Math 112C Final Project
Heat Sink Shape Optimization

Kyle Spink

June 10th 2024

Contents

1 Overview of Optimal Control & Shape Optimization

Optimal control is a branch of optimization for problems which involve a parameter or control variable, a state variable and a partial differential equation (with appropriate boundary conditions) over the domain defined by the state variables. [13]

Optimal shape design, also known as shape optimization is a branch of computational mechanics and a special case of optimal control where the control is the boundary of the shape itself. Data that describes the behavior of a shape (parameters or control variables) subject to governing laws/equations is obtained and one refines the structure until certain optimal properties are achieved. [9]

Some common examples of quantities one would aim to extremize are related to material costs (surface area or volume), lift-to-drag ratio, heat transfer, electromagnetic interference, or structural integrity (stress distribution) [Fig. 1]

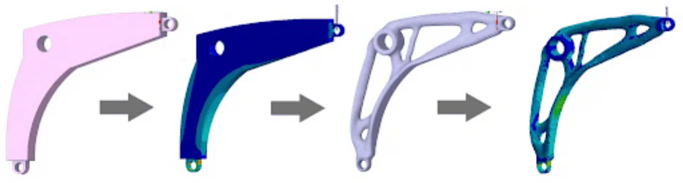


Fig. 1: Optimization of a bell crank to cut down material in low-stress areas[17]

2 Problem Formulation

The setup for the shape optimization of a heat sink profile is as follows: constant heat is emitted from a source to the heat sink profile which is surrounded by room temperature air. We want to determine the profile of the heat sink that's the Pareto optimal (for equal weights) with respect to both the total area of the heat sink as well as the average temperature within the shape's domain after sufficient time has passed.

The prior statement "after a sufficient time has passed" implies that the temperature distribution is governed by the Laplace Equation ("steady-state heat equation").

The term "Pareto optimal" is coined from economics and refers to a state when no interaction or change can make one individual better off without making the another worse off. In the context of optimal control, a point in the admissible space is Pareto optimal if there is no other point that is better in at least one objective and not worse in any other objective. By introducing parameters that act as weights for each objective, the solutions form a Pareto front which represents the set of all Pareto optimal solutions. [Fig. 2]

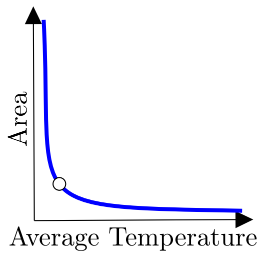


Fig. 2: Schematic of Pareto Front[7]

Mathematically, there will be a Dirichlet boundary condition on the bottom face that's in direct contact with the source, and convective (Robin) boundary conditions for all other points on the boundary of the profile. Convective boundary conditions are applied when there's heat transfer between a solid body and a fluid (air). [Fig. 3] When the surface undergoes convective heat transfer, it's governed by Newton's Law of Cooling and the temperature distribution is asymptotic toward some ambient/free stream temperature, u_∞ far away from the body. [1, 16]

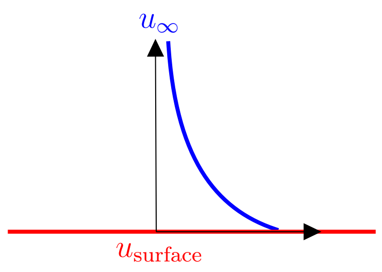


Fig. 3: Hot plume from convective boundary condition, red is the surface and blue is how the temperature transitions from u_{surface} to the ambient temperature, u_∞

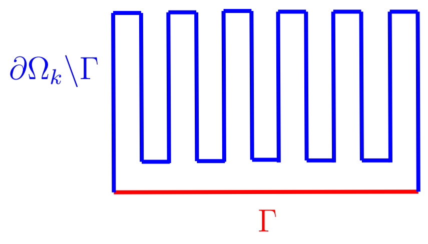


Fig. 4: Boundary of Heat Sink

To summarize, we seek to minimize $J[u_k] = \lambda_1 \iint_{\Omega_k} dxdy + \lambda_2 \cdot \frac{1}{|\Omega_k|} \iint_{\Omega_k} u_k(x,y)dxdy$ (sum of the area of the domain and average temperature over the domain) such that

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \coloneqq \nabla^2 u = 0 & (x,y) \in \Omega_k \\ u = u_{\text{source}} & (x,y) \in \Gamma \\ -k\frac{\partial u}{\partial n} = h(u - u_\infty) & (x,y) \in \partial\Omega_k \backslash \Gamma \end{cases}$$

3 The Finite Element Method

Although past material covered in Math 112C makes one more inclined to solve the 2D Laplace Equation using a Finite Differences Scheme, this proves to be less robust and more challenging to implement on arbitrary domains compared to the Finite Element Method. A derivation of the Finite Element Method's setup for the Poisson (non-homogeneous Laplace) Equation is as follows.

We aim to solve for u on a planar domain, Ω

$$\begin{cases} \nabla^2 u = 0 & (x,y) \in \Omega \\ u(x,y) = f & (x,y) \in \delta\Omega \end{cases} \tag{1}$$

Let $v(x)$ be a scalar function on Ω such that it vanishes on the boundary, $v(x)\big|_{\delta\Omega} = 0$. Multiplying (1) by v and applying integration by parts yields

$$\int_\Omega f(x,y)v(x)dA = \cancelto{0}{v(x) \cdot \nabla u\big|_{\delta\Omega}} - \int_\Omega (\nabla u \cdot \nabla v)dA \tag{2}$$

by construction. Then generating a tiling of Ω by triangular subdivisions such that for each interior vertex of the triangles labeled $v_i, \ldots, v_n$, we pick a set of basis functions $h_i(x, y) = ax + by + c$ on each triangle with vertices as the indices i, j, k such that

1. At the i^{th} vertex, v_i, $h_i(v_i) = 1$ and at every other vertex, $h_i(v_j) = h_i(v_k) = 0$
2. Within each triangle, h_i is a linear function and piece-wise linear throughout the rest of the domain Ω and
3. a, b, c solve the following system of linear equations:

$$h_i(v_i) = av_{i,x} + bv_{i,y} + c = 1$$
$$h_i(v_j) = av_{j,x} + bv_{j,y} + c = 0$$
$$h_i(v_k) = av_{k,x} + bv_{k,y} + c = 0$$

The solution to the Poisson Equation is given by the linear combination of these basis functions

$$u(x, y) \approx \sum_{i=1}^{n} u_i h_i(x, y)$$

Imposing the conditions from (2),

$$\int_\Omega f(x, y)h_j(x, y)dA = -\int_\Omega (\nabla u \cdot \nabla h_j)dA = -\sum_{j=1} u_i \int_\Omega (\nabla h_i \cdot \nabla h_j)dA$$

which is equivalent to

$$K\boldsymbol{u} = \boldsymbol{f} \Rightarrow \boldsymbol{u} = K^{-1}\boldsymbol{f}$$

for $k_{ij} = -\int_\Omega (\nabla h_i \cdot \nabla h_j)dA$ and $f_j = \int_\Omega f(x, y)h_j(x, y)$ with

$$K := \begin{bmatrix} k_{11} & \cdots & k_{1j} \\ \vdots & \ddots & \vdots \\ k_{i1} & \cdots & k_{ij} \end{bmatrix} \qquad \boldsymbol{u} := \begin{bmatrix} u_1 \\ \vdots \\ u_j \end{bmatrix} \qquad \boldsymbol{f} := \begin{bmatrix} f_1 \\ \vdots \\ f_j \end{bmatrix}$$

K is called the spring or stiffness matrix, $\boldsymbol{u}$ is the solution, and $\boldsymbol{f}$ is called the load vector. Accounting for $f = 0$ for the Laplace Equation and different boundary conditions result in modifications to K and $\boldsymbol{f}$. To summarize, the Finite Element Method frames a boundary value problem as a system of algebraic equations and enables us to solve the Laplace Equation on an arbitrary domain. [2, 6]

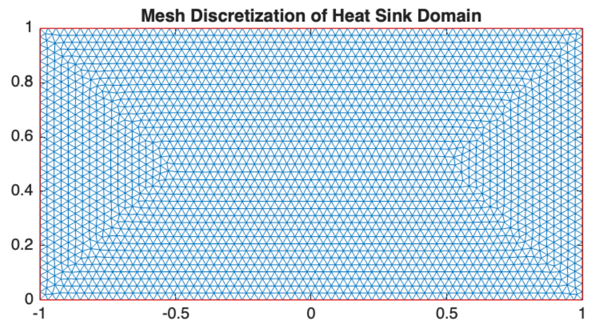


Fig. 5: Mesh discretization of the heat sink domain formed by triangular subdivisions

4 The Adjoint Method

To introduce the discrete Adjoint Method in its form used for shape optimization of the heat sink, we will first view a variation of it within the context of linear algebra. Consider

$$\min_x \{C^T x\} \text{ subject to } Ax = b(S)$$

Then, assuming A is invertible,

$$x = A^{-1}b(S) \Rightarrow C^T x = C^T A^{-1} b(S)$$

Rather than solving $Ax = b(S)$ after each new S and updating $C^T x$ accordingly, one can "pre-compute" $\hat{A} := C^T A^{-1}$ and optimize the original function by only varying S. [18]

$$\min_{S \in \Omega} \{\hat{A} b(S)\}$$

This demonstrates how *the order of operations plays a significant role in a problem's computational complexity.* [18]

Now let $J(b_i)$, $i = 1, \ldots, N$ be an objective function and $R(u_k) \equiv 0$, $k = 1, \ldots, M$ be a constraint/state function. For shape optimization of the heat sink, R is the residual function—a measure of the approximate solutions' resemblance to the true solution and is defined as $R(u_k) = Ku - F = 0$ where K and F are the spring matrix and load vector from the Finite Element Method respectively. Then the total derivatives of both functions with respect to the design variables are given by [15, 11]

$$\frac{dJ}{db_i} = \frac{\partial J}{\partial b_i} + \frac{\partial J}{\partial u_k}\frac{du_k}{db_i} \quad (3) \qquad\qquad \frac{dR}{db_i} = \frac{\partial R}{\partial b_i} + \frac{\partial R}{\partial u_k}\frac{du_k}{db_i} = 0 \quad (4)$$

which can be equivalently expressed as

$$\frac{dJ}{db_i} = \frac{\partial J}{\partial b_i} + \psi_k \frac{\partial R}{\partial b_i}$$

such that

$$\frac{\partial J}{\partial u_k} + \psi_k \frac{\partial R}{\partial u_k} = 0 \quad (5)$$

The value ψ_k is referred to as the adjoint, or costate variable and (5) is referred to the adjoint equation. To view this equivalence, clearly from (4), $\frac{\partial R}{\partial b_i} = -\frac{\partial R}{\partial u_k}\frac{du_k}{db_i}$ and from (5)

$$\psi_k = -\frac{\left(\frac{\partial J}{\partial u_k}\right)}{\left(\frac{\partial R}{\partial u_k}\right)}$$

Thus, $\psi_k \cdot \frac{\partial R}{\partial b_i}$ is given by

$$-\frac{\left(\frac{\partial J}{\partial u_k}\right)}{\left(\frac{\partial R}{\partial u_k}\right)} \cdot -\frac{\partial R}{\partial u_k}\frac{du_k}{db_i} = \frac{\partial J}{\partial u_k}\frac{du_k}{db_i}$$

as in (3). Once calculated, $\frac{dJ}{db_i}$ are then used in a gradient based optimizer to generate the next evolved shape.

It's important to see how the adjoint equation, (5) has no dependence on the design parameters, b_i but rather is solely dependent on the pseudo-time step, k. Although both $\frac{\partial F}{\partial U_k}$, $\frac{\partial R}{\partial U_k}$ (and, as a result, the adjoint variable, ψ_k) change with each new iteration of the *set* of design variables b_i, the adjoint variable only requires to be computed once per iteration and is used to compute the total derivatives with respect to *all* design variables b_i, therefore significantly reducing the computational complexity.

Informally, there are "efficient methods to obtain sensitivities of *many* functions with respect to a *few* design variables" (direct) as well as "efficient methods to obtain sensitivities of a *few* functions with respect to *many* design variables." (adjoint). [5] Another reason for the superiority of the Adjoint Method is that the gradient points towards a better solution where evolutionary/genetic algorithms are generally unable to exploit local information when the solution is refined. [15]

To demonstrate the connection between the adjoint equation and the "primer" that was introduced for linear algebra, consider the scenario when the objective function, $J = \gamma^T u_k$ and state function, u_k satisfies the linear system $Au_k = \phi$. The adjoint equation $A^T \psi = \gamma$ is first solved, from which $J = \psi^T \phi$. [15] For our proposed problem,

$$R(u_k) := K_k u_k - f = 0 \Rightarrow \frac{\partial R}{\partial u_k} = K_k \Rightarrow \psi_k = -K_k^{-1}\left(\frac{\partial J}{\partial u_k}\right)^T$$

All partial derivatives may be numerically calculated using central finite differences:

$$\frac{\partial J}{\partial u_k} \approx \frac{\Delta J}{\Delta u_k} \qquad \frac{\partial R}{\partial b_i} \approx \frac{\Delta K_k}{\Delta b_{i,k}} + \frac{\Delta U_k}{\Delta b_{i,k}} - \frac{\Delta f_k}{\Delta b_{i,k}} \qquad \frac{\partial J}{\partial b_i} \approx \frac{\Delta J_k}{\Delta b_{i,k}}$$

Here, $b_{i,k}$ refers to the i^{th} design variable in the (fixed) k^{th} pseudo-time step, and $(\cdot)_k$ is the value with respect to the i^{th} design variable $(\Delta(\cdot) = (\cdot)^+ - (\cdot)^-$ around the value of $b_i)$.

5 Considerations for a Feasible Design

In order for the model to converge to an optimal profile, the following constraints/choices were implemented.

First involves properly applying the shape gradient. Depending on its magnitude, pushing nodes according to their gradient may result in a tangled mesh. This can lead to poor-quality elements that can cause numerical instability and inaccurate results. [8] To prevent this from occurring, a threshold for the gradient of the objective function with respect to the design variables was defined, $\left|\frac{dJ}{db_i}\right| > \left|H_{\max}\right|$ which, if met, will remove or add elements and initiate a remesh of the current geometry. In practice, $H_{\max}$ was specified to be the maximum height of each individual element of the original mesh.

The second choice was only updating the boundary of the shape, not every point within the shape's domain. Although this is seen more for shapes subject to no-penetration boundary conditions such as an airfoil, at the time this was justified due to the homogeneity of the shape's thermal conductivity which was thought to imply that the objective function was solely dictated by the boundary of the shape itself.

Thirdly, normalizing the area and the average temperature. Recall the original objective function was:

$$J_k = \lambda_1 \iint_{\Omega_k} dxdy + \lambda_2 \cdot \frac{1}{|\Omega_k|} \iint_{\Omega_k} u_k(x,y)dxdy$$

which was changed to

$$J = \lambda_1 \frac{A_k}{A_{\text{ref}}} + \lambda_2 \frac{\bar{u}_k}{\bar{u}_{\text{ref}}}$$

where A_k is the area of the shape and $\bar{u}$ is the average temperature of the shape in the k^{th} pseudo-time step, with A_{ref} and $\bar{u}_{\text{ref}}$ being reference areas and temperatures (taken at $k = 0$). This was done to prevent a bias of minimizing the average temperature that would result in a significant increase of the area which is unwanted. [Fig. 6]

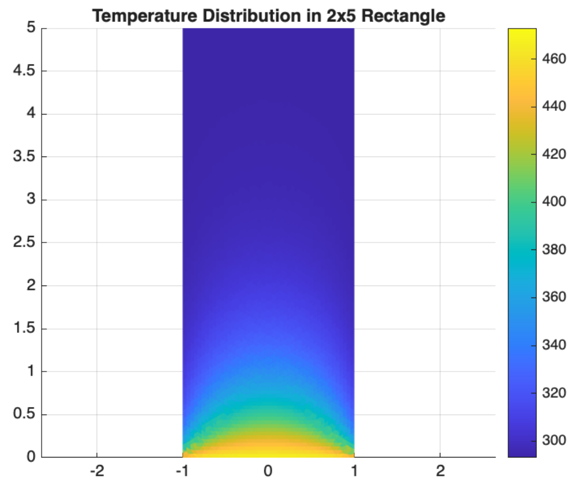


Fig. 6: Area: 10, Average Temperature: 317 $\Rightarrow J = 327$, less than J for a 2×1 rectangle

6 Numerical Scheme

Initialization:

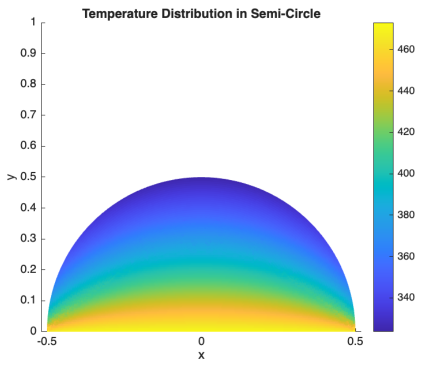
1. Create and import initial 2D domain
2. Mesh domain using the Delaunay Triangulation Algorithm[10] (Fig. 7)
3. Solve primal equation (Laplace) with prescribed boundary conditions using the Finite Element Method[14]
   - $k = 237$ W/m $\cdot$ K (thermal conductivity for aluminum)[19]
   - $h = 7$ W/m^2 $\cdot$ K (natural convective heat transfer coefficient between air and aluminum)
4. Evaluate the objective function, J_0
5. Perturb boundary nodes to enable first gradient calculation

Then for $k = 1, \ldots M$:

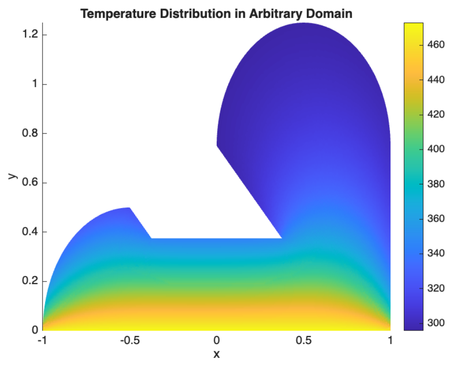
1. Solve primal equation for u_k
2. Evaluate the objective function, $J[u_k]$
3. Solve the adjoint equation system

4. Compute the sensitivity information for each boundary node
5. Update the geometry based on Step 4 using a standard gradient descent algorithm, $b_k = b_{k-1} - \alpha \frac{dJ}{db}$, $\alpha = 0.1$
   (a) If necessary, remesh domain
6. Repeat if $\big|J[u_k] - J[u_{k-1}]\big| > \epsilon$ (convergence is not met)

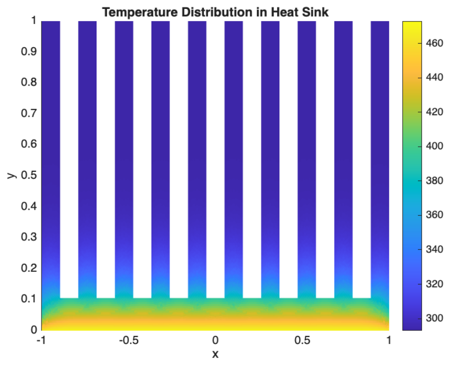
7 Results



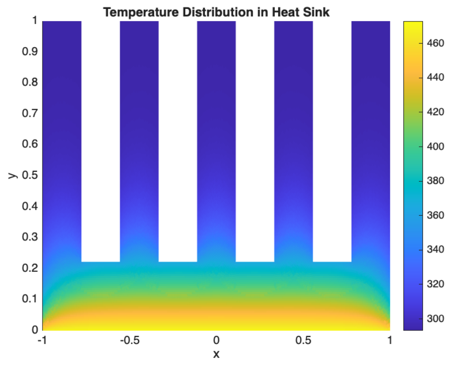
Area: 0.39, Average Temperature: 391.42



Area: 1.46, Average Temperature: 358.74

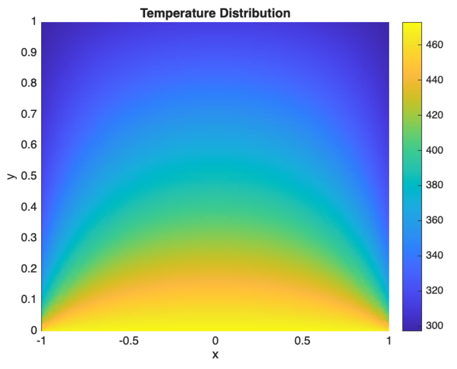


Area: 1.15, Average Temperature: 324.56

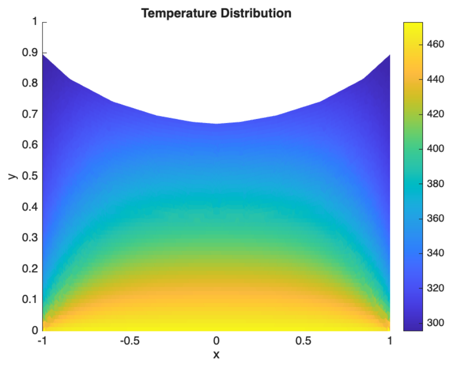


Area: 1.31, Average Temperature: 341.40

The shapes' profile in above figures are manually drawn—not a result of the performing optimization, although as one may expect, an increase in the number of nodes subject to convective boundary conditions (as in the bottom left figure) results in a lower average temperature.



Initial Iteration: Area: 2, Average Temperature: 372.63,
Objective Function: 2



Final Iteration: Area: 1.5, Average Temperature: 379.22,
Objective Function: 1.761

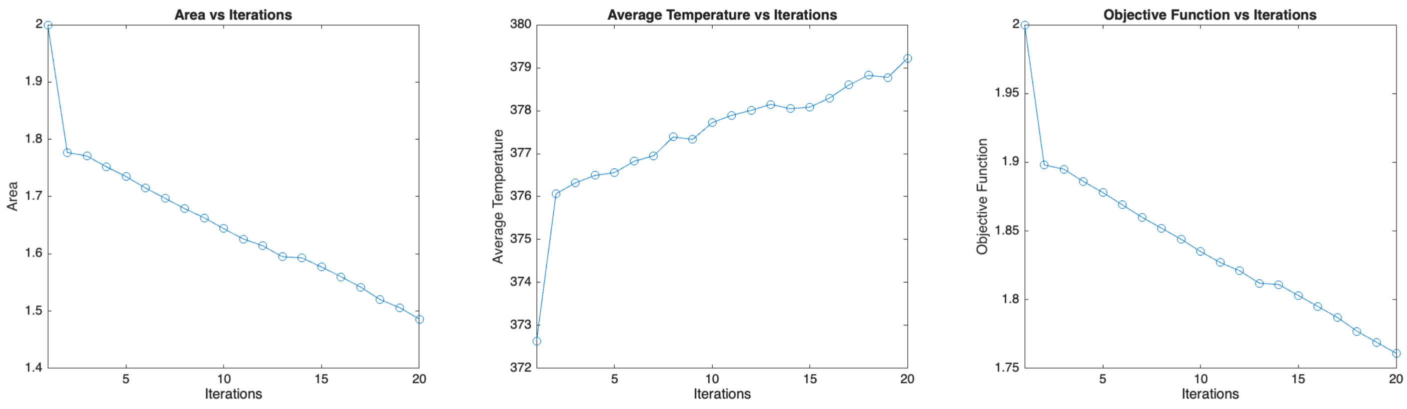


Fig. 7: Subplot of the area (m^2), average temperature (K), and objective function vs. iterations

8 Concluding Remarks

I believe that multiple oversights resulted in the shape profile converging to what wouldn't be expected. Due to a lack of time (and experience), I wasn't able to automate the full for loop that would generate and mesh an evolved profile. Instead, the profiles in each one of the twenty iterations had to be manually drawn, exported, imported to MATLAB, ran, and saved which was inefficient...

That being said, its promising that the objective function in Fig. 7 is shown to decrease as expected. One slight caveat is that as also shown in Fig. 7, this decrease was a result of a decrease in the area but an unwanted increase in the average temperature. This could be interpreted to mean that the upside of a given decrease in the area outweighed the downside of its corresponding increase in the average temperature (with respect to the initial shape). To remedy this, in the future one may want to adjust the corresponding weights, λ_1 and λ_2 accordingly or redefine the objective function altogether.

Other potential changes that could be made include a) using a stochastic gradient descent algorithm, b) adding noise to each design variable, c) updating both the boundary points and the interior points, and d) accounting for the interaction of emitted heat between two edges of the shape that are close in proximity to one another (e.g., two adjacent heat sink fins).

Also worth mentioning is how this model doesn't allow for the generation of shapes without a simply connected boundary (e.g., a through hole) which is common to see in some heat sink fin designs to enable faster cooling. Rather than the Adjoint Method, the Level Set Method is implemented for shape optimization problems that require this topological change.

For the Level Set Method, let the boundary of the shape domain, $\Gamma = \{(x, y) : \varphi(x, y) = 0\}$ where φ is called the level set function, a three dimensional surface that represents the evolution of the two dimensional boundary. [Fig. 9]

What dictates the evolution of Γ is the Hamilton-Jacobi partial differential equation,

$$\frac{\partial \varphi}{\partial t} = v|\nabla \varphi|$$

where t is time and v is the speed the curve moves in the normal direction. This formulation enables the shape to split in two, the development of holes, and the reverse of the two previous operations. [Fig. 8] [12]

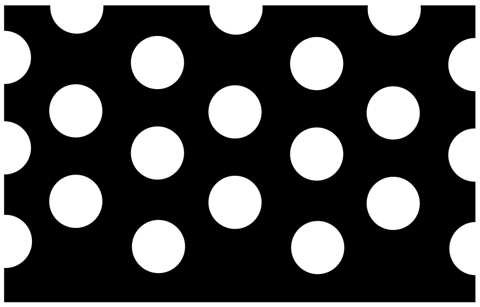


Fig. 8: Potential initial design for the Level Set Method

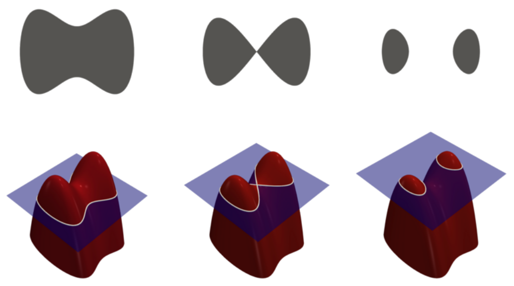


Fig. 9: An illustration of the Level Set Method[3]

9 Other Applications—Airfoil Optimization

Although the Navier-Stokes Equations best describe the flow, because of the complexity, the flow is often modeled as two-dimensional, *incompressible* (constant density) and *inviscid* (zero viscosity). Then the problem is to find a stream function, φ ($v = \operatorname{curl}(\varphi)$ for the velocity, v) where $\varphi := \varphi(b_i)$ and constant $\beta := \beta(b_i)$ such that [9]

$$\begin{cases} -\Delta\varphi = 0 & (x,y) \in \Pi\backslash\Omega \\ \varphi = \beta & (x,y) \in \partial\Omega(b_i) \\ \varphi = \varphi_\infty & (x,y) \in \partial\Pi \\ \nabla\varphi \text{ is regular in a vicinity of the trailing edge, } b_N & \end{cases}$$

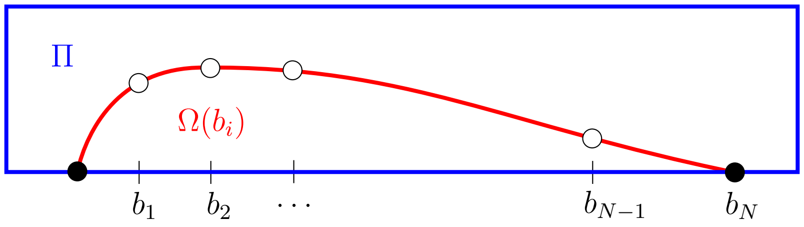


Fig. 10: Shape parameterization of the upper surface of an airfoil. Solid points are fixed and open points are allowed to move in the y direction

"$\nabla\varphi$ being regular in the vicinity of the trailing edge" is also commonly referred to as the Kutta Condition. In simple terms, when the Kutta Condition is satisfied, fluid that approaches the trailing edge from above and below meets at the trailing edge and then flows away from the body without flowing around the sharp corner which would introduce nonphysical behavior. Due to the radius of the sharp trailing edge being zero, the fluid's velocity around the trailing edge would be infinitely fast. [4]

References

[1] May 2021. URL: `https://www.youtube.com/watch?v=ygR31_vhvJI`.

[2] May 2024. URL: `https://en.wikipedia.org/wiki/Finite_element_method`.

[3] May 2024. URL: `https://en.wikipedia.org/wiki/Level-set_method`.

[4] Feb. 2024. URL: `https://en.wikipedia.org/wiki/Kutta_condition#:~:text=The%20Kutta%20condition%20is%20a,mathematician%20and%20aerodynamicist%20Martin%20Kutta.`.

[5] Juan J Alonso. *A Crash-Course on the Adjoint Method for Aerodvnamic Shape Optimization.* Mar. 2005. URL: `http://aero-comlab.stanford.edu/aa200b/lect_notes/aa200b-lect18.pdf`.

[6] Birne Binegar. URL: `https://math.okstate.edu/people/binegar/4263/4263-l21.pdf`.

[7] S. Brisset and F. Gillon. *Approaches for Multi-Objective Optimization in the Ecodesign of Electric Systems.* 2015. URL: `https://www.sciencedirect.com/topics/engineering/pareto-optimality`.

[8] Pascal Frey. *Shape Optimization.* URL: `https://www.ljll.fr/~frey/cours/Teheran/P.%20Frey,%20chap7.pdf`.

[9] J. Haslinger and R. A. E. Mäkinen. *Introduction to Shape Optimization: Theory, Approximation, and Computation.* SIAM, Society for Industrial and Applied Mathematics, 2003.

[10] *MATLAB PDE Toolbox Initial 2D Mesh.* URL: `https://www.mathworks.com/help/pde/ug/initmesh.html`.

[11] Dimitri J. Mavriplis. *Adjoint-Based Sensitivity Analysis for Computational Fluid Dynamics.* URL: `https://www.cerfacs.fr/musaf/PDF_MUSAF_DAY3/MAVRIPLIS/Mavriplis.pdf`.

[12] David Herrero Pérez. *Level Set Method Applied to Topology Optimization.* 2024. URL: `https://www.upct.es/goe/level-set-slides.pdf`.

[13] Olivier Pironneau. "Chapter 1: Optimal Shape Design". In: URL: `https://www.ljll.fr/pironneau/publi/bmopBOOK/opt_1_osd.pdf`.

[14] Gilbert Strang. URL: `http://math.mit.edu/~gs/cse/codes/femcode.m`.

[15] Dominique Thévenin and Gábor Janiga. *Optimization and Computational Fluid Dynamics.* Springer-Verlag Berlin Heidelberg, 2008.

[16] Ilan Toledano. *Heat Conduction Boundary Conditions.* Mar. 2024. URL: `https://www.wattco.com/2023/06/heat-conduction-boundary-conditions/#:~:text=A%20convection%20boundary%20condition%20occurs,by%20Newton%E2%80%99s%20law%20of%20cooling.`.

[17] *Topology Optimization of a Bell Crank.* URL: `https://courses.ansys.com/index.php/courses/topology-optimization-using-ansys-mechanical/lessons/topology-optimization-of-a-bell-crank-lesson-1/`.

[18] Qiqi Wang. *Introduction to the Adjoint Method*. May 2018. URL: `https://www.youtube.com/watch?v=EybH_Q-QTZ8&list=LL&index=10&t=369s`.

[19] Ailing Zhang and Yanxiang Li. *Thermal Conductivity of Aluminum Alloys*. Apr. 2023. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10144406/`.