Wall-to-wall heat transfer with unsteady flow perturbations

Nicole Vuong, PI: Dr. Silas Alben

August 9, 2024

Abstract

The efficiency of many real-life appliances relies on its ability to cool effectively. Thus, optimizing heat transfer in fluids via advection (the movement of a fluid) is a point of interest. In this paper, we explore the effect of adding unsteady perturbations to a previously derived optimal steady flow, with a fixed time-averaged power, on the heat transfer in a 2D channel. We also discuss current progress towards finding optimal unsteady flows that maximize heat transfer in the 2D channel via the genetic algorithm.

Contents

1 Introduction

Historically, forced convection of fluid flows has been a common strategy for increasing heat transfer. The advent of the plate fin heat sink is one such example. The shape of a plate fin heat sink forces convection to occur, which enhances cooling of the airflow that flows through it [6]. In recent years, Hassanzadeh, Chini, and Doering studied forced incompressible flows in a 2D channel heated at the bottom and cooled at the top, in order to find optimal steady flows of a given mean rate of viscous dissipation (Pe^2) that maximize the rate of heat transfer from the walls [2]. In addition to the incompressibility and energy budget (rate of viscous dissipation) constraints, they imposed a free slip condition on their choices of fluid flows.

In 2023, Alben found optimal steady, horizontally periodic flows for maximizing heat transfer from the walls of a similar geometry, with the exception of enforcing no slip conditions on the fluid flows instead of free slip conditions [1]. He found that at lower Pe^2 values, the optimal flows took on the shape of convective rolls while at higher Pe^2 values, the optimal flows found had increasingly elongated and asymmetrical branching near the walls of the channel. His optimization method both optimized for the flow shape and the period of the flow in the x-direction.

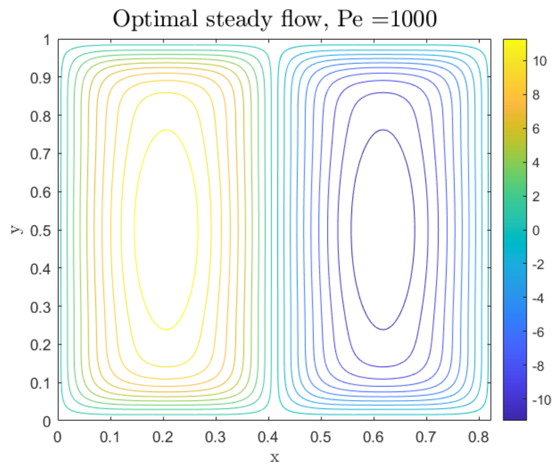


Figure 1: An optimal steady flow for Pe = 1000, from Alben [2023].

We now seek to build upon the work of Alben in [1] by also considering optimal unsteady flows. That is, can unsteady flows of a given averaged rate of viscous dissipation result in improved heat transfer compared to steady flows? As a starting point, we first investigate unsteady flows that are largely similar to the optimal steady flows found by Alben - we consider the optimal steady flows with small unsteady perturbations added to them. Additionally, we only consider flows with averaged rate of viscous dissipation $\mathrm{Pe}^2 = 10^6$. We choose this smaller Pe^2 value because the behavior of the optimal steady flow found by Alben at $\mathrm{Pe}^2 = 10^6$ is simplest. Thus, numerical simulations working with slightly perturbed versions of the optimal steady flow with $\mathrm{Pe}^2 = 10^6$ value will not be as computationally expensive.

2 Problem setup

Consider a layer of fluid in the 2D rectangular channel $\{0 \leq x \leq L_x,\ 0 \leq y \leq 1\}$. ($L_x$ here is the x-direction period of the optimal steady flow found by Alben for $\mathrm{Pe}^2 = 10^6$. L_x is approximately 0.83.) The bottom wall $y = 0$ has temperature $T = 1$, the top wall $y = 1$ has temperature $T = 0$, and the temperature is periodic in x with period L_x. Also, the initial temperature of the fluid in the domain is the temperature field associated with the optimal steady stream function ψ_0 for Pe = 1000, and the temperature is time-periodic with period τ.

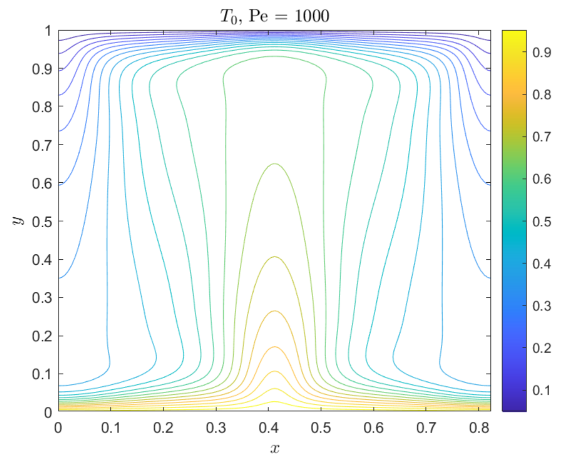


Figure 2: The initial temperature field.

We wish to find optimal unsteady fluid flows $(u(x,y,t), v(x,y,t))$ that maximize the time- and horizontally-averaged heat transfer Nu out of the bottom wall, which is given by

$$Nu = \frac{1}{\tau L_x} \int_0^{\tau} \int_0^{L_x} -\partial_y T\Big|_{y=0} dx\, dt. \tag{1}$$

Note that the averaged heat transfer out of the bottom wall is the same as the averaged heat transfer into the top wall.

We solve for the temperature field $T(x,y,t)$ needed to compute (1) via the unsteady advection-diffusion equation

$$\partial_t T + u\partial_x T + v\partial_y T - (\partial_{xx}T + \partial_{yy}T) = 0. \tag{2}$$

We also impose the following restrictions the unsteady fluid flows must follow:

1. Incompressibility - that is, $\partial_x u + \partial_y v = 0$. To follow convention, we represent unsteady fluid flows in this paper as stream functions $\psi(x,y,t)$, where (u,v) is set to $(\partial_y\psi, -\partial_x\psi)$ to automatically obey incompressibility [5].

2. Periodic in x with period L_x.

3. Periodic in time with time period τ, the same time period as that of the temperature field T.

4. No-slip conditions - that is, zero velocity at the top and bottom walls [3].

5. The fluid flow must have Pe = 1000, the same Pe as that of the previously derived optimal steady flow.

Condition 5 above is equivalent to $\mathrm{Pe}^2 = 10^6$, where Pe^2 is given by

$$\widetilde{\text{Power}} = \mathrm{Pe}^2 = \frac{1}{\tau L_x} \int_0^{\tau} \int_0^{1} \int_0^{L_x} \partial_{xx}\psi^2 + \partial_{yy}\psi^2 - 2\partial_{xx}\psi\partial_{yy}\psi + 4\partial_{xy}\psi^2 \, dx\, dy\, dt. \tag{3}$$

2.1 Construction of stream functions

The stream functions we consider for this paper are of the form

$$\psi_j(x,y,t) = \frac{1}{\sqrt{1+\epsilon^2}}\psi_0(x,y) + \frac{\epsilon}{\sqrt{1+\epsilon^2}} f_j(x,y)\cos(2\pi t/\tau_j), \tag{4}$$

where ψ_0 is the previously derived optimal steady stream function. We do this to ease the computational time of calculating Nu (see Section 3 for a discussion of this), and also to have a starting point for finding optimal unsteady stream functions. f_j is a superposition of modes, each being a product of a Fourier mode in the x-direction and a linear combination of Chebyshev polynomials $Y_i(y)$ in the y-direction. In other words,

$$f_j(x,y) = \sum_i \sum_k a_{ik} Y_i(y) \cos(2\pi k x/L_x) + b_{ik} Y_i(y) \sin(2\pi k x/L_x). \tag{5}$$

In Matlab, f_j is discretized as a matrix $\mathbf{V}$ (see Section 3.1 for a discussion of the domain discretization), with each mode being a column of $\mathbf{V}$. For this paper, we opt to use 357 modes. Each mode is normalized to have $\mathrm{Pe}^2 = 2 * 10^6$ and the modes are made to be pairwise orthogonal. Then, the coefficents of the modes a_{ik}, b_{ik} are represented by a vector b_j, which must be normalized to have norm 1. This ensures that the discretized version of f_j in Matlab created by setting $f_j \equiv \mathbf{V} b_j / \|b_j\|$ has $\mathrm{Pe}^2 = 2 * 10^6$.

From (3), notice that if f_j has $\mathrm{Pe}^2 = 2 * 10^6$, then $f_j(x,y)\cos(2\pi t/\tau_j)$ has $\mathrm{Pe}^2 = 10^6$. This is because the time-average of $\cos^2(2\pi t/\tau_j)$ over its period τ_j is 1/2. Then, substituting (4) into (3) and rearranging terms, we have that Pe^2 of (4) is given by

$$\mathrm{Pe}^2 = \left(\frac{1}{1+\epsilon^2}\right)\left(\frac{1}{\tau_j L_x}\right)\int_0^{\tau_j}\int_0^1\int_0^{L_x} \partial_{xx}\psi_0^2 + \partial_{yy}\psi_0^2 - 2\partial_{xx}\psi_0\partial_{yy}\psi_0 + 4\partial_{xy}\psi_0^2 \, dx\, dy\, dt \tag{6}$$

$$+ \left(\frac{\epsilon^2}{1+\epsilon^2}\right)\left(\frac{1}{\tau_j L_x}\right)\int_0^{\tau_j}\int_0^1\int_0^{L_x} \left(\partial_{xx}\left(f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right)\right)\right)^2 + \dots \, dx\, dy\, dt \tag{7}$$

$$+ \left(\frac{2\epsilon}{1+\epsilon^2}\right)\left(\frac{1}{\tau_j L_x}\right)\int_0^{\tau_j}\int_0^1\int_0^{L_x} \partial_{xx}\psi_0 \partial_{xx}\left(f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right)\right) + \dots \, dx\, dy\, dt, \tag{8}$$

where (7) is $\frac{\epsilon^2}{1+\epsilon^2}$ times the Pe^2 of $f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right)$ and the terms of (8) are all proportional to $\cos\left(\frac{2\pi t}{\tau_j}\right)$. (8) evaluates to zero because the time-average of $\cos\left(\frac{2\pi t}{\tau_j}\right)$ over its period τ_j is zero. So, assuming that f_j has $\mathrm{Pe}^2 = 2 * 10^6$, it follows from (6) and (7) that Pe^2 of (4) is $\left(\frac{1}{1+\epsilon^2}\right)(10^6) + \left(\frac{\epsilon^2}{1+\epsilon^2}\right)(10^6) = 10^6$. In other words, (4) does indeed have the same Pe^2 as the optimal steady stream function ψ_0.

3 Estimating Nu

Directly solving the advection-diffusion equation for a prescribed stream function ψ_j is numerically very time-consuming. So, it is desirable to approximate the time- and horizontally- averaged heat transfer, Nu, via more time-efficient means. To this end, we employ techniques from perturbation theory. We begin by expanding ψ_j and T in powers of ϵ,

$$\psi_j(x,y,t) = \psi_0(x,y,t) + \epsilon\psi_1(x,y,t) + \epsilon^2\psi_2(x,y,t) + \dots \tag{9}$$

$$T(x,y,t) = T_0(x,y,t) + \epsilon T_1(x,y,t) + \epsilon^2 T_2(x,y,t) + \dots, \tag{10}$$

where we can substitute (10) into (2) to obtain an analogous expansion for Nu given by

$$Nu = Nu_0 + \epsilon Nu_1 + \epsilon^2 Nu_2 + \dots. \tag{11}$$

Since ψ_j is explicitly defined in (4), we can find ψ_0, ψ_1 and ψ_2, which will be relevant in future derivations. Via the second order Taylor series expansions for $\frac{1}{\sqrt{1+\epsilon^2}}$ and $\frac{\epsilon}{\sqrt{1+\epsilon^2}}$, we see that

$$\psi_j(x,y,t) \approx \left(1 - \frac{\epsilon^2}{2}\right)\psi_0(x,y) + \left(\epsilon - \frac{\epsilon^3}{2}\right) f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right) \tag{12}$$

$$= \psi_0(x,y) + \epsilon f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right) - \frac{\epsilon^2}{2}\psi_0(x,y) + \dots, \tag{13}$$

implying that $\psi_0(x,y,t)$ is simply the optimal steady stream function, $\psi_1(x,y,t) = f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right)$, and $\psi_2(x,y,t) = -\frac{\psi_0(x,y)}{2}$.

We now solve (2) at each power of ϵ to obtain Nu_0, Nu_1, and Nu_2. At $O(\epsilon^0)$ (2) is

$$\partial_y\psi_0\partial_x T_0 - \partial_x\psi_0\partial_y T_0 - \partial_{xx}T_0 - \partial_{yy}T_0 = 0. \tag{14}$$

This is precisely the steady advection-diffusion equation with the optimal steady flow ψ_0, so Nu_0 is simply the heat transfer associated with the optimal steady flow. At $O(\epsilon^1)$ equation (2) is

$$\partial_t T_1 + \partial_y\psi_0\partial_x T_1 - \partial_x\psi_0\partial_y T_1 - \partial_{xx}T_1 - \partial_{yy}T_1 = -\partial_y\psi_1\partial_x T_0 + \partial_x\psi_1\partial_y T_0. \tag{15}$$

Observe that because $\psi_1(x,y,t) = f_j(x,y)\cos\left(\frac{2\pi t}{\tau_j}\right)$, the right hand side of (15) is equal to

$$-\cos\left(\frac{2\pi t}{\tau_j}\right)\left(\partial_y f_j(x,y)\partial_x T_0(x,y) + \partial_x f_j(x,y)\partial_y T_0(x,y)\right). \tag{16}$$

This means we can write T_1 as $T_{1A}(x,y)\cos(2\pi t/\tau_j) + T_{1B}(x,y)\sin(2\pi t/\tau_j)$. However, because the cosine and sine terms average to zero over its period τ_j, the associated heat transfer Nu_1 must also be zero. It is typical to simply evaluate up to the first-order term in (11) to achieve an approximation of Nu. But, since $Nu_1 = 0$, we must evaluate (2) at an additional power of ϵ to see if the added unsteady perturbation in (4) improves the total heat transfer Nu compared to the optimal steady flow ψ_0.

At $O(\epsilon^2)$ equation (2) is

$$\begin{aligned}&\partial_t T_2 + \partial_y\psi_0\partial_x T_2 - \partial_x\psi_0\partial_y T_2 - \partial_{xx}T_2 - \partial_{yy}T_2\\ &= -\partial_y\psi_1\partial_x T_1 + \partial_x\psi_1\partial_y T_1 - \partial_y\psi_2\partial_x T_0 + \partial_x\psi_2\partial_y T_0.\end{aligned} \tag{17}$$

Plugging in the expressions for ψ_1, ψ_2, and T_1 into the right hand side of (17), we have that the right hand side of (17) can be rewritten as

$$\begin{aligned}&\frac{1}{2}\cos\left(\frac{4\pi t}{\tau_j}\right)\left(-\partial_y f_j\partial_x T_{1A} + \partial_x f_j\partial_y T_{1A}\right) + \frac{1}{2}\sin\left(\frac{4\pi t}{\tau_j}\right)\left(-\partial_y f_j\partial_x T_{1B} + \partial_x f_j\partial_y T_{1B}\right)\\ &-\frac{1}{2}\partial_y f_j\partial_x T_{1A} + \frac{1}{2}\partial_x f_j\partial_y T_{1A} + \frac{1}{2}\partial_y\psi_0\partial_x T_0 - \frac{1}{2}\partial_x\psi_0\partial_y T_0.\end{aligned} \tag{18}$$

We now write T_2 as $T_{2s}(x,y) + T_{2u}(x,y,t)$ (splitting T_2 into a steady and unsteady part) and rewrite (17) as the following two equations:

$$\begin{aligned}&\partial_t T_{2u} + \partial_y\psi_0\partial_x T_{2u} - \partial_x\psi_0\partial_y T_{2u} - \partial_{xx}T_{2u} - \partial_{yy}T_{2u}\\ &= \frac{1}{2}\cos\left(\frac{4\pi t}{\tau_j}\right)\left(-\partial_y f_j\partial_x T_{1A} + \partial_x f_j\partial_y T_{1A}\right) + \frac{1}{2}\sin\left(\frac{4\pi t}{\tau_j}\right)\left(-\partial_y f_j\partial_x T_{1B} + \partial_x f_j\partial_y T_{1B}\right)\end{aligned} \tag{19}$$

and

$$\begin{aligned}&\partial_y\psi_0\partial_x T_{2s} - \partial_x\psi_0\partial_y T_{2s} - \partial_{xx}T_{2s} - \partial_{yy}T_{2s}\\ &= -\frac{1}{2}\partial_y f_j\partial_x T_{1A} + \frac{1}{2}\partial_x f_j\partial_y T_{1A} - \partial_y\psi_2\partial_x T_0 + \partial_x\psi_2\partial_y T_0\end{aligned} \tag{20}$$

Analogous to the T_1 case, (19) implies that $T_{2u}(x,y,t) = T_{2u,1}(x,y)\cos(4\pi t/\tau_j) + T_{2u,2}(x,y)\sin(4\pi t/\tau_j)$. This means that the unsteady part of T_2 does not contribute anything to Nu_2. It suffices to evaluate for T_{2s} in (20) to calculate Nu_2, given by

$$Nu_2 = \frac{1}{\tau_j L_x}\int_0^{\tau_j}\int_0^{L_x} -\partial_y T_{2s}\Big|_{y=0} dxdt. \tag{21}$$

Notice that the right hand side of (20) requires T_{1A}. T_{1A} can be solved for via the following system of coupled equations:

$$\frac{2\pi}{\tau_j}T_{1B} + \partial_y\psi_0\partial_x T_{1A} - \partial_x\psi_0\partial_y T_{1A} - \partial_{xx}T_{1A} - \partial_{yy}T_{1A} = -\partial_y f_j\partial_x T_0 + \partial_x f_j\partial_y T_0 \tag{22}$$

$$-\frac{2\pi}{\tau_j}T_{1A} + \partial_y\psi_0\partial_x T_{1B} - \partial_x\psi_0\partial_y T_{1B} - \partial_{xx}T_{1B} - \partial_{yy}T_{1B} = 0. \tag{23}$$

(22) and (23) are obtained by directly substituting the expression for T_1 into (15), then regrouping all terms proportional to $\cos(2\pi t/\tau_j)$ into (22) and all terms proportional to $\sin(2\pi t/\tau_j)$ into (23).

In summary, calculating an approximation of Nu involves solving for T_0, T_{1A}, T_{1B}, and T_{2s}, which involves solving three steady PDEs. This should be much less computationally expensive than directly solving the unsteady advection-diffusion equation with many time steps.

3.1 Checking the estimate of Nu

From above, we claim that we can estimate Nu as $Nu_0 + \epsilon^2 Nu_2$, where Nu_2 is the leading order change in heat transfer due to the unsteady perturbation added to the optimal steady flow in (4). We wish to check that this approximation is close to the true Nu. To that end, we first fix the stream function ψ_j, which is dictated by setting ϵ, choosing coefficients of f_j and setting $\tau_j = 0.5$. The coefficients of f_j are determined by initializing a vector of coefficients b_j, with entries independently sampled from a standard normal distribution. After, we find $Nu_0 + \epsilon^2 Nu_2$: we first compute Nu_0 using T_0 solved from (14). Then, we compute T_{2s} via the three steady PDEs derived above, and use T_{2s} to finally compute Nu_2. We check this method of approximation against numerically solving the unsteady advection-diffusion equation using the Crank-Nicolson scheme (run with $t = 10 \cdot \tau_j$ and the same prescribed stream function ψ_j as in the approximation method).

From a test case with $\epsilon = 0.05$ and $\tau_j = 0.5$, the two different methods described above appear to yield very similar computations. The temperature field $T(x, y, t)$ acquired by solving the advection-diffusion equation using Crank-Nicolson is very similar to that of the approximated temperature field $T_0(x, y) + \epsilon T_1(x, y, t) + \epsilon^2 T_{2s}(x, y)$ calculated from the three steady PDEs described above. (The time-averaged values of these two temperature fields only differ by values in the order of 10^{-5} at worst.) Most importantly, $Nu - Nu_0 - \epsilon^2 Nu_2$ is calculated to be $-5.8128 \cdot 10^{-5}$, which means the estimate $Nu_0 + \epsilon^2 Nu_2$ is sufficiently close to Nu. We expect this estimate to improve as the choice of ϵ is made smaller (that is, decreasing the influence of the added unsteady perturbation in (4)).

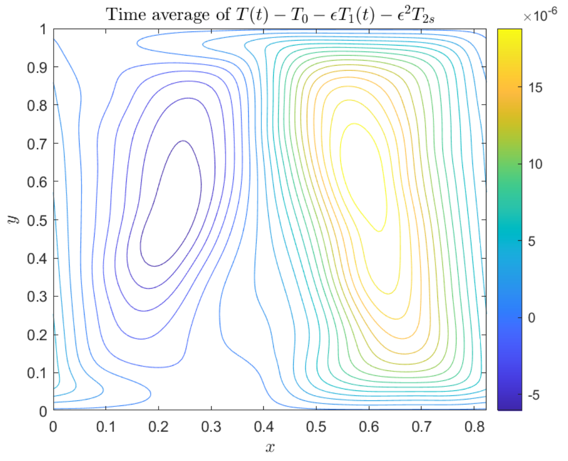


Figure 3: The time-averaged difference between $T(x, y, t)$ and $T_0(x, y) + \epsilon T_1(x, y, t) + \epsilon^2 T_{2s}(x, y)$.

For both methods above, we discretize the x-domain uniformly with 129 points. Thus, the spacing between adjacent points in the x-direction is $\frac{L_x}{128}$. In the y-direction however, we discretize with a non-uniform grid with 129 points. To construct this non-uniform grid, we start with a uniform discretization of 129 points for the y-domain (so the grid spacing is also $\frac{1}{128}$). Then, we map each point γ in the uniform y-discretization to our desired points γ^* for the non-uniform y-discretization via

$$\gamma^* = \gamma - \frac{0.997}{2\pi}\sin(2\pi\gamma). \tag{24}$$

This mapping ensures that the grid spacing is minimum near the boundaries $y = 0$ and $y = 1$, while maximum near the center $y = 1/2$. Past results show potential for complex branching behavior near the boundaries of optimal steady flows, so having a finer mesh grid near the boundaries to handle this added complexity is necessary for accurate computations of the temperature fields [1]. For subsequent trials described in next sections, we keep this choice of domain discretization constant.

4 Optimization method

To search for a set of locally optimal stream functions, we use the genetic algorithm which is inspired by natural selection in biological evolution [4]. This entails initializing a population of stream functions ψ_j with different choices of coefficients for f_j (see Section 2.1 for further details on the construction of the ψ_j's). In other words, we initialize coefficient vectors b_1, ..., b_N, where N is the population size, to define our initial population of ψ_j's. For this paper, the entries of the b_j's are sampled from the standard normal distribution, then normalized to each have norm 1. For a fixed population, we choose to keep τ_j fixed to maximize computational efficiency. (The matrices involved in solving for T_{1A} and T_{1B} in (22)-(23) can be kept constant for all members of the population if τ_j is fixed. This greatly decreases the amount of time spent to solve for T_{1A}, T_{1B} for each population member, as Matlab's backslash operator for solving matrices can be used here with many right hand side vectors at a time.) We choose to set $N = 1000$ for the initial population size.

For each population member, we solve for its associated Nu_2 using the method detailed in Section 3. Solving for Nu_2 is sufficient to see which population members are locally optimal, as Nu_0 is the same across all the population members. We select the b_j's that correspond to the Nu_2's in the best performing half, and duplicate those b_j's. To create "generation 2", we perturb these b_j's by adding a small perturbation to each entry of each b_j. For this paper, the small perturbation added to each coefficient is of the form $(0.1/N_g)$ times ρ, where ρ is a random number sampled from the standard normal distribution and N_g is the generation number. The added perturbations are independently sampled from each other. In other words, a sampled small perturbation is not held constant for all entries of a given b_j coefficient vector. Note also that the population size of "generation 2" is still $N = 1000$, the population size of "generation 1".

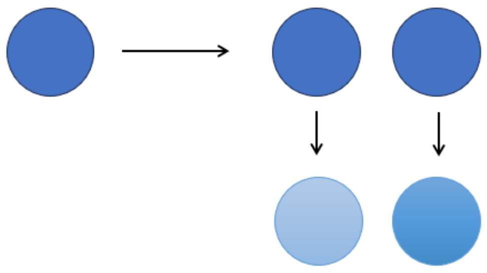


Figure 4: A schematic of the "duplication" process followed by the "perturbation" process. A member of the top-performing portion from a generation, the "parent", is used to create two "daughter" members in the next generation.

We repeat the process of calculating Nu_2 for each population member from "generation 2" and selecting the b_j's that correspond to the Nu_2's in the best performing portion. However, this time, we select only the b_j's corresponding to the top 25 Nu_2 values from "generation 2." Following the same procedure used to create "generation 2", we duplicate each of these top 25 b_j's and add a small perturbation to each entry of each b_j to create "generation 3." We then repeat the above steps of calculating Nu_2 for each population member from "generation 3" and picking the top 25 b_j's to produce "generation 4" consisting of 50 members. For subsequent generations, the population size of 50 is held. We run this genetic algorithm for a total of 1500 generations and repeat the genetic algorithm for new initial populations with different choices of fixed τ_j.

The reason we choose to run "generation 1" and "generation 2" of the genetic algorithm with 1000 members is simple: to gain access to top performing outliers. Starting "generation 1" with only 50 members makes it unlikely to encounter such top performing outliers. Starting "generation 1" with 1000 members and decreasing the population size to 50 at "generation 3" does not make a difference as the number of generations run approaches infinity. But, for shorter genetic algorithm algorithm runs of, say, 500 generations, this "switch" in population size enables improved Nu_2 values by the last generation with a near-negligible increase in computational time.

5 Preliminary results

5.1 Effects of parameters b_j and τ_j on Nu_2

Without running any subsequent generations, we ran a simulation with a population comprised of 50000 members. These members were formed by creating an array of 50 τ_j's and 100 randomly sampled b_j's vectors (entries independently sampled from a standard normal distribution), then calculating the Nu_2 for all possible combinations. The τ_j's range from 0.01 to 0.5, with finer incrementing at smaller values. Prior simulations with a fixed b_j showed a roughly monotonic relationship between the choice of τ_j and the top Nu_2 for that given τ_j, with sharp increase/decrease at small τ_j values followed by a plateau in the top Nu_2 value achieved past $\tau_j \approx 0.5$. So, we wanted to focus our investigation on smaller τ_j values, hence the limited range of τ_j values chosen.

Interestingly, of the two parameters b_j and τ_j, the choice of b_j appears to have a much stronger influence on the resulting Nu_2 value than the choice of τ_j. The top graph of Figure 5 shows that there is a "dominant" b_j that yields the best Nu_2 regardless of the choice of fixed τ_j. In this particular simulation, all the τ_j's but $\tau_j = 0.01$ share the "best" b_j.

Nevertheless, there still appears to a trend in the choice of "favorable" τ_j's. The bottom graph of Figure 5 has scattered dots outside of $\tau_j = 0.01$, but most of the dots are concentrated near $\tau_j = 0.01$. For a fixed b_j, a dot not at $\tau_j = 0.01$ or $\tau_j = 0.5$ indicates a non-monotonic relationship between τ_j and the top Nu_2 achieved for the given τ_j. A dot at $\tau_j = 0.01$, on the other hand, is a necessary condition for a monotonic (decreasing) relationship between τ_j and the top Nu_2 achieved for each τ_j (for a fixed b_j).

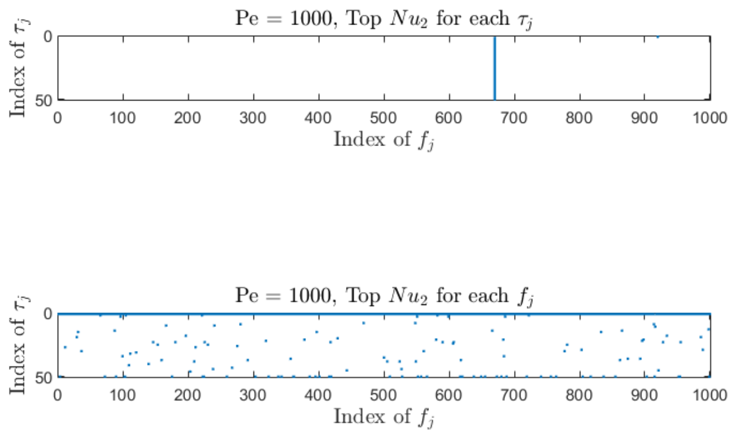
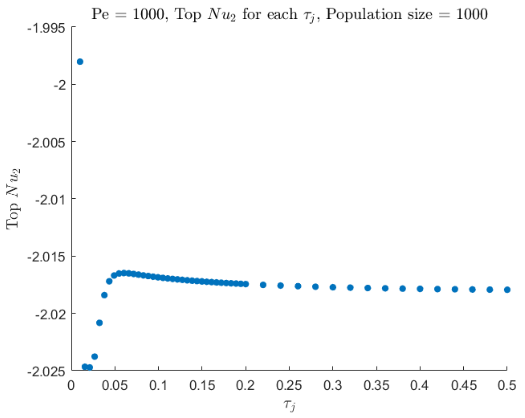


Figure 5: Top row: The index of f_j that yields the top Nu_2 for a fixed τ_j. The τ_j's are in increasing order - that is, $\tau_1 = 0.01$ and $\tau_{50} = 0.5$. Bottom row: The index of τ_j that yields the top Nu_2 for a fixed f_j.

When the top Nu_2 for each τ_j is plotted, there is still an exhibited sharp decrease at smaller τ_j values followed by a plateau. Additionally, it is pertinent to note that the top Nu_2 values achieved are negative.

That is, without any optimization strategies run, the addition of the unsteady perturbation term to the optimal steady stream function does not improve the total averaged heat transfer Nu.



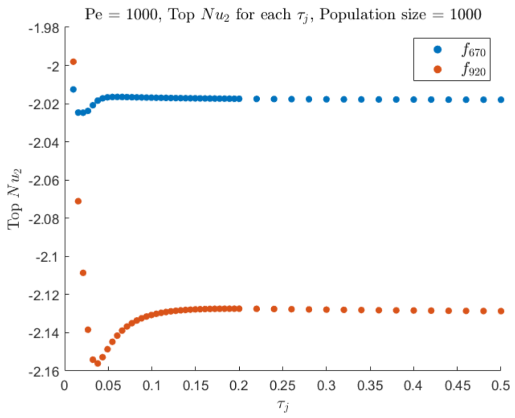


Figure 6: Left: The top Nu_2 for each fixed τ_j. Population size = 1000 means 1000 choices of fixed b_j shared across all τ_j's. Right: The Nu_2's over the τ_j's yielded by the two optimal f_j's.

5.2 Genetic algorithm run with a range of τ_j's

We ran the genetic algorithm with the same choice of τ_j's as described in Section 5.1. For each of the 50 initialized populations (each corresponding to a τ_j), we opted to run 1500 generations. Note that the initial coefficient vectors b_j were independently sampled across the different τ_j's.

After 1500 generations, there appears to be a small improvement in the top Nu_2 values achieved compared to the Nu_2 values seen at generation 1 in a separate simulation (see Section 5.1). Nevertheless, the values are still negative. Additionally, the clear relationship exhibited in Section 5.1 between τ_j and the top Nu_2 (for a fixed b_j) is not present here. There does not seem to be a correlation in the τ_j and top Nu_2 achieved.

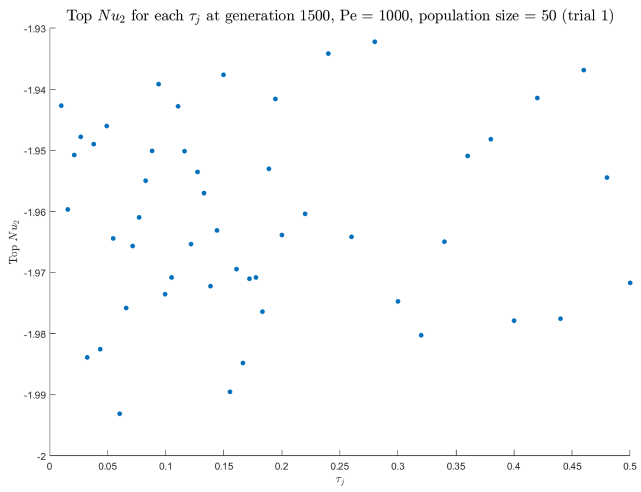


Figure 7: The top Nu_2 achieved for each τ_j at the last generation run, generation 1500.

Additionally, it is difficult to make out any distinguishing features in the f_j's corresponding to the top Nu_2's over the τ_j's, despite the similarity in the Nu_2 values across the different τ_j's.

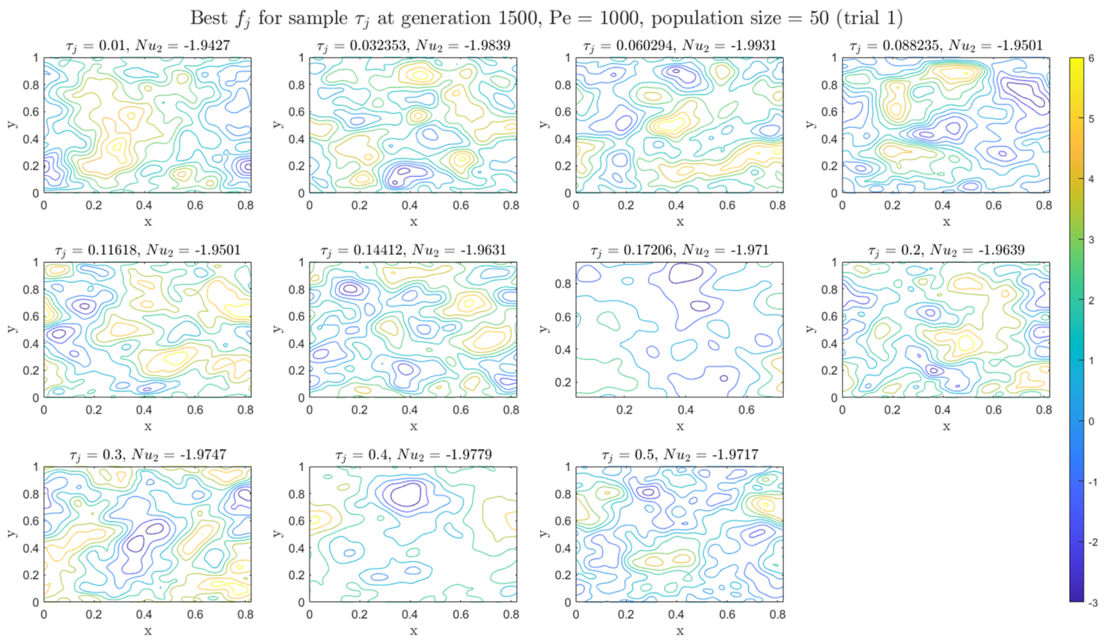


Figure 8: Contour plots of the f_j corresponding to the top Nu_2 for select τ_j at the last generation run, generation 1500.

5.3 Case study with $\tau_j = 0.5$

It would have been preferable to run each the genetic algorithm for each τ_j above with more generations. However, due to computational time, we were not able to do so and instead, opted to run a longer genetic algorithm simulation with 20000 generations for a sample $\tau_j = 0.5$. Despite the increase in generation ran, there does not appear to be a significant improvement in the top Nu_2 achieved by the final generation. (The top Nu_2 for $\tau_j = 0.5$ from Section 5.2 after 1500 generations was -1.9717, compared to a top Nu_2 of of -1.9319 here after 20000 generations.) It also appears that this version of the genetic algorithm allows for the most improvement in the initial generations, followed by continued yet smaller steady improvements as the generations progress.

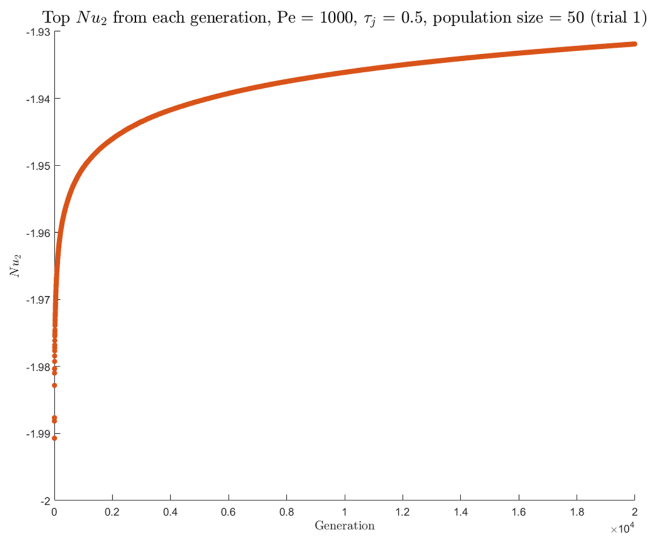


Figure 9: The progression of the top Nu_2 value from each generation.

Additionally, the optimal f_j's associated with the top 5 Nu_2 values at the last generation run look different from the optimal f_j at generation 1500 for $\tau_j = 1500$ from Section 5.2. This could be because of the difference in generations run. However, in past simulations the shapes of optimal f_j do not change much from, say, generation 1000 to generation 20000. (The f_j values near $y = 0$ change slightly, accounting for the small but steady increase in Nu_2 over the generations.) So, the difference could simply be a case of an alternative locally optimal f_j found by the genetic algorithm. Also observe that the f_j's corresponding to the top 5 Nu_2 values (which happen to all be the same) at generation 20000 look indistinguishable from one another. That is, the genetic algorithm converges to a locally optimal f_j.

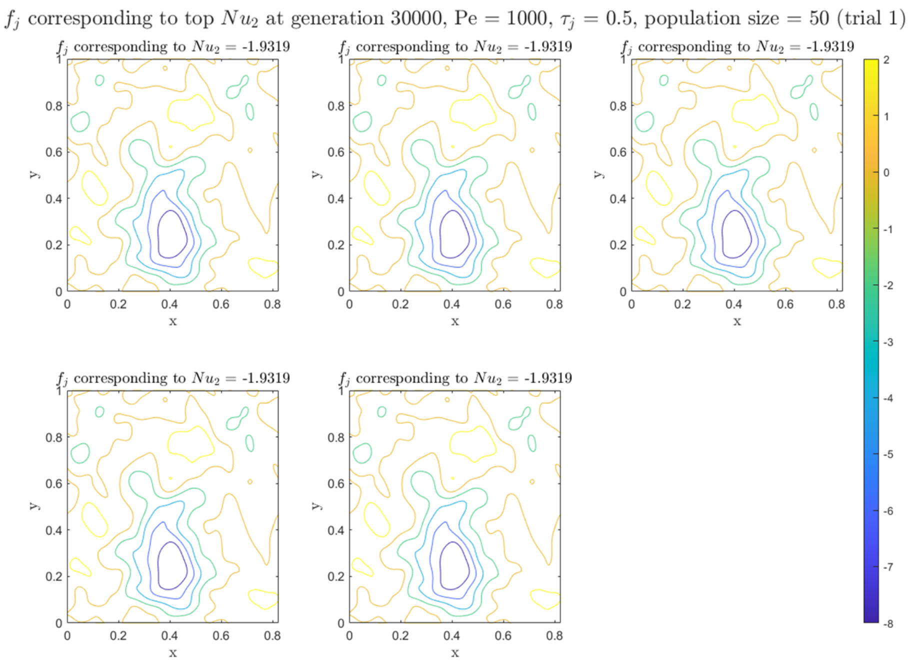


Figure 10: Contour plots of the f_j's corresponding to the top 5 Nu_2 values at the last generation, generation 20000.

6 Conclusions and discussion

In this paper, we build upon the work done by Alben in [1] by investigating heat transfer in a 2D domain with unsteady fluid flows of a fixed viscous energy dissipation Pe = 1000. These unsteady fluid flows are formed by adding small unsteady perturbations to the optimal steady flow found by Alben in [1]. To shorten the computational time of computing the Nu associated with each sample fluid flow, we decompose the unsteady advection-diffusion equation into three steady PDEs. We then calculate Nu_2, the leading order change in averaged heat transfer due to the added small unsteady perturbation to the optimal steady flow. Preliminary simulations show that of the two parameters responsible for uniquely determining ψ_j, b_j (the coefficients of f_j) and τ_j, b_j has a much more dominant influence on the resulting Nu_2 value. However, for a fixed choice of b_j, there is a roughly monotonic relationship between τ_j and locally optimal Nu_2 for each τ_j. Both increasing and decreasing relationships have been observed in simulations, depending on the choice of b_j.

In an attempt to find locally optimal perturbed versions of the optimal steady flow in [1], we use the genetic algorithm. Genetic algorithm imulations have shown that the locally optimal Nu_2 inproves consistently as the number of generations run increases. However, the optimal Nu_2 values fail to be nonnegative. This means that instead of improving the averaged heat transfer Nu, the choices of added small unsteady perturbations used in this paper actually decrease Nu.

Since the choice of b_j has the largest influence on the Nu_2 of ψ_j, a pertinent next step would be to try more combinations for b_j. In this paper the b_j's are formed by sampling entries from a standard normal distribution. It is unclear if certain flow modes in (5) drastically worsen or improve Nu_2. To examine this, for each $j = 0, 1, \ldots, 357$, we could set $b_j = [0, 0, \ldots, 1, 0, \ldots, 0]$ (where the j-th entry is 1 and 0 elsewhere), and calculate Nu_2 for ψ_j formed with each b_j and a fixed τ_j. We could then place more weight on the flow modes with better Nu_2 in hopes of improving Nu_2.

The setup of the genetic algorithm can also be reexamined beyond simply increasing the sample size or the number of generations run. As exhibited in Section 5.3, while the locally optimal Nu_2 value consistently increases, it increases rather slowly (especially at later generations). This is probably because of the choice of "perturbation update" used in this paper to create daughter flows. $0.1/N_g$ multiplied by a randomly sampled number from a standard normal distribution, the number added to a flow mode coefficient to perturb it, becomes very small as the generation number N_g goes to infinity. This means at later generations, adding a number of this form to a coefficient essentially does not change the coefficient. This also means the overall ψ_j barely changes, hence the very slow change in Nu_2. Thus, an idea to speed up the current version of the generation algorithm would be to increase the magnitude of the "perturbation update". We could instead add 0.1 multiplied by a randomly sampled number from a standard normal distribution to each flow mode coefficient. Then, the average magnitude of such random perturbations will not decrease over the generations, which might enable us to see better Nu_2 at earlier generations.

Lastly, it would be interesting to run simulations for different fixed average rate of viscous energy dissipation Pe^2. Perhaps at larger Pe^2 values, we will be able to see improved Nu values by considering slightly perturbed versions of optimal steady flows with that given Pe^2. The procedure to calculate Nu_2 discussed in Section 3 would need to be modified: the domain discretization would need to be made finer, to account for the complex branching behavior exhibited by optimal steady flows at higher Pe^2 values in [1]. This would increase the computational time needed to run genetic algorithm simulations.

7 Acknowledgements

I would like to extend my thanks to my principal investigator, Dr. Silas Alben, for the project idea and the welcome stay in Ann Arbor this summer. I would also like to thank Dr. Xiaojia Wang for her patience and guidance in discussing computational methods and reviewing Matlab code with me. Lastly, I would like to thank the U(M) Math REU coordinators for organizing the logistics of this summer's REU.

References

[1] Silas Alben. "Transition to branching flows in optimal planar convection". In: *Physical Review Fluids* 8.7 (2023), p. 074502.

[2] Pedram Hassanzadeh, Gregory P. Chini, and Charles R. Doering. "Wall to wall optimal transport". In: *Journal of Fluid Mechanics* 751 (June 2014), pp. 627–662.

[3] R Huilgol and N Phan-Thien. "7 - Computational Viscoelastic Fluid Dynamics". In: *Fluid Mechanics of Viscoelasticity*. Vol. 6. Rheology Series. Elsevier, 1997, pp. 397–472.

[4] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future". In: *Multimedia tools and applications* 80 (2021), pp. 8091–8126.

[5] Singiresu S. Rao. "Chapter 17 - Basic Equations of Fluid Mechanics". In: *The Finite Element Method in Engineering (Fifth Edition)*. Ed. by Singiresu S. Rao. Fifth Edition. Boston: Butterworth-Heinemann, 2011, pp. 549–569.

[6] P Teertstra, MM Yovanovich, and JR Culham. "Analytical forced convection modeling of plate fin heat sinks". In: *Journal of Electronics Manufacturing* 10.04 (2000), pp. 253–261.