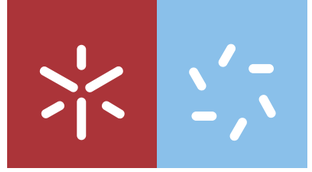
Alexandre Silva **Study of friction mechanisms between texturized surfaces**

UMinho | 2021



Universidade do Minho
Escola de Ciências

Alexandre Daniel Mendonça Faria da Silva

Study of friction mechanisms between texturized surfaces

dezembro 2021

Universidade do Minho
Escola de Ciências

Alexandre Daniel Mendonça Faria da Silva

Study of friction mechanisms between texturized surfaces

Dissertação de Mestrado
Mestrado em Física

Trabalho efetuado sob a orientação do:
Doutor Veniero Lenzi
e do
Professor Doutor Luís Silvino Alves Marques

dezembro 2021

Acknowledgements

Firstly, I must state my profound gratitude toward my supervisors, Dr. Veniero Lenzi and Prof. Luís Marques for their support during the development of this work. In particular, to Dr. Veniero Lenzi for the scientific discussion that motivated me to develop each component, for his guidance in the ways of scientific research, for his continuous advice on the improvement of this thesis and also for discussions on possible future work. To Prof. Luís Marques for motivating me to pursue and develop research in the field of computational physics ever since my bachelors degree, I'm deeply grateful for his continuous guidance in the field. I must also acknowledge and thank Dr. Sergey Pyrlin for the scientific discussions on machine learning and particularly for his help in advancing this chapter. As a whole, I'm very grateful to every member in the computational and theoretical physics laboratory for all the support.

I would also like to thank my family, namely my mother for her continuous support during the development of this thesis. To my girlfriend in particular during the pandemic for enduring my stress and never-ending thoughts about the development of this work. It would not have been possible without their support.

This work was supported by the Portuguese Foundation for Science and Technology (in the framework of the Strategic Funding UIDB/04650/2020 project PTDC/EME- SIS/30446/2017 and Advanced Computing Project CPCA/A2/4513/2020, awarded by FCT I.P., for accessing MACC-BOB HPC resources.

Statement of Integrity

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Resumo

Estudo dos mecanismos de fricção entre superfícies texturizadas

A necessidade de reduzir o impacto ambiental das atividades humanas lidera o esforço coletivo na procura pela redução do atrito entre as partes móveis em contato lubrificado em dispositivos mecânicos, como motores de combustão. Entre as várias soluções possíveis, a investigação em tribologia identificou as técnicas de texturização de superfície entre as mais promissoras na redução do atrito. Nesse sentido grande parte dos esforços científicos tem sido na procura pela melhor textura que reduza a fricção para uma determinada aplicação. Embora tenham sido testadas diversas texturas com padrões regulares e também texturas inspiradas na natureza, a identificação da textura otíma é puramente baseada em abordagens de tentativa e erro. Estas experiências tem um custo em tempo e também monetário intrínsecamente associado, o que torna este problema de otimização ineficiente. Métodos numéricos podem certamente acelerar este processo, mas ainda assim o problema de otimização da textura é demasiado complexo para ser resolvido diretamente, devido ao tamanho do conjunto de texturas possíveis e pela relação não óbvia entre texturas e atrito.

Nesta tese, desenvolvemos e aplicamos uma ferramenta baseada em aprendizagem automática para superar estas limitações e resolver o problema de otimização da textura. A nossa abordagem consiste na implementação de uma rede neuronal (DNN) capaz de previsões precisas em comparação com abordagens experimentais e numéricas, estas tipicamente são lentas e ineficazes à base de tentativa e erro. Para gerar o conjunto de dados para treino da DNN de forma precisa e rápida, desenvolvemos uma ferramenta baseada no método de elementos finitos para a resolução da equação de Reynolds. A nossa implementação resulta na produção de resultados até duas ordens de magnitude mais rápidamente quando comparadas às alternativas na literatura. Graças à inclusão da iteração inexata de Newton que permite um tratamento correto dos limites das regiões de cavitação conseguimos a obtenção de resultados fisicamente robustos. Finalmente, usamos a DNN para calcular o atrito para todos os padrões possíveis no espaço de configuração da textura, reduzindo assim o problema de otimização da textura a uma simples operação de procura de informação.

O objetivo desta tese é demonstrar como as técnicas numéricas e de aprendizagem automática podem resolver problemas de outra forma intratáveis, mas também apresentar a aprendizagem automática como uma opção viável e bem-sucedida na investigação em física, particularmente em problemas de otimização.

Palavras-chave: fricção, texturização, tribologia, método dos elementos finitos, aprendizagem automática.

Abstract

Study of friction mechanisms between texturized surfaces

The need of reducing the environmental footprint of human activities is leading the demand for friction reduction between moving parts in lubricated contact in mechanical devices, such as combustion engines. Within the various possibilities, research on tribology has identified surface texturing techniques among the most promising ones. In this regard, much effort is put in looking for the best friction-reducing texture in a specific application. Even if diverse textures, from regular patterns to nature-inspired ones, have been tested, the identification of the best performing one is still based upon a trial and error approach. Such experiments have a high cost in time and money, and are intrinsically unable to find the best friction-reducing pattern. Numerical methods can certainly speed up this process, but still the texture optimization problem is too complex to be directly solved, because of the size of the set of possible textures and by the non obvious relationship between textures and friction.

In this thesis, we develop and apply a machine learning based tool to overcome these limitations and solve the texture optimization problem. Our machine learning approach consists in a deep neural network (DNN) capable of accurate predictions compared to otherwise slow and ineffective trial-and-error experimental and numerical approaches. To generate the necessary training data set for the DNN in an accurate and fast way, we developed a finite element method solver for the Reynolds equation. Our implementation results in speedups of up to two orders of magnitude when compared to alternatives in literature, while remaining physically accurate thanks to the inclusion of the Inexact Newton iteration that allows for a correct treatment of the cavitation boundaries. Finally, we use the DNN to solve the inverse problem, by calculating the coefficient of friction curve for all the possible patterns in the given texture space, thereby reducing the texture optimization problem to a simple data search.

The aim of this thesis is not only to demonstrate how numerical and machine learning techniques can solve otherwise untractable problems, but also to introduce machine learning as a viable and successful option in physics research, particularly in optimization problems.

Keywords: friction, texturing, tribology, finite element method, machine learning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Friction and wear are ubiquitous phenomena and have profound and tangible effects in our environment. On a large scale, they contribute in shaping our world through a large series of phenomena, which directly affects our environment. For instance, the consistent erosion from high velocity flow on the river bed causes rivers to change their course [3]; the erosion of seabed causes its roughness to change, inducing alterations on wave formation patterns [4, 5] which must be taken into account for seaside construction safety. The presence of friction is fundamental to interact with the environment: without it we could not walk or hold any object, making our living impossible. As humans, we even evolved to detect friction with our tongue, for example, to avoid foods that could wear our teeth significantly [6].

As friction is a natural component of our world, animals have learned how to adapt to it through many methods, through millions of years of evolution. Common fishes display remarkable capabilities of reducing drag while swimming by secreting mucous, studies have shown that some species are capable of reducing drag by up to 66% [7] improving swimming speed, stamina and helping with safety from predators. The top part of the body of orcas, sharks, manta rays and dusky dolphins evolved to be darker when compared to the bottom part of their body, so that, by absorbing a higher quantity of radiation thus increasing their skin temperature they alter the conditions of boundary flow. Interestingly, numerical calculations in [8] demonstrated that this coloring scheme actually reduces drag.

Apart from color and substances secretion, one of the most effective way in which nature controls friction is by surface texturing. The skin of snakes and certain lizards developed scales that reduce dry contact friction, the effect of which has been successfully replicated experimentally with different materials and has been shown to provide a significant friction reduction [9]. The same reduction has been observed in sharks, whose skin is covered by denticles which, combined with the secretion of mucous, provides an effective drag reduction [10]. Conversely, specific patterns found in tree toads feet have been shown to

increase friction, thus granting them a better grip on vertical surfaces [11], the same principle is behind the amazing capability of geckos to get grip and walk on glass [12]. Hence, nature through evolution, found that surface texturing is an extremely effective way to control friction, both in reducing it as for water-based life, or to increase it to allow grip on vertical surfaces. As we will see below, one of the most important fields in which we can draw inspiration from nature is surface engineering.

Concerning our activity as humans, friction reduction is nowadays more important than ever, since it directly relates with the reduction of our environmental impact. Transportation is one of the largest industries, as well as one of the most responsible ones in polluting emissions, and more than a third of all fuel energy in passenger cars worldwide is used with the sole purpose of overcoming friction forces [13].

Advancements in the study of friction and wear, termed tribology by Jost [14], allowed us to consistently perform measurements of these quantities in controlled laboratory conditions, and opened the way to the study of those mechanism that could reduce - or increase - friction[15]. Tribometers in configurations such as pin-on-disc with lubricated contacting surfaces were a major developments in experimental tribology [16]. By quantifying friction as a function of the relative sliding speed of the contacting surfaces, a curve that is a major fundamental concept in tribology [17] can be drawn. The Stribeck curve, as it is called, allowed for direct experimental insight into the different regimes of friction [18].

The main advancement of tribology as a field began mainly as an effort to reduce overall costs within the industrial sectors, in particular the transportation and energy sectors [19]. Through means of careful analysis of the current methodology available in tribology, the report by Jost et al. [14] and many following studies [20, 21] estimated potential savings in terms of total GDP of up to 1.4% [22] in industrialized countries. Particularly, in the transportation, energy and utilities sectors, where the application of new friction and wear reducing techniques were estimated to potentially save upwards of 11% of total energy expenses in first world countries [20, 22]. The combined impact in industry led to a rampant investment in research and development of new techniques that were proven to be incredible return-on-investment opportunities [19]. Quite recently in fact, research and development into environmentally friendly options came to the forefront in friction optimization due to European legislation (EURO 6) within the automobile industry additives banning the use of lubricants and surface coatings containing substances such as phosphorus and zinc [23]. Concerning its potential impact, advances in the field of tribology are important in addressing environmental and economical challenges as a society.

The reduction of friction between lubricated parts in contact within mechanical equipment, such as engine pistons and bearings, increasing their lifetimes and reducing friction losses is one of the fields in which tribology is more active [24]. In the interest of this goal, research on tribology has developed

different approaches such as the use of surface coatings [25, 26], research and devolopment of novel lubricant formulations [26] and finally the application of surface modification techniques [27]. There has also been a push in tribology research for the combination of multiple techniques [28].

As a matter of fact, thanks to advances in texturing techniques [21] leading to more efficient textured surface generation processes, and its environmentally friendly nature, surface texturing has been launched into the front-line of the global research effort in tribology [18]. Surface texturing combined with lubrication provides a very wide range of possible parameters that have a great impact on the overall performance of the system in contact [29]. When the goal is the optimization of a certain process, a wide variety of degrees of freedom is generally preferred, turning surface texturing into a major candidate for friction reduction, and even nature-inspired textures have been employed successfully [11]. However, the study of textured surfaces is a process with high associated costs, which typically requires pristine laser textured samples and complex experimental setups. Moreover, it is affected by substandard reproducibility of results, because the samples are subject to significant wear at every experimental run, which can significantly affect the outcomes [30].

Due to this, research and the resulting literature within the field of surface texturing has largely moved to a combination of numerical and experimental work [18]. Numerical simulations offer a natural way to tackle this problem, being significantly cheaper, faster and more easily reproducible with respect to experiments. The easy numerical representation of textured samples is also a major point as it allows us, for example, to easily reproduce experiments with bio-inspired textures [9] that are typically hard to manufacture and therefore experimentally harder to test.

There is a growing interest in numerical simulations [18] to perform the texture optimization of hydrodynamic lubrication problems by solving simplified models that use the Reynolds equation [31] derived from the more general, and computationally much more demanding, Navier-Stokes equations [32]. In [33], a finite element solver for the contact mechanics was employed together with a Reynolds equation solver to solve complex elastic-hydrodynamic (EHD) systems. Through coupling of the governing Reynolds equation with an energy equation, thermodynamic effects were also considered by Habchi et al. [34] for Newtonian fluids. The full compressible Navier-Stokes equations were also solved by Bruyere et al. [35] providing full physical description even across the lubrication film height. Effective methods combining both experimental and numerical work have also been developed to predict friction in pin-on-disc systems [36, 37] allowing also the study of a surface design made by rows of dimples, which are essentially wells in the material carved in by laser pulses. These dimples were placed on a texture with a particular shift between consecutive rows, and it was shown that there is an optimal shift value that minimizes friction

within the mixed lubrication regime. Other parameters are also commonly considered in research works, such as dimple shape, dimple density, dimple depth and dimple radius [21, 37, 38, 39]. In any case, a consensus on the underlying mechanisms behind friction reduction has not been reached within the scientific community. Moreover, neither of the above attempts at an approach to the optimization problem, that is, finding the best friction-reducing texture given a set of changing texture-defining parameters have been successful. One should calculate the Stribeck curve for every possible texture, the number of which can be exceedingly large to attempt a solution with standard numerical techniques.

In some of the previously mentioned numerical methods, a common pitfall in terms of physical accuracy is the exclusion of considerations about cavitation, as the pressure drop in the lubricant film can lead to the formation of vapor-filled cavities [40]. These cavities are crucial when combating friction and especially wear as they are one of the major driving forces behind the wearing out of contacting surfaces [41]. This exclusion from the numerical problem is due to the sharp increase in complexity required by its presence. The proper treatment of cavitation boundaries drastically changes the nature of the numerical problem into what is commonly known as a linear complementarity problem [42]. With increased computational complexity the required computational power also increases, because in this case the numerical problem requires a more careful treatment which typically leads to slower convergence [2, 43].

The introduction of exact treatments of cavitation boundaries, as the Fischer-Burmeister-Newton-Schur (FBNS) algorithm [44] represented an important step in solving lubricated contact problems by developing a mass-conserving cavitation scheme, which was further improved with the introduction of the inexact Newton iteration (INE) [2]. Unlike the former, the latter ensures that the non-negativity conditions of the linear complementarity problem are correctly treated, meaning that the solution is always reasonable from a physical point of view. In addition, the INE method presents an overall speed increase over previous methods, such as FBNS, of up to two orders of magnitude [2]. However, even considering the most recent advances in numerical methods, such as the INE implementation, the finding of an optimal texture within a texture optimization technique remains out of reach for classical approaches.

Machine learning encompasses a range of algorithms and modeling tools used for large data processing tasks and it is still relatively new as a wide-spread tool for physics research [45, 46]. It has two typical applications, being classification and regression problems. For example, in [47], a network was trained in order to attempt the classification of the phases of matter using Monte Carlo simulations of matter at different phases and temperatures. Extrapolating this classification to configurations not present in the training set lead to the possible determination of phase transitions and respective order parameters. In [48], recurrent neural networks were used to predict trajectories of chaotic dynamical systems for use in

weather prediction. In the field of optics, a deep neural network (DNN) was trained to replace, to a certain degree, a complex solver for the Maxwell equations [49], and was shown to be extremely fast in providing solutions.

The possibility of replacing complex systems of equations such as the Maxwell and Navier-Stokes equations and their respective boundary conditions with simple deep neural networks is very appealing. However, its success as a replacement for a fully fledged numerical simulation largely depends on the quality of its training, which is still entirely a physics problem. As noted in [50], while we can develop DNNs capable of accurately predicting results from a physics problem, the DNN itself is not capable of learning physics, it is only capable of predicting trends based on past experience. Results obtained from a DNN application should always be treated and regarded with this consideration in mind.

The aim of this thesis is to show how we can apply machine learning techniques to provide a fast and accurate solution for the texture optimization problem. To our knowledge, this approach is novel in the field of tribology. To achieve this, we will replace the traditional numerical method of solving the Reynolds equations which governs hydrodynamic lubrication problems, with a deep neural network (DNN) that, given a textured surface, is capable of predicting the corresponding Stribeck curves, with an accuracy level comparable to direct solution methods, but in a tiny fraction of their solving time. Since good quality data is required to train the DNN properly, we will develop a finite element method direct solver that is faster than the existing implementations and correctly calculates the cavitation regions by implementing the INE algorithm. Next, the direct solver will be used to generate the training data set and a DNN will be trained upon it, its performance will be evaluated. Finally, we will show how the DNN can be used to calculate the Stribeck curves for each one of the possible patterns in a 5x5 array of dimples, thereby reducing the texture optimization problem to a data search, some notes on possible future work to do with the texture optimization problem will also be given.

Chapter 2

Theoretical foundations and methods

This chapter provides an introduction of the main concepts used in this thesis, and details the methods adopted. Firstly, the definition of friction and wear, the different friction regimes in lubricated contacts and the governing equations of the hydrodynamic lubricated contact are introduced. Then, the finite element method technique is discussed and finally the main ideas behind machine learning and deep neural networks, as well as how they are implemented, are given.

2.1 Friction and wear

Kinetic friction occurs when two surfaces in contact move relative to each other. When a normal force F_N is applied to the surfaces to press them together, a friction force F_f is generated, which is linearly proportional to the normal force. The proportionality constant is known as the coefficient of friction μ_f:

$$F_f = \mu_f F_N \,. \tag{2.1}$$

In most industrially relevant applications, lubricant is always used to reduce friction between moving parts, where the lubricant is a fluid with desired characteristics. Thus, for the remainder of this work, lubricated contacts will always be assumed, unless otherwise specified.

The coefficient of friction is typically measured using a tribometer, a device that recreates the sliding motion of the surfaces and, given a known external load (the normal force), measures the friction force F_f, allowing to derive the coefficient of friction. A diagram representing a tribometer is reported in figure 1. Its configuration depends on which of the surfaces are in motion, the direction of the motion and of the type of contact, which can be conformal when the two surfaces are flat or non-conformal when one of the surfaces is curved. A measurement of the applied load F_N and the tangential force in the system F_f allows us to compute the coefficient of friction μ_f.

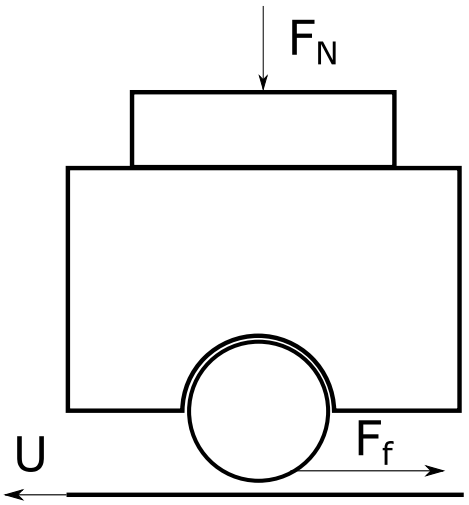


Figure 1: Schematic representation of a tribometer used to measure the coefficient of friction for a moving untextured surface in contact with a still non-conformal surface.

Considering different sliding velocities U for fixed applied force and lubricant, the tribometer can be used to measure the Stribeck curve, which reports the coefficient of friction as a function of the dimensionless quantity H, termed the Hersey number and defined as:

$$H = \frac{\mu U}{F_N} \,. \tag{2.2}$$

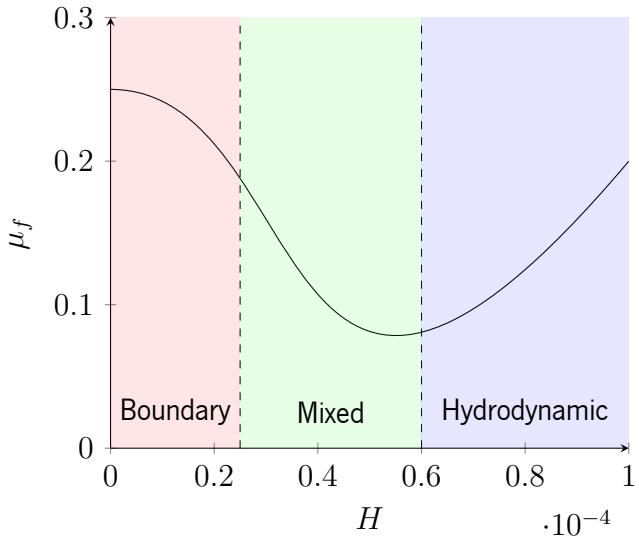


Figure 2: Stribeck curve as a function of the dimensionless Hersey number, the curve is split between three friction regimes: boundary friction region; mixed friction region; hydrodynamic friction region;

A typical Stribeck curve is shown in figure 2, where three different regimes of friction can be clearly distinguished. For low H, the boundary friction regime (in red) is characterized by a nearly dry contact,

meaning that the surfaces are very close and there is almost no lubricant between them. In dry contacts, friction forces are caused by the interlocking roughnesses of the two surfaces, yet at a deeper level, interactions at the molecular scale are also relevant, including changes in the composition of the surfaces due to reactions between themselves or with molecules present in the environment. At intermediate H, we can identify the mixed friction regime (in green). Although the surfaces can be regarded as still in contact, a drop in friction is observed due to presence of a lubricant film between them, which provides an additional hydrodynamic lift that counteracts the applied load. Finally, at large H, the hydrodynamic friction regime (in blue) is characterized by the absence of contact between the surfaces, while the applied load is entirely countered by the hydrodynamic lift of the lubricant. An increase in friction is observed again, but in this case its origin is hydrodynamic.

2.2 Reynolds equation

2.2.1 Derivation from Navier-Stokes

Within the lubricant film, the Navier-Stokes equations provide the most accurate way to model the physics of the fluid. However, with simple considerations in relation to hydrodynamic lubrication problems the Navier-Stokes equations can be simplified into a more commonly utilized formulation. The Reynolds equation which follows from the Navier Stokes has two very important advantages, it is easier to implement and less computationally complex [18].

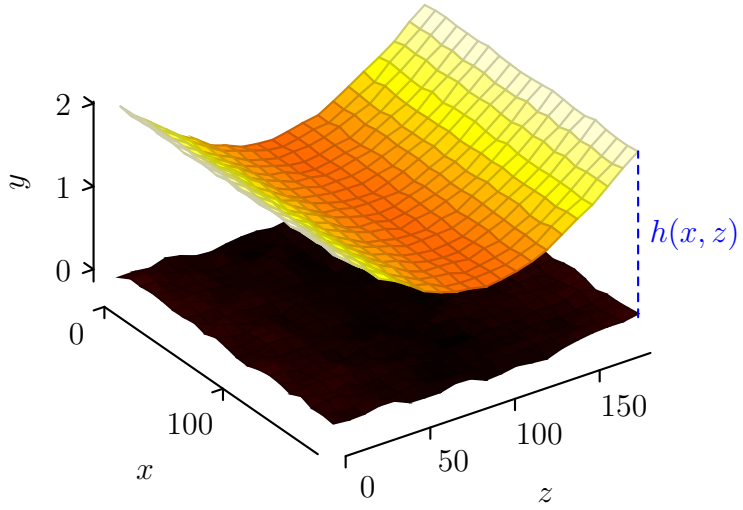


Figure 3: Schematic representation of the thin film height $h(x, z)$ for two contacting rough surfaces. The characteristic lengths for x and z are considered much bigger than $h(x, z)$.

Consider a thin film of lubricant between two surfaces, as represented in figure 3. The characteristic dimensions of the surfaces are L_{xz} in the x and z direction. The function $h(x,z)$ represents the distance between the two surfaces in the y direction and defines a characteristic length L_y approximately equal to the maximum value of $h(x,z)$.

Assuming a steady-state and incompressible flow of the lubricant film it is possible to write the Navier-Stokes equations as follows [51]:

$$\begin{cases} \boldsymbol{\nabla} \cdot \vec{\mathbf{u}} = 0 & (2.3) \\ \rho\left(\vec{\mathbf{u}} \cdot \boldsymbol{\nabla}\right)\vec{\mathbf{u}} + \boldsymbol{\nabla} p = \mu \nabla^2 \vec{\mathbf{u}}\,, & (2.4) \end{cases}$$

where $\vec{\mathbf{u}} = (u_x, u_y, u_z)$ is the lubricant flow velocity vector, p is the film pressure, ρ is the density of the lubricant and μ the lubricant viscosity. In order to derive the Reynolds equation from the set of 4 Navier-Stokes equations it is easier to consider their dimensionless version, obtained using the following relations:

$$\begin{cases} \tilde{x} = \dfrac{x}{L_{xz}} \quad \tilde{u}_x = \dfrac{u_x}{U_{xz}} & (2.5) \\ \tilde{y} = \dfrac{x}{L_y} \quad \tilde{u}_y = \dfrac{u_y}{U_y} & (2.6) \\ \tilde{z} = \dfrac{x}{L_{xz}} \quad \tilde{u}_z = \dfrac{u_z}{U_{xz}}\,, & (2.7) \end{cases}$$

where U_{xz} and U_y are characteristic velocities defined along the x, z and y axes. To obtain a dimensionless pressure we define the Reynolds number as

$$R_e = \frac{\rho U_{xz} L_y}{\mu}\,, \tag{2.8}$$

such that

$$\tilde{p} = R_e \epsilon \frac{p}{\rho U_{xz}^2} \text{ where } \epsilon = \frac{L_y}{L_{xz}}\,. \tag{2.9}$$

By substituting the coordinates and velocities in their dimensionless form into equation (2.3) and considering that $U_y = \frac{L_y}{L_{xz}} U_{xz}$ holds we obtain

$$\frac{\partial \tilde{u}_x}{\partial x} + \frac{\partial \tilde{u}_y}{\partial y} + \frac{\partial \tilde{u}_z}{\partial z} = 0\,. \tag{2.10}$$

Finally, replacing the dimensionless variables into equations (2.4) and dropping the tilde notation for clarity brings us to the set of equations

$$\begin{cases} R_e\epsilon\left(u_x\dfrac{\partial u_x}{\partial x} + u_y\dfrac{\partial u_x}{\partial y} + u_z\dfrac{\partial u_x}{\partial z}\right) = -\dfrac{\partial p}{\partial x} + \dfrac{\partial^2 u_x}{\partial y^2} + \epsilon^2\left(\dfrac{\partial^2 u_x}{\partial x^2} + \dfrac{\partial^2 u_x}{\partial z^2}\right) & (2.11) \\ \epsilon^2\left[R_e\epsilon\left(u_x\dfrac{\partial u_y}{\partial x} + u_y\dfrac{\partial u_y}{\partial y} + u_z\dfrac{\partial u_y}{\partial z}\right) - \dfrac{\partial^2 u_y}{\partial y^2} - \epsilon^2\left(\dfrac{\partial^2 u_y}{\partial x^2} + \dfrac{\partial^2 u_y}{\partial z^2}\right)\right] = -\dfrac{\partial p}{\partial y} & (2.12) \\ R_e\epsilon\left(u_x\dfrac{\partial u_z}{\partial x} + u_y\dfrac{\partial u_z}{\partial y} + u_z\dfrac{\partial u_z}{\partial z}\right) = -\dfrac{\partial p}{\partial z} + \dfrac{\partial^2 u_z}{\partial y^2} + \epsilon^2\left(\dfrac{\partial^2 u_z}{\partial x^2} + \dfrac{\partial^2 u_z}{\partial z^2}\right) & (2.13) \end{cases}$$

To further simplify these equations the following considerations are made: the lubricant viscosity is constant, the characteristic distances in the x, z directions are much larger than in the y direction, such that $\epsilon \to 0$; a small Reynolds number is assumed, such that a laminar flow regime is established and $R_e\epsilon \to 0$. Dropping the irrelevant terms and going back to the dimensional forms we get

$$\begin{cases} \dfrac{\partial p}{\partial x} = \mu \dfrac{\partial^2 u_x}{\partial y^2} & (2.14) \\ \dfrac{\partial p}{\partial z} = \mu \dfrac{\partial^2 u_z}{\partial y^2} & (2.15) \\ \dfrac{\partial u_x}{\partial x} + \dfrac{\partial u_y}{\partial y} + \dfrac{\partial u_z}{\partial z} = 0\,. & (2.16) \end{cases}$$

Since the pressure is constant across the thin film we can perform an integration twice and by considering that the surfaces move only along the x direction, such that

$$\begin{cases} u_x(0) = U_1; \quad u_x(h) = U_2 & (2.17) \\ u_z(0) = 0; \quad u_z(h) = 0\,, & (2.18) \end{cases}$$

the double integration yields

$$\begin{cases} u_x = \dfrac{1}{2\mu}\dfrac{\partial p}{\partial x}\left(y^2 - yh\right) + \left(1 - \dfrac{y}{h}\right)U_1 + \dfrac{y}{h}U_2 & (2.19) \\ u_z = \dfrac{1}{2\mu}\dfrac{\partial p}{\partial z}\left(y^2 - yh\right)\,. & (2.20) \end{cases}$$

The average value of u_y across the film is simply

$$\overline{u}_y = \int_0^h (x,z) u_y \, dy = \frac{\mathrm{d}h}{\mathrm{d}t}\,, \tag{2.21}$$

since we are considering a steady state solution we have $\overline{u}_y = 0$. By performing the same integration over the film in the continuity equation (2.16) we get

$$0 = -\int_0^h (x,z)\frac{\partial u_x}{\partial x}\, dy - \int_0^h (x,z)\frac{\partial u_z}{\partial z}\, dy\,. \tag{2.22}$$

Finally, substituting the relevant terms and performing the integrations yields the Reynolds equation [31] which allows us to compute the pressure profile p given a thin film height $h(x,z)$ and adequate boundary conditions

$$\boldsymbol{\nabla} \cdot \left(h^3 \boldsymbol{\nabla} p - 6\mu \vec{\mathbf{U}} h\right) = 0 \tag{2.23}$$

where $\vec{\mathbf{U}}$ is defined as the relative sliding velocity of the surfaces, normally chosen to only have an x component.

2.2.2 Cavitation

Equation (2.23) was derived under a set of precise hypotheses, and it clearly is an approximation of the Navier-Stokes equation, meaning that any results computed from it are considered valid only if the problem falls under the conditions defined as follows:

- Density ρ, viscosity μ and temperature T are constant across and through out the lubricant film.
- Pressure p is constant across the lubricant film.
- The physical dimension of the gap between the contacting surfaces is considered negligible when compared to the physical dimensions of the contacting surfaces (thin film approximation).
- The flow within the film is considered to be laminar which corresponds to low Reynolds numbers.
- Inertial effects which lead to the deformation of the sliding surfaces are entirely neglected.

Under these conditions, the problem can be described by the Reynolds equation which allows us to correctly compute the lubricant pressure profile p:

$$\boldsymbol{\nabla} \cdot \left(\rho h^3 \boldsymbol{\nabla} p\right) = 6\mu \vec{\mathbf{U}} \cdot \boldsymbol{\nabla} \left(\rho h\right) . \tag{2.24}$$

Without any loss of generality we can set the relative sliding velocity as being always directed along the x axis such that

$$\boldsymbol{\nabla} \cdot \left(\rho h^3 \boldsymbol{\nabla} p\right) = 6\mu U \frac{\partial(\rho h)}{\partial x} . \tag{2.25}$$

In some conditions, especially in divergent regions where $\frac{\partial h}{\partial x} < 0$, it is common to observe cavitation, which consists in the formation of vapour filled cavities within the fluid. Using the cavitation model introduced by Elrod and Adams in [43] based on the Jakobsson-Floberg-Olsson (JFO) model [52], we can rewrite the equation in terms of pressure p and cavitation fraction θ as follows

$$\boldsymbol{\nabla} \cdot \left(h^3 \boldsymbol{\nabla} p\right) + 6\mu U \frac{\partial(h\theta)}{\partial x} = 6\mu U \frac{\partial h}{\partial x} . \tag{2.26}$$

The cavitation fraction is defined as

$$\theta = \frac{\rho_0 - \rho}{\rho_0} \tag{2.27}$$

in active regions (without cavitation) where $p > 0$ the film density ρ is that of the reference ρ_0 while in inactive regions (with cavitation) we have $p = 0$ and ρ is no longer equal to the reference ρ_0 as the fluid is a mixture of liquid/gas.

These cavitation considerations complement equation (2.26) providing a closed set of equations that allow us to determine both the pressure p and cavitation fraction θ, this constitutes a linear complementarity problem (LCP) [42], we thus look to solve the following set of equations:

$$\begin{cases} \boldsymbol{\nabla} \cdot \left(h^3 \boldsymbol{\nabla} p\right) + 6\mu U \dfrac{\partial (h\theta)}{\partial x} = 6\mu U \dfrac{\partial h}{\partial x} & (2.28) \\ p\theta = 0 & (2.29) \\ p \geq 0 & (2.30) \\ \theta \geq 0\,. & (2.31) \end{cases}$$

2.2.3 Contact model

Overall friction in the system exists due to two components: pressure gradients present within the film and the direct interaction between asperities in the contacting surfaces. To account for both of these components we have to model how the surfaces interact down to the asperity scale. The contact model used in this work is that of the Greenwood and Tripp model [53], where the overall roughness is considered to be on both contacting surfaces. They show with this model that the load and area of contact remain independent of the detailed aspects of asperities in the surfaces, achieving a good approximation of real behavior with simple considerations.

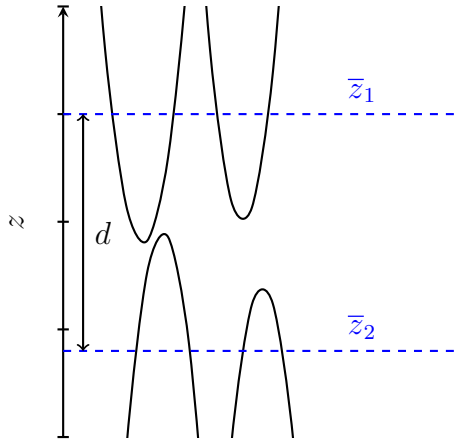


Figure 4: Schematic contact of two rough surfaces, all asperities are assumed to have the same shape, $\overline{z}_1$ and $\overline{z}_2$ are the reference planes of each surface corresponding to the average height of their roughness and d is the distance between these references.

The contact between rough surfaces can be seen microscopically as pairs of asperities modeled as identical contacting pairs of hills. These asperities are assumed to have a paraboloidal shape with an

average radius of curvature η, a distribution along both surfaces of density k and a gaussian asperity height distribution with standard deviation σ.

The sliding surface has a load F applied in the normal direction to the sliding, this force is opposed by two contributions: the hydrodynamic pressure and the load carried by the interaction between the rough asperities

$$W = W_h + W_a \tag{2.32}$$

where the load carried by the lubricant film is the direct integral of the generated pressure we calculate by solving the Reynolds equation

$$W_h = \int_\Omega p \, d\Omega \, . \tag{2.33}$$

The load carried by the interaction between asperities can be written as [53]

$$W_a = \frac{16\sqrt{2}}{5}\pi(\eta k\sigma)^2\sqrt{\frac{\sigma}{k}}E^{'}F_{5/2}\left(\frac{h}{\sigma}\right) \tag{2.34}$$

the dimensionless parameters $\eta k\sigma$ are known as the roughness parameter, the term σ/k is the typical asperity slope, h/σ is the Stribeck oil film parameter typically denoted by λ, $E^{'}$ is the reduced (effective) elastic Young modulus of the contacting surface pair. The statistical function $F_{5/2}(\lambda)$ introduces the Gaussian distribution of asperities which can be calculated from a parametric fit such as in [54].

The apparent contact area between the contacting surfaces can be obtained from the cavitation fraction profile θ, in regions where $\theta \approx 0$ we have a fluid lubricant region

$$A(x,y) = \begin{cases} 0 \text{ if } \theta(x,y) = 0 \\ 1 \text{ if } \theta(x,y) > 0 \end{cases}$$

however, the asperity contact area itself is calculated as [54]

$$A_a = \pi^2(\eta k\sigma)^2 A F_2\left(\frac{h}{\sigma}\right) \, , \tag{2.35}$$

where the statistical function $F_2(\lambda)$ can be estimated with a parametric fit such as in [54]. The asperity contact area is much smaller than the apparent contact area, so the viscous contact area A_v can be written as

$$A_v = A - A_a \approx A \, . \tag{2.36}$$

In the mixed regime of lubrication there are two contributions to overall friction: viscous shear of the lubricant and interactions between asperities

$$f_{total} = f_{viscous} + f_{boundary} \, , \tag{2.37}$$

the viscous shear of the lubricant can be obtained as a function of the lubricant film height h

$$\tau = \frac{h}{2}\frac{\partial p}{\partial x} + \frac{U\mu}{h}\,, \tag{2.38}$$

and the viscous friction is calculated as

$$f_v = \tau A_v \approx \tau A\,. \tag{2.39}$$

The boundary friction is obtained as a sum of two contributions: the direct contact of asperities and the non-Newtonian shear introduced by a thin film of lubricant entrapped between the interspatial cavities of asperities. We can write boundary friction as

$$f_b = \tau_0 A_a + \xi W_a \tag{2.40}$$

τ_0 is the Eyring shear stress, W_a is the load carried by asperities and ξ is the pressure coefficient of the boundary shear stress [51].

We can then calculate the total friction force as a sum given by

$$f_t = \tau A + \tau_0 A_a + \xi W_a\,, \tag{2.41}$$

the coefficient of friction can be calculated by integrating over the domain Ω

$$\mu_f = \frac{1}{F\mathcal{A}}\int_\Omega f_t\, d\Omega \tag{2.42}$$

where F is the applied load and $\mathcal{A}$ is the area of the domain.

2.3 Finite element method

2.3.1 Basic concepts

The finite element method is a powerful tool in its application to real world problems that involve complicated physics and geometry [55]. A complex domain defined by an arbitrarily curved boundary which is typically impossible to represent under finite difference methods is split up into a collection of simple subdomains called finite elements. It is in fact easier to represent a complicated function as a piecewise collection of polynomials, as represented in figure 5.

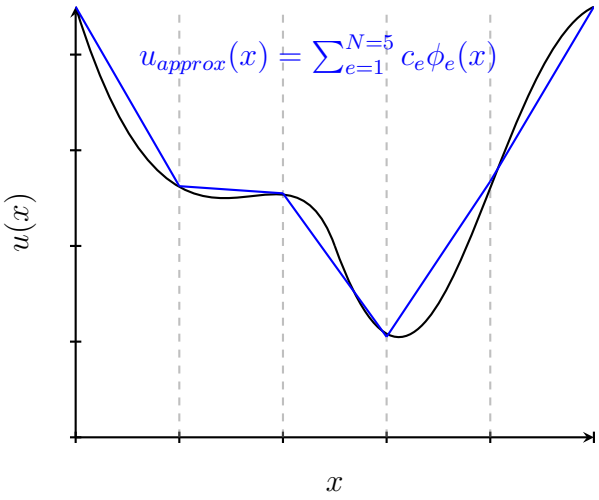


Figure 5: Piecewise first order approximation of a function $u(x)$ by a collection of 5 elements.

The coefficients c_e are entirely determined by satisfying the governing equations we are attempting to solve. The finite element method (FEM) steps are as follows:

- Divide the domain geometry into subdomains called finite elements, this collection is called finite element mesh.
- Seek an approximation of the solution as a linear combination of basis functions in these finite elements and derive the algebraic relations between nodal values of the solution for each element.
- Assemble the elements into a global system that fully determines the solution as a whole.

The finite element method can be viewed as an elementwise application of a variational formulation method [55], in theory our goal is to build a functional with respect to the variables that includes the governing equations of our problem, therefore by minimizing this functional form of the problem we approach the nodal solutions of our governing equations.

Consider a domain $x \in [0, L]$ and a simple differential equation commonly appearing in heat exchange problems

$$-\frac{\partial}{\partial x}\left[a(x)\frac{\partial u}{\partial x}\right] = f(x)\,, \tag{2.43}$$

for the function $u(x)$ subject to known boundary conditions

$$u(0) = u(L) = u_0\,, \tag{2.44}$$

suppose we seek an approximation of $u(x)$ in the form

$$u(x) \approx u_N(x) = \phi_0(x) + \sum_{j=1}^{N} c_j \phi_j(x)\,, \tag{2.45}$$

where $\phi_j(x)$ are the element basis functions. We must find the set of coefficients c_j that satisfy the differential equation (2.43), by substituting the approximation of $u(x)$ we obtain

$$-\frac{\partial}{\partial x}\left[a(x)\frac{\partial u_N}{\partial x}\right] = f(x)\,, \tag{2.46}$$

since we are now working with an approximate solution the equality is not entirely satisfied so we are working with a residual $R(x, c_j)$ that depends on coefficients c_j

$$R(x, c_j) = -\frac{\partial}{\partial x}\left[a(x)\frac{\partial u_N}{\partial x}\right] - f(x)\,. \tag{2.47}$$

A way to determine the coefficients c_j is to require them to vanish in a weighted residual sense

$$\int_0^L w_i(x)R(x, c_j)\,dx = 0 \tag{2.48}$$

where $w_i(x)$ are a set of linearly independent functions called weight functions. There exist several special cases of the weighted residual method that are distinguishable by the option of weight functions [55], some are listed as follows

- Petrov-Galerkin FEM: $w_i(x) \equiv \psi_i(x)$, where $\psi_i(x)$ is an element basis set of functions different from the set $\phi_i(x)$.
- Galerkin FEM: $w_i(x) \equiv \phi_i(x)$
- Least-Squares FEM: $w_i(x) \equiv \frac{\partial}{\partial x}\left[a(x)\frac{\partial \phi_i}{\partial x}\right]$
- Collocation FEM: $w_i(x) \equiv \delta(x - x_i)$

2.3.2 Reynolds equation weak form

Consider the Reynolds equation (2.26) we derived previously, together with appropriate boundary conditions it is known as the strong form of the problem, considering Dirichlet boundary conditions:

$$\begin{cases} \boldsymbol{\nabla}\cdot\left(h^3\boldsymbol{\nabla}p\right) + 6\mu U\dfrac{\partial(h\theta)}{\partial x} = 6\mu U\dfrac{\partial h}{\partial x} & (2.49)\\ p(\Gamma) = p_0 & (2.50)\\ \theta(\Gamma) = \theta_0 & (2.51) \end{cases}$$

where Γ signifies the boundary of the domain Ω. Our approach will be to utilize the weighted residual Galerkin finite element method (GFEM) [56] to derive the weak form of the Reynolds equation, the weighted residual can be written as

$$\int_\Omega w\left[\boldsymbol{\nabla}\cdot\left(h^3\boldsymbol{\nabla}p\right) + 6\mu U\frac{\partial(h\theta)}{\partial x} - 6\mu U\frac{\partial h}{\partial x}\right]d\Omega = 0\,. \tag{2.52}$$

Considering the first term of the integral we can write it as

$$\int_\Omega w\boldsymbol{\nabla}\cdot\left(h^3\boldsymbol{\nabla}p\right)\,d\Omega=\int_\Omega w(\boldsymbol{\nabla}h^3)\cdot(\boldsymbol{\nabla}p)\,d\Omega+\int_\Omega wh^3\nabla^2p\,d\Omega \tag{2.53}$$

where the divergence theorem allows us to rewrite the second order derivative term in the previous equation as

$$\begin{aligned}\int_\Omega wh^3\nabla^2p\,d\Omega&=-\int_\Omega(\boldsymbol{\nabla}\left(wh^3\right))\cdot(\boldsymbol{\nabla}p)\,d\Omega+\int_\Gamma h^3(\boldsymbol{\nabla}p)\cdot\hat{\mathbf{n}}\,d\Gamma\\&=-\int_\Omega h^3(\boldsymbol{\nabla}w)\cdot(\boldsymbol{\nabla}p)\,d\Omega-\int_\Omega 3h^2w(\boldsymbol{\nabla}h)\cdot(\boldsymbol{\nabla}p)\,d\Omega\,,\end{aligned} \tag{2.54}$$

the boundary Γ term can be dropped in the case of Dirichlet boundary conditions, however, it is necessary to set Neumann or mixed boundary conditions. Substituting equation (2.54) into equation (2.53) the weak form of the term reads

$$\int_\Omega w\boldsymbol{\nabla}\cdot\left(h^3\boldsymbol{\nabla}p\right)\,d\Omega=-\int_\Omega h^3(\boldsymbol{\nabla}w)\cdot(\boldsymbol{\nabla}p)\,d\Omega\,. \tag{2.55}$$

Only first order derivatives of the relevant variables are present in the equation thus we have derived the weak form of the Reynolds equation, which reads

$$\int_\Omega h^3(\boldsymbol{\nabla}w)\cdot(\boldsymbol{\nabla}p)\,d\Omega-6\mu U\int_\Omega w\frac{\partial(h\theta)}{\partial x}\,d\Omega+6\mu U\int_\Omega w\frac{\partial h}{\partial x}\,d\Omega=0\,, \tag{2.56}$$

the approximate solutions for pressure p and cavitation fraction θ can be written as

$$\begin{cases}p_N=\displaystyle\sum_{j=1}^{N}p_j\phi_j(x,y) & (2.57)\\ \theta_N=\displaystyle\sum_{j=1}^{N}\theta_j\phi_j(x,y) & (2.58)\end{cases}$$

where N is the number of finite elements and ϕ_j are the finite element basis functions.

Considering the GFEM where the weight functions are chosen to correspond to the element basis set and substituting the approximate solutions of pressure and cavitation fraction into the weak form of the Reynolds equation we obtain a set of N equations of the form

$$\overline{\boldsymbol{K}}^e_p\boldsymbol{p}^e+\overline{\boldsymbol{K}}^e_\theta\boldsymbol{\theta}^e=\boldsymbol{c}^e\,, \tag{2.59}$$

subject to the conditions $\boldsymbol{p}\cdot\boldsymbol{\theta}=\mathbf{0}$, $\boldsymbol{p}\geq\mathbf{0}$ and $\boldsymbol{\theta}\geq\mathbf{0}$, where

$$K^{e,p}_{ij}=\int_{\Omega_e}h^3\left(\boldsymbol{\nabla}\phi_i\cdot\boldsymbol{\nabla}\phi_j\right)\,d\Omega_e \tag{2.60}$$

$$K^{e,\theta}_{ij}=-6\mu U\int_{\Omega_e}\phi_i\frac{\partial(h\phi_j)}{\partial x}d\Omega_e \tag{2.61}$$

$$c^e_i=-6\mu U\int_{\Omega_e}\phi_j\frac{\partial h}{\partial x}\,d\Omega_e \tag{2.62}$$

where Ω_e refers to the single j-th finite element domain.

The standard GFEM method described so far introduces a truncation error for hyperbolic equations [57] in the form of a diffusive operator. In regions where cavitation is present the equation becomes hyperbolic which results in oscillatory solutions. In order to mitigate these effects an alternative method is utilized. The streamline upwind Petrov-Galerkin (SUPG) finite element method introduces a modification to the Galerkin finite element which is equivalent to introducing artificial diffusion to the equation, this modification results in a stabilized solution for cavitation.

The stabilization term is added to the weak form of the Reynolds equation as an extra term of the form

$$\int_{\Omega} \mathcal{P}(w)\, \tau\, R(\theta)\, d\Omega\,, \tag{2.63}$$

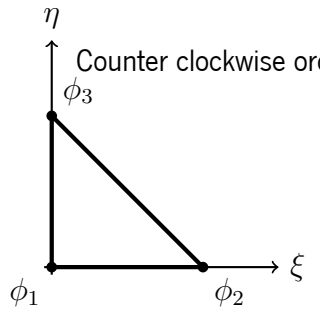
for SUPG stabilization we have

$$\begin{cases} \mathcal{P}(w) = U\dfrac{\partial w}{\partial x} & (2.64) \\ \tau = \dfrac{\beta \Delta x}{2U} & (2.65) \\ R(\theta) = -6\mu U \dfrac{\partial (h\theta)}{\partial x} & (2.66) \end{cases}$$

where Δx defines the largest x component of a finite element distance in the finite element mesh and β is a parameter that controls the amount of artificial diffusion. We obtain an altered and stabilized coefficient matrix written as

$$K_{ij}^{\theta} = -6\mu U \int_{\Omega_e} \left(\phi_i + \frac{1}{2}\beta \Delta x \frac{\partial \phi_i}{\partial x} \right) \frac{\partial (h\phi_j)}{\partial x} d\Omega_e\,. \tag{2.67}$$

For a triangular first order element (3 local nodes) with local element coordinates (ξ, η) the basis functions ϕ_i are defined as follows



$$\phi_1(\xi, \eta) = \xi$$
$$\phi_2(\xi, \eta) = \eta$$
$$\phi_3(\xi, \eta) = 1 - \xi - \eta$$

Figure 6: Triangular first order element in coordinate space (ξ, η) and respective basis functions. The element in regular cartesian coordinates is translated through the use of a Jacobian to its local coordinates and vice-versa.

In 2D we can write the cartesian coordinates (x, y) in terms of the element local coordinates (ξ, η)

as a linear combination of the previously defined basis function as follows

$$\begin{cases} x(\xi,\eta) = \sum_{j=1}^{N_h} x_j^e \phi_j(\xi,\eta) & (2.68) \\ y(\xi,\eta) = \sum_{j=1}^{N_h} y_j^e \phi_j(\xi,\eta)\,, & (2.69) \end{cases}$$

where N_h is the node number of an element and (x_j^e, y_h^e) are the known nodal cartesian coordinates. We can express (ξ,η) derivatives of the basis functions in terms of the (x,y) derivatives as

$$\begin{bmatrix} \frac{\partial \phi_i}{\partial \xi} \\ \frac{\partial \phi_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \end{bmatrix} \tag{2.70}$$

we can identify the 2×2 matrix as the Jacobian coordinate transformation matrix $\overline{\boldsymbol{J}}_e$, the entries of the Jacobian matrix can be calculated from equation (2.68). In order to convert the derivatives in our coefficient integrals to be in terms of (ξ,η) we require the inverse Jacobian $\overline{\boldsymbol{J}}_e^{-1}$ which is trivially calculated as a 2×2 matrix.

The full form of the K_{ij}^p coefficient matrix in terms of (ξ,η) for example can now be written as

$$K_{ij}^{e,p} = -6\mu U \int_{-1}^{1}\int_{-1}^{1} h^3(\xi,\eta) \left[\left(J_{e,11}^{-1} \frac{\partial \phi_i}{\partial \xi} + J_{e,12}^{-1} \frac{\partial \phi_i}{\partial \eta} \right) \left(J_{e,11}^{-1} \frac{\partial \phi_j}{\partial \xi} + J_{e,12}^{-1} \frac{\partial \phi_j}{\partial \eta} \right) + \left(J_{e,21}^{-1} \frac{\partial \phi_i}{\partial \xi} + J_{e,22}^{-1} \frac{\partial \phi_i}{\partial \eta} \right) \left(J_{e,21}^{-1} \frac{\partial \phi_j}{\partial \xi} + J_{e,22}^{-1} \frac{\partial \phi_j}{\partial \eta} \right) \right] |J_e| \, d\xi \, d\eta \tag{2.71}$$

where $h(\xi,\eta)$ is the shortened notation of $h(x(\xi,\eta), y(\xi,\eta))$. This transformation into local element coordinates enables us to utilize Gauss quadrature integration [57] to compute the coefficients.

The assembly of the finite element coefficient matrices into the global system is done through a local-to-global mapping matrix that translates a local element numbering into a global node numbering.

2.3.3 Inexact Newton method

The inexact Newton iteration was shown in [2] to be an effective and efficient way to solve the LCP problem. In the following a brief report of the INE iteration steps is given. We can cleverly rewrite our set of coupled equations and conditions as

$$\boldsymbol{\Psi}(\boldsymbol{p},\boldsymbol{\theta}) = \boldsymbol{0} \tag{2.72}$$

$$\boldsymbol{p} \geq \boldsymbol{0};\ \boldsymbol{\theta} \geq \boldsymbol{0}\,, \tag{2.73}$$

in which

$$\boldsymbol{\Psi}(\boldsymbol{p},\boldsymbol{\theta}) = \begin{pmatrix} \boldsymbol{\Psi}_0(\boldsymbol{p},\boldsymbol{\theta}) \\ \overline{\overline{\boldsymbol{P}}}\,\overline{\overline{\boldsymbol{\Theta}}}\hat{\boldsymbol{e}} \end{pmatrix} = \begin{pmatrix} \overline{\overline{\boldsymbol{K}}}_{\boldsymbol{p}}\boldsymbol{p} + \overline{\overline{\boldsymbol{K}}}_{\boldsymbol{\theta}}\boldsymbol{\theta} - \boldsymbol{c} \\ \overline{\overline{\boldsymbol{P}}}\,\overline{\overline{\boldsymbol{\Theta}}}\hat{\boldsymbol{e}} \end{pmatrix} \tag{2.74}$$

and where diagonal matrices $\overline{\overline{\boldsymbol{P}}}$ and $\overline{\overline{\boldsymbol{\Theta}}}$ have the values of $\boldsymbol{p}$ and $\boldsymbol{\theta}$ as their entries, and $\hat{\boldsymbol{e}}$ represents the unit vector.

The Newton iteration is defined as

$$\begin{pmatrix} \boldsymbol{p}^{k+1} \\ \boldsymbol{\theta}^{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{p}^{k} \\ \boldsymbol{\theta}^{k} \end{pmatrix} + \alpha^k \begin{pmatrix} \Delta\boldsymbol{p}^{k} \\ \Delta\boldsymbol{\theta}^{k} \end{pmatrix} ;\ k = 0, 1, 2, ... \tag{2.75}$$

where $(\Delta\boldsymbol{p}^k, \Delta\boldsymbol{\theta}^k)$ are the updates for each iteration dampened by a factor α_k and are calculated by solving the linear system

$$\boldsymbol{J}^k(\boldsymbol{p},\boldsymbol{\theta}) \begin{pmatrix} \Delta\boldsymbol{p}^{k} \\ \Delta\boldsymbol{\theta}^{k} \end{pmatrix} = -\boldsymbol{\Psi}^k(\boldsymbol{p},\boldsymbol{\theta})\,, \tag{2.76}$$

As described in [2], a common pitfall with this method is a stalling of the iteration leading to convergence failure. To avoid this, a perturbation term $\eta_k\tilde{\boldsymbol{e}}$ with $\tilde{\boldsymbol{e}} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{1} \end{pmatrix}$ is included:

$$\boldsymbol{J}^k(\boldsymbol{p},\boldsymbol{\theta}) \begin{pmatrix} \Delta\boldsymbol{p}^{k} \\ \Delta\boldsymbol{\theta}^{k} \end{pmatrix} = -\boldsymbol{\Psi}^k(\boldsymbol{p},\boldsymbol{\theta}) + \eta_k\tilde{\boldsymbol{e}}\,. \tag{2.77}$$

The perturbation coefficient η_k is defined as

$$\eta_k = \sigma_k\mu_k\,, \tag{2.78}$$

where $\sigma_k \in [0, 1]$ is the forcing parameter and μ_k is a parameter in the range

$$\frac{\boldsymbol{p}^k \cdot \boldsymbol{\theta}^k}{N} \leq \mu_k \leq \frac{\left\|\boldsymbol{\Psi}^k(\boldsymbol{p},\boldsymbol{\theta})\right\|}{\sqrt{N}}\,. \tag{2.79}$$

The choice of relaxation parameter α^k is required to enforce the convergence of the method and this choice is composed of 3 conditions or steps: feasibility, centrality and backtracking.

The feasibility condition provides the initial value of α_k and reads

$$\alpha^k = \min\left(\min_{\Delta p_i^k<0} \frac{-p_i^k}{\Delta p_i^k}, \min_{\Delta\theta_i^k<0} \frac{-\theta_i^k}{\Delta\theta_i^k}, 1\right)\,, \tag{2.80}$$

this guarantees that conditions $\boldsymbol{p} \geq \boldsymbol{0}$ and $\boldsymbol{\theta} \geq 0$ are both satisfied after each iteration.

The centrality conditions define the coefficient η_k in the forcing term and ensure $\boldsymbol{p}^k$ and $\boldsymbol{\theta}^k$ are bounded away from $\boldsymbol{p} = 0$ and $\boldsymbol{\theta} = 0$ and are defined as

$$\begin{aligned} \phi_1^k(\alpha_k) &= \min\left(\boldsymbol{p}^k(\alpha_k) \circ \boldsymbol{\theta}^k(\alpha_k)\right) - \frac{\gamma_k\tau_1}{N}\boldsymbol{p}^k(\alpha_k)\cdot\boldsymbol{\theta}^k(\alpha_k) \geq 0 \\ \phi_2^k(\alpha_k) &= \boldsymbol{p}^k(\alpha_k)\cdot\boldsymbol{\theta}^k(\alpha_k) - \gamma_k\tau_2\left\|\boldsymbol{\Psi}_0^k(\boldsymbol{p}(\alpha_k),\boldsymbol{\theta}(\alpha_k))\right\| \geq 0\,, \end{aligned} \tag{2.81}$$

where

$$\tau_1^{max} = N\frac{\min\left(\boldsymbol{p}^0 \circ \boldsymbol{\theta}^0\right)}{\boldsymbol{p}^0 \cdot \boldsymbol{\theta}^0}; \quad \tau_2^{max} = \frac{\boldsymbol{p}^0 \cdot \boldsymbol{\theta}^0}{\|\boldsymbol{\Psi}_0^0(\boldsymbol{p}, \boldsymbol{\theta})\|} \tag{2.82}$$

and $\gamma_k \in [0.5, 1.0]$. The centrality conditions also impose a lower bound for the forcing parameter:

$$\sigma_k > \max\left(\delta_k \frac{\sqrt{N} + \tau_1\gamma_k}{1 - \tau_1\gamma_k}, \delta_k \frac{\sqrt{N} + \tau_2\gamma_k}{\sqrt{N}}\right), \tag{2.83}$$

a recursive reduction of α_k can be used to find a value that satisfies both conditions (2.81).

The backtracking condition is an additional safeguard that reduces the value of α_k by a factor λ^t with $\lambda \in [0, 1]$ and t a non-negative integer that increases per iteration until the following condition is met

$$\left\|\boldsymbol{\Psi}(\boldsymbol{p}^k + \alpha_k\Delta\boldsymbol{p}^k, \boldsymbol{\theta}^k + \alpha_k\Delta\boldsymbol{\theta}^k)\right\| \leq (1 - \beta\alpha_k(1 - \sigma_k - \delta_k))\left\|\boldsymbol{\Psi}(\boldsymbol{p}^k, \boldsymbol{\theta}^k)\right\|, \tag{2.84}$$

where $\beta \in [0, 1]$ and $\sigma_k + \delta_k = 1$.

The parameters described within this section are set as follows for every case in the validation, the results and the data generation that was obtained

$$\begin{aligned}\sigma_k &= \min\left(0.01\left\|\boldsymbol{\Psi}^k(\boldsymbol{p}, \boldsymbol{\theta})\right\|, 0.9\right) \\ \delta_k &= \min\left(\frac{\sigma_k}{N\max\left(\frac{\sqrt{N}+\tau_1+\gamma_k}{1-\tau_1\gamma_k}, \frac{\sqrt{N}+\tau_2\gamma_k}{\sqrt{N}}\right)}\right).\end{aligned} \tag{2.85}$$

τ_1	τ_2	μ_k	β	λ	γ_k
$\min\left(0.5\,\tau_2^{max} \times 10^{-7}, 0.99\right)$	$\tau_2^{max} \times 10^{-7}$	μ_k^{max}	10^{-4}	0.5	0.5

Table 1: Parameters utilized in the numerical experiments described in [2] for validation of the INE method implementation.

The stopping criteria for the INE iteration is defined as

$$\begin{aligned}c_1 &= \left\|\boldsymbol{\Psi}^{k+1}(\boldsymbol{p}, \boldsymbol{\theta})\right\| \leq \epsilon_1 \\ c_2 &= \left\|\alpha_k \begin{pmatrix}\Delta\boldsymbol{p}^k \\ \Delta\boldsymbol{\theta}^k\end{pmatrix}\right\| \leq \epsilon_2,\end{aligned} \tag{2.86}$$

where ϵ_1, ϵ_2 define the convergence threshold of the solver.

2.4 Machine learning

2.4.1 Basic concepts

Consider we have a problem P that requires an answer (or output O) for a set of parameters (or input) I. Consider a set of algorithms A that correctly solves our problem P with varying degrees of efficiency and accuracy, ideally we are most interested in the algorithm $A_k \in A$ that achieves the most accurate answer in the most efficient way, this is of course considering however that we know the set of algorithms A.

For some problems P we do not know an algorithm that is capable of solving the problem, however for many others we know some algorithms but they can be impossibly inefficient to be of practical use. One example of this is spam filters [58] for e-mail clients, the input I is the e-mail itself, the problem P is simply stated as "Is the e-mail I spam?" and the output O is a binary response, yes or no. As simple as a problem like this might seem, it depends entirely on what the individual considers to be spam, therefore there is no clear and objective way to define a set of instructions capable of solving the problem. Despite this, we can provide millions of examples of spam and non-spam e-mails in such a way that we can systematically learn from these by letting a computer for example, artificially extract the algorithm just from the raw data. Obviously the "algorithm" we are extracting from data will not be a perfect solution to the problem, yet it might be a good approximation supported by a degree of accuracy that we can calculate. This process of automated learning is commonly known as machine learning (ML).

2.4.2 Types of machine learning applications

Machine learning applications are commonly split into two main categories: classification problems, where to a given input I is assigned a class, for example numerical handwritting recognition (figure 7), and regression problems of which a good example is the objective of this thesis, that is fitting a set of surface-defining parameters to a Stribeck curve, a highly non-linear problem.

In the case of figure 7 the correct digit class is 6, and the prediction is commonly associated with a probability of being the correct class, meaning that we have an implicit accuracy associated with the prediction.

In the context of this thesis, we considered a regression application, a simple example of which is that of performing fits to data, meaning that the application takes a collection of N points (x_i, y_i) and

provides a fit function $f(x_i)$ that closely describes the real output y_i

$$y_i^{pred} \equiv f(x_i) \tag{2.87}$$

Figure 7: Handwritten 6 for which the classification ML application should give a correct prediction.

The accuracy of the fit function in describing the input can be judged by direct comparison between the predicted values and the input values, for example with the root mean squared error (RMSE) defined as follows

$$\sigma_i = \sqrt{\left(y_i^{pred} - y_i\right)^2}, \tag{2.88}$$

we can take the overall prediction error for new input I to be the maximum RMSE or the average RMSE both defined respectively as

$$\sigma_{max} = \max \boldsymbol{\sigma} \tag{2.89}$$

$$\sigma_{avg} = \frac{\sqrt{\sum_{i=1}^{N}\left(y_i^{pred} - y_i\right)^2}}{N}. \tag{2.90}$$

2.4.3 Perceptron

The simplest automated learning unit is the perceptron, schematically represented in figure 8. The inputs x_i are associated with a connection weight $w_i \in \Re$ and the output is computed as follows

$$y = f\left(w_0 + \sum_{i=1}^{n} w_i x_i\right), \tag{2.91}$$

this process can also be written in vector form:

$$y = f\left(\boldsymbol{w}^T \boldsymbol{x}\right). \tag{2.92}$$

The function f is commonly referred to as the activation function, which can be a step-like function such as

$$\Theta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0\,. \end{cases} \tag{2.93}$$

More complex functions might be considered as activation functions, such as non-linear activation functions which allow small networks of perceptrons to compute nontrivial behavior. Indeed, research on activation functions is a very active area in the research of the machine learning field as they have been shown to have a significant effect on the efficiency of certain learning problems [59].

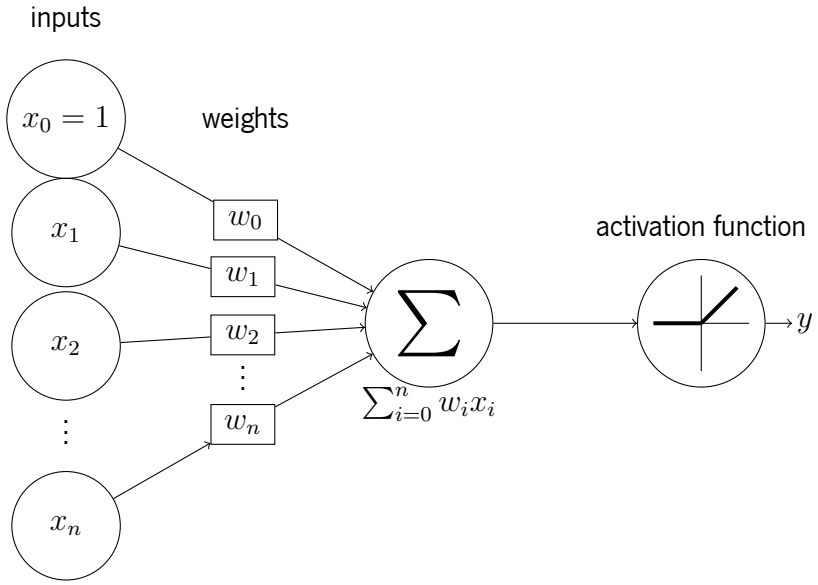


Figure 8: Simple perceptron, x_i are the input values, $x_0 = 1$ is the bias, w_i are the weight connections and y is the output value.

Consider a set of $K \geq 2$ perceptrons, fully connected with inputs x_j and connections w_{ij} from input j to perceptron i, schematically shown in figure 9. The output of perceptron i can be computed as

$$y_i = f\left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j\right)\,, \tag{2.94}$$

or

$$\boldsymbol{y} = f\left(\boldsymbol{W}\boldsymbol{x}\right)\,, \tag{2.95}$$

where $\boldsymbol{W}$ is a matrix whose rows are the weight vector $\boldsymbol{w}_i$ of perceptron i, in the case of figure 9 we have $\boldsymbol{W}$ as a $2 \times n + 1$ matrix.

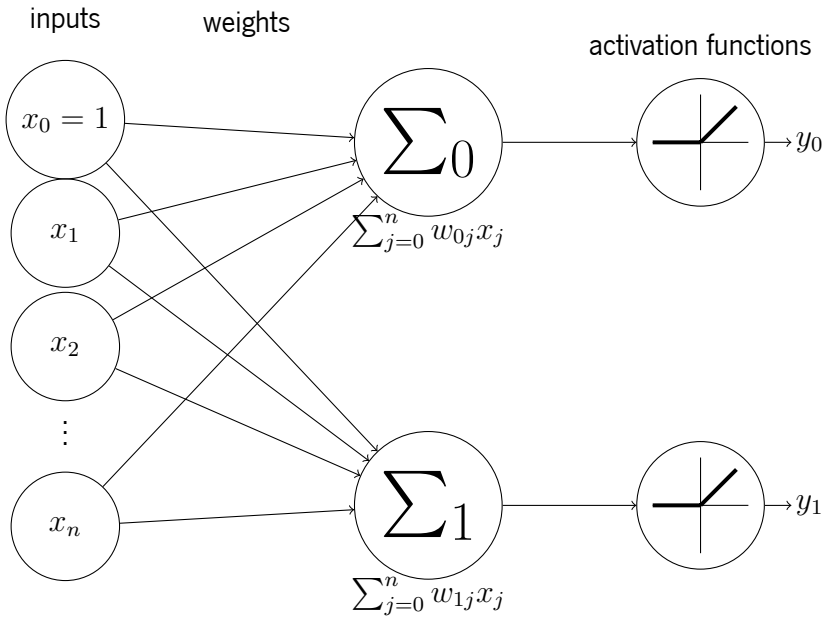


Figure 9: Set of two perceptrons, x_j are the input values, $x_0 = 1$ is the bias, w_{ij} are the weight connections from input j to perceptron i and y_i is the output value.

Considering a single perceptron with randomized initial weights $\boldsymbol{w}$ for a input $\boldsymbol{x}$, we can compute the perceptrons output with equation (2.92), defining a loss function $E(y, y_p)$ that compares the perceptron output y_p with the desired output y. Aiming for the smallest possible difference between the predicted output y_p and the true output y we expect to find the minimum of the loss function $E(y, y_p)$, which is a function of the weights. However, the global minimum in real applications is incredibly hard to find, in most cases we reach local minima.

The training of the perceptron is typically done by consistently updating the weights in order to find a minimum of the loss function. If we update weight w_j by considering the negative derivative of the loss function we should step closer to the minimum of the loss function. In theory, this update should be done in small portions in order to not overshoot it, thus we write the weight update

$$\Delta w_j = -\eta \frac{\partial L(y, y_p)}{\partial w_j} \tag{2.96}$$

where η is the learning rate, this method of weight updating is referred to as stochastic gradient descent (SGD).

Typically a small learning rate, while ensuring that it is less likely that we miss local minima due to shorter updates along the slope of the loss function, takes a much longer time to converge. It can also converge toward plateau-like regions that stall learning. Large learning rates can lead to overshooting the local minima as well as completely diverging in loss which leads to a failure in convergence.

We can further write equation (2.96) as follows, by using the chain rule

$$\Delta w_j = -\eta \frac{\partial L(y, y_p)}{\partial y_p} \frac{\partial y_p}{\partial w_j} \,, \tag{2.97}$$

from equation (2.91) we have

$$\frac{\partial y_p}{\partial w_j} = f^{'} \left(\sum_{i=0}^{n} w_i x_i \right) x_j \,, \tag{2.98}$$

where $x_0 = 1$ for the bias weight. The bias of a layer is extremely important particularly for regression problems, the reason why is clearer by considering a single neuron and input and the resulting weighted sum

$$y = a\, x + b \,, \tag{2.99}$$

with the bias term we fit a function that has control over slope as well as intercept, then the bias can be seen as one more input representing the part of the output that is actually independent from the input (a sort of background).

Substituting into equation (2.97) we have a generalized way to determine the update for weight j

$$\Delta w_j = -\eta \frac{\partial L(y, y_p)}{\partial y_p} f^{'} \left(\sum_{i=0}^{n} w_i x_i \right) x_j \,. \tag{2.100}$$

Lets consider a special case where we define the loss function and activation function to be

$$L(y, y_p) = \frac{1}{2} \left(y - y_p \right)^2 \tag{2.101}$$

$$f(x) = x \tag{2.102}$$

$$f^{'}(x) = 1 \tag{2.103}$$

the update for weight j can be computed as

$$\Delta w_j = \eta \left(y - y_p \right) x_j \,. \tag{2.104}$$

In the following section we will describe how to generalize SGD to a network of perceptrons which can learn and adapt to non-linear behavior.

2.4.4 Deep neural network

A deep neural network (DNN) is essentially a mathematical function $\mathcal{F}$ that takes input $\boldsymbol{x}$ and predicts output $\boldsymbol{y}$. They are trained based on statistical techniques [58] such as supervised, unsupervised and adversarial learning among others.

Perceptrons are limited in what they can predict, because there is only a single layer of weights that can only approximate linear behavior and cannot solve any sort of nonlinear problem. This limitation is not present if we consider a network of interconnected perceptrons, which are in this case called "neurons" due to their resemblance with biological neuron functions. Let us consider an input $\boldsymbol{x}$ with k entries, with weighted connections to N_1 perceptron-like weighted sum plus their associated activation functions. The output y_1 of these N_1 neurons further connects in a weighted sense to additional N_2 neurons, constituting what is regarded as a "feed-forward" network with hidden layers between input and output, as it is visualized in figure (10). Deep neural networks are networks that contain a number of inner layers and are perfectly capable of solving non-linear regression problems.

Consider a deep neural network with 2 hidden layers, with N_{h_1} and N_{h_2} neurons respectively, an input layer with N_i neurons and output layer with N_o neurons. This means that 3 sets of weighted connections exist, between the input layer and first hidden layer, between the first hidden layer and the second hidden layer and finally between the second hidden layer and the output layer.

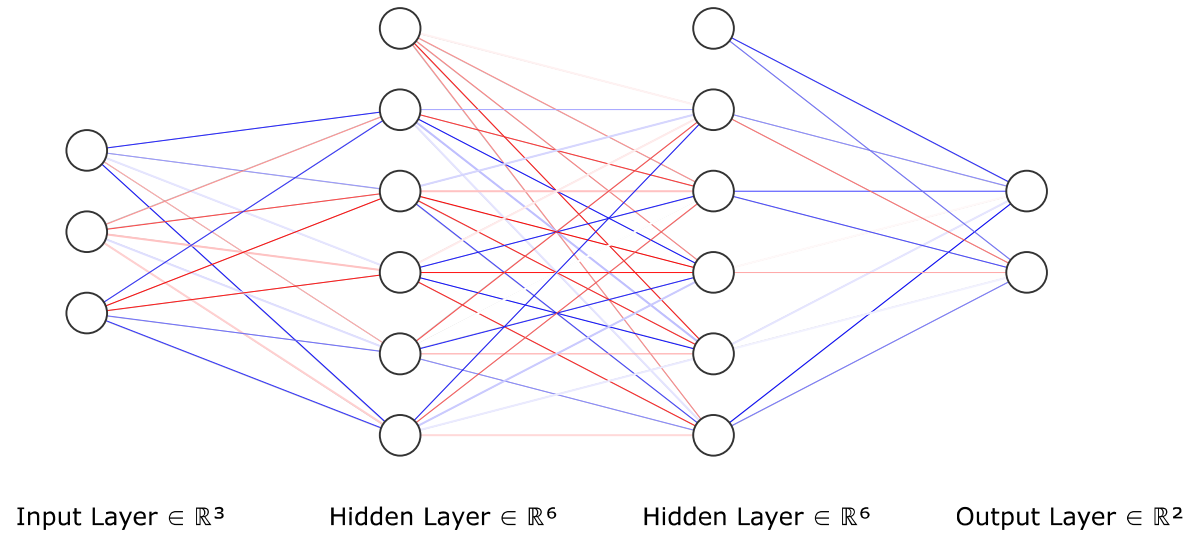


Figure 10: Schematic representation of a deep neural network [1], consisting of 2 hidden layers with 5 neurons + 1 bias neuron each, the colored connections represent the value of weights in the range $[-1, 1]$.

As the input layer does not have weighted connections to it, its the output is simply the input $\boldsymbol{x}$, the input neurons perform no calculations. As shown in equation (2.95) we can obtain the first hidden layer output by computing

$$\boldsymbol{y}_{h_1} = \sigma_{h_1} \left(\boldsymbol{W}_{ih_1} \boldsymbol{y}_i \right) , \tag{2.105}$$

where $\boldsymbol{y}_{h_1}$ is the output of the first hidden layer, σ_1 is the activation function of the first hidden layer, $\boldsymbol{W}_{ih_1}$ are the weights from input layer to the first hidden layer and $\boldsymbol{y}_i$ is the output of the input layer (which is the original input).

We can similarly write the same equation for the output of the second hidden layer and finally the output layer, as a whole we can write the final output of the network as

$$\boldsymbol{y}_o = \sigma_o\left(\boldsymbol{W}_{h_2o}\sigma_{h_2}\left(\boldsymbol{W}_{h_1h_2}\sigma_{h_1}\left(\boldsymbol{W}_{ih_1}\boldsymbol{y}_i\right)\right)\right) \tag{2.106}$$

this computation is known as the feed-forward algorithm [60] of a deep neural network. Naturally for a randomly generated set of weights, the output $\boldsymbol{y}_o$ will be random when compared to the expected output after learning.

2.4.5 Learning algorithm

As explained previously, in order to train the perceptron we sequentially attempt to match its output $\boldsymbol{y}_p$ given an input $\boldsymbol{x}$ to the expected output $\boldsymbol{y}$, this is done by slowly changing the weights of the perceptron by minimizing a given loss function. Consider we have a set of data with N entries, with these entries consisting of pairs $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, if the complete loss function $\mathcal{L}$ is defined as

$$\mathcal{L} \equiv L(\boldsymbol{y}_k^p, \boldsymbol{y}_k)\,, \tag{2.107}$$

which is calculated from a single pair $(\boldsymbol{y}_k^p, \boldsymbol{y}_k)$ we can expect the learning rate to be slow as the algorithm is attempting to adjust only to a single pair per step. An ideal choice is the following complete loss function

$$\mathcal{L} = \sum_{k=0}^{N} L(\boldsymbol{y}_k^p, \boldsymbol{y}_k)\,. \tag{2.108}$$

In which the complete loss function is a function of the entire data set. However, whereas this approach is the most complete, as the weight updates would adjust to the entire data set, it is too computationally demanding for realistic applications with hundreds of thousands of data samples and very large complex networks.

A common compromise is to split up the set of N entries into small batches N_b typically being 32 samples in most applications [61], such that each step computes a complete loss function as

$$\mathcal{L} = \sum_{k\in\text{batch}} L(\boldsymbol{y}_k^p, \boldsymbol{y}_k) \tag{2.109}$$

which results in significantly faster convergence of the learning while remaining computationally feasible. For the sake of simplicity when discussing the main learning algorithm of deep neural networks we will consider the loss function as defined in (2.107).

Consider an arbitrary network with layers $0, 1, 2, \ldots, L-1, L$ where L is the total number of layers with $N_0, N_1, N_2, \ldots, N_{L-1}, N_L$ neurons in layer L, defining $\boldsymbol{a}^L$ as the activation of layer L, $\boldsymbol{a}^{L-1}$ as

the activation of layer $L-1$, weights $\boldsymbol{W}^L$ as the weights that connect $L-1$ to L and σ^L as the activation function for layer L we can compute $\boldsymbol{a}^L$ as follows

$$\boldsymbol{a}^L = \sigma^L\left(\boldsymbol{z}^L\right) , \tag{2.110}$$

where

$$\boldsymbol{z}^L = \boldsymbol{W}^L\boldsymbol{a}^{L-1} . \tag{2.111}$$

Let us focus on layer L as the output layer, during training we provide a set of data $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ the input of which is fed through the network providing a prediction $\boldsymbol{a}^L$. The loss when compared to $\boldsymbol{y}_k$ can be computed by the loss function $\mathcal{L}(\boldsymbol{a}^L, \boldsymbol{y}_k)$, utilizing the chain rule we can compute the derivative of the loss function $\mathcal{L}$ with respect to weight W^L_{jk}

$$\frac{\partial \mathcal{L}}{\partial W^L_{jk}} = \frac{\partial \mathcal{L}}{\partial a^L_j}\frac{\partial a^L_j}{\partial z^L_j}\frac{\partial z^L_j}{\partial W^L_{jk}} \tag{2.112}$$

following the stochastic gradient descent process described in section 2.4.3 we can use as an update for weight W^L_{jk}

$$\Delta W^L_{jk} = -\eta\frac{\partial \mathcal{L}}{\partial W^L_{jk}} , \tag{2.113}$$

where η is the learning rate.

We can also compute the derivative of the loss function $\mathcal{L}$ with respect to the activation of the previous layer $L-1$

$$\frac{\partial \mathcal{L}}{\partial a^{L-1}_k} = \sum_{j=0}^{N_L}\frac{\partial z^L_j}{\partial a^{L-1}_k}\frac{\partial a^L_j}{\partial z^L_j}\frac{\partial \mathcal{L}}{\partial a^L_j} , \tag{2.114}$$

the summation over all neurons in layer L is necessary as the activation a^{L-1}_k has a connection to N_L neurons in layer L. This means that computing the derivative of the loss function $\mathcal{L}$ with respect to weight W^{L-1}_{jk} becomes trivial

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W^{L-1}_{jk}} &= \frac{\partial z^{L-1}_j}{\partial W^{L-1}_{jk}}\frac{\partial a^{L-1}_j}{\partial z^{L-1}_j}\frac{\partial \mathcal{L}}{\partial a^{L-1}_j} \\ &= \frac{\partial z^{L-1}_j}{\partial W^{L-1}_{jk}}\frac{\partial a^{L-1}_j}{\partial z^{L-1}_j}\left(\sum_{m=0}^{N_L}\frac{\partial z^L_m}{\partial a^{L-1}_j}\frac{\partial a^L_m}{\partial z^L_m}\frac{\partial \mathcal{L}}{\partial a^L_m}\right) .\end{aligned} \tag{2.115}$$

This method can be repeated for layers $L-2, L-3, \ldots, 0$ which allows us to compute update weights for every layer in our network, this process is known as back-propagation.

Consider an example network with 3 layers (for simplicity we are ignoring bias units), represented in figure 11, where the output layer is layer L, the hidden layer is layer $L-1$ and the loss function we will consider is defined as

$$\mathcal{L}(\boldsymbol{a}^L, y) = \frac{1}{2}\sum_{j=0}^{N_L}\left(y_j - a^L_j\right)^2 . \tag{2.116}$$

Importantly, let us mention the dimensions of our quantities: the weight matrix $\boldsymbol{W}^L$ has dimensions 2×4, $\boldsymbol{W}^{L-1}$ has dimensions 4×3, $\boldsymbol{a}^L$ is a column vector with 2 entries (corresponding to the output), $\boldsymbol{a}^{L-1}$ is a column vector with 4 entries.

To update all weights in our network we have to compute in total 20 weight updates, considering the activation function $\sigma^L(z) = z$ (linear activation) we can obtain an equation for the update of weight W^L_{jk} for our network

$$\begin{aligned} \Delta W^L_{jk} = -\eta \frac{\partial \mathcal{L}}{\partial W^L_{jk}} &= \eta a^{L-1}_k \left(y_j - a^L_j\right) \\ &= \eta \boldsymbol{a}^{L-1} \cdot \left(\boldsymbol{y} - \boldsymbol{a}^L\right) \end{aligned} \tag{2.117}$$

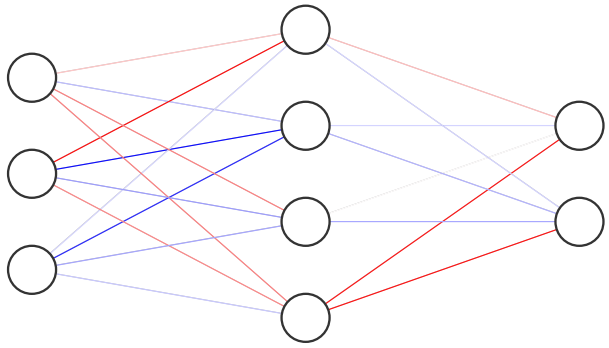


Figure 11: Schematic representation of a deep neural network [1], consisting of 1 hidden layer with 4 neurons + 1 bias neuron (hidden for visibility), the colored connections represent the value of weights in the range $[-1, 1]$.

Utilizing equation (2.115) and considering the activation function $\sigma^{L-1}(z) = \tanh z$ we can similarly calculate the weight updates for layer $L-1$

$$\begin{aligned} \Delta W^{L-1}_{jk} = -\eta \frac{\partial \mathcal{L}}{\partial W^{L-1}_{jk}} &= \eta x_k \left[1 - \tanh^2\left(\sum_{n=0}^{N_{L-1}} W^{L-1}_{jn} x_n\right)\right] \sum_{m=0}^{N_L} W^L_{mj} \left(y_m - a^L_m\right) \\ &= \eta\, \boldsymbol{x} \cdot \left[\left(\mathbf{1}^{N_0} - \tanh^2\left(\boldsymbol{W}^{L-1}\boldsymbol{x}\right)\right) \odot \boldsymbol{W}^L(\boldsymbol{y} - \boldsymbol{a}^L)\right] , \end{aligned} \tag{2.118}$$

where $\mathbf{1}^{N_0}$ is the identity vector of $N_0 = 3$ entries, $\odot$ is the matrix elementwise multiplication operator. This set of equations that define the update weights of layer L and layer $L-1$ as well as the learning rate allow for the systematic training of our network to our data set.

Chapter 3

FELINE: a Reynolds equation solver

In chapters 1 and 2 we discussed the importance of the Reynolds equation solver in the context of texture optimization as obtaining the film pressure profile is fundamental to compute friction in a system. The work developed in this chapter details our implementation of a fast and robust Reynolds equation solver termed FELINE, based on a FEM implementation of the inexact newton method to deal with the linear complementarity problems (LCP) that naturally arise from the introduction of cavitation. A noticeable improvement in speed is observed when compared to similar work in literature while enforcing physically correct treatment of the boundary of cavitation regions.

3.1 Results

In this section, results pertaining to the utilization of FELINE are shown, ranging from a set of problems explored in [2] that are used as validation for the solver to others we identified to be of industrial interest that used dimpled textures. A benchmark of the solver is also provided to show the performance increase of our solution when compared to existing solvers.

The INE solver was implemented in C++17 utilizing the linear algebra library Armadillo [62][63]. The inner solver of choice is SuperLU [64], a sparse direct solver preceded by an equilibration of the sparse Jacobian matrix to help reduce computation times.

3.1.1 Validation

To allow for a direct comparison with the results reported by Mezzadri and Galligani [2], we used their same algorithm parameters, which are reported in table 1, page 21. However, we note that those parameters can, and should, be optimized to achieve faster convergence times in real applications.

To validate our results, we considered a 2D square domain of length $L = 100\mu m$, a lubricant viscosity $\mu = 0.015\ Pa \cdot s$ and a relative surface sliding velocity $U = 5 \times 10^{-3}\ m \cdot s^{-1}$ and we ran three different test cases matching problems 1-3 of [2]. We can define a Convergent-Divergent (C-D) profile and a Divergent-Convergent (D-C) profile according to the following function

$$h_{\pm}(x,y) = \pm\frac{h_{max} - h_{min}}{2}\sin\left(\frac{2\pi x}{L} + \frac{\pi}{2}\right) + 2\left(h_{max} - h_{min}\right)\ , \tag{3.1}$$

with $h_{max} = 0.025\ \mu m$ and $h_{min} = 0.015\ \mu m$, that, when combined with the different boundary conditions, allows us to define three test cases according to table 2.

Problem	Profile	$p_{d\Omega}$	$\theta_{d\Omega}$
1	$h_+(x,y)$	0	0
2	$h_+(x,y)$	1	0
3	$h_-(x,y)$	1	0

Table 2: Test case definition where C-D $\equiv h_+(x,y)$, D-C $\equiv h_-(x,y)$ and $p_{d\Omega}$ and $\theta_{d\Omega}$ are the values of p and θ at the domain boundary.

The following solutions were obtained in a 200×200 element mesh, with 2D and sliced pressure and cavitation profiles for problem 1 shown in figures 12 and 13. Our results are in excellent agreement with profiles of references [2] (in figure 18 page, 36) and [65]. The 2D and sliced results for problem 2 and 3 are shown in figures (14, 15) and (16, 17) respectively, closely matching the 1D profiles of [2].

The difference in pressure magnitudes are due to the fact that, in [2] these problems are solved in 1D while we are solving them in 2D. We further notice that the cavitation is correctly reproduced where expected, that is in correspondence of divergent profile region.

It is also important to highlight that in every solved example the complementarity conditions were obeyed at every iteration, thus providing a robust physical solution. The feasibility condition was sufficient to fulfill the complementarity conditions in these 3 cases, although centrality and backtracking might be necessary to ensure convergence in more complex cases. The cavitation model and inexact Newton iteration method implemented in this work ensure physically realistic solutions.

After direct comparison with the results reported in [2] we concluded that the solver performs correctly when predicting pressure and correctly treating the cavitation boundaries by ensuring the non-negativity conditions. In the following section we will detail some more realistic and possibly industrially relevant cases that our solver also handles correctly.

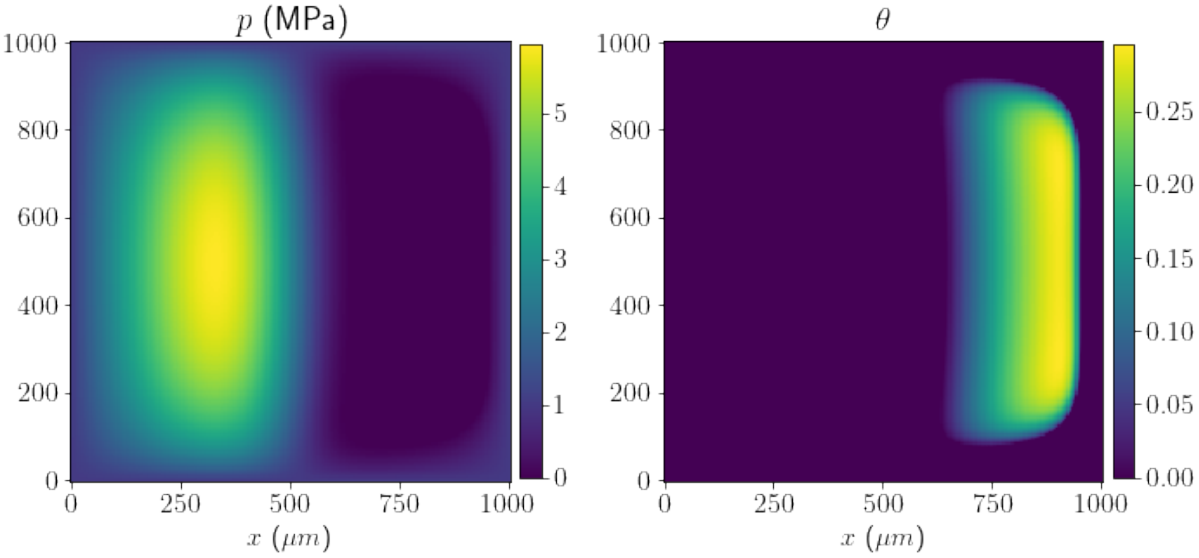


Figure 12: Surface plots of $h/h_{max}(x, y)$, $p(x, y)$ and θ of problem 1 solved in a 200×200 element mesh.

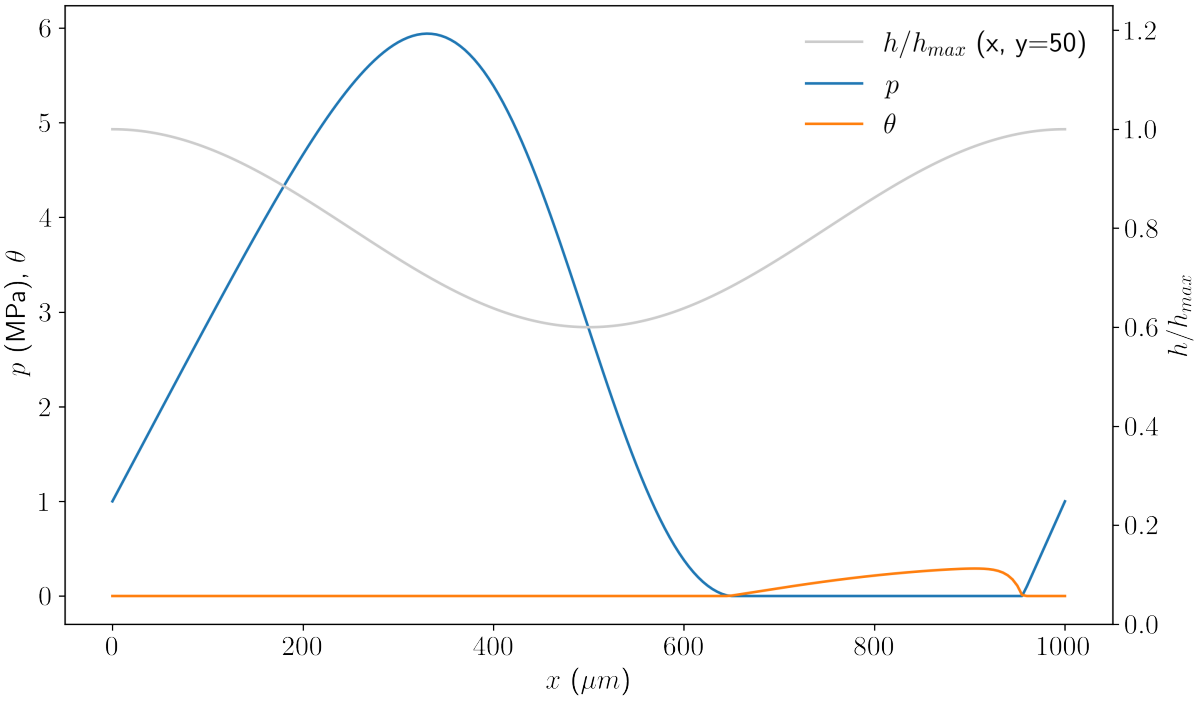


Figure 13: Plots of a slice at $y = 50\ \mu m$ of $h/h_{max}(x, y)$, $p(x, y)$ and θ of problem 1 solved in a 200×200 element mesh.

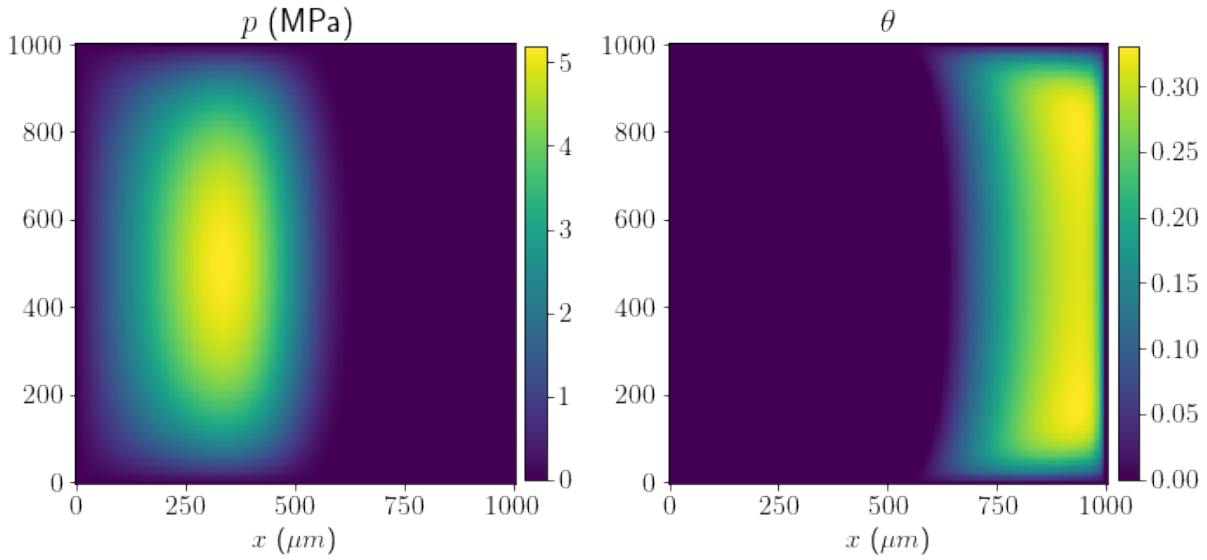


Figure 14: Surface plots of $h/h_{max}(x, y)$, $p(x, y)$ and $\theta(x, y)$ of problem 2 solved in a 200x200 element mesh.

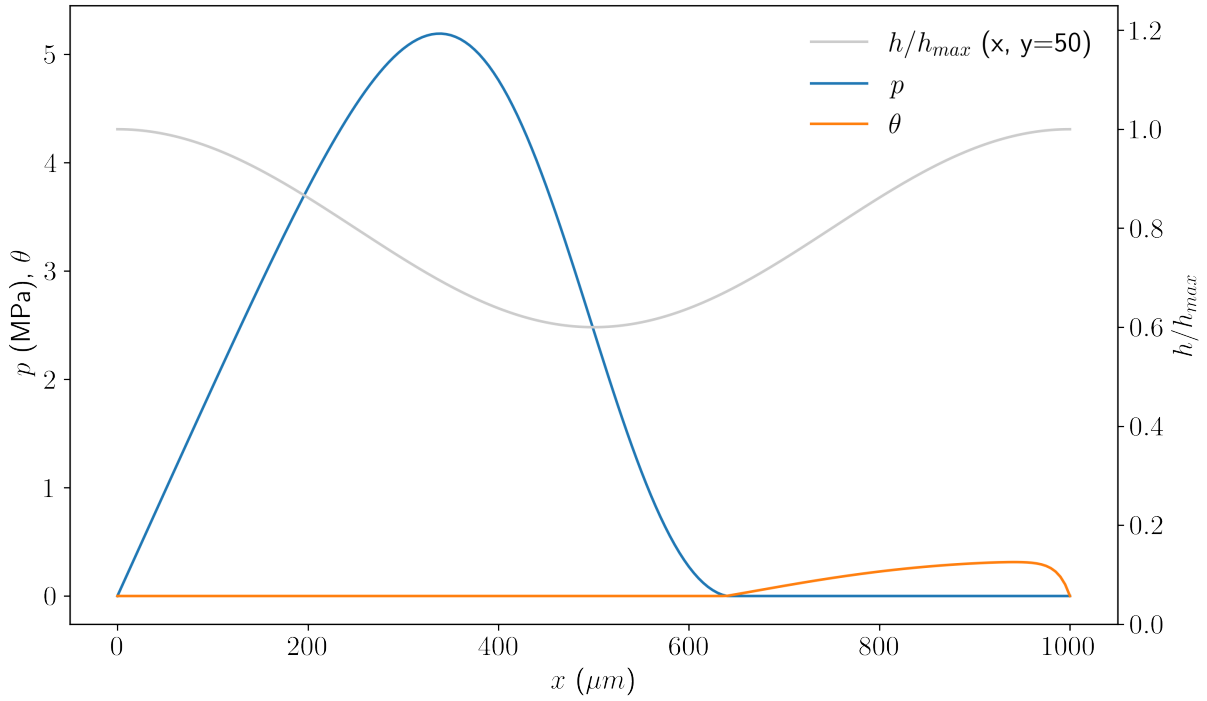


Figure 15: Plots of a slice at $y = 50\ \mu m$ of $h/h_{max}(x, y)$, $p(x, y)$ and $\theta(x, y)$ of problem 2 solved in a 200×200 element mesh.

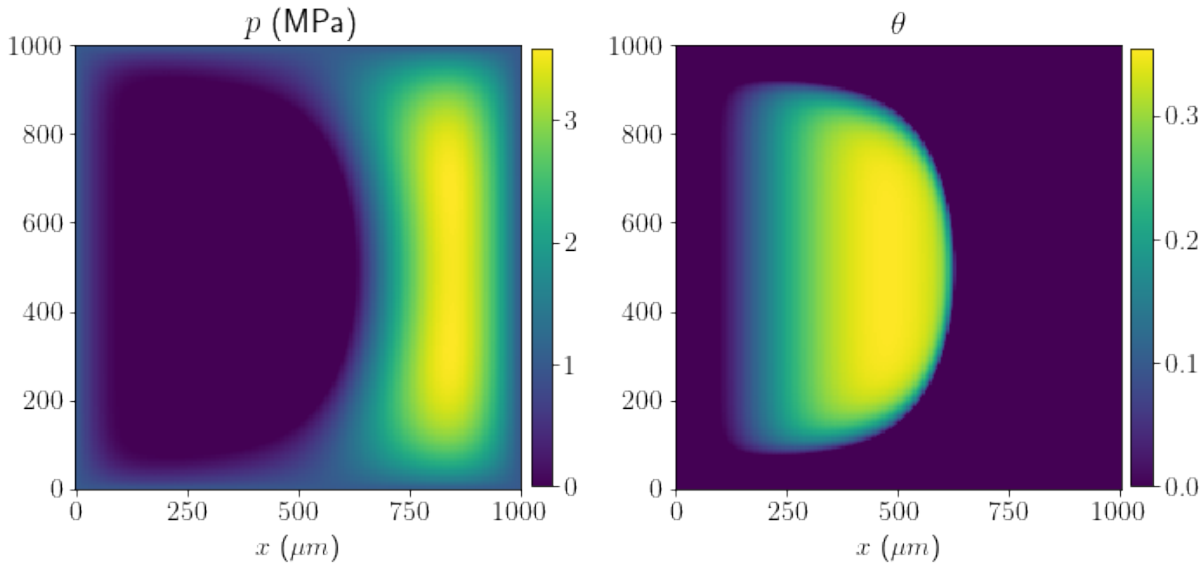


Figure 16: Surface plots of $h/h_{max}(x, y)$, $p(x, y)$ and $\theta(x, y)$ of problem 3 solved in a 200×200 element mesh.

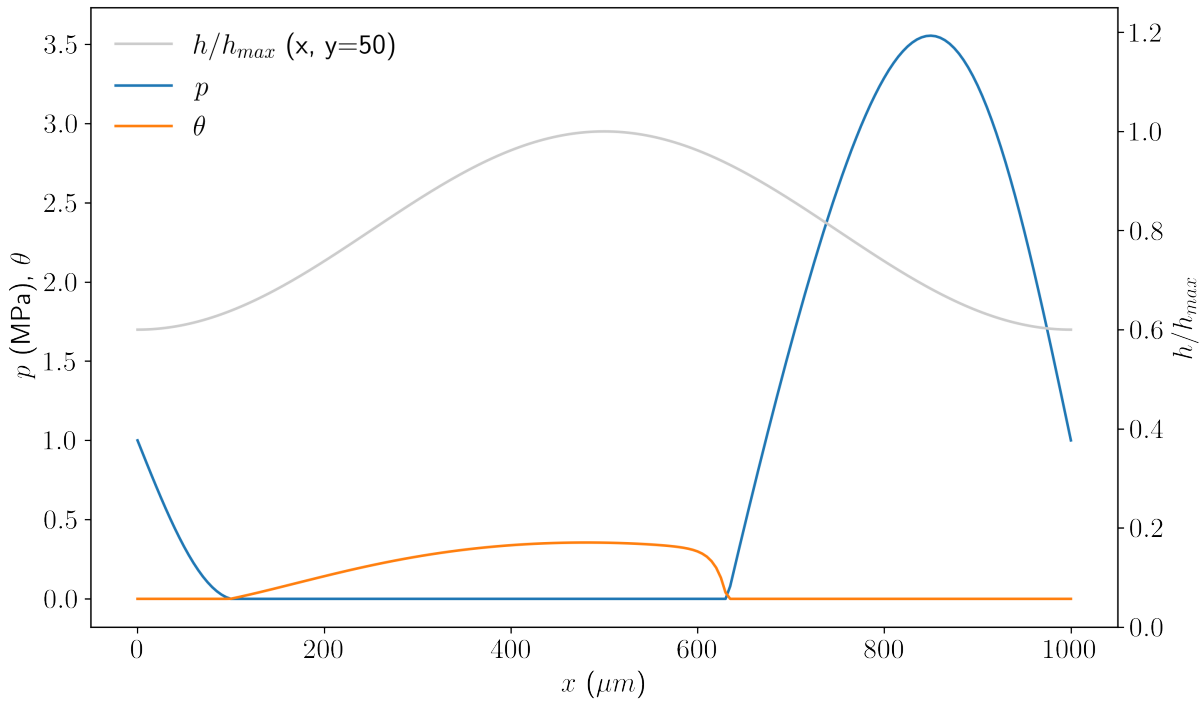


Figure 17: Plots of a slice at $y = 50\ \mu m$ of $h/h_{max}(x, y)$, $p(x, y)$ and θ of problem 3 solved in a 200×200 element mesh.

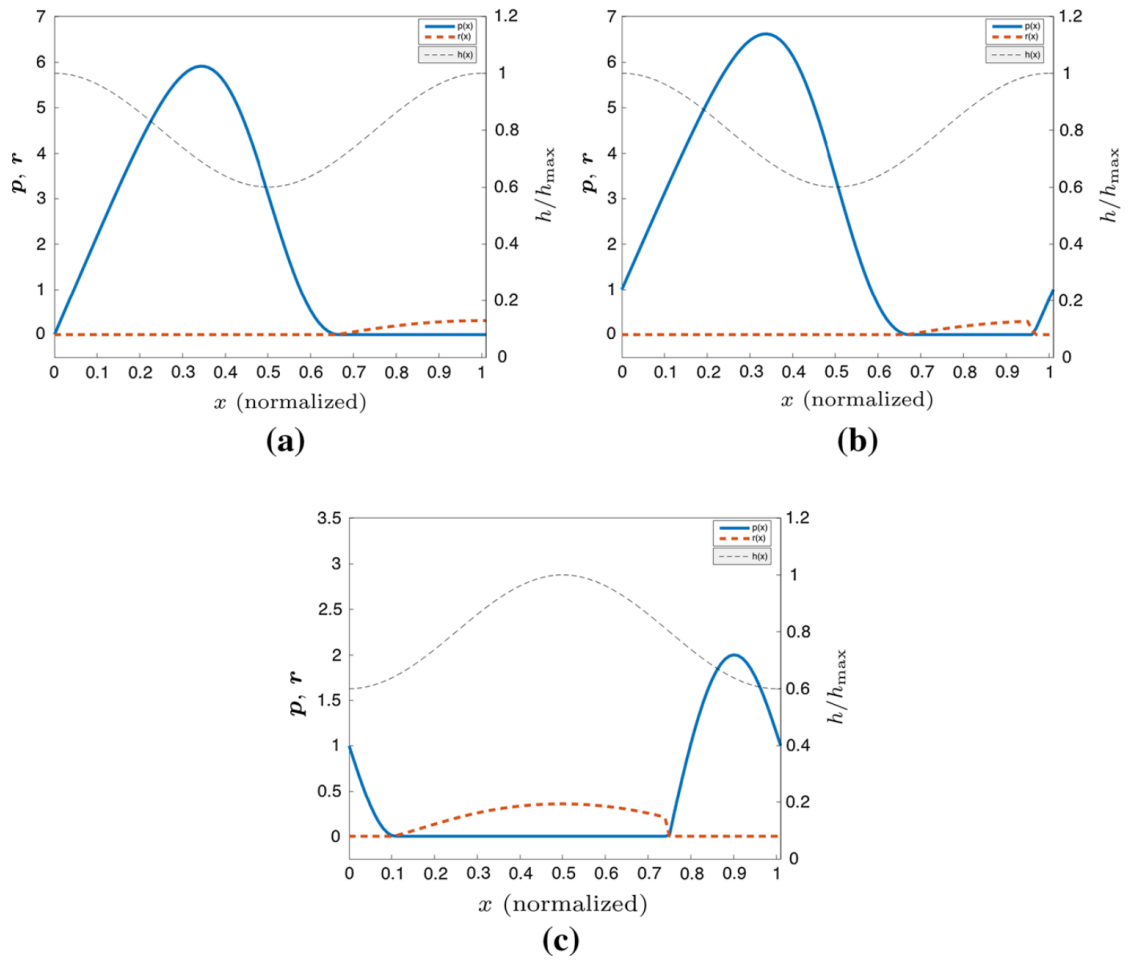


Figure 18: Solution profiles of the Reynolds equation in a 1D setup in [2] for the 3 cases described in table 2. r in these plots refers to our cavitation fraction θ. Reproduced with permission from [2].

3.1.2 Dimpled textures

The code was also tested for a dimpled texture, which is an example of a texture of industrial relevance. For this particular case, we consider a set of 9 equally spaced dimples, with depth $D_d = 10\ \mu m$ and radius $D_r = 60\ \mu m$, placed on a parabolic profile

$$h(x, y) = 0.1 + 2 \times 10^{-5} \cdot (x - 500)^2 \quad (\mu m). \tag{3.2}$$

Considering a 2D square domain of side $L = 1000\ \mu m$, lubricant viscosity $\mu = 0.035\ Pa \cdot s$ and relative sliding velocity $U = 4.3 \times 10^{-2}\ m \cdot s^{-1}$ the resulting height, pressure, and cavitation 2D profiles and 1D slice at $y = 500\ \mu m$ are shown in figures 19 and 20 respectively.

One can observe that the central cavitation region is diminished along with pressure. Notice also that dimples far from the center have a much lower influence on pressure as they are relatively far away from the contact region, where the fluid pressure is maximal.

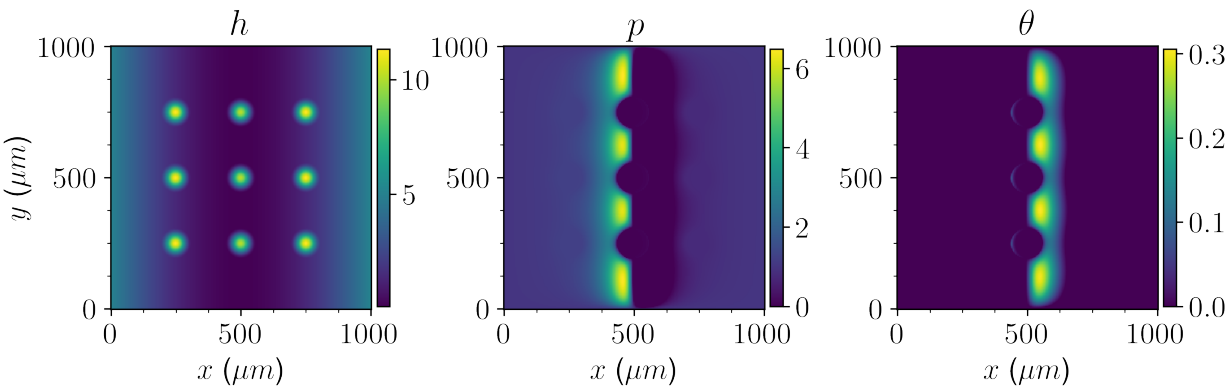


Figure 19: Surface plots of $h/h_{max}(x, y)$, $p(x, y)$ in atm and θ of the dimpled texture solved in a 200×200 element mesh.

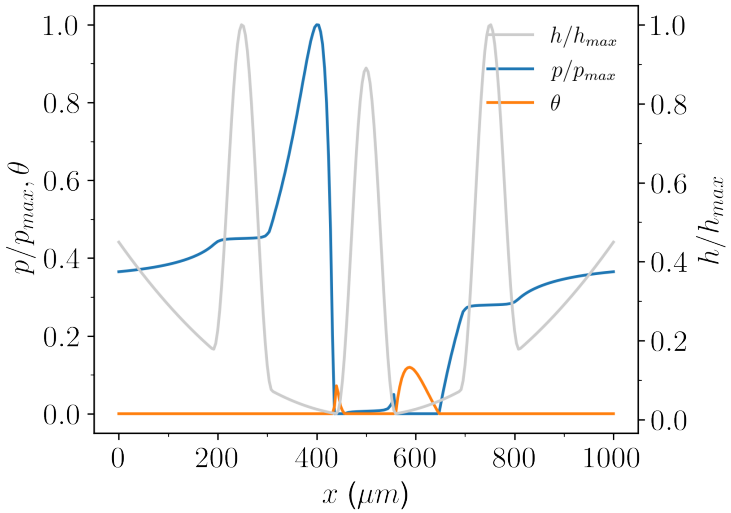


Figure 20: Normalized plots of a slice at $y = 500\ \mu m$ of $h/h_{max}(x, y)$, $p/p_{max}(x, y)$ and θ of the dimpled texture solved in a 200×200 element mesh.

To further verify the robustness of our implementation, we calculated the pressure/cavitation profiles for a set of 9 equally spaced inverted dimples with depth $D_d = 1\ \mu m$ and radius $D_r = 100\ \mu m$ placed on a parabolic profile

$$h(x, y) = 1.05 + 0.2 \times 10^{-5} \cdot (x - 500)^2.\ (\mu m) \tag{3.3}$$

Considering a 2D square domain of side $L = 1000\ \mu m$, lubricant viscosity $\mu = 0.035\ Pa \cdot s$ and relative sliding velocity $U = 4.3 \times 10^{-1}\ m \cdot s^{-1}$ the resulting normalized height, pressure and cavitation 2D profiles and 1D slice at $y = 500\ \mu m$ are shown in figures 21 and 22 respectively.

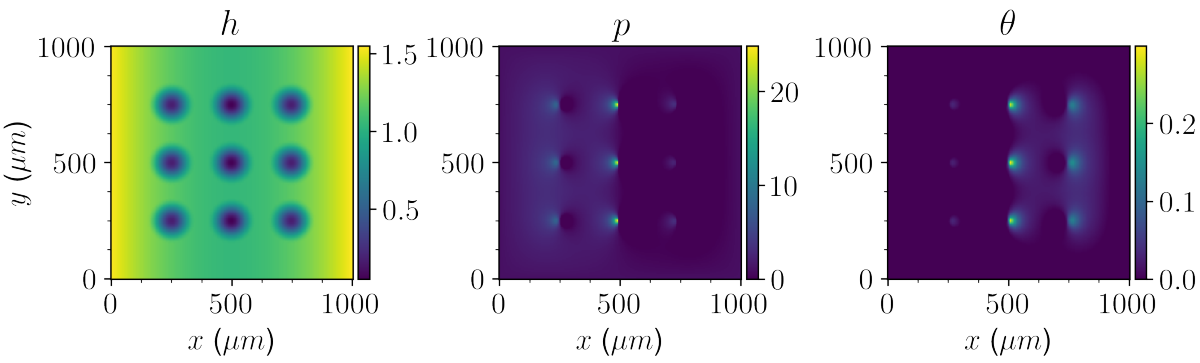


Figure 21: Surface plots of $h/h_{max}(x, y)$, $p(x, y)$ in atm and θ of the inverted dimpled texture solved in a 200×200 element mesh.

This is typically a hard to solve problem as the inverted dimples introduce very localized contacts between the surfaces as seen in figure 21, hence extremely high pressures at the dimple peaks are found. For each of the inverted dimples, a cavitation region is generated.

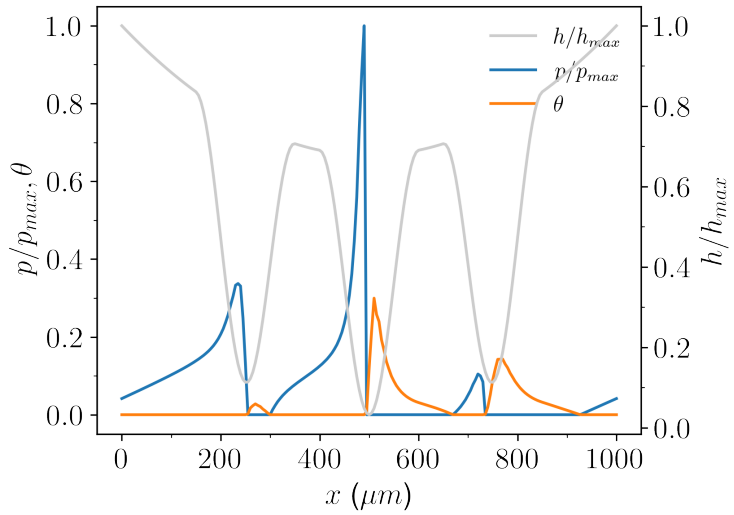


Figure 22: Normalized plots of a slice at $y = 500\ \mu m$ of $h/h_{max}(x, y)$, $p/p_{max}(x, y)$ and θ of the inverted dimpled texture solved in a 200×200 element mesh.

This demonstrates that FELINE is robust enough to successfully solve complex cases, with the centrality condition being called twice and taking around the double of the number of iterations to achieve convergence compared to the previous example.

3.1.3 Benchmark

To assess the performance of FELINE, we ran the code for a dimpled texture case and increasing mesh sizes $N \times N$ on a single core of a Intel(R) Xeon(R) CPU E5-2697 v2. The number of iterations, total

execution time and convergence parameters defined in equations (2.86) are shown in 3.

From the reported times, we can see that the solver is rather fast even if, as expected the computational times increase exponentially with mesh size since the Jacobian matrix scales as N^2. FELINE also shows an impressive 100-fold speedup when compared with the results of a 100×100 mesh reported in [2].

Mesh size	**Total computation time** (s)	**Iterations**	$\mathbf{C_1}$, $\mathbf{C_2}$ **defined in (2.86)**
50	1.51	19	5.45×10^{-10}, 6.79×10^{-5}
100	6.07	24	4.04×10^{-11}, 2.76×10^{-5}
150	16.8	24	8.36×10^{-12}, 2.76×10^{-5}
200	49.6	26	1.24×10^{-11}, 9.73×10^{-5}
250	87.2	29	4.82×10^{-11}, 6.91×10^{-5}

Table 3: Benchmark of the INE algorithm results utilizing SuperLU for varying levels of mesh coarseness.

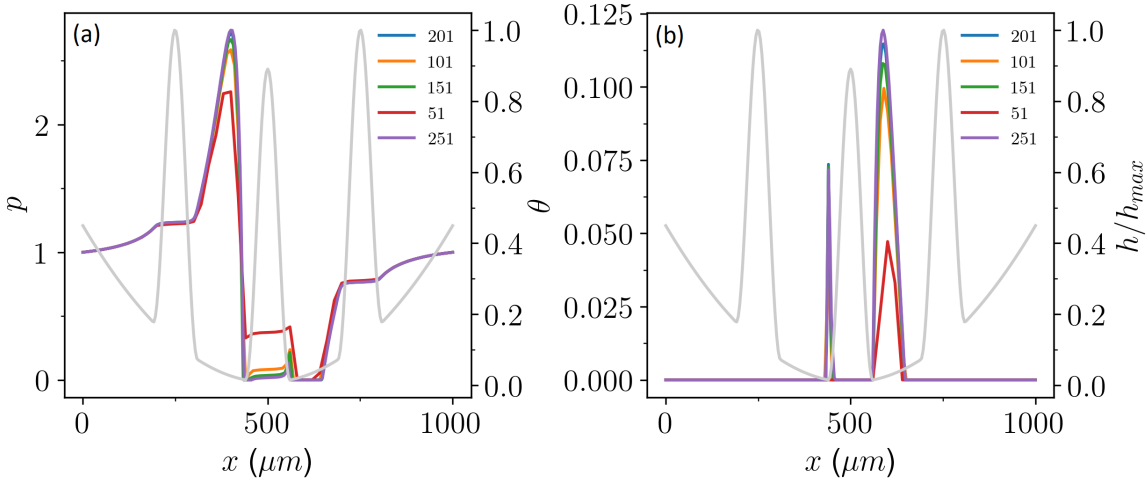


Figure 23: Plots of: (a) a slice at $y = 500\ \mu m$ of $h/h_{max}(x, y)$, $p(x, y)$ in atm; (b) a slice at $y = 500\ \mu m$ of $h/h_{max}(x, y)$, $\theta(x, y)$ of the dimpled texture solved in a $N \times N$ element mesh.

The effect of mesh coarseness on the solution correctness was verified. In figure 23 the pressure and cavitation profiles at $y = 500\ \mu m$ is shown for different mesh sizes. The main difference in the pressure profile is in the region at $400 \leq x \leq 600$, where smaller mesh sizes appear to overestimate the pressure within the dimple region, while severely underestimating the first pressure peak at $x = 400$. Concerning cavitation, the region observed at $x \approx 450$ is completely absent for $N = 50$, while for all the other cases the cavitation region boundaries are matching.

3.2 Stribeck curve

3.2.1 Equilibrium between load and lift

In chapter 2 we discussed how to compute a coefficient of friction from results obtained by solving the Reynolds equation, namely from the pressure profile p and cavitation fraction θ and a set of friction parameters. In this section we will discuss results obtained utilizing this process together with FELINE.

Much of the thesis goal is to explore a optimization strategy that deals with a textured surface that can be parameterized quite simply in order to reduce friction in the system. The surface profile $h(x, y)$ can lead to the equation system turning singular if $\frac{\partial h}{\partial x} = 0$, by definition we introduce a term $h_p(x, y)$ such that $\frac{\partial h_p}{\partial x} \neq 0$ ensuring a non-conformal contact. The portion of the profile we attempt to optimize is added to the parabolic non-conformal part as

$$h(x, y) = h_{min} + h_p(x, y) + h_{tex}(x, y)\,, \tag{3.4}$$

where h_{min} is the minimum distance established by hydrodynamic lift from any point in the texture to the sliding surface, ensuring we are not in a dry contact regime. In $h_{tex}(x, y)$ we encode the particular texture we look to study.

As previously discussed, FELINE provides a framework to solve and therefore obtain pressure and cavitation fraction profiles for a given height profile $h(x, y)$ and sliding speed U, which allows us, as discussed in chapter 2, to compute hydrodynamic lift and friction. For a given applied load to the surfaces F_N we look for the equilibrium situation where the hydrodynamic and contact lift counterbalance the applied load. We follow the process described in figure 24 to obtain a coefficient of friction of the system for a load F_N, a range of Hersey numbers, a height profile $h(x, y)$ and known surface roughness parameters.

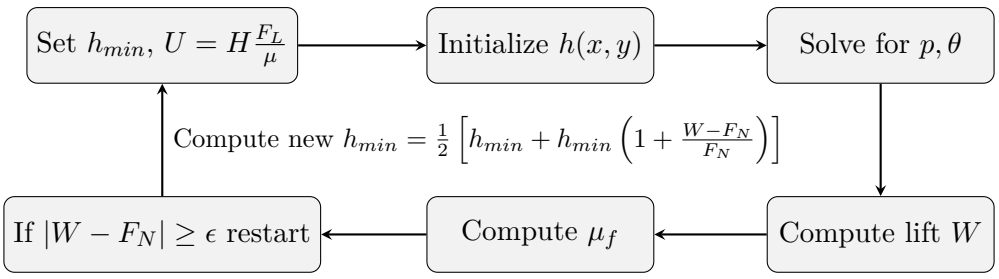


Figure 24: Flowchart of the process of determining the coefficient of friction, the iteration is considered complete if $|W - F_N| \leq \epsilon$ providing a converged value of μ_f.

3.2.2 Dimpled textures

Consider the following setup: a 2D square domain of dimensions $L_x \equiv L_y = 1\ mm$, lubricant viscosity $\mu = 0.035\ Pa\,s$, applied load $F_N = 1.5\ N$ and the non-conformal component of the height profile is defined as

$$h_p(x, y) = 2.0 \times 10^{-5} \left(x - L_x/2\right)^2 \quad (\mu m)\,. \tag{3.5}$$

The texture profile $h_{tex}(x, y)$ is constructed by placing at (x_0, y_0) dimples of square cosine line-shape with depth D_d and radius D_r. The dimples are placed within a square 5×5 grid leaving a $50\ \mu m$ border around the texture, as it is represented in figure 25.

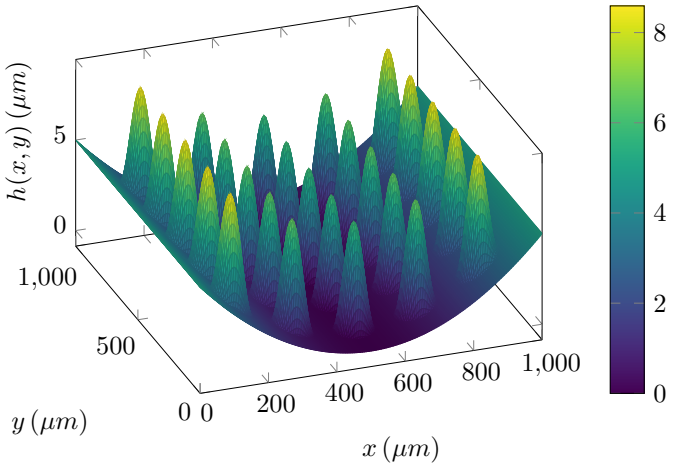


Figure 25: Height profile $h_{tex}(x, y)$ generated by a 5×5 grid of dimples with $D_d = 6\ \mu m$ and $D_r = 60\ \mu m$ with a $50\ \mu m$ border.

We studied the effect of varying D_r and keeping $D_d = 6\ \mu m$ fixed in the overall Stribeck curve of the system, and in figure 26 we represent a set of Stribeck curves for a few different dimple radii within a Hersey number range $H \in [10^{-5}, 10^{-2}]$.

It is immediately obvious that the dimple radius has effects both in the mixed and hydrodynamic regimes of lubrication, up to 30% for some Hersey numbers. In order better quantify these differences we must set a reference: the untextured case of this setup where $h_{tex}(x, y) = 0$ can show us whether the texturing works to reduce friction as well as which radii does so most effectively.

Defining Stribeck curve of the untextured case to be $\mu_f^0(H)$ and the textured Stribeck curve of dimple radius D_r to be $\mu_f^{D_r}(H)$ we can calculate the deviation from this curve as a simple difference

$$\Delta\mu_f^{D_r}(H) = \mu_f^{D_r}(H) - \mu_f^0(H)\,, \tag{3.6}$$

where if $\Delta\mu_f^{D_r}(H) < 0$ the texture is successful in reducing the friction in the system at a specific Hersey number H.

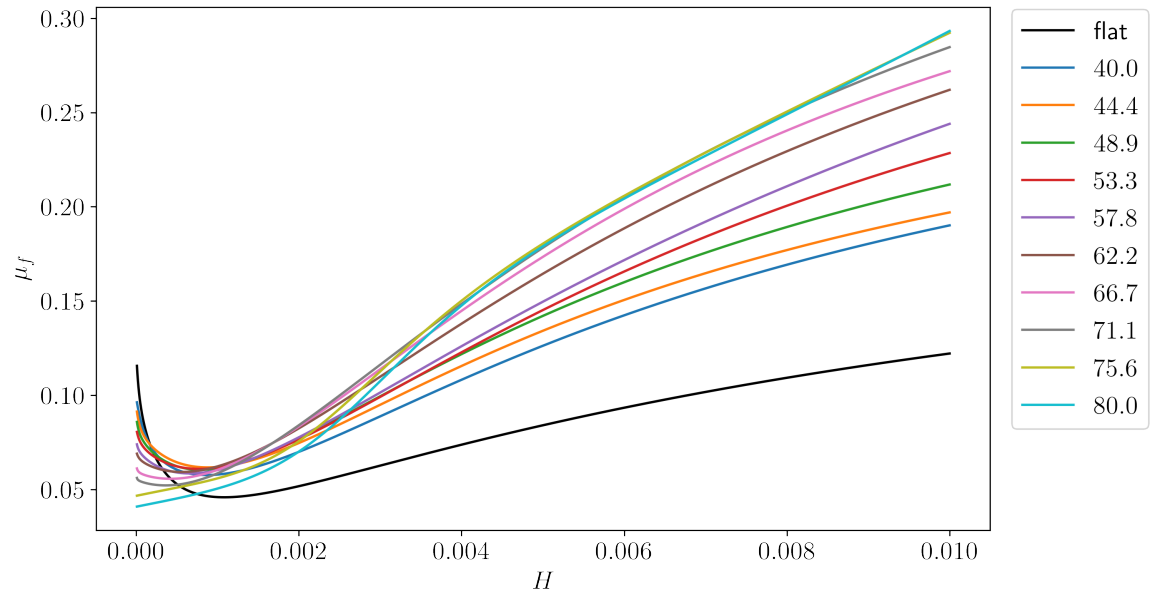


Figure 26: Stribeck curves of the fully dimpled case described in this section with varying dimple radius in μm as well as the flat untextured case.

If the interest is global reduction of friction (not regime dependent) we can study the overall deviation of coefficient of friction over the entire Hersey spectra, as follows

$$M_f^{D_r} = \int \Delta\mu_f^{D_r}(H)\, dH \tag{3.7}$$

for a range of 10 dimple depths from $40\ \mu m$ to $80\ \mu m$ we plot in figure 27 the deviation from the reference untextured case as defined in 3.6.

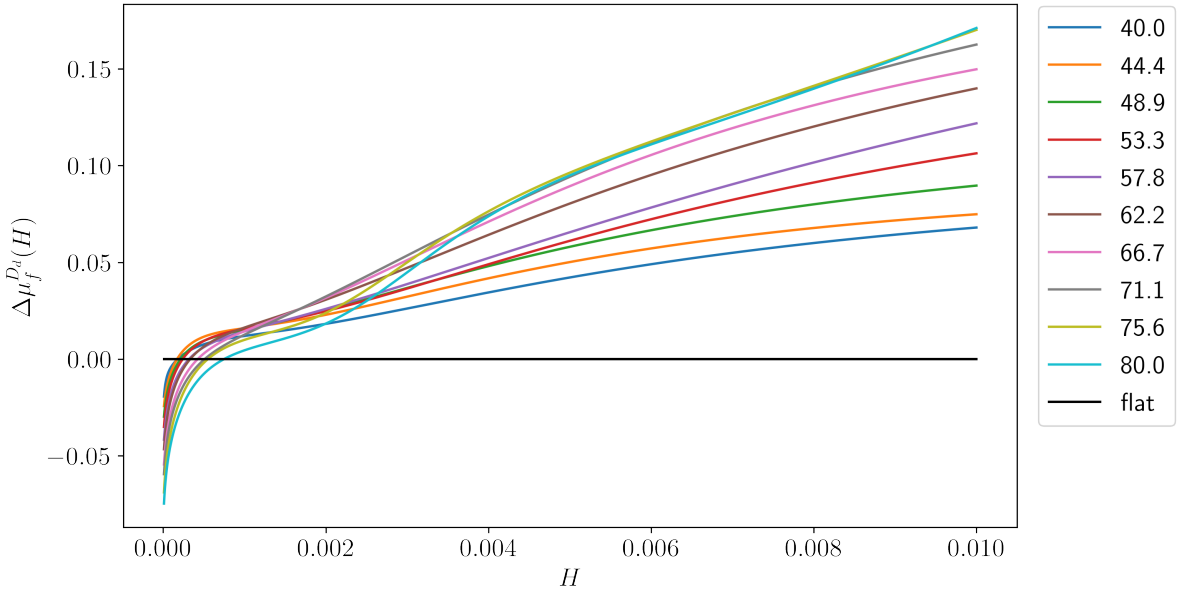


Figure 27: Difference between Stribeck curves of varying dimple radius (in μm) textures and the reference untextured surface.

In general over the entire Stribeck curve no texture performs better overall than the flat case, as we can see by computing M_f^{Dr} for each dimple radius, as shown in figure 28. This means that if the goal is to operate a part under different lubrication regimes it is unlikely that a full dimpled would be of any use.

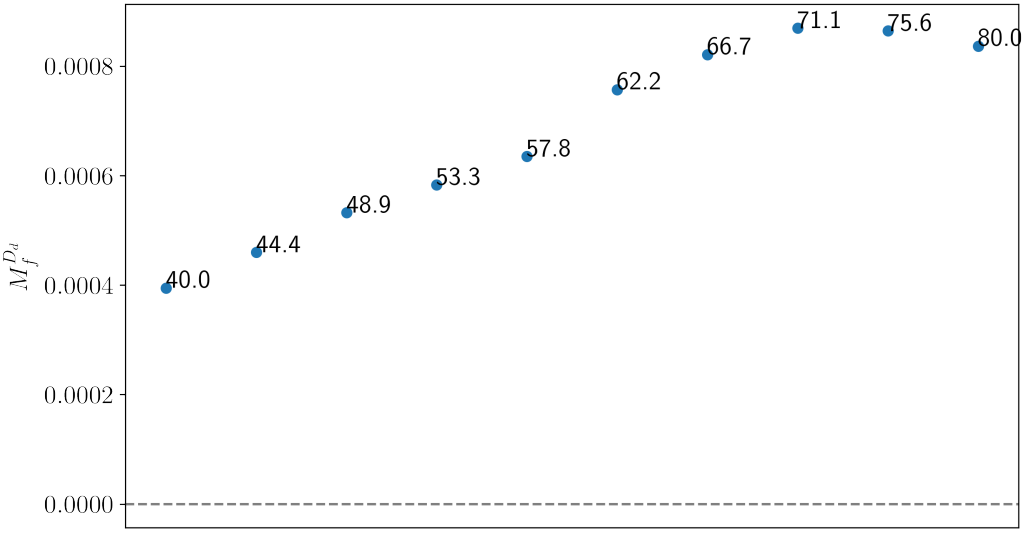


Figure 28: Total coefficient of friction difference between dimpled cases with radius D_r and the flat case defined in equation 3.7 for the mixed and hydrodynamic regions.

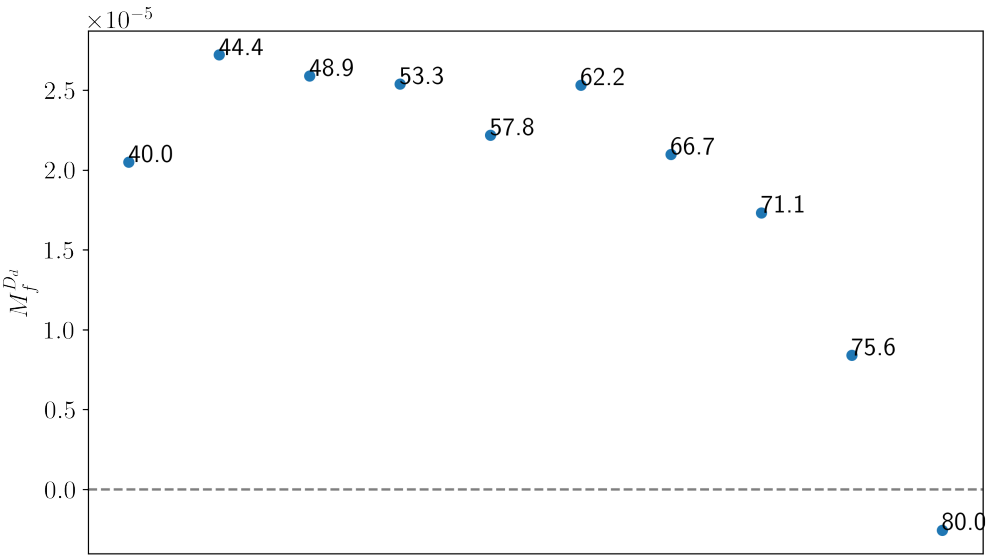


Figure 29: Total coefficient of friction difference between dimpled cases with radius D_r and the flat case defined in equation 3.7 for the mixed region.

However, assuming that we are only working with one lubrication regime, then it can be interesting to assess the behavior and performance of the textures. Setting the mixed lubrication regime to be in the interval of Hersey number from 0 to 0.002 we calculated equation 3.7 as shown in figure 29. By isolating

the mixed regime we can highlight a particular trend: for $D_r < 65\ \mu m$ there is no qualitative effect of lowering or increasing the dimple radius, however from $65\ \mu m$ to $80\ \mu m$ we see a sharp decrease in overall friction in the mixed regime. This decrease results in $D_r = 80\ \mu m$ showing improvement over the flat case in the mixed region, this is associated with textures acting as lubricant reservoirs, which typically leads to higher radii performing better in the mixed lubrication regime.

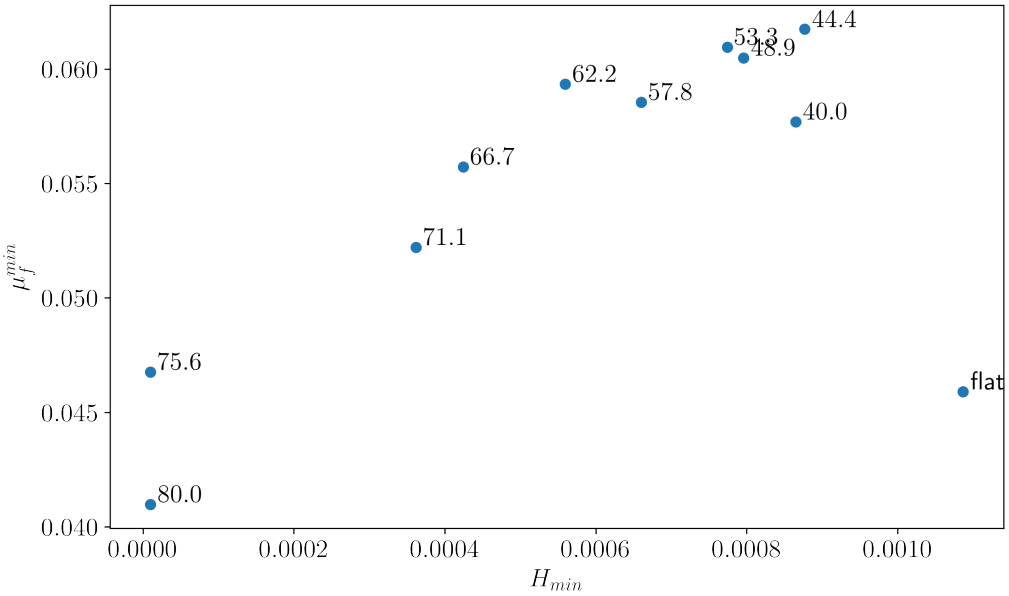


Figure 30: Stribeck minimum coordinates of the fully dimpled case described in this section with varying dimple depth in μm as well as the flat untextured case.

It is also possible to check how the minimum of the Stribeck curves changes with dimple radius to have an idea of the overall position of the mixed region. The coordinates of the minimum of each curve are represented in figure 30. Clearly there is leftwards trend with growing dimple radius as well as decreasing magnitude of the minimum, however, it is important to highlight that for $75.6\ \mu m$ and $80\ \mu m$ we seem to miss the growing behavior of the coefficient of friction in the region where $H < 10^{-5}$.

We can perform similar simulations but varying the dimple depth instead of the dimple radius, by keeping $D_r = 60\ \mu m$ and allowing D_d a range from $6\ \mu m$ to $12\ \mu m$. We plotted the resulting Stribeck curves in figure 31.

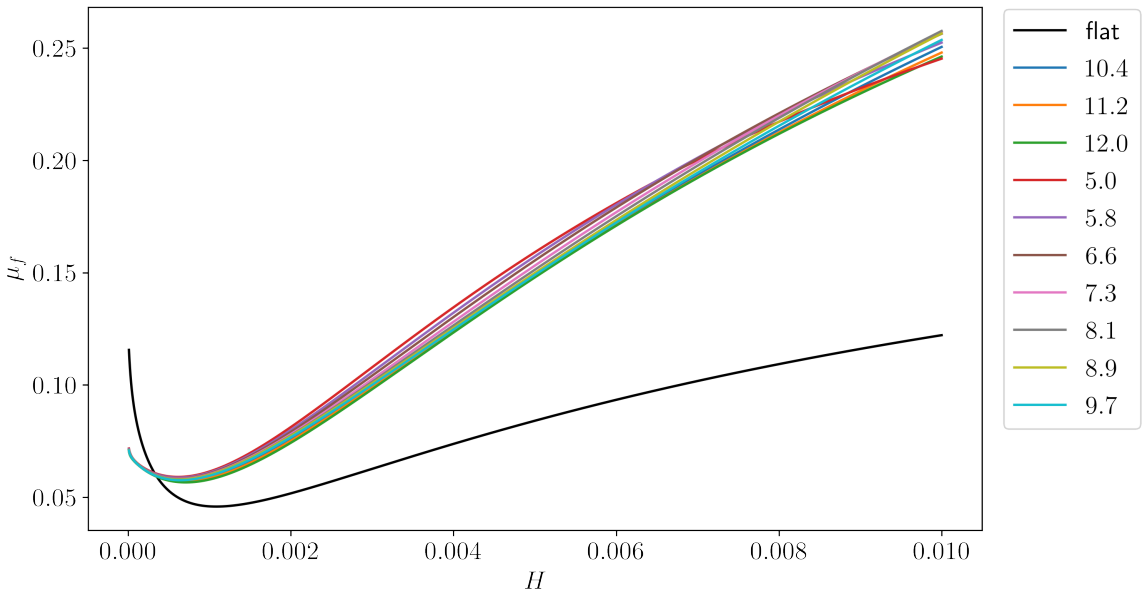


Figure 31: Stribeck curves of the fully dimpled case described in this section with varying dimple depth in μm as well as the flat untextured case.

Compared to the change in dimple radius in figure 26 we can see that the change in dimple depth has a less significant effect in the Stribeck curves, since the overall change with dimple depth between curves is under 4%. The overall change when compared to the flat case is represented in figure 32.

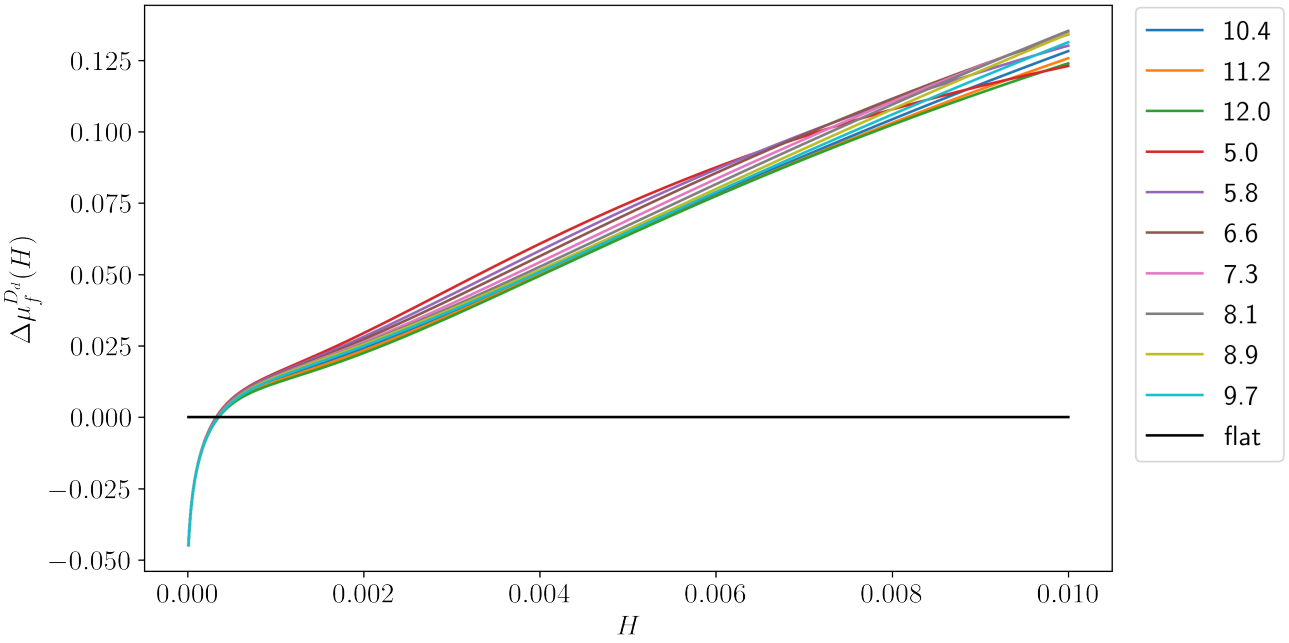


Figure 32: Difference between Stribeck curves of varying dimple depth (in μm) textures and the reference untextured surface.

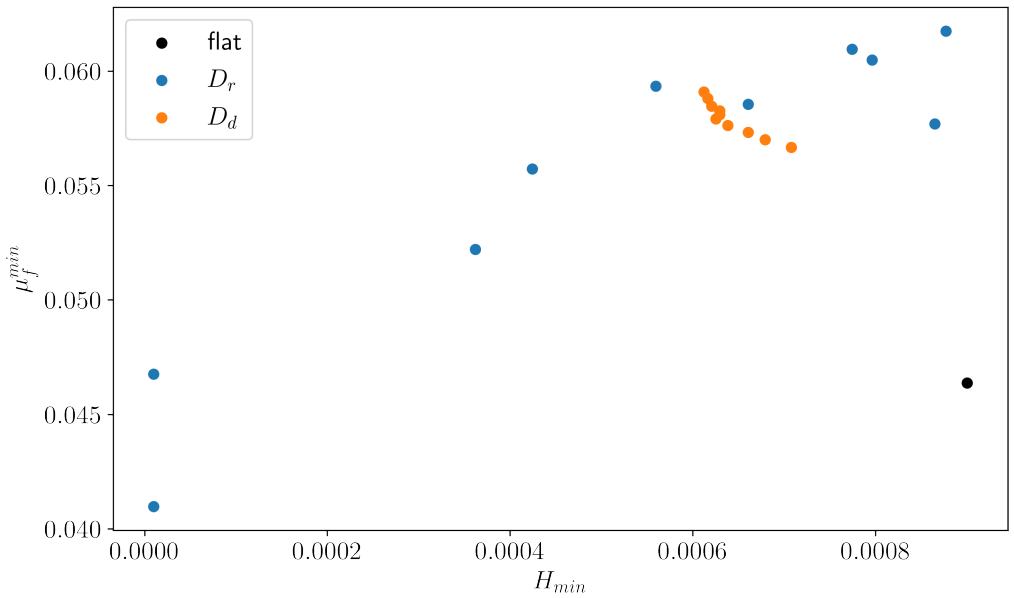


Figure 33: Minimum position for the flat case in black, for the radii cases in blue and for the depth cases in orange. A difference in impact is apparent from the relative movement of the minimum in terms of the relevant parameter.

It is quite intuitive from figure 32 that in general none of the textures performs better than the flat case. It is worth noting that while there was one texture with varying D_r which performed better up to $H = 0.002$, every texture with varying D_d performs worse in this range, indicating that changing the radii provides a larger impact over the Stribeck curves, represented in figure 33. This means that we can regard the dimple radius as a more promising parameter to optimize, while neglecting the dimple depth.

3.2.3 Friction optimization

A crucial problem is to minimize the friction while maximizing the lift, at some regimes or at others. In principle, this could be achieved by applying a texture which has been optimized for some conditions (viscosity, load, etc..). However, since the relationship between textures and Stribeck curves is not clear, finding the optimal texture is practically impossible with direct approaches. For instance, in the previous section only two cases have been discussed: a flat untextured case and a fully dimpled texture case. However, considering the same setup of a 5×5 grid of dimples, if we allow the possibility of placing or not a dimple at spot (i, j) where $i, j = 0, 1, ..., 4$ in the grid we have a total of $2^{25} \approx 33$ million possible textures. Simple considerations of symmetry in the Reynolds equations allow us to reduce the number of possible textures by half. Textures P and P' in figure 34 are entirely equivalent. A mirror reflection along the x axis and at $y = L_y/2$, L_y being the length along y, for any pattern P results in an equivalent pattern P' that yields the same Stribeck curve as shown in figure 35.

Since flat and fully dimpled textures do not have a distinguishable mirrored equivalent, the total number of different possible Stribeck curves is approximately $2^{25}/2 \approx 1.68 \times 10^7$ which constitutes a close enough approximation for simplicity. This way, we reduced by half the number of possible textures, but the set is still large.

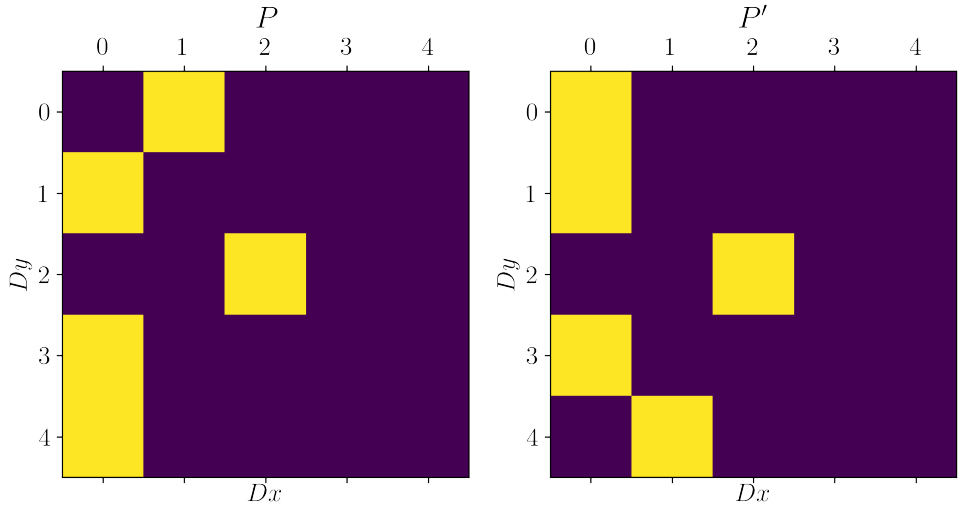


Figure 34: Regular pattern P and mirrored result P' represented as a dimple mapping where (D_x, D_y) are the coordinates of the dimple map and the colors give information about the placement (or not) of a dimple at (D_x, D_y): yellow means an existing dimple and purple a free spot.

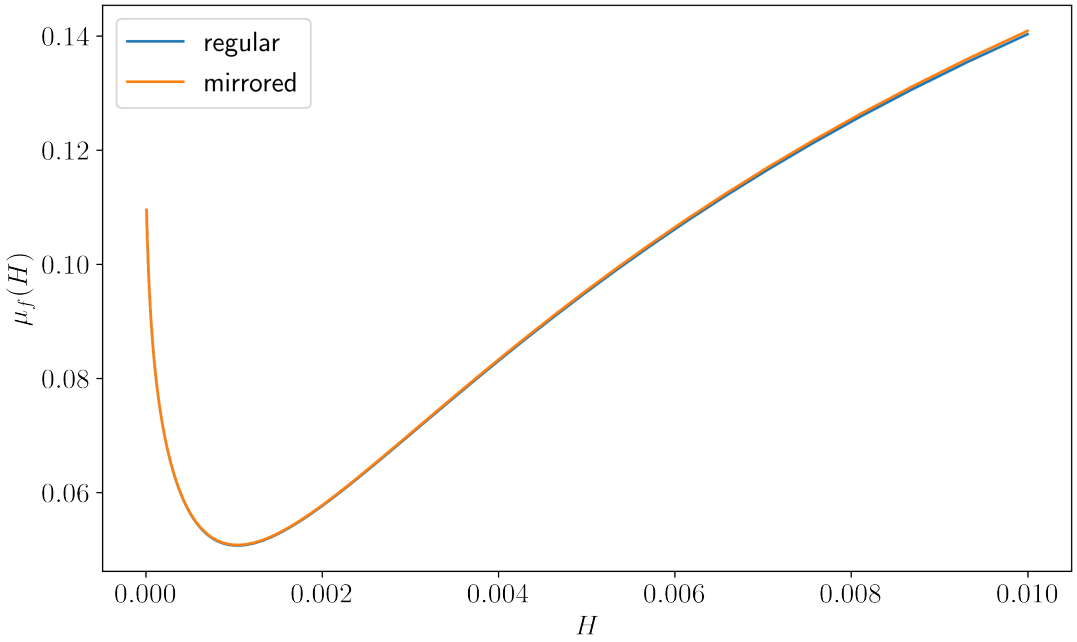


Figure 35: Regular pattern P and mirrored result P' stribeck curves, shows the symmetric property of the Reynolds equation.

This calculation does not consider parameters such as D_d and D_r which can be considered global, such that they apply to every dimple, or local where each dimple k has a set of parameters (D_d^k, D_r^k). Consider a range of 10 possible values for both D_d and D_r that apply locally to each dimple, the total number of possible textures considering symmetry is

$$N = \frac{1}{2}(2 \times 10 \times 10)^{25} \approx 1.68 \times 10^{57}\,, \tag{3.8}$$

this number is 50 orders of magnitude bigger than the possibilities without the two dimple parameters.

A simpler yet still interesting configuration is to consider as discussed in previous sections that depths D_d have a negligible impact on the Stribeck curves when compared to the radii D_r. As such we can remove a degree of freedom, and we can also assume that the parameter D_r is global, such that all dimples in the texture have radius D_r. With these considerations in mind and considering 10 possible values of D_r there are now

$$N = \frac{1}{2}\,10 \times 2^{25} \approx 1.68 \times 10^{8}\,, \tag{3.9}$$

possible configurations. Still there are an order of magnitude more configurations than our first consideration but it does include the physics of the dimple radii. For each global parameter that we add with n possible values, we multiply the configuration space of the problem by n.

Keeping in mind our benchmark of FELINE in chapter 2, the computation of a Stribeck curve takes anywhere from 15 to 20 minutes, even considering the best possible scenario and assuming it takes 10 minutes per curve, if our goal was to compute our entire configuration space (3.9) in order to find the best performing configuration, the total amount of time required would be above 3000 years. This clearly demonstrates that to solve the texture optimization problem a better and faster approach is needed. The next chapter will address exactly this point, showing how this can be achieved by a machine learning application.

Chapter 4

Deep neural network

In chapters 2 and 3 we discussed the details of our implementation of a fast and robust Reynolds equation solver termed FELINE and showed its powerful capabilities for the calculation of Stribeck curves. In section 3.2.3 we demonstrated that while the solver is an improvement in terms of speed and accuracy when compared with existing solutions, it is still too slow to be directly used in our optimization problem. In this chapter we will apply the fundamental concepts of machine learning discussed in chapter 2 to our problem, namely we will design, train and use a deep neural network to replace our FELINE solver and use it to solve the pattern optimization problem.

4.1 The forward problem

4.1.1 Definition

In this section, we will define the forward problem in terms of the parameters of the physical problem and introduce the dimensions of our problem, such as data set size and ratio of data set to configuration space. Furthermore, we will show the training process for a variety of neural network configurations in order to give some insight into the neural network functioning. Finally, we will show results that validate our neural network. For the implementation of the work done within this chapter, we used commonly available libraries in *Python* and the machine learning library *TensorFlow* [66].

To train our neural network we need to carefully define the problem that we are trying to solve. As there exist a large set of parameters that we could theoretically use to describe the problem, such as the texture dimensions, the type of pattern, the pattern specific parameters, the load applied to the system, the lubricant in use, etc., we restricted the set to a relatively small number to simplify the optimization

problem. Otherwise, as discussed in section 3.2.3, the configuration space of our problem might become far too big to be solvable even by machine learning methods, due to time or data constraints.

Thus, we consider the following fixed parameters for our problem: a texture with fixed dimensions (L_x, L_y) where $L_x = L_y = 1\ mm$, a fixed grid-like dimpled texture with grid dimensions (D_x, D_y) where $D_x = D_y = 5$, a fixed lubricant viscosity $\mu = 0.035\ Pa\,s$, fixed non-conformal profile and fixed roughness parameters (among other surface related parameters). The reason why these parameters are considered fixed is because they are problem specific, meaning that one can retrain the network for any given problem by adjusting these parameters during data generation. The more important point is to ensure that we are capable of obtaining a network that provides Stribeck curves for problems with varying dimple placement and dimple radius, which can be used for the texture optimization problem.

The variables in our system (or, in a machine learning sense, inputs) are: a 5×5 grid D_{map} corresponding to dimple placement, where a 1 corresponds to placing a dimple in the texture and 0 not placing a dimple and a value for the global dimple radius D_r, this in total means 26 input variables for our network $\{D_{map}, D_r\}$.

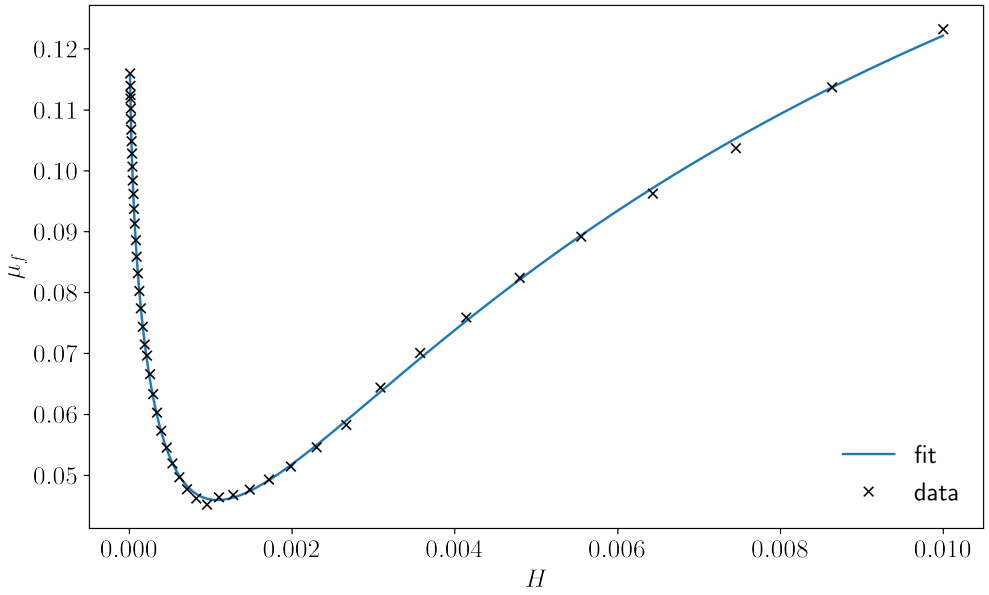


Figure 36: Fit versus data for the flat Stribeck curve case detailed in chapter 3.

More importantly, let us discuss the output of the network. In chapter 3, the resulting Stribeck curves were functions plotted from N coefficient of friction values calculated by the solver for N Hersey numbers. It is possible for certain input values $\{D_{map}, D_r\}$ that the solver fails. This failure typically happens closer to the edge between the mixed lubrication regime and the contact lubrication regime as it falls outside the validity of our solver.

The issue arising from this possibility is that if we choose to calculate a Stribeck curve composed of N points we may not obtain N points in every case. To circumvent this problem we identified a fitting function that allowed us to represent a Stribeck curve with an arbitrary number of points with only a set of $k < N$ parameters. Specifically, we fitted the Stribeck curve to a rational polynomial, defined as:

$$f_n^m(x) = \frac{p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1}}{x^m + q_1 x^{m-1} + \cdots + q_{m-1} x + q_m} \tag{4.1}$$

where n, m are the fit degrees and $\boldsymbol{p}, \boldsymbol{q}$ are the fit coefficients. We identified f_3^3 to be the most promising compromise between RMSE of the fit and the number of parameters, an example of this fit function is represented in figure 36. In summary: we have an input composed of 26 parameters representing the dimple placement and dimple radius and an output composed of 7 parameters that lets us to reconstruct a fit for the corresponding Stribeck function.

When looking in literature for similar applications of machine learning to physics problems we identified a recent work [49] that dealt with a problem of similar dimensions with configuration space approximately four times larger than ours. Considering the magnitude of their problem and the size of their training data set, it is possible to have a qualitative estimation on how much data the training of our network would require. In [49] the training was performed using ≈ 21000 data points, corresponding to $\approx 0.002\%$ of the total configuration space. In our case, we computed a total of around 120000 Stribeck curves from randomly generated dimple patterns and 12 possible radii within in a range of $D_r \in [40, 80]\ \mu m$. This corresponds to $\approx 0.03\%$ of the total configuration space, a much larger percentage than the one in [49], indicating that the amount of training data should be enough to obtain a well-behaved DNN. Clearly, an *a posteriori* check of this can be made once the network is trained. This data generation was done in approximately 300 simultaneously running processes in Intel(R) Xeon(R) CPU E5-2697 v2 cores for 11 days in total.

4.1.2 Data treatment

For a rational fit one common problem is that the parameters have wildly different magnitudes, a fact which is not ideal when it comes to the learning process of the network, because the step sizes calculated in the stochastic gradient descent (SGD) method used during the training process can wildly change for each feature. Scaling the data set in such a way that every parameter has the same order of magnitude helps to ensure that the steps in SGD are smoother. This greaty reduces the time required to achieve learning convergence. Hence, we applied a standard scaling to the input and output on our data points, which essentially converts the parameter values to their *z-score* [67], which is a measure of how far a data

point deviates from the mean of the data set and is defined as

$$z_k = \frac{x_k - \overline{x}_k}{\sigma_k}\,, \tag{4.2}$$

One of the key factors for a successful training of a machine learning application is good quality data, because outliers in data lead to networks that are unable to adapt to the full spectra of the data set. Thus, we evaluated the quality of our data by employing the *z-score* test.

After organizing and representing in histograms the individual output parameters that reconstruct the Stribeck curve fit, we noticed that each distribution had outliers. Investigating this issue, we realized that for larger dimple radius cases (approaching $80\ \mu m$) the respective Stribeck curve within the mixed region became flatter, leading to a fit that was noticeably different with respect to all the other Stribeck curves. This is due to the fixed range of Hersey numbers we used in our calculations, which is not adjusted to the leftwards minimum shift occurring for increasing dimple radius (see figure 33 page 46).

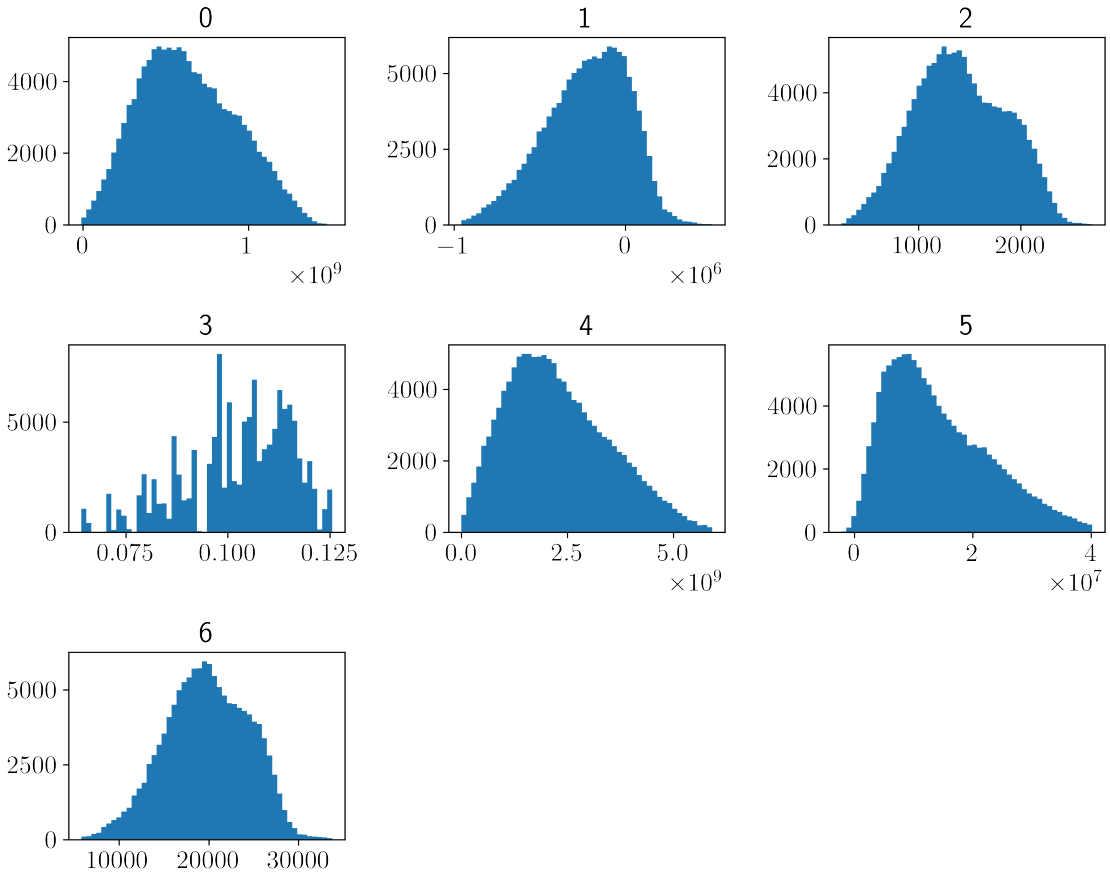


Figure 37: Histogram representation of each fit parameter from the 120000 case data set after removing the outliers.

Even if the outlier data removing procedure improves the overall learning, it makes the network perform worse for cases with larger dimple radius, because these are the cases mainly affected by the outlier removal. Considering a data rejection threshold of 3σ, where σ is the standard deviation for parameter

p, we removed the data point from the data set if $|Z| > 3\sigma$ where Z is the *z-score*. After cleanup, the histograms of each of the 7 parameters in our data set, which are represented in figure 37, were entirely free of outlier cases.

As discussed in 3.2.3 we have a mirror symmetry along the x-axis at $y = 0.5L$ in our system, so that for any given pattern P there exists a pattern P' that generates an identical Stribeck curve S. To make sure that this symmetry is enforced during the training of our network, we associated each Stribeck curve with its two symmetrically equivalent textures, resulting in a total of ≈ 240000 data points.

4.1.3 Training

To train the network we must divide the data into two sets: a training and a validation set. From our full data set we randomly selected 90% of cases to constitute the training set and the remaining 10% of cases were left as validation set. The purpose of the validation set is to assess how the network is generalizing the problem to data it has not been trained for. This is an important sanity check because, in principle, it is possible to obtain a network that performs very well within our training data, but terribly for any data it has not yet seen, an issue commonly known as *overfitting*. By distinguishing the training loss and validation loss we can tune the network parameters (such as number of layers/neurons) to prevent overfitting.

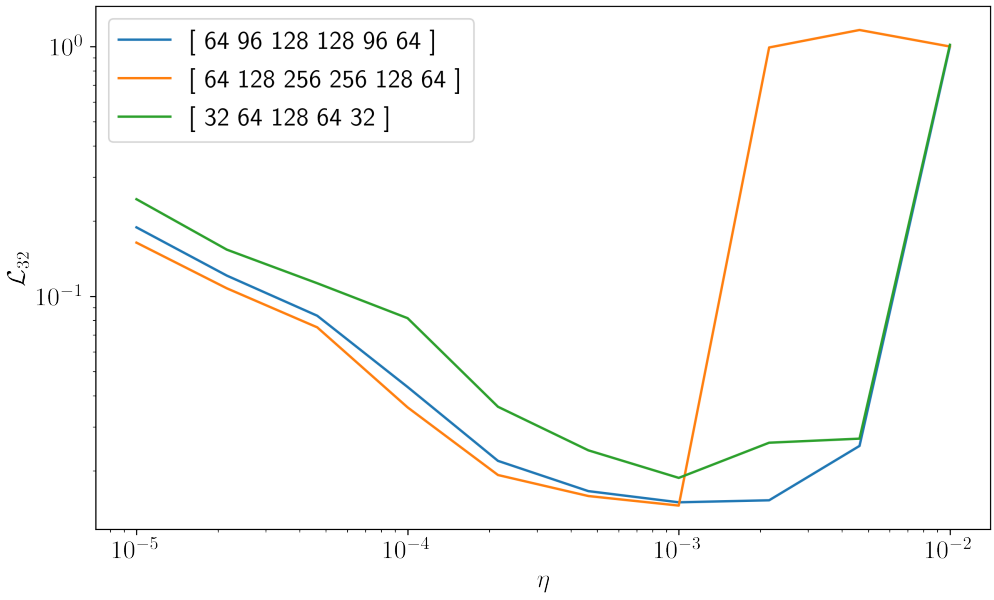


Figure 38: Learning rate polling for 3 different neural network models consisting of different numbers of layers and amounts of neurons in each layer respectively. The activation function utilized across every hidden layer was the $ReLU$ function.

The loss function of the network is the mean squared error (MSE) defined in equation 2.90, while

the activation function used across every hidden layer is the $ReLU$ function [68] and the loss function optimization algorithm is the stochastic gradient descent. These choices are safe and common options for regression problems [69]. More sophisticated learning algorithms such as *Adam* [70] can lead to faster convergence of the learning process, however these are typically applied to larger scale problems, which is not our case. In the present work we are mostly interested in demonstrating the feasibility and reliability of machine learning applications to texture optimization problems rather than finding the fastest algorithm.

To identify the best learning rate we adopted a technique called learning rate polling [71]. The main idea is to compare the loss of a neural network model M after N steps of learning for different learning rates η. We polled the loss in terms of learning rate of 3 distinct neural network models, with different layer counts and neuron counts per layer. After 10 steps (or epochs) we stopped the learning and determined the corresponding loss, we performed this analysis for our 3 networks and represented the corresponding losses in figure 38. By finding the minimum of each of these curves we can determine the optimal learning rate for each network model. We trained these 3 networks with their respective optimal learning rate for a total of 100 epochs and the respective training loss and validation loss over these 100 epochs are represented in figure 39.

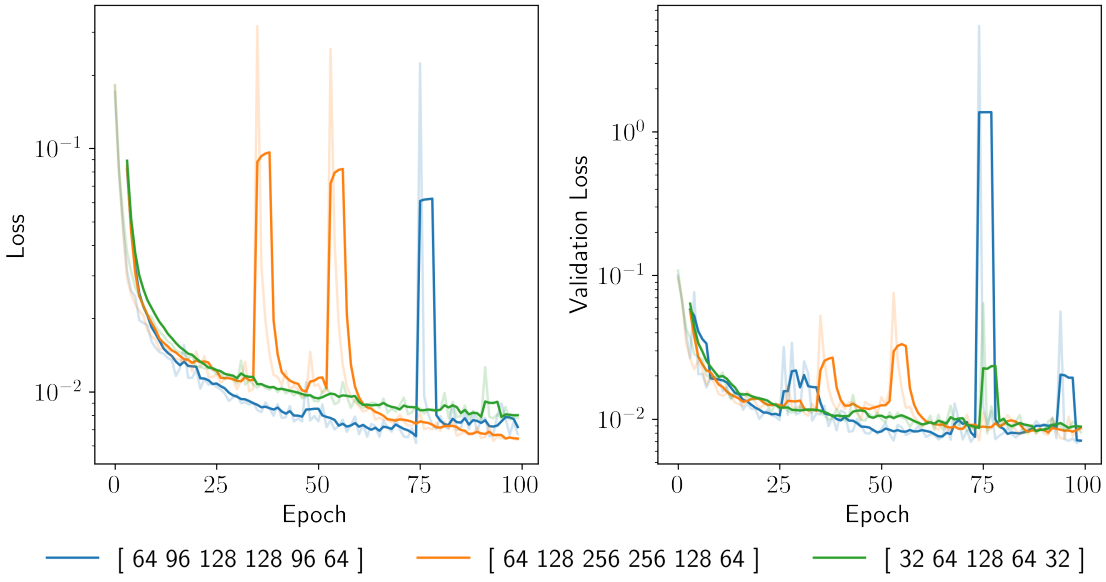


Figure 39: Learning curves of training and validation loss for 3 different neural network models consisting of different numbers of layers and amounts of neurons in each layer respectively.The solid lines are a moving average of the real losses, allowing to see more clearly the performance of each network model.

As one can see in figure 39, the learning process is not a simple descent to convergence but a clear stochastic process. Although it is possible to train the networks for more than 100 epochs, the rate at which they learn afterwards is minimal when compared to these first 100 epochs. The orange model

appears to perform better in the training data set after 100 epochs, however it is slightly outperformed by the blue model in validation loss, which provides the smallest validation set loss after 100 epochs of training.

After observing these curves, we decided to adopt the blue model in figure 39, which is composed of six inner layers with a maximum layer number of neurons (128) in inner layers 3 and 4. Once the training is completed, to validate the DNN performance we must take a set of cases from the validation set and compare the data to the DNN output. In the next section we will explore some comparisons between the DNN results and the solver direct ouptut.

4.1.4 Results

In order to verify that our DNN accurately predicts results calculated by the solver, which are our reference data, we selected six random cases from the validation data set and plotted both the solver results and the DNN predictions in figure 40. We can see that the DNN accurately predicts the cases with a significant deviation only in the hydrodynamic region of the Stribeck curve. Since the relevant region in real life applications is the mixed region, for cases of interest the DNN provides an incredibly accurate results, as it will be clear below.

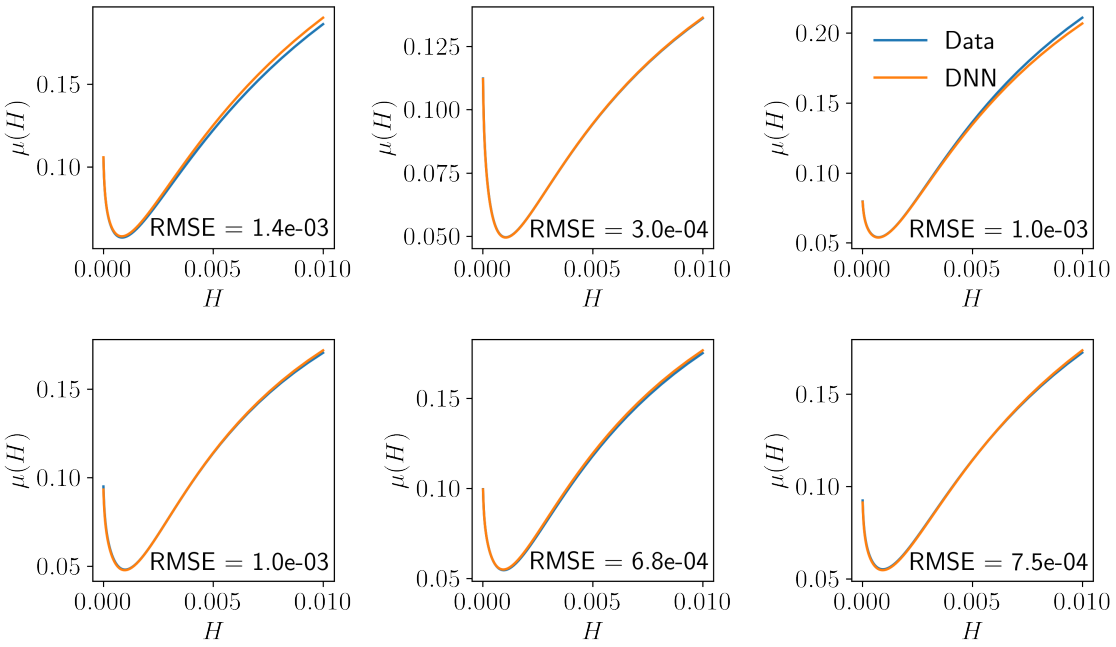


Figure 40: Comparison between the results calculated by the DNN after training and the data from the solver for random arrangements of dimples and dimple radii. The RMSE value provides a measurement for the performance of the DNN.

The RMSE associated with each of the cases also provides us with an estimate on the average deviation from data: since the units of the RMSE are that of the coefficient of friction, this means we deviate at most 2% from the solver result in the case at the top left.

In order to identify a more useful measurement of the DNN error we computed the RMSE from data for each case in the validation set. By representing these in a histogram, as plotted in figure 41, we can see how the RMSE is distributed and we can estimate an average RMSE for the whole network. By plotting the RMSE for the full Stribeck curve, and by specifying it for the mixed and hydrodynamic regions, it is possible to see that the DNN performs better in the mixed region, thus supporting our previous claim.

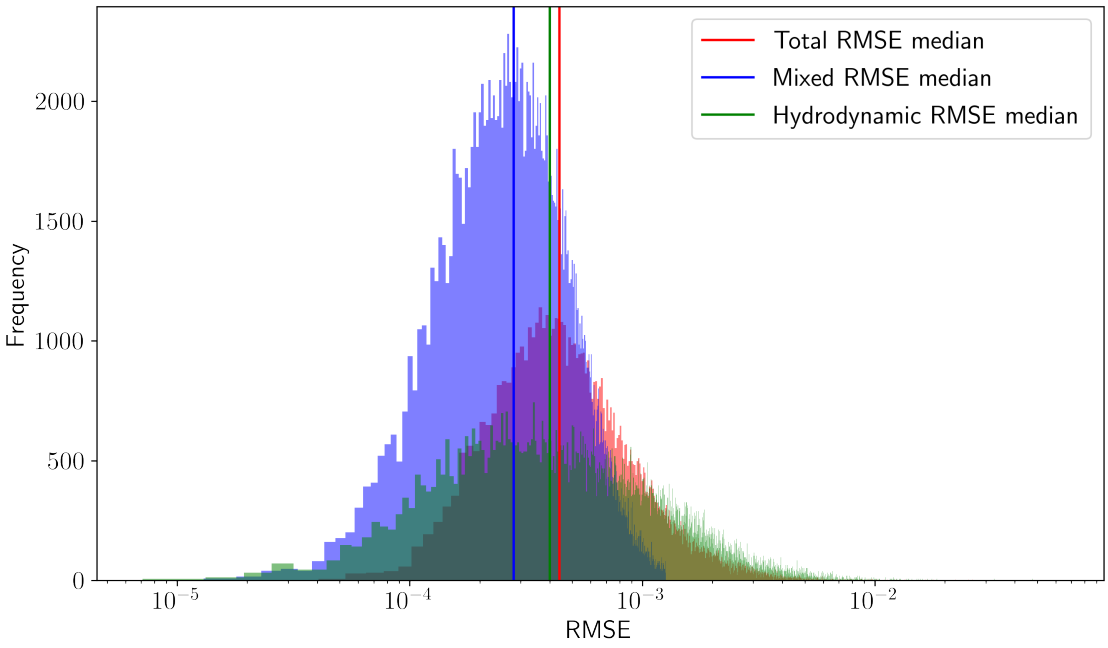


Figure 41: RMSE distributions and median for the total Stribeck, mixed region and hydrodynamic region. This result shows us that the DNN performs slightly better for the mixed region of the Stribeck curve.

The measured median RMSE for the entire Stribeck curve is 4.4×10^{-4}, for the mixed region of the Stribeck curve defined as $H \in [0, 0.002]$ it is 2.8×10^{-4} and for the hydrodynamic region defined as $H \in [0.002, 0.01]$ it is 4.0×10^{-4}.

In section 3.2.2 we showed the Stribeck curves for a fully dimpled texture case displayed a peculiar trend for increasing radii, with the minimum of the curve shifting leftwards and for cases closer to $D_r = 80\ \mu m$ the curves flattening out more in the mixed region. Importantly, during the discussion about the data treatment we mentioned that, due to this trend, some cases with $D_r \approx 80\ \mu m$ became outliers in data and were consequently removed. The fully dimpled case with $D_r \to 80\ \mu m$ is at the boundaries of the configuration space, with the data treatment it is likely to be the worst performing case of the DNN

prediction when compared to data.

To verify that even in these extreme cases the network is still robust enough to capture the trends of the underlying physics we performed the same analysis as in 3.2.2 for the range of radii and represented the results in figure 42. What we observed was that while the DNN results differ significantly from the solver results in this extreme case the trend of shifting minimum positions and flattening of the curves is still present and well represented. However, for $D_r > 71.1\ \mu m$ we were not able to obtain good enough results due to these cases being outliers in terms of the fit and being the least data supported cases in the entire data set.

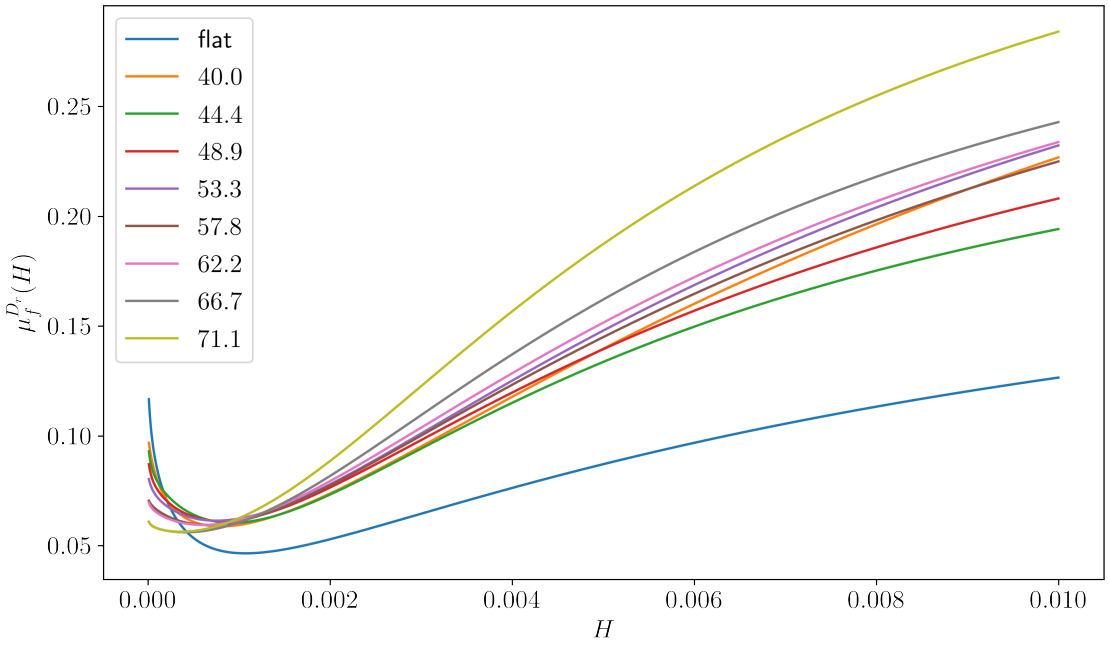


Figure 42: Stribeck curves calculated by the DNN of the fully dimpled case from section 3.2.2 with varying dimple radius in μm as well as the flat untextured case.

Since we have established that our DNN replacement for the FELINE solver works sufficiently well in predicting results it is also important to establish an idea of its performance when compared to the direct solver. We estimated that it would take more than 3000 years to fully compute our entire configuration space using FELINE on the same hardware we used to produce the benchmark in section 3.1.3. In comparison, our DNN running on a single core of a AMD Ryzen™ 5 3580U CPU, a low-power cpu used in consumer laptops, takes only about 2 hours to fully compute the configuration space with approximately 200 million distinct possible cases. This results in approximately a 25 million times speedup when compared to the regular solver. The next section will explore how we can take advantage of this speed up in performance.

4.2 Inverse problem

4.2.1 Algorithm

In this section, we will discuss how we can take advantage of the performance our DNN by solving the texture optimization problem for the particular system we have defined in the forward problem section. The goal is to use our DNN to solve every possible configuration in our configuration space, of size $N \approx 400$ million. This is done by first generating a matrix containing all possible configurations of dimples and radii. We can take advantage of the fact that we can represent each case as a binary number from 0 to 2^{25} combined with a range of allowed dimple radii that should fall within the training range $D_r \in [40, ..., 80]\ \mu m$ to easily represent our configuration space. For example, a randomly generated dimple case with dimple radius $50\ \mu m$ would be represented as follows:

$$0101100001010011110110100\ 50\,. \tag{4.3}$$

The entries of the matrix for pattern 0 for a particular range of possible dimple radii $D_r \in [40, 50, 60]\ \mu m$, have the form:

$$0000000000000000000000000\ 40$$

$$0000000000000000000000000\ 50$$

$$0000000000000000000000000\ 60$$

and for pattern 522,

$$1000001010000000000000000\ 40$$

$$1000001010000000000000000\ 50$$

$$1000001010000000000000000\ 60$$

To obtain the dimple map from one of these entries we can reshape the 25 bit array into a 5×5 matrix in a row-major sense, to comply with the standard set by the solver and DNN. The random pattern in (4.3) would take the following form

$$00011$$

$$10110$$

$$00011$$

$$10000$$

$$11110$$

In the following discussion, we will consider the full configuration space of the dimple mapping and a range of 12 possible dimple radii contained in the range $[40, 80]$ as our input matrix. However, we need to prepare this data for use with our DNN. As discussed in section 4.1.2, we scaled our data so that both our input and output features have the same order of magnitude. By using the same scaling function that was determined during training, we can scale our matrix of input data and allow the DNN to predict the fit functions for all Stribeck curves. One can calculate the total amount of data we have to store: with 12 dimple radii for each possible pattern (2^{25}), we have a total of 402653184 entries with 26 parameters each in our input matrix, the total amount of storage memory required to store this information is 78 GB. To speed up the calculation of Stribeck curves for such a massive matrix of data, we have split the data into 32 equally sized chunks that can be processed in parallel. This way, each process can calculate the Stribeck curves without memory constraints while also providing an overall faster process than processing the full matrix at once. With each of the 32 distinct chunks of Stribeck curves determined, we can collate the information into a single Stribeck curve fit parameter table.

4.2.2 Results

Since we calculated the stibeck curve for each possible configuration, we are now in the condition to perform the texture optimization. The user can specify the optimization criteria: in this case, we will consider the best performing pattern to be the one that generates the Stribeck curve with the lowest value of the coefficient of friction. To determine the minimum of each Stribeck curve, one can take advantage of the fact that the fit is a well-behaved function in the Hersey number interval we have calculated, this condition being guaranteed by the outlier removal data treatment. Since we know the analytical derivative of each fit, determining the minimum of the Stribeck curve is a simple root-finding problem, for which we can employ fast root finding algorithms such as Brent's method [72] [73], which is valid for intervals $[a, b]$ where:

$$f(a) \cdot f(b) < 0\,, \tag{4.4}$$

a condition which the fit applied to our Stribeck curves fulfills.

After determining the minimum position of every single possible case in our configuration space, we can simply perform a linear search through the list to find the smallest possible Stribeck curve minimum. This process can also take advantage of parallelization. In figure 43 we represent the predicted best performing pattern with a dimple radius $D_r = 72.7\ \mu m$, the respective Stribeck curves computed by the DNN and FELINE and the "control" Stribeck curve generated by a flat surface for performance comparison.

As we can see, the predicted Stribeck curve for the pattern appears to vastly outperform the flat untextured case. We can also see that the Stribeck curve calculated by FELINE looks entirely different from the DNN prediction, going against our demonstration in section 4.1.4, where our DNN performed quite remarkably for the validation set, supposedly validating the network performance for the entire configuration space. As we know from the discussion in section 4.1.2, we identified and removed outliers in data through use of pure statistical analysis (the *z-score* test).

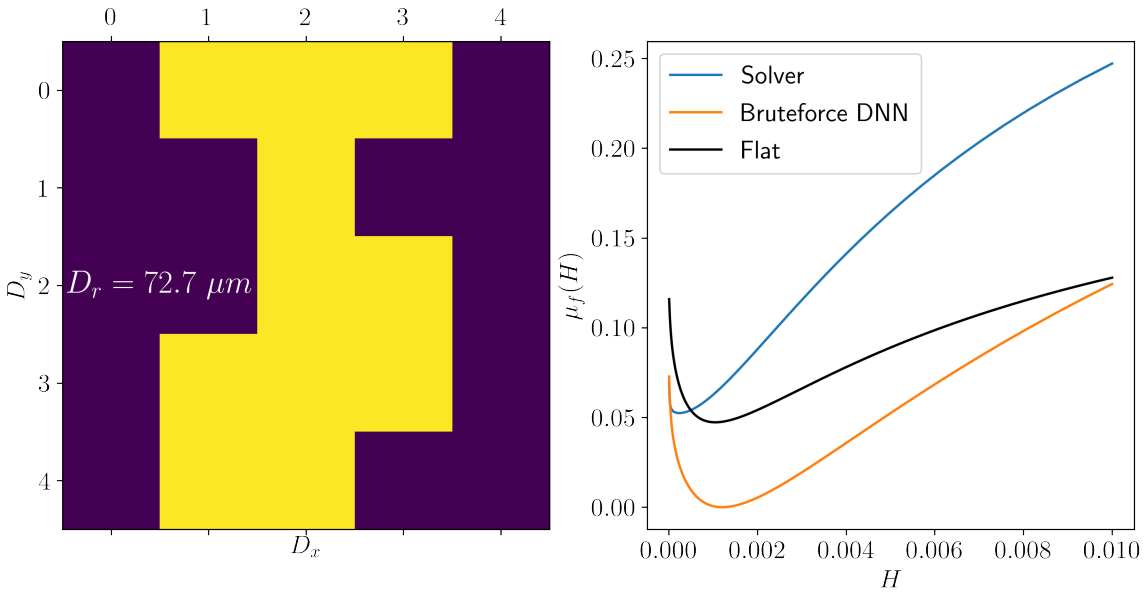


Figure 43: (a) predicted best performing pattern with dimple radius $D_r = 72.7\ \mu m$. (b) corresponding Stribeck curve in orange computed with the DNN; Stribeck curve in blue computed with FELINE; non-textured surface as a control for friction minimization.

The reason why the RMSE distribution in section 4.1.4 doesn't appear to have outliers with such disagreement is entirely due to the fact that the validation set itself is clean from them. It is thus important in such scenarios that we do not forget the physics behind such calculations. While statistical analysis of data in deep neural network training is certainly one key for its success, the real key to obtaining good results in this scenario comes from purely physical arguments. As a matter of fact, removing outlier cases and training the network while afterwards attempting to apply the brute-force search method in a range that certainly includes them is inherently doomed to fail, because we are essentially attempting to extrapolate a behaviour our network was not trained for.

Furthermore we must shed light on the physical reasons why cases with high dimple radius were not adapting well to our network. Since we observed a leftwards moving minimum of the Stribeck curve (see section 3.2.2), this meant that these cases were likely moving towards a dry contact regime for lower

Hersey numbers. Which we statistically identified as the appearance of outliers in fit parameters, due to the curve flattening. In this regime, the Reynolds equation and the Greenwood-Tripp contact model might not apply. To verify this, we decided to carefully look at the pressure profiles that the solver was calculating, by randomly choosing a dimpled case with a dimple radius $D_r = 80\ \mu m$, and representing the resulting maximal pressure and cavitation fraction found at each Hersey number in figure 44.

We found that, due to the increasing dimple radius, meaning also an increase in the dimple to total area ratio, the effective contact area between the texture and the moving surface became very localized. This localization of contact leads to exponentially high pressure points for lower Hersey numbers, as the minimal distance between the surfaces is lower. At pressures of over $20 - 25$ atm we can no longer guarantee the validity of the Reynolds equation, from figure 44 we can see pressures of up to 250 atm. Meaning that these cases cannot be treated in a physically correct way by our solver, and thus have no place in our training set.

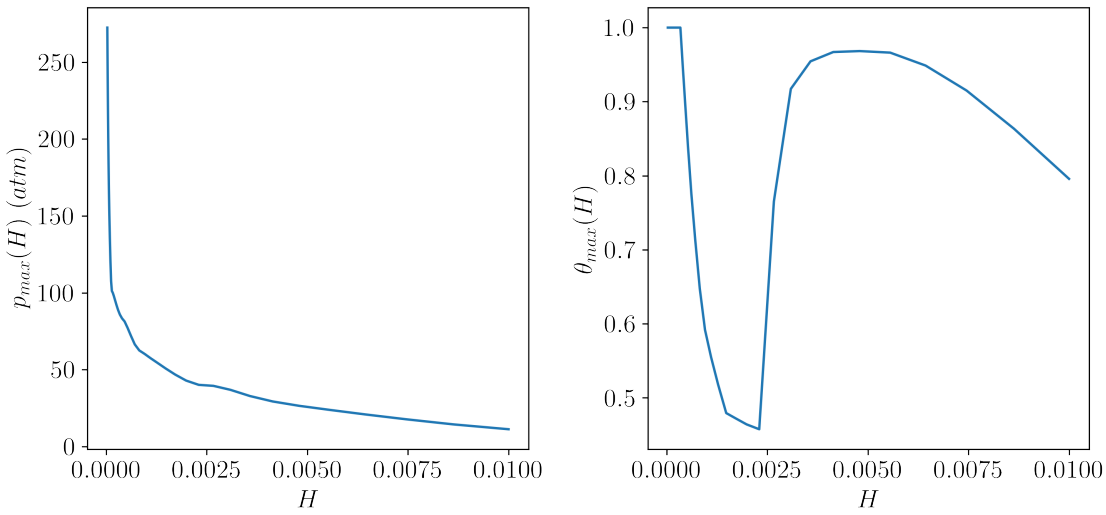


Figure 44: Evolution of maximum pressure and cavitation with respect to Hersey number for a randomly dimpled texture with dimple radius $D_r = 80\ \mu m$.

To verify that our brute-force search method still yields good results, we limited our dimple radius range to 10 values in the interval $D_r \in [40, 60]\ \mu m$. Once again, we generated our input matrix by combining the total pattern configuration space with the 10 possible dimple radii, and then we scaled it according to the same scaling function used in the DNN training. After providing the correctly scaled input matrix to the DNN, we obtained the output matrix containing the corresponding Stribeck curve fits and from these we computed the minimum of each Stribeck curve. After a linear search we identified a family of patterns that yield the best reduction to the Stribeck curve minimum, the most effective of which is represented in figure 45 and has dimple radius $D_r = 60\ \mu m$. We also represented the corresponding Stribeck curves

computed by the DNN and FELINE as well as the flat case for texturing performance comparisons.

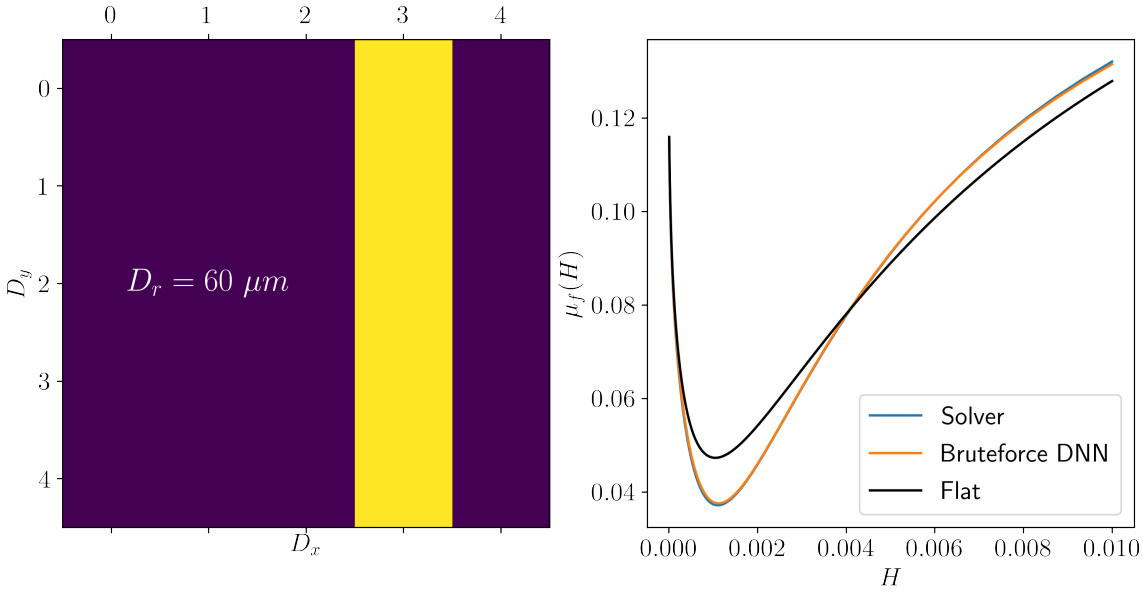


Figure 45: (a) correctly predicted best performing pattern with dimple radius $D_r = 60\ \mu m$. (b) respective Stribeck curve in orange computed with the DNN; Stribeck curve in blue computed with FELINE; non-textured surface as a control for friction minimization.

At its peak of friction reduction, the optimized texture outperforms the flat case by up to 20%. Moreover, our predicted optimal solution is in excellent agreement with the corresponding direct solver calculation. This is perhaps the most critical point of our method, because in order for the brute-force search method to be reliable, its predictions must agree with the solver output.

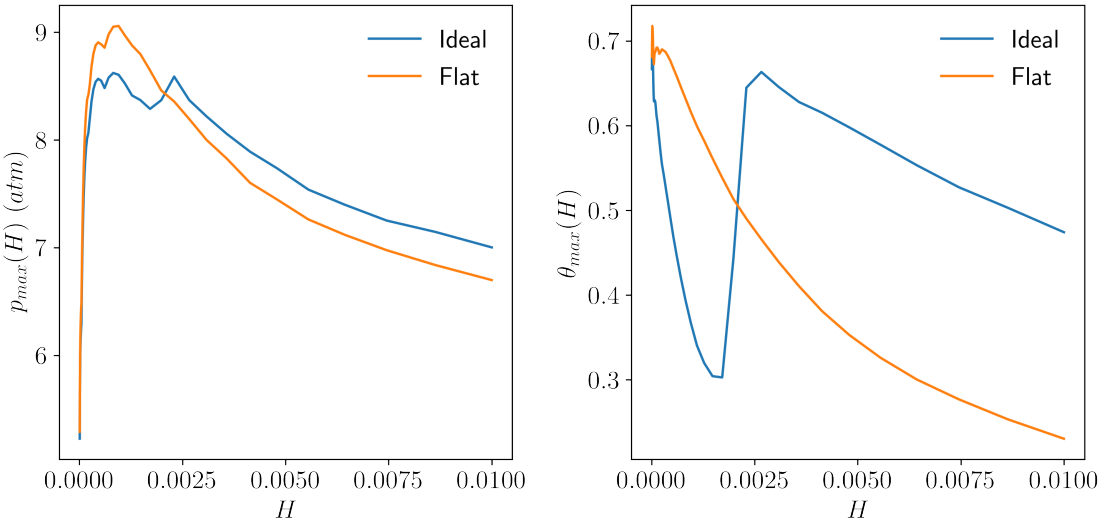


Figure 46: Evolution of maximum pressure and cavitation with respect to Hersey number for our bruteforce search method optimized pattern with $D_r = 80\ \mu m$ and the flat texture.

The optimal pattern is made by a row of dimples normal to the sliding direction after the contact. Similarly to the analysis done in figure 44, one can look at the maximum pressure and maximum cavitation fraction as a function of the Hersey number for our optimized pattern to gain insight into how it achieves friction reduction. In figure 46 we represented this for both the optimized pattern and flat surface, for comparison purposes. We observed that in terms of pressure, our optimized pattern achieves smaller maximum values in the mixed region pressures, while slightly larger values are found in the hydrodynamic region. In terms of cavitation, we have a noticeable increase in maximum cavitation fraction in the hydrodynamic region while observing a steep decrease in the mixed region. This sharp decrease in maximum cavitation fraction occurs within the mixed region, that is in the regime in which our optimized pattern vastly outperforms the flat texture. We can choose a Hersey number within the mixed region of the Stribeck curves in order to visually verify this cavitation region decrease. For $H = 1.1 \times 10^3$ we represented the optimized texture and drew the cavitation boundaries for both the optimized texture and flat texture in figure 47. With this representation of the cavitation boundaries we have shown how the optimized texture determined by the brute-force search method decreases overall friction in the mixed regime.

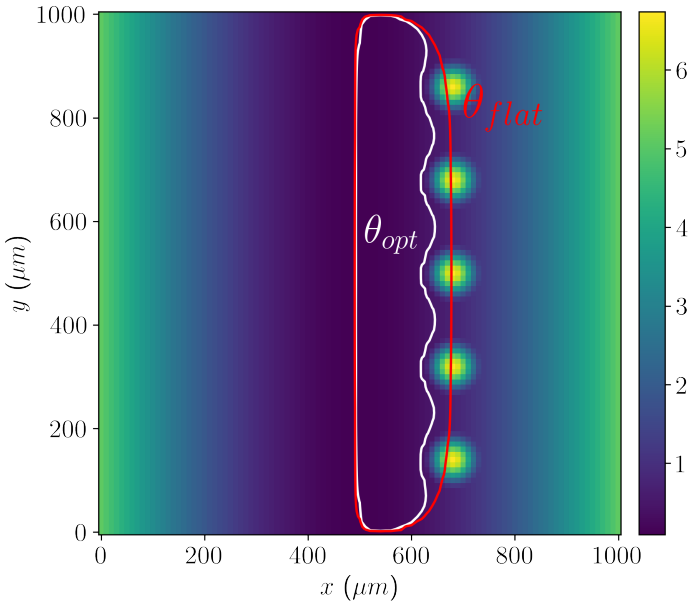


Figure 47: Representation of the optimized texture h, and countours for $\theta = 0.01$ for both the flat textured case and optimized texture case, allowing us to visualize a clear reduction of the cavitation boundary.

The effect of texturing applied after the contact region has been noted previously[38, 39], and thus the best performing pattern might also have been guessed before any calculation, however, since we could not manually check each and every pattern, we could not conclude this for certain. What the brute-force

search method combined with our DNN has allowed is an efficient texture optimization process. Not only it does provide us with the best-performing textures according to our requests, but also ensures that those are the best among all the possible ones within the configuration space.

A powerful feature of this texture optimization method is that it provides a family of optimal/close to optimal patterns. In the maximal friction reducing case, however, it is clear from figure 48 that the optimal patterns follow the same trend of placing a column of dimples after the contact region.

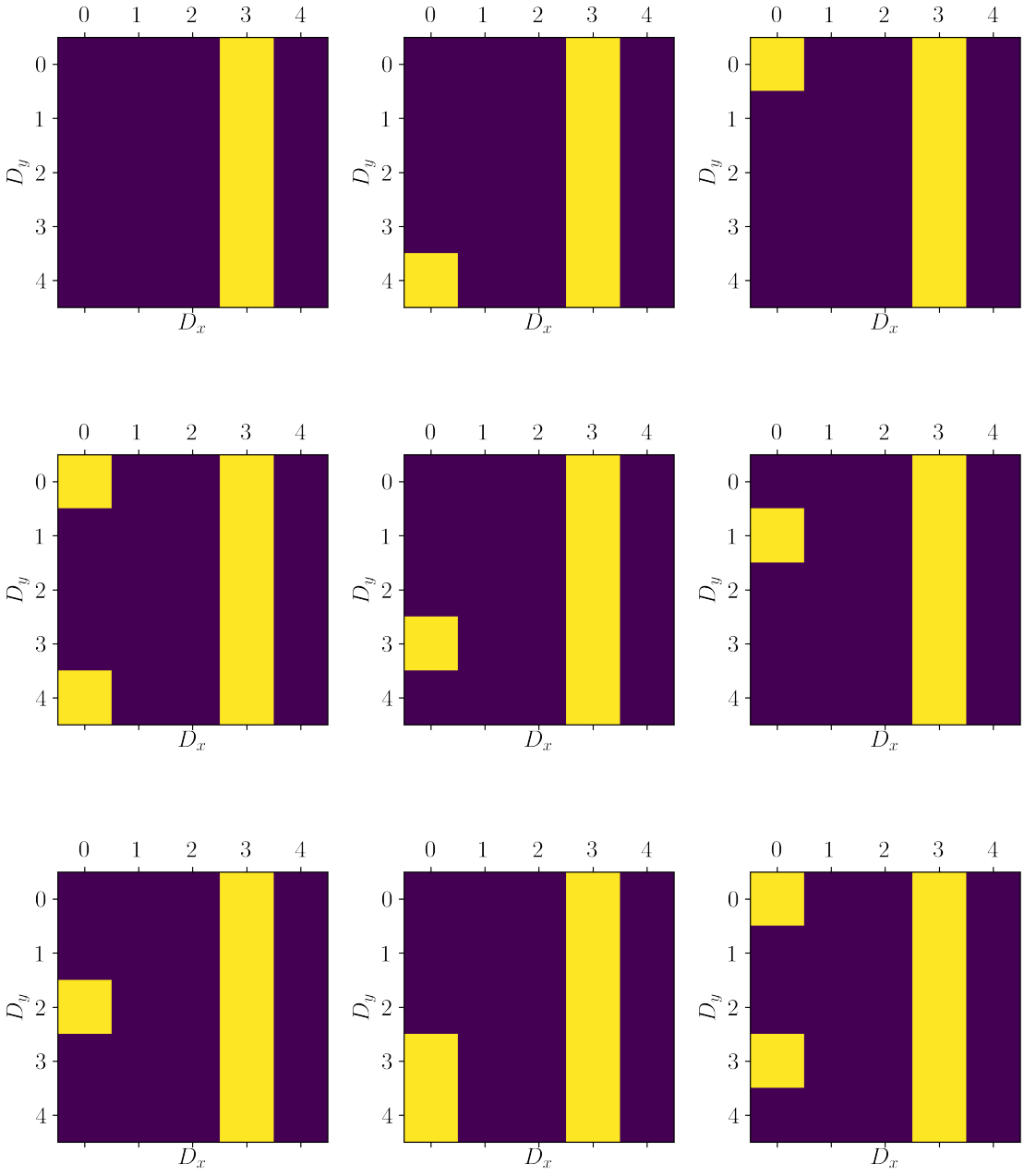


Figure 48: Family of 9 patterns with the best performance excluding the one analyzed in chapter 4. These patterns are closely related to the optimal pattern determined by the brute-force search method.

The dimples at the first column which make up this set of 9 close to optimal patterns hardly have any

effect on friction, when compared to the column of dimples after contact, because they are too far from the contact region and they are placed in a region where cavitation does not occur. The Stribeck curves generated by these patterns generate are extremely close to the one generated by the optimal pattern.

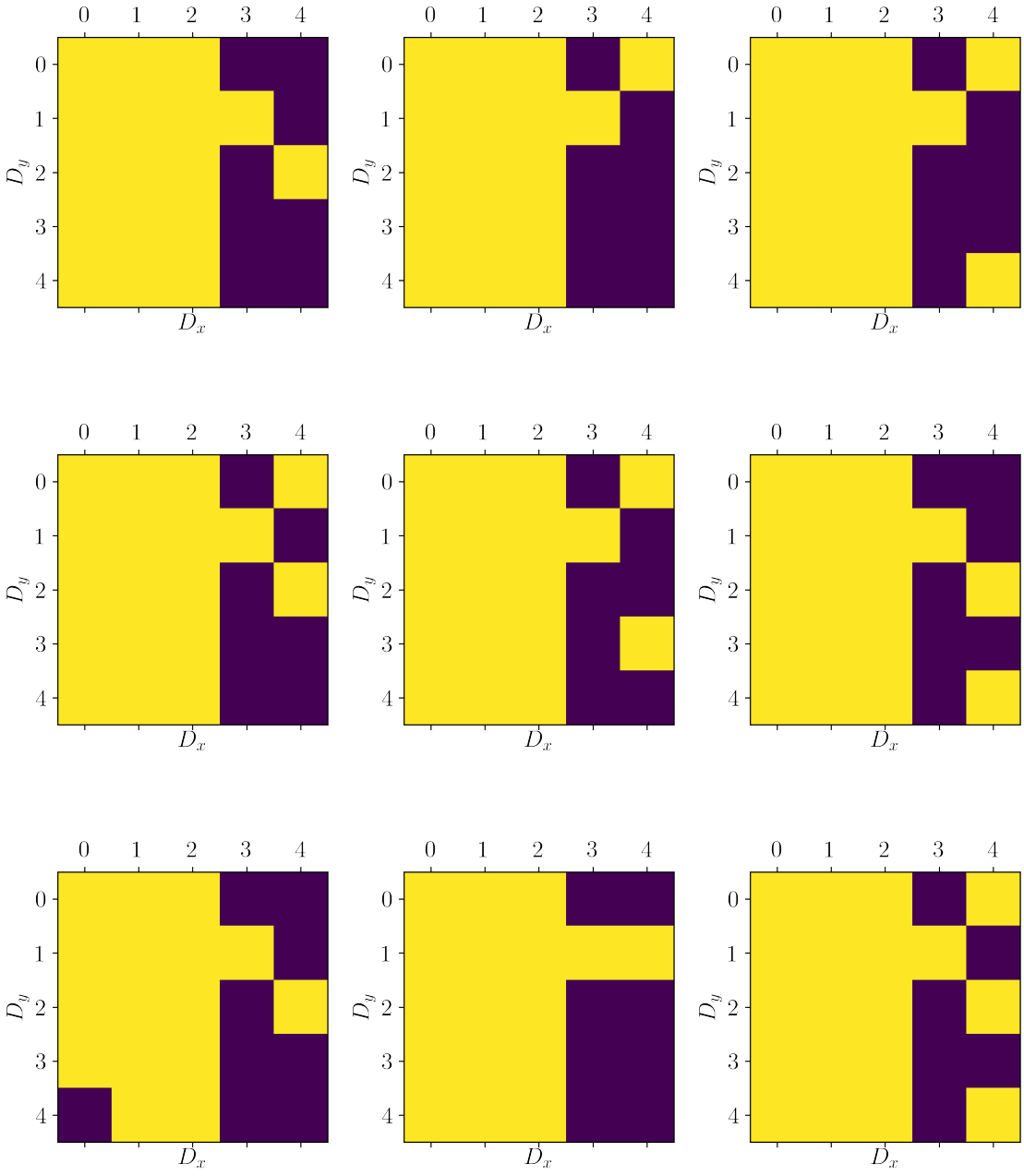


Figure 49: Family of 9 patterns with displayng the worst performance in terms of friction reduction.

It is also worth checking which patterns generate the largest mixed region minimum value of the coefficient of friction, which would be the worst performing case in terms of our previous minimum search. In fact, in some applications an increase of friction is desired, such as in [74, 75]. To do this, we performed a linear search for the highest minima and represented the found patterns in figure 49. Conversely to the best performing case, these patterns have in most cases a dimple radius of was $40\ \mu m$. We also notice

that patterns are related to the best reducing pattern in the sense that they appear to avoid dimples on the column right after contact.

We can now compare the flat case with the best and worst performing textured cases, represented in figure 50, giving us the full idea of how much surface texturing truly influences friction, therefore showing its potential as a friction manipulation tool. This representation of the cases can also be interpreted as a window into how much more room for failure rather than success there is in texturing for friction reduction. In absolute terms, the percent difference between the worst case and flat compared to the best case and flat is considerably higher through the entire curve.

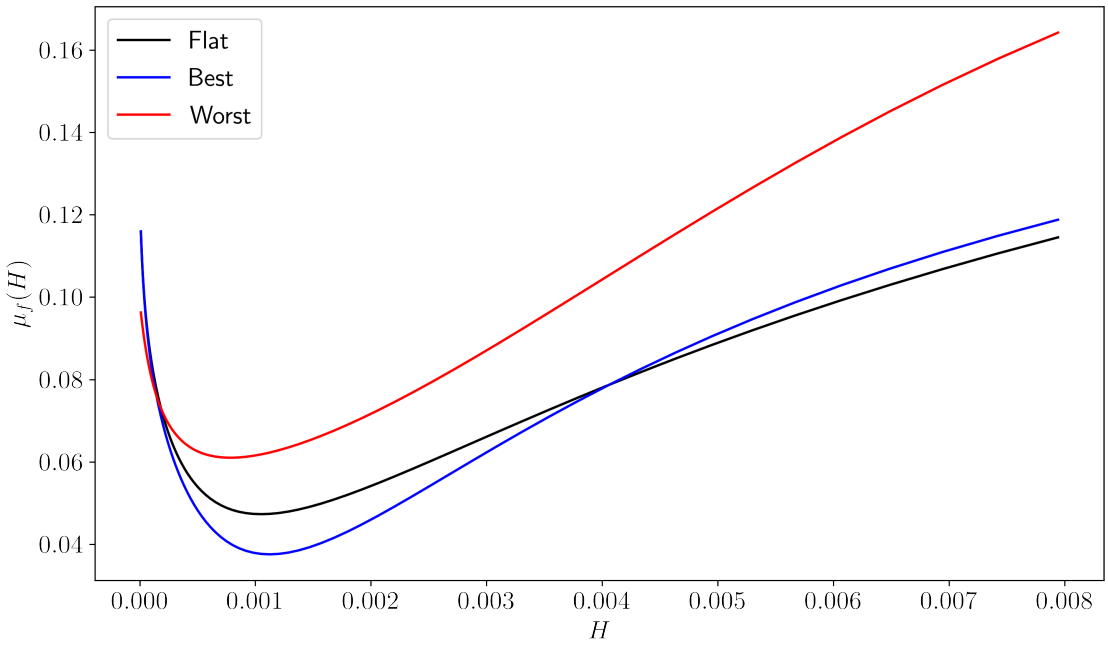


Figure 50: Stribeck curve comparison between the flat, worst and best pattern cases determined by the brute-force search method.

Chapter 5

Conclusions

During the state of the art discussion in chapter 1, we concluded that despite the recent developments in traditional numerical methods utilized to solve the Reynolds equation and allowing for the prediction of friction between moving parts in contacts, a direct optimization of surface parameters for a specific application is still unreliable, being based upon a trial-and-error approach. Due to this fact, we proposed a novel approach to this problem, based upon a machine learning application.

Firstly, we implemented and validated a robust and extremely fast Reynolds equation solver, which uses the finite element method combined with the novel inexact Newton iteration and a mass-conserving cavitation scheme. We showed that it performed successfully both against common cases in literature as well as industrially relevant cases, providing physically accurate results that correctly describe cavitation regions and ensure the non-negativity conditions. Further, we described how we could use the solver combined with a surface contact scheme to consistently predict friction stemming from hydrodynamic and asperity contact, which are calculated using coefficients that can be approximated through experimental data fitting. We verified that our friction calculations correctly took into account the surface geometry. We studied the effect of both dimple radius and depth on the Stribeck curves, concluding that the effect of radius is more apparent and significant, thus making it a promising candidate as an optimization variable.

The overall physical accuracy of our Stribeck curve solver, coupled with its excellent performance in terms of timing, allowed us to calculate the large number of data points requested in the deep neural network (DNN) training, while ensuring the data quality. After generating a total of more than 120 thousand distinct randomly generated grid of dimples we solved these cases and obtained the corresponding Stribeck curves. The DNN that we trained upon this data is capable of predicting Stribeck curves for textured dimples in milliseconds, with a very high degree of accuracy when compared with direct solver results.

Finally, we showed how we could take advantage of our newly trained DNN and its extremely fast

prediction speeds to predict the Stribeck curves for all the possible configurations of our problem, a task that would have been required thousands of years of computation time. This allowed the texture optimization problem to be treated as a a linear search on the dictionary of the predicted Stribeck curves, and we provided two practical examples of this.

The introduction of machine learning into physics is still a recent event in the grand scheme of things, while it has been studied, understood and consistently applied over the years to other fields. We must never forget that the networks can only learn correctly if we teach them correctly, and as such, a close eye on the physics at hand is necessary to ensure successful applications of the technique [76]. Particularly that of the physical limitations of our models and data, as we experienced during the development of our brute-force search method.

As further work, a generalization of the direct Reynolds solver in terms of mesh manipulation and also optimization work could provide us with the possibility of tackling more complex cases while remaining computationally competitive and perhaps even achieving greater levels of speed. this could provide us with the capability to attempt more sophisticated approaches to the texture optimization process such as the training of a deep neural network that would be capable of predicting the texturing pattern in terms of a full, or portioned Stribeck curve, in other words the inverse problem. This would largely eliminate the need for large computations that occur through the brute-force search methods. However, due to a leap in complexity it could possibly require more advanced machine learning strategies such as adversarial neural networks [71] and possibly extremely large data sets. One major problem is the existence of a one-to-many mapping of solutions as there can be a set of textures that generate very similar Stribeck curve solutions. A few cases of this were observed during discussion in chapter 4 of this thesis with closely related textures of the optimal Stribeck curve.

The introduction of machine learning and its successful application and validation in the field of tribology represents the main achievement of the present work. Showing that it is indeed a powerful tool that can be used to help us manipulate friction in novel ways can create an incentive for the pursuit of further exploration of the technique, both in tribology and physics as a whole.

Bibliography

[1] A. LeNail, Nn-svg: Publication-ready neural network architecture schematics, Journal of Open Source Software 4 (33) (2019) 747. doi:10.21105/joss.00747.

[2] F. Mezzadri, E. Galligani, An inexact newton method for solving complementarity problems in hydrodynamic lubrication, Calcolo 55 (1) (feb 2018). doi:10.1007/s10092-018-0244-9.

[3] D. Mason, D. Cobby, M. Horritt, P. Bates, Floodplain friction parameterization in two-dimensional river food models using vegetation heights derived from airborne scanning laser altimetry, Hydrological Processes - HYDROL PROCESS 17 (2003) 1711–1732. doi:10.1002/hyp.1270.

[4] H. K. Johnson, H. Kofoed-Hansen, Influence of bottom friction on sea surface roughness and its impact on shallow water wind wave modeling, Journal of Physical Oceanography 30 (7) (2000) 1743 – 1756. doi:10.1175/1520-0485(2000)030<1743:IOBFOS>2.0.CO;2.

[5] Y. Shao, Physics and Modelling of Wind Erosion, 2nd Edition (Atmospheric and Oceanographic Sciences Library, 37), 2nd Edition, 2008.

[6] J. Prinz, R. de Wijk, L. Huntjens, Load dependency of the coefficient of friction of oral mucosa, Food Hydrocolloids 21 (3) (2007) 402–408. doi:https://doi.org/10.1016/j.foodhyd.2006.05.005.

[7] M. W. ROSEN, N. E. CORNFORD, Fluid friction of fish slimes, Nature 234 (5323) (1971) 49–51. doi:10.1038/234049a0.

[8] M. Hassanalian, H. Abdelmoula, S. Mohammadi, S. Bakhtiyarov, J. Goerlich, U. Javed, Aquatic animal colors and skin temperature: Biology's selection for reducing oceanic dolphin's skin friction drag, Journal of Thermal Biology 84 (2019) 292–310. doi:https://doi.org/10.1016/j.jtherbio.2019.07.018.

[9] C. Greiner, M. Schäfer, Bio-inspired scale-like surface textures and their tribological properties, Bioinspiration amp Biomimetics 10 (2015) 044001. doi:10.1088/1748-3190/10/4/044001.

[10] G. Liu, Z. Yuan, Z. Qiu, S. Feng, Y. Xie, D. Leng, X. Tian, A brief review of bio-inspired surface technology and application toward underwater drag reduction, Ocean Engineering 199 (2020) 106962. doi:https://doi.org/10.1016/j.oceaneng.2020.106962.

[11] T. Endlein, A. Ji, D. Samuel, N. Yao, Z. Wang, W. J. P. Barnes, W. Federle, M. Kappl, Z. Dai, Sticking like sticky tape: tree frogs use friction forces to enhance attachment on overhanging surfaces, Journal of the Royal Society, Interface 10 (80) (Jan 2013). doi:10.1098/rsif.2012.0838.

[12] A. Y. Stark, I. Badge, N. A. Wucinich, T. W. Sullivan, P. H. Niewiarowski, A. Dhinojwala, Surface wettability plays a significant role in gecko adhesion underwater, Proceedings of the National Academy of Sciences of the United States of America 110 (16) (Apr 2013). doi:10.1073/pnas.1219317110.

[13] K. Holmberg, P. Andersson, A. Erdemir, Global energy consumption due to friction in passenger cars, Tribology International 47 (2012) 221–234. doi:https://doi.org/10.1016/j.triboint.2011.11.022.

[14] H. Jost, Tribology — origin and future, Wear 136 (1) (1990) 1–17. doi:https://doi.org/10.1016/0043-1648(90)90068-L.

[15] S. FURUHAMA, Tribology on reciprocating internal combustion engines, JSME international journal 30 (266) (1987) 1189–1199. doi:10.1299/jsme1987.30.1189.

[16] V. Lampaert, F. Al-Bender, J. Swevers, Experimental characterization of dry friction at low velocities on a developed tribometer setup for macroscopic measurements, Tribology Letters 16 (1) (2004) 95–105. doi:10.1023/B:TRIL.0000009719.53083.9e.

[17] B. Jacobson, The stribeck memorial lecture, Tribology International 36 (11) (2003) 781–789, nORDTRIB symposium on Tribology 2002. doi:https://doi.org/10.1016/S0301-679X(03)00094-X.

[18] D. Gropper, L. Wang, T. J. Harvey, Hydrodynamic lubrication of textured surfaces: A review of modeling techniques and key findings, Tribology International 94 (2016) 509–529. doi:https://doi.org/10.1016/j.triboint.2015.10.009.

[19] R. W. Carpick, A. Jackson, W. G. Sawyer, N. Argibay, P. Lee, A. Pachon, R. M. Gresham, The tribology opportunities study: can tribology save a quad?, Tribology & Lubrication Technology 72 (5) (2016) 44.

[20] K. Holmberg, A. Erdemir, Influence of tribology on global energy consumption, costs and emissions, Friction 5 (2017) 263–284. doi:10.1007/s40544-017-0183-5.

[21] I. Etsion, State of the Art in Laser Surface Texturing , Journal of Tribology 127 (1) (2005) 248–253. doi:10.1115/1.1828070.

[22] O. Pinkus, D. F. Wilcock, Strategy for energy conservation through tribology (1 1977).

[23] S. C. Tung, M. L. McMillan, Automotive tribology overview of current advances and challenges for the future, Tribology International 37 (7) (2004) 517–536, the New Trends and Frontiers in Tribology. doi:https://doi.org/10.1016/j.triboint.2004.01.013.

[24] M. Nakada, Trends in engine technology and tribology, Tribology International 27 (1) (1994) 3–8, special Issue Tribology for Automobiles in Japan. doi:https://doi.org/10.1016/0301-679X(94)90056-6.

[25] N. Shaigan, W. Qu, D. G. Ivey, W. Chen, A review of recent progress in coatings, surface modifications and alloy developments for solid oxide fuel cell ferritic stainless steel interconnects, Journal of Power Sources 195 (6) (2010) 1529–1542. doi:https://doi.org/10.1016/j.jpowsour.2009.09.069.

[26] A. E. Somers, P. C. Howlett, D. R. MacFarlane, M. Forsyth, A review of ionic liquid lubricants, Lubricants 1 (1) (2013) 3–21. doi:10.3390/lubricants1010003.

[27] P. Lu, R. Wood, Tribological performance of surface texturing in mechanical applications - a review, Surface Topography: Metrology and Properties 8 (09 2020). doi:10.1088/2051-672X/abb6d0.

[28] X. Lu, M. M. Khonsari, An experimental investigation of dimple effect on the stribeck curve of journal bearings, Tribology Letters 27 (2) (2007) 169. doi:10.1007/s11249-007-9217-x.

[29] P. Stoyanov, R. R. Chromik, Scaling effects on materials tribology: From macro to micro scale, Materials (Basel, Switzerland) 10 (5) (2017) 550. doi:10.3390/ma10050550.

[30] N. Myshkin, M. Petrokovets, A. Kovalev, Tribology of polymers: Adhesion, friction, wear, and mass-transfer, Tribology International 38 (11-12) (2005) 910–921.

[31] O. Reynolds, Iv. on the theory of lubrication and its application to mr. beauchamp tower's experiments, including an experimental determination of the viscosity of olive oil, Philosophical transactions of the Royal Society of London (177) (1886) 157–234.

[32] T. J. R. Hughes, Recent progress in the development and understanding of supg methods with special reference to the compressible euler and navier-stokes equations, International Journal for Numerical Methods in Fluids 7 (11) (1987) 1261–1275. doi:https://doi.org/10.1002/fld.1650071108.

[33] W. Habchi, D. Eyheramendy, P. Vergne, G. Morales-Espejel, A Full-System Approach of the Elastohydrodynamic Line/Point Contact Problem, Journal of Tribology 130 (2), 021501 (03 2008). doi:10.1115/1.2842246.

[34] W. Habchi, D. Eyheramendy, P. Vergne, G. Morales-Espejel, Stabilized fully-coupled finite elements for elastohydrodynamic lubrication problems, Advances in Engineering Software 46 (1) (2012) 4–18, cIVIL-COMP. doi:https://doi.org/10.1016/j.advengsoft.2010.09.010.

[35] V. Bruyere, N. Fillot, G. Morales-Espejel, P. Vergne, Computational fluid dynamics and full elasticity model for sliding line thermal elastohydrodynamic contacts, Tribology International 46 (1) (2012) 3–13. doi:https://doi.org/10.1016/j.triboint.2011.04.013.

[36] J. Schneider, D. Braun, C. Greiner, Laser textured surfaces for mixed lubrication: Influence of aspect ratio, textured area and dimple arrangement, Lubricants 5 (2017) 32. doi:10.3390/lubricants5030032.

[37] C. Greiner, T. Merz, D. Braun, A. Codrignani, F. Magagnato, Optimum dimple diameter for friction reduction with laser surface texturing: The effect of velocity gradient, Surface Topography: Metrology and Properties 3 (2015) 044001. doi:10.1088/2051-672X/3/4/044001.

[38] R. Ausas, P. Ragot, J. Leiva, M. Jai, G. Bayada, G. C. Buscaglia, The Impact of the Cavitation Model in the Analysis of Microtextured Lubricated Journal Bearings, Journal of Tribology 129 (4) (2007) 868–875. doi:10.1115/1.2768088.

[39] R. F. Ausas, M. Jai, G. C. Buscaglia, A mass-conserving algorithm for dynamical lubrication problems with cavitation, Journal of Tribology 131 (3) (jun 2009). doi:10.1115/1.3142903.

[40] F. R. Young, Cavitation, The Journal of the Acoustical Society of America vol. 91 iss. 6 91 (jun 1992). doi:10.1121/1.402820.

[41] M. S. Ahmed, K. Hokkirigawa, K. Kikuchi, J. Higuchi, R. Oba, Sem studies of particles produced by cavitation erosion, JSME International Journal Series B 36 (4) (1993) 517–523. doi:10.1299/jsmeb.36.517.

[42] R. W. Cottle, J.-S. Pang, R. E. Stone, The linear complementarity problem, SIAM, 2009.

[43] H. Elrod, A computer program for cavitation and starvation problems, Journal of Lubrication Technology (1975). doi:10.1115/1.3251669.

[44] T. Woloszynski, P. Podsiadlo, G. W. Stachowiak, Efficient solution to the cavitation problem in hydrodynamic lubrication, Tribology Letters 58 (1) (2015) 18. doi:10.1007/s11249-015-0487-4.

[45] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. 91 (2019) 045002. doi:10.1103/RevModPhys.91.045002.
URL https://link.aps.org/doi/10.1103/RevModPhys.91.045002

[46] M. Biehl, A. Mietzner, Statistical mechanics of unsupervised learning, Europhysics Letters (EPL) 24 (5) (1993) 421–426. doi:10.1209/0295-5075/24/5/017.

[47] J. Carrasquilla, R. G. Melko, Machine learning phases of matter, Nature Physics 13 (5) (2017) 431–434. doi:10.1038/nphys4035.

[48] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Phys. Rev. Lett. 120 (2018) 024102. doi:10.1103/PhysRevLett.120.024102.
URL https://link.aps.org/doi/10.1103/PhysRevLett.120.024102

[49] C. C. Nadell, B. Huang, J. M. Malof, W. J. Padilla, Deep learning for accelerated all-dielectric metasurface design, Opt. Express 27 (20) (2019) 27523–27535.

[50] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics 357 (2018) 125–141. doi:https://doi.org/10.1016/j.jcp.2017.11.039.

[51] B. O. J. Bernard J. Hamrock, Steven R. Schmid, Fundamentals of Fluid Film Lubrication (Mechanical Engineering), 2nd Edition, CRC Press, 2004.
URL libgen.li/file.php?md5=fe907a9b885f68d442af2ee2e40e5bb1

[52] B. Jakobsson, L. Floberg, The finite journal bearing, considering vaporization (das gleitlager von endlicher breite mit verdampfung), 1957.

[53] J. A. Greenwood, J. H. Tripp, The contact of two nominally flat rough surfaces, Proceedings of the Institution of Mechanical Engineers 185 (1) (1970) 625–633. doi:10.1243/PIME_PROC_1970_185_069_02.

[54] R. Gohar, H. Rahnejat, Fundamentals of Tribology, IMPERIAL COLLEGE PRESS, 2008. doi:10.1142/p553.

[55] J. Reddy, An Introduction to the Finite Element Method, 3rd Edition, 3rd Edition, McGraw-Hill Education (ISE Editions), 2005.

[56] D. Griffiths, J. Lorenz, An analysis of the petrov-galerkin finite element method, Computer Methods in Applied Mechanics and Engineering 14 (1978) 39–64. doi:10.1016/0045-7825(78)90012-9.

[57] A. H. Jean Donea, Finite Element Methods for Flow Problems, Wiley, 2003.

[58] E. Alpaydin, Introduction to Machine Learning, Second Edition (Adaptive Computation and Machine Learning), 2nd Edition, Adaptive Computation and Machine Learning, The MIT Press, 2010.

[59] M. M. Noel, A. L, A. Trivedi, P. Dutta, Growing cosine unit: A novel oscillatory activation function that can speedup training and reduce parameters in convolutional neural networks (2021). arXiv:2108.12943.

[60] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85–117.

[61] D. Mishkin, N. Sergievskiy, J. Matas, Systematic evaluation of CNN advances on the imagenet, CoRR abs/1606.02228 (2016).

[62] C. Sanderson, R. Curtin., Armadillo: a template-based c++ library for linear algebra., Journal of Open Source Software, Vol. 1, pp. 26 (2016).

[63] C. Sanderson, R. Curtin., A user-friendly hybrid sparse matrix class in c++., Lecture Notes in Computer Science (LNCS), Vol. 10931, pp. 422-430 (2018).

[64] X. S. Li, An overview of SuperLU: Algorithms, implementation, and user interface, ACM Trans. Math. Softw. 31 (3) (2005) 302–325.

[65] M. H. Tber, An active-set mixed finite element solver for a transient hydrodynamic lubrication problem in the presence of cavitation, ZAMM Journal of applied mathematics and mechanics: Zeitschrift für angewandte Mathematik und Mechanik 98 (06 2017). doi:10.1002/zamm.201700193.

[66] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `https://www.tensorflow.org/`

[67] A. Ge□ron, Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems, O'Reilly Media, Sebastopol, CA, 2017.

[68] A. F. Agarap, Deep learning using rectified linear units (relu), CoRR abs/1803.08375 (2018). arXiv:1803.08375.

[69] Y. Li, Y. Yuan, Convergence analysis of two-layer neural networks with relu activation, CoRR abs/1705.09886 (2017).

[70] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[71] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recognition 77 (2018) 354–377. doi:https://doi.org/10.1016/j.patcog.2017.10.013.

[72] R. P. Brent, An algorithm with guaranteed convergence for finding a zero of a function, Comput. J. 14 (1971) 422–425.

[73] R. P. Brent, Algorithms for minimization without derivatives, Courier Corporation, 2013.

[74] J.-P. Roberge, W. Ruotolo, V. Duchaine, M. Cutkosky, Improving industrial grippers with adhesion-controlled friction, IEEE Robotics and Automation Letters 3 (2) (2018) 1041–1048. doi:10.1109/LRA.2018.2794618.

[75] K. Maghsoudi, R. Jafari, G. Momen, M. Farzaneh, Micro-nanostructured polymer surfaces using injection molding: A review, Materials Today Communications 13 (2017) 126–143. doi:https://doi.org/10.1016/j.mtcomm.2017.09.013.

[76] R. Rai, C. K. Sahu, Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus, IEEE Access 8 (2020) 71050–71073. doi:10.1109/ACCESS.2020.2987324.