NATIONAL TECHNICAL UNIVERSITY OF ATHENS

School of Naval Architecture and Marine Engineering
Division of Ship and Marine Hydrodynamics

Acceleration of BEM using CUDA/GPU programming with application to marine renewable energy extraction

---

DIPLOMA THESIS

February 14, 2019

Author: Panagiotis Koutsogiannakis

Supervisor: K. A. Belibassakis, Professor NTUA

Committee: G. A. Athanassoulis, Professor NTUA
G. K. Politis, Professor NTUA

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

Abstract

Division of Ship and Marine Hydrodynamics

School of Naval Architecture and Marine Engineering

DiplomaThesis

Acceleration of BEM using CUDA/GPU programming with application to marine renewable energy extraction

by Panagiotis Koutsogiannakis

In the present thesis a biomimetic flapping-foil device is studied for the exploitation of marine renewable energy resources. As a first approximation, the lifting body is submerged far from the free surface, neglecting the interaction with the additional boundary. The device is semi-activated, i.e. a pitching motion is enforced and a heaving motion is induced by an alternating lift, produced in the presence of the current. For this purpose a Boundary Element Method (BEM) in its 3-dimensional version is applied. A Morino-type Kutta condition is imposed on the trailing edge and, by linearizing the trailing vortex sheet dynamics, a simplified wake model is used. However, the motion of the body and the initial shape of the wake are not linearized. The present method, after enhancement and further verification, can be applied to the design and control of such biomimetic devices extracting energy from waves and tidal currents nearshore.

The performance of various integration quadratures for the calculation of singular integrals emerging in BEM are also investigated. Simple Gauss-Lobatto and Newton-Cotes quadratures, of arbitrary order, are embedded in an adaptive routine, enabling the treatment of singularities associated with the single/double layer potential induction factors. Different partitioning schemes are examined and the Richardson extrapolation technique is used to accelerate the convergence of the recursive quadrature routine. The developed numerical integration method is able to evaluate efficiently integrals with multiple singularities on N-dimensional hypercubes.

The in-house GPU-accelerated computational code, developed by E.S. Filippas, is reprogrammed, by using object-oriented programming, extending the method to solve the problem of the semi-activated system. The parallelization parameters that affect performance are determined and a mixed precision arithmetic scheme is used to optimize the performance of the algorithm. The concepts of polymorphism and inheritance, incorporated in object-oriented programming, encourage the extension of the solver to treat different problems with similar structure in an elegant manner.

Acknowledgements

The completion of my Diploma thesis would not be possible without the guidance of my supervisor Professor Kostas Belibassakis and the doctoral student Evangelos Filippas. I want to thank them for their useful advice, knowledge and the support they gave me throughout the time I was working on my thesis project.

I also want to thank my friends and family for their support in times of need. Without their motivation I would not be able to be the person I am now and I would not be capable of handling situations when time was at a limit.

Furthermore, I want to express my gratitude to Professor Theodoros Gerostathis, from University of West Attica, for providing material and technical assistance.

Finally, I want to give a huge shoutout to all the anonymous contributors on stackoverflow.com and the NVidia forums for providing solutions to programming conundrums, without whom my work would be still at early stages.

Contents

List of Figures

Chapter 1

Introduction

1.1 Marine energy

Marine currents can have many causes [38]. They can be wind-driven, due to the wind stress on the sea. Along the coast, the movement of water is restricted by the seabed and the coasts and the sea surface rises, creating a pressure gradient that forces the water mass to move producing gradient currents. Moreover, waves can cause mass transport, a second order effect, that can increase the current velocity rapidly with wave height. Tides also contribute to the ocean currents and will be examined more in the next section. Finally, temperature and salinity gradients are related to the great oceanic currents, like the Golf Stream, that are responsible for most deep water movement and the energy transported by them is responsible for the climate conditions in many parts of the world.

The hydrokinetic energy of the currents consists most of the energy found in the sea and is a resource that can be tapped in the upcoming future.

1.1.1 Tidal currents

Tidal currents are a major part of the total currents of the Earth's oceans and in many regions, the effect of tidal currents is intense. For example, in the North Sea the measured velocity of tidal currents can reach $5.0 m/sec$ (Pentland Firth) [40]. A case study shows that the potential annual power production in the UK, from tidal energy, could be as high as $27700 GWh$ [57], which is equal to 1.6% of the total energy consumption in the country for the year 2017 [18].

Tidal currents are a component of the wave motion produced primarily by the relative motion of the Earth and the Sun and Moon and the respective gravitational pull of the oceanic water mass. The primary constituents of tidal currents are the principal lunar and solar semidiurnal cycles, which have periods of 12h 25min and 12h respectively. The tides are more evident when Earth, Moon and Sun are aligned (spring tides) and milder when the Sun, Earth and Moon form a right angle (neap tides), as the effect of the gravitational pull of the Moon partially cancels out with the effect of the Sun.

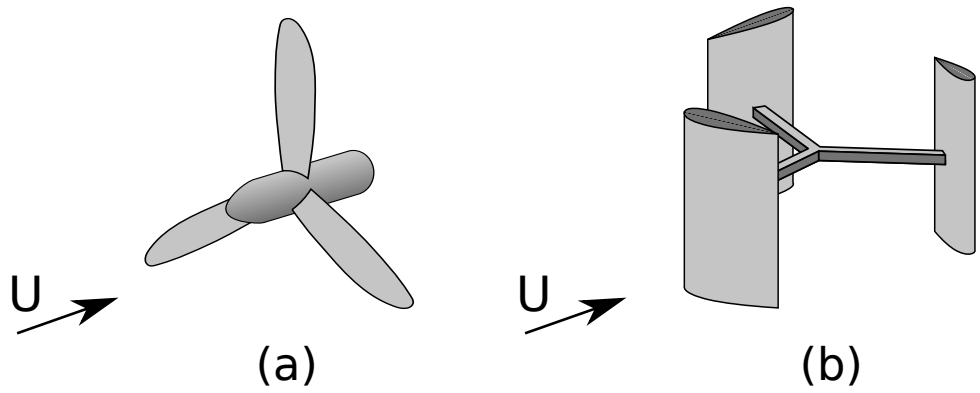

FIGURE 1.1: Horizontal (a) and vertical (b) axis turbine design.

1.1.2 Types of marine energy extraction systems

Various devices have been proposed in the last years to exploit these energy reserves. These systems can be either classical turbines or unconventional systems, such as flutter vanes, the use of piezoelectric materials, vortex induced vibrations, oscillating hydrofoils and sails [29]. In the following paragraphs the turbines and the nature inspired oscillating hydrofoils will be introduced, as they present the most viable of the proposed designs.

Turbines

Turbine systems consist of rotating blades that drive the rotor of a generator. Based on the rotation axis position relative to the flow velocity, the turbines can be described to be of the horizontal or vertical axis kind. The most common design is the horizontal axis turbine that resembles classical wind turbines, with the rotation axis being parallel to the flow velocity vector. In contrast, the vertical configuration has blade elements that rotate perpendicular to the flow.

The popularity of the horizontal axis devices can be attributed to the fact that this configuration exhibits higher efficiency than vertical systems, due to the fact that the underlying mechanism is based on lift forces, while vertical axis turbines are based on drag forces. Khan et al. [29] discusses the different hydrokinetic energy conversion methods and the techno-economic parameters for the choice of the turbine rotor configuration. In figure 1.1 these two principal designs are presented.

Horizontal axis turbine systems suffer from the Betz limit [21], independently published by Betz [7], Lanchester [33] and Joukowsky [26]. The limit prohibits the turbines to harness more than 16/27 of the power available in the water passing through the cross-section of the device. For the horizontal configuration the cross-section area would be the area occupied by the rotating propeller. This limit was corrected by Garrett and Cummins [21] to account for the constraint of the flow in a channel and the ratio of coverage of the channel cross-section with turbines. Garrett and Cummins [20] also note that the coverage of the channel with turbines in practice will be limited by navigational and ecological reasons.

Turbine farms have been proposed to extract the energy of tidal currents (e.g. [36, 14]). Vennell [66] has studied the efficiency of turbine farms placed inside a channel

and reports that with as many as 15 rows of turbines occupying 20% of the current cross-section in a strait a farm will only make 40% of the tidal energy available.

Concerning the horizontal axis turbines, Batten et al. [5] have conducted experiments with a scaled model in a cavitation tunnel. From their results, it can be concluded that the power coefficient of the turbine varies significantly with the current velocity (V_{bg}) and the tip speed ratio (TSR). The maximum power coefficient reported was 0.45 for $TSR = 5.5$ and $V_{bg} = 1.73m/s$.

One of the most recent effort to commercialize this technology is a joint venture between International Power, Morgan Stanley and Atlantis Resources Corporation [3]. The MeyGen Project, as it is called, is proposed to operate for 25 years, off the northern coast of Scotland, and will generate about 400MW of power. The aim of the first phase of the project is to assess their environmental impact and provide a proof of concept for future development.

Despite the benefits of turbine devices for power extraction, the impact of marine turbines on the marine ecology can be significant. Bryden et al. [10] argue that by extracting 10% of the energy flux in a channel, the velocity of the current will be decreased by 3%. Increasing the energy extraction will further reduce the velocity of the flow potentially producing major complications to the local marine environment. Neill et al. [45] studied the effect of turbines on sediment transport on the seabed and developed a numerical model for predictions. Their research concluded that the change in sediment transportation dynamics can affect the topography of the seabed, not only near the turbine or the turbine farm, but as far as 50km away. The sediment that is being transferred from one location to another contains microorganisms that could possibly affect the biodiversity of the remote regions. Furthermore, changes of the sea floor shape and reduction of current velocity can alter the flow patterns of the currents on a large scale, disturbing the living organisms that depend on them and limiting the overall available power for extraction.

The tips of turbine blades can reach high speeds (more than 12-15 m/s [69]), generating noise that will disturb wildlife. Also, the high speed of the blades, together with the significant fraction of the current cross-section area they occupy, raises concerns that blades could hit aquatic animals that use the currents to move. If this happens, not only will the animals be in danger, but also damage will be inflicted to the turbine, adding to the maintenance cost.

Biomimetic approaches

Bio-mimicry is innovation inspired by nature and has become popular in the design disciplines mainly because people are looking for more sustainable ways to manage their activities. Living organisms have evolved for 3.8 billion years to adapt to their environments. Many ideas have been tested in nature and those that work survived till today. So, it would be wise to look in nature for potential solutions to technological problems.

Many commercial bio-mimetic devices have been proposed to harvest energy from waves such as the Pelamis Wave Energy Converter [35] and the Oyster Wave Energy Converter [23], but today none of them succeeded to make the technology scalable and financialy reasonable. This means that more research is needed in order for this technology to mature and be more affordable and reliable. Recently the company

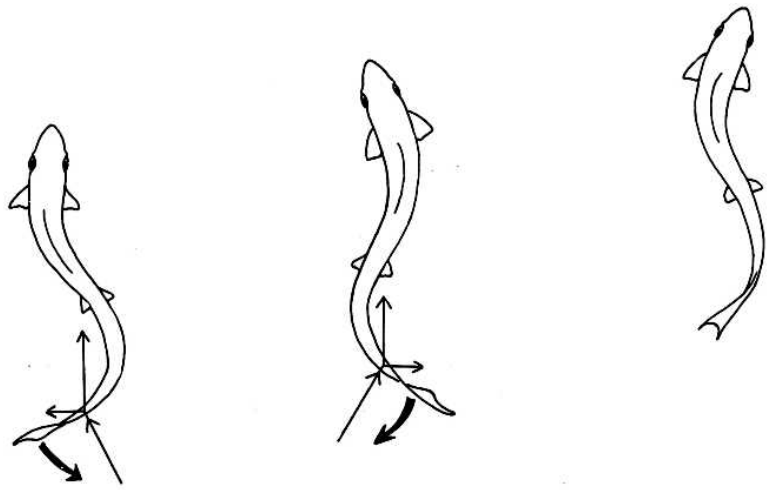

FIGURE 1.2: Fish propulsive motion.

BPS launched a wave generator named bioWAVE [9] that operates like the Oyster Wave Energy Converter.

Devices that can harness the power of currents have also been inspired by nature. Flapping foils, that are being used as propulsors, are modeled after the tail fins of marine animals. In the next section, devices that use oscillating foils for energy extraction will be examined in more detail. These devices operate in the same way as flapping foils, with the only difference being the phase between the heaving and pitching motions.

1.1.3 Oscillating foils for marine energy harvesting

Fishes and other aquatic species, like dolphins and whales, have developed unique ways to propel themselves through water. They move their tail fin in a flapping motion to produce thrust. This motion can be so efficient that led British zoologist Sir James Gray to state in 1936 that the power dolphins can produce through their muscular system does not suffice for the acceleration and velocities observed in nature. This is known in literature as the Gray's paradox.

The flapping oscillation combines a pitching motion with a heaving motion of the foil. The primary mechanism for the generation of thrust or drag by an oscillating foil is the generation of a wake of vortices behind the foil. Koochesfahani [32] visualizes the pattern of vortices on the wake of a flapping foil and shows a jet-like effect. This mechanism is known to aerodynamicists since the early decades of aviation and Von Karman and Burgers [67] explained the effect mathematically. By analyzing both experimentally and computationally the motion patterns of the wings of insects in flight, Birch and Dickinson [8] and Shyy and Liu [61] report that the lift of flapping foils can be enhanced by a secondary mechanism of leading edge vortices (LEV). Zhu and Peng [74] explain, that for high angles of attack, the LEV interacts with the foil creating an area of low pressure on the upper face of the wing. In contrast with the fllaping foil, an oscillating foil would produce a jet in the opposite direction lowering the hydrokinetic energy of the flow.

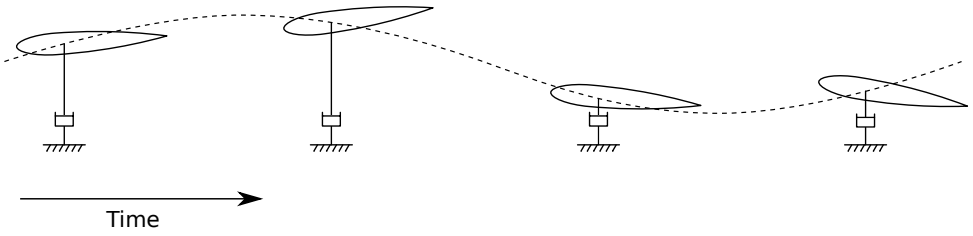

FIGURE 1.3: Motion of an oscillating foil for energy harnessing.

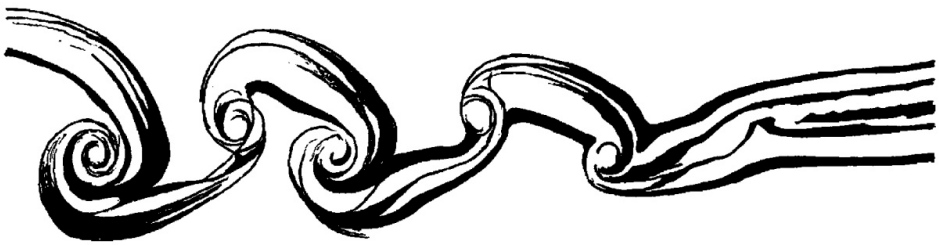

FIGURE 1.4: Wake vortex formation behind a flapping foil [58].

For the design of a ship, this technology can find application in the reduction of resistance due to waves and/or as the main propulsor of the ship (e.g. [60, 64, 52]). Politis and Tsarsitalidis [53] have studied the effect the planform shape has on thrust characteristics of flapping-foil propulsors. They used a BEM simulation to examine fish-like foils.

The idea to use an oscillating foil to extract energy from currents is not new. McKinney and DeLaurier [39] have proposed in 1981 a device to harvest energy by oscillating wings and some first experiments were conducted by DeLaurier and Harris [13] the year after. Parametric studies have been performed (e.g. [31, 25, 73]) to map the performance of the power extraction for different pitching amplitudes and oscillation frequencies. Huxham et al. [25] report a maximum efficiency of 23.8% in their experiments.

Many researchers argue that energy harvesters utilizing oscillating foils can be more efficient than rotary turbines as they are not governed by the Betz limit (e.g. [12, 72]). This is attributed to the fact that oscillating foils can take advantage of the unsteadiness of the flow and vortex dynamics. For example, Kinsey and Dumas [30] have reported a 59.9% efficiency for two oscillating foils in a tandem configuration. For foils in tandem configuration the efficiency can reach asymptotically values as high as 0.9 [27], while the equivalent for value for turbines is 0.66 [72].

Furthermore, Filippas et al. [16] have shown that the energy extraction can increase significantly when operating under waves and they report a maximum power efficiency of 53.5% for a single foil with pitching amplitude of 25^o and motion parameters tuned appropriately to take advantage of the wave. It is significant to note that the ability of the oscillating foil devices to harness energy from both currents and waves gives them an advantage over turbines, where disturbances in the uniformity of the flow can affect negatively the performance of the system.

Kinsey and Dumas [31] have introduced the term "feathering limit" to denote the transition from propulsion to energy extraction. This transition is depicted in figure

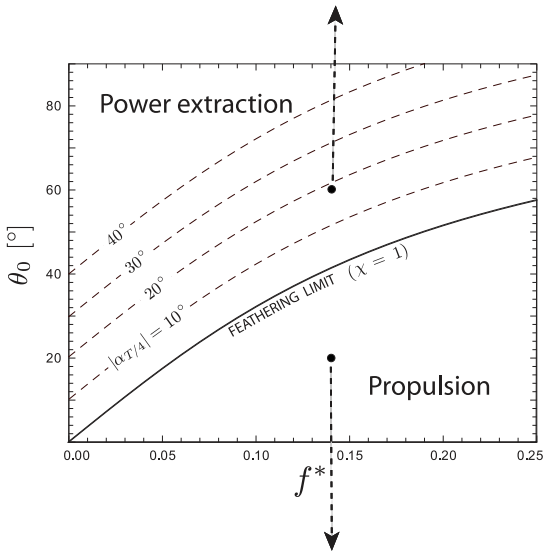

FIGURE 1.5: Feathering limit for power extraction [31]. Here χ is the feathering parameter: $\chi = \frac{\theta_0}{tan^{-1}(H_0\gamma/V_{bg})}$ and f^* is the reduced frequency defined in chapter 2.

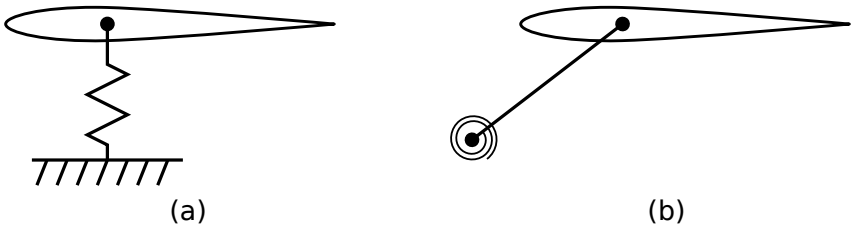

FIGURE 1.6: Two different designs of oscillating foil energy harvesters. Type (a) is a linear actuator, while type (b) is a rotary actuator.

1.5. This means that there is an area of operating conditions where the efficiency of the energy harvesting system is negative and therefore more energy is added to the current rather than extracted and the device operates more like a thruster.

Two designs for oscillating foil energy harvesters have been proposed, with the difference found on the way the foil's motion is constrained. In the first design the foil moves linearly in the heaving direction, while in the other design the foil is constrained to move in a circle by a lever and produces torque on the axis of rotation. In literature the first design is mostly studied, but the latter design may be simpler to realize and find commercial use.

Depending on the degrees of freedom of the heaving and pitching motions of the foil the oscillating foil devices can be categorized as fully-activated, semi-activated and passive. Fully-activated devices enforce both heaving and pitching motions, semi-activated devices enforce only the pitching motion, and passive devices allow the foil to move by itself. Passive devices can be more efficient than other formats, but maintaining correct operation may be difficult. In the present work the semi-activated oscillating foil will be studied. In chapter 2 the mathematical formulation of the problem will be presented. In chapter 5 the derived method will be validated and the device operational characteristics will be evaluated.

The impact of oscillating foil energy converters on the marine environment has not

been studies in depth yet. It can be said, though, that these devices operate at lower frequencies, reducing the noise footprint of the device and the impact on marine wildlife, compared to the classical turbines. Also, collisions with animals will be less frequent and the consequences less severe.

1.2 Numerical methods for lifting flows

The first methods concerning flows with circulation induced by foils begun to emerge in the 1920s. The first approach was to find an analytical solution to the problem, with the most noteworthy results being the thin airfoil theory developed by Munk [44] for his work at NACA. The idea behind the theory is the separation of the foil's shape (the "mean camber line") and the foil's varying thickness. The theory was enriched by Glauert [22] and an unsteady version of it was developed by Theodorsen and Sears [46]. These methods relied on a conformal transformation of the complex plane and are not suitable for 3D problems. A theory for 3D foils was formulated by Scherer to study oscillating foils [59].

Of course, this method has some serious limitations imposed by the assumptions made to achieve an analytical solution. The foil under consideration must be thin as the solution ignores the variable thickness across the chord and the shape of the foil cannot be complex. Furthermore, the solution is linearized and as a consequence the results obtained by the thin airfoil theory are not satisfactory in areas where non-linear effects are important.

Advances in computer technology since the 1970s facilitated the use of numerical methods to solve engineering problems. This technological progress lead to the development of various numerical methods for resolving fluid flows, creating the field of Computational Fluid Dynamics (CFD). Such methods can be categorized as Field Methods, that require discretization of the entire domain of the problem and Boundary Element Methods (BEM), that only require discretization of the boundary of the domain.

Regarding Field Methods, many approaches have been proposed for CFD, such as the Finite Element Method (FEM), Finite Difference Method (FDM) and Finite Volume Method (FVM). As the numerical resolution of computers cannot capture both small scale effects, such as turbulence, and large scale effects the Navier-Stokes equations are averaged for a time period longer than the time scale of the turbulent motion, producing the Reynold's Averaged Navier-Stokes equations (RANS). There are many commercially available CFD solvers that can solve a large variety of problems, but for niche applications they may exhibit poor performance in comparison with dedicated software.

While field methods demand discretization of the problem domain, BEM reduces the problem by one dimension, needing calculations only for the boundaries. This renders the solution of the problem computationally cheaper. On the other hand, the BEM needs knowledge of a fundamental solution of the flow's field equation. Such a solution is not required by other methods that perform calculations on the entire domain of the problem. The use of BEM is widespread for the solution of potential flow problems where a fundamental solution is known.

One of the first publications on the use of BEM for fluid flow problems is that of Hess and Smith [24]. They refer on their method as the Panel Method, as the boundary elements used are flat quadrilaterals. This method is capable of solving steady potential flows around 3D objects of arbitrary geometry. Morino [43] derived a Boundary Element Method for unsteady flows. In his publication he discusses the wake geometry and motion and the constraints that must apply to the trailing edge to eliminate the pressure jump through it.

In chapter 2 the Boundary Element Method will be used for the calculation of hydrodynamic forces acting on the foil.

1.3 General purpose parallel computing on GPUs

In recent years there have been significant technological advancements related to GPUs. GPUs evolved from simple graphics accelerator units to more complex devices, ready to perform trillions of calculations every second. Therefore, GPUs emerge as an interesting alternative to CPUs for solution of scientific problems. This use of GPUs is commonly known as General-Purpose computing on Graphics Processing Units (GPGPU). GPUs can provide more computational power for less electrical power consumption than CPUs, rendering them ideal for intensive computational tasks. For example, an NVidia GTX Titan X has 11 TFLOPS of single precision (FP32) performance, consuming 250 Watts nominal power, while an Intel Core i7-8700K has 61.4 GFLOPS FP32 and consumes 86.2 Watts of power. This means a rating of 44 GFLOPS/Watt for the NVidia GPU and 0.71 GFLOPS/Watt for the Intel CPU.

There are two kinds of GPUs concerning their memory type. The first category includes those GPUs that share a part of the main system memory (RAM) with the CPU and have no dedicated memory. These GPUs, while compact, have performance issues as the main system memory is not optimized for GPUs. The second category is constituted by GPUs that have dedicated memory on their board. GPUs of this kind are discrete from the CPU and communicate with the rest of the system via the PCIe bus standard. The GPUs used for all the work presented in this Thesis are of the latter kind and therefore the term GPU will be used from this point forward to refer to GPUs with dedicated memory.

Modern GPUs have large memory spaces allowing them to store data for complex problems on the device, lowering the PCIe bus communication bandwidth used. This can potentially eliminate communication bottlenecks and speed up the application significantly. The data can be loaded once on the GPU and after the execution of the application the results can be retrieved. In this maner, the CPU stays free of any computational load and can be used for other tasks.

For general purpose computing on GPUs there are two Application Programming Interfaces (APIs) that can be used. The first is called OpenCL (Open Computing Language) that allows the programmer to write the same code for all different processors on a machine. While this ability to standardize software development across devices with very different architectures is appealing to programmers, it can have an impact on performace due to the way it is implemented. The second solution is the CUDA API developed by NVidia for the GPUs of the same brand. CUDA is optimized for NVidia GPUs and as a result offers higher performance. Technologies like OpenCL and CUDA ease the programming of a modern GPU and reduce

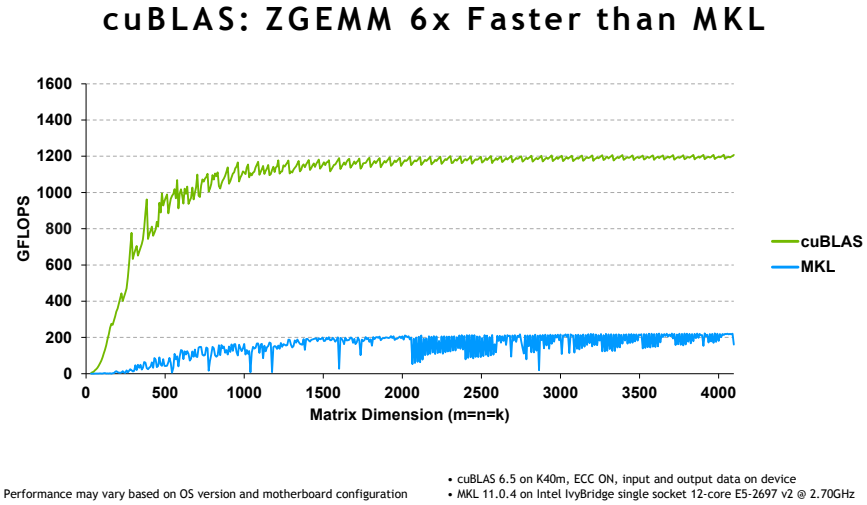

FIGURE 1.7: cuBLAS ZGEMM operation performance comparison with Intel MKL library [48].

the development time. In this manner, the process of writing parallel programs for GPUs get simplified and the code becomes easier to maintain.

1.3.1 GPUs for scientific simulations

Only in recent years GPUs have reached a level of versatility to be able to run general codes. Simulation algorithms should be able to achieve the highest performance possible on a machine, as they tend to be time intensive. So, by offloading the time consuming parts of the algorithm to a co-processor that can quickly process the tasks at hand the computation time gets significantly lower. Most algorithms used to solve physical systems, like FEM, BEM, SPH and more, are massively parallelizable and therefore can efficiently be adapted to the GPU pipeline, which historically has been evolved for graphics rendering.

Of course, a large amount of simulation data must be transferred to the GPU in order to be accessed by the co-processor. Today's GPUs can achieve high transfer rates between the RAM and the GPU memory and the data can be stored in the ever growing GPU memory and cached in the co-processor chip, for great performance benefits.

Many libraries and software exist that utilize GPUs to accelerate the computation process and achieve higher performance standards that scientific research demands. NVidia provides a BLAS implementation for GPUs of the same brand (cuBLAS and cuSPARSE), as well as a system solver package (cuSolver), for free. Other libraries provided by NVidia include cuRAND for random number generation, cuFFT for Fast Fourier Transforms, cuDNN consisting of primitives for deep neural networks and Thrust containing a collection of parallel algorithms and data structures.

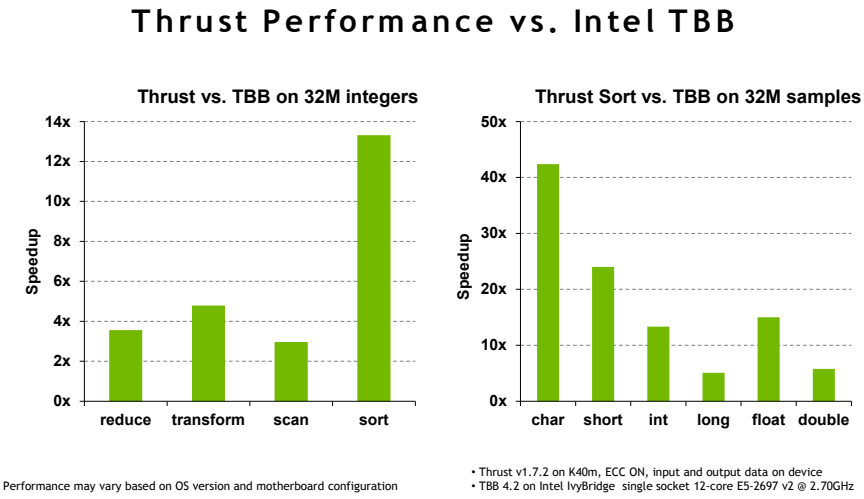

FIGURE 1.8: Thrust library performance comparison with Intel TBB library [48].

Chapter 2

Mathematical Formulation

In this chapter the semi-activated oscillating foil energy harvesting device is studied and the problem is described in a mathematically formal way. The device under study is performing a combined pitching and heaving motion underwater.

The foil geometry is defined by a cross-section profile along the span (s) and the chord (c). The cross-section of the foil and the chord can be constant along the span or variable. For the first part of chapter 5, where simulation results are presented, the cross-section of the foil and the chord are constant along the span and next the effects of a skewed foil and variable chord are studied.

The pitching motion ($\theta(t)$) is enforced on the hydrofoil rotating it around a body-fixed axis on the camber line, at a distance a from the leading edge. In this work a current is considered creating a background velocity field with constant magnitude with direction from the leading edge to the trailing edge of the foil and perpendicular to the axis of rotation. The heaving motion ($h(t)$) is induced by the pitching motion, changing the angle of attack of the flow, translating the foil vertically. In this way a circulation of water is created around the foil producing lift. Depending on the angle of attack the lift will be pointing upwards or downwards. This excitation force makes the foil perform a vertical heaving oscillation.

With right choice of pitching motion parameters, such as frequency, amplitude and profile, a heaving oscillation will occur that maximizes the power output of the device. These parameters should be carefully picked in order to avoid large angles of attack that will stall the foil, decreasing the lift produced.

Oscillating foil energy harvesting devices usually operate near the sea bed. For the purposes of this thesis, it is assumed that the foil operates in an infinite domain, ignoring the ground effect. A study on the ground effect can be found in [75]. A flat sea bed can be considered by using a symmetrical (with respect to the sea bottom) Green function.

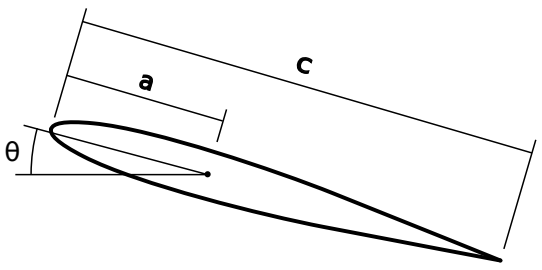

FIGURE 2.1: Schematic of the foil.

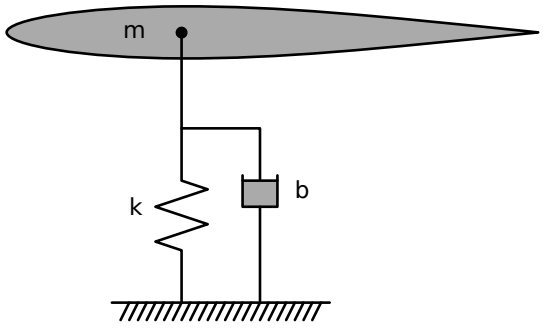

FIGURE 2.2: Device schematic.

In literature a device that produces rotary motion can also be found (e.g. [25]). The evaluation of the present method is done with both models. The mathematical formulation for both types of devices is similar. For the sake of simplicity only the linear actuator model will be described in the next sections.

The mathematical problem is composed by the Boundary Value Problem (BVP) of the lifting flow, and the Initial Value Problem (IVP) describing the motion of the foil along the heaving direction.

2.1 The semi-activated oscillating foil energy harvesting device

The semi-activated oscillating foil energy harvesting device with prescribed pitching motion is modeled by a damped oscillator described by the equation

$$m\ddot{h} + b\dot{h} + kh = F_L(t) \tag{2.1}$$

where m is the mass of the hydrofoil, b is the damping coefficient, modeling the energy generator, k is the spring coefficient, representing the elastic mount and F_L is the lift generated by the oscillating foil. The lift (F_L) depends upon the heave position and the respective time derivative ($\dot{h}$), introducing an implicit nonlinearity to the problem.

The pitch of the foil is given as a periodic function of time. Here a sinusoidal profile will be assumed

$$\theta(t) = \theta_0 sin(\omega_p t) \tag{2.2}$$

where θ_0 is the amplitude of the pitching motion and ω_p is the angular frequency. The frequency $f = \frac{\omega_p}{2\pi}$ of the oscillation is often associated with the non-dimensional quantity called the Reduced Frequency (f^*) so that

$$f^* = \frac{fc}{V_{bg}} \tag{2.3}$$

In order to optimize the operation of the device, the parameter space $\{\theta_0, f^*\}$ can be searched. Further investigation of the pitch profile in time can be conducted in the context of the design of a control system for the device.

The power consumed for the pitching motion is

$$P_{consumed}(t) = M(t)\dot{\theta}(t) \tag{2.4}$$

where $M(t)$ is the moment produced by the pitching motion.

The power produced by the generator is

$$P_{produced}(t) = b\dot{h}^2(t) \tag{2.5}$$

The mean net power generated by the system in one period of the heaving motion is [71]

$$\bar{P} = \frac{1}{T}\int_{t}^{t+T} \left[b\dot{h}^2(t) - M(t)\dot{\theta}(t)\right] dt \tag{2.6}$$

and the efficiency of energy harvesting is

$$\eta = \frac{\bar{P}}{\frac{1}{2}\rho U_\infty^3 Y_p s} \tag{2.7}$$

where Y_p is the difference between the highest and lowest points of the heaving motion, s is the span of the foil, and $\frac{1}{2}\rho U_\infty^3 Y_p s$ is the incoming flow kinematic energy flux. In literature Y_p is defined either as the difference between the highest and lowest points of the pitching axis or as the difference between the highest and lowest positions of any point of the foil. In this work the second approach is adopted, because it allows for better comparison with devices that have very different characteristics, i.e. turbines.

In the case of a device with rotating oscillation the angle of the shaft can be described by the differential equation

$$I\ddot{\phi} + b\dot{\phi} + k\phi = M_H(\phi, \dot{\phi}) \tag{2.8}$$

where M_H is the torque produced on the rotating shaft due to the hydrodynamic forces.

The heaving amplitude is then given by the relation

$$h = Rsin(\phi) \tag{2.9}$$

where R is the length of the lever. The power produced by the device is then

$$P_{produced}(t) = b\dot{\phi}^2(t) \tag{2.10}$$

2.2 Computing the lift of the foil with the Boundary Element Method

The foil geometry disturbs the flow and creates circulation around the wing. This circulation produces a pressure differential between the two sides of the foil and thus a lifting force is generated.

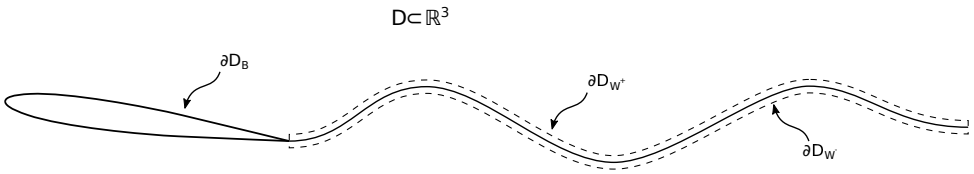

FIGURE 2.3: Domain of the lifting flow problem

In this section it will be demonstrated how the velocity field around the foil can be found using the Boundary Element Method. Next, a relation to link the velocity field with the pressure field will be derived and the lift produced will be evaluated by integrating the pressure on the foil surface.

2.2.1 The lifting flow problem

The hydrofoil is performing an unsteady motion, submerged in an infinite fluid domain $D \subseteq \mathbb{R}^3$ on a background velocity field ($\mathbf{V}_{bg}$). The flow of the fluid is assumed to be incompressible and irrotational. The boundary (∂D) of the domain D is composed by the hydrofoil surface (∂D_B) and the upper and lower side of the wake (∂D_{W^+} and ∂D_{W^-} respectively). The boundary (∂D) is considered smooth everywhere, except for the trailing edge.

The total velocity field ($\mathbf{V}_T$) of the fluid at a point $\mathbf{x} \in D$ is composed of the known background field ($\mathbf{V}_{bg}$) and the unknown disturbance velocity field ($\mathbf{V}$) caused by the hydrofoil.

$$\mathbf{V}_T = \mathbf{V}_{bg} + \mathbf{V} \tag{2.11}$$

The equation of mass conservation states that

$$\nabla \mathbf{V}_T = 0 \tag{2.12}$$

and the equations of motion are

$$\frac{D\mathbf{V}_T}{Dt} = \mathbf{g} - \frac{\nabla p}{\rho} \tag{2.13}$$

where $\frac{D}{Dt}(\cdot)$ denotes the material derivative operator

$$\frac{D}{Dt}(\cdot) = \frac{\partial(\cdot)}{\partial t} + \mathbf{V}_T \nabla(\cdot) \tag{2.14}$$

and $\mathbf{g}$ denotes the acceleration of gravity that will be ignored for the purposes of the present work. Under the assumption of weakly rotational flow, the potential of the disturbance velocity field (Φ) can be defined as

$$\nabla \Phi = \mathbf{V} \tag{2.15}$$

The potential is by definition smooth in the entirety of D. On the boundary ∂D the potential is defined by its trace Φ^* to allow for discontinuities through it. For a point $\mathbf{r}$ on the boundary ∂D the trace of the potential is defined as

$$\Phi^*(\mathbf{r}) = \lim_{\delta \to 0} \Phi(\mathbf{r} + \mathbf{v}\delta) \tag{2.16}$$

where $\mathbf{v}$ is a vector non-tangent to the surface of the boundary and δ is a small parameter. Defining the potential on the boundary in terms of the trace lets potential have different values on the two sides of the wake, while keeping the potential continuous in the entire domain D.

For the potential flow the equation of mass conservation 2.12 becomes

$$\Delta\Phi(\mathbf{x}) = 0, \mathbf{x} \in D \tag{2.17}$$

where

$$\Delta(\cdot) = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2} \tag{2.18}$$

is the Laplace operator. This equation represents the conservation of mass for irrotational, incompressible and inviscid fluids.

By integrating the equations of motion, following a similar procedure as the one found in [4, Ch. 6, p. 382], the approximate Bernoulli's principle for weakly rotational flows can be derived. Using the identity

$$\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) = \mathbf{u}\cdot\nabla\mathbf{u} + \mathbf{u}\times\boldsymbol{\omega} \tag{2.19}$$

where $\boldsymbol{\omega} = \nabla\times\mathbf{u}$ is the rotational velocity of the fluid and assuming no exterior force fields ($\mathbf{g} = 0$), equation 2.13 becomes

$$\frac{\partial\mathbf{V}_T}{\partial t} - \mathbf{V}_T\times(\nabla\times\mathbf{V}_T) + \frac{1}{2}\nabla(\mathbf{V}_T\cdot\mathbf{V}_T) + \frac{\nabla p}{\rho} = 0 \tag{2.20}$$

By substituting $\mathbf{V}_T$ with the right hand side of equation 2.11 and using the irrotational property of $\nabla\Phi$

$$\nabla\times(\nabla\Phi) = 0 \tag{2.21}$$

equation 2.20 becomes

$$\begin{aligned}\frac{\partial(\nabla\Phi + \mathbf{V}_{bg})}{\partial t} + \frac{1}{2}\nabla(\nabla\Phi)^2 + \frac{1}{2}\nabla\mathbf{V}_{bg}^2 + \nabla(\nabla\Phi\cdot\mathbf{V}_{bg}) \\ - \nabla\Phi\times(\nabla\times\mathbf{V}_{bg}) - \mathbf{V}_{bg}\times(\nabla\times\mathbf{V}_{bg}) + \frac{\nabla p}{\rho} = 0\end{aligned} \tag{2.22}$$

By rewriting the equations of motion for the background velocity field only

$$\frac{\partial\mathbf{V}_{bg}}{\partial t} + \frac{1}{2}\nabla\mathbf{V}_{bg}^2 - \mathbf{V}_{bg}\times(\nabla\times\mathbf{V}_{bg}) + \frac{\nabla p_{bg}}{\rho} = 0 \tag{2.23}$$

and subtracting it from equation 2.22 we get

$$\frac{\partial\nabla\Phi}{\partial t} + \frac{1}{2}\nabla(\nabla\Phi)^2 + \nabla(\nabla\Phi\cdot\mathbf{V}_{bg}) - \nabla\Phi\times(\nabla\times\mathbf{V}_{bg}) + \frac{\nabla p}{\rho} = 0 \tag{2.24}$$

Here the rotational term $\nabla\Phi\times(\nabla\times\mathbf{V}_{bg})$ can be ignored under the assumption of weakly rotational flow and p_{bg} is considered zero. After some algebraic manipulation the equation above can be rewritten as

$$\nabla\left[\frac{\partial\Phi}{\partial t} + \frac{1}{2}(\nabla\Phi)^2 + \nabla\Phi\cdot\mathbf{V}_{bg} + \frac{p}{\rho}\right] = 0 \tag{2.25}$$

and by integrating 2.25 along a curve in D

$$\frac{\partial \Phi}{\partial t} + \frac{1}{2}(\nabla \Phi)^2 + \nabla \Phi \cdot \mathbf{V}_{bg} + \frac{p}{\rho} = f(t) \tag{2.26}$$

where f is a function of time only due to equation 2.25. This term can be eliminated from the right hand side of 2.26 by redefining the potential as

$$\tilde{\Phi} = \Phi + \int_{t_0}^{t} f(\tau) d\tau \tag{2.27}$$

giving the final form of the Bernoulli's principle for weakly irrotational flows

$$\frac{\partial \Phi}{\partial t} + \frac{1}{2}(\nabla \Phi)^2 + \nabla \Phi \cdot \mathbf{V}_{bg} + \frac{p}{\rho} = 0 \tag{2.28}$$

Boundary Conditions

On the foil, the fluid cannot enter the body, giving the no entrance boundary condition on the body boundary (∂D_B)

$$\nabla_{\mathbf{n}_B} \Phi = \big(\mathbf{V}(\mathbf{x}) - \mathbf{V}_{bg}\big) \cdot \mathbf{n}_B, \mathbf{x} \in \partial D_B \tag{2.29}$$

where $\mathbf{n}_B$ is the unit normal vector to the boundary surface[1]and $\mathbf{V}(\mathbf{x}) = \mathbf{V}_0 + \omega \times \mathbf{R}(\mathbf{x})$, with $\mathbf{V}_0$ denoting the velocity of the center of rotation, ω the angular velocity of the body rotating around the pitching axis and $\mathbf{R}(\mathbf{x})$ the relative pozition of point $\mathbf{x}$ from the pitching axis.

At infinite distance from the hydrofoil the disturbance field vanishes and only the background field is left

$$\lim_{\|\mathbf{x}\| \to \infty} \nabla \Phi = 0 \tag{2.30}$$

The wake of the hydrofoil is modeled as a free shear layer surface (∂D_W). The wake as a free surface cannot carry loading, giving the constraint

$$p^+ = p^- \tag{2.31}$$

where the plus (+) sign denotes the upper side of the layer and the minus (-) sign the lower. This constraint is termed the dynamic boundary condition. In addition the two sides of the free shear layer are not allowed to separate, giving the kinematic boundary condition. This condition states that the velocity vector components perpendicular to the boundary surface on both sides of the free shear layer are the same

$$\mathbf{n}_W \nabla \Phi^+ = \mathbf{n}_W \nabla \Phi^- \Leftrightarrow \mathbf{n}_W (\nabla \Phi^+ - \nabla \Phi^-) = 0 \tag{2.32}$$

where $\mathbf{n}_W$ is the normal to the wake vector[1].

[1]The normal to the boundary surface vectors are considered pointing towards the interior of D.

Using the approximate Bernoulli's theorem for unsteady weakly rotational flow 2.28 we can write for the pressure difference $p^+ - p^-$ that

$$\frac{p^+ - p^-}{\rho} = -\frac{\partial(\phi^+ - \phi^-)}{\partial t} - \frac{1}{2}(\nabla\Phi^+ + \nabla\Phi^-)(\nabla\Phi^+ - \nabla\Phi^-) - (\nabla\Phi^+ - \nabla\Phi^-)\mathbf{V}_{bg} \tag{2.33}$$

Setting

$$\mu = \phi^+ - \phi^- \tag{2.34}$$

as the potential jump on the wake and satisfying both 2.31 and 2.32 it can be shown that on the wake

$$\frac{D\mu}{Dt} = 0 \tag{2.35}$$

Due to the dynamic boundary condition the wake surface should move with the flow. In the present work a simplified wake model is used [54]. The wake geometry is specified by the movement of the trailing edge (TE) of the hydrofoil and is moving according to the background velocity $\mathbf{V}_{bg}$.

As the foil moves the wake is expanded adding one more unknown quantity to the problem. That is the potential jump at the trailing edge ($\mu(TE)$). Therefore, for the closure of the mathematical problem one more condition is needed. In order to eliminate infinite values of the spatial rate of change of the physical quantities the pressure on the two sides of the trailing edge must be the same. This condition is known as the Pressure-Kutta condition.

In the present work a Morino-type Kutta condition will be used, demanding continuity of the velocity at the TE. Mathematically this is written as

$$\mu(TE) = \Phi^+(TE) - \Phi^-(TE) \tag{2.36}$$

The Morino-type Kutta condition is derived from the more general pressure-type Kutta condition, demanding pressure continuity through the TE, under the assumption of linearized quasi-steady flow.

2.2.2 The Boundary Integral Equation

Given the Green's second formula and denoting with $\frac{\partial G}{\partial \mathbf{n}}$ the derivative of the function G in the direction of the normal to the boundary unit vector $\mathbf{n}$, the following equation stands

$$\int_D (G(\mathbf{x};\mathbf{y})\Delta\Phi(\mathbf{y}) - \Phi(\mathbf{y})\Delta G(\mathbf{x};\mathbf{y}))dV(\mathbf{y}) = \int_{\partial D} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi(\mathbf{y})}{\partial\mathbf{n}} - \Phi(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y}) \tag{2.37}$$

where $G(\mathbf{x};\mathbf{y})$ is the fundamental solution (Green function) of the Laplace operator, at a point $\mathbf{y}$, satisfying the condition

$$\Delta G(\mathbf{x};\mathbf{y}) = -\delta(\mathbf{x} - \mathbf{y}) \tag{2.38}$$

where δ denotes the Dirac delta function with the property

$$\int_D u(\mathbf{y})\delta(\mathbf{x}-\mathbf{y})dV(\mathbf{y}) = u(\mathbf{x}) \tag{2.39}$$

This is the sampling property of the Dirac delta function.

As we seek a solution of the Laplace problem 2.17 and using equation 2.38, equation 2.37 becomes

$$\Phi(\mathbf{x}) = \int_{\partial D} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi(\mathbf{y})}{\partial\mathbf{n}} - \Phi(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y}) \tag{2.40}$$

The above results allows us to represent the solution of the BVP in the entire domain D as a function of the values on the boundary. The first term of the integral equation 2.40 is called the single layer potential and the second double layer potential.

For the 3-dimensional domain $D \subset \mathbb{R}^3$ the fundamental solution G is

$$G(\mathbf{x};\mathbf{y}) = -\frac{1}{4\pi\left\|\mathbf{x}-\mathbf{y}\right\|} \tag{2.41}$$

and the partial derivative in the direction of the normal to the boundary vector $\frac{\partial G}{\partial\mathbf{n}}$ is

$$\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}} = \mathbf{n}\cdot\nabla G(\mathbf{x};\mathbf{y}) = \frac{\mathbf{n}\cdot(\mathbf{x}-\mathbf{y})}{4\pi\left\|\mathbf{x}-\mathbf{y}\right\|^3} \tag{2.42}$$

By splitting the boundary into the surface of the foil (∂D_B) and the upper and lower side of the wake (∂D_{W^+} and ∂D_{W^-} respectively) so that $\partial D = \partial D_B \cup \partial D_{W^+} \cup \partial D_{W^-}$, equation 2.40 becomes

$$\begin{aligned}\Phi_B(\mathbf{x}) = &\int_{\partial D_B} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi_B(\mathbf{y})}{\partial\mathbf{n}} - \Phi_B(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y}) \\ &+ \int_{\partial D_{W^+}} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi_{W^+}(\mathbf{y})}{\partial\mathbf{n}} - \Phi_{W^+}(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y}) \\ &+ \int_{\partial D_{W^-}} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi_{W^-}(\mathbf{y})}{\partial\mathbf{n}} - \Phi_{W^-}(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y})\end{aligned} \tag{2.43}$$

Applying the same process on a point near the boundary ∂D_B and taking the limit as the point $\mathbf{x}$ gets closer to the boundary

$$\begin{aligned}\frac{1}{2}\Phi_B(\mathbf{x}) = &\int_{\partial D_B} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi_B(\mathbf{y})}{\partial\mathbf{n}} - \Phi_B(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y}) \\ &+ \int_{\partial D_{W^+}} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi_{W^+}(\mathbf{y})}{\partial\mathbf{n}} - \Phi_{W^+}(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y}) \\ &+ \int_{\partial D_{W^-}} (G(\mathbf{x};\mathbf{y})\frac{\partial\Phi_{W^-}(\mathbf{y})}{\partial\mathbf{n}} - \Phi_{W^-}(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial\mathbf{n}})dS(\mathbf{y})\end{aligned} \tag{2.44}$$

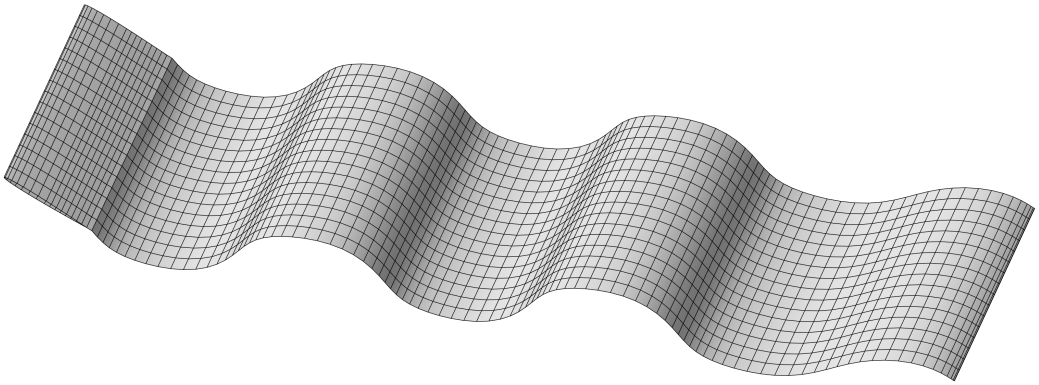

FIGURE 2.4: Discetization of the boundary.

The 1/2 term in equation 2.44 occurs when the point of interest approaches the boundary surface, as the support of the Dirac delta function becomes a hemisphere in the interior of D.

From 2.44 and using the boundary conditions 2.29 and 2.32 the boundary integral equation to calculate the potential on the surface of the foil is derived.

$$\begin{aligned}\frac{1}{2}\Phi_B(\mathbf{x}) + \int\limits_{\partial D_B} \Phi_B(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial \mathbf{n}} dS(\mathbf{y}) = &\int\limits_{\partial D_B} (\mathbf{V}(\mathbf{y}) - \mathbf{V}_{bg})\cdot \mathbf{n}_B G(\mathbf{x};\mathbf{y}) dS(\mathbf{y}) \\ &- \int\limits_{\partial D_W} \mu_W(\mathbf{y})\frac{\partial G(\mathbf{x};\mathbf{y})}{\partial \mathbf{n}} dS(\mathbf{y})\end{aligned} \tag{2.45}$$

where $\mu_W = \Phi_{W^+} - \Phi_{W^-}$ is the potential jump on the wake.

2.2.3 Discretization of the BIE

Initially the geometry of the boundaries ∂D_B and ∂D_W is approximated with the aid of bilinear elements (Ω_i) (see appendix A). On each bilinear element the potential is considered constant. So, for the N_B panels of the foil and the N_W panels of the wake, and with appropriate numbering of them, the potential and its normal derivative are:

$$\Phi_B(\mathbf{x}_i) = \Phi_{Bi}, \text{ at panel i}, i = 1(1)N_B \tag{2.46}$$

$$\mu_W(\mathbf{x}_i) = \mu_{Wi}, \text{ at panel i}, i = 1(1)N_W \tag{2.47}$$

$$\frac{\partial \Phi_B(\mathbf{x}_i)}{\partial \mathbf{n}} = \frac{\partial \Phi_{Bi}}{\partial \mathbf{n}} = \left[(\mathbf{V}_B - \mathbf{V}_g)\cdot \mathbf{n}_B\right]_i = g_i, \text{ at panel i}, i = 1(1)N_B \tag{2.48}$$

Using the collocation method, equation (2.45) is required to be satisfied for a finite number of collocation points $\{P_i : i = 1(1)N_B\}$ on ∂D_B. Here the collocation points are taken to be the centers of the bilinear elements. Then, the discretized BIE is written

$$\sum_{j=1}^{N_B}\left(\frac{\delta_{ij}}{2} + B_{ij}\right)\Phi_{Bj} = \sum_{j=1}^{N_B}(A_{ij})g_j + \sum_{j=1}^{N_F}(-B_{ij})\mu_{Wj} \tag{2.49}$$

for all $\mathbf{P}_i$, where δ_{ij} is Kronecker's delta and (A_{ij}) and (B_{ij}) are the induced potential factors at collocation point $\mathbf{P}_i$ due to a unit source and doublet distribution, at panel j, respectively.

The induction factors are defined as

$$A_{ij} = \int_{\Omega_j} G(\mathbf{P}_i; \mathbf{y}_j) dS(\mathbf{y}_j) \tag{2.50}$$

$$B_{ij} = \int_{\Omega_j} \frac{\partial G(\mathbf{P}_i; \mathbf{y}_j)}{\partial \mathbf{n}} dS(\mathbf{y}_j) \tag{2.51}$$

The integrands of the diagonal elements (A_{ii}) and (B_{ii}) have no finite value at point P_i when it coincides with η. For the evaluation of these elements an adaptive method will be used together with a fast quadrature (see chapter 3 for more details on the computation of the above integrals).

The strip of wake elements adjoin to the trailing edge is called the Kutta strip. The geometry of these elements is constant on the body fixed reference frame. The role of this special strip is to direct the dipole intensity (μ) according to the flow characteristics near the trailing edge.

The Morino condition must be satisfied on the collocation points adjoin to the trailing edge. Denoting these collocation points with the subscript k^+ for the upper side and k^- for the lower side, the Morino condition can be written as

$$\mu_{Wk} = \Phi_{Bk^+} - \Phi_{Bk^-} \tag{2.52}$$

Equations (2.49) and (2.52) can be written together as

$$\sum_{j=1}^{N_B} \left(\frac{\delta_{ij}}{2} + \sum_{k^+} \delta_{k^+,j} B_{k^+,j} - \sum_{k^-} \delta_{k^-,j} B_{k^-,j} + B_{ij} \right) \Phi_{Bj} = \sum_{j=1}^{N_B} (A_{ij}) g_j + \sum_{j=1(1)N_F} (-B_{ij}) \mu_{Wj} \tag{2.53}$$

By solving the above system of equations the values of the potential on P_i are found. The potential anywhere in the domain D can be calculated by discretizing the equation 2.43, as follows

$$\Phi_i = \sum_{j=1}^{N_B} (A_{ij}) g_j + \sum_{j=1}^{N_B} (-B_{ij}) \Phi_{Bj} + \sum_{j=1}^{N_F} (-B_{ij}) \mu_{Wj} \tag{2.54}$$

2.2.4 Computing velocities on boundaries

Assuming that the potential on the boundary is known, and by using equation (2.15), the disturbance velocity can be computed. As the potential gradient is calculated on a curvilinear coordinate system on the boundary surface the following transformation will be used to calculate potential gradient with respect to the global coordinate system

$$\frac{\partial \Phi}{\partial x_i} = \frac{\partial \Phi}{\partial e_j} \frac{\partial e_j}{\partial x_i} \tag{2.55}$$

where repeating indices indicate summation, and e_j is the curvilinear coordinate system on the surface at the point where the velocity will be calculated. e_1 and e_2 are

the directions tangent to the boundary surface and e_3 is the direction normal to the boundary and pointing outwards from the body. The local curvilinear coordinate system is body fixed, meaning that it moves along with the boundary. Also, for the BIE described above, the potential is expressed in terms of it's trace. For the trace potential differential, the following is true

$$d\Phi^* = \nabla\Phi \cdot \left(\frac{\partial \mathbf{x}}{\partial e_1} de_1 + \frac{\partial \mathbf{x}}{\partial e_2} de_2 + \frac{\partial \mathbf{x}}{\partial t} dt \right) + \frac{\partial \Phi}{\partial t} dt \tag{2.56}$$

For $e_1, e_2 = const$, (2.56) can be written as

$$d\Phi^* = \nabla\Phi \cdot \frac{\partial \mathbf{x}}{\partial t} dt + \frac{\partial \Phi}{\partial t} dt \tag{2.57}$$

or

$$\frac{d\Phi^*}{dt} = \nabla\Phi \cdot \frac{\partial \mathbf{x}}{\partial t} + \frac{\partial \Phi}{\partial t} = \nabla\Phi \cdot V_B + \frac{\partial \Phi}{\partial t} \tag{2.58}$$

For $t = const$, (2.56) can be written as

$$d\Phi^* = \nabla\Phi \cdot \left(\frac{\partial \mathbf{x}}{\partial e_1} de_1 + \frac{\partial \mathbf{x}}{\partial e_2} de_2 \right) = \nabla\Phi \cdot d\mathbf{s} \tag{2.59}$$

for a displacement vector $d\mathbf{s}$ tangent to the surface. By definition the differential of the trace of potential on the boundary is

$$d\Phi^* = \nabla\Phi^* \cdot d\mathbf{s} \tag{2.60}$$

By subtracting (2.60) from (2.59) we get

$$(\nabla\Phi - \nabla\Phi^*) \cdot d\mathbf{s} = 0 \tag{2.61}$$

meaning that $(\nabla\Phi - \nabla\Phi^*)$ is a vector normal to the surface. Therefore, the gradient of the potential can be written as

$$\nabla\Phi = \nabla\Phi^* + (\mathbf{n}\nabla\Phi) \cdot \mathbf{n} \tag{2.62}$$

The velocities are calculated using a second order central finite differences scheme as

$$u_P^1 = \frac{1}{2} \left(\frac{\Phi_E - \Phi_P}{d(P,E)} + \frac{\Phi_P - \Phi_W}{d(W,P)} \right) \tag{2.63}$$

$$u_P^2 = \frac{1}{2} \left(\frac{\Phi_N - \Phi_P}{d(P,N)} + \frac{\Phi_P - \Phi_S}{d(S,P)} \right) \tag{2.64}$$

where $d(\cdot,\cdot)$ is the Euclidean distance between two points

$$d(A,B) = \sqrt{(x_A^1 - x_B^1)^2 + (x_A^2 - x_B^2)^2 + (x_A^3 - x_B^3)^2} \tag{2.65}$$

and the subscripts P, N, W, S and E denote the points seen on figure 2.6. The superscripts 1 and 2 on the velocities denote the components of the velocity parallel to the directions e_1 and e_2 respectively.

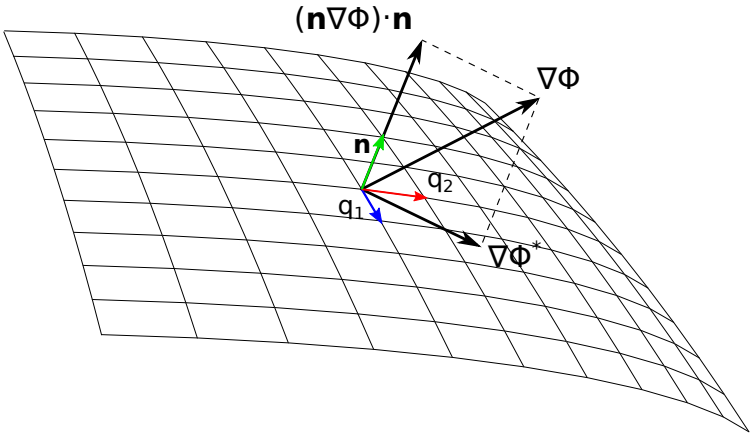

FIGURE 2.5: Geometric representation of the relation between the potential gradient and its trace.

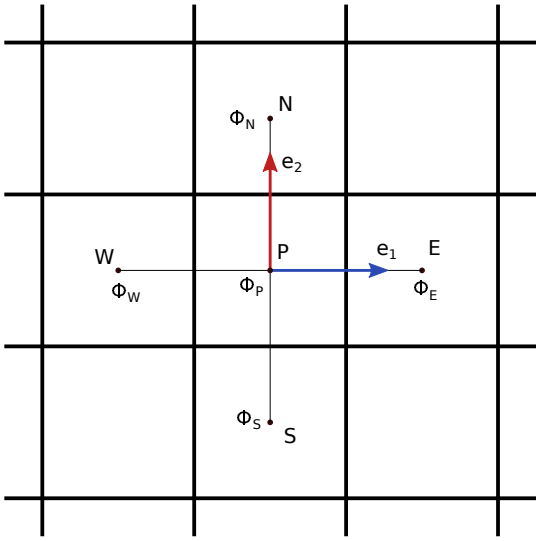

FIGURE 2.6: Grid used for velocity calculations. The points N,W,S and E are named after the four cardinal directions North, West, South and East.

The gradient of the trace potential on the body-fixed 3D coordinate system is then

$$\nabla\Phi^* = \begin{pmatrix} u_P^1 \\ u_P^2 \\ 0 \end{pmatrix} \tag{2.66}$$

The velocity u_P^3 is zero because of the no entrance boundary condition on the body surface, as the velocity is expressed on the body-fixed curvilinear coordinate system.

The boundary (∂D_B), defined in this chapter, is in general a closed boundary. In the case of the foil discretization, however, it is possible to not model the side surfaces of the wing tips, without losing too much information. This simplification produces an open boundary with edges at the wing tips. The trailing edge is also considered an edge of the boundary, as the sharp corner would make it difficult to evaluate the potential derivative with a central difference scheme. So, in the case of calculating velocities on an element at the edge of the boundary, a higher order finite difference scheme is used in the direction that the aforementioned methodology cannot be used. For example, to compute the velocity on a collocation point near the "easternmost" edge of the boundary the following scheme is used.

$$u_P^1 = \frac{\Phi_{W2}^* - 4\Phi_{W1}^* + 3\Phi_P^*}{3d(P, W_1) - d(W_1, W_2)} \tag{2.67}$$

where W_1, W_2 are the points one and two places respectively away from point P in the "west" direction. On the opposite edge of the boundary the equation becomes

$$u_P^1 = \frac{-\Phi_{E2}^* + 4\Phi_{E1}^* - 3\Phi_P^*}{3d(P, E_1) - d(E_1, E_2)} \tag{2.68}$$

where E_1, E_2 are the points one and two places respectively away from point P in the "east" direction.

Correction for non-orthogonal local basis

If the cross-section of the foil varies in the spanwise direction, the local coordinate system on collocation points is not orthogonal. In this more general case the velocity on the foil surface can be calculated using the covariant derivative of the potential

$$\nabla\Phi^* = \mathbf{e}^i \frac{\partial\Phi^*}{\partial q_i} \tag{2.69}$$

where q_i is the basis on the local coordinate system and the vectors $\mathbf{e}^i$ form the contravariant basis. For the unit vectors $\mathbf{e}_i$ along the local coordinate system axes, the contravariant basis is given by

$$\mathbf{e}^i = \frac{\mathbf{e}_j \times \mathbf{e}_k}{\sqrt{g}} \tag{2.70}$$

where $\sqrt{g} = \mathbf{e_1} \cdot (\mathbf{e_2} \times \mathbf{e_3})$ and (i, j, k) is a cyclic permutation.

2.2.5 Calculation of pressure on a boundary point

To calculate the pressure on the surface of the hydrofoil, the unsteady version of the Bernoulli's equation 2.28 is used. Using equations (2.58) and (2.62) the pressure can be evaluated from the BEM results as

$$\frac{p}{\rho} = -\frac{d\Phi_B^*}{dt} + \nabla\Phi_B^* \cdot (\mathbf{V}_B - \mathbf{V}_g) - \frac{1}{2}(\nabla\Phi_B^*)^2 + \frac{1}{2}[\mathbf{n} \cdot (\mathbf{V}_B - \mathbf{V}_g)]^2 \tag{2.71}$$

The pressure coefficient can be found as

$$\begin{aligned} C_p =& \frac{p}{\frac{1}{2}\rho U^2} \\ =& \left\{ -\frac{d\Phi_B^*}{dt} + \nabla\Phi_B^* \cdot (\mathbf{V}_B - \mathbf{V}_g) - \frac{1}{2}(\nabla\Phi_B^*)^2 + \frac{1}{2}[\mathbf{n} \cdot (\mathbf{V}_B - \mathbf{V}_g)]^2 \right\} / \frac{1}{2}U^2 \end{aligned} \tag{2.72}$$

2.2.6 Calculation of force and moment on the hydrofoil

The force can be calculated as the surface integral of the pressure on the body surface

$$\mathbf{F} = -\iint_{\partial D_B} p\mathbf{n}_B dS \tag{2.73}$$

and the moments as

$$\mathbf{M} = \iint_{\partial D_B} p(\mathbf{x} - \mathbf{x}_{ref}) \times \mathbf{n}_B dS \tag{2.74}$$

The non-dimensional lift and drag coefficients are defined respectively as

$$C_L(t) = \frac{L(t)}{\frac{1}{2}\rho U^2 cs} \tag{2.75}$$

and

$$C_D(t) = \frac{D(t)}{\frac{1}{2}\rho U^2 cs} \tag{2.76}$$

Also, the power coefficient is defined

$$C_P(t) = \frac{P(t)}{\frac{1}{2}\rho U^3 cs} \tag{2.77}$$

2.3 Fluid-Structure interaction

As mentioned previously, the foil performs a heaving motion expressed as a linear oscillation. The mounting system is modeled by a spring and the energy generator is modeled by a damper. This oscillator is driven by the lift generated by the hydrofoil. In order to solve the problem numerically, a method to couple the hydrodynamic part of the problem with the oscillation has to be designed.

In the works [74] and [51] such a method is presented. An iterative scheme is used to compute the heaving velocity of the foil.

Here a method to solve the non-linear equation of the oscillator will be presented combining the Crank-Nicholson method for differential equations and the Newton's method for non-linear equations, in order to produce a stable and fast method for the solution of the nonlinear system.

2.3.1 Nonlinear driven oscillator

The oscillation is governed by the rule

$$m\ddot{h} + b\dot{h} + kh = F_L(h, \dot{h}) \tag{2.78}$$

where $F_L(h, \dot{h})$ is the lifting force acting on the foil, as shown in equation 2.1.

By setting

$$\mathbf{x} = \begin{pmatrix} h \\ \dot{h} \end{pmatrix}$$

the equation above can be written as the system of equations

$$P = \begin{cases} m\dot{x}_2 + b\dot{x}_1 + kx_1 = F_L(x_1, x_2) \\ \dot{x}_1 = x_2 \end{cases}$$

or in a matrix form

$$\begin{bmatrix} b & m \\ 1 & 0 \end{bmatrix} \dot{\mathbf{x}} = \begin{bmatrix} -k & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{pmatrix} F_L \\ 0 \end{pmatrix}$$

and setting

$$M = \begin{bmatrix} b & m \\ 1 & 0 \end{bmatrix}$$

and

$$S = \begin{bmatrix} -k & 0 \\ 0 & 1 \end{bmatrix}$$

the equation becomes

$$M\dot{\mathbf{x}} = S\mathbf{x} + \begin{pmatrix} F_L \\ 0 \end{pmatrix} = f(\mathbf{x})$$

Also by the Crank-Nicolson method with no spatial derivatives the equality that has to be satisfied at a given timestep is

$$M\dot{\mathbf{x}}^{t+1} = \frac{1}{2}[f(\mathbf{x}^{t+1}) + f(\mathbf{x}^t)]$$

where t denotes the previous timestep and $t+1$ the current timestep. By approximating the time derivative of $\mathbf{x}$ as

$$\dot{\mathbf{x}^{t+1}} = \frac{1}{\Delta t}(\mathbf{x}^{t+1} - \mathbf{x}^t)$$

the system becomes

$$U(\mathbf{x}^{t+1}) = M(\mathbf{x}^{t+1} - \mathbf{x}^t) - \frac{\Delta t}{2}(S\mathbf{x}^{t+1} + F_L^{t+1} + f(\mathbf{x}^t)) \tag{2.79}$$

The appearance of the lifting force in U introduces an implicit nonlinearity that can be treated using the Newton's method.

For a system of equations the value of the next iteration is obtained by the solution of the system

$$J(f(\mathbf{x}_n)) \cdot (\mathbf{x}_{n+1} - \mathbf{x}_n) = -f(\mathbf{x}_n) \tag{2.80}$$

where $\mathbf{x}$ is a vector and $J(f(\mathbf{x}_n))$ is the Jacobian matrix of the function f. The Jacobian matrix is defined for a function f as

$$J(f) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \tag{2.81}$$

Therefore, at each iteration the new heave position is calculated by the solution of the system

$$J(U)(\mathbf{x}_{n+1}^{t+1} - \mathbf{x}_n^{t+1}) = -U(\mathbf{x}_n^{t+1})$$

2.3.2 Half oscillator

In the special case that the mass of the foil is considered small, the inertial term of equation 2.1 vanishes, reducing the system above to the single equation

$$b\dot{h} + kh = F_L(h, \dot{h}) \tag{2.82}$$

This case will be used to compare the results of the present method with Zhu et al. [75], as they consider the mass of the foil small compared to the added mass due hydrodynamic effects.

Discretizing in time and approximating the time derivative of heave with an implicit Euler scheme the equation becomes

$$b\frac{h^{t+1} - h^t}{\Delta t} = -kh^{t+1} + F_L(h^{t+1}, \frac{h^{t+1} - h^t}{\Delta t}) = f(h^{t+1}) \tag{2.83}$$

Also, by the Crank-Nicolson method, the equality that has to be satisfied at a given timestep is

$$b\frac{h^{t+1}-h^t}{\Delta t} = \frac{1}{2}(f(h^{t+1}) + f(h^t)) \tag{2.84}$$

or

$$U(h^{t+1}) = h^{t+1} - \frac{bh^t + \frac{\Delta t}{2}F_L(h^{t+1}, \frac{h^{t+1}-h^t}{\Delta t}) + \frac{\Delta t}{2}f(h^t)}{b + \frac{\Delta t}{2}k} = 0 \tag{2.85}$$

and

$$\dot{h}^{t+1} = \frac{h^{t+1}-h^t}{\Delta t} \tag{2.86}$$

Again the Newton's method is used to calculate the new heave position. At each iteration the new heave position is calculated as

$$h_{n+1}^{t+1} = h_n^{t+1} - \frac{U(h_n^{t+1})}{U'(h_n^{t+1})}$$

where the derivative of U is

$$U'(h_n^{t+1}) = 1 - \frac{\Delta t}{2(b + \frac{\Delta t}{2}k)}\frac{dF_L}{dh}(h_n^{t+1}, \frac{h_n^{t+1}-h_n^t}{\Delta t})$$

and $\frac{dF_L}{dh}$ can be approximated as

$$\frac{dF_L}{dh}(h_n^{t+1}, \frac{h_n^{t+1}-h_n^t}{\Delta t}) = \frac{F_L(h_n^{t+1}+\epsilon, \frac{h_n^{t+1}+\epsilon-h_n^t}{\Delta t}) - F_L(h_n^{t+1}-\epsilon, \frac{h_n^{t+1}-\epsilon-h_n^t}{\Delta t})}{2\epsilon}$$

where ϵ is a small positive real number.

Chapter 3

Calculation of induction factors

For the method derived in chapter 2 the performance (execution time and accuracy) of the algorithm depends on the robustness of the calculation of the induction factors. In this chapter the integration techniques used in the present work for the calculation of induction factors will be discussed.

Initially, simple quadratures will be presented for one-dimensional domains, along with an adaptive algorithm for recursive integration of subintervals. Next, the adaptive algorithm will be extended for surface integrals and generalized for arbitrary number of dimensions.

3.1 Simple quadratures

There are many families of integration quadratures. Here the Gauss-Legendre, Newton-Cotes and Gauss-Lobatto quadratures will be reviewed. In the most general case, a quadrature rule can be given by the formula

$$If = \int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{3.1}$$

where x_i are the evaluation nodes of the function f and w_i are weights. Both the nodes and the weights are dependent of specific methods.

With a simple change of variable the quadrature can evaluate the integral on any interval $D = [a, b]$ as follows:

$$\int_{a}^{b} f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}x + \frac{a+b}{2}\right) dx \tag{3.2}$$

A quadrature can be either open or closed. When the evaluation nodes include the boundaries of the integration interval ($x_0 = a$ and $x_n = b$) the quadrature is considered closed, while when the interval $[x_0, x_n]$ does not coincide with the interval D the quadrature is said to be open.

3.1.1 Gauss-Legendre Quadrature

The Gauss-Legendre quadrature is mentioned here for completeness, as it can efficiently evaluate integrals of well-behaving functions. That is the case for the evaluation of induced potential on a collocation point afar from the boundary element.

The weights for the Gauss-Legendre quadrature are given by the formula [1, p. 887]

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2} \tag{3.3}$$

where P_n are the Legendre polynomials normalized so that $P_n(1) = 1$. The Legendre polynomials are the polynomial solutions to the Legendre differential equation

$$\frac{d}{dx}\left[(1-x^2)\frac{dP_n(x)}{dx}\right] + n(n+1)P_n(x) = 0 \tag{3.4}$$

One useful expression for the Legendre polynomials is given by Rodrigues [56]:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n \tag{3.5}$$

The nodes x_i of the quadrature are given by the roots of P_n. It must be noted that the Gauss-Legendre quadrature is open, as the values of P_n are either -1 or 1 at $x = -1$ and 1 at $x = 1$.

The Gauss-Legendre quadrature is able to integrate exactly polynomials of order $2n-1$ [1, p. 888].

3.1.2 Newton-Cotes Quadrature

The Newton-Cotes Quadratures are a useful family of numerical integration techniques. The general formula for the Newton-Cotes quadratures (NCQ) is the following

$$\int_a^b f(x)dx = \sum_{i=1}^{n} w_i f(x_i),$$

where

$$w_i = \int_a^b l_i(x)dx$$

are the weights corresponding to the nodes

$$x_i = a + (i-1)\frac{b-a}{n-1}$$

and l_i are the Lagrange basis polynomials. In the general form for an n-point NCQ the weights can be found by the formula

$$w_i = H_{n,i=r+1} = \frac{(-1)^{n-r}}{r!(n-r)!}\int_0^n t(t-1)\cdots(t-r+1)(t-r-1)\cdots(t-n)dt$$

referenced from [70].

The NCQs can be divided into the two categories of Closed and Open NCQs. Newton-Cotes formulas are considered "closed" if the entire interval $[x_0 = a, x_n = b]$ is included in the fit, and "open" otherwise.

Although these methods of integration are useful for most applications, they tend to be unstable for large n, as they can suffer from Runge's phenomenon, where the error grows exponentially for large n. Stable Newton-Cotes formulae can be constructed using least-square approximation instead of interpolation.

3.1.3 Gauss-Lobatto Quadrature

The Gauss-Lobatto quadrature (GLQ) is a Gaussian quadrature with weighting function $W(x) = 1$ in which the endpoints of the canonical interval [-1,1] are included in a total of n nodes, giving r=n-2 free nodes ([68]). Nodes are symmetrical about the origin, and the general formula is

$$\int_{-1}^{1} f(x)dx = w_1 f(-1) + w_n f(1) + \sum_{i=2}^{n-1} w_i f(x_i).$$

The free nodes x_i for $i = 2, ..., n-1$ are the roots of the polynomial $P'_{n-1}(x)$, where P(x) is a Legendre polynomial. The weights of the free nodes are

$$w_i = -\frac{2n}{(1-x_i^2)P''_{n-1}(x_i)P'_m(x_i)} = \frac{2}{n(n-1)[P_{n-1}(x_i)]^2},$$

and of the endpoints are

$$w_{1,n} = \frac{2}{n(n-1)}.$$

The error term of the quadrature is given by

$$E = -\frac{n(n-1)^3 2^{2n-1}[(n-2)!]^4)}{(2n-1)[(2n-2)!]^3} f^{(2n-2)}(\xi) \tag{3.6}$$

The GLQ integrates exactly polynomials of degree $2n-3$ or less, as can be deduced from the error term.

Here the interval of integration is [-1,1] but every interval [a,b] can be scaled and translated appropriately to fit exactly the interval [-1,1]. Of course, the integrand f must be multiplied with the scale factor.

To calculate the nodes of the quadrature, the roots of the polynomial $P'_{n-1}(x)$ must be found first. The zeros of P'_{n-1} are the same as those of the associated Legendre polynomial $P_n^1(x)$ through the relation

$$P_n^1(x) = \sqrt{x^2 - 1} P_n'(x)$$

One can then approximate the zeros of $P_n^1(x)$ in terms of the successive zeros of the Bessel function $J_1(x)$. Mitchels ([41]) proposed the approximation

$$x_{n,k} = cos \left\{ \frac{j_{1,k}}{\sqrt{(n - 1/2)^2 + \left(\frac{\pi^2 - 4}{4\pi^2}\right)}} \right\}$$

where $j_{1,k}$ is the k-th zero of the Bessel function. Using a Newton-Raphson iterative process one can refine the place of the nodes as shown below.

$$(x_{n,k})_{i+1} = (x_{n,k})_i - \frac{P'_{n-1}(x_{n,k})_i}{P''_{n-1}(x_{n,k})_i}$$

3.2 Adaptive integration

The main idea behind the adaptive quadrature is the recursive subdivision of the integration interval (D). A simple quadrature is used to obtain a first estimate (Q) of the integral. Next, the interval is partitioned to m subintervals $D_i = [a_i, b_i]$, i=1(1)m such that $D = \cup D_i$ and the simple quadrature is used on each subinterval. Then, the first estimation is compared to the sum of the subinterval estimations to determine the convergence. This procedure is repeated on every subinterval until a convergence criterion is satisfied.

Lyness [37] studied an adaptive quadrature based on the simple Simpson's (1-4-1) rule. He proposed an accurate convergence criterion and modified the quadrature to include a term for convergence acceleration.

Gander and Gautschi [19] present an adaptive quadrature based on the four-point Gauss-Lobatto rule with two successive Kronrod extensions. In that paper special attention is given on the convergence criterion of the adaptive routine.

The basic procedure for the adaptive quadrature is presented in algorithm 1. The adaptive routine takes as first arguments the integration interval $[a, b]$ and the integrand function f. Next, there are two parameters that control the algorithm. The tolerance (tol) determines when the algorithm is considered to have converged from one recursion level to the next. The maximum recursion depth (maxDepth), the adaptive routine is allowed to reach, determines the maximum allowable refinement of the integration interval. When the routine takes many recursions to converge it can affect the performance and setting a cap at the recursion level can improve the time needed for the operation by sacrificing some precision. Other arguments of the adaptive routine are the simple quadrature used internally (quadr) and the approximation of the integral on a coarser subdivision of the integration domain (Q).

Closed quadrature rules are preferred, as the integrand function evaluations for the first node of the first subinterval and the last node of the last subinterval at a given recursion are known from the upper recursion level. Therefore, Gauss-Legendre and open Newton-Cotes quadratures are not optimal choices for this application.

Algorithm 1 Adaptive quadrature recursive algorithm

```
1: procedure ADAPTIVEROUTINE(a, b, f, m, tol, maxDepth, quadr, Q)
2:     for i = 1(1)m do
3:         a_i ← a + ((b−a)/m)(i − 1)
4:         b_i ← a + ((b−a)/m) i
5:         S_i ← quadr(a_i, b_i, f)
6:     S ← Σ_{i=1}^{m} S_i
7:     if (|S-Q| < tol OR maxDepth==0) then return S
8:     else
9:         for i = 1(1)m do
10:            Q_i ← AdaptiveRoutine(a_i, b_i, f, m, tol/m, maxDepth − 1, quadr, S_i)
11:        return Σ_{i=1}^{m} Q_i
```

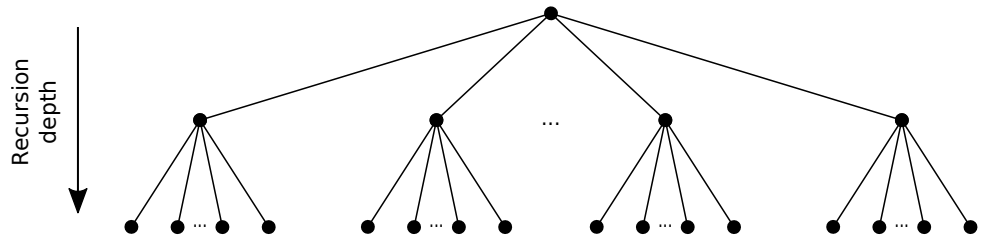

FIGURE 3.1: m-division strategy.

3.2.1 Partitioning scheme and termination criterion

For the adaptive routine to work optimally, the partitioning scheme must be selected carefully. Uniform bisection and trisection of the interval have been studied and trisection is proven to be better [62]. Berntsen et al. [6] discuss a nonuniform 3-division. Their proposed algorithm estimates the width of a difficulty and divides the interval according to the width and the location of the difficulty. They report a 50% reduction in the work needed for the evaluation of integrals of singular functions.

In the present work a general uniform m-division is studied. The interval is divided in m equal parts and the performance is evaluated in order to choose the best partitioning scheme for the adaptive integrator in the case of the single and double layer integrals found in BEM.

The convergence criterion can be absolute or relative to a "characteristic length" of the domain. Here an absolute criterion is used to test the convergence.

3.2.2 Richardson extrapolation

The Richardson extrapolation is a technique to increase the convergence rate of a numerical method with no additional function evaluations.

As stated by Richardson and Gaunt [55] an approximation of the actual value of an unknown function $f(x)$ at a point x is

$$\phi(x,h) = f(x) + hf_1(x) + h^2 f_2(x) + h^3 f_3(x) + ... \tag{3.7}$$

where the functions f_1, f_2, f_3... are usually unknown.

If there is a numerical method (A) that can produce a result with accuracy of order k for a step h, the actual value of the function will be

$$f(x) = A(x,h) + h^{k+1} f_{k+1}(x) + h^{k+2} f_{k+2}(x) + ... \tag{3.8}$$

or using the Big O notation

$$f(x) = A(x,h) + h^{k+1} f_{k+1}(x) + O(h^{k+2}) \tag{3.9}$$

By using two different values for the step h that are connected with the relation

$$h_2 = \frac{h_1}{t} \tag{3.10}$$

where t in the case of the adaptive integration routine is equal to m. For the two different step sizes equation (3.13) becomes

$$f(x) = A(x,h_1) + h_1^{k+1} f_{k+1}(x) + O(h_1^{k+2}) \tag{3.11}$$

$$f(x) = A(x,\frac{h_1}{t}) + \left(\frac{h_1}{t}\right)^{k+1} f_{k+1}(x) + O(h_1^{k+2}) \tag{3.12}$$

By multiplying equation 3.12 by t^{k+1} and subtracting 3.11 we get

$$f(x) = \frac{t^{k+1} A\left(\frac{h_1}{t}\right) - A(h_1)}{t^{k+1} - 1} + O(h^{k+2}) \tag{3.13}$$

For the Gauss-Lobatto quadrature of order n the truncation error of the approximation is $O(h^{2n-2})$ (see eq. 3.6) and for the adaptive routine with m uniform subdivisions at each recursion level the convergence can be accelerated by applying the Richardson extrapolation as

$$If = \frac{m^{2n-2} GLQ_n\left(\frac{h_1}{m}\right) - GLQ_n(h_1)}{m^{2n-2} - 1} + O(h_1^{2n-1}) \tag{3.14}$$

The above equation can be rewriten as

$$(If - GLQ_n\left(\frac{h_1}{m}\right))(m^{2n-2} - 1) = GLQ_n\left(\frac{h_1}{m}\right) - GLQ_n(h_1) \tag{3.15}$$

and by replacing $(If - GLQ_n\left(\frac{h_1}{m}\right))$ by the error ϵ we get the termination condition

$$|GLQ_n\left(\frac{h_1}{m}\right) - GLQ_n(h_1)| < \epsilon(m^{2n-2} - 1) \tag{3.16}$$

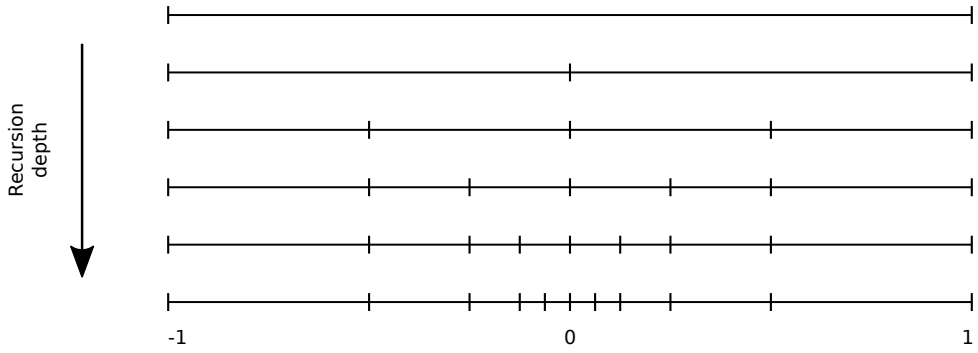

FIGURE 3.2: Interval subdivision for a function with singularity at $x = 0$ for $m = 2$.

3.3 Treatment of singularities

The adaptive routine will automatically create denser partitions around a singularity. The function near a singularity cannot be approximated correctly by the classes of functions considered in producing the Gauss-Lobatto and Newton-Cotes quadratures, and as a result the method will not converge for the current recursion level and will continue one level lower, subdividing the domain even more.

In figure 3.2, the uniform bisection of the integration integral is drawn for an integrand function that displays a singularity at $x = 0$. The interval is refined near the singularity, where the accuracy of the quadrature used does not suffice to evaluate the integral correctly.

3.4 Surface integrals

To compute a surface integral, there are two levels of integration that are required, one outer integration and one inner. To evaluate a singular surface integral it is important to have a very good approximation of the inner integral, as this is the integral that bears the singularity. Then, the outer integration can be done by a less accurate routine. In this way the benefits of accuracy and robustness of the intricate routine used for the inner integration are combined with the great speed of the simpler routine of the outer integration. Finally a fast routine for evaluating singular surface integrals is produced.

3.5 N-Dimensional

The adaptive routine can be generalized for integration over N-dimensional domains. By passing an adaptive integrator of (N-1) dimensions as an integrand function for the N-dimensional integrator and giving the N-th element of the position vector as a parameter, the routine will perform integrations recursively on all dimensions.

It is apparent that as the number of dimensions increases, the computational complexity will increase exponentially. Therefore, even if it is possible to integrate in any number of dimensions, this is practically unachievable.

3.6 Performance

In this section the adaptive routine will be used to compute the single and double layer potential terms. In order to compare the results, analytical methods for the calculation of the double and single layer integrals will be presented. Next, the convergence of the adaptive quadrature will be tested for different recursion levels.

3.6.1 Analytical methods

Induced potential of constant source and dipole distribution on flat bilinear element

The analytical solution to the induced factor calculation for a flat quadrilateral with constant source or doublet distribution can be found in [28, Chapters 10.4.1 and 10.4.2]. Note that in this solution the z axis must be perpendicular to the plane of the quadrilateral. For the source distribution the solution is

$$\begin{aligned}\Phi = \frac{-\sigma}{4\pi}\Bigg\{\Bigg[&\frac{(x-x_1)(y_2-y_1)-(y-y_1)(x_2-x_1)}{d_{12}} ln\frac{r_1+r_2+d_{12}}{r_1+r_2-d_{12}} \\ &+\frac{(x-x_2)(y_3-y_2)-(y-y_2)(x_3-x_2)}{d_{23}} ln\frac{r_2+r_3+d_{23}}{r_2+r_3-d_{23}} \\ &+\frac{(x-x_3)(y_4-y_3)-(y-y_3)(x_4-x_3)}{d_{34}} ln\frac{r_3+r_4+d_{34}}{r_3+r_4-d_{34}} \\ &+\frac{(x-x_4)(y_1-y_4)-(y-y_4)(x_1-x_4)}{d_{41}} ln\frac{r_4+r_1+d_{41}}{r_4+r_1-d_{41}}\Bigg] \\ &-|z|\Bigg[tan^{-1}\left(\frac{m_{12}e_1-h_1}{zr_1}\right)-tan^{-1}\left(\frac{m_{12}e_2-h_2}{zr_2}\right) \\ &+tan^{-1}\left(\frac{m_{23}e_2-h_2}{zr_2}\right)-tan^{-1}\left(\frac{m_{23}e_3-h_3}{zr_3}\right) \\ &+tan^{-1}\left(\frac{m_{34}e_3-h_3}{zr_3}\right)-tan^{-1}\left(\frac{m_{34}e_4-h_4}{zr_4}\right) \\ &+tan^{-1}\left(\frac{m_{41}e_4-h_4}{zr_4}\right)-tan^{-1}\left(\frac{m_{41}e_1-h_1}{zr_1}\right)\Bigg]\Bigg\}\end{aligned} \tag{3.17}$$

where

$$d_{12} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \tag{3.18}$$

$$d_{23} = \sqrt{(x_3-x_2)^2+(y_3-y_2)^2} \tag{3.19}$$

$$d_{34} = \sqrt{(x_4-x_3)^2+(y_4-y_3)^2} \tag{3.20}$$

$$d_{41} = \sqrt{(x_1-x_4)^2+(y_1-y_4)^2} \tag{3.21}$$

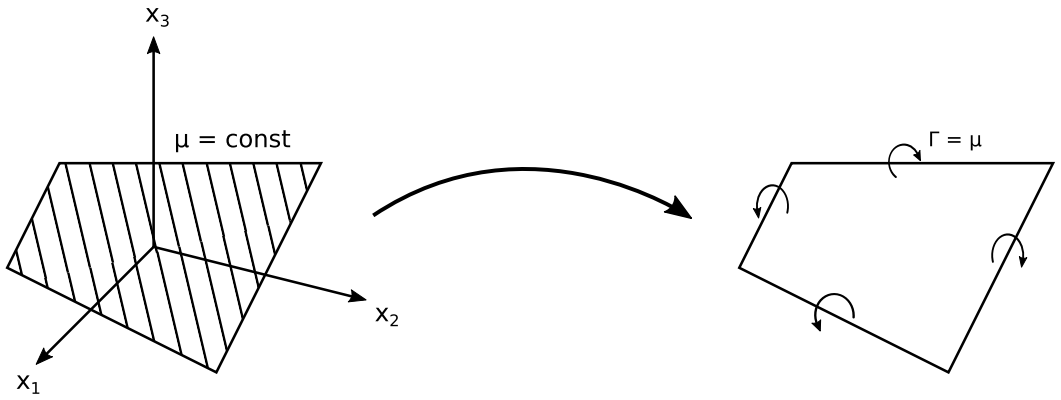

FIGURE 3.3: Geometry of a flat quadrilateral with constant doublet distribution and the equivalent vortex ring.

$$m_{12} = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.22}$$

$$m_{23} = \frac{y_3 - y_2}{x_3 - x_2} \tag{3.23}$$

$$m_{34} = \frac{y_4 - y_3}{x_4 - x_3} \tag{3.24}$$

$$m_{41} = \frac{y_1 - y_4}{x_1 - x_4} \tag{3.25}$$

and

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + z^2} \qquad i = 1,2,3,4 \tag{3.26}$$

$$e_i = (x - x_i)^2 + z^2 \qquad i = 1,2,3,4 \tag{3.27}$$

$$h_i = (x - x_i)(y - y_i) \qquad i = 1,2,3,4 \tag{3.28}$$

For the dipole distribution the solution is

$$\begin{aligned}\Phi = \frac{\mu}{4\pi}\Bigg[& tan^{-1}\left(\frac{m_{12}e_1 - h_1}{zr_1}\right) - tan^{-1}\left(\frac{m_{12}e_2 - h_2}{zr_2}\right) \\ & + tan^{-1}\left(\frac{m_{23}e_2 - h_2}{zr_2}\right) - tan^{-1}\left(\frac{m_{23}e_3 - h_3}{zr_3}\right) \\ & + tan^{-1}\left(\frac{m_{34}e_3 - h_3}{zr_3}\right) - tan^{-1}\left(\frac{m_{34}e_4 - h_4}{zr_4}\right) \\ & + tan^{-1}\left(\frac{m_{41}e_4 - h_4}{zr_4}\right) - tan^{-1}\left(\frac{m_{41}e_1 - h_1}{zr_1}\right)\Bigg]\end{aligned} \tag{3.29}$$

Induced potential of constant unit strength dipole distribution on non-flat bilinear element

The induced potential from a dipole distribution on a bilinear element surface to a point P can be calculation by evaluating the surface integral

$$\Phi = \frac{1}{4\pi} \int_A \mathbf{n} \frac{\mathbf{r}}{r^3} dS(Q) \tag{3.30}$$

where

A: the quadrilateral element's area

n: the normal vector at the center of the element

r: the vector from point Q to point P, and

r: the magnitude of **r** ($r = x_P - x_Q$ and $r = ||\mathbf{r}||$)

As stated by Thermos [63], it can be observed that, if there was a sink of unit strength at point P, the sink flow rate through A would be the same as the potential we want to evaluate.

$$Q(A) = -\frac{1}{4\pi} \int_A \mathbf{n} \frac{\mathbf{x}_Q - \mathbf{x}_P}{r^3} dS(Q) = \Phi \tag{3.31}$$

The sink flow rate through the element's area is the same as the flow rate through the area of a sphere with unit radius and center the point P and bounded by the projection of the edges of the bilinear element on the sphere surface. Then, the induced potential is easily calculated as

$$\Phi = \int_{S(c=x_P, r=1)} \frac{-1}{4\pi} \frac{\mathbf{x}_Q - \mathbf{x}_P}{r^3} dS = -\frac{1}{4\pi} \int_S r^{-2} dS = -\frac{\alpha}{4\pi} \rightarrow \pm \frac{\alpha}{4\pi} \tag{3.32}$$

where α is the solid angle of the element. To evaluate the solid angle one can divide the bilinear element into two elements with three vertices each, as shown in figure **??**. The three vertices of each element along with the point P form a tetrahedron whose solid angle with respect to point P is known and equal to

$$\alpha_{tetrahedron} = 2\pi - \phi_1 - \phi_2 - \phi_3 \tag{3.33}$$

where ϕ_1, ϕ_2 and ϕ_3 are the angle between the different planes of the faces of the tetrahedron with common vertex the point P. The solid angle of the bilinear element will be the sum of the solid angles of the two tetrahedrons and finally the induced potential is

$$\Phi = \frac{\alpha_1 + \alpha_2}{4\pi} \tag{3.34}$$

It should be noted that this method can yield an analytical solution only if the point P is not inside the region defined by the simplified geometry and the real geometry of the element. In the case that the evaluation point is in this region the result will have wrong sign.

3.6.2 Test cases

In the present work two bilinear elements will be studied. The first (A) is flat with nodes $\{X_1 = (-1,-1,0)^T, X_2 = (-1,1,0)^T, X_3 = (1,1,0)^T, X_4 = (1,-1,0)^T\}$.The induced potential will be evaluated on points along the line $z = 0$, passing through the elements, for $z \in [-1,1]$.

The second (B) is slightly distorted with nodes $\{X_1 = (-1,-1,-0.25)^T, X_2 = (-1,1,0.25)^T, X_3 = (1,1,-0.25)^T, X_4 = (1,-1,0.25)^T\}$. The evaluation points belong to the line that passes from the center of the element and is normal to it. The line is parametrized by the parameter t, that denotes the distance from the center of the element.

For more information on the geometry of the bilinear elements see appendix A.

The tolerance is set to machine precision to allow the algorithm to reach the recursion limit.

3.6.3 Convergence

Figures 3.4 to 3.9 show the convergence and the time needed to calculate the integrals, for the single and double layer induction factors for case A, and figures 3.10 to 3.12 show the convergence and the time needed to calculate the integrals for the double layer potential, for case B. The outer integration in both test cases is performed by the GLQ with 3 nodes (GLQ3).

For case A, the inner integral was evaluated by the GLQ with 3 and 11 nodes (GLQ3 and GLQ11) and the NCQ with 5 nodes (NCQ5). The NCQ5 method was systematically faster than the GLQs, for low recursion depths but the error was higher than that of GLQ11. In the case that more recursion depths were needed the speed advantage of the NCQ becomes insignificant and the GLQ11 shows faster convergence.

For case B, the GLQ3 and GLQ11 were tested, together with the seven-node GLQ (GLQ7). In this case the GLQ7 and GLQ11 showed similar performance with a small trade-off of accuracy and speed between the two.

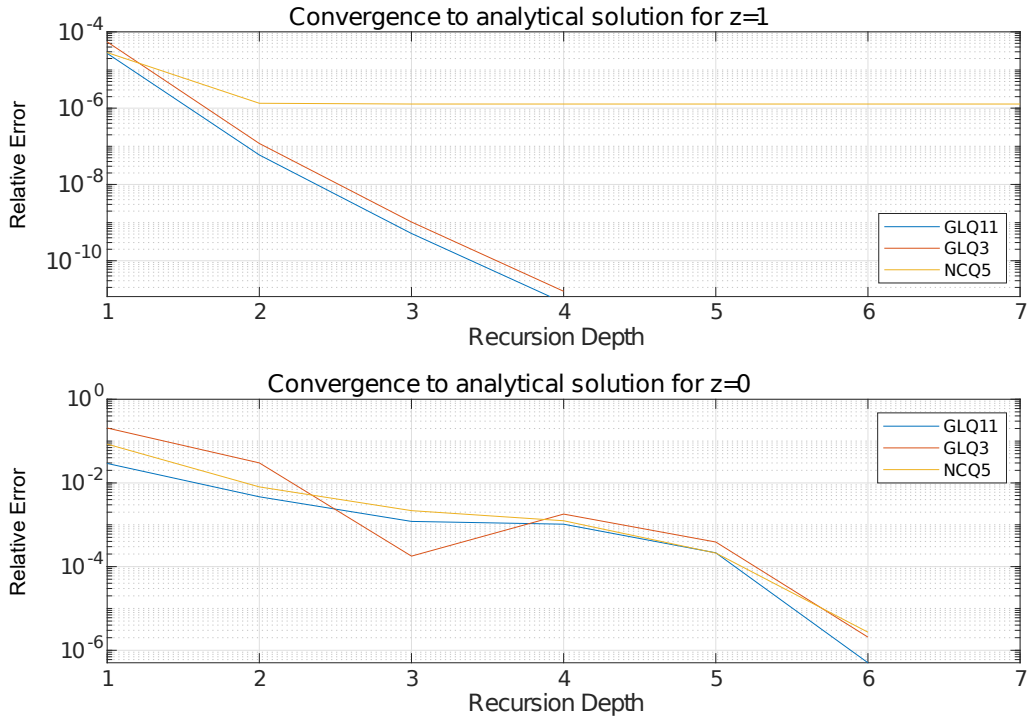

FIGURE 3.4: Convergence of GLQ to exact solution of single layer induced potential for increasing recursion depths.

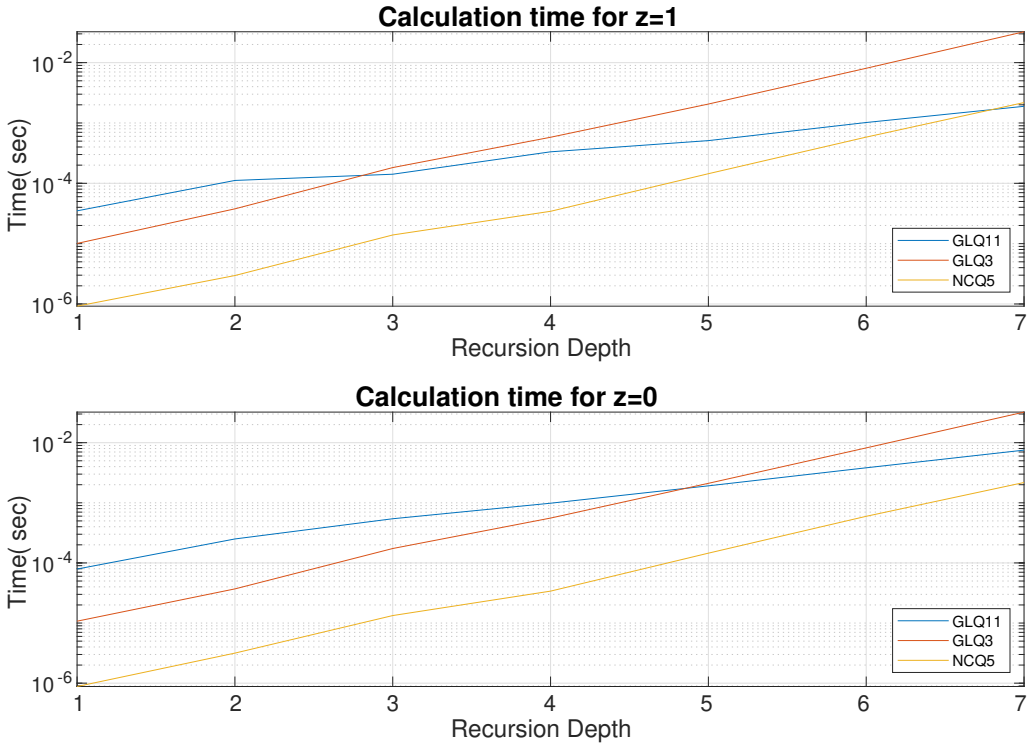

FIGURE 3.5: Calculation time for single layer induced potential with GLQ for increasing recursion depths.

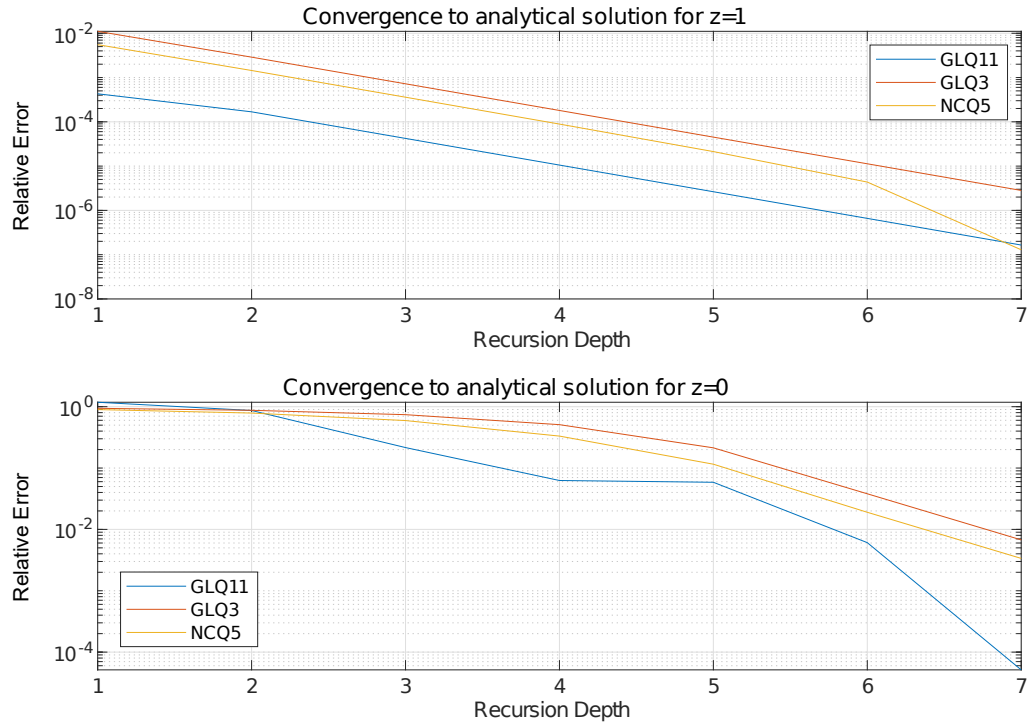

FIGURE 3.6: Convergence of GLQ to exact solution of double layer induced potential for increasing recursion depths.

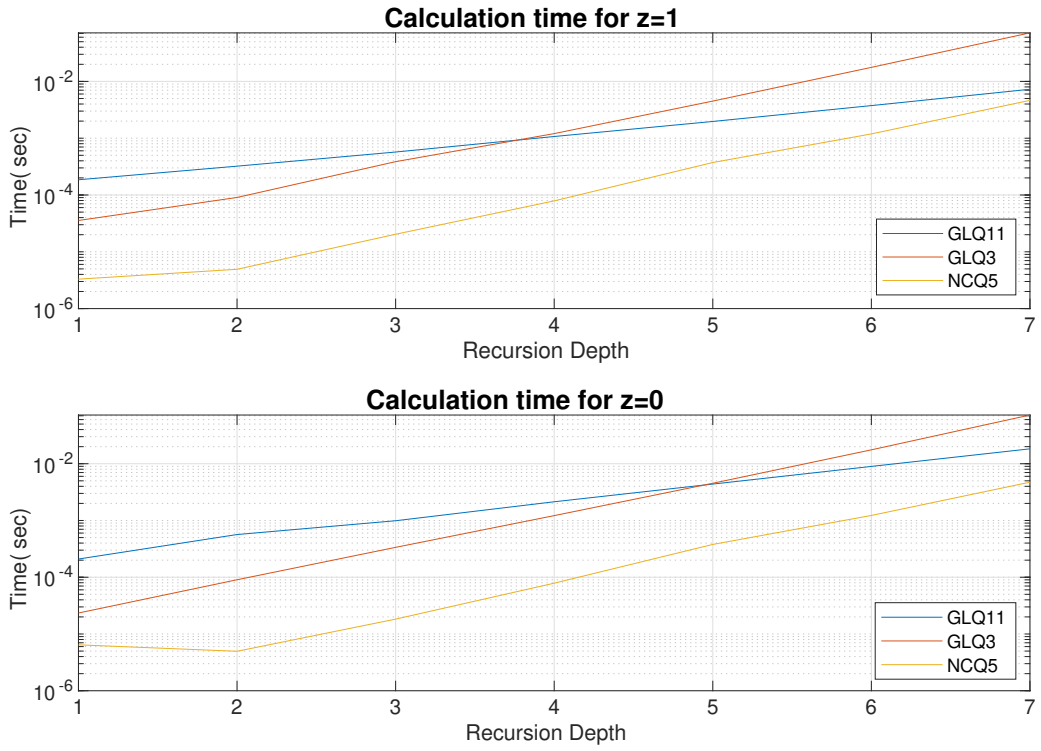

FIGURE 3.7: Calculation time for double layer induced potential with GLQ for increasing recursion depths.

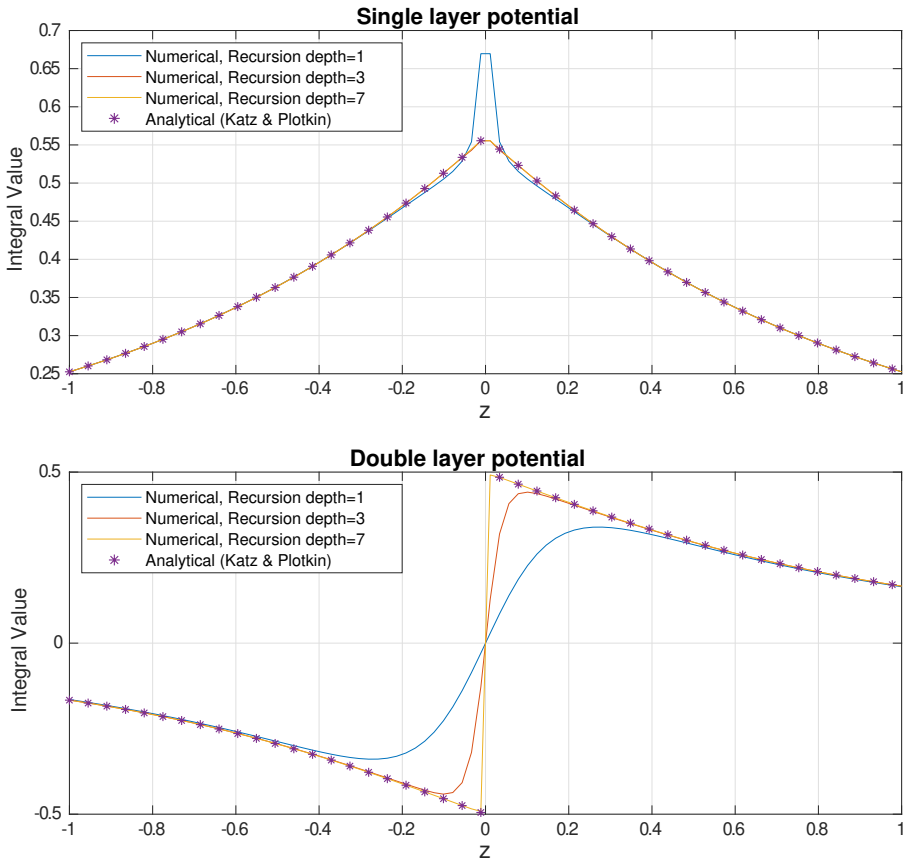

FIGURE 3.8: Comparison of calculated single and double layer induction factors by the GLQ3 with analytical solution [28, Chapters 10.4.1 and 10.4.2].

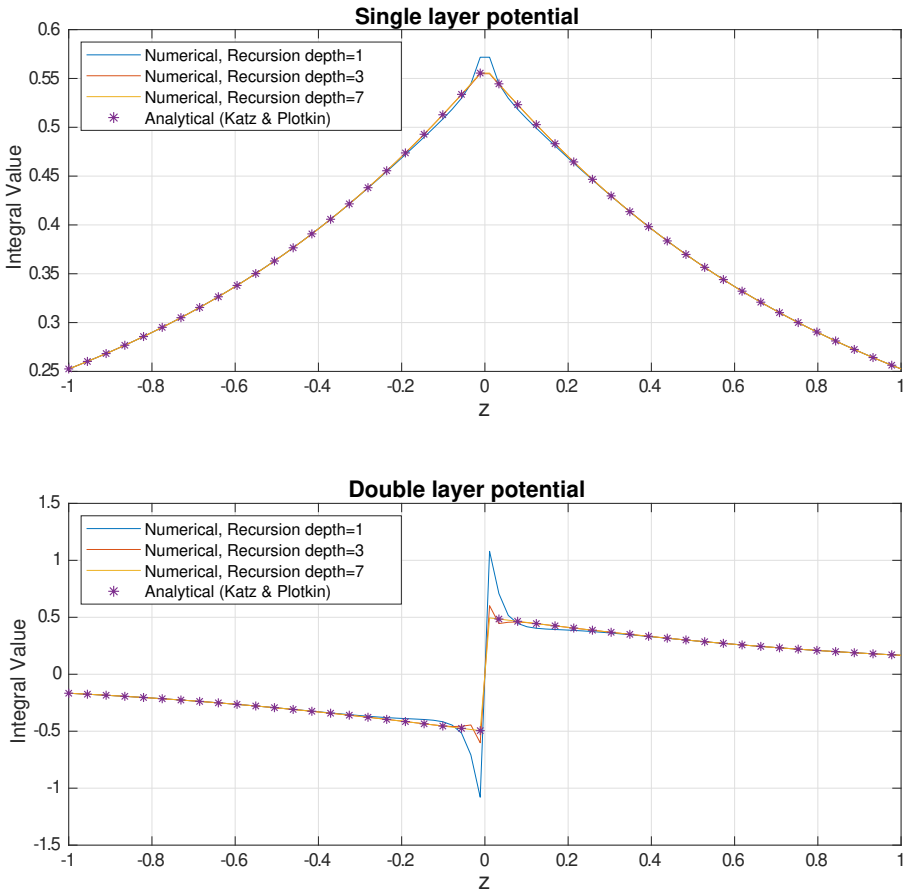

FIGURE 3.9: Comparison of calculated single and double layer induction factors by the GLQ11 with analytical solution [28, Chapters 10.4.1 and 10.4.2].

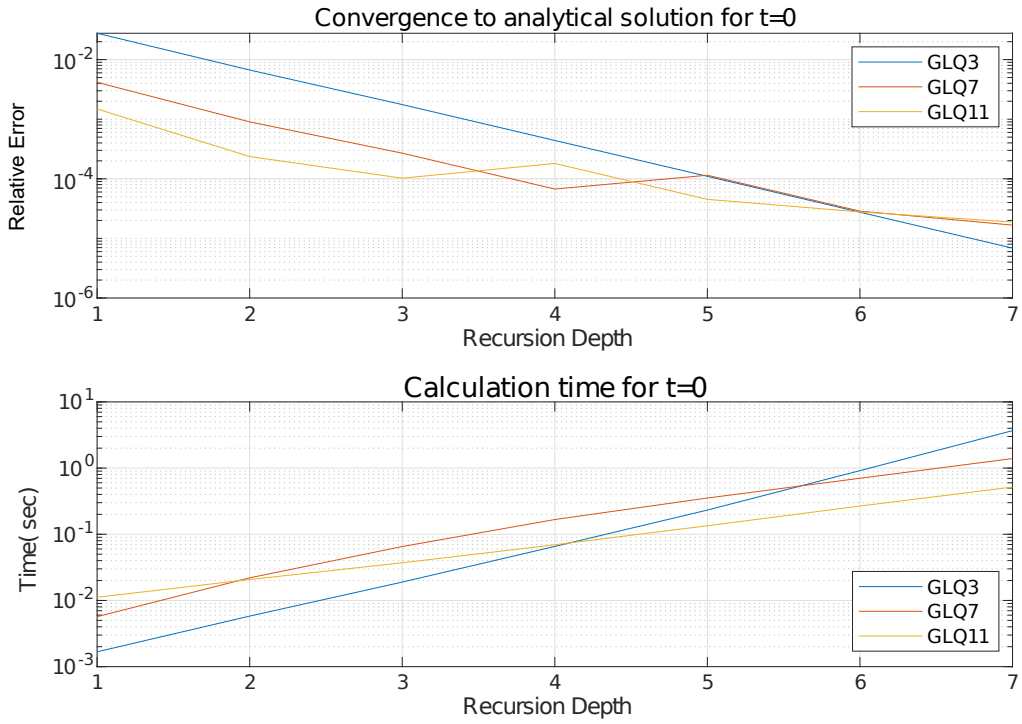

FIGURE 3.10: Comparison of calculated double layer induction factors by the GLQ with analytical solution [63], for distance $t = 6$ from the center of the distorted bilinear element.

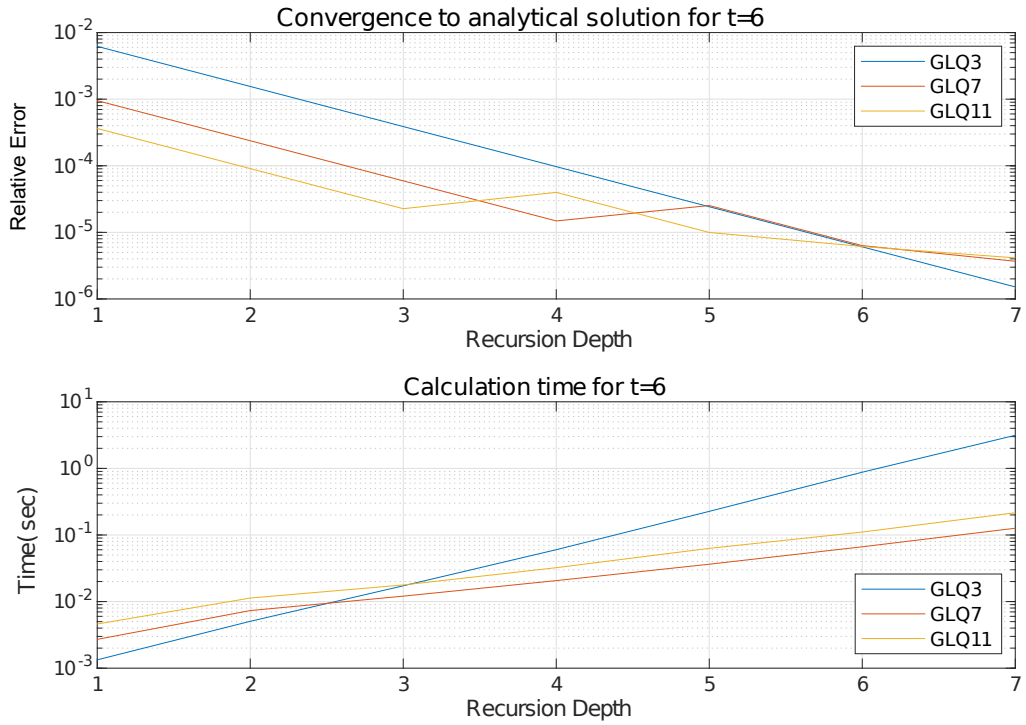

FIGURE 3.11: Comparison of calculated double layer induction factors by the GLQ with analytical solution [63], for distance $t = 6$ from the center of the distorted bilinear element.

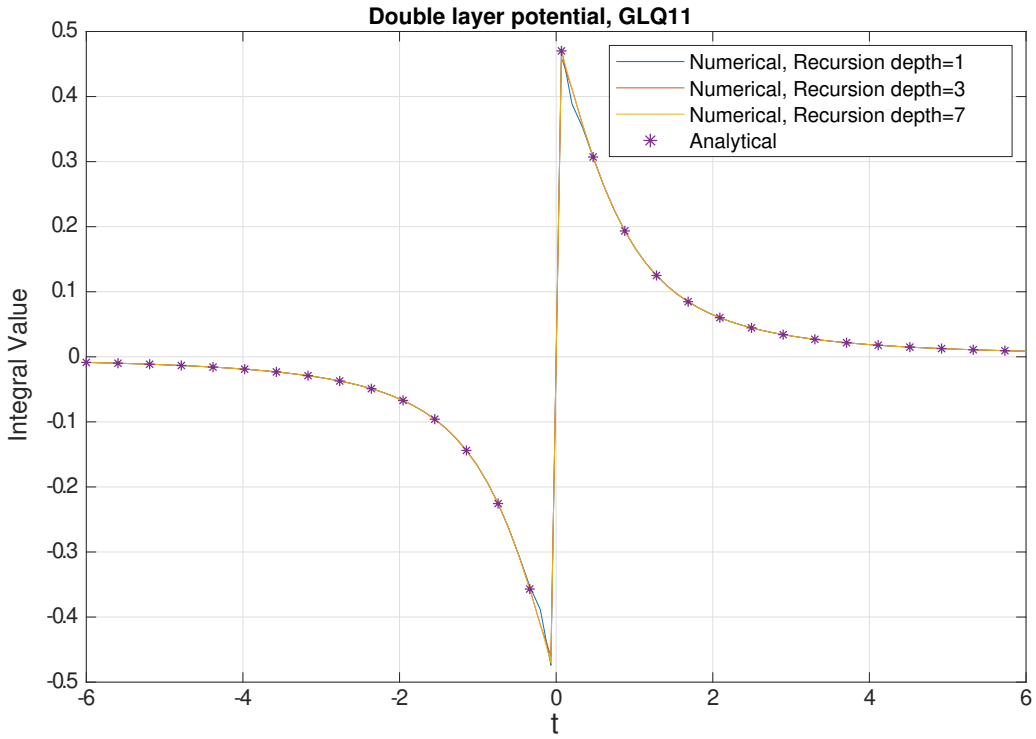

FIGURE 3.12: Comparison of calculated double layer induction factors by the GLQ11 with analytical solution [63], for different values of the parameter t, for the distorted bilinear element.

Chapter 4

GPU implementation

In this chapter we will focus on the implementation of the BEM on NVidia GPUs. For this purpose the Compute Unified Device Architecture (CUDA) will be examined to determine the best programming practices for the problem at hand. The code produced by the present work is an extension of the code developed by E. S. Filippas during his PhD studies [15]. The code was reprogrammed using object-oriented programming.

The GPUs share many features with modern CPUs, but have some differences that make GPUs more robust for some numerical algorithms. The GPU core is inferior to a CPU core in terms of performance and complexity, as the cores of the GPU are slower and simpler on hardware level. This means a CPU core can deliver more FLOPS than a single GPU core. On the other hand, contemporary GPUs can have thousands of cores, giving them a significant performance boost when executing parallel code. The independent calculations of the parallel algorithm are distributed across all the cores and as a result the independent calculations can be executed simultaneously. If an algorithm can be rewritten as a parallel one, it can benefit from the high core count of the GPU.

In order to create software that is efficient, it is of high importance to study the architecture of the hardware it will run on. In the next section the design aspects of a typical NVidia GPU will be discussed and the details of the GPUs used to test the code will be examined.

Information about NVidia GPUs in the next sections is derived from the *CUDA C PROGRAMMING GUIDE* [49] and the whitepapers for the specific GPU architectures used.

For the implementation of the method described in Chapter 2 the cuBLAS and cuSOLVER libraries were used for matrix-vector, matrix-matrix operations and linear system solvers. In addition, the jsonpp open-source library was used to read the simulation parameters written using the Javascript Object Notation. The use of this format allows the parameters to be human readable.

4.1 GPU architecture

For our testing purposes two different GPUs were used. The first is the NVidia GeForce GTX 650 Ti BOOST, using the GK106 Kepler compute architecture, and the second is the GTX 1050, using the GP107 Pascal architecture. In later sections the two architectures will be discussed.

In order to understand the different architectures, a general overview of Nvidia's GPU design should be described first. The design of the GPU was historically driven by the content creation industry and many features of the modern GPU are a result of the effort to increase the functionality of the hardware and adapt it to the graphics generation pipeline. Nowadays, the paradigm of GPU design has shifted to include the general purpose computing community, with many features dedicated to complex mathematical tasks.

In the following section the building blocks of a GPU application will be discussed in order to better understand the structure of the device. The data presented are true for GPUs of compute capability (CC) 2.x and above. Older GPUs had slight variations of this general model, such as the division of warps in two half-warps.

4.1.1 Threads, Blocks and Grids

The basic unit to processing is the thread. The thread can be perceived as a set of instructions to be executed by the assigned core. A GPU, as any other processing unit, can handle a fixed maximum number of threads at a time. This maximum number of threads is determined by the capabilities of the GPU and cannot be more than the number of available cores.

The threads are organized in groups called blocks. A block cannot contain more than 1024 threads, and the optimal number of threads per block are determined by the demand in registers a thread has, the number of cores available and the nature of the algorithms used. When a kernel is launched the user sets the number of threads in a block and the number of blocks. This structure of blocks and threads is referred to as a grid. The grid and the blocks can be 1D, 2D or 3D, with different number of dimensions having advantages in different cases. The threads are assigned a unique ID in the block they belong and the blocks in the grid for easier identification, facilitating the access of thread specific data and the execution of thread specific instructions.

The instruction execution model adopted by NVidia for their GPUs is that of "Single Instruction Multiple Data" (SIMD). The threads of a block are grouped together in "warps" of 32 threads. The threads belonging to a warp must execute the same instruction simultaneously, but the data onto which the threads operate can differ. This model of instruction execution is ideal for massively parallel applications and the simplicity of the hardware implementation allows for greater efficiency and enhanced performance. On the other hand, the SIMD model can have some drawbacks in certain applications. For example, when an *if* statement is introduced in the application and differentiates the execution flow of the threads in the same warp to two blocks, the threads executing the second block will stall until the threads executing the first block have finished, affecting the performance significantly.

More specifically, in the case of adaptive integration an *if* statement is used to check the convergence and exit the recursion as seen in chapter 3. This means the threads of the induction factors calculation kernel that converge fast will wait for the threads with more demanding tasks. For this reason, the calculation of the singular integrals in the diagonal of the induction factors matrix were calculated separately.

4.1.2 Streams

To improve concurrency, NVidia added streams to the programming model of CUDA. A stream is a sequence of operations that execute in issue-order on the GPU. In this way, two or more different kernels can run simultaneously on the GPU and the operations from different streams may be interleaved. This means that in the case that a kernel call does not saturate the GPU computational capacity, some operations can be executed concurrently to reduce the total time needed for the total operation.

4.1.3 General Architecture of modern NVidia GPUs

As previously mentioned, the GPU may have thousands of cores. In order for the GPU to assign tasks to each of them efficiently the cores are organized in Streaming Multiprocessors (SMs) and the SMs are grouped in Graphics Processing Clusters (GPCs).

Each SM also contains Special Function Units (SFUs) used to evaluate some common mathematical functions. The SFUs only support single precision floating point arithmetic at the moment. Functions evaluated with SFUs are approximations and have limited precision as documented in ([49], table 7) and therefore are not fit for scientific work. NVidia provides a Math API that implements the C Math library for the GPU.

The SM also contains Texture Units or Texture Mapping Units (TMUs) responsible for transformations of bitmap texture images, such as rotation, resize and distortion. The TMUs are used in graphics applications to map textures on a given 3D geometry. For general purpose computing those hardware modules can be used to apply transformations to data. Also the interpolation of data is implemented in TMU hardware, rendering them suitable for approximating arbitrary functions with data tables.

The tasks are assigned to cores by the warp schedulers. Each SM can have multiple warp schedulers. The warp scheduler can dispatch two independent instructions per warp simultaneously.

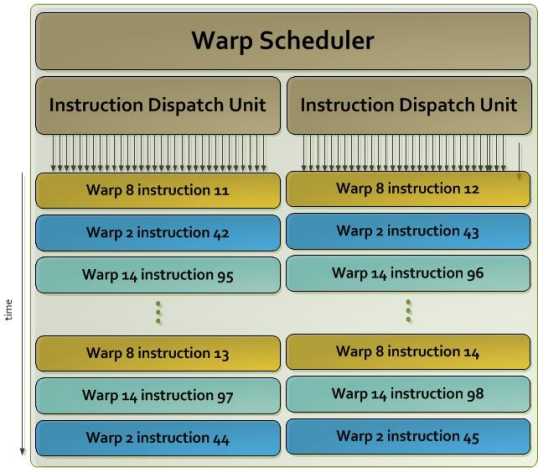

FIGURE 4.1: Warp scheduler and instruction dispatch units issuing instructions to warps

4.1.4 Memory types

There are many different types of memory available to a GPU program, each with advantages for various applications. Based on thread visibility, memory spaces can fit in one of the following three categories:

- Per-thread private local memory
- Per-block shared memory
- Per-application context global memory

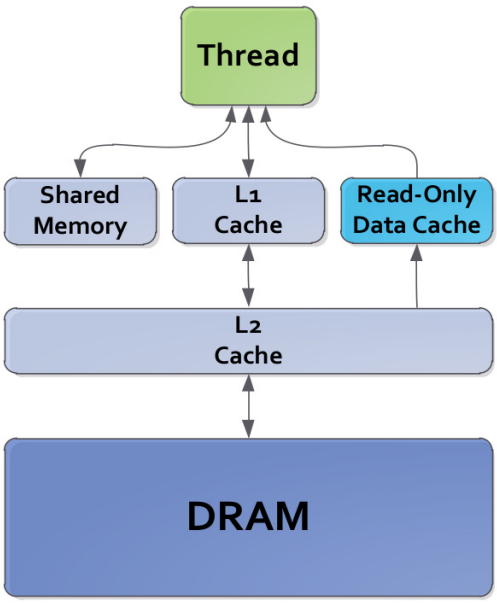

FIGURE 4.2: Memory hierarchy and layers [50].

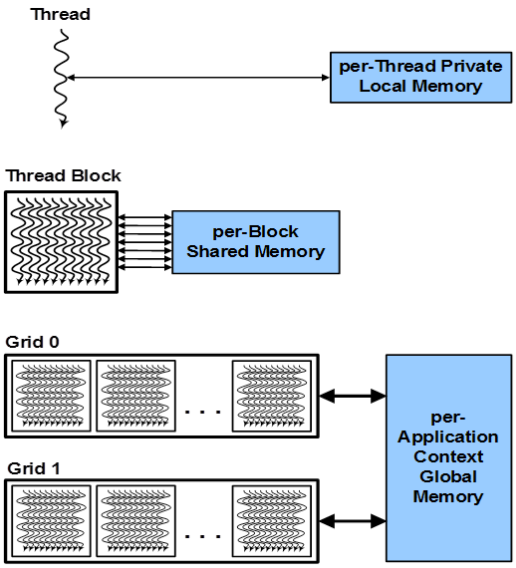

FIGURE 4.3: Hierarchy of threads, blocks, and grids, with corresponding per-thread private, per-block shared, and per-application global memory spaces [50].

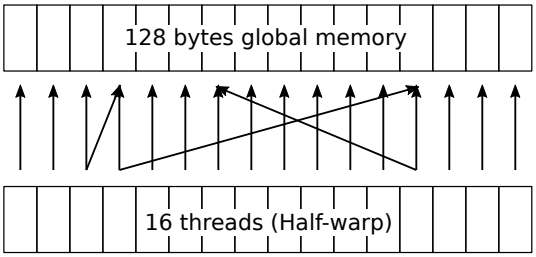

FIGURE 4.4: Example of efficient global memory access pattern.

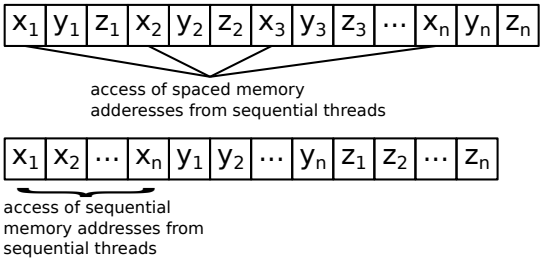

FIGURE 4.5: Example of two different models for storing elements of a vector array.

Global memory

The global memory is the main memory of the GPU, accessible by all the threads. The read or write operations from or to the global memory are expensive in time, taking some hundreds of clock cycles.

The GPU coalesces the load and store operations issued by threads within a warp to the global memory into as few transactions as possible, minimizing the device main memory bandwidth requirements. The warp perceives the global memory space as a series of 32, 64, or 128-byte partitions of data addresses that can be accessed in one operation. Therefore, when the addresses of the data to be read or written by the warp are spread in as few of these memory chunks as possible the access time will be optimal. While the access pattern is crucial for older GPUs, newer models allow more flexible access patters, such as the arbitrary access order within a warp as seen in figure 4.4.

For example, when storing the elements of a 3D-vector array, as is the boundary geometry in BEM, the programmer can choose to store the elements of each vector together in triplets or store the elements in one direction of all the vectors together (see fig. 4.5). In order to make a choice the memory access patterns should be examined. In the first case, the memory addresses that must be accessed in order to get the first elements of each vector from threads with sequential IDs are spaced so that there are two memory addresses between the elements. In the second case the memory addresses are sequential. As an effect, the width of the memory partition that has to be accessed by the first implementation is three time as long as the one needed in the second implementation. If there are enough registers available for all the active threads, the global memory access will happen only the first time the data are needed and then stored in the register file, reducing the latency significantly the next time the specific data are needed.

Shared memory

The shared memory is part of the same memory space as the L1 cache, giving it the same performance. Shared memory can be accessed by all the threads in the same block, allowing to exchange data between them. The size of the memory space where shared memory and L1 cache reside can be split between them based on the programmer's preferences. The programmer can usually select during runtime between modes that are different for different architectures. For example, for the GK110 Kepler architecture, out of the 64KB memory space, the programmer can select to split it as 48KB/16KB, 16KB/48KB or 32KB/32KB of shared memory and L1 cache [50].

The shared memory can be randomly accessed by the threads of the same block. It can be used to share common data among threads, eliminating the need to compute mentioned data for each thread separately or using the slow global memory. Special care should be taken to avoid data races and for that reason synchronization of threads should be implemented in the application when needed.

Local memory

The local memory is accessible by only one thread and it is part of the same memory space as the global memory. It is usually used when the demand of registers is high and cannot be satisfied by the hardware. As it is evident, the local memory is characterized by slow read and write operations and the same access patterns as the global memory. The local memory is structured in the way that minimizes the number of memory partitions a warp accesses, as the optimal standard dictates. The access of the local memory is accelerated by the L1 and L2 cache.

4.1.5 Special memory types

There exist two special memory types in a GPU, the texture memory and the constant memory. Initially, they were designed to store textures and constants, for graphics generation. Both constant and texture memory are referred to as read-only memories and are cached on chip. Of course, while these memory types are optimized for specific uses, when there is a cache miss they are very slow as the data must be read from the device global memory.

Texture memory

Texture memory is a special type of memory that resides in the device global memory space. An interesting feature of the texture memory is that it can be linearly interpolated in hardware. This means that given an non integer memory address the value returned by the access operation is the linear combination of the two values nearest to the "address" given.

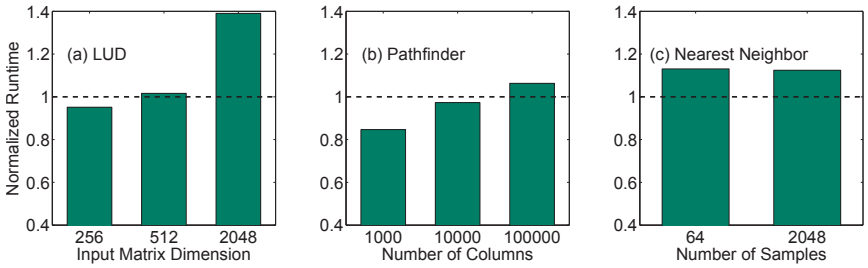

FIGURE 4.6: Performance results of the Unified memory access (UMA) version of the Rodinia[1]benchmarks normalized to the non-UMA version [34].

Constant memory

Constant memory also resides in the device global memory. It is optimized for when a warp of threads requires access to the same memory address. When this condition is not satisfied, the performance decreases significantly as the access is serialized.

4.1.6 CPU-GPU communication

Unified memory address space

Nvidia has introduced the unified memory technology to facilitate the development of HPC codes. This technology allows the use of RAM and device memory (even for multiple GPUs) in a unified address space that hides the data migration from and to the RAM. The data transfer is managed by the CUDA API and as a result the programmer does not need to devote time into optimizing this process accelerating development.

4.1.7 GK106 Kepler Architecture

The GK106 die consists of 3 GPCs, two containing 2 SMs and one with one SM, as seen in figure 4.7. Each SM has 192 cores, 32 SFUs, 32 LD/ST modules, 16 texture units and 4 warp schedulers.

4.1.8 GP107 Pascal Architecture

The GP107 die consists of 2 GPCs, containing 3 SMs each, as seen in figure 4.8. Each SM has 128 cores, 16 SFUs, 16 LD/ST modules, 8 texture units and 4 warp schedulers. The GTX 1050 model uses the GP107 architecture but has one SM inactive, with a total of 640 cores.

[1]Rodinia [11] is a benchmarking suite to measure the performance of different heterogeneous computer architectures and program design.

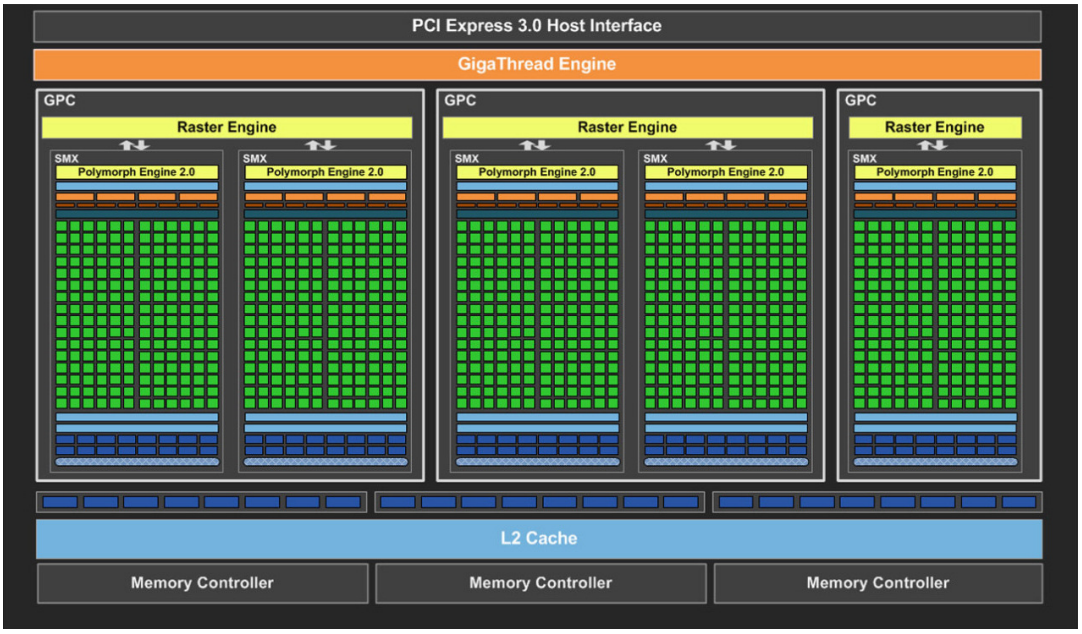

FIGURE 4.7: GK106 Kepler architecture.

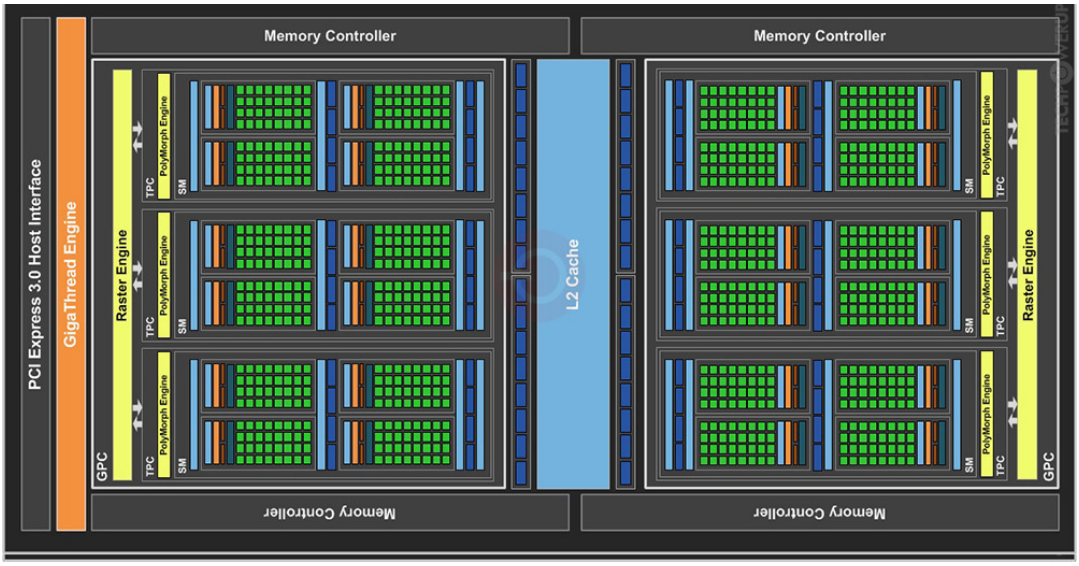

FIGURE 4.8: GP107 Pascal architecture.

4.2 Parallelization techniques for GPUs

The parallel algorithms can be divided into two categories, based on hardware limitations. The memory bounded algorithms are those that saturate the bandwidth of the memory bus, demanding higher data transfer speeds from the memory to the ALU than the hardware can handle. As a result the ALU must wait several clock cycles idle, while the needed data are fetched and consequently the execution time is negatively affected. In case of such an algorithm special attention must be paid during the programming phase to minimize the access to the memory.

The other category is that of computationally intensive algorithms. These algorithms require a lot of computations with a small amount of data. This means the memory access time is small compared to the computation time and therefore the performance of the algorithm is impacted only by the number of floating point operations needed to complete the task.

Performance wise, the manner one has to program a GPU application and things one has to be cautious about depend on whether the algorithms used are memory bounded or computationally intensive.

4.3 The Algorithm

In algorithm 2 the time marching procedure is presented. First, the parameters of the simulation are determined by reading the configuration and job files. In the configuration file, parameters concerning the GPU and the precision are defined. In the job file there are parameters that define the foil geometry, the temporal and spatial discretization and the oscillation.

Second, the geometry is initialized and discretized. Here the bilinear elements of the foil are generated and the elements of the Kutta strip are created.

For each timestep the wake is moved according to the background velocity field and expanded by one element near the Kutta strip. Next, the state of the foil (position and velocity) are approximated by the means of an implicit Euler scheme, the hydrodynamic force acting on the foil is calculated and the validity of equation 2.79 is checked. If equation 2.79 does not hold, the Newton Method is used to refine the state so that the equation is satisfied within a small tolerance. Finally the values of the time derivatives is calculated and the results are appended on the result file.

On the right hand side of the linear system (2.53) there are terms that don't change over time for a rigid boundary and terms that depend on previous timesteps. This means that the constant terms can be calculated when initializing the geometry and update only the values of the terms needed. The terms that need to be changed are these related with the moving wake and should be calculated on each timestep and each iteration of the Newton Method.

4.4 Parallelization of BEM

Concerning the Boundary Elements Method, most of the calculation in a step of the solution are independent of one another. The computation of matrices A and B

Algorithm 2 Time marching algorithm

```
1: procedure TIME MARCHING
2:     initializeSimulationParameters()
3:     initializeGeometry()
4:     initializeDtN()
5:     for i = 1(1)N_timesteps do
6:         moveWake()
7:         expandWake()
8:         S[i] ← approximateStateWithImplicitEulerMethod(S[i − 1])
9:         F ← calculateForce(S[i])
10:        e ← calculateError(F)
11:        while e ≥ tolerance do
12:            S[i] ← refineStateWithNewtonMethod(S[i])
13:            F ← calculateForce(S[i])
14:            e ← calculateError(F)
15:    updateTimeDerrivatives()
16:    writeTimestepResultsToFile()
```

Algorithm 3 DtN matrix initialization algorithm

```
1: procedure INITIALIZEDTN
2:     for i = 1(1)DoF do
3:         for j = 1(1)DoF do
4:             if i == j then
5:                 A_ij ← slIntegrate<double>(Element_i, CollocationPoint_j)
6:                 B_ij ← dlIntegrate<double>(Element_i, CollocationPoint_j)
7:             else
8:                 A_ij ← slIntegrate<float>(Element_i, CollocationPoint_j)
9:                 B_ij ← dlIntegrate<float>(Element_i, CollocationPoint_j)
10:    N ← getNeumannData()
11:    LHS ← 0.5 · I + B
12:    RHS ← A · N
13:    DtN ← LHS^−1 · RHS
```

Algorithm 4 Hydrodynamic force calculation algorithm

```
1: procedure CALCULATEFORCE(S)
2:     updateElementsPosition(S)
3:     updateNeumannData(S)
4:     updateDtN()
5:     D ← getDirichletData()
6:     Ḋ ← calculatePotentialTimeDerivative(D)
7:     V ← calculateVelocities(D)
8:     P ← calculatePressure(Ḋ,V)
9:     F ← calculateForce(P)
```

Algorithm 5 DtN update algorithm

```
1: procedure UPDATEDTN
2:     LHS ← LHS_init + morinoTerms()
3:     RHS ← RHS_init + wakeTerms()
4:     DtN ← LHS^−1 · RHS
```

in equations 2.50 and 2.51 respectively need only the geometry of the foil and are independent among the elements, and as a result they can be executed in parallel without any complication. Figure 4.14 shows the speed-up a GPU can produce for the computation of the discrete Dirichlet to Neumann (DtN) map for the most demanding application of double-precision arithmetic. It can be seen that the time for parallel execution on the GPU can be more that 44 times lower than the single thread execution on the CPU.

4.5 Algorithm optimization and hardware awareness

4.5.1 GPU occupancy

In order to achieve the maximum computational capacity of the GPU all the available CUDA cores must be used. There are many parameters that must be determined to accomplish the maximum occupancy of the GPU [2].

First of all, the workload of the threads within a block must be balanced. This means that the threads should take the same time to finish the scheduled work. If some threads need more time to execute, at the end of the kernel there will be less active warps, using only a subset of the cores.

The same is true about the unbalanced workload of blocks within a grid. In this case, limiting the size of the blocks and distributing the workload to more blocks will increase the efficiency of the algorithm, because new blocks will be launched as the previous ones will finish their execution.

Another issue that can have an impact on the occupancy is when too few blocks are launched. If the number of blocks launched is less than the maximum number of blocks that can run simultaneously across all SMs then the occupancy is limited. The maximum number of blocks that can run at once on the GPU is called a "full wave". When a kernel is launched with more blocks that a full wave then the blocks that cannot fit into an SM will be scheduled for after another block has finished.

Moreover, each thread of a block will use the registers available to that block. This means that the limited register file must be adequate for all the threads of the block. To achieve this the block size may need to be decreased or the register count per thread my be limited at compilation.

Finally, the last wave of operations scheduled may be partial and thus limit the occupancy. This means that for a problem that needs the last wave to be full, if other resources are adequate, the GPU will be used more efficiently than a problem that will not occupy completely the GPU for the final block executions. Of course, this is inevitable and generally in large problems the last wave takes a small portion of the total time as it can be seen in figure 4.11.

Taking into consideration all the above effects, the size of the block and the number of registers allocated to each thread should be determined for the hardware the code is run on. To determine these factors, the NVidia occupancy calculator can be used. For the card with compute capability (CC) 6.1, to achieve maximum warp occupancy on each SM (64 warps/SM), the register count for each thread are limited to 32. As it can be observed in figure 4.10 this value is the maximum number of registers that does not decrease the number of warps that can run simultaneously. Then, a set

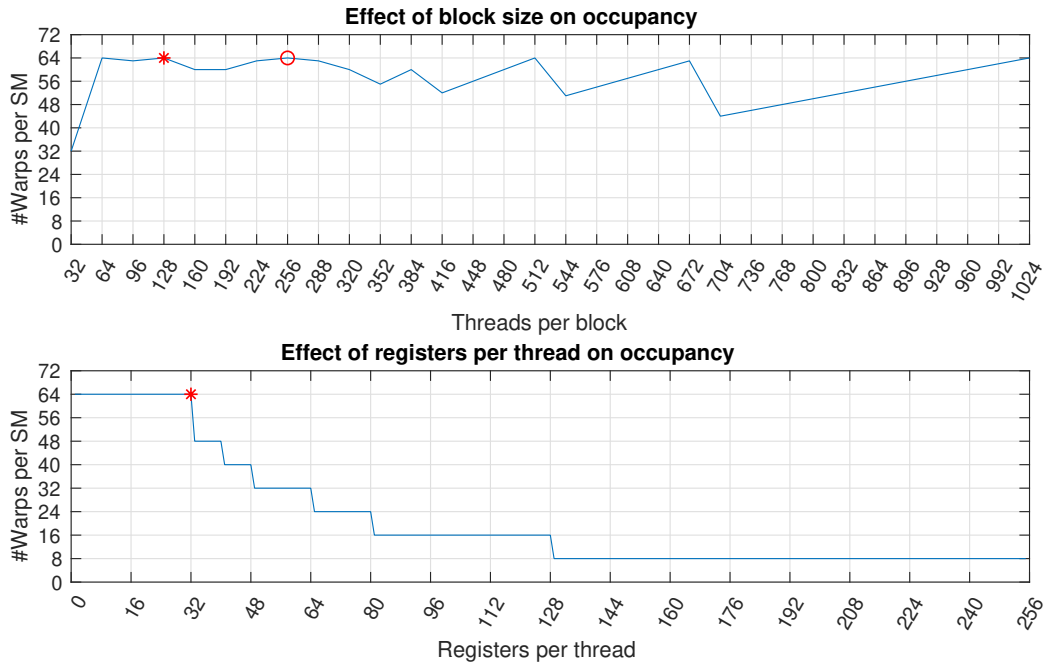

FIGURE 4.9: Effect of block size and register count per thread on GPU occupancy for CC=6.1.

of block sizes that maximize the occupancy can be found and the block size of 128 threads is chosen for 1D block and 256 threads for 2D blocks. The number of blocks is then enough to occupy all the SMs. In the developed code no shared memory is used and therefore it's effect on occupancy will not be examined.

In figure 4.11 the effect of the number of degrees of freedom introduced to the problem of building the Induced Factors Matrix on the last wave size is presented. The number of threads launched is the number of DoFs squared (equal to the number of elements in the matrix). It can be seen that for a small problem size, the average utilization of the GPU can become low and, as the problem becomes bigger, the utilization asymptotically approaches the ideal value.

4.5.2 Mixed precision arithmetic

Most commercial GPUs have more cores capable of single precision operations than double precision. This leads to a higher total throughput of single precision calculations, meaning that the same problem can be solved faster using single precision floating point arithmetic. For example, the NVidia Pascal architecture SM has 128 FP32 (single-precision) cores but only 4 FP64 (double-precision) cores.

Furthermore, storing large matrices in single precision can reduce significantly the memory usage of the software. In this way, larger problems can be solved entirely on the GPU without the need of data transactions between the RAM and the device memory.

Of course some calculations (e.g. multiplication of a very large number with a small one) need double precision in order to be evaluated correctly. So, a mixed precision arithmetic scheme should be devised to achieve the best performance possible.

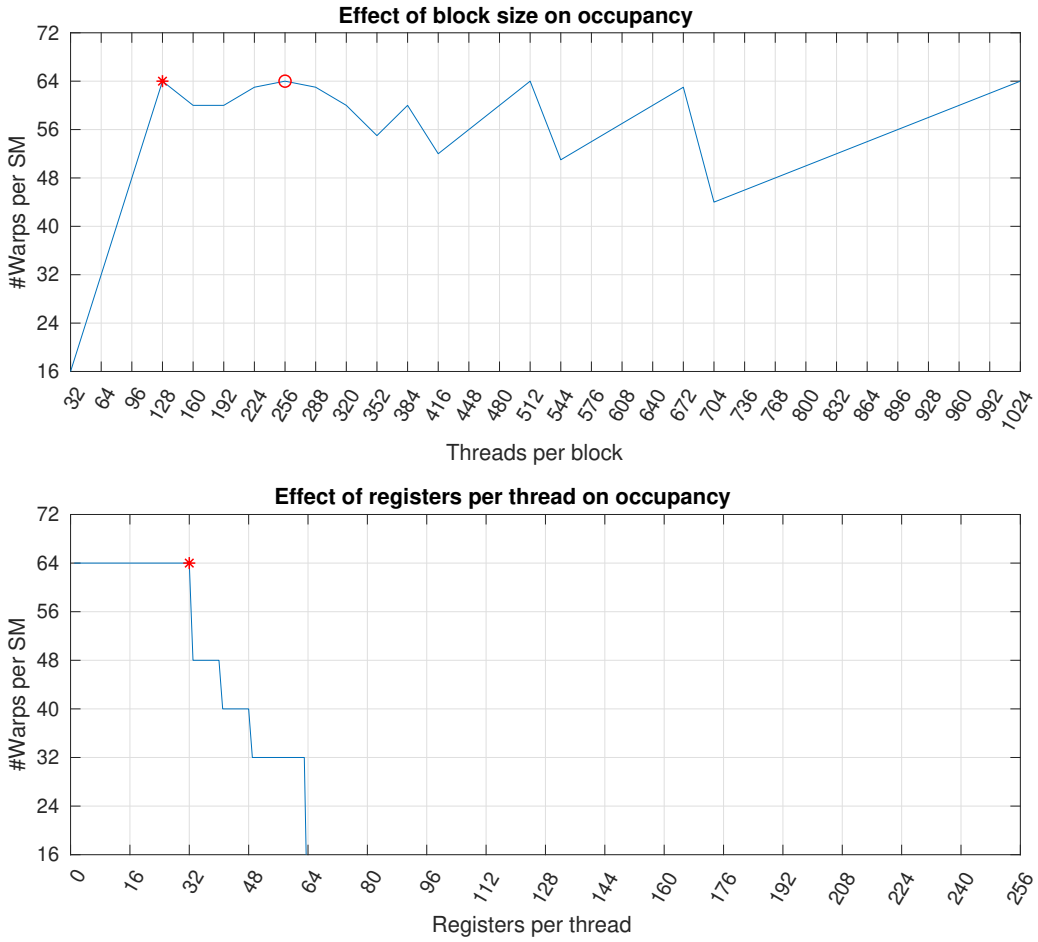

FIGURE 4.10: Effect of block size and register count per thread on GPU occupancy for CC=3.0.

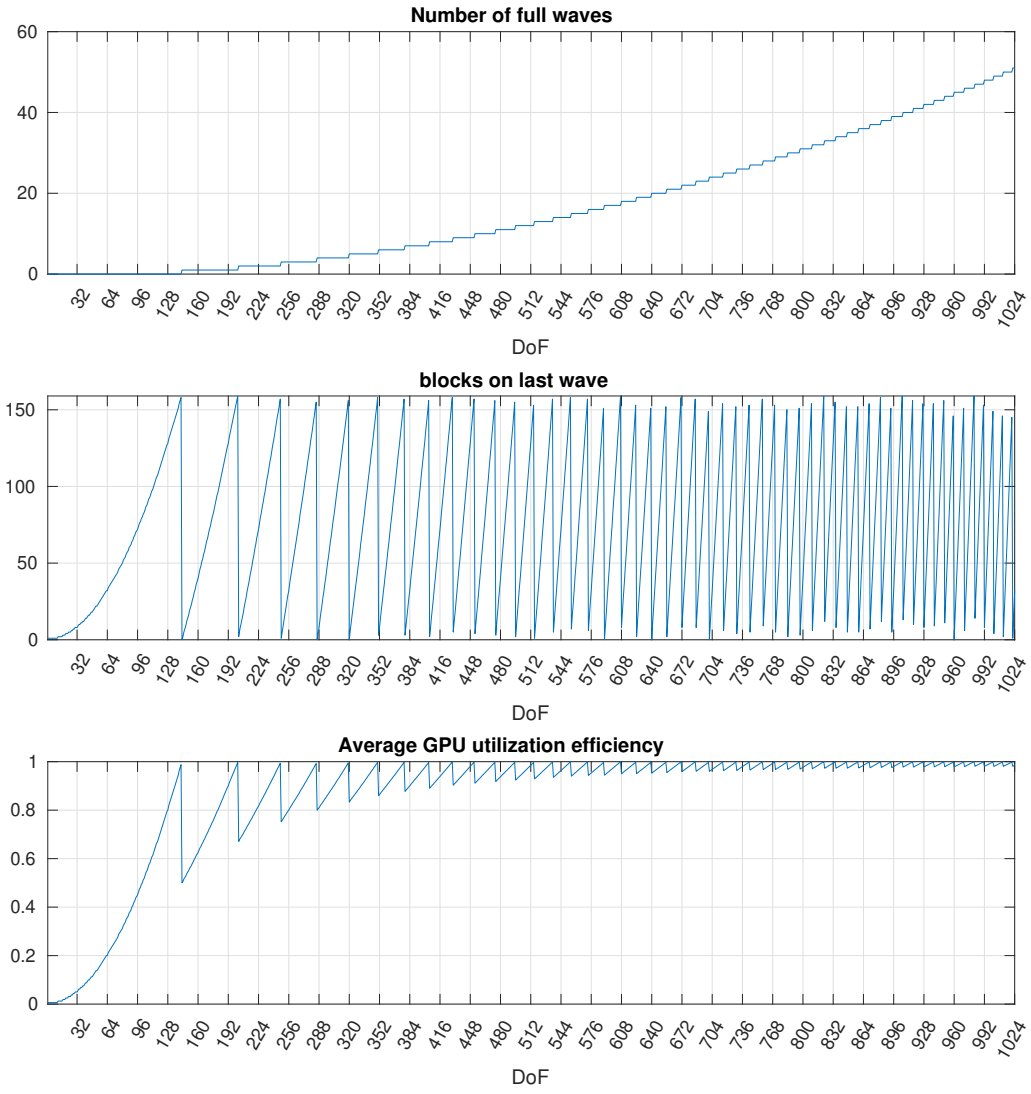

FIGURE 4.11: Effect of problem size (DOFs) on GPU occupancy for block-size=256, 5 SMs and a maximum of 32 blocks per SM.

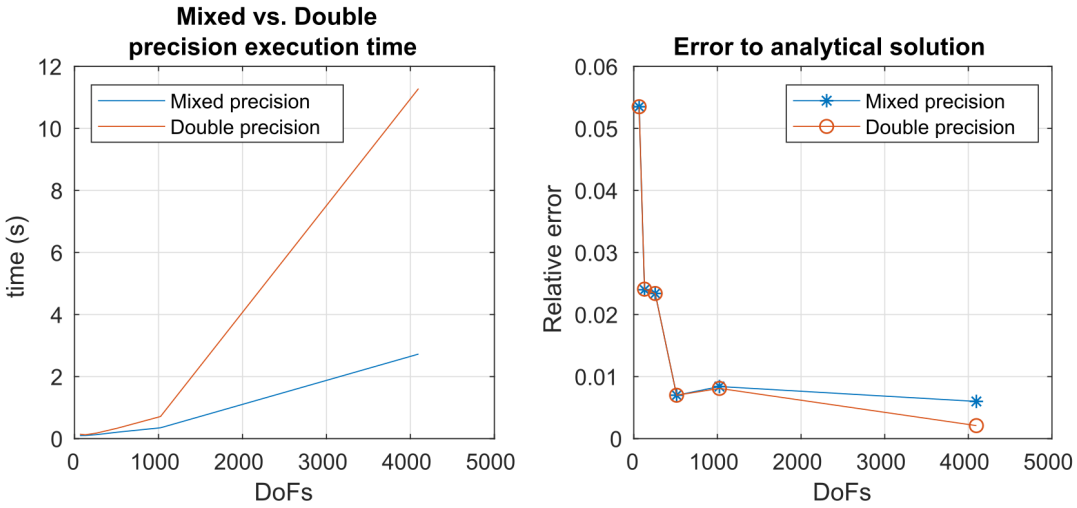

FIGURE 4.12: Execution time comparison and relative error for double and mixed precision on GTX 650 Ti BOOST graphics card. The results are compared to the analytical solution [42].

In the present work, the mixed precision scheme is implemented as proposed by Trompoukis [65] for a data intensive CFD application and used for a compute-intensive BEM method by Filippas [15]. In the case of the Boundary Element Method, the calculation of the singular integrals, for the diagonal elements of matrices A and B, that appear in the evaluation of the self-induced potential demand double-precision arithmetic. The results are then converted to single-precision and stored together with the induced potential factors calculated with single-precision arithmetic. The solution of the linear system is then performed in single-precision.

Extensive use of templates allows for fine control over precision. In this way a precision scheme can be found that balances computation time and solution error. In figure 4.12 the execution times and the error for double precision and mixed precision arithmetic are compared. For the error evaluation, the results are compared to the analytical solution [42]. From those figures can be inferred that for an insignificantly greater error mixed precision can decrease execution times threefold.

4.5.3 Integrators

The BEM algorithm is parallelized in a coarse level and thus the adaptive integration routine is not parallelized. Newer GPU architectures allow for recursive kernel launching, making the parallelization easy. But executing the adaptive algorithm on multiple GPU threads would not give a considerable advantage. Such an algorithm would recursively launch new kernels of small size adding a significant amount of overhead to the process.

It should be noted that in order for the adaptive integration to work properly the stack size of the GPU must be increased to accommodate as many recursion levels as needed by the adaptive routine.

Functors were used to define the integrant functions. In this way the user can easily define an integrant function with parameters as is the case of the Green functions. Also, functors allow the definition of nested integrators. This means that one

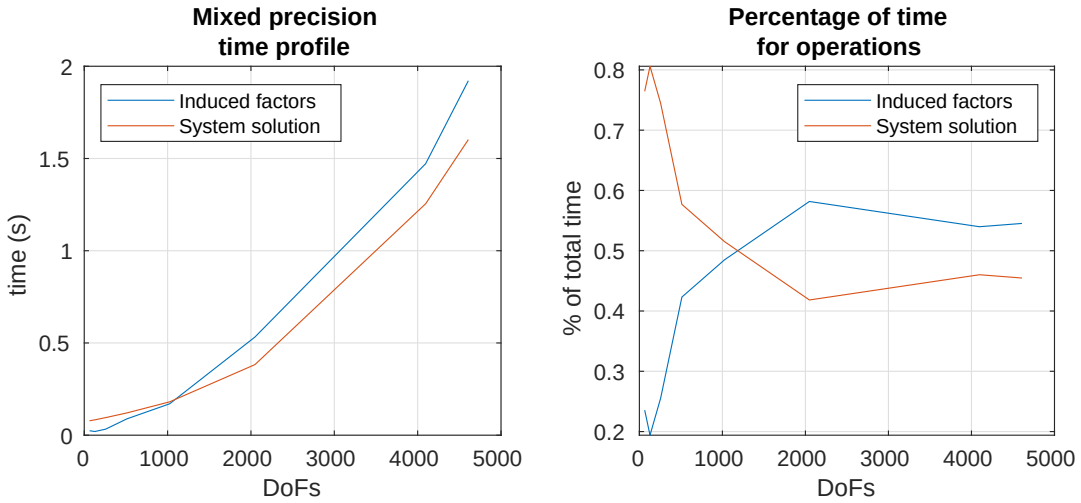

FIGURE 4.13: Profile of computation time for calculation of potential on an ellipsoid on GTX 650 Ti BOOST.

integrator can be used as the integrant function of another integrator allowing to evaluate 2D surface integrals easily without re-implementing the integrators.

For the calculation of self-induction factors, a semi-analytical method was used.

4.6 Performance analysis

The most time consuming operations in the solver are the calculation of the induction factors and the solution of the produced system of equations. In figure 4.13 the time needed to execute each operation is plotted against the number of degrees of freedom. It can be observed that for a few degrees of freedom the solution of the system takes slightly more time. As the problem size increases the calculation of induction factors becomes the most demanding operation, taking 58% of the total time. This means that the effort to augment the performance of the implementation should be concentrated on developing faster techniques for the calculation of the induction factors. One such method is to approximate far-field interactions by point sources and dipoles rather than distributions (see the Fast Multipole Method [47]).

Due to the use of an adaptive algorithm for integration, a performance penalty is introduced, as some threads will integrate near-field induction factors and other threads will integrate far-field induction factors. The integrant of the single and double layer integrals become increasingly singular when the evaluation point approaches the surface of source or dipole distribution. As a result, the adaptive routine will converge more slowly for the calculation of near-field induction factors making the thread execution time range heavily. When threads of each category are placed in the same warp the block execution time will be the maximum of the execution times of the threads in the block, making the faster threads become idle until all threads have finished. This problem can be partially tackled by sorting thread IDs based on the distance of the evaluation point from the surface. In this approach, however, extra computation time is needed to sort the threads.

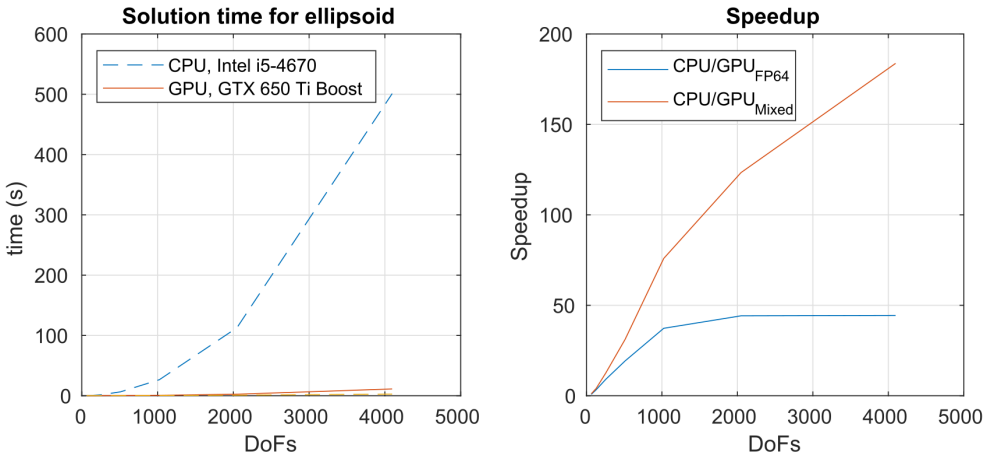

FIGURE 4.14: Comparison of CPU (Intel i5-4670 3.4GHz) single-thread performance with GPU (NVidia GTX 650 Ti BOOST) parallel performance for the calculation of potential on an ellipsoid.

In order to evaluate the performance of the GPU code, the execution times of CPU and GPU implementations for the calculation of the potential on an ellipsoid are compared as seen in figure 4.14.

Chapter 5

Numerical Results

5.1 Convergence study

It is customary to begin the numerical study by proving the convergence of the method as both spatial and temporal discretization becomes finer. For this purpose, numerical experiments can be conducted for a foil in steady motion. The foil starts moving smoothly in the forward direction with a constant angle of attack and retains a constant velocity (U) for ten chord lengths of forward motion. In figure 5.1 the position of the foil and it's velocity are plotted against time. The velocity of the foil is given by the formula

$$u(t) = U(1 - e^{-ft^2}) \tag{5.1}$$

where t is the time and f is a filter parameter taken equal to 2.

Next, for the spatial convergence the foil was discretized in grids of N elements per dimension for $N = \{8, 16, 32, 64\}$. The convergence of the non-dimensional coefficients C_L, C_D and C_M can be observed in figure 5.2.

The same procedure was followed with time discretization. The simulation was repeated with timesteps $\Delta t = \{0.025, 0.05, 0.01, 0.1, 1\}sec$. The convergence of the non-dimensional coefficients C_L, C_D and C_M as the timesteps become smaller can be observed in figure 5.3.

In figure 5.4 the convergence of the Morino-type Kutta condition to the Pressure-type is shown as the motion of the foil becomes steady.

In figure 5.5 the convergence of the efficiency of the oscillating foil is shown for the case of $AR = 10$, $c = 1$, NACA0005 cross-section, $a/c = 0.5$, $f^* = 1/\pi$, $V_{bg} = 1$, $\theta_0 = 10^o$, $b = \pi\rho csV_{bg}$, $k = 0$ and $m = 0$.

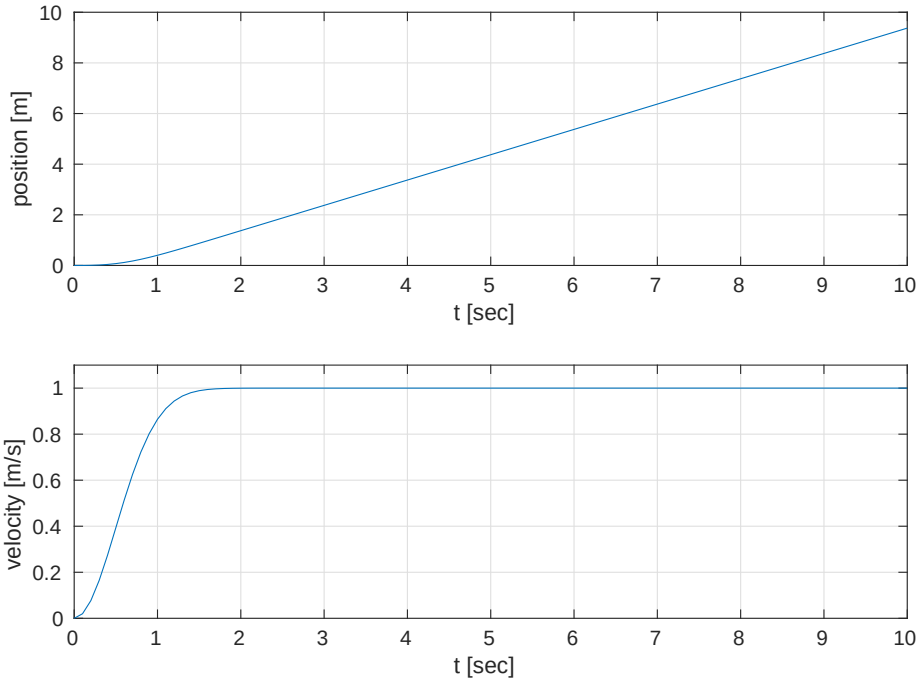

FIGURE 5.1: Motion of foil for the convergence study for $f = 2$ and $U = 1$. Time step is $\Delta t = 0.04sec$.

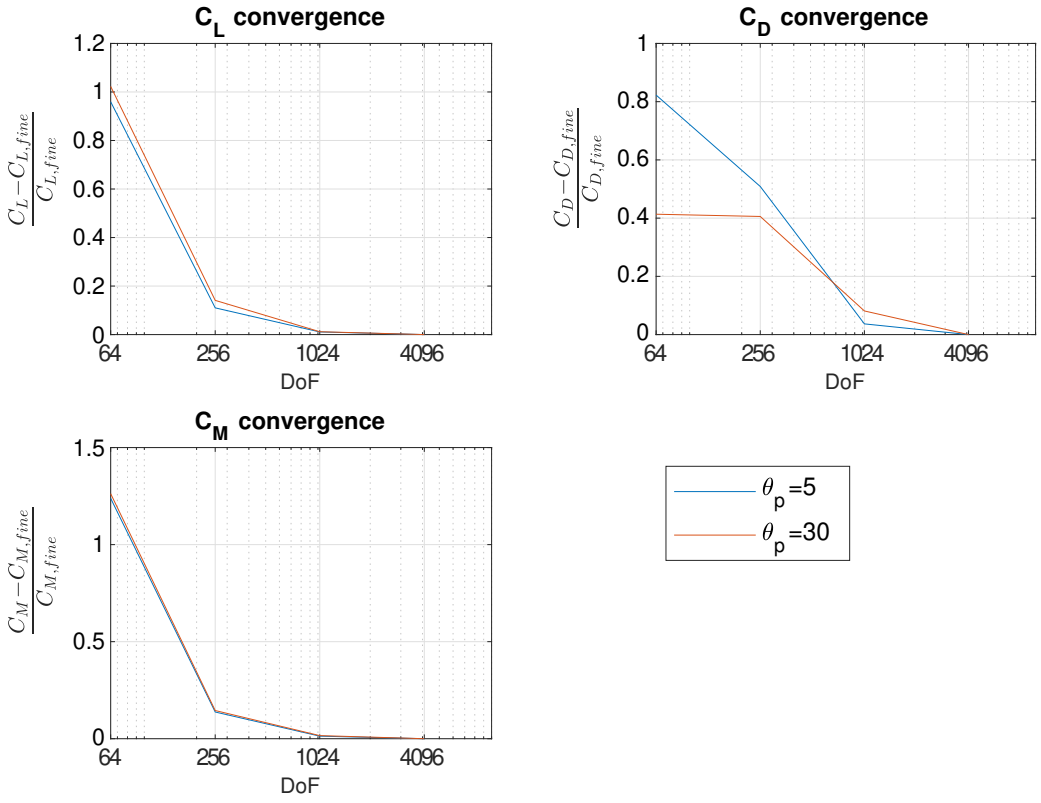

FIGURE 5.2: Convergence of C_L, C_D and C_M for finer grids

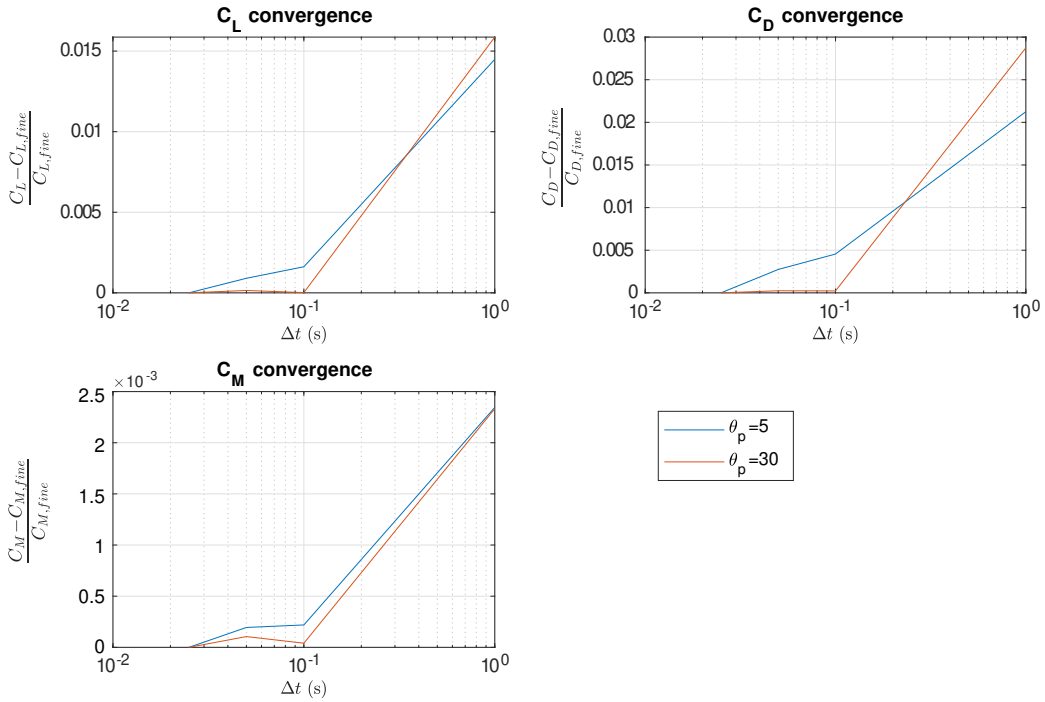

FIGURE 5.3: Convergence of C_L and C_D for smaller timesteps

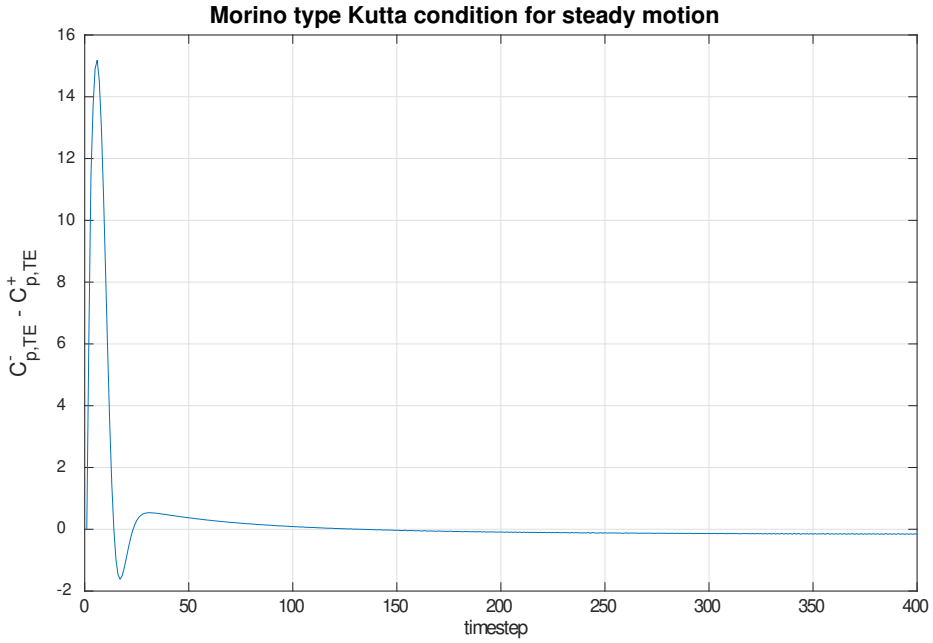

FIGURE 5.4: Morino-type Kutta condition convergence to pressure-type for steady flow.

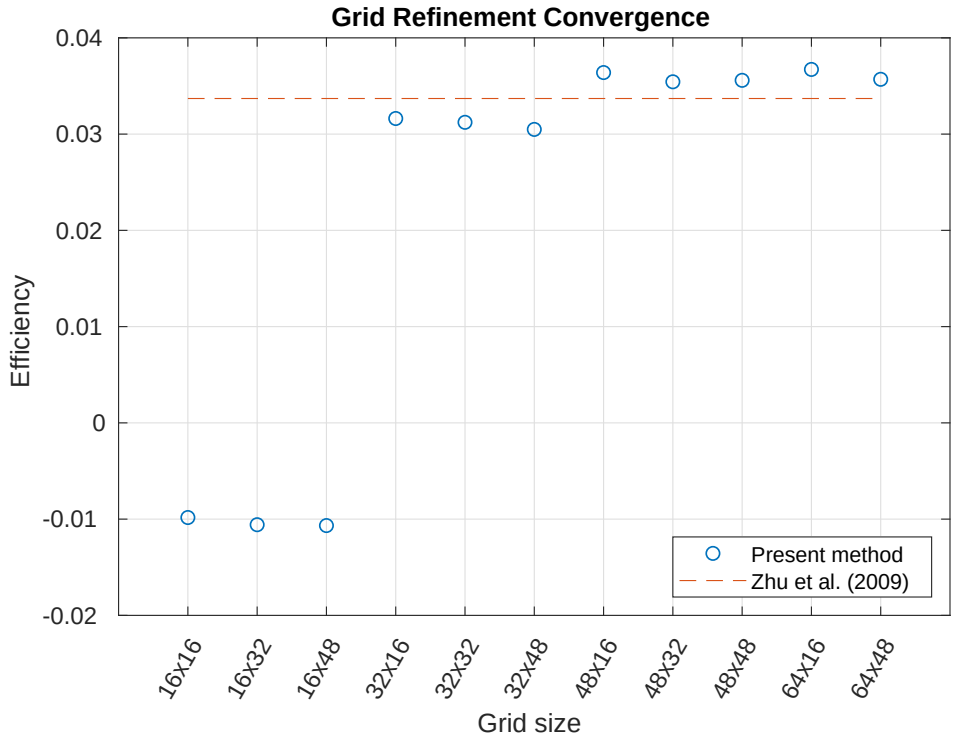

FIGURE 5.5: Convergence of efficiency with grid refinement. The first number in grid size is the number of elements in the chord-wise direction and the second is the number of elements in the span-wise direction.

5.2 Validation of present method with experimental data

In this section the experiment described on [25] is replicated with a computer model and the results are compared with the original experiment. In this way, the accuracy of the method presented in chapter 2 is tested and the limitations of it can be shown.

The hydrofoil is attached vertically, at about 1/4th of its chord, to the end of an arm of 0.3m length, that is free to rotate around the other end. The lift and drag of the foil create a moment at the axis of rotation of the arm and a sensor is placed at the shaft to measure this moment. The hydrofoil section has the NACA0012 geometry with span equal to $s = 0.34$ meters and chord $c = 0.1$ meters. The aspect ratio of the foil is therefore $AR = 3.4$. Care was taken to place the edges of the foil near the ceiling and floor of the water tunnel in order to reduce the tip vortices. This is an attempt to reduce the 3D effects on the flow and increase the efficiency of the oscillation. Two identical torque transducers are used to measure the torque needed for the pitching motion and for the torque exerted on the rotation axis of the arm by the hydrodynamic forces acting on the foil. A dashpot is placed on the arm shaft to dampen the motion and act like a generator.

Following the design of the experiment above a series of numerical simulations was performed. In figure 5.6 the heaving amplitude of the pitch center of the foil is plotted for different reduced frequencies and various pitching amplitudes (ranging from 10 deg to 60 deg). The foil used in the simulations is of large aspect ratio to counteract the effect the plates near the wingtips have on the wake. A deviation from the

experiment is apparent but it is expected as the method described in chapter 2 considers only a perfect fluid with irrotational flow and applies the Morino condition on the TE. For large pitching amplitudes the response of the foil seems to continue increasing at a constant rate using the present method. In reality, in large angles of attack there would be flow separation occurring near the foil, a phenomenon that decreases the lifting potential of the foil.

For the reduced frequency $f^* = 0.1$ the curve produced with the method presented in 2 is very close to the experimental curve. That is because more information about the initial phase difference between the heave and pitch motions was given. Furthermore, the observed angle of attack was lower for this frequency and increased as the frequency became higher or lower.

For other reduced frequencies the phase difference was chosen by trial and error. The phase difference between the heave and pitch motions is important because any deviation from the correct value will introduce asymmetries in the hydrodynamic forces between the different phases of the oscillatory motion of the wing, affecting the behavior of the oscillating mechanism. To reduce these asymmetries a spring could be used to return the foil on the desired orbit, something not used in the experiment.

An attempt to reduce this deviation is made by the use of empirical corrections. The drag of the foil can be corrected for the skin friction resistance including the effect of the angle of attack. To achieve this, the drag coefficient of the foil can be increased by the term

$$c_r = \frac{0.0858}{[log_{10}Re - 1.22]^2} + c_\alpha(Re)\alpha^2 \tag{5.2}$$

where $Re = V_{bg}c/v$ is the Reynolds number of the experiment, v is the kinematic viscosity of the water and α is the effective angle of attack. This coefficient will increase the drag of the foil for higher values of α and as a result reduce the total torque on the shaft, limiting the heaving amplitude.

In figure 5.6 the corrected curves are plotted against the curves without corrections. It can be seen that for reduced frequencies $f^* = 0.025, 0.2$ the correction yields the desired result, but it is apparent that better and more detailed model should be used the incorporate viscous effects into the simulation. Furthermore, to acchieve more accurate results the pressure-Kutta condition should be imposed on the trailing edge of the foil.

The torque produced by the hydrodynamic forces exerted on the foil for one numerical experiment are shown in figure 5.8 as a function of time along with the corresponding power output. These quantities are compared with the data provided by Huxham et al. [25]. It is apparent that the power produced in the experiment is significantly higher that the results for the real aspect ratio of the foil. The output power increases in relation with the aspect ratio. At $AR = 10$ it can be observed the same effect as that of the plates near the wing tips at the experiment.

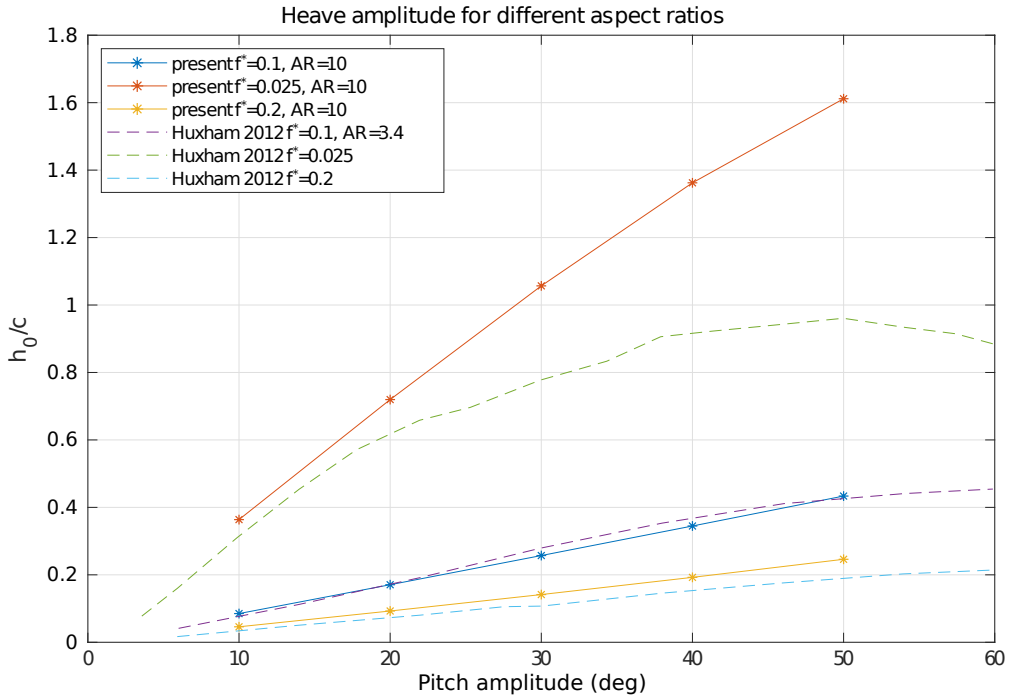

FIGURE 5.6: Heave amplitude comparison between the present method and the experiments conducted by Huxham et al. [25] for reduced frequencies $f^* = 0.025, 0.1, 0.2$. In this figure the results of present method are of a foil with large aspect ratio to approximate the effect of the plates near the wingtips.

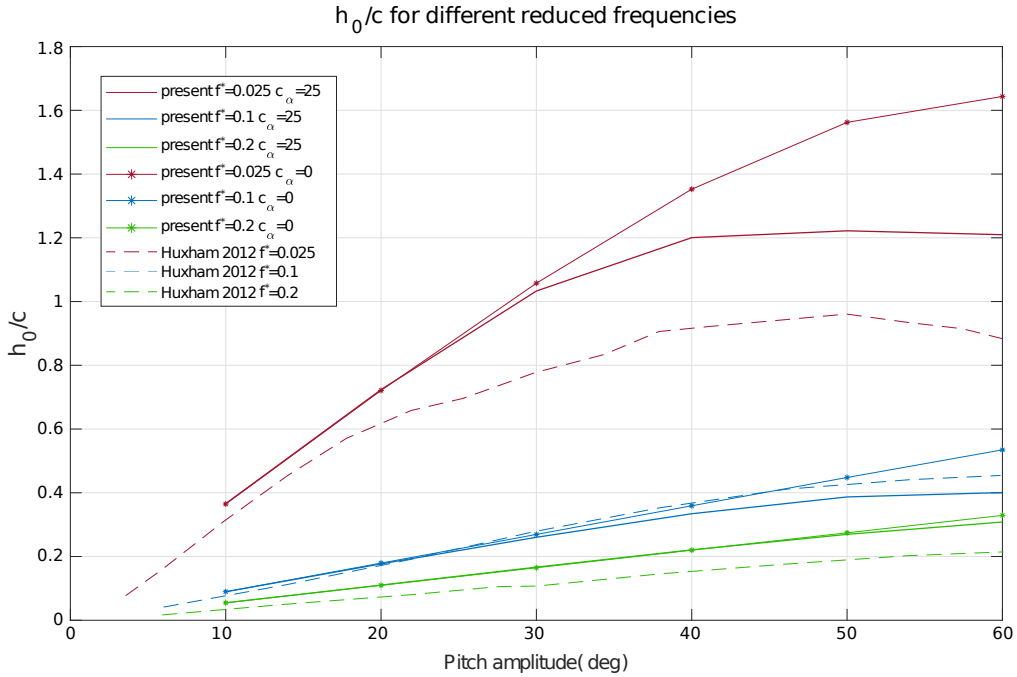

FIGURE 5.7: Heave amplitude comparison between the present method with corrections for resistance and the experiments conducted by Huxham et al. [25] for reduced frequencies $f^* = 0.025, 0.1, 0.2$.

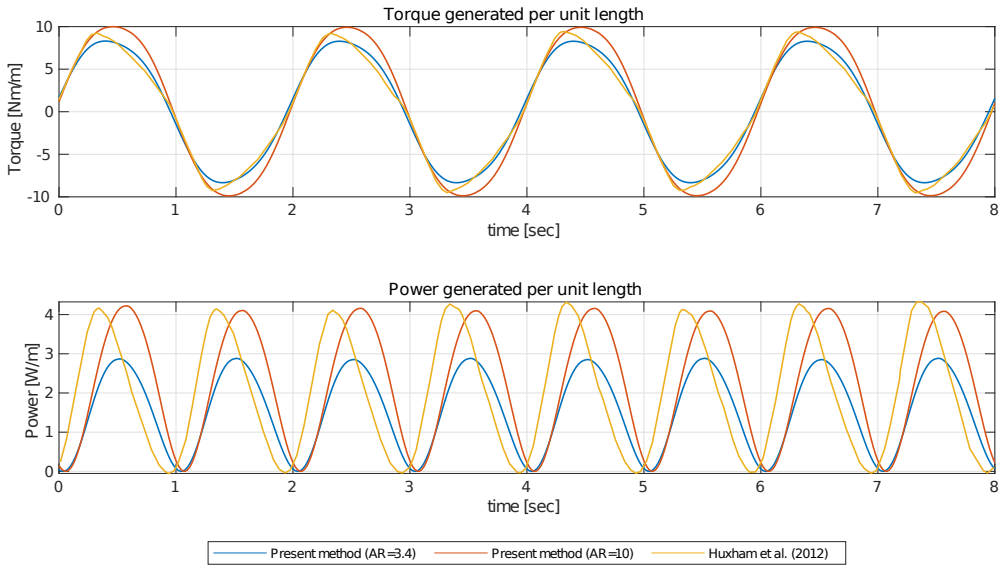

FIGURE 5.8: Torque output and power output timeseries between the present method and the experiments conducted by Huxham et al. [25] for $f^* = 0.1$, $\alpha_0 = 50^o$ and $V_{bg} = 0.5$.

5.3 Validation of fluid-structure interaction

Here the device examined consists of a foil moving linearly, instead of moving on a circle constrained by the arm as it was in the experiment described above. It is easier to examine a device like this because the initial phase between the heave and pitch positions do not introduce asymmetries in the heave oscillation. This mean that there is one less parameter that determines the quality of the simulation.

The results are compared with the numerical experiments of Zhu et al. [75]. The mass of the foil is ignored for the sake of comparison. In figures 5.9 to 5.11 the heaving amplitude, the power extraction and the efficiency of the device is drawn as a function of reduced frequency. The simulations were performed for foils of different aspect ratios and compared with the 2D and 3D results of Zhu et al. [75]. In these runs, the pitch amplitude is equal to 10^o and the damper coefficient is $b = \pi\rho csV_{bg}$, that is value that yields the theoretical maximum power extraction. The chord of the foil is $c = 1$ and the current velocity is $V_{bg} = 1$. The power is normalized to the theoretical maximum power $P_{max} = \frac{\pi}{8}\rho csV_{bg}^3\theta_0^2$ [75].

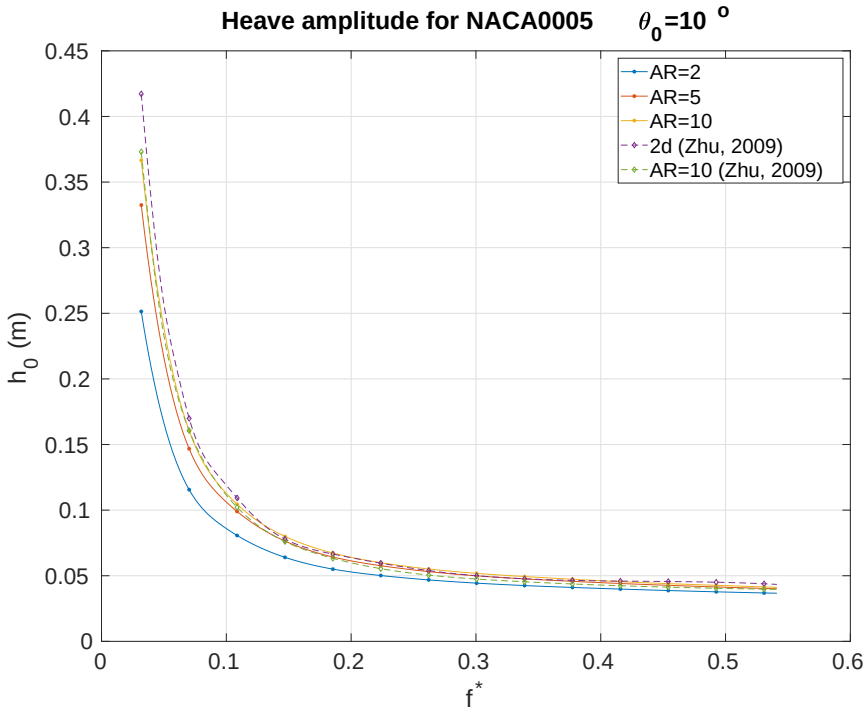

FIGURE 5.9: Effect of aspect ratio (AR) on the heave amplitude of an oscillating foil in energy harvesting mode and comparison with [75].

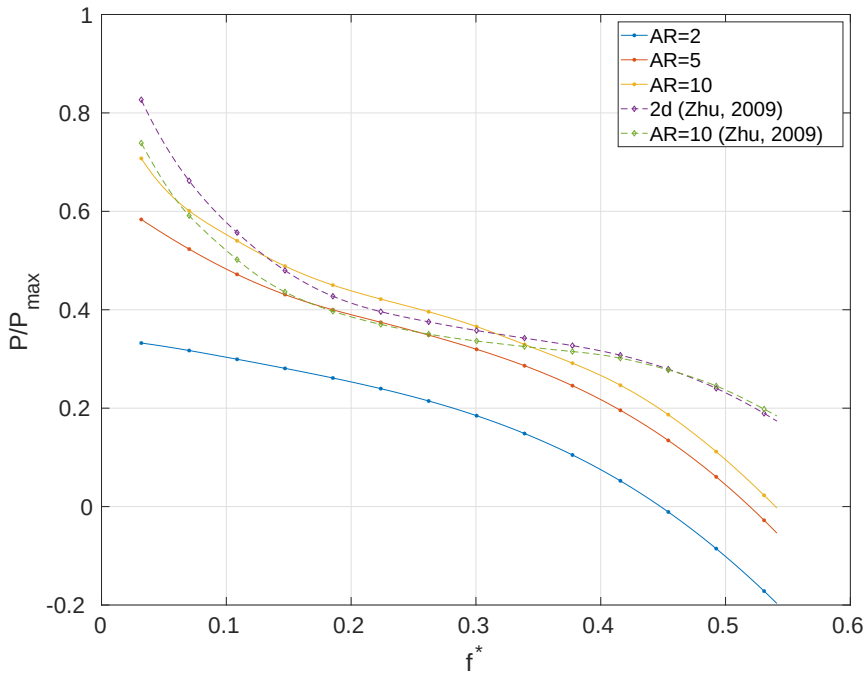

FIGURE 5.10: Effect of aspect ratio (AR) on the power extraction of an oscillating foil in energy harvesting mode and comparison with [75].

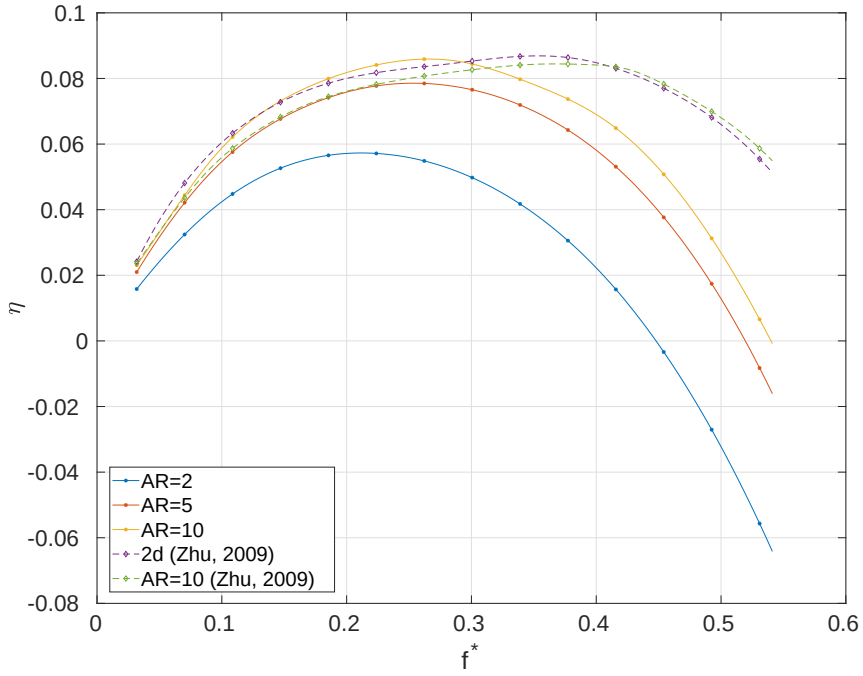

FIGURE 5.11: Effect of aspect ratio (AR) on the heave amplitude of an oscillating foil in energy harvesting mode and comparison with [75].

While the heaving response of the foil is the same as the one reported by Zhu et al. [75], the power extraction and the efficiency of the foil with the present method presentd differences as the frequency of the oscillation is becoming higher. This could be attributed to the Morino condition and the linearised wake dynamics.

5.4 Optimization and comparison with turbines

In order to compare the oscillating foil energy converter to the more traditional turbines in terms of energy conversion efficiency a suitable performance index should be devised. The oscillating foil sweeps a cross-section area (A_{sw}) equal to the area define by the maximum and minimum positions of the foil and the span of the device. Similarly, the turbines occupy a space equal to the area the blades sweep. This difference is shown in figure 5.13. Therefore, the power extracted by each device can be normalized by the power available for harnessing in the sweep area, yielding comparable performance indices

$$PI = \frac{\bar{P}_{net}}{\frac{1}{2}\rho V_{bg}^3 A_{sw}} \tag{5.3}$$

It is important that the height of the area swept by the oscillating foil is measured as the difference of the uppermost and lowermost points of the total motion of the foil, taking into consideration the pithing motion, and not only the heaving motion. In figure (5.12) the difference between the two definitions is shown. The performance of the foil is much lower when the pitching motion is taken into account.

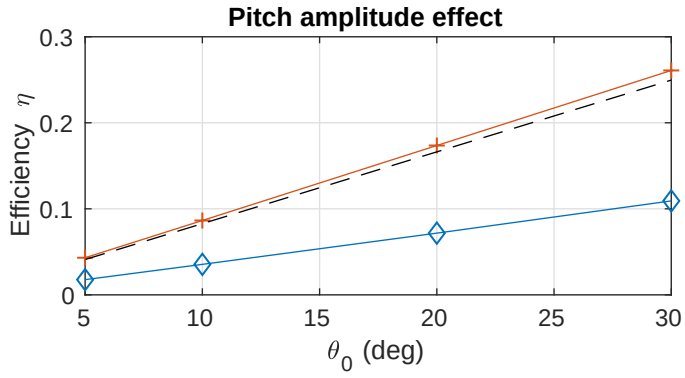

FIGURE 5.12: Effect of the cross-section definition on efficiency. The performance of the foil is much lower when the pitching motion is taken into account.

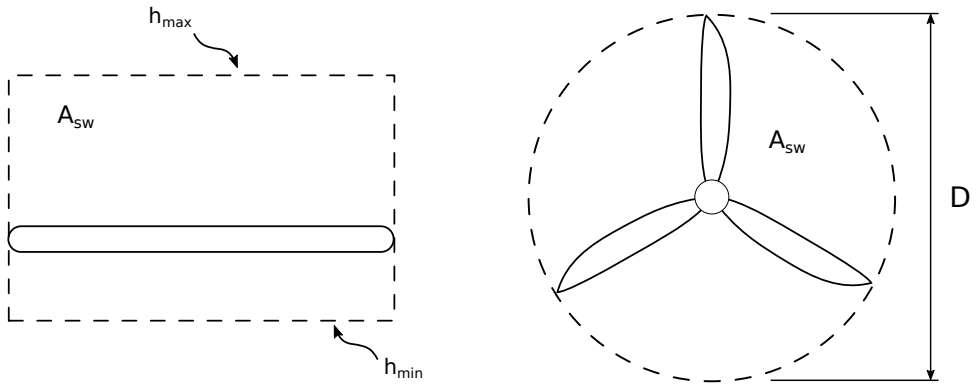

FIGURE 5.13: Sweep area of an oscillating foil and a horizontal-axis turbine.

For the oscillating foil device the sweep area can be calculated as

$$A_{sw} = (h_{max} - h_{min})s \tag{5.4}$$

where h_{max} and h_{min} are the maximum and minimum heaving positions of any point on the surface of the foil. For the turbine the sweep area is

$$A_{sw} = \pi \frac{D^2}{4} \tag{5.5}$$

where D is the diameter of the blades.

As the heaving amplitude of the oscillating foil changes with the different parameters of the pitching motion and the damping coefficient, it is essential to perform a study to find the optimal set of parameters (that maximize the performance index), before attempting any comparison with the turbines. In figures 5.14 and 5.15 the performance index for the oscillating foil is plotted against the damping coefficient and the frequency for different positions of the pitching axis (a/c) from the leading edge, for a rectangular NACA0005 foil. In the following simulations, the values of the different parameters are $c = 1$, $V_{bg} = 1$, $AR = 10$, $\theta_0 = 10, 25$. The mass of the foil is ignored and the spring coefficient is set to zero. The simulations are performed for reduced frequencies in the range $f^* = 0.08 - 0.80$ and damping coefficient in the

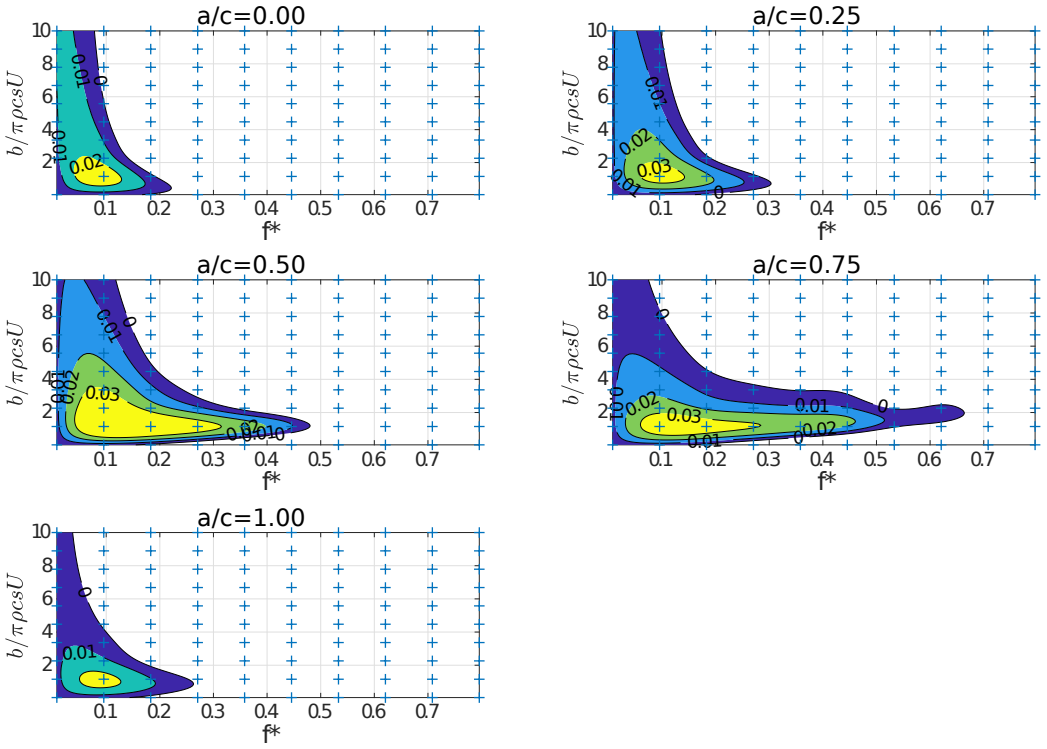

FIGURE 5.14: Parametric study of performance index of oscillating foil for $\theta_0 = 10^o$ and $a/c = [0, 0.25, 0.5, 0.75, 1]$.

range $\frac{b}{\pi\rho csV_{bg}} = 0 - 10$. In this way it is found that the best performing configuration is at $f^* = 0.11$, $\frac{b}{\pi\rho csV_{bg}} = 1.46$ and $a/c = 0.5$ for a pitching amplitude of 25 degrees. The performance index for this configuration is $PI_{max} = 0.126$. The results are qualitatively similar to those of Zhu et al. [75] and confirm the proposed region of maximum power extraction.

The Performance index of the oscillating foil increases monotonically with the pitch amplitude (see fig. 5.16) and confirms the observations of Filippas et al. [16] about the present method. The method described in Chapter 2 can give reliable results at small angles of attack and as a result the pitching amplitude has to be constrained at a maximum of 25 degrees. Here it should be noted that in literature the maximum performance is reported to be at about 75 degrees [31]. It is also apparent that a higher aspect ratio will increase the performance index slightly.

Further optimization of the oscillating foil can be done on other parameters, such as the planform and spanwise chord distribution. As the parameter space that needs to be searched for the maximum performance index is getting larger, optimization algorithms can be used. In the present study the foil will be skewed symmetrically by an angle θ_{skew} and the spanwise chord distribution will be considered linear, taking the value c_{mid} at the middle of the foil and the value c_{tip} at the wing tip. For these values the chord ratio can be defined as $CR = \frac{c_{tip}}{c_{mid}}$. The aspect ratio ($AR = s/c_{mid}$) will be kept constant at 5 and the pitch amplitude at 25 degrees. The oscillation frequency (f) and the damping coefficient (b) will be those maximizing the performance index in the case shown in figure 5.15.

The effect of the skew and the chord ratio on the performance of the foil is presented in figure 5.19. As the chord ratio approaches 1 the performance index increases

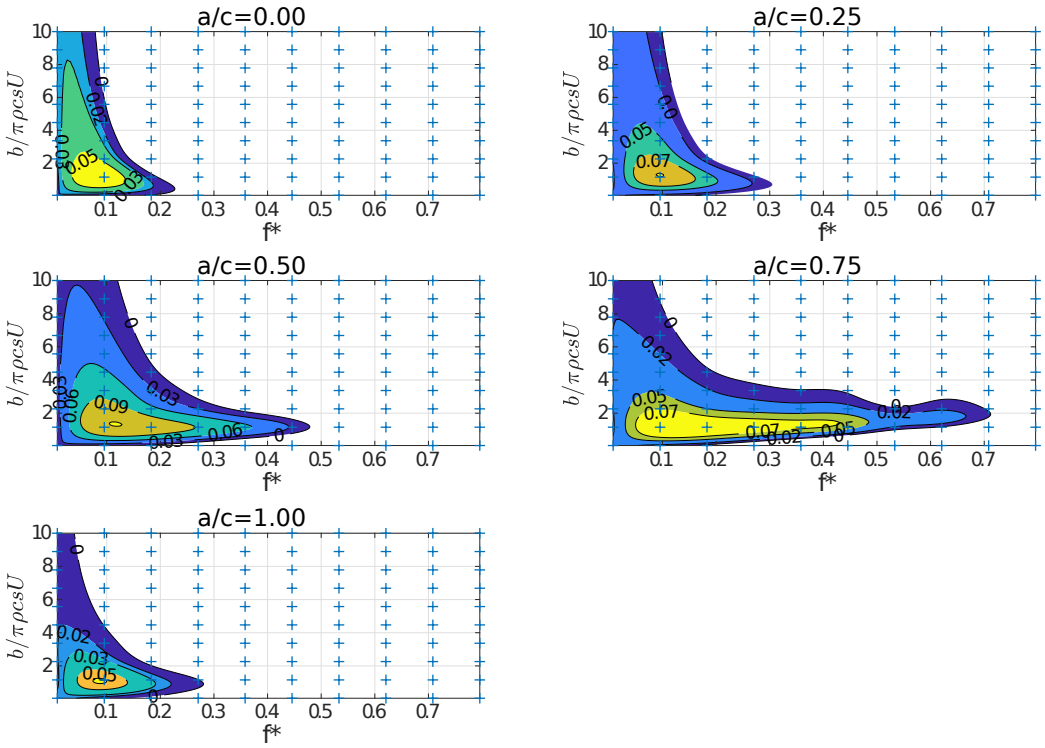

FIGURE 5.15: Parametric study of performance index of oscillating foil for $\theta_0 = 25^o$ and $a/c = [0, 0.25, 0.5, 0.75, 1]$.

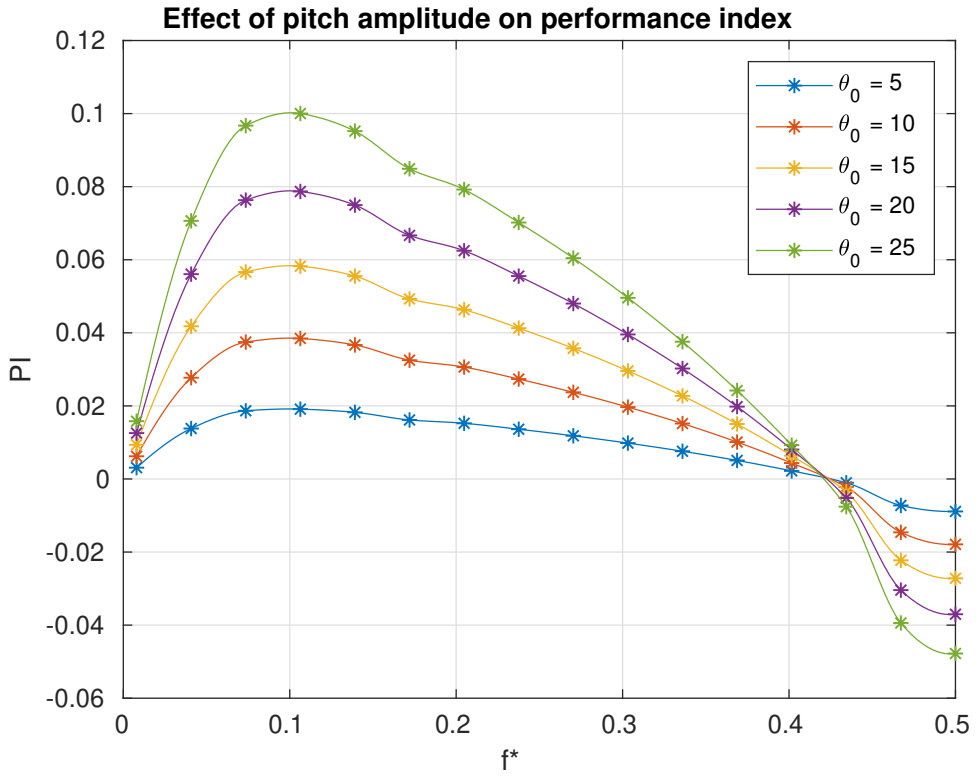

FIGURE 5.16: Performance index of oscillating foil over frequency for $\frac{b}{\pi\rho csV_{bg}} = 1.46$, $a/c = 0.5$ at different pitch amplitude angles.

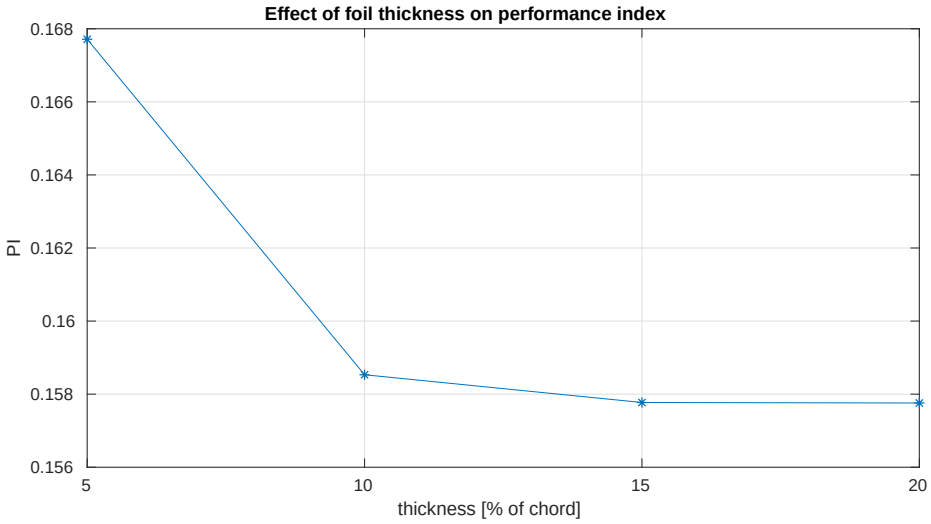

FIGURE 5.17: Performance index of oscillating foil over thickness for $\frac{b}{\pi\rho csV_{bg}} = 1.46$, $a/c = 0.5$, $\theta_0 = 25^o$ and $f = 0.11Hz$.

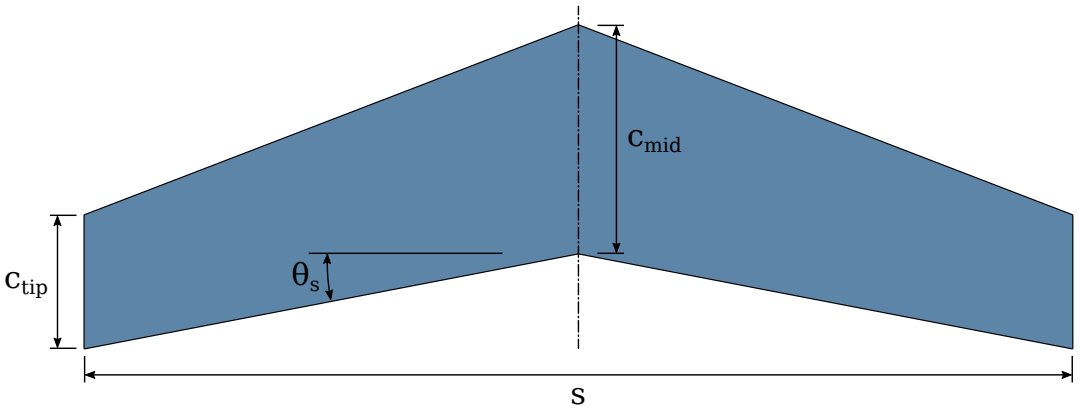

FIGURE 5.18: Geometry of a foil with sweep and non-constant chord length used in the numerical study.

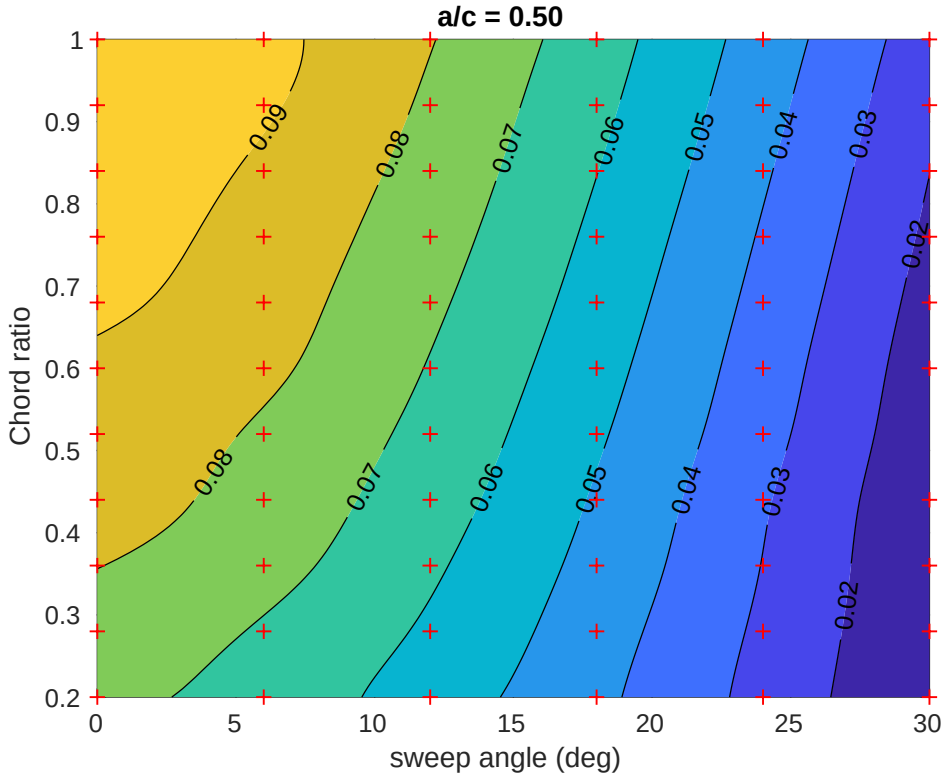

FIGURE 5.19: Effect of skew and chord ratio on performance index for $\frac{b}{\pi\rho csV_{bg}} = 1.46$, $a/c = 0.5$, $AR = 5$, $\theta_0 = 25^o$ and $f = 0.11Hz$.

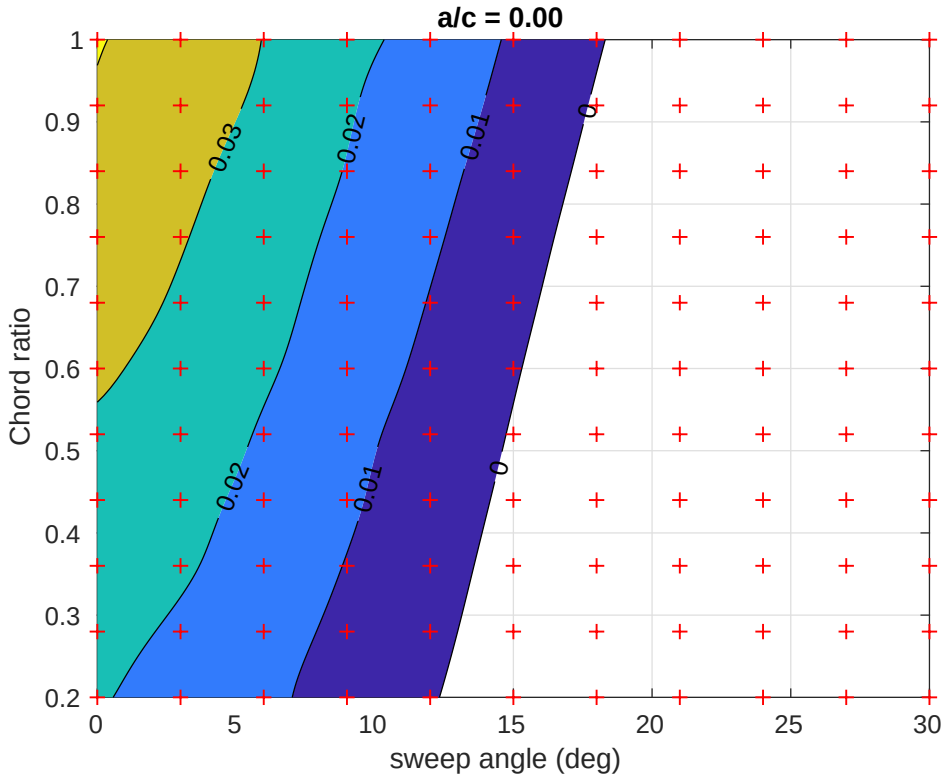

FIGURE 5.20: Effect of skew and chord ratio on performance index for $\frac{b}{\pi\rho csV_{bg}} = 1.46$, $a/c = 0$, $AR = 5$, $\theta_0 = 25^o$ and $f = 0.11Hz$.

monotonically. The simulation data show that a rectangular planform is the optimal shape for the foil.

Figure 5.20 shows the effect of the skew and the chord ratio on the performance when the pitching axis is at $a/c = 0$. In this configuration, the power production limit is for about 18 degrees of sweep angle. This results is explained by the fact that the distance between the center of hydrodynamic forces and the center of rotation of the foil are larger, compared to the case shown in figure 5.19, and thus, the device requires more power in order to counteract the hydrodynamic moments and maintain the pitching motion.

In literature an optimized horizontal axis turbine can be found to have a performance index of about $0.4 - 0.5$ in ideal environmental conditions [5], about 4-5 times higher than the oscillating foil device. Of course for environmental concerns this advantage may not be observed in practice, as researchers suggest a 10% energy absorptions from the current as a safe limit for minimizing the impact to the local marine ecosystem (see ch. 1). Moreover, investigation of multiple foils in tandem configuration [27] have shown that the overall power coefficient of an oscillating foil farm can reach asymptotically values higher than 0.9 for 15 foils in tandem or more.

Chapter 6

Conclusion and Future work

In the present work a study of the semi-activated oscillating foil marine energy extracting device was conducted. For this purpose, a GPU-accelerated solver was programmed using the Boundary Element Method, with the aid of the code developed by Filippas [15], for hydrodynamic calculations and a coupling algorithm for the fluid-structure interactions. The algorithm was accelerated by calculating the induction factors in parallel using the CUDA C/C++ API and the integrations were evaluated with an adaptive quadrature routine. In order to study the whole range of parameters the present method should be extended or another method (CFD,FEM) should be used to capture the involved phenomena more accurately.

The present Boundary Element Method extensively uses numerical integration to calculate the induction factors. To reduce the number of integrations, the Fast Multipole Method (FMM) can be used [47]. When the boundary elements are far from the evaluation point, the source and dipole distribution on the elements can be approximated by a single point source or doublet.

One aspect of the oscillating foil energy harvester that has not yet been investigated in depth is the effect the planform has on the efficiency of the device. All the studies so far use foils of rectangular planform. In the present work skewed foils were studied yielding low efficiency for high angles of skew, but further investigation is needed in order to verify this result and find the optimal geometry.

When the device described in the present work operates in real environmental conditions the optimal operating point may vary. To always approach the best performance of the foil, a robust control system has to be designed. Simulink provides a great suite for control system design and testing. In order to integrate the code developed for the purposes of this thesis into Simulink it is useful to make the solver compatible with the FMI protocol for co-simulation, that has recently begun being supported by Mathworks. The FMI protocol defines some routines that must be available in any simulation model to be able to communicate correctly with other models and programs. Therefore, the developed solver can become the plant model for a controller designed in Simulink. The simulation of the foil dynamics can be offloaded to a remote machine and communicate the results to the machine running Simulink via the local network, as the communication speed will not be the main bottleneck.

Appendix A

Bilinear element geometry

The bilinear element is a patch surface bounded by the four line segments connecting the four vertices of the element, as seen in the figure A.1. The exact geometry of the surface is given by the relation

$$\mathbf{x}(u,v) = \mathbf{x}_n N_n(u,v), i = 1,2,3 \tag{A.1}$$

where repeating indices denote sumation over the index variable, and N_n are the shape functions

$$N_1(u,v) = 0.25(1-u)(1-v) \tag{A.2}$$
$$N_2(u,v) = 0.25(1+u)(1-v) \tag{A.3}$$
$$N_3(u,v) = 0.25(1+u)(1+v) \tag{A.4}$$
$$N_4(u,v) = 0.25(1-u)(1+v) \tag{A.5}$$

The curvilinear coodrinate system created by the parametrization of the curve is

$$e_1(u,v) = \frac{\partial \mathbf{x}}{\partial u} \tag{A.6}$$
$$e_2(u,v) = \frac{\partial \mathbf{x}}{\partial v} \tag{A.7}$$
$$e_3(u,v) = e_1 \times e_2 \tag{A.8}$$

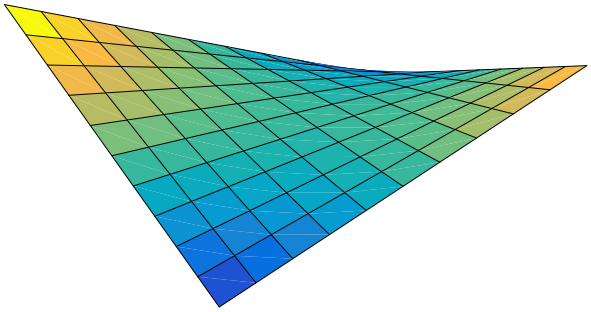

FIGURE A.1: Bilinear element element geometry

A descrete boundary consisted of bilinear elements has C^0 continuity every where and C^1 continuity within each element.

Appendix B

Further study of the flow around the foil

B.1 Span-wise effects

In figure (B.1) the spanwise distribution of the lift is plotted for different timeframes of a period and for aspect ratios $AR = [3.4, 10, 20, 50]$, for the device of Huxham et al. [25]. It is apparent that the lift distribution tends to become constant along the span of the wing for high values of the aspect ratio.

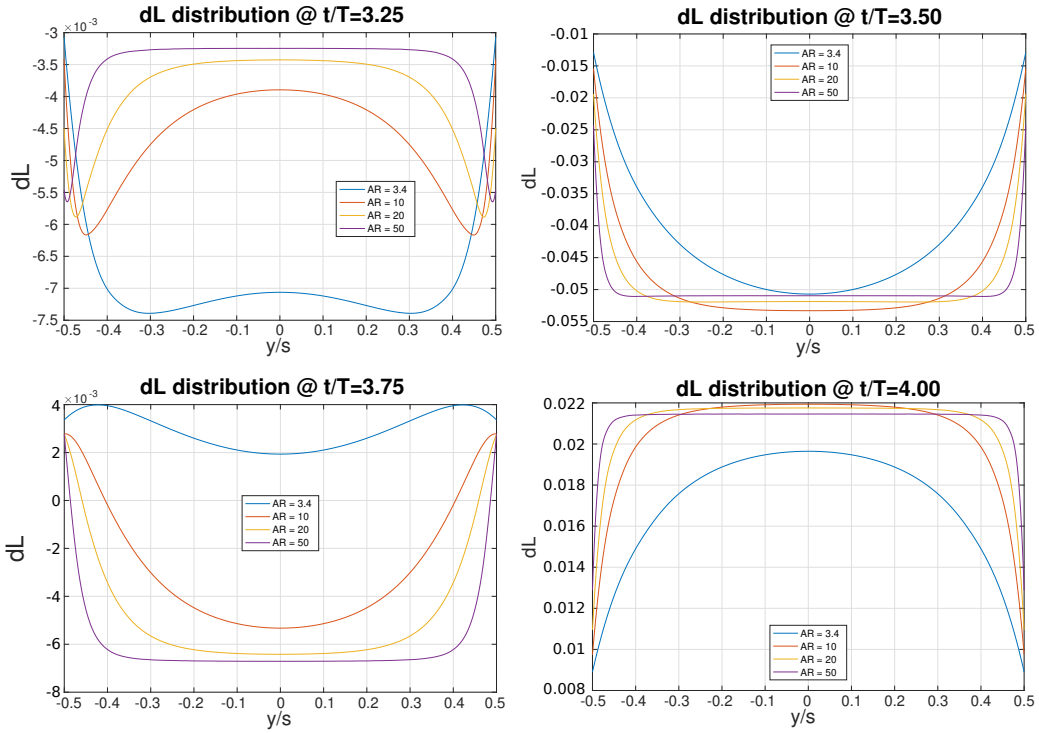

FIGURE B.1: Distribution of the lift on the foil in the spanwise direction for aspect ratios $AR = 3.4, 10, 20, 50$ at reduced frequency $f^* = 0.10$.

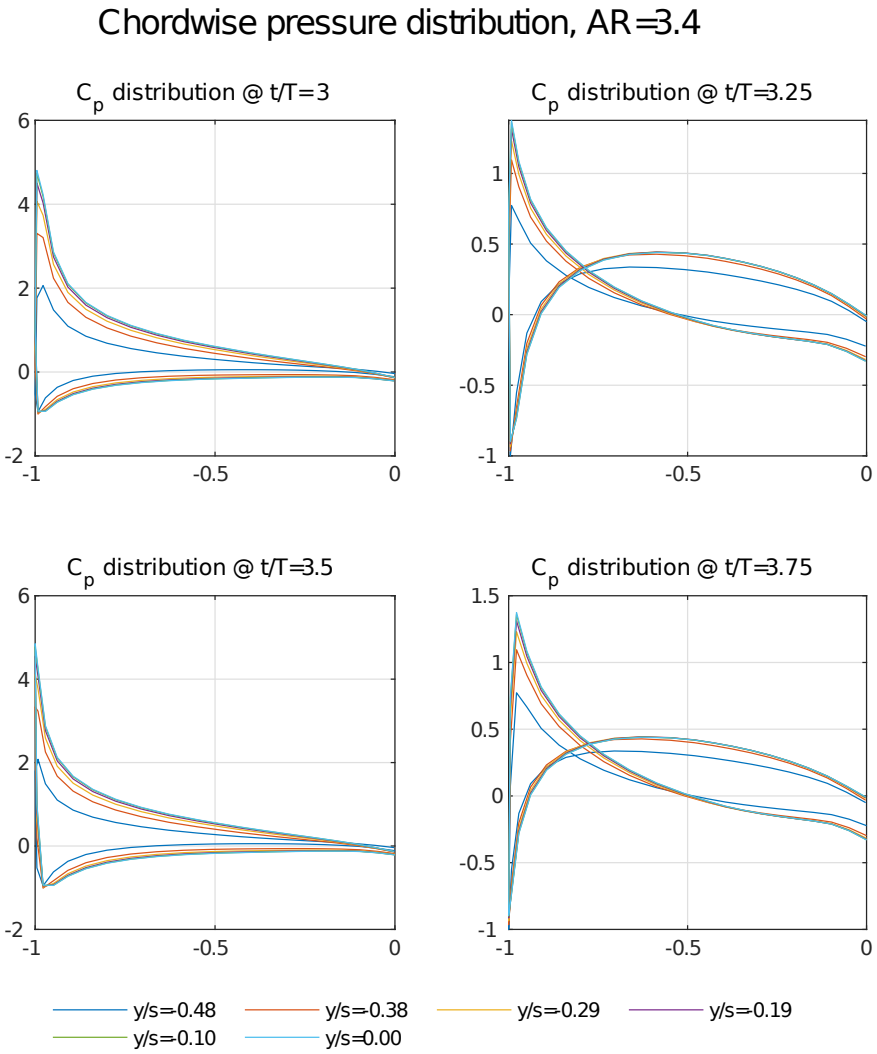

FIGURE B.2: Chord-wise distribution of pressure coefficient for different sections along the span of the foil at key timeframes for $AR = 3.4$.

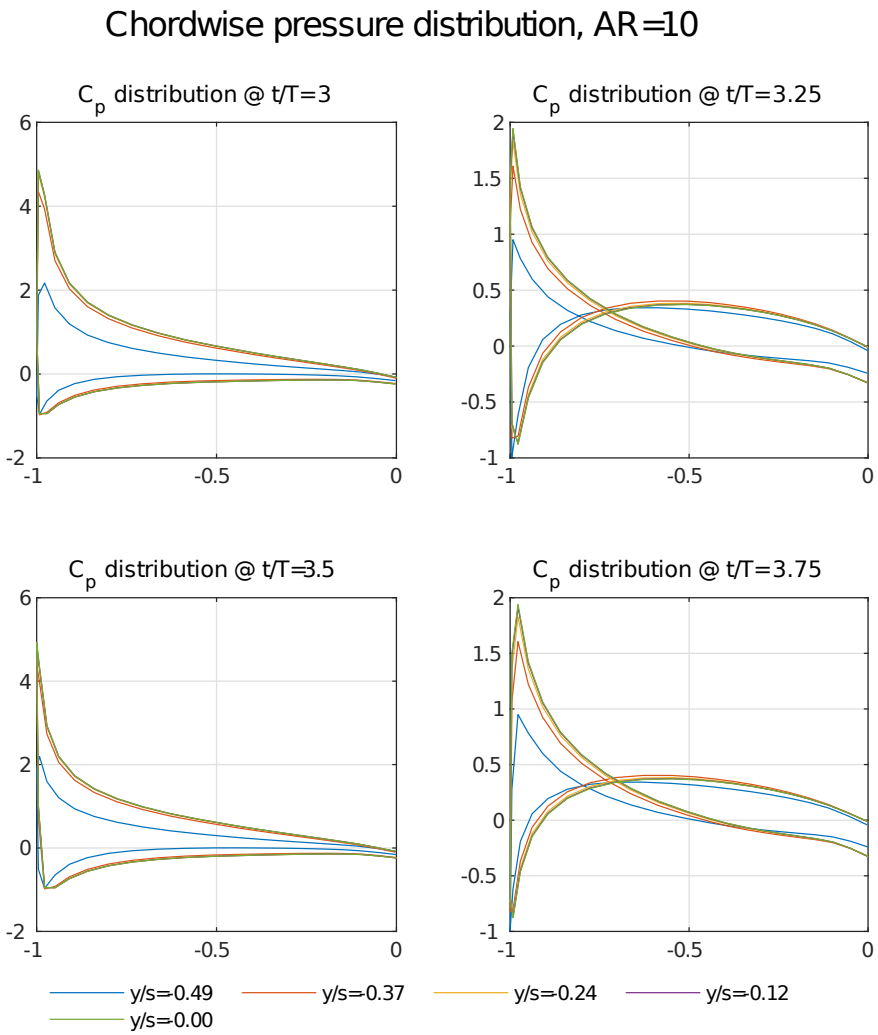

FIGURE B.3: Chord-wise distribution of pressure coefficient for different sections along the span of the foil at key timeframes for $AR = 10$.

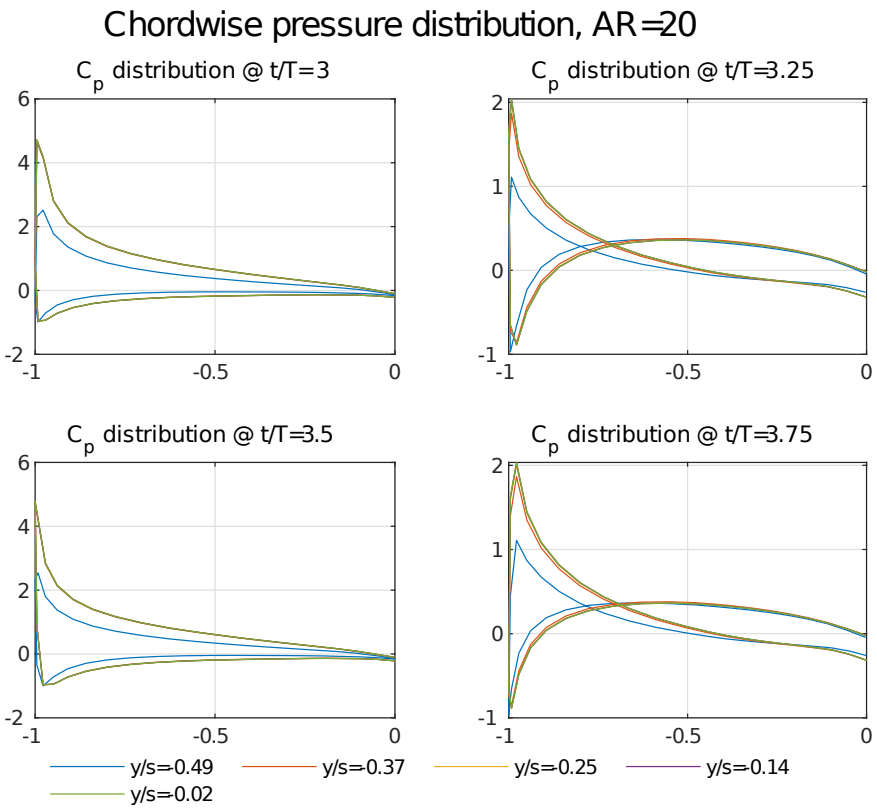

FIGURE B.4: Chord-wise distribution of pressure coefficient for different sections along the span of the foil at key timeframes for $AR = 20$.

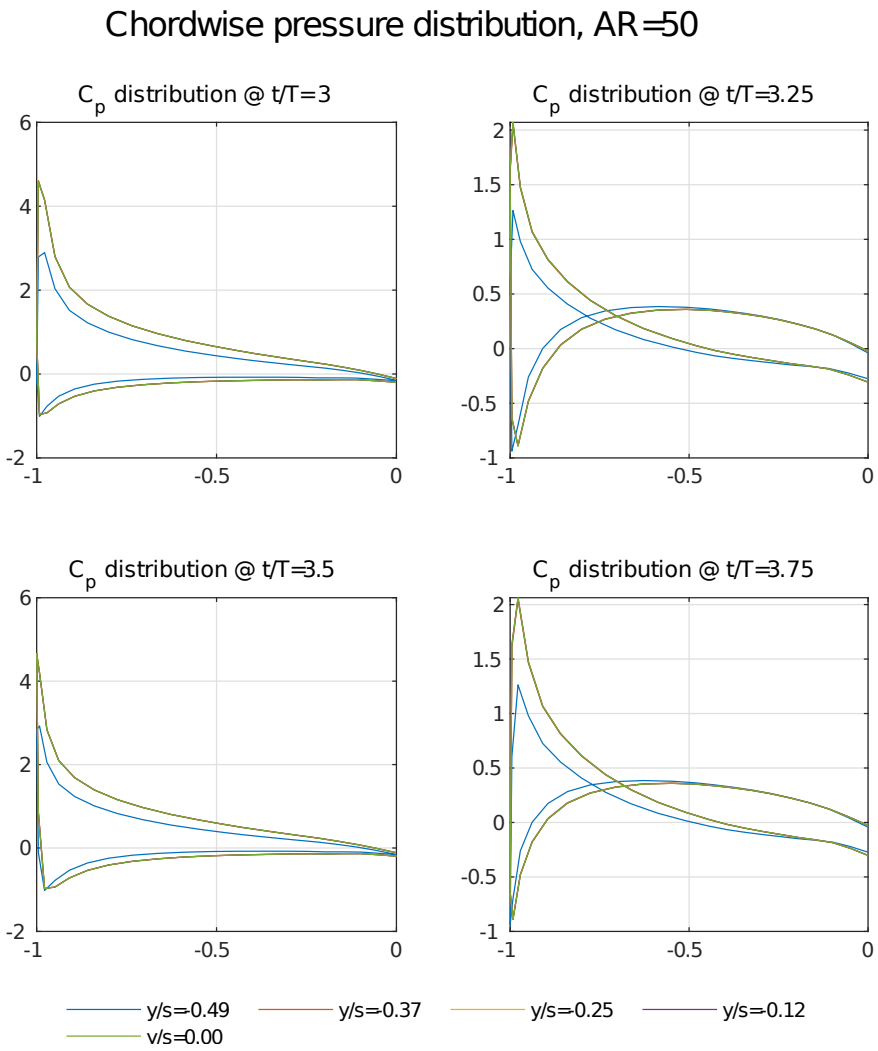

FIGURE B.5: Chord-wise distribution of pressure coefficient for different sections along the span of the foil at key timeframes for $AR = 50$.

Appendix C

Efficiency of biomimetic foil shape

The developed code can be used to examine the performance characteristics of complex geometries. Such examples are the flippers of dolphins and whales. Here the performace of the humpback whale (megaptera novaeangliae) flipper is examined.

Fish and Battle [17] have studied the geometry of the humpback whale's flipper, from a hydrodynamic point of view. For the animal examined in their study, the flipper had an aspect ratio of 6.1 and an elliptical planform. Also, the outer 1/3rd of the wing was swept backwards by 19 degrees. The thickness to chord ratio varies along the flipper with an average of 0.22-0.23. They also report that the cross-section of the flipper is streamlined and near the midspan resembles the design of the $NACA63_4 - 021$ foil.

In figure C.2 the flipper is compared with an engineerd wing of the same aspect ratio and constant spanwise chord and thickness distribution. The damping coefficient is 1.46, that of the optimal foil (see 5). The chord of the foil is taken to be 1m and the thickness equal to 22% of the chord, as it is the average of the whale flipper. In figure C.3 the performance of the foil is drawn for a range of frequencies and dampening coefficients. The performance index of the whale flipper is slightly lower than that of the foil (about 1%). The maximum efficiency is observed at the same frequency for both geometries and the overall shape of the performance index curves remains similar.

It is obvious that the geometry of the flipper is complex and does not give an advantage in performance. Further analysis can be performed with biomimetic foils

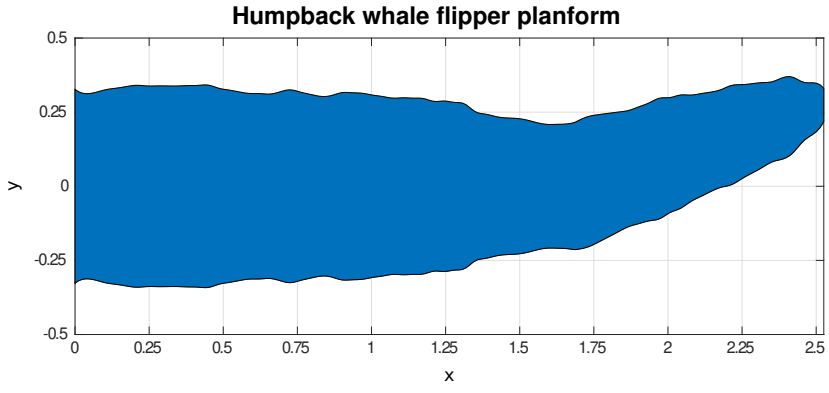

FIGURE C.1: Humpback whale flipper planform. Data drawn from [17].

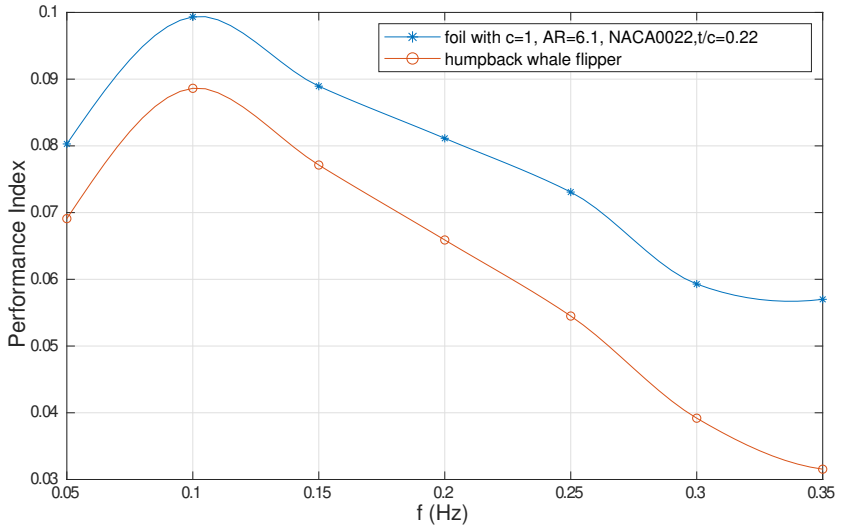

FIGURE C.2: Comparison of humpback whale flipper [17] as a oscillating foil marine energy harvester with a foil of constant cross-section.

inspired from other species, i.e. dolphins. In order to make a better comparison deformation of the flipper troughout the motion should be taken into account, something that the current version of the software does not allow.

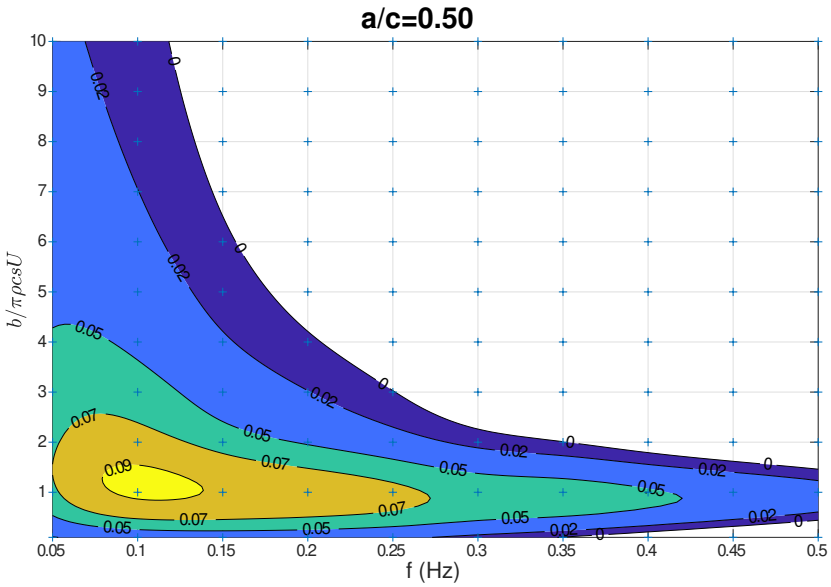

FIGURE C.3: Parametric study of humpback whale flipper [17] as a oscillating foil marine energy harvester.

Bibliography

[1] Milton Abramowitz. 1974. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,* Dover Publications, Inc., New York, NY, USA. ISBN: 0486612724.

[2] [n. d.] Achieved Occupancy. https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/achievedoccupancy.htm.

[3] Press Association. 2010. Major scottish tidal project unveiled. *New Civil Engineer*, (October 2010). https://www.newcivilengineer.com/news/energy/major-scottish-tidal-project-unveiled/8607525.article.

[4] G.K. Batchelor. 2000. *An Introduction to Fluid Dynamics*. *Cambridge Mathematical Library*. Cambridge University Press. ISBN: 9780521663960. https://books.google.gr/books?id=Rla7QihRvUgC.

[5] WMJ Batten, AS Bahaj, AF Molland, JR Chaplin, Sustainable Energy Research Group, et al. 2007. Experimentally validated numerical method for the hydrodynamic design of horizontal axis tidal turbines. *Ocean engineering*, 34, 7, 1013–1020.

[6] Jarle Berntsen, Terje O. Espelid, and Tor Sørevik. 1991. On the subdivision strategy in adaptive quadrature algorithms. *Journal of Computational and Applied Mathematics*, 35, 1, 119 –132. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(91)90201-T. http://www.sciencedirect.com/science/article/pii/037704279190201T.

[7] A Betz. 1920. Das maximum der theoretisch möglichen ausnutzung des windes durch windmotoren. *Z. Gesamte Turbinenwesen*, 26.

[8] James M Birch and Michael H Dickinson. 2001. Spanwise flow and the attachment of the leading-edge vortex on insect wings. *Nature*, 412, 6848, 729.

[9] bps. [n. d.] bioWAVE. http://bps.energy/biowave.

[10] IG Bryden, T Grinsted, and GT Melville. 2004. Assessing the potential of a simple tidal channel to deliver useful energy. *Applied Ocean Research*, 26, 5, 198–204.

[11] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. Lee, and K. Skadron. 2009. Rodinia: a benchmark suite for heterogeneous computing. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 44–54. DOI: 10.1109/IISWC.2009.5306797.

[12] John O Dabiri. 2007. Renewable fluid dynamic energy derived from aquatic animal locomotion. *Bioinspiration & biomimetics*, 2, 3, L1.

[13] JD DeLaurier and JM Harris. 1982. Experimental study of oscillating-wing propulsion. *Journal of Aircraft*, 19, 5, 368–373.

[14] Morlais Anglesay Marine Energy. [n. d.] The West Anglesey Demonstration Zone. http://www.morlaisenergy.com/.

[15] E.S. Filippas. 2019. *Hydrodynamic analysis of ship and marine biomimetic systems in waves using GPGPU programming*.

[16] ES Filippas, Th P Gerostathis, and KA Belibassakis. 2018. Semi-activated oscillating hydrofoil as a nearshore biomimetic energy system in waves and currents. *Ocean Engineering*, 154, 396–415.

[17] Franke E Fish and Juliann M Battle. 1995. Hydrodynamic design of the humpback whale flipper. *Journal of Morphology*, 225, 1, 51–60.

[18] Department for Business Energy & Industrial Strategy. 2017. Energy Consumption in the UK. (July 2017).

[19] Walter Gander and Walter Gautschi. 2000. Adaptive quadraturerevisited. *BIT Numerical Mathematics*, 40, 1, 84–101.

[20] Chris Garrett and Patrick Cummins. 2008. Limits to tidal current power. *Renewable Energy*, 33, 11, 2485–2490.

[21] Chris Garrett and Patrick Cummins. 2007. The efficiency of a turbine in a tidal channel. *Journal of fluid mechanics*, 588, 243–251.

[22] H. Glauert. 1983. *The Elements of Aerofoil and Airscrew Theory*. *Cambridge Science Classics*. Cambridge University Press. DOI: 10.1017/CBO9780511574481.

[23] Alan Henry, Kenneth Doherty, Lachlan Cameron, Trevor Whittaker, and Ronan Doherty. 2010. Advances in the design of the oyster wave energy converter, (January 2010).

[24] John L Hess and AM Smith. 1962. Calculation of non-lifting potential flow about arbitrary three-dimensional bodies. Technical report. Douglas Aircraft Co Long Beach CA.

[25] GH Huxham, S Cochard, and J Patterson. 2012. Experimental parametric investigation of an oscillating hydrofoil tidal stream energy converter. In *18th Australasian Fluid Mechanics Conference (AFMC), Launceston, Australia, Dec*, 3–7.

[26] NE Joukowsky. 1920. Windmill of the nej type. *Transactions of the Central Institute for Aero-Hydrodynamics of Moscow*.

[27] Hamid Reza Karbasian, JA Esfahani, and E Barati. 2015. Simulation of power extraction from tidal currents by flapping foil hydrokinetic turbines in tandem formation. *Renewable Energy*, 81, 816–824.

[28] Joseph Katz and Allen Plotkin. 2001. *Low-speed aerodynamics*. Volume 13. Cambridge university press.

[29] MJ Khan, G Bhuyan, MT Iqbal, and JE Quaicoe. 2009. Hydrokinetic energy conversion systems and assessment of horizontal and vertical axis turbines for river and tidal applications: a technology status review. *Applied energy*, 86, 10, 1823–1835.

[30] Thomas Kinsey and Guy Dumas. 2012. Optimal tandem configuration for oscillating-foils hydrokinetic turbine. *Journal of fluids engineering*, 134, 3, 031103.

[31] Thomas Kinsey and Guy Dumas. 2008. Parametric study of an oscillating airfoil in a power-extraction regime. *AIAA journal*, 46, 6, 1318–1330.

[32] Manoochehr M Koochesfahani. 1989. Vortical patterns in the wake of an oscillating airfoil. *AIAA journal*, 27, 9, 1200–1205.

[33] FW Lanchester. 1915. A contribution to the theory of propulsion and the screw propeller. *Transactions of the Institution of Naval Architects*, 57, 98–116.

[34] Raphael Landaverde, Tiansheng Zhang, Ayse K Coskun, and Martin Herbordt. 2014. An investigation of unified memory access performance in cuda. In *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*. IEEE, 1–6.

[35] Ocean Power Delivery Ltd. [n. d.] The Pelamis Wave Energy Converter: An update on performance and operational aspects. https://energiatalgud.ee/img_auth.php/3/3a/Carcas%2C_M._The_Pelamis_Wave_Energu_Converter.pdf.

[36] Verdant Power Ltd. 2008. Verdant Powers Free Flow Turbines. https://www.verdantpower.com/.

[37] J. N. Lyness. 1969. Notes on the adaptive simpson quadrature routine. *J. ACM*, 16, 3, (July 1969), 483–495. ISSN: 0004-5411. DOI: 10.1145/321526.321537. http://doi.acm.org/10.1145/321526.321537.

[38] Stanislaw R Massel. 1989. *Hydrodynamics of coastal zones*. Volume 48. Elsevier.

[39] William McKinney and James DeLaurier. 1981. Wingmill: an oscillating-wing windmill. *Journal of energy*, 5, 2, 109–115.

[40] B.L. Méhauté and D.M. Hanes. 2005. *Ocean Engineering Science*. *The Global Coastal Ocean: Multiscale Interdisciplinary Processes*. Harvard University Press. ISBN: 9780674017399.

[41] HH Michels. 1963. Abscissas and weight coefficients for lobatto quadrature. *Mathematics of Computation*, 17, 83, 237–244.

[42] L.M. Milne-Thomson. 1996. *Theoretical Hydrodynamics*. *Dover Books on Physics Series*. Dover Publications. ISBN: 9780486689708.

[43] Luigi Morino. 1974. A general theory of unsteady compressible potential aerodynamics.

[44] Max M. Munk. 1922. General Theory of Thn Wing Sections. NACA.

[45] Simon P Neill, Emmer J Litt, Scott J Couch, and Alan G Davies. 2009. The impact of tidal stream turbines on large-scale sediment dynamics. *Renewable Energy*, 34, 12, 2803–2812.

[46] J.N. Newman. 1977. *Marine Hydrodynamics*. Wei Cheng Cultural Enteroprise Company. ISBN: 9780262140263.

[47] Naoshi Nishimura. 2002. Fast multipole accelerated boundary integral equation methods. *Applied mechanics reviews*, 55, 4, 299–324.

[48] nVidia. [n. d.] CUDA 6.5 Performance Report. http://developer.download.nvidia.com/compute/cuda/6_5/rel/docs/CUDA_6.5_Performance_Report.pdf.

[49] NVIDIA. 2017. CUDA C PROGRAMMING GUIDE. Technical report. (December 2017). https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf.

[50] NVIDIA. [n. d.] NVIDIAs Next Generation CUDA Compute Architecture: Kepler GK110/210. Technical report.

[51] Zhangli Peng and Qiang Zhu. 2009. Energy harvesting through flow-induced oscillations of a foil. *Physics of fluids*, 21, 12, 123602.

[52] Gerasimos K Politis and Vasileios T Tsarsitalidis. 2014. Flapping wing propulsor design: an approach based on systematic 3d-bem simulations. *Ocean Engineering*, 84, 98–123.

[53] Gerasimos K Politis and Vassileios Tsarsitalidis. 2009. Simulating biomimetic (flapping foil) flows for comprehension, reverse engineering and design. In *First International Symposium on Marine Propulsors*.

[54] G.K. Politis. 2011. *The Boundary Element Method for 3-D Fluid Flow Problems*.

[55] Richardson and Gaunt. 1927. Viii. the deferred approach to the limit. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 226, 636-646, 299–361. ISSN: 0264-3952. DOI: 10.1098/rsta.1927.0008. eprint: http://rsta.royalsocietypublishing.org/content/226/636-646/299.full.pdf. http://rsta.royalsocietypublishing.org/content/226/636-646/299.

[56] Olinde Rodrigues. 1815. *De l'attraction des sphéroïdes*. PhD thesis.

[57] Fergal O Rourke, Fergal Boyle, and Anthony Reynolds. 2010. Tidal energy update 2009. *Applied Energy*, 87, 2, 398 –409. ISSN: 0306-2619. DOI: https://doi.

org/10.1016/j.apenergy.2009.08.014. http://www.sciencedirect.com/science/article/pii/S030626190900347X.

[58] Kirill V Rozhdestvensky and Vladimir A Ryzhov. 2003. Aerohydrodynamics of flapping-wing propulsors. *Progress in aerospace sciences*, 39, 8, 585–633.

[59] J Otto Scherer. 1968. Experimental and theoretical investigation of large amplitude oscillation foil propulsion systems. Technical report. HYDRONAUTICS INC LAUREL MD.

[60] Lionel Schouveiler, FS Hover, and MS Triantafyllou. 2005. Performance of flapping foil propulsion. *Journal of Fluids and Structures*, 20, 7, 949–959.

[61] Wei Shyy and Hao Liu. 2007. Flapping wings and aerodynamic lift: the role of leading-edge vortices. *AIAA journal*, 45, 12, 2817–2819.

[62] T Sørevik. 1988. *Reliable and efficient adaptive quadrature algorithms*. PhD thesis. Department of Informatics, University of Bergen Norway.

[63] Leonidas Thermos. 2017. *A Boundary Element Method for the hydrodynamic analysis of floating bodies in variable bathymetry regions*. (November 2017).

[64] Michael S Triantafyllou, Alexandra H Techet, and Franz S Hover. 2004. Review of experimental work in biomimetic foils. *IEEE Journal of Oceanic Engineering*, 29, 3, 585–594.

[65] X.S. Trompoukis. 2012. *Numerical solution of aerodynamic-aeroelastic problems on Graphics Processing Units*.

[66] Ross Vennell. 2012. Realizing the potential of tidal currents and the efficiency of turbine farms in a channel. *Renewable Energy*, 47, 95–102.

[67] T. Von Karman and J. M. Burgers. 1935. *Aerodynamic Theory - Perfect Fluids*. Volume 2. Springer Berlin, 308.

[68] Eric W. Weisstein. [n. d.] Lobatto quadrature. http://mathworld.wolfram.com/LobattoQuadrature.html. ().

[69] Adam Westwood. 2004. Ocean power: wave and tidal energy review. *Refocus*, 5, 5, 50 –55. ISSN: 1471-0846. DOI: https://doi.org/10.1016/S1471-0846(04)00226-4. http://www.sciencedirect.com/science/article/pii/S1471084604002264.

[70] ET Whittaker and G Robinson. 1944. *The calculus of observations: a treatise on numerical mathematics*. Blackie & Son London.

[71] Qing Xiao and Qiang Zhu. 2014. A review on flow energy harvesters based on flapping foils. *Journal of fluids and structures*, 46, 174–191.

[72] John Young, Joseph CS Lai, and Max F Platzer. 2014. A review of progress and challenges in flapping foil power generation. *Progress in Aerospace Sciences*, 67, 2–28.

[73] Qiang Zhu. 2011. Optimal frequency for flow energy harvesting of a flapping foil. *Journal of fluid mechanics*, 675, 495–517.

[74] Qiang Zhu and Zhangli Peng. 2009. Mode coupling and flow energy harvesting by a flapping foil. *Physics of Fluids*, 21, 3, 033601.

[75] Qiang Zhu, Max Haase, and Chin H Wu. 2009. Modeling the capacity of a novel flow-energy harvester. *Applied Mathematical Modelling*, 33, 5, 2207–2217.