POLITECNICO DI TORINO

VON KARMAN INSTITUTE

MASTER'S DEGREE IN AEROSPACE ENGINEER
PROPULSION SYSTEMS

Closed-Loop Liquid Film Control with a 3D Spectral Solver

Supervisors:
Prof. Sandra PIERACCINI
Prof. Miguel A. MENDEZ
Mr. Fabio PINO

Candidate:
Edoardo FRACCHIA

December 2023

Abstract

Hot-dip galvanization is a ubiquitous process involved in steel production, which consists in covering the alloy with a layer of molten zinc. Gas jets impinge on the surface in order to control the thickness in a procedure called jet wiping. However, this causes instabilities that manifest as undulations on the final product. Being able to accurately simulate how a jet perturbs the liquid zinc layer on an upward-moving substrate is thus essential to obtain a thin and smooth final coating. A 3D spectral solver has been improved and tested; it takes into account the surface tension terms and it features a perfectly matched layer. Some efforts have been made to make the solver faster while being able to simulate a larger number of jets. Jet wiping is not the only control method that has been investigated: since the liquid zinc is paramagnetic at those temperatures the flow could also be controlled by electromagnets. These simulations are at the basis of the final aim of this work, which is to optimize the controllers characteristics and behavior in order to obtain the desired final coating. A fast solver is an essential aspect when trying to run machine learning simulations of such systems, on this note another idea that could speed up the time integration involves computing the Jacobian of the system. Some steps have been taken into this direction but dealing with simplified equations.

Contents

List of Figures

List of Tables

List of Abbreviations

BC	**B**oundary **C**onditions
IBL	**I**ntegral **B**oundary **L**ayer
BLEW	**B**oundary **L**ay**E**r **W**iping
FT	**F**ouerier **T**ransform
IFT	**I**nverse **F**ouerier **T**ransform
DFT	**D**iscrete **F**ouerier **T**ransform
RHS	**R**ight **H**and **S**ide
PML	**P**erfectly **M**atched **L**ayer
PPO	**P**roximal **P**olicy **O**ptimization
RL	**R**einforcement **L**earning
ML	**M**achine **L**earning

List of Symbols

$\hat{h}$	Dimensionless height of the liquid film
$[h]$	Reference height of the liquid film
U_p	Vertical velocity of the metal strip
$\boldsymbol{\tau}_w$	Wall shear stress distribution
p_g	Gas pressure distribution
Ca	Capillary number
Re	Reynolds number
Ka	Kapitza number
Ha	Hartmann number
B	Magnetic field
ν_l	Liquid kinematic viscosity
μ_l	Liquid dynamic viscosity
ρ_l	Liquid density
ρ_g	Gas density
σ_l	Liquid surface tension
d	Jet diameter
H	Stand-off distance
$\hat{q}$	Dimensionless flow rate
W	DFT matrix
$\overline{W}$	Inverse DFT matrix
$\mathcal{F}$	Fourier transform
$\mathcal{F}^{-1}$	Inverse Fourier transform
$s(t)$	State at time t
$a(t)$	Action at time t
$R(s_t, a_t)$	Reward
$J(\theta)$	Objective function
θ	Objective function parameters
$\pi(s_t, t)$	Policy

Introduction

Corrosion has always been a critical problem since the dawn of the metal ages. Affecting the mechanical properties of steel and iron, corrosion degrades a wide range of components, leading to profound economic consequences. To give some figures, the degradation of metal due to rust in the United States alone cost 300 billion dollars in 1995 or 3% of the nation's Gross Domestic Product (GDP) [6, 14]. This degradation not only necessitates high maintenance costs, but may also prompt the complete replacement of parts, resulting in additional expenses associated with material production and transportation. The impact corrosion is not merely economical as it can also lead to the waste of natural resources, hazardous failures, and many other indirect costs. To exacerbate matters, global steel production is anticipated to rise in the coming decades, primarily fuelled by emerging economies [10]. Given the imperative to reduce sector carbon emissions, coupled with the escalating demand, there is an increasing emphasis on enhancing the durability of new steel. Thus, protecting steel from corrosion becomes paramount in ensuring its longevity.

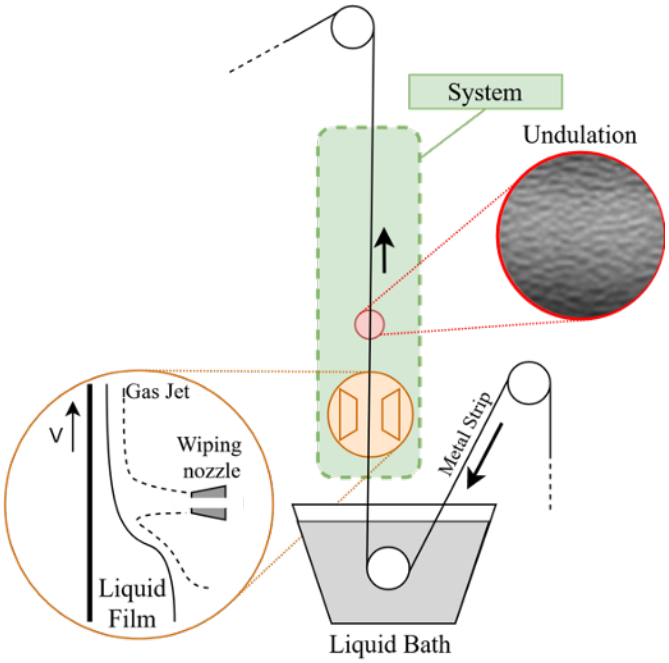

FIGURE 1: Scheme of the hot-dip galvanization process with the metal strip dip into a bath of molten zinc and then withdrawn vertically with the wiping jet (gold circle) removing the liquid excess and the wavy pattern in the final coating caused by the undulation instability (red circle).

One of the common methods to protect steel from corrosion consists in coating it with a thin layer of protective material. Fig.1 shows the hot-dip galvanizing process where a metal strip is immersed in a bath of molten zinc and then withdrawn vertically, creating a thin layer of zinc on the metal surface. This process is inherently simple and provides three layers of defence to the coated steel. Firstly, on a mechanical level, an adherent barrier is created, isolating the steel

from the electrolytes in the environment. This barrier, in particular, can boast good resistance in harsh environments while also being resistant to abrasion. Once the first protection level is breached, the cathodic protection comes into play. Zinc protects steel by being consumed before the steel. Finally, when zinc begins to corrode, as a consequence of being exposed to the atmosphere, corrosion by-products will naturally form on the coating surface. The formation of these by-products creates a patina, which, once fully developed, slows the corrosion rate, acting as an additional passive barrier for coating.

The smoothness of the final coating is a major industrial stake. In this respect, the use of control jets or magnets becomes crucial. However, the resultant film often falls short in thinness and homogeneity, as highlighted in [9], leading to wastage of materials or inadequate finishes. To address this issue, gas jets impinge upon the film, generating a runback flow, as illustrated in Figure 1.1. This process is not exclusive to steel strips; its applicability extends to various coating processes, including those involved in paper and film manufacturing, where the deposition of a thin layer is fundamental. Being able to apply a thin and homogeneous film results in less material being used and a more effective and durable finish. Jet wiping is a coating process using gas jets to control the coating thickness coming out of the bath of molten zinc. Despite bing simple and cost-effective, this technique suffers from a drawback, limiting the production rate and the coating smoothness. At high strip velocities, the gas jet starts to oscillate, leading to the formation of wavy patterns in the final coating surface. The final goal of this work is to numerically simulate the zinc film on a moving substrate and to introduce Reinforcement Learning in order to find a control law for the jets or magnets that guarantees an adequate final coating.

Chapter 1

Mathematical modelling

This chapter presents the reduced order model used to simulate the liquid film undulation (Section 1.2) and the modelling of the control actuators (Section 1.4). Starting from the Navier-Stokes equations for the 2D problem will be described with the corresponding boundary conditions (BC). Then, the steady state solution will be presented; it will be assumed that the flow has peculiar characteristics so that the derivatives along certain directions can be neglected.

After that, the focus will be shifted to the area that the jets impinge upon, so the gas pressure and shear stress are imposed by the external airflow.

Before introducing the integral model, the equations are scaled with reference quantities; this allows us to understand which terms are the dominant ones, considering the physics and geometry at play.

The model investigated exploits BC in the 'long-wave' form, derived by scaling the cross streamwise direction with a reference length $[h]$, which is much smaller than the streamwise reference length $[x]$. This is because, in reality, the liquid film layer, corresponding to the zinc in the galvanizing process, is extremely thin compared to the dimensions of the strip it is on. From that, some assumptions will be made in order to arrive at the integral boundary layer model (IBL). The first part of the chapter follows the work done in Mendez, Gosset, and Buchlin [15]. These models of lower dimensionality proved to be useful in simulating such industrial processes, which are not yet accessible by direct high-fidelity simulations because of the prohibitive computational cost. With this approach, the dynamics of the liquid film flow are described in terms of film thickness streamwise and spanwise flow rates, as opposed to the Navier-Stokes equations where the film thickness, pressure and velocity fields must be computed (Mendez et al. [16]). So far, for matters of simplicity, only two directions have been considered, but it has also been extended to the 3D case in the paper byIvanova et al. [12], and its validity has been assessed by Barreiro-Villaverde et al. [1]. Until now, the control has been carried out by gas jets; however, magnets can also exert a force on the molten zinc. This introduces new terms, but the modus operandi for modelling the equations remains analogous to the jets' case. Finally, in the last section of the chapter, the actuators, jets and magnets are taken into consideration, showing how the pressure, shear stress and magnetic field are represented.

1.1 Scaling quantities

The liquid film thickness is much smaller than the film extension. This leads to the assumption that the streamwise $[x]$ (along x) and spanwise $[z]$ (along z) scales are much bigger than the film thickness scale $[h]$:

$$\epsilon = [h]/[x] \ll 1, \qquad \epsilon = [h]/[z] \ll 1, \tag{1.1}$$

where ϵ is the *film parameter*. This implies that the terms with a higher power of ϵ will be negligible with respect to $\mathcal{O}(1)$ or $\mathcal{O}(\epsilon^1)$. The stramwise velocity scale $[u]$ and the film thickness scales are defined as:

$$[u] = U_p \qquad\qquad [h] = \sqrt{\frac{U_p \nu}{g}}. \tag{1.2}$$

Based on these quantities, we can calculate the remaining scaling quantities which are reported in table 1.1.

Reference Quantity	Definition	Expression
$[h]$	$(\nu_l [u]/g)^{1/2}$	$(\nu_l U_p/g)^{1/2}$
$[x]$	$[h]/\varepsilon$	$(\nu_l U_p/g)^{1/2} Ca^{-1/3}$
$[u]$	U_p	U_p
$[v]$	εU_p	$U_p Ca^{1/3}$
$[p]$	$\rho_l g [x]$	$(\mu_l \rho_l g U_p)^{1/2} Ca^{-1/3}$
$[\tau]$	$\mu_l [u]/[h]$	$(\mu_l \rho_l g U_p)^{1/2}$
$[t]$	$[x]/[u]$	$(\nu_l/U_p g)^{1/2} Ca^{-1/3}$

TABLE 1.1: Scaling reference quantities

Performing the scaling operation the set of equations becomes

$$\partial_{\hat{x}} \hat{u} + \partial \hat{v}_{\hat{y}} = 0 \tag{1.3}$$

$$\epsilon Re \left(\partial_{\hat{t}} \hat{u} + \hat{u} \partial_{\hat{x}} \hat{u} + \hat{v} \partial_{\hat{y}} \hat{u}\right) = -\partial_{\hat{x}} \hat{p}_l + \partial_{\hat{y}\hat{y}} \hat{u} + 1 \tag{1.4}$$

$$0 = \partial_{\hat{y}} \hat{p}_l \tag{1.5}$$

1.2 Liquid Film Model Overview

The first case to be considered is the one where jets impinge of the liquid surface. The effect of such jets are a pressure $p_g(x,t)$ and a shear stress distribution $\tau_g(x,t)$ that varies along the x axis of the plate, see figure 1.1

$$\epsilon = \frac{[h]}{[x]} \ll 1$$

In the adopted notation, the values in square brackets are the reference quantities and the ones with the hat are the dimensionless variables (e.g. $h/[h] = \hat{h}$). The liquid film flow in isothermal conditions is governed by the Navier-Stokes equations. For a 2D incompressible flow with constant properties (density ρ_l, dynamic viscosity μ_l and surface tension σ), the continuity and the momentum equations along x and y read:

$$\partial_x u + \partial_y v = 0 \tag{1.6}$$

$$\rho_l \left(\partial_t u + u \partial_x u + v \partial_y u\right) = -\partial_x p_l + \mu_l \left(\partial_{xx} u + \partial_{yy} u\right) + \rho_l g \tag{1.7}$$

$$\rho_l \left(\partial_t v + u \partial_x v + v \partial_y v\right) = -\partial_y p_l + \mu_l \left(\partial_{xx} v + \partial_{yy} v\right) \tag{1.8}$$

The subscript "l" refers to the liquid, while "g" to the gas.

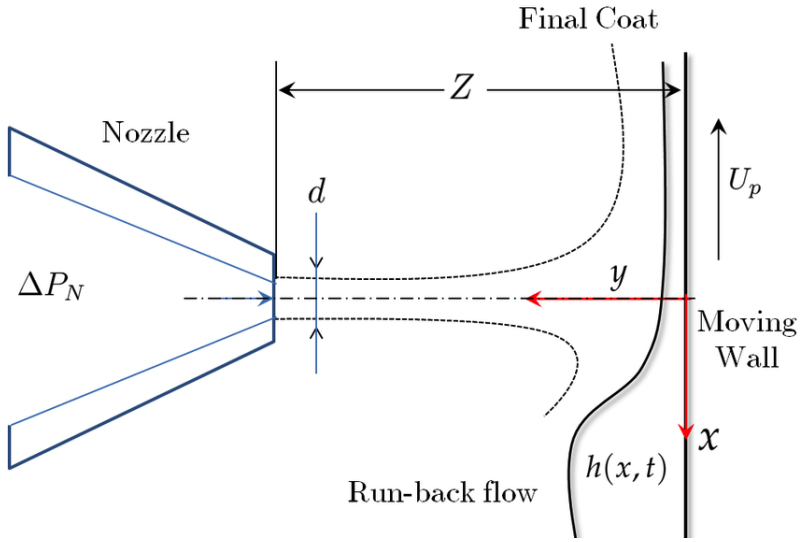

FIGURE 1.1: A schematic of the jet wiping process where the nozzle and the substrate moving with speed U_p are shown

The kinematic boundary conditions at the wall and at the gas-liquid interface set are:

$$\begin{cases} (u,v) = (-1,0) & in\ y = 0 \\ v = \partial_t h + u\partial_x h & in\ y = h \end{cases} \tag{1.9}$$

The film flow is bounded by the wall at $y = 0$ and by a moving interface at $y = h(x,t)$. The boundary conditions at the wall are kinematic and ensure that the flow does not slip or permeate the moving substrate. The second equation of the set ensures that the interface between the two fluids remains continuous as seen by a Eulerian observer. The interface shape is ultimately determined by this latter expression.

This set of equations governs the 2D problem, no assumptions have been introduced a part from considering constant material proprieties.

1.2.1 Steady State solutions

Far Field

The domain can be considered as made up by two different areas: one close to the impinging point and one far from it. Where the influences of the jets can be neglected it is possible to approximate the solution as with the steady state. There the height of the film is not changing and so $\partial_t \approx 0$ and it is also flat, $\partial_x \approx 0$. It also follows that the velocity has only one component along x, so $v = 0$. The equation 1.6 becomes

$$0 = \nu_l \partial_{yy} u + g \tag{1.10}$$

Now integrating twice along y and using the BCs allows to obtain the velocity profile

$$u(y) = \int \left(\int \frac{g}{vl} dy \right) dy = -\frac{1}{2}\frac{g}{v} y^2 + \frac{g}{\nu_l} hy - U_p \tag{1.11}$$

The velocity profile far from the impinging jets, or in a steady state, is parabolic. The flow rate is obtained by integrating this relation

$$q = \int_0^h u(y) = \frac{1}{3}\frac{g}{v_l}h^3 - hU_p \tag{1.12}$$

This last equation set some boundaries on the values of h since if it exceeds the critical value then the $q < 0$, but this is not physical since it would mean that more liquid would be falling than being pulled by the substrate.

Impinging Area

Let us now consider what happens in the other region, the one close to the impinging jets. Since the jet is now acting on the fluid it is not anymore possible to assume that the $\partial_x \approx 0$. This leads to the 1D set of equations

$$\partial_x u = 0 \tag{1.13}$$

$$0 = -\frac{1}{\rho}\partial_x p_l + v_l\left(\partial_{yy} u\right) + g \tag{1.14}$$

$$0 = -\partial_y p_l \tag{1.15}$$

Moreover the kinematic condition, under the assumption of a gently changing profile ($\partial_x h \approx 0$), can be expressed as

$$p_l - p_g(x) + \sigma\partial_{xx} h = 0; \quad \text{and} \quad \partial_y u = \frac{1}{\mu}\tau_g(x) \tag{1.16}$$

Finally combining 1.16 and 1.15 the velocity along y and the corresponding flow rate are determined as in Mendez, Gosset, and Buchlin [15]. This makes it possible to compute them with a solver. This simplified case shares the same process with the higher order models shown in the upcoming sections.

1.2.2 The Integral Formulation of the Unsteady Problem

Once the set of dimensionless equations is provided it is necessary to integrate the equations, however this requires some assumptions on the shape of the velocity profile. The purpose of an integral model is to write these two contributions as function of the flow rate $\hat{q}$ and film thickness $\hat{h}$.

it is possible to obtain the following integral form of the continuity equation

$$\int_0^{\hat{h}(\hat{x},\hat{t})} \left(\partial_{\hat{x}}\hat{u} + \partial v_{\hat{y}}\right) d\hat{y} = \partial_{\hat{x}}\hat{q} + \partial_{\hat{t}}\hat{h} = 0 \tag{1.17}$$

The integration for 1.4 with the pressure written as

$$\hat{p}_l|_{\hat{h}} = \hat{p}_g - \frac{\epsilon^3}{Ca}\partial_{\hat{x}\hat{x}}\hat{h} \tag{1.18}$$

yields [15]

$$\epsilon Re\left(\partial_{\hat{t}}\hat{q}+\partial_{\hat{x}}\mathcal{F}\right)=\hat{h}\left(1-\partial_{\hat{x}}\hat{p}_g+\partial_{\hat{x}\hat{x}\hat{x}}\hat{h}\right)+\hat{\tau}_g+\hat{\tau}_w \tag{1.19}$$

Where $\mathcal{F}=\partial_{\hat{x}}\int_0^{\hat{h}(\hat{x},\hat{t})}\hat{u}^2d\hat{y}$ is the integral advection term. This is the last missing term, but to integrate the $\hat{u}$ it is necessary to make an assumption regarding its distribution. As seen in equation 1.11 the velocity profile in the steady state case has a parabolic shape. If it can be assumed that the newly introduced terms, inertia and surface tension, do not affect this balance in a significant way, then the profile for the unsteady case has the same quadratic trend. The constants depend on $\hat{h}$ and $\hat{q}$, and therefore change in space and time. By exploiting the BC two of the three constants can be determined, leading to

$$\hat{q}=\int_0^{\hat{h}}\left(\frac{1}{2}C_1\hat{y}^2+(\hat{\tau}_g-C_1\hat{h})\hat{y}-1\right)dy \tag{1.20}$$

From this equation C_1 is obtained and with that the final velocity profile and integrating it results in the following advection term

$$\mathcal{F}(\hat{h},\hat{q})=\int_0^{\hat{h}}\hat{u}^2d\hat{y}=\frac{24\hat{h}^2+48\hat{h}\hat{q}+144\hat{q}^2+\left(6\hat{h}^2\hat{q}+6\hat{h}^3\right)\hat{\tau}_g+\hat{h}^4\hat{\tau}_g^2}{120\hat{h}}. \tag{1.21}$$

the shear stress on the other hand has the expression:

$$\tau_w=-\partial_{\hat{y}}\hat{u}=\frac{\hat{h}^2\hat{\tau}_g-6\hat{q}-6\hat{h}}{2\hat{h}^2} \tag{1.22}$$

By substituting this last couple of relations in 1.19 the 2D model for the jet wiping case is complete. To summarize, the final equations set governs the 2D isothermal jet case with constant properties and one-way coupling between the gas flow and the liquid flow. In this approach, the impinging gas jet enters into the modeling solely in terms of distributed (and possibly time dependent) sources of pressure gradient and shear stress.

1.2.3 Vectorial Representation

The set of equations 1.17 and 1.19 can be written in the following manner

$$\partial_t\mathbf{V}(x,t)+\nabla\cdot\mathbf{F}(x,\mathbf{V})=S(x,t,\mathbf{V}) \tag{1.23}$$

The state vector is represented by:

$$\mathbf{V}=\begin{bmatrix}\hat{h}\\ \hat{q}\end{bmatrix} \tag{1.24}$$

The flux term is

$$\mathbf{F}=\begin{bmatrix}-\hat{q}\\ -\hat{\mathcal{F}}(\hat{h},\hat{q})\end{bmatrix} \tag{1.25}$$

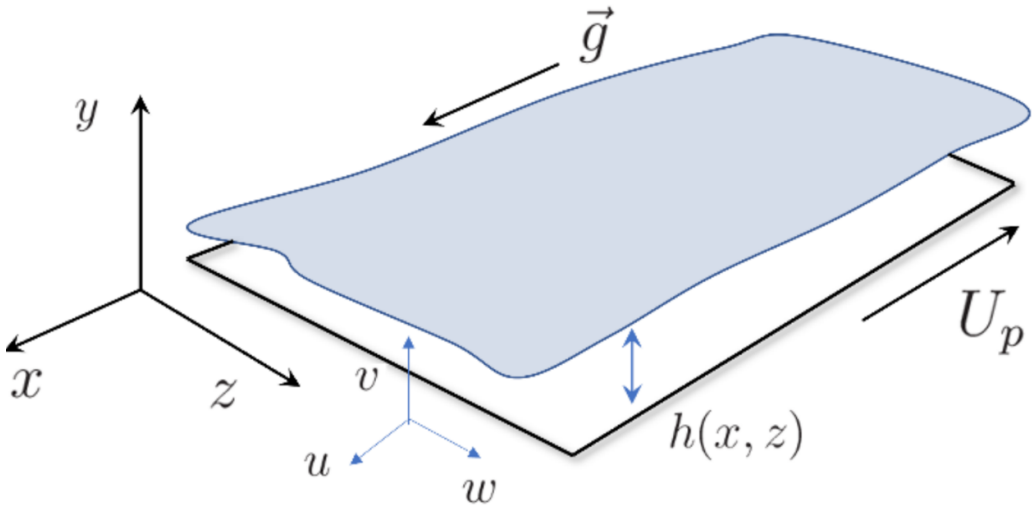

FIGURE 1.2: Sketch of the flow domain for a liquid film on a vertically moving substrate

Finally the source term is

$$\mathbf{S} = \begin{bmatrix} 0 \\ \hat{h}\left(1 - \partial_{\hat{x}}\hat{p}_g + \partial_{\hat{x}\hat{x}\hat{x}}\hat{h}\right) + \hat{\tau}_g + \hat{\tau}_w \end{bmatrix} \tag{1.26}$$

This allows for the numerical solving of the set of PDEs through finite volumes or spectral methods.

1.2.4 The 3D Integral Formulation and Validation

It is possible to further extend the model to the 3D case, refer to Ivanova et al. [12] for further details. In this case the set of equations would become

$$\partial_{\hat{x}}\hat{u} + \partial\hat{v}_{\hat{y}} + \partial_{\hat{z}}\hat{w} = 0 \tag{1.27}$$

$$\epsilon Re\left(\partial_{\hat{t}}\hat{u} + \hat{u}\partial_{\hat{x}}\hat{u} + \hat{v}\partial_{\hat{y}}\hat{u} + \hat{w}\partial_{\hat{z}}\hat{u}\right) = -\partial_{\hat{x}}\hat{p}_l + \partial_{\hat{y}\hat{y}}\hat{u} + 1 \tag{1.28}$$

$$0 = \partial_{\hat{y}}\hat{p}_l \tag{1.29}$$

$$\epsilon Re\left(\partial_{\hat{t}}\hat{w} + \hat{u}\partial_{\hat{x}}\hat{w} + \hat{v}\partial_{\hat{y}}\hat{w} + \hat{w}\partial_{\hat{z}}\hat{w}\right) = -\partial_{\hat{z}}\hat{p}_l + \partial_{\hat{y}\hat{y}}\hat{w} \tag{1.30}$$

The kinematic BCs become

$$\begin{cases} (\hat{u},\hat{v},\hat{w}) = (-1,0,0) \quad in\ \hat{y} = 0 \\ \hat{v} = \partial_{\hat{t}}\hat{h} + \hat{u}\partial_{\hat{x}}\hat{h} + \hat{w}\partial_{\hat{z}}\hat{h} \quad in\ \hat{y} = \hat{h} \end{cases} \tag{1.31}$$

while the dynamic BC, in $\hat{y} = \hat{h}$, considering only the terms $\mathcal{O}(\epsilon^1)$ is

$$\hat{p} = \hat{p}_g - \left(\partial_{\hat{x}\hat{x}}\hat{h} + \partial_{\hat{z}\hat{z}}\hat{h}\right) \tag{1.32}$$

$$\partial_{\hat{y}}\hat{u} = \hat{\tau}_{g,x} \tag{1.33}$$

$$\partial_{\hat{y}}\hat{w} = \hat{\tau}_{g,z} \tag{1.34}$$

A process similar to the 2D case is followed for the integration, where a self-similar parabolic velocity profile for both the streamwise $\hat{u}$ and the spanwise $\hat{w}$ velocity components is assumed.

$$\hat{u}\left(\hat{h}, \hat{q}_x, \hat{q}_z\right) = \frac{3}{4\hat{h}^3}\left(\hat{\tau}_{g,x}\hat{h}^2 - 2\hat{h}^{-2}\hat{q}_x\right)\hat{y}^2 \tag{1.35}$$

$$+ \frac{6\hat{h} + 6\hat{q}_x - \hat{\tau}_{g,x}\hat{h}^2}{2\hat{h}^2}\hat{y} - 1, \tag{1.36}$$

$$\hat{w}\left(\hat{h}, \hat{q}_x, \hat{q}_z\right) = \frac{3}{4\hat{h}^3}\left(\hat{\tau}_{g,z}\hat{h}^2 - 2\hat{q}_z\right)\hat{y}^2 \tag{1.37}$$

$$+ \frac{6\hat{q}_z - \hat{\tau}_{g,z}\hat{h}^2}{2\hat{h}^2}\hat{y}. \tag{1.38}$$

The source terms are

$$\vec{S} = \begin{pmatrix} 0 \\ \frac{1}{\delta}\left[\hat{h}\left(-\partial_{\hat{x}}\hat{p}_x + \partial_{\hat{x}\hat{x}\hat{x}}\hat{h} + \partial_{\hat{x}\hat{z}\hat{z}}\hat{h} + 1\right) + \Delta\hat{\tau}_x\right] \\ \frac{1}{\delta}\left[\hat{h}\left(-\partial_{\hat{z}}\hat{p}_z + \partial_{\hat{z}\hat{z}\hat{z}}\hat{h} + \partial_{\hat{z}\hat{x}\hat{x}}\hat{h}\right) + \Delta\hat{\tau}_z\right] \end{pmatrix} \tag{1.39}$$

The shear stress at the wall

$$\hat{\tau}_{w,x} = \frac{1}{2}\hat{\tau}_{g,x} - \frac{3\hat{q}_x}{\hat{h}^2} + \alpha\frac{3}{\hat{h}} \tag{1.40}$$

$$\hat{\tau}_{w,z} = \frac{1}{2}\hat{\tau}_{g,z} - \frac{3\hat{q}_z}{\hat{h}^2} \tag{1.41}$$

Finally the flux matrix contains has components

$$\mathbf{F} = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \end{pmatrix} = \begin{pmatrix} \hat{q}_x & \int_0^{\hat{h}} \hat{u}^2 d\hat{y} & \int_0^{\hat{h}} \hat{u}\hat{w} d\hat{y} \\ \hat{q}_z & \int_0^{\hat{h}} \hat{u}\hat{w} d\hat{y} & \int_0^{\hat{h}} \hat{w}^2 d\hat{y} \end{pmatrix} \tag{1.42}$$

The different terms are determined as

$$F_{11} = \int_0^{\hat{h}} \hat{u} d\hat{y} \tag{1.43}$$

$$F_{21} = \int_0^{\hat{h}} \hat{w} d\hat{y} \tag{1.44}$$

$$F_{12} = \frac{1}{120\hat{h}} \left(144\hat{q}_x^2 + 6\hat{\tau}_{g,x}\hat{h}^2\hat{q}_x + \hat{\tau}_{g,x}\hat{h}^4 \right. \tag{1.45}$$

$$\left. -\alpha \left(48\hat{h}\hat{q}_x + 6\hat{\tau}_{g,x}\hat{h}^3 + 24\hat{h}^2 \right) \right) \tag{1.46}$$

$$F_{22} = \frac{1}{120\hat{h}} \left(144\hat{q}_x\hat{q}_z + 3\hat{\tau}_{g,x}\hat{h}^2\hat{q}_z + 3\hat{\tau}_{g,z}\hat{h}^2\hat{q}_x \right. \tag{1.47}$$

$$\left. +\hat{\tau}_{g,x}\hat{\tau}_{g,z}\hat{h}^4 - \alpha \left(24\hat{h}\hat{q}_z + 3\hat{\tau}_{g,z}\hat{h}^3 \right) \right) \tag{1.48}$$

$$F_{13} = F_{22} \tag{1.49}$$

$$F_{23} = \frac{144\hat{q}_z^2 + 6\hat{\tau}_{g,z}\hat{h}^2\hat{q}_z + \hat{\tau}_{g,z}^2\hat{h}^4}{120\hat{h}} \tag{1.50}$$

This is the model implemented in the spectral 3D BLEW solver. The model robustness has been studied in Barreiro-Villaverde et al. [1], where the IBL model was integrated using the Finite Volume BLEW. The latter combines the two-step second-order Lax-Wendroff and the two-step first-order Lax- Friedrich schemes. The model was validated with and without the surface tension term. An interesting observation, made by the author, is that technically the term ϵRe exceeds $\epsilon Re \sim \mathcal{O}(1)$, this means that the asymptotic expansion is not guaranteed to hold, however the results between IBL and DNS still seem to agree. The paper concludes that using the DNS the mesh had to fulfill stricter restrictions, in terms of cell size. Despite minor differences in the profiles, the consistency was deemed acceptable as the shape, amplitude, and speed of the waves remained largely unchanged in both scenarios. Additionally, both solvers successfully depicted the initial wave growth, followed by capillary damping. Finally it is important to highlight that the computational expense of DNS is significantly higher than that of IBL simulations, owing to mesh demands and the intricate nature of the governing equations.

1.3 Magnetic Model

Impinging on the liquid film through gas jets it is not the only way to exert a force on the flow. When molten zinc is exposed to a magnetic field, it begins to conduct a small amount of electricity, which creates an oppositely charged magnetic field. Since the two fields are oppositely charged, they repel each others, causing the liquid zinc to move away from the magnet. So a new term has to be considered in the steady state equation 1.6, thus becoming

$$0 = -\frac{1}{\rho_l}\partial_x p_l + \nu_l(\partial_{yy}u) + g - \frac{\sigma_M}{\rho_l}B^2 u \tag{1.51}$$

$$\tag{1.52}$$

The magnetic terms is caused by the Lorentz force, since the film is in relative motion with respect to the magnetic field, **B**. Generally the Lorentz force, in absence of an electric field, is

written as

$$\mathbf{F_L} = q(\dot{\mathbf{x}} \times \mathbf{B}) \tag{1.53}$$

where $\dot{\mathbf{x}}$ is the velocity vector of a charged particle. Considering only a infinitesimal volume and writing it in terms of current density $\mathbf{j} = \rho\dot{\mathbf{x}}$, where ρ is the charge density.

$$\mathbf{j} = \sigma_m(\dot{\mathbf{x}} \times \mathbf{B}) \tag{1.54}$$

By also neglecting the electric potential difference the equation of the Lorentz force for the unit volume becomes

$$\mathbf{f_L} = \mathbf{j} \times \mathbf{B} = (f_{L,x}, 0, f_{L,z})^T = (-\sigma_M B^2 u, 0, -\sigma_M B^2 w)^T \tag{1.55}$$

The boundary and dynamic conditions remain unaltered and once again the non-dimensional quantities are introduced in the expression

$$0 = -\frac{[p]}{[x]}\partial_{\hat{x}}\hat{p}_g + \frac{\sigma[h]}{[x]^3}\partial_{\hat{x}\hat{x}\hat{x}}\hat{h} + \mu_l\frac{U_p}{[h]^2}\partial_{\hat{y}\hat{y}}\hat{u} + \rho_l g - \sigma_M[B]^2 U_p \hat{B}^2\hat{u}. \tag{1.56}$$

$[B]$ is assumed as the maximum value of the magnetic field ($[B] = sup(B(x,t))$. All terms are then divided by the gravitational term $\rho_l g$ and ϵ is introduced. Finally taking $[p] = \rho_l g[x]$, $[\tau_g] = \mu_l U_p/[h]$ and the definition of h according to 1.1, as in the jet formulation, the following dimensionless version of equation 1.52 is given:

$$0 = -\partial_{\hat{x}}\hat{p}_g + \frac{\epsilon^3}{Ca}\partial_{\hat{x}\hat{x}\hat{x}}\hat{h} + \partial_{\hat{y}\hat{y}}\hat{u} + 1 - Ha_0^2\hat{B}^2\hat{u}, \tag{1.57}$$

Ha is the Hartmann number, weighting the importance of the electromagnetic to the gravitational force. This steady state equation can be seen as a wave equation with a constant term A_1

$$A_1 = -\partial_{\hat{x}}\hat{p}_g + \frac{\epsilon^3}{Ca}\partial_{\hat{x}\hat{x}\hat{x}}\hat{h} \tag{1.58}$$

Meaning that the equation can be written as

$$0 = A_1 + \partial_{\hat{y}\hat{y}}\hat{u} + 1 - Ha_0^2\hat{B}^2\hat{u} \tag{1.59}$$

which has an homogeneous solution in the form

$$\hat{u}(\hat{y}) = k_1 e^{\left(Ha_0\hat{B}\hat{y}\right)} + k_2 e^{\left(-Ha_0\hat{B}\hat{y}\right)} + \frac{A_1 + 1}{Ha_0^2\hat{B}^2}, \tag{1.60}$$

The k_1 and k_2 terms can be computed starting from the boundary conditions. It is now possible to write an expression for the velocity distribution $\hat{u}(\hat{y})$, from which also the flow rate per unit width is obtained ($\hat{q} = \int_0^{\hat{h}} \hat{u}(\hat{y}')\ d\hat{y}'$). This eventually leads to an expression for $\hat{u}(\hat{y})$ for the steady state. Moving on to the unsteady case the equation to be solved is

$$\partial_{\hat{t}}\hat{u} + \hat{u}\partial_{\hat{x}}\hat{u} + \hat{v}\partial_{\hat{y}}\hat{u} = -\partial_{\hat{x}}\hat{p}_l + \partial_{\hat{y}\hat{y}}\hat{u} + 1 - Ha_0^2\hat{B}^2\hat{u} \tag{1.61}$$

However an assumption has to be made and, as for the jet wiping model, it is assumed that the velocity profile has the same shape as in the steady case

$$\hat{u}(\hat{y}) = k_{1,u}e^{H_{a_0}\hat{B}\hat{y}} + k_{2,u}e^{-H_{a_0}\hat{B}\hat{y}} + k_{3,u}\frac{A_1+1}{H_{a_0}^2\hat{H}^2} \quad \text{with } A_1 = -\partial_{\hat{x}}\hat{p}_g + \frac{\epsilon^3}{Ca}\partial_{\hat{x}\hat{x}\hat{x}}\hat{h} \tag{1.62}$$

This means that the inertial forces have a small influence over said velocity distribution. The final set of equations that have been coded into the 3D spectral solver is shown in appendix A. In the code implementation it was noticed that a fully magnetic model, that is one that solves the equations introduced in this section, would not be stable. This is probably due to the fact that the magnetic field far from the magnets takes on values close to zero, causing numerical issues. To solve this problem different approaches have been tested. The one that seemed to strike the best balance between stability and adherence to the mathematical model is the one chosen for the simulations. The flux terms of 1.23 have been computed assuming that the magnetic field has a marginal influence, thus neglecting it. This means that the magnetic field only appears in the source term. In these terms the square of the magnetic field is added to the gas pressure gradient, to the surface tension, to gravitational and to the wall shear stress terms. As shown in the appendix A the wall shear stresses are also functions of the magnetic field.

1.4 Actuators Characterization

As mentioned before, it is being assumed that the jets impinge on the liquid film without being influenced by the flow itself. This is a consequence of the fact that the distance between the plate and jet nozzle H is much bigger than the liquid film thickness h. These actuators determine a pressure and shear distribution on the flow, following empirical relations that will be illustrated in the upcoming sections.

1.4.1 Jet Characterization

Impinging circular jets are a special type of jets that is commonly used in jet wiping and characterized by their circular shape and perpendicular impact on the surface to be wiped. As it will be seen later those distribution can be computed through empirical models for which it is necessary to know the gas velocity.

The process for obtaining the gas velocity value starts by assuming that both, the maximum pressure and the shear stress, depend exclusively on the nozzle gauge stagnation pressure, ΔP_N , the nozzle opening, d, its discharge coefficient, C_d, and the standoff distance, H. The losses are expressed as

$$C_D^2 = \frac{1}{2}\frac{\rho U_j^2}{\Delta P_N} = \frac{U_j^2}{U_{id}^2} \tag{1.63}$$

Generally, for wiping jets, C_D as a value of 0.8, it has to be less than one.

Pressure and shear stress distribution

To model the height of the liquid film, firstly one must evaluate how the jet wiping interacts with the liquid coating. With an experimental facility one can to compute the relationship of shear stress and pressure distribution for circular jets impinging in a strip. They both depend

on the value of stand-off distance and the diameter of the nozzle. For this kind of application it is $H/d \geq 6$, which is a compromise between wiping performance and the nozzle gas flow rate. Another crucial parameter that must be set up is the gauge of pressure in the nozzle, so the project variables are ΔP and H/D. The pressure and shear stress distribution laws at the wall interface are modeled according to the semi-empirical relations proposed in the paper by Beltaos and Rajaratnam [2], according to which

$$p_w = p_{max} e^{-144\lambda^2} \tag{1.64}$$

(1.65)

while the shear stress distribution is given by

$$\tau_w = \tau_{max} \left[0.18 \frac{1 - e^{-144\lambda^2}}{\lambda} - 9.43\lambda e^{-144\lambda^2} \right] \tag{1.66}$$

The parameter $\lambda = r/H$ is defined as the ratio between the distance from the impinging point r and H. The pressure has a maximum value directly beneath the impinging jet and then it decreases. On the other hand the maximum shear stress is located at some distance from the impinging point. It is possible to notice that the maximum pressure and shear stress are only functions of three parameters, that is

$$p_{max} = p_{max}(U_j, H, d) = 50 \frac{\rho_g U_j^2}{2} \left(\frac{H}{d} \right)^{-2} \tag{1.67}$$

$$\tau_{max} = \tau_{max}(U_j, H, d) = 0.16 \rho_g U_j^2 \left(\frac{H}{d} \right)^{-2} \tag{1.68}$$

These empirical equations approximate the quantities in the impingement region of the jet ($\lambda < 0.22$), where the wall shear and the pressure gradients are most significant. In the actual code implementation an additional artificial term has been introduced in the shear distribution. The term is the square of λ, this allows for a distribution closer to the experimental one and since the values under a certain range are set to zero, this also assures that the jets do not influence themselves too much, as the shear stress goes to zero more rapidly.

$$\tau_{g,mod} = \max \left(0, \tau_{max} \left(0.18 \left(\frac{1 - e^{-114\lambda^2}}{\lambda} \right) - 9.43\lambda e^{-114\lambda^2} - \lambda^2 \right) \right) \tag{1.69}$$

Equations 1.65 and 1.69 are the expression on which the actuators were implemented in the final solver. However the waves that have to be controlled are only quasi 3D, this lead to the idea that a 2D jet might also have chances of flattening it. Since the changes along the streamwise direction, $\hat{x}$, are much more pronounced than the ones in the spanwise one, $\hat{z}$ is is reasonable to assume that a 2D jet will also perform well. For this reason the equations of the three-dimensional pressure and shear stress distribution were modified by neglecting the terms along $\hat{z}$. The results are distributions that vary only along $\hat{x}$. Alternatively this could have been achieved by putting multiple jets close to one another, however this would unnecessarily increase the computational cost.

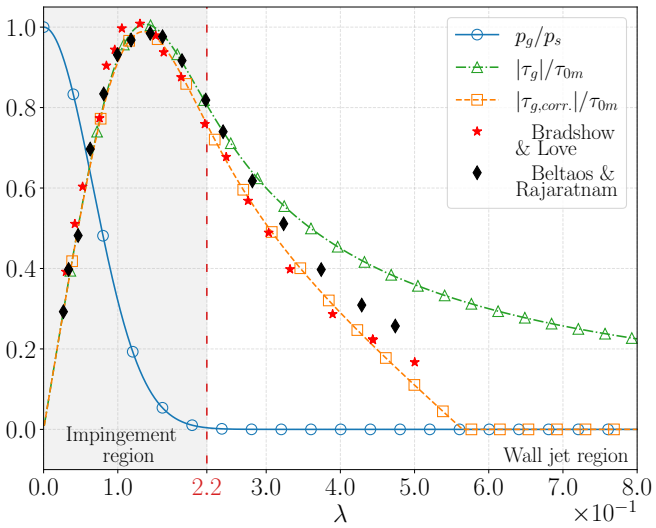

FIGURE 1.3: Pressure (blue continuous line with circles) and shear stress (green dash-dotted line with triangles) distributions at the strip level in the radial direction $\lambda = r/H$ for a planar 2D jet (a) and for a circular 3D jet with the modified shear stress distribution (orange dashed line with squares) and a highlight of the impingement and the wall jet regions (b)

1.5 Magnets Characterization

Just like the jets, the magnets have the effect of causing a force distribution over the domain, the value of which changes in space. The modeling of the magnets is based on the hypothesis that the radial component, the induced field and the hysteresis effects are negligible

$$B = B_{\text{ext}} + B_{\text{ind}} \tag{1.70}$$

The absence of induction can also be seen in terms of magnetic Reynolds defined as

$$Re_m = \mu\sigma_M[u][L] \ll 1 \tag{1.71}$$

The final and most relevant approximation is the fact that the filed has the shape of a Gaussian

$$B(\hat{x},\hat{z},\hat{t}) = B_t(\hat{t})e^{\frac{-(\hat{x}-\hat{x}_0)^2}{2\gamma_{\hat{x}}^2}+\frac{-(\hat{z}-\hat{z}_0)^2}{2\gamma_{\hat{z}}^2}} \tag{1.72}$$

(x_0, z_0) is the center of the Gaussian and (γ_x, γ_z) are the standard deviations along x and z respectively and $B_t(t)$ is bounded between 0 and the maximum prescribed value for the induction field $\sup(B)$. So the shape does not change over time but it is scaled with a factor that is time dependent. A critical point of this modelling is the lack of a hysteresis term in the magnetic field. In reality, the magnetic field does not change instantaneously when we change B_t from one instant to the other; it has a response time linked to the magnetic properties of the solenoid's core material. Although enough for the level of accuracy of this research, a more advanced model should also consider the hysteresis effect Bertotti and Mayergoyz [3] and Przybyłowicz and Szmidt [20].

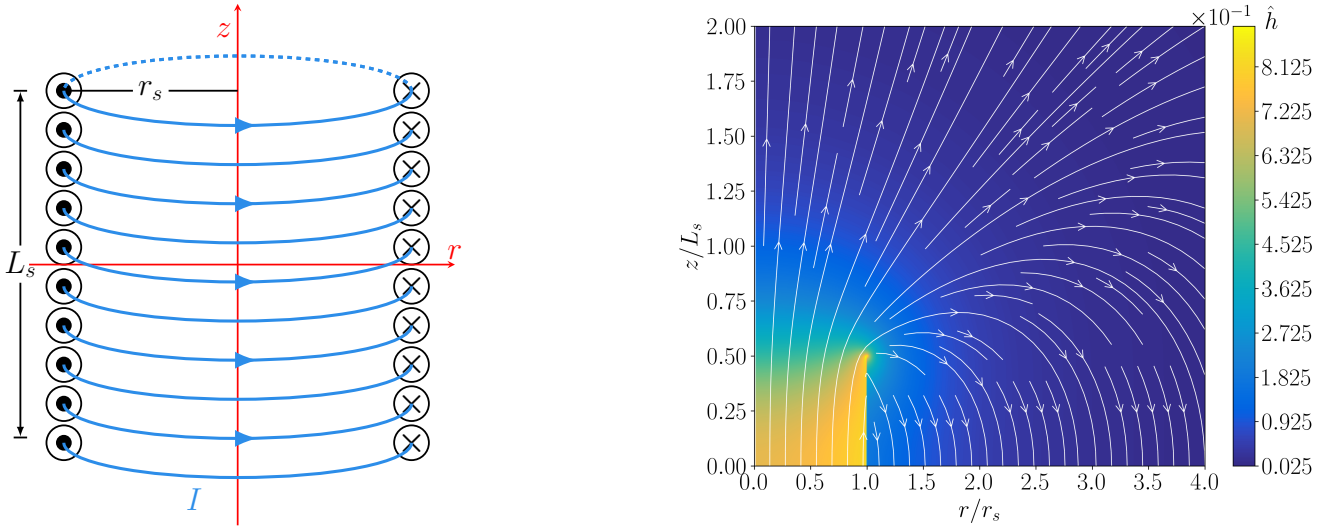

FIGURE 1.4: Scheme of the solenoid of length L_s, radius r_s with a current I with the cylindrical reference frame centred in the middle (a) and the relative magnetic induction field in terms of relative magnetic intensity $|B|/B_0$ and magnetic lines (b)

Chapter 2

Simplified Model Implementation

This chapter focuses on the numerical implementation of the reduced order model presented in Chapter 1 via the Fourier pseudo-spectral method. A previous version of BLEW used finite volume schemes and suffered the computation of the third-order derivatives that appear in the surface tension terms. Moreover, those finite volume schemes had a high numerical diffusion, which negatively affected the liquid film development.

It was, therefore, necessary to find an alternative. As seen in the upcoming sections, spectral methods are really promising options as they can allow for faster and more accurate solutions. This chapter begins with a brief introduction to the Fourier transform, the main mathematical tool behind spectral methods, highlighting its strengths and weaknesses. This approach will be put to the test by showing how the spatial derivatives are computed and how this compares to other conventional methods.

A fast solver is essential when many simulations have to be carried out, as in the case of reinforcement learning. Therefore, in the second part of this chapter it has been investigated how it is possible to use the Fourier transform with the goal of obtaining the Jacobian matrix of a function. This matrix will then be given to a time integrator to speed up the process. The equations in question are not the ones solved in BLEW, but it was decided to test the idea on less complex functions that would still present similar terms. Finally, the integrating factor method was applied to the viscous Burgers' equation in order to provide an additional alternative.

2.1 Fourier Transform

The Fourier Transform converts a function $f(x)$ from its original domain to the frequency domain. This is given by the integral transformation:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x}\, dx, \tag{2.1}$$

where $\hat{f}$ is the transformed function and ξ are the frequencies. For each frequency, the magnitude of the complex value, $\hat{f}(\xi)$, represents the amplitude of a constituent complex sinusoid with that frequency integrated over the domain. The inverse transformation is given by:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\xi) e^{i2\pi\xi x} d\xi, \tag{2.2}$$

In short, the Fourier Transform "dissects" the original function into its underlying frequencies, revealing the contribution of each frequency component. If a particular frequency is not present in the original function, the corresponding value in the transform is zero.

Real signals are measured at discrete points in time or space. The Discrete Fourier Transform (DFT) is an adaptation of the Fourier Transform for such discrete signals. Let us now consider a discrete signal x with N samples. The DFT takes this discrete signal and maps the function values to the frequency domain, similar to the continuous Fourier Transform. However, instead of using an integral, the DFT employs a sum. The DFT of a discrete signal x at frequency k/N is computed using the formula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}}, \tag{2.3}$$

In the adopted formulation the sum starts with $n = 0$, this will make it easier to write the DFT as a matrix in section 2.4 and it is also the way that python indexes variables. The corresponding inverse function is expressed as

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k e^{i2\pi \frac{kn}{N}}. \tag{2.4}$$

Here, k ranges from 0 to $N-1$, representing frequency intervals. Each $\hat{f}_k$ is a complex number that reveals the contribution of the sinusoidal component of frequency k/N in the signal.

The DFT proves to be valuable for numerical estimation and calculation. However, its efficiency diminishes significantly for very large values of N, due to the basic formulation involving multiplication by a non-sparse $N \times N$ matrix, necessitating $\mathcal{O}(N^2)$ operations. The computational cost can be reduced by employing the Fast Fourier Transform (FFT) algorithm, which scales at $\mathcal{O}(Nlog(N))$, meaning that as N grows, the algorithm approaches linear scaling.

Not every wave number can be considered; as a matter of fact, the maximum is determined by the Nyquist-Shannon sampling theorem, which states that to capture frequencies in a signal accurately, the sampling rate must be at least twice the frequency of the highest frequency component. From another perspective, the highest considerable frequency is half the sample frequency. The frequency corresponds to the wave number when the discretization is in space.

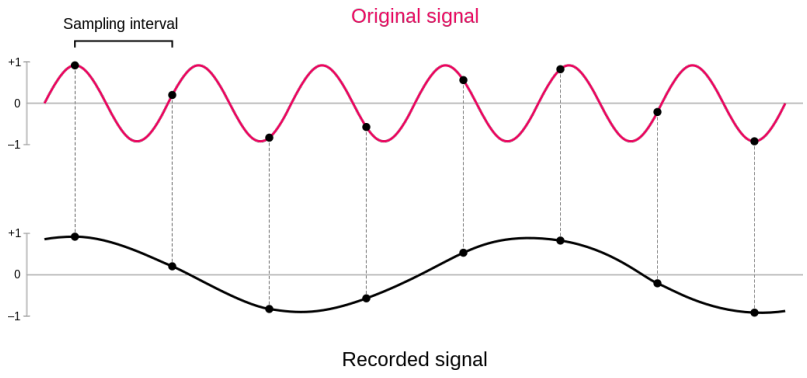

FIGURE 2.1: Aliasing might lead to an incorrectly reconstructed signal if the sampling ratio is not appropriate

So far, the function considered was one-dimensional. However, it is possible to extend it

to multiple dimensions; this will be the case with the BLEW solver. More focus is on the two-dimensional case in section 2.5. An important phenomenon that has to be considered is called *aliasing*, which causes a distortion of the frequency content of a signal, as shown in figure 2.1. Aliasing primarily occurs when the signal contains high-frequency components that exceed the Nyquist frequency. To clarify, consider the following example. Imagine dealing with a continuous signal that carries a pure tone at a frequency of 1000Hz. If this signal is sampled at a rate only slightly above 2000Hz (just above the Nyquist rate for 1000Hz), the Nyquist criterion is fulfilled. However, if there are any higher-frequency components present in the signal beyond 1000Hz, they will "fold back" due to aliasing. These higher-frequency components might manifest as seemingly lower-frequency components in the sampled signal. Imagine a small, additional signal component at 1500Hz in the original continuous signal (which was not initially noticeable), which exceeds the Nyquist frequency. Due to aliasing, this 1500Hz component will fold back and appear as a lower frequency. In the sampled signal, one might detect a component at 500Hz ($2000v - 1500$Hz), which looks like a genuine part of the original signal. This is how the spurious lower-frequency components emerge. They are the result of aliasing, where higher-frequency information is folded back into the lower-frequency range. Employing strategies like low-pass filtering can mitigate this issue. Low-pass filters remove high-frequency components that could lead to aliasing, ensuring that the sampled signal accurately reflects the true composition of the original continuous signal.

2.2 Spectral Methods

Spectral methods are the best numerical methods to solve partial differential equations (PDEs) with high accuracy on a relatively simple domain if the data defining the problem is smooth. They can achieve higher accuracy with respect to the other methods, with the same number of grid points (Dutykh [7]). This ultimately results in less complexity as well as a reduced computational cost. The main idea behind spectral techniques is to write the solution of the differential equation as a sum of certain *basis functions* and then to choose the coefficients in the sum in order to satisfy the differential equation (Canuto et al. [5]).

$$u(x,t) \approx u_n(x,t) = \sum_{k=0}^{N} v_k(t)\phi_k(x) \tag{2.5}$$

where $\phi_k(x)_{k=0}^{\infty}$ are the basis functions. This set of functions should guarantee three general properties:

- The approximation should converge to the solution as the number of grid points increases
- It should be relatively straight forward to determine the sum coefficients
- It should not be computationally expensive to reconstruct the solution starting from the sum weights

In spite of all those advantages, spectral methods assume the periodicity as a BC. This requirement is not explicitly stated, but it is implicit which means that the function that represents the initial condition should be periodic in the physical domain. If this condition is not respected then huge oscillations in the solution are introduced at the boundary of the domain. Having a periodic domain means that the flow that exits on the left enters, at the same time, the domain from the right. To make up for this restrictive hypothesis it is possible to apply a particular

layer to the domain. It is a zone where a strong damping is numerically implemented, such that in these areas waves are flattened. In this way the flow that exits, and then reenters, the domain is therefore free of any kind of perturbations.

2.2.1 Equation discretization

In practice the liquid film height $\hat{h}$, the streamwise $\hat{q}_{\hat{x}}$ and the spanwise $\hat{q}_{\hat{z}}$ flow rates are approximated with $N \times M$ plane waves in two dimensions:

$$\hat{h} \approx \hat{h}^N(\hat{x},\hat{z},\hat{t}) = \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} \tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})} \tag{2.6}$$

$$\hat{q}_{\hat{x}} \approx \hat{q}_{\hat{x}}^N(\hat{x},\hat{z},\hat{t}) = \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} \tilde{q}_{\hat{x}}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})} \tag{2.7}$$

$$\hat{q}_{\hat{z}} \approx \hat{q}_{\hat{z}}^N(\hat{x},\hat{z},\hat{t}) = \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} \tilde{q}_{\hat{z}}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})} \tag{2.8}$$

where $k_{\hat{x}}$ and $k_{\hat{x}}$ are the streamwise and the spanwise wave numbers and $\tilde{H}$, $\tilde{Q}_{\hat{x}}$ and $\tilde{Q}_{\hat{z}}$ are the matrices containing the Fourier coefficients $\tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]}$, $\tilde{q}_{\hat{x}}^{[k_{\hat{x}},k_{\hat{z}}]}$ and $\tilde{q}_{\hat{z}}^{[k_{\hat{x}},k_{\hat{z}}]}$ respectively. The next step consists in computing the derivatives, using the property show in expression 1.23 for the 1D case:

$$\frac{\partial^n \hat{h}^N}{\partial \hat{x}^n} = \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} (ik_{\hat{x}})^n \tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})} \tag{2.9}$$

$$\frac{\partial^n \hat{h}^N}{\partial \hat{z}^n} = \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} (ik_{\hat{z}})^n \tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})} \tag{2.10}$$

Then the residual is projected onto a base of Dirac delta functions, $\delta_{\hat{x}_i,\hat{z}_j}$, and set to zero. Those Dirac functions are defined over an equispaced grid $\hat{x}_i = \frac{2\pi}{N} i$ and $\hat{z}_j = \frac{2\pi}{M} j$ with $i \in \mathbb{N} : i \leq N$ and $j \in \mathbb{N} : j \leq M$. Exploiting the Dirac function properties, the inner product with respect to the residual results into a discrete inverse Fourier transform giving the value of the state quantities at the grid point, also known as collocation points:

$$\langle \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} \tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})}, \delta_{\hat{x}_i,\hat{z}_j} \rangle = \tag{2.11}$$

$$= \int_\Omega \sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}=-M/2}^{M/2-1} \tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}+k_{\hat{z}}\hat{z})} \delta_{\hat{x}_i,\hat{z}_j} \, d\hat{x}d\hat{z} = \tag{2.12}$$

$$= \underbrace{\sum_{k_{\hat{x}}=-N/2}^{N/2-1} \sum_{k_{\hat{z}}-M/2}^{M/2-1} \tilde{h}^{[k_{\hat{x}},k_{\hat{z}}]} e^{i(k_{\hat{x}}\hat{x}_i+k_{\hat{z}}\hat{z}_i)}}_{IDFT} = \hat{h}(\hat{x}_i,\hat{z}_i). \tag{2.13}$$

Hence, a linear system of equations of dimension $N \times M$ is obtained in terms of the state quantities values at the collocation points.

At each time step, the nonlinear terms in the physical space are computed to find the values of the associated Fourier coefficients to then compute the spatial derivative in the wave numbers space. This section summarizes the main mathematical idea behind spectral methods. To further familiarize with these concepts some study cases are provided in the upcoming sections.

2.3 Practical implementation of the spectral methods

Let us now consider the advection equation to see how the Fourier transform can practically be employed in order to discretise in space a differential equation, such as the advection one:

$$\partial_t u + a\partial_x u = 0 \tag{2.14}$$

Just like the finite differences, spectral methods provide a way to evaluate the spatial derivatives. So that the initial equation will be rewritten as

$$\partial_t u(x) = f(u(x)) \tag{2.15}$$

Such an ODE can then be solved in different ways, the simplest of which is an explicit Euler scheme. Other alternatives are the Runge-Kutta schemes or also using the odeint function in python is a viable option. First, let's go back to the spatial discretization. Considering the convection term it is possible to rewrite it as

$$\partial_x u(x) = \mathcal{F}^{-1}\mathcal{F}\left(\partial_x u(x)\right) \tag{2.16}$$

However it is possible to see that

$$\mathcal{F}(\partial_x u) = ik_x\mathcal{F}(u) \tag{2.17}$$

Where i is the imaginary unit. This is because

$$\mathcal{F}(f(x)) = \int_{-\infty}^{\infty} f(x)e^{-ikx}dx \tag{2.18}$$

$$\mathcal{F}\left(\partial_x f(x)\right) = \int_{-\infty}^{\infty} \partial_x f(x)e^{-ikx}dx \tag{2.19}$$

Using integration by parts, the above integral can be expressed as:

$$\mathcal{F}\left(\partial_x f(x)\right) = \left[f(x)e^{-ikx}\right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(x)(-ik)e^{-ikx}dx \tag{2.20}$$

Assuming that $f(x)$ approaches zero as $|x| \to \pm\infty$, the boundary terms vanish:

$$\mathcal{F}\{\partial_x f(x)\} = \int_{-\infty}^{\infty} f(x)(ik)e^{-ikx}dx = ik\mathcal{F}(f(x)) \tag{2.21}$$

So the original equation 2.14 can be written as

$$\partial_t u = a\mathcal{F}^{-1}\left[ik\mathcal{F}(u)\right] \tag{2.22}$$

Which is in the form of equation 2.16. On a practical level the right hand side can be computed using built in functions of numpy for calculating the direct and inverse Fourier transform and by multiplying the result with the wave number vector.

2.3.1 Derivative computation

As seen in the previous section, Fourier transforms can be employed to compute the spatial derivatives of a function. Other methods, such as the finite differences, might require a higher number of points to reach the same accuracy, although they might be best suited if the solution presents sharps changes. In this example a Gaussian function in the middle of the domain is derived first analytically, than with the finite differences and finally with the Fourier transform. It is important to notice that the equation is periodic on the considered domain and that the domain itself is $x \in [0, 2\pi]$. The spectral derivative is computed by using equation 2.17 and then taking the inverse transform to go back to the original domain. The code 2.2 also shows how to perform the transform by multiplying a vector by the DFT matrix, more on that in section 2.4. The Fourier derivative is more accurate than the finite difference one, this is because it is taking information from the whole domain to compute the basis function coefficients. It is worth noting that with the same number of points the FD method would be faster than the spectral one. One final remark is that the spectral scheme does not provide the solution only in the collocation points but in the whole domain, because the test sinusoidal functions are defined everywhere in the domain.

LISTING 2.1: Python code for computing the derivative of a periodic function

```
import numpy as np
import matplotlib.pyplot as plt

# Define domain parameters
L = 2 * np.pi
n_x = 10
dx = L / n_x
x = np.linspace(0,  L, n_x, endpoint=False)

# Initial Condition: Cosine function, which is periodic on the
   domain [0, L]
u0 = np.exp(-(x-np.pi) ** 2 / 0.5)

# Calculate Analytical Derivative
finite_dif = np.gradient(u0, dx)


# Calculate Spectral Derivatives using different methods

# Method 1: Using Fourier Transform matrices
N = len(x)
k = np.fft.fftfreq(N, dx)
```

```
W = np.fft.fft(np.eye(n_x))
W_bar = np.fft.ifft(np.eye(n_x))
spectral_derivative1 = (W_bar @ ((1j * k) * (W @ u0))) * L

# Method 2: Using FFT directly
spectral_derivative2 = np.fft.ifft(1j * k * np.fft.fft(u0)) * L

# Analythical solution
x_an = np.linspace(0,  L, 10*n_x, endpoint=False)
u0_an = np.exp(-(x_an-np.pi) ** 2 / 0.5)
analytical_derivative = -4*(x_an-np.pi)*u0_an
# Plotting ...
```

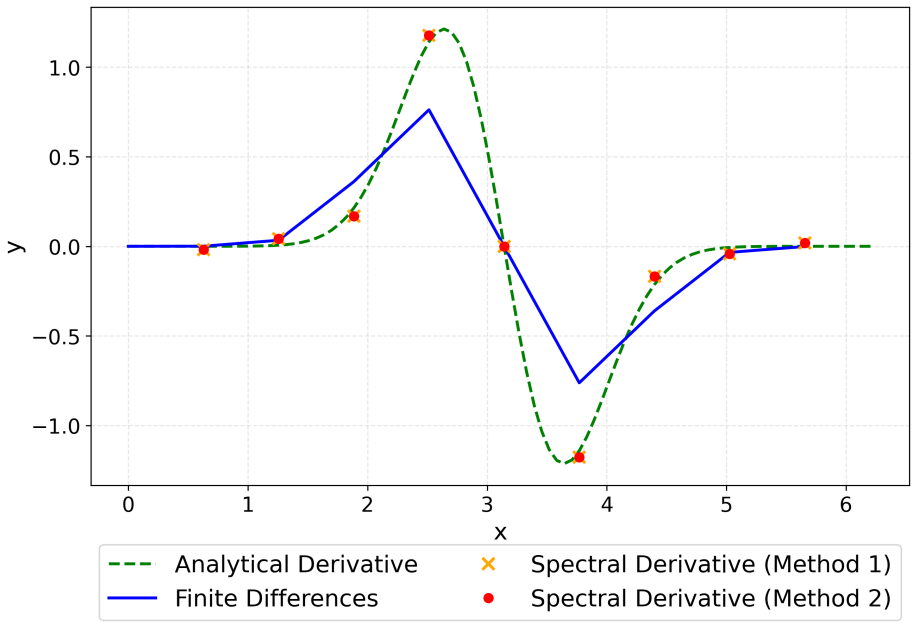

FIGURE 2.2: Comparison between analytical and numerical approximations of the derivative of a Gaussian. The numerical derivative is first computed with the finite differences method and then with the Fourier transform, using two analogous formulations

2.4 Fourier Transform Matrix Formulation

The Fourier transform can be seen as an orthogonal function space spanned by sines and cosines [4] and it is also a linear operator, meaning that

$$\mathcal{F}(\alpha f(x) + \beta g(x)) = \alpha \mathcal{F}(f) + \beta \mathcal{F}(g) \tag{2.23}$$

As a consequence of that it is possible to compute the discrete Fourier transform as a matrix multiplication. So if x is the vector of the sampled data, X the DFT of the signal and W the DFT

matrix.

$$X = Wx \tag{2.24}$$

This is because of the fact that the DFT of a function, at a certain wave number k, has the form

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi nk/N} \tag{2.25}$$

where N is the length of the sample column vector x_n. For the k^{th} wave number the expression can be written in matrix form as

$$X_{k_1} = \left[e^{-i2\pi 0k_1/N}, e^{-i2\pi 1k_1/N}, ..., e^{-i2\pi(N-1)k_1/N}\right] [x_0, x_2, ..., x_{N-1}]^T \tag{2.26}$$

So the complete expression, considering all the wave numbers, becomes

$$X = \begin{bmatrix} e^{-2\pi i(0)(0)/N} & e^{-2\pi i(0)(1)/N} & \dots & e^{-2\pi i(0)(N-1)/N} \\ e^{-2\pi i(1)(0)/N} & e^{-2\pi i(1)(1)/N} & \dots & e^{-2\pi i(1)(N-1)/N} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-2\pi i(N-1)(0)/N} & e^{-2\pi i(N-1)(1)/N} & \dots & e^{-2\pi i(N-1)(N-1)/N} \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

It is worth noting that the matrix is symmetrical. From now on this operation will be written as in equation 2.24. The inverse transformation on the other hand can be expressed as

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi nk/N} \tag{2.27}$$

The notation with the term $1/N$ in the inverse transform term is chosen because it is compliant with the SciPy module *scipy.fft* of Python. The matrix associated with the inverse transform will be addressed as $\overline{W}$. This is due to the fact that, not considering the $1/N$ factor, the matrices W and $\overline{W}$ are one the complex conjugate of the other, moreover since they are both symmetrical the complex conjugate and the hermitian operators coincide.

$$W = W^T \tag{2.28}$$

$$\overline{W} = \overline{W}^T = W^H \tag{2.29}$$

It follows that multiplying each term of 2.24 on the left by the hermitian W^H

$$W^H X = W^H W x \longrightarrow W^H X = x \tag{2.30}$$

Since $\overline{W}W = \mathbf{I}$, this is in accordance with the statement

$$\mathcal{F}^{-1}\left[\mathcal{F}(x)\right] = x \tag{2.31}$$

$$\overline{W}Wx = \mathbf{I}x = x \tag{2.32}$$

In the previous section it was mentioned that $\mathcal{F}(\partial_x u) = ik_x\mathcal{F}(u)$. This useful property can also be expressed in matrix form as

$$W\partial_x u = ik_x W u \tag{2.33}$$

In this case k is a vector that contains the frequency bin centers in cycles per unit of the sample spacing (with zero at the start). The multiplication between k_x and W is not a vector-matrix product but rather an element-wise product.

2.5 2D Fourier Transform Matrix Formulation

When working on a 2D domain the setup changes slightly. Provided that the domain has the same number of equally spaced points in the two directions, the Fourier transform is now

$$\mathcal{F}_2(x) = WxW^T = X \tag{2.34}$$

Where x is now a matrix. The DFT matrix is the same as the one of the 1D case and so it still benefits from its properties. Analogously the inverse function is expressed as

$$\mathcal{F}_2^{-1}(X) = \overline{W}X\overline{W}^T = x \tag{2.35}$$

The subscript "2" is to stress out that the FT is now bi-dimensional, but it will be dropped in the following sections. It is interesting to notice to the derivatives in a 2D domain, the element-wise product between k_x and k_y are multiplied to the right and to the left of x respectively

$$\partial_x u = \overline{W}Wx(ik_xW^T)\overline{W}^T = x(ik_xW^T)\overline{W}^T \tag{2.36}$$

$$\partial_y u = \overline{W}(ik_yW)xW^T\overline{W}^T = \overline{W}(ik_yW)x \tag{2.37}$$

Intuitively this is due to the fact that performing a 2D FT corresponds to separately perform a FT along the rows and then along the columns.

2.6 1D Jacobian for the viscous Burgers equation

It has been said that using spectral methods can lead to the solution of equations such as 2.14 or Burgers'. A question that might arise is whether it is possible to somehow obtain the Jacobian of such system in order to then feed it to the time integrator to speed it up. This strategy will first be applied to the 1D Burgers equation and then to a scalar version of the 2D Burgers equation, more on that in a following section. First of all let's see how the equation can be rewritten and split into the diffusive and convective terms. Then the Jacobian will be introduced and it will be shown how it is possible to compute it starting from the matrix formulation of the problem at hand. The initial equation reads

$$\partial_t u + u\partial_x u = \nu\partial_{xx} u \tag{2.38}$$

$$\partial_t u = \nu\partial_{xx} u - \frac{1}{2}\partial_x(u^2) \tag{2.39}$$

Applying the Fourier transform to both sides and using the derivative property it becomes

$$\mathcal{F}(\partial_t u) = -\nu k^2 \mathcal{F}(u) - \frac{1}{2} ik \mathcal{F}(u^2) \tag{2.40}$$

Now the inverse transform is applied and it yields

$$\mathcal{F}^{-1}\left[\mathcal{F}(\partial_t u)\right] = -\nu \mathcal{F}^{-1}\left[k^2 \mathcal{F}(u)\right] - \frac{1}{2}\mathcal{F}^{-1}\left[ik\mathcal{F}(u^2)\right] \tag{2.41}$$

The first term on the RHS is the diffusive term, it is linear with respect to the solution u. The second one is the convective term, it is non-linear. The equation can therefore be split in the sum of two terms

$$\partial_t u = f(u) = f_c(u) + f_d(u) \tag{2.42}$$

2.6.1 Introduction to the Jacobian

The Jacobian can in some way be thought of as the extension of the derivative for the one dimensional case. When $f : \mathbb{R} \to \mathbb{R}$

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x \tag{2.43}$$

This still holds when dealing with multivariate functions $g : \mathbb{R}^n \to \mathbb{R}^m$

$$g(x + \Delta x) \approx g(x) + g'(x)\Delta x \tag{2.44}$$

Now x and Δx are vectors of dimension n by 1, whereas on the right hand side $g(x)$ is m by 1 , this means that in order to have matching dimensions $g'(x)$ must be $\in \mathbb{R}^{m\times n}$. The Jacobian has, according to the adopted notation, the following structure

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} \cdots \frac{\partial f_1}{\partial x_n} \\ \vdots \ddots \vdots \\ \frac{\partial f_m}{\partial x_1} \cdots \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{2.45}$$

2.6.2 Jacobian of a matrix-vector product

Now let's consider a matrix-vector multiplication $X = Wx$, where W does not depend on x. X has dimensions $m \times 1$, x has dimensions $n \times 1$ and W $m \times n$. It can be seen that

$$\frac{\partial X}{\partial x} \equiv J_X = \frac{\partial (Wx)}{\partial x} = W \tag{2.46}$$

this is because the ith element of X is given by

$$X_i = \sum_{k=1}^{n} w_{ik} x_k \,, \quad i = 1, \ldots, n \tag{2.47}$$

where w_{ij} is an element of matrix W and n is the number of elements of x. It follows that

$$\frac{\partial X_i}{\partial x_j} = w_{ij}\,, \quad i = 1, \dots, n \quad j = 1, \dots, m \tag{2.48}$$

So the Jacobian of a matrix-vector multiplication is the matrix itself. This extends also in the case of higher powers of x, so for instance in the case of $Y = Wx^2$ the Jacobian becomes $J_Y = 2Wx$.

2.6.3 Burgers equation written with matrices

A question that might arise is whether those matrix properties can be useful in finding the Jacobian equations such as 2.41. Following what was done in section 2.4, this expression can be written in matrix form as follows

$$\partial_t u = -\nu \overline{W} ik^2 W u - \frac{1}{2} \overline{W} ikW u^2 \tag{2.49}$$

Both terms $\overline{W}ik^2W$ and $\overline{W}ikW$ are matrices, that do not depend on u so it is indeed possible to compute the Jacobian in this case. This expression can once again be split in the convective and in the diffusive term as in 2.42 and since derivation is a linear operator

$$J_f = \frac{\partial f(u)}{\partial u} = \frac{\partial f_c(u)}{\partial u} + \frac{\partial f_d(u)}{\partial u} \tag{2.50}$$

So the Jacobian of the whole expression is the sum of the Jacobians of the convective and diffusive terms. Analogously to 2.46, the Jacobian of the diffusive term is $-\nu\overline{W}ik^2W$, whereas for the convective, non-linear term it is $-\overline{W}ikWu$. It is worth noting that the Jacobian of the diffusive term is constant, since it does not depend on the solution. This means that it can be computed only once at the start of the integration process. Therefore by applying the direct and inverse Fourier transform to a PDE and then writing the expression in terms of matrices, the Jacobian of the 1D viscous Burgers equation was obtained. As mentioned before, this Jacobian can be used in the time integration process to speed it up. In the following section this aspect will further be looked into.

2.6.4 Jacobian computing time

When the Jacobian is not provided, it is approximated by python functions, such as *odeint*, by means of finite differences. In order to obtain the approximate Jacobian the function is perturbed in each direction. The Jacobian can be approximated with forward difference as

$$J_{ij} = \frac{f_i(x_j + \Delta x_j) - f_i(x_j)}{\Delta x_j} \tag{2.51}$$

The computational cost does not scale well with large domains [21], especially when the function at hand is hard to evaluate. On the flip side the analytical Jacobian does not need to evaluate the function but rather it consists of a matrix-vector multiplication for each time step as the Jacobian of the diffusive term is constant. In figure 2.3 it is possible to see that as the grid becomes finer the time saved grows, while the norm 2 of the difference between the Jacobian computed with finite differences and the analytical one has acceptable values.

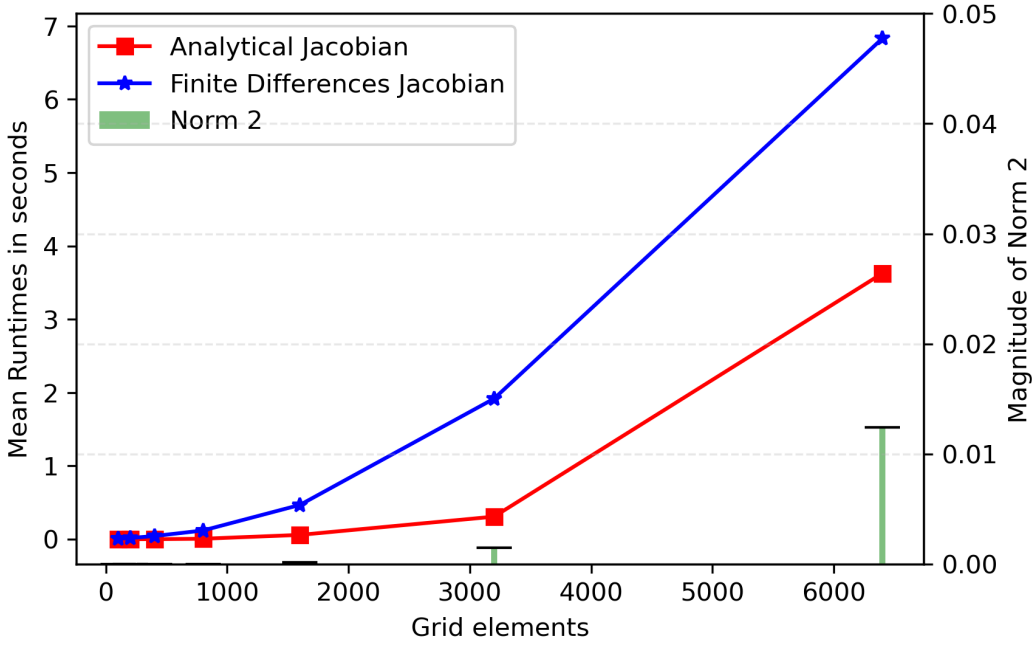

FIGURE 2.3: Finite differences and analytical Jacobian computation time comparison for the 1D Burgers equation. The norm is computed on the difference between the FD and analytical Jacobian

At this point one may wonder how the time saved in the Jacobian calculation translates to a faster time integration. Figure 2.4 shows how the Jacobian did indeed speed up the integrating factor, but by a smaller margin compared to the Jacobian calculation time alone.
By observing the Jacobian matrix it was noticed that the predominant term was the one related to diffusion. This remark led to the idea of approximating the whole Jacobian with it, $J_f \approx \partial f_d(u)/\partial u$. Since this term only had to be computed once the Jacobian would remain constant. However time spared in this way will be lost during the process of solving the linear system since the odeint function is now being fed an approximated function. In the end it was found that for this specific case the integration process was indeed sped up. Nonetheless this approach is very much empirical and not so easy to generalize. One other idea in this direction could be to update the Jacobian with the convective term every n steps, in the end this was not further investigated due to its low generalizability.

2.7 2D Jacobian for the viscous Burgers equation

Since calculating the Jacobian had a beneficial effect on the 1D Burgers equation it is natural to ask what happens when dealing with a 2D domain. After all the long term goal would be to find the Jacobian of the BLEW equations. The main difference compared to the 1D case is that now u is a matrix and not a vector anymore, since the solution is computed at each grid point. This means that the derivative property of the matrix-vector product in 2.46 no longer applies. Moreover, depending on the axis with respect to which the equations are being derived, the order of the terms changes. Equations 2.36 and 2.37 can respectively be seen as BxA and AxB, with B being the identity matrix. To investigate this idea a simplified version of the Burgers

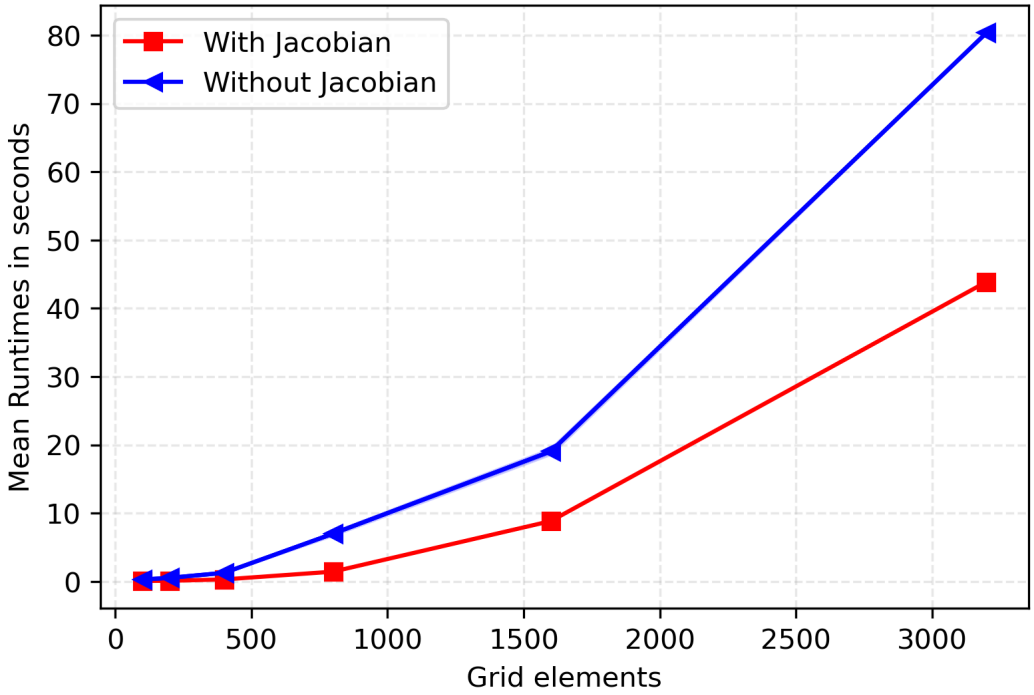

FIGURE 2.4: Integration time with and without analytical Jacobian for the 1D Burgers equation

equation was chosen

$$\partial_t u = -u\partial_x u - a\partial_y u + \nu(\partial_{xx} u + \partial_{yy} u) \tag{2.52}$$

Nonetheless the equation contains both diffusive and convective terms and derivatives of them in both directions so extending the Jacobian calculation to the full 2D Burger equation should not be too tedious.

2.7.1 Linear Terms

As in the 1D case the diffusive terms are linear with respect to the solution u. The term $\partial_{yy} u$ can be written as $-\overline{W}(k_y^2 W)u$ which corresponds to AuB, where B is the identity matrix. Knowing that the derivative of a constant matrix $dC = 0$ and that the chain rules applies

$$dXY = (dX)Y + XdY \tag{2.53}$$

it follows that

$$d(AuB) = A(du)B \tag{2.54}$$

As a matter of fact A and B are constant matrices. One other ingredient is the vectorization operation for which holds [18]

$$vec(AuB) = (B^T \otimes A)vec(u) \tag{2.55}$$

Therefore

$$vec(AduB) = (B^T \otimes A)vec(du) \tag{2.56}$$

So that

$$\frac{d(AuB)}{du} = B^T \otimes A \tag{2.57}$$

Using relation 2.57 the Jacobian for the diffusive term can be expressed as

$$J_{f_{d,y}} = I \otimes -\overline{W}(k_y^2 W) \tag{2.58}$$

For the term $\partial_{xx}u$ the process is completely analogous and it can be written as $f_{d,x} = -u(W^T k_x^2)\overline{W}^T = AuB$, where now A is the identity. The Jacobian is thus $J_{f_{d,x}} = \left((W^T k_x^2)\overline{W}^T\right)^T \otimes I$. The last linear term remaining is $a\partial_y u$ which has a Jacobian analogous to expression 2.58 but with the first power of k_y and positive sign.

2.7.2 Non-linear Terms

This term is written in matrix form as $u^2(W^T ik_x)\overline{W}^T$. If it could be rewritten as in 2.57, the Jacobian could be computed . However now the elements of u are squared and so

$$\frac{d(Au^2B)}{du} \neq \frac{d(AXB)}{dX} \tag{2.59}$$

The problem is that the derivative is still with respect to the solution and not to the squares of its values. That is where the chain rule comes in handy once again since

$$\frac{d(Au^2B)}{du} = \frac{d(Au^2B)}{du^2}\frac{du^2}{du} \tag{2.60}$$

Which is really convenient as the first term can now be expressed as $B^T \otimes A$, while the second one is the derivative of a matrix with element-wise squared elements with respect to the non squared element matrix. Therefore

$$\frac{du^2}{du} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} & \cdots & \frac{\partial f_1}{\partial x_{nm}} \\ \vdots & \ddots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} & \cdots & \frac{\partial f_n}{\partial x_{nm}} \\ \vdots & & \vdots & \ddots & \vdots \\ \frac{\partial f_{nm}}{\partial x_1} & \cdots & \frac{\partial f_{nm}}{\partial x_n} & \cdots & \frac{\partial f_{nm}}{\partial x_{nm}} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_1^2}{\partial u_1} & \cdots & \frac{\partial u_1^2}{\partial u_n} & \cdots & \frac{\partial u_1^2}{\partial u_{nm}} \\ \vdots & \ddots & \vdots & & \vdots \\ \frac{\partial u_n^2}{\partial u_1} & \cdots & \frac{\partial u_n^2}{\partial u_n} & \cdots & \frac{\partial u_n^2}{\partial u_{nm}} \\ \vdots & & \vdots & \ddots & \vdots \\ \frac{\partial u_{nm}^2}{\partial u_1} & \cdots & \frac{\partial u_{nm}^2}{\partial u_n} & \cdots & \frac{\partial u_{nm}^2}{\partial u_{nm}} \end{bmatrix} \tag{2.61}$$

Which simplifies to a diagonal matrix with elements

$$\frac{du^2}{du} = \begin{bmatrix} 2u_1 & & 0 \\ & \ddots & \\ 0 & & 2u_{nm} \end{bmatrix} \tag{2.62}$$

The Jacobian of the convective term is computed by multiplying these two terms. In the final code some adjustments had to be made, the signs of the matrix had to be inverted apart from the diagonal terms in order to get the same matrix as the one found with finite differences. Since two sparse matrices have to be multiplied together it is important to treat them as such. This made the difference between taking much more time than odeint without Jacobian, to being faster. Once this iterative process is finished the final Jacobian of equation 2.52 can be assembled by summing all the three linear terms and this last non-linear one. The final result was compared to the finite differences one, the norm of the difference of the two final Jacobians was computed and deemed acceptable.

2.7.3 2D Jacobian run times

Before analyzing the results it is important to have a feeling for the dimension of the matrices that are under consideration. When dealing with a numerical 2D domain of 120 by 120, for instance, the Jacobian has dimensions 120^2 by 120^2, which means dealing with 120^4 elements, that is, more than 200 millions elements. The Euclidean norm between the two Jacobians in this case is around 10^{-2}, as shown in figure 2.5. It is therefore necessary to used sparse matrices whenever possible and to try and make every step free from unnecessary operation, while also categorically avoiding for loops. In the end, the results compare the matrix computed with the finite differences with the analytical one. As happened for the 1D study case the diffusive term has a constant derivative with respect to the solution and so the corresponding Jacobian can be computed only once.

As for the monodimensional case this faster computed Jacobian does in fact translate to a time saving during the integration process. Once again the time saved shrinks compared to the time required to compute exclusively the Jacobian. Nonetheless the improvement is noticeable.

The integration took place calling the following commands

LISTING 2.2: Python code for computing the derivative of a periodic function

```
t_span = (t_i, t_f)
t2 = time.time()
solution2 = solve_ivp(burger_ivp, t_span, u0.ravel(), method='LSODA
    ', jac=jaco_ivp)
t2_end = time.time()
elapsed2 = t2_end - t2
list_run_times_nuove.append(elapsed2)
```

During the coding of the script some arrangements have been made. For example the odeint function only takes one dimensional initial conditions, but since the u scalar field is 2D it was necessary to flatten it at each time step, with the command *ravel()*. Whenever *odeint* is used it is necessary to provide a function that evaluates the RHS of the function, in this case it was called *burger_ivp*. For the reason given before the function took a flattened vector input that has to be reshaped into the original matrix form. All in all it would be interesting to further work on the

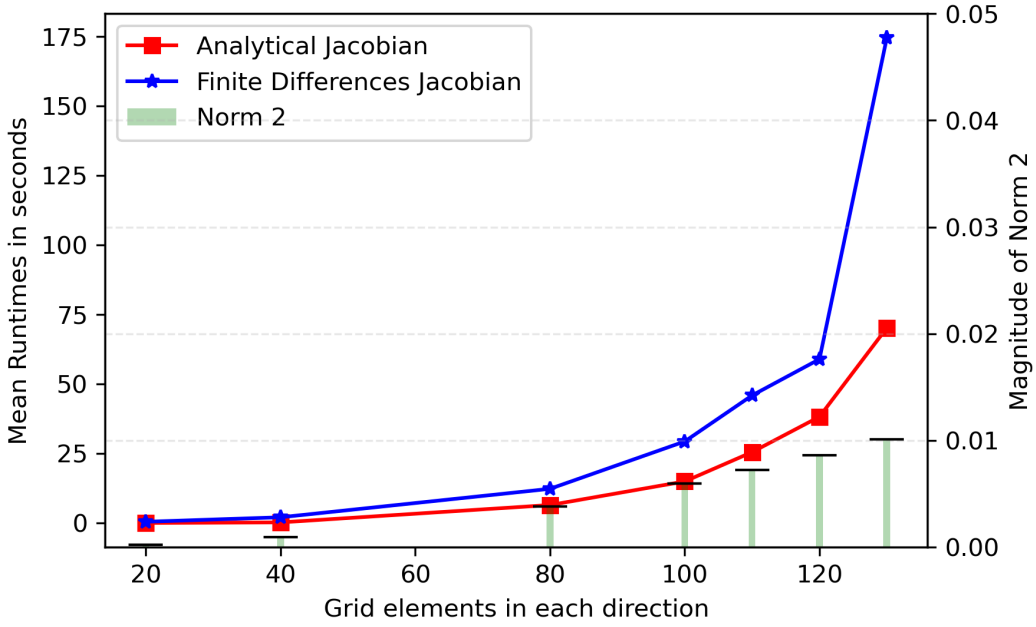

FIGURE 2.5: Finite differences and analytical Jacobian computation time comparison for the 2D Burgers equation. The norm is computed on the difference between the FD and analytical Jacobian

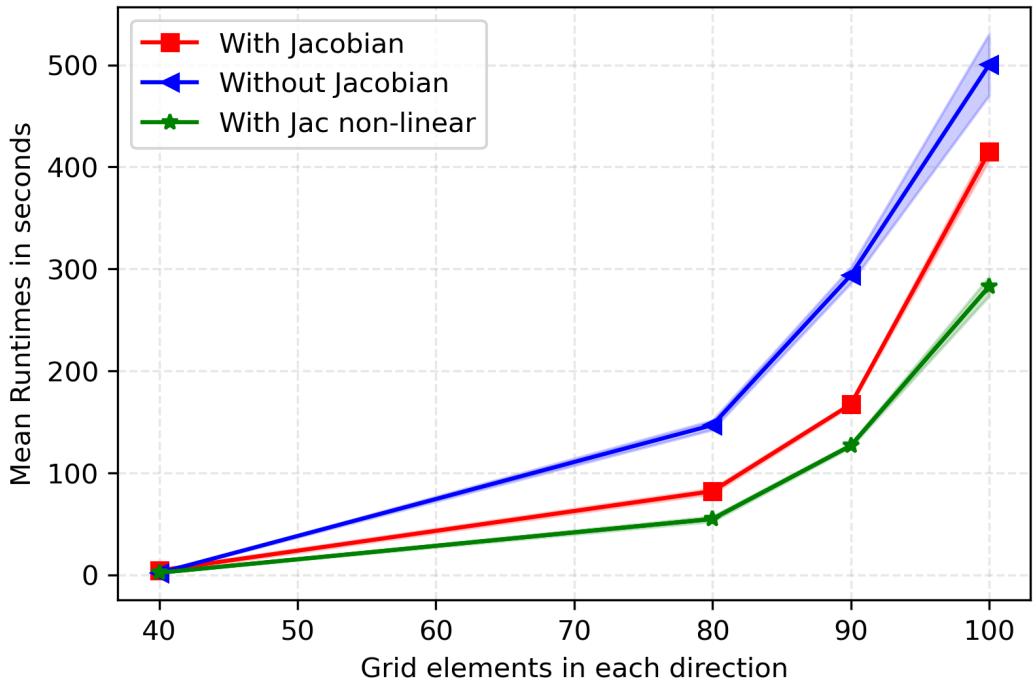

FIGURE 2.6: Integration time with and without analytical Jacobian for the 1D Burgers equation. The green line represents the case where only the convective term is computed at each time step

subject to further generalize the method and to run some more test cases to see whether it is possible to make it more general. One problem that might arise when trying this approach on BLEW is related to the perfectly matched layer. As a matter of fact this layer that takes care of damping the perturbations introduces new equations that have to be solved. This means that it would be also necessary to calculate the Jacobian for those equations.

2.8 Integrating Factor

In this section it will be shown how it is possible to leverage spectral methods to build a fast integrator that in a way does not solve the linear terms. The process follows the "Program 27" in [25]. The process starts by considering 1D Burgers equation in form 2.39, defined in a generic domain, $\tilde{x}$, centered with respect to the origin. The domain is then re-scaled between $[0, 2\pi]$, with $x = \frac{2\pi}{L}\tilde{x}$. It follows that

$$\frac{\partial x}{\partial \tilde{x}} = \frac{2\pi}{L} \tag{2.63}$$

So

$$\partial_{\tilde{x}} u = \frac{\partial u}{\partial \tilde{x}} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial \tilde{x}} = \frac{2\pi}{L}\frac{\partial u}{\partial x} \tag{2.64}$$

The process is analogous for the second order derivative

$$\frac{\partial^2 u}{\partial \tilde{x}^2} = \frac{\partial}{\partial \tilde{x}}\left(\frac{\partial u}{\partial \tilde{x}}\right) = \tag{2.65}$$

$$\frac{\partial}{\partial \tilde{x}}\left(\frac{2\pi}{L}\frac{\partial u}{\partial x}\right) = \frac{2\pi}{L}\frac{\partial}{\partial \tilde{x}}\left(\frac{\partial u}{\partial x}\right) \tag{2.66}$$

Now inverting the order of the derivatives

$$\frac{\partial^2 u}{\partial \tilde{x}^2} = \frac{2\pi}{L}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial \tilde{x}}\right) = \frac{4\pi^2}{L^2}\frac{\partial^2 u}{\partial x^2} \tag{2.67}$$

The scaled Burgers equation then becomes

$$\partial_t u + \frac{2\pi}{2L}\partial_x u^2 - \frac{4\pi^2}{L^2}\nu\partial_{xx} u = 0 \tag{2.68}$$

Taking the Fourier transform of both sides yields

$$\frac{d}{dt}(\hat{u}) + \frac{\pi i k}{L}\widehat{u^2} + \frac{4\pi^2 k^2 \nu}{L^2}\hat{u} = 0 \tag{2.69}$$

Where the sign of the diffusive term has changed as a consequence of taking the square of the imaginary unit. Now the goal is finding a factor, K, that multiplied to the equation leads to a form

$$\frac{d}{dt}\left[\hat{u}(t)K\right] = K\frac{d}{dt}(\hat{u}) + K\frac{4\pi^2 k^2 \nu}{L^2}\hat{u} \tag{2.70}$$

So that the whole expression could be written as

$$\frac{d}{dt}\left[\hat{u}(t)K\right] + \frac{\pi i k}{L} K\widehat{u^2} = 0 \tag{2.71}$$

It tuns out that said *integrating factor*, K, is $e^{\frac{4\pi^2 k^2 \nu}{L^2}t}$. Thus it is possible to rewrite 2.69 as

$$\frac{d}{dt}\left[\hat{u}(t)e^{\frac{4\pi^2 k^2 \nu}{L^2}t}\right] + \frac{\pi i k}{L} e^{\frac{4\pi^2 k^2 \nu}{L^2}t}\widehat{u^2} = 0 \tag{2.72}$$

To verify one can take the derivative of the function product and divide by the factor. At this point it is convenient to change variables

$$w = \hat{u}(t)e^{\frac{4\pi^2 k^2 \nu}{L^2}t} \tag{2.73}$$

and so

$$\hat{u} = we^{-\frac{4\pi^2 k^2 \nu}{L^2}t} \tag{2.74}$$

$$\widehat{u^2} = \mathcal{F}\left(\left[\mathcal{F}^{-1}\left(we^{-\frac{4\pi^2 k^2 \nu}{L^2}t}\right)\right]^2\right) \equiv f^*\left(we^{-\frac{4\pi^2 k^2 \nu}{L^2}t}\right) \tag{2.75}$$

Calling $A = \frac{4\pi^2 k^2 \nu}{L^2}$ yields

$$\frac{dw}{dt} = \frac{-\pi i k}{L} e^{At} f^*\left(we^{-At}\right) \tag{2.76}$$

This is now an ODE that could be solved with the Euler explicit method for instance

$$w_k^{n+1} = w_k^n + \Delta t \frac{\pi i k}{L} e^{At^n} f^*\left(w_k e^{-At^n}\right) \tag{2.77}$$

However the solution of interest is $\hat{u}$ and not w and so using the relation 2.74 it becomes

$$\hat{u}_k^{n+1} e^{At^{n+1}} = \hat{u}_k^n e^{At^n} + \Delta t \frac{\pi i k}{L} e^{At^n} f^*\left(\hat{u}_k e^{-At^n} e^{At^n}\right) \tag{2.78}$$

So in the end

$$\hat{u}_k^{n+1} = \hat{u}_k^n e^{A\Delta t} + \Delta t \frac{\pi i k}{L} e^{A\Delta t} f^*\left(\hat{u}_k\right) \tag{2.79}$$

The effectiveness of this approach resides in the fact that in order to evaluate $\partial_t u$ it is necessary to only compute $f^*(u)$. This equates to an inverse FT, squaring the result and then performing a direct FT. Usually this had to be done for both the convective and diffusive term. In this case the diffusive term is somehow embedded in the time derivative term. This approach appears to be hard to extend to more complex functions. This is because the core idea is to multiply each term with a factor that will then allow to write the time derivative as a product between two functions. However when the number of terms increases it becomes difficult to group them together.

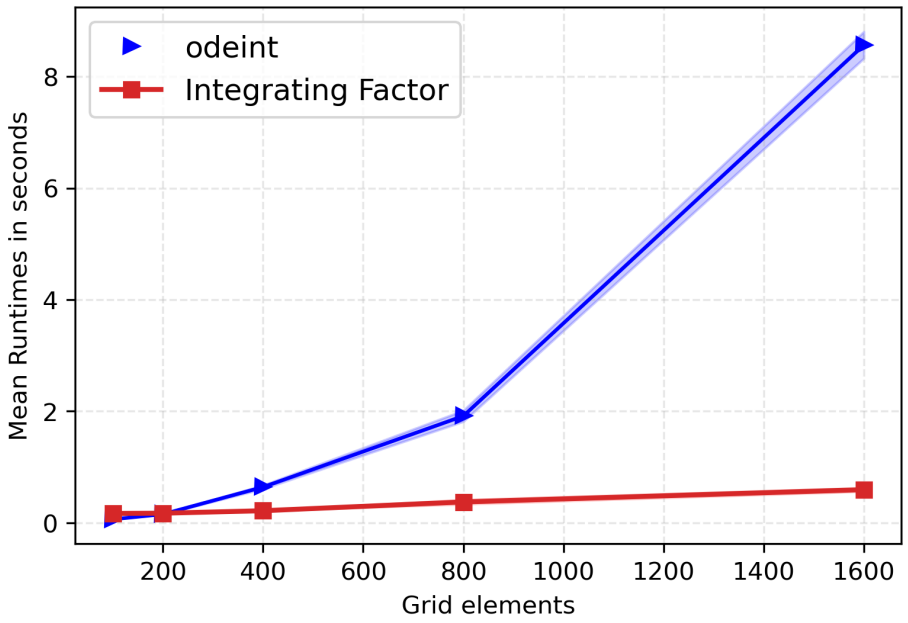

FIGURE 2.7: Time integration comparison with the same number of steps between odeint and the integrating factor method

Chapter 3

BLEW Implementation

As seen in chapter 2 spectral methods are powerful tools, however it is necessary to take some precautions before implementing it to the integral model introduced in chapter 1. The main issue is related to the periodicity requirements, that will be dealt with using a Perfectly Matched Layer (PML). After this is done, everything will be ready to be coded up. Nonetheless it will still be necessary to file some corners in order to make the code faster. In the second half of the chapter it will be discussed how this was achieved by optimizing how the distributions caused by jets and magnets are computed.

3.1 Perfectly Matched Layer

Many physical phenomena involve waves propagating over an indefinitely extended domain. This poses an important problem to numerical simulations. Once the discretization method has been addressed, it is a matter of defining the boundary conditions. Neumann and Dirichlet conditions are usually the first options considered. These come at the cost of spurious phenomena, though. The outgoing waves impinging on the boundary are reflected into the domain, lowering the quality of the numerical solution. A simple workaround would be periodic boundary conditions. However, this assumes period actuators over the coating line, which will not be the case in industrial applications, where the actuators will act in a small portion of the domain near the wiping jets. So the question is how to simulate the liquid film on an open domain without reflections. A method, previously applied to the 3D BLEW with finite volumes, is an absorbing layer at the boundary. This way, the system is fooled into thinking it extends indefinitely over the boundary. Although very efficient, this method does not suit the spectral BLEW 3D environment with periodic boundary conditions. Instead of an absorbing boundary, we can think of an absorbing layer, damping the waves before these attain the periodic boundary. This is the concept of the perfectly matched layer (Johnson [13]).

With this approach boundary conditions do not influence the layer, which, in a way, is an extension of the domain where the waves are damped. However a new problem arises, that is when the wave crosses from one material to another, it reflects.
On a broad level the process of creating the PML can be broken down in two steps: first the domain is analytically continued in complex coordinates, then it is brought back to conventional real coordinates.

3.1.1 Complex coordinate stretching

To begin with the domain is extended in the complex plane. For example let us consider a wave in the form $w(x) = We^{ikz-\omega t}$. It can be shown that, if the following couple of assumptions are

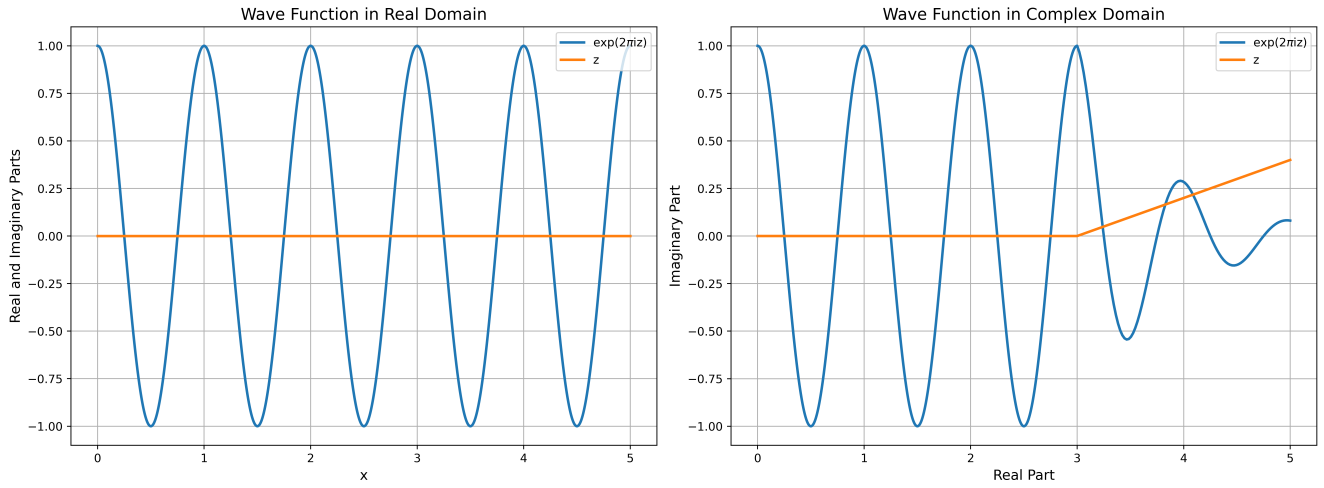

FIGURE 3.1: Without changing the wave function, but just the contour z, the solution in the right hand side is damped and goes to zero

met, then a generic wave can be decomposed in a sum of waves that have that expression, that is, they are plane ([13]). The mentioned prerequisites are

- The region of the PML is homogeneous
- The region of the PML is does not depend on time and is linear

To see how such a wave can be damped by analytically extending the domain the wave e^{ikz} is considered. If the contour, z, has real values then the wave assumes the shape of a periodic oscillating function, according to the Euler formulation for complex numbers. However if $z = x + iy$, then $e^{ikz} = e^{ikx}e^{-ky}$ and this is the product of the original wave times a damped wave. It is important to notice that if z it real and only after a certain value becomes complex, like in figure 3.1, then in the real section the solution will not change.

3.1.2 Bringing the axis back to real

So the new contour was introduced, $z = x + if(x)$, it follows that $\partial z = \left(1 + i\frac{df}{dx}\right)\partial x$. For a matter of convenience, that allows to have a decay rate that is not influenced by the frequency, the latter term is generally written as

$$\frac{df}{dx} = \frac{\sigma_x(x)}{\omega} \tag{3.1}$$

In practice the substitution to be made is

$$\frac{\partial}{\partial x} \rightarrow \frac{1}{1 + i\frac{\sigma_x(x)}{\omega}}\frac{\partial}{\partial x} \tag{3.2}$$

Once the oscillations are damped, it is possible to set conventional boundary conditions on the domain. For instance in the BLEW case the condition will be that there the height of the liquid film will be that of the unperturbed flow. So far the PML has been shown to be a general and flexible strategy to deal with oscillations, however it does come with a couple of problems. The

first of which is the fact that when a solution is not solved analytically but on a discrete domain it is no longer true that there are no reflections at all. However the magnitude of the disturbs can be reduced in two main ways: either by making the domain larger or by having a finer grid. Moreover it is important not to have an extremely steep function σ, usually a quadratic or cubic form is chosen.

3.1.3 1D Burgers equation with PML

The example of figure 3.1 shows how a linear wave is damped by a certain domain that is continued in the analytical plane. However one of the most common issues of the PML is dealing with non linear equations, and as chance has it the system of equations shown in the first chapter does not fall in that category. In this section the PML for the non linear 1D Burgers equation will be put to a test. The final implementation on the BLEW solver will follow the same method. The equation 2.38 can be rewritten by introducing the substitution 3.2, yielding

$$\partial_t u = -u\left(1-\frac{i\sigma}{\omega}\right)^{-1}\partial_x u + \nu\left(1-\frac{i\sigma}{\omega}\right)^{-1}\partial_x\left[\left(1-\frac{i\sigma}{\omega}\right)^{-1}\partial_x u\right] \tag{3.3}$$

Auxiliary variables are now introduced

$$\alpha_1 = \left(1-\frac{i\sigma}{\omega}\right)^{-1}\partial_x u \qquad \alpha_2 = \left(1-\frac{i\sigma}{\omega}\right)^{-1}\partial_x \alpha_1 \tag{3.4}$$

Which allows to rewrite the Burgers' equation as

$$\partial_t u = -u\alpha_1 + \nu\alpha_2 \tag{3.5}$$

Moreover the two equations 3.4 imply that

$$\partial_t\left(\alpha_1 - \partial_x u\right) = -\sigma_x\alpha_1 \qquad \partial_t\left(\alpha_2 - \partial_x\alpha_1\right) = -\sigma_x\alpha_2 \tag{3.6}$$

This equation can be verified considering that equation 3.4 can be rewritten as

$$\alpha_1 - \partial_x u = -\frac{i\sigma_x}{\omega}\alpha_1 \tag{3.7}$$

Now both sides are derived with respect to time

$$\frac{\partial}{\partial t}\left(\alpha_1 - \partial_x u\right) = -\frac{\partial}{\partial t}\left(\frac{i\sigma_x}{\omega}\alpha_1\right) = -\mathcal{F}\mathcal{F}^{-1}\left[\frac{\partial}{\partial t}\left(\frac{i\sigma_x}{\omega}\alpha_1\right)\right] \tag{3.8}$$

At this point the following property can be used

$$\mathcal{F}^{-1}\left[\frac{\partial}{\partial t}\left(\frac{i}{\omega}f(t)\right)\right] = \mathcal{F}^{-1}(f(t)) \tag{3.9}$$

This stems from the definition of the FT

$$\mathcal{F}^{-1}\left[\frac{\partial}{\partial t}\left(\frac{i}{\omega}f(t)\right)\right] = \int_{-\infty}^{+\infty}\frac{\partial}{\partial t}\frac{i}{\omega}f(t)e^{i\omega t}dt \tag{3.10}$$

which can be further be integrated by parts

$$\frac{i}{\omega}f(t)e^{i\omega t}\Big|_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty}\frac{i}{\omega}f(t)i\omega e^{i\omega t}dt = \int_{-\infty}^{+\infty}f(t)e^{i\omega t} = \mathcal{F}^{-1}(f(t)) \tag{3.11}$$

Now relations 3.6 have been justified. So taking a step back, there is a system with equations 3.5 and the two 3.6. A new set of auxiliary variables is introduced

$$A_1 = \alpha_1 - \partial_x u \qquad A_2 = \alpha_2 - \partial_x \alpha_1 \tag{3.12}$$

So now the two equations 3.6 can be written as

$$\partial_t A_1 = -\sigma_x \alpha_1 \qquad \partial_t A_2 = -\sigma_x \alpha_2 \tag{3.13}$$

Hence the final system reads

$$\begin{cases} \partial_t u = -u\alpha_1 + v\alpha_2 \\ \partial_t A_1 = -\sigma_x \alpha_1 \\ \partial_t A_2 = -\sigma_x \alpha_2 \end{cases} \tag{3.14}$$

A python function was written to return the RHS of the system at each time step. As inputs it took the three variables and time. Moreover at each time step also α_1 and α_2 have to be updated and it is done according to equation 3.12. The necessary derivatives are computed using the Fourier transform. It was chosen to have a σ_x that has a value of zero in the physical domain and then it grows quadratically in the PML region.

Figure 3.2 compares the results with and without layer. The layer successfully damps the waves while maintaining the solution unaltered where it is not present.

3.1.4 BLEW solver implementation

The process of the previous example is illustrative also of what happens in BLEW, where the final set of equations for the 3D jet wiping case can be written as

$$\partial_{\hat{t}}\hat{h} + \partial_{\hat{x}}\hat{q}_{\hat{x}} + \partial_{\hat{z}}\hat{q}_{\hat{z}} = 0 \tag{3.15}$$

$$\partial_{\hat{t}}\hat{q}_{\hat{x}} + \partial_{\hat{x}}F_{12} + \partial_{\hat{z}}F_{22} = \delta^{-1}\left[\hat{h}\left(-\partial_{\hat{x}}\hat{p}_g + \partial_{\hat{x}\hat{x}\hat{x}}\hat{h} + \partial_{\hat{x}\hat{z}\hat{z}}\hat{h} + 1\right) + \Delta\tau_{\hat{x}}\right] \tag{3.16}$$

$$\partial_{\hat{t}}\hat{q}_{\hat{z}} + \partial_{\hat{x}}F_{13} + \partial_{\hat{z}}F_{23} = \delta^{-1}\left[\hat{h}\left(-\partial_{\hat{z}}\hat{p}_g + \partial_{\hat{z}\hat{x}\hat{x}}\hat{h} + \partial_{\hat{z}\hat{z}\hat{z}}\hat{h}\right) + \Delta\tau_{\hat{z}}\right] \tag{3.17}$$

The change of variables introduced in equation 3.2 is carried out

$$\partial_{\hat{x}} \to \left(1 + \frac{i\sigma_x}{\omega}\right)^{-1}\partial_{\hat{x}} \quad \partial_{\hat{z}} \to \left(1 + \frac{i\sigma_z}{\omega}\right)^{-1}\partial_{\hat{z}} \tag{3.18}$$

For simplicity it will be shown the complete process for the first equation only, but it is analogous for the other two. Performing the substitution

$$\partial_{\hat{t}}\hat{h} + \left(1 + \frac{i\sigma_x}{\omega}\right)^{-1}\partial_{\hat{x}}\hat{q}_{\hat{x}} + \left(1 + \frac{i\sigma_z}{\omega}\right)^{-1}\partial_{\hat{z}}\hat{q}_{\hat{z}} = 0 \tag{3.19}$$

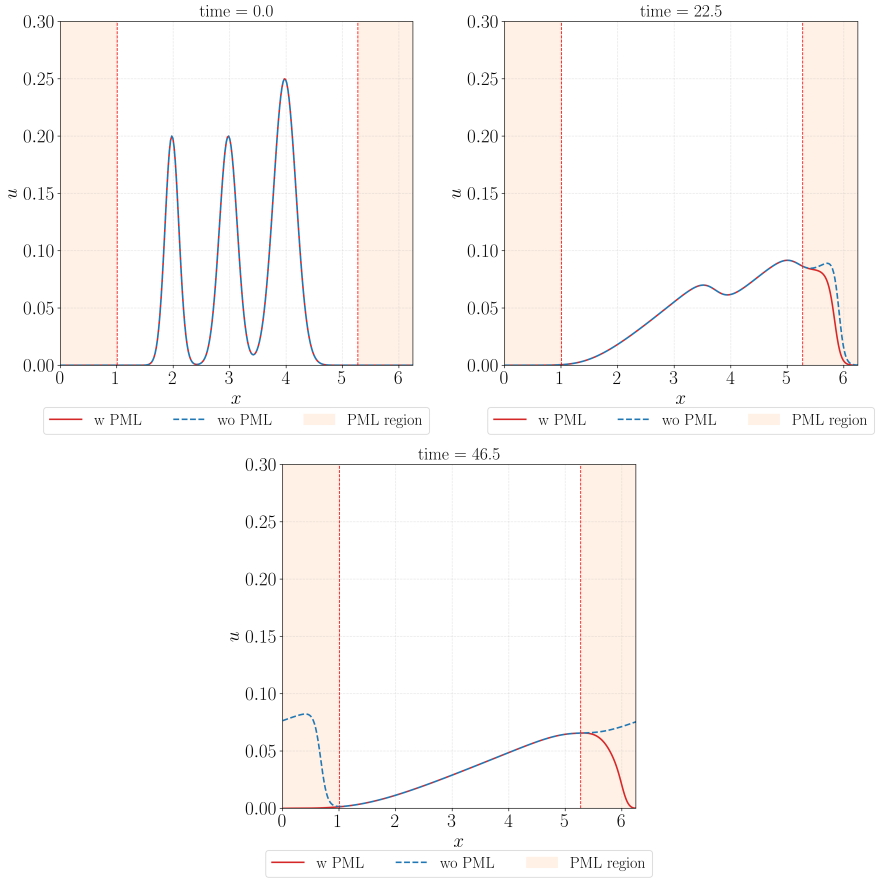

FIGURE 3.2: The figures show how the solution evolves with time. At the beginning there is no difference, but when the wave reaches the PML area the wave that interacts with the PML is flattened. It is worth noting how the solution remains unchanged in the physical domain

Now introducing another set of variables

$$\zeta_1 = \left(1 + \frac{i\sigma_x}{\omega}\right)^{-1} \partial_{\hat{x}} \hat{q}_{\hat{x}} \tag{3.20}$$

$$\zeta_2 = \left(1 + \frac{i\sigma_z}{\omega}\right)^{-1} \partial_{\hat{z}} \hat{q}_{\hat{z}} \tag{3.21}$$

The final result for the continuity equation then becomes

$$\partial_{\hat{t}} \hat{h} + \zeta_1 + \zeta_2 = 0 \tag{3.22}$$

Finally the different ζ are related to the other variables like

$$\partial_{\hat{t}} (\zeta_1 - \partial_{\hat{x}} \hat{q}_{\hat{x}}) = -\sigma_{\hat{x}} \zeta_1 \tag{3.23}$$

The process is repeated for all the additional variables and so the perfectly matched layer is added to the BLEW solver.

3.2 Jets pressure and shear stress distribution

In section 1.4 the pressure and shear stress expressions have been introduced. Considering equation 1.66 it is possible to see how the term can be decomposed in a geometric term, that does not change unless the jets are moved, and a term that depends on the jets "throttle", τ_{max}. This equation is not scalar, unlike the pressure one, meaning that the wall shear stress has a component along x and one along z. To compute them is therefore necessary to take the sine and cosine, increasing the computational cost.

For clarity sake let us go over the process of computing the stress field in the case of a single jet:

- The jet position is chosen
- The distance of each grid point from the jet is computed. Since $\lambda = r/k$, where k represents a constant and r the distance, this univocally determines the geometrical term

$$g_\tau = 0.18\frac{1 - e^{-144\lambda^2}}{\lambda} - 9.43\lambda e^{-144\lambda^2} \tag{3.24}$$

- τ_{max} is computed, depending on how strongly the jet operates
- The two terms are multiplied

$$\boldsymbol{\tau} = \tau_{max}\mathbf{g}_\tau \tag{3.25}$$

- The components along x and z are obtained

$$\tau_x = \tau_{max}g_\tau cos(\theta) \quad \tau_z = \tau_{max}g_\tau sin(\theta) \tag{3.26}$$

  When dealing with matrices those multiplications are element-wise operations

If more jets were to be added these steps would have to be repeated for each of them. The distribution for each jet would be computed separately and then they would all be summed together. To speed up the process the following measures are taken

- For each jet the distance and angle matrices are saved in a class object. So now, if n_j is the jet number, we will have n_j distance and n_j angle matrices
- With the angle matrix the cosine and sine matrices are computed, once in for all
- With the distance matrix the geometrical term is computed, which is again constant as long as the position of the jet does not change. We will have such a matrix for each jet.
- The geometrical term is multiplied element by element with the sine and cosine matrix

$$f_{\tau,x} = g_\tau cos(\theta) \quad f_{\tau,z} = g_\tau sin(\theta) \tag{3.27}$$

  This value is saved for each jet as an object of the class

- τ_{max} for each jet is computed and saved in a vector

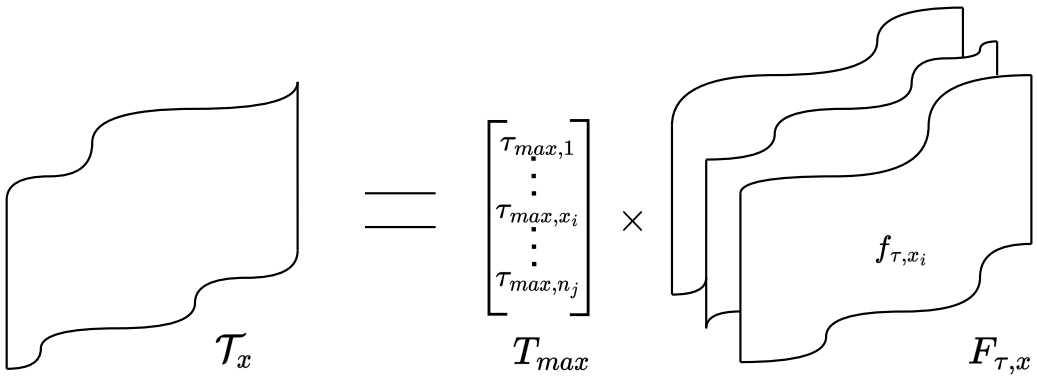

FIGURE 3.3: Final tensor-vector operation to compute the shear stress distribution along x

With this set up the only operation to be carried out at each time step is

$$\tau_{x_i} = (\tau_{max})_i f_{\tau,x_i} \quad \text{with} \quad i = 1, \cdots, n_j \tag{3.28}$$

Considering all jets at the same time $F_{\tau,x}$ is a tensor of dimension $n_j \times n \times m$ where n and m are the number of elements along x and z respectively, while T_{max} is $n_j \times 1$. The capital letters indicate variables considering all the jets together. Instead of performing a for loop as before a tensor-vector multiplication is computed

$$\mathcal{T}_x = T_{max} F_{\tau,x} \quad \mathcal{T}_z = T_{max} F_{\tau,z} \tag{3.29}$$

So the shear stress distributions $\mathcal{T}_x$ and $\mathcal{T}_z$ for the whole domain are computed in two operations: computing T_{max} and then multiplying it with the constant tensor,F_τ. The time saved, with respect to the previous implementation is shown in figure 3.4. For the shear stress the difference is more pronounced because in this case it was also avoided projecting the shear in the two directions.

Magnetic field distribution

An analogous approach was taken also for the magnets, in that the spatial distribution was computed for each magnet and then stored in a tensor. Each magnet can have a different maximum value and so vector-tensor multiplication was carried out avoiding the for loop. In this case the spatial distribution is a Gaussian as discussed in section 1.4.

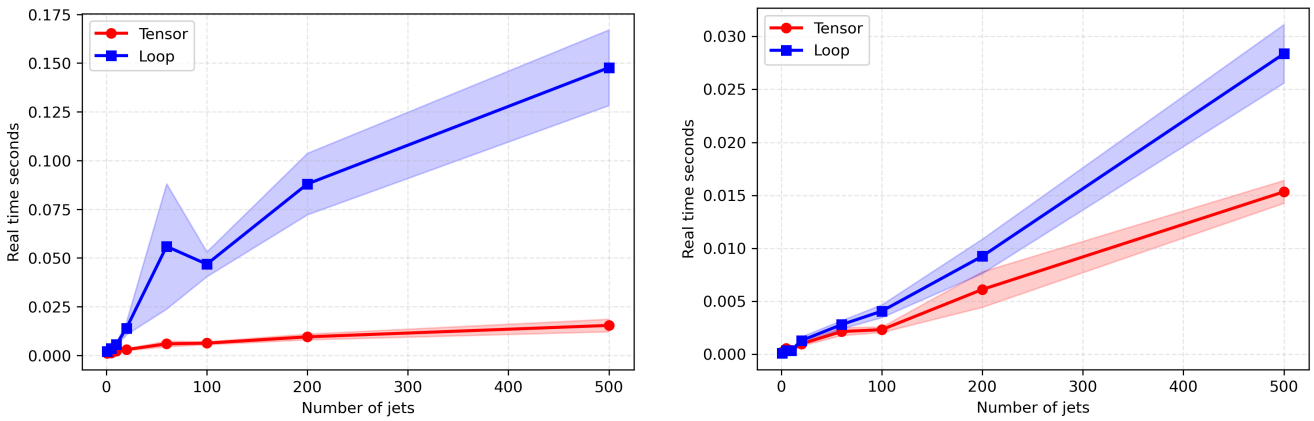

FIGURE 3.4: Time comparison between the tensor multiplication implementation (new) with respect with the old one with the for loop

Chapter 4

Machine Learning Control

At this point the theoretical model can be efficiently be solved by the numerical methods, which were touched upon in 2, exploiting the measures of 3. The jet impinging over the liquid film simulation can take place, it is now the focus of the work to investigate which is the best way to operate such jets in order to obtain the desired final coating. To do so Machine Learning (ML) techniques will be exploited. It is a vast and ever growing topic and therefore the goal of this chapter is to provide an overview of the main models that have been employed.

The techniques introduced are the same ones that have been compared in the work by Pino et al. [19], which is recommended for more insight and quantitative comparisons between the different approaches.

4.1 Reinforcement Learning Closed-Loop Control

The problem at stake is closed-loop. It involves a system whose behavior is influenced by the outcomes of previous actions, guided by feedback from the environment. In this scenario, an agent makes decisions to achieve a desired goal or maintain a specific state, adapting its actions based on the received feedback. The liquid film with its constitutive relations is the environment. The jets are the agent since they take actions on the environment. Their aim is to obtain an homogeneous film, this information (reward), together with the incoming film height (observations) will be given as a feedback.

In Figure 4.1, the closed-loop system is illustrated, depicting the agent's progression through states, actions, and rewards over time. At each time step t, the agent is in a certain state s_t, takes an action a_t, transitions to the next state s_{t+1}, and receives a reward $r(s_{t+1})$ from the environment.

4.2 Reinforcement Learning Framework

Objective of the Agent

The primary objective of the agent is to maximize a reward function $r(s_{t+1})$ by selecting appropriate actions. However, the challenge lies in determining how the agent should choose its actions. This leads to the goal of the agent: learning an optimal policy, represented as $\pi(s_t, t)$, which maps states to actions.

Value Function

In reinforcement learning (RL), the value function ($v_\pi(s)$) plays a crucial role in the decision-making process of an agent. This function is fundamental for assessing the long-term rewards

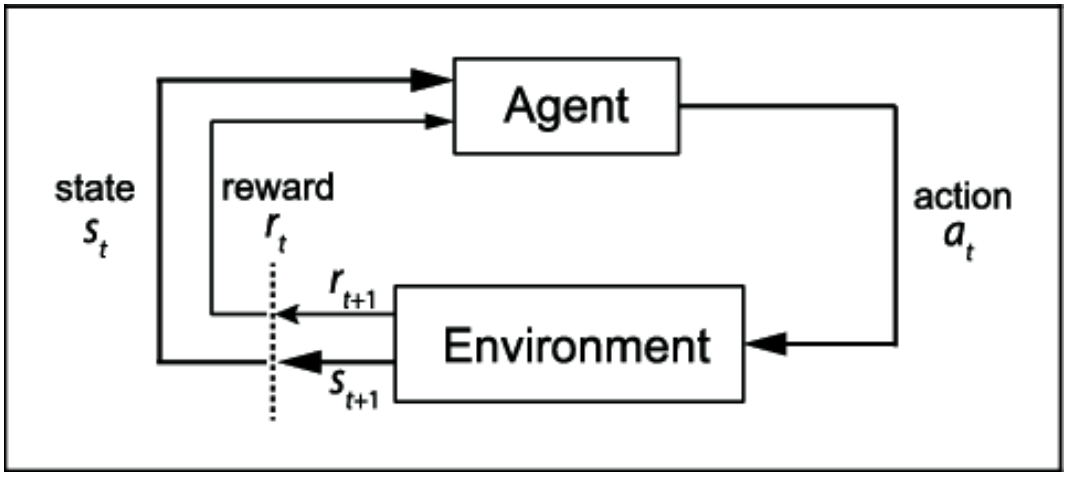

FIGURE 4.1: The agent operates within a loop with the environment

associated with a particular state under a given policy. In simpler terms, it helps the agent understand how beneficial a state is in the context of achieving its goals.

Definition of the Value Function

The value function is formally defined by the expected return (G_t), denoted as $v_\pi(s)$:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

This expectation represents the average sum of rewards the agent anticipates when starting from state s and following policy π. However, expressing this expectation directly as a sum over an infinite horizon is often impractical. To address this, the expectation can be expressed as the sum of discounted rewards:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_t = s \right]$$

Here,

- γ is the discount factor, representing the agent's preference for immediate rewards over delayed ones.
- R_{t+1} is the reward obtained at time $t + 1$.

This formulation encapsulates the idea that the agent values immediate rewards more than future rewards, and the discount factor ensures that future rewards are appropriately weighed. The summation extends over an infinite time horizon, capturing the agent's consideration of long-term consequences.

Role of the Value Function

The value function serves as a critical guide for the agent. By estimating the value of different states, the agent can make informed decisions about which actions to take. The objective is to navigate towards states with higher values, indicating greater potential for accumulating

rewards. In essence, the value function influences the selection of actions by helping the agent prioritize states that contribute more to its long-term success.

The agent in reinforcement learning (RL) has the flexibility to learn either a policy directly or, alternatively, learn a value function. This choice profoundly influences the agent's decision-making strategy.

Learning a Policy

Learning a policy ($\pi(s_t, t)$) involves the agent directly mapping states to actions. In this approach, the agent aims to understand the optimal sequence of actions for each state, without explicitly estimating the value of each state. While direct policy learning is intuitive, it may require extensive exploration to discover effective strategies, especially in complex environments.

Learning a Value Function

Alternatively, the agent can choose to learn a value function ($v_\pi(s)$). The value function provides estimates of the expected long-term rewards associated with being in a specific state under a given policy. This allows the agent to assess the potential benefits of different states, guiding its decision-making process.

Role of the Value Function in Policy Adoption

The value function becomes particularly powerful when the agent adopts a greedy policy. A greedy policy involves selecting actions that lead to states with higher values. In other words, the agent prioritizes actions that are expected to maximize cumulative rewards over time.

Q-Function in Reinforcement Learning

Another crucial concept in reinforcement learning is the Q-function ($Q_\pi(s, a)$), representing the expected cumulative rewards of taking action a in state s under a policy π. The Q-function is instrumental in evaluating the desirability of different actions, providing valuable insights into the optimal decision-making process.

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right]$$

Understanding the Q-function is integral to the exploration of action spaces and the refinement of policies for achieving better long-term rewards.

Unsupervised Nature of Reinforcement Learning

Reinforcement learning is inherently an unsupervised learning technique. Unlike conventional supervised learning scenarios, RL operates autonomously, with data not provided by a database or an expert but rather generated by the algorithm itself during its interactions with the underlying system[24]. Traditional methods of formulating effective control strategies often involve delving deeply into the underlying system dynamics and constructing formal models. These approaches can be resource-intensive and time-consuming. Moreover a control law explicitly based on a mathematical or physical model might lead to relying too much on pre-existing assumptions, resulting in less flexible strategies.

Exploration-Exploitation Trade-off

A critical challenge in RL is the exploration-exploitation trade-off. Exploration involves trying new actions to understand the environment, while exploitation focuses on selecting known optimal actions. Striking the right balance is essential for effective learning. Stochastic policies, incorporating randomness, naturally support exploration, whereas deterministic policies may struggle in uncertain environments.

4.3 Policy Optimization

In the realm of reinforcement learning, policy optimization stands as a pivotal approach to enhancing an agent's decision-making capabilities. At its core, policy optimization revolves around refining the agent's strategy, or policy, to maximize cumulative rewards in a given environment. The objective function serves as the compass for the agent, providing a quantitative measure of its performance and guiding the adjustments to its policy. In the context of reinforcement learning, the objective function encapsulates the final goal of the agent: to maximize cumulative rewards over time. The reward function assigns a numerical value to each state-action pair, quantifying the immediate desirability of a particular transition. The objective function aggregates these rewards over time, encapsulating the agent's aspiration to achieve the highest possible cumulative reward.

$$J(\theta) = \mathbb{E}[r] \tag{4.1}$$

So if the objective function is high then the model is getting a lot of reward during an episode. The process of policy optimization often involves the meticulous search for the optimal policy parameters. This search is driven by the gradient of the objective function, a vector that points in the direction of the steepest increase in the objective function. Effectively, the gradient guides the agent in adjusting its policy to traverse the landscape of possible actions and states, aiming for regions associated with higher cumulative rewards. This process of searching for the gradient reflects the iterative nature of policy optimization. As the agent explores and interacts with the environment, it continually refines its policy parameters to align with the direction that promises enhanced performance.

How can this gradient be computed in practice? Considering the policy parameters, θ, of the objective function, $J(\theta)$, its gradient can be expressed as:

$$\nabla_\theta J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_1}, ..., \frac{\partial J(\theta)}{\partial \theta_n} \right) \tag{4.2}$$

However if the gradient is not known we can think of perturbing the parameters in every directions and see how this changes the objective function. This gives a numerical estimation of the gradient. This is not particularly efficient because for each component of the gradient a perturbation and a function evaluation are needed.

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{J(\theta_j + \epsilon) - J(\theta_j)}{\epsilon} \tag{4.3}$$

To overcome this problem it is possible to analytically reformulate the gradient of the objective function. If the case where only one action is taken and then the episode terminates, the

objective function can be rewritten as:

$$J(\theta) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \mathcal{R}_{s,a} \tag{4.4}$$

J is the sum of the probabilities of being in each state ($d(s)$), multiplied by the sum of the probabilities of taking each action in that state ($\pi_\theta(s,a)$), multiplied by the immediate rewards obtained ($\mathcal{R}_{s,a}$). In simple terms, it's a way for the agent to calculate the expected cumulative rewards over all possible states and actions, considering the probabilities of encountering each state and taking each action. The agent's goal is to adjust its policy parameters θ to maximize this expected sum of rewards.

But the gradient of this function is what is needed in order to update the parameters θ

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \tag{4.5}$$

To get to it, the gradient of 4.4 is taken

$$\nabla_\theta J(\theta) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \nabla_\theta log(\pi_\theta(s,a)) \mathcal{R}_{s,a} \tag{4.6}$$

Where, for convenience, the gradient of the policy is expressed as

$$\nabla \pi = \pi \nabla log(\pi) \tag{4.7}$$

This is useful because it allows to rewrite the equation 4.6 as an expectation

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta log(\pi_\theta(s,a)) \mathcal{R}_{s,a}] \tag{4.8}$$

This means that it can be estimated with a sample mean. If a set of trajectories is collected

$$\mathcal{D} = \{\tau_i\}_{i=1,\dots,D} \tag{4.9}$$

Where each trajectory is obtained by letting the agent act in the environment using the policy π_θ, the policy gradient can be estimated with

$$\nabla_\theta J(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) \mathcal{R}_{s,a} \tag{4.10}$$

where $|\mathcal{D}|$ is the number of trajectories in $\mathcal{D}$. The equation 4.10 is fundamental, to recap once more it allows to compute the gradient just by knowing the expression of the policy and by running some trajectories, by doing so all the components of the gradient will be updated. Knowing the policy expression is not so uncommon as it might seem. Once the gradient is known it is possible to update each of the parameters θ according to it in order to ascend the objective function. For a better understanding of the topic the reader is redirected to the work [11].

4.4 PPO Algorithm

Proximal Policy Optimization (PPO) is a family of model-free reinforcement learning algorithms. PPO algorithms are policy gradient methods, which means that they search the space

of policies rather than assigning values to state-action pairs. So rather than selecting the actions that maximize the action value function, this algorithm will directly try to learn the q-function. This is done in order to allow the model to scale, that is to be able to solve problems with more states and with higher complexity. Moreover policy methods have generally better convergence properties and are able to learn stochastic (as opposed to deterministic) policies.

PPO builds on policy gradient ideas but it replaces the two main terms in equation 4.8 in order to take the biggest possible improvement step on a policy. The step should not be too large, otherwise it might cause performance to collapse. The algorithm was first introduced in the paper by Schulman et al. [23]. The way the policy parameters are updated in time from time step m to the following $m+1$ is

$$\theta_{m+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_m}} \left[L(s,a,\theta_m,\theta) \right], \tag{4.11}$$

Or to be more adherent to the code

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t,a_t),\ g(\epsilon, A^{\pi_{\theta_k}}(s_t,a_t)) \right) \tag{4.12}$$

The term summed at each time step is L, is the *surrogate advantage*, a measure of how policy performs relative to the old policy:

$$L(s,a,\theta_m,\theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_m}(a|s)} A^{\pi_{\theta_m}}(s,a),\ \ g(\epsilon, A^{\pi_{\theta_m}}(s,a)) \right) \tag{4.13}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases} \tag{4.14}$$

Just considering the first term of 4.13 it is possible to see a resemblance with 4.10, the policy gradient term $\nabla_\theta log(\pi_\theta(s,a))$ has been replaced with a ratio that is greater than one if an action is more probable under the new policy and smaller otherwise. The reward term is written in terms of the so called advantage function

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{4.15}$$

The advantage function $A^\pi(s,a)$, of a policy π, describes how much better it is to take a specific action a in state s, over randomly selecting an action according to $\pi(\cdot|s)$. Supposing that the advantage for that state-action pair is positive, in which case its contribution to the objective reduces to

$$L(s,a,\theta_m,\theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_m}(a|s)}, (1+\epsilon) \right) A^{\pi_{\theta_m}}(s,a). \tag{4.16}$$

Because the advantage is positive, the objective will increase if the action becomes more likely, that is, if $\pi_\theta(a|s)$ increases. But the min in this term puts a limit to how much the objective can increase. Once $\pi_\theta(a|s) > (1+\epsilon)\pi_{\theta_m}(a|s)$, the min kicks in and this term hits a ceiling of $(1+\epsilon)A^{\pi_{\theta_m}}(s,a)$. PPO is a powerful tool and the detail of its implementation are beyond the scope of this work. The takeaway of this section is that the state of the art algorithms in RL are capable of striking a balance between effectiveness and ease of use. Such an algorithm was not

coded from scratch but a library was exploited, that required the BLEW environment to be set up in a specific manner. In section 4.6 an example of its effectiveness will be provided, while also illustrating how to set up an appropriate environment. Before that some of the parameters used during the training phase are introduced.

4.5 Machine Learning Parameters

When running reinforcement learning simulation it is important to monitor how the training process is evolving. However it is not always possible to directly visualize the episodes as they progress, therefore it is important to introduce some metrics that are provided by the library used, in this case stable baselines. The most intuitive way to measure how the agent is behaving is by observing the episode mean reward. It is a sum of all the rewards collected by the agent during one episode. In BLEW the reward function implemented reflects the non-smoothness of the liquid film. Therefore having a less negative episode mean reward means that the agent is obtaining a final flow with fewer ripples. It is important to always compare this parameter to the value it has in the non controlled case, that is when the control jets are not taking actions. An agent that behaves well will therefore achieve a lower episode mean reward compared to the non-controlled case. Although the episode mean reward is by far the most indicative metrics, there are more technical parameters that also show how the algorithm is performing during the training process. In the case of PPO, an on-policy actor-critic method, the main parameters are the following, as found in Raffin et al. [22]

- Approx kl: Approximate mean KL divergence between old and new policy, it is an estimation of how much changes happened in the update. So the KL divergence is a measure of how one probability distribution diverges from a second expected probability distribution. It measures the difference between the old policy and the new policy in terms of the actions they select.

- Entropy loss: Mean value of the entropy loss. During the training process, the entropy loss should decrease over time. This is because, as the agent learns and the training progresses, the policy becomes more deterministic and converges towards near-optimal strategy. As a result, the entropy loss decreases, indicating that the policy becomes more certain about its actions and that there is less and less need for exploring.

- Std: Current standard deviation of the noise. This value indicates how much variability or randomness is introduced into the agent's actions at a particular point in the training process. A higher standard deviation leads to more exploratory and random actions, which can be useful in the early stages of training when the agent needs to explore the environment thoroughly. As training progresses and the agent becomes more knowledgeable about the environment, the standard deviation might be reduced to focus more on exploitation.

- Value loss: Current value for the value function loss for on-policy algorithms, usually error between value function output and Monte-Carlo estimate. During training, the agent uses this value function to assess the quality of its actions and states. The value loss measures the discrepancy between the predicted values from the current value function and the values obtained from actual interactions with the environment.

- Explained variance: Fraction of the return variance explained by the value function. It measures the proportion of the variance in the returns that can be explained by the predictions made by the value function. In other words, it assesses how accurately the value function approximates the true expected returns. A higher explained variance indicates that the value function is doing a good job at predicting the returns and capturing the underlying patterns in the environment. Conversely, a lower explained variance suggests that the value function is not accurately capturing the variance in the observed returns, indicating a need for improvement in the value function's predictions.

4.6 Application of reinforcement learning

In the upcoming pages examples on how to leverage these RL ideas will be shown. What stood out the most is the fact that such algorithms are well documented and ready to use with minimal adjustments from the user. This has been made possible thanks to packages such as Stable Baselines [22], which provide open source implementations of Reinforcement Learning algorithms based on OpenAI Baselines. To get familiar with the topic some examples from the Gym environment were considered. The strength of this tool resides in the common structure of said environments. These environments range from short games, to arcades. Each one of them is defined by an action and state space. The *Mountain Car* environment, for instance, presents itself as follows: a car is placed at the bottom of an hill and has to reach the top, situated on the right of the vehicle. The agent, or driver, can only decide when to accelerate, this is the action. For the sake of simplicity let us assume that the action space is discrete (non continuous) and therefore our driver will only be able to choose between three different actions at each time step. It will either accelerate to the right, to the left or just do nothing. However here is the twist, the agent will never reach the top if it just keeps on accelerating to the right. In order to succeed it has to somehow learn how to swing in order to progressively increase its speed. The agent which has to learn how and when to accelerate has access to the environment thanks to the observation that it receives. In this specific case the observation space is a two dimensional array, the first value is the position of the car along the x-axis and the second one is the velocity of the car. The last fundamental information that the environment provides to the agent is the reward, in this example since the goal is to reach the flag placed on top of the right hill as quickly as possible, the agent is penalized with a reward of -1 for each time step. Meaning that an agent that reaches the top after 300 steps will end the episode with a score of -300 and therefore will have a better performance than an agent that finished with a score of -500. There is also a limit on the episode length, so that an agent that is not able to reach the top will still be terminated after a set number of time steps. This also means that all the agents that are not able to climb the hill will end up with the same final score, regardless of how close they got to it. To summarize, a gym environment is characterized by the possible actions the agent can take, which according to the internal dynamics lead to the next state. The agent will be informed on the state reached, by the observation vector, and will also receive a score, the reward. The hidden obstacle of this example is that if the car never reaches the top it will never learn anything since it only gets a reward (in this case the reward is the absence of negative rewards) when it completes the climb. Moreover at the start of the episode it does not know anything and by acting randomly it will take a long time to reach the top, if it manages at all. There are two ways solve this problem: either to change how the reward process works or letting the episode go on for a long time, making it more likely that even a bad agent will get to the top after a while. To give a feel for how the different concepts introduced in section

4.2 might be interpreted, let us first consider the value function. In this example a well trained value functions would probably associate a greater value to a state with the car in a high point and speed, since if this is the state the agent is more likely to reach the hill top. The q-function of such a state-action pair should be greater if the action is not opposed to the motion, since to create the swinging the acceleration should always be concordant with the velocity vector.

4.6.1 Mass-Spring-Damper System

In order to familiarize with a control loop on a simple system, the problem consisting of a mass with a spring and a damper was tackled. The goal is to train an agent able to bring the system to rest, given initial speed and displacement. To make things more interesting the mass is forced by a term with the resonance frequency. Therefore if the agent doesn't take any actions the oscillations would initially keep on growing. There are some similarities with the BLEW environment, since the waves that perturb the liquid film are comparable to the forcing term and the agent, though the jets has to bring the system to a stop, just like in this case. The system, represented in figure 4.2, is governed by the equation

$$m\ddot{x} + c\dot{x} + kx = F(t) + U(t) \tag{4.17}$$

$F(t)$ is the forcing term and $U(t)$ is the force applied by the agent. Notably the forcing term is $F(t) = Asin(ft)$, where the frequency is $f = \sqrt{k/m}$, where k and c are respectively the spring and the damping constants and m is the mass. Alongside the dynamics of the system also the action and state spaces are specified. In this particular case the actions allowed are continuous and ranging between -1 and 1. On the other hand the observation state goes from minus infinity to plus infinity, meaning that the all values of x, the displacement, and the speed, $\dot{x}$ are allowed as long as they agree with equation 4.17. The last characteristic to be defined is the loss function, which is a function that maps a certain state to a scalar, the goal of the agent is to minimize its value, it has the same role as the reward. The chosen relation is

$$loss = \int_0^T 100x^2 + 100\dot{x}^2 dt \tag{4.18}$$

As long as the model is concerned the agent will have few losses when it obtains low values of displacementand speed $\dot{x}$.

The code is structured as follows: first a new class is defined, it inherits from the gym.Env class, making it compatible with the OpenAI Gym interface for reinforcement learning environments. Then the constructor initializes the environment. It sets up the action and observation spaces, as well as the parameters for the mass-spring-damper system. Next up is the step method, which defines how the environment evolves when a step is taken. It calculates the next state, reward, and whether the episode is done based on the current state and the action taken. From equation 4.17 the acceleration a is calculated (using the speed at the previous step, $\dot{x}^n$) and then the following assumptions are made

$$\dot{x}^{n+1} = \dot{x}^n + \Delta t a \tag{4.19}$$

$$x^{n+1} = x^n + \Delta t \dot{x}^{n+1} \tag{4.20}$$

The last method to be defined is the reset, as the name suggests, it resets the environment to its initial state. It resets the state variables and returns the initial state. It is now time to train the agent, doing so does not require much coding at all, as long as the environment is properly

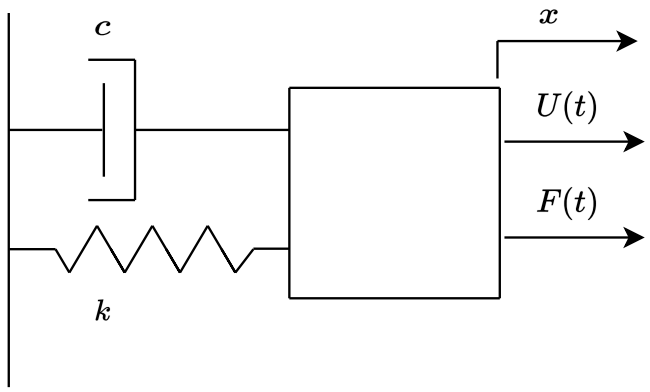

FIGURE 4.2: Dynamical system with a forcing term that pulses at the resonance frequency

set. However choosing the algorithm and its parameters is not so trivial. For this example the PPO algorithm was chosen and it proved itself to be a fast and adaptable option. Some of the parameters that it is possible to change are the batch size, the activation function, the size of the two neural networks (actor and value function) and the number of steps to run for each environment per update. During the code testing some general trends have emerged, for instance having a policy network with fewer layers compared to the value function networks seemed to speed up the learning process without loosing performance. The batch size also influences the learning curve: having a smaller batch size resulted in more policy updates and therefore in a usually slower training. The model was trained for a total of half a million time steps. The results are shown in figure 4.3. It is possible to see that the model long term strategy is to apply a force that is out of phase with respect to the forcing term. Whereas at the start of the episode it applies the maximum force it is allowed to, that is -1. It is worth noting that the displacement never goes below zero, meaning that the spring is first extended with respect to the rest position, then the displacement reaches a maximum and then the mass starts going back and it is stopped when it reaches the starting position without ever compressing the spring. A question that arose it was whether this control was a closed or open loop and whether the model which was trained only with one set of parameters would be able to also control systems with other mass, or spring constant values. In order to verify it the agent was trained with the same parameters as before but an episode was run with different values. For instance changing the spring constant or having a lighter mass, which resulted in faster oscillations (as it also changed the forcing natural frequency). The agent behaved quite well and was able to bring the system to a rest in all instances. It struggled only when the amplitude of the forcing term was particularly high, probably because the agent would have wanted to take stronger actions, but the maximum value allowed was still ± 1. One other question was whether the model would be able to adapt in the middle of an episode. So one episode was run with a frequency that would gradually change, becoming faster with time. Once again the agent managed to control the mass and was able to apply an opposed and out of phase force adapting in the middle of the episode as shown in figure 4.3.

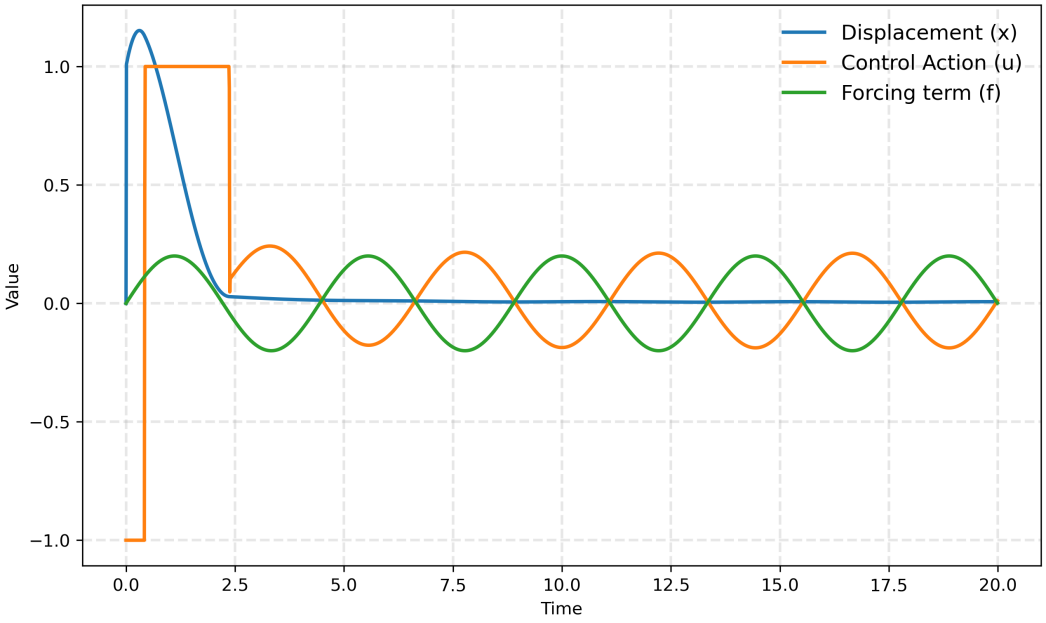

FIGURE 4.3: At the start of the episode the mass is in the resting position but the velocity is not zero so it starts moving (blue). The forcing term (green) and the force applied by the agent (yellow) are out of phase

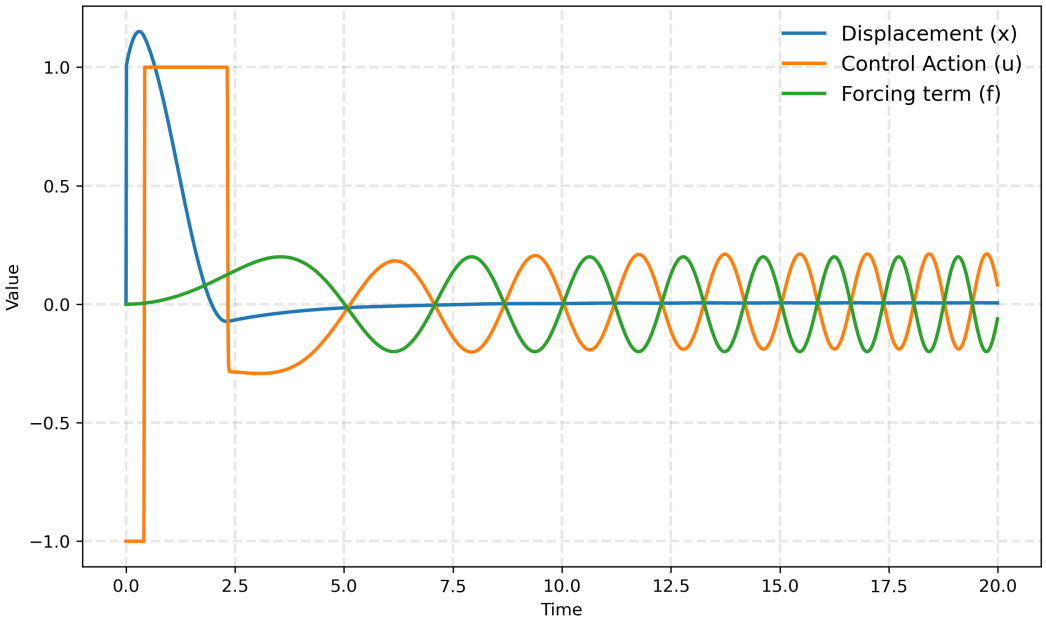

FIGURE 4.4: Unlike the other episode the mass is forced with a varying frequency. Nonetheless the agent is still able to bring the system to a stop

The last aspect that was briefly analyzed is whether having an open loop control law would manage to bring the system to rest. For this purpose an episode was run with an action that was no longer chosen by the trained agent but it was something hard coded in a way that would resemble the yellow control action of figure 4.3. The advantage of an open loop control is the fact that the controller action is independent from the "process output" and therefore there are

no sensors involved. For this reason when writing down an action strategy that resembled the one of the trained controller we did not want the relation to be too complex. The first of the two strategies that were tried out is a completely out of phase action,with respect to the forcing term. The other action was first trying to brake the system with a constant action like in graph 4.3 and then opposing the forcing term. Both methods performed significantly worse than the closed-loop control case but they both managed to at least damp the oscillations. This is due to the fact that there is a damper that acts on the mass and since the control action cancels out with the forcing term the damper will eventually bring the system to a rest. In figure 4.5 two phase plots are shown, that is the velocity as a function of the displacement for two episodes. One is controlled by the agent and the other one with the completely out of phase action.

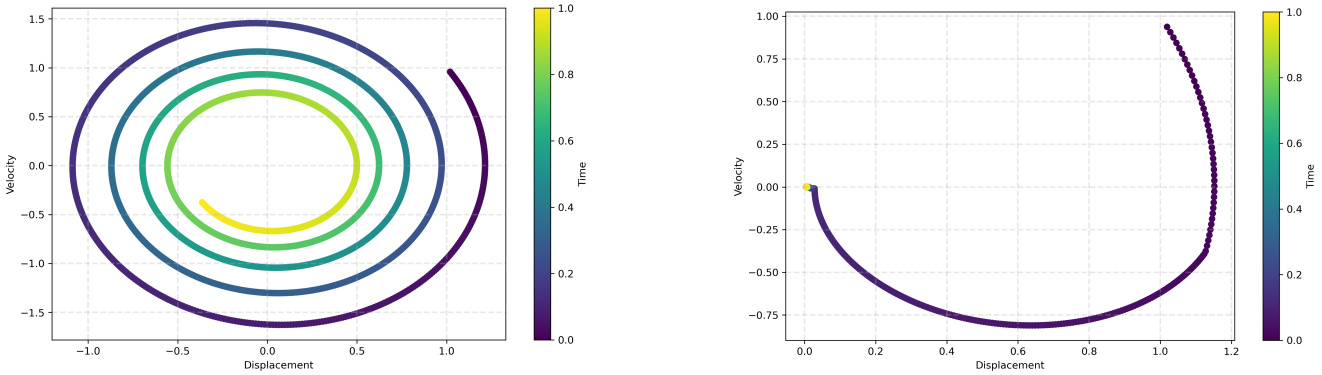

FIGURE 4.5: A darker color indicates the episode start, with the following steps it fades into a lighter color. It can be seen that on the openly controlled system the damping will eventually lead the spiral towards the point (0, 0). On the right the controller brings the mass to rest quicker

4.7 BLEW Environment

So far the mathematical model has been discussed in chapter 1. The numerical methods employed in order to solve such equations and some other necessary tools have been shown in the chapters 2 and 3 respectively. In chapter 4 the ML models that are going to be used have been introduced. At this point is thus necessary to explain how these different aspects come together in order to supply the required results.

In order to optimize the agent with stable baselines it is necessary to set up the code in a way that is compatible with gymnasium. As briefly discussed in chapter 4.6 a gym environment has three fundamental functions within it. It has to contain a function where all the variables are initialized, for instance the domain, the perfectly matched layer and the jet positions. The step function is where the time integration takes place and where the action of the jet is defined. The action will be defined by the agent trying to maximise the reward function. Finally a reset function takes care of bringing the system to its initial state. The machine learning algorithm will receive observations in terms of $\hat{h}$, the adimensional height of the liquid film, in a certain number of points. Depending on the number of observation points the observation space of the environment will be defined. The action space is also continuous but its limits are set in order to reflect the technical specifications of the actual jets. The dimension of the action vector depends on the number of control jets, this tells the model how many output values it has to return.

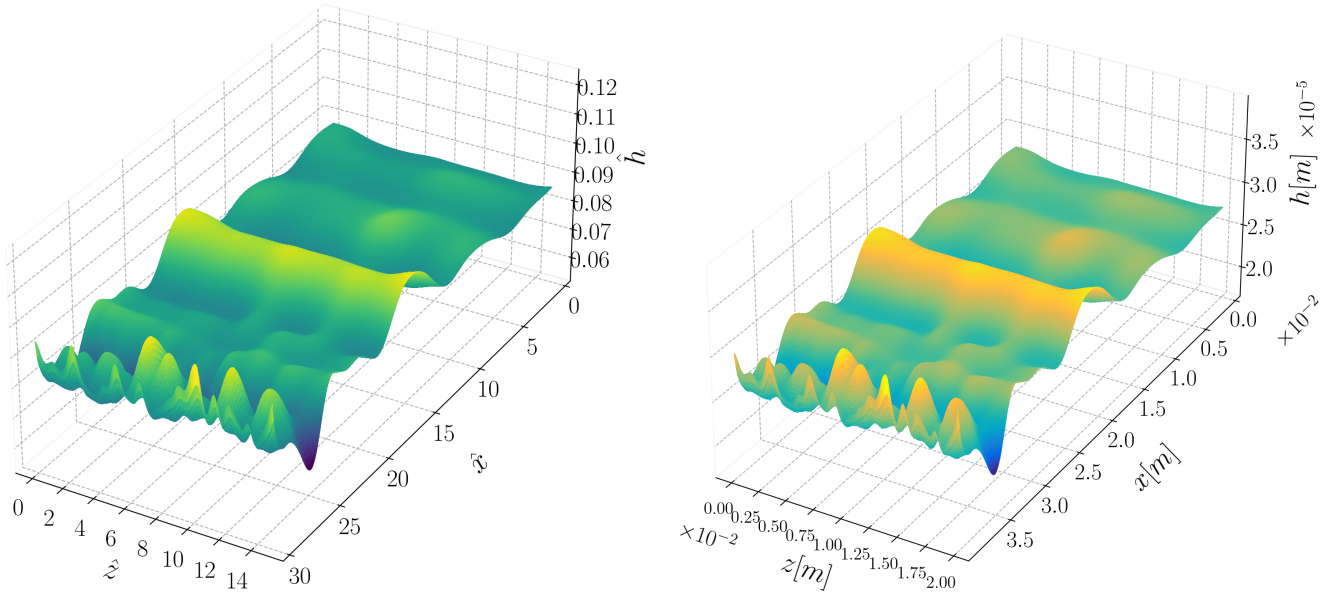

FIGURE 4.6: 3D plot of the non-dimensional (a) and dimensional (b) liquid film with the undulation instability generated by the unsteady Dirichlet boundary conditions at $\hat{x} = 30$.

Some additional measures have been taken in order to create an environment that has some levels of randomness to it, otherwise the agent might learn how to behave just in that one instance, this problem is generally referred to as *overfitting*. Meaning that the agents achieves the goal for a specific data set but if faced with different initial conditions it will perform significantly worse. For this reason the actuation law of the forcing jets, determined by 4.25, had some randomness to it. An interval of amplitudes, frequencies and phases were chosen in such a was that would still make the perturbation of the undulation kind (see subsection 4.7), but without repeating itself. Moreover when dealing with multiple jets a phase shift was added from one to the other in order to achieve a more random 3D wave. The forcing jets have the task of creating the perturbation wave that cannot be introduced as a boundary condition because of the periodicity requirements of the spectral methods. Finally since having faster simulation is a crucial aspect, the domain was set up with jets positioned quite close to the PML in order to exploit the space as well as possible. Moreover the number of points in the spanwise direction was reduced since the direction in which the substrate is moving is more relevant.

Spectral environment with zinc

This environment simulates the undulation instability in galvanizing line conditions using the 3D Integral boundary layer model with magnetic and blowing gas jet actuators. We use the physical properties of the liquid zinc (reported in Table 4.1) with a strip velocity $U_p = 2.5[m/s]$ and a magnetic field $[b] = 0.23[\mathrm{T}]$, which results in a reduced Reynolds number $\delta = 395.2$ and a Hartmann number of $Ha = 6$. The liquid is assumed flat with an initial thickness of $\hat{h}_0 = 0.1$ $(34[\mu m])$.

Figure 4.7 shows the computational domain, Ω, which is composed of a physical domain Ω_1 of dimensions $L_x = 30$ and $L_z = 15$, surrounded by a perfectly matched layer (Ω_{PRL}) of dimension $L_{\hat{x},PML} = L_x/6$ and $L_{\hat{z},PML} = L_z/6$ equal on all directions with periodic boundary conditions on the boundaries $\partial\Omega$. The physical domain comprises a set of control actuators

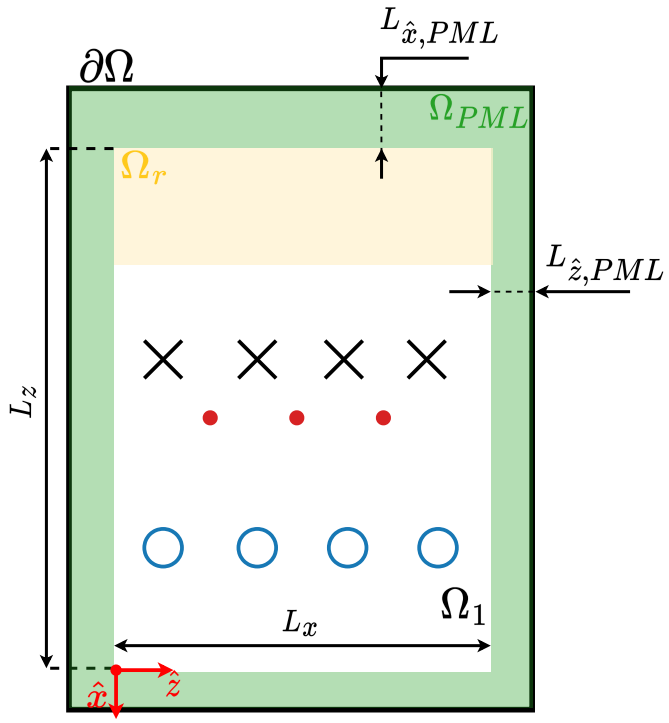

FIGURE 4.7: Scheme of the spectral environment with the physical domain Ω_1 containing the actuators (black crosses), the observations (red dots), the forcing jets (blue circles) and the reward area (light orange region), surrounded by an absorbing perfectly matched layer (green region) with periodic boundary conditions on the found boundaries ($\partial\Omega$). The figure is not to scale

	$\boldsymbol{\rho}$ (kg/m^3)	$\boldsymbol{\mu}\cdot 10^{-3}$ (Pa s)	$\boldsymbol{\sigma}\cdot 10^{-3}$ (N/m)	Ka	Re
Liquid Zinc	6570	3.5	700	11525	$14-43\cdot 10^{-3}$

TABLE 4.1: Liquid properties (density, dynamic viscosity, and surface tension from left to right), Kapitza number (Ka) and range of Reynolds numbers (Re) for substrate velocity $U_p \in [0.1, 100]$ for liquid zinc

(black crosses), some observation of the liquid film height (red dots) and a set of artificial forcing jets aligned along the $\hat{z}$ direction (blue circles).

These synthetic jets serve as a means to reproduce the undulation instability, appearing on the right-hand side of the source term, as additional pressure and shear stress distributions:

$$\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 0 \\ \delta^{-1}\Big[\hat{h}\Big(-\partial_{\hat{x}}\hat{p}_g - \partial_{\hat{x}}\hat{p}_f + \partial_{\hat{x}\hat{x}\hat{x}}\hat{h} + \partial_{\hat{x}\hat{z}\hat{z}}\hat{h} + 1\Big) + \hat{\tau}_{g,x} + \hat{\tau}_{f,x} + \hat{\tau}_{w,x} - Ha\hat{B}^2\hat{q}_x\Big] \\ \delta^{-1}\Big[\hat{h}\Big(-\partial_{\hat{z}}\hat{p}_g - \partial_{\hat{z}}\hat{p}_f + \partial_{\hat{z}\hat{z}\hat{z}}\hat{h} + \partial_{\hat{z}\hat{x}\hat{x}}\hat{h}\Big) + \hat{\tau}_{g,z} + \hat{\tau}_{f,z} + \hat{\tau}_{w,z} - Ha\hat{B}^2\hat{q}_z\Big] \end{pmatrix} \tag{4.21}$$

where $\hat{B} = \sum \hat{B}_i$ is the total nondimensional magnetic field, and the subscripts g and f refer to the total nondimensional pressure and shear stress given by the contribution of the control and forcing actuators, respectively.

As described in section 1.4, the magnetic field of the single actuators $\hat{B}_i$ is modelled as a Gaussian centred at $(\hat{x}_{0,i}, \hat{z}_{0,i})$:

$$\hat{B}_i(\hat{x}, \hat{z}, \hat{t}) = B_{t,i}(t) e^{\frac{-(\hat{x}-\hat{x}_{0,i})^2}{2\gamma_{\hat{x}}^2} + \frac{-(\hat{z}-\hat{z}_{0,i})^2}{2\gamma_{\hat{z}}^2}} \tag{4.22}$$

where $\gamma_{\hat{x}} = 1$ and $\gamma_{\hat{z}} = 1$ are the standard deviations and $B_{t,k} \in [0, 1]$ is the control action at $k-th$ time step. The pressure and shear distribution of the control and forcing jets are modelled using the experimental correlations for impinging circular jets (see section 1.4) with the velocity at the jet nozzle exit $U_j \in [0, 50]$[m/s] used as the control parameter.

The undulation instability appears in the liquid film as 2D waves in the streamwise direction with slight variations along $\hat{z}$. Numerical evidence from LES[1] and 2D IBL[16] simulations suggest that these waves have a nondimensional frequency $\hat{f} \in [0.05, 0.2]$, an amplitude peak-to-peak A between 5 and 10 per cent of the unperturbed film thickness and wavelengths in the range $\lambda \in [0.25, 0.35]$[mm]. The undulation instability obtained with a LES is shown in pictures 4.6.

These waves are reproduced, setting the nozzle exit velocities $U^i_{j,f}$ of the forcing jets as a harmonic function of time with a phase shift $\phi \in [0.2\pi, 0.5\pi]$ to create a small variation in the spanwise direction:

$$U^i_{j,f} = A\cos\left(2\pi\hat{f}\hat{t} + \phi_i\right) \tag{4.23}$$

To give more variability in the simulation and to foster the learning of robust control functions, the amplitude A and the frequency $\hat{f}$ are modelled as finite Fourier series with randomly

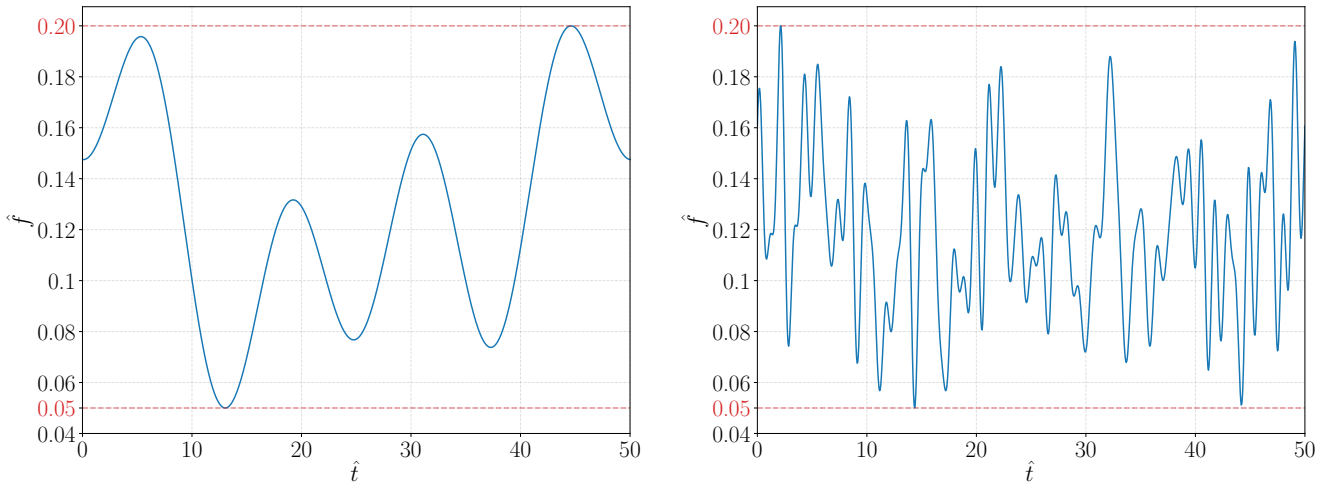

FIGURE 4.8: Evolution of the forcing jets' frequency using 4 (a) and 50 (b) harmonics bounded by in the rage of admissible undulation frequency (red dashed lines)

chosen coefficients [8]:

$$A(\hat{t}) = \sqrt{2}\sum_{j=1}^{m}\left[a_j \cos\left(2\pi j\hat{t}\right) + b_j \sin\left(2\pi j\hat{t}\right)\right] \tag{4.24}$$

$$\hat{f}(\hat{t}) = \sqrt{2}\sum_{j=1}^{m}\left[a_j \cos\left(2\pi j\hat{t}\right) + b_j \sin\left(2\pi j\hat{t}\right)\right] \tag{4.25}$$

where $m = 4$ is the number of harmonics and the coefficient a_j and b_j are sampled from the normal distribution $\mathcal{N}(0, 1/(2m+1))$. Before being passed to the forcing jets, these functions are rescaled in the range described above. The number of harmonics m defines the variability of the randomness of the final function. Figure 4.9 shows $\hat{f}$ for $m = 4$(a) and $m = 50$(b). As the number of harmonics increases the function is capable of more sudden variations.

To reproduce open-flow conditions, avoiding the waves re-entering the domain from the boundaries, a linear perfectly matched layer, described in 3.1, was included.

$\sigma_x(\hat{x}, \hat{z})$ and $\sigma_z(\hat{x}, \hat{z})$ are positive functions which determine the strength of the absorbing layer, defined as:

$$\sigma_x(\hat{x}, \hat{z}) = \begin{cases} 0, & \text{if } (\hat{x}, \hat{z}) \in \Omega_1 \\ \hat{x}^3, & \text{if } (\hat{x}, \hat{z}) \in \Omega_{PRL} \end{cases} \qquad \sigma_z(\hat{x}, \hat{z}) = \begin{cases} 0, & \text{if } (\hat{x}, \hat{z}) \in \Omega_1 \\ \hat{z}^3, & \text{if } (\hat{x}, \hat{z}) \in \Omega_{PRL} \end{cases} \tag{4.26}$$

The complete set of the equations was discretized in space using the Fourier spectral method over a uniform grid of 180 points along $\hat{x}$ and 88 along $\hat{z}$. The explicit Euler formula was used for the time integration with a time step $\Delta\hat{t} = \Delta\hat{x}/80$.

16 forcing jets placed at $\hat{x} = 3$ along the all length of the physical domain L_z. Figure 4.9 shows the evolution of the undulation waves produced by these jets in dimensional (a) and nondimensional form (b). The wavelength is between 25 and 27$[\mu$m$]$.

In the only magnetic case, we used the flux matrix $\mathbf{F}$ and the wall shear stress vector $\hat{\boldsymbol{\tau}}_w^M$. For the jets and the hybrid simulations $\mathbf{F}$ and $\hat{\boldsymbol{\tau}}_w$. The observations and jests positions are discussed in more detail. The objective of the control problem is to bring to rest the undulation

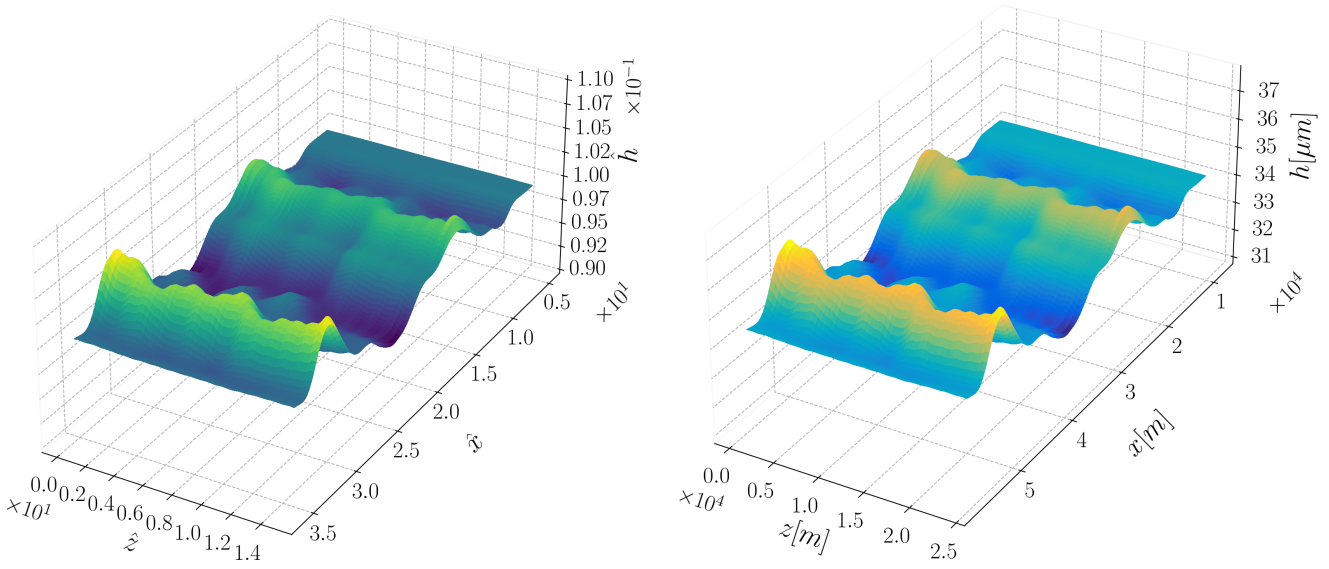

FIGURE 4.9: 3D plot of the non-dimensional (a) and dimensional (b) liquid film with the undulation instability generated by the forcing gas jets

perturbation with a reward function defined as:

$$r_k = e^{\text{std}((\hat{h})_{\Omega_r} - \hat{h}_0)\cdot 50} - 1, \tag{4.27}$$

where $\text{std}((\hat{h})_{\Omega_r} - \hat{h}_0)$ is the standard deviation of the liquid film at time step k in the reward area $\Omega_r = \{\hat{x}, \hat{z} |\ 9 < \hat{x} < 22\ \ \&\ \ 1 < \hat{z} < 14\}$.

The optimal control law was found using the reinforcement learning algorithm PPO. The training is run in parallel on $20 \div 30$ cpus for thousands of episodes. Different sizes of neural network for the PPO have been tried, generally having two hidden layers and with a number of neurons ranging in $128 \div 512$.

4.8 Observations Location

The information about the environment that the agent disposes of is represented by the observation points and the reward area. Feeding the algorithm with data that is representative of the problem, while not supplying an excessive amount of hard to decode numbers is really important. Not to mention the fact that in the actual production line those observation points would be expensive sensors. Measuring the thickness of a moving molten liquid film undulations that measure only tens of microns is not an easy task. The use of laser sensors is further complicated by the scattering effect of liquid metal, and their placement must be strategic, considering factors such as the proximity to impinging jets or magnets. Moreover they have a volume that has to be taken into consideration. Therefore finding the number and location of the observation points is not only a trade-off between cost and information fed to the agent, but several other aspects are to be taken into account. In previous versions of the code, it was tried to look for the points that were the least correlated to each others. The process started by sampling points according to the Latin hypercube method, a pattern that resembles rooks on a chess board that do not threaten each others. Once the observation points are chosen it is possible to estimate how much the information seen from a point is related to another one. So it was

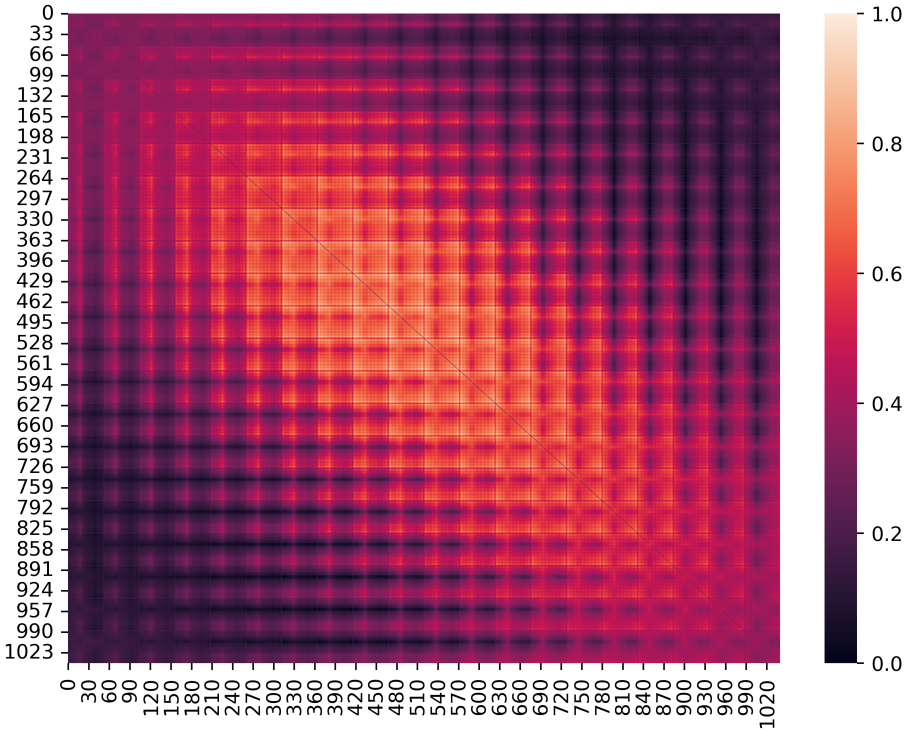

FIGURE 4.10: The correlation matrix shows with warmer colors a strong correlation, the points along the diagonal are more related to each others because those are the neighboring points

seen which points were the most related and one of the two was discarded. This was done in order to choose the minimum number of points, while retaining as much state representation as possible.

Something similar was carried out in this work, but instead of considering only certain points the full area in front of the control jets was sampled, this area had dimensions (m, n). For convenience the matrix was flattened into a column vector of size $(mn, 1)$. Then for each point of the area the observations at each time step of an episode were stored, for a total of n_t observations for each point. In the end the data matrix had dimensions (mn, n_t). Each columns represents all the observations for the whole domain at a certain time step. In order to see how much the observations of a point were correlated to all other points the following expression has been used

$$C_{ij} = \frac{O_i \cdot O_j}{\|O_i\| \|O_j\|} \tag{4.28}$$

If the correlation coefficient, C_{ij}, is 0, it means that the two signals are not correlated, otherwise if it is 1 they are fully correlated. The resulting matrix, C, is show in figure 4.10. The values of the matrix were computed only for the upper triangular part since it is symmetric (the correlation of point i to j is the same as j to i) and the values have been normalized, since, in general, the values were all quite close to each others. The matrix does not provide too much useful information as it is basically saying that points that are the most apart from each others also carry unrelated observations, as one would expect. Vertical and horizontal lines appear in the matrix mainly because in the transformation from a matrix to a vector some points that were

nearby in the 2D area are now far apart and vice versa.

However, considering the strong 2D character of the undulation it was decided to place the observation points along an horizontal line. This approach does, in a way, convey additional information to the agent. What is implicitly being expressed to the agent is the fact that it only needs to know how the height is distributed along a straight line given that this aligns with the direction of the wave peaks or valleys. Moreover having aligned observations makes it so that all the observed points will arrive underneath the line of jets (which are also horizontally aligned as in figure 4.7) at the same time. This is likely to make the observations easier to interpret. In the end it was decided to also place the control jets horizontally, this is mainly a matter of making the learning process easier. Moreover this arrangement is also more convenient if all the jets are actuated harmonically, and it also respects the physical constraints of the real sensors.

So far the observations have been taken as single points, that is the height value of a single cell. However another alternative would be to take a mean observation spanning multiple cells. It could happen that the value extracted from a single point does not reflect what is going on in its neighboring cells. Though a new issue would arise, that is the determination of the best radius for this purpose. One could think to consider a radius that is the one of the actual sensor employed on the production line, but since this information is not available and given the strong 2D character of the waves this path was not further explored.

4.9 Time step in the environment

The choice of the time step in the solver is primarily driven by the need for stability in simulations, this is were the choice of $\Delta\hat{t} = \Delta\hat{x}/80$ comes from. While a larger integration time step would lead to faster simulations, it could also compromise stability. However, it is essential to distinguish between the solver's time step and the time step in the environment, representing the time passed between consecutive actions. Allowing the agent to make frequent decisions within a short time span might not be conducive to effective learning. Due to the system's inherent reaction time, the agent might not observe the actual consequences of its actions, hindering the learning process. To have a feeling for the reason let us consider a Newton disc which has segments in different colors. Selecting too many actions is comparable to spinning the disc too fast. In this case it will be impossible for the naked eye to tell all the different colors apart and it will just be perceived as white. This is somewhat similar to what happens to the agent that is not able to trace a correlation between a certain action and reward. To address this issue, the decision was made to diminish the number of action selections, so that it would interact with the environment less often. This deliberate delay ensures that the agent can witness the outcomes of its actions, fostering a meaningful connection between its behavior and the rewards received. For example, suppose the agent decides to exert significant force on a liquid. By introducing a delay before the agent can choose its next action, sufficient time is given to the liquid to respond, approximately 20 numeric time steps. In summary, maintaining a clear distinction between the numerical time step for the solver and the perceived time step for the agent, as shown in figure 4.11, is crucial. By allowing the agent to interact less frequently with the environment simulations can be expedited without compromising the learning process or the integration stability. In practice this made quite a big difference, after this was tried the results showed a much improved learning curve.

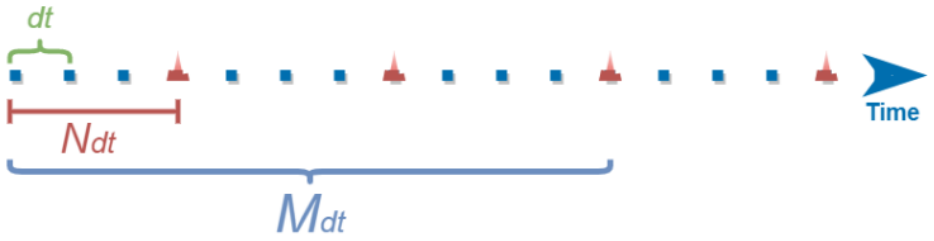

FIGURE 4.11: The green dt is set according to stability reasons. The red line indicates the time step seen by the agent, while the blue one is the length of a episode. The picture is not to scale

4.10 Reward

The reward expresses how well the agent is behaving and together with the observations it is how the agent acquires information from the environment. The reward has to convey the fact that the goal of the agent is to flatten the liquid film by bringing the flow close to the unperturbed initial height. Therefore it has to consider the liquid surface after the control jets and estimate the roughness. The reward expression can be broken down into two major aspects: the area considered and the law used to evaluate the performance. A reward function should be:

- Aligned with objectives: it should align with the ultimate goal of the task
- Sparse but Informative: it should provide enough information for the agent to learn the optimal policy without overwhelming it with constant feedback
- Smooth and Continuous: sudden jumps or discontinuities in the reward landscape can lead to difficulties in convergence.
- Avoiding Reward Hacking: exploiting loopholes in the reward function to achieve high rewards without actually fulfilling the task's goal.

4.10.1 Reward function

Since the final goal is to have a flat and homogeneous film, the reward function has to somehow evaluate the deviation of the film with respect to the undisturbed case. This was done considering the standard deviation between the ideal (completely flat, $h = h_0$) and the actual height distribution

$$\sigma(h_{rew}) = \sqrt{\frac{1}{h_{rew}} \sum_{\Omega_{rew}} (h_{rew} - h_0)^2} \tag{4.29}$$

However it would not be practical to just give this parameter as reward, one issue with it is that it is not bounded. Different approaches have been investigated in the work carried out by a previous student on similar control problems [17]. In that report different functions of the standard deviation had been tried, for instance linear negative with $r(s_t) = -\sigma(h_{rew})$,

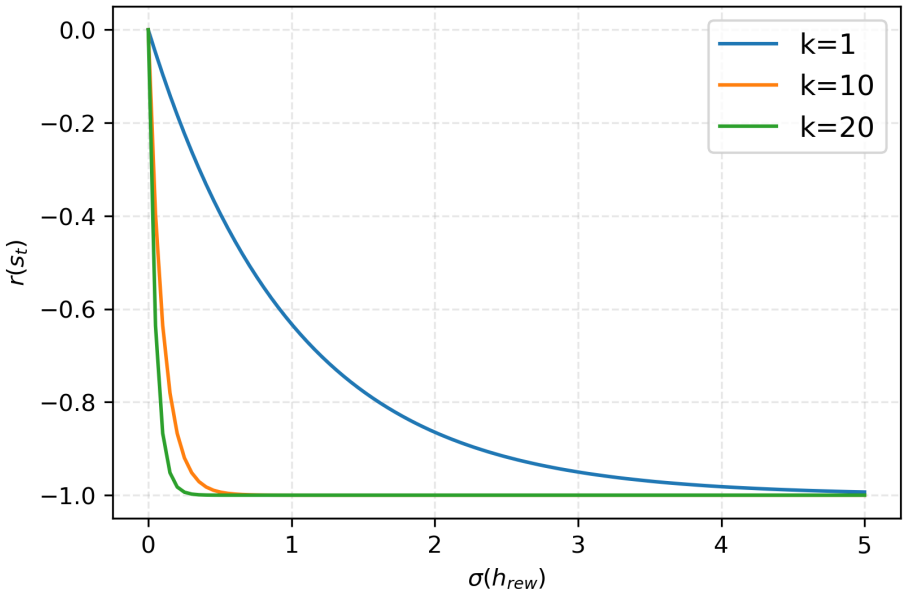

FIGURE 4.12: The reward function is approximately linear when σ has a value close to zero. Then it becomes increasingly more flat. This means that when σ is close to zero a change in σ will result in a bigger change in the reward

hyperbolic and Gaussian. The one that obtained the best results is the negative exponential:

$$r(s_t) = e^{-k\sigma(h_{rew})} - 1 \tag{4.30}$$

Therefore it was decided to adopt it as a reward function with $k = 50$.

If the surface becomes more wavy then the standard deviation grows and the reward tends towards negative one, if, on the other hand, the standard deviation is close to zero the controller will receive a higher score with a maximum of zero. A problem of this equation can be noticed by considering figure 4.12. When the standard deviation takes on bigger value the agent will barely notice a difference, this means that the reward is not particularly good at telling a bad action from a disastrous one. In practice the standard deviation at the exponent was multiplied with a constant, this might increase the problem just discussed but it results in bigger rewards. Having small rewards might be an issue, since for neural networks small gradients can lead to numerical instability during training. When the gradients become too small, the updates to the model parameters may not be effective, and the learning process can slow down.
Another idea to increase the reward would be to normalize it with reference values so that it remains bounded between 0 and 1. However for this to work it would be necessary to have the values of the extremes of the reward, for the best and worst case. The best case is know, since it would be a flat film, however the worst case is not so easily obtained.
Simply put, the reward function has to make the agent understand that it has to push on the peaks and take no action on the valleys, since it can in no way bring them up. The reward function proposed so far does not give many hints in this direction, so it was thought to work around it. Let us, for example, consider a term in the form

$$r(s_t) = (h - h_0)action \tag{4.31}$$

This is telling to the agent that if there is a peak in the film ($\Delta h = h - h_0 > 1$) then a bigger

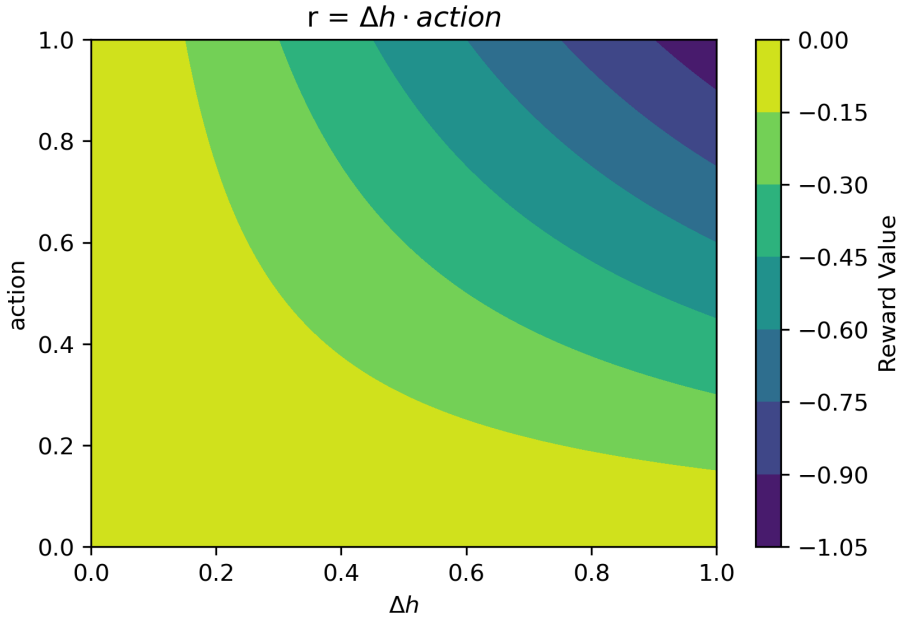

FIGURE 4.13: With a reward function that is linear with respect to the two variables strong actions are rewarded when not needed

action is desirable in order to maximize the reward. On the other hand, if a valley is considered, the term $h - h_0$ is negative and so to maximize the reward the action should be zero. So a simple idea would be to sum this term to the original value function 4.30. However there are still some issues with this function, as the agent is driven to always take actions with the maximum value, as long as the height difference term is positive. Plotting the reward as a function of the action and of the height difference, as in figure 4.13, it is clear that even with a not so tall peak, $(h - h_0) \approx 0^+$, the action that would yield the best reward would be a really strong pushing. A better solution would involve a function that, for minor peaks, promotes only a gentle push.

One function that satisfies this requirement could be something along the lines of

$$r(s_t) = log\left([(h - h_0) - action]^2 + 1\right) \tag{4.32}$$

As shown in figure 4.14 for small values of Δh it is best, in terms of reward, to select a moderate action. However also this reward function has its problems. Even though it is more informative as it is hinting to the agent which actions to take based on the physics of the problem, its objective it is no longer to strictly reduce the waviness of the film. In other words it is less aligned with the objective. Moreover, by summing this two terms, 4.30 and 4.31, part of the reward is based on the action that the agent is taking at a certain instant, while the other term is considering the waviness of the fluid caused by actions taken earlier. This could lead to sub-optimal learning.

4.10.2 Reward Area

Another important aspect of the reward is which area of the fluid to consider, that is defining the h_{rew} of equation 4.30. In the finite volumes this zone ended earlier, because the numerical dispersion was more pronounced and therefore it run into the risk of believing the actions were better than in reality because of the numerical smoothing. What happened was that the

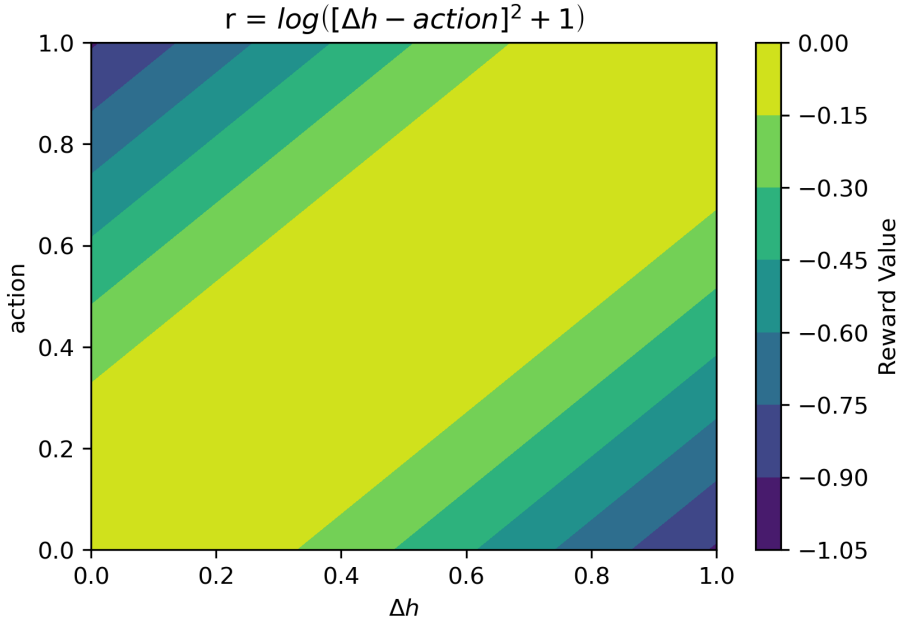

FIGURE 4.14: The best reward is not always given to the strongest actions

agent would take random actions and those would make the film more uneven, leading to higher numerical dissipation. With a spectral solver this does not occur, but having a smaller reward area also allows for quicker feedback. If it were really extended than the new incoming perturbation would only slightly alter the overall value. For this reason it was decided to place it close to the jets and sizing it in a way that would let the area encompass a single peak or valley. This does come at the cost of neglecting some long term effects, such as the one of surface tension which generally has a higher characteristic time. One final caveat is to place the area a bit downstream of the jets, otherwise they would affect the reward even before they have time to properly act on the perturbations.

4.11 Further considerations

One other idea that was considered for implementation was to clip the observation in a way that would make the agent only see the peaks but not the valleys. Since the jets can only blow and not suck there is little they can do to flatten a valley in the film. So if the agent considers all depression in the same way as it does with a flat surface it is possible that it will best learn how to act exclusively on the peaks. This is because it has a simplified view of the problem, however in the carried out tests this approach has not proven successful.
One other approach that was tested was the introduction of an observation log. Ideally the observations would be taken where the jets or magnets are, so that they can see the fluid that is directly beneath them. To work around this problem it is possible to collect the observation and store them in a log. Then the observations will be given to the agent in the time that the waves are estimated to travel the distance from the observation point to the jets position. For example, if from the last observation point to the jets the waves take two adimensional seconds, then two seconds worth of observation will be stored in a variable. Then at the following step an observation will be added to the storing variable and the first observation to be added will be given to the agent. One thing that has to be checked before running the simulations concerns the action that the control jets are allowed to take. Since the goal is to flatten the peaks it is

important to test that the most intense actions that the agent is allowed to take are actually strong enough to flatten the most pronounced peaks. Everything is ready, but before finally moving to the results, there is one last remark to be made. All the different aspects of the set up have non trivial interplay one with the other. Since simulations are a time consuming process it was sometimes the case that more than one adjustment were made from one iteration to the other. Following this approach might have led to results that were not properly interpreted, since not only one parameter was being isolated. However the complexity of the problem makes it so that it would have been really hard to actually understand the true influence of a single parameter, even as simple as the extent of the reward area. To properly do it would have called for an extensive parameter study, considering a multitude of set ups, requiring time and resources. In the end most of the paths followed were sustained by intuitive reasoning more than purely empirical data.

Chapter 5

BLEW Control Results

In this section, the results obtained by the reinforcement learning algorithms are presented. To begin with, a 2D jet will be considered, followed by the evaluation of a configuration involving four magnets. All the results of this section are compared to the *no control* case. Since the final goal is to have a film that is flatter than that in the absence of jets, it is necessary to estimate this value. Due to the randomness of the perturbation waves, introduced for robustness reasons as mentioned in section 4.7, it is not possible only to run a couple of episodes without jets to have no control value. For this reason, a larger number of simulations with only perturbation jets were carried out. In each instance, the mean reward per episode was computed, along with the corresponding standard deviation.

The learning is generally considered successful if it manages to reach at least the no-control value. Otherwise, it would be preferable to operate without control jets altogether. Before achieving successful simulations, numerous attempts failed. Some encountered errors, while others had configurations incompatible with effective learning. Even when signs of progress were evident, a few persistent issues arose. The first notable one was the emergence of a *passive* controller, wherein the agent learned that the optimal action was to take no action. This typically manifested as a learning curve plateauing at the no-control value. Another recurrent suboptimal solution was *extra wiping*. Essentially, the agent recognized that acting on the peaks was beneficial, but also did so in the valleys. This resulted in a uniform film that was thinner than the desired outcome.

5.1 2D Jets control

This case has been investigated by letting the agent choose its actions in two different ways. Initially, the agent was permitted to select the single jet action, represented by a value between -1 and 1, which was subsequently appropriately rescaled. For the second case, it was decided to restrain the action selection by imposing a harmonic control law. This could be done by asking the agent to return three values, the first of which would become the amplitude, A, the second the frequency, f, and the third phase, ϕ. These values were assembled into a periodic function, $g = Asin(2\pi ft + \phi)$. Each of these values had specific intervals from which they could be sampled, providing the agent with hints about the ideal control law. Just like in the example with the forced mass attached to a spring and a damper of subsection 4.6.1, it is likely that the long-term optimal control would be an action opposed to the forcing term. In this case, the forcing term is the one creating the perturbation, and so it was decided to let the agent select values of A, f and ϕ that are similar to the forcing jets. This second approach proved to be successful since it managed to give additional meaningful data to the agent, which resulted in a more stable and faster learning process. In figure 5.1, the harmonic and non-harmonic

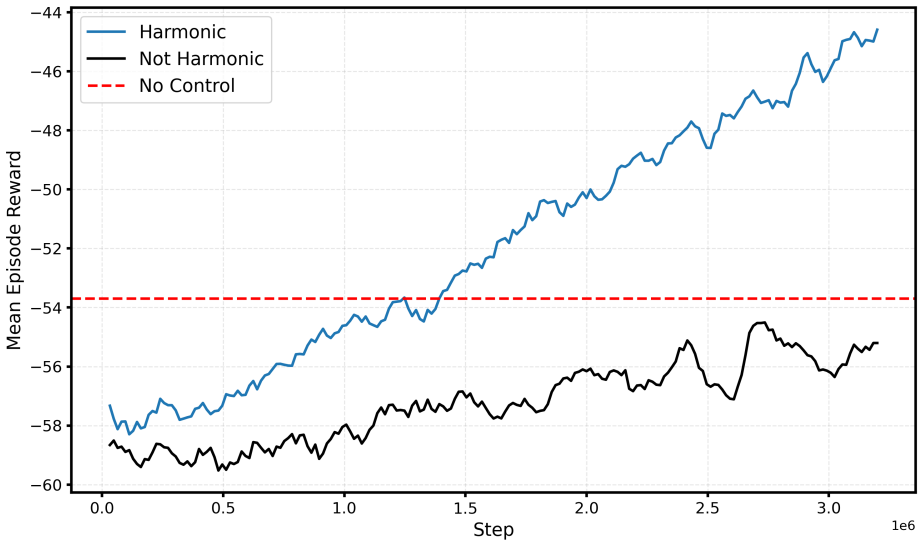

FIGURE 5.1: Learning curve comparison for the harmonic and non-harmonic control actions

training are compared. The two setups were identical as part of the action implementation, and so also, the no control value for the two is the same.

In the end, both simulations were successful, and the not-harmonic one managed to surpass the no-control value. This is because after the learning has finished, ten evaluation episodes with the best control law were carried out. Both managed to flatten the peaks, as shown in figure 5.2. Moreover, the learning process was probably not completely done since usually, the learning curve reaches a horizontal asymptote as the agent does not explore anymore but just takes the best actions,

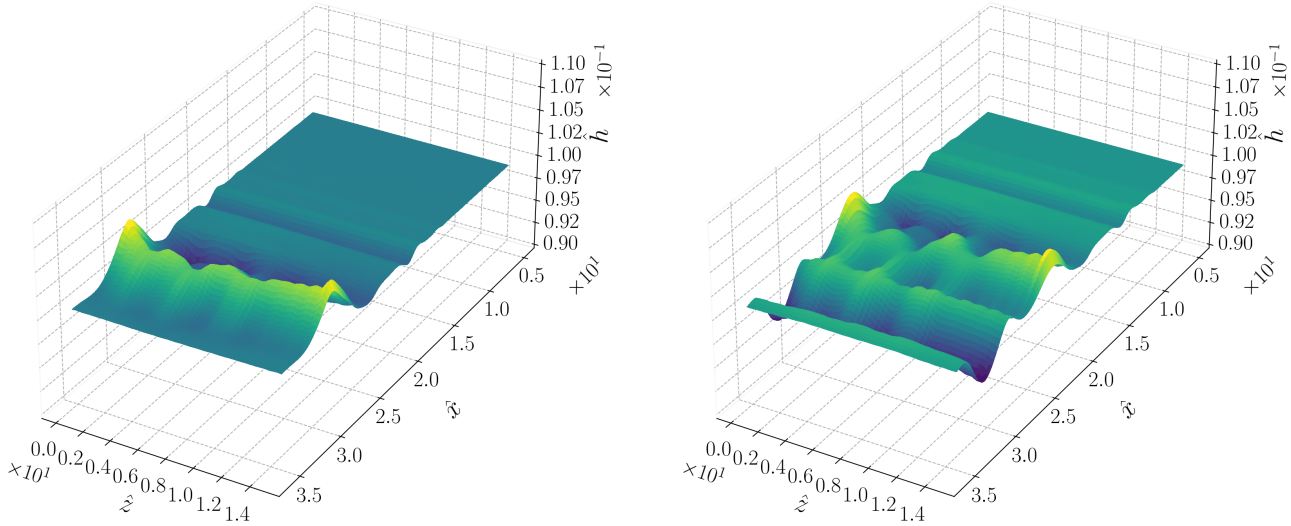

FIGURE 5.2: 3D plot of the non-dimensional liquid film. The 2D jet with harmonic actions manages to flat the wave crest located at $\hat{x} \approx 2.5$ in the left figure

An additional aspect to consider is the relationship between observations and subsequent actions. Figure 5.3 illustrates that when the observation has a peak, the amplitude exhibits a corresponding peak, aligning with expectations. This behaviour indicates a strategy of exerting more force on the peaks and less on the valleys. Notably, when a peak is observed, there is a time delay before the agent reacts. This delay most likely stems from the relative position of the jets and observations, where the height is measured before the jet line, making it crucial for the agent to wait before initiating a reaction.

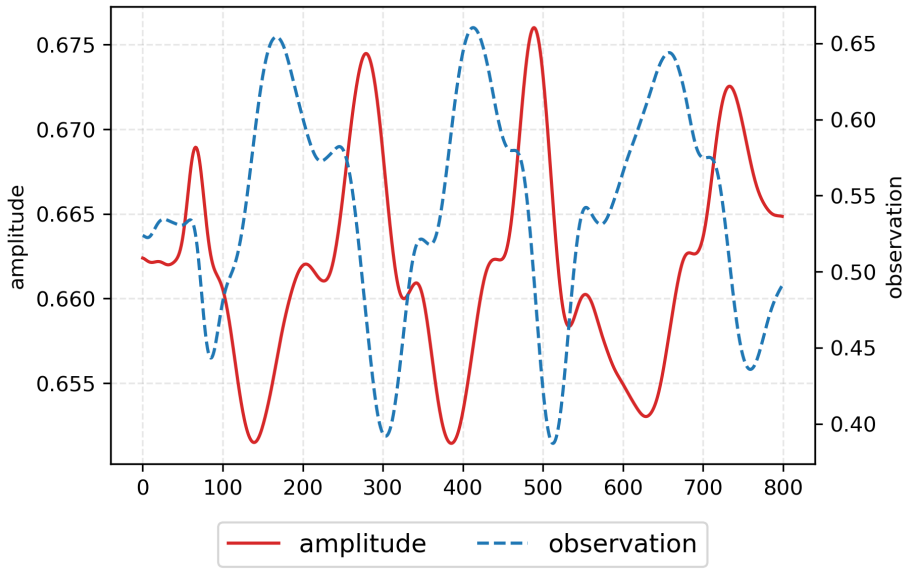

FIGURE 5.3: The normalized amplitudes (which determine the action) have a strong correlation with the observation for the whole length of the episode, as the x axis represents time step index

5.2 3D Magnets control

For this second study case, it was decided to use actuators that would only act on specific points and not all along the $\hat{z}$ axis. Even though 2D jets and magnets are technically feasible, it is also interesting to see how a 3D agent would behave. For instance, breaking the wave into multiple horizontal segments causes the rate of change to increase along $\hat{z}$ and that, in turn, might lead to surface tension or even viscosity to have a stronger flattening effect.

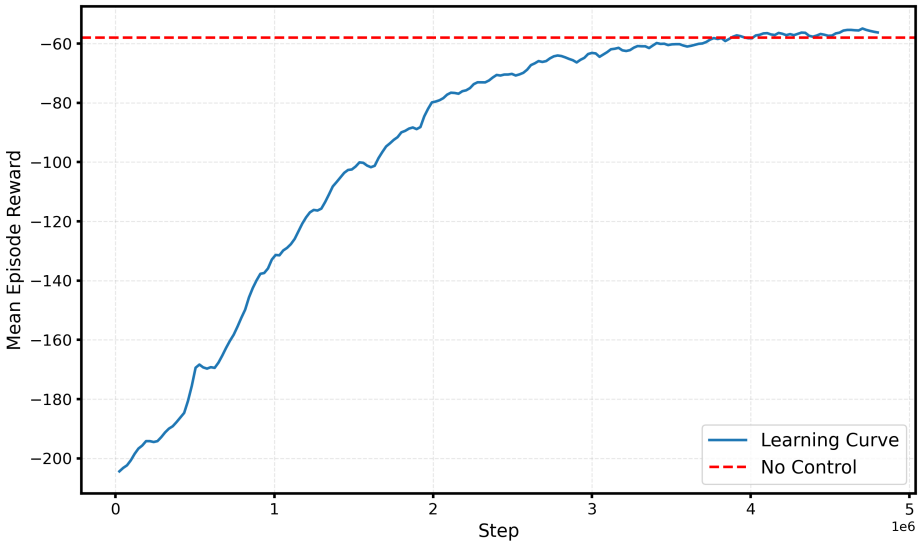

FIGURE 5.4: Learning curve plateaus close to the no control value, in a way similar to the passive agents

Working with the magnets also involves a different interaction between the liquid and the controllers. For starters, the force applies to the bulk of the fluid and not only its surface. Moreover, the force does not only push the fluid down like the gas jets. The good news is that once the differences are accounted for in the solver, the machine learning set-up does not change much, and the observations, actions and rewards can remain the same. Figure 5.4 shows the final learning curve, whose final agent managed to reach an episode mean reward of -41.5, improving more than 25% the no control value.

Figure 5.5 illustrates the different effects the magnets can exert by pulling up the valleys. Also for this reason, it would be interesting to combine magnets and jets, since they might have a complementary effect.

Finally the strategy found by the agent is no longer in phase with the observations as in the 2D jet with harmonic actions. As a matter of fact the agent actions shown in figure 5.6 seems to have a different strategy, that is acting mainly on the valleys, while turning down as the peaks approach.

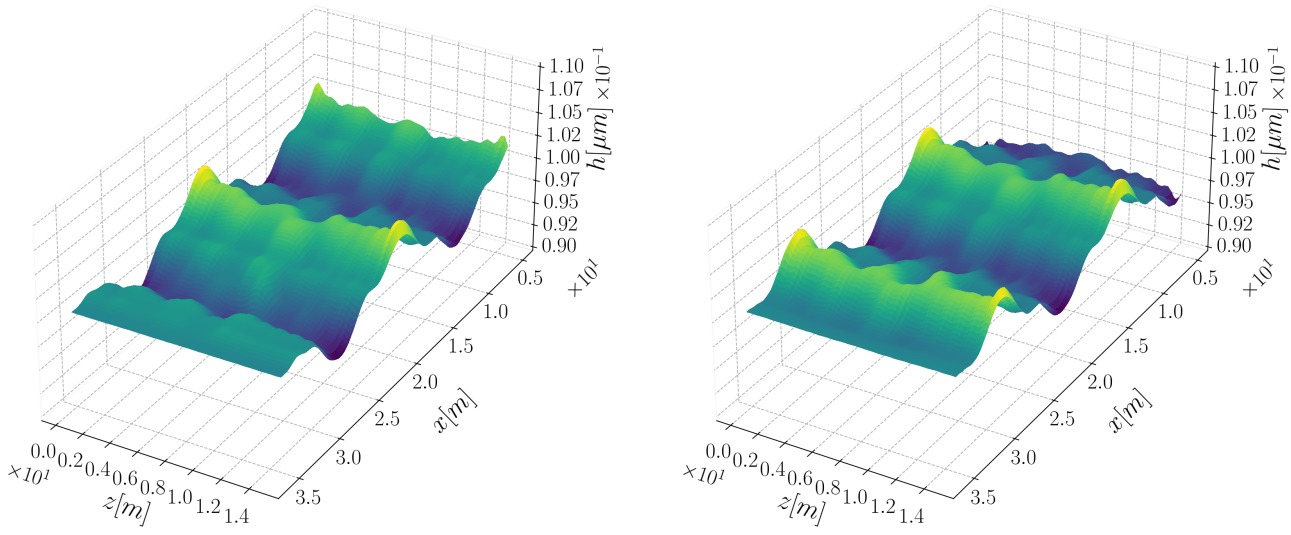

FIGURE 5.5: 3D plot of the non-dimensional liquid film. The four magnets manage to pull up the wave valley located at $\hat{x} \approx 2.5$ in the left figure, which is then found in $\hat{x} \approx 2$ in the right one

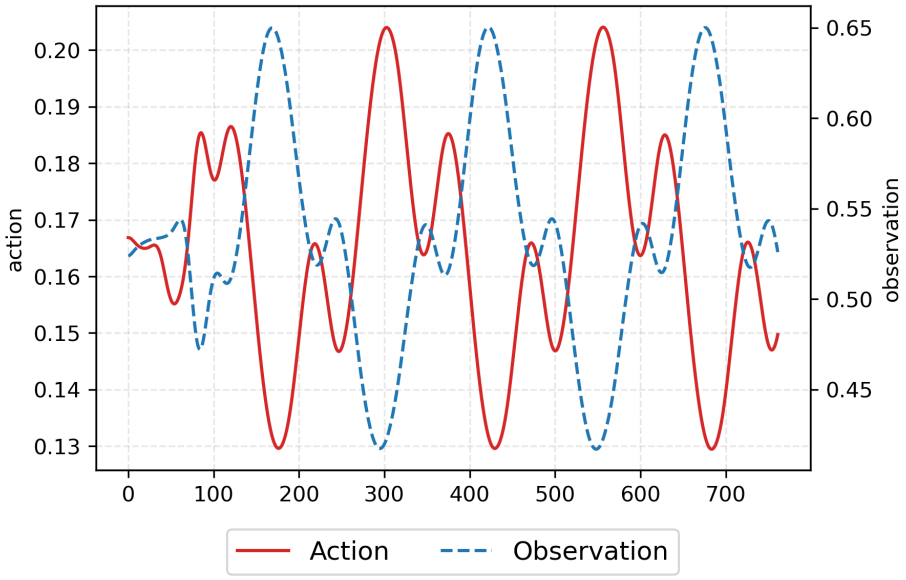

FIGURE 5.6: In this case, the approach seems to strongly differ, with normalized actions that are maximum in correspondence to the valley and turned down for the peaks

Chapter 6

Conclusion and Prospects

All things considered the control laws developed in this work proved to be successful and in line with the main objective. The control law makes the film significantly more homogeneous and does it by acting differently on the valley and on the peaks, in a way that depends on the chosen actuator. The final simulations have been run on less than a couple of days and still have room for improvement. Nonetheless if the control were to be implemented in the the actual factory lines with similar results they would reduce the used material while also guaranteeing a better superficial finishing at the expense of powering the jets or magnets. Now that a benchmark has been found that shows to which extent it is indeed possible to flatten the perturbations with the current spectral solver, it would be interesting to explore more in depth how different configurations of jets or magnets of observations would perform. For instance by finding the minimum number of observation points in order to reduce the sensors cost or by trying to minimize the actuators strength with the idea of saving resources and power. Another idea would be to place multiple lines of jets so that if the first one fails, it is still possible for the second one to properly flatten the film. It would also be interesting to combine the jets and the magnets together, since it has been shown that they have intrinsically different effects on the flow, the most evident of which is the ability of the magnets of pulling up valleys. Therefore, placing a series of jets followed by a line of magnets could potentially yield results not reachable by jets or magnets alone.
The possible developments are not limited to the reinforcement learning, efforts could be made also in the direction of expanding the model, by dropping some of the assumptions taken in chapter 1. For instance one direction would be not to consider the surface tension of the liquid zinc constant, as it has been shown to vary significantly close to the plate borders. Further investigations could also be made with the aim of making the time integration process faster and more stable by improving on the explicit Euler scheme. Additionally, in the present work only the PPO algorithm has been evaluated, however a vast number of RL methods exists and to investigate them only minimal adjustments would be required. Last but not least it would be worthwhile validating the solver with experimental data.

Appendix A

Unsteady electromagnetic model 3D

The full set of equations solved by the complete magnetic model are presented here

$$\partial_{\hat{t}}\hat{h} + \partial_{\hat{x}}\hat{q}_{\hat{x}} + \partial_{\hat{z}}\hat{q}_{\hat{z}} = 0 \tag{A.1}$$

$$\begin{aligned}\partial_{\hat{t}}\hat{q}_{\hat{x}} + \partial_{\hat{x}}\int_0^{\hat{h}} \hat{u}^2\, d\hat{y} \\ + \partial_{\hat{z}}\int_0^{\hat{h}} \hat{u}\hat{w}\, d\hat{y} = \delta^{-1}\Big(- \hat{h}\partial_{\hat{x}}\hat{p}_g + \hat{h}(\partial_{\hat{x}\hat{x}\hat{x}}\hat{h} + \partial_{\hat{x}\hat{z}\hat{z}}\hat{h}) \\ + \hat{\tau}_{g,x} + \hat{\tau}_{w,x} + \hat{h} - H_a^2\hat{B}^2(\hat{x},\hat{z})\hat{q}_{\hat{x}}\Big)\end{aligned} \tag{A.2}$$

$$\begin{aligned}\partial_{\hat{t}}\hat{q}_{\hat{z}} + \partial_{\hat{x}}\int_0^{\hat{h}} \hat{u}\hat{w}\, d\hat{y} \\ + \partial_{\hat{z}}\int_0^{\hat{h}} \hat{w}^2\, d\hat{y} = \delta^{-1}\Big(- \hat{h}\partial_{\hat{z}}\hat{p}_g + \hat{h}(\partial_{\hat{z}\hat{z}\hat{z}}\hat{h} + \partial_{\hat{z}\hat{x}\hat{x}}\hat{h}) \\ + \hat{\tau}_{g,z} + \hat{\tau}_{w,z} - H_a^2\hat{B}^2(\hat{x},\hat{z})\hat{q}_{\hat{z}}\Big)\end{aligned} \tag{A.3}$$

The velocity profiles can be expressed as

$$\hat{u}(\hat{y}) = k_{1,u}e^{H_{a_0}\hat{B}\hat{y}} + k_{2,u}e^{-H_{a_0}\hat{B}\hat{y}} + k_{3,u}\frac{A_1+1}{H_{a_0}^2\hat{H}^2} \quad with A_1 = -\partial_{\hat{x}}\hat{p}_g + \frac{\epsilon^3}{Ca}\partial_{\hat{x}\hat{x}\hat{x}}\hat{h}, \tag{A.4}$$

$$\hat{w}(\hat{y}) = k_{1,z}e^{H_{a_0}\hat{B}\hat{y}} + k_{2,z}e^{-H_{a_0}\hat{B}\hat{y}} + k_{3,z}\frac{A_1+1}{H_{a_0}^2\hat{H}^2} \quad with A_1 = -\partial_{\hat{x}}\hat{p}_g + \frac{\epsilon^3}{Ca}\partial_{\hat{x}\hat{x}\hat{x}}\hat{h}, \tag{A.5}$$

While the three conditions for the streamwise velocity profile along $\hat{x}$ are:

$$\hat{u}(0) = -1 \qquad \partial_{\hat{y}}\hat{u}(\hat{h}) = \hat{\tau}_{g,x} \qquad \hat{q}_{\hat{x}} = \int_0^{\hat{h}} \hat{u}(\hat{y})\, d\hat{y}. \tag{A.6}$$

and similarly for the spanwise velocity they can be written as:

$$\hat{u}(0) = 0 \qquad \partial_{\hat{y}}\hat{u}(\hat{h}) = \hat{\tau}_{g,z} \qquad \hat{q}_{\hat{z}} = \int_0^{\hat{h}} \hat{w}(\hat{y})\, d\hat{y}. \tag{A.7}$$

Now some auxiliary variables are introduced

$$\alpha_1 = \frac{1}{Ha^2\hat{B}^2} \quad \alpha_2 = Ha\hat{B} \quad \alpha_3 = e^{Ha\hat{B}\hat{h}} \quad \alpha_4 = e^{-Ha\hat{B}\hat{h}} \tag{A.8}$$

$$\alpha_5 = \hat{\tau}_{\hat{x}} \quad \alpha_6 = \hat{h} \quad \alpha_7 = \hat{q}_{\hat{x}} \quad \alpha_8 = \hat{q}_{\hat{z}} \quad \alpha_9 = \hat{\tau}_{\hat{z}} \tag{A.9}$$

Additional variables for the streamwise direction are

$$k_{1,u} = \frac{\alpha_2^2\alpha_4(-\alpha_7-\alpha_6)+\alpha_5(\alpha_2\alpha_6+\alpha_4-1)}{\alpha_4(\alpha_2^2\alpha_6+\alpha_2)+\alpha_3(\alpha_2^2\alpha_6-\alpha_2)}, \tag{A.10a}$$

$$k_{2,u} = \frac{\alpha_2^2\alpha_3(-\alpha_7-\alpha_6)+\alpha_5(-(\alpha_2\alpha_6)+\alpha_3-1)}{\alpha_4(\alpha_2^2\alpha_6+\alpha_2)+\alpha_3(\alpha_2^2\alpha_6-\alpha_2)}, \tag{A.10b}$$

$$k_{3,u} = -\frac{\alpha_4(\alpha_2-\alpha_2^2\alpha_7)+\alpha_3(-(\alpha_2^2\alpha_7)-\alpha_2)+(\alpha_4+\alpha_3-2)\alpha_5}{\alpha_1(\alpha_4(\alpha_2^2\alpha_6+\alpha_2)+\alpha_3(\alpha_2^2\alpha_6-\alpha_2))}, \tag{A.10c}$$

Whereas for the spanwise direction:

$$k_{1,w} = \frac{(\alpha_2^2\alpha_6+\alpha 4-1)\alpha_9-\alpha_2^2\alpha_4\alpha_8}{\alpha_4(\alpha_2^2\alpha_6+\alpha_2)+\alpha_3(\alpha_2^2\alpha_6-\alpha_2)}, \tag{A.11a}$$

$$k_{2,w} = \frac{(-(\alpha_2\alpha_6)+\alpha_3-1)\alpha_9-\alpha_2^2\alpha_3\alpha_8}{\alpha_4(\alpha_2^2\alpha_6+\alpha_2)+\alpha_3(\alpha_2^2\alpha_6-\alpha_2)}, \tag{A.11b}$$

$$k_{3,w} = -\frac{(\alpha_4+\alpha_3-2)\alpha_9-\alpha_2^2\alpha_4\alpha_8-\alpha_2^2\alpha_3\alpha_8}{\alpha_1(\alpha_4(\alpha_2^2\alpha_6+\alpha_2)+\alpha_3(\alpha_2^2\alpha_6-\alpha_2))}, \tag{A.11c}$$

The closure terms are given by:

$$\hat{\tau}_{w,x} = -\partial_{\hat{y}}\hat{u}|_{\hat{y}=0} = -k_{1,u}H_a\hat{B} + k_{2,u}H_a\hat{B} \tag{A.12a}$$

$$\hat{\tau}_{w,z} = -\partial_{\hat{y}}\hat{w}|_{\hat{y}=0} = -k_{1,w}H_a\hat{B} + k_{2,w}H_a\hat{B} \tag{A.12b}$$

Combining everything it is possible to write the flux terms as:

$$\begin{aligned}\int_0^{\hat{h}} \hat{u}^2\, d\hat{y} = \Big(& e^{-2\alpha_2\alpha_6}(2\alpha_1^2\alpha_2\alpha_6 e^{2\alpha_2\alpha_6}k_{3,u}^2 + ((4\alpha_1 e^{2\alpha_2\alpha_6} - 4\alpha_1 e^{\alpha_2\alpha_6})k_{2,u} + \\ & + (4\alpha_1 e^{3\alpha_2\alpha_6} - 4\alpha_1 e^{2\alpha_2\alpha_6})k_{1,u})k_{3,u} + (e^{2\alpha_2\alpha_6}-1)k_{2,u}^2 + \\ & + 4\alpha_2\alpha_6 e^{2\alpha_2\alpha_6}k_{1,u}k_{2,u} + (e^{4\alpha_2\alpha_6} - e^{2\alpha_2\alpha_6})k_{1,u}^2)\Big)/(2\alpha_2)\end{aligned} \tag{A.13a}$$

$$\begin{aligned}\int_0^{\hat{h}} \hat{u}\hat{w}\, d\hat{y} = \Big(& e^{-2\alpha_2\alpha_6}((2\alpha_1^2\alpha_2\alpha_6 e^{2\alpha_2\alpha_6}k_{3,u} + (2\alpha_1 e^{2\alpha_2\alpha_6} - 2\alpha_1 e^{\alpha_2\alpha_6})k_{2,u} + \\ & + (2\alpha_1 e^{3\alpha_2\alpha_6} - 2\alpha_1 e^{2\alpha_2\alpha_6})k_{1,u})k_{3,w} + ((2\alpha_1 e^{2\alpha_2\alpha_6} - 2\alpha_1 e^{\alpha_2\alpha_6})k_{2,w} + \\ & + (2\alpha_1 e^{3\alpha_2\alpha_6} - 2\alpha_1 e^{2\alpha_2\alpha_6})k_{1,w})k_{3,u} + ((e^{2\alpha_2\alpha_6}-1)k_{2,u} + \\ & + 2\alpha_2\alpha_6 e^{2\alpha_2\alpha_6}k_{1,u})k_{2,w} + 2\alpha_2\alpha_6 e^{2\alpha_2\alpha_6}k_{1,w}k_{2,u} + \\ & + (e^{4\alpha_2\alpha_6} - e^{2\alpha_2\alpha_6})k_{1,u}k_{1,w})\Big)/(2\alpha_2)\end{aligned} \tag{A.13b}$$

$$\int_0^{\hat{h}} \hat{w}^2 \, d\hat{y} = \Big(e^{-2\alpha_2\alpha_6}(2\alpha_1^2\alpha_2\alpha_6 e^{2\alpha_2\alpha_6} k_{3,w}^2 + ((4\alpha_1 e^{2\alpha_2\alpha_6} - 4\alpha_1 e^{\alpha_2\alpha_6})k_{2,w} + \\ + (4\alpha_1 e^{3\alpha_2\alpha_6} - 4\alpha_1 e^{2\alpha_2\alpha_6})k_{1,w})k_{3,w} + (e^{2\alpha_2\alpha_6} - 1)k_{2,w}^2 + \\ + 4\alpha_2\alpha_6 e^{2\alpha_2\alpha_6} k_{1,w}k_{2,w} + (e^{4\alpha_2\alpha_6} - e^{2\alpha_2\alpha_6})k_{1,w}^2) \Big) / (2\alpha_2) \tag{A.13c}$$

Bibliography

[1] David Barreiro-Villaverde et al. "Damping of three-dimensional waves on coating films dragged by moving substrates". In: *Physics of Fluids* 35 (June 2023). URL: arXiv:2305.16139v3.

[2] Spyridon Beltaos and Nallamuthu Rajaratnam. "Impinging circular turbulent jets". In: *Journal of the hydraulics division* 100.10 (1974).

[3] Giorgio Bertotti and Isaak D Mayergoyz. *The Science of Hysteresis: 3-volume set*. Elsevier, 2005.

[4] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[5] Claudio Canuto et al. *Spectral methods: evolution to complex geometries and applications to fluid dynamics*. Springer Science & Business Media, 2007.

[6] R. Joseph Davis. *Corrosion: Understanding the Basics*. 2000, pp. 1–20.

[7] Denys Dutykh. "A brief introduction to pseudo-spectral methods: application to diffusion problems". In: *arXiv preprint arXiv:1606.05432* (2016).

[8] Silviu Filip, Aurya Javeed, and Lloyd N Trefethen. "Smooth random functions, random ODEs, and Gaussian processes". In: *SIAM Review* 61.1 (2019), pp. 185–205.

[9] Anne Gosset and Jean-Marie Buchlin. "Jet Wiping in Hot-DipGalvanization". In: *Journal of Fluids Engineering* 129 (2007), pp. 466–475. DOI: 10.1115/1.2436585.

[10] IEA. *Iron and Steel Technology Roadmap*. 2020, pp. 50–60. URL: https://www.iea.org/reports/iron-and-steel-technology-roadmap.

[11] "Intro to Policy Optimization". In: (). URL: https://spinningup.openai.com.

[12] Tsvetelina Ivanova et al. "Evolution of waves in liquid films on moving substrates". In: *Under consideration for publication in J. Fluid Mech.* (Jan. 2023). URL: arXiv:2004.13400v2.

[13] Steven G. Johnson. *Notes on Perfectly Matched Layers (PMLs)*. Tech. rep. MIT, 2008.

[14] Gerhardus Koch. "Cost of corrosion". In: Dec. 2017, pp. 3–30. ISBN: 9780081011058. DOI: 10.1016/B978-0-08-101105-8.00001-2.

[15] M.A. Mendez, A. Gosset, and J.-M. Buchlin. "Integral Modeling of the Jet Wiping Process: Part 1, Introduction to the Self-Similar Formulation". In: *Technical Note of the Von Karman Institute* 216a (May 2018). URL: DOI:10.13140/RG.2.2.32798.23363.

[16] M.A. Mendez et al. "Dynamics of the Jet Wiping Process via Integral Models". In: *Journal of Fluid Mechanics* 911 (Nov. 2020). URL: arXiv:2004.13400v2.

[17] Desmet Mitja. *Reinforcement Learning for active flow control*. 2020.

[18] Kaare B. Petersen and Michael S. Pedersen. *The Matrix Cookbook*. 2008, p. 54. URL: http://matrixcookbook.com.

[19] Fabio Pino et al. “Comparative analysis of machine learning methods for active flow control”. In: *Journal of Fluid Mechanics* 958 (Mar. 2023).

[20] Piotr M Przybyłowicz and Tomasz Szmidt. “Electromagnetic damping of a mechanical harmonic oscillator with the effect of magnetic hysteresis”. In: *Journal of theoretical and applied mechanics* 47.2 (2009), pp. 259–273.

[21] Krishnan Radhakrishnan and Alan C. Hindmarsh. *Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations*. Lawrence Livermore National Laboratory, 1993, pp. 66–69. URL: https://www.iea.org/reports/iron-and-steel-technology-roadmap.

[22] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[23] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: (Aug. 2017). URL: arXiv:1707.06347.

[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction. Second*. Vol. 2. 2018, pp. 25–130. URL: http://incompleteideas.net/book/the-book-2nd.html.

[25] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. SIAM, 2000. DOI: 10.1137/1.9780898719598.