TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen, Institut für Strömungsmechanik, Professur für Strömungsmechanik

Diploma Thesis

Coupling of an artificial neural network with LES-LBM to improve wind farm control

submitted in partial fulfillment of the requirements for the degree "Diplomingenieur"

Henry Torsten Korb

born on 02.03.1996

in Ratingen

submitted on 13. 09. 2020

1st reviewer Prof. Dr.-Ing. habil. J. Fröhlich

2nd reviewer PD Dr.-Ing. habil. J. Stiller

Supervisor M. Sc. H. Asmuth

Co-Supervisor M. Sc. R. Jain

Kurzfassung

Kopplung eines künstlichen neuronalen Netzwerks mit LES-LBM zur Verbesserung einer Windpark-Steuerung

15 zeilen

Abstract

Coupling of an artificial neural network with LES-LBM to improve wind farm control

15 lines

Contents

Nomenclature

Latin Symbols	Unit	Description
a	$-$	Axial interference / induction factor
a'	$-$	Tangential interference / induction factor
$\mathbf{a}$	$-$	Action vector
$\hat{\mathbf{a}}$	$-$	Applied action vector
A	m^2	Area
$\mathbf{A}$	$-$	Action vector
b	$-$	Weights
$\mathbf{b}$	$-$	Vector of biases
$\mathbf{c}$	$\mathrm{m/s}$	Constant velocity vector
c	$\mathrm{m/s}$	Constant velocity
$\mathbf{c}$	$-$	Cell state vector
C	$-$	Coefficient
$C_{\alpha,\beta,\gamma}$	$-$	Cumulant
D	m	Rotor diameter
$\mathbf{e}$	m	Basis vector
E	$\mathrm{J/m}^3$	Energy density
E	$-$	Expected value
f	$\mathrm{kgs}^3/\mathrm{m}^6$	Distribution function
f	$-$	Activation function
$\mathbf{F}$	N	Force
F	$-$	Distribution function in frequency space
G	$-$	Return

I	kgm^2	Moment of inertia
J	–	Performance measure / Objective
k	–	Central moment
K	–	Number of layers
L	m	Length
M_{aero}	Nm	Aerodynamic torque
M_{gen}	Nm	Generator torque
$M_{\alpha,\beta,\gamma}$	–	Raw moment
M	–	Number of nodes in layer
Ma	–	Mach number
$\mathbf{n}$	–	Normal vector
N_b	–	Number of blades
$N_{A,e}$	–	Number of actions per episode
$N_{e,b}$	–	Number of episodes per batch
$N_{t,A}$	–	Number of timesteps per action
N	–	Number of inputs into layer
p	Pa	Pressure
p	–	State transition function
pdf	–	Probability density function
pr	–	Probability ratio
P	W	Power
Pr	–	Probability
q	–	Action-value function
r	m	Radius / radial coordinate
r_f	–	Forget ratio

r	$-$	Reward
r_p	$-$	Probability ratio
R	m	Rotor radius
$\hat{R}$	$\mathrm{J/kg/K}$	Specific gas constant
R	$-$	Reward
s	$-$	Solidity
$\mathbf{S}$	$-$	State vector
t	s	Time
t	$-$	Sensitivity
T	N	Thrust force
$\hat{T}$	K	Temperature
T	$-$	Length of Episode
$\mathbf{u}$	$\mathrm{m/s}$	Macroscopic velocity vector
u	$\mathrm{m/s}$	Streamwise velocity
$\mathbf{v}$	$\mathrm{m/s}$	Macroscopic velocity vector
v	$\mathrm{m/s}$	Wallnormal velocity
v	$-$	Value function
$\mathbf{V}$	$\mathrm{m/s}$	Wind Velocity
V	m^3	Volume
V_0	$\mathrm{m/s}$	Mean wind speed
w	$\mathrm{m/s}$	Streamwise velocity
W	$-$	Weight matrix of layer
$\mathbf{x}$	m	Vector of position
x	m	Streamwise coordinate
$\mathbf{x}$	$-$	Activation
y	m	Wallnormal coordinate

$\mathbf{y}$	$-$	Layer output
z	m	Spanwise coordinate
$\mathbf{z}$	$-$	Layer input

Greek Symbols	Unit	Description
α	rad	Angle of attack
α_f	$-$	Decay rate of exponential filter
β	$-$	Probability clipping ratio
γ	$-$	Discount rate
δ	$-$	Delta distribution
ϵ	m	Smearing width
ε_{p,l_2}	$-$	Policy l_2 regularization coefficient
$\varepsilon_{v,l}$	$-$	Policy l_2 regularization coefficient
ε_{v,l_2}	$-$	Value loss coefficient
ζ	$\mathrm{m/s}$	Microscopic velocity
Z	$-$	Wavenumber
η	$-$	Gaussian filter kernel
$\boldsymbol{\theta}$	$-$	Parameter vector
θ	$-$	Angle
Θ	$-$	Azimuth angle
κ	$\mathrm{Nms^2}$	Greedy controller proportionality constant
λ	$-$	Tip-speed ratio
μ	$-$	Probability parameter
ν	$\mathrm{m^2/s}$	Kinematic viscosity

$\boldsymbol{\xi}$	$\mathrm{m/s}$	Microscopic velocity vector
ξ	$\mathrm{m/s}$	Microscopic velocity
$\boldsymbol{\Xi}$	$-$	Wavenumber vector
Ξ	$-$	Wavenumber
π	$-$	Policy
ρ	$\mathrm{kg/m^3}$	Density
σ	$-$	Sigmoid function
υ	$\mathrm{m/s}$	Microscopic velocity
Υ	$-$	Wavenumber
ϕ	rad	Angle
φ	$-$	General control variable
Φ	rad	Pitch angle
ψ	$-$	Learning rate
Ψ	rad	Yaw angle
ω	$\mathrm{rad/s}$	Angular velocity
$\hat{\omega}$	$1/\mathrm{s}$	Relaxation frequency
Ω	$\mathrm{kgs^2/m^6}$	Collision operator

Indices	Description
D	Drag
i	running index
I	Induced
j	running index
k	running index
l	running index
L	Lift

m	Mean
max	Maximum
n	Normal
P	Power
r	Radial
s	Sound
t	tangential
T	Thrust
x	Streamwise
θ	Azimuthal

Additional Symbols	Description
$\|\|(.)\|\|$	Euclidian norm
∇	Nabla operator
$\delta_{i,j}$	Kronecker delta
Δ	Step
$\mathcal{O}(.)$	Order
$(.) \cdot (.)$	Inner product
$(.)'$	Fluctuation
$[.]^T$	Transposed vector

Abbreviations	Description
ABL	Atmospheric boundary layer
ALM	Actuator line model
ANN	Artificial Neural Network
BEM	Blade element / momentum theory

BGK	Bhatnagar-Gross-Krook
CFD	Computational Fluid Dynamics
DNS	Direct numerical simulation
DRL	Deep Reinforcement Learning
GPU	Graphical Processing Unit
HAWT	Horizontal axis wind turbine
IEA	International Energy Agency
LBM	Lattice Boltzmann Method
LBE	Lattice Boltzmann Equation
LES	Large-Eddy Simulation
LSTM	Long short-term memory
MDP	Markov Decision Process
MRT	Multiple Relaxation Times Operator
NREL	National Renewable Energy Laboratory
NSE	Navier-Stokes-Equations
pdf	Particle-distribution Function
PPO	Proximal policy optimization
RL	Reinforcement Learning
SGD	Stochastic Gradient Descent

1. Introduction

While human made climate change only begins to effect the countries of Europe and North America, the predictions for the near and far future are devastating [23]. One of the key causes for global warming is increase in atmospheric greenhouse gases like carbondioxide and methane. Electricity production accounts up to a third of greenhouse gas emissions in major industrial countries like the United States [22] and Germany [37]. Therefore, the energy sector is shifting to renewable sources of energy such as wind energy [25]

The momentum of the wind is used to power wind turbines, which generate electricity. To reduce investment and maintenance costs, wind turbines are often arranged in wind farms of multiple, sometimes hundreds of turbines. However, this also reduces the overall efficiency of the turbines due to wake losses. These losses can be mitigated by various means. Placing turbines further apart reduces the wake interaction but also some of the advantages of wind farms. Furthermore, this can not be changed for farms already built. A different approach is to change the controller of turbines in order to reduce wake interaction. This has the advantage that it is not only applicable to newly built parks, but can also be implemented at already existing sites.

One possible way to reduce wake interaction is by curtailing the upstream turbines. The

2. Theoretical Background

2.1. Wind Turbines

2.1.1. Description of a Turbine and Flow Features

First a few basic definitions of the model used for wind turbines will be given. Todays mainstream wind turbines consist of a rotor with three blades, with a horizontal axis, therefore they are called horizontal axis wind turbines (HAWT), and the rotor points upwind. This rotor is mounted to the nacelle, which holds the gearbox and the generator. The nacelle sits on top of the tower. The diameter of the rotor is D. A schematic of a HAWT is given in Figure 2.1. It shows the wind velocity $\mathbf{V}$, the azimuthal angle θ, angular velocity of the rotor ω, the pitch angle of one of the blades Φ_0, the yaw angle Ψ and the torque due to aerodynamic forces and the generator, M_{aero} and M_{gen}, respectively. Due to the nature of the atmospheric boundary layer (ABL) the wind speed is higher at larger hub heights, and a turbine with a larger diameter can produce more power since the availabe power P_{avail} is proportional to the area A swept by the rotor. This leads to a steady increase in turbine sizes and nominal capacity [41]. In the academic world theoretical reference turbines such as the NREL 5 MW wind turbine [27] or the recently proposed IEA Wind 15-Megawatt Offshore Reference Wind Turbine [11] are used, since data from real turbines usually is not publicly available.[20]
The flow regime a wind turbine is subjected to is the ABL, which features wind velocities of the order of $\mathcal{O}(1\,\mathrm{m/s})$ to $\mathcal{O}(10\,\mathrm{m/s})$. Furthermore the flow is turbulent and sheared. For further information on the ABL and the challenges in modelling it, the reader is referred to [28] and [24]. The turbine induces a wake, characterized by an increase of turbulence intensity and a decrease of average velocity by 50 percent and more [2]. More details on wakes can be found in [6].

2.1.2. Wind Turbine Control

To generate power from the aerodynamic torque acting on the rotor a generator applies a countering torque. Furthermore, the turbine is equipped with different actuators, which can be used to manipulate the behaviour in order to achieve certain goals. The operating conditions of wind turbines can be classified into three regions, depending on the wind speed. In region I, below the cut-in speed, no power is generated and the wind is used to speed up the rotor. In between cut-in and rated speed lays region II, where the goal is to maximize the generated power. At rated speed, the turbine generates the maximum power. Above rated speed, in region III, the control is focussed on the quality of the generated electricity and to minimize loads on the turbine. An example of a detailed control curve can be found in [27]. [6]

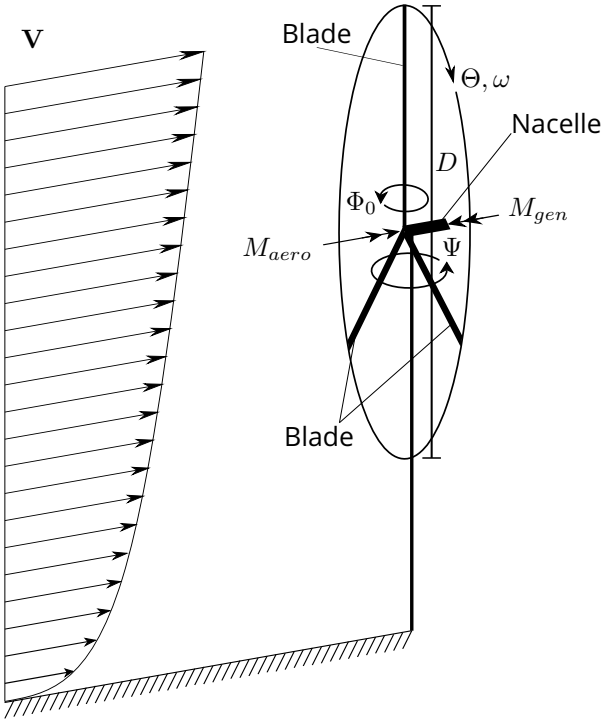

Figure 2.1.: Schematic of a HAWT and relevant angles, lengths, velocities and moments.

The controllable variables are Ψ, Φ and M_{gen}. To maximize the power, the yaw has to be adjusted so the turbine points upwind. The pitch and generator torque have to be adjusted so that M_{gen} and ω are maximized, since the generated power P_{gen} is:

$$P_{gen} = \omega M_{gen}. \tag{2.1}$$

The blades are designed so that the optimal blade pitch is zero. At a given wind speed and constant blade pitch, it can be shown that $M_{aero} \propto \omega^2$. Applying the law of conservation of angular momentum to the rotor and generator, with I being the moment of inertia of rotor and generator, yields:

$$I\frac{\mathrm{d}\omega}{\mathrm{d}t} = M_{aero} - M_{gen}. \tag{2.2}$$

Therefore, controlling the torque in region II according to

$$M_{gen} = \kappa\omega^2 \tag{2.3}$$

maximizes P_{gen}, with κ being a proportionality constant that can be found experimentally. This control mechanism will be referred to as greedy control, since it seeks to maximize generated power of a single turbine. [20, p.63 - 77]

2.1.3. Blade Element / Momentum Theory

To study the physical phenomena connected to wind turbines, models of the turbines and the airflow have been developed. Blade Element / Momentum Theory combines one-dimensional momentum theory and local forces on a section of a blade, the so called blade element, to give a quick way to analyze the loads on a rotor. Momentum theory assumes the flow to be steady, inviscid, incompressible and axisymmetric. The rotor is assumed to have an infinite number of blades and thus becomes a permeable disc. It is based on the integral forms of conservation of mass, momenta and energy:

$$\oint_{\partial V} \rho \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}A = 0 \tag{2.4}$$

$$\oint_{\partial V} \rho u_x \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}A = T - \oint_{\partial V} p \mathbf{n} \cdot \mathbf{e}_x \, \mathrm{d}A \tag{2.5}$$

$$\oint_{\partial V} \rho r u_\theta \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}A = M_{aero} \tag{2.6}$$

$$\oint_{\partial V} \left(p + \frac{1}{2} \rho \|\mathbf{u}\|^2 \right) \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}A = P. \tag{2.7}$$

The surface of the control volume V is denoted as ∂V, density is denoted as ρ, the velocity vector is $\mathbf{u} = [u_x, u_r, u_\theta]$ and $\mathbf{n}$ is the normal vector pointing outwards of the control volume. T is the thrust force acting on the rotor in streamwise direction and P the power extracted by the rotor. The three dimensionless quantities tip speed ratio λ, thrust coefficient C_T and power coefficient C_P are defined as follows:

$$\lambda = \frac{\omega R}{V_0} \tag{2.8}$$

$$C_T = \frac{T}{\frac{1}{2} \rho A V_0^2} \tag{2.9}$$

$$C_P = \frac{P}{\frac{1}{2} \rho A V_0^3}, \tag{2.10}$$

with R being the radius of the rotor and and V_0 is the mean wind speed. Applying these equations to a control volume enclosed by the stream tube around the rotor disc under the assumption that $u_x = u_r = const$ in the rotor disc, yields these basic relationships for the thrust and power:

$$T = 2\rho A V_0^2 a(1-a), \quad C_T = 4a(1-a) \tag{2.11}$$

$$P = 2\rho A V_0^3 a(1-a)^2, \quad C_P = 4a(1-a)^2. \tag{2.12}$$

The axial interference factor a is a measure for the influence of the rotor disc on the velocity in the stream tube and defined as

$$a = 1 - \frac{u_r}{V_0}. \tag{2.13}$$

From (2.12) it is possible to find a maximum power coefficient, which is given by

$$C_{P,max} = \frac{16}{27} \approx 59.3\%, \quad a = \frac{1}{3}, \tag{2.14}$$

and is referred to as the Betz-Joukowski limit. In practice this limit can not be reached due to higher dimensional effects such as the rotation of the wake. Sørenensen gives an assessment of the assumptions made in [47]. [47, p. 7 - 11]

The local forces acting on the blade are calculated assuming a 2D flow around an airfoil. The forces and velocities in the local coordinate system of the blade are shown in Figure 2.2, with x being the global streamwise direction. The forces $\mathbf{F}_n$ and $\mathbf{F}_t$ are the forces on the blade element in normal and tangential direction, respectively, while the lift and drag forces are $\mathbf{F}_L$ and $\mathbf{F}_D$. The undisturbed wind speed is V_0, ωr is the velocity due to the rotation of the rotor and the induced velocity is defined as $\mathbf{V}_i = [-aV_0, a'\omega r]$, with axial and tangential induction factor a and a'. The sum of these velocities is the relative velocity $\mathbf{V}_{rel}$.

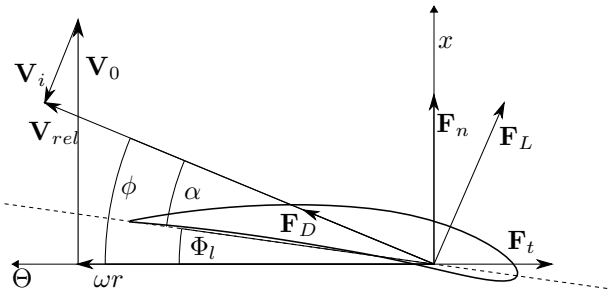

Figure 2.2.: Schematic of the local forces and velocities at a blade element.

The thrust and torque on the rotor within an infinitesimally thick stream tube are calculated as follows:

$$\frac{\mathrm{d}T}{\mathrm{d}r} = N_b F_n = \frac{1}{2} N_b \rho L_c \|\mathbf{V}_{rel}\|^2 C_n \tag{2.15}$$

$$\frac{\mathrm{d}M_{aero}}{\mathrm{d}r} = N_b r F_t = \frac{1}{2} N_b \rho L_c \|r\mathbf{V}_{rel}\|^2 C_t, \tag{2.16}$$

with N_b being the number of blades and L_c the chord length of the airfoil. The force coefficients C_n and C_t can be found by projecting lift and drag of the airfoil, which can usually be found as tabulated values, into the global coordinate system. Lift and drag depend on the local angle of attack α and the Reynolds number. The angle of attack α can be found through the difference of the angle between rotor plane and $\mathbf{V}_{rel}$, denoted as ϕ and the local pitch Φ_l. Combining (2.15) and (2.16) with the thrust and torque found by applying momentum theory to the same stream tube

yields:

$$a = \frac{1}{4\sin(\phi)^2/\left(sC_n\right)+1} \tag{2.17}$$

$$a' = \frac{1}{4\sin(\phi)\cos(\phi)/\left(sC_t\right)-1}, \tag{2.18}$$

with the solidity $s = N_b c/(2\pi r)$. This allows for an iterative computation of the infinitesimal torque and thrust as described in Algorithm 1. Integrating these with respect to the radius yields M_{aero} and T. In practice, the blade is defined as finite sized blade elements and the integration becomes a summation. [47, p. 100 - 103]

Many corrections for BEM have been found to increase the accuracy of the model. Among them is

Algorithm 1 Algorithm to compute thrust and torque on a blade element

for all elements **do**
  $a \leftarrow 0$
  $a' \leftarrow 0$
  repeat
    $\phi \leftarrow \tan^{-1}((1-a)/(\lambda r/R(1+a')))$
    $\alpha \leftarrow \phi - \Phi_l$
    compute C_n and C_t from lift and drag tables
    calculate new a and a' from (2.17) and (2.18)
  until a and a' converge
end for
calculate $\int \mathrm{d}T$ and $\int \mathrm{d}M$ according to (2.15) and (2.16)

the Prandtl tip loss factor, which proposes a factor to correct for the error due to the assumption of an infinite number of blades. Furthermore, above an axial induction factor of $1/3$, the assumptions made by BEM become inaccurate, as momentum theory predicts an expansion of the wake that is significantly too large. A correction for the calculation of C_t has been proposed by Glauert. Many other corrections for effects caused by the wake have also been proposed, such as a coupled near and far wake model by Pirrung et al. [38]. [47, p. 103 - 104]

2.1.4. Actuator Line Model

To study the interaction of fluid and turbine in detail, the flow field has to be resolved by means of computational fluid dynamics (CFD). Since fully resolving the shape of the blades would require a prohibitively fine resolution, the influence of the blade on the fluid is modelled. An overview over the models applied can be found in [7] and [30]. One such model is the actuator line model (ALM), which models each blade as a line of forces on the fluid. Like in BEM, the forces on the blade are calculated from local velocities and airfoil data at the points r_j along the ith actuator line $\mathbf{e}_i$. These

forces are then distributed by applying a convolution with a gaussian filter kernel $\eta(d)$:

$$\eta(d) = \frac{1}{\epsilon^2 \pi^{3/2}} e^{(-(d/\epsilon)^2)} \tag{2.19}$$

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{N_b} \int_0^R \left(\mathbf{F}_n(r) + \mathbf{F}_t(r)\right) \eta(\|\mathbf{x} - r\mathbf{e}_i\|)\, \mathrm{d}r. \tag{2.20}$$

The parameter ϵ is referred to as the smearing width, since it controls the stretching of the bell curve. As shown for example by Asmuth et. al [4], the ALM does not account for velocity induced by the root and tip vortices. This leads to an overprediction of forces on the blade, most significantly of the tangential force near the tip. Among others, Meyer-Forsting et al. have proposed corrections based on an iterative correction of the relative velocity [35]. [46]

2.2. Machine Learning for a Continuous Control Problem

2.2.1. Reinforcement Learning

Markov Decision Process

The mathematical formulation on which RL is based is the Markov Decision Process (MDP). Its two main components are the agent and the environment. Given a state $\mathbf{S}_t$ the agent takes an action $\mathbf{A}_t$. The environment responds to the action $\mathbf{A}_t$ with changing its state to $\mathbf{S}_{t+1}$ and giving feedback to the agent in form of a reward R_t. The interaction takes place at discrete time steps t and the sequence of state, action and reward is referred to as the trajectory. The dynamics of the MDP are described by the state transition function p that is defined as:

$$\mathrm{p}(\mathbf{s}', \mathrm{r}|\mathbf{s}, \mathbf{a}) \doteq Pr(\mathbf{S}_t = \mathbf{s}', \mathrm{R}_t = \mathrm{r}|\mathbf{S}_{t-1} = \mathbf{s}, \mathbf{A}_{t-1} = \mathbf{a}). \tag{2.21}$$

It defines the probability of state $\mathbf{s}'$ with reward r occuring, given the state $\mathbf{s}$ and action $\mathbf{a}$. Note that the following derivations will be constricted to finite MDPs, meaning that state and action space are discrete. However, the concepts all are transferable to continuous action and state space.
For a process to be a Markov Decision Process, p must only depent on $\mathbf{s}$ and $\mathbf{a}$. Therefore, $\mathbf{s}$ must include all information necessary to determine the future behaviour of the environment. This is not limited to information currently present in the environment. When thinking of this in terms of the wind farm problem at hand, the state could include data about wind speeds at the current time but also from time steps in the past. This approach allows to model virtually any interaction as a MDP, simply by including every bit of information from the beginning of time into the state. Obviously, this is not feasible and therefore a careful choice of the information in the state is necessary.

The goal of the learning process is to maximize the sum of the rewards in the long run. Therefore a new quantity is defined, the return G_t that includes not only R_t but also the rewards received in future time steps. While in many applications of RL, the process naturally comes to an end, referred to as the terminal state $\mathbf{S}_T$, in problems of continuous control this is not the case. Therefore the timeline is broken up into episodes of length T. This allows for a finite computation of G_t. A typical formulation of G_t, referred to as a discounted return is:

$$G_t \doteq R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} ... = \sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}, \quad \gamma \in [0, 1]. \tag{2.22}$$

It includes a discount rate γ, that emphasizes rewards in the near future. If $\gamma = 0$, $G_t = R_t$, if $\gamma = 1$, the return is the sum of all future rewards. [48, p. 47- 57]
Now the goal of the learning process is defined, but not what to learn. There exist three possible answers to this question: a model of the environment, a value function or a policy. Also combinations of these components is possible. In the case of continuous control, most common approaches are model-free, meaning either learning a value function, a policy or both. Therefore model-based methods will not be discussed further and the reader is referred to the book by Sutton and Barto [48].
In loose terms, the policy π guides the agent on which action to take. More formally, $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$ is the probability of action $\mathbf{A}_t = \mathbf{a}$ given a state $\mathbf{S}_t = \mathbf{s}$ under the policy π with its parameters set to $\boldsymbol{\theta}$. The value function of a state $\mathbf{s}$ under a policy π is denoted as $v_\pi(\mathbf{s})$ and is the expected return if the agent acts according to π, starting from state $\mathbf{s}$. Note that for convenience the parameters of π were dropped. For MDPs it can be defined as:

$$v_\pi(\mathbf{s}) \doteq \mathrm{E}_\pi \left[G_t | \mathbf{S}_t = \mathbf{s} \right] = \mathrm{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} | \mathbf{S}_t = \mathbf{s} \right] \tag{2.23}$$

$$= \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \sum_{\mathbf{s}'} \sum_{r} p(\mathbf{s}', r|\mathbf{s}, \mathbf{a}) \left(r + \gamma v_\pi(\mathbf{s}') \right). \tag{2.24}$$

In the form of (2.24), the equation is referred to as the Bellmann equation and its unique solution is the value function v_π. Analogously, the action-value function is the expected reward of taking action $\mathbf{a}$ at state $\mathbf{s}$ under the policy π, denoted by $q_\pi(\mathbf{s}, \mathbf{a})$. It is defined by [48, p. 58-59]:

$$q_\pi(\mathbf{s}, \mathbf{a}) \doteq \mathrm{E}_\pi \left[G_t | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a} \right] = \mathrm{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a} \right]. \tag{2.25}$$

Policy Gradient Methods

With a known value function, it is possible to construct a policy and by improving the value function, the policy can be improved. However, there exist advantages to directly improve the policy,

especially for continuous state and action spaces. Policy gradient methods use the gradient of a perfomance measure $\mathsf{J}(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ of a policy. This gradient can be used in optimization algorithms, such as stochastic gradient descent [48, p. 201] or its extensions such as Adam [31]. In its basic form, SGD performs an update according to:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \psi \nabla_{\boldsymbol{\theta}} \mathsf{J}(\boldsymbol{\theta}). \tag{2.26}$$

The parameter ψ is called the learning rate. The policy gradient methods differ now only in the performance measure. The first such algorithm proposed is the REINFORCE algorithm [50]. Its definition of $\nabla_{\boldsymbol{\theta}} \mathsf{J}(\boldsymbol{\theta})$ is:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathsf{J}(\boldsymbol{\theta}) &= \mathrm{E}_{\pi}\left[\sum_{\mathbf{a}} \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{S}_t)\mathsf{q}_{\pi}(\mathbf{S}_t, \mathbf{a}) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{S}_t)}\right] && (2.27)\\
&= \mathrm{E}_{\pi}\left[\mathsf{G}_t \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}\right] && (2.28)\\
&\approx \frac{1}{N_{e,b}} \sum^{N_e,b} \frac{1}{\mathsf{T}} \sum_{t}^{\mathsf{T}} \mathsf{G}_t \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}. && (2.29)
\end{aligned}$$

It follows from the policy gradient theorem and substitution of all values of **A** and **S** with the actions and states from one trajectory. Thus, all the values necessary for the computation of the gradient are known. The expected value can approximated by the mean of a batch of episodes with $N_{e,b}$ being the number of episodes per batch. [48, p.324-328]
While this algorithm makes a computation possible, it is inefficient. Therefore newer methods have been devised such as the Trust Region Policy Optimization (TRPO) [43] and the Proximal Policy Optimization (PPO) [44]. They are closely related and both propose a surrogate performance measure that limits the size of the gradient to ensure monotonic improvement in the case of TRPO and a close approximate in the case of ppo. However, the computation of the surrogate in PPO is simpler and more efficient, which helped policy gradient methods to become one of the most used algorithms in continuous control problems. One version of the surrogate performance measure, which is also referred to as objective, is:

$$\begin{aligned}
\mathsf{pr}_t(\boldsymbol{\theta}) &\doteq \frac{\pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{A}_t|\mathbf{S}_t)} && (2.30)\\
\mathsf{J} &\doteq \mathrm{E}_{\pi}\left[\min\left(\mathsf{pr}_t(\boldsymbol{\theta})\mathsf{a}_{\pi}(\mathbf{A}_t|\mathbf{S}_t), \mathrm{clip}(\mathsf{pr}_t(\boldsymbol{\theta}), 1-\beta, 1+\beta)\mathsf{a}_{\pi}(\mathbf{A}_t|\mathbf{S}_t)\right)\right]. && (2.31)
\end{aligned}$$

The probability ratio pr_t compares the policy after the update to the policy before the update. Therefore $\pi_{\boldsymbol{\theta}}$ has to be approximated. $\mathsf{a}_{\pi}(\mathbf{A}_t|\mathbf{S}_t)$ is an estimator of the advantage function, which is defined as $\mathsf{a}_{\pi}(\mathbf{A}_t|\mathbf{S}_t) = \mathsf{q}_{\pi}(\mathbf{A}_t|\mathbf{S}_t) - \mathsf{v}_{\pi}(\mathbf{S}_t)$. This estimator also has to be found, however, this is a regular optimization problem, which can be solved by use of SGD or Adam. The parameter β is referred to as

Table 2.1.: Activation functions commonly used in neural networks

Name	Domain	Range	Definition	Derivative
$\tanh$	$(-\infty, +\infty)$	$(-1, -1)$	$\tanh(x) = (e^x - e^{-x})(e^x + e^{-x})^{-1}$	$1 - \tanh^2(x)$
σ	$(-\infty, +\infty)$	$(0, -1)$	$\sigma(x) = (1 + e^{-x})^{-1}$	$\sigma(x)\sigma(-x)$
$\mathrm{softplus}$	$(-\infty, +\infty)$	$(0, +\infty)$	$\mathrm{softplus}(x) = \ln(e^x + 1)$	$\sigma(x)$

the clipping ratio. [44]
In addition to the ratio probability clipping, other regularizations can be added to the objective function, for example an l_2 regularization, that adds a penalty proportional to $\|\boldsymbol{\theta}\|^2$.

2.2.2. Artificial Neural Networks

Feed-forward networks

In the section above, a way to update parameters of the policy or value function was described, however, no description of the policy function itself was given. In principal any function with a set of parameters can be used, but usually, an artificial neural network (ANN) is used. ANNs are comprised of layers of neurons. More precisely, a layer k of N neurons takes as input a vector $\mathbf{z}$ of length M. The output of the layer is a new vector $\mathbf{y}^{\mathsf{k}}$, which is then the input for the next layer of the network. In a simple feed-forward layer $\mathsf{y}_j^{\mathsf{k}}$ is computed according to:

$$\mathsf{y}_j^{\mathsf{k}} = \mathsf{f}(\mathsf{w}_{i,j}^{\mathsf{k}}\, \mathsf{z}_i^{\mathsf{k}} + \mathsf{b}_j^{\mathsf{k}}). \tag{2.32}$$

The entries $\mathsf{w}_{i,j}^{\mathsf{k}}$ of the matrix $\mathbf{W}^{\mathsf{k}}$ are called the weights of the layer and the entries $\mathsf{b}_j^{\mathsf{k}}$ of the vector $\mathbf{b}^{\mathsf{k}}$ are called its bias. f is called the activation function, which can be chosen freely, but typical choices include the hyperbolic tangent ($\tanh$), the sigmoid-function (σ) or the softplus ($\mathrm{softplus}$) function. A small overview over some of the key features of these functions is given in Table 2.1. One desirable property for an activation function is that it is differentiable at least once in the entire domain. Thus a network of two layers with $\tanh$ as an activation function describes the function

$$\mathbf{y} = \mathbf{tanh}(\mathbf{b^1} + \mathbf{W^1} \cdot \mathbf{tanh}(\mathbf{b^0} + \mathbf{W^0} \cdot \mathbf{z})), \tag{2.33}$$

with $\mathbf{z}$ being the input to the network and $\mathbf{y}$ its output and the activation function being applied element-wise to the vectors. [9, p. 2-2 - 2-12].

Recurrent neural networks

It is obvious that a feed-forward network only produces an output influenced by the current activation. There can be no influence of previous activations for example in a time-series. Yet there exist many applications for which such a feature would be useful, for example the field of natural language processing [17] or transient physical processes. This led to the development of recurrent neural networks. These include a hidden state, which is preserved. Especially the development of the long short-term memory (LSTM) cell has had great impact on the success of recurrent neural networks due to its ability to preserve hidden states over a longer period of time [21]. An LSTM layer consists of one or multiple cells, with the input being the entire or parts of the activation. Each cell passes a vector called the cell state $\mathbf{c}$ as well as its output $\mathbf{y}$ to itself in the next time step. Thus a cell has as inputs at time t the cell state and output of the previous time step, $\mathbf{c}_{t-1}$ and $\mathbf{y}_{t-1}$, as well as the current activation $\mathbf{z}_t$. On the inside, the cell consists of a forget-gate, an input-gate and an output-gate. The forget-gate determines the influence of the cell-state, the input-gate determines the influence of the current time-step on the cell-state. The output-gate then computes the output based on the updated cell-state. A schematic of the cell is given in Figure 2.3. From left to right the sigmoid layers are the forget-, input- and output-gate, whereas the $\tanh$-layer corresponds most closely to the layer of a feed-forward-network.

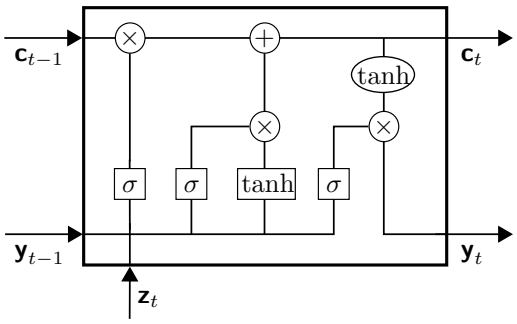

Figure 2.3.: Schematic representation of an LSTM cell, operation in boxes represent layers with trainable weights, operations in ellipses are pointwise operations.

Deep Reinforcement Learning

If RL is combined with a neural network, it is often referred to as deep reinforcement learning or DRL. In case of a policy gradient method, the neural network is part of the policy. For example, the state can be used as the input of a neural network and the output of the network can be used to set the parameters of a distribution function. Assuming that each entry in the action vector $\mathbf{A}$ is a statistically indepedent random variable with the probability density function pdf, parameterized

by μ_i, the policy can be written as

$$\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S}) = \prod_i pdf(\mathsf{A}_i, \mu_i(\mathbf{S}, \boldsymbol{\theta})). \tag{2.34}$$

If μ_i is the output of a K-layered network with activation function f, it is:

$$\mu_i(\mathbf{S}, \boldsymbol{\theta}) = \mathsf{f}^{\mathsf{K}}(\mathsf{b}_i^{\mathsf{K}} + \mathsf{w}_{i,j}^{\mathsf{K}} \mathsf{f}^{\mathsf{K}-1}(...\mathsf{f}^0(\mathsf{b}_k^0 + \mathsf{w}_{k,l}^0 \mathsf{S}_l)...)). \tag{2.35}$$

Then the parameters $\boldsymbol{\theta}$ of the policy $\pi_{\boldsymbol{\theta}}$ are the weights and biases of the neural network:

$$\boldsymbol{\theta} = \left[\mathsf{b}_j^{\mathsf{k}}, \mathsf{w}_{i,j}^{\mathsf{k}}\right]^T. \tag{2.36}$$

A schematic of DRL with a policy gradient method is given in Figure 2.4.

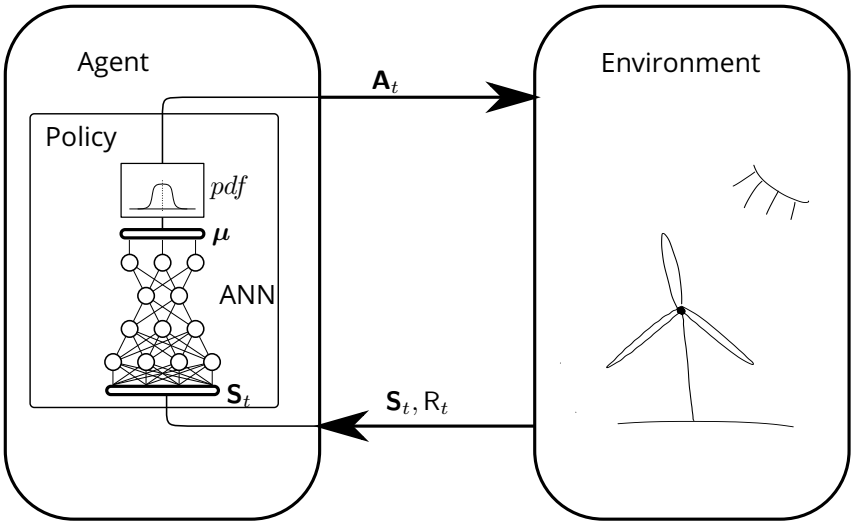

Figure 2.4.: Schematic of a markov decision process for a policy gradient method with a neural network.

Training of a Neural Network

To use a neural network as policy in a policy gradient method algorithm, a way to compute the gradient of the objective J has to be found. The parameters of the policy are the weights and biases of the network, therefore the partial derivatives of J with respect to all weights and biases have to be found. Both described policy gradient methods express the objective gradient as a function j of the gradient of the policy:

$$\nabla_{\boldsymbol{\theta}} \mathsf{J} = \mathsf{j}\left(\frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})}{\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})}\right). \tag{2.37}$$

Under the same assumptions as above, the gradient term is: [48, p. 335]

$$\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})}{\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})} = \prod_i \frac{\partial pdf(\mathsf{A}_i, \mu_i(\mathbf{S},\boldsymbol{\theta}))}{\partial \mu_i(\mathbf{S},\boldsymbol{\theta})} \frac{\nabla_{\boldsymbol{\theta}}\mu_i(\mathbf{S},\boldsymbol{\theta})}{pdf(\mathsf{A}_i, \mu_i(\mathbf{S},\boldsymbol{\theta}))}. \tag{2.38}$$

Computing the gradient

$$\nabla_{\boldsymbol{\theta}}\mu_i|_{\mathsf{S}} = \left[\left.\frac{\partial \mu_i}{\partial \mathsf{b}_j^{\mathsf{k}}}\right|_{\mathsf{S}}, \left.\frac{\partial \mu_i}{\partial \mathsf{w}_{j,k}^{\mathsf{k}}}\right|_{\mathsf{S}} \right]^T \tag{2.39}$$

can now be done efficiently via backpropagation:

$$\mathsf{x}_i^{\mathsf{k}} = \mathsf{b}_i^{\mathsf{k}} + \mathsf{w}_{i,j}^{\mathsf{k}}\mathsf{z}_j^{\mathsf{k}}, \quad \mathsf{z}_i^{\mathsf{k}+1} = \mathsf{f}^{\mathsf{k}}(\mathsf{x}_i^{\mathsf{k}}), \quad \mathsf{z}_i^0 = \mathsf{S}_i \tag{2.40}$$

$$\mathsf{t}_{i,j}^{\mathsf{K}} = \left.\frac{\partial f^{\mathsf{K}}(x)}{\partial x}\right|_{\mathsf{x}_i^{\mathsf{K}}} \delta_{i,j}, \quad \mathsf{t}_{i,j}^{\mathsf{k}} = \mathsf{t}_{i,k}^{\mathsf{k}+1}\mathsf{w}_{k,j}^{\mathsf{k}} \left.\frac{\partial f^{\mathsf{k}}(x)}{\partial x}\right|_{\mathsf{x}_i^{\mathsf{k}}} \tag{2.41}$$

$$\left.\frac{\partial \mu_i}{\partial \mathsf{b}_j^{\mathsf{k}}}\right|_{\mathsf{S}} = \mathsf{t}_{i,j}^{\mathsf{k}}, \quad \left.\frac{\partial \mu_i}{\partial \mathsf{w}_{j,k}^{\mathsf{k}}}\right|_{\mathsf{S}} = \mathsf{t}_{i,j}^{\mathsf{k}}\mathsf{z}_k^{\mathsf{k}} \tag{2.42}$$

The name refers to the fact, that due to the chain rule, the gradient of a layer is easily expressed through the gradient of the previous layer, which allows for an efficient computation. The sensitivity $\mathsf{t}_{i,j}^{\mathsf{k}}$ determines how sensible the action is to changes of this parameter. [9, p.11-7 - 11-13]

2.3. The Lattice Boltzmann Method

2.3.1. The Boltzmann Equation

The agent described in the section above needs to interact with an environment in order to train. In the case studied in this thesis the environment is a wind farm with multiple turbines. Therefore the fluid field within the wind park has to be calculated. Traditionally solvers based on the discretized Navier-Stokes-Equations (NSE) are used. However, parallelization has proven to be difficult. A newer approach is the Lattice Boltzmann Method (LBM). The basics of its theory will be layed out in this chapter.

LBM is based on a discretization of the Boltzmann Equation derived from the kinetic theory of gases. This description is often referred to as a mesoscopic description, since it stands in between the scale of continuum theory and the scale of single particles, which will be referred to as macroscopic and microscopic scale, respectively. While the NSE describe the medium through density ρ, velocity $\mathbf{u}$ and pressure p, the Boltzmann Equation considers the particle density function (pdf) f, which describes the distribution of particles in six dimensional phase space. It is therefore a functductiion of space $\mathbf{x}$, microscopic velocity $\boldsymbol{\xi} = [\xi, \upsilon, \zeta]^T$ and time t. However, the macroscopic quantities can

be recovered from the pdf by taking its zeroth and first moments in velocity space:

$$\rho(\mathbf{x},t) = \int_{-\infty}^{\infty} f(\mathbf{x},\boldsymbol{\xi},t)\mathrm{d}\boldsymbol{\xi} \tag{2.43}$$

$$\rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t) = \int_{-\infty}^{\infty} \boldsymbol{\xi} f(\mathbf{x},\boldsymbol{\xi},t)\mathrm{d}\boldsymbol{\xi}. \tag{2.44}$$

Similarly the energy can be found by the third moment. The pressure can not be recovered directly, instead it is calculated by a state equation such as the ideal gas law. The pdf tends towards an equilibrium, which is given by the Maxwell equilibrium. . The Boltzmann Equation describes the change of the pdf over time. The change in time is governed by the collision operator Ω. It describes the change of f due to collisions of particles. In the Boltzmann Equation it is an integral operator, that accounts for all possible collisions. Therefore it is mathematically rather cumbersome which renders it unuseful for numerical discretization. Thus another formulation for Ω is used in LBM, which is an active area of research in the LBM community [8]. The Boltzmann Equation is:

$$\frac{Df(\mathbf{x},\boldsymbol{\xi},t)}{Dt} = \frac{\partial f(\mathbf{x},\boldsymbol{\xi},t)}{\partial t} + \frac{\partial f(\mathbf{x},\boldsymbol{\xi},t)}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial t} + \frac{\partial f(\mathbf{x},\boldsymbol{\xi},t)}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial t} = \Omega(f). \tag{2.45}$$

The first form is the total derivative of f with respect to time. The second term in the second form is a convection term, while the third term corresponds to a forcing term. In comparison to the NSE the Boltzmann Equation lacks a diffusive term. Diffusion occurs only through the collision operator. Through collision the particles tend towards equilibrium, the pdf of that equilibrium is known as the Maxwell equilibrium. It can be described by macroscopic quantities only:

$$f^{eq}(\mathbf{x},\boldsymbol{\xi},t) = \rho\left(\frac{1}{2\pi\hat{R}\hat{T}}\right)e^{-\|\mathbf{v}\|^2/(2\hat{R}\hat{T})}, \tag{2.46}$$

with $\hat{R}$ being the specific gas constant and $\hat{T}$ the temperature. The velocity $\mathbf{v}$ is defined as $\mathbf{v} = \boldsymbol{\xi} - \mathbf{u}$. [32, p. 15- 21]

2.3.2. Discretization

In order to use the Boltzmann Equation for numerical simulations it needs to be discretized in space, velocity and time and a collision operator has to be defined. The velocity discretization can be based on the Hermite Series expansion, since its generating function is of the same form as the equilibrium distribution. To recover the first three moments of the distribution correctly, i.e. density, velocity and energy, truncation of the series after the first three terms is sufficient. The roots of the Hermite polynomials up to order three are the necessary discrete velocities $\boldsymbol{\xi}_i$. Together with the weights b_i, that are used in the Gauss-Hermite quadrature, the velocities form a velocity set. Velocity sets are

denoted in the DdQq-notation, with d being the number of spatial dimensions and q the number of discrete velocities. In three dimensions D3Q15, D3Q19 and D3Q27 can be used to recover the Navier-Stokes-Equations, for high Reynolds number flows D3Q27 is the most suitable[29]. [32, p. 73-93]
Time is discretized via the explicit Euler forward scheme, which can be shown to be second order accurate. Space is discretized uniformly into a cubic grid, so that discrete pdfs move from one node of the grid to the other in one timestep. The ratio of timestep to lattice width is called the lattice speed of sound c_s. The computation of a timestep is usually separated into two parts, the streaming step, in which populations are advected from one node to the other, and the collision step, in which the collision operator is applied. Applying the entire discretization gives the Lattice Boltzmann Equation (LBE), with $\mathbf{c}_i$ being $\boldsymbol{\xi}_i/\sqrt{3}$ for convenience:

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, \mathbf{c}_i, t + \Delta t) = \Omega_i(f) + f_i(\mathbf{x}_i, \mathbf{c}_i, t). \tag{2.47}$$

It is, compared to discretized NSE, very simple and the separation into collision and streaming makes it easily parallelizible. Furthermore the equations to compute density and velocity are: [32, p. 94-98]

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t) \tag{2.48}$$

$$\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) = \sum_i \mathbf{c}_i f_i(\mathbf{x}, t) \tag{2.49}$$

Lastly a collision operator has to be found. The first applicable collision operator proposed was the Bhatnager-Gross-Krook (BGK) operator:

$$\Omega_i^{BGK} = \hat{\omega}\left(f_i - f_i^{eq}\right). \tag{2.50}$$

It is based on the fact, that the distributions tend to the equilibrium distribution, therefore the BGK operator relaxes the distributions towards equilibrium with a constant relaxation frequency $\hat{\omega}$. Via Chapmann Enskog analysis it can be shown that this relaxation frequency is related to the kinetic viscosity ν: [32, p. 98-100, 112]

$$\nu = c_s^2\left(\frac{1}{\hat{\omega}} - \frac{\Delta t}{2}\right). \tag{2.51}$$

While the BGK operator is sufficient for low Reynolds number flows, it becomes unstable at higher Reynolds numbers. Therefore more sophisticated methods had to be developed. One approach is to transform the distributions into moment space and relaxe the moments independently, leading to multiple relaxation time (MRT) methods. Geier et al. argue, that they cannot be relaxed separately, since these moments are not statistically independent. However, cumulants of a distri-

bution are statistically independent by design, therefore they can also be relaxed independently. Furthermore, they fulfill Galilean invariance, which is not always the case for MRT methods. It can be shown that with a parametrization this method can be fourth order accurate[14]. In underresolved flows, the parameter of this parametrization can be used to influence the numerical diffusivity of the cumulant operator, acting like an implicit subgrid scale model for Large-Eddy-Simulations (LES)[3]. However, the exact behaviour of the underresolved cumulant operator is not yet known. More details on the cumulant LBM as well as a derivation for a refinement algorithm can be found in Appendix A.
While LBM has some advantages over NSE-based algorithms, the treatment of boundary conditions is often more complicated. This is due to the fact that it is necessary to prescribe the populations in the boundary nodes. No-slip and full slip boundaries are imposed by bounce-back and bounce forward, respectively and velocity boundary conditions can be imposed by prescribing the corresponding equilibrium distributions or by extending the bounce back approach [32, p. 175 - 189, 199 – 207]. Another problem often arising in LBM is the reflection of acoustic waves, especially at inlet and outlet boundaries. There exist methods to cancel these waves, while another approach is to simply increase the viscosity in a so called sponge layer near the outlet [32, p. 522 - 526].

2.4. Previous Works

2.4.1. Wind Farm Control

In subsection 2.1.2 the greedy control strategy for a single turbine was described. However, this control strategy does not take into consideration the effects of the wake on other turbines within a wind farm. Therefore more sophisticated strategies with the objective of maximizing the generated power of multiple turbines or a whole wind farm have been proposed. They can be divided into two main categories, axial induction control and wake redirection control [6]. Axial induction control tries to lower the power intake of the first turbine, so that the wind speed at the following turbines is higher. Wake redirection control aims at steering the wake away from the next turbine downstream. A fairly recent review of the vast amount of studies dedicated to this topic can be found in [30].
Overall, it shows that static axial induction control is able to increase total power, in LES-simulations usually around 5 percent. However, the magnitude of the increase depends on the simulation method and flow conditions. Aerodynamic phenomena not captured by BEM-like models can significantly decrease the generated power. An increase in turbulence intensity also leads to a decrease in power. This highlights the importance of LES simulations for studies on control. Compared to the static approaches, dynamic approaches showed to be more promising. Based on a receding-

horizon optimal control, Goit and Meyers [18] were able to report an increase in power of 16 percent. However, this approach requires knowledge of the full flow field as well as adjoint simulations and thus not applicable to real control. Based on these results, a simplified control strategy based on sinusoidal variation of the induction was proposed by Munters und Meyers [36]. This approach was further explored by Frederik et. al [10], leading to the development of the helix approach, based on a periodic variation of pitch angles. This results in a sinusoidal moment exerted on the wake and thus increasing wake mixing. They could report an increase in power of up to 7.5 percent. These results prove the theoretical potential for power increase by dynamic axial induction control and that simple control strategies can achieve considerible gains.
Static and dynamic wake redirection have also been explored, and the review in [30] concludes that it shows an even greater potential for improving power. However, this work will only focus on induction control.

2.4.2. Machine Learning in Active Flow Control

Formulating control strategies for active flow is inherently difficult, due to the non-linear nature of fluid-flow. This motivates the interest in methods of machine learning, since it has been shown in other areas that these methods can develop new strategies and deal with large state spaces, most famously in games such as Go [45], but also . Recently, some studies applying reinforcement learning have been conducted. A review of some of them can be found in [12]. The work by Verma et al. focussed on collectively swimming fishes and employed Q-learning, in which the action-value function is approximated by a neural network [49]. Rabault et al. were the first ones to apply policy gradient methods. They used it to minimize the drag on a two-dimensional von-Kármán vortex street via jets [40]. It showed the applicability of these methods to active flow control. In subsequent studies, which focussed on similar problems, i.e. two-dimensional laminar flow, more aspects of the application of DRL to flow control were studied. The use of parallel environments to reduce the wall time was proposed in [39]. In [5], policy gradient methods were successfully applied to control of a one-dimensional, unstable falling liquid film. While these studies showed promising results, the complexity of the controlled flows was still limited in comparison to a fully turbulent three-dimensional LES simulation.

3. Setup of Simulation

3.1. Implementation

3.1.1. Implementation Strategy

In this work the performance of two controllers in two simulation environments is studied. As a baseline case a greedy controller is implemented. This is compared to a controller that is governed by an RL-agent. They are compared in a one-dimensional BEM model of a turbine, which is used for a more rapid development and an initial parameter study, and an LBM-ALM environment with multiple turbines. Everything is implemented in Python with the exception of the LBM-ALM, which is implemented in C++.
In contrast to most other applications of machine learning, in this case the main program is not the one governing the machine learning, but rather the simulation environment. This is due to the fact that the LBM-ALM simulation environment already existed prior to this work and it was expected to be simpler to design the RL-controller so that it could be called by the simulation environment, rather than breaking up the already existing code. Furthermore the LBM-ALM consumes the vast amount of computational time, so any potential for optimization in this part should be used. However, this approach also created some difficulties, especially regarding the concurrency of events and scope of objects, many of which were solved by using techniques of object oriented programming.

3.1.2. RL-Controller

The RL-Controller relies on the library TF-Agents [19], which implements many RL algorithms and is based on the Python API of TensorFlow [1]. The controller implements an MDP, with an environment and an agent and also governs the interaction between them. The frequency of interactions is not related to the size of the timestep of the simulation environment, since the relevant time scales are not necessarily connected [40]. It is governed by a parameter, the number of timesteps per action, $N_{t,A}$.
An instance of the class `PPOAgent` represents the agent and contains the methods to calculate actions based on the observations and to train the agent. It is based on the PPO-algorithm explained in 2.2.1. The actor network consists of three layers, of which the first layer is either a fully connected LSTM cell or a regular feed forward layers The activation function of these layers is $\tanh$. They have the same width, which is adjustable. The last layer is a feed forward layer of $\mathrm{softplus}$ nodes or $\tanh$, depending on the controlled quantity. The value network is a three layered feed forward network

with a $\mathrm{softplus}$ activation.
The environment is implemented in the class `TurbineEnvironment`. The action a provided by the agent is smoothed with an exponential filter

$$\hat{\mathbf{a}}_t = \hat{\mathbf{a}}_{t-1} + \alpha_f(\mathbf{a}_t - \hat{\mathbf{a}}_{t-1}) \tag{3.1}$$

to make it continuous as done for example in [40]. The decay rate α_f is $\alpha_f = r_f^{1/N_{t,A}}$ with a forget ratio r_f of 0.99. $\hat{\mathbf{a}}$ is applied as control of the turbine. To allow for more refined action space, the value of the control variables is calculated by

$$\varphi(t) = \varphi_{base} + \sum_{i=0}^{N_a} A_{\varphi,i} \cos(2\pi f_{\phi,i} t), \tag{3.2}$$

with φ being the controllable quantity, φ_{base} is a base value that is modulated by N_a cosine waves with frequencies $f_{\varphi,i}$ and amplitudes $A_{\phi,i}$. Unless controlled by the agent, these quantities are set according to the greedy control strategy, that is all amplitudes and base values are zero except the base-value of the T_{gen}.
The observations and reward are also filtered exponentially, with the same rate as the action. Observations can include quantities from turbines, such as ω, but also velocities sampled from the flow-field. They can also be from multiple past timesteps in addition to the current timestep. If the environment reaches a terminal state, because a rotor turns too slow or too fast, the environment is controlled by a greedy controller, until it is in a stable state again. This was implemented since a reinitialization of the LBM-ALM simulation environment would be very costly. To reduce wall time, multiple environments can be run at the same time, each represented by a single instance of `TurbineEnvironment`. Therefore, all the TurbineEnvironments are wrapped in a single instance of a `ParallelTurbineEnvironment`.

3.1.3. Greedy Controller

The greedy controller is an implementation of the baseline generator-torque controller given in [27], which is based on the greedy control strategy explained in 2.1.2. To simplify the calculations the controllable torque is the torque at the rotor, losses due to the transmission are ignored. The proportionally constant κ was optimized using the BEM environment and found to be $1.268 \times 10^6\,\mathrm{kgm}^2$.

3.1.4. LBM-ALM Environment

The LBM-ALM environment is based on the work of Asmuth et al. [3, 4]. The actuator line model is implemented in elbe, a cumulant LBM code utilizing graphical processing units (GPU) [26, 13]. For this work the code was extended to include a second order accurate refinement scheme according to Schönherr et al. [42], for further information the reader is referred to section A.2. Multiple independent domains can be instantiated in one call to the main function, with creating only one instance of the controller, allowing for a significant reduction in wall time [39]. The inlet is a uniform inflow, that can be superimposed with fluctuations of a Mann box [34], created by a synthetic turbulence generator. The outlet uses a sponge layer, as described in section 2.3.

3.1.5. Fast Implimentation of a BEM Environment

To provide the physical inputs of a one-dimensional model of the flow field and the turbine a steady state BEM-code for the computation of C_t and C_n was implemented. To minimize computational time, a table of 400 values of M_{aero} is precomputed and then linearly interpolated. The fluid field is composed of a base velocity and an optionally superimposed turbulent fluctuation of a Mann box, which is computed by a synthetic turbulence generator. The base velocity can also be varying in time, either by defining a sine wave or by regularly sampling a random velocity within a given interval.

3.1.6. Interaction of Controller and Simulation Environment

To provide a layer of abstraction between simulation environments and controllers, exchange of information takes place via the so-called visitors. Each turbine and probe is represented by an instance of a visitor. It is provided with input by the simulation environments such as M_{aero} or the velocity, but can also set control variables such as M_{gen}. This allows for an easy recombination of simulation environments and controllers.

As an example for the interaction between the controller and the simulation environment, a timestep of a simulation with the ALM-LBM and a RL-Controller is given in 3.1. It shows that the probes and turbines hold information but not execute methods themselves. It also shows the implementation languages. To simplify communication across languages, objects that exchange information across languages are mirrored in both languages. This includes the visitors as well as the controller.

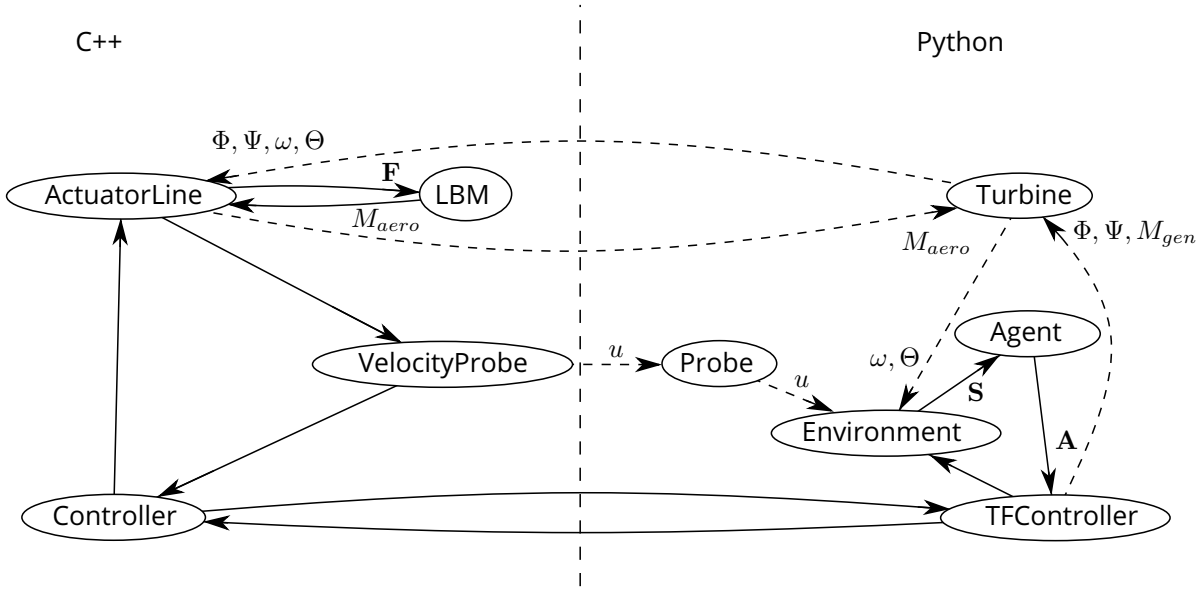

Figure 3.1.: Schematic of a timestep sequence. To trace order of execution, follow solid arrows and go around ellipsis clockwise, dashed arrows represent exchange of information.

3.2. Setup

3.2.1. Preliminary Studies in the BEM environment

To validate the implementation of the controllers and do studies of some of the hyperparameters of the agent the BEM environment is used. The airfoil data and physical properties are taken from the NREL5 reference [27]. The mean wind speed is $V_0 = 10.5\,\mathrm{m/s}$ and a velocity probe is placed ten and a half meters upstream of the turbine. The control variable is M_{gen}. The state of the environment consists of the angular velocity of the turbine and the streamwise velocity of the velocity probe. Training is conducted in six environments in parallel, with independently generated turbulent inflow. The size of the timestep of the simulation is $\Delta t = 0.1\,\mathrm{s}$. A parameter study of the following parameters is conducted:

- variance of action σ_A
- policy l_2 regularization coefficient ε_{p,l_2}
- value l_2 regularization coefficient ε_{v,l_2}

These parameters were chosen since they do not directly depend on the physics and layout of the problem. The parameters not varied are set according to Table 3.1.

Table 3.1.: Parameters of agent and environment

Parameter	Symbol	Value
Time per action		$1.0\,\mathrm{s}$
Number of actions per episode	$N_{A,e}$	500
Number of episodes per batch	$N_{e,b}$	12
Width of actor network		100
Width of value network		200
Variance of action	σ_A	0.1
Importance ratio clipping	β	0.2
Discount factor	γ	0.95
Policy l_2 regularizaion coefficient	ε_{p,l_2}	0.1
Value l_2 regularization coefficient	ε_{v,l_2}	0.1
Value network loss coefficient	$\varepsilon_{v,l}$	0.5
Learning rate	ψ	10^{-3}

3.2.2. The LBM-ALM Environment

The properties of the turbine are again taken from the NREL5 reference [27]. The diameter D of the rotor is therefore $D = 126\,\mathrm{m}$. Three turbines are placed in the center of the cross-stream plane with a distance of five diameters in streamwise direction. The first turbine is placed three diameters downstream of the inlet. The domain is 19 diameters long and six diameters wide and high. In order to reduce computational cost, a more refined domain is placed within the first domain. It is placed in the center of the crosswise plane and one diameter downstream of the inlet. The inflow is turbulent. The parameters of the domain are gathered in Table 3.2.

Training of the agents is again conducted in six environments in parallel.

Table 3.2.: Parameters of the domain

Parameter	Value
Rotor diameter	$126\,\mathrm{m}$
Outer domain H x W x L	$6\,\mathrm{D} \times 6\,\mathrm{D} \times 19\,\mathrm{D}$
Outer domain resolution	$8\,\mathrm{nodes/D}$
Inner domain H x W x L	$5\,\mathrm{D} \times 5\,\mathrm{D} \times 16\,\mathrm{D}$
Inner domain resolution	$16\,\mathrm{nodes/D}$
Turbulence intensity	$5\,\%$
Mean wind speed	$10.5\,\mathrm{m/s}$

3.2.3. Optimizing Specific Behaviour

3.2.4. Learning New Behaviour

4. Results

4.1. Validation

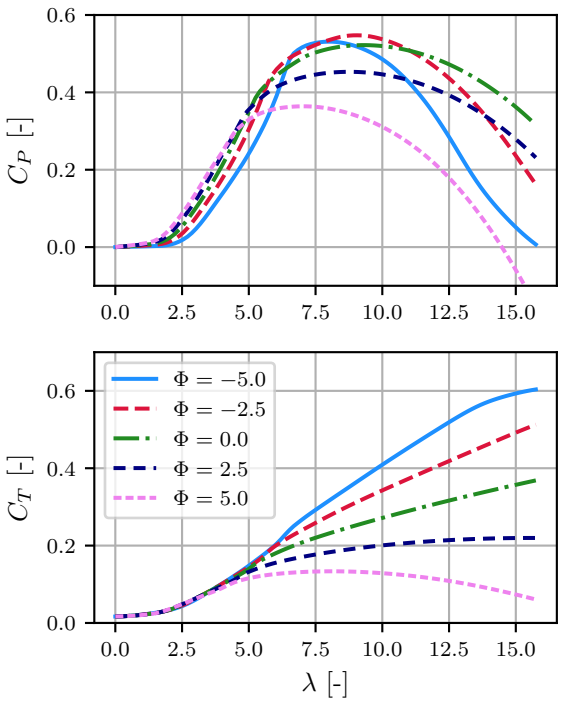

Figure 4.1.: C_P and C_T as functions of tip-speed ratio and pitch angle for the NREL 5MW turbine.

As a first step, the implementation of the BEM environment is validated. Figure 4.1 shows C_P and C_T computed as functions of the tip-speed ratio λ and the collective pitch angle Φ. The results appear reasonable compared to results given in [20], where a smaller turbine is used. For the NREL5 MW or any of the newer reference turbines data is not available. Furthermore, values obtained through interpolation of the precomputed table were compared to calculated values for 100 tip-speed ratios. The difference was found to be of $\mathcal{O}\left(10^{-5}\right)$ or smaller, while the interpolation is 170 times faster. Figure 4.1 shows that the power coefficient has a maximum between $\lambda = 6$ and $\lambda = 10.0$, depending on the pitch-angle. For $\Phi = 0$, the maximum C_P of 0.522 occurs at $\lambda = 9.23$. However, the thrust coefficient grows with tip-speed ratio for most pitch angles. The thrust influences the wake deficit and lifetime of wind turbines, therefore a reduction in λ below the value maximizing C_P can be advisable.

To validate the greedy controller, the controller curve is computed in the BEM environment. It is shown in Figure 4.2, for a range of values of κ. The choice of κ has little influence on the power in comparison to the wind velocity V_0. However, the tip-speed ratio is greatly influenced by the choice

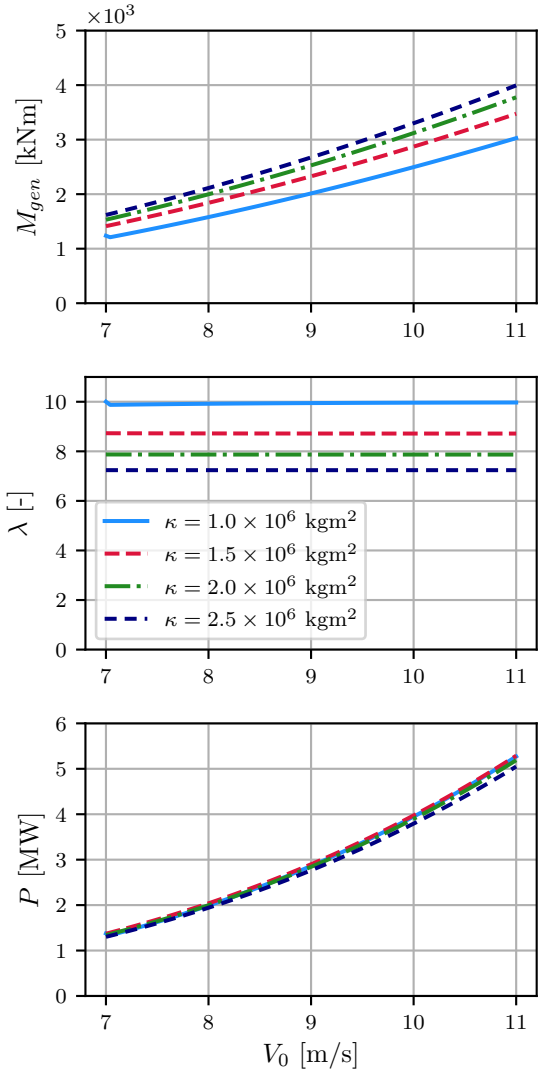

Figure 4.2.: Controller curve of greedy controller for different values of the proportionality constant κ

of κ. As discussed aboved, tip-speed ratio influences C_T and C_P. Thus a higher κ can reduce the thrust exerted on the turbine, with only a small reduction in power. Comparing the controller curve to the curve given in [27], a good agreement in P and M_{gen} is found for $\kappa = 2.5 \times 10^6\,\mathrm{kgm}^2$. However, since the lifetime of the turbine is not of concern, thus κ maximizing C_P, $\kappa = 1.268 \times 10^6\,\mathrm{kgm}^2$, is chosen.

The LBM-ALM with a constant angular velocity of the rotor has been validated by Asmuth et al. in [4]. Therefore only the combination of the greedy controller with the LBM-ALM will be validated here. Figure 4.3 shows M_{gen} over ω for three turbines in a setup as described in subsection 3.2.2. The data gathered from the turbines agrees well with the expected value given by $\kappa\omega^2$. Deviations from the line are due to the turbulent inflow and the inertia of the turbine, but the figure shows,

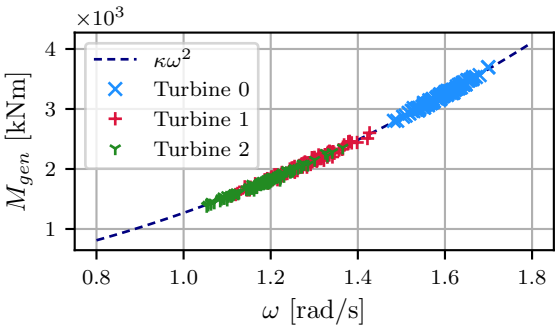

Figure 4.3.: Generator torque over angular velocity of three turbines controlled by greedy controller. Sampled every 200th timestep.

that the turbine can follow the changing conditions closely. Furthermore, the influence of the wake loss can be seen. While the first turbine operates near the values expected at rated wind speed, the second turbine has lower angular velocity and torque and thus generates less power. The third turbine has even lower values, although the difference between the second and third turbine is not as big as between the first and the second.
Thus the environments and the greedy controller exhibit the expected behaviour and the coupling of them is working as well.

4.2. Parameter Study

To gain insight into the behaviour of the RL-agent and the influence of some of the parameters, a parameter study is conducted. To reduce the effect of the random initialization of the networks, for each tested value of a parameter, three independent agents are trained. The evolution of generator torque, angular velocity and power throughout training for a fixed number of timesteps is compared, since these values are used as action, state and reward, respectively.
First the influence of the variance of the action, σ_A is studied. The results of the training for agents with variance of 0.01, 0.05 and 0.1 are shown in Figure 4.4. The comparison of the three values shows a clear trend. A higher variance leads to a faster increase in power. This can be expected, since a larger vairance in the action leads to trying a wider range of actions. Therefore finding a coarse estimate of a beneficial behaviour, that is behaviour that increases the advantage, is more likely. Furthermore it is visible, that the two agents with a higher variance reach similar behaviour. This increases the confidence, that this behaviour represents a local optimum. However, they both reach a maximum of generated power in the first half of training, but are not able to sustain that value throughout the rest of the training. Comparing the different agents with the same set of parameters show that a certain stochasticity is present in training of the agents. After around two thirds of

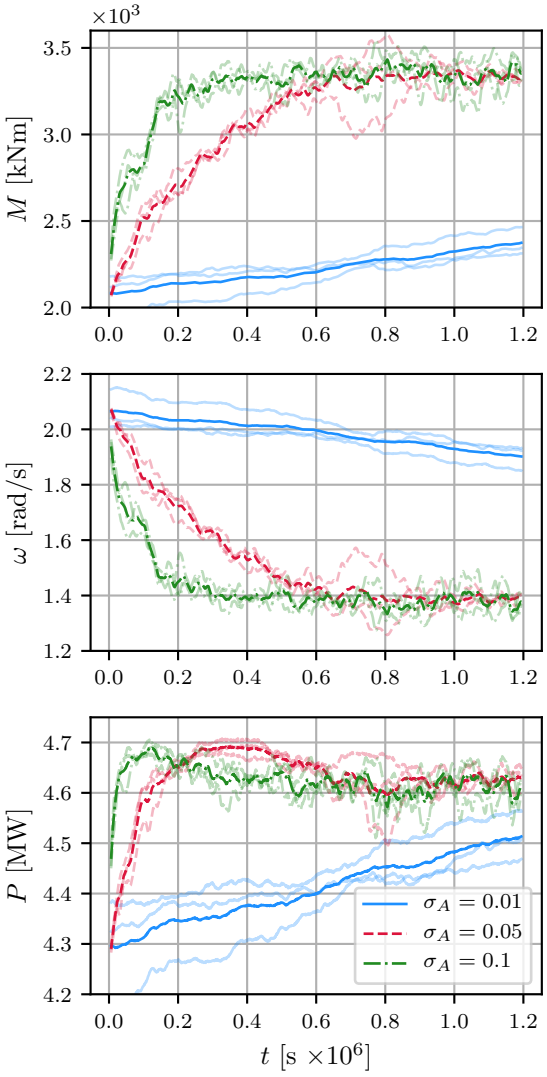

Figure 4.4.: Sensitivity to σ_A. Thin lines are individual runs, thick lines are the average of all runs.

training time, one of agents with medium variance shows a significant drop in power, while another performs significantly better than average. However, towards the end of the training, the agents with medium variance perform more similar, more stable and slightly better than the agent with a higher variance. It can therefore be assumed that the choice of variance has to be a balance between a fast increase in the beginning and unstable behaviour in the end. It can be expected that the training of the agent controlling three turbines will take significantly more timesteps and the computational cost of a single timestep is also much higher. Therefore, a faster increase is favored and a the variance is chosen to be $\sigma_A = 0.1$.

Next, the influence of an l_2 regularization of the policy network is examined. The results of the trainings without regularization and with regularization coefficients of 0.1 and 0.2 are shown in Fig-

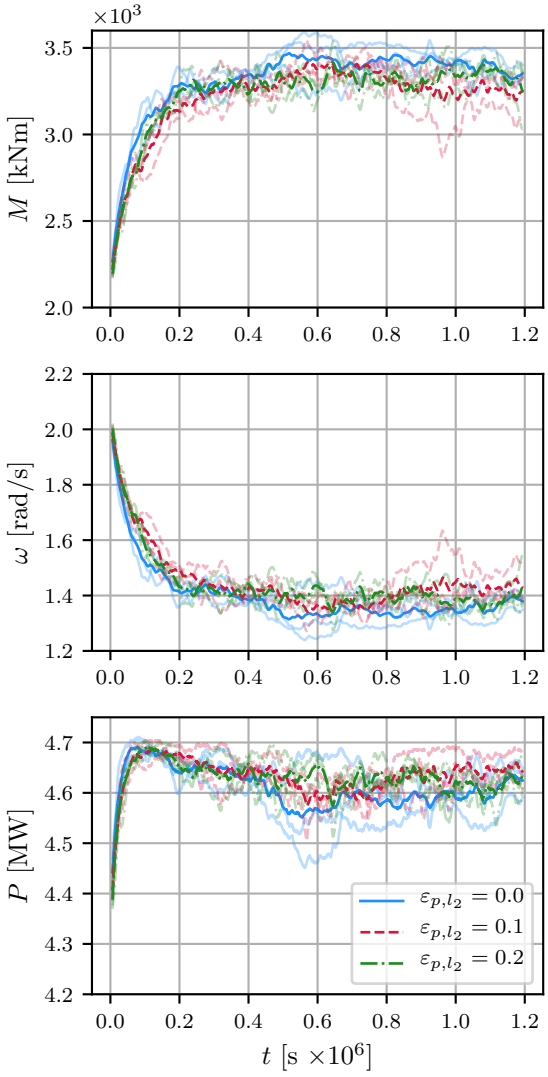

Figure 4.5.: Sensitivity to ε_{p,l_2}. Thin lines are individual runs, thick lines are the average of all runs.

ure 4.5. The sensitivity is significantly smaller to this parameter than to an increase in σ_A. While almost no difference is visible in the beginning for trainings with regularization, no regularization allows for a faster increase in power. However, on average, this agents drop the most afterwards. The agents also differ the most in power towards the end of the training. Remarkably, this is not the case for the generator torque, which differs most for $\varepsilon = 0.1$. To explain this discrepancy in action and reward, looking back at Figure 4.1 shows, that C_P is not very sensitive to changes in tip-speed ratio near the optimum. The optimum tip-speed ratio corresponds to an angular velocity of $\omega = 1.58\,\mathrm{m/s}$. Figure 4.5 shows that most of the agents operate at angular velocities below the optimum. The aforementioned agent sets a lower generator torque, therefore the angular velocity increases and is actually closer to the optimum. The difference in the no regularization case is

caused by an increased torque, which has a bigger effect on C_P. In the last third of training, the agents with $\varepsilon_{p,l_2} = 0.1$ consistently perform best. Therefore this value is chosen.

Finally, the regularization of the value network is tested as well. The same coefficients as for the

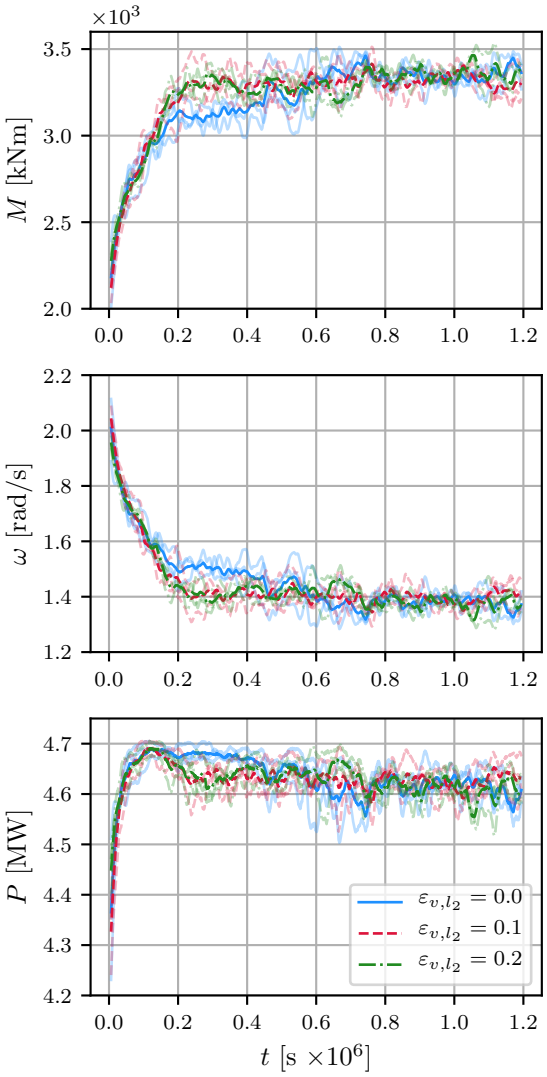

Figure 4.6.: Sensitivity to ε_{v,l_2}. Thin lines are individual runs, thick lines are the average of all runs.

regularization of the policy network are tested. The sensitivity to a change in this parameter is similar to that of the policy regularization. A noticable difference between the trainings is a delayed drop for trainings without regularization. However, this is only a delay and later on the reductions in power are of similar magnitude as for the other cases, while the differences in generator torque are even larger than for the other agents. The second half of the training shows no clear trend, for the best performance. The differences are small and the none of the averaged values is consistently better than the other. Since none of the tested values offers a clear advantage, the same value as

for the regularization of the policy network is chosen.
The parameter study revealed a high sensitivity to a change in variance of the action and showed that an l_2 regularization benefits stability of the trained network while too much influence of the regularization might inhibit optimization. In general, all of the studied trainings showed that the initial value of the generator torque is significantly lower than the optimal value. This is done by design, since a initial value that is too high leads to slowing down the turbine and a breakdown of M_{aero}, which ultimately makes a restart of the turbine necessary. Furthermore, almost all cases showed that P reached a maximum value and then decresead. This was caused by a generator torque that is consistently set higher than what would be expected from a comparison with the controller curve of the greedy controller in Figure 4.2. It can also already noted that a lot of simulated time is necessary for the training of the agent. It can be expected that the necessary time will be at least similar, probably more for the training of an agent controlling three turbines.

4.3. Optimizing Parameters of a Dynamic Behvaviour

4.4. Learning a New Behaviour

5. Conclusion

Bibliography

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] M. Abkar, A. Sharifi, and F. Porté-Agel. Wake flow in a wind farm during a diurnal cycle. *Journal of Turbulence*, 17(4):420–441, 2016.

[3] H. Asmuth, H. Olivares-Espinosa, and S. Ivanell. Actuator line simulations of wind turbine wakes using the lattice Boltzmann method. *Wind Energy Science*, 5(2):623–645, 2020.

[4] H. Asmuth et al. The Actuator Line Model in Lattice Boltzmann Frameworks: Numerical Sensitivity and Computational Performance. *Journal of Physics: Conference Series*, 1256:012022, 2019.

[5] V. Belus et al. Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film. *AIP Advances*, 9(12):125014, 2019.

[6] S. Boersma et al. A tutorial on control-oriented modeling and control of wind farms. In *2017 American Control Conference (ACC)*. 2017 American Control Conference (ACC), pages 1–18, 2017.

[7] S.-P. Breton et al. A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2091):20160097, 2017.

[8] C. Coreixas, B. Chopard, and J. Latt. Comprehensive comparison of collision models in the lattice Boltzmann framework: Theoretical investigations. *Physical Review E*, 100(3):033305, 2019.

[9] H. B. Demuth et al. *Neural Network Design*. Martin Hagan, Stillwater, OK, USA, 2nd edition, 2014.

[10] J. A. Frederik et al. The helix approach: Using dynamic individual pitch control to enhance wake mixing in wind farms. *Wind Energy*, 2020.

[11] E. Gaertner et al. IEA Wind TCP Task 37: Definition of the IEA 15-Megawatt Offshore Reference Wind Turbine. NREL/TP-5000-75698, 1603478, National Renewable Energy Laboratory, 2020.

[12] P. Garnier et al. A review on Deep Reinforcement Learning for Fluid Mechanics, 2019.

[13] M. Gehrke, C. F. Janßen, and T. Rung. Scrutinizing lattice Boltzmann methods for direct numerical simulations of turbulent channel flows. *Computers & Fluids*. Ninth International Conference on Computational Fluid Dynamics (ICCFD9), 156:247–263, 2017.

[14] M. Geier and A. Pasquali. Fourth order Galilean invariance for the lattice Boltzmann method. *Computers & Fluids*, 166:139–151, 2018.

[15] M. Geier, A. Pasquali, and M. Schönherr. Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion part I: Derivation and validation. *Journal of Computational Physics*, 348:862–888, 2017.

[16] M. Geier et al. The cumulant lattice Boltzmann equation in three dimensions: Theory and validation. *Computers & Mathematics with Applications*, 70(4):507–547, 2015.

[17] S. Ghosh et al. Contextual LSTM (CLSTM) models for Large scale NLP tasks, 2016.

[18] J. P. Goit and J. Meyers. Optimal control of energy extraction in wind-farm boundary layers. *Journal of Fluid Mechanics*, 768:5–50, 2015.

[19] S. Guadarrama et al. TF-Agents: A library for reinforcement learning in TensorFlow, 2018.

[20] M. O. L. Hansen. *Aerodynamics of Wind Turbines*. Earthscan, London ; Sterling, VA, 2nd ed edition, 2008. 181 pages.

[21] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[22] L. Hockstad and L. Hanel. Inventory of U.S. Greenhouse Gas Emissions and Sinks. EPA 430-R-20-002, Environmental Protection Agency, 2018.

[23] O. Hoegh-Guldberg et al. Impacts of 1.5°C of Global Warming on Natural and Human Systems, IPCC, 2019.

[24] A. A. M. Holtslag et al. Stable Atmospheric Boundary Layers and Diurnal Cycles: Challenges for Weather and Climate Models. *Bulletin of the American Meteorological Society*, 94(11):1691–1706, 2013.

[25] International Energy Agency. *Global Energy Review 2020: The Impacts of the Covid-19 Crisis on Global Energy Demand and CO2 Emissions*. OECD, 2020.

[26] C. F. Janßen et al. Validation of the GPU-Accelerated CFD Solver ELBE for Free Surface Flow Problems in Civil and Environmental Engineering. *Computation*, 3(3):354–385, 3, 2015.

[27] J. Jonkman et al. Definition of a 5-MW Reference Wind Turbine for Offshore System Development. NREL/TP-500-38060, National Renewable Energy Laboratory, 2009.

[28] J. C. Kaimal and J. J. Finnigan. *Atmospheric Boundary Layer Flows: Their Structure and Measurement*. Oxford University Press, New York, 1994. 289 pages.

[29] S. K. Kang and Y. A. Hassan. The effect of lattice models within the lattice Boltzmann method in the simulation of wall-bounded turbulent flows. *Journal of Computational Physics*, 232(1):100–117, 2013.

[30] A. C. Kheirabadi and R. Nagamune. A quantitative review of wind farm control with the objective of wind farm power maximization. *Journal of Wind Engineering and Industrial Aerodynamics*, 192:45–73, 2019.

[31] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2017.

[32] T. Krüger et al. *The Lattice Boltzmann Method: Principles and Practice*. Graduate Texts in Physics. Springer International Publishing, Cham, 2017.

[33] K. Kutscher, M. Geier, and M. Krafczyk. Multiscale simulation of turbulent flow interacting with porous media based on a massively parallel implementation of the cumulant lattice Boltzmann method. *Computers & Fluids*, 193:103733, 2019.

[34] J. Mann. Wind field simulation. *Probabilistic Engineering Mechanics*, 13(4):269–282, 1998.

[35] A. R. Meyer Forsting, G. R. Pirrung, and N. Ramos-García. A vortex-based tip/smearing correction for the actuator line. *Wind Energy Science*, 4(2):369–383, 2019.

[36] W. Munters and J. Meyers. Towards practical dynamic induction control of wind farms: analysis of optimally controlled wind-farm boundary layers and sinusoidal induction control of first-row turbines. *Wind Energy Science*, 3(1):409–425, 2018.

[37] E. Örtl. *Entwicklung der spezifischen Kohlendioxid-Emissionen des deutschen Strommix in den Jahren 1990 - 2019*. Umweltbundesamt.

[38] G. R. Pirrung et al. A coupled near and far wake model for wind turbine aerodynamics. *Wind Energy*, 19(11):2053–2069, 2016.

[39] J. Rabault and A. Kuhnle. Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach. *Physics of Fluids*, 31(9):094105, 2019.

[40] J. Rabault et al. Deep Reinforcement Learning achieves flow control of the 2D Karman Vortex Street, 2018.

[41] K. Rohrig et al. Powering the 21st century by wind energy—Options, facts, figures. *Applied Physics Reviews*, 6(3):031303, 2019.

[42] M. Schönherr. *Towards Reliable LES-CFD Computations Based on Advanced LBM Models Utilizing (Multi-) GPGPU Hardware*. PhD Thesis, TU Braunschweig, Institut für rechnergestützte Modellierung im Bauingenieurswesen, 2015.

[43] J. Schulman et al. Trust Region Policy Optimization. In *International Conference on Machine Learning*. International Conference on Machine Learning, pages 1889–1897, 2015.

[44] J. Schulman et al. Proximal Policy Optimization Algorithms, 2017.

[45] D. Silver et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, 2017.

[46] J. N. Sørensen and W. Z. Shen. Numerical Modeling of Wind Turbine Wakes. *Journal of Fluids Engineering*, 124(2):393–399, 2002.

[47] J. N. Sørensen. *General Momentum Theory for Horizontal Axis Wind Turbines*, volume 4 of *Research Topics in Wind Energy*. Springer International Publishing, Cham, 2016.

[48] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. 526 pages.

[49] S. Verma, G. Novati, and P. Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854, 2018.

[50] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

A. The Cumulant Lattice Boltzmann Method

A.1. Derivation of cumulants

To provide a more detailed explanation of the cumulant LBM, a more thorough derivation is given here. Note that in the following, the linear index i of the discrete populations is split into three indeces i, j, k representing the lattice directions, running from -1 to 1. First, the discrete distribution is written in a continuous form and transformed to frequency space, making it indepent of the frame of reference:

$$f(\boldsymbol{\xi}) = \sum_{i,j,k} f_{i,j,k}\delta(ic_s - \xi)\delta(jc_s - \upsilon)\delta(kc_s - \zeta) \tag{A.1}$$

$$F(\boldsymbol{\Xi}) = \int_{-\infty}^{\infty} f(\boldsymbol{\xi}) e^{-\boldsymbol{\Xi}\cdot\boldsymbol{\xi}} \mathrm{d}\boldsymbol{\xi}, \tag{A.2}$$

with $\boldsymbol{\Xi} = [\Xi, \Upsilon, \mathrm{Z}]^T$ as the frequency vector. This distribution is then used in the cumulant generating function:

$$C_{\alpha,\beta,\gamma} = c_s^{-(\alpha+\beta+\gamma)} \left.\frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial \mathrm{Z}^\gamma} \ln F(\Xi, \Upsilon, \mathrm{Z})\right|_{\Xi=\Upsilon=\mathrm{Z}=0}. \tag{A.3}$$

However, from that form, the cumulants are not directly computable. Comparing the cumulant generating function to the moment generating function

$$M_{\alpha,\beta,\gamma} = c_s^{-(\alpha+\beta+\gamma)} \left.\frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial \mathrm{Z}^\gamma} F(\Xi, \Upsilon, \mathrm{Z})\right|_{\Xi=\Upsilon=\mathrm{Z}=0} = \sum_{i,j,k} i^\alpha j^\beta k^\gamma f_{i,j,k} \tag{A.4}$$

shows their similarity. Computing the cumulants of an arbitrary function shows how they can be expressed in terms of moments, two examples are given here:

$$C_{200} = \frac{M_{200}}{M_{000}} - \frac{M_{100}^2}{M_{000}^2} \tag{A.5}$$

$$C_{110} = \frac{M_{110}}{M_{000}} - \frac{M_{100} M_{010}}{M_{000}^2}. \tag{A.6}$$

Cumulants can also be computed from central moments, which is requires less computations [16]. Cumulants of zeroth and first order stay constant throughout the collision. The cumulants upwards from order two can now be relaxed indivdually according to

$$C^*_{\alpha,\beta,\gamma} = (1 - \hat{\omega}_{\alpha,\beta,\gamma})\, C_{\alpha,\beta,\gamma}, \tag{A.7}$$

therefore each has its own relaxation frequency. The first frequency $\hat{\omega}_1$ governs the shear viscosity and $\hat{\omega}_2$ the bulk viscosity of the fluid by the same relation as (2.51). The rest of the frequencies can be chosen freely and are usually set to one, resulting fully relaxing the cumulants, however a parametrized version of the collision operation exists, that changes some of the higher order relaxation frequencies, leading to fourth order accurate diffusion in a fully resolved simulation [15]. In underresolved simulations the parameter introduced in the parametrization can be used to influence the diffusivity and therefore acting as an implicit subgrid scale model, however so far only empirical evidence exists for this.[16] The density, velocity and the terms of velocity gradient tensor can be identified with cumulants:

$$\delta\rho = M_{000} = C_{000} \tag{A.8}$$

$$u = \frac{M_{100}}{M_{000}} = C_{100} \tag{A.9}$$

$$D_x v + D_y u = -3\hat{\omega}_1 C_{110} = k_{xy} \tag{A.10}$$

$$D_x u = -\frac{\hat{\omega}_1}{2\rho}\left(2C_{200} - C_{020} - C_{002}\right) - \frac{\hat{\omega}_2}{2\rho}\left(C_{200} + C_{020} + C_{002} - \delta\rho\right) \tag{A.11}$$

$$D_x u - D_y v = -\frac{3\hat{\omega}_1}{2}\left(C_{200} - C_{020}\right) = k_{xx-yy}. \tag{A.12}$$

Note, that in this derivation the well-conditioned cumulant is used as described in the appendix of [16] and therefore the zeroth order cumulant is not the density but its deviation from unit density $\delta\rho$. Also the additional moments k_{xy} and k_{xx-yy} are introduced for use in section A.2.

A.2. Refinement

As was stated before, LBM is usually used on a uniform grid. However, often a variation in grid spacing is desirable, as coarser grids need less memory and have a coarser timestep, whereas a finer grid is able to resolve finer turbulent structures. To balance these two interests, the domain can be partitioned into blocks with different levels of refinement, as areas of high turbulence, which need high resolution, usually occur only in known parts of the domain. The question is now how to treat the borders of the blocks. In [42] the compact interpolation scheme is proposed that is second order accurate, therefore it is consistent with the order of accuracy of the LBM in general. However, the derivation given was found to be neither exhaustive, nor fully correct, as there seem to be misprints, making it difficult to follow. Therefore a more detailed derivation is given here, that is based on [42] as well as [33], which gave a version for an incompressible form of LBM, but suffers from similar issues. In each coarse timestep, a layer of coarse nodes is interpolated from fine nodes and two layers of fine nodes are interpolated from coarse nodes. The layers overlap, so that at no point in time invalid populations are propagated outside the overlap [42]. The node

being interpolated is called the receiver node while the nodes from which is interpolated is referred to as the donor node. Each receiver node has eight donor nodes. A local coordinate system in the center of cube is defined, with the donor nodes laying in the corners at $x = -0.5, 0.5$, $y = -0.5, 0.5$ and $z = -0.5, 0.5$.
First, a interpolation function for the density, $\delta\hat{\rho}$ and the velocities $\hat{u}$, $\hat{v}$ and $\hat{w}$ is defined. Note that the density only requires to be first order accurate:

$$\delta\hat{\rho} = d_0 + d_x x + d_y y + d_z z + d_{xy} xy + d_{xz} xz + d_{yz} yz + d_{xyz} xyz \tag{A.13}$$

$$\hat{u} = a_0 + a_x x + a_y y + a_z z + a_{xx} x^2 + a_{xy} xy + a_{xz} xz + a_{yy} y^2 + a_{yz} yz + a_{zz} z^2 + a_{xyz} xyz \tag{A.14}$$

$$\hat{v} = b_0 + b_x x + b_y y + b_z z + b_{xx} x^2 + b_{xy} xy + b_{xz} xz + b_{yy} y^2 + b_{yz} yz + b_{zz} z^2 + b_{xyz} xyz \tag{A.15}$$

$$\hat{w} = c_0 + c_x x + c_y y + c_z z + c_{xx} x^2 + c_{xy} xy + c_{xz} xz + c_{yy} y^2 + c_{yz} yz + c_{zz} z^2 + c_{xyz} xyz \tag{A.16}$$

To evualate the interpolation functions, constraints have to be defined. The velocity and density in the donor node place 32 constraints, since there are eight of each, thus nine more are required for the 41 degrees of freedom. The second derivative of velocities in the center of the local coordinate system are chosen as these additional constraints. They can be computed by taking the first order central difference of the terms of the velocity gradient tensor, which will be denoted as $D_x xu$ and so on. However, the terms in the main diagonal of the velocity gradient tensor are lengthy and include $\hat{\omega}_2$. This is somewhat undesirable and it was shown in (A.10) that differences of the terms can be expressed more directly in differences of cumulants. Therefore the remaining nine constraints are chosen like this:

$$\frac{\partial^2}{\partial x^2}\hat{v} + \frac{\partial^2}{\partial y \partial x}\hat{u} = D_{xx} v + D_{xy} u \tag{A.17}$$

$$2\frac{\partial^2}{\partial x^2}\hat{u} - \frac{\partial^2}{\partial x \partial y}\hat{v} - \frac{\partial^2}{\partial x \partial z}\hat{w} = 2D_{xx} u - D_{xy} v - D_{xz} w \tag{A.18}$$

$$2\frac{\partial^2}{\partial y^2}\hat{v} - \frac{\partial^2}{\partial x \partial y}\hat{u} - \frac{\partial^2}{\partial y \partial z}\hat{w} = 2D_{yy} v - D_{xy} u - D_{yz} w \tag{A.19}$$

$$2\frac{\partial^2}{\partial z^2}\hat{w} - \frac{\partial^2}{\partial y \partial z}\hat{v} - \frac{\partial^2}{\partial x \partial z}\hat{u} = 2D_{zz} w - D_{yz} v - D_{xz} u, \tag{A.20}$$

the missing five constraints by the off diagonal can easily be extrapolated.
Thus a system of linear equations is established that can be solved for the coefficients of the inter-

polation functions:

$$a_0 = \frac{1}{32}\sum_{x,y,z} -4D_{xx}u - 2D_{xy}v - 4D_{yy}u - 2D_{xz}w + -4D_{zz}u + 4u_{xyz} + 4xyv_{xyz} + 4xzw_{xyz} \tag{A.21}$$

$$= \frac{1}{32}\sum_{x,y,z} -x\left(k_{xx-yy} + k_{xx-zz}\right) - 2yk_{xy} - 2zk_{xz} + 4u_{xyz} + 4xyv_{xyz} + 4xzw_{xyz} \tag{A.22}$$

$$a_x = \frac{1}{2}\sum_{x,y,z} xu_{xyz} \tag{A.23}$$

$$a_{xx} = \frac{1}{8}\sum_{x,y,z} x\left(k_{xx-yy} + k_{xx-zz}\right) + 4xyv_{xyz} + 4xzw_{xyz} \tag{A.24}$$

$$a_{yy} = \frac{1}{8}\sum_{x,y,z} yk_{xy} - 4xyv_{xyz} \tag{A.25}$$

$$a_{xy} = 2\sum_{x,y,z} xyu_{xyz} \tag{A.26}$$

$$a_{xyz} = 8\sum_{x,y,z} xyzu_{xyz} \tag{A.27}$$

$$d_0 = \frac{1}{8}\sum_{x,y,z} \delta\rho_{xyz}, \tag{A.28}$$

in order to reduce the number of equations only unique descriptions are given, the other coefficients can easily extrapolated. Furthermore note that the coefficients of $\delta\hat{\rho}$ are the same as for the other interpolation functions with the exception of d_0. Thus the velocities are computed to second order. To arrive at equations for the second order cumulants, (A.10) - (A.12) are solved for cumulants. A term including the divergence of the velocity is neglected, since LBM is valid in a weakly compressible range and the term is thus much smaller.To compute the cumulants, the derivatves of $\hat{u}$, $\hat{v}$ and $\hat{w}$ are used and the constant terms in the derivatives are replaced with averaged values of second order moments. Also $\hat{\omega}_1$ has to be scaled with a factor σ_{rf} to account for the change in

refinement, it is 2 when scaling from fine to coarse and $1/2$ when scaling from coarse to fine:

$$A_{011} = \frac{\partial \hat{v}}{\partial z} + \frac{\partial \hat{w}}{\partial y} - b_z - c_y \tag{A.29}$$

$$= b_{xz}x + b_{yz}y + 2b_{zz}z + b_{xyz}xy + c_{xy}x + 2c_{yy}y + c_{yz}z + c_{xyz}xz \tag{A.30}$$

$$A_{101} = \frac{\partial \hat{u}}{\partial z} + \frac{\partial \hat{w}}{\partial x} - a_z - c_x \tag{A.31}$$

$$= a_{xz}x + a_{yz}y + 2a_{zz}z + a_{xyz}xy + 2c_{xx}x + c_{xy}y + c_{xz}z + c_{xyz}yz \tag{A.32}$$

$$A_{110} = \frac{\partial \hat{u}}{\partial y} + \frac{\partial \hat{v}}{\partial x} - a_y - b_x \tag{A.33}$$

$$= a_{xy}x + 2a_{yy}y + a_{yz}z + a_{xyz}xz + 2b_{xx}x + b_{xy}y + b_{xz}z + b_{xyz}yz \tag{A.34}$$

$$B = \frac{\partial \hat{u}}{\partial x} - \frac{\partial \hat{v}}{\partial y} - a_x + b_y \tag{A.35}$$

$$= 2a_{xx}x + a_{xy}y + a_{xz}z + a_{xyz}yz - b_{xy}x - 2b_{yy}y - b_{yz}z - b_{xyz}xz \tag{A.36}$$

$$C = \frac{\partial \hat{u}}{\partial x} - \frac{\partial \hat{w}}{\partial z} - a_x + c_z \tag{A.37}$$

$$= 2a_{xx}x + a_{xy}y + a_{xz}z + a_{xyz}yz - c_{xz}x - c_{yz}y - 2c_{zz}y - c_{xyz}xy \tag{A.38}$$

$$C_{011} = -\frac{\sigma_{rf}\rho}{3\hat{\omega}_d}\left(\overline{k_{yz}} + A_{011}\right) \tag{A.39}$$

$$C_{101} = -\frac{\sigma_{rf}\rho}{3\hat{\omega}_d}\left(\overline{k_{xz}} + A_{101}\right) \tag{A.40}$$

$$C_{110} = -\frac{\sigma_{rf}\rho}{3\hat{\omega}_d}\left(\overline{k_{xy}} + A_{110}\right) \tag{A.41}$$

$$C_{200} = \frac{\delta\rho}{9} - \frac{2\sigma_{rf}\rho}{9\hat{\omega}_d}\left(\overline{k_{xx-yy}} + B + \overline{k_{xx-zz}} + C\right) \tag{A.42}$$

$$C_{020} = \frac{\delta\rho}{9} - \frac{2\sigma_{rf}\rho}{9\hat{\omega}_d}\left(-2\left(\overline{k_{xx-yy}} + B\right) + \overline{k_{xx-zz}} + C\right) \tag{A.43}$$

$$C_{002} = \frac{\delta\rho}{9} - \frac{2\sigma_{rf}\rho}{9\hat{\omega}_d}\left(\overline{k_{xx-yy}} + B - 2\left(\overline{k_{xx-zz}} + C\right)\right). \tag{A.44}$$

Now the cumulants can be transformed back to distributions, with the central moments up to order two and cumulants of order higher than two set to zero.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag der Professur für Strömungsmechanik eingereichte Diplomarbeit zum Thema

Kopplung eines künstlichen neuronalen Netzwerks mit LES-LBM zur Verbesserung einer Windpark-Steuerung

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Dresden, 06. Januar 2020

Henry Korb