TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen, Institut für Strömungsmechanik, Professur für Strömungsmechanik

Diplomarbeit

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

vorgelegt zur Erlangung des akademischen Grades '"Diplomingenieur"'

Henry Torsten Korb

geboren am 02. März 1996

in Ratingen

eingereicht am 00. Monat 2020

1. Gutachter Prof. Dr.-Ing. habil. J. Fröhlich

2. Gutachter Dipl.-Ing. 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
$\mathbf{c}$	$\mathrm{m/s}$	Constant Velocity vector
c	$\mathrm{m/s}$	Constant Velocity
f	$\mathrm{kgs^3/m^6}$	Distribution function
E	$\mathrm{J/m^3}$	Energy Density
H	m	Channel half-height
Ma	-	Mach Number
p	Pa	Pressure
R	$\mathrm{J/kg/K}$	Specific gas Constant
Re	-	Reynolds Number
t	s	Time
T	K	Temperature
$\mathbf{u}$	$\mathrm{m/s}$	Macroscopic Velocity Vector
u	$\mathrm{m/s}$	Velocity in streamwise Direction
v	$\mathrm{m/s}$	Velocity in wallnormal Direction
w	$\mathrm{m/s}$	Velocity in spanwise Direction
$\mathbf{x}$	m	Vector of position
x	m	Coordinate in streamwise Direction
y	m	Coordinate in wallnormal Direction
z	m	Coordinate in spanwise Direction

Greek Symbols	Unit	Description
κ	-	Adiabatic Index
μ	$\mathrm{kg/m/s}$	Dynamic Viscosity
ν	$\mathrm{m^2/s}$	kinematic Viscosity
$\boldsymbol{\xi}$	$\mathrm{m/s}$	Microscopic Velocity
ρ	$\mathrm{kg/m^3}$	Density
Ω	$\mathrm{kgs^2/m^6}$	Collision Operator

Indices	Description
τ	Friction
b	Bulk
cp	Centerplane
f	Fluid
m	Mean
pwm	Plane-wise mean
rms	Root-mean square
s	Solid, Sound
w	Wall

Additional Symbols	Description
$\|\mathbf{a}\|$	Euclidian Norm
∇	Nabla Operator
Δ	Step
$\mathbf{a} \cdot \mathbf{b}$	Scalar Product, Matrix multiplication
a'	Fluctuation

Abbreviations	Description
BGK	Bhatnagar-Gross-Krook
DNS	Direct numerical Simulation
LBM	Lattice-Boltzmann-Method
LBE	Lattice-Boltzmann Equation
LES	Large-Eddy Simulation
MRT	Multiple Relaxation Times Operator
NSE	Navier-Stokes-Equations
pdf	Particle-distribution Function

1. Theoretical Background

1.1. Wind Turbines

1.1.1. Description of a Turbine and Flow Features

First a few basic definitions of the model used for windturbines will be given. Todays mainstream windturbines 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 hub height H is defined as the distance from the ground to the axis of rotation of the rotor and is approximately the same as the rotor diameter D. 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 lead to a steady increase in turbine sizes and nominal capacity [35]. In the academic world theoretical reference turbines such as the NREL 5 MW wind turbine [22] 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.[18]
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 [23] and [20]. Behind the turbine exists a wake, characterized by an increase of turbulence intensity and a decrease of average velocity by $50\,\%$ and more [2]. More details on wakes can be found in [5]. To study the physical phenomena connected to wind turbines, models of the turbines and the airflow have been developed.

1.1.2. Blade Element / Momentum Theory

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 inegral forms of conservation of mass, momenta and energy, as shown in (1.1) through (1.4), respectively. 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 $\mathrm{d}\mathbf{A}$ is the area vector pointing outwards of the control volume. T is the force acting on the rotor in streamwise direction and M_{aero} the torque and P the power

extracted by the rotor.

$$\oint_{\partial V} \rho \mathbf{u} \cdot \mathrm{d}\mathbf{A} = 0 \tag{1.1}$$

$$\oint_{\partial V} \rho u_x \mathbf{u} \cdot \mathrm{d}\mathbf{A} = T - \oint_{\partial V} p \mathrm{d}\mathbf{A} \cdot \mathbf{e}_x \tag{1.2}$$

$$\oint_{\partial V} \rho r u_\theta \mathbf{u} \cdot \mathrm{d}\mathbf{A} = M_{aero} \tag{1.3}$$

$$\oint_{\partial V} \left(p + \frac{1}{2}\rho \|\mathbf{u}\|^2 \right) \mathbf{u} \cdot \mathrm{d}\mathbf{A} = P \tag{1.4}$$

Furthermore the dimensionless tip speed ratio is denoted as λ, it is defined by the angular velocity of the rotor, ω, its radius R and the mean wind speed V_0. [40, p. 7 - 8]

$$\lambda = \frac{\omega R}{V_0} \tag{1.5}$$

$$C_T = \frac{T}{\frac{1}{2}\rho A V_0^2} \tag{1.6}$$

$$C_P = \frac{P}{\frac{1}{2}\rho A V_0^3} \tag{1.7}$$

Applying these equations to a controll volume enclosed by the stream tube around the rotor disc under the assumption that $u_x = u_R = const$ in the rotor disc, yields the basic relationships for the thrust (1.9) and power (1.10). The axial interference factor a is a measure for the influence of the rotor disc on the velocity in the stream tube. From (1.10) it is possible to find a maximum power coefficient, which is given in (1.11) 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. [40, p.10 - 11] Sørenensen gives an assessment of the assumptions made in [40].

$$a = 1 - \frac{u_R}{V_0} \tag{1.8}$$

$$T = 2\rho A V_0^2 a(1-a), \quad C_T = 4a(1-a) \tag{1.9}$$

$$P = 2\rho A V_0^3 a(1-a)^2, \quad C_P = 4a(1-a)^2 \tag{1.10}$$

$$C_{P,max} = \frac{16}{27} \approx 59.3\%, \quad a = \frac{1}{3} \tag{1.11}$$

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 1.1. 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 is the induced velocity defined as $\mathbf{w}_i = [-aV_0, a'\omega r]$, with axial and tangential induction factor a and a'. The sum of these velocities is the relative velocity $\mathbf{V}_{rel}$.

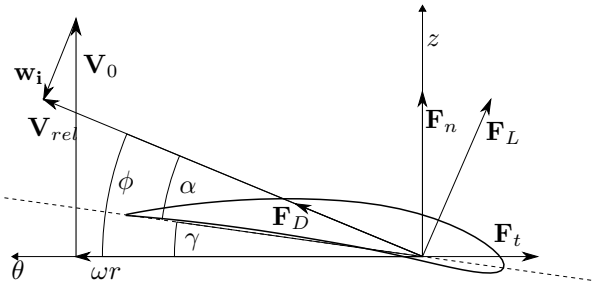

Figure 1.1.: Schematic of the local forces and velocities at a blade element.

The thrust and torque on the rotor within an infinitesimally thick stream tube are calculated according to (1.12) and (1.13). The force coefficients 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 can be found through the difference of the angle between rotor plane and $\mathbf{V}_{rel}$, denoted as ϕ and the local pitch γ.

$$\frac{\mathrm{d}T}{\mathrm{d}r} = N_b F_n = \frac{1}{2} N_b \rho c \|\mathbf{V}_{rel}\|^2 C_n \tag{1.12}$$

$$\frac{\mathrm{d}M_{aero}}{\mathrm{d}r} = N_b r F_t = \frac{1}{2} N_b \rho c \|r\mathbf{V}_{rel}\|^2 C_n \tag{1.13}$$

Combining (1.12) and (1.13) with the thrust and torque found by applying momentum theory to the same stream tube yields (1.14) and (1.15), with the solidity $\sigma = N_b c/(2\pi r)$. This allows for an iterative computation of the infinitesimal torque and thrust as described in algorithm 1. Integrating these with respect to the radius thus yields the total torque and thrust, T and M_{aero}. In practice the blade is defined as finite sized blade elements and the integration becomes a summation. [40, p. 100 - 103]

$$a = \frac{1}{4\sin(\phi)^2/\left(\sigma C_n\right) + 1} \tag{1.14}$$

$$a' = \frac{1}{4\sin(\phi)\cos(\phi)/\left(\sigma C_t\right) - 1} \tag{1.15}$$

Many corrections for BEM have been found to increase the accuracy of the model, among them 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, therefore Glauert proposed a correction for the calculation of C_t. [40, p. 103 - 104]

Many other corrections for effects caused by the wake have also been proposed, such as a coupled

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

for all elements **do**
  $a \leftarrow 0$
  $a' \leftarrow 0$
  repeat
    $\phi \leftarrow \tan^{-1}((1-a)/(\lambda r/R(1+a')))$
    $\alpha \leftarrow \phi - \gamma$
    compute C_n and C_t from lift and drag tables
    calculate new a and a' from (1.14) and (1.15)
  until a and a' converge
end for
calculate $\int \mathrm{d}T$ and $\int \mathrm{d}M$ according to (1.12) and (1.13)

near and far wake model by Pirrung et al. [32].

1.1.3. Actuator Line Model

To study the influence of the wind turbine on the fluid field, the fluid field has to be resolved by means of CFD. Since fully resolving the shape of the blades would require a prohibitingly fine resolution, the influce of the blade on the fluid is modelled. An overview over the models applied can be found in [6] and [25]. One such model is the actuator line model (ALM). 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 as body forces by applying a convolution with a gaussian filter kernel $\eta(d)$, a definition for a three dimensional kernel is given in (1.16). The parameter ϵ is referred to as the smearing width, since it controls the stretching of the bell curve. [39]

$$\eta(d) = \frac{1}{\epsilon^2 \pi^{3/2}} e^{(-(d/\epsilon)^2)} \tag{1.16}$$

$$\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{N_b} \int_0^R \left(\mathbf{F_n}(r) + \mathbf{F_t}(r)\right) \eta(\|x - r\mathbf{e}_i\|)\mathrm{d}r \tag{1.17}$$

As shown for example by Asmuth et. al [4], the ALM does not account for velocity induced by the root and tip vortices, leading 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 [30].

1.1.4. Wind Turbine Control

To generate power from the aerodynamic torque acting on the rotor a generator applys 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 [22]. [5]
The quantities controllable are the yaw υ, the blade pitch γ_p and the torque of the generator M_{gen}. To maximize the power, the yaw has to be adjusted so the turbine points upwind and the pitch and generator torque have to be adjusted so that $\mathbf{F}_t$ and ω are balanced, since $P = \omega M_{aero} = \omega N_b \int_0^R r\mathbf{F}_t \cdot \mathbf{e}_\theta \mathrm{d}r$. The blades are designed so that the optimal blade pitch is zero and the optimal power only depends on the tip speed ratio. At a given wind speed and constant blade pitch, M_{aero} becomes a function only depending on ω. 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 (1.18). Therefore, controlling the torque in region II according to (1.19) maximizes the generated power, with k 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. [18, p.63 - 77]

$$I\frac{\mathrm{d}\omega}{\mathrm{d}t} = M_{aero} - M_{gen} \tag{1.18}$$

$$M_{gen} = k\omega^2 \tag{1.19}$$

However, this control strategy also maximizes the wake and 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, which tries to lower the power intake of the first turbine, so that the wind speed at the following turbines is higher, and wake redirection control, which aims at steering the wake away from the second turbine. [5] Recently also the development of dynamic approaches has begun [10]. Many different approaches have been taken, from extremum seeking control [7] to physically motivated dynamic steering [10]. The review in [25] shows that theoretical approaches for axial induction like [42] and [31] achieved increases of $20\,\%$ and more, however these methods are not applicable to real turbines.

1.2. Machine Learning for a Continuous Control Problem

1.2.1. 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 that given a state $\mathbf{S}_t$ takes an action $\mathbf{A}_t$, and the environment, that 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 function p that is defined in (1.20). It defines the probability of the 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.

$$p(\mathbf{s}', r|\mathbf{s}, \mathbf{a}) \doteq Pr(\mathbf{S}_t = \mathbf{s}', R_t = r|\mathbf{S}_{t-1} = \mathbf{s}, \mathbf{A}_{t-1} = \mathbf{a}) \tag{1.20}$$

For a process to be a Markov Decision Process, the 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. This allows for a finite computation of G_t. A typical formulation of G_t, referred to as a discounted return is given in (1.21). 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. [41, p. 47- 57]

$$G_t \doteq R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} ... = \sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}, \quad \gamma \in [0, 1] \tag{1.21}$$

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 combina-

tions 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 [41].

The policy π is the mapping from states to actions with a set of adjustable parameters. Under the policy π with its vector of parameters set to $\boldsymbol{\theta}$, the probability of Action $\mathbf{A}_t = \mathbf{a}$ given a state $\mathbf{S}_t = \mathbf{s}$ is denoted as $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s})$. The value function of a state 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 be dropped. For MDPs it can be defined as (1.22).

$$v_\pi \doteq \mathrm{E}_\pi \left[G_t | \mathbf{S}_t = \mathbf{s}\right] = \mathrm{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} | \mathbf{S}_t = \mathbf{s}\right] \tag{1.22}$$

$$= \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \sum_{\mathbf{s}'} \sum_{r} p(\mathbf{s}', r|\mathbf{s}, \mathbf{a}) \left(r + \gamma v_\pi(\mathbf{s}')\right) \tag{1.23}$$

In the form of (1.23), 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 (1.24). [41, p. 58-59]

$$q_\pi(\mathbf{s}, \mathbf{a}) \doteq \mathrm{E}_\pi \left[G_t | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right] = \mathrm{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} | \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}\right] \tag{1.24}$$

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 space. Policy gradient methods use the gradient of a perfomance measure $J(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ of a policy. This gradient can be used in optimization algorithms, such as stochastic gradient descent [41, p. 201] or its extensions such as Adam [26]. In its basic form, SGD performs an update according to 1.25. The parameter α is called the learning rate.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{1.25}$$

The policy gradient methods differ now only in the performance measure. The first such algorithm proposed is the REINFORCE algorithm [43]. Its definition of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is presented in (1.26). It follows from the policy gradient theorem and substituting all values of $\mathbf{A}$ and $\mathbf{S}$ with the actions and states from one trajectory. Thus all the values necessary for the computation of the gradient are known.

[41, p.324-328]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathrm{E}_{\pi}\left[\sum_{a} \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{S}_t) q_{\pi}(\mathbf{S}_t, \mathbf{a}) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{S}_t)}\right] \tag{1.26}$$

$$= \mathrm{E}_{\pi}\left[G_t \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}\right] \tag{1.27}$$

While this algorithm makes a computation possible, it is inefficient. Therefore newer methods have been devised such as the Trust Region Policy Optimization (TRPO) [38] and the Proximal Policy Optimization (ppo) [37]. 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 given in (1.29). The probability ratio r_t compares the policy after the update to the policy before the update. Therefore $\pi_{\boldsymbol{\theta}}$ has to be approximated. $a_{\pi}(\mathbf{A}_t|\mathbf{S}_t)$ is an estimator of the advantage function, which is defined as $a_{\pi}(\mathbf{A}_t|\mathbf{S}_t) = q_{\pi}(\mathbf{A}_t|\mathbf{S}_t) - v_{\pi}(\mathbf{S}_t)$. This estimator also has to be found, however, this is a regular optimization problem, which can be solved via SGD or Adam.

$$r_t(\boldsymbol{\theta}) \doteq \frac{\pi_{\boldsymbol{\theta}}(\mathbf{A}_t|\mathbf{S}_t)}{\pi_{\boldsymbol{\theta}_{old}}(\mathbf{A}_t|\mathbf{S}_t)} \tag{1.28}$$

$$J \doteq \mathrm{E}_{\pi}\left[\min\left(r_t(\boldsymbol{\theta}) a_{\pi}(\mathbf{A}_t|\mathbf{S}_t), \mathrm{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) a_{\pi}(\mathbf{A}_t|\mathbf{S}_t)\right)\right] \tag{1.29}$$

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 the size of $\boldsymbol{\theta}$.

1.2.2. Artificial Neural Networks

Feed-forward networks

In the section above, a way to update parameters of the policy or value function were described, however, no description of the 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 $\S$ neurons takes as input a vector $\mathbf{p}$ of length R. The output of the layer is a new vector $\mathbf{a}^k$, wich then is the input for the next layer of the network. In a simple feed-forward layer a_j^k is computed according to (1.30). The entries $w_{i,j}^k$ of the matrix $\mathbf{W^k}$ are called the weights of the layer and the entries b_j^k of the vector $\mathbf{b}^k$ are called its bias. f is called the activation function, which can be chosen freely, but typical choices include the hyperbolic tangent

Table 1.1.: Activation functions commonly used in neural network

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	$(-\infty, +\infty)$	$(0, -1)$	$\sigma(x) = (1 + e^{-x})^{-1}$	$\sigma(x)\sigma(-x)$
softplus	$(-\infty, +\infty)$	$(0, +\infty)$	$\mathrm{softplus}(x) = \ln(e^x + 1)$	$\sigma(x)$

($\tanh$), 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 1.1. One desirable property for activation functions 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 given in (1.31), with $\mathbf{p}$ being the input to the network and $\mathbf{a}$ its output and the activation function being applied element-wise to the vectors. [9, p. 2-2 - 2-12].

$$a_j^k = f(w_{i,j}^k p_i^k + b_j^k) \tag{1.30}$$

$$\mathbf{a} = \mathbf{tanh}(\mathbf{b^1} + \mathbf{W^1} \cdot \mathbf{tanh}(\mathbf{b^0} + \mathbf{W^0} \cdot \mathbf{p}) \tag{1.31}$$

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 [15] or transient physical processes. This led to the development of recurrent neural networks, in which the network includes has 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 [19]. 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 $\mathbf{a}$ 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 $\mathbf{a}_{t-1}$, as well as the current activation $\mathbf{p}_t$. 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 1.2. 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.

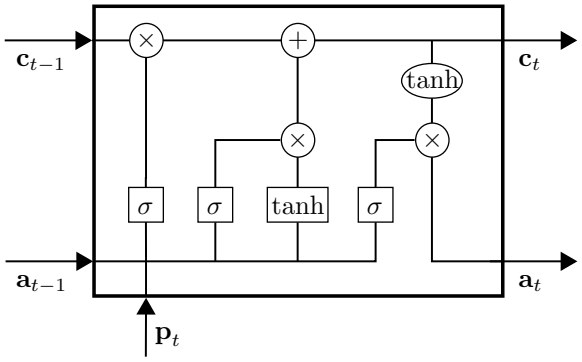

Figure 1.2.: Schematic representation of an LSTM cell, operation in boxes represent layers with trainable weights, operations in ellipses are pointwise operations.

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 funtion j of the gradient of the policy as shown in (1.32). Under the assumption 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 (1.33). Thus the gradient term is (1.34). [41, p. 335]

If μ_i is the output of a K-layered network, it can be written as (1.35). Computing the gradient (1.36) can now be done efficiently via backpropagation. 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 $s_{i,j}^k$ determines how sensible the action is to

changes of this parameter. [9, p.11-7 - 11-13]

$$\nabla_{\boldsymbol{\theta}} J = j\left(\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})}{\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})}\right) \tag{1.32}$$

$$\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S}) = \prod_i pdf(A_i, \mu_i(\mathbf{S}, \boldsymbol{\theta})) \tag{1.33}$$

$$\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})}{\pi_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{S})} = \prod_i \frac{\partial pdf(A_i, \mu_i(\mathbf{S}, \boldsymbol{\theta}))}{\partial \mu_i(\mathbf{S}, \boldsymbol{\theta})} \frac{\nabla_{\boldsymbol{\theta}}\mu_i(\mathbf{S}, \boldsymbol{\theta})}{pdf(A_i, \mu_i(\mathbf{S}, \boldsymbol{\theta}))} \tag{1.34}$$

$$\mu_i(\mathbf{S}, \boldsymbol{\theta}) = f^K(b_i^K + w_{i,j}^K f^{K-1}(...f^0(b_l^0 + w_{l,m}^0 S_m)...)) \tag{1.35}$$

$$\nabla_{\boldsymbol{\theta}}\mu_i|_{\mathbf{S}} = \left[\left.\frac{\partial \mu_i}{\partial b_j^k}\right|_{\mathbf{S}}, \left.\frac{\partial \mu_i}{\partial w_{j,l}^k}\right|_{\mathbf{S}}\right]^T \tag{1.36}$$

$$x_i^k = b_i^k + w_{i,j}^k p_j^k, \quad p_i^{k+1} = f^k(x_i^k), \quad p_i^0 = S_i \tag{1.37}$$

$$s_{i,j}^K = \left.\frac{\partial f^K(x)}{\partial x}\right|_{x_i^K} \delta_{i,j}, \quad s_{i,j}^k = s_{i,l}^{k+1} w_{l,j}^k \left.\frac{\partial f^k(x)}{\partial x}\right|_{x_i^k} \tag{1.38}$$

$$\left.\frac{\partial \mu_i}{\partial b_j^k}\right|_{\mathbf{S}} = s_{i,j}^k, \quad \left.\frac{\partial \mu_i}{\partial w_{j,l}^k}\right|_{\mathbf{S}} = s_{i,j}^k p_l^k \tag{1.39}$$

1.3. The Lattice Boltzmann Method

1.3.1. basics

To conduct the training of the agent, it needs to interact with the environment. 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. The traditional approach would be to discretize the Navier-Stokes-Equations (NSE), however, solvers based on the NSE require many CPU-hours. A newer approach is the Lattice Boltzmann Method. The basics of its theory will be layed out in this chapter.
The Lattice Boltzmann Method (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 function of space $\mathbf{x}$, microscopic velocity $\boldsymbol{\xi}$ and time. However, the macroscopic quantities can easily be recovered from the pdf by taking its zeroth and first moments in velocity space, as shown in (1.40) and (1.41). 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. It can be described by only the macroscopic quantities and is shown in (1.42), with R being

the specific gas constant and T the temperature. The velocity $\mathbf{v}$ is defined as $\mathbf{v} = \boldsymbol{\xi} - \mathbf{u}$.[27, p. 15-21]

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 Equaiton 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 in LBM another formulation for Ω is used, which is an active area of research in the LBM community [8]. The Boltzmann Equation is shown in (1.43). The first form is the total derivative of f with repsect 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. [27, p. 21]

$$\rho(\mathbf{x}, t) = \int_{-\infty}^{\infty} f(\mathbf{x}, \boldsymbol{\xi}, t) \mathrm{d}\boldsymbol{\xi} \tag{1.40}$$

$$\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) = \int_{-\infty}^{\infty} \boldsymbol{\xi} f(\mathbf{x}, \boldsymbol{\xi}, t) \mathrm{d}\boldsymbol{\xi} \tag{1.41}$$

$$f^{eq}(\mathbf{x}, \boldsymbol{\xi}, t) = \rho \left(\frac{1}{2\pi RT} \right) e^{-\|\mathbf{v}\|^2/(2RT)} \tag{1.42}$$

$$\frac{Df(\mathbf{x}, \boldsymbol{\xi}, t)}{Dt} = \frac{\partial f(\mathbf{x}, \boldsymbol{\xi}, t)}{\partial t} + \frac{\partial f(\mathbf{x}, \boldsymbol{\xi}, t)}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial t} + \frac{\partial f(\mathbf{x}, \boldsymbol{\xi}, t)}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial t} = \Omega(f) \tag{1.43}$$

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 is 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 and the roots of the Hermite polynomials up to order three are the necessary discrete velocities $\boldsymbol{\xi}_i$. Together with the weights w_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, but only D3Q27 is suitable for high Reynolds number flows [24].[27, 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 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), which is given in (1.44), with $\mathbf{c}_i$ being $\boldsymbol{\xi}_i/\sqrt{3}$ for convenience. 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 given in (1.45) and (1.46). [27, p. 94-98]

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, \mathbf{c}_i, t + \Delta t) = \Omega(f)_i + f_i(\mathbf{x}_i, \mathbf{c}_i, t) \tag{1.44}$$

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t) \tag{1.45}$$

$$\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) = \sum_i \mathbf{c}_i f_i(\mathbf{x}, t) \tag{1.46}$$

Lastly a collision operator has to be found. The first applicable collision operator proposed was the Bhatnager-Gross-Krook (BGK) operator. 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 ω. This leads to (1.47). Via Chapmann Enskog analysis this relaxation frequency is related to the kinetic viscosity ν by (1.48). [27, p. 98-100, 112]

$$\Omega_i^{BGK} = \omega\left(f_i - f_i^{eq}\right) \tag{1.47}$$

$$\nu = c_s^2\left(\frac{1}{\omega} - \frac{\Delta t}{2}\right) \tag{1.48}$$

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, since these moments are not statistically independent, they cannot be relaxed separately. However, cumulants of a distribution 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. To transform the discrete distributions into cumulant space, the discrete distributions are redefined in terms of continuous velocity and then transformed into frequency space for Galilean invariance. From the transformed distributions the cumulants are generated. These quantities can now be relaxed independently. It can be shown that with a parametrization this method can be fourth order accurate[12]. However in underresolved flows, this parametrization is used to add numerical diffusivity, acting like an implicit subgrid scale model for Large-Eddy-Simulations (LES). 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 one of its weaknesses, as usually it is not straight forward, since 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 [27, 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 [27, p. 522 - 526].

2. Setup of Simulation

2.1. Implementation

2.1.1. Basic Implementation Strategy

In this work the performance of two controllers in two simulation environments is studied. As a baseline case a greedy controller according to the description given above 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 mainly used for rapid development, and an LBM-ALM environment with multiple turbines.
To provide a layer of abstraction between simulation environments and controllers, the so called visitors were introduced. Each turbine and velocity 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 controllable quantities such as M_{gen}. This allows for an easy recombination of simulation environments and controllers.
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.

2.1.2. RL-Controller

The RL-controller relies on the library TF-Agents [17], which implements many RL algorithms and is based on the Python API of TensorFlow [1]. The controller implements an MDP, with an environment, 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 [34].
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 above. The actor network consists of three layers, of which the first two layers are either fully con-

nected lstm cells or regular feed forward layers. The activation function of these layers is $\tanh$. The width of these layers is adjustable. The last layer is a feed forward layer of $\mathrm{softplus}$ nodes. The value network is a three layered feed forward network with a $\mathrm{softplus}$ activation.
The interaction with the visitors is governed by an instance of the class `TurbineEnvironment`.The action provided by the agent is smoothed with an exponential filter to make it continuous. The decay rate α is chosen according to (2.1), with a forget ratio FR of 0.99. $N_{t,A}$ is the number of time steps in the simulation in between actions of the network. The observations are also filtered exponentially, with the same rate as the action and can include 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`.

$$\alpha = FR^{1/N_{t,A}} \tag{2.1}$$

$$x_t = x_{t-1} + \alpha(\hat{x}_t - x_{t-1}) \tag{2.2}$$

2.1.3. Greedy Controller

The greedy controller is an implementation of the baseline generator-torque controller given in [22].

2.1.4. LBM-ALM Environment

The LBM-ALM environment is based on the work of Asmuth et al. [3]. The actuator line model is implemented in elbe, a cumulant LBM code [21]. For this work the code was extended to include a second order accurate refinement scheme according to Schönherr et al. [36], for further information the reader is referred to 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 [33]. A sequence diagramm of a timestep is given in 2.1 to demonstrate the interactions between elbe and the controller and visitors. The inlet is a uniform inflow, that can be superimposed with fluctuations of a Mann box [29]. The outlet uses a sponge layer, as described in section 1.3.

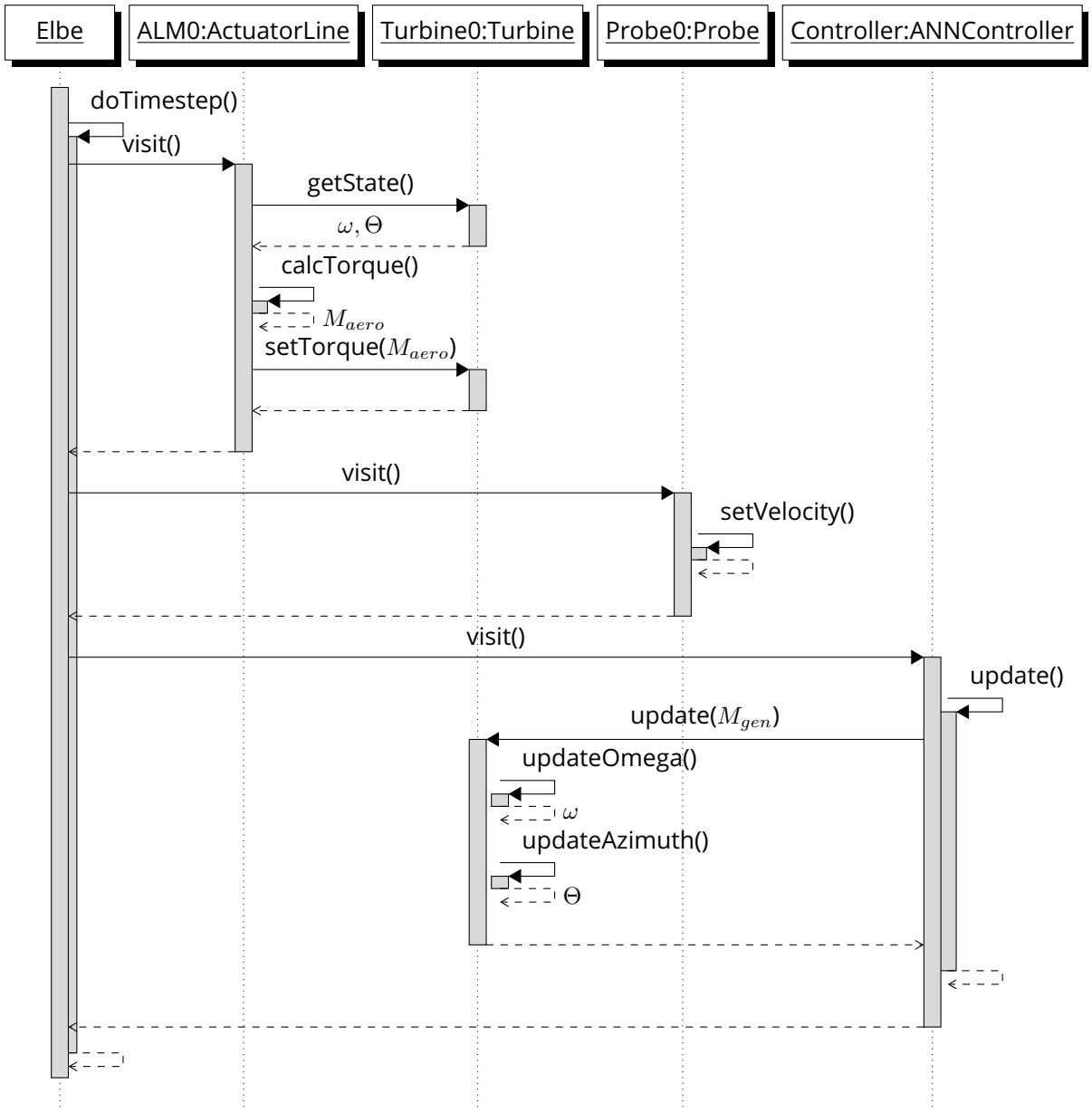

Figure 2.1.: UML sequence diagram of simulation timestep.

2.1.5. 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 4000 values of M_{aero} are precomputed and is then linearly interpolated. The fluid field is composed of a base velocity and an optionally superimposed turbulent fluctuation, which is computed by the synthetic turbulence generator TuGen [16]. 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. The interaction of the BEM environment with the coupling mechanism is designed to be as similar as possible to the interaction between elbe and the controller.

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 (Apr. 2016). _eprint: https://doi.org/10.1080/14685248.2015.1127379, pp. 420–441.

[3] H. Asmuth, H. Olivares-Espinosa, and S. Ivanell. "Actuator Line Simulations of Wind Turbine Wakes Using the Lattice Boltzmann Method". English. *Wind Energy Science Discussions* (Dec. 2019), pp. 1–33.

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

[5] S. Boersma et al. "A Tutorial on Control-Oriented Modeling and Control of Wind Farms". *2017 American Control Conference (ACC)*. May 2017, pp. 1–18.

[6] S.-P. Breton et al. "A Survey of Modelling Methods for High-Fidelity Wind Farm Simulations Using Large Eddy Simulation". en. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 375.2091 (Apr. 2017), p. 20160097.

[7] U. Ciri et al. "Large-Eddy Simulations with Extremum-Seeking Control for Individual Wind Turbine Power Optimization". en. *Wind Energy* 20.9 (2017), pp. 1617–1634.

[8] C. Coreixas, B. Chopard, and J. Latt. "Comprehensive Comparison of Collision Models in the Lattice Boltzmann Framework: Theoretical Investigations". en. *Physical Review E* 100.3 (Sept. 2019), p. 033305.

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

[10] J. A. Frederik et al. "The Helix Approach: Using Dynamic Individual Pitch Control to Enhance Wake Mixing in Wind Farms". *Wind Energy* n/a.n/a (May 2020).

[11] E. Gaertner et al. *IEA Wind TCP Task 37: Definition of the IEA 15-Megawatt Offshore Reference Wind Turbine*. en. Tech. rep. NREL/TP-5000-75698, 1603478. Mar. 2020, NREL/TP–5000–75698, 1603478.

[12] M. Geier and A. Pasquali. "Fourth Order Galilean Invariance for the Lattice Boltzmann Method". en. *Computers & Fluids* 166 (Apr. 2018), pp. 139–151.

[13] M. Geier, A. Pasquali, and M. Schönherr. "Parametrization of the Cumulant Lattice Boltzmann Method for Fourth Order Accurate Diffusion Part I: Derivation and Validation". en. *Journal of Computational Physics* 348 (Nov. 2017), pp. 862–888.

[14] M. Geier et al. "The Cumulant Lattice Boltzmann Equation in Three Dimensions: Theory and Validation". en. *Computers & Mathematics with Applications* 70.4 (Aug. 2015), pp. 507–547.

[15] S. Ghosh et al. “Contextual LSTM (CLSTM) Models for Large Scale NLP Tasks”. en (Feb. 2016).

[16] L. Gilling. “TuGen: Synthetic Turbulence Generator, Manual and User’s Guide”. English (2009).

[17] S. Guadarrama et al. “TF-Agents: A Library for Reinforcement Learning in TensorFlow” (2018).

[18] M. O. L. Hansen. *Aerodynamics of Wind Turbines*. 2nd ed. OCLC: ocm86172940. London ; Sterling, VA: Earthscan, 2008.

[19] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. en. *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780.

[20] A. A. M. Holtslag et al. “Stable Atmospheric Boundary Layers and Diurnal Cycles: Challenges for Weather and Climate Models”. en. *Bulletin of the American Meteorological Society* 94.11 (Nov. 2013), pp. 1691–1706.

[21] C. F. Janßen et al. “Validation of the GPU-Accelerated CFD Solver ELBE for Free Surface Flow Problems in Civil and Environmental Engineering”. en. *Computation* 3.3 (July 2015), pp. 354–385.

[22] J. Jonkman et al. *Definition of a 5-MW Reference Wind Turbine for Offshore System Development*. en. Tech. rep. NREL/TP-500-38060, 947422. Feb. 2009, NREL/TP–500–38060, 947422.

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

[24] 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”. en. *Journal of Computational Physics* 232.1 (Jan. 2013), pp. 100–117.

[25] A. C. Kheirabadi and R. Nagamune. “A Quantitative Review of Wind Farm Control with the Objective of Wind Farm Power Maximization”. en. *Journal of Wind Engineering and Industrial Aerodynamics* 192 (Sept. 2019), pp. 45–73.

[26] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. en. *arXiv:1412.6980 [cs]* (Jan. 2017).

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

[28] 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”. en. *Computers & Fluids* 193 (Oct. 2019), p. 103733.

[29] J. Mann. “Wind Field Simulation”. en. *Probabilistic Engineering Mechanics* 13.4 (Oct. 1998), pp. 269–282.

[30] A. R. Meyer Forsting, G. R. Pirrung, and N. Ramos-García. “A Vortex-Based Tip/Smearing Correction for the Actuator Line”. en. *Wind Energy Science* 4.2 (June 2019), pp. 369–383.

[31] 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”. en. *Wind Energy Science* 3.1 (June 2018), pp. 409–425.

[32] G. R. Pirrung et al. “A Coupled near and Far Wake Model for Wind Turbine Aerodynamics”. en. *Wind Energy* 19.11 (2016). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.1969, pp. 2053–2069.

[33] J. Rabault and A. Kuhnle. “Accelerating Deep Reinforcement Learning Strategies of Flow Control through a Multi-Environment Approach”. *Physics of Fluids* 31.9 (Sept. 2019), p. 094105.

[34] J. Rabault et al. “Deep Reinforcement Learning Achieves Flow Control of the 2D Karman Vortex Street”. en. *arXiv:1808.10754 [physics]* (Aug. 2018).

[35] K. Rohrig et al. “Powering the 21st Century by Wind Energy—Options, Facts, Figures”. *Applied Physics Reviews* 6.3 (Sept. 2019), p. 031303.

[36] M. Schönherr. “Towards Reliable LES-CFD Computations Based on Advanced LBM Models Utilizing (Multi-) GPGPU Hardware”. PhD Thesis. July 2015.

[37] J. Schulman et al. “Proximal Policy Optimization Algorithms”. en. *arXiv:1707.06347 [cs]* (Aug. 2017).

[38] J. Schulman et al. “Trust Region Policy Optimization”. en. *International Conference on Machine Learning*. June 2015, pp. 1889–1897.

[39] J. N. Sørensen and W. Z. Shen. “Numerical Modeling of Wind Turbine Wakes”. en. *Journal of Fluids Engineering* 124.2 (June 2002), pp. 393–399.

[40] J. N. Sørensen. *General Momentum Theory for Horizontal Axis Wind Turbines*. en. Vol. 4. Research Topics in Wind Energy. Cham: Springer International Publishing, 2016.

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

[42] M. Vali et al. “Adjoint-Based Model Predictive Control of Wind Farms: Beyond the Quasi Steady-State Power Maximization * *This Work Has Been Funded by the Ministry for Sciences and Culture of the Federal State of Lower Saxony, Germany as Part of the PhD Programme on System Integration of Renewable Energies (SEE) and by the German Ministry of Economic Affairs and Energy (BMWi) in the Scope of the WIMS-Cluster Project (FKZ 0324005).” en. *IFAC-PapersOnLine* 50.1 (July 2017), pp. 4510–4515.

[43] R. J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. en. *Machine Learning* 8.3 (May 1992), pp. 229–256.

A. The Cumulant Lattice Boltzmann Method

A.1. Derivation of cumulants

To provide a more detailed explanation of the cumulant lattice boltzmann method, a more thorough derivation is given here. First, the discrete distribution is written as in a continuous form as shown in (A.1). This is transformed to frequency space, to make the cumulants independent of frame of reference as shown in (A.2). This distribution is then used in the cumulant generating function, displayed in (A.4). However, from that form, the cumulants are not directly computable. Comparing the cumulant generating function to the moment generating function in (A.3) shows their similarity. Computing the cumulants of a arbitrary function shows how they can be expressed in terms of moments, two examples for second order cumulants are given in (A.5) and (A.6). Cumulants can also be computed from central moments, which is requires less computations [14]. Cumulants of zeroth and first order stay constant throughout the collision. The cumulants upwards from order two can now be relaxed indivdually like (A.7), therefore each has its own relaxation frequency. The first frequency ω_1 governs the shear viscosity and ω_2 the bulk viscosity of the fluid by the same relation as (1.48). 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 [13]. 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.[14]

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

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

$$M_{\alpha,\beta,\gamma} = c_s^{-(\alpha+\beta+\gamma)} \left.\frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial Z^\gamma} F(\Xi, \Upsilon, Z)\right|_{\Xi=\Upsilon=Z=0} = \sum_{i,j,k} i^\alpha j^\beta k^\gamma f_{i,j,k} \tag{A.3}$$

$$C_{\alpha,\beta,\gamma} = c_s^{-(\alpha+\beta+\gamma)} \left.\frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial Z^\gamma} \ln F(\Xi, \Upsilon, Z)\right|_{\Xi=\Upsilon=Z=0} \tag{A.4}$$

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

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

$$C_{\alpha,\beta,\gamma}* = (1 - \omega_{\alpha,\beta,\gamma})\, C_{\alpha,\beta,\gamma} \tag{A.7}$$

The density, velocity and the terms of velocity gradient tensor can be identified with cumulants as shown in (A.8) - (A.12). Note, that in this derivation the well-conditioned cumulant is used as described in the appendix of [14] 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 the chapter on refinement.

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

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

$$D_x v + D_y u = -3\omega_1 C_{110} = k_{xy} \tag{A.10}$$

$$D_x u = -\frac{\omega_1}{2\rho}\left(2C_{200} - C_{020} - C_{002}\right) - \frac{\omega_2}{2\rho}\left(C_{200} + C_{020} + C_{002} - \delta\rho\right) \tag{A.11}$$

$$D_x u - D_y v = -\frac{3\omega_1}{2}\left(C_{200} - C_{020}\right) = k_{xx-yy} \tag{A.12}$$

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 [36] 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 [36] as well as [28], 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 [36]. 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. 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 ω_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 shown in (A.17) - (A.20), the missing five constraints by the off diagonal can easily be extrapolated.

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

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

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

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

Thus a system of linear equations is established that can be solved for the coefficients of the interpolation functions. They are listed below, in order to reduce the number length 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.

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

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

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

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

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

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

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

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

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 ω_1 has to be scaled with a factor σ 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.

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

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

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

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

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

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

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

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

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

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

$$C_{011} = -\frac{\sigma\rho}{3\omega_d}\left(\overline{k_{yz}} + A_{011}\right) \tag{A.39}$$

$$C_{101} = -\frac{\sigma\rho}{3\omega_d}\left(\overline{k_{xz}} + A_{101}\right) \tag{A.40}$$

$$C_{110} = -\frac{\sigma\rho}{3\omega_d}\left(\overline{k_{xy}} + A_{110}\right) \tag{A.41}$$

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

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

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

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