TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen, Institut für Strömungsmechanik, Professur für Strömungsmechanik

Diplomarbeit

Wie ich darauf achte, dass zumindest auf der Titelseite keine Fehler sind

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

Validierung eines Wandmodels für Large Eddy Simulationen auf Basis der Lattice-Boltzmann Methode

15 zeilen

Abstract

Validation of a wall model for Large Eddy Simulations based on the Lattice Boltzmann Method

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
ERCOFTAC	European Research Community On Flow, Turbulence And Combustion
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
rms	Root mean square
WM-LES	Wall modelled Large-Eddy Simulation

1 Basic Windturbine Stuff

1.1 Description

1.2 Actuator Line Model

1.3 Greedy Control

2 Lattice Boltzmann Method

2.1 basics

2.2 boundary conditions

3 ANN stuff

3.1 Reinforcement Learning

3.1.1 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 S_t takes an action A_t, and the environment, that responds to the action A_t with changing its state to 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 (3.1). It defines the probability of the state s' with reward r occuring, given the state s and action 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.

$$p(s', r|s, a) \doteq Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a) \tag{3.1}$$

For a process to be a Markov Decision Process, the p must only depent on s and a. Therefore 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 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 (3.2). 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. [4, p. 47- 57]

$$G_t \doteq R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} ... = \sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}, \quad \gamma \in [0, 1] \tag{3.2}$$

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 [4].

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 $A_t = a$ given a state $S_t = s$ is denoted as $\pi_{\boldsymbol{\theta}}(a|s)$. The value function of a state s under a policy π is denoted as $v_\pi(s)$ and is the expected return if the agent acts according to π, starting from state s. Note that for convenience the parameters of π were be dropped. For MDPs it can be defined as (3.3).

$$v_\pi \doteq \mathrm{E}_\pi \left[G_t | S_t = s\right] = \mathrm{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} | S_t = s\right] \tag{3.3}$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left(r + \gamma v_\pi(s')\right) \tag{3.4}$$

In the form of (3.4), 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 a at state s under the policy π, denoted by $q_\pi(s, a)$. It is defined by (3.5). [4, p. 58-59]

$$q_\pi(s, a) \doteq \mathrm{E}_\pi \left[G_t | S_t = s, A_t = a\right] = \mathrm{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'} | S_t = s, A_t = a\right] \tag{3.5}$$

3.2 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 [4, p. 201] or its extensions such as Adam [1]. In its basic form, SGD performs an update according to 3.6. The parameter α is called the learning rate.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}) \tag{3.6}$$

The policy gradient methods differ now only in the performance measure. The first such algorithm proposed is the REINFORCE algorithm [5]. Its definition of $\nabla J(\boldsymbol{\theta})$ is presented in (3.7). It follows from the policy gradient theorem and substituting 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. [4, p.324-328]

$$\nabla J(\boldsymbol{\theta}) = \mathrm{E}_\pi \left[\sum_a \pi_{\boldsymbol{\theta}}(a|S_t) q_\pi(S_t, a) \frac{\nabla \pi_{\boldsymbol{\theta}}(a|S_t)}{\pi_{\boldsymbol{\theta}}(a|S_t)} \right] \tag{3.7}$$

$$= \mathrm{E}_\pi \left[G_t \frac{\nabla \pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}}(A_t|S_t)} \right] \tag{3.8}$$

While this algorithm makes a computation possible, it is inefficient. Therefore newer methods have been devised such as the Trust Region Policy Optimization (TRPO) [3] and the Proximal Policy Optimization (ppo) [2]. 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 (3.10). 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. $a_\pi(A_t|S_t)$ is an estimator of the advantage function, which is defined as $a_\pi(A_t|S_t) = q_\pi(A_t|S_t) - v_\pi(S_t)$. This estimator also has to be found, however, this is a regular optimization problem, which can be solved via SGD or Adam.

$$r_t(\boldsymbol{\theta}) \doteq \frac{\pi_{\boldsymbol{\theta}}(A_t|S_t)}{\pi_{\boldsymbol{\theta}_{old}}(A_t|S_t)} \tag{3.9}$$

$$J \doteq \mathrm{E}_\pi \left[\min \left(r_t(\boldsymbol{\theta}) a_\pi(A_t|S_t), \mathrm{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) a_\pi(A_t|S_t) \right) \right] \tag{3.10}$$

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 $\boldsymbol{\theta}$.

3.3 backpropagation

3.4 ANN Design

Bibliography

[1] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". en. *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 02/13/2020).

[2] J. Schulman et al. "Proximal Policy Optimization Algorithms". en. *arXiv:1707.06347 [cs]* (Aug. 2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347 (visited on 02/12/2020).

[3] J. Schulman et al. "Trust Region Policy Optimization". en. *arXiv:1502.05477 [cs]* (Apr. 2017). arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477 (visited on 02/27/2020).

[4] 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.

[5] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". en. *Machine Learning* 8.3-4 (May 1992), pp. 229–256. URL: http://link.springer.com/10.1007/BF00992696 (visited on 02/27/2020).

Selbstständigkeitserklärung

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

Validierung eines Wandmodels für Large Eddy Simulationen auf der Basis der Lattice-Boltzmann Methode

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

Dresden, 06. Januar 2020

Henry Korb