Evaluation of a lattice Boltzmann-based wind-turbine actuator line model against a Navier-Stokes approach

Helen Schottenhamml[1], Ani Anciaux-Sedrakian[2], Frédéric Blondel[2], Adria Borras-Nadal[2], Pierre-Antoine Joulin[2], Ulrich Rüde[1,3]

[1] Friedrich-Alexander-Universität, Cauerstrasse 11, 91058 Erlangen, Germany

[2] IFP Energies nouvelles, 1-4 avenue du Bois Préau, 92852 Rueil-Malmaison, France

[3] CERFACS, 42 Avenue Gaspard Coriolis, 31100 Toulouse, France

E-mail: helen.schottenhamml@fau.de

Abstract. Due to the cost and difficulty to precisely measure aerodynamic quantities in onshore and offshore wind farms, researchers often rely on high-fidelity large eddy simulation, based on Navier-Stokes flow solvers. However, the cost of such simulation is very high and does not allow, in practice, extensive parametric studies for large wind farms. Among others, the lattice Boltzmann method is a good candidate for much faster, ExaScale wind farm flow simulations. The present paper aims to assess the validity of a lattice Boltzmann-based actuator line model and highlights its strengths and potential weaknesses. With this intent, comparisons against a Navier-Stokes approach commonly used in the wind energy community are performed. We assess the potential of the lattice Boltzmann method to reduce the computational cost of such simulations by analyzing the performance of the different solvers and their scalability. The lattice Boltzmann-based waLBerla solver reduces the computational costs significantly compared to SOWFA while maintaining the same accuracy as the Navier-Stokes-based method. Furthermore, we show that a multi-GPU implementation leads to an even more drastic reduction of the computational time, achieving faster-than-real-time simulations. This performance will allow extensive parametric studies over large wind farms in future studies.

1 Introduction

Wind farm flows are complex and challenging to predict, although their understanding is crucial for proper wind farm design and layout optimization. During the design phase, so-called "engineering" flow models are used to estimate wind farm power production [1, 2]. However, the accuracy of these models is still limited. Several phenomena are as yet poorly understood, such as overlapping wakes, turbulence generation in wakes, wake deflection, wake meandering, or even wind farm blockage effects and atmospheric stability, i.e., thermal effects. Accurate reference data is required to support the development of such models. However, it remains problematic to obtain reliable on-site data as measurements are complex and subject to the atmosphere's random nature. On the other hand, wind tunnel experiments are expensive and less flexible than numerical models. Thus, a common approach

consists in using high-fidelity solvers for parametric studies after validating them against a limited number of wind tunnel measurements. Hence, researchers rely on high-fidelity large eddy simulation (LES)-based solvers, such as SOWFA [3]. The solvers often build upon actuator-line methods [4]: based on the airfoil's aerodynamic properties, i.e., lift and drag curves, and the blade element theory, aerodynamic forces acting on the blades are estimated over simple lines and projected onto the flow, instead of fully resolving the turbines geometrically. At the wind farm scale, one is more interested in wake developments, interactions, and coupling with the atmospheric boundary layer than in the detailed flow around the blades. Moreover, previous studies have already shown that the detailed wind turbine geometry has a limited impact on the far wake properties [5]. Avoiding the fine and body-fitted meshes reduces their required sizes. Nevertheless, using these solvers for parametric studies and complete wind farm flows remains challenging. Different length scales must be resolved to capture the correct physical behaviour, e.g., atmospheric eddies with kilometric scale and wind turbine wake eddies with meters. These challenges inevitably require massive computational resources and leading-edge high-performance simulation software. Lattice Boltzmann-based approaches constitute a relevant alternative. There are no fundamental limitations for lattice Boltzmann method (LBM) to deal with atmospheric boundary layer (ABL) flows and wind turbine modelling capabilities with the same precision as Navier-Stokes (NS)-based approaches, as demonstrated in [6, 7, 8]. Furthermore, they exhibit much lower computational costs and are well-adapted to large-scale heterogeneous architectures. Recent developments, e.g., improved collision operators, enable the LBM for high-fidelity high-Reynolds number flows needed for wind farm flow simulations. The present work aims to compare two LES-based flow solvers, including a recently developed LBM approach based on the waLBerla flow solver [9, 10, 11], in terms of physics and computational efficiency, including weak and strong scaling. The other solver is the OpenFOAM-based LES flow solver SOWFA, which is among the most commonly used in the wind energy community [12]. Although this comparison is purely numerical, a validation of the waLBerla solver against wind tunnel experiments can be found in [13], and a validation study of SOWFA against a full-scale wind farm can be found in [14]. While this article does not deal with entire wind farms, the investigations on a single turbine are a crucial step towards large-scale simulations.

2 Lattice Boltzmann method

The following section provides a brief introduction to the lattice Boltzmann method. A more comprehensive treatment can be found in, e.g., [15] or [16]. In contrast to conventional solvers, the LBM does not build upon a discretisation of the Navier-Stokes equations. Instead, it discretises the Boltzmann equation, defined through the so-called particle distribution function (PDF) f. Firstly, the continuous velocity space is reduced to a set of discrete velocities $\mathbf{c}_i$. The discretisation in velocity space also introduces the notion of stencils which are a crucial ingredient of lattice Boltzmann implementations. By custom, a $DdQq$ stencil represents a d-dimensional LBM with q discrete velocities. While the choice of stencil impacts numerical properties like the accuracy and stability of an LB method, it also influences the computational efficiency and memory requirements. Typically, a stencil that includes more neighbours lowers numerical errors and improves stability. However, this comes at the cost of reduced computational performance which is typically memory-bound in LBM. Secondly, the Boltzmann equation is discretised in space and time. Clasically, the LBM acts on uniform and regular grids with an explicit time stepping scheme. Eventually, this discretisation results in the second-order accurate lattice Boltzmann equation, here without forcing terms,

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t). \tag{1}$$

Typical implementations split this equation into a streaming part and a collision part. The collision step evaluates the right-hand side of Equation 1, including the collision operator Ω. Since this step is usually cell-local, it is very well-suited for parallelisation and enables the LBM for large-scale applications. The second step, the streaming, then copies the post-collision distribution to

the neighbouring cells. This step only acts on direct neighbours, dictated by the chosen velocity set. Hence, it does not compromise the computational efficiency too much. Until now, we have not yet discussed the importance of the collision operator Ω. In addition to the stencil, it defines a specific lattice Boltzmann method and dictates its stability and accuracy properties. The most straightforward approaches act directly on the PDFs and relax them towards their equilibrium with one or several relaxation times. While they are easier to implement, they restrict the simulation's stability. Hence, more sophisticated collision operators have been introduced that overcome this issue. The approach that we chose for our simulations is the cumulant lattice Boltzmann method (CLBM) by Geier et al. [17]. Instead of employing a specific force model, we rely on the velocity shift proposed in [17] to incorporate the forcing terms.

3 Framework description

This section briefly describes the frameworks used for the comparisons. Furthermore, it details their numerical setups, subgrid-scale (SGS) models and grid structuredness.

3.1 waLBerla

waLBerla [9, 10] is a massively parallel multi-physics framework developed at Friedrich-Alexander University Erlangen-Nuremberg (FAU) with focus on simulations using the lattice Boltzmann method. Its specific software design combined with the code generation framework lbmpy [18] allows for highly optimised, performance-portable solutions on block-structured grids for various application areas, like phase-field simulations, and particle-laden flows [19, 20]. It supports shared and distributed memory parallelism with OpenMP and MPI, respectively, automated SIMD vectorisation, and the execution on NVIDIA graphics cards. For the wind turbine simulations in the present study, we use our recently added holistic actuator line model (ALM) approach [13], which provides a mutual code base for CPU and GPU. Not only does this approach conserve the performance-portability of waLBerla, but it also reduces the maintenance efforts. For the LBM setup, we chose a D3Q27 stencil and a cumulant collision model with a Smagorinsky SGS model.

3.2 SOWFA

SOWFA (Simulator fOr Wind Farm Applications) [3] provides a set of solvers, boundary conditions, and physical extensions based on the OpenFOAM CFD solver [21]. Containing wind turbine models, e.g., an actuator-line model, it enables the simulation of wind turbines and wind farms operating in neutral or stratified ABLs. In the wind-energy research community, SOWFA is the most widely used solver for wind farm flows [12]. In the present study, we use second-order spatial schemes and a Pressure-Implicit with Splitting of Operators (PISO) pressure-velocity calculation procedure with three sub-iterations per timestep using a geometric agglomeration-based algebraic multigrid solver, together with an implicit scheme for time integration. Moreover, we employ a Smagorinsky SGS model. OpenFOAM is an unstructured grid solver which enhances the approximation of complex geometries but comes at the price of reduced performance compared to structured grids. For parallelisation, the domain is decomposed and distributed among different processes using MPI.

3.3 castor

CASTOR is a so-called free-wake, lifting-line flow solver based on a vortex filament discretisation of the wake and the wind-turbine blades. The underlying N-body problem is solved on GPU-systems using the CUDA programming language to increase the computational performance. It has been validated against wind tunnel measurement [22]. In the present study, CASTOR acts as a reference for blade force distributions: this kind of solver is supposed to be accurate for blade force predictions and intrinsically accounts for tip-losses. Furthermore, CASTOR is based on a lifting-line approach, which allows a fair comparison with actuator-line solvers, as the same limitations are expected, e.g., no three-dimensional effects such as stall delay are accounted for. The present

implementation follows the work of Sebastian et al. [23] and uses Van Garrel's core model [24]. A first-order advection scheme is employed. The time step is chosen to have a hub rotation of ten degrees per iteration, while more than 20 full rotor rotations are kept in the wake.

3.4 Force spreading methods

Part of the actuator line method is usually the smearing of aerodynamic forces over several cells around the blade elements. For doing so, often linear or Gaussian regularisation kernels are employed. More details and generic guidelines can be found in, e.g., [25].
SOWFA implements several force projection kernels, including isotropic and anisotropic ones. Here, we employ the standard isotropic Gaussian kernel using a standard deviation of $\epsilon = 2\Delta x$ with Δx being the width of the mesh cells. In the waLBerla setups, we use two different kernels for the force smearing function. On the one hand, there is the tight kernel by Roma et al. [26] which approximates the Dirac delta over three cells per spatial dimension. On the other hand, we have a wider Gaussian kernel that spreads the forces over nine cells per spatial direction with $\epsilon = 2\Delta x$. With this discretisation of the smearing functions, waLBerla avoids the numerically expensive convolution product that is typically evaluated for Gaussian kernels. The setups with the thinner and the wider kernel are called "waLBerla-thin" and "waLBerla-wide" in the following.

4 Methodology

The proposed study uses the generic DTU 10MW reference wind turbine [27] which is representative of modern large offshore wind turbines. Contrary to previous validation studies presented in [13], this case exhibits very high chord-based Reynolds numbers of the order of ten million. These Reynolds numbers could have been challenging to deal with using standard methods used in the early stages of LBM. We compare the time-averaged aerodynamic force distributions along the blade, the wake characteristics, i.e., velocity deficit and turbulent intensity profiles, predicted by the different solvers, and the code performance and computational times. All approaches employ an actuator-line type of model at a single operating point with a wind speed of $8m/s$, leading to a rotational velocity of $\Omega \approx 0.673 rad/s$, no blade pitch, and airfoil data retrieved from the public wind turbine model [28]. Hub tilt and blade precone are neglected. At such operating conditions, the simulated wind turbine thrust and power coefficients are $C_T \approx 0.814$ and $C_P \approx 0.476$, respectively, in accordance with the HAWCStab2 configuration in [28]. Thus, one expects a strong wake velocity deficit due to the large thrust coefficient considered here. The airfoil lift and drag coefficients used in the present study can be found in [28]. Linear interpolations of the provided data with respect to the local blade thickness are performed, enabling smooth transitions between the different airfoils. We discretise the computational domain of 25 rotor diameters D in length, $5D$ in height and $5D$ in width with similar meshes between the solvers. A single turbine with $D = 178.32m$ and a tower height of $118m$ is located at $5D$ from the inlet. To enforce a mutual boundary setup between the solvers, we use free-slip conditions at the bottom and the top of the domain, periodic lateral boundaries, and a constant uniform inflow at the inlet. The reference vortex solver cannot fulfil this setup as it does not implement wall boundary conditions. A constant wind velocity is used all over the domain in order to advect the vortex filament in a Lagrangian way, together with the so-called induced velocities. The domain is unbounded, there are neither outflow nor top or bottom boundaries. For the grid-based approaches, the mesh resolution reaches 64 cells per rotor diameter, which is sufficiently fine to predict the wake properties correctly [29]. However, we cannot accurately resolve the tip vortices in the near wake at such mesh resolutions, and thus so-called tip losses are probably not correctly captured. Still, we do not use a tip loss model to minimise the potential source of discrepancies and characterise the solvers and not the aerodynamic corrections. While the force spreading methods slightly differ between the solvers, i.e., waLBerla spreads the forces over either three cells in the narrow kernel or nine cells in the wide kernel, they all use a trilinear interpolation for the wind quantities. Enforcing a rotation of approximately one degree per time step, we have a time step size

of $0.026s$ and an equivalent Mach number of $\mathcal{M} \approx 0.13$ for the LBM simulations. Asmuth et al. [8] provide more details on the relation between Mach number and time step size in LBM wind turbine simulations. With this time step size, the blade elements do not cross more than one cell per time step, even at the tip. However, convergence studies provided in Appendix A show that this time step might be too high to predict the far wake flow accurately. The grid-free vortex solver discretises the blades using 35 cosine-spaced vortex filaments. Other solvers use a linear discretisation that includes at least 50 elements per blade. The simulations start with an initial period of 10 min of physical time, i.e., approximately 64 rotations of the wind turbine, during which the wake develops. Then, we evaluate the quantities of interest averaged over 10 additional minutes of physical time.

5 Results

This section details the physical results of the solvers. In particular, we will address the aerodynamic force distribution and wake properties, as the velocity deficit and turbulence intensity.

5.1 Blade force distribution

We focus on the analysis of the blade force distribution first. Recall that we employ two force spreading kernels in waLBerla: a wide one in waLBerla-wide runs for comparing to SOWFA, and another thin kernel in waLBerla-thin runs to study the impact of the spreading method, see 3.4. In Figure 1 (left) that depicts the distribution of the normal forces, the agreement of the different solvers is overall good. waLBerla-wide and SOWFA predict eminently similar blade force distributions, even near the blade root and tip where the forces are over-estimated compared with the vortex solver. waLBerla-thin's normal force distribution is closer to the CASTOR reference results. The observed force reduction when changing the kernel is attributed to a better resolution of the tip vortex, leading to a better prediction of the tip-losses.

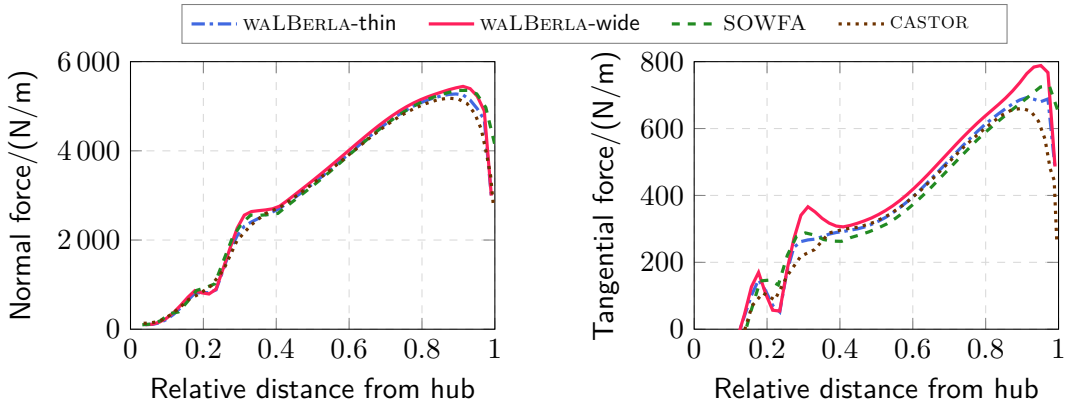

Figure 1. Normal (left) and tangential (right) force distribution along the blade, 64 cells per diameter

Figure 1 (right) shows that the results for the tangential forces are less aligned. waLBerla and SOWFA results differ from CASTOR, especially near the blade root and tip. SOWFA predicts in general lower tangential forces than waLBerla, CASTOR results being between the two. Despite some noticeable differences at the blade root and tip regions, the waLBerla-thin results are aligned better with CASTOR. Nevertheless, the agreement between the solvers is still overall satisfactory. The mentioned discrepancies between the two LES solvers can be attributed mainly to the differences in the force spreading and the numerical methods.

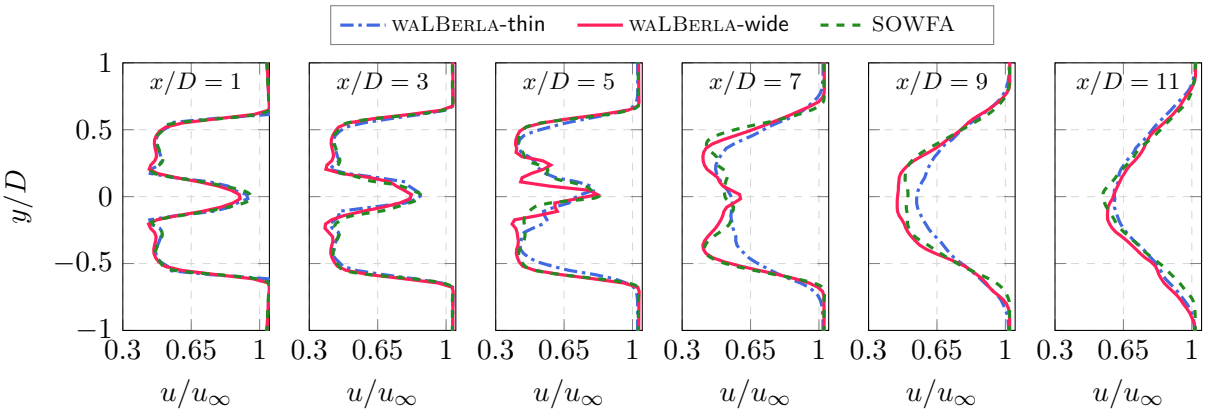

Figure 2. Wake velocity profiles, 64 cells per diameter

5.2 Wake velocity and turbulence intensity profiles

Next, we will look at the velocity profiles at several downstream positions in the wake. Figure 2 shows a globally satisfying agreement between the solvers. The transition to a turbulent wake occurs at a similar location for the wider kernels in SOWFA and waLBerla-wide. Even nearly Gaussian profiles are observed in the far wake ($x/D \geqslant 9$). waLBerla-wide predicts a slightly higher velocity deficit at $x/D = 9$. Using the thinner kernel, the waLBerla-thin velocity profiles tend to be smoother, indicating an earlier transition to a turbulent wake. Martinez-Tossas et al. [30] predict this kind of behaviour, reasoning that with a narrower kernel thinner and more unstable tip vortices emerge. In the far wake ($x/D = 11$), the impact of the force spreading kernel is low. This postulation is consistent with the observed turbulence intensity profiles in Figure 3. We define the axial turbulence intensity as the ratio of the root-mean-square velocity fluctuations to the mean velocity, sampled in a space-fixed reference frame at every simulation time step. It does include coherent structures such as the tip vortices. In the near-wake at $x/D = 3$, waLBerla-thin results in a higher turbulence level near the rotor blade tips than waLBerla-wide. As shown in Appendix A, reducing the time step also delays the transition to a turbulent wake. More investigations are required to understand this phenomenon. In particular, using strictly identical force spreading methods is mandatory before drawing any definitive conclusions. In terms of code-to-code turbulence intensity profiles comparisons, there are noticeable differences. Especially in the far wake, waLBerla-wide surprisingly predicts higher turbulence intensities than SOWFA. Even though the presented comparisons are globally

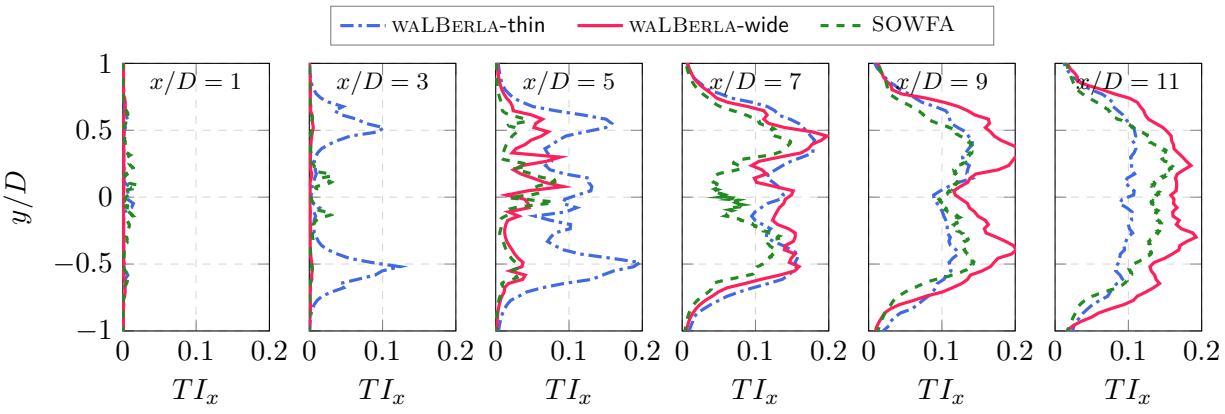

Figure 3. Wake TI profiles, 64 cells per diameter

satisfactory, they drag behind those of Asmuth et al. [8]. A key difference is that Asmuth et al. used identical force spreading methods and a coarser mesh resolution. Rerunning our setup with 32 cells per rotor diameter improved the agreement between waLBerla-wide and SOWFA.

6 Code performances

In this section, we compare the performance of NS-based and LBM-based ALM solvers for wind turbines in a parallel environment. In particular, we investigate their weak and strong scaling behaviour and their overall performance. All simulations run on the *Topaze* supercomputer at

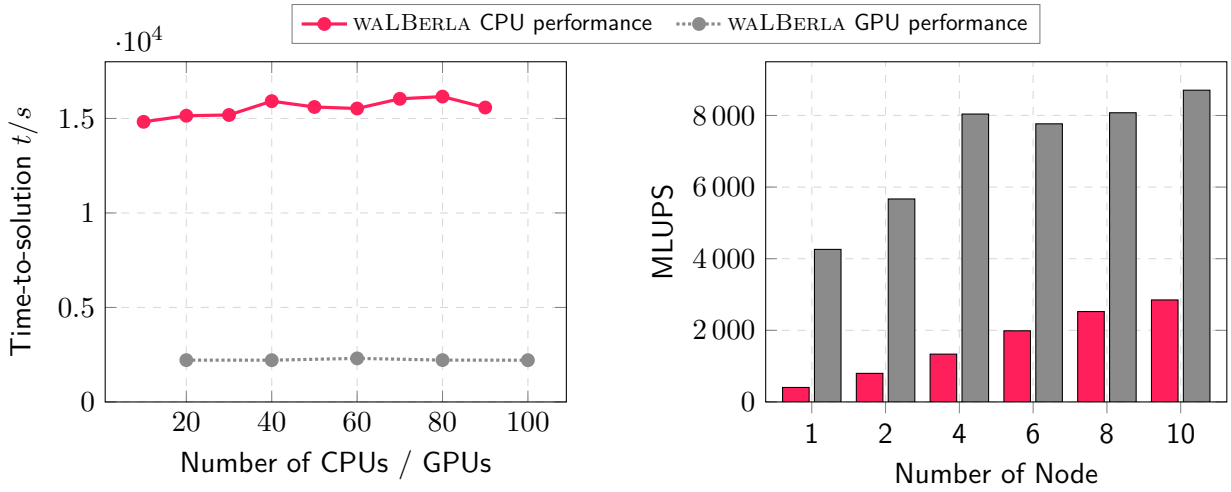

Figure 4. waLBerla scaling experiments for $1\,200s$ simulated physical time with $\Delta t = 0.010054s$: weak scaling with $40\,960\,000$ cells per node (left), strong scaling with $163\,840\,000$ cells (right)

CCRT/CEA and follow the setup described in Section 4. Topaze has 864 compute nodes based on 2 AMD Milan@2.45GHz (AVX2) CPUs with 64 cores per CPU. Furthermore, it includes an accelerated partition with 48 compute nodes with 4 Nvidia A100 GPUs each.
Firstly, we focus on the performance aspects of waLBerla and its holistic ALM approach. Figure 4 (left) depicts the weak scaling of waLBerla for CPU and GPU runs and clearly shows that the excellent behaviour of waLBerla, e.g., reported in [10], was not compromised by the ALM implementation. Figure 4 (right), on the other hand, compares the strong scaling behaviour of the CPU and the GPU implementations of waLBerla in terms of *mega lattice site updates per second* (MLUPS). Here, we observe no perfect but still a favourable performance increase with an increasing number of compute nodes. For the GPU runs, the performance increase saturates for more than four nodes with four GPUs each. The simulation domain in these runs was too small to add sufficient workload to all GPUs, hence not further decreasing the time-to-solution. However, the weak scaling proves that we still scale when we provide a sufficient workload.

Table 1. Simulation times in s for the strong scaling runs for waLBerla and SOWFA

Node	GPU	CPU	Cores	waLBerla GPU	waLBerla CPU	SOWFA
2	8	4	256	1 351	9 652	-
5	20	10	640	852	5 368	401 376
10	40	20	1 280	864	2 734	191 215
15	60	30	1 920	-	1 597	135 817
20	80	40	2 560	-	1 390	91 652

Next, we will directly compare the two solvers against each other. The exact performance results in "time to solution" are given in Table 1 for a total of $163\,840\,000$ cells and $1\,200s$ of simulated physical time. Note that we did not include the results for more than ten nodes for the waLBerla GPU simulations. Again, the workload per GPU was not sufficient to obtain representative results. Also, SOWFA lacks the 2-node simulations as the runs were too slow and, therefore, not feasible to perform. As illustrated in the left of Figure 5, the speed-up of all three solvers is perfect up to 20 CPU

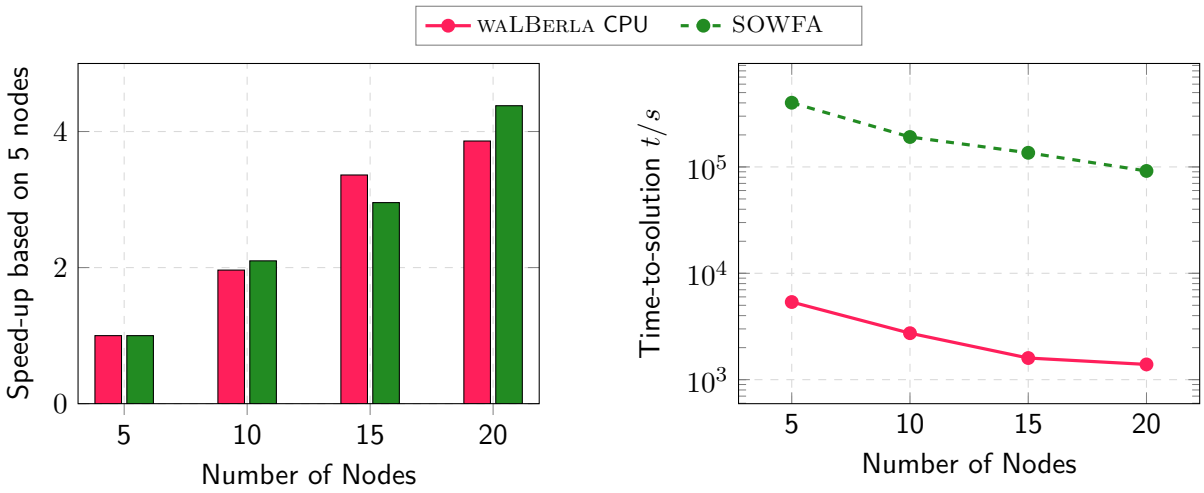

Figure 5. Strong scaling experiment with $163\,840\,000$ cells, $1\,200s$ simulated physical time with $\Delta t = 0.026s$: Speed-up based on the simulation time on five nodes (left), performance in time-to-solution (right)

nodes. As a basis for the speed-up, we used the simulation times of the solvers' 5-node runs. While the speed-up is a crucial factor towards large-scale runs, as it allows exploiting large clusters, we must not neglect the actual time-to-solution, given logarithmically in the right of Figure 5. The difference between the LBM-based waLBerla and the NS-based SOWFA is immense, with waLBerla being on average 73 times faster than SOWFA. On five nodes, the GPU version of waLBerla performs even better with a factor of 471 to SOWFA. Moreover, considering the simulated physical time of $1\,200s$, the GPU runs on five nodes perform faster than real-time.

7 Conclusion

This study is a preliminary step towards validating a lattice Boltzmann flow solver for wind farm flow simulations using the actuator line model to represent the turbines' rotor. We focused on the blade force distribution and wake properties of a single wind turbine subject to a constant inflow. Comparisons against a Navier-Stokes-based flow solver were performed and show that the lattice Boltzmann method leads to very similar results, considering identical numerical setups. In terms of computational cost, the LBM performed about 70 times faster than the SOWFA library. Using the same number of nodes but running on GPUs, this factor even increases to about 470. LBM seems an auspicious candidate for highly performant and physically realistic wind farm flow simulations. The next step will consist in the extension of the currently employed waLBerla framework to ABL simulations, allowing the study of realistic wind farm flows.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824158 (EoCoEII).

Appendix A Convergence study

For the sake of completeness, we perform a convergence study in this appendix. This study is based on the lattice Boltzmann solver since it is the main interest of this work. The Roma kernel is employed, allowing a single layer of ghost cells, thus slightly reducing the computational cost. Four different Mach numbers are used, i.e., $\mathcal{M} = 0.02$, 0.05, 0.1 and 0.15 with corresponding time steps of $\Delta t = 0.004$, 0.010, 0.020 and 0.030, respectively. Figure A1 shows the Mach number

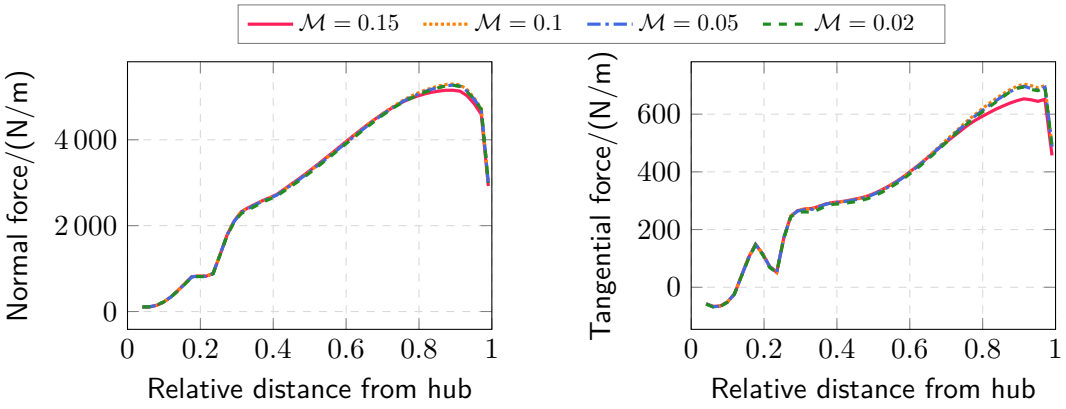

Figure A1. Force distribution along the blade: Mach number impact

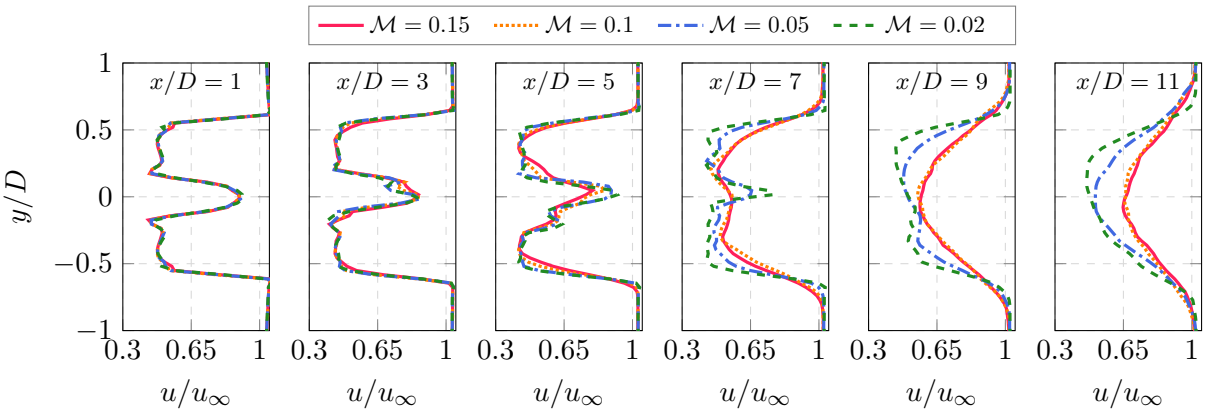

Figure A2. Wake velocity profiles: Mach number impact

impact on the blade force distribution. Some discrepancies appear near the tip of the blade at the largest Mach number, i.e., at the compressible limit. Otherwise, results converge rapidly towards the same solution: no differences are noticed between $\mathcal{M} = 0.1$ and lower Mach numbers. The Mach number convergence results for the wake velocity and turbulence intensity profiles are presented in Figures A2 and A3. In the near-wake, i.e., $x/D < 5$, the velocity profiles are not strongly impacted by the time step, although lowering it reduces the turbulence intensity, especially near the rotor tip. Further downstream, the impact is higher. Lowering the time step sharpens the velocity deficit, and one also observes a lower turbulence intensity. In other words, decreasing the time step size tends to delay the transition to a turbulent wake further downstream. Also, note that we do not reach convergence in the far wake, at $x/D > 7$. This result is unexpected since we should be able to capture higher wavenumbers with smaller time steps, leading to an earlier transition to turbulence [31]. As mentioned in [30], the force spreading method has a similar impact on the tip vortex.

The shear layer is thinner and more unstable using narrow kernels, allowing an earlier transition to turbulence. A convergence study based on a Navier-Stokes solver should be undertaken to confirm these observations. To investigate the impact of the kernel, we ran additional simulations using different kernel widths. Although it seems to help in the convergence, the same phenomenon is observed. Note that in practice, actuator-line simulations also consider a turbulent atmospheric boundary layer. The ABL allows a much faster wake recovery compared with the present constant inflow simulations and probably help in the convergence.

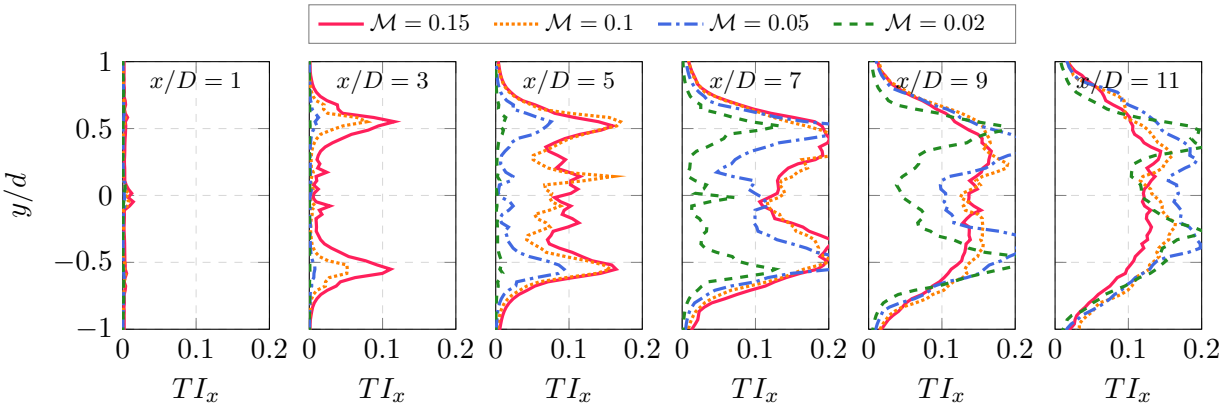

Figure A3. Wake turbulence intensity profiles: Mach number impact

References

[1] Bastankhah M and Porté-Agel F 2014 *Renew. Energy* **70** 116–123
[2] Blondel F and Cathelain M 2020 *Wind Energy Sci.* **5** 1225–36
[3] Sowfa: Simulator for wind farm applications `www.nrel.gov/wind/nwtc/sowfa.html` accessed: 2022-01-23
[4] Sørensen J N and Shen W Z 2002 *J. Fluids Eng.* **124** 393–399
[5] Aubrun S, Loyer S, Hancock P and Hayden P 2013 *J. Wind Eng. Ind. Aerodyn.* **120** 1–8
[6] Feng Y, Miranda-Fuentes J, Guo S, Jacob J and Sagaut P 2021 *J. Adv. Model. Earth Syst.* **13**
[7] Asmuth H, Janßen C F, Olivares-Espinosa H and Ivanell S 2021 *Phys. Fluids* **33** 105111
[8] Asmuth H, Olivares-Espinosa H and Ivanell S 2020 *Wind Energy Sci.* **5** 623–645
[9] Feichtinger C, Donath S, Köstler H, Götz J and Rüde U 2011 *J. Comput. Sci.* **2** 105–112
[10] Bauer M, Eibl S, Godenschwager C, Kohl N, Kuron M, Rettinger C, Schornbaum F, Schwarzmeier C, Thönnes D, Köstler H and Rüde U 2021 *Comput. Math. Appl.* **81** 478–501
[11] Schornbaum F and Rüde U 2016 *SIAM J. Sci. Comput.* **38** C96–C126
[12] Martínez-Tossas L A, Churchfield M J and Leonardi S 2015 *Wind Energy* **18** 1047–60
[13] Schottenhamml H, Anciaux-Sedrakian A, Blondel F and Ruede U 2021 Actuator-line simulation of wind turbines using the lattice boltzmann method (*Preprint* `10.13140/RG.2.2.10928.89603`)
[14] Churchfield M, Lee S, Moriarty P, Martínez Tossas L, Leonardi S, Vijayakumar G and Brasseur J 2012 *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*
[15] Krüger T, Kusumaatmaja H, Kuzmin A, Shardt O, Silva G and Viggen E M 2017 *The Lattice Boltzmann Method* (Springer, Cham)
[16] Succi S 2018 *The Lattice Boltzmann Equation* (Oxford University Press)
[17] Geier M, Schönherr M, Pasquali A and Krafczyk M 2015 *Comput. Math. Appl.* **70** 507 – 547
[18] Bauer M, Köstler H and Rüde U 2021 *J. Comput. Sci.* **49** 101269
[19] Holzer M, Bauer M, Köstler H and Rüde U 2021 *Int. J. High Perform. Comput. Appl.* **35** 413–427
[20] Rettinger C, Eibl S, Rüde U and Vowinckel B 2022 *J. Fluid Mech.* **932** A1
[21] Weller H G, Tabor G, Jasak H and Fureby C 1998 *Comput. Phys.* **12** 620–631
[22] Schepers J, Lutz T, Boorsma K, Gomez-Iradi S, Herraez I, Oggiano L, Rahimi H, Schaffarczyk P, Pirrung G, Madsen H A and et al 2018 (ECN)
[23] Sebastian T and Lackner M 2012 *Renew. Energy* **46** 269–275
[24] van Garrel A 2003 Development of a wind turbine aerodynamics simulation module Tech. rep.
[25] Jha P K, Churchfield M J, Moriarty P J and Schmitz S 2014 *J. Sol. Energy Eng.* **136**
[26] Roma A M, Peskin C S and Berger M J 1999 *J. Comput. Phys.* **153** 509 – 534
[27] Bak C, Zahle F, Bitsche R, Kim T, Yde A, Henriksen L, Hansen M, Blasques J, Gaunaa M and Natarajan A 2013 The dtu 10-mw reference wind turbine danish Wind Power Research 2013 URL `https://orbit.dtu.dk/en/publications/the-dtu-10-mw-reference-wind-turbine`
[28] Project page dtu 10mw rwt `https://rwt.windenergy.dtu.dk/dtu10mw/dtu-10mw-rwt` accessed: 2022-01-23
[29] Benard P, Viré A, Moureau V, Lartigue G, Beaudet L, Deglaire P and Bricteux L 2018 *Comput. Fluids* **173** 133–139
[30] Martínez-Tossas L A and Leonardi S 2013 Wind turbine modeling for computational fluid dynamics Nrel/sr-5000-55054 National Renewable Energy Laboratory URL `https://nrel-primo.hosted.exlibrisgroup.com/permalink/f/1a440ct/NREL_ALMA5148606600003216`
[31] Martínez-Tossas L A, Churchfield M J, Yilmaz A E, Sarlak H, Johnson P L, Sørensen J N, Meyers J and Meneveau C 2018 *J. Renew. Sustain. Energy* **10** 033301