Computers and Electronics in Agriculture 175 (2020) 105546

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

On the feasibility of using open source solvers for the simulation of a turbulent air flow in a dairy barn

David Janke[a,*], Alfonso Caiazzo[b], Naveed Ahmed[c], Najib Alia[b], Oswald Knoth[d], Baptiste Moreau[b], Ulrich Wilbrandt[b], Dilya Willink[a], Thomas Amon[a,f], Volker John[b,e,1]

[a] Leibniz-Institut für Agrartechnik und Bioökonomie e.V. (ATB), Max-Eyth-Allee 100, 14469 Potsdam, Germany
[b] Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr. 39, 10117 Berlin, Germany
[c] Department of Mathematics and Natural Sciences, Gulf University for Science and Technology, Kuwait City, Kuwait
[d] Leibniz-Institut für Troposphärenforschung (TROPOS), Permoserstraße 15, 04318 Leipzig, Germany
[e] Freie Universität Berlin, Department of Mathematics and Computer Science, Arnimallee 6, 14195 Berlin, Germany
[f] Freie Universität Berlin, Institute for Animal Hygiene and Environmental Health, Robert-von-Ostertag-Str. 7-13, 14163 Berlin, Germany

ARTICLE INFO

Keywords:
Naturally ventilated barns
Transient simulations
Wind tunnel
OpenFOAM
ParMooN

ABSTRACT

Two transient open source solvers, OpenFOAM and ParMooN, and the commercial solver Ansys Fluent are assessed with respect to the simulation of the turbulent air flow inside and around a dairy barn. For this purpose, data were obtained in an experimental campaign at a 1: 100 scaled wind tunnel model. All solvers used different meshes, discretization schemes, and turbulence models. The experimental data and numerical results agree well for time-averaged stream-wise and vertical-wise velocities. In particular, the air exchange was predicted with high accuracy by both open source solvers with relative differences less than 4% and by the commercial solver with a relative difference of 9% compared to the experimental results. With respect to the turbulent quantities, good agreements at the second (downwind) half of the barn inside and especially outside the barn could be achieved, where all codes accurately predicted the flow separation and, in many cases, the root-mean-square velocities. Deviations between simulations and experimental results regarding turbulent quantities could be observed in the first part of the barn. These deviations can be attributed to the utilization of roughness elements between inlet and barn in the experiment that were not modeled in the numerical simulations. Both open source solvers proved to be promising tools for the accurate prediction of time-dependent phenomena in an agricultural context, e.g., like the transport of particulate matter or pathogen-laden aerosols in and around agricultural buildings.

1. Introduction

In Europe, dairy cows are mainly housed in naturally ventilated barns (NVBs). Also in pigs and poultry farming, the trend is towards naturally ventilated systems or systems with increased spout areas, mainly to increase the animal welfare and the consumers' acceptance. The NVBs act as sources of airborne pollutants, both gaseous and particle-associated. The gaseous pollutants are mainly ammonia (NH_3), methane (CH_4), carbon dioxide (CO_2), or dinitrogen monoxide (N_2O). Pollutants associated with particles are, e.g., particulate matter (PM) or droplets. These particles can act as carriers for pathogens which arise through sick animals inside the barn. Carried out of the barn, the pathogens can spread diseases that are harmful either for other animals or (in case of zoonosis) for human beings. In order to assess and possibly mitigate the risk of airborne pathogen spreading out of NVBs, it is necessary to obtain insights on the flow fields inside and around the barns and to assess the air exchange (AE). The direct coupling of the inside flow regime with the ambient, turbulent weather conditions makes it hard to measure the flow conditions and the AE. The buildings and their openings are very large, velocities and gaseous concentrations are heterogeneously distributed and vary in time and space (König et al., 2018).

Computational fluid dynamics (CFD) is a useful tool for acquiring detailed insight into the complex flow fields encountered in NVBs. In

* Corresponding author.
E-mail addresses: djanke@atb-potsdam.de (D. Janke), caiazzo@wias-berlin.de (A. Caiazzo), ahmed.n@gust.edu.kw (N. Ahmed), alia@wias-berlin.de (N. Alia), knoth@tropos.de (O. Knoth), moreau@wias-berlin.de (B. Moreau), wilbrandt@wias-berlin.de (U. Wilbrandt), dwillink@atb-potsdam.de (D. Willink), tamon@atb-potsdam.de (T. Amon), john@wias-berlin.de (V. John).
[1] ORCID 0000-0002-2711-4409.

https://doi.org/10.1016/j.compag.2020.105546
Received 23 October 2019; Received in revised form 14 May 2020; Accepted 29 May 2020
Available online 18 June 2020

particular, CFD allows to predict the main features of the flow fields also at locations where measurements are practically infeasible, or to perform virtual assessment studies with the help of computational models. The use of CFD to describe the flow characteristics in NVBs has considerably gained popularity in recent years.

The flow under consideration is time-dependent, and even turbulent, see the beginning of Section 2.8 for a discussion on some features of turbulent flows. Such kinds of flows can be modeled mathematically by the evolutionary incompressible Navier-Stokes equations. In fact, the Reynolds number of the considered flow is so large that the steady-state Navier-Stokes equations do not possess a stable (weak) solution. Since the flow is turbulent, standard numerical discretizations, like central finite differences or the Galerkin finite element method, have to be extended with additional terms that model the impact of turbulence. The concrete numerical solution depends heavily on the kind of turbulence modeling that is used. In particular, there are turbulence modeling techniques that compute time-averaged (steady-state) flow fields, like some RANS (Reynolds averaged Navier-Stokes) approaches. They are highly efficient and have been used in scientific and industrial applications for more than 40 years. In fact, in agricultural applications, the vast majority of numerical flow simulations rely on RANS approaches, where the turbulence is completely parameterized (see e.g. (Lee et al., 2013) or (Bjerg et al., 2013)). In most cases, proprietary software is used for the flow simulations. Commonly applied commercial codes are, e.g., Ansys (containing Fluent and CFX), StarCCM+, or Comsol.

The transport and dispersion of gases and particles in a turbulent flow is by nature a dynamic (time-dependent) process. Therefore, important information might be lost when transport is modeled with a time-averaged flow field such that the use of time-averaged flow fields could lead to inaccurate results. Transient simulations resolve important unsteady scales and so the dynamic characteristics of a flow. In our opinion, the use of transient simulations is expected to provide more accurate results for the simulation of gas and particle transport. However, the gained accuracy comes at the price of greater computational costs. The use of computer clusters, where several processors solve the problem in parallel, is often necessary. Depending on the license policy of the software companies, this can lead to high costs, e.g., when for every additional processor node, an extra license needs to be paid. Also, many of the computer clusters freely accessible to research institutes do not even offer the use of proprietary software. Moreover, commercial tools do not give access to the source code, thus making it difficult to understand the details of numerical methods used within the solver. Open source solvers can represent here an appealing alternative. Besides being free of charge, they provide complete control over the numerical methods and give also the possibility of customizing the code and implementing tailored methods. In the context of research, they favor exchange of data and source codes, and naturally allow for reproducibility of numerical results by different groups. We also refer to the discussion of the benefits of open source software in the introduction of Wilbrandt et al. (2017). Research in the area of CFD brought in recent years the release of several open source packages for flow simulations, such as, to mention some, OpenFOAM (OpenFOAM, 2016), deal.II (Bangerth et al., 2016), FEniCS (Alnæs et al., 2015), DUNE (Blatt et al., 2016; Dedner et al., 2010). Further available options include the in-house research codes ASAM[2] (Jähn et al., 2014) and ParMooN[3] (Ganesan et al., 2016; Wilbrandt et al., 2017), which are currently developed in the research groups of some authors of this paper. To the best of our knowledge, the only studies using non-commercial solvers to investigate the flow inside or around agricultural buildings are the ones from a research group around Lee and co-workers (see e.g. Lee et al., 2007 or Hong et al., 2017), using OpenFOAM with a steady-state RANS approach, and from a research group around Kateris and co-workers, using Galerkin finite-element-methods with an in-house code written in FORTRAN (Kateris et al., 2012). Studies with transient simulations in the agricultural context are the exception. Only few papers can be found, e.g. Villagran et al. (2019), where 2D transient simulations with Ansys Fluent were carried out with the focus on greenhouses.

This study therefore aims to present a contribution to fill this gap and to investigate the use of transient open source solvers for turbulent flow simulations inside and around agricultural buildings. The main goal of this paper is to demonstrate that open source codes are able to provide accurate simulations of the flow through and around a NVB. Two open source codes with different features are involved in the numerical studies: OpenFOAM and ParMooN. For comparison, a commercial solver (Ansys Fluent) is also included in the study. To this end, a benchmark problem with a typical naturally ventilated barn with cross flow is defined. To achieve the objective of this paper, the benchmark configuration was investigated experimentally in a wind tunnel for obtaining data to compare with. These data sets will be published as well, which we consider to be very useful for the definition of a realistic benchmark problem for turbulent flow simulations. They can be used for assessing turbulent flow solvers in the future. Both codes support different types of grids, discretizations, turbulence models, and solvers. By sharing the experiences made in this study, we hope to reduce eventually existing reservations towards the use of open source codes and promote their application in the agricultural community.

The paper is organized as follows. Section 2 describes the experimental setup to measure the benchmark data set and introduces the corresponding mathematical models, which will be the basis for the numerical studies. Some features of the two considered open source packages as well as the used commercial code will be described in detail in Section 3, while the assessment of the numerical studies is presented in Section 4. Finally, Section 5 contains the conclusions of our investigations.

2. Experimental setup and mathematical model

2.1. The studied dairy barn

The studied building is a naturally ventilated dairy barn with a capacity for 375 cows, located in Northern Germany, near the city of Rostock. The barn has a length of 96 m, a width of , while the roof height varies from 4.2 m at the side walls up to 10.7 m at the gable top. The side walls are completely open, while the gable walls are partially open as sketched in Fig. 1. Further detailed information about the barn can be found in König et al. (2018).

2.2. Wind tunnel setup

Experimental airflow measurements were obtained on a 1:100 scaled model (Fig. 1, right) of the above described barn in the *atmospheric boundary layer wind tunnel* (ABLWT) of ATB Potsdam (Fig. 2). Within the wind tunnel, a fully developed turbulent flow with a logarithmic vertical velocity profile was generated by the presence of roughness elements at the inflow section. The vertical velocity profile and the vertical distribution of turbulence intensity of the generated boundary layer are shown in Fig. 3. The inflow profile fulfilled the criteria for a boundary layer over a moderately rough terrain according to VDI (2000). More detailed information about the ABLWT can be found in Yi et al. (2018). The model was positioned with a 90° angle to the flow direction, so that the barn was under cross-flow.

[2] developed at Leibniz Institute for Tropospheric Research (TROPOS) by Dr.Oswald Knoth

[3] developed at Weierstrass Institute for Applied Analysis and Stochastics (WIAS)

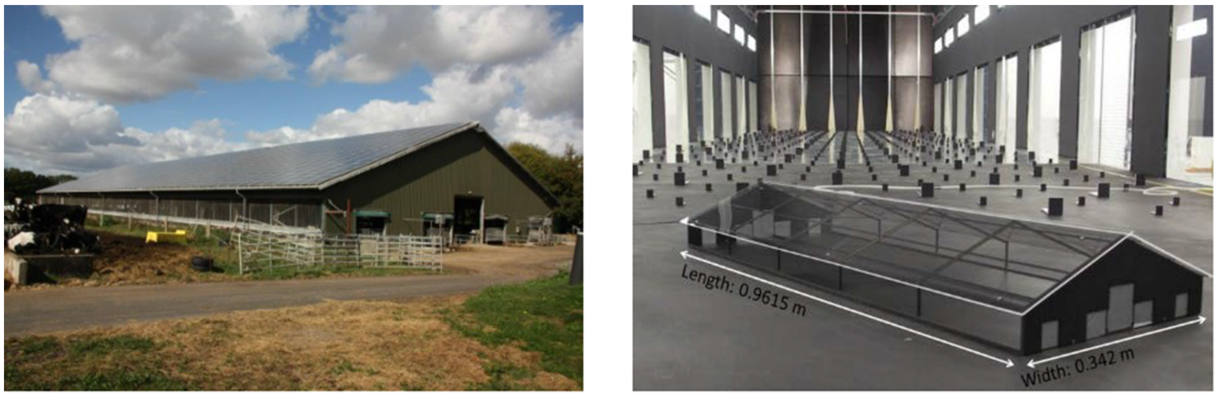


Fig. 1. Left: Real scale barn. Right: 1:100 scale model in the wind tunnel.

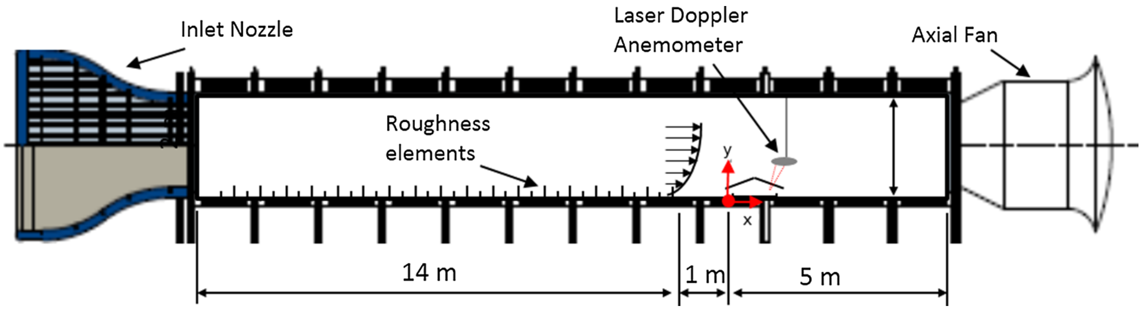


Fig. 2. Sketch of the ABLWT with the scaled model.

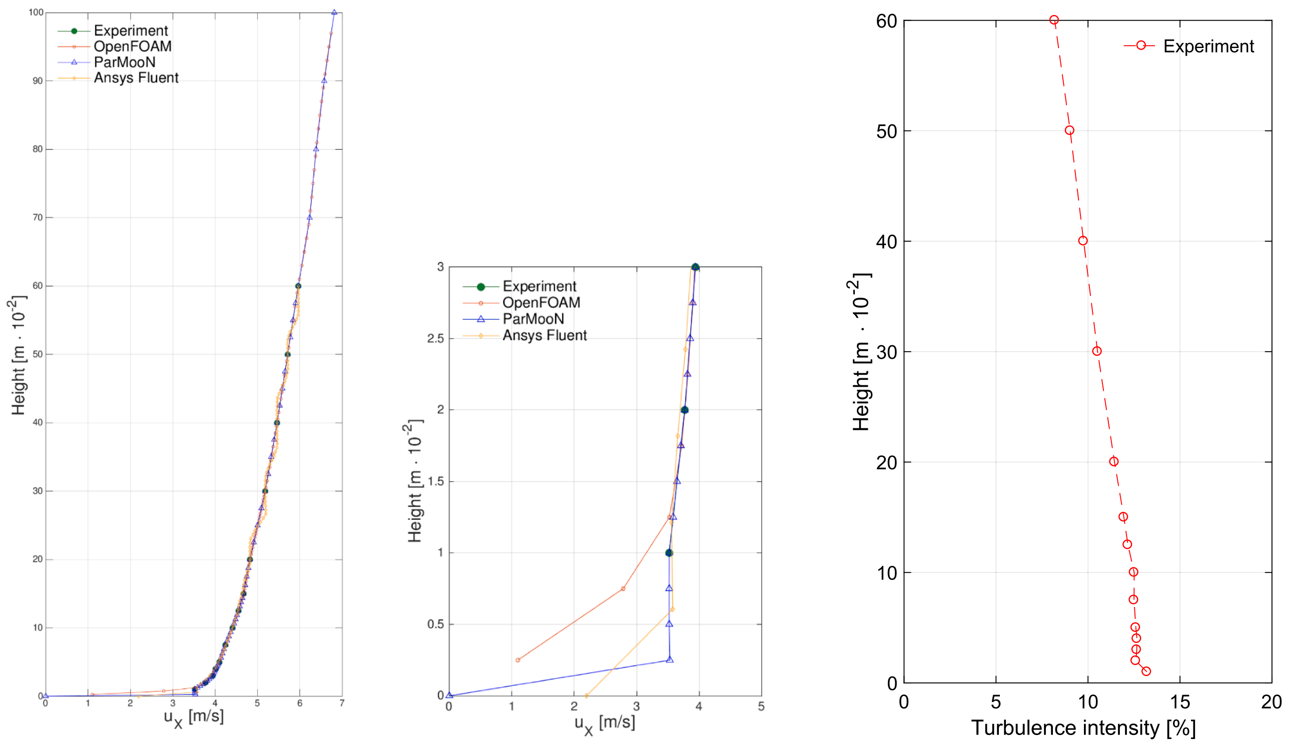


Fig. 3. Left: comparison between the measured inflow profile and the interpolated data. Center: Zoom on the first part (0 to 0.03 meters) of the inlet region. Right: Turbulence intensity at the inflow boundary.

2.3. Velocity measurements

The velocity components along the vertical direction were measured at several vertical sample lines (v1,..,v10) shown in Fig. 4 and their coordinates are given in Table 1. For each point at the sample lines, the velocity components in x-direction (stream-wise) and y-direction (vertical-wise) were measured with a two-dimensional Laser Doppler Anemometer (LDA) (Dantec Dynamics, Skovlunde, Denmark), mounted on a computer-controlled positioning traverse. The LDA measured the velocity with a sample rate between 20 and 100Hz, depending on the

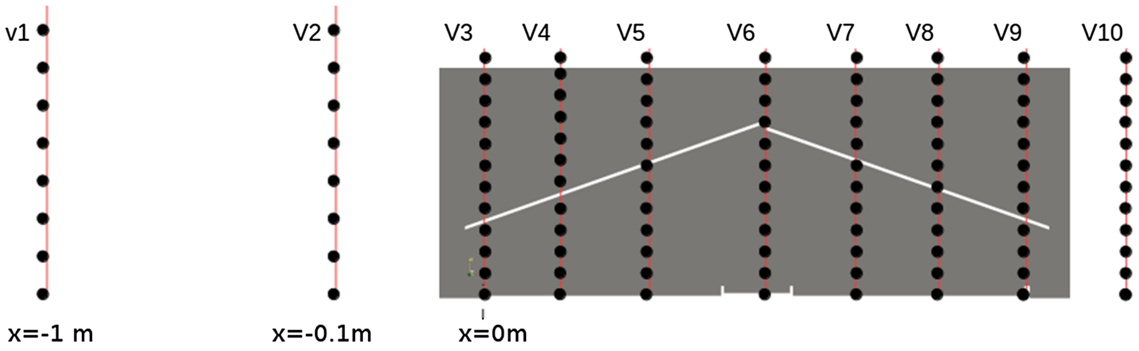


Fig. 4. Visualization of the measured sample lines. All data were acquired in the middle cross-section in the *z*-direction.

density of measured seeding particles. Each point was measured for a duration of 3min, which was found to be sufficiently long enough to capture the features of the flow with an uncertainty for the mean velocities smaller than 2% and for the root-mean-square velocities smaller than 5%.

2.4. Mathematical model

Air is a compressible fluid. However, since the range of velocities observed in the livestock husbandry is much lower than the speed of sound in air, the flow can be considered as incompressible, expecting only a minor impact on the numerical results.

Let $\Omega \subset \mathbb{R}^3$ be the computational flow domain and let $T > 0$ be the final time, which in the simulations was set to $T = 7$ s. Moreover, let us denote with $\boldsymbol{u}\,[\text{m/s}]$ the air velocity (where u and v stand for the horizontal and vertical components, respectively), and with $p\,[\text{Pa}]$ the air pressure. Without any external force, velocity and pressure obey the incompressible Navier-Stokes equations given by

$$\begin{aligned} \partial_t \boldsymbol{u} - \nu \nabla \cdot \mathbb{D}(\boldsymbol{u}) + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p &= \boldsymbol{0} \quad \text{in}(0, T] \times \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in}(0, T] \times \Omega. \end{aligned} \tag{1}$$

The velocity deformation tensor $\mathbb{D}(\boldsymbol{u})$ is the symmetric part of the velocity gradient, i.e., $\mathbb{D}(\boldsymbol{u}) := (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)/2$. In (1), the kinematic viscosity $\nu\,[\text{m}^2/\text{s}]$ is the ratio of the dynamic viscosity $\eta\,[\text{kg/ms}]$ over the density $\rho\,[\text{kg/m}^3]$. For air at 15 °C, these parameters are $\eta = 1.81 \cdot 10^{-5}$kg/ms and $\rho = 1.225$kgm^3, and therefore $\nu = 1.48 \cdot 10^{-5}$m^{2}s.

2.5. Computational domain

The computational domain for the model problem is sketched in Fig. 5, see also Table 2 for additional information. It is a rectangle of 3 m length and 1 m height with the floor and roof geometry of the wind tunnel model. The coordinate system origin is placed on the bottom edge of the windward side of the barn. As already mentioned, the horizontal coordinate is denoted by x and the vertical coordinate by y.

In order to solve the Navier-Stokes Eqs. (1) numerically, the computational domain needs to be discretized by a mesh, i.e., decomposed in a set of polyhedral mesh cells that cover the domain. On the one hand, the size, and consequently the number, of the mesh cells defines the number of degrees of freedom of the discrete problem to be solved. On the other hand, the size of the mesh cells allows for increasing the accuracy of the approximation in regions of interest. Therefore, the computational mesh plays a crucial role in CFD and it has a direct impact on the results of the numerical solvers. In the study presented in this paper, different meshes were employed, whose structure depended on the particular code. For the sake of clarity, details will be described for each code separately in Section 3.

Remark 1. The geometry used for defining the computational model considered a straight roof (Fig. 5). However, due to the weight of the material, the cross section of the roof in the experiment is a slightly convex curve. For this reason, the coordinates of few measurement points were in the experiment on top of the roof but in the simulations below the roof. These points were not considered for the numerical assessment presented in Section 4.

2.6. Initial condition and boundary conditions

In order to obtain a closed system, the Navier-Stokes Eqs. (1) have to be equipped with an initial condition ($t = 0$) and with boundary conditions on the boundary of Ω.

In practice, the initial condition is not known. For this reason, one has to start with an arbitrary initial flow field, to run a simulation until the flow is fully developed, and then to start with monitoring the flow field. Note that the actual initial condition possesses only an impact on the time interval until a fully developed flow field is reached.

At the inlet boundary ($x = -0.5$ m), the velocity profile was prescribed based on experimental data. Concretely, the velocity was measured in the wind tunnel without the scaled barn model at different heights, see Table 3. For the simulations, some interpolation of this profile was used, compare Fig. 3. Further details on the boundary conditions that were applied in the simulations with the individual codes are given in Section 3.

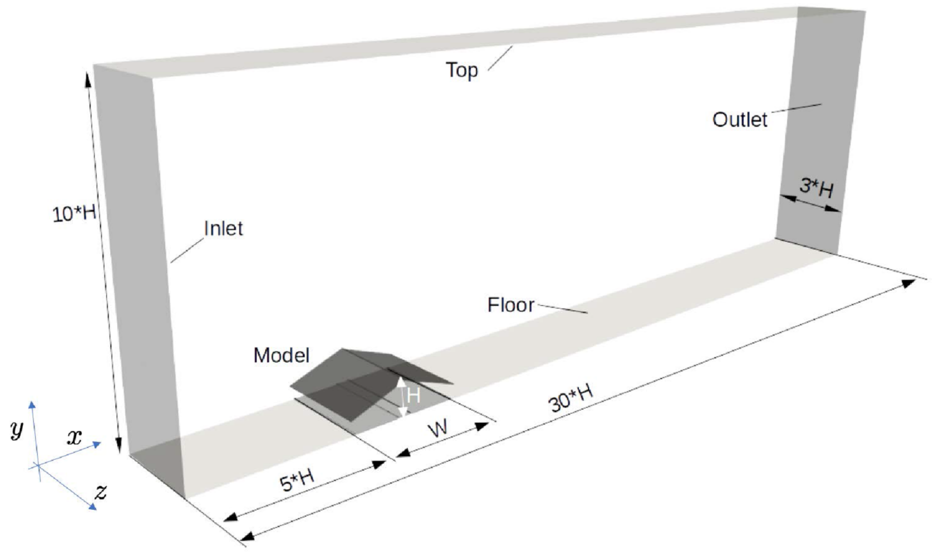
In order to characterize the flow regime, the Navier-Stokes equations can be non-dimensionalized by introducing characteristic length, velocity, and time scales. Choosing as characteristic length scale $L = 0.11$ m (approximate height of the barn) and as characteristic velocity scale $U = 5$ m/s (approximate maximal velocity in a neighborhood of the barn), the Reynolds number of the flow is given by

$$\text{Re} = \frac{LU}{\nu} \approx 37200.$$

This number indicates that the flow is turbulent, requiring therefore suitable numerical methods for its simulation, see Sections 2.8 and 3.

Table 1
Horizontal coordinates and vertical extrema of the measurement sample lines (see also Fig. 4).

	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
x [m]	−1	−0.1	0	0.05	0.1	0.165	0.242	0.292	0.342	0.39
y_{end} [m]	0.6	0.6	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15



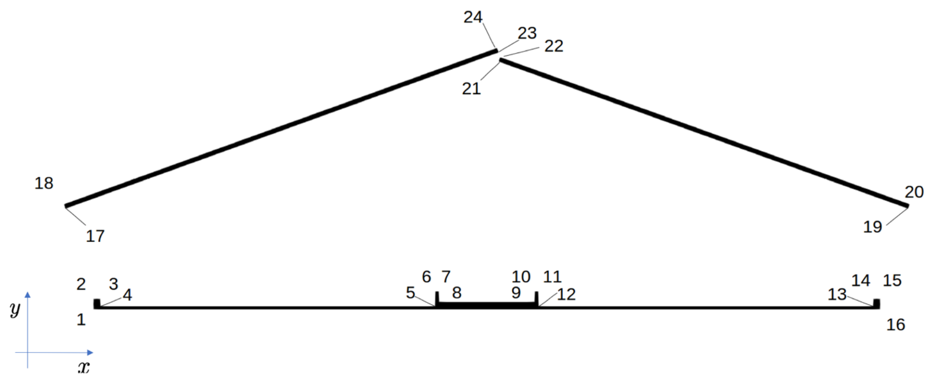


Fig. 5. Top: Sketch of the computational domain. Bottom: Detailed view of the 2D projection of the barn. The numbers 1–24 refer to the edges, whose coordinates are listed in Table 2.

Table 2
Coordinates of the points defining the model geometry (see the sketch in Fig. 5).

Point	x [m]	y [m]
1	0	0
2	0	0.0048
3	0.0032	0.0048
4	0.0032	0.0016
5	0.1485	0.0016
6	0.1485	0.008
7	0.150	0.008
8	0.1503	0.0034
9	0.1917	0.0034
10	0.1917	0.0080
11	0.1935	0.0080
12	0.1935	0.0016
13	0.3388	0.0016
14	0.3388	0.0048
15	0.342	0.0048
16	0.3420	0
17	−0.0120	0.044
18	−0.01253	0.04577
19	0.354	0.044
20	0.35451	0.04562
21	0.1761	0.108
22	0.1767	0.1099
23	0.1761	0.112.0
24	0.1757	0.1138

Table 3
Experimental data from wind tunnel measurements at the inlet.

y[m]	0	0.01	0.02	0.03	0.04	0.05	0.075	0.1	0.125	0.15
$\boldsymbol{u}$[m/s]	0	3.52	3.77	3.94	4.01	4.10	4.24	4.41	4.55	4.67
y[m]	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
$\boldsymbol{u}$[m/s]	4.82	5.18	5.46	5.71	5.96	6.23	6.38	6.57	6.81	

2.7. Time interval

The choice of the final time is briefly motivated in this section. The quantities of interest are time-averaged velocity profiles. Hence, one needs a sufficiently long time interval for obtaining statistically converged results. The area of interest is inside the barn, which has a maximum height of $H \approx 0.11$ m. According to the inflow profile, see Table 3, an air parcel starting at the half of this height has a velocity of about $u = 4$ m/s. Consequently, the parcel passes the whole length of the domain ($30H$) in an interval of time of approximately 0.8 s. The area of interest inside the barn has a width of $W = 0.34$ m. The given parcel passes this width around three times in one second. Based on these considerations, we assumed that a fully developed full profile can be obtained within a time interval of 1 s. Furthermore, a time interval of 6 s is assumed to be sufficient to achieve statistically converged velocity profiles. These estimates were validated a posteriori by the results of our numerical simulations.

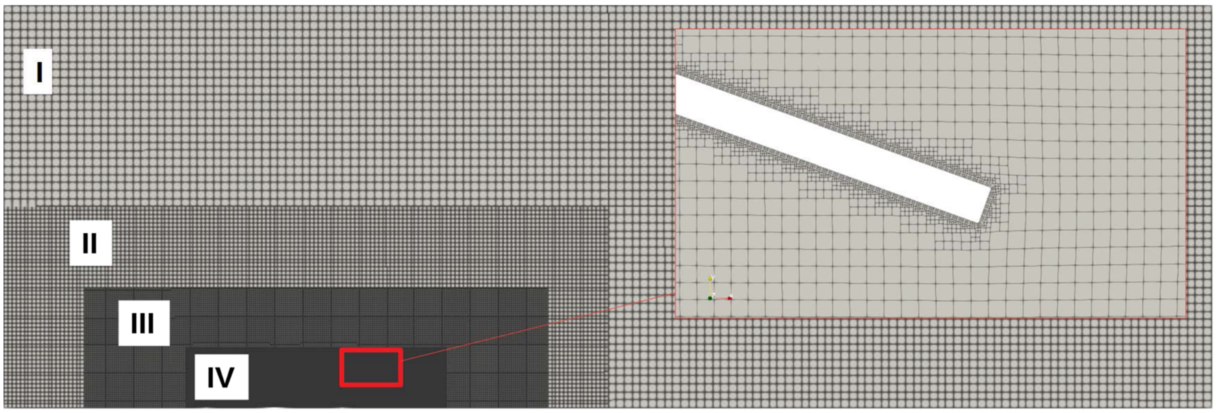


Fig. 6. Domain meshed for OpenFOAM with snappyHexMesh.

2.8. Remarks to turbulence modeling

There is no mathematical definition of what turbulence is. A flow is considered to be turbulent when its dynamics possess a wide spectrum of scales (eddies) – ranging from large scales to very small scales – and with the very small scales being of utmost importance for the physical character of the flow (energy dissipation). The Navier-Stokes Eqs. (1) are a proper mathematical model for describing such flows.

Standard discretizations of the Navier-Stokes equations, like the Galerkin finite element method or central finite differences, try to resolve all important scales of the flow. However, the ability of a numerical method to approximate the flow dynamics of small scales depends on the level of resolution of the spatial discretization, i.e., the computational grid. In particular, most of the important scales in turbulent regimes are usually so small that it is not even possible to represent them on computationally affordable grids. These scales are called *unresolved* scales. Of course, scales that cannot be represented cannot be simulated. Because of these unresolved scales, standard discretizations fail for the simulation of turbulent flows, which usually results in a blow-up of the numerical simulations. The remedy consists in augmenting standard discretizations by including so-called turbulence models, which have the purpose to account for the impact of the unresolved scales onto the simulated (resolved) scales. From the numerical point of view, turbulence models introduce additional viscosity into the discrete problem.

Turbulence modeling has been an active field of research for more than forty years. Although numerous turbulence models were proposed, there is neither a standard model nor, in some sense, a best model. There are models, whose derivation is based on physical insight in turbulent flows and there are models, which were derived purely with mathematical arguments. Considering all the different motivations, assumptions, and approximations behind the derivation of turbulence models, it is therefore not surprising that different numerical results can be obtained using different turbulence models.

Commercial codes, e.g., as used for the simulations presented in Saha et al. (2011) and Shen et al. (2013), provide in general classical two-equation turbulence models, like the k-ϵ and k-ω model. The properties of these models, in particular their shortcomings, are well described in (Chapters 10 and 11 Pope, 2000). The codes used in the study presented in this paper offer the possibility to use turbulence models of different types, which will be explained briefly in the description of the individual codes.

3. Used CFD software

This section provides some information on the three software packages that were used in our study as well as on specific choices in the setup of the numerical simulations. The open source solver OpenFOAM is widely used by scientists and engineers for flow simulations. The other open source code ParMooN is a more specialized in-house research code. The third solver is the popular commercial package Ansys Fluent.

3.1. Open source package OpenFOAM

OpenFOAM (Open Source Field Operation And Manipulation) is an open source software package containing different applications to model and simulate problems in fluid dynamics, The OpenFOAM Foundation (2016). It is written in C++ and designed in an object-oriented fashion that allows the choice amongst many solvers for both the compressible or incompressible Navier-Stokes equations, including also RANS or Large Eddy Simulation (LES) turbulence models. Since OpenFOAM is a popular open source code and all information are readily available at The OpenFOAM Foundation (2016), we decided to provide here only a brief description of this solver that concentrates on those aspects that are important for our numerical studies.

Meshing can be done in several ways, either by integrated meshing routines or by importing meshes with external (open or closed source) meshing routines like e.g. Gmsh, GAMBIT or SALOME. For this study, the domain was decomposed and meshed with the snappyHexMesh utility, which generates 3-dimensional meshes containing mainly hexahedra and split-hexahedra cells automatically from triangulated geometries in.stl format (e.g. Gisen, 2014). The domain consisted of 4 refinement boxes (I, II, III and IV) as depicted in Fig. 6. The refinement

Table 4
Grid parameters for the coarse, medium and fine grid used in OpenFOAM. 'Size' refers to the edge length in the respective refinement boxes. 'Distance wall' is the distance of the first grid cell's midpoint to the wall in the roof region. y^+ values were chosen as the maximum values from the time-averaged solution at the roof walls. The number of degrees of freedom corresponds to the number of mesh cells.

Mesh	N^o cells	Size I [mm]	Size II [mm]	Size III [mm]	Size IV [mm]	Distance wall [mm]	y^+
Coarse	1,434,580	20	10	5	2.5	1	6–14
Medium	4,608,675	10	5	2.5	1.25	0.4	4–8
Fine	14,188,950	5	2.5	1.25	0.625	0.4	4–8

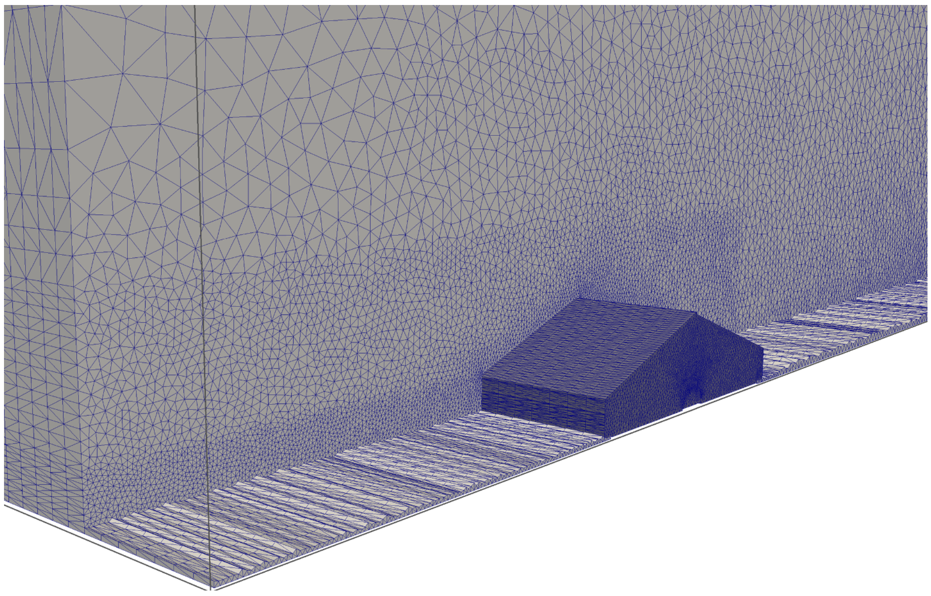


Fig. 7. Cut view of Grid 4 used in ParMooN.

was factor 1 for each box, meaning that the edge length of a cell in box IV (inside and nearby the barn) was 8 times smaller than the original edge length in box I. Table 4 provides information on the used grids. (See Fig. 7).

For the simulations presented in this paper, incompressible LES was setup using the pimpleFoam solver, which merges the well-known PISO (Pressure-Implicit with Splitting of Operators, Issa, 1986) algorithm with the SIMPLE (Semi-implicit Method for Pressure Linked Equations) algorithm, Patankar and Spalding (1983), resulting in a fast convergence for transient simulations, described, e.g., by Holzmann (2016). An adaptive time stepping was chosen with the constraint of a Courant number not higher than 3, resulting in time steps in average of $2{\cdot}10^{-4}$ s.

The subgrid scale turbulence was modeled with a one equation eddy viscosity model (`simulationType:LES, LESModel:kEqn`), where the not resolved scales are solved similarly to common RANS approaches with an additional equation for the turbulent kinetic energy, described in detail by Yoshizawa (1986). The spatial discretization was done using second order `linearUpwind` schemes, the time variable was discretized with the second order `backward` scheme, both described in The OpenFOAM Foundation (2016).

The experimentally derived time-averaged velocity profile, shown in Fig. 3, was mapped as boundary condition onto the inlet face for the velocity, while the pressure was defined with a Dirichlet `zeroGradient` boundary condition. At the outlet, the boundary condition for velocity was set to `zeroGradient` and the pressure to `fixedValue`.

All simulations were performed at the *North-German Supercomputing Alliance* computer cluster (HLRN) on the Cray-MPI system, using Intel Xeon Haswell compute nodes with 2500 MHz CPUs. For the simulation of the medium sized mesh, the domain was decomposed with the `scotch` algorithm and distributed with `openMPI` on 120 CPUs. The computation time for reaching 7 s was around 8.6 h.

To make the computation times of the different codes comparable in some way, we introduce the unit of CPU hours [CPUh], which is the computing time calculated down to one single CPU. Consequently, to reach 7 s in flow time, the 8.6 h computing time on 120 CPU have an equivalent of 8.6 h * 120 CPU = 1032 CPUh.

3.2. Open source package ParMooN

The software ParMooN (Parallel Mathematics and object-oriented Numerics) (Ganesan et al., 2016; Wilbrandt et al., 2017) is a C++ finite element library, based on a hybrid MPI/OpenMP parallelization, developed for numerical simulations of partial differential equations from fluid dynamics. This parallel version builds upon the code MooNMD (John and Matthies, 2004), which was used for performing simulations for more than one hundred journal papers. The development of MooNMD paid a special emphasis on implementing turbulence models for incompressible flow simulations, both for academic benchmark problems and applications.

Currently, more than 250 finite elements in two or three dimensions are implemented in ParMooN, including high order polynomials, bubble functions, and discontinuous elements. The software supports both quadrilateral/hexahedral and triangular/tetrahedral meshes. Unstructured simplicial grids in 2D and 3D can be provided using the MEDIT `.mesh` format, which is supported by several established mesh generation packages such as Gmsh (Geuzaine and Remacle, 2009) or TetGen (Si, 2015).

Concerning the temporal discretization, ParMooN supports methods with different derivations, complexity, and accuracy. A first group of methods, so-called θ-schemes, include the explicit and the implicit Euler methods as well as the Crank-Nicolson scheme. Furthermore, the fractional-step θ-scheme and the backward differentiation formula of second order (BDF2) can be applied. Moreover, higher order variational type time stepping schemes that include the continuous Galerkin-Petrov (k) (cGP(k)), for $k \geqslant 2$, and the discontinuous Galerkin(k) (dG(k)), for $k \geqslant 1$, which are accurate of order $k+1$, are available and have been successfully applied to several classes of problems.

Linear solvers implemented in ParMooN include iterative approaches (Krylov subspace methods) and several preconditioners, including geometric multigrid methods. Moreover, an extended choice of direct and iterative methods is available via the PETSc library Balay et al. (2016). This library offers, among others, the Boomer AMG method and the parallelized sparse direct solver MUMPS (Amestoy et al., 2006).

ParMooN supports a number of turbulence models, in particular some LES models and variational multiscale (VMS) methods, e.g., see Ahmed et al. (2017) and John (2016).

Altogether, ParMooN is a code whose development is driven by the state-of-the-art research on problems in numerical analysis and scientific computing. It offers many discretizations in time and space as well as many solvers and it is designed such that extensions in many directions are possible. The drawback of this flexibility is that the implementations might not be tailored to be very efficient for specific

Table 5
Information on the grids and the degrees of freedom (dof) on these grids used with ParMooN. All grids were especially refined within and around the barn (see also Fig. 7).

	Cells	dof velocity	dof pressure	Description
Grid 1	697 500	2 933 685	128 352	Medium refinement
Grid 2	895 770	3 757 479	163 968	More refinement in front and around the barn
Grid 3	995 985	4 175 142	182 080	Similar to Grid 1, but overall somewhat more refined
Grid 4	1 014 030	4 251 681	185 456	Similar to Grid 2, but even more refined in front and around the barn

problems or discretizations. For instance, all integrals are evaluated via a transform to a reference mesh cell, which is advantageous for the flexibility of using general finite elements, but which introduces some computational overhead. This approach is not necessary if, e.g., only linear finite elements are implemented, since integrals can be evaluated efficiently on the physical mesh cells in this case.

In the following, the setup of the simulations with ParMooN is described. The domain was decomposed with unstructured tetrahedral grids that were generated with Gmsh. The three-dimensional grid was obtained from an unstructured triangular grid in the (x, y)-plane, extruding it via several layers in the z-direction.

Several grids with different resolution were used, see Table 5 for information on the number of mesh cells. In all cases, a rather coarse grid was used in the bulk of the domain. In the neighborhood of the barn, the grids became gradually finer and the highest refinement was within and closely around the barn.

For the spatial discretization, the popular inf-sup stable pair of finite elements P_2/P_1, a so-called Taylor–Hood pair, was utilized. That means that the discrete velocity is a continuous function, piecewise quadratic on each mesh cell, and the discrete pressure is a continuous function, piecewise linear on each mesh cell.

As time stepping scheme, the Crank-Nicolson scheme was utilized. It turned out that the length of the time step $\Delta t = 2.5\cdot 10^{-4}$s was an appropriate choice in terms of computational time and sensitivity of the results. Hence, the simulation of the whole time interval of 7 s required 28000 time steps. A fully implicit approach was used. The stopping criterion for the solution of the nonlinear problem in each time instant was that the Euclidean norm of the residual vector was below 10^{-5}. This criterion was usually satisfied after one iteration. The arising linear problems were solved with a flexible GMRES (FGMRES) method (Saad, 1993) and the so-called least squares commutator (LSC) preconditioner (Elman et al., 2014) was applied. This preconditioner has been proven to be very efficient for time-dependent incompressible flow problems in the recent study (Ahmed et al., 2018). Its application was similar as described in Ahmed et al. (2018), i.e., the arising pressure Poisson problems were solved directly using MUMPS and the velocity subproblems were solved inexactly with an iterative method (GMRES with SSOR preconditioner, relaxation parameter $\omega = 1$).

A crucial algorithmic component for the simulations of flows through the barn is the turbulence model. For the simulations with ParMooN, a popular LES model, the Smagorinsky model (Smagorinsky, 1963), was used. The motivation for choosing this model consists in demonstrating that with an easy-to-implement extension of an existing solver for laminar flow problems, it is possible to perform simulations also for quite challenging applications. In fact, for the Smagorinsky LES model, the only extension consists in replacing the viscous term of the Navier-Stokes equations (1) by $\nabla\cdot((\nu + \nu_{\mathrm{T}})\mathbb{D}(\boldsymbol{u}))$ with the turbulent viscosity ν_{T} given by

$$\nu_{\mathrm{T}} = C_{\mathrm{Sma}}\delta^2\|\mathbb{D}(\boldsymbol{u})\|_F,$$

where C_{Sma} is the user-chosen Smagorinsky constant, δ is the so-called filter width, and $\|\mathbb{A}\|_F$ is the Frobenius norm of a tensor $\mathbb{A} = (a_{ij})_{i=1}^3$ defined by $\|\mathbb{A}\|_F = (\sum_{i,j=1}^3 a_{ij}^2)^{1/2}$. The filter width δ is a measure of the locally smallest resolved scales of the flow. Thus, it is linked to the local mesh width. One can use different measures for the local mesh width. In our experience, e.g., (John, 2016, Example 8.128), the length of the shortest edge of a mesh cell K is an appropriate unit for the Smagorinsky model and δ should be set on K by two times the length of the shortest edge of K. In fact, we observed that the results obtained with this filter width were much more accurate than using two times the diameter of K. For brevity, studies with respect to the choice of the filter width will not be presented in this paper. Typical values for the Smagorinsky constant C_{Sma} in academic benchmark problems are of order 0.01. Smagorinsky constants of this order were also used in the simulations presented in this paper. Since the considered flow is more complex than in usual academic test problems, we could observe that smaller constants than 0.01, depending on the grid even 0.01, resulted in a blow-up of the simulations. In the considered application, the flow field in front of the barn is much less complex than in and after the barn, since there are no big vortices in front of the barn. To account for this difference and to reduce the viscous effect of the Smagorinsky model in front of the barn, the used Smagorinsky constant was scaled with 10^{-2} for $x \leqslant -0.05$ m.

The inlet boundary condition was constructed on the basis of the experimental data, see Table 3. In particular, inlet velocity values were interpolated between the measured points, except for the points between $y = 0$ m and $y = 0.01$ m, where a constant value, which was equal to the measured velocity at height $y = 0.01$ m, was assigned to all degrees of freedom. As usual in simulations of turbulent flows, boundary layers cannot be resolved, in particular the boundary layer at the bottom. A preliminary numerical study, whose results are omitted for brevity, showed that one gets a quite smeared boundary layer of the computed solution already in front of the barn, at sample line v2, if a linear interpolation is used in this interval. With the described approach, a notable improvement could be obtained. At the top and lateral boundaries, the free-slip condition was imposed. At the outlet boundary ($x = 2.5$ m), a stress-free condition (the so-called do-nothing condition) was applied, i.e., by imposing $(2\nu\mathbb{D}(\boldsymbol{u}) - p\mathbb{I})\cdot\boldsymbol{n} = 0$,where $\mathbb{I}$ is the unit tensor and $\boldsymbol{n}$ the outward pointing normal vector at the outlet. This condition states that the flow should leave the computational domain in the form it is arriving there. The do-nothing condition is a standard approach at outlets, in particular, in situations where no other information on the downstream domain is available. Concerning the roof boundaries and the bottom boundary, the no-slip condition $\boldsymbol{u} = \boldsymbol{0}$ m/s was utilized.

The inlet condition was extrapolated horizontally in the domain and the resulting function was the initial condition for all simulations with ParMooN. At the time 1 s, a fully developed flow field was reached and the collection of the data was performed in the time range $[1, 7]$ (s). The comparison with different time intervals, e.g., $[1, 6]$ s or $[2, 7]$ s, showed that the obtained results can be considered to be statistically converged.

All simulations with ParMooN were performed on compute servers HP BL460c Gen9 2xXeon, Fourteen-Core 2600 MHz, using 50 processors. The simulation of one time step, including the calculation of the quantities of interest, took around 10 s, such that the computation for the whole time interval took between 80h and 100h, which corresponds to an average duration of 4500CPUh. Although this computing time is significantly longer than for OpenFOAM, it is not straightforward to draw conclusions based on the CPU hours. First of all, the simulations were performed on different hardware architectures (high

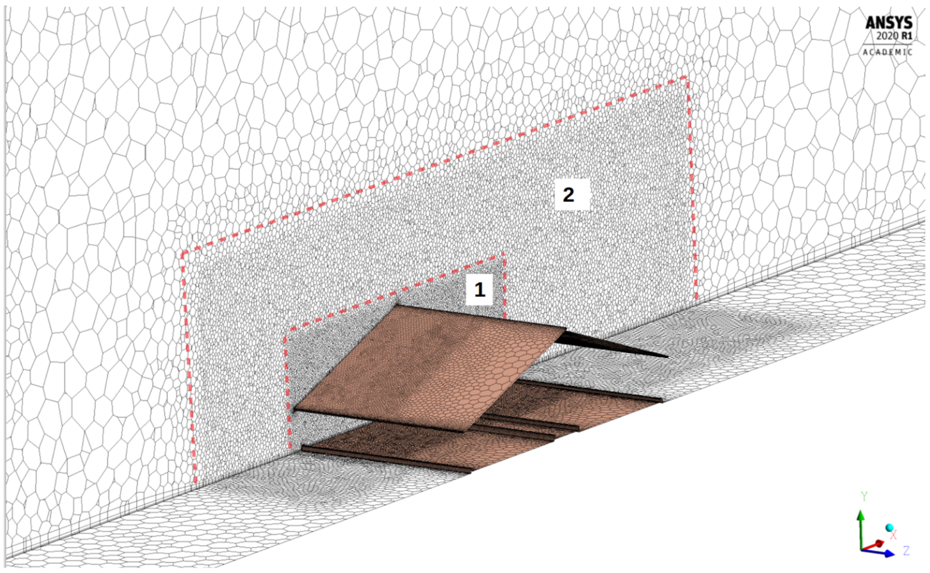


Fig. 8. Cut view of the grid used for the Ansys Fluent simulations. Red marked Sections 1 and 2 show the refinement regions around the barn.

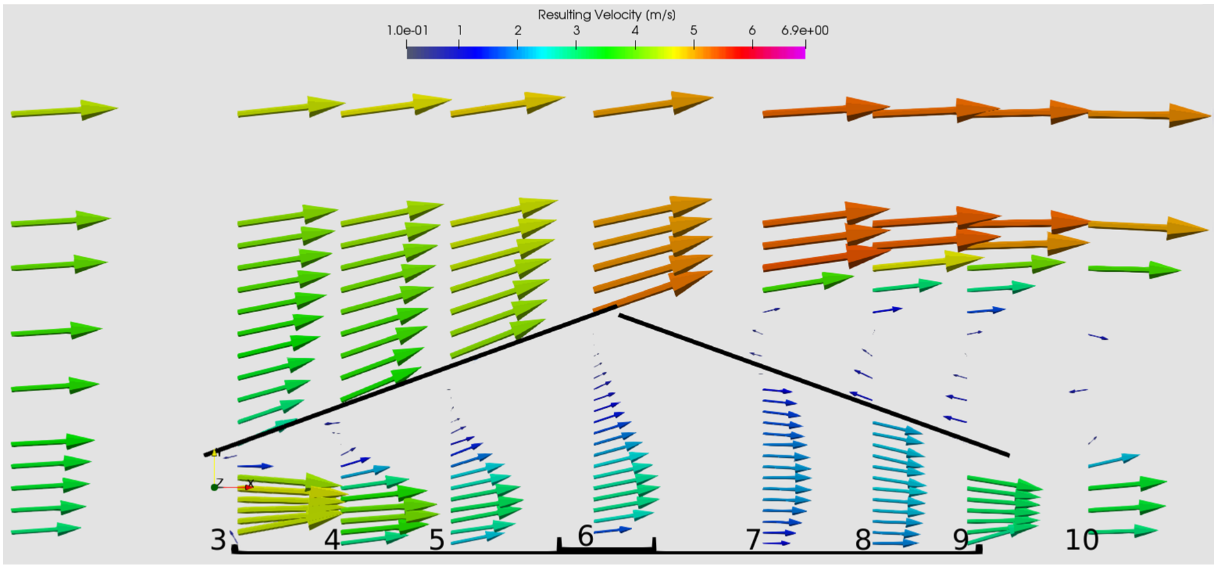


Fig. 9. Experimental results for the time-averaged velocity field. The 2D vectors represent the resulting velocity from the measured vectors in x and y direction. Their origin is at the respective sampling location on the vertical sampling lines, labeled with numbers 3–10.

performance compute cluster with distributed memory vs. workstation with shared memory). Second, since the essential goal of the numerical studies was to compute accurate results, the simulations were not optimized with respect to efficiency, e.g., by using weaker stopping criteria for iterative solvers. Finally, also the feature that ParMooN is a flexible research code and not tailored for the considered class of problems, as explained above, of course contributes to the computing times. For all these reasons, in particular the first one, the CPU hours cannot be considered as an accurate measurement of the codes' efficiency. They are rather used here for providing a rough comparison.

3.3. Ansys Fluent

To have a comparison between open source and commercial software, simulations were also carried out with the well-known Ansys Fluent solver. For the mesh, the dimensions from the OpenFOAM middle mesh were taken for the cell size and near-wall meshing. Due to license limitations, a mesh study was not conducted.

The processes of creating the geometry, meshing, solver setup, calculation, and post-processing were done within the simulation environment Ansys Workbench. The Ansys Meshing software was used to generate an initial unstructured tetrahedral mesh with an inflation layer on the ground and on the model boundaries. The distance of the first inflation layer was set to 0.4 mm, with a number of 5 layer and the *smooth transition* option. Two boxes around the barn were created, where the initial cell size of the domain of 20 mm was gradually refined. In the outer box, the cell size was set to 2.5 mm, in the inner box, the cell size was set to 1.25 mm, see Fig. 8. After the initial meshing, the Ansys Fluent meshing algorithm was used to convert the mesh into a polyhedral mesh, consisting of a total number of 1.950.000 cells.

The experimental values for the velocity in x-direction (Fig. 3) were taken as boundary condition for the velocity at the inlet. At the bottom, the top and the walls of the domain, no-slip conditions were set. On the side walls, symmetry conditions were applied.

LES was performed, with the *Wall-Adapting local Eddy Viscosity* (WALE) model as a subgrid scale turbulence model. Default settings

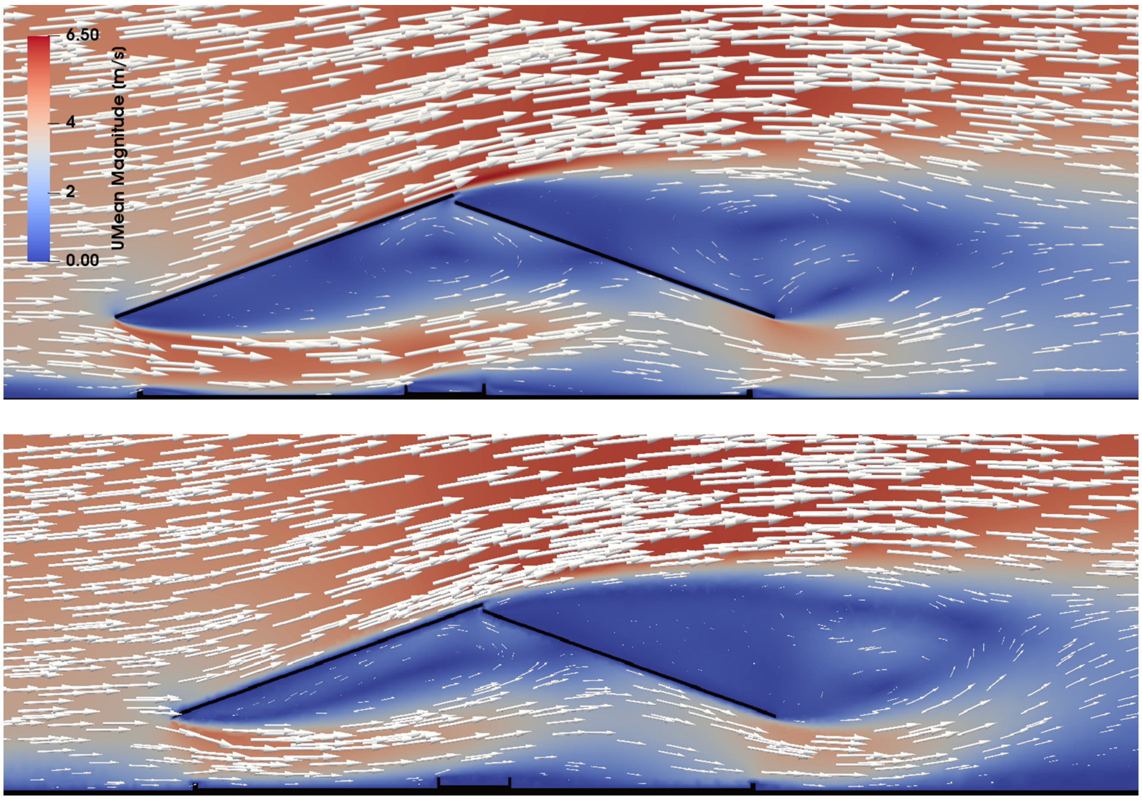


Fig. 10. Time-averaged velocity fields for OpenFOAM (top) and ParMooN (bottom). Results were obtained with the finest considered discretization (fine grid for OpenFOAM and Grid 4 for ParMooN).

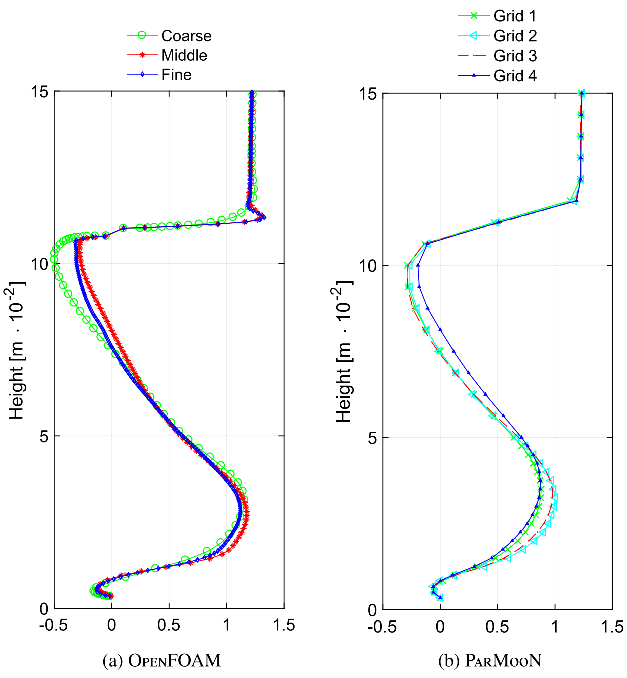


Fig. 11. Grid independence study. Shown are the normalized velocities in x-direction at sample line v6, see Fig. 4. Left: OpenFOAM, right: ParMooN.

were utilized for the near-wall treatment, which is a law-of-the-wall approach dependent on the respective y^+-value. For further descriptions, see the Ansys Fluent theory guide.

As solver scheme, the SIMPLE algorithm was chosen. For the spatial discretization, the *Least Squares Cell Based* scheme for Gradients, the *Second Order* scheme for pressure, and the *Bounded Central Differencing* scheme for momentum were used. As temporal discretization, the *Bounded Second Order Implicit* scheme was applied. The above described settings were chosen following the recommendations from the Ansys Fluent interactive guidance software. A time step of $\Delta t = 1.5 \cdot 10^{-4}$ s was taken and the number of iterations per time step was set to be 8. The simulations were carried out at the HLRN computer cluster, described in Section 3.1, using 80 processors, which resulted in an average computation time of around 2.47 s per time step and an overall simulation time of 32h. This corresponds to 2560CPUh, which is approximately twice the duration of OpenFOAM (1032CPUh) and half the duration of ParMooN (~4500CPUh).

4. Results and discussion

This section presents the results of the numerical simulations and compares them with the data obtained in the experimental campaign. The experimental data can be accessed as open source data set (Janke, 2020).

4.1. Experimental results

The measured velocities in the ABLWT experiments are qualitatively shown in Fig. 9. Each sampling point is the origin of the time-averaged 2D velocity vector measured by the LDA, where the length and color of the arrows represent the magnitude of the vector. The following flow pattern attributes can be observed:

(1) when the flow enters the barn at the inlet, it is accelerated and the vectors are directed towards the center line of the opening.
(2) In the first half of the barn under the roof (at sampling lines 3 until 5), a small re-circulation zone in anti-clockwise direction can be seen.

Table 6
Comparison (experimental vs. numerical simulations) of the volume flow through the barn. The flow has been calculated interpolating first the numerical results on the measurement coordinates and then approximating the surface integral with a first order quadrature rule.

	Experimental [m^3s^{-1}]	OpenFOAM [m^3s^{-1}]	ParMooN[m^3s^{-1}]	Ansys Fluent [m^3s^{-1}]
Model inlet (v3)	$13.60{\cdot}10^{-2}$	$13.44{\cdot}10^{-2}$ (−1.2%)	$13.77{\cdot}10^{-2}$ (+1.2%)	$14.55{\cdot}10^{-2}$ (+7.0%)
Model outlet (v9)	$13.40{\cdot}10^{-2}$	$13.84{\cdot}10^{-2}$ (+3.3%)	$13.86{\cdot}10^{-2}$ (+3.4%)	$14.62{\cdot}10^{-2}$ (+9.1%)

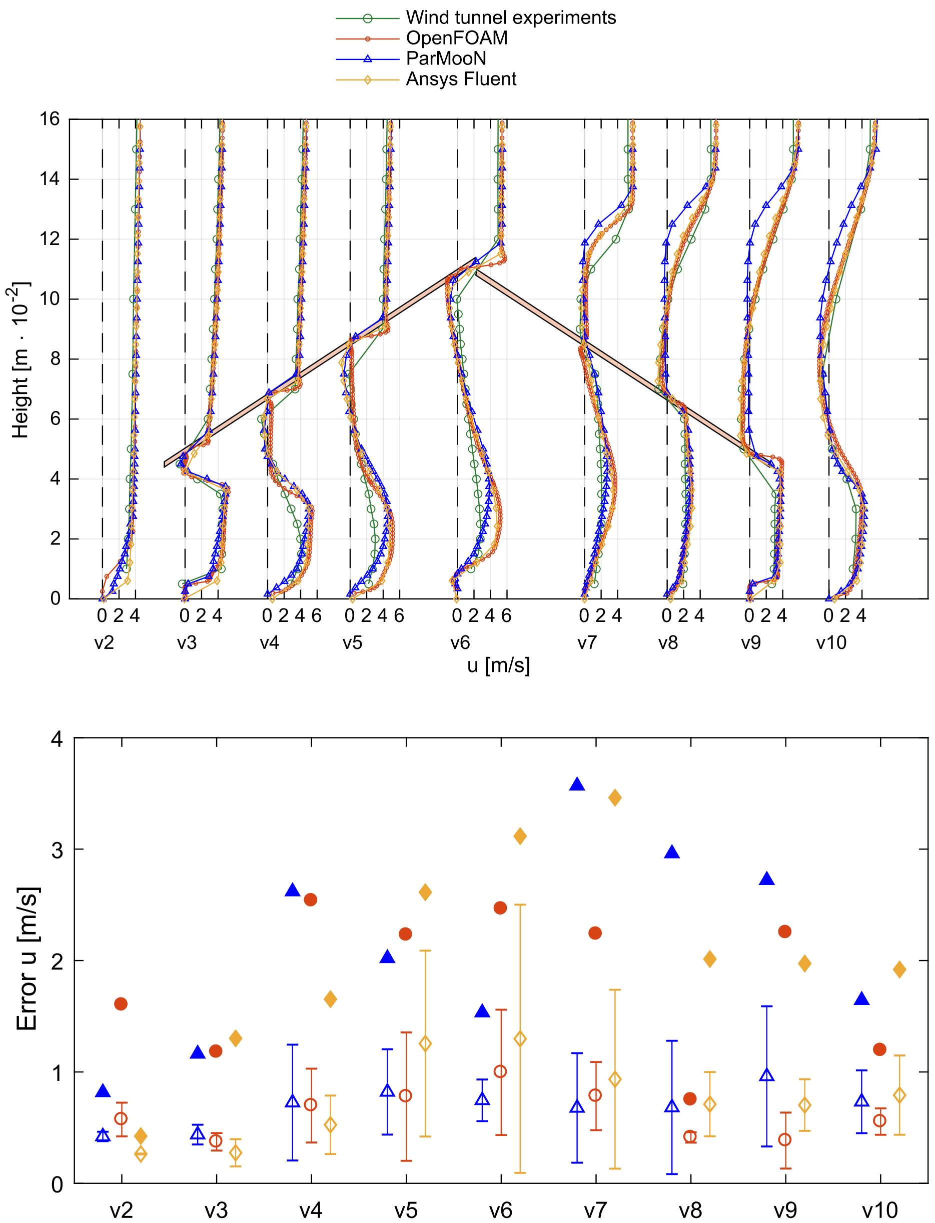


Fig. 12. Comparison between experimental data and numerical results for the horizontal velocity (top) with corresponding error statistics (bottom): bold symbol – maximal error, open symbol – average error, interval – standard deviation. Note the effect of the slightly different form of the roof in the experiment and the simulations explained in Remark 1.

(3) After having passed the inlet towards the middle of the barn (sampling lines 4, 5 and 6), the vertical component of the flow is positive, resulting in a drift towards the roof.
(4) From the middle of the barn towards the outlet (sampling lines 7 and 8), the vertical component is negative, resulting in a drift towards the floor.
(5) Outside the barn over the downwind side half of the roof, a large re-circulation zone in clockwise direction has formed.

Yi et al., 2018 conducted wind tunnel experiments with a setup similar to this study, where the flow inside and around a NVB under cross-wind direction and different opening geometries was studied. For an opening configuration comparable to this study, attributes (1), (3), (4), and (5) could also be observed in their study. However, the re-circulation zone described as attribute (2) could not be observed. This is probably due to the different scale of the model, which was 1:40 in the study of Yi et al. (2018) and resulted in a larger opening.

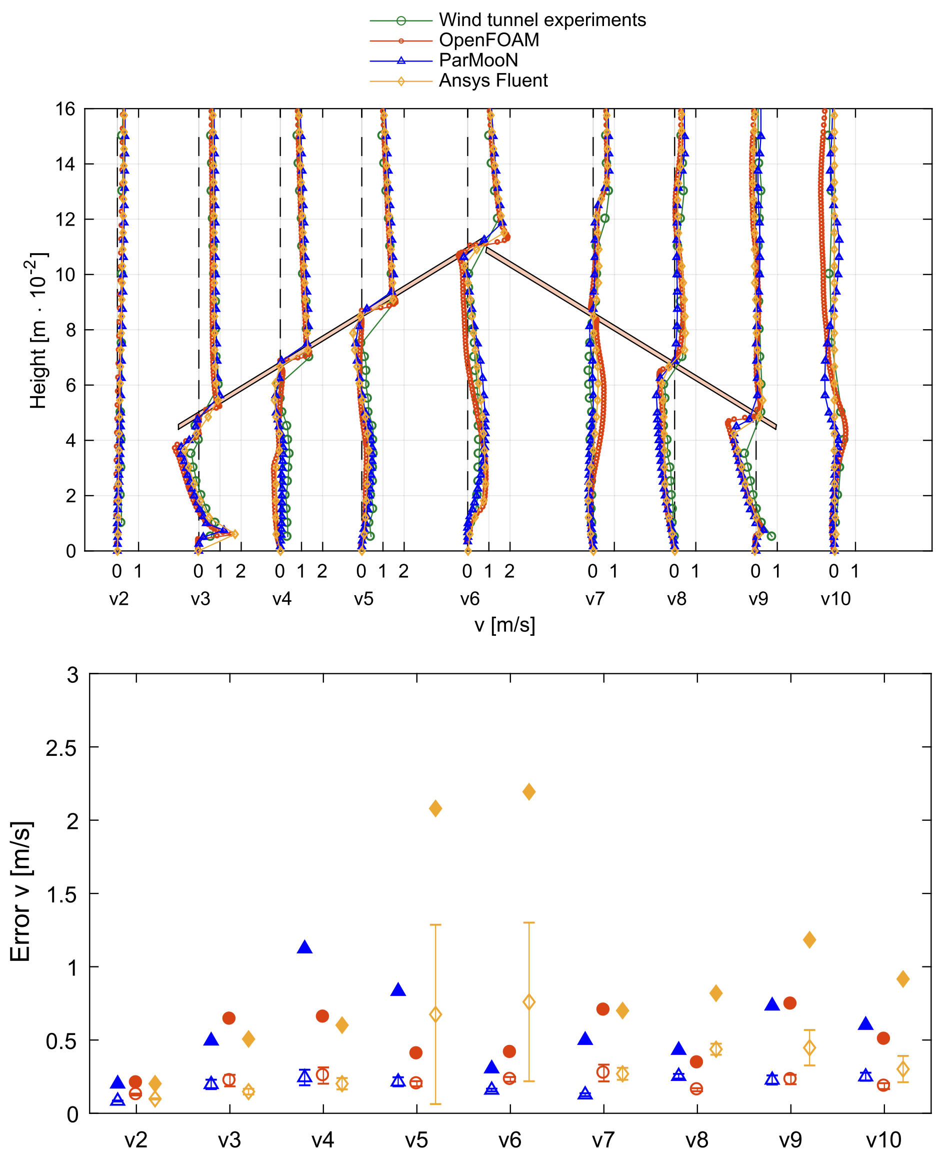


Fig. 13. Comparison between experimental data and numerical results for the vertical velocity (top) with corresponding error statistics (bottom): bold symbol – maximal error, open symbol – average error, interval – standard deviation. Note the effect of the slightly different form of the roof in the experiment and the simulations explained in Remark 1.

Several on-farm measurement campaigns focusing on the inside air flow pattern were conducted in the NVB that served as model for the 1:100 scale model in this study. For wind situations with an orthogonal inflow, attributes (1), (3), and (4) were observed by (Fiedler et al., 2013; Fiedler et al., 2014) at the on-farm measurements. Hempel et al. (2015) conducted additional measurements under the roof and observed also the anti-clockwise re-circulation zone as described in attribute (2).

4.2. Simulation results: air flow pattern

As a first qualitative validation of the open source solvers, Fig. 10 shows the simulated time-averaged 2D air flow patterns inside and around the barn on the symmetry plane of the computational domain. The upper picture shows results for OpenFOAM and the lower picture shows results for ParMooN. Both pictures, which are qualitatively very similar, use the same color scale for the resulting velocity. The white arrows represent the respective velocity vectors with their length scaled by the resulting velocity.

The previously described air flow pattern attributes (1) to (5) from the wind tunnel experiments are reproduced by both codes. In particular, the re-circulation zone under the roof (attribute (2)) and the large re-circulation zone over the downwind roof side (attribute (5)) are clearly visible. Regarding attribute (5), both codes produce two vortices forming the re-circulation zone, one larger in anti-clockwise direction, and under that, probably originating from the right roof edge, a smaller vortex in clockwise direction. The formation of these two vortices is more pronounced in the OpenFOAM simulations. Since the experimental measurements were not resolved sufficiently fine in this region to detect this feature, this is an example where numerical simulations provide more information of the flow field than experimental data.

4.3. Simulation results: vertical sampling lines

The simulation results were compared to the experimental results at the sampling points on the vertical sampling lines sketched in Fig. 4. In particular, we compared the time-averaged horizontal velocity ($\bar{u}$), the time-averaged vertical velocity ($\bar{v}$) and the corresponding root-mean-square (rms) of their turbulent fluctuations u_{rms} and v_{rms}, defined as:

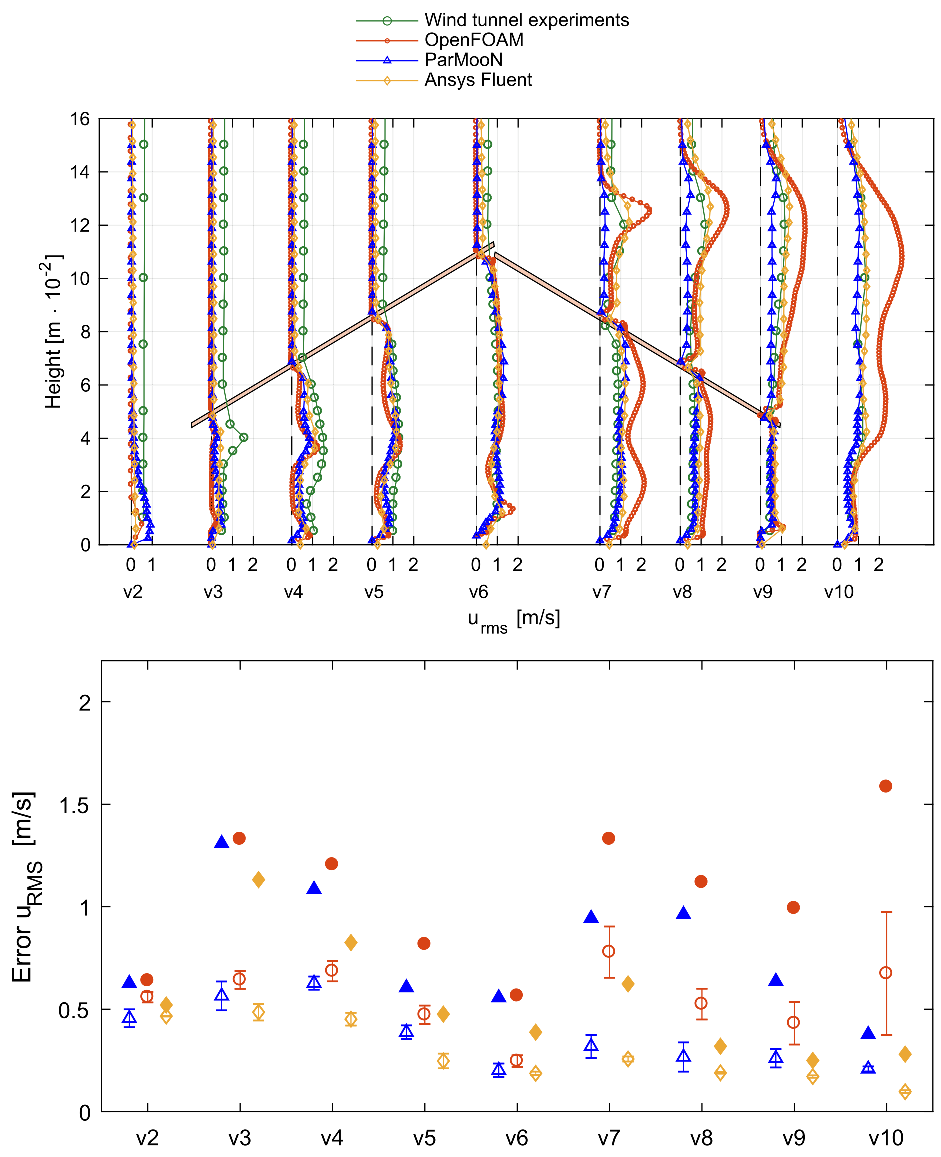


Fig. 14. Comparison between experimental data and numerical results for the root-mean-square of the horizontal velocity (top) with corresponding error statistics (bottom): bold symbol – maximal error, open symbol – average error, interval – standard deviation. Note the effect of the slightly different form of the roof in the experiment and the simulations explained in Remark 1.

$$u_{\mathrm{rms}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} u'u'} \quad \text{and} \quad v_{\mathrm{rms}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} v'v'},$$

with N being the number of time instants for which the velocity was monitored in the simulations. Values superscribed with an apostrophe are the fluctuating parts of the velocities, defined as $u' = u - \bar{u}$ and $v' = v - \bar{v}$.

First of all, the dependency of the numerical results on the grids will be briefly discussed for the open source solvers. Considering sampling line v6, which is in the center of the barn (Fig. 4), one can see that for OpenFOAM, the results on the medium refined and the fine grid are almost identical, see Fig. 11. Also in the case of ParMooN, the results obtained on all grids are quite similar.

In the following, the numerical results at the sample lines are evaluated, including the results obtained with Ansys Fluent. Deviations Δ between the simulated ($\mathrm{res}_{\mathrm{sim}}$) and the experimental results ($\mathrm{res}_{\mathrm{exp}}$) will be presented as the modulus of the relative differences in percents, with the experimental results as reference, i.e., $\Delta = |(\mathrm{res}_{\mathrm{exp}} - \mathrm{res}_{\mathrm{sim}})/\mathrm{res}_{\mathrm{exp}}| \cdot 100$.

Comparing the volume flow through the barn (Table 6), at the barn's inlet (v3) the differences between simulated and experimental results are less than 2% for both OpenFOAM and ParMooN, and about 7% for Ansys Fluent. Considering the volume flow through the barn with the velocity profile at the barn's outlet (v9), the difference between experimental data and simulations are of the order of 3% for both open source solvers and of the order of 9% for Ansys Fluent.

The horizontal component of the velocity (u) is the most important one, since it represents the main flow direction. Fig. 12 presents the comparison of the experimental data and the numerical results as well as the corresponding error statistics. At sample line v2, the results computed with all codes show a good agreement with the experimental data. Only at the first measurement point at the bottom, OpenFOAM and ParMooN underestimated the horizontal velocity somewhat. This is certainly due to the boundary layer at the wall, where the friction is overestimated compared with the real situation in the turbulent flow. But altogether, one can state that the flow profile given at the inlet, compare Fig. 3, was transported accurately through the domain by all codes. At the inlet of the barn, sample line v3, a very good agreement between the results obtained with OpenFOAM and ParMooN and the

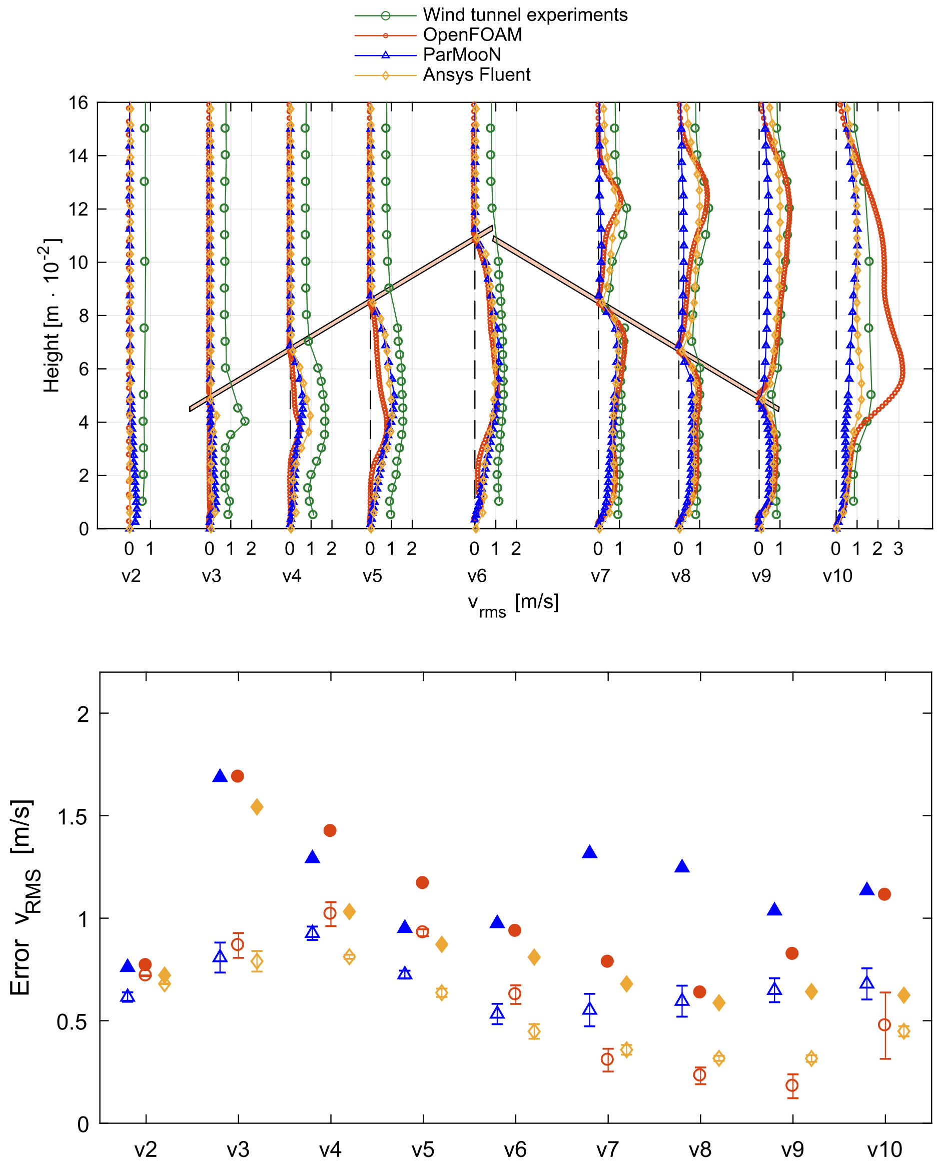


Fig. 15. Comparison between experimental data and numerical results for the root-mean-square of the vertical velocity (top) with corresponding error statistics (bottom): bold symbol – maximal error, open symbol – average error, interval – standard deviation. Note the effect of the slightly different form of the roof in the experiment and the simulations explained in Remark 1.

experimental data is visible. The curve from Ansys Fluent shows some deviations from the experimental data close to the bottom. Inside the barn, sample lines v4–v9, all codes computed qualitatively correct results. There are some deviations from the experimental data and the errors are somewhat larger than at the sample lines v2 and v3. Often, the curves obtained with OpenFOAM and Ansys Fluent are quite close. Based on the mean errors, one can conclude that all codes computed the horizontal velocity with similar accuracy.

The results for the vertical velocity component are presented in Fig. 13. The magnitude of this component is much smaller than for the horizontal component. A generally good agreement between experimental and the numerical results with all used codes was obtained, with slight differences in details and small average errors.

Results for the root-mean-square of the horizontal velocity component are depicted in Fig. 14. One can observe that there are notable differences in some parts between the experimental data and the numerical results. However, it is important to observe that some modeling aspects already introduced differences between the computational and the experimental setup. For example, in the experiments roughness elements were used to induce turbulence of the flow already in front of the barn (Section 2.2), whereas in the numerical model, turbulent disturbances of the flow in front of the barn are not introduced. We think that this difference is the reason why the experimental root-mean-square velocities are larger in the first part of the barn, i.e., in sample lines v2-v5. For the flow field in the second part of the barn, sample lines v6-v9, not longer the flow in front of the barn but the flow field in its first part possesses the dominating impact. In this region, the numerical results obtained with ParMooN and Ansys Fluent are quite close to the experimental data and one can see that the average errors at sample lines v6-v9 are notably smaller than at sample lines v2-v5 for these CFD solvers. It can be observed generally that in regions where the results computed with the different codes are different, the curves obtained with Ansys Fluent are closer to those from ParMooN than to those predicted with OpenFOAM.

Fig. 15 presents the results for the root-mean-square of the vertical velocity component. Concerning the first half of the barn, the same comments apply as for the root-mean-square of the horizontal velocity. Again, the average errors in the second part of the barn are smaller than in the first part, most notably for OpenFOAM and Ansys Fluent. The results of both codes show in particular a comparatively good agreement with the experimental data at the sample lines v7-v9 above the roof of the barn.

5. Conclusions and outlook

This paper assessed the potential of two open source solvers (OpenFOAM and ParMooN) for simulating the turbulent air flow inside and around a naturally ventilated barn. In particular, results of numerical simulations were compared with experimental data measured in a wind tunnel. The main goal consisted in assessing the solvers in terms of their capability of simulating the transient flow with sufficient accuracy. Additionally, simulations were performed with the widely established commercial code Ansys Fluent and the results were compared with the results from the two open source solvers.

In our opinion, there is a clear result of our studies: both OpenFOAM and ParMooN represent competitive choices for the numerical simulation of the considered application. There was a good agreement of the time-averaged velocities at the sample lines. All three codes computed the velocity with similar accuracy. The differences of the root-mean-square velocities at the first sample lines are due to a difference between the experimental setup and the numerical model. At the other sample lines, there is often a good agreement within the barn, in particular for ParMooN with respect to the horizontal root-mean-square velocity and for OpenFOAM with respect to the vertical root-mean-square velocity. We expect that the agreement of the numerical results and the experimental data, especially for the turbulent characteristics, can be enhanced on the one hand by the introduction of an appropriate turbulent inflow and on the other hand by the application of more sophisticated turbulence models. These topics will be subject of future research.

It is well known that the simulation of turbulent flows, in particular with advanced approaches like LES, is computationally demanding and requires appropriate hardware, which is usually not available in many institutes and companies. The use of multiple processors is indispensable, which requires that the used solver has to support parallel computing. In this respect, external cloud computing services might become a more and more attractive feature in future. They have gained already popularity in recent years, which is connected with declining cost for simulation time.

A limiting factor for the wider spread of open source software might be the more complex handling and work-flow compared with commercial codes, e.g., because of the absence of a graphical user interface (GUI) and, correspondingly, the use of the command line. For OpenFOAM, various efforts have been made to provide GUI-based platforms, where the whole work-flow can be handled via one interface. Less known codes could benefit from providing such GUIs and gain more popularity. Besides that, to promote a wider use of open source solvers, a clear and accessible documentation or handbook of the code is indispensable as well as assembled tutorial cases, from which a non-experienced user can learn most.

CRediT authorship contribution statement

David Janke: Conceptualization, Software, Writing - original draft, Writing - review & editing. **Alfonso Caiazzo:** Conceptualization, Software, Visualization, Writing - original draft, Writing - review & editing. **Naveed Ahmed:** Software, Writing - original draft, Writing - review & editing. **Najib Alia:** Software, Writing - original draft, Writing - review & editing. **Oswald Knoth:** Software, Writing - original draft. **Baptiste Moreau:** Software, Writing - original draft. **Ulrich Wilbrandt:** Software, Writing - original draft, Writing - review & editing. **Dilya Willink:** Software, Data curation. **Thomas Amon:** Supervision, Writing - original draft, Writing - review & editing. **Volker John:** Supervision, Conceptualization, Software, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.compag.2020.105546.

References

Ahmed, N., Bartsch, C., John, V., Wilbrandt, U., 2018. An assessment of some solvers for saddle point problems emerging from the incompressible Navier-Stokes equations. Comput. Methods Appl. Mech. Eng. 331, 492–513. https://doi.org/10.1016/j.cma.2017.12.004.

Ahmed, N., Chacón Rebollo, T., John, V., Rubino, S., 2017. A review of variational multiscale methods for the simulation of turbulent incompressible flows. Arch. Comput. Methods Eng. 24 (1), 115–164. https://doi.org/10.1007/s11831-015-9161-0.

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M., Wells, G., 2015. The fenics project version 1.5. Archive of Numerical Software 3 (100).http://journals.ub.uni-heidelberg.de/index.php/ans/article/view/20553.

Amestoy, P.R., Guermouche, A., L'Excellent, J.-Y., Pralet, S., 2006. Hybrid scheduling for the parallel solution of linear systems. Parallel Comput. 32 (2), 136–156.

Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zampini, S., Zhang, H., Zhang, H., 2016. PETSc Web page. URLhttp://www.mcs.anl.gov/petsc.

Bangerth, W., Davydov, D., Heister, T., Heltai, L., Kanschat, G., Kronbichler, M., Maier, M., Turcksin, B., Wells, D., 2016. The deal.II library, version 8.4. J. Numer. Math. 24.

Bjerg, B., Cascone, G., Lee, I.-B., Bartzanas, T., Norton, T., Hong, S.-W., Seo, I.-H., Banhazi, T., Liberati, P., Marucci, A., et al., 2013. Modelling of ammonia emissions from naturally ventilated livestock buildings. part 3: Cfd modelling. Biosyst. Eng. 116 (3) 259–275.

Blatt, M., Burchardt, A., Dedner, A., Engwer, C., Fahlke, J., Flemisch, B., Gersbacher, C., Gräser, C., Gruber, F., Grüninger, C., Kempf, D., Klöfkorn, R., Malkmus, T., Müthing, S., Nolte, M., Piatkowski, M., Sander, O., 2016. The distributed and unified numerics environment, version 2.4. Arch. Numer. Software 4 (100), 13–29.

Dedner, A., Klöfkorn, R., Nolte, M., Ohlberger, M., 2010. A generic interface for parallel and adaptive discretization schemes: abstraction principles and the DUNE-FEM module. Computing 90 (3–4), 165–196. https://doi.org/10.1007/s00607-010-0110-3.

Elman, H.C., Silvester, D.J., Wathen, A.J., 2014. Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. Numerical Mathematics and Scientific Computation, second ed. Oxford University Press, Oxford.

Fiedler, M., Berg, W., Ammon, C., Loebsin, C., Sanftleben, P., Samer, M., von Bobrutzki, K., Kiwan, A., Saha, C.K., 2013. Air velocity measurements using ultrasonic anemometers in the animal zone of a naturally ventilated dairy barn. Biosyst. Eng. 116 (3), 276–285.

Fiedler, M., Fischer, J., Hempel, S., Saha, C., Loebsin, C., Berg, W., Amon, B., Brunsch, R., Amon, T., 2014. Flow fields within a dairy barn-measurements, physical modelling and numerical simulation. In: Proceedings. International Conference of Agricultural Engineering AgEng, pp. 1–5.

Ganesan, S., John, V., Matthies, G., Meesala, R., Abdus, S., Wilbrandt, U., 2016. An object oriented parallel finite element scheme for computing pdes: design and implementation. In: IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW). IEEE, Hyderabad, pp. 106–115.

Geuzaine, C., Remacle, J.-F., 2009. Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Methods Eng. 79 (11), 1309–1331. https://doi.org/10.1002/nme.2579.

Gisen, D., 2014. Generation of a 3d mesh using snappyhexmesh featuring anisotropic refinement and near-wall layers. In: ICHE 2014. Proceedings of the 11th International Conference on Hydroscience & Engineering. pp. 983–990.

Hempel, S., Wiedemann, L., Ammon, C., Fiedler, M., Saha, C., Janke, D., Loebsin, C., Fischer, J., Amon, B., Hoffmann, G., et al., 2015. Determine the through-flow characteristics of naturally ventilated dairy barns to optimise barn climate. In: Bau, Technik und Umwelt in der landwirtschaftlichen Nutztierhaltung 2015. Kuratorium für Technik und Bauwesen in der Landwirtschaft eV, Darmstadt, pp. 346–351.

Holzmann, T., 2016. Mathematics, Numerics, Derivations and OpenFOAM. Holzmann CFD. URLhttps://www.researchgate.net/publication/307546712?channel=doi&linkId=57cc47b508ae59825185d94f&showFulltext=true.

Hong, S.-W., Exadaktylos, V., Lee, I.-B., Amon, T., Youssef, A., Norton, T., Berckmans, D., 2017. Validation of an open source cfd code to simulate natural ventilation for agricultural buildings. Comput. Electron. Agricul. 138, 80–91.

Issa, R.I., 1986. Solution of the implicitly discretised fluid flow equations by operator-splitting. J. Comput. Phys. 62 (1), 40–65.

Jähn, M., Knoth, O., König, M., Vogelsberg, U., 2014. Asam v2. 7: a compressible atmospheric model with a cartesian cut cell approach. Geoscientific Model Development Discussions 7 4.

Janke, D., Yi, Q., Hempel, S., Thormann, L., Amon, B., Amon, T., 2020. Velocity measurements of a 1:100 scaled model of a naturally ventilated dairy barn in an atmospheric boundary layer wind tunnel. PUBLISSO Repository for Life Sciences. URLhttps://doi.org/10.4126/FRL01-006420859.

John, V., 2016. Finite element methods for incompressible flow problems. vol. 51 of Springer Series in Computational Mathematics. Springer, Cham.https://doi.org/10.1007/978-3-319-45750-5.

John, V., Matthies, G., 2004. MooNMD—a program package based on mapped finite element methods. Comput. Vis. Sci. 6 (2–3), 163–169.

Kateris, D., Fragos, V., Kotsopoulos, T., Martzopoulou, A., Moshou, D., 2012. Calculated external pressure coefficients on livestock buildings and comparison with eurocode 1. Wind Struct. 15 (6), 481–494.

König, M., Hempel, S., Janke, D., Amon, B., Amon, T., 2018. Variabilities in determining air exchange rates in naturally ventilated dairy buildings using the co2 production model. Biosyst. Eng. 174, 249–259.

Lee, I.-B., Bitog, J.P.P., Hong, S.-W., Seo, I.-H., Kwon, K.-S., Bartzanas, T., Kacira, M., 2013. The past, present and future of cfd for agro-environmental applications. Comput. Electron. Agricul. 93, 168–183.

Lee, I.-B., Sase, S., Sung, S.-H., 2007. Evaluation of cfd accuracy for the ventilation study of a naturally ventilated broiler house. Japan Agricult. Res. Q.: JARQ 41 (1), 53–64.

OpenFOAM, 2016. OpenFOAM Web page. URLhttp://openfoam.org/.

Patankar, S.V., Spalding, D.B., 1983. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In: Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion. Elsevier, pp. 54–73.

Pope, S.B., 2000. Turbulent flows. Cambridge University Press, Cambridge.

Saad, Y., 1993. A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Comput. 14 (2), 461–469.

Saha, C.K., Wu, W., Zhang, G., Bjerg, B., 2011. Assessing effect of wind tunnel sizes on air velocity and concentration boundary layers and on ammonia emission estimation using computational fluid dynamics (cfd). Comput. Electronics Agricul. 78 (1), 49–60 URL http://www.sciencedirect.com/science/article/pii/S0168169911001268.

Shen, X., Zhang, G., Wu, W., Bjerg, B., 2013. Model-based control of natural ventilation in dairy buildings. Comput. Electronics Agricul. 94, 47–57. URLhttp://www.sciencedirect.com/science/article/pii/S016816991300046X.

Si, H., 2015. Tetgen, a delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. 41 (2) 11:1–11: 36. URL http://doi.acm.org/10.1145/2629697.

Smagorinsky, J., 1963. General circulation experiments with the primitive equations. Mon. Wea. Rev. 91, 99–164.

The OpenFOAM Foundation, 2016. Openfoam user guide version 4.0.http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf.

VDI, 2000. Physical modelling of flow and dispersion processes in the atmospheric boundary layer—application of wind tunnels. VDI Verein Deutcher Ingenieure, Düsseldorf.

Villagran, E.A., Romero, E.J.B., Bojaca, C.R., 2019. Transient cfd analysis of the natural ventilation of three types of greenhouses used for agricultural production in a tropical mountain climate. Biosyst. Eng. 188, 288–304.

Wilbrandt, U., Bartsch, C., Ahmed, N., Alia, N., Anker, F., Blank, L., Caiazzo, A., Ganesan, S., Giere, S., Matthies, G., Meesala, R., Shamim, A., Venkatesan, J., John, V., 2017. ParMooN—a modernized program package based on mapped finite elements. Comput. Math. Appl. 74 (1), 74–88. https://doi.org/10.1016/j.camwa.2016.12.020.

Yi, Q., König, M., Janke, D., Hempel, S., Zhang, G., Amon, B., Amon, T., 2018. Wind tunnel investigations of sidewall opening effects on indoor airflows of a cross-ventilated dairy building. Energy Build. 175, 163–172.

Yoshizawa, A., 1986. Statistical theory for compressible turbulent shear flows, with the application to subgrid modeling. Phys. Fluids 29, 2152–2164.