energies

MDPI

Review

A Survey on the Application of Machine Learning in Turbulent Flow Simulations

Maciej Majchrzak [1,*], Katarzyna Marciniak-Lukasiak [2] and Piotr Lukasiak [1,*]

[1] Institute of Computing Sciences, Faculty of Computing and Telecommunications, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland
[2] Institute of Food Sciences, Faculty of Food Assessment and Technology, Warsaw University of Life Sciences (WULS-SGGW), Nowoursynowska 159c, 02-776 Warsaw, Poland
* Correspondence: maciej.majchrzak@doctorate.put.poznan.pl (M.M.); piotr.lukasiak@put.poznan.pl (P.L.)

Abstract: As early as at the end of the 19th century, shortly after mathematical rules describing fluid flow—such as the Navier–Stokes equations—were developed, the idea of using them for flow simulations emerged. However, it was soon discovered that the computational requirements of problems such as atmospheric phenomena and engineering calculations made hand computation impractical. The dawn of the computer age also marked the beginning of computational fluid mechanics and their subsequent popularization made computational fluid dynamics one of the common tools used in science and engineering. From the beginning, however, the method has faced a trade-off between accuracy and computational requirements. The purpose of this work is to examine how the results of recent advances in machine learning can be applied to further develop the seemingly plateaued method. Examples of applying this method to improve various types of computational flow simulations, both by increasing the accuracy of the results obtained and reducing calculation times, have been reviewed in the paper as well as the effectiveness of the methods presented, the chances of their acceptance by industry, including possible obstacles, and potential directions for their development. One can observe an evolution of solutions from simple determination of closure coefficients through to more advanced attempts to use machine learning as an alternative to the classical methods of solving differential equations on which computational fluid dynamics is based up to turbulence models built solely from neural networks. A continuation of these three trends may lead to at least a partial replacement of Navier–Stokes-based computational fluid dynamics by machine-learning-based solutions.

Keywords: machine learning; computational fluid dynamics; turbulence; turbulence modeling

Citation: Majchrzak, M.; Marciniak-Lukasiak, K.; Lukasiak, P. A Survey on the Application of Machine Learning in Turbulent Flow Simulations. *Energies* **2023**, *16*, 1755. https://doi.org/10.3390/en16041755

Academic Editors: Marcin Sosnowski, Jaroslaw Krzywanski, Karolina Grabowska, Dorian Skrobek, Ghulam Moeen Uddin, Yunfei Gao, Anna Zylka, Anna Kulakowska and Bachil El Fil

Received: 28 December 2022
Revised: 31 January 2023
Accepted: 7 February 2023
Published: 9 February 2023

1. Introduction

Fluid mechanics is used in many branches of engineering as well as meteorology, oceanography, biology, and astrophysics to name just a few. Some level of understanding of fluid flow must have already been possessed by prehistoric people, manifested in the construction of spears, arrows, or boats. Ancient civilizations applied the rules of fluid motion successfully to create systems for irrigation, drainage, or for transporting and storing water, many of which, such as Roman aqueducts, fountains, and sewers, are still preserved today. The first known attempts to scientifically describe the behavior of liquids took place in Greece where Archimedes developed the law of buoyancy in 250 BCE. Many modern scientists have studied various aspects of fluid mechanics such as the relationship between the height of the fluid column and pressure (Torricelli and Pascal), viscosity and friction (Guglielmini, Newton, and Pitot), or the relationship between flow velocity and pressure (Bernoulli, d'Alembert, and Euler). The latter in particular focused on describing the motion of liquids using mathematical formulas. Established by Helmholtz in 1858, the laws of vertex motion are still used today. From 1822 to 1850, Claude-Louis Navier and George Gabriel Stokes developed partial differential equations describing the motion

of viscous fluids. Named after their creators, Navier–Stokes equations are used to this day in fluid mechanics in an unchanged form. Osborne Reynolds at the end of the 19th century published several papers on fluid mechanics, particularly turbulent flow [1]. The Reynolds number and Reynolds averaging developed by him have been widely used in fluid mechanics ever since.

Despite hundreds of years of development of fluid mechanics, the attempt to determine by analytical methods the flow parameters for any system more complex than flow in a simple channel or smooth body flow is impractical due to the complexity of the problem. As early as the beginning of the 20th century, the idea of numerical flow simulation appeared—the finite difference method developed by Sheppard in 1899 [2] was later developed by Richardson [3] and used in 1910 to calculate the load acting on a dam. The latter attempted to simulate the state of the weather on a specific day but, after six weeks of manual calculations, was only able to determine the values for the two vertical columns. Guided by this experience, he calculated that to simulate the state of the entire Earth's atmosphere once every three hours, assuming a resolution of 200 km, would require 64,000 people running calculations continuously. In 1928 Courant, Fredrichson, and Lewy defined stability criteria for time-stepping in flow simulation [4]. In 1933, Thom published results of simulating flow around a cylinder; his approach was based on iterative, manual calculation of the value of the velocity vector in the domain until a stable solution was obtained [5]. Both the Courant number and the iterative approach are still used today in computational fluid dynamics (CFD).

The large amount of time required for calculations has made the term "numerical simulation" synonymous with "computer simulation" today. Immediately after the end of World War II, there were ideas to use fledgling computers to simulate flow—in 1950 Von Neumann, Charney, and Fjörtoft simulated weather changes in the U.S. 24 h in advance [6]. Von Neumann, along with Fermi and Ulam, used a computer to simulate the operation of the first hydrogen bomb. These first attempts proved the practicality of using computers to simulate physical phenomena. Further development of CFD, due to the small number of available computers, was limited to military centers and major universities, and simulations were concerned with military, aerospace, and meteorological applications. The development of computers has increased their availability and performance while decreasing their price [7] which has contributed to increased interest in computer simulations. The first textbook focusing on CFD simulations was published in 1972 by P. J. Roache [8]. Since the 1980s and the advent of widely available commercial CFD software, the field of application of this simulation method has expanded continuously. Initially, due to the high cost of computing power and the multitude of limitations of the method, it was mainly used in the aerospace industry. However, interest in the method grew steadily which contributed to its development concerning many aspects of CFD. The structural grids initially used gave way to unstructured and tetrahedral grids [9,10]. The ability to simulate complex systems was improved by the advent of chimera [11] and overset [12] meshes, which also made it possible to simulate parts that move relative to each other. A significant step in the development of CFD was the emergence of the ability to run calculations on multiple processors [13]. After the advent of the message passing interface (MPI) [14], this approach became widespread in flow simulations, allowing for shorter computation times and simulations of systems that were previously too computationally demanding.

Today, thanks to the wide availability of fast computers and advances in the field, CFD calculations are also routinely used in the automotive industry, shipbuilding, HVAC design, urban planning, medicine, and many other fields. A number of professional commercial forms of software, together with wizard-type solutions found in some CAD software packages as well as free open-source programs [15], allow the use of CFD simulations even by small companies, universities, etc.

Turbulent flow is characterized by chaotic changes in the velocity and pressure of the flowing medium, as opposed to laminar flow in which the fluid moves in parallel, unmixed layers. The cause of turbulence is the presence of areas in the moving fluid where kinetic energy overcomes the damping effect of viscosity. The susceptibility of a flow to turbulence

is determined by the Reynolds number, a dimensionless value that is the ratio of the kinetic energy of the flow to the damping caused by viscosity.

Figure 1 shows an example of turbulent flow encountered in everyday life. Pouring milk into coffee results in the appearance of areas characterized by a significantly higher velocity of fluid movement—the hot coffee in the mug also moves due to convection caused by temperature differences, but the velocity of this movement is much lower. In these areas, there is a combination of flow velocity, characteristic dimension—in this case the cup's inner diameter—and fluid viscosity that causes turbulence to appear. These three quantities determine the Reynolds number [1], one of the so-called characteristic numbers, that indicates the type of flow. In areas where the velocity remains lower, the flow still remains laminar and the distribution of areas of turbulence and their absence changes dynamically. The viscosity of the fluid causes a gradual conversion of the kinetic energy of the fluid's motion into thermal energy, causing a slight increase in temperature. As the velocity decreases, the turbulent flow disappears and, after a while, the movement of the coffee in the cup is again caused solely by convection. Accurately determining the moment of transition from laminar to turbulent flow is one of the most important issues in CFD simulations.

Figure 1. Coffee mixing with milk as an example of turbulent fluid motion.

The turbulence occurring in turbulent flow varies in size—the largest turbulence is usually about the size of the body around which the flow occurs or the size of the channel in which the flow takes place, the size of the smallest is determined by the so-called Kolmogorov microscale [16]. Large vortices acquire kinetic energy from mean flow. Smaller vortices acquire energy from larger ones and transfer it to smaller ones until they reach the smallest vortices at the level where energy is converted directly into heat. The concept of an "energy cascade" involving the transfer of energy from the largest to the smallest vortices is one of the foundations of the current understanding of turbulence [17].

Since turbulent flows are highly irregular in both space and time, no numerically stable solution of the Navier–Stokes equation has yet been developed and a statistical description is used to quantitatively describe the values associated with turbulent flow.

Most flows found in nature and artificial systems are turbulent [18]. In engineering applications, turbulence is treated, depending on the situation, as an undesirable or desirable phenomenon. Figure 2. shows an example of flow around a vehicle—the vast majority of turbulence in such flows is considered unfavorable. The visualization shows the periodic detachment of vortices from the edge of the side mirror. The direction of rotation of the detaching vortices is alternating; vortices with different directions differ in color.

Figure 2. Visualization of LES calculation results showing turbulence as an unwanted phenomenon.

The periodicity of vortex detachment is the main cause of the problems—it causes alternating increases and decreases in the pressure acting on the mirror and vehicle body along the path of vortex movement. Pressure changes cause vibrations and, if the frequency is high enough, audible noise. If the vortices are large enough, they can affect the movement of the vehicle, impairing its handling; smaller scale turbulence, in turn, can contribute to fatigue damage to vehicle components. Since kinetic energy is turned into heat in turbulent flow, all turbulence significantly increases the drag and fuel consumption of a vehicle—this is the main reason for engineers' efforts to achieve the most laminar flow around the cars.

An example of turbulence being a beneficial phenomenon is shown in Figure 3. In laminar flow, adjacent fluid layers do not mix with each other and, consequently, heat transfer occurs mainly through conduction. In turbulent flow, however, layers are absent and fluid particles also move across the flow direction, carrying thermal energy along with kinetic energy. For this reason, heat exchangers are designed to ensure turbulent fluid motion, significantly increasing their efficiency by achieving a higher heat transfer coefficient. In the visualization, it can be seen that at the air inlet to the exchanger (right side) both the turbulence intensity (cross-section) and heat flux (visualized on tube surfaces) are low. The increase in turbulence intensity as air enters the tube bundle also increases the heat flux.

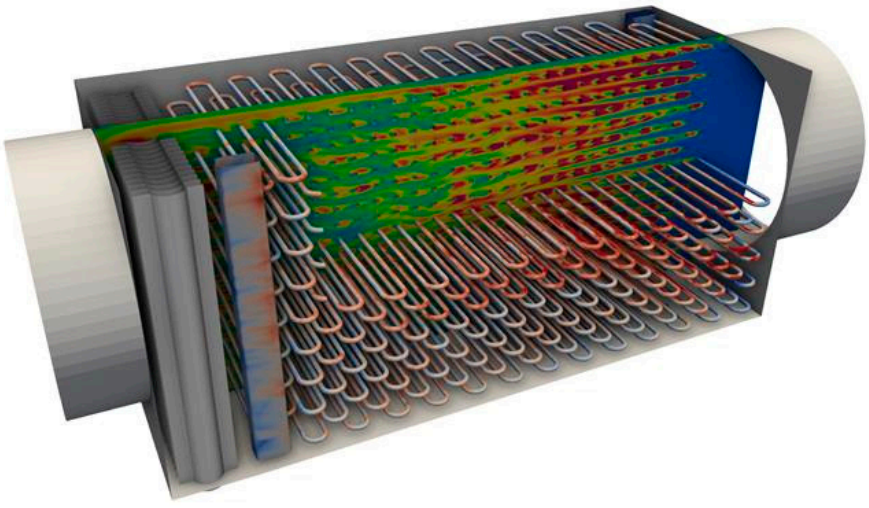

Figure 3. Visualization of turbulence intensity (cross-section) and heat flux (tubes) in a tube bundle heat exchanger (from blue—lowest value, to red—highest value respectively).

Factors affecting the occurrence and intensity of turbulence:

- The scale of the problem—body size and channel size;
- The shape of the body and the cross-section of the channel;
- Flow velocity;
- Viscosity of the fluid;
- Compressibility of the fluid;
- Pressure;
- Fluid-structure interactions;
- Surface roughness.

2. Practical Approach to the Problem of Turbulence in CFD

Two variables are needed to describe the movement of a fluid—velocity and pressure. The relationship between them is described by the Navier–Stokes equations mentioned earlier:

- The momentum equation—application of Newton's law of motion to a fluid flow;
- The continuity equation—also known as the mass conservation equation;
- The energy equation.

They are the heart of the CFD method but at the same time the source of its limitations—it is assumed that as nonlinear second-order partial differential equations with four independent unknowns, they do not have a general analytical solution for 3D flow. Proving the existence of a unique solution of the Navier–Stokes equation for 3D flow is one of the so-called millennium problems [19,20]. The existence of a solution for 2D flow was proven in 1933 by Leray [21]. Most CFD calculation methods are therefore based on finding a numerical solution. Using certain simplifications, such as the assumption of incompressibility of the fluid, the complexity of the equations can be reduced. The flow domain is then divided into elements forming a computational mesh (discretization). The use of the finite difference method makes it possible to obtain the discrete form of the Navier–Stokes equations which can be solved using an iterative solver. Calculations are carried out taking into account the so-called boundary conditions and time discretization, except for steady-state calculations in which the time dimension is omitted.

The problem of simulating turbulent flow arises directly from the size of the smallest turbulence—to accurately simulate the behavior of the fluid at these smallest turbulence scales, the grid elements must be smaller than they are. Meshes made with this assumption often have billions of elements making their use in calculations prohibitively expensive due to time and hardware requirements. The increase in the computational capabilities of computers has made it possible to conduct CFD simulations for much larger (geometrically) problems. A significant increase in the performance of CFD programs is due to the ability to run calculations on multiple processors, both within a single computer and on dedicated computing clusters. Unfortunately, it is impossible to simulate arbitrarily large domains using simply more CPUs—the need for continuous communication between processors means that adding each additional core gives a progressively lower reduction in calculation time. For a given problem, there is an optimal number of processors above which adding more processors slows down calculations.

The abovementioned limitations led to the fact that already in the 1980s the search began for the possibility of improving the accuracy and reducing the time of CFD calculations by using solutions based on machine learning. An additional motivation for researching the feasibility of using ML in turbulence modeling is the fact that it is used in a wide range of engineering and scientific problems such as reliability assessment [22], nanotechnology [23], virology [24], immunology [25], bioinformatics [26], or CFD results post-processing [27], for example.

3. Turbulence Modeling Methods

In practice, the vast majority of flow simulations are performed using three turbulence modeling methods. The first, the most accurate and least frequently used, is direct

numerical simulation (DNS), which involves using a grid with a cell size consistent with the scale of the turbulence. The second, the most popular but also the least accurate, is Reynold-averaged Navier–Stokes (RANS). It uses the grids with the lowest resolution and so-called turbulence models that determine the effect of vortices too small to resolve on the flow. The third method, less popular than RANS but still widely used in industry, is large eddy simulation (LES). The meshes used in this method are more accurate than those used in RANS thus allowing to fully resolve the largest of the turbulence and giving more accurate results.

Figure 4 shows the differences in the appearance of the results obtained in simulations carried out using the three methods. The difference between DNS and LES is evident in the lack of small-scale turbulence in the case of LES—the large turbulences at the top of the image look similar. The influence of these smallest turbulences in LES is taken into account using the turbulence model and is visible as a small velocity gradient at the edge of the jet, not visible in DNS. The results of RANS simulations are completely different—there are no vortices but only a smooth gradient of the velocity field. This is due to the use of turbulence models based on Reynolds averaging [1]—the apparent distribution of velocity is the sum of its constant component and the average of fluctuations calculated by the model. Averaging the DNL and LES results over time would yield a similar velocity gradient to the RANS results.

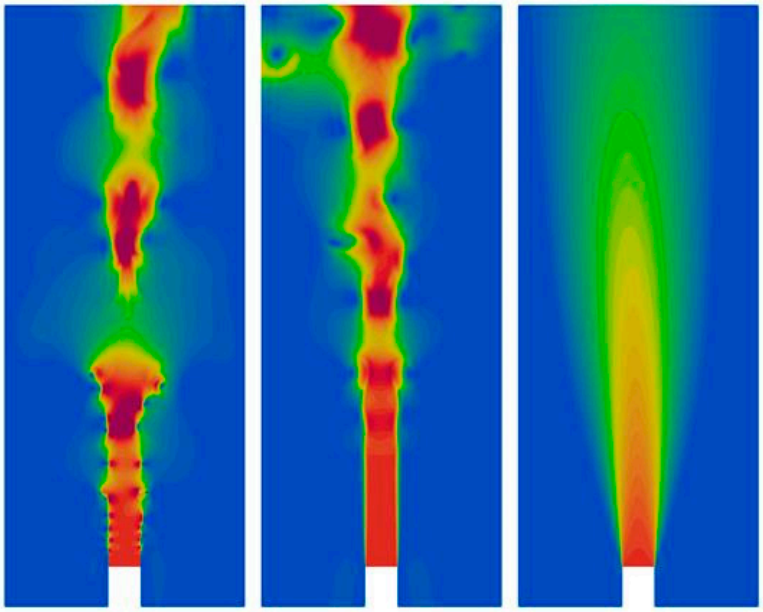

Figure 4. The results of the jet flow simulation using three methods: DNS, LES, and RANS-visualisation of flow velocity (from blue—lowest value, to red—highest value respectively).

3.1. Direct Numerical Simulation (DNS)

This involves simulating the full range of turbulence scales, from the largest turbulence drawing energy directly from the mean flow to the smallest dissipative scales. The maximum size of grid elements is determined by the so-called Kolmogorov scale η—the ratio of grid cell size to integral scale L—usually the diameter of the channel or the size of the body around which the fluid flows. The relationship between the Kolmogorov scale η, the turbulence kinetic energy dissipation rate ε, and the kinematic viscosity of fluid ν is described by the equation:

$$\eta = (\nu^3/\varepsilon)^{\frac{1}{4}} \quad (1)$$

The required number of grid points depends on the Reynolds number of the flow—it is assumed that the size of the grid cells must be small enough so that the Reynolds number for each cell is at most 1—and thus the size of the grid increases rapidly with the size of the simulation domain and the speed of the flow. In addition, the small size of the

grid cells forces the use of very small time-steps so as to satisfy the Courant–Friedrichs–Lewy condition [4]. These two relationships mean that the required number of floating-point operations needed to simulate a given flow increases with the cube of the Reynolds number. In practice, these calculations give high accuracy but at the expense of the highest requirements of all CFD methods in terms of computing power and computation time, which is why they are used mainly for research purposes, to study flows whose behavior is difficult to measure experimentally and in the development of turbulence models where they serve as a reference. Engineering applications are limited to the most demanding applications and are conducted on small fragments of systems, such as a wing profile.

Figure 5 shows the effect of grid resolution on the values obtained in DNS calculations. The upper left shows velocity values in a cross-section of the homogeneous isotropic turbulence field at the beginning of the simulation. The upper right corner shows the values after 1 s of simulation on a 100 × 100 × 100 grid while the lower left corner shows the values after the same time but computed on a 50 × 50 × 50 grid. The initial values generated on the higher-resolution grid were resampled onto the lower-resolution grid. After the calculations were completed, the results from the lower-resolution grid were resampled back to the high-resolution grid and the differences between the obtained values are shown in the lower right corner. Something that is worth noting is the high irregularity of the obtained differences.

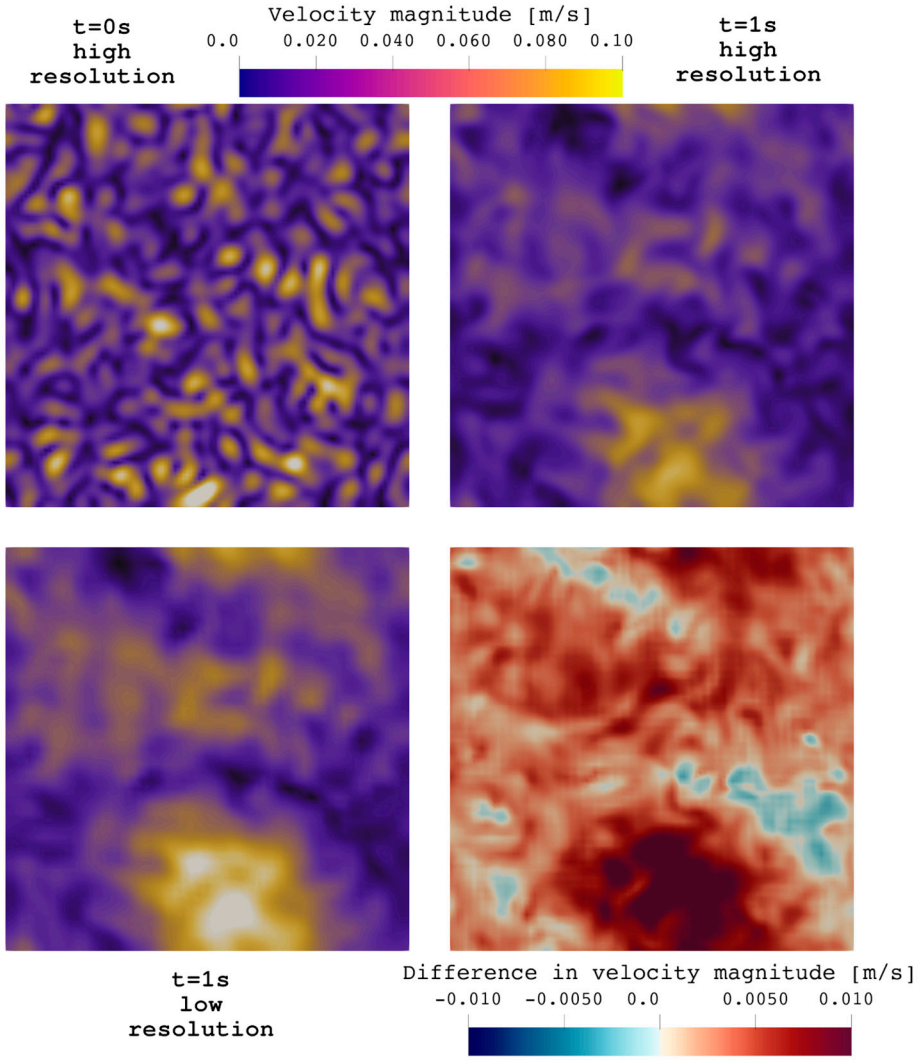

Figure 5. Comparison of the velocity fields from DNS calculations of the 3D homogenous isotropic turbulence for high resolution gridwith results obtained for a low resolution grid.

The advantage of DNS is the high accuracy of calculations that allow it to treat this method as an alternative to physical experiments, while the disadvantages of high hardware requirements and a high calculation time make them almost impossible for practical application in engineering problems.

Among the examples of the application of machine learning to improve DNS calculations, we can mention, for example, the method proposed by Pathak et al. [28] to accelerate CFD calculations by using deep learning in combination with classical partial differential equations (PDEs) [29]. Their solution is based on resolving with full accuracy only some of the iterations—the remaining iterations are calculated on a grid of lesser accuracy (the results are projected onto a less accurate grid which involves a loss of accuracy). The problem on which the solution was tested was two-dimensional convection-induced flow. The deep learning model learns by comparing the appearance of the results for full and reduced accuracy and, for intermediate iterations, corrects the inaccuracies by modifying the results obtained on the less accurate grid and then projected onto the denser grid (Figure 5). ML-PDE obtained a smaller error than the classical simulation on the same grid when compared to the baseline simulation [28].

A more complex approach proposed by Kochkov et al. [30] involves running simulations using the DNS method with a grid size and time-step size larger than required for this method in order to reduce computation time consumption. This paper proposes two solutions, learned correction [31] and learned interpolation [29], with the latter giving better results. Learned interpolation is based on the use of convolutional neural networks to interpolate the results of calculations carried out on a coarser grid so as to obtain a distribution of physical values consistent with the results of calculations on a grid compliant with the requirements of the DNS method. The network is trained using the results of DNS calculations at full scale using supervised learning—the difference between the predicted values and the ground truth is used as a loss function. The results show that it is possible to use a grid size 10 times larger and the same amount of time-steps and obtain results consistent with full-scale DNS simulation. The ML-based solver is 12 times slower than the traditional one but since the computational resolution is 103 times smaller (102 times fewer grid points in 2D analysis multiplied by 10 times larger time-steps consistent with Courant's criterion), this gives a reduction in computation time of about $1000/12 \approx 80$ times [30].

3.2. Reynolds Averaged Navier–Stokes (RANS)

Simulations of this type are based on solving the time-averaged form of the Navier–Stokes equation. This approach is based on the so-called Reynolds decomposition [32] assuming that at each instant of time, a given value describing the flow can be divided into an average value and a fluctuation value (Reynolds stress). Due to the advantage of the number of unknowns over the number of equations (closure problem), it is impossible to accurately calculate all the factors of the equation describing the flow. Boussinesq proposed a way to solve this problem in the form of the eddy viscosity hypothesis [33] assuming a linear relationship between Reynolds stress and the mean strain rate. This hypothesis is used despite its apparent physical incorrectness [34]. Closure coefficients—constants used in the equations whose values cannot be determined due to the closure problem—have been established experimentally as the average of the values determined by comparing test calculations with measurements of actual flows. An example of such a coefficient is $C\mu$ used in the k-epsilon model to determine the turbulent dissipation rate ϵ depending on the turbulence kinetic energy k and turbulence scale l. The default value of $C\mu$ is 0.09.

These limitations mean that calculations using RANS cannot be universal—for each flow, the values of the coefficients giving the highest accuracy are different [35]. Moreover, in complex analyses, the nature of the flow in different areas of the computational domain may be different making even the best-suited coefficients only a compromise that does not give high accuracy results.

One of the key features of simulations using RANS is the use of so-called turbulence models for calculating Reynolds stress, the most important of which are:

- k-epsilon (k-ε)—[36] the most commonly used in RANS-type CFD calculations. It is based on two equations calculating k—turbulence kinetic energy and ε—turbulence kinetic energy dispersion rate. This model deals well with flow away from walls hence it is preferred for simulation of external flows, such as vehicle aerodynamics
- k-omega (k-ω)—[37] together with k-ε constituting the standard used in industry for the vast majority of calculations. It is based on two equations calculating k and ω—the specific dispersion rate of turbulence kinetic energy. This model is best suited for simulating flows near walls; hence, it is used for calculations of systems consisting of pipes, ducts, and narrow spaces
- k-omega SST—[38] a special variant of k-ω which switches between k-ω and k-ε depending on the distance of a given grid point from the nearest wall—"best of both worlds"
- Spalart–Allmaras (S-A)—[39] a model used mainly in the aerospace industry, not suitable for simulating typical "industrial" flows. It is based on a single equation calculating the turbulent viscosity of vortices.

Several other turbulence models exist, but they are less frequently used and no ML-based improvements have been found for them.

Despite not being limited by the Kolmogorov scale requirements, the grids used in RANS calculations cannot have an arbitrarily chosen cell size. In practice, the size of the grid cells is selected according to the expected size of the turbulence and also the size of the details of the geometric model used in the calculations. The relevance of a given area also affects the size of cells; for example, the immediate vicinity of a car body will be meshed very densely, but the area of the computational domain further away will consist of cells that are many times larger. This approach makes it possible to increase the accuracy of simulation in relevant areas without excessively increasing the number of elements and computation time. Such grid sizing requires a lot of experience on the part of the person creating the mesh.

To help determine the correct grid size, Roache proposed the grid convergence index (GCI) in 1994 [40]. CGI is an analytical tool that facilitates the determination of the required level of grid density by comparing the differences in error obtained in calculations performed using grids of different cell sizes. The idea is based on the fact that the differences between the results of the calculations when continuously increasing the resolution of the grid are becoming smaller and smaller and converging towards a certain value, hence the method's name. This phenomenon can be used to determine whether further improvements in grid accuracy will translate into significant improvements in the quality of results. This method is usually not applied to every simulation—once a grid resolution that gives good enough quality results is established, it can be used for calculations of similar geometries.

GCI is now commonly used in CFD and mesh-based FEM simulations, applied both manually and using specialized algorithms. The manual approach is to perform simulations for several different grid resolutions and to calculate the GCI value using available tools. An example of an algorithmic approach was presented by Sosnowski et al. [41] in 2019 in the form of a method based on fuzzy logic.

Additional grid size requirements for RANS simulations are due to the need for accurate near-wall zone modeling. The so-called y+ parameter must be kept within ~1, i.e., the thicknesses of the wall layers must correspond to the scale of the smallest turbulence [42], unless, in the case of the k-ε and k-ω models, the so-called wall functions are used that allow the use of a thicker (y+ ranging from 30 to 150) first wall layer.

NASA's CFD Vision 2030 report [43], published in 2014, notes that most of the aerospace CFD research community shares the belief that RANS models have reached a plateau. It also points out that CFD calculations are either not fully trustworthy or too expensive making them a poor substitute for physical experiments.

The undoubted advantages are the much lower hardware requirements and time-consuming calculations. In addition, meshes in RANS calculations can be very coarse allowing fast, low-precision calculations often quite sufficient for engineering applications.

Disadvantages can be considered the inherent inaccuracy of the results, the need to choose the right turbulence model, which requires experience in applying the method, and the representation of turbulent, variable flow as averaged values, which makes it impossible to visualize the actual turbulence (Figure 4).

RANS is the most popular variation of CFD and has thus attracted the attention of many researchers looking to improve flow simulation methods. Most solutions using ML to support RANS calculations focus on reducing inaccuracies due to closure coefficients and Reynolds stress anisotropy with only a few aimed solely at making RANS simulations faster.

The first such solution was published in 2012 by Yarlanki et al. [44] proposing the use of artificial neural networks (ANNs) to establish turbulence model coefficients for data center flow simulations. The network was trained to minimize the error, defined as the difference between the results of CFD calculations and measurements taken in the test room. The coefficients thus determined reduced the average temperature error by 35% compared to calculations with standard coefficients [44]. The values of the coefficients can be determined without ML by trial and error, but this requires time and experience. This approach is one of many solutions mentioned in this work that is based on ANNs—computing systems whose principles are loosely based on the way their biological counterparts work in animal brains. Biological neural networks consist of a large number of highly interconnected cells called neurons. The output signal sent by each neuron is the result of the weighted sum of signals received from its neighbors. The weights are determined in the learning process and define the behavior of the entire network. Similarly, artificial neural networks use artificial neurons organized in multiple layers. The first layer is supplied with an input, the last layer gives the output, and between them is a series of so-called hidden layers obtaining their input from the previous layer and sending their output to the next layer.

Tracey et al. [45] presented a method for reducing the error resulting from the closure problem by using extended kernel regression [46]. Their solution was to compare the results of a DNS simulation performed on a high-accuracy grid of a constricting channel with the results of a RANS simulation on a low-accuracy grid and train a model that corrects the errors of the RANS method (Reynolds stress anisotropy). This approach improved the accuracy of both typical CFD analysis and CFD analysis combined with combustion simulation [45] with limited applicability for flows on which the model was not trained or trained on larger datasets.

The same researchers [47] proposed using neural networks as a tool in improving existing models and creating new turbulence models. In their publication, they described an attempt to replace the S-A turbulence model with a neural network trained on the results of airfoil simulations conducted using said model. A good match [47] was obtained between the results of a standard RANS simulation and the results of calculations performed using this network, even for unseen cases.

In 2016, Ling et al. [48] conducted a comparison of the effectiveness of various machine learning methods—their proposed tensor based neural network (TBNN) and multi-level perceptron (MLP) [49]—as well as the standard linear eddy viscosity model (LEVM) and the more advanced QEVM [50] for evaluating Reynolds stress anisotropy and improving the accuracy of RANS calculations. The dataset consisted of nine cases for which both RANS and DNS simulation results (or fully resolved LES) were available—duct and channel flows with different values of Reynolds number, perpendicular and inclined jets in crossflow, flow around a square cylinder, and flow through a constricting channel. The models were trained on this dataset and the results were compared; the most commonly used LEVM and MLP gave the worst predictions of Reynolds stress anisotropy; TBNN gave by far the best results for every test case [48]. A key element of TBNN is the preservation of Galilean invariance through the use of a tensor basis.

Singh et al. [51] proposed using machine learning to improve the accuracy of simulations carried out using the S-A model by investigating the differences between the results of airfoil simulations and experimental results. The discrepancies found were used to correct the equations of the turbulence model using artificial neural networks (ANNs), and the model

obtained was used for calculations. The resulting improvement in the accuracy of the results was also preserved for airfoils and flow characteristics not taken into account when learning the model [51]. One of the conclusions reached was the confirmation of the validity of comparing simulation results with experiments to investigate the nature of discrepancies.

Duraisamy et al. [52] proposed an alternative to models created by comparing the results of RANS and DNS or LES simulations. Because DNS simulations are prohibitively time-consuming and expensive, their results are only available for small problems and low Reynolds number values. The lack of a baseline model for more complex flows motivated the authors to develop field inversion and machine learning (FIML) [47,53,54]. The operation of this algorithm is based on using inverse modeling to extract the spatial distribution of model discrepancies and then using machine learning to transform the discrepancy information into corrective model forms. The use of this method significantly improved the accuracy of the results of simulating turbulent flow around the wing profile compared to classical RANS S-A calculations [52], even for wing profiles on which the model was not trained.

Wang et al. [55] proposed the creation of a physics-informed machine learning (PIML) framework [56] aimed at modeling turbulence. As an example of such a solution, they presented the use of a random forest algorithm to determine the discrepancy in Reynolds stresses between the results of RANS and DNS simulations. The algorithm was trained on the results of DNS and LES calculations of two systems—a square duct with a fully developed turbulent flow and a flow with massive separations. The model was then used to correct the results of the RANS simulation performed with different flow velocities and on different geometries obtaining excellent convergence with the results of the DNS simulation [55].

An alternative solution to the problem of determining Reynolds stress anisotropy using Bayesian deep neural networks [57,58] was proposed by Geneva and Zabaras in 2019 [59]. Their model was trained on five flows: a constricting channel, square cylinder, periodic hills, square duct, and tandem cylinders, using the Stein variational gradient descent algorithm [60]. Tests of the method were conducted on a backward-facing step and wall-mounted cube. In choosing this approach, they were guided by the desire to simultaneously determine the likely ranges of pressure and velocity values, as well as the confidence and uncertainty of prediction. The results were closer to those of the LES simulation than those of the RANS. However, the authors noted the difficulty in obtaining satisfactory predictions for unseen geometries [59].

In 2019 Maulik et al. [61] presented a method for determining eddy viscosity using machine learning instead of a standard turbulence model. Their work involved 2D turbulent airflow over a backward-facing step. The model was trained on the results of calculations carried out in OpenFOAM using the S-A turbulence model; the variables were the dimensions of the step and the velocity of the flow and the location of a given point relative to the step, with the output in turn being the eddy viscosity value. Such a trained model (a neural network with 6 hidden layers and only 40 neurons) was integrated into OpenFOAM, replacing the calculation of eddy viscosity by a typical turbulence model; the rest of the simulation (determination of pressure and velocity distribution) was carried out in a typical manner. Importantly, the model correctly determined the eddy viscosity value also for boundary conditions outside the range found in the training set. The resulting reduction in the calculation time is significant; ML-CFD achieved the result in ~14% of the time required by S-A [61]. In addition, this method allowed for the use of significantly higher relaxation factors, 0.9 for both pressure and velocity, when for S-A, it was necessary to use 0.5 for pressure, 0.9 for velocity, and 0.3 for turbulent viscosity, which translates into a several-fold reduction in the number of steady-state solution iterations needed.

Holland et al. [62] further developed field inversion and machine learning (FIML) by embedding a neural network directly in the flow solver; in the solution proposed by Duraisamy et al. in 2016 [53] the machine learning training step was decoupled from the inversion procedure. This yielded improved results [60] over the original FIML for simulating turbulent, detached flow around a wind turbine blade.

Ching et al. [63] proposed a method for using neural networks to improve the accuracy of simulations of complex separated flows. The network is trained on the results of LES simulations and then used to modify the values of turbulence kinetic energy obtained from RANS calculations using a k-ε model. A Gaussian mixture model [64] is used to detect areas where the neural network extrapolates the result; the standard RANS model is used in these areas. Both the NN and Gaussian mixture model are coupled directly to the RANS solver so that the correction occurs in each iteration of the calculation. The method was tested on two cases—a wall-mounted cube and a skewed bump, with both obtaining improved prediction of turbulence kinetic energy values and average flow velocity [63].

Matai and Durbin [65] presented a sharply different method from most solutions for using machine learning to address the inaccuracy caused by fixed values of closure coefficients. A decision tree algorithm is used to divide the model into zones, depending on local flow characteristics. For each zone of the training case—two-dimensional flow over a bump—optimized values of closure coefficients giving the highest accuracy are determined. The k-ω [37] model modified in this way yields higher accuracy than a model using the same value of coefficients throughout the domain [65].

Kaandorp and Dwight [66] proposed in 2020 to use a modified random forest algorithm [67] to predict Reynolds stress anisotropy. Their algorithm, called tensor-basis random forest (TBRF) guarantees Galilean invariance by using a tensor basis. The algorithm was trained on the results of five DNS/LES simulations: periodic hills, a constricting channel, a curved backward-facing step, a backward-facing step, and a square duct. It was then used to predict Reynolds stress anisotropy in RANS simulations using the k-ε model. The results were compared with those of RANS simulations, DNS simulations, and the alternative tensor-basis neural network (TBNN) method proposed by Ling et al. [48] Results similar to baseline DNS and TBNN and significantly better than RANS simulations were obtained [66]. The advantage of the proposed solution is a simpler implementation than TBNN.

Obiols-Sales et al. [68] developed an approach based on replacing some of the iterations needed to find a complete RANS solution with a model based on machine learning. CFDNet, described by the authors as a "physics solver-CNN model–physics solver coupled framework," is a coupling of the open-source CFD code OpenFOAM with a model based on convolutional neural networks. The calculation begins with an experimentally determined number of initial iterations run by classical CFD software; then, the results for each physical quantity are stored as an image with the number of pixels equal to the number of grid cells. The neural network predicts how the distribution of physical quantities will evolve and the resulting values are entered back into the CFD model. The quality of the results obtained in this way is checked, and if the assumed convergence (a decrease of usually 4–5 orders of magnitude) has been achieved, the calculations are ended, otherwise, the CFD software continues until convergence is achieved. The idea is that such a process should be shorter than performing the same calculations using only classical CFD. The experiment was conducted for several different geometries, such as cylinders or airfoils, achieving a reduction in the calculation time of between a factor of 1.9 and 7.4 depending on the type of flow [68].

Another example of a departure from neural networks in determining closure coefficients in a turbulence model is the method developed by Beetham and Capecelatro [69] based on sparse regression [70]. The authors point out one disadvantage of NN-based solutions—these models act like a 'black box' and cannot be represented in algebraic form. The use of regression makes it possible to obtain the model in an algebraic form that allows its interpretation and integration of such obtained form in existing CFD simulation methods. In addition, thanks to the thoughtful tailoring of feature space, Galilean invariance can be guaranteed. The solution was tested by stimulating the flow through a periodically constricted channel and comparing it with the results of DNS, RANS, and also TBNN [48] simulations. A 41% reduction in the error of the Reynolds stress anisotropy value was obtained [69] and the results were comparable to those obtained by Ling et al. [48].

In 2017, Milani et al. [71] presented a way to improve the accuracy of heat flow estimation in RANS simulations; current models yield poor temperature predictions. The goal was to obtain more accurate values of turbulent diffusivity αt—one of the key parameters describing heat flow between a fluid and a solid. A supervised learning algorithm calibrated against data from DNS calculations of an inclined coolant jet in crossflow (validated against experimental data with good agreement) was used, with variables calculated in a RANS simulation using the k-ε model as input, while turbulent diffusivity values were the result. This resulted in a 50% reduction in error; the error level is still relatively high, but the best improvement was obtained in areas of high gradients near the walls so that the error in determining heat transfer itself can be significantly smaller [71].

Zhang et al. [72] proposed another way to use deep learning; using the standard k-ε model [36] as a benchmark, they replaced some of the source terms in the rate of dissipation of turbulent kinetic energy equation by a physically constrained deep learning model. The equation for turbulent kinetic energy was left unchanged because it follows directly from the Navier–Stokes equations. Baseline data used to train the model came from simulations of flow in the channel using the LES method. The model reduced the error by 51.7% compared to the standard k-ε model [72]. Additionally described is a method for correcting the training datasets if the data they contain deviate from the theoretical characteristic values of the k-ε model—this further reduced the error by 6.2% compared to the model without correction.

Ho and West [73] presented in 2021 an example of using FIML proposed by Duraisamy et al. in 2017 [53] to improve the accuracy of simulations of three-dimensional detached flow, studying, among other things, what the impact of correcting once is as opposed to correcting each iteration of the calculation. The model was trained on four two-dimensional cases: channel flow, a curved backward-facing step, a wall-mounted hump, and a cylinder, and one three-dimensional case: a conical hill. The model was used to correct the $k-\omega$SST model, yielding improvements in the prediction of detached flow [73].

3.3. Large Eddy Simulation (LES)

A method of turbulent flow simulation proposed by Smagorinski [74] and developed by Deardorff [75] is based on ignoring the smallest scale of turbulence and fully resolving (as in DNS) only the large, most significant vortices. Smaller vortices are filtered out however their influence on fluid behavior cannot be ignored and is determined using specialized models. The method is less computationally demanding and time-consuming than DNS but more so than RANS.

The advantage of LES is that the requirements are lower than in DNS, allowing LES to be used for engineering calculations while explicitly simulating larger turbulence. The disadvantage, however, is lower accuracy than DNS due to modeling smaller-scale turbulence. The method's wider range of practical applications has translated into more ML-based solutions developed for it.

Gamahara and Hattori [76] presented an artificial neural network (ANN)-based alternative to the subgrid models currently used in LES calculations. Since in LES calculations the flow is decomposed into grid scale (GS, resolved turbulent motion) and subgrid scale (SGS, modeled fluctuations of smaller scale turbulence), the effect of subgrid scale turbulence on grid scale flow (so-called SGS stress tensor) must be properly accounted for to obtain good results. There are many subgrid models, such as the Smagorinsky model [74], similarity model [77], or gradient model [78], and their modifications and combinations. None of these models has an advantage over the others in every type of flow. The authors proposed using ANN to predict the relationship between GS flow and SGS stress tensor, training their model on the results of DNS calculations of channel flow. The resulting model obtained results similar to the gradient model but did not show a significant advantage over classical models [76]. However, the mere fact that ANN found a relationship between GS flow and SGS stress tensor can be considered a success.

In 2017, Maulik and San [79] proposed a method for blind deconvolution of turbulent flows using an artificial neural network (ANN) motivated by recent advances in the application of machine learning for the reconstruction of noisy or blurred images [80]. Such deconvolution can allow the reconstruction of turbulent flow at a scale smaller than the model grid size—for example, in LES computational results where only larger-scale turbulence is fully resolved. Their approach is based on the assumption that the relationship between the flow fields described at high and low resolution can correspond to a convolution using a blur kernel. Their ANN attempts to determine the parameters of this kernel so that it can be inverted to produce a higher resolution flow field; previous attempts at deconvolution have relied on knowledge of the filter kernel used; for example, in LES simulations [81], their solution does not use such information so it is more universal. They ran their trials on the results of three flow simulations: 2D Kraichan, 3D Kolmogorov, and compressible stratified turbulence. The results obtained are similar to those obtained in deconvolution conducted with knowledge of the filter kernel [78].

Beck et al. [82] presented a method for modeling turbulence in LES calculations based on deep neural networks (DNNs). This approach is similar to those used in methods designed to improve RANS calculations; a neural network is used to establish closure coefficients. The grid scale (GS) flow values were used as inputs and the network itself was trained on the results of DNS and LES calculations of decaying homogeneous isotropic turbulence (HIT) [83]; LES used a grid coarsened by a factor of 8 in any direction. It was found that convolutional neural networks, especially their variant called residual neural networks, give good results. The convergence of the best results to DNS was about 45% (73% when omitting outer points) [82]. The authors presented that network performance is limited by the available amount of data.

Wang et al. [84] presented TF-Net, a combination of two classical CFD methods, RANS and LES, with machine learning. The combination of the computational efficiency of RANS and the ability to resolve turbulent flow offered by LES yields a method that is less time-consuming and easier to apply than classical LES. The approach is based on using a neural network to divide the velocity field into the mean flow, resolved fluctuations, and unresolved, modeled, fluctuations while maintaining the correct direction of energy flow. CFD typically uses predefined filters for this purpose. TF-Net is based on convolutional neural networks (CNNs); a graphical representation is created for each part of the decomposed velocity field produced by LES simulation of a Rayleigh-Bénard convection flow [85,86] and the images obtained are the input for the CNNs. Based on these fields, the model determines the values of the kinetic energy of turbulence in the same domain and also, this is the feature that distinguishes TF-Net from other solutions and allows it to simulate the evolution of the flow by generating graphical representations of the velocity field for successive time-steps. The CNNs used in TF-Net performed better than all the neural networks with which it was compared [84].

Frezat et al. [87] presented a method for modeling subgrid-scale turbulence using physics-informed neural networks (NNs). The motivation behind their work was the desire to use NNs to improve LES calculations and at the same time to ensure physical correctness by adding constraints based on physical laws into the model; neural networks trained on results from DNS may not preserve physical rules which puts the possibility of their practical application into question. Their solution called the subgrid transport neural network (SGTNN) guarantees compliance with the laws of physics by, among other things, satisfying the Galilean invariance condition. The network was trained on the results of DNS homogeneous isotropic turbulence (HIT) [82] calculations and obtained an accuracy of predictions better than the convolutional neural network (CNN) and multi-level perceptron (MLP) [87].

3.4. Examples of Additional CFD Methods

Detached eddy simulation (DES) [88] is a combination of RANS and LES calculations. During the calculations, the ratio of the turbulence scale to the size of the grid elements at a given location is continuously checked; if the turbulence scale is larger, a given region is resolved

using the LES approach, otherwise, RANS is used. This method allows the use of grids of lower resolution than that required by LES so that calculations are less time-consuming.

Coherent vortex simulation (CVS) [89] is based on a similar approach to LES—splitting the flow into two components, but unlike LES the components are coherent vortex motion and non-coherent random background flow. The way of filtering also differs, where LES uses a linear low-pass filter whereas CVS filtering is based on wavelets [90].

More varied are the solutions to improve the results of RANS simulations, more numerous due to the higher popularity of the method (Table 1, Figure 6). Two trends can be distinguished among them: the drive to improve the quality of results—the vast majority of publications—and the drive to reduce computation time. Solutions focusing on the quality of results try to solve the problem of inherent error resulting from the assumptions and simplifications on which RANS is based—the use of constant, empirical closure coefficients and assumptions about the value of Reynolds stress (anisotropy and dependence on mean flow). All of the aforementioned solutions achieved quality improvements for the flows at which they were trained, some also outside the range included in the training data. However, none of the cases investigated the possibility of using such models in the calculation of flows of a completely different nature (e.g., internal flow vs. external flow) than the one on which the model was trained. An important aspect considered by some of the authors was compliance with the laws of physics, such as satisfying the Galilean invariance condition, hence many solutions are physics informed. Solutions range from simply finding the best closure coefficients [44] through to correcting the Reynolds stress anisotropy [48] up to completing surrogate models for RANS [61]. The problem posed by some of the ML-based solutions stems from them being a "black-box"; without a way to interpret the principle of their operation, there is a risk that the results obtained will be seemingly correct but subject to a larger error than that of classical RANS, even if the solution is "physics informed." In this regard, the method developed by Beetham and Capecelatro [69] based on regression stands out, allowing the model to be obtained in algebraic form, enabling interpretation and examination of compliance with the laws of physics. Of the solutions aimed at speeding up calculations, an interesting example is that proposed by Obiols-Sales et al. [68] where machine learning speeds up the intermediate iterations of the solution, but the final result is calculated in the classical way thus not differing in the quality of results from RANS.

Table 1. The list of selected papers with their corresponding methodology (marked with a bullet in the appropriate column).

Authors	DNS	RANS	LES	ANN	CNN	Regression	DT
Pathak [28]	•			•			
Kochkov [30]	•			•			
Yarlanki [44]		•		•			
Tracey [45]		•				•	
Tracey [47]		•		•			
Ling [48]		•		•			
Singh [51]		•		•			
Duraisamy [52]		•		•			•
Wang [55]		•					•
Geneva [59]		•		•			
Maulik [61]		•		•			
Holland [62]		•		•			•

Table 1. *Cont.*

Authors	DNS	RANS	LES	ANN	CNN	Regression	DT
Ching [63]		•		•			
Matai [65]		•					•
Kaandorp [66]		•					•
Obiols-Sales [68]		•			•		
Beetham [69]		•				•	
Milani [71]		•					•
Zhang [72]		•		•			
Ho [73]		•		•			•
Gamahara [76]			•	•			
Maulik [79]			•	•			
Beck [82]			•	•			
Wang [84]		•	•		•		
Frezat [87]			•	•			

ANN = Artificial Neural Network, CNN = Convolutional Neural Network, DT = Decision Tree.

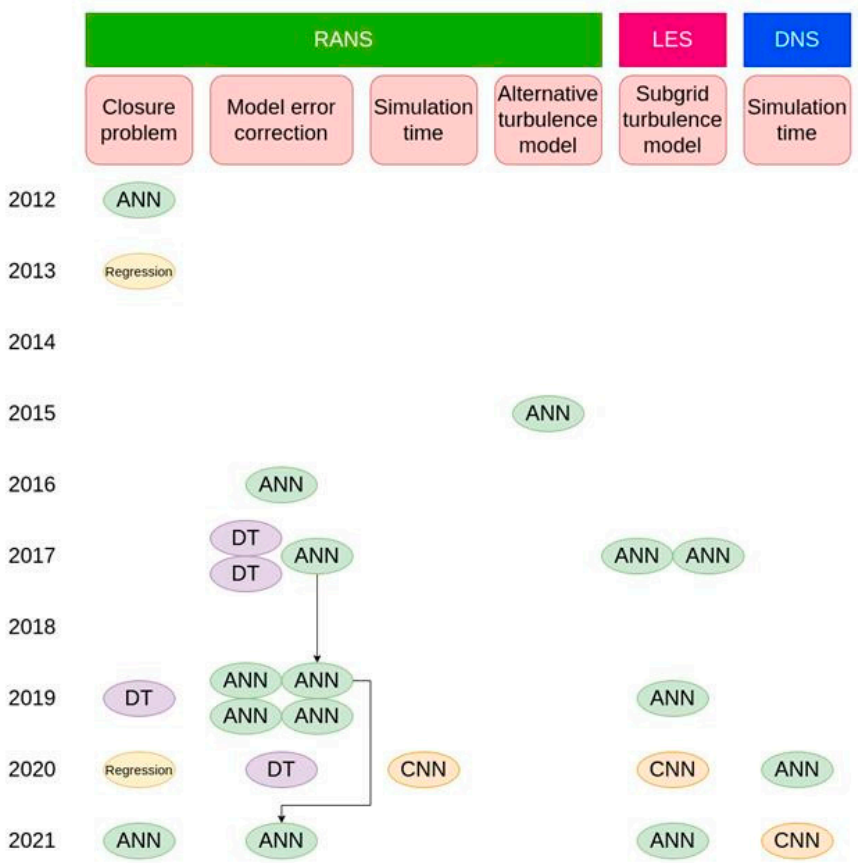

Figure 6. The progress and evolution of the analyzed methods over the last years sorted by the problem defined.

4. Conclusions

The solutions described above (in the case of machine learning applications aimed at improving DNS calculations) focus on reducing hardware requirements and computation time as the main drawbacks of the method. This effect is achieved by conducting calcula-

tions using a lower-resolution grid, projecting its results onto a full-resolution grid, and using machine learning to reconstruct the lost information. These solutions produce the desired effect but due to the low availability of sample results of DNS calculations—especially in 3D—and their problematic size, they were trained on a limited number of examples. No 3D simulation results are available for real-life examples hence the use of idealized examples such as decaying homogeneous isotropic turbulence (HIT) in such work. These limitations, although solvable thanks to the growing capabilities of computers, mean that the ability of such solutions to generalize beyond the range on which they were trained may be questionable.

In the scope of the conclusions of the CFD 2030 Vision report regarding classical turbulence models reaching the limit of their development potential and the still extremely time-consuming nature of DNS calculations that do not use these models, the possibilities offered by machine-learning-based methods seem to be an interesting alternative. Reducing computation time by whole orders of magnitude or even running them in real-time while reducing hardware requirements could translate into tangible financial benefits for institutions running flow simulations.

However, the proposals outlined above usually apply to a single example application and may not be able to generalize well, which may limit interest in these methods, and without interest from industry, the development of these methods may be hampered. What is noteworthy is the number of solutions to improve each simulation method—most are for RANS, a smaller number are for LES, and very few are for DNS, which is consistent with the popularity of these methods.

One can see the evolution of solutions from the simple determination of closure coefficients through to more advanced attempts to use machine learning as an alternative to the classical methods of solving differential equations on which CFD is based, up to turbulence models built solely from neural networks. The second clear trend is the increasing importance being placed on ensuring that the results comply with the laws of physics—something that in the case of classical flow simulation methods is guaranteed by their reliance on the Navier–Stokes equations. Finally, a shift can be seen from highly specialized solutions aimed at a specific problem to more universal solutions integrated with existing CFD software, which may be key in terms of industry acceptance. A continuation of these three trends may lead to at least a partial replacement of Navier–Stokes-based CFD by machine-learning-based solutions in the future.

Classic CFD simulation methods such as DNS, RANS, and LES still have one advantage—their limitations are well known and can be taken into account when analyzing simulation results. When OpenFOAM—the most popular free and open-source CFD software today—appeared in 2004 and flow calculations became available to a wide number of institutions previously discouraged by the high price of commercial software, turbulence models were already decades old and well studied.

It is fair to ask whether even a significant advantage in the areas of computational speed and hardware requirements will prompt CFD software users to move away from proven paths and adopt new solutions based on machine learning.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Reynolds, O. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Philos. Trans. R. Soc.* **1883**, *174*, 935–982. [CrossRef]
2. Sheppard, W.F. Central-Difference Formulæ. *Proc. Lond. Math. Soc.* **1899**, *1*, 449–488. [CrossRef]
3. Richardson, L.F. *Weather Prediction by Numerical Process*; Cambridge University Press: Cambridge, UK, 1922; p. 219.
4. Courant, R.; Friedrichs, K.; Lewy, H. Über die partiellen Differenzengleichungen der mathematischen Physik. *Math. Ann.* **1928**, *100*, 32–74. (In German) [CrossRef]

5. Thom, A. The Flow Past Circular Cylinders at Low Speeds. *Proc. R. Soc. Lond.* **1933**, *141*, 651–669.
6. Von Neumann, J.; Richtmyer, R.D.J. A Method for the Numerical Calculation of Hydrodynamic Shocks. *Appl. Phys.* **1950**, *21*, 232–237, Republished in Collected Works; Pergamon: New York, NY, USA, 1961; Volume 6, pp. 380–385.
7. Moore, G.E. Cramming more components onto integrated circuits. *Electronics* **1965**, *38*, 114ff.
8. Roache, P.J. *Computational Fluid Dynamics*; Hermosa Publications: Albuquerque, NM, USA, 1972.
9. Baker, T.J. Unstructured meshes and surface fidelity for complex shapes. In Proceedings of the 10th Computational Fluid Dynamics Conference, Honolulu, HI, USA, 24–26 June 1991; pp. 714–725.
10. Melton, J.E.; Pandya, S.A.; Steger, J.L. 3D Euler flow solutions using unstructured Cartesian and prismatic grids. In Proceedings of the 31st Aerospace Sciences Meeting, Reno, NV, USA, 11–14 January 1993.
11. Steger, J.L.; Dougherty, F.C.; Benek, J.A. A Chimera grid scheme. In Proceedings of the ASME Mini-Symposium on Advances in Grid Generation, Houston, TX, USA, June 1982.
12. Buning, P.G.; Chiu, I.T.; Obayashi, S.; Rizk, Y.M.; Steger, J.L. Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent. In Proceedings of the 15th Atmospheric Flight Mechanics Conference, Minneapolis, MN, USA, 15–17 August 1988.
13. Message Passing Interface Forum. MPI: A Message Passing Interface. In Proceedings of the Supercomputing 93, Portland, OR, USA, 15–19 November 1993; IEEE Computer Society Press: Washington, DC, USA, 1993; pp. 878–883.
14. Riggins, D.; Underwood, M.; McMillin, B.; Reeves, L.; Jui-Lin, E. Modeling of Supersonic Combustor Flows using Parallel Computing. *Comput. Sci. Tech. Rep.* **1993**, *36*, 217–229. [CrossRef]
15. OpenCFD Ltd. OpenFOAM Launched 10th December 2004; Archived from the original on 8 February 2005. Retrieved 20 August 2019. Available online: https://web.archive.org/web/20050208124617/http://www.opencfd.co.uk/openfoam/launch.html (accessed on 15 December 2022).
16. Kolmogorov, A.N. The local structure of turbulence in incom-pressible viscous fluids at very large Reynolds numbers. *Dokl. Akad. Nauk. SSSR* **1941**, *30*, 299–303, Reprinted in *Proc. R. Soc. London A* **1991**, *434*, 9–13.
17. Baez, J.C. *Open Questions in Physics, Usenet Physics FAQ*; Riverside: Department of Mathematics, University of California: Berkeley, CA, USA, 2006.
18. Tennekes, H.; Lumley, J.L. *A First Course in Turbulence*; MIT Press: Cambridge, MA, USA, 1972; ISBN 9780262200196.
19. Devlin, K.J. *The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*; Basic Books: New York, NY, USA, 2002; ISBN 0-465-01729-0.
20. Orszag, S.A. Analytical Theories of Turbulence. *J. Fluid Mech.* **1970**, *41*, 363–386. [CrossRef]
21. Leray, J. Sur le mouvement d'un liquide visqueux emplissant l'espace. *Acta Math.* **1934**, *63*, 193–248.
22. Shafiq, A.; Çolak, A.B.; Sindhu, T.N. Reliability In-vestigation of Exponentiated Weibull Distribution Using IPL through Numerical and Artificial Neural Network Modeling. In *Quality and Reliability Engineering International*; Wiley: Hoboken, NJ, USA, 2022; pp. 1–16.
23. Çolak, A.B. Experimental study for thermal conductivity of water-based zirconium oxide nanofluid: Developing optimal artificial neural network and proposing new correlation. *Int. J. Energy Res.* **2020**, *45*, 2912–2930.
24. Shafiq, A.; Çolak, A.B.; Sindhu, T.N.; Lone, S.A.; Alsubie, A.; Jarad, F. Comparative Study of Artificial Neural Network versus Parametric Method in COVID-19 data Analysis. *Results Phys.* **2022**, *38*, 105613. [CrossRef]
25. Blazewicz, J.; Borowski, M.; Chaara, W.; Kedziora, P.; Klatzmann, D.; Lukasiak, P.; Six, A.; Wojciechowski, P. GeVaDSs–decision support system for novel Genetic Vaccine development process. *BMC Bioinform.* **2012**, *13*, 91.
26. Blazewicz, J.; Lukasiak, P.; Wilk, S. New machine learning methods for prediction of protein secondary structures. *Control. Cybern.* **2007**, *36*, 183–201.
27. Majchrzak, M.; Jakubowski, M.; Starosta, R. AI-based Method of Vortex Core Tracking as an Alternative for Lambda2. *Vib. Phys. Syst.* **2020**, *31*, 10.
28. Pathak, J.; Mustafa, M.; Kashinath, K.; Motheau, E.; Kurth, T.; Day, M. Using Machine Learning to Augment Coarse-Grid Computational Fluid Dynamics Simulations. *arXiv* **2020**, arXiv:2010.00072. [CrossRef]
29. Bar-Sinai, Y.; Hoyer, S.; Hickey, J.; Brenner, M.P. Learning data-driven discretizations for partial differential equations. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 15344. [CrossRef]
30. Kochkov, D.; Smith, J.A.; Alieva, A.; Wang, Q.; Brenner, M.P.; Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2101784118. [CrossRef]
31. Sirignano, J.; MacArt, J.L.; Freund, J.B. A deep learning pde augmentation method with application to large-eddy simulation. *J. Comput. Phys.* **2020**, *423*, 109811.
32. Reynolds, O. On the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. *Philos. Trans. R. Soc. Lond. A* **1895**, *186*, 123–164.
33. Boussinesq, J. Essai sur la théorie des eaux courantes. In *Mémoires Présentés par Divers Savants à l'Académie des Sciences*; Imprimerie Nationale: Paris, France, 1897; Volume 23, pp. 1–680.
34. Schmitt, F.G. About Boussinesq's turbulent viscosity hypothesis: Historical remarks and a direct evaluation of its validity. *Comptes Rendus Mécanique* **2007**, *335*, 617–627. [CrossRef]
35. Margheri, L.; Meldi, M.; Salvetti, M.V.; Sagaut, P. Epistemic uncertainties in RANS model free coefficients. *Comput. Fluids* **2014**, *102*, 315–335.
36. Hanjalic, K.; Launder, B. A Reynolds stress model of turbulence and its application to thin shear flows. *J. Fluid Mech.* **1972**, *52*, 609–638.

37. Wilcox, D.C. Formulation of the k-omega Turbulence Model Revisited. *AIAA J.* **2008**, *46*, 2823–2838. [CrossRef]
38. Menter, F.R. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. *AIAA J.* **1994**, *32*, 1598–1605.
39. Spalart, P.; Allmaras, S. A one-equation turbulence model for aerodynamic flows. In Proceedings of the 30th Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 6–9 January 1992.
40. Roache, P.J. Perspective: A method for uniform reporting of grid refinement studies. *J. Fluids Eng.* **1994**, *116*, 405–413. [CrossRef]
41. Sosnowski, M.; Krzywanski, J.; Scurek, R. A fuzzy logic approach for the reduction of mesh-induced error in CFD analysis: A case study of an impinging jet. *Entropy* **2019**, *21*, 1047.
42. Von Kármán, T. Mechanische Ähnlichkeit und Turbulenz. In *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Fachgruppe 1 (Mathematik)*; NACA: Washington, DC, USA, 1930; Volume 5, pp. 58–76.
43. Slotnick, J.; Khodadoust, A.; Alonso, J.; Darmofal, D.; Gropp, W.; Lurie, E.; Mavriplis, D. *Cfd Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*; Langley Research Center, National Aeronautics and Space Administration: Washington, DC, USA, 2014.
44. Yarlanki, S.; Rajendran, B.; Hamann, H. Estimation of turbulence closure coefficients for data centers using machine learning algorithms. In Proceedings of the 13th InterSociety Conference on Thermal and Thermomechanical Phenomena in ElectronicSystems, San Diego, CA, USA, 30 May–1 June 2012.
45. Tracey, B.D.; Duraisamy, K.; Alonso, J. Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. In Proceedings of the 51stAIAA Aerospace Sciences Meeting, Dallas, TX, USA, 7–10 January 2013.
46. Matta, A.; Li, N.; Lin, Z.; Shanthikumar, J.G. Operational Learning of Approximate Analytical Methods for Performance Evaluation of Manufacturing Systems. In Proceedings of the 10th Conference on Stochastic Models of Manufacturing and Service Operations SMMSO, Volos, Greece, 1–6 June 2011.
47. Tracey, B.D.; Duraisamy, K.; Alonso, J. A machine learning strategy to assist turbulence model development. In Proceedings of the 53rd American Institute of Aeronautics and Astronautics Aerospace Sciences Meeting, Kissimmee, FL, USA, 5–9 January 2015.
48. Ling, J.; Kurzawski, A.; Templeton, J. Reynolds averaged turbulence modeling using deep neural networks with embedded invariance. *J. Fluid Mech.* **2016**, *807*, 155–166.
49. Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*; Spartan Books: Washington, DC, USA, 1961.
50. Ray, J.; Lefantzi, S.; Arunajatesan, S.; Dechant, L. Learning an Eddy Viscosity Model Using Shrinkage and Bayesian Calibration: A Jet-in-Crossflow Case Study. *ASME J. Risk Uncertain. Part B. Mar.* **2018**, *4*, 011001. [CrossRef]
51. Singh, A.P.; Duraisamy, K. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA J.* **2017**, *55*, 2215. [CrossRef]
52. Duraisamy, K.; Singh, A.P.; Pan, S. Augmentation of Turbulence Models Using Field Inversion and Machine Learning. In Proceedings of the 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, USA, 9–13 January 2017; pp. 2215–2227.
53. Duraisamy, K.; Zhang, Z.J.; Singh, A.P. New Approaches in Turbulence and Transition Modeling Using Data-driven Techniques. In Proceedings of the 53rd AIAA Aerospace Sciences Meeting, Kissimmee, FL, USA, 5–9 January 2015.
54. Parish, E.J.; Duraisamy, K. A paradigm for data-driven predictive modeling using field inversion and machine learning. *J. Comput. Phys.* **2016**, *305*, 758–774.
55. Wang, J.X.; Wu, J.L.; Xiao, H. A physics informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Phys. Rev. Fluids* **2017**, *2*, 0346039.
56. Wu, J.-L.; Xiao, H.; Paterson, E. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Phys. Rev. Fluids* **2018**, *3*, 074602.
57. Ben Gal I Bayesian Networks. *Encyclopedia of Statistics in Quality and Reliability*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
58. Blundell, C.; Cornebise, C.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural networks. *arXiv* **2015**, arXiv:1505.05424.
59. Geneva, N.; Zabaras, N. Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks. *J. Comput. Phys.* **2019**, *383*, 125. [CrossRef]
60. Liu, Q.; Wang, D. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 2378–2386.
61. Maulik, R.; Sharma, H.; Patel, S.; Lusch, B.; Jennings, E. Accelerating RANS turbulence modeling using potential flow and machine learning. *arXiv* **2019**, arXiv:1910.10878.
62. Holland, J.R.; Baeder, J.D.; Duraisamy, K. Field inversion and machine learning with embedded neural networks: Physics-consistent neural network training. In Proceedings of the AIAA Aviation 2019 Forum, Dallas, TX, USA, 17–21 June 2019.
63. Ching, D.S.; Banko, A.J.; Milani, P.M.; Eaton, J.K. Machine learning modeling for RANS turbulence kinetic energy transport in 3D separated flows. In Proceedings of the 11th International Symposium on Turbulence and Shear Flow Phenomena, Southampton, UK, 22–30 July 2022.
64. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the em algorithm. *J. R. Stat. Soc. Ser. B (Methodol.)* **1977**, *39*, 1–22.
65. Matai, R.; Durbin, P.A. Zonal eddy viscosity models based on machine learning. *Flow Turbul. Combust.* **2019**, *103*, 93–109.
66. Kaandorp, M.L.; Dwight, R.P. Data-driven modeling of the Reynolds stress tensor using random forests with invariance. *Comput. Fluids* **2020**, *202*, 104497. [CrossRef]
67. Ho, T.K. Random Decision Forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, USA, 14–16 August 1995; pp. 278–282.

68. Obiols-Sales, O.; Vishnu, A.; Malaya, N.; Chandramouli Sharan, A. CFDNet: A deep learning-based accelerator for fluid simulations. In Proceedings of the 34th ACM international conference on supercomputing, Barcelona, Spain, 29 June–2 July2020; pp. 1–12.
69. Beetham, S.; Capecelatro, J. Formulating turbulence closures using sparse regression with embedded form invariance. *Phys. Rev. Fluids* **2020**, *5*, 084611. [CrossRef]
70. Donoho, D.L.; Elad, M. Optimally sparse representation in general (nonorthogonal) dictionaries via L1 minimization. *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 2197–2202.
71. Milani, P.M.; Ling, J.; Saez-Mischlich, G.; Bodart, J.; Eaton, J.K. A machine learning approach for determining the turbulent diffusivity in film cooling flows. *J. Turbomach.* **2018**, *140*, 021006.
72. Zhu, L.; Zhang, W.; Sun, X.; Liu, Y.; Yuan, X. Turbulence closure for high Reynolds number airfoil flows by deep neural networks. *Aerosp. Sci. Technol.* **2021**, *110*, 106452. [CrossRef]
73. Ho, J.; West, A. Field Inversion and Machine Learning for turbulence modeling applied to three-dimensional separated flows. In Proceedings of the AIAA Aviation 2021 Forum, Virtual, 2–6 August 2021; pp. 2021–2903.
74. Smagorinsky, J. General Circulation Experiments with the Primitive Equations. *Mon. Weather. Rev.* **1963**, *91*, 99–164.
75. Deardorff, J. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *J. Fluid Mech.* **1970**, *41*, 453–480.
76. Gamahara, M.; Hattori, Y. Searching for turbulence models by artificial neural network. *Phys. Rev. Fluids* **2017**, *2*, 054604. [CrossRef]
77. Bardina, J.; Ferziger, J.H.; Reynolds, W.C. *Improved Turbulence Models Based on LES of Homogeneous Incompressible Turbulent Flows, Rep. TF-19*; Department of Mechanical Engineering: Stanford, CA, USA, 1984.
78. Clark, R.A.; Ferziger, J.H.; Reynolds, W.C. Evaluation of subgrid-scale models using an accurately simulated turbulent flow. *J. Fluid Mech.* **1979**, *91*, 1. [CrossRef]
79. Maulik, R.; San, O. A neural network approach for the blind deconvolution of turbulent flows. *J. Fluid Mech.* **2017**, *831*, 151–181. [CrossRef]
80. Cichocki, A.; Amari, S. *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2002.
81. Stolz, S.; Adams, N.A.; Kleiser, L. The approximate deconvolution model for large-eddy simulations of compressible flows and its application to shock-turbulent-boundary-layer interaction. *Phys. Fluids* **2001**, *13*, 2985–3001. [CrossRef]
82. Beck, A.; Flad, D.; Munz, C.-D. Deep neural networks for data-driven LES closure models. *J. Comput. Phys.* **2019**, *398*, 108910. [CrossRef]
83. Orszag, S.A.; Patterson, G.S. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.* **1972**, *28*, 76–79. [CrossRef]
84. Wang, R.; Kashinath, K.; Mustafa, M.; Albert, A.; Yu, R. Towards physics informed deep learning for turbulent flow prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 1457–1466.
85. Bénard, H. Les tourbillons cellulaires dans une nappe liquide [Cellular vortices in a sheet of liquid]. *Rev. Générale Des Sci. Pures Et Appliquées* **1900**, *11*, 1261–1271. (In French)
86. Rayleigh, L. On the convective currents in a horizontal layer of fluid when the higher temperature is on the under side. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1916**, *32*, 529–546. [CrossRef]
87. Frezat, H.; Balarac, G.; Le Sommer, J.; Fablet, R.; Lguensat, R. Physical invariance in neural networks for subgrid-scale scalar flux modeling. *Phys. Rev. Fluids* **2021**, *6*, 024607. [CrossRef]
88. Spalart, P.R. Comments on the feasibility of LES for wing and on a hybrid RANS/LES approach. In Proceedings of the 1st Asosr Conference on DNS/LES, Arlington, TX, USA, 4–8 August 1997.
89. Farge, M.; Schneider, K. Coherent Vortex Simulation (CVS), A Semi-Deterministic Turbulence Model Using Wavelets. *Flow Turbul. Combust.* **2001**, *66*, 393–426. [CrossRef]
90. Haar, A. Zur Theorie der orthogonalen Funktionensysteme. *Math. Ann.* **1910**, *69*, 331–371.