Article

Computational Efficiency Assessment of Adaptive Mesh Refinement for Turbulent Jets in Crossflow

Ante Sikirica ^{1,2} , Luka Grbčić ¹ , Marta Alvir ¹ and Lado Kranjčević ^{1,*}

¹ Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia; ante.sikirica@uniri.hr (A.S.); luka.grbic@riteh.hr (L.G.); malvir@riteh.hr (M.A.)

² Center for Advanced Computing and Modelling, University of Rijeka, Radmile Matejčić 2, 51000 Rijeka, Croatia

* Correspondence: lado.kranjcevic@riteh.hr; Tel.: +385-51-651-554

Abstract: Numerical analyses of environmental discharges are commonly conducted on pre-generated numerical grids with refinements implemented in regions of interest or influence on the flow field. This approach to problem formulation relies on insights into the flow specifics so that appropriate attention is given to relevant segments of the domain. In this paper we investigated the applicability of adaptive mesh refinement (AMR) on a commonly considered environmental problem—a jet in crossflow. The assessment was made using the OpenFOAM toolbox. Several RANS turbulence models and grid generation approaches were compared in terms of accuracy to previous studies and experimental results. Main emphasis is given to the computational efficiency of the methodology with a focus on load distribution. Our findings indicate that the results are acceptable in terms of accuracy with load balancing providing significant computational savings thus enabling AMR methodology to outperform the conventional approach.

Keywords: jet in crossflow; turbulent mixing; adaptive mesh refinement; load balancing

Citation: Sikirica, A.; Grbčić, L.; Alvir, M.; Kranjčević, L. Computational Efficiency Assessment of Adaptive Mesh Refinement for Turbulent Jets in Crossflow. *Mathematics* **2022**, *10*, 620. <https://doi.org/10.3390/math10040620>

Academic Editors: Mohammad Mehdi Rashidi, Mostafa Safdari Shadloo and Alessio Alexiadis

Received: 13 January 2022

Accepted: 15 February 2022

Published: 17 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Adaptive mesh refinement (AMR) is a valuable method for numerous computational fluid dynamic (CFD) problems as it brings about a reduction in computational requirements and ultimately computational time. Effectiveness of the methodology has been assessed throughout the years, however, in the case of OpenFOAM, due to performance degradation associated with the processor overload and imbalance, it has not always been the optimal or fastest tool [1,2]. With the advancements pertaining to the load balancing, it is now paramount to investigate and present AMR in the context of current developments.

Numerous studies have so far considered the AMR approach for interface tracking between phases. Maric et al. [3] proposed a geometrical VOF algorithm that employs dynamic AMR for interface reconstruction. The overall accuracy was acceptable with negligible computational overhead. The phase field method for wetting phenomena presented by Cai et al. [4] relied on AMR rather than the conventional grid generation approach in order to achieve computational efficiency. Cavitation modeling conducted by Li et al. [5] employed AMR to reduce computational requirements of an LES-VOF model. Similarly, Wang et al. [6] used AMR in order to assess cavitation around a hydrofoil. Due to adaptive refinement, complex structures were captured at a reduced computational cost. Adaptive mesh refinement was used in a study by Ismail et al. [7] to model the combustion process of a fuel spray. The obtained results indicated errors of up to 15% and a reduction in computational time by up to 60%. Fuel spray modeling using LES and AMR was further investigated by Hindi et al. [8]. The authors introduced a new criterion for adaptive mesh refinement and showed that the proposed approach can improve results without the need for any additional tuning. Deising et al. [1] introduced an integrated framework for the simulation of dilute species transfer. The framework unified the VOF approach, AMR and

load balancing. Adaptive mesh refinement is an essential part of the fireFoam toolkit [9]. Recently, based on current advancements, load balancing was also included [10]. Presented works predominantly focused on the general methodology and applicability of AMR for specific problems. The underlying algorithms and implementations are an independent topic and are briefly examined in order to substantiate the need for AMR assessment in view of recent advancements.

Basic AMR algorithm applicable to purely hexahedral grids and with limited flexibility is included with the OpenFOAM distributions. Innate isotropic implementation has been assessed and an algorithm extending the capabilities has been proposed by Karlsson [11]. This study provided a functional implementation of the anisotropic AMR. Joshi [12] proposed a modified algorithm that supports arbitrary cells and a refinement criterion based on the concept of multiresolution analysis. Load balanced AMR introduced by Rettenmaier et al. [2] improved current AMR capabilities by extending the AMR to axisymmetric cases, adding multi-criteria refinement and additionally included native load balancing. The paper by Ribeiro et al. [13] presented a dynamic load balancing technique, which aims to efficiently redistribute the workload based on system heterogeneity and overall use. Performance gains of up to 38% have been noted. Since the potential of the AMR approach has been previously established, these developments prompt further assessment.

Improvements and applicability of AMR will hence be assessed on a turbulent mixing problem known as a jet in crossflow (JIC). Although common in engineering applications (e.g., mixers), most notably, the topic of a jet in crossflow can be associated with pollutant dispersion (e.g., chimney discharge). Due to interactions of the influents, these flows are typically complex, unsteady and nonlinear. Navier–Stokes equations govern the flow. By incorporating a scalar transport equation, intricacies of the turbulent mixing can be investigated, as noted in works by Galeazzo et al. [14,15]. Aforementioned studies investigated the applicability of Reynolds-averaged Navier–Stokes (RANS) turbulence models and large eddy simulations (LES). The overall agreement with the measurements for mean values as well as jet penetration was acceptable; however, RANS models struggled to predict coherent structures and were hence deemed inadequate. Similarly, a study on turbulent Schmidt number conducted by Ivanova et al. [16] indicated that the accuracy of LES was superior to RANS, yet for practical use, RANS models were deemed acceptable. Both studies [15,16] suggested the use of different Schmidt numbers for LES and RANS cases. Consequently, the best approach remains uncertain. In order to improve the effectiveness of RANS models, specifically $k - \epsilon$ model, Lefantzi et al. [17] tried to optimize model parameters. The authors have shown that the calibration can improve accuracy of the simulations. Valero and Bung [18] assessed RANS turbulence models and their applicability for jet in crossflow configurations. RANS models, although inherently flawed, were considered competent for general use.

Analyses of jets in crossflow for Reynolds numbers $Re \leq 5000$ were conducted by Su and Mungal [19] and Denev et al. [20]. These papers investigated associated coherent structures. Influence of the jet nozzle shape was assessed by Salewski et al. [21]. Nozzles with a high aspect ratio as well as blunt nozzles were deemed superior in the context of turbulent mixing. Buoyant and non-buoyant jets in crossflow were analyzed by Cintolesi et al. [22]. The comparison revealed that non-buoyant jets (jets) show lower vertical velocity, consequently lower trajectory, have a higher eccentricity and do not produce similar turbulent structures as buoyant jets (plumes). Study conducted by Ryan et al. [23] considered skewed jet and employed LES to assess the single-vortex structure. Numerical results and experimental observations indicated that the assumption of a uniform Schmidt number for the problem in question, which is common for RANS, is inaccurate. Recently, a study by Milani et al. [24] proposed the use of deep learning to predict tensorial diffusivity, which is subsequently used in averaged scalar transport equations. Introduced modifications improved alignment and accuracy of the scalar flux vector and mean scalar field.

In this study, three distinct but intertwined avenues of investigation are considered: general flexibility of the AMR methodology, load balancing associated computational effi-

ciency and applicability of AMR for plume modeling. The code utilized in this paper is a custom modified OpenFOAM variant with integrated recent algorithmic advancements which ought to improve the computational efficiency. Efficiency of AMR with implemented load balancing is generally underinvestigated, hence a systematic assessment of the adaptive mesh refinement and analysis of the influence of load balancing were conducted. The significance of AMR in the HPC environment was evaluated for a jet in crossflow problem. As this is one of the most common yet fundamental pollutant dispersion problems, computational savings can be of great importance. Additionally, savings can be twofold: in terms of preprocessing related to grid generation as well as total evaluation time. Overall accuracy for different RANS turbulence models was also investigated. Comparison with the conventional grid generation approach in terms of accuracy and computational efficiency was conducted. Finally, results are compared with the numerical results and experimental measurements from previous studies. The primary goal of this study was to assess the applicability of AMR and establish a simple workflow and set of guidelines that should facilitate future assessments of the JIC problem by utilizing a computationally efficient approach.

2. Materials and Methods

2.1. Experimental Measurements

Considered experimental results and relevant setup have been previously described in [14,15]. Results have been obtained using the particle image velocimetry (PIV)/laser-induced fluorescence (LIF) methodology. The test setup consisted of a rectangular channel with a square cross section and a pipe. The pipe was mounted flush with the wall thus creating a round jet entrance. Diameter of the jet inlet equals $D = 8 \times 10^{-3}$ m. Crossflow entrance side equals $a = 0.108$ m. At the crossflow inlet, velocity is 9.08 m/s hence the Reynolds number is 6.2×10^4 . Velocity at the jet inlet is 37.72 m/s with $Re = 1.9 \times 10^4$. The velocity ratio equals $R = (\rho_j U_j^2) / (\rho_c U_c^2) = 4.15$, where ρ and U are respective densities and velocities. Additional information on the experimental setup and specifics can be found in the original paper [14].

2.2. Numerical Setup

Numerical simulations have been conducted in OpenFOAM [25]. A simplified computational domain has been used in all test cases [15]. An overview of the computational domain and relevant dimensions in meters has been given in Figure 1. All henceforth considered numerical grids are hexahedral.

Figure 1. Computational domain and relevant dimensions in meters.

At the crossflow and jet inlet, appropriate velocities are prescribed. Remaining patches, except for the outlet, are fixed walls. Value of the dimensionless wall distance, y^+ , is below 4.52 in all test cases. According to [26], since this is a global maximum, design should be adequate as the first cell is within the viscous sublayer. Invariability of the first cell thickness in all AMR tests is ensured by excluding the relevant part of the domain from the refinement. Appropriate wall treatment is used for all considered turbulence models with blending enabled for turbulent viscosity through the employed `nutUBlendedWallFunction` boundary condition [27]. The considered fluid is air with a constant density and kinematic viscosity of $1.57 \times 10^{-5} \text{ m}^2/\text{s}$. Turbulent mixing is assessed based on the concentration of the passive scalar, c , which is governed by the advection–diffusion equation

$$\frac{\partial c}{\partial t} = D_t \nabla^2 c - \mathbf{u} \nabla c \quad (1)$$

where t is time, D_t turbulent diffusivity and $\mathbf{u}$ velocity. The aforementioned transport equation as well as its steady variant are integrated in appropriate incompressible solvers. Value of the Schmidt number, unless stated otherwise, is $Sc_t = 0.9$. Influence of the molecular diffusion was neglected.

Employed numerical schemes are second-order accurate; implicit, second-order scheme is employed for time, `Gamma01` scheme for transport equation, while `limitedLinearV` and `limitedLinear` schemes are used for remaining terms. Transient simulations lasted 0.5 s with the Courant number limited to $Co \leq Co_{max} = 0.9$. Averaging is conducted on the final 20% of the simulation i.e., 0.1 s. Convergence criteria is set to 10^{-5} . Steady simulations adhere to the same guidelines, with the iteration limit set at 3×10^4 .

Three RANS turbulence models have been considered: $k - \epsilon$, $k - \omega$ TNT and $k - \omega$ SST. According to [14], two-equation $k - \epsilon$ model [28] produces underwhelming results, however, since it is still commonly used, it is included in the assessments. $k - \omega$ TNT model is a modification of the Wilcox's $k - \omega$ model and introduces additional constraints and new diffusion coefficients, thus eliminating the freestream dependency of the original model [29]. $k - \omega$ SST is nowadays used as a universal turbulence model mainly due to its flexibility; the model freely switches from $k - \epsilon$ in the freestream to $k - \omega$ formulation near the walls, thus, mitigating the deficiencies of both models [30]. Furthermore, the SST model has been previously used to assess the turbulent mixing [15] and is thus an excellent starting point to comparatively evaluate the adaptive mesh refinement approach.

The adaptive mesh refinement employed in this study integrates three different conditions that ultimately govern the mesh generation. Initially, allowed refinement zones are defined within the domain so that, despite the inevitable push towards further refinement near the walls, cells in the boundary layer remain intact. This serves two purposes: it ensures equivalence at the wall between innately different grids while maintaining the smallest cell spacing and hence limiting the variable time-stepping due to enforced Courant value. Turbulence kinetic energy, k , serves as an intermediary refinement condition; it is used to pre-refine a zone. Cells in which $l_k \leq k \leq u_k$ are refined in order to smoothen the transition from coarse towards finer regions of the mesh. Upper bound, u_k , is an arbitrarily large number whereas $l_k = 0.4$ for the coarsest mesh and is subsequently scaled down by a factor of 2 in each finer grid. Finally, concentration of the passive scalar, c , controls the fine refinement. Given that the results for a scalar transport problem are typically considered for $c = 0.01$, bounds for refinement are set $0.0065 \leq c \leq 0.012$. These bounds are similarly altered; each finer grid has its lower bound decreased by 5×10^{-4} and higher bound increased by 6×10^{-3} , thus widening the zone of refinement. Refinement level i.e., maximum number of allowed cell subdivisions, has been capped at 2.

2.3. Testing Environment and Methodology

The test system is an Intel Xeon E5-2690v3 based cluster. Each cluster node has 24 physical cores and 64 GB of RAM. Unless otherwise noted, 2 nodes are used exclusively with hyperthreading disabled. Scaling tests rely on the same per-node resource distribution

and simply utilize multiple nodes. Communication is facilitated by the Infiniband FDR link with a throughput of 51 Gb/s.

Assessments are made using the OpenFOAM v2012 with Infiniband enabled OpenMPI 4.0.3. The employed adaptive mesh refinement with load balancing is an adapted port of the code introduced in [2]. Custom solvers based on simpleFoam and pimpleFoam are also created. These integrate an additional transport equation and address other conflicts/issues. Furthermore, code for the $k - \omega$ TNT variant by Fürst [31] is used.

Initially, coarse, medium and fine grids have been created. Based on the GCI results, appropriate mesh has been chosen and subjected to a series of evaluations. Several aspects have been tested; difference between steady state and transient results, influence of the Schmidt number, Courant number and overall scaling for a chosen grid. Subsequently, AMR grids have been assessed with varying refinement approaches so as to determine the most appropriate setup for comparison with previous results. The initial AMR grid is a variant of the coarse grid unrefined by a factor of 1.3. The boundary layer has been kept intact.

Results pertaining to the computational efficiency are three-run averages. Reported average timestep and memory use represent the average values during the final 20% of the simulation, analogously to the previously noted averaging. Certain test cases necessitated the increase in the number of outer loops in order to ensure computational stability. These modifications are indicated.

3. Results and Discussion

3.1. Grid Convergence

A grid convergence study has been conducted for the conventional mesh generation approach. Three sets of grids have been generated, namely coarse, medium and fine. Grids have approximately 7.2×10^5 , 1.6×10^6 and 3.7×10^6 cells, respectively. A fragment of the computational domain near the jet centerline for each grid is given in Figure 2. The first cell is within the viscous sublayer for all cases. The downstream section of the computational domain in which the interaction occurs is resolved with $N_x = 198$, $N_y = 111$ and $N_z = 109$ grid cells for the fine grid. Grid convergence index (GCI) was calculated for three considered locations downstream of the jet centerline [32]. Values presented in the Table 1 support the conclusion that the attained results are in the asymptotic range of convergence i.e., $GCI_{m,f} / (r^p \times GCI_{c,m}) > 0.95$ for all cases. Consequently, fine grid has been chosen for further assessments.

Figure 2. Grids used in grid convergence study. Images are fragments of the computational domain near the jet centerline: (a) coarse grid; (b) medium grid; (c) fine grid.

Table 1. Grid details and GCI for the conventional approach. Assessment is made at three locations downstream of the jet centerline for the passive scalar c .

Mesh	Cells n	Ratio r	$x/D = 0.4$		$x/D = 1.2$		$x/D = 2.0$	
			e [%]	GCI [%]	e [%]	GCI [%]	e [%]	GCI [%]
Coarse	7.179×10^5	-	-	-	-	-	-	-
Medium	1.638×10^6	1.3	3.419	0.359	13.900	9.157	10.762	0.424
Fine	3.695×10^6	1.3	0.266	0.022	5.039	3.320	0.330	0.013

3.2. Conventional Approach

Multiple setups and simulation parameters were assessed in order to determine the appropriate settings for future AMR simulations. Steady state and transient simulations have been conducted using the $k - \omega$ TNT model. Additionally, transient simulation using the $k - \omega$ SST model has been conducted. Obtained results are compared to experimental results and RANS results calculated using the SST model and presented in [15]. Comparison is given in Figure 3.

Overall, presented results match the experimental curve slightly better than the RANS results from [15]. This is true for steady and transient results. At $z/D = 3$, downstream of the point $x/D = 1.4$, scalar predictions are consistent but slightly worse. Still, for $z/D = 4.5$, experimental trendline is better matched. Mean velocity profiles, U/U_{cross} and W/U_{cross} , show a better trend as well. Mean values of passive scalar and velocity for steady and transient cases are virtually indistinguishable. This can be partially attributed to averaging conducted in order to report the results. Further validation is conducted for the velocity component U/U_{cross} . Similarly to scalar concentration, values are assessed at different z/D sections. Results are given in Figure 4.

Based on Figure 4, velocity profile is reasonably reconstructed. There are, however, some notable discrepancies at $z/D = 1.5$, $x/D = 0.2$. Differences can be attributed to employed turbulence models and simplifications (domain simplification and missing inlet profiles). Two-dimensional plots of the scalar c at $y/D = 0$ and $z/D = 1.5$ are shown in Figure 5. The shape of the jet varies slightly for considered cases. Notably, TNT and SST model yield different inner shapes of the jet with slightly wider lower jet shear-layer obtained when using the SST model. Steady simulation delivers a moderately attenuated concentration distribution. In general, coherent structures are poorly predicted, particularly for the steady case. Transient simulations produce a distinguishable counter-rotating vortex pair. Results are as expected and similar to [15].

Several aspects of the current setup have been additionally assessed. Value of the turbulent Schmidt number, $Sc_t = 0.9$, is commonly suggested for RANS simulations [15,18,33]. Often, however, for specific turbulence models and depending on the problem, this value is modified. This aspect has therefore been considered and results for a range $0.3 \leq Sc_t \leq 1.3$ are presented in Appendix A, Figure A1. It is evident that satisfactory results are obtained for $Sc_t = 0.9$. Higher values are able to conform to peaks, but are lacking for the majority of the relevant region. Lower values on the other hand produce more sensible results but are inadequate when assessing concentration peaks. Utilized Sc_t is hence an appropriate choice and has been thus kept throughout.

Courant number of $Co_{max} = 1$ is commonly used as a strict guideline to ensure the numerical stability for explicit and semi-implicit schemes. Since implicit second-order scheme is employed, aforementioned stability requirement is not necessary. As computational efficiency is a key aspect of this study, by increasing the Courant value and consequently the time step, overall computational time might be reduced. This approach, however, must attain the same levels of accuracy i.e., physics of the problem must be resolved. Figure A2 suggests that the change in Courant value does not affect the accuracy of the result; however, in order to ensure numerical stability, number of outer loops had to be increased, $n_{loops} = 50$, which led to a significant increase in computational time, as is demonstrated by Table A1. Accordingly, initial choice of the Courant number,

$Co_{max} = 0.9$, has been kept for all cases, since for $Co_{max} = 10$ there is only a 13% reduction in computational time.

Figure 3. Comparison between the results presented in [15] and results obtained on the fine grid: (a) scalar concentration at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) where $y/D = 0$; (b) profiles of mean velocity components for $y/D = 0$ and $z/D = 1$.

Figure 4. Velocity component U/U_{cross} at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) for $y/D = 0$.

Figure 5. Two-dimensional plots of the scalar c at $y/D = 0$ (top) and $z/D = 1.5$ (bottom). (a,d) steady simulation using the $k - \omega$ TNT model; (b,e) transient simulation using the $k - \omega$ TNT model; (c,f) transient simulation using the $k - \omega$ SST model.

3.3. Adaptive Mesh Refinement

Adaptive mesh refinement enables the creation of dynamically changing grids based on specific condition(s). The methodology is not novel, yet it is not commonly utilized. This is partially due to inherent variability of the numerical grids and consequently questionable accuracy of the results. Key aspects of the AMR process is now discussed.

Typically, a scalar value or phase fraction are used to refine the mesh in the zone of interest. It is questionable whether this is the most appropriate choice, namely for

cases where physics of the problem can cause abrupt changes in mesh refinement levels. These changes contribute to the numerical diffusion and can hence impact the final result. Additionally, for more complex models, i.e., LES, refinement jumps and unrefined zones will govern which eddies are to be resolved. Still, AMR shows great potential and with proper use can be an acceptable approach even for complex LES simulations, where reduction in the computational requirements can lead to significant savings.

In accordance with the previous argument and as noted in Section 2.2, turbulence kinetic energy has been adopted in this study as an additional refinement condition that ensures slower transition between refinement levels. This effect can be clearly seen in Figure 6. It is important to note that this significantly increases the overall mesh size, but based on our findings, is an acceptable trade off.

Finally, stability of the numerical mesh should be a key consideration when using AMR. Although intuitively one could say that AMR will generate adequate grids as it is governed by the flow, issue arises when modeling the flow near the wall. The problem is twofold; when using turbulence models that rely on wall functions, excessive refinement in near-wall regions will lead to a change in first layer thickness thus altering the y^+ value and consequently will impact the calculations near the wall. Furthermore, additional refinement of boundary layer cells, which typically are the smallest cells in the domain, will lead to a notable change in time-stepping in order to maintain the CFL value. Presented problem, however, can be addressed. By limiting the regions of refinement to a zone which excludes the boundary layer, near-wall immutability can be achieved.

Unrefined preliminary grid employed for AMR in this study contained approximately 5.3×10^5 cells. The initial grid has iteratively been transformed according to methodology presented in Section 2.2. Results of the mesh refinement are presented in Figure 6.

Figure 6. Grids generated using the AMR approach. Variances in mesh refinement are a direct consequence of innate differences between turbulence models. Images are fragments of the computational domain: (a) $k - \epsilon$ model; (b) $k - \omega$ TNT model; (c) $k - \omega$ SST model.

Applicability for the presented problem is henceforth assessed. Figure 7 demonstrates the agreement with the experiment for scalar c and profiles of mean velocity components

when the mesh size is similar to the fine grid. At $z/D = 1.5$, results mostly match the numerical results from [15] and previously noted results on the fine grid. Despite comparative equivalence, all RANS cases are unable to reproduce experimental results. The $k - \epsilon$ model shows a deviation from the trend at $z/D = 3$, downstream of point $x/D = 0.8$. Although similar behavior is reported, current results are an obvious improvement. Similarly, at $z/D = 4.5$, a minor variation can be observed. $k - \omega$ SST model produced convincing results throughout, followed by the TNT model. Rationale for this can be found in profiles of mean velocity where for U/U_{cross} SST model approaches experimental results, surpassing the results on the fine grid. Results for the TNT model are similar to those on the fine grid. It is evident that the discrepancies are governed by resulting velocity fields i.e., are caused by the turbulence models.

Figure 7. Results obtained for the AMR approach for cases where mesh size is similar to the fine grid: (a) scalar concentration at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) where $y/D = 0$; (b) profiles of mean velocity components for $y/D = 0$ and $z/D = 1.5$.

Figure 8 further underscores turbulence-model-associated discrepancies for velocity component U/U_{cross} . It is interesting to note that at $z/D = 1.5$ and $z/D = 3$, the $k - \epsilon$

model seemingly surpasses other models; however, for $z/D = 4.5$ it drops off, which manifests as a variation in the scalar concentration at that section. $k - \omega$ variants, for the most part, produce similar values. Overall, results are comparable to results on conventional grids.

Figure 8. Velocity component U/U_{cross} when using AMR at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) for $y/D = 0$.

Two-dimensional plots of scalar concentration presented in Figure 9 show good agreement with the results for the fine grid (Figure 5). Results for $y/D = 0$ demonstrate a strong and amplified downstream trend. This is particularly true for $k - \omega$ variants. The SST model consistently assumes, based on values of c , broadest mixing region. For $z/D = 1.5$, distribution zones of c are slightly enlarged when compared to Figure 5. The overall accuracy is satisfactory. Inability to fully reconstruct the experiment is typical for RANS models; they tend to provide acceptable mean values but are unable to fully resolve flow specifics.

3.4. Computational Efficiency and Load Balancing

One of the major hindrances when using AMR is associated performance deterioration. Typically, a complex OpenFOAM case on a system with multiple cores (hereinafter processors) is executed in parallel and decomposed. Each of the processors is assigned a specific subdomain and data are exchanged only for shared faces. The described approach is appropriate for static, immutable grids; however, when using adaptive refinement, regions assigned to a specific processor can be, depending on the refinement methodology, significantly altered, both topologically and in terms of the overall cell count. Once a processor is assigned a significantly disproportional number of cells or is at maximum utilization, the overall computational efficiency will drop as the remaining processors are forced to wait. In order to address this, load balancing is introduced. Methodology behind the load balancing is rather simple: software tracks the disbalance between the processor loads and redistributes the mesh accordingly.

Figure 9. Two-dimensional plots of the scalar c at $y/D = 0$ (top) and $z/D = 1.5$ (bottom) for cases that employed the AMR approach: (a,d) $k - \epsilon$ model; (b,e) $k - \omega$ TNT model; (c,f) $k - \omega$ SST model.

Table 2 demonstrates the main problem of using AMR without load balancing; computational times for similarly sized grids are significantly longer (up to 200%). Cell size discrepancy for $k - \omega$ TNT and SST models when using the AMR approach is a direct consequence of the innate differences between the turbulence models; for identical setups, calculated values of scalar c and kinetic energy k govern the refinement of the primary grid. Still, despite that, the SST model has the longest computational time.

Table 2. Computational efficiency on fine grid and when using the AMR approach without load balancing.

Case	Fine $k - \omega$ TNT	AMR $k - \omega$ TNT	AMR $k - \omega$ SST
Cells, n	3,695,236	3,770,395	2,850,847
Total time, t [s]	61,183	104,631	123,465
Avg. timestep, t_{ts} [s]	1.010	2.364	2.463
RAM, m [GB]	9.754	17.108	15.080

With load balancing, computational times can be, on average, reduced by more than 45%, as is shown in Table 3. Since there is no observable change in the overall accuracy of the results, attained savings should encourage the use of the AMR. Additional information pertaining to the performance of the AMR methodology is included in the Appendix B. Furthermore, comparison of computational times on similarly sized grids does not take into consideration the main advantage of the AMR methodology; the given problem can be resolved by utilizing a less computationally demanding case (i.e., numerical grid), since refinement is conducted only in relevant zones/region of interest.

Aforementioned supposition is affirmed by results presented in Figure 10. Values of passive scalar c and velocity profile components are contrasted for three different cases with 1.6×10^6 , 2.15×10^6 and 2.85×10^6 cells, respectively. As can be seen, even though there are slight variations for the coarsest case, it is evident that the results are mostly equivalent.

Table 3. Impact of load balancing on computational efficiency when using AMR.

Case	Balancing Disabled			Balancing Enabled		
	$k - \epsilon$	TNT	SST	$k - \epsilon$	TNT	SST
Cells, n	3,743,793	3,770,395	2,850,847	3,746,180	3,770,836	2,851,503
Total time, t [s]	548,479	104,631	123,465	282,719	57,127	66,171
Avg. timestep, t_{ts} [s]	24.807	2.364	2.463	12.636	1.296	1.333
RAM, m [GB]	9.462	17.108	15.080	14.876	13.685	12.314

Figure 10. Results for different refinement steps when using AMR and $k - \omega$ SST model. Cases 1, 3 and 5 are considered. Case specifics are discussed in Section 2.2 and are noted in Appendix C: (a) concentration of passive scalar c at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) where $y/D = 0$; (b) profiles of mean velocity components for $y/D = 0$ and $z/D = 1$.

Thus far, noted AMR cases permitted refinement every 1000 iterations (timesteps) with load balancing conducted when the imbalance reaches 20%. Detailed results for all considered turbulence models and with regards to the load balancing are included in Appendix C (Tables A5–A10). Additionally, refinement frequency and imbalance level were

assessed i.e., their influence on the computational efficiency. These results are included in Tables A11–A14. Results demonstrate that with the increase in refinement frequency, computational time is also increased. This is expected, since frequent refinement steps introduce non-computational delays which lengthen the overall computation and for cases without load balancing can quickly lead to a general slowdown. Although it might seem favorable to instigate the refinement step frequently, only for exceedingly time-varying (changing) cases this is pertinent. For common problems, mesh refinement will reach an equilibrium where refinement and unrefinement will largely cancel each other out whilst the mesh will remain mostly constant. It is also interesting to note that lower imbalance limits are favorable for frequent refinements. Overall, however, imbalance limits have little influence on the computational time as is shown in Tables A13 and A14.

3.5. LES Considerations

The fundamental prerequisite for LES modeling is proper mesh resolution. Adaptive mesh refinement and inherent inconstancy of the grids make the assessment of the grid applicability impractical. This in itself is a lengthy topic outside of the scope of this paper and hence will not be further discussed here. It is important to note, however, that the predictions of coherent structures and the accuracy of results are directly related to the quality of the computational mesh (refinement) as scales smaller than the employed cell sizing cannot be resolved, rather they are modeled [34].

Numerical setup described in Section 2.2 is analogously used for the considered wall-adapting local eddy-viscosity (WALE) model [35]. WALE is adopted as it does not require damping in near-wall regions and should hence be more stable for use with AMR. limitedLinear01 1 and LUST schemes are used for c and U , respectively. Schmidt number $Sc_t = 1$ suggested in [15] is employed. Due to pressure instabilities, a constant number of outer loops, $n_{loops} = 5$, is set. At the inlets, a synthetic eddy generator, turbulentDFSEMInlet is used. Necessary values of the Reynolds stress tensor, integral length scales and velocities at the inlets are obtained from SST simulations. Refinement is conducted with regards to the geometrical constraints (at least two cell layers near the walls are immutable) and Q-criterion. This was performed so that the zones containing vortices and vortex structures are properly captured. Number of cells is approximately 5.5×10^6 . Results for noted setup are presented in Figure 11.

It is evident that the LES results are in better agreement with the experimental data. This is especially true for passive scalar at $z/D = 3$ and $z/D = 4.5$. For $z/D = 1.5$ scalar distribution is nearly perfectly reconstructed. There are some discrepancies, however, namely for U/U_{cross} downstream of $x/D = 0.5$ and W/U_{cross} near $x/D = 0$. Although it is fair to assume that this is caused by the AMR, result similarity with other LES studies indicates that this might be a problem inherent to the LES modeling or missing experimental data. These disparities require further investigation. Nevertheless, it has been established that AMR and inherent computational savings related to load balancing provide the ability to accurately and efficiently assess the JIC problem. Additionally, the isosurface of scalar $c = 0.01$ and its distribution at $y/D = 0$ calculated using WALE model and AMR are given in Figure 12.

Figure 11. Results obtained using the WALE model and AMR: (a) concentration of passive scalar c at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) where $y/D = 0$; (b) profiles of mean velocity components for $y/D = 0$ and $z/D = 1$.

Figure 12. Results obtained using the WALE model with AMR: (a) isosurface of the passive scalar $c = 0.01$; (b) distribution of scalar c at $y/D = 0$.

4. Conclusions

Adaptive mesh refinement and its applicability is a complex topic that requires extensive investigation. On the face of it, benefits of the approach are obvious and numerous; it allows for quick and accurate estimates in resource-constrained computational environments. However, as is the case with all techniques, there are some drawbacks and potential pitfalls which must be considered.

Use of AMR on non-HPC systems and systems with limited processing power as well as general use, which does not include problem parallelization, fails to uncover one of the main issues of the AMR natively included in OpenFOAM distributions which is load-associated performance deterioration. From a computational standpoint, this is a major hindrance to AMR use, namely for larger, complex and real-world problems.

This study tried to address and clarify key aspects of AMR use by analyzing a turbulent mixing problem of a jet in crossflow. Overall accuracy for a given problem is assessed and compared to experimental and previous numerical results. Additionally, load balancing, its parameters and significance are investigated and discussed.

Initially, conventional approach using RANS turbulence models to analyze turbulent mixing of a jet in crossflow was considered. Results showed good agreement with previous numerical studies. At worst, predicted scalar concentration differed by $c = 0.1$ whereas velocity components U/U_{cross} and W/U_{cross} deviated by 0.5 and 0.3. These discrepancies, however, are an improvement as they lead to better agreement with the experimental data. Expanding on this, an unrefined coarse grid was employed for a series of AMR case studies for varying turbulence models and parameters. Results were encouraging and acceptable and showed the potential of the adaptive mesh refinement; even the coarsest considered cases were satisfactory. Differences between scalar concentrations were negligible (for best cases). Subsequently, emphasis was given to load balancing and its influence on computational time. When normalized by the cell count, considered AMR cases without load balancing exhibited up to 72% longer computational times than conventionally defined problems. With the inclusion of load balancing, AMR cases completed in approximately 7% less time. Based on our findings, load balancing can reduce AMR computational times by more than 50%. Furthermore, it has been shown that AMR cases can achieve accuracy similar to conventional methodology with significantly lower cell count, thus further reducing computational time. Key observations and suggestions will now be briefly noted:

- AMR is suitable for turbulent mixing problems. Accuracy is on par with the conventional approach.
- Use for complex problems with bounded regions of interest is desirable.
- AMR approach can be employed to generate problem-specific grids as opposed to generating grids using conventional methodology.
- In order to avoid abrupt changes in mesh refinement, multi-criteria approach is suggested.
- Ensuring immutability of the boundary layer is a key factor in maintaining computational efficiency. This should also ensure CFL stability.
- Refinement frequency is the principal parameter governing the computational effort in AMR. Frequent refinement steps are typically not necessary, at least for JIC problems and RANS. Accordingly, additional computational savings can be achieved.
- Load balancing is indispensable for complex problems; however, simpler problems can also benefit from it. Due to load balancing, computational times can be halved.

Author Contributions: Conceptualization, L.K. and L.G.; methodology, A.S.; validation, A.S., M.A. and L.G.; writing—original draft preparation, A.S.; writing—review and editing, A.S., M.A. and L.G.; supervision, L.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the KLIMOD project funded by the Ministry of Environment and Energy of the Republic of Croatia and the European structural and investment funds under grant no. KK.05.1.1.02.0017.

Acknowledgments: The authors acknowledge the support of the Center of Advanced Computing and Modelling at the University of Rijeka.

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

Appendix A. Results on Conventional Grids

Figure A1. Concentration of passive scalar c at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) for different values of Sc_t where $y/D = 0$. Fine grid and $k - \omega$ TNT model.

Table A1. Performance on the fine grid for different CFL values. Assessment is made for the $k - \omega$ TNT model. Presented values are three-run averages. A total of 48 cores are used for all tests. Due to numerical instabilities, the number of outer loops had to be increased ($n_{loops} = 50$) for $Co_{max} > 0.9$.

CFL	0.9	1.5	3	5	10
Cells, n	3,695,236				
Total time, t [s]	61,183	314,339	172,683	105,603	53,854
Avg. timestep, t_{ts} [s]	1.010	8.381	9.743	9.863	9.790
RAM, m [GB]	9.754	11.190	10.345	9.937	9.556

Table A2. Scalability and performance on the fine grid for original setup. Assessment is made for the $k - \omega$ TNT model. Presented values are three-run averages.

Cores, c	12	24	48	96	192	384
Cells, n	3,695,236					
Total time, t [s]	145,500	131,312	61,183	27,941	15,671	10,513
Avg. timestep, t_{ts} [s]	2.410	2.173	1.010	0.458	0.238	0.172
RAM, m [GB]	6.116	8.011	9.754	13.770	21.966	39.331

Figure A2. Results for different CFL values on fine grid with $k - \omega$ TNT model: (a) concentration of passive scalar c at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) where $y/D = 0$; (b) profiles of mean velocity components for $y/D = 0$ and $z/D = 1$.

Appendix B. Results on AMR Generated Grids

Figure A3. Jet shape for the SST model where $c = 0.01$: (a) results on the initial grid used for AMR after 0.5 s (no refinement); (b) AMR results after 0.5 s.

Figure A4. Reynolds stress component $\overline{u'u'}/U_{cross}^2$ at $z/D = 1.5$, $z/D = 3$, $z/D = 4.5$ (top to bottom) where $y/D = 0$. AMR approach and different turbulence models.

Table A3. Performance for different CFL values when using AMR approach with grid size similar to the fine grid. Testing methodology is equal to the one defined for Table A1. Refinement frequency is set to 1000 timesteps. Load balancing is performed when the imbalance reaches 10%.

CFL	0.9	1.5	3	5	10
Cells, n	3,772,957	3,773,398	3,744,222	3,737,663	3,694,963
Total time, t [s]	56,999	286,900	150,495	79,317	32,841
Avg. timestep, t_{ts} [s]	1.291	11.124	11.497	11.794	12.443
RAM, m [GB]	13.981	15.358	14.214	14.154	13.347

Table A4. Scalability and performance for AMR approach when the grid size is similar to the fine grid. Assessment is made for the $k - \omega$ TNT model. Refinement frequency is set to 1000 timesteps. Load balancing is performed when the imbalance reaches 10%. Presented values are three-run averages.

Cores, c	12	24	48	96	192	384
Cells, n	3,771,074	3,770,465	3,772,957	3,772,712	3,775,442	3,777,577
Total time, t [s]	134,665	120,315	56,999	27,394	15,291	9449
Avg. timestep, t_{ts} [s]	3.035	2.722	1.291	0.619	0.333	0.193
RAM, m [GB]	9.036	11.781	13.981	19.823	32.623	67.008

Appendix C. Load Balancing and AMR Refinement Frequency

Table A5. AMR generated grids when using the $k - \epsilon$ model. Refinement frequency is set to 1000 timesteps. Load balancing is disabled. AMR specifics are defined in the manuscript. Presented values are three-run averages. A total of 48 cores are used for all tests. Due to numerical instabilities, the number of outer loops had to be increased ($n_{loops} = 20$).

Cells, n	1,635,190	2,131,756	2,699,904	3,256,761	3,743,793	4,171,500
Total time, t [s]	213,972	293,283	416,898	492,013	548,479	610,740
Avg. timestep, t_{ts} [s]	9.428	13.329	17.508	22.222	24.807	26.648
RAM, m [GB]	7.274	7.723	8.024	8.744	9.462	10.271

Table A6. Results for setup identical to Table A5 with load balancing enabled and performed when the imbalance reaches 20%.

Cells, n	1,674,068	2,160,666	2,682,159	3,256,502	3,746,180	4,182,035
Total time, t [s]	121,036	152,235	191,997	245,430	282,719	328,035
Avg. timestep, t_{ts} [s]	5.326	7.003	8.752	10.991	12.636	14.474
RAM, m [GB]	10.048	10.929	12.780	13.915	14.876	16.068

Table A7. AMR generated grids when using the $k - \omega$ TNT model. Refinement frequency is set to 1000 timesteps. Load balancing is disabled. AMR specifics are defined in the manuscript. Presented values are three-run averages. A total of 48 cores are used for all tests.

Cells, n	1,648,562	2,153,262	2,567,886	3,139,800	3,770,395	4,568,486
Total time, t [s]	44,703	56,000	75,137	93,349	104,631	122,059
Avg. timestep, t_{ts} [s]	1.014	1.196	1.696	2.098	2.364	2.673
RAM, m [GB]	9.736	11.674	13.106	15.321	17.108	18.757

Table A8. Results for setup identical to Table A7 with load balancing enabled and performed when the imbalance reaches 20%.

Cells, n	1,670,976	2,150,553	2,569,461	3,142,047	3,770,836	4,568,983
Total time, t [s]	23,683	29,469	37,496	46,964	57,127	69,733
Avg. timestep, t_{ts} [s]	0.538	0.639	0.850	1.076	1.296	1.486
RAM, m [GB]	9.128	10.184	11.583	12.558	13.685	16.707

Table A9. AMR generated grids when using the $k - \omega$ SST model. Refinement frequency is set to 1000 timesteps. Load balancing is disabled. AMR specifics are defined in the manuscript. Presented values are three-run averages. A total of 48 cores are used for all tests.

Cells, n	1,593,696	1,890,216	2,149,496	2,445,512	2,850,847	4,127,374
Total time, t [s]	57,857	72,595	84,600	98,321	123,465	161,944
Avg. timestep, t_{ts} [s]	1.180	1.471	1.728	2.009	2.463	3.030
RAM, m [GB]	10.093	11.268	12.305	13.363	15.080	18.380

Table A10. Results for setup identical to Table A9 with load balancing enabled and performed when the imbalance reaches 20%.

Cells, n	1,607,775	1,891,996	2,150,862	2,445,944	2,851,503	4,128,115
Total time, t [s]	31,972	39,160	42,714	50,833	66,171	89,405
Avg. timestep, t_{ts} [s]	0.663	0.807	0.872	1.110	1.333	1.689
RAM, m [GB]	9.036	9.834	10.559	11.013	12.314	14.863

Table A11. Influence of the refinement frequency on the computational time. Setup is identical to Table A7 with load balancing enabled and performed when the imbalance reaches 5%.

Frequency	10	50	100	500	1000
Cells, n	3,793,285	3,788,252	3,775,575	3,768,449	3,773,790
Total time, t [s]	147,421	91,853	65,316	58,514	57,493
Avg. timestep, t_{ts} [s]	2.595	1.562	1.438	1.315	1.306
RAM, m [GB]	16.164	15.697	14.702	15.162	13.985

Table A12. Influence of the refinement frequency on the computational time. Setup is identical to Table A7 with load balancing enabled and performed when the imbalance reaches 20%.

Frequency	10	50	100	500	1000
Cells, n	3,789,764	3,785,494	3,774,413	3,767,392	3,770,836
Total time, t [s]	149,921	105,332	68,027	58,432	57,127
Avg. timestep, t_{ts} [s]	2.679	1.909	1.520	1.311	1.296
RAM, m [GB]	16.430	15.755	15.106	14.989	13.685

Table A13. Influence of the load balancing (imbalance adjustment) on the computational time. Setup is identical to Table A7 with refinement frequency set to 10 timesteps.

Imbalance	2.5%	5%	10%	20%	40%
Cells, n	3,794,307	3,793,285	3,791,325	3,789,764	3,788,651
Total time, t [s]	128,477	147,421	142,727	149,921	146,456
Avg. timestep, t_{ts} [s]	1.225	2.595	2.483	2.679	2.605
RAM, m [GB]	15.263	16.164	16.355	16.430	16.493

Table A14. Influence of the load balancing (imbalance adjustment) on the computational time. Setup is identical to Table A7 with refinement frequency set to 1000 timesteps.

Imbalance	2.5%	5%	10%	20%	40%
Cells, n	3,777,773	3,773,790	3,772,957	3,770,836	3,770,892
Total time, t [s]	57,932	57,493	56,999	57,127	57,755
Avg. timestep, t_{ts} [s]	1.333	1.306	1.291	1.296	1.311
RAM, m [GB]	14.234	13.985	13.981	13.685	13.577

References

1. Deising, D.; Bothe, D.; Marschall, H. Direct numerical simulation of mass transfer in bubbly flows. *Comput. Fluids* **2018**, *172*, 524–537. [\[CrossRef\]](#)
2. Rettenmaier, D.; Deising, D.; Ouedraogo, Y.; Gjonaj, E.; De Gersem, H.; Bothe, D.; Tropea, C.; Marschall, H. Load balanced 2D and 3D adaptive mesh refinement in OpenFOAM. *SoftwareX* **2019**, *10*, 1–10. [\[CrossRef\]](#)
3. Maric, T.; Marschall, H.; Bothe, D. voFoam-A geometrical Volume of Fluid algorithm on arbitrary unstructured meshes with local dynamic adaptive mesh refinement using OpenFOAM. *arXiv* **2013**, arXiv:1305.3417.
4. Cai, X.; Marschall, H.; Wörner, M.; Deutschmann, O. A phase field method with adaptive mesh refinement for numerical simulation of 3D wetting processes with OpenFOAM®. In Proceedings of the 2nd International Symposium on Multiscale Multiphase Process Engineering (MMPE), DECHEMA, Hamburg, Germany, 24–27 September 2014; pp. 116–121.

5. Li, L.; Hu, D.; Liu, Y.; Wang, B.; Shi, C.; Shi, J.; Xu, C. Large eddy simulation of cavitating flows with dynamic adaptive mesh refinement using OpenFOAM. *J. Hydrodyn.* **2019**, *32*, 398–409. [\[CrossRef\]](#)
6. Wang, Z.; Li, L.; Cheng, H.; Ji, B. Numerical investigation of unsteady cloud cavitating flow around the Clark-Y hydrofoil with adaptive mesh refinement using OpenFOAM. *Ocean. Eng.* **2020**, *206*, 107349. [\[CrossRef\]](#)
7. Ismail, H.M.; Ng, H.K.; Gan, S.; Lucchini, T. *Application of Adaptive Local Mesh Refinement (ALMR) Approach for the Modeling of Reacting Biodiesel Fuel Spray Using OpenFOAM*; Technical Report, SAE Technical Paper; Society of Automotive Engineers: Warrendale, PA, USA, 2014.
8. Hindi, G.; Paladino, E.; de Oliveira, A., Jr. Effect of mesh refinement and model parameters on LES simulation of diesel sprays. *Int. J. Heat Fluid Flow* **2018**, *71*, 246–259. [\[CrossRef\]](#)
9. Lapointe, C.; Wimer, N.T.; Glusman, J.F.; Makowiecki, A.S.; Daily, J.W.; Rieker, G.B.; Hamlington, P.E. Efficient simulation of turbulent diffusion flames in OpenFOAM using adaptive mesh refinement. *Fire Saf. J.* **2020**, *111*, 102934. [\[CrossRef\]](#)
10. Lapointe, C.; Wimer, N.T.; Simons-Wellin, S.; Glusman, J.F.; Rieker, G.B.; Hamlington, P.E. Efficient Simulations of Propagating Flames and Fire Suppression Optimization Using Adaptive Mesh Refinement. *Fluids* **2021**, *6*, 323. [\[CrossRef\]](#)
11. Karlsson, J. Implementing Anisotropic Adaptive Mesh Refinement in OpenFOAM. Master's Thesis, Chalmers University of Technology, Göteborg, Sweden, 2013.
12. Joshi, S.V. Adaptive Mesh Refinement in OpenFOAM with Quantified Error Bounds and Support for Arbitrary Cell-Types. Master's Thesis, The Technical University of Munich, Munich, Germany, 2016.
13. Ribeiro, R.; Santos, L.P.; Nóbrega, J.M.; nSharma: Numerical Simulation Heterogeneity Aware Runtime Manager for OpenFOAM. In Proceedings of the International Conference on Computational Science, Wuxi, China, 11–13 June 2018; Springer: Berlin, Germany, 2018; pp. 429–443.
14. Galeazzo, F.C.C.; Donnert, G.; Habisreuther, P.; Zarzalis, N.; Valdes, R.J.; Krebs, W. Measurement and simulation of turbulent mixing in a jet in crossflow. *J. Eng. Gas Turbines Power* **2011**, *133*, 061504. [\[CrossRef\]](#)
15. Galeazzo, F.C.C.; Donnert, G.; Cárdenas, C.; Sedlmaier, J.; Habisreuther, P.; Zarzalis, N.; Beck, C.; Krebs, W. Computational modeling of turbulent mixing in a jet in crossflow. *Int. J. Heat Fluid Flow* **2013**, *41*, 55–65. [\[CrossRef\]](#)
16. Ivanova, E.M.; Noll, B.E.; Aigner, M. A numerical study on the turbulent Schmidt numbers in a jet in crossflow. *J. Eng. Gas Turbines Power* **2013**, *135*, 011505. [\[CrossRef\]](#)
17. Lefantzi, S.; Ray, J.; Arunajatesan, S.; Dechant, L. *Tuning a RANS ke Model for Jet-in-Crossflow Simulations*; Technical Report; Sandia National Lab. (SNL-CA): Livermore, CA, USA, 2014.
18. Valero, D.; Bung, D.B. Sensitivity of turbulent Schmidt number and turbulence model to simulations of jets in crossflow. *Environ. Model. Softw.* **2016**, *82*, 218–228. [\[CrossRef\]](#)
19. Su, L.; Mungal, M. Simultaneous measurements of scalar and velocity field evolution in turbulent crossflowing jets. *J. Fluid Mech.* **2004**, *513*, 1–45. [\[CrossRef\]](#)
20. Denev, J.A.; Fröhlich, J.; Bockhorn, H. Large eddy simulation of a swirling transverse jet into a crossflow with investigation of scalar transport. *Phys. Fluids* **2009**, *21*, 015101. [\[CrossRef\]](#)
21. Salewski, M.; Stankovic, D.; Fuchs, L. Mixing in circular and non-circular jets in crossflow. *Flow Turbul. Combust.* **2008**, *80*, 255–283. [\[CrossRef\]](#)
22. Cintolesi, C.; Petronio, A.; Armenio, V. Turbulent structures of buoyant jet in cross-flow studied through large-eddy simulation. *Environ. Fluid Mech.* **2019**, *19*, 401–433. [\[CrossRef\]](#)
23. Ryan, K.J.; Bodart, J.; Folkersma, M.; Elkins, C.J.; Eaton, J.K. Turbulent scalar mixing in a skewed jet in crossflow: Experiments and modeling. *Flow Turbul. Combust.* **2017**, *98*, 781–801. [\[CrossRef\]](#)
24. Milani, P.M.; Ling, J.; Eaton, J.K. Turbulent scalar flux in inclined jets in crossflow: Counter gradient transport and deep learning modelling. *J. Fluid Mech.* **2021**, *906*, A27.1–A27.30. [\[CrossRef\]](#)
25. OpenCFD Ltd. OpenFOAM v2012 User Guide. 2012. Available online: <https://www.openfoam.com> (accessed on 1 December 2021).
26. Tu, J.; Yeoh, G.H.; Liu, C. *Computational Fluid Dynamics: A Practical Approach*; Butterworth-Heinemann: Oxford, UK, 2018.
27. Menter, F.; Ferreira, J.C.; Esch, T.; Konno, B.; Germany, A. The SST turbulence model with improved wall treatment for heat transfer predictions in gas turbines. In Proceedings of the International Gas Turbine Congress. Gas Turbine Society of Japan (GTSJ), Tokyo, Japan, 2–7 November 2003; Volume 1, pp. 2–7.
28. Launder, B.E.; Spalding, D.B. The numerical computation of turbulent flows. In *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion*; Pergamon: New York, NY, USA, 1983; pp. 96–116.
29. Kok, J.C. Resolving the dependence on freestream values for the k-turbulence model. *AIAA J.* **2000**, *38*, 1292–1295. [\[CrossRef\]](#)
30. Menter, F.R. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA J.* **1994**, *32*, 1598–1605. [\[CrossRef\]](#)
31. Fürst, J. Turbulence Models. 2021. Available online: <https://github.com/furstj/myTurbulenceModels/wiki/User-guide> (accessed on 1 December 2021).
32. Roache, P.J. *Verification and Validation in Computational Science and Engineering*; Hermosa: Albuquerque, NM, USA, 1998; Volume 895.
33. Yimer, I.; Campbell, I.; Jiang, L.Y. Estimation of the turbulent Schmidt number from experimental profiles of axial velocity and concentration for high-Reynolds-number jet flows. *Can. Aeronaut. Space J.* **2002**, *48*, 195–200. [\[CrossRef\]](#)

-
34. Grbčić, L.; Kranjčević, L.; Lučin, I.; Sikirica, A. Large Eddy Simulation of turbulent fluid mixing in double-tee junctions. *Ain Shams Eng. J.* **2021**, *12*, 789–797. [[CrossRef](#)]
 35. Nicoud, F.; Ducros, F. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow Turbul. Combust.* **1999**, *62*, 183–200. [[CrossRef](#)]