Nuclear Engineering and Technology xxx (xxxx) xxx

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

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

Original Article

Thermal stratification prediction in reactor system based on CFD simulations accelerated by a data-driven coarse-grid turbulence model

Zijing Liu [a,b], Pengcheng Zhao [a,*], Badea Aurelian Florin [b], Xu Cheng [b]

[a] School of Nuclear Science and Technology, University of South China, Hengyang, 421001, China
[b] Institute of Applied Thermofluidics, Karlsruhe Institute of Technology, Karlsruhe, 76131, Germany

ARTICLE INFO

Keywords:
Thermal stratification
Data-driven turbulence model
Machine learning
OpenFOAM
TensorFlow

ABSTRACT

Thermal stratification in large enclosures is an integral phenomenon to nuclear reactor system safety. Currently, the effective model for thermal stratification utilizes a multi-scale method that integrates 1-D system-level and 3-D CFD code, which offers thermal stratification details while supplying system-level data across various domains. Nonetheless, harmonizing two codes that operate on different spatial and temporal scales presents a significant challenge, with high-resolution CFD simulations requiring substantial computational resources. This study introduced a data-driven coarse-grid turbulence model based on local flow characteristics at a significantly coarser scale, targeting improved efficiency and accuracy in reactor safety analysis concerning thermal stratification. A machine learning framework has been introduced to expedite the RANS-solving process by coupling OpenFOAM and TensorFlow, which entails training a deep neural network with fine-grid CFD-generated data to predict turbulent eddy viscosity. The feasibility of the developed data-driven turbulence model was proven through the SUPERCAVNA experimental facility problem validation.

Nomenclature

ML	Machine Learning	ANN	Artificial Neural Network
OLHS	Orthogonal Latin Hypercube Sampling	FG	Fine Grid
DNN	Deep Neural Network	CG	Coarse Grid
FNN	Feedforward Neural Network	DD	Data-Driven
CNN	Convolutional Neural Network	kNN	k-Nearest-Neighbor
RANS	Reynolds-averaged N-S equation	LSTM	Long Short Term Memory

1. Introduction

Thermal stratification in expansive pools or enclosures represents a critical phenomenon essential to nuclear reactor safety [1]. In advanced light-water reactors, thermal stratification can occur within the passive containment, thereby obstructing natural circulation [2]. In pool-type reactors, thermal stratification can cause reactor physics and thermal-hydraulic instabilities, leading to thermal fatigue that could damage the reactor vessel and in-vessel components, and further impede natural circulation [1,3,4]. Accurate prediction of pool temperature and density distribution is imperative for these reactor systems' safety analyses and design optimization.

Generally, methods for analyzing thermal stratification can be categorized into two main groups. The first category encompasses system-level methods offering rapid yet approximate computations, whereas the second category includes CFD methods providing high-resolution computations at considerable computational cost [1]. System-level codes utilize lumped-volume-based 0-D models or coarse 1-D models for thermal mixing, which struggle to accurately calculate sudden temperature changes during transients due to complex 3-D effects or buoyancy phenomena, particularly at the start of natural convection [5]. The feasibility of CFD modeling for the 3-D effects of thermal stratification has been confirmed when a fine grid (FG) is employed to resolve the substructure. The choice of the turbulence model and the spatial mesh configuration around the thermal stratification interface in the direction of gravity are crucial for CFD modeling of thermal stratification. Moreover, the flow pattern within a large enclosure influences the thermal stratification interface [6]. To effectively model thermal stratification phenomena, a multi-scale method has been proposed. This approach integrates 1-D system-level code and

* Corresponding author.
E-mail address: pengcheng.zhao@usc.edu.cn (P. Zhao).

https://doi.org/10.1016/j.net.2024.10.050
Received 14 April 2024; Received in revised form 20 October 2024; Accepted 28 October 2024
Available online 3 November 2024

3-D CFD code, offering detailed thermal stratification insights while also supplying system-level data in other areas. However, coupling two codes operating at different spatial and temporal scales that can differ by orders of magnitude remains a very challenging task, and the high-resolution CFD simulation is computationally intensive [7].

Recent advances have explored super-resolution reconstruction as a means to accelerate CFD computations. Fukami et al. proposed using convolutional neural networks (CNNs) to enhance the spatial resolution of low-resolution CFD data. This approach allows the reconstruction of high-resolution flow fields from coarse input data, significantly reducing computational costs while maintaining accuracy [8]. However, this method requires extensive training data and may struggle with generalization when applied to conditions different from the training dataset. Another promising approach is hybrid computing, which integrates machine learning with traditional CFD methods to accelerate the overall simulation process. Jeon et al. developed a method that combines the finite volume method with neural networks to reduce computation time for unsteady CFD simulations while preserving accuracy [9]. This method, however, can be complex to implement and may require significant tuning depending on the specific flow conditions. In the context of incompressible flow simulations, Ajuria Illarramendi et al. introduced a pressure projection method that combines deep learning with traditional CFD solvers to accelerate the pressure correction step. This method aims to reduce the computational load associated with solving the pressure-velocity coupling, which is often the bottleneck in incompressible flow simulations [10]. While this approach can significantly speed up computations, its effectiveness depends on the quality of the training data, and it may face challenges in robustness when applied to scenarios outside the training range.

Data-driven turbulence models represent a growing area of research focused on reducing the computational burden of CFD simulations by leveraging machine learning techniques to predict turbulence parameters, such as turbulent eddy viscosity, from coarse grid data. Tracey et al. assessed the feasibility of creating a Feed-forward Neural Network (FNN) based turbulence model designed to replicate the Spalart-Allmaras model. The trained FNN model demonstrated proficiency in simulating a broad range of flow conditions, from 2D flat plate boundary layers to 3D transonic wings, even successfully handling flow scenarios that were not encountered during training [11,12]. Sun et al. and Zhu et al. further explored the use of FNNs as surrogate models for the Spalart-Allmaras turbulence model. They employed the optimal brain surgeon technique to identify the relevance of input features, effectively performing a sensitivity analysis on the input neurons' weights [13–15]. This approach allows the neural network to focus on the most impactful features, improving model efficiency and accuracy. Chang et al. introduced an empirical strategy using feature coverage mapping rooted in t-distributed Stochastic Neighbor Embedding (t-SNE) to quantify data coverage in machine learning-based closures. Through the application to a backward-facing step flow, the study demonstrated that neural networks could decipher inherent correlations in fluid data and be integrated into the RANS (Reynolds-Averaged Navier-Stokes) solving mechanism to forecast flow attributes without compromising numerical stability [16]. In their approach, training data derived exclusively from RANS simulations utilizing the k-ε model and the machine learning model aimed to estimate Reynolds stress, sourced from the spatial derivatives of input velocity fields. During the conservation equation-solving process, closure relations were continually extracted from the machine learning model. Zhu et al. developed a turbulence eddy viscosity model based on FNNs to address 3D thermal stratification issues using a coarse-grid (CG) CFD code. The authors also tackled the issue of imbalanced training datasets, suggesting over-sampling and under-sampling techniques as potential solutions [17]. Iskhakov et al. employed an invariant neural network architecture to model Reynolds stress and turbulent heat flux in forced convection flows, making it suitable for simulating flow in reactor downcomers. Their data-driven model, validated under various fluid conditions relevant to advanced reactors, showed promise without needing to adjust the turbulent Prandtl number [18].

Maulik et al. introduced a turbulent eddy-viscosity surrogate modeling framework for RANS simulations. By training neural networks to predict steady-state turbulent eddy viscosity fields, this model significantly reduced computational time while maintaining high accuracy. Their framework, tested on various turbulence closure models and validated on a 2D backward-facing step flow problem, demonstrated robust performance across different grid refinements and geometries [19]. Liu et al. proposed a data-driven coarse mesh turbulence model based on convolutional recurrent neural network (CNN) and long short term memory (LSTM) model to predict turbulent eddy viscosity distribution under transient conditions for transient analysis of thermal mixing and stratification in sodium-cooled fast reactor [20]. In this approach, the CNN functions as an encoder-decoder, compressing high-dimensional turbulent eddy viscosity data into a lower-dimensional feature space and then reconstruction. This reduces the computational burden on the LSTM while preserving key physical features. The LSTM captures the temporal evolution of turbulent eddy viscosity. However, the CNN-LSTM model faces challenges in prediction accuracy and generalization. The CNN's use of convolutional kernels often results in overly smooth predictions, limiting its ability to capture sharp changes in turbulent eddy viscosity. Additionally, the encoding and decoding process can lead to the loss of important local flow features. Meanwhile, the LSTM's memory effect can accumulate errors over long-term predictions, and its tendency to smooth transitions can introduce significant inaccuracies when dealing with the rapid changes typical of turbulent flows.

This research presents a data-driven coarse-grid (CG) turbulence model for thermal stratification designed to enhance the efficiency and precision of reactor system safety analysis. The 3-D fluid conservation equations are solved using a data-driven (DD) turbulence model, which is based on local flow characteristics at a significantly coarser scale. This approach ensures that the mesh configuration in the CFD code is consistently aligned with the 1-D system-level code. To expedite the RANSsolving process, a machine learning (ML) framework was developed, wherein an Artificial Neural Network (ANN) was trained using substantial fine-grid CFD-generated data to predict turbulent eddy viscosity. OpenFOAM was selected as the simulation framework, with TensorFlow integrated as the ML framework, and the TensorFlow C API was utilized to couple OpenFOAM with TensorFlow, enabling the implementation of the DD turbulence model. The coupling is implemented via a surrogate model prediction task for the turbulence eddy viscosity in the context of the SUPERCAVNA experimental facility [21] problem. Inspired by Liu's approach of using CNN-LSTM to develop a turbulent eddy viscosity surrogate model [20], we further enhanced prediction accuracy, generalization ability, and training efficiency by dividing the surrogate model into steady-state and transient. The steady-state surrogate model was efficiently trained using deep neural networks (DNN) and high-fidelity CFD data with various initial and boundary conditions, enabling rapid and accurate prediction of turbulent eddy viscosity with better generalization ability. For the transient surrogate model, an improved LSTM architecture, accelerated by the CUDA Deep Neural Network library (CuDNN), was adopted. This architecture incorporates a Self-Attention mechanism and Bi-directional LSTM (Bi-LSTM). The Self-Attention mechanism mitigates prediction bias caused by residual memory effects in LSTM. At the same time, Bi-LSTM enhances the model's ability to capture sharp changes in turbulent eddy viscosity by processing both forward and backward information flows simultaneously. This paper focuses on the framework of the data-driven turbulence model and the development of the steady-state turbulent eddy viscosity surrogate model.

2. Methods

This section summarizes the theoretical basis and framework of the

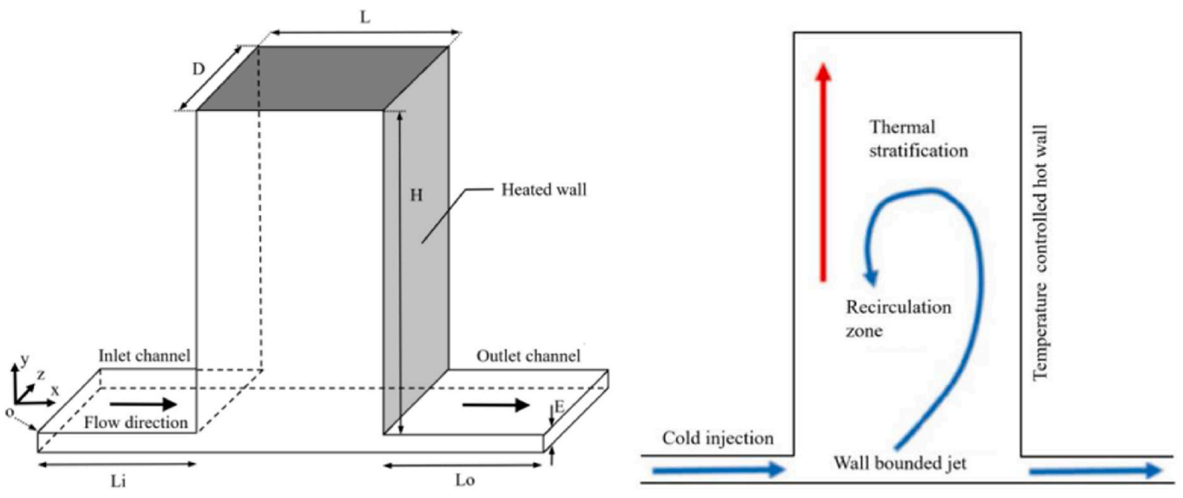

Fig. 1. SUPERCAVNA test section and experimental representation.

Table 1
Geometry parameters of the test section in SUPERCAVNA.

Symbol	Parameter	Value
H	Height of cavity	3.2 m
L	Length of cavity	1.6 m
D	Depth of cavity	0.8 m
E	Thickness of channel	0.03 m
Li	Length of inlet channel	1.52 m
Lo	Length of outlet channel	1.52 m

Table 2
SUPERCAVNA flow conditions.

Symbol	Parameter	Value
Tc	Maximum temperature in the cavity	294.4 °C
T_O	Global initial temperature	250 °C
V_{fi}	Cavity mean inlet velocity	0.69 m/s
T_{fi}	Cavity mean inlet temperature	250 °C
T_h	Heated wall mean temperature	303.1 °C

Table 3
Physical properties of sodium.

Parameter	Correlation
Density, ρ, kg/m^3	P = 1015.03–0.23393 T - 0.305×10^{-5}T^2
Thermal conductivity, λ, W/m/K	λ = 110–0.0648 T + 1.16 × 10^{-5}T^2
Dynamic viscosity, μ, Pa × s	μ = 8.85141 × 10^{-4} - 9.4 × 10^{-7}T
Thermal expansion coefficient, β, 1/K	β = 2.523 × 10^{-4} (T-523.15)
Specific heat capacity, c$_p$, J/kg/K	C$_p$ = 1.63624845 × 10^3 - 0.8363338 T + 4.64113 × 10^{-4}T^2

DD turbulence model for thermal stratification analysis in the reactor system. In subsection 2.1 we describe the RANS equation and the principles of computational speed-up and CG setup in CFD code. In subsection 2.2 we discuss the key procedures and components of construction of the DD turbulence model.

2.1. RANS equation

The Reynolds-averaged Navier-Stokes (RANS) equations represent a time-averaged variant of the Navier-Stokes motion equations for fluid flow. Stemming from the Reynolds decomposition principle, these equations partition the instantaneous quantity into its time-averaged and fluctuating segments. These equations are utilized to characterize the behavior of turbulent flows [22]. The RANS equation is presented in the following form.

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{1}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = \overline{f}_i - \frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j} \tag{2}$$

Where $\overline{u}_i$ represents the filtered/averaged velocity, $\overline{p}$ denotes the filtered/averaged pressure, and the overbar $^-$ signifies the filtering/averaging operation.

Reynolds stress $-\overline{u_i' u_j'}$ represents a nonlinear term that necessitates an additional model to close the RANS equation. Most methods address this tensor with an explicit model, employing additional algebraic or differential equations. According to Boussinesq's turbulent eddy viscosity theory, the mathematical relationship between Reynolds stress and the strain rate is as follows:

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{3}$$

Where ν_t is turbulent eddy viscosity, $u_i^{'}$ is the fluctuating component of velocity, $k = \frac{1}{2}\overline{u_i' u_i'}$ is turbulence kinetic energy, δ_{ij} is Kronecker delta.

This study aims to formulate a surrogate model for turbulent eddy viscosity, thereby eliminating the requirement for solving supplementary closure equations. In the k-ε turbulence framework, $\nu_t\ (k,\varepsilon)$ characterizes the turbulent eddy viscosity as a function of kinetic energy and dissipation rate. Conversely, within the DD turbulence framework, the turbulent eddy viscosity is expressed in terms of local system variables, denoted as $\nu_t\ (V,\ T,\ p,\ \ldots)$. By excluding k and ε from the conservation equation, the computationally demanding challenges inherent in traditional CFD can be tackled using a coarser grid setup.

2.2. CFD simulation of thermal stratification

In this study, we demonstrate the applicability of the proposed RANS simulation with an embedded DD turbulence model in a case study of thermal stratification. The CEA's SUPERCAVNA facility has served as the experimental site for the analysis of sodium flow and thermal stratification interaction. The facility includes a rectangular cavity with heated sidewalls. The flow is driven by a cold jet located at the cavity's base, restricted by the wall. Experimental data on temperature distribution inside the cavity are available for both steady-state and transient flow scenarios. Fig. 1 depicts the 3-D computational domain of the

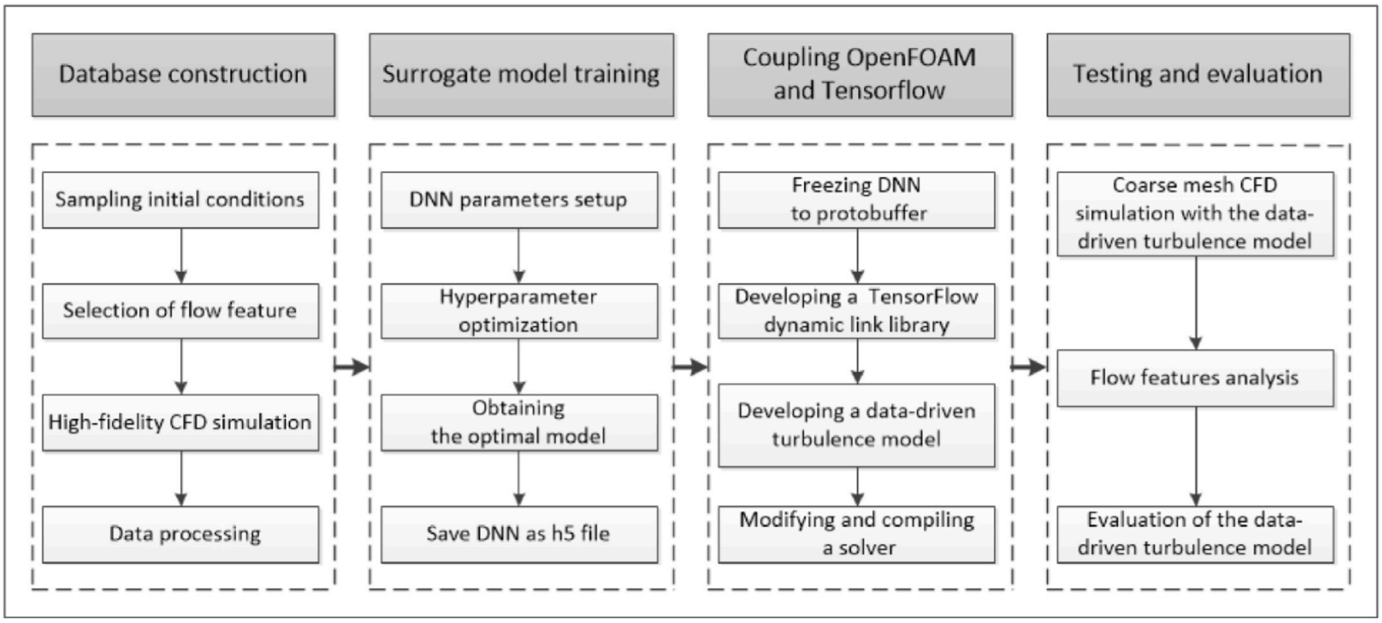
Database construction
Sampling initial conditions
Selection of flow feature
High-fidelity CFD simulation
Data processing
Surrogate model training
DNN parameters setup
Hyperparameter optimization
Obtaining the optimal model
Save DNN as h5 file
Coupling OpenFOAM and Tensorflow
Freezing DNN to protobuffer
Developing a TensorFlow dynamic link library
Developing a data-driven turbulence model
Modifying and compiling a solver
Testing and evaluation
Coarse mesh CFD simulation with the data-driven turbulence model
Flow features analysis
Evaluation of the data-driven turbulence model

Fig. 2. The framework of the data-driven turbulence model.

SUPERCAVNA test section [21]. The geometry parameters of the test section in SUPERCAVNA facility are shown in Table 1. The flow conditions are summarized in Table 2.

Simulations of the SUPERCAVNA test section are conducted using the open-source CFD software, OpenFOAM. It serves as a reference tool for estimating the accuracy of flow field predictions through CFD and generating data for the development of a DD turbulence model.

In the 3-D steady-state modeling, the physical properties of sodium, including density ρ, thermal conductivity λ, dynamic viscosity μ, thermal expansion coefficient β and Specific heat capacity c_p were utilized according to the correlations listed in Table 3 [23]. A mesh comprising approximately 1,163,200 trimmed hexahedral cells was generated using blockMesh. A pressure outlet was implemented at the outlet channel of the cavity, with all walls assumed to be adiabatic, except for the heating wall. A constant turbulent Prandtl number, Prt = 0.9 [24], was utilized. The buoyantFoam heat transfer solver was employed for flow and energy equations, with the standard k-ε model, utilized to depict system turbulence. This model was employed alongside the thermal-stratification model to account for the buoyancy production of dissipation.

BuoyantFoam is a pressure-based solver designed for steady-state and transient simulations of compressible and incompressible flows. It handles laminar and turbulent, single-phase flows with temperature and density variations. In incompressible flows modeling, the solver use the Boussinesq approximation, which simplifies the computations related to buoyancy by linearly relating the density changes in the fluid to temperature changes, based on a reference temperature. The solver is particularly accurate and efficient when the changes in density compared to a reference density are small. The solver uses the SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm for pressure-momentum coupling, augmented by under-relaxation techniques to enhance convergence. It supports Multiple Reference Frames (MRF) and porosity modeling and allows easy integration of passive scalar transport equations and source terms.

The standard k-ε turbulence model strikes a balance between precision and robustness. It has demonstrated reasonable accuracy in past simulations, especially those involving buoyant jets and the erosion of stratified layers [25]. In the SUPERCAVNA facility, the standard k-ε model was used to simulate thermal stratification, and the results showed good agreement with experimental measurements. This indicates that the standard k-ε model can provide reasonable predictions in specific applications [18]. Additionally, non-linear eddy viscosity modeling might further improve the quality of the simulation. For transient scenarios, high-fidelity CFD simulations utilizing unsteady Reynolds-Averaged Navier-Stokes (uRANS) were necessary for the development of DD turbulence models.

2.3. *Framework of data-driven turbulence model*

Due to its nonparametric modeling nature, a DNN allows for the adaptive assimilation of very complex data. Based on DNN techniques, a surrogate model of turbulent eddy viscosity can be developed and integrated into OpenFOAM by incorporating the data-driven (DD) closure into the physics-based solving process, i.e., the proposed framework of the DD turbulence model. This framework builds upon the method proposed by Maulik et al., who focused on constructing a surrogate model for turbulent eddy viscosity using flow field characteristics for a 2D backward-facing step flow problem. This research extends this approach to 3D natural convection thermal stratification problems. Specifically, we construct the surrogate model for turbulent eddy viscosity in the standard k-ε two-equation model, utilizing both flow and temperature field characteristics. To further enhance the model's applicability, the k-Nearest Neighbor (kNN) algorithm has been employed to transform high-fidelity data, enabling the turbulent eddy viscosity surrogate model to be applied to coarse grid CFD calculations. This method ensures that the surrogate model maintains high accuracy even at a coarser setup, thereby significantly enhancing computational efficiency. The key procedures and components of the framework, including database construction, surrogate model training, coupling OpenFOAM and TensorFlow software, testing, and evaluation, are discussed in detail. Fig. 2 presents the framework of the DD turbulence model.

2.3.1. *Database construction*

The first step in developing a DD turbulence model is to build a database for training and validation. This includes sampling inlet and initial conditions, selecting flow features, performing high-fidelity CFD simulations, and processing data.

For the generalization ability of the trained surrogate model of turbulent eddy viscosity ν_t, the database should contain data for a variety of

Table 4
Inlet and initial conditions of SUPERCAVNA for training cases.

Training case	V [m/s]	T [K]	k	ε	Re	Gr
0 (baseline)	0.69	523.15	1.79E-03	1.55E-04	2.38 EE+06	2.50 E+12
1	0.42	520.89	6.72E-04	3.58E-05	1.45 E+06	2.59 E+12
2	0.45	484.37	7.60E-04	4.31E-05	1.43 E+06	3.81 E+12
3	0.49	534.14	8.86E-04	5.41E-05	1.71 E+06	2.03 E+12
4	0.54	499.38	1.10E-03	7.53E-05	1.78 E+06	3.36 E+12
5	0.57	532.74	1.23E-03	8.82E-05	2.01 E+06	2.10 E+12
6	0.58	524.83	1.27E-03	9.29E-05	2.01 E+06	2.44 E+12
7	0.62	458.33	1.43E-03	1.11E-04	1.87 E+06	4.44 E+12
8	0.67	453.37	1.68E-03	1.42E-04	2.02 E+06	4.55 E+12
9	0.72	473.34	1.95E-03	1.76E-04	2.25 E+06	4.10 E+12
10	0.75	481.01	2.13E-03	2.02E-04	2.39 E+06	3.90 E+12
11	0.78	552.63	2.27E-03	2.22E-04	2.86 E+06	1.12 E+12
12	0.81	463.79	2.47E-03	2.52E-04	2.49 E+06	4.32 E+12
13	0.83	495.94	2.58E-03	2.69E-04	2.70 E+06	3.47 E+12
14	0.89	512.46	3.00E-03	3.38E-04	3.01 E+06	2.91 E+12
15	0.92	506.78	3.16E-03	3.65E-04	3.06 E+06	3.11 E+12
16	0.99	539.53	3.69E-03	4.61E-04	3.54 E+06	1.78 E+12
17	0.92	541.79	3.19E-03	3.70E-04	3.31 E+06	1.68 E+12

operating conditions and the range of initial conditions should be wide. We choose to keep the temperature of the heating wall as a fixed value, and extract 22 groups of data from the velocity V and temperature T of the inlet channel in each range [V: 0.4–1] m/s, [T: 453.15–553.15] K by orthogonal Latin hypercube sampling (OLHS) method. The OLHS allows for a random, uniform sampling of data points that represent almost the entire design area. For given values of V and T, turbulent energy k and dissipation rate ε, Reynolds number Re and Grashof number Gr are calculated by Eqs. (4)–(7).

$$k = \frac{3}{2}(VI)^2 \tag{4}$$

$$\varepsilon = \frac{C_\mu^{0.75} k^{1.5}}{l} \tag{5}$$

Here, V represents the mean flow velocity, while I denotes the turbulence intensity. The C_μ refers to a turbulence model constant, typically assigned a value of 0.09, and l signifies the turbulent length scale.

$$Re = \frac{\rho U L}{\mu} \tag{6}$$

ρ denotes the fluid density, μ represents the dynamic viscosity, while U and L correspond to the characteristic velocity and characteristic length of the flow field, respectively.

$$Gr = \frac{g\beta\Delta T L^3}{\gamma^2} \tag{7}$$

Here, g represents the gravitational acceleration due to Earth, β denotes the coefficient of volume expansion, L signifies the vertical length, γ is the kinematic viscosity, and ΔT represents the temperature differential between the surface and the bulk medium.

The 17 sets of initial conditions by OLHS and 1 set of baseline conditions constitute the training cases, which are shown in Table 4. The remaining 4 sets of initial conditions by OLHS constitute the test cases, as shown in Table 7. Based on the data in OpenFOAM simulations of SUPERCAVNA with FG were conducted to generate the accurate data sets of the database.

One of the important parts of the ML algorithm is the selection of input and output features. The objective of this study is to establish a CG closure for isotropic turbulent eddy viscosity, aimed at supporting steady-state CFD simulations of reactor systems with a range of inlet and boundary conditions. The ideal DNN model should enable a field-to-field mapping that can be smoothly integrated into OpenFOAM. For this research, we have adopted a field-to-field mapping method that uses local flow features as input. The input and output features identify the region in CFD simulation domain space through mesh-centered coordinates and the initial conditions. Thus, 3 velocity components (*Vx, Vy, Vz*), and temperature T from the initial phase of CFD simulation, as well as the mesh-centered coordinates (*Cx, Cy, Cz*), were selected as input features. The turbulent eddy viscosity ν_t from the steady-state of CFD simulation was selected as the output feature. The buoyantFoam heat transfer solver was employed to resolve the RANS equations. The pressure effects are indirectly captured through the velocity field due to the inherent coupling in the buoyantFoam solver. In thermal stratification phenomena, buoyancy effects resulting from temperature gradients are the dominant forces driving the flow. These effects are primarily captured by the temperature and velocity fields. Since pressure variations in such scenarios are often secondary to the thermal buoyancy effects, the pressure term can be omitted from the training input features. This omission helps to reduce the computational complexity in surrogate modeling and improve training efficiency.

Data processing is an important part of building ML databases, mainly including data transformation and scaling. High-resolution nonuniform mesh-defined physical quantity fields are unsuitable for training the proposed ML-based model due to 2 main challenges. Firstly, the unequal data weighting from nonuniform mesh grids complicates neural network training, which typically attributes equal importance to each data point. Secondly, the high-resolution configuration clashes with the aim of establishing a consistent relationship for a CG 3-D CFD

Table 5
KS test results for different coarse grid numbers.

Dataset	Grid number	T KS statistic	T p-value	Vz KS statistic	Vz p-value	ν_t KS statistic	ν_t p-value
Channel-1	68,896	0.027972	0.828674	0.108891	0.000014	0.177822	0.030774
Channel-2	137,792	0.014985	0.999875	0.100899	0.000074	0.076472	0.341717
Channel-3	275,584	0.008697	0.999945	0.086723	0.001277	0.041958	0.412983
Channel-4	551,168	0.005994	1.000000	0.079920	0.003332	0.034965	0.573537
Cavity-1	68,896	0.011554	0.983459	0.066922	0.011129	0.113892	0.014489
Cavity-2	137,792	0.009972	0.995139	0.055775	0.046159	0.050796	0.071435
Cavity-3	275,584	0.009184	0.998069	0.044538	0.195781	0.038835	0.252781
Cavity-4	551,168	0.008279	0.999385	0.037824	0.309272	0.031175	0.512714

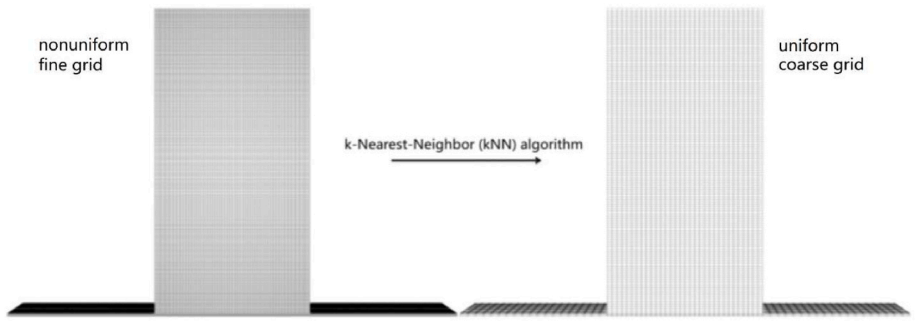

Fig. 3. Conversion between nonuniform fine grid and uniform coarse grid.

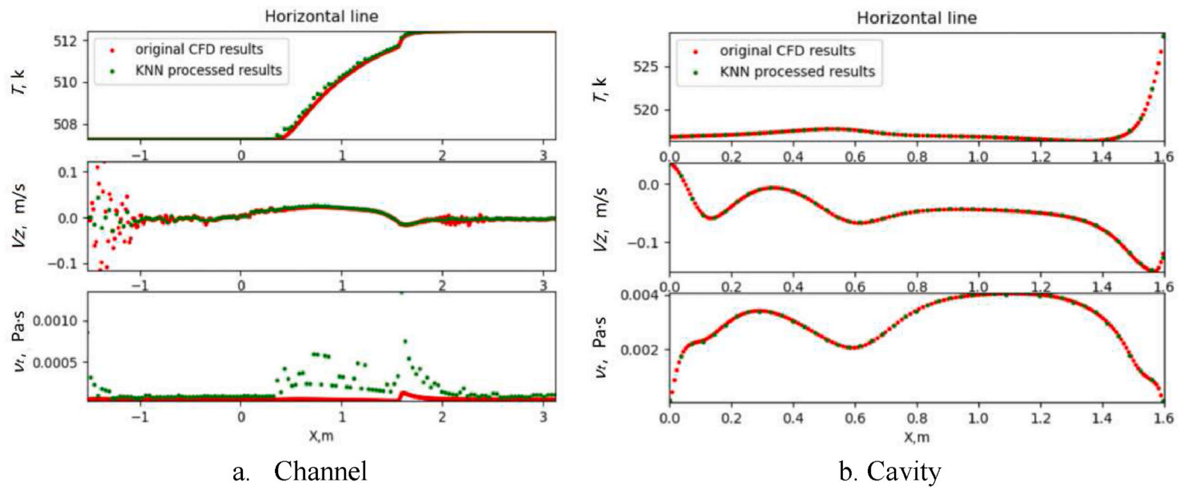

Fig. 4. Flow features comparison of original CFD solution and kNN algorithm b) Surrogate model training.

Table 6
Deep neural network parameters.

Parameter	Value
Hidden layer number	6
Neurons number	80
Batch_size	1024
Epochs number	500
Optimizer	Adam
Learning rate	0.001
Validation split	0.1
Activation function	ReLU
Loss function	mean squared error

Table 7
Comparison of calculation time of 2 turbulence models.

Parameters	Standard k-ε model	Data-driven turbulence model
Mesh quantity	1,163,200	68,896
CPU type	Intel Core-i7	Intel Core-i7
Iteration number	200	25
Calculation time	896 core-hours	2 core-hours

code integrated with the system code. To address these challenges, the k-Nearest-Neighbor (kNN) algorithm from the scikit-learn ML library [26] is employed to translate the original CFD outcomes into uniform CG data. During this transformation, for a specific point A in the uniform CG structure, the kNN algorithm identifies a set number of nearest points from the nonuniform FG to point A. Subsequently, it forecasts point A's value using a distance-weighted average derived from these neighboring points' values. Fig. 3 shows the transition from a nonuniform FG to a uniform CG.

As preparation for the DNN model training, the local flow features (velocity components *Vx*, *Vy*, *Vz*, temperature *T*, pressure *p*, turbulent eddy viscosity ν_t) of SUPERCAVNA domain with 1,163,200 cells were converted to the CG uniform data with 68,896 cells by using a kNN algorithm. The CG uniform data, comprising both input and output features, was normalized to have a unit mean and zero variance for every flow feature, facilitating smoother training.

We verified our data preprocessing approach's effectiveness by conducting qualitative and quantitative comparisons between the original CFD outputs and the processed kNN results. To analyze the consistency between the transformed coarse grid data and the fine grid data, we adopted the Kolmogorov-Smirnov (KS) test, the results of the KS test are presented in Table 5. The KS statistic is the maximum absolute difference between the two cumulative distribution functions (CDFs). The smaller the KS statistic, the more similar the distributions of the two samples. The p-value is calculated based on the KS statistic, representing the probability of observing the current or more extreme KS statistic under the null hypothesis. The larger the p-value, the more substantial the evidence supporting the null hypothesis, indicating that the two sample distributions may be the same.

Table 5 shows that a higher coarse grid number results in greater consistency with the channel and cavity fine grid data. This is evidenced by the lower KS Statistics and higher p-values in datasets with higher grid number. Increasing the grid number improves the detail and accuracy of data transformation by the kNN algorithm, leading to distributions that are more aligned with the high-resolution channel and cavity fine grid data.

Fig. 4 illustrates CG uniform data with 68,896 cells of 3 pivotal physical parameters: the velocity component *Vz*, temperature *T*, and

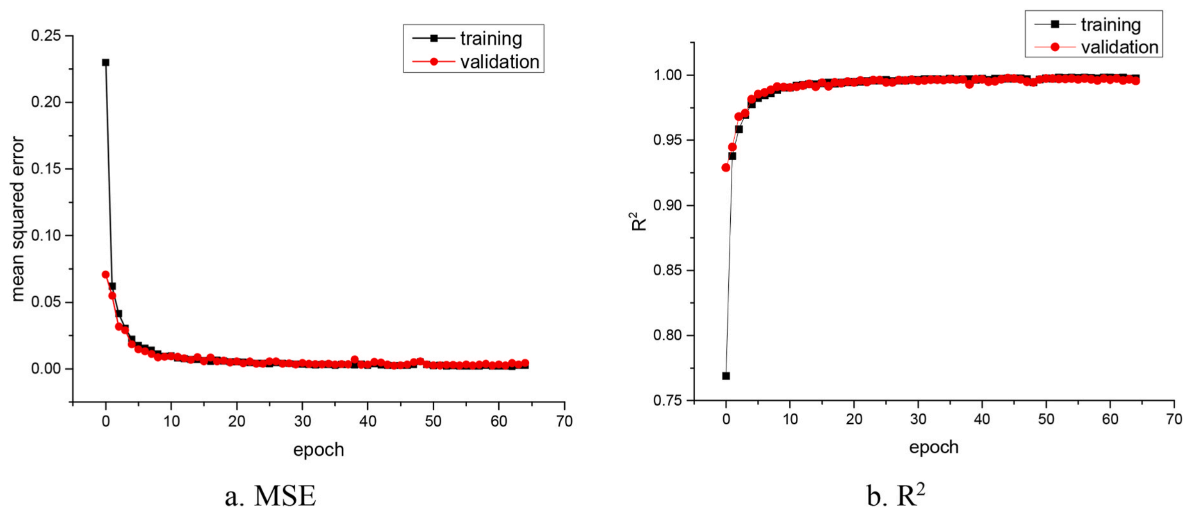

Fig. 5. Convergence while training the DNN model.

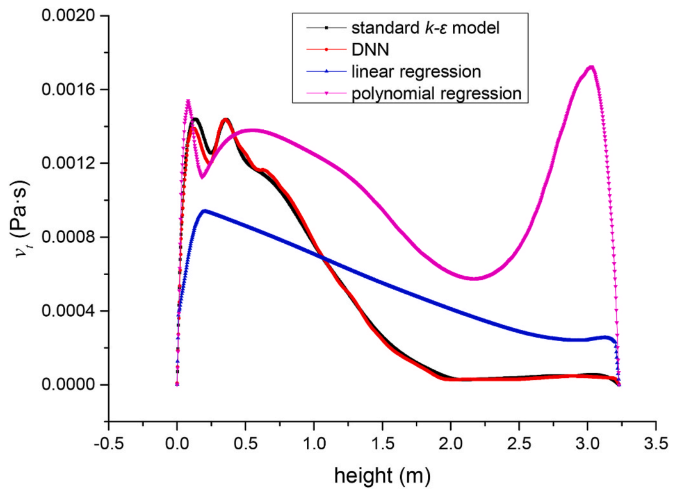

Fig. 6. Turbulent eddy viscosity predicted by the standard *k*-*ε* model, CNN, linear regression and polynomial regression at X = 2.32 m.

turbulent eddy viscosity ν_t. The data obtained from the kNN algorithm on the CG aligns well with the high-resolution CFD data along the cavity's centerline. But the velocity at the entrance of the channel and the ν_t at the bottom of the cavity is slightly different from the FG CFD data because the data at these locations are more discrete. This observation substantiates that data fidelity is maintained during the transition from a FG nonuniform configuration to a CG uniform one.

In this study, we detailed our input and output features. The DD turbulence model identifies deployment regions using inputs such as finite-volume cell-centered coordinates and initial conditions. This study focuses on training a ML surrogate model, as described below.

$$\mathbb{M}_1 : V_x(\mathbf{x}), V_y(\mathbf{x}), V_z(\mathbf{x}), T(\mathbf{x}), C_x(\mathbf{x}), C_y(\mathbf{x}), C_z(\mathbf{x}) \rightarrow \nu_t(\mathbf{x}) \tag{8}$$

Here $\mathbf{x}$ is the mesh point, $V_x(\mathbf{x}), V_y(\mathbf{x}), V_z(\mathbf{x}), T(\mathbf{x})$ indicate the velocity components derived from the initial conditions, and $C_x(\mathbf{x}), C_y(\mathbf{x}), C_z(\mathbf{x})$ denote the mesh-centered coordinates within the domain. The output, $\nu_t(\mathbf{x})$, signifies the steady-state turbulent eddy viscosity, obtainable from different RANS closure methods.

TensorFlow [27], a renowned ML library, facilitates the development of intricate data-driven techniques, including fully connected neural networks and convolutional neural networks. We utilized it to train our surrogate model. The architecture of the DNN was automatically determined using RandomizedSearchCV [28], which explored various configurations to identify the optimal structure. The final DNN consists of 6 hidden layers, each containing 80 neurons, the structural parameters of the DNN model are shown in Table 6. The activation function used is rectified linear units (ReLU), and the loss function is Mean Squared Error (MSE) as defined by Eq. (9). In addition to MSE, the performance of the optimized model is assessed using the R^2 value, computed as per Eq. (10), where an R^2 value approaching 1 indicates a strong regression fit of the model. We trained the DNN model using the entire dataset for 500 epochs with a batch size of 1024. The parameters

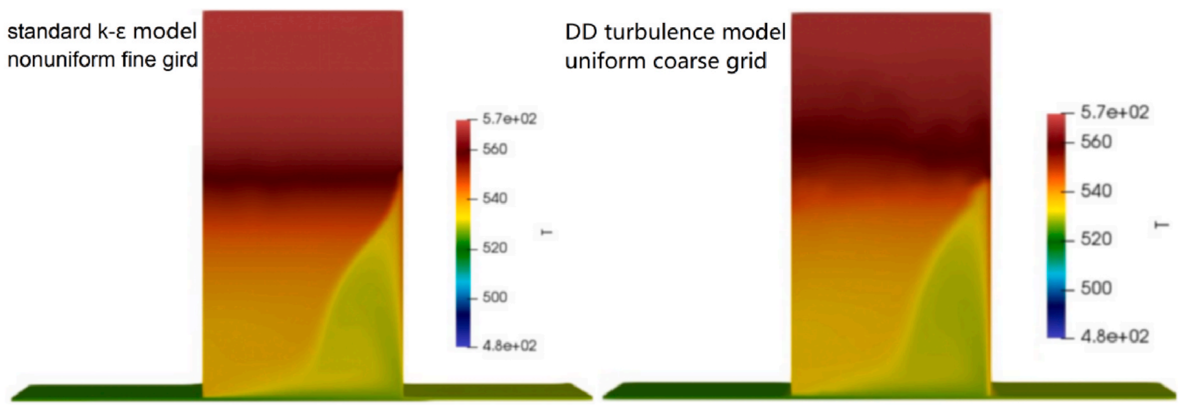

Fig. 7. Temperature field with k-ε model (left) and data-driven turbulence model (right).

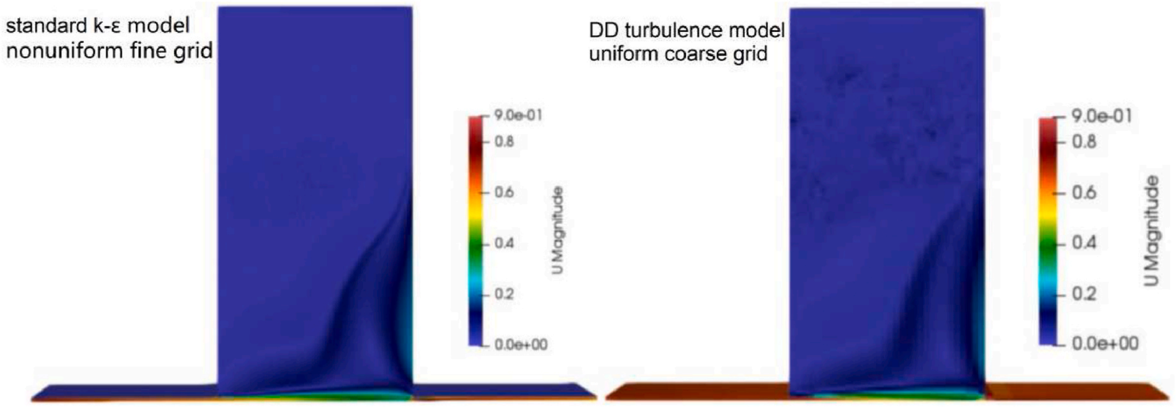

Fig. 8. Velocity field with k-ε model (left) and data-driven turbulence model (right).

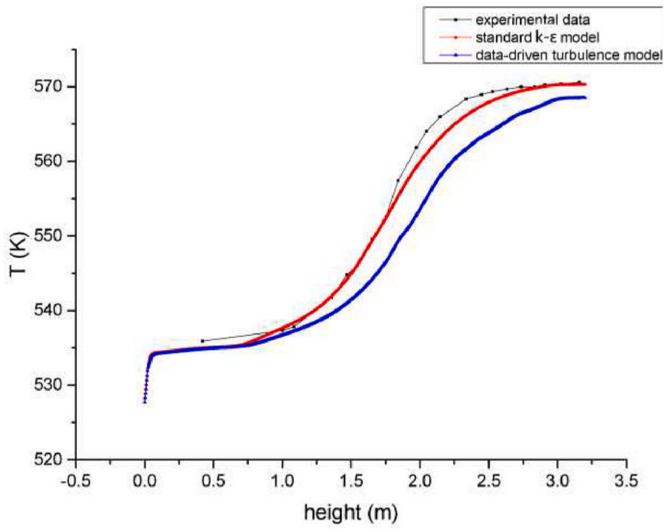

Fig. 9. T comparison of experimental and CFD.

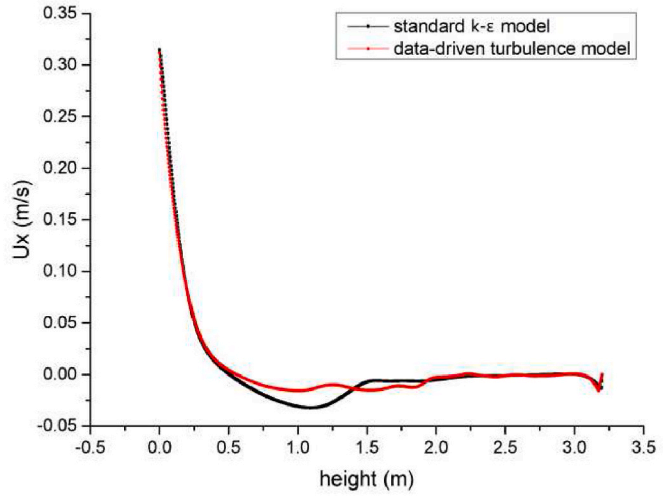

Fig. 10. V comparison of CFD results with k-ε model results (case 0) and data-driven turbulence model (case 0).

Table 8
Inlet and initial conditions of SUPERCAVNA for test cases.

test case	U [m/s]	T [K]	*k*	*ε*	*Re*	*Gr*
1	0.52	515.02	9.95E-04	6.44E-05	1.74 E+06	2.82 E+12
2	0.65	472.67	1.59E-03	1.30E-04	2.03 E+06	4.11 E+12
3	0.87	492.27	2.82E-03	3.07E-04	2.80 E+06	3.58 E+12
4	0.95	546.28	3.41E-03	4.10E-04	3.46 E+06	1.45 E+12
5	0.44	480.31	7.17E-04	3.94E-05	1.38 E+06	3.92 E+12

of the DNN, including weights and biases, were optimized using the Adam optimizer [29], which was initialized with a learning rate of 0.001. Adam is an adaptive learning rate optimization algorithm that effectively combines the benefits of both AdaGrad and RMSProp. By maintaining per-parameter learning rates and leveraging estimates of the first and second moments of the gradients, Adam ensures efficient and robust optimization throughout the training process. Of the total dataset, 90 % was allocated for training, while the remaining 10 % was reserved for validation.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 \tag{9}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y_i})^2} \tag{10}$$

Where n is the number of predictions, y_i denotes the actual value of the data, $\widehat{y}_i$ signifies the value predicted by the model, $\overline{y_i}$ is the average true value across the data.

In the DNN training process, an early-stopping criterion was employed, ceasing the training once the observed metric exhibited no further improvement. This approach aims to minimize the loss function, ensuring the derivation of an optimal model. Upon determining and training the optimal model, its exportation to non-Python environments necessitates the utilization of a function to stabilize the model weights, thereby fixing all trainable parameters and operations inherent to the DNN architecture. Consequently, the finalized optimal DNN model was preserved in an h5 file format.

2.3.2. *Coupling OpenFOAM and Tensorflow*

The coupling of OpenFOAM and TensorFlow should be implemented to successfully invoke the surrogate model to predict turbulent viscosity based on flow features. The coupling mechanism is described below.

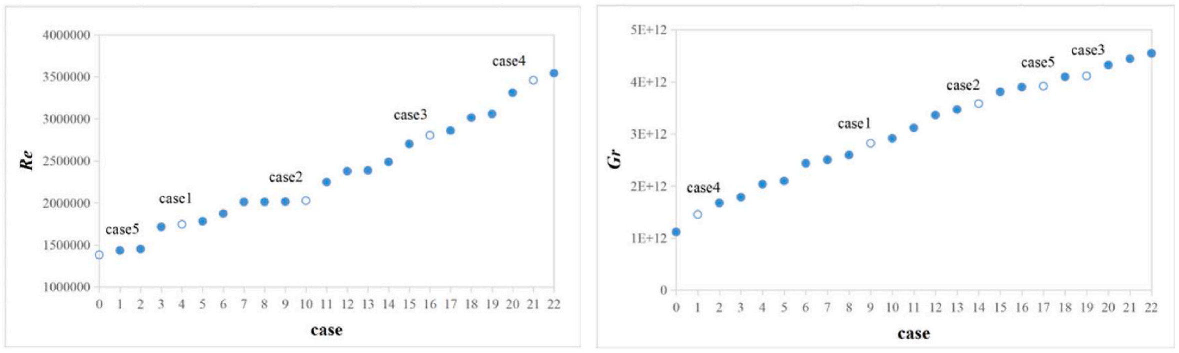

Fig. 11. *Re* and *Gr* of training and test cases.

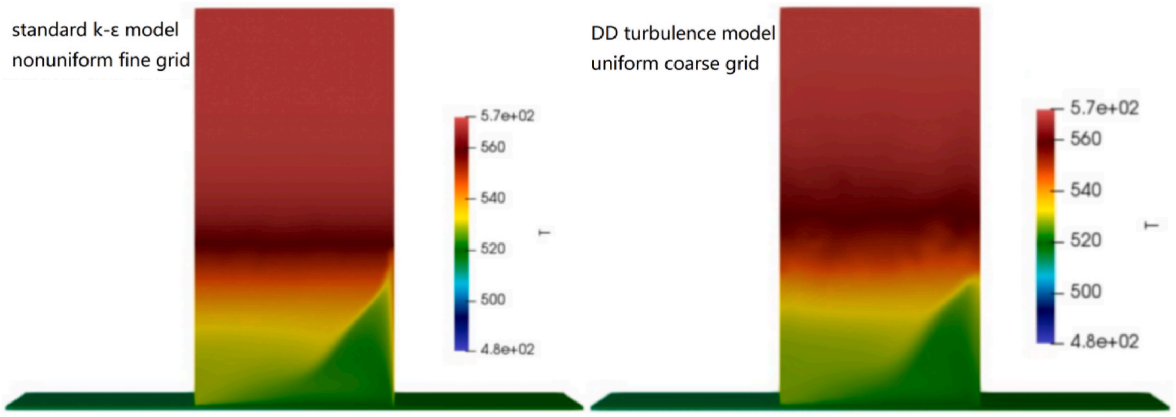

Fig. 12. Temperature field with k-ε model (left) & data-driven turbulence model (right) in test case 1.

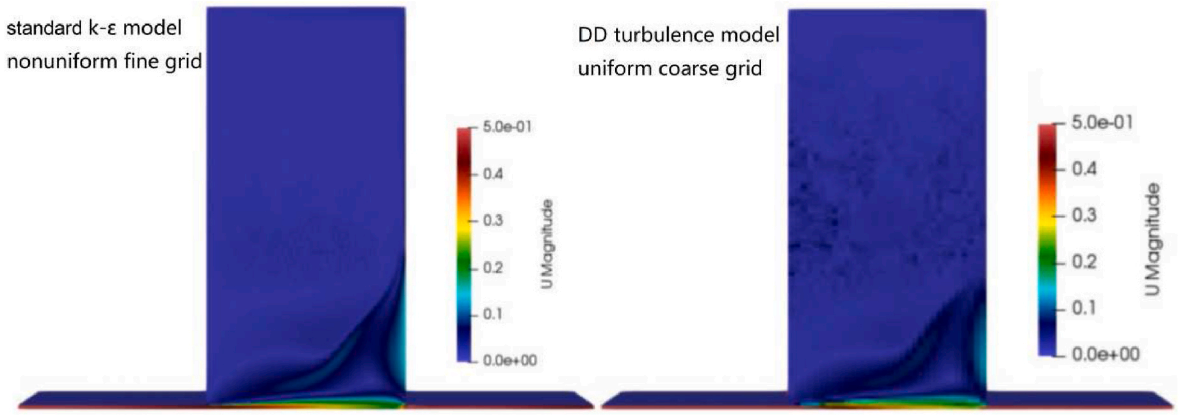

Fig. 13. Velocity field with k-ε model (left) & data-driven turbulence model (right) in test case 1.

The procedure to couple OpenFOAM with TensorFlow leverages the TensorFlow C API. Initially, the trained DNN model was converted from the h5 format to the protobuffer (pb) format to facilitate its export to a C++ environment. Subsequently, the pb file was loaded and integrated with OpenFOAM data structures during the solution process. Next, using the TensorFlow C API, a graph can be loaded and inferences executed within standard C++ code. The DNN model, formatted in pb, is importable to OpenFOAM. Then, a new turbulence model was modified and compiled, linking to TensorFlow C dynamic libraries, which can predict turbulence eddy viscosity, replacing the k and ε equation solutions. Lastly, a new buoyantFoam solver was modified and compiled to match the turbulence model and call for a ML prediction within OpenFOAM.

3. Results and discussion

In the subsequent section, we delineate the efficacy of our ML approach using a priori analyses and then discuss the results stemming from its application as a DD turbulence model.

3.1. Machine learning

We delineated the efficacy of our ML approach using statistical performance metrics. The convergence trajectory for the ML frameworks is depicted in Fig. 5. After 64 epochs, a satisfactory parameterization of the input-output relationship was achieved, prompting the termination of the training via the early-stopping criteria. For the surrogate model of

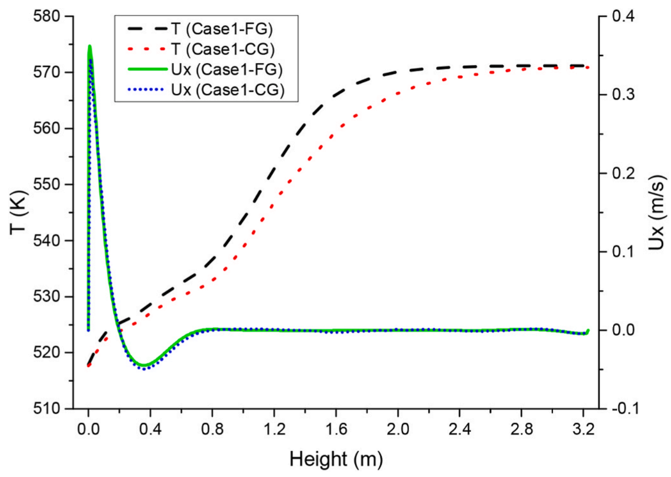

Fig. 14. *T* & *V* comparison in test case 1.

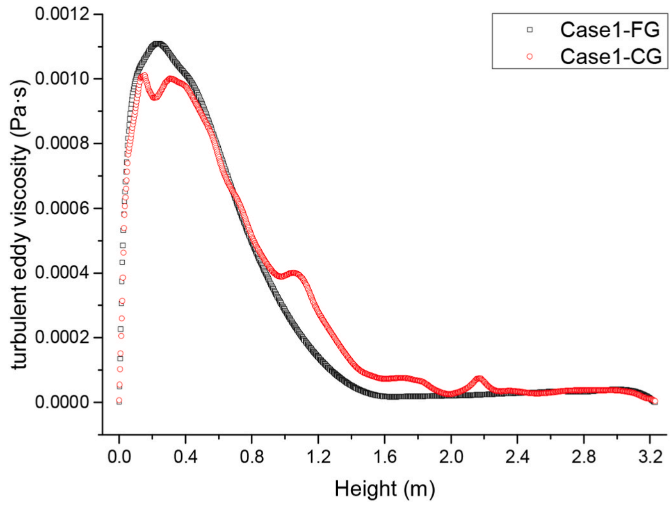

Fig. 15. ν_t comparison in test case 1.

turbulence eddy viscosity, the fully trained DNN achieves the mean squared error MSE = 0.0026 and a coefficient of determination of R^2 = 0.997 for training data sets, the MSE = 0.0045 and R^2 = 0.996 for validation data sets, indicating a successful parameterization.

To further explore efficient and accurate methods for predicting turbulent eddy viscosity, we employed DNN, linear regression, and polynomial regression to establish a mapping between the flow field and temperature field features and the turbulent eddy viscosity distribution. The predictions generated by these three methods were compared with the turbulent eddy viscosity obtained using the standard *k-ε* model with fine-grid CFD, as illustrated in Fig. 6. The results indicate that the DNN model effectively captures the complex variation trends of turbulent eddy viscosity, particularly in regions with low turbulent eddy viscosity, where the DNN's predictions closely align with those of the standard *k-ε* model. In regions with greater height, there is some deviation between the DNN prediction curve and the reference curve. Overall, the DNN model demonstrates strong learning capabilities, effectively handling the nonlinear relationships inherent in turbulent eddy viscosity. In contrast, the linear regression model fails to adequately capture the nonlinear characteristics of turbulent eddy viscosity, particularly in regions with high turbulent eddy viscosity, where its predictions are significantly lower than the reference results obtained from the standard *k-ε* model. This indicates that linear regression is unsuitable for predicting complex turbulent characteristics. The polynomial regression model of order 5, while performing better than linear regression in capturing local variations in turbulent eddy viscosity, exhibits overfitting in regions with low turbulent eddy viscosity, resulting in a significant deviation from the reference results. The above results indicate that the ν_t magnitude can be captured accurately by the predictions of the DNN surrogate model.

3.2. Data-driven turbulence model evaluation

Since the aim of this framework was to develop a DD turbulence model to replace the twoequation model and implement CG CFD simulation, it was a natural choice to validate the DD turbulence model by comparing temperature and velocity field propagated from DNN model and that of standard *k-ε* model.

In assessing the ML model's performance, CFD results were extracted for both qualitative and quantitative analyses against the model's predictions. For this study, data from two planes were analyzed qualitatively, whereas data from two specific lines (including a vertical line aligned with the domain's center) were used for quantitative analyses.

The OpenFOAM simulation results with the standard *k-ε* model and with 1,163,200 cells and those of the DD turbulence model with 68,896 cells of baseline case (case 0) can be seen in Figs. 7 and 8, respectively. These figures showcase both the velocity magnitude and temperature distributions. Within the SUPERCAVNA test section, when liquid sodium is introduced into the cavity, a vertically spreading wall-bounded jet forms. A portion of this jet leaves the cavity via the outlet channel, while the remaining portion impacts the right wall, creating a recirculation zone. The size of this zone varies based on the sodium injection rate at the inlet and the intensity of thermally stratified layers forming at the cavity's top due to the temperature gradient established by the heating wall.

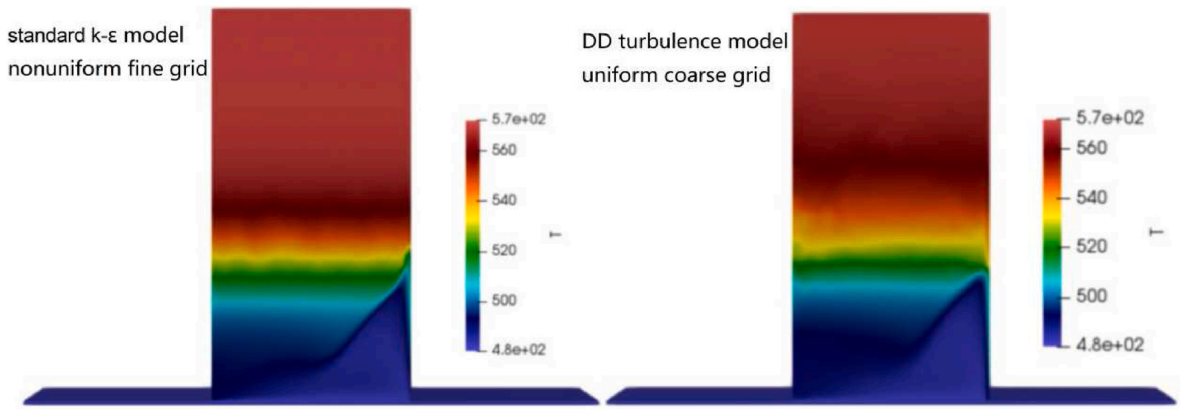

Fig. 16. Temperature field with k-ε model (left) & data-driven turbulence model (right) in test case 2.

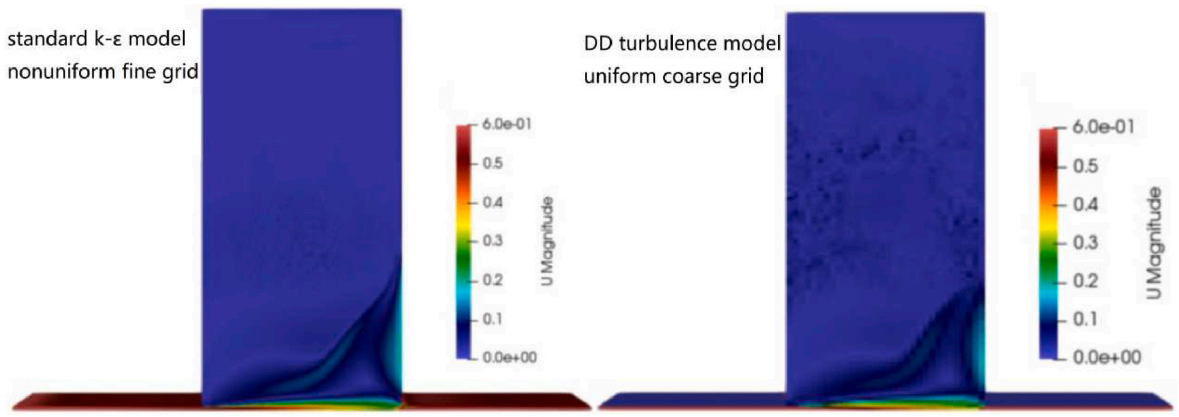

Fig. 17. Velocity field with k-ε model (left) & data-driven turbulence model (right) in test case 2.

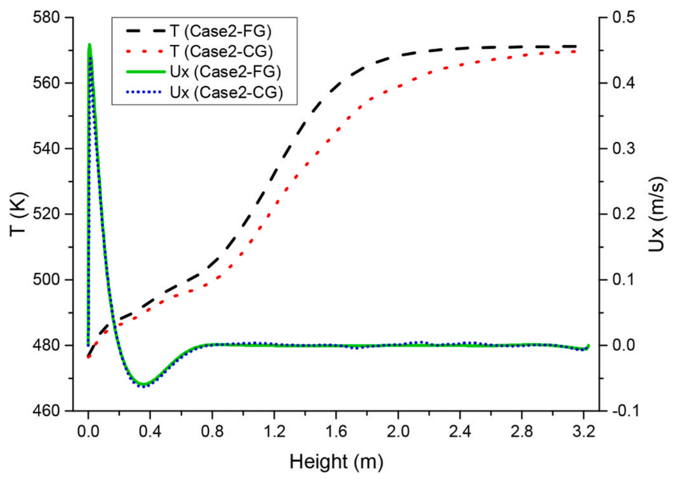

Fig. 18. *T* & *V* comparison in test case 2.

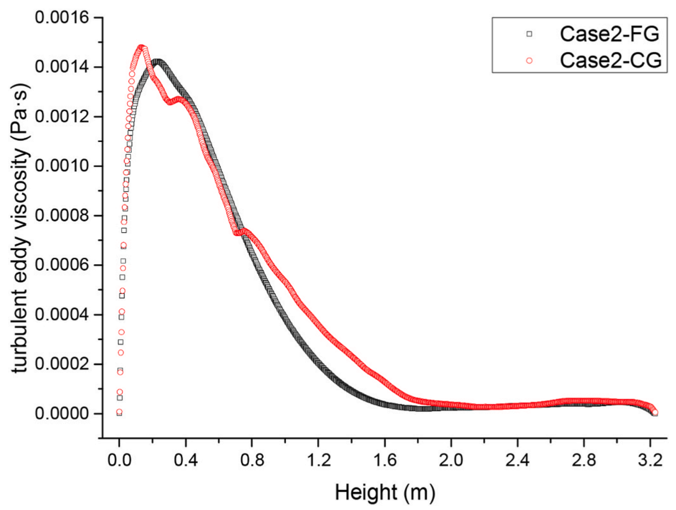

Fig. 19. ν_t comparison in test case 2.

The temperature and velocity distributions in training case 0 (refer to Table 4), both on fine and CGs, are approximately in the same range. For training case 0, both the predictions from the standard k-ε model and the DD turbulence model show the formation of thermally stratified layers in the cavity's upper region, indicating significant buoyancy effects due to temperature gradients from the heated wall. In both scenarios, the jets tend to ascend within the cavity as they approach the right wall, a pronounced thermal barrier prevents further intrusion into the upper cavity. Moreover, as depicted by the profiles and temperature distribution fields, temperatures remain approximately constant horizontally across various vertical positions, post the recirculation zone. Such a trend signifies pronounced thermal stratification layers, effectively captured in the CG OpenFOAM simulation utilizing the DD turbulence model.

A comparison between the standard k-ε model, the DD turbulence model, and the experimental temperature profiles at locations X = 1.62 m is given in Fig. 9. For case 0, the FG CFD simulations with the standard k-ε model are in good agreement with the experimental data. The CG CFD with DD turbulence model predicts the temperature profiles along the height of the cavity; some differences appear between FG CFD with the standard k-ε model and experimental results; the most significant difference occurs above the thermal stratification layer, while the flow in the lower side of the cavity is well predicted. The CG CFD data profiles differ by up to 7 °C from the experimental data, while they differ by up to 5 °C from the FG CFD data.

There are 4 main sources of the above errors: ① CG setup when solving the conservation equations, ② conversion from FG data to CG data by kNN algorithm, ③ training errors in the DNN model, ④ errors of the data based on solving the RANS equations. In follow-up research, we will consider reducing the error of the simulation results of the DD turbulence model by the following approaches. ① increasing the number of CG appropriately, ② improving the training accuracy of the DNN model, ③ adding some of the available experimental data in the training procedure.

Fig. 10 presents a comparison of the velocity component profiles between the standard k-ε model and the DD turbulence model at X = 1.62 m. The results from CG CFD simulations using the DD turbulence model and FG CFD simulations with the standard k-ε model show good agreement. There is only a slight difference in the velocity component in the middle of the cavity. This indicates that the CG CFD with the DD turbulence model has a good prediction accuracy for the velocity component.

3.3. Speedup from data-driven turbulence model

The presented framework offers a significant advantage by eliminating the need for an extra partial differential equation in computing ν_t. Upon training the DNN model, the steady-state turbulent eddy viscosity predictor can be immediately employed at the onset of the simulation, taking into account the initial conditions. As a result, only the equations pertaining to pressure and velocity require iterative resolution for convergence. Additionally, adopting a CG configuration enhances the computational efficiency of the CFD simulation.

All numerical experiments were conducted using OpenFOAM in

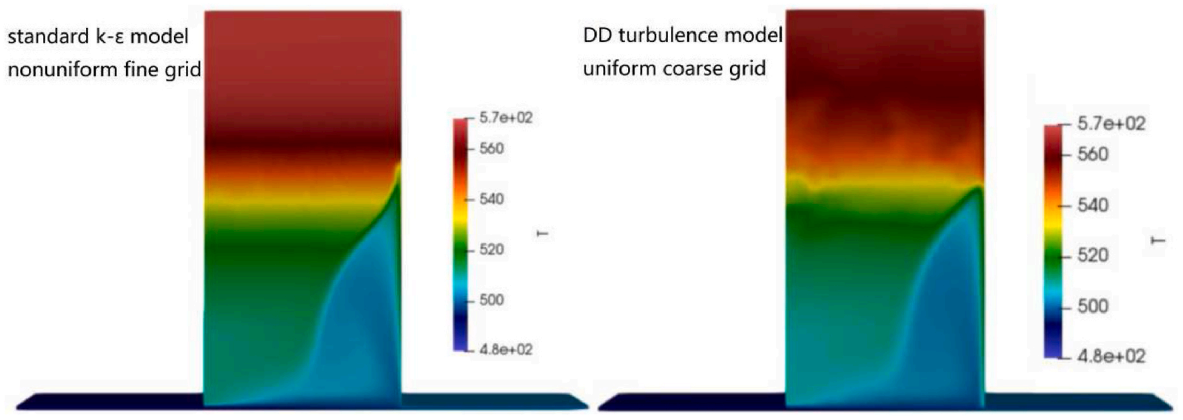

Fig. 20. Temperature field with k-ε model (left) & data-driven turbulence model (right) in test case 3.

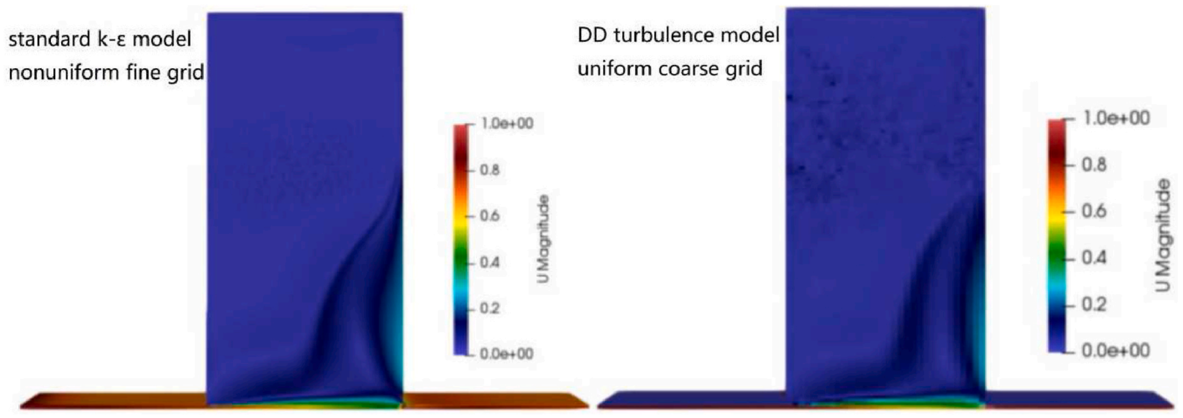

Fig. 21. Velocity field with k-ε model (left) & data-driven turbulence model (right) in test case 3.

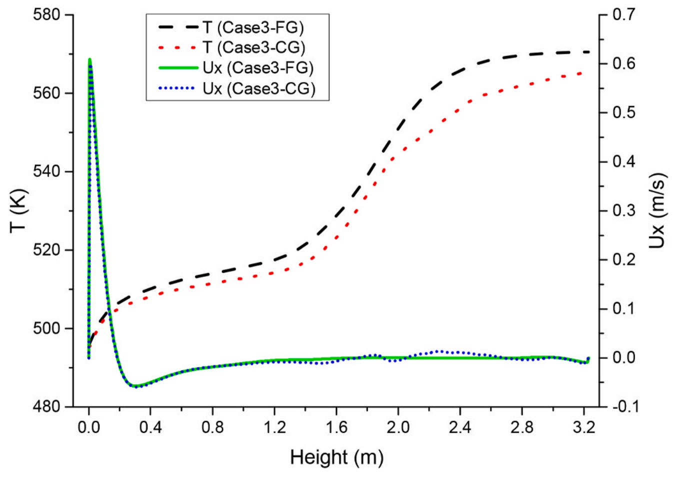

Fig. 22. *T* & *V* comparison in test case 3.

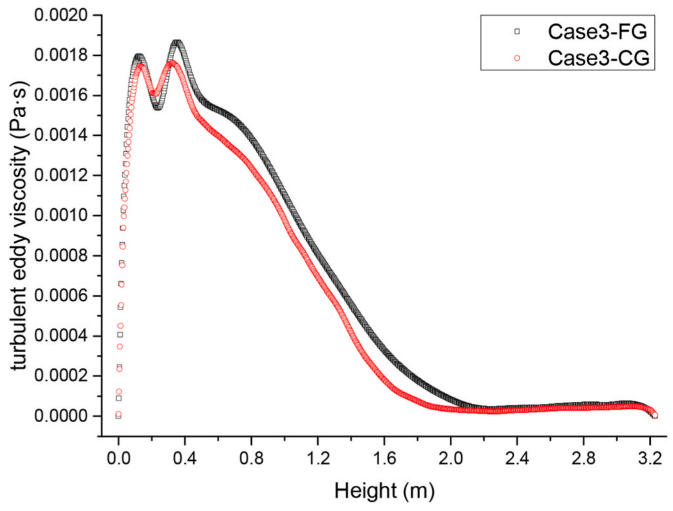

Fig. 23. ν_t comparison in test case 3.

parallel on an Intel Core-i7 processor. For a 1,163,200-cell mesh, the CFD simulations using the standard *k-ε* model took about 896 core-hours, converging in 200 iterations with the steady-state solver. For a 68,896-cell mesh, the CFD simulations using the DD turbulence model took roughly 2 core-hours, reaching convergence in just 25 iterations.

3.4. Generalization ability of data-driven turbulence model

The generalization ability is the key to determining whether the DD turbulence model could adapt properly to new data and be applied to various conditions in CFD simulations. In this work, the generalization ability of the DD turbulence model is evaluated with 5 steady-state test cases, with inlet and initial conditions as shown in Table 8. The fluid velocity and temperature of the inlet channel in these 5 test cases (from Table 8), further to be denoted as cases 1–5, are different from those 18 training cases (from Table 4). The Reynolds numbers and Grashof numbers of the 18 training cases and 5 test cases basically present a

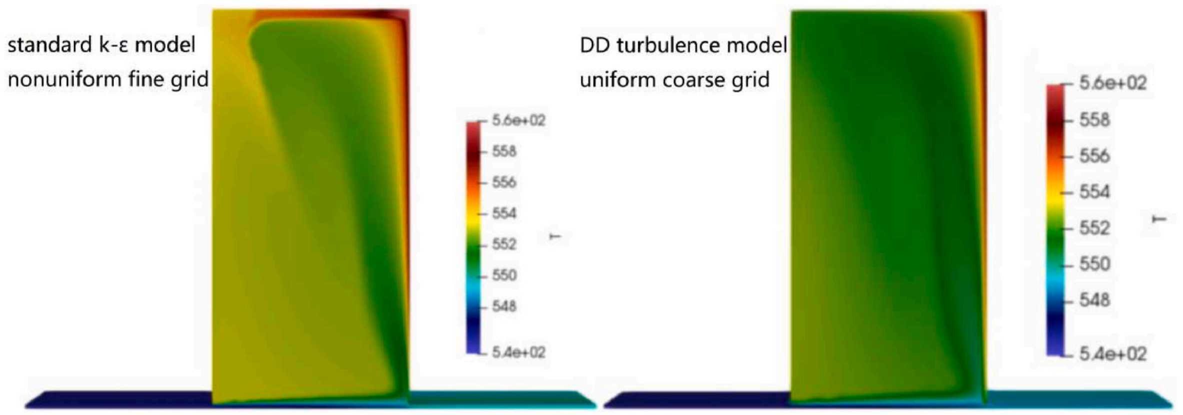

Fig. 24. Temperature field with k-ε model (left) & data-driven turbulence model (right) in test case 4.

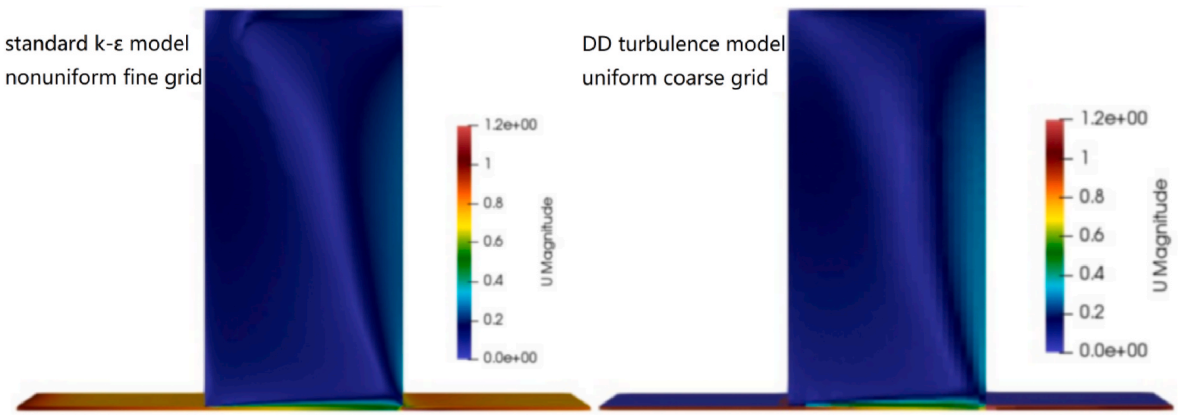

Fig. 25. Velocity field with k-ε model (left) & data-driven turbulence model (right) in test case 4.

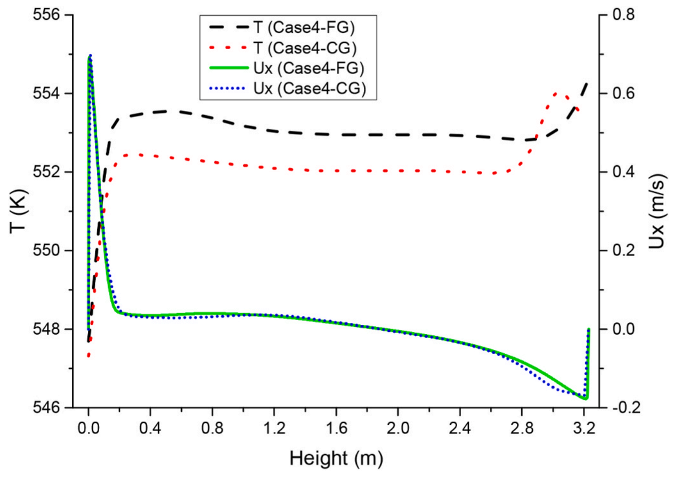

Fig. 26. *T* & *V* comparison in test case 3.

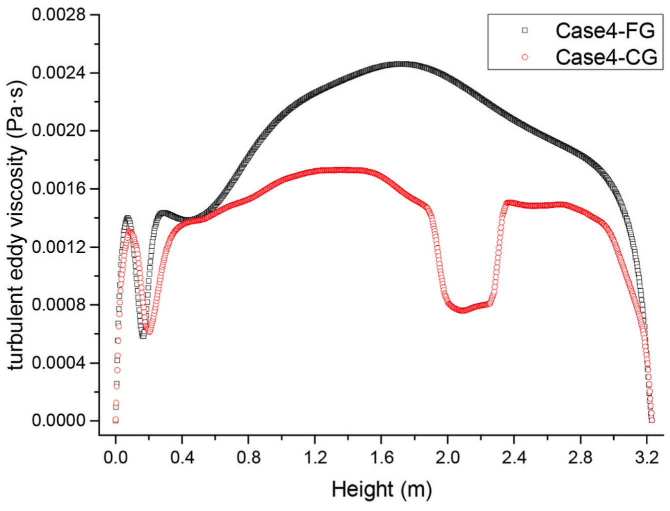

Fig. 27. ν_t comparison in test case 3.

linear distribution, as shown in Fig. 11. The 5 test cases are evenly selected according to the Reynolds number and the Grashof number to evaluate the prediction performance of the DD turbulence model for different fluid flow patterns.

Following the approach used for the baseline case evaluation, we obtained the temperature, horizontal velocity component, and turbulent viscosity field for cases 1–5. Steady-state snapshots from both prediction methods are utilized for comparison. Figs. 12–31 illustrate the qualitative and quantitative comparisons.

Figs. 15, 19, 23, 27 and 31 show the turbulent eddy viscosity ν_t comparisons for test cases 1–5 at the horizontal location X = 2.32m. From the figures, we note that the turbulent eddy viscosity field predicted by the DD turbulence model aligns well with the trends and distributions of the standard *k*-*ε* model for cases 1–3, despite minor acceptable discrepancies. However, in case 4, the predicted turbulent eddy viscosity is inconsistent with the calculation results of the standard *k*-*ε* model, especially at the location around the height of 2.0 m; the

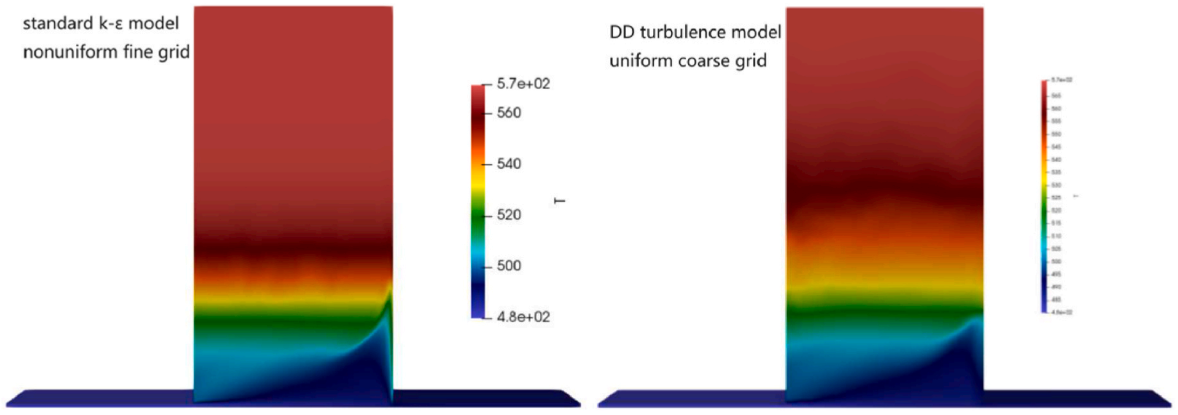

Fig. 28. Temperature field with k-ε model (left) & data-driven turbulence model (right) in test case 5.

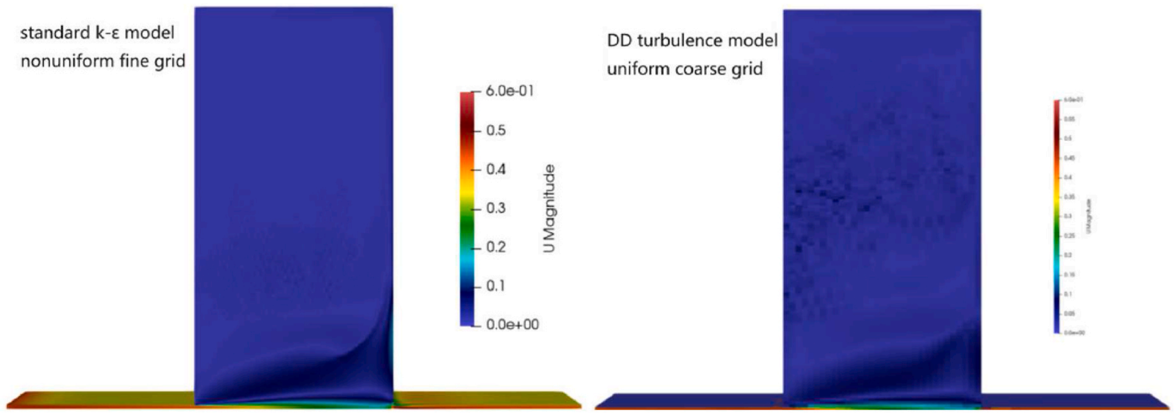

Fig. 29. Velocity field with k-ε model (left) & data-driven turbulence model (right) in test case 5.

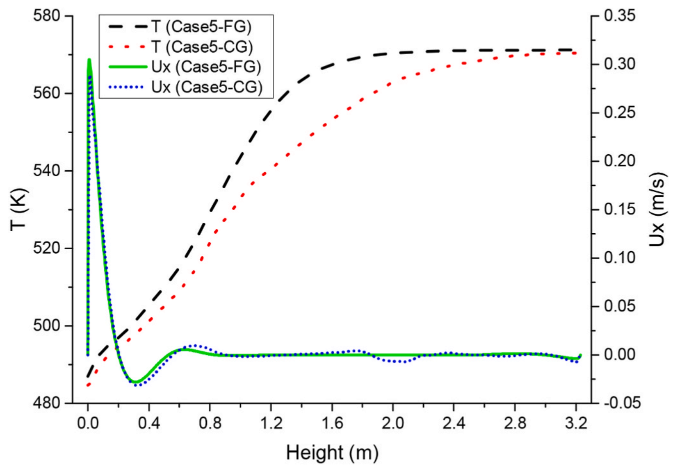

Fig. 30. T & V comparison in test case 5.

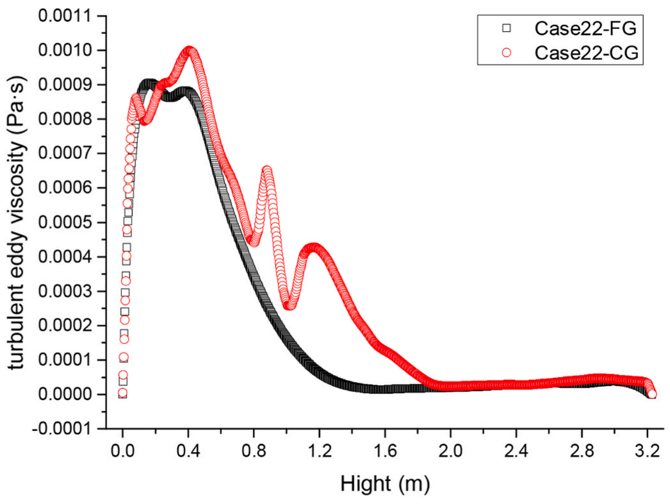

Fig. 31. ν_t comparison in test case 5.

prediction seriously underestimated the turbulent eddy viscosity. The reason is that the fluid flow pattern of test case 4 is significantly different from the previous 3 cases, and only 2 training cases have flow patterns similar to case 4. The imbalance of training data leads to a large prediction deviation of turbulent eddy viscosity in case 4. In order to test the extrapolation ability of the DD turbulence model, case 5 with Reynolds number outside the training set was simulated. In case 5, the turbulent eddy viscosity predicted by the DD turbulence model and calculated by the standard k-ε model exhibit similar overall trends, but with notable differences in the middle height regions (0.5–1.5m) of the cavity, the former displays higher oscillations compared to later. The above results indicate that under the condition of sufficient and balanced training data, the DD turbulence model has the ability to cope with moderately perturbed inlet and initial conditions. Despite local fluctuations and differences, the overall trend consistency and the gradual convergence in various regions indicate that the DD turbulence model possesses a certain degree of extrapolation capability to predict the turbulent eddy viscosity.

Figs. 14, 18, 22, 26 and 30 display for the cases 1–5 corresponding line plots for the horizontal velocity component and temperature magnitudes at the location X = 2.32 m. We can observe a good agreement of the horizontal velocity component between the DD turbulence model and the standard *k*-*ε* model in all test cases, because the velocity and pressure solvers remain intact and the convergence fields maintain their respective symmetries during CFD simulations with the DD turbulence model.

The figures show that the temperature field modeled with the DD turbulence model is generally consistent with the standard *k*-*ε* model in terms of trend and distribution across all five test cases. In cases 1–3, the maximum temperature difference consistently occurs above the thermal stratification interface, reaching 7.2 K, 14.4 K, and 10.5 K, respectively. In case 4, the thermal stratification phenomenon is absent, and the maximum temperature difference of 1.2 K is observed in the lower part of the cavity. In case 5, the location of the maximum temperature difference aligns with the position of the largest fluctuation in turbulent eddy viscosity, which is in the middle height region of the cavity. The maximum temperature difference in case 5 is 11.9 K.

The evaluation results demonstrate that the DD turbulence model can generalize to inlet and initial conditions. When the training data is sufficient and balanced, the DNN model's generalization ability is high enough to make predictions for any untrained case at the level of training cases.

4. Conclusion

In this research, we proposed a DD modeling approach based on deep neural networks to develop data-driven closure to support CG multi-dimensional modeling in CFD code and system code. We developed a framework of a DD turbulence model, which includes database construction, surrogate model training, coupling OpenFOAM and Tensorflow, testing and evaluation. We demonstrated the feasibility of this approach with a DD turbulence model applied during steady-state simulations that involve thermal mixing and stratification phenomena in the SUPERCAVNA test section. The proposed DD turbulence model uses a CG setup for computational efficiency and is trained on FG CFD data to maintain high accuracy. The DD turbulence model has a certain generalization ability to adapt to different inlet and initial conditions for steady-state OpenFOAM simulation. We will further investigate DD turbulence models suitable for transient conditions, focusing on predicting the temporal evolution and dependencies of turbulent eddy viscosity distributions. To achieve this, the temporal sequence prediction algorithms will be incorporated to train surrogate models. Additionally, to improve the generalization capability of data-driven turbulence models across different geometries, the following strategies will be investigated: ① non-dimensionalization of input features: Transform physical quantities into dimensionless numbers to standardize data from various geometries and flow conditions, ensuring accuracy and robustness. ② geometric parameterization: include geometric parameters as input features to help the model adapt to different geometries, enhancing flexibility without extensive retraining. ③ transfer learning: use a pre-trained model on a specific geometry and fine-tune it with a smaller dataset from a new geometry to reduce the need for extensive training data and computational resources, enabling quick adaptation to different geometries.

CRediT authorship contribution statement

Zijing Liu: Writing – original draft. **Pengcheng Zhao:** Funding acquisition, Conceptualization. **Badea Aurelian Florin:** Resources, Investigation. **Xu Cheng:** Writing – review & editing, Resources.

Acknowledgement

This work is supported by the Joint Fund of the Ministry of Education for Equipment Pre-research (Grant No. 8091B032243), China Scholarship Council (grant no. 202008430068).

References

[1] Zeyun Wu, et al., A status review on the thermal stratification modeling methods for Sodium-cooled Fast Reactors, Prog. Nucl. Energy 125 (2020) 103369.
[2] N. Aksan, J.H. Choi, Y.J. Chung, et al., Passive Safety Systems and Natural Circulation in Water Cooled Nuclear Power Plants, International Atomic Energy Agency (IAEA), 2009, p. 159. IAEA-TECDOC-1624.
[3] Zhao Haihua, Per F. Peterson, An Overview of Modeling Methods for Thermal Mixing and Stratification in Large Enclosures for Reactor Safety Analysis, 2010.
[4] Sarah Morgan, et al., Thermal stratification modeling for sodium-cooled fast reactors: a status update[C], in: International Conference on Nuclear Engineering, vol. 51531, American Society of Mechanical Engineers, 2018 V009T16A078.
[5] T.H. Fanning, et al., The sas4a/SASSYS-1 safety analysis code system, ANL/NE-16/19, in: Nuclear Engineering Division, Argonne National Laboratory, 2017 (ANL).
[6] Shibahara Makoto, Takashi Takata, Akira Yamaguchi, Numerical study on thermal stratification phenomena in upper plenum of LMFBR "MONJU", Nucl. Eng. Des. 258 (2013) 226–234.
[7] David Pialla, et al., Overview of the system alone and system/CFD coupled calculations of the PHENIX Natural Circulation Test within the THINS project, Nucl. Eng. Des. 290 (2015) 78–86.
[8] K. Fukami, K. Fukagata, K. Taira, Super-resolution reconstruction of turbulent flows with machine learning, J. Fluid Mech. 870 (2019) 106–120.
[9] J. Jeon, J. Lee, S.J. Kim, Finite volume method network for the acceleration of unsteady computational fluid dynamics: non-reacting and reacting flows, Int. J. Energy Res. 46 (8) (2022) 10770–10795.
[10] E. Ajuria Illarramendi, A. Alguacil, M. Bauerheim, et al., Towards an hybrid computational strategy based on deep learning for incompressible flows[C], AIAA Aviation 2020 Forum (2020) 3058.
[11] S.R. Allmaras, F.T. Johnson, Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence model[C]//Seventh International Conference on Computational Fluid Dynamics (ICCFD7), 2012, p. 1902.
[12] Brendan D. Tracey, Karthikeyan Duraisamy, Juan J. Alonso, A Machine Learning Strategy to Assist Turbulence Model development[C], 53rd AIAA aerospace sciences meeting, 2015, p. 1287.
[13] Liang Sun, A.N. Wei, L.I.U. Xuejun, et al., On developing data-driven turbulence model for DG solution of RANS, Chin. J. Aeronaut. 32 (8) (2019) 1869–1884.
[14] Linyang Zhu, et al., Machine learning methods for turbulence modeling in subsonic flows around airfoils, Phys. Fluids 31 (1) (2019) 015105.
[15] Jinhua Xu, Daniel WC. Ho, A node pruning algorithm based on optimal brain surgeon for feedforward neural networks[C]//Advances in Neural Networks-ISNN 2006, in: Third International Symposium on Neural Networks, Chengdu, China, May 28-June 1, 2006, Proceedings, Part I 3, Springer Berlin Heidelberg, 2006, pp. 524–529.
[16] Chang Chih-Wei, Jun Fang, T. Dinh Nam, Reynolds-averaged turbulence modeling using deep learning with local flow features: an empirical approach, Nucl. Sci. Eng. 194 (8–9) (2020) 650–664.
[17] Y. Zhu, R. Hu, et al., Development of a data-driven turbulence model for 3d thermal stratification simulation during reactor transients, NURETH-18, American Nuclear Society, 2019, pp. 2223–2234.
[18] A.S. Iskhakov, C.K. Tai, I.A. Bolotnov, et al., Data-driven RANS turbulence closures for forced convection flow in reactor downcomer geometry, Nucl. Technol. 210 (7) (2024) 1167–1184.
[19] R. Maulik, H. Sharma, S. Patel, B. Lusch, E. Jennings, A turbulent eddy-viscosity surrogate modeling framework for Reynolds-averaged Navier-Stokes simulations, Comput. Fluid 227 (2021) 104777.
[20] Y. Liu, R. Hu, A. Kraus, et al., Data-driven modeling of coarse mesh turbulence for reactor transient analysis using convolutional recurrent neural networks, Nucl. Eng. Des. 390 (2022) 111716.
[21] U. Bieder, G. Ziskind, A. Rashkovan, CFD analysis and experimental validation of steady state mixed convection sodium flow, Nucl. Eng. Des. 326 (2018) 333–343.
[22] Reynolds O.IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion[J]. Phil. Trans. Roy. Soc. Lond., 1895 (186): 123-164.].
[23] Werner Pfrang, Dankward Struwe, Assessment of Correlations for Heat Transfer to the Coolant for Heavy Liquid Metal Cooled Core designs[M], FZKA, Karlsruhe, BW, 2007.
[24] Ling Zou, Daniel Nunez, Rui Hu, Development and Validation of SAM Multi-Dimensional Flow Model for Thermal Mixing and Stratification Modeling[R], Argonne National Lab.(ANL), Argonne, IL (United States), 2020.
[25] A. Kraus, S. Aithal, A. Obabko, et al., Erosion of a large-scale gaseous stratified layer by a turbulent jet-simulations with URANS and LES approaches[C]//16th international topical meeting on nuclear reactor thermal hydraulics. NURETH 2015, American Nuclear Society, 2015, pp. 1448–1461.
[26] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[27] M. Abadi, A. Agarwal, P. Barham, et al., Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2016 arXiv preprint arXiv:1603.04467.
[28] P. Probst, M.N. Wright, A.L. Boulesteix, Hyperparameters and tuning strategies for random forest, Wiley Interdisciplinary Reviews: Data Min. Knowl. Discov. 9 (3) (2019) e1301.
[29] Diederik P. Kingma, Jimmy Ba. Adam: a method for stochastic optimization, arXiv preprint arXiv:1412. (2014) 6980.