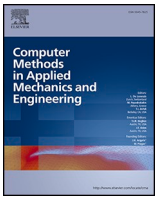
Computer Methods in Applied Mechanics and Engineering 417 (2023) 116444

Contents lists available at ScienceDirect

Computer Methods in Applied Mechanics and Engineering

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



A super-real-time three-dimension computing method of digital twins in space nuclear power

Enping Zhu, Tao Li, Jinbiao Xiong, Xiang Chai *, Tengfei Zhang, Xiaojing Liu

School of Nuclear Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

ARTICLE INFO

Keywords:
Digital twin
GPU and multi-core CPU
Machine learning
Super-real-time
Space nuclear reactor

ABSTRACT

Digital twins (DTs) have attracted widespread attention in academia and industry in recent years. It can accurately reflect the physical world in real-time, enabling online monitoring, control, and prediction operations. Their foundation is super-real-time computing and high data representation capabilities. However, current DTs do not achieve 3D super-real-time computing. This study proposes a novel 3D computational method for solving fluid–solid coupling problems in a super-real-time. The method is based on a mixed solution framework that combines traditional numerical methods with deep learning operators. Specifically, the method employs multi-core CPU parallel acceleration to solve the solid equations while leveraging the computing power of GPU to solve the fluid equations. The fluid–solid coupling is achieved through information exchange between the GPU and the multi-core CPU. In addition, the proposed method introduces a new deep learning operator framework based on the DeepONET. The framework is accompanied by a database structure that facilitates model training and validation and a loss function that guides the training. The space nuclear reactor, an improved TOPAZ-II system, was selected to demonstrate its feasibility. Four non-training transient conditions were simulated to test the generalization performance. The results show that the proposed method achieves an average error between the calculated results and reference values below 2.5%, with the average error of thermodynamic parameters below 1.5%. The average deviation between system parameter peak values during the transient process and the reference value was less than 5 s. The result meets the acceptable error level and satisfies the super-real-time requirements with a time acceleration ratio of approximately 1.17, which is 60 times faster than traditional numerical methods. The results demonstrate the accuracy and efficiency of the proposed method for DT.

1. Introduction

The term “Digital Twin” has garnered significant attention in both academic and industrial domains, despite its conceptualization dating back several decades [1]. The concept of twins was initially employed in NASA’s Apollo program [2]. NASA constructed two identical spacecraft, with one remaining on Earth and referred to as the twin. This twin served as a prototype that replicated real operating conditions, enabling real-time behavior simulation. Prior to the mission, the twin was utilized for training purposes. During the flight, the twin simulated various scenarios on Earth to support astronauts in critical decision-making situations [3].

DT is a simulation process that maximizes data utilization, including physical models, sensor updates, and operational history. It

* Corresponding author.
E-mail address: xiangchai@sjtu.edu.cn (X. Chai).

https://doi.org/10.1016/j.cma.2023.116444
Received 8 June 2023; Received in revised form 6 September 2023; Accepted 7 September 2023
Available online 28 September 2023

integrates multiple disciplines, physical aspects, scales, and probabilistic mappings in a virtual space, thereby reflecting the entire lifecycle of the corresponding physical system, as shown in Fig. 1. DTs find widespread applications in various fields [4–7]. On the one hand, they intuitively and accurately depict the real state of physical systems, assisting in engineering design, operation, and maintenance [8]. On the other hand, DTs enable real-time online monitoring and even super-real-time prediction [9,10], facilitating control over the physical world and aiding operators in making informed decisions. Zhou Y et al.. analyzed the needs and issues of DTs in industrial energy systems and proposed requirements, including achieving super real-time computing [11]. Li C et al. provided a comprehensive explanation of the intelligent energy grid supported by DT, underscoring the significance of super real-time monitoring and health management as crucial assurances for DTs [12]. Zheng Y et al. devised a DT for the doubly fed induction generator, featuring remarkable super real-time simulation capabilities to facilitate fault diagnosis and continuous monitoring [13].

The foundation for realizing the aforementioned functions of DTs lies in their ability to achieve super-real-time computing and real-time interaction with the physical world [14]. When DTs receive measurement data from the physical world, they dynamically adjust parameters to accurately reflect the real state of the physical system. Building upon this capability, DTs facilitate the analysis and prediction of the system's transient behavior [15], providing valuable assistance to operators and maintenance personnel in effectively controlling and managing the physical system. This necessitates the achievement of super-real-time computing in the DT's code, ensuring timely execution of parameter updates, prediction, control, and various other operations [16–18]. Overall, super-real-time computing aims to enhance the implementation and expand the functionality of DTs. Additionally, to achieve precise mapping of the real physical world, DTs must employ high-dimensional models for simulating physical systems [19,20]. Consequently, high-dimensional and super-real-time system computing pose significant challenges to the development of DTs.

Table 1 provides an overview of the existing DTs. Analysis of Table 1 reveals a prominent trend in the advancement of DTs towards achieving super-real-time computing capabilities. However, the majority of current super-real-time computing methods rely on the surrogate model (SM) approach, such as neural networks [21] and support vector machines [22] which learn 1D model data to predict future states. Unfortunately, these approaches do not realize super-real-time computing for 3D models [23–25]. The 1D model's data representation ability is limited, and the available system state points are insufficient to support engineering design, optimization, and monitoring. Furthermore, most 1D models rely on empirical relationships and lack a foundation in mechanism models. Consequently, when a physical system or sensor malfunctions, the 1D model fails to make accurate judgments and identify the cause of the malfunction.

It is worth noting that surrogate models do not necessarily need to be one-dimensional. Currently, numerous scholars are investigating reduced-order models for real-time simulation by leveraging numerical data from 3D/2D numerical simulations in combination with machine learning. Fresca et al. proposed a method called POD-DL-ROMs to address the issues of traditional reduced-order models [32]. They applied randomized proper orthogonal decomposition for a priori dimensionality reduction on the snapshots of the full-order model, effectively reducing the training cost of the deep learning. Furthermore, Fresca et al. demonstrated that the proposed POD-DL-ROMs method can achieve real-time computation and analysis in solving fluid mechanics problems and micro-electro-mechanical systems [33,34]. Conti et al. proposed a method that employs autoencoder neural networks and parametric sparse identification of nonlinear dynamics to construct a low-dimensional dynamical model [35]. Makkar et al. used artificial neural networks to minimize the number of high-fidelity data points required for correcting the outputs of the low-fidelity model [36]. Drakoulas et al. proposed a framework called FastSVD-ML-ROM, which utilizes singular value decomposition to compute a linear subspace of the multi-fidelity solutions during the simulation process and train neural networks to establish the relationship between input parameters, latent space, and prediction results [37].

The 3D DTs listed in Table 1 employ the SM approach to facilitate real-time calculations [27–30]. In comparison to 1D models, they demonstrate enhanced data representation capabilities. However, it is important to note that further improvements are necessary to enhance the accuracy and reliability of their outcomes. Alongside improving computing efficiency and achieving 3D super-real-time computing, the generalization ability of the results obtained from the SM requires further verification. The reliability of the SM is

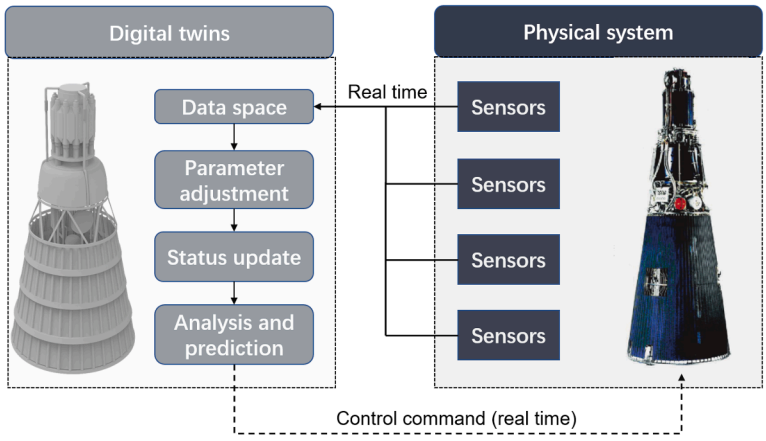


Fig. 1. Real time interaction between digital twins and the physical world.

Table 1
Summary of existing digital twins.

Year	Researcher	Model dimension	Real-time	Super-real-time	Work
2019	Zhou M [26]	1D	Yes	No	A DT has been proposed for real-time data online analysis of power grids. The presentation, discussion, and performance testingare based on a large-scale grid network model.
2020	T.I. Zohdi [27]	3D	Yes	No	Employed a variant of the genetic algorithm to fine-tune model parameters, enabling almost real-time fire simulation calculations through rapid alignment with experimental observations
2021	Gunasegaram D R et [28]	3D	Yes	No	Proposed the application of DT in metal additive manufacturing, and considered creating multiscale and multiphysics models and their SM, and explained the necessity of developing SMs through machine learning.
2022	Fahim M [23]	1D	Yes	Yes	Developed a 5G-NG-RAN assisted cloud-based digital twins framework to monitor the SCADA units of wind turbines. A machine learning pipeline is designed over the temporal CNN and kNN regression to forecast wind and power generation.
2022	Chen J [29]	3D	Yes	No	Used low/high-resolution and global/local field scanning data to construct DT and developed a neural network optimization mechanism to assist in designing the porosity and orientation of materials.
2022	Wang S [25]	1D	Yes	Yes	Constructed a single-fidelity SM using a calibration result from a multi-fidelity SM based on sensor data as input, to predict and visualize the quantities of interest in real-time
2023	He W et al [30]	3D	Yes	No	Introduced a digital twin method for monitoring mechanical structural performance, which includes a hybrid machine learning model with high computational speed and sufficient accuracy that can utilize sensor data to evaluate the structural performance in real-time;
2023	Galeazzi A [24]	1D	Yes	Yes	Developed a SM for an industrial amine washing process based on process simulation in Aspen HYSYS. The model was validated using real data collected during steady-state operation
2023	Yi Y [31]	1D	Yes	Yes	A real-time temperature prediction and degradation mode analysis method based on DT and LSTM is proposed, to describe the dynamic thermal behavior of LIB and identify parameters through calculation.

influenced by various limiting factors, including the quality and representativeness of the training data, the chosen model architecture, and parameter selection [38]. Therefore, when utilizing the outcomes derived from the SM, it is crucial to conduct a comprehensive analysis of their generalizability [39].

The training process of the SM is grounded in empirical data. By learning patterns and trends in the data, the SM captures the relationship between input and output. However, it does not directly explain the underlying physical laws [40]. Instead, it relies on mathematical representations of data and models to make predictions. Consequently, there are stringent requirements for the quantity and quality of data, making its application in certain fields challenging. For example, in the nuclear energy sector, operational data is severely limited [41], and acquiring and organizing data is prohibitively expensive. Moreover, nuclear power systems exhibit significant nonlinearity, particularly in space nuclear reactors (SNRs). To date, successful development and operation of SNRs have been achieved only by the United States and the former Soviet Union [42,43]. Traditional SM results possess uncertainties, which demand cautious handling when applied to safety considerations. On the other hand, for an accurate representation of the physical model, a DT should employ multiple high-dimensional system simulations. Relying solely on an SM would exponentially increase model complexity and decrease calculation accuracy [44,45], while incorporating multiple SMs significantly increases workload without providing reliable accuracy guarantees. To ensure result reliability, it is advisable to minimize the use of SMs, preserve the physical model, or adopt interpretable SMs while considering the interaction between the SM and the physical model.

To reduce the need for large datasets, improve model generalization, and increase result reliability, many researchers have recently begun investigating neural networks with physical constraints [46–49]. Physics-informed neural networks(PINNs) use these constraints to encapsulate the output's specific structure and qualities, which are known to hold due to domain knowledge, such as known physical laws like conservation of momentum, mass, and energy. The physical laws are written into the loss function, and the neural network performs many random samples in the discrete grid coordinate system. Theoretically, this method eliminates the need for raw data and grid partitioning to solve the equations. However, the solving speed of the PINN is too slow and cannot be used to solve complex transient systems. Although the PINN has been continuously improved in recent years [50–52], its problems have not been fundamentally resolved.

The proposal of the PINN has made researchers realize the need for additional constraints when constructing neural networks. Many researchers have started looking for alternative ways to constrain neural networks to accelerate the solution of PDE equations [53–55]. Discovering the rules from limited prior data is challenging, thus necessitating a PDE solving framework that can rapidly solve PDE equations under various initial and boundary conditions. DeepONet proposed in is one of the possible ways to learn the PDE solution operators from the labeled input–output datasets [56]. DeepONet is proposed based on the universal approximation theorem [57]. Both PINNs and traditional data-driven models solve problems by approximating functions. The extended universal approximation theorem states that a single hidden layer neural network can accurately approximate nonlinear continuous functionals and nonlinear operators. DeepONet learns the operator by inputting functions, thus realizing a mapping from an element in one vector space to an element in another vector space. Recently, many researchers have used the DeepONet framework to solve practical

engineering problems [58]. There are many versions of DeepONet [59–61], but this study employs the original DeepONet framework without modification.

This study proposes a 3D computational method to achieve super-real-time and high data performance in DTs, specifically applied to state monitoring, prediction, and decision-making in SNRs. A novel numerical calculation and neural network coupled mixed solution framework is presented, utilizing multi-core CPUs to accelerate the solution of solid equations and GPUs to load deep learning network models for solving fluid equations. Additionally, GPU and CPU information communication is employed to simulate the interaction between the fluid and solid regions. To enhance the generalization ability, the study introduces a network structure based on the DeepONet framework and classification thinking, a new data set generation method, and a modified loss function. These modifications result in improved generalization ability, reduced training time, and lower training difficulty. To validate the effectiveness of the proposed method, the TOPAZ-II reactor is selected as the research object, and data from non-training conditions, including multiple continuous power up/down, power changes with varying rates, and power scaling beyond the training conditions, are used to verify the model's generalization ability.

The main contributions of this article are summarized as follows:

1. A 3D computing method based on separation framework of GPU and CPU has been proposed. It combines numerical computation with deep learning to solve equations, thus achieving precise, comprehensive, super-real-time monitoring and prediction of system status;

2. A novel network structure utilizing the DeepONet framework and classification concept, a novel method for generating datasets, and a new loss function have been proposed. These modifications provide stronger generalization capabilities.

The paper is structured as follows: the proposed computing method for SNR's DT is presented in Section 2. The experiment design and results are given in Section 3 and conclusion are given in Section 4.

2. Methodology

2.1. Framework

The proposed method framework in this study is illustrated in Fig. 2, which can be generally categorized into three stages: (1) region separation; (2) solution based on a CPU and GPU separation architecture; (3) data transmission between a GPU and multiple CPUs. The method flowchart depicted in Fig. 3 presents a detailed illustration of these stages, which include the following steps:

(1) Firstly, the solid and fluid regions are separated in the CPU memory space, and the data space of the fluid region is loaded into the memory space of the GPU;
(2) Secondly, a parallel algorithm is employed to partition the solid region into multiple sub-regions, and the acceleration calculation of the solid region is achieved through multi-core CPUs. Additionally, the fluid region equations are solved using a deep learning network model, and the calculation of the fluid region is accomplished through GPU parallel computing;
(3) The data space for the fluid region in a GPU is partitioned into multiple subspaces and stored in distinct data cache spaces. As such, various CPUs read the corresponding data cache spaces and overwrite the fluid data space in the CPU, thus enabling information communication between the CPU and GPU;
(4) An evaluation is conducted on the CPU to determine whether the fluid and solid regions have reached convergence. If convergence has been achieved, the process will terminate. Otherwise, the previous steps will be iteratively repeated.

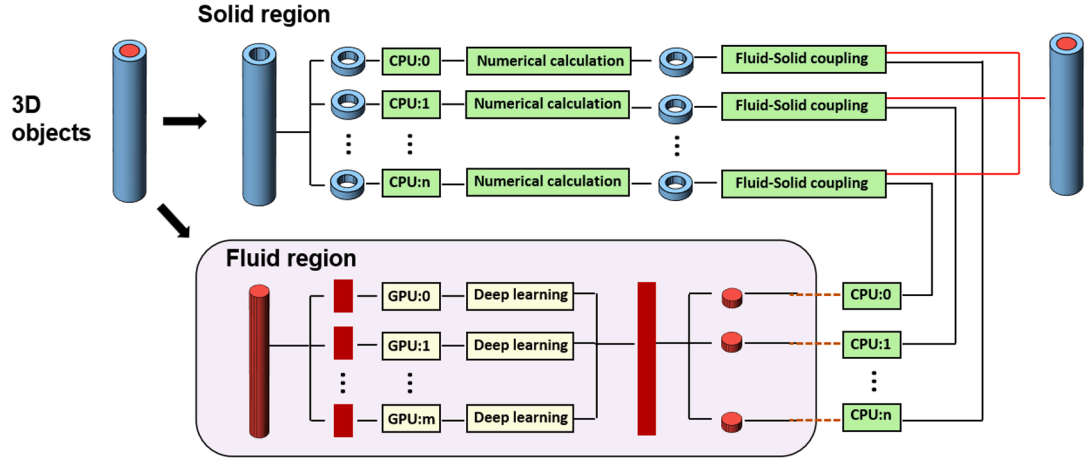


Fig. 2. Super-real-time computing framework.

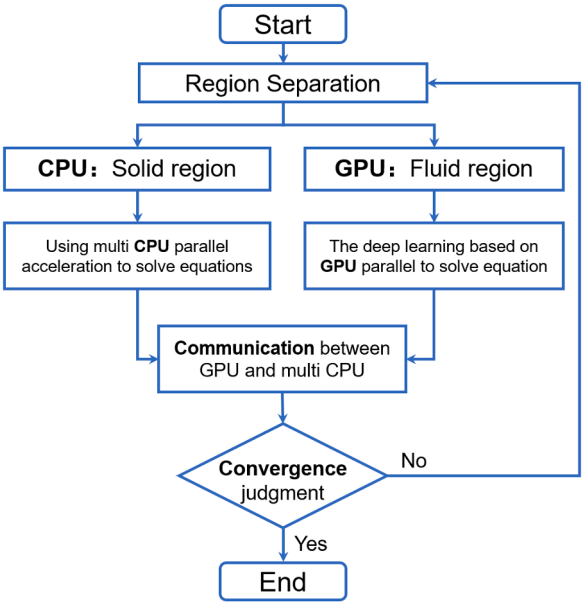


Fig. 3. Super-real-time computing method flowchart.

2.2. Deep learning for solving equations in fluid regions

This section will introduce how to construct and train deep learning networks to solve fluid region equations. Section 2.2.1 introduces a neural network non-linear operator called DeepONet, Section 2.2.2 presents the construction of a database for training deep learning networks. Section 2.2.3 presents the loss function for training deep learning networks.

2.2.1. Deep operator networks

The DeepONET framework is shown in Fig. 4. To approximate the operator, DeepONet uses the discrete values of function $u(x)$ as input for the branch network. This means that the numerical values of function $u(x)$ on a set of points $\{x_1, x_2, \ldots, x_m\}$, denoted by $\{u(x_1), u(x_2), \ldots, u(x_m)\}$, serve as the input for the branch network. The input y for the trunk network represents the independent variable of the operator G acting on u. Specifically, $G(u)(y)$ denotes the result obtained by the G when acting on function $u(x)$ under the condition of y. The universal approximation theorem states that the function of the operator G can be approximated by Eq. (1):

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{\text{branch}} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{\text{trunk}} \right| < \varepsilon \tag{1}$$

where: σ is a continuous nonpolynomial function, n,p,m are positive integers, $c_i^k, \xi_{i,j}^k, \theta_i^k, \zeta_k$ are constant. $x_j \in K_1$, $i = 1, \ldots, n$, holds for all $y \in K_2$, $u \in V$. X is a Banach Space, $K_1 \in X$, $K_2 \in R^d$ are two compact sets in X and R^d, respectively. V is a compact set in $C(K_1)$.

According to Eq. (1), neural networks can approximate any nonlinear operator G. However, this theorem does not provide insight into how to learn the nonlinear operator. In response to this challenge, DeepONet has emerged as a flexible framework that does not

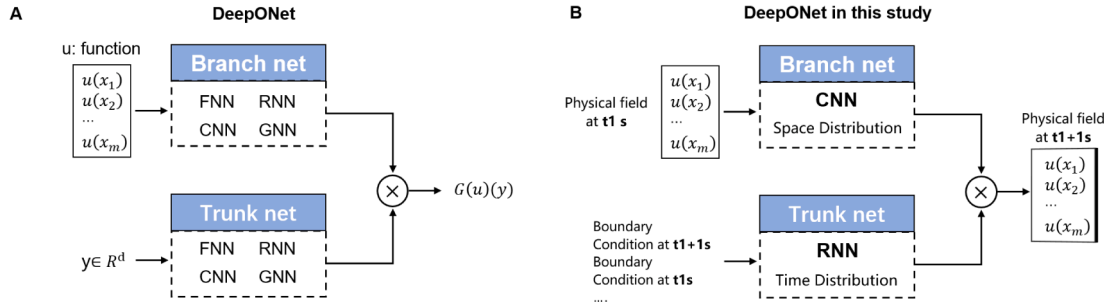


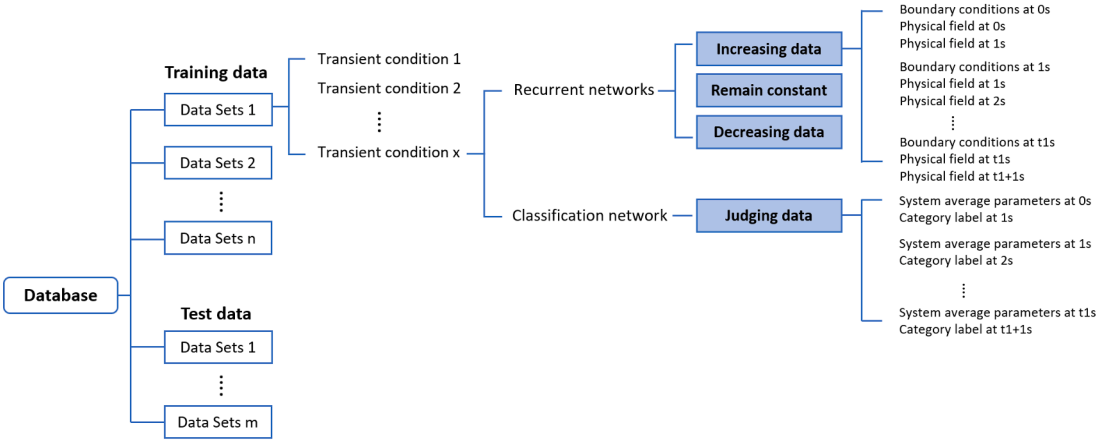
Fig. 4. DeepONet framework (left: standard DeepONet framework; right: framework used in this work).

impose any limitations on the structure of the branch and trunk networks. It is important to note that the effectiveness of learning the nonlinear operator depends heavily on the type of neural network employed. Utilizing different network structures may result in notable variations in the degree to which DeepONet can effectively learn the nonlinear operator [59,62]. The DeepONet used in this study incorporates a branch network that employs a convolutional neural network(CNN), while the trunk network uses an recurrent neural network(RNN). This architectural decision is motivated by the characteristics of the initial temperature field, which is a high-dimensional and data-dense distribution that exhibits spatial regularity. By adopting a CNN for the branch network, the model can more effectively acquire and learn information [60]. The trunk network employs an RNN to acquire more information and capture temporal patterns [61], which take as input the boundary conditions at different time points. Compared to using fully connected layers in the DeepONet, employing CNN and RNN enables the acquisition of more feature information, resulting in higher precision and better fitting capability, and reducing training complexity.

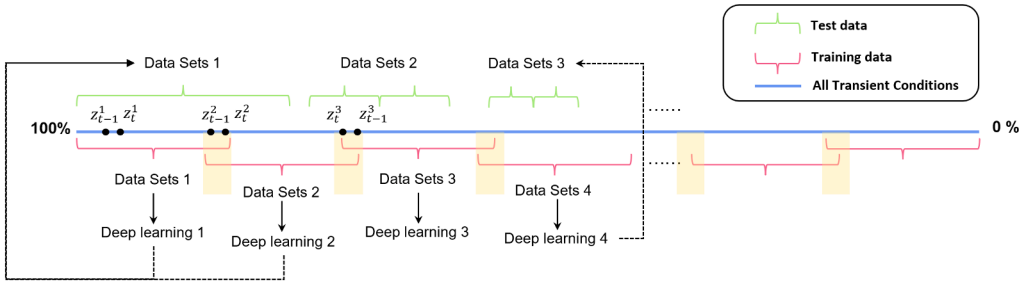
2.2.2. Database

Databases play a crucial role in the training of neural networks. For this research, data is obtained from a fully physics-based DT and is partitioned into various types to train distinct networks [63,64]. This approach effectively decreases the complexity and training difficulty of individual networks. By utilizing a dataset that encompasses a substantial amount of digital twin data, the aim is to further streamline the training process and alleviate complexities in training individual networks. The data in the database is divided into two primary categories: training data and testing data. The database comprises three levels: (1) The first level, which includes a collection of transient operational datasets; (2) The second level, which covers transient operational conditions; (3) The third level of the database consists of training data for the monotonic nonlinear operator DeepONet. The data is classified into three categories based on the average value of a physical field at the t + 1 s and the previous time step t s: **(1). Increasing data**: $(u_{t+1} > u_t)$ and $|u_{t+1} - u_t| > \xi$; **(2). Decreasing data**: $(u_{t+1} < u_t)$ and $|u_{t+1} - u_t| > \xi$; **(3). Constant output data:** $|u_{t+1} - u_t| < \xi$. Additionally, the data includes information that determines whether the output is increasing, decreasing, or unchanged. The relationship among these categories is depicted in Fig. 5.a. The methodology for constructing the training and validation sets is presented in Fig. 5.b.

A training set from the database is utilized in a deep learning network, whereas the validation set can involve multiple deep learning network. Specifically, this is achieved by utilizing several deep learning networks to simulate a broader range of transient conditions that are more complex. To validate the generalization capability of the deep learning network, the validation set should encompass transient conditions that are not present in the training set. The selection of the training set ought to be such that it is evenly sampled, ensuring comprehensive coverage of all transient conditions. The size of the training set is dependent on the types and scope of the transient conditions, as well as the size of the deep learning network and the training costs. To enable multiple deep learning



(a)



(b)

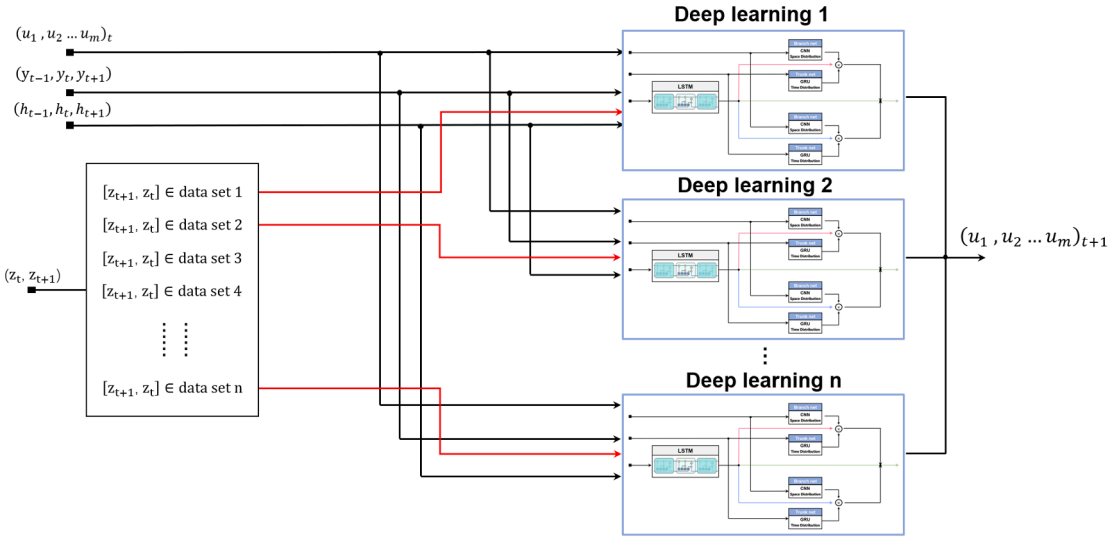
Fig. 5. Database structure and usage.

networks to collaborate more effectively, it is essential that the training data sets overlap with adjacent sets at their borders. Additionally, the density of transient cases at the interface of the training sets ought to be increased, thus ensuring enhanced efficacy of the network architecture.

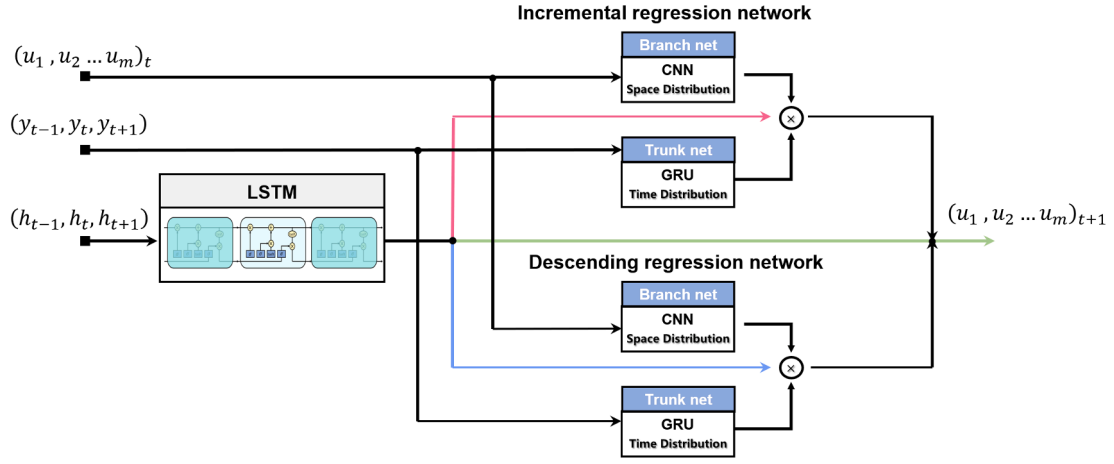
2.2.3. *The structure of deep learning networks*

Ensemble learning methods, which involve constructing and combining multiple learners, are recognized for their superior generalization performance compared to single learners. The two most common integrated learning methods are Boosting [65] and Bagging [66]. Additionally, recent years have seen the emergence of novel ensemble learning approaches, such as those based on D-S evidence theory [67] and hybrid expert systems [68] . Among these, Mixture of experts (MoE) have gained widespread use for large-scale training [69,70], employing a combination of gating and expert models. When a set of inputs is received, the system selects the trusted expert model and consolidates the predicted results. In this work, the deep learning operator network framework proposed in this study embraces the MoE concept. Transient conditions are characterized by significant differences arising from variations in initial states and computational parameters to train multiple deep learning networks and demonstrate that a variety of transient conditions can be simulated using either a single or multiple deep learning networks.

The deep learning network architecture proposed in this study is shown in Fig. 6.a. Where z_t represents the system parameters at time **t**, which are used to determine the deep learning network to be used at that time. h_t represents another set of system parameters at



(a)



(b)

Fig. 6. Deep learning network structure.

time t that are used to determine the relationship between the average value of the physical field at the time t + 1 and that at the time t. This relationship is then used to decide whether to employ an increasing regression network, a decreasing regression network, or assume that the physical field remains unchanged. The structure of the deep learning network is shown in Fig. 6.b. The code first determines which deep learning network to use, and then transfers the data to the selected network. Next, the LSTM network [71] receives a time series of (h_{t-1}, h_t, h_{t+1}) and determines which DeepONet framework to use or assumes that the physical field remains unchanged.

The fundamental criterion for selecting the appropriate neural network for the program is based on whether the training set of the chosen deep learning network contains the system state at both time t and t − 1. Fig. 5.b illustrates that the (z^1_{t-1}, z^1_t) data points should be utilized for training deep learning network 1. There is an intersection between adjacent training sets. When the system state at time t and t − 1 are both in the intersection space of the sets, a decision should be made based on the direction of the system state changes at times t and t − 1. For example, as shown in Fig. 5.b, (z^2_{t-1}, z^2_t) should be processed using Deep Learning Network 2, while (z^3_{t-1}, z^3_t) should also be processed using Deep Learning Network 2. To reduce the error caused by neural network noise, this study uses a relaxation factor to adjust the calculated results. The equation is as follows:

$$u_{t+1} = a \times u_{t+1} + (1 - a) \times u_t \tag{2}$$

where: a is the relaxation factor, which takes values between 0 and 1. The value determines the algorithm's sensitivity. A larger value leads to a more sensitive algorithm but lower smoothness, whereas a smaller value leads to a less sensitive algorithm but higher smoothness.

Gathering all the data for operating conditions and training the deep learning operator networks is an extensive undertaking. This work demonstrates this model's capability to solve the fluid region equations effectively. This necessitates validating two key points: firstly, that a single deep learning operator exhibits high accuracy and generalization ability within its training case, and secondly, that two deep learning operators can jointly simulate a non-training lift-power case. Once these two points are established, it can be inferred that with sufficient data, our integrated model can efficiently compute the vast majority of working conditions in super-real time while maintaining only a small loss of accuracy.

2.2.4. Loss function

The loss function of the increasing/decreasing neural network comprises three components: (1) the cumulative error f_{sum}, which is the total sum of absolute value deviations between the physical field obtained by real-time calculation and the physical field obtained by DT calculation at discrete grid points; (2) the average and maximum absolute value deviations, denoted as f_{ave} and f_{max}, respectively. (3) the penalty for violation of monotonicity in the super-real-time calculation, denoted as f_{punish}, and its expression is as follows:

$$f = f_{\text{sum}} + a_1 f_{\text{ave}} + a_2 f_{\text{punish}} \tag{3}$$

where: a_1 and a_2 are constant coefficients. The values of a_1 and a_2 are dependent on the batch size used during network training. To be more precise, the difference in magnitude between f_{ave}, f_{max}, and f_{sum} is considered when determining the values of these coefficients. Batch size refers to the number of samples simultaneously inputted into the network during one weight update. Choosing an appropriate batch size can significantly impact training speed and enhance the model's generalization ability. It should be noted that the batch size setting leads to a notable disparity between the cumulative error, denoted as f_{sum}, and the average and maximum absolute value deviations, represented as f_{ave} and f_{max}, respectively. To rectify this discrepancy, coefficients a_1 and a_2 must be introduced to align the magnitudes of f_{ave} and f_{max} with that of f_{sum}. The f_{punish} depends on the relationship between the predicted results and the input results. For example, in the increasing neural network, if the average value of the physical field input to the neural network is smaller than the average value of the physical field output from the neural network, f_{punish} is 0; otherwise, f_{punish} takes a very large value.

The loss function of the neural network is determined by using the cross-entropy loss function. In machine learning, the cross-entropy function measures the difference between the true probability distribution and the predicted probability distribution, and is widely used to train classification models [72]. The smaller the value of cross-entropy, the better the prediction performance of the model. The expression of the cross-entropy function is as follows:

$$f = \frac{1}{N}\sum_i f_i = -\frac{1}{N}\sum_i \sum_{c=1}^{M} y_{ic} log(p_{ic}) \tag{4}$$

where N is the number of samples, M is the number of classes, y_{ic} is a sign function, where y_{ic} equals 1 if i equals c, and 0 otherwise. p_{ic} is the probability of judging i as c.

2.3. Multi-core CPU parallel acceleration algorithm

A region decomposition method divides the problem of solving the solid region equations into multiple subproblems and then solves these subproblems in parallel on different computing nodes. This decomposition aims to speed up the overall solution process by distributing the computational load evenly across multiple processors or computers, as shown in Fig. 7. Common regional decompositions for parallel solving are [73]: 1. Regional decomposition based on spatial decomposition; 2. Regional decomposition based on temporal decomposition; 3. Regional decomposition is based on physical process decomposition. In this study, based on the

OpenFOAM parallel framework [74,75], the region decomposition method based on spatial decomposition is used to solve the solid region equations, and the information interaction is realized with the GPU. According to the decomposition of the solid region, the GPU divides the fluid domain accordingly and overwrites the fluid domain data in the corresponding CPU memory. The OpenFOAM parallel framework is implemented based on OpenMPI. Based on OpenMPI, it is not only able to run on a cluster [76], but also able to run on a single machine with multiple cores, which makes the code more expandable [77].

3. Experiment

3.1. Model description

The DT in this study was developed based on the former Soviet TOPAZ-II reactor. TOPAZ-II is a well-established space nuclear power source that underwent extensive research during the TOPAZ International Program (TIP) in the 1990s [78]. The reactor core consists of 37 thermionic fuel elements that directly convert thermal energy into electrical energy to power spacecraft. Waste heat that is not converted into electrical energy is dissipated into space through a loop-type radiative cooler. The DT focuses on the improved TOPAZ-II reactor system that replaces the traditional pumped-loop radiator with a heat pipe radiator to overcome the single-point failure phenomenon. Fig. 8 respectively show schematic diagrams of the improved TOPAZ-II system and the thermionic fuel elements [79]. The fuel generates heat through neutron fission, which in turn heats the emitter and receiver, leading to the generation of electrical energy. The surplus heat is then effectively dissipated by a flow of Nak coolant.

This work used a transient coupling code for the TOPAZ-II reactor based on OpenFOAM as a reference. The reactor neutron physics model, multi-region heat transfer model, thermoelectric conversion model, electromagnetic pump model, and heat pipe radiator model are included. A detailed 3D reactor core model was built and used in the developed code. A 2D heat pipe model using the finite element method is applied to model the heat pipe radiator. The boundary conditions of reactor core are shown in Fig. 9:

The energy conservation equations of the solid region can be written as follow:

$$\frac{\partial(\rho h)}{\partial t} = \frac{\partial}{\partial x_j}\left(\dot{\alpha}\frac{\partial h}{\partial x_j}\right) \tag{5}$$

where $\dot{\alpha}$ is the thermal diffusivity, h is the total energy of element, ρ is the density, x is the direction vector, t is time. The heat flux entering one region at one side of the interphase should be equal to the heat flux leaving the other region on the other side of the domain, and the grid temperature on the interface is the same, as follows:

$$\kappa_f \frac{dT_f}{d\vec{n}} = -\kappa_s \frac{dT_s}{d\vec{n}} \tag{6}$$

where $\vec{n}$ represents the direction normal to the wall. k_f and k_s are the thermal conductivities of the fluid and solid, respectively. The thermal power of the reactor is generated by neutron fission in the fuel, which can be obtained by solving the point kinetics equations. Assuming a constant spatial distribution of reactor power [80], a function expansion method can be employed to map the zero-dimensional power distribution to a three-dimensional unstructured grid [81]. In this code, the axial and radial mappings of the power distribution are achieved using Zernike functions and Legendre functions, respectively. The point-kinetics equations may be expressed as follows:

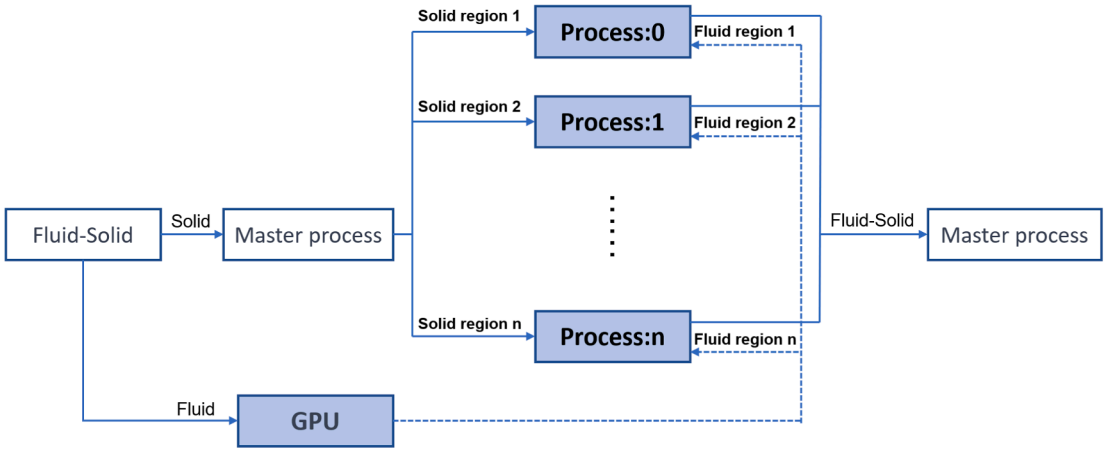


Fig. 7. Code structure and basic framework of MPI.

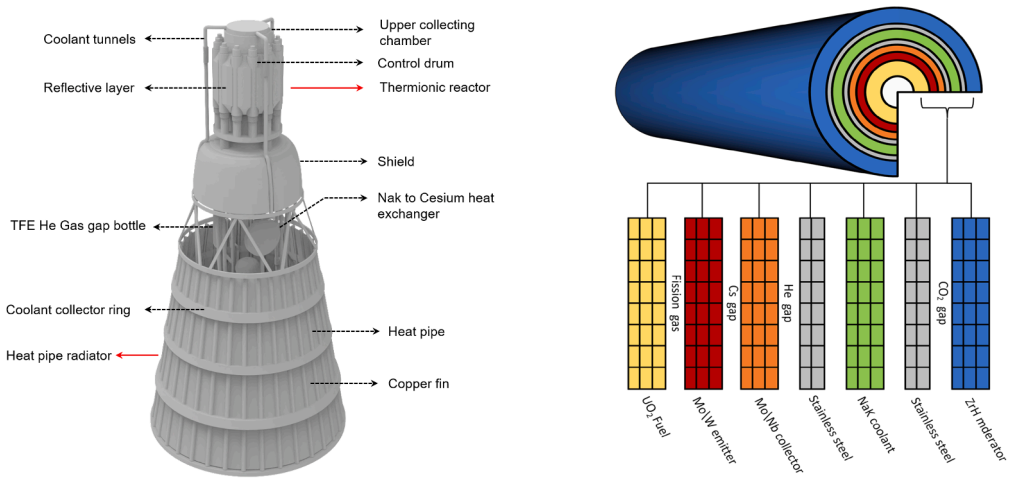
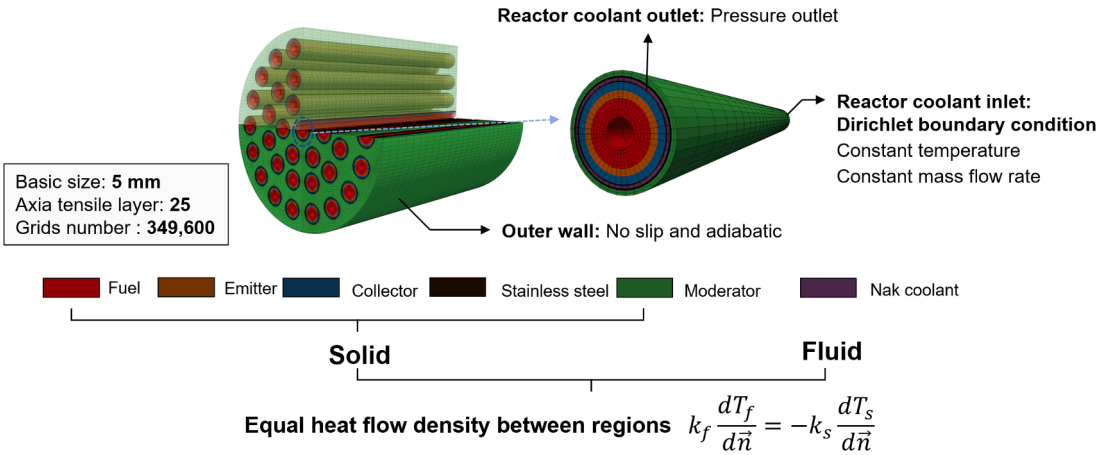
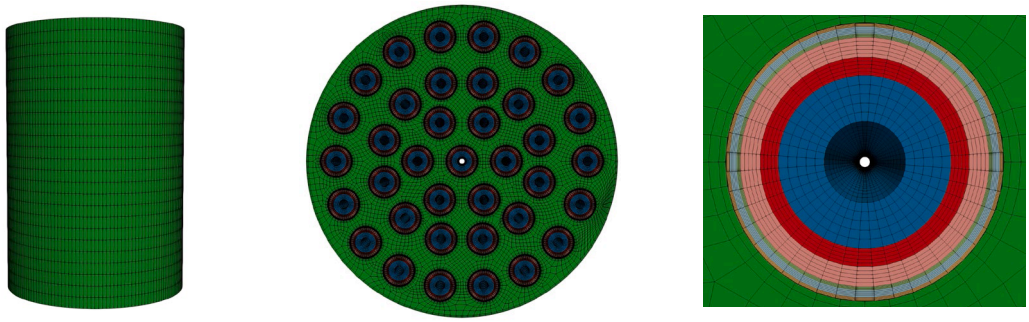


Fig. 8. The improved TOPAZ-II system and the high-dimensional model.



a. Boundary conditions of reactor model



b. Axial mesh c. Radial mesh

Fig. 9. Boundary conditions and mesh grid of reactor model.

$$\frac{dn(t)}{dt} = \frac{\bar{\rho}(t) - \beta}{\Lambda} n(t) + \sum_{i=1}^{6} \lambda_i C_i(t) \tag{7}$$

$$\frac{dC_i(t)}{dt} = \frac{\beta_i}{\Lambda} n(t) - \lambda_i C_i(t) \quad (i = 1, 2 \ldots 6) \tag{8}$$

where n is the neutron flux density; $\bar{\rho}(t)$ is the total reactivity; β is the delayed neutron share; Λ is the neutron generation time; C_i is the ith delayed neutron precursor concentration; t is the time; and λ_i is the ith group decay constant of the precursor. The neutron flux density is proportional to the fission power; therefore, the total reactivity $\bar{\rho}(t)$ is related to the fission power according to Eq. (7). The total reactivity $\bar{\rho}(t)$ is affected by many physical variables, and it is given as follows:

$$\bar{\rho}(t) = \bar{\rho}_{\text{external}}(t) + \sum \bar{\rho}(t) \tag{9}$$

where $\bar{\rho}_{external}(t)$ is the external reactivity and $\sum \bar{\rho}(t)$ is the sum of the reactivity feedback caused by various physical phenomena. Among them, the Doppler broadening effect is transient and plays a significant role in reactor safety. The Doppler reactivity $\bar{\rho}_{doppler}$ caused by the Doppler broadening effect can be written as follows:

$$\bar{\rho}_{\text{doppler}} = 1 - \left(1 - \bar{\rho}_{ref}\right) e^{\alpha_T ln\left(\frac{T_{fuel}}{T_{ref}}\right)} \tag{10}$$

where $\bar{\rho}_{ref}$ is the reference reactivity; α_T is the Doppler coefficient; T_{ref} is the reference temperature; and T_{fuel} is the fuel temperature. According to Eq. (10), the Doppler reactivity will be introduced when the fuel temperature deviates from the reference temperature.

3.2. Database: training data and test data

The TOPAZ-II modifies its power output by introducing reactivity through rotating control drums. This study presents two training data sets, which are displayed in Table 2. There are two data sets for power reduction: (1) starting at 100% power, with the final stabilized power ranging from 90% to 70% after the introduction of reactivity; (2) starting at 75% power, with the final stabilized power ranging from 70% to 60% after reactivity is introduced. Reactivity is introduced linearly over a period of 20 s.

To demonstrate the generalizability of the proposed deep learning network, it is crucial that the selection of the validation dataset differs from that of the training dataset. Specifically, in addition to the transient operating condition types included in the training set, the validation set should also encompass four distinct types of transient operating conditions: (1) Multiple continuous power up/down; (2) Power changes with different up/down rates; (3) Scaling up/down of power beyond the training conditions; (4) Simulating multiple consecutive power ramps by using multiple deep learning networks, as shown in Table 3.

The Trunk network of the incremental/decremental neural network is conditioned on the boundary conditions at time t + 1, t, and t − 1, corresponding to the thermal power, mass flow rate, and average coolant temperature at inlet. Meanwhile, the Branch network is conditioned on the coolant temperature field at time t. The decision network takes the total reactivity, Doppler reactivity, and the running time after introducing external reactivity at time t + 1, t, and t − 1 as input and outputs whether to use the incremental or decremental neural network or to assume that the coolant temperature field at time t + 1 is the same as that at time t.

3.3. Super real time code framework

The super real-time computational framework presented in Fig. 10 differs from the traditional numerical DT by modifying the multi-region heat conduction model to divide the computation domain into solid and fluid regions. The computer configuration used in this work consists of an Intel 13th Generation Core i9 13900KF CPU, 64 GB of RAM, a GeForce RTX 4090 24 GB GPU, and the Ubuntu 20.04 operating system. It is worth noting that super real-time computing requirements still place stringent demands on the CPU and RAM, while the GPU need not be particularly high-end. During program execution, it was observed that GPU memory usage was less than 8%, suggesting that the GPU configuration could be further optimized. However, a powerful GPU can significantly reduce the

Table 2
Training data set.

Data set	Initial power	Stable power	Data set	Initial power	Stable power
Data sets 1	115 kW	103.50 kW	Data sets 2	86.25 kW	82.10 kW
(100% to [90%,70%])	115 kW	100.63 kW	(75% to [70% 60%])	86.25 kW	78.89 kW
	115 kW	94.88 kW		86.25 kW	74.96 kW
	115 kW	89.13 kW		86.25 kW	71.81 kW
	115 kW	83.38 kW		86.25 kW	69.01 kW
	115 kW	80.50 kW			

Data set	Incremental neural network		Descending neural network		Judgment neural network	
Data sets 1	Input: 704 × 3 × 3,	Output:	Input: 2035 × 3 × 3,	Output:	Input: 3835 × 3 × 3	Output:
(100% to [90%,70%])	704 × 255 × 11	704 × 255 × 44	2035 × 255 × 11	2035 × 255 × 44		3835 × 1
Data sets 2	Input: 507 × 3 × 3,	Output:	Input: 1237 × 3 × 3,	Output:	Input: 3592 × 3 × 3	Output:
(75% to [70% 60%])	507 × 255 × 11	507 × 255 × 44	1237 × 255 × 11	1237 × 255 × 44		3592 × 1

Table 3
The validation data set.

Data set	Thermal power	Time	Data set	Thermal power	Time
Data set 1	115 kW – 97.75 kW	20 s	Data set 3	115 kW – 95.78 kW	10 s
	115 kW – 92.15 kW	20 s		115 kW – 95.78 kW	40 s
	115 kW – 87.22 kW	20 s		115 kW – 95.78 kW	60 s
Data set 2	115 kW – 103.13 kW – 83.16 kW	20 s		115 kW – 95.78 kW	80 s
	115 kW – 97.75 kW – 82.8 kW	20 s	Data set 4	115 kW – 70.63 kW	20 s
	115 kW – 96.60 kW – 88.55 kW	20 s		115 kW – 80.5 kW – 78.87 kW	20 s

time required for training deep learning networks. The number of processors impacts computational efficiency, and a sensitivity analysis of the number of processors needs to be carried out. In this work, 20-core CPUs are used.

The workflow of this part can be summarized as follows:

1. First, according to the total reactivity, the reactor neutron physical model first obtains the power and delayed neutron precursor concentration by solving the point reactor kinetic equation. Then, the zero-dimensional thermal power is mapped onto a three-dimensional unstructured grid by the function expansion method.

2. Then, the reactor temperature field is solved, which is roughly divided into 3 steps: (1). CPU:0 passes the boundary conditions and the temperature field of the fluid region at times $t + 1$, t, and $t - 1$ to the deep learning network on the GPU through the API interface. After solving the equation, the deep learning network obtains the temperature field of the fluid region at time $t + 1$ and saves it in a file. (2). CPU:0 sends a signal, and the other CPUs exit their waiting state upon receiving the signal. Second, CPUs 0–19 access the file, cut it, unlock it, and read the relevant data to overwrite the fluid region data space on CPUs 0–19. (3). Solving solid equations in a multi-core CPU-based framework using numerical analysis methods. Fig. 11 presents the inputs and outputs of the framework.

3. Next, the thermoelectric conversion model calculates the electric power based on the initial temperature distribution. The code extracts the average fuel temperature and updates the Doppler reactivity.

4. Furthermore, the code provides a judgment. If the output current exceeds the set value, the code will update the mass flow using the electromagnetic pump module. Otherwise, it will keep the default value. Then, the code uses the heat pipe radiator model to obtain the average coolant temperature at the reactor inlet.

Finally, the code judges whether the calculated results satisfy the convergence criterion. If the result meets the convergence criterion, it will output the result, and the calculation of the next time step will start; if not, the calculation within the time step will be repeated.

Fig. 12 provides super real-time DT calculation results under a typical transient operating condition with multiple fluid–solid coupling iterations within a single time step. The results indicate that convergence can be achieved by using multiple couplings. However, the convergence result only deviates slightly from the initial result, with a maximum deviation of no more than 0.25%. The average time required for a single coupled iteration is approximately 1e−3 units.

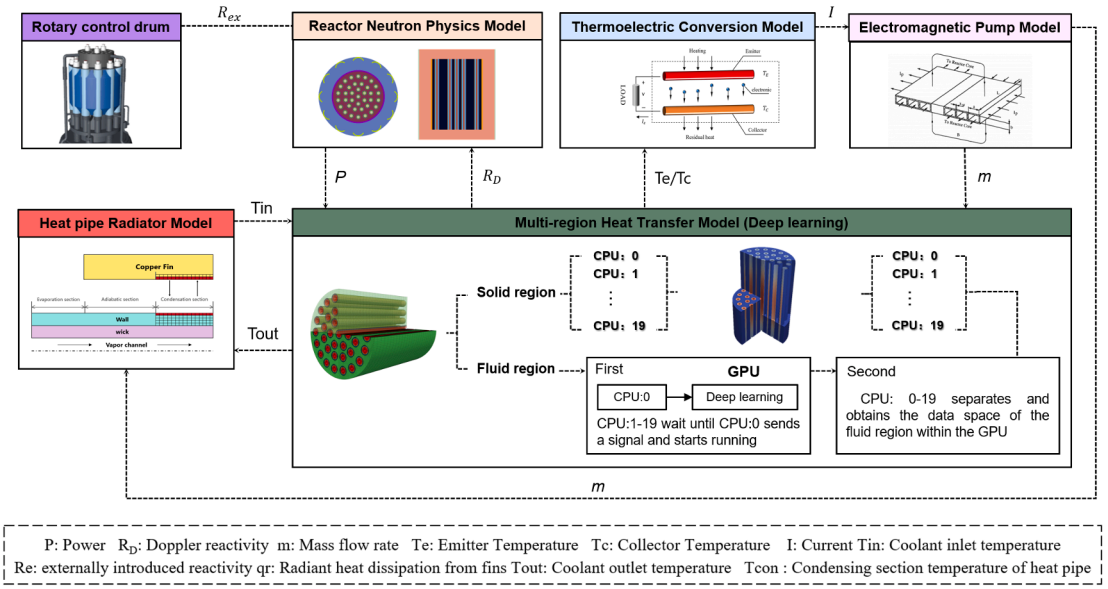


Fig. 10. Super-real-time computing framework.

3.4. Deep learning operator

The network architecture used in this study is depicted in Fig. 6, and the specific parameters of the deep learning network structure are listed in Table 4. Details of the training setup, including learning rate, epochs, optimizer, and training time for the models, are provided in Table 5. The parameter settings are obtained through sensitivity analysis and continuous adjustment. All models are trained on GPU for optimized performance. In the proposed framework, the number of grids significantly impacts computations in the solid and fluid–solid regions. Increasing the number of grids leads to higher CPU workload and requires additional memory and graphics memory for CPU–GPU information exchange. However, deep learning operator networks are minimally affected by the number of grids due to effective data complexity reduction through pooling and interpolation methods, resulting in only minor precision loss that has little effect on the final results. The framework employs a maximum pooling layer for dimensionality reduction of the input temperature field and a convolutional neural network to extract important feature parameters to improve the model's accuracy.

The Trunk network comprises a hybrid of a GRU network and an FNN network, where the former receives time series information, while the latter models the relationship between the time series and the output data. In comparison to LSTM, GRU has fewer parameters, leading to faster training speed and greater computational efficiency [82]. Nonetheless, in certain tasks, LSTM may outperform GRU, particularly when processing longer sequences or intricate contextual relationships. Hence, the LSTM is used for the judgment network, and GRU is utilized for the trunk network.

The deep learning operator takes input from three components:

- A 3×3 matrix comprising external reactivity, Doppler reactivity, and runtime after reactivity introduction at $t-1$, t, and $t+1$;
- A 3×3 matrix containing reactor inlet temperature, mass flow rate, and thermal power at $t-1$, t, and $t+1$;
- A downscaled coolant temperature field at t represented as a 255×11 matrix.

The first component serves as input to the judgment neural network, while the last two components act as inputs to the regression neural network. The ultimate output is a 255×44 matrix representing the coolant temperature field at $t+1$.

The results of the sensitivity analysis of the relaxation factor are given in Fig. 13, with transient conditions 115 kW–92.75 kW as the benchmark and the average temperature of inner stainless steel as the object. This is to consider that the fluid region greatly influences the average temperature of inner stainless steel, and its deviation from the reference value in the calculation is large. The relaxation factors for the increasing and decreasing neural networks are 0.7 and 0.3, respectively, considering the smoothness of the curves and the deviation from the reference values.

3.5. Results and discussions

The parameters listed below are utilized as indicators to evaluate the results of solving transient fluid region equations using deep learning networks. The DT based on traditional numerical methods is denoted as DT_{tra} using symbols, and the super-real-time DT is denoted as DT_{super}.

(1) Maximum/Average system parameter error.

Apart from the fluid region, the maximum and average errors between the system parameters obtained through super-real-time code and their respective reference values at each time point can be determined using the formula provided below:

the maximum errors of system parameters Γ_{max}:

$$\left\{\left(|\omega_1-\omega_2|_{t=0}\right)_{max},\left(|\omega_1-\omega_2|_{t=1}\right)_{max},\ \ldots.,\ \left(|\omega_1-\omega_2|_{t=T}\right)_{max}\right\}_{max} \tag{11}$$

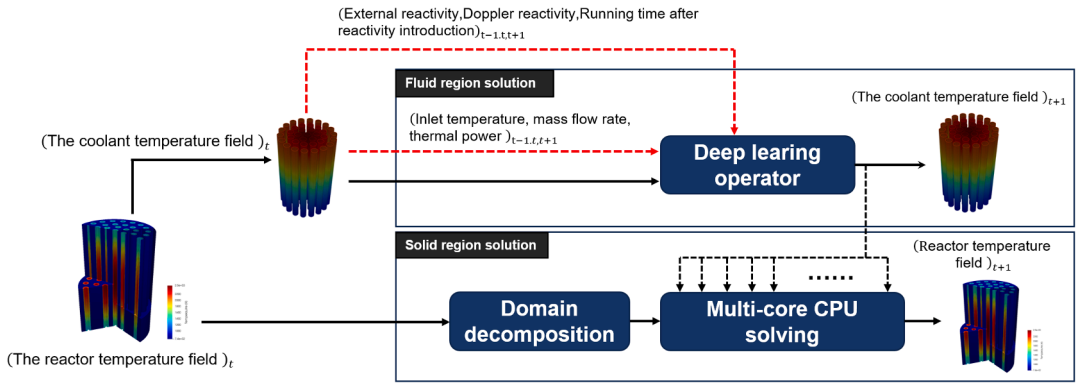


Fig. 11. Inputs and outputs of the framework.

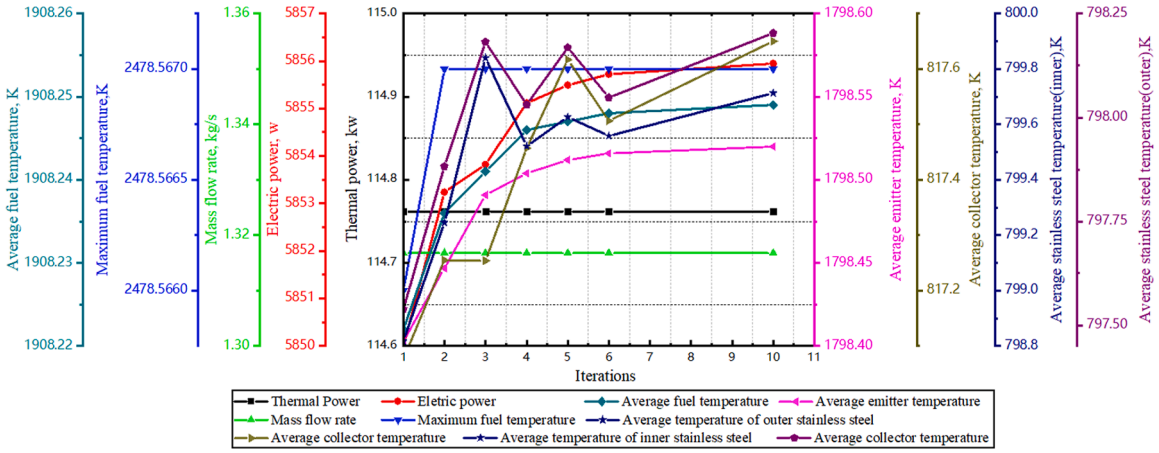


Fig. 12. Multiple iteration results for fluid–solid coupling of DT.

Table 4
Parameters of deep learning networks.

Network type	LSTM	FNN
Judgment neural network	Input: 3 × 3 Output: 200 Number in each layer: 200 LSTM layers: 4	Input: 200 Output: 3 Structure: 200 × 500 × 500 × 3 Number of network layers: 3
Incremental neural network	Branch	Trunk
	1. Maximum pooling layer maximum pooling window: (5, 6) move Step Length: (5, 6) **2. Downsampling** multiplier for spatial size: 5 bilinear interpolation **3. FNN** Structure: 102 × 500 × 102	**GRU** Input: 3 × 3 Output: 150 Number in each layer: 150 GRU layers: 4 **FNN** Input: 150 Output: 102 Structure: 150 × 500 × 1000 × 1000 × 1000 × 102 Number of network layers: 5
Descending neural network	Branch	Trunk
	1. Maximum pooling layer maximum pooling window: (5, 6) move Step Length: (5, 6) **2. Downsampling** multiplier for spatial size: 5 bilinear interpolation **3. FNN** Structure: 102 × 500 × 102	**GRU** Input: 3 × 3 Output: 150 Number in each layer: 150 GRU layers: 4 **FNN** Input: 150 Output: 102 Structure: 150 × 500 × 1000 × 1000 × 1000 × 102 Number of network layers: 5

Table 5
Training setup of deep learning networks.

Model	Learning rate	Optimizer	Epochs	a/a1/a2	Training/ Testing	Batch size	Training time
Incremental network	Initial value: 1e−4 After 2500 iterations: Learning rate = Learning rate × 0.8	Adam	60 000	0.7/1000/500	0.8/0.2	100	70 min 32 s
Decreasing network	Initial value: 5e−4 After 2500 iterations: Learning rate = Learning rate × 0.8	Adam	80 000	0.3/1000/500	0.8/0.2	64	346 min 40 s
Judgment network	Initial value: 1e−2 After 5000 iterations: Learning rate = Learning rate × 0.5	Adam	50 000		0.85/0.15	320	19 min 44 s

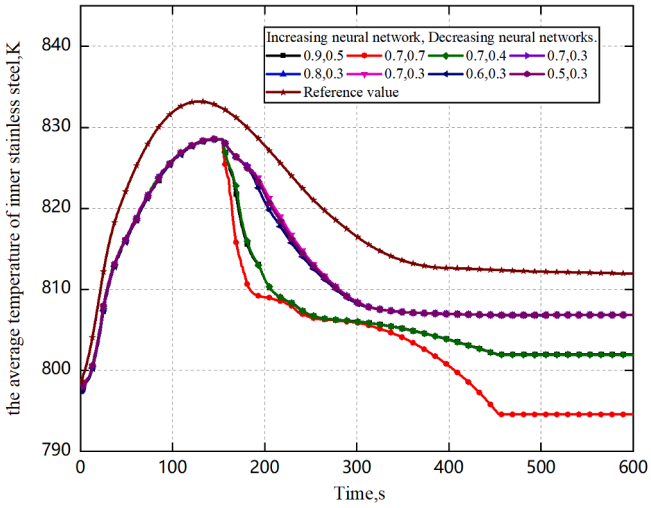


Fig. 13. Average temperatures of inner stainless steel obtained from calculations at different relaxation factors as a function of time.

the average errors of system parameters Γ_{ave}:

$$\frac{\sum_{t=0}^{T}(\omega_1-\omega_2)_t}{T} \tag{12}$$

where ω_1 represents the system parameters obtained through DT_{super}, ω_2 represents the reference system parameters through DT_{tra}, and T represents the total operating time. The system parameters encompass several variables, namely: **A.** thermal power; **B.** maximum fuel temperature; **C.** average fuel temperature; **D.** average emitter temperature; **E.** average collector temperature; **F.** average temperature of inner/outer stainless steel; **G.** average moderator temperature; **H.** output electric power; **I.** mass flow rate; **J.** voltage; **K.** current.

(2) Maximum/Average deviation of the peak time

The maximum/average deviation between the time at which each system parameter reaches its peak value can be expressed as follows:

Maximum deviation of the peak time T_{max}:

$$\left\{\left(|T_1-T_2|_{m=1}\right)_{max},\left(|T_1-T_2|_{m=2}\right)_{max},\ldots,\left(|T_1-T_2|_{m=M}\right)_{max}\right\}_{max} \tag{13}$$

Average deviation of the peak time T_{ave}:

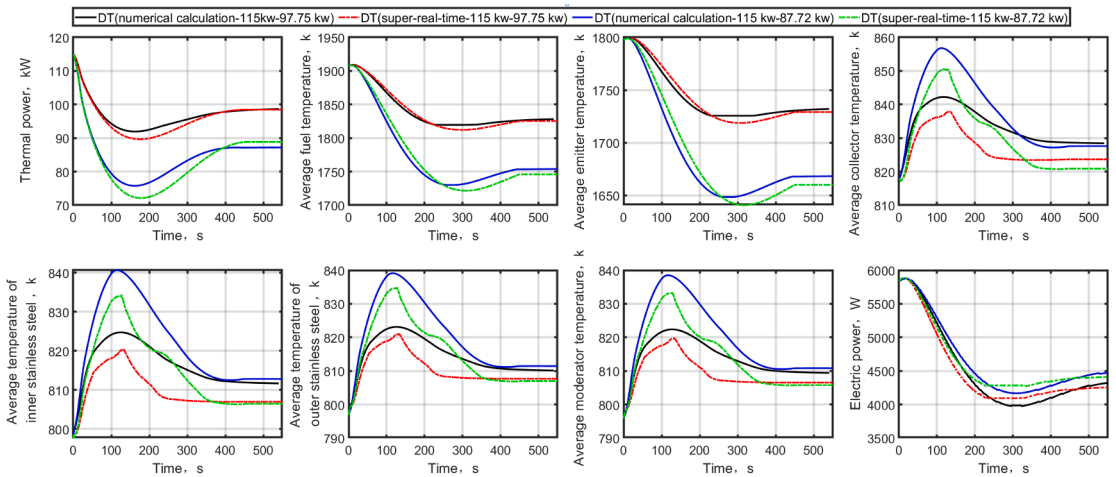


Fig. 14. The variation of reactor system parameters over time (97.75 kW and 87.22 kW).

$$\frac{\sum_{m=1}^{M}(T_1 - T_2)_m}{M} \tag{14}$$

In the formula, T_1 is the array of peak times of each system parameter obtained through DT_{super}, T_2 is the array of peak times of each system parameter obtained through DT_{tra}, and M represents the number of peak values that occur during a transient condition.

(3) The maximum/average error of the fluid region grid U_{max}/U_{ave}

The maximum absolute deviation and average deviation between the fluid region grid obtained through super-real-time code and the reference value at each time point can be expressed using the same formulas as those in Eqs. (5) and (6).

(4) Time acceleration ratio. γ

The time acceleration ratio γ is defined as the ratio of the time T_{real} required for a transient condition to occur in the real physical world to the time T_{code} required to simulate the same transient condition using DT_{super}. γ greater than 1 indicates that the code is running faster than real-time, and a higher value of γ corresponds to a more effective acceleration of the simulation.

(1) Introducing external reactivity reduces thermal power

This study simulated the transient conditions of data set 1 in the validation data set, and it is worth mentioning that these transient conditions type are consistent with those in the training set. Fig. 14 illustrates the time-varying average parameters of the reactor system within the validation dataset data1, specifically for power ranges of 115 kW–97.75 kW and 115 kW–87.22 kW. Meanwhile, Table 6 provides an overview of the evaluation metrics for data1 in the validation dataset.

Fig. 14 demonstrates that the curve derived from the super-real-time computing code is essentially congruous with the trend of the curve acquired from DT_{tra}, albeit exhibiting some irregularity. This deviation can be attributed to the presence of errors in the deep learning network model, which tend to propagate during computation. Nevertheless, the overall magnitude of the error remains within an acceptable threshold. Table 6 indicates that the maximum deviation in the thermal parameters of the reactor, denoted as Γ_{max}, is no more than 6%. Similarly, the average error, Γ_{ave}, is less than 1% and the average error in the power parameters is below 2.5%. Moreover, the maximum deviation in the peak time, Γ_{max}, is less than 13 s. Importantly, the time acceleration ratio, γ, for all operational conditions is greater than 1, attesting to the successful realization of super-real-time computing. Conventional numerical methods require nearly 12 h of computation on a 20-core CPU, to perform the same calculation. In contrast, the super-real-time computing code proposed in this study substantially reduces the time cost while maintaining a minimal loss of precision, and its solving speed has been increased by approximately 60 times.

In the transient condition ranging from 115 kW to 87.22 kW, Table 6 shows a maximum deviation of the mass flow rate is 10.58%. This deviation is influenced by various parameters, including thermal power, emitter temperature, and collector temperature, resulting in relatively significant deviations in the mass flow rate. However, the average deviation is less than 2.5%. The mass flow rate is a sensitive parameter. It exhibits a maximum deviation of over 10% between numerical analysis-based code calculations and experimental measurements, highlighting the necessity for alternative methods to accurately estimate its true value and determine the reactor system's actual state.

(2) Multiple continuous power up/down

Deep learning network 1 was developed using training data set 1, where the initial power was set at 115 kW. By introducing a reactivity, the power level is lowered to the range of 90%–70% of the initial power. The validation set, data set 2, was used to assess the generalization performance of deep learning network 1. It incorporates two reactivities that lead to a continuous reduction in power. Fig. 15 displays the average system parameters over time for data set 2, which are 115 kW–96.60 kW–88.55 kW and 115 kW–92.49 kW–82.72 kW. Fig. 16 illustrates the temperature distribution of the reactor over time for the transient operating condition of 115 kW–97.75 kW–82.8 kW, obtained from the super-real-time computing code and OpenFOAM solver. Table 7 summarizes the evaluation metrics for data set 2.

It can be seen form Fig. 15 that the error after the introduction of the second reactivity is slightly larger than that after the introduction of the first reactivity, but it remains within an acceptable range and is consistent with the curve obtained from DT_{tra}. Fig. 16 shows that the temperature distribution map obtained by the super-real-time computing code is free from significant deviations, and the temperature distribution of the reactor/coolant changes reasonably versus time. Compared to the coolant temperature field obtained by the OpenFOAM solver, the distributions are partially biased, but the general distribution is consistent. It can be observed from Table 7 that the maximum system error is less than 10%, the average error is less than 1.5%, the maximum time deviation is 15 s, and the time acceleration ratio is greater than 1. The super-real-time computing code achieves super-real-time calculation, and from the evaluation metrics, the calculation deviation of data set 2 is similar to that of data set 1. This indicates that the proposed deep learning network has good generalization ability in continuous power increase/decrease operating conditions.

(3) Power changes with different up/down rates

In actual operating conditions, the speed of controlling drum rotation is adjustable. Therefore, the time for introducing reactivity during each power increase/decrease is different. In this section, data set 3 form validation set is selected to verify the generalization ability of the deep learning network. In this work, transient operating conditions with different ramp rates are simulated, where the thermal power changes from 115 kW to 95.78 kW and the time for introducing external reactivity are 10 s, 40 s, 60 s, and 80 s, respectively. Fig. 17 shows the variation of system parameters over time from the data set 3.

According to Fig. 17, the deviation caused by the time difference is relatively small. As external reactivity time increases, the fluctuations in heat power become smoother, and the time required for each system parameter to reach its peak value is extended. Fig. 17 reveals that the effects of deviation resulting from the introduction of external reactivity time are relatively minor. As external reactivity time increases, the fluctuations in heat power become smoother, and the time required for each system parameter to reach its

Table 6
Evaluation indicators for dataset 1 form validate dataset.

Case	Γ_{max}		Γ_{ave}		T_{max}		T_{ave}		U_{max}		U_{ave}		γ
	Ther	Elec	Ther	Elect	Ther	Elect	Ther	Elect	Θ_1/K	Θ_2	Θ_1/K	Θ_2	
	A	H			E	K							
97.75 kW	2.83%	2.98%	0.54%	0.88%	13	1 s	5.0 s	0.6 s	16.52	1.95%	−5.87	−0.71%	1.12
	B	H			G	I							
92.15 kW	4.18%	5.42%	0.64%	1.34%	−6	15 s	−1.5 s	4.6 s	24.13	2.80%	−5.06	−0.61%	1.18
	A	I			F	I							
87.22 kW	5.84%	10.59%	0.81%	2.45%	12 s	−8 s	5.0 s	0 s	19.04	2.23%	−4.02	−0.49%	1.17

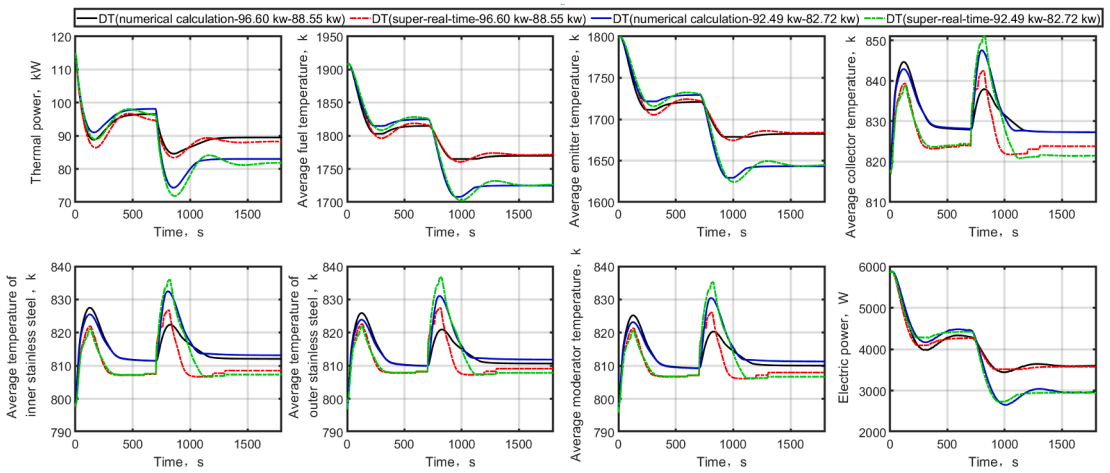


Fig. 15. The variation of reactor system parameters over time (115 kW–96.60 kW–88.55 kW, 115 kW–92.49 kW–82.72 kW).

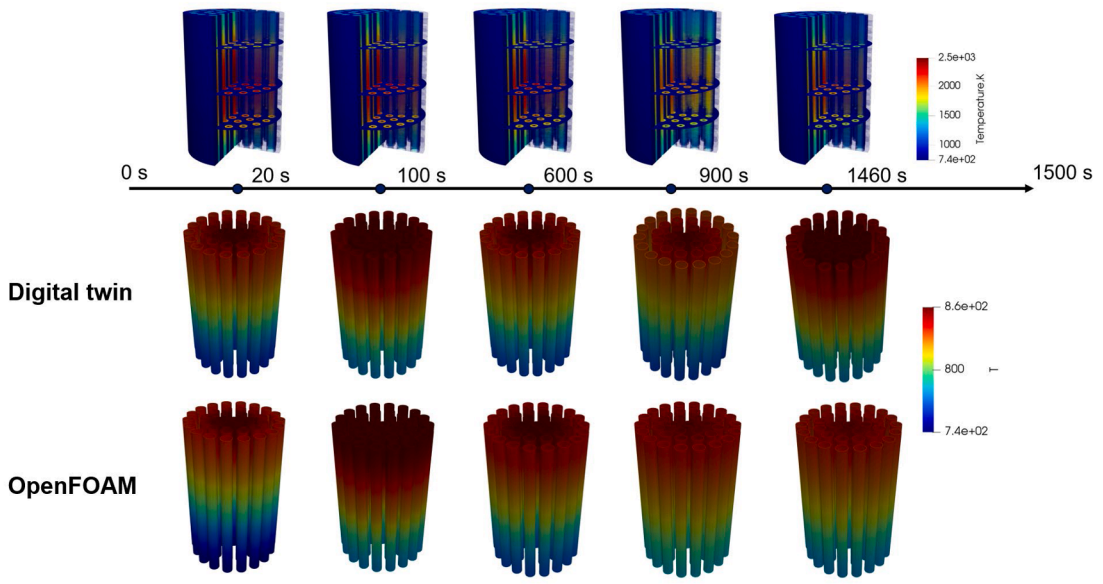


Fig. 16. The temporal evolution of temperature distribution in the improving TOPAZ-II (Top: Reactor; Bottom: Coolant temperature field).

peak value is extended. The super-real-time computing code proposed in this work is capable of simulating the effects resulting from time differences. The relative changes between each curve are in good agreement with the results obtained by traditional numerical methods, demonstrating the generalization ability of the model proposed in this work.

(4) Scaling up/down of power beyond the training conditions

Table 7
Evaluation indicators for dataset 2 form validate dataset.

Case	Γ_{max}		Γ_{ave}		T_{max}		T_{ave}		U_{max}		U_{ave}		γ
	Ther	Elec	Ther	Elect	Ther	Elect	Ther	Elect	Θ_1/K	Θ_2	Θ_1/K	Θ_2	
103.48 kW	A	H			G	I							
87.23 kW	5.08%	9.71%	0.77%	1.28%	−11 s	−2 s	−2 s	−0.4 s	23.69	2.77%	−3.03	−0.37%	1.18
97.75 kW	A	H			E	J							
82.8 kW	3.83%	5.20%	0.79%	1.27%	14 s	2 s	4.6 s	0.6 s	36.88	4.32%	−2.08	−0.27%	1.17
96.60 kW	A	I			F	J							
88.55 kW	3.14%	4.07%	0.50%	0.80%	9 s	1 s	4.4 s	1.6 s	18.12	2.10%	−3.42	−0.42%	1.17
92.49 kW	A	I			G	I							
82.72 kW	4.19%	5.40%	0.60%	1.26%	−6 s	15 s	−1.5 s	4.6 s	22.93	2.67%	−4.04	−0.49%	1.17

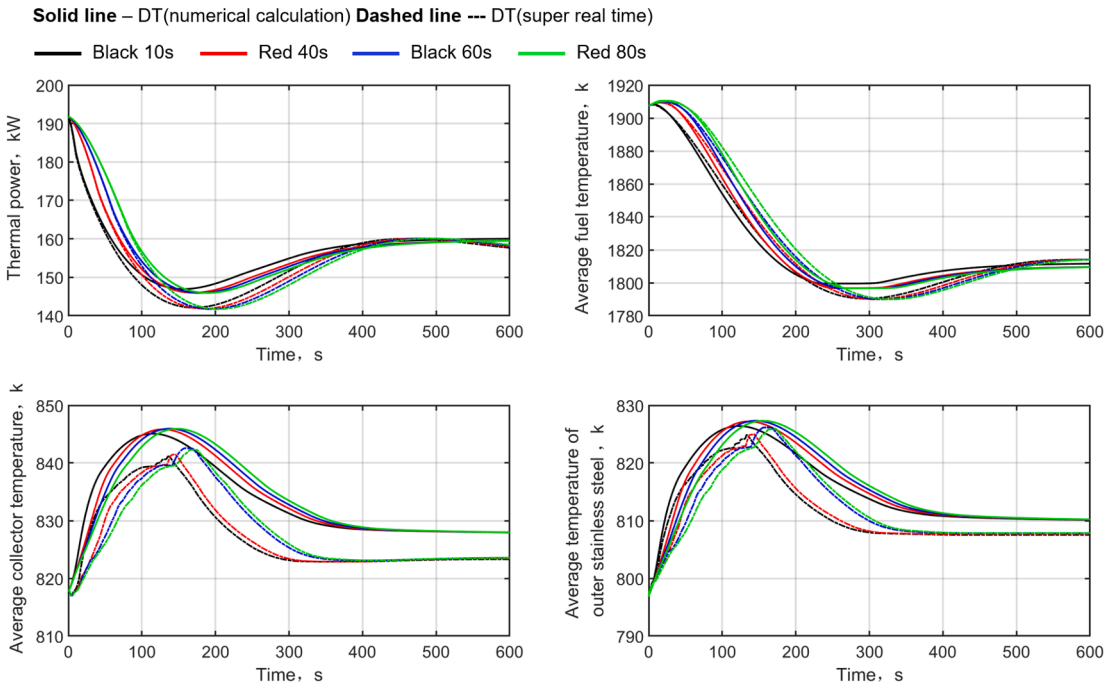


Fig. 17. Reactor system parameters for the 115 kW to 95.78 kW operating conditions at different power ramp rates.

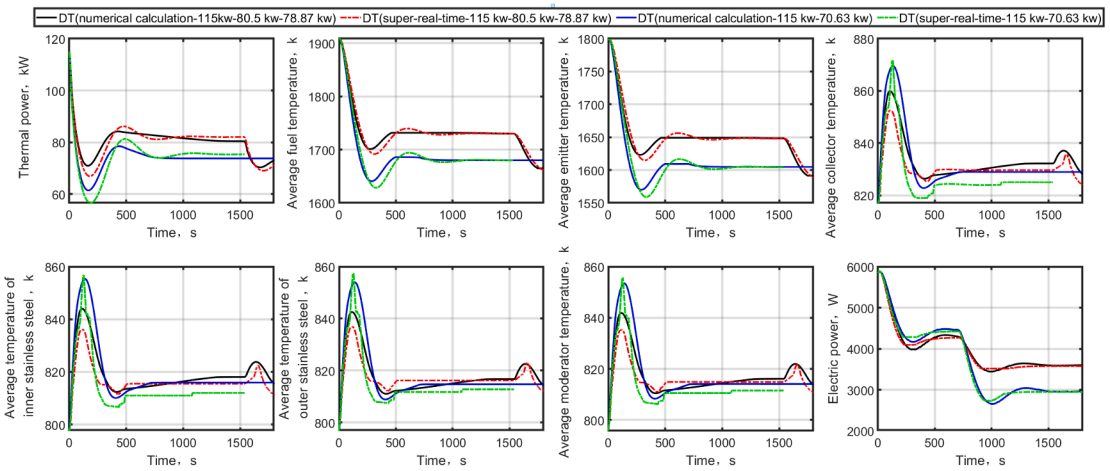


Fig. 18. The variation of reactor system parameters over time (115 kW–80.5 kW–78.87 kW and 115 kW–70.63 kW).

In this section, data set 4 in validation sets was selected to verify the generalization ability of the deep learning network. In this validation set, two deep learning networks are used to simulate non-training transient conditions. Data set 4 has two types of operating conditions: one is continuous power up/down beyond the training data set, and the other is a one-time power up/down beyond the training set. Fig. 18 shows the comparison between the super-real-time computing results and the reference results. For the calculation of the operation condition of 115 kW–80.5 kW–78.87 kW, deep learning network 1, which was trained based on the data set 1, was used for the first reactivity insertion. Deep learning network 2 was used for the second reactivity insertion, and the network switch time was 2200 s. The total running time of the super-real-time code was 1865 s, meeting the requirement for real-time, and the time acceleration ratio is about 1.18. From the results, the average error between the super-real-time computing results and the reference values is less than 1.5% within the considered range of system parameters.

In the calculation of the operating conditions ranging from 115 kW to 70.63 kW, the range of power variation exceeds that of the training conditions. Hence, the use of two deep learning networks is necessary to simulate the system accurately. After the code has been running for 67 s, a decreasing trend in power is observed, and the normalized power falls below the set value. At this time, the deep learning network switches to deep learning network 2, enabling the calculation of operating conditions to adapt to the over-trained data. Based on the results, it is evident that within the range of considered system parameters, the average relative error between the results form the super-real-time code and the reference value is less than 1%. Additionally, the time acceleration ratio is approximately 1.17, which satisfies the super real-time calculation requirements.

From the preceding calculations, it is clear that the temperature deviation of stainless steel is significantly larger than that of other solid regions. This deviation is mainly due to the deviation of the coolant temperature field, which directly affects the distance between each solid region and the fluid region; the closer the distance, the more pronounced the effect of the fluid region deviation. One must consider the potential of incurring substantial training costs because of the marginal effects to mitigate model errors and enhance the accuracy of deep learning operator networks. Thus, it becomes necessary to adopt a real-time monitoring and correction approach [83,84], enabling the code to reflect the physical state of the target faithfully. The proposed super real-time computing method serves precisely this purpose, forming the groundwork for future research. Subsequently, we aim to develop a digital twin framework based on real-time measurement data, allowing the correction of the code's calculated results to achieve a more realistic representation of the physical system. Furthermore, the entire process will be designed to operate in real-time to meet performance requirements.

4. Conclusion

In this study, a three-dimensional numerical solution method with super-real-time and high data performance is proposed. The method is based on a mixed solution framework that combines traditional numerical methods with deep learning networks. Specifically, the method utilizes parallel computing of multiple CPU cores to efficiently solve solid regions, while a deep learning network is loaded onto a GPU to solve fluid regions. The interaction between fluid and solid regions is achieved through communication between multiple CPU cores and GPUs. A deep learning network framework was proposed based on the DeepONet framework for solving fluid equations. A database structure and loss function were also proposed. The model was trained using a classification approach to reduce the complexity of the deep learning network.

The improved TOPAZ-II reactor system was selected as the research object. Four operating conditions other than the training condition were selected to verify the generalization ability of the model. The results demonstrate that, within the range of system parameters considered, the maximum error in thermodynamic parameters is less than 6%, with an average error of less than 1%. The maximum error in electromechanical parameters is less than 11%, with an average error of less than 2.5%. On average, the time deviation of the system parameters reaching the peak is less than 5 s. Increasing external reactivity time delays the peak values of system parameters. The super-real-time computing code can accurately simulates time differences and agrees well with traditional numerical methods, as indicated by relative curve changes. All validation computations met the requirements for super-real-time processing, with a time acceleration ratio of approximately 1.17. Compared to traditional numerical solution methods, the proposed method increased the calculation speed by approximately 60 times. The method proposed in this work satisfies the requirements for super-real-time and high data performance, and demonstrates good generalization performance. The results are within an acceptable range of deviation from traditional calculations. Therefore, the proposed method can be applied to DT of SNRs.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This work is financially support by National Key R&D Program of China (Grant No. 2020YFB1901900), National Natural Science Foundation of China (Grant No. 12275175), Special Fund for Strengthening Industry of Shanghai (Grant No. GYQJ-2018-2-02), Shanghai Rising-Star Program, China (Grant No. 21QA1404200), and LingChuang Research Project of China National Nuclear

Corporation .

References

[1] M. Singh, R. Srivastava, E. Fuenmayor, et al., Applications of Digital Twin across industries: a review, Appl. Sci. 12 (11) (2022) 5727.
[2] M.R. Enders, N. Hozbach, Dimensions of digital twin applications-a literature review, 2019.
[3] B. Schleich, N. Anwer, L. Mathieu, et al., Shaping the digital twin for design and production engineering, CIRP Ann. 66 (1) (2017) 141–144.
[4] Z. Lv, C. Cheng, H. Lv, Digital twins for secure thermal energy storage in building, Appl. Energy 338 (2023), 120907.
[5] S. de López Diz, R.M. López, F.J.R. Sánchez, et al., A real-time digital twin approach on three-phase power converters applied to condition monitoring, Appl. Energy 334 (2023), 120606.
[6] M. Wang, S. Feng, A. Incecik, et al., Structural fatigue life prediction considering model uncertainties through a novel digital twin-driven approach, Comput. Methods Appl. Mech. Engrg. 391 (2022), 114512.
[7] Z. Chen, L. Huang, Digital twins for information-sharing in remanufacturing supply chain: A review, Energy 220 (2021), 119712.
[8] M. Liu, S. Fang, H. Dong, et al., Review of digital twin about concepts, technologies, and industrial applications, J. Manuf. Syst. 58 (2021) 346–361.
[9] H. Pan, Z. Dou, Y. Cai, et al., Digital twin and its application in power system, in: 2020 5th International Conference on Power and Renewable Energy (ICPRE), IEEE, 2020, pp. 21–26.
[10] T.Y. Lin, Z. Jia, C. Yang, et al., Evolutionary digital twin: A new approach for intelligent industrial product development, Adv. Eng. Inform. 47 (2021), 101209.
[11] Y. Zhou, P. Su, J. Wu, et al., Digital twins for flexibility service provision from industrial energy systems, in: 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI), IEEE, 2021, pp. 274–277.
[12] C. Li, D. Yang, Construction of power grid digital twin model based on GAN, in: 2021 China Automation Congress (CAC), IEEE, 2021, pp. 7767–7771.
[13] Y. Zheng, Y. Hua, P. Wei, et al., Application of faster real time simulation based on digital twin in fault monitoring of doubly fed induction generator, in: Annual Conference of China Electrotechnical Society, Springer Nature Singapore, Singapore, 2022, pp. 1158–1167.
[14] M. Singh, R. Srivastava, E. Fuenmayor, et al., Applications of Digital Twin across industries: a review, Appl. Sci. 12 (11) (2022) 5727.
[15] A. Rasheed, O. San, T. Kvamsdal, Digital twin: Values, challenges and enablers from a modeling perspective, Ieee Access 8 (2020) 21980–22012.
[16] Z. Zhaoyun, L. Linjun, Application status and prospects of digital twin technology in distribution grid, Energy Rep. 8 (2022) 14170–14182.
[17] R. Hu, L. Shang, N. Ma, et al., Research on the low-voltage governance and evaluation method for new distribution system based on the digital twin, in: 2022 Power System and Green Energy Conference (PSGEC), IEEE, 2022, pp. 806–813.
[18] P. Zhang, B. Chen, P. Feng, et al., Imaging scheduling of satellite constellation based on digital parallel system, in: Advances in Guidance, Navigation and Control: Proceedings of 2022 International Conference on Guidance, Navigation and Control, Springer Nature Singapore, Singapore, 2023, pp. 3470–3479.
[19] R. Liu, H. Li, Z. Lv, Modeling methods of 3D model in digital twins, CMES-Comput. Model. Eng. Sci. 136 (2) (2023) 985–1022.
[20] K. Zvarikova, J. Horak, S. Downs, Digital twin algorithms, smart city technologies, and 3D spatio-temporal simulations in virtual urban environments, Geopolit. Hist. Int. Relat. 14 (1) (2022) 139–154.
[21] Y. Zhang, J. Bryan, G. Richards, et al., Development and comparative selection of surrogate models using artificial neural network for an integrated regenerative transcritical cycle, Appl. Energy 317 (2022), 119146.
[22] S. Funk, A. Airoud Basmaji, U. Nackenhorst, Globally supported surrogate model based on support vector regression for nonlinear structural engineering applications, Arch. Appl. Mech. 93 (2) (2023) 825–839.
[23] M. Fahim, V. Sharma, T.V. Cao, et al., Machine learning-based digital twin for predictive modeling in wind turbines, IEEE Access 10 (2022) 14184–14194.
[24] A. Galeazzi, K. Prifti, C. Cortellini, et al., Development of a surrogate model of an amine scrubbing digital twin using machine learning methods, Comput. Chem. Eng. 174 (2023), 108252.
[25] S. Wang, X. Lai, X. He, et al., Building a trustworthy product-level shape-performance integrated digital twin with multifidelity surrogate model, J. Mech. Des. 144 (3) (2022).
[26] M. Zhou, J. Yan, D. Feng, Digital twin framework and its application to power grid online analysis, CSEE J. Power Energy Syst. 5 (3) (2019) 391–398.
[27] T.I. Zohdi, A machine-learning framework for rapid adaptive digital-twin based fire-propagation simulation in complex environments, Comput. Methods Appl. Mech. Engrg. 363 (2020), 112907.
[28] D.R. Gunasegaram, A.B. Murphy, A. Barnard, et al., Towards developing multiscale-multiphysics models and their surrogates for digital twins of metal additive manufacturing, Addit. Manuf. 46 (2021), 102089.
[29] J. Chen, C. Meng, Y. Gao, et al., Multi-fidelity neural optimization machine for digital twins, Struct. Multidiscip. Optim. 65 (12) (2022) 1–15.
[30] W. He, J. Mao, K. Song, et al., Structural performance prediction based on the digital twin model: A battery bracket example, Reliab. Eng. Syst. Saf. 229 (2023), 108874.
[31] Y. Yi, C. Xia, C. Feng, et al., Digital twin-long short-term memory (LSTM) neural network based real-time temperature prediction and degradation model analysis for lithium-ion battery, J. Energy Storage 64 (2023), 107203.
[32] S. Fresca, A. Manzoni, L. Dedè, et al., POD-enhanced deep learning-based reduced order models for the real-time simulation of cardiac electrophysiology in the left atrium, Front. Physiol. 12 (2021), 679076.
[33] S. Fresca, A. Manzoni, Real-time simulation of parameter-dependent fluid flows through deep learning-based reduced order models, Fluids 6 (7) (2021) 259.
[34] S. Fresca, G. Gobat, P. Fedeli, et al., Deep learning-based reduced order models for the real-time simulation of the nonlinear dynamics of microstructures, Internat. J. Numer. Methods Engrg. 123 (20) (2022) 4749–4777.
[35] P. Conti, G. Gobat, S. Fresca, et al., Reduced order modeling of parametrized systems through autoencoders and SINDy approach: continuation of periodic solutions, Comput. Methods Appl. Mech. Engrg. 411 (2023), 116072.
[36] G. Makkar, C. Smith, G. Drakoulas, et al., A machine learning framework for physics-based multi-fidelity modeling and health monitoring for a composite wing, in: ASME International Mechanical Engineering Congress and Exposition, Vol. 86625, American Society of Mechanical Engineers, 2022, V001T01A008.
[37] G.I. Drakoulas, T.V. Gortsas, G.C. Bourantas, et al., FastSVD-ML–ROM: A reduced-order modeling framework based on machine learning for real-time applications, Comput. Methods Appl. Mech. Engrg. 414 (2023), 116155.
[38] A. Bhosekar, M. Ierapetritou, Advances in surrogate based modeling, feasibility analysis, and optimization: A review, Comput. Chem. Eng. 108 (2018) 250–267.
[39] B. Sudret, S. Marelli, J. Wiart, Surrogate models for uncertainty quantification: An overview, in: 2017 11th European Conference on Antennas and Propagation (EUCAP), IEEE, 2017, pp. 793–797.
[40] Y. Lu, B. Wang, Y. Zhao, et al., Physics-informed surrogate modeling for hydro-fracture geometry prediction based on deep learning, Energy 253 (2022), 124139.
[41] J. An, X. Yin, R. Chen, et al., Integrative taxonomy of the subfamily Orbioninae (Crustacea: Isopoda) based on mitochondrial and nuclear data with evidence that supports Epicaridea as a suborder, Mol. Phylogenet. Evol. 180 (2023), 107681.
[42] E.F. Araújo, L.N.F. Guimarães, American space nuclear electric systems, J. Aerosp. Technol. Manag. (2018) 10.
[43] S.S. Voss, TOPAZ II design evolution, in: AIP Conference Proceedings, Vol. 301, American Institute of Physics, 1994, pp. 791–802 (1).
[44] X. Song, G. Sun, G. Li, et al., Crashworthiness optimization of foam-filled tapered thin-walled structure using multiple surrogate models, Struct. Multidiscip. Optim. 47 (2013) 221–231.
[45] N. Demo, M. Tezzele, G. Rozza, A DeepONet multi-fidelity approach for residual learning in reduced order modeling, 2023 https://doi.org/10.1186/s40323-023-00249-9.
[46] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[47] S. Cai, Z. Mao, Z. Wang, et al., Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sinica 37 (12) (2021) 1727–1738.

[48] B. Huang, J. Wang, Applications of physics-informed neural networks in power systems-a review, IEEE Trans. Power Syst. (2022).
[49] C. Wu, M. Zhu, Q. Tan, et al., A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 403 (2023), 115671.
[50] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, J. Comput. Phys. 425 (2021), 109913.
[51] J. Yu, L. Lu, X. Meng, et al., Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Engrg. 393 (2022), 114823.
[52] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Engrg. 374 (2021), 113547.
[53] K. Ishitsuka, W. Lin, Physics-informed neural network for inverse modeling of natural-state geothermal systems, Appl. Energy 337 (2023), 120855.
[54] A.D. Jagtap, G.E. Karniadakis, in: Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations, AAAI Spring Symposium: MLPS, 2021, pp. 2002–2041.
[55] M. Yin, X. Zheng, J.D. Humphrey, et al., Non-invasive inference of thrombus material properties with physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 375 (2021), 113603.
[56] L. Lu, P. Jin, G. Pang, et al., Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[57] L. Lu, P. Jin, G.E. Karniadakis, Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, 2019 arXiv preprint arXiv:1910.03193.
[58] S. Goswami, M. Yin, Y. Yu, et al., A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials, Comput. Methods Appl. Mech. Engrg. 391 (2022), 114587.
[59] L. Lu, R. Pestourie, S.G. Johnson, et al., Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport, Phys. Rev. Res. 4 (2) (2022), 023210.
[60] L. Lu, P. Jin, G. Pang, et al., Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[61] S. Garg, H. Gupta, S. Chakraborty, Assessment of DeepONet for time dependent reliability analysis of dynamical systems subjected to stochastic loading, Eng. Struct. 270 (2022), 114811.
[62] M. Zhu, H. Zhang, A. Jiao, et al., Reliable extrapolation of deep neural operators informed by physics or sparse observations, Comput. Methods Appl. Mech. Engrg. 412 (2023), 116064.
[63] W. Wu, M. Peng, A data mining approach combining $ K $-means clustering with bagging neural network for short-term wind power forecasting, IEEE Internet Things J. 4 (4) (2017) 979–986.
[64] X. Cao, Y. Liu, J. Wang, et al., Prediction of dissolved oxygen in pond culture water based on K-means clustering and gated recurrent unit neural network, Aquac. Eng. 91 (2020), 102122.
[65] R.E. Schapire, A brief introduction to boosting, Ijcai 99 (999) (1999) 1401–1406.
[66] S. Dudoit, J. Fridly, Bagging to improve the accuracy of a clustering procedure, Bioinformatics 19 (9) (2003) 1090–1099.
[67] S. Masoudnia, R. Ebrahimpour, Mixture of experts: a literature survey, Artif. Intell. Rev. 42 (2014) 275–293.
[68] P. Li, C. Wei, An emergency decision-making method based on DS evidence theory for probabilistic linguistic term sets, Int. J. Disaster Risk Reduct. 37 (2019), 101178.
[69] J. He, J. Qiu, A. Zeng, et al., Fastmoe: A fast mixture-of-expert training system, 2021 arXiv preprint arXiv:2103.13262.
[70] Y. Zhao, G. Zheng, S. Mukherjee, et al., Admoe: Anomaly detection with mixture-of-experts from noisy labels, Proc. AAAI Conf. Artif. Intell. 37 (4) (2023) 4937–4945.
[71] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[72] S. Mannor, D. Peleg, R. Rubinstein, The cross entropy method for classification, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 561–568.
[73] V. Dolean, P. Jolivet, F. Nataf, An Introduction To Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation, Society for Industrial and Applied Mathematics, 2015.
[74] S. Keough, Optimising the Parallelisation of Openfoam Simulations, Defence Science and Technology Organisation Fishermans Bend (Australia) Maritime Div, 2014.
[75] M. Towara, M. Schanen, U. Naumann, MPI-parallel discrete adjoint OpenFOAM, Procedia Comput. Sci. 51 (2015) 19–28.
[76] E. Gabriel, G.E. Fagg, G. Bosilca, et al., Open MPI: Goals, concept, and design of a next generation MPI implementation, in: Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting Budapest, Hungary, September (2004) 19-22. Proceedings 11, Springer, Berlin Heidelberg, 2004, pp. 97–104.
[77] N. Hjelm, An evaluation of the one-sided performance in open mpi, in: Proceedings of the 23rd European MPI Users' Group Meeting, 2016, pp. 184–187.
[78] R.A. Johnson, W.T. Morgan, S.R. Rocklin, Design, ground test and flight test of SNAP 10A, first reactor in space, Nucl. Eng. Des. 5 (1) (1967) 7–21.
[79] E. Zhu, Z. Wang, J. Ma, et al., Transient multiphysics characteristics of a space thermionic reactor based on a coupling analysis, Nucl. Eng. Des. 401 (2023), 112064.
[80] Q. Ma, P. Sun, X. Wei, et al., Dynamic model construction of a space thermionic nuclear reactor, Appl. Therm. Eng. (2022), 118644.
[81] W. Xiao, X. Li, P. Li, et al., High-fidelity multi-physics coupling study on advanced heat pipe reactor, Comput. Phys. Comm. 270 (2022), 108152.
[82] K. Cho, B.Van. Merriënboer, C. Gulcehre, et al., Learning phrase representations using RNN encoder–decoder for statistical machine translation, 2014 arXiv preprint arXiv:1406.1078.
[83] H. Song, M. Song, X. Liu, Online autonomous calibration of digital twins using machine learning with application to nuclear power plants, Appl. Energy 326 (2022), 119995.
[84] T.I. Zohdi, A machine-learning digital-twin for rapid large-scale solar-thermal energy system design, Comput. Methods Appl. Mech. Engrg. 412 (2023), 115991.