Korean J. Chem. Eng., 39(3), 515-528 (2022)
DOI: 10.1007/s11814-021-0979-x

pISSN: 0256-1115
eISSN: 1975-7220

Physics-informed deep learning for data-driven solutions of computational fluid dynamics

Solji Choi*, Ikhwan Jung, Haeun Kim***, Jonggeol Na****,†, and Jong Min Lee*,†**

*School of Chemical and Biological Engineering, Seoul National University, Seoul 08826, Korea
**Research & Development Institute, Hanwha Chemical Corporation, Gyeonggi-do 13488, Korea
***Department of Chemical Engineering and Materials Science, Ewha Womans University, Seoul 03760, Korea
****Department of Chemical Engineering and Materials Science, Graduate Program in System Health Science and Engineering, Ewha Womans University, Seoul 03760, Korea
(*Received 20 May 2021 • Revised 9 October 2021 • Accepted 10 October 2021*)

Abstract−Computational fluid dynamics (CFD) is an essential tool for solving engineering problems that involve fluid dynamics. Especially in chemical engineering, fluid motion usually has extensive effects on system states, such as temperature and component concentration. However, due to the critical issue of long computational times for simulating CFD, application of CFD is limited for many real-time problems, such as real-time optimization and process control. In this study, we developed a surrogate model of a continuous stirred tank reactor (CSTR) with van de Vusse reaction using physics-informed neural network (PINN), which can train the governing equations of the system. We propose a PINN architecture that can train every governing equation which a chemical reactor system follows and can train a multi-reference frame system. Also, we investigated that PINN can resolve the problem of neural network that needs a large number of training data, is easily overfitted and cannot contain physical meaning. Furthermore, we modified the original PINN suggested by Raissi to solve the memory error and divergence problem with two methods: Mini-batch training and weighted loss function. We also suggest a similarity-based sampling strategy where the accuracy can be improved up to five times over random sampling. This work can provide a guideline for developing a high performance surrogate model of the chemical process.

Keywords: Physics-informed Neural Network, Surrogate Model, Computational Fluid Dynamics, Chemical Reactor

INTRODUCTION

Computational fluid dynamics (CFD) is a powerful tool that uses numerical approaches to solve engineering problems involving fluid motion and heat transfer. CFD has been employed in a wide range of research fields, such as aerodynamics [1], weather dynamics [2], environmental engineering, and industrial system modeling [3]. However, a critical issue of CFD simulations is their long computational times. Consequently, the application of CFD in analyzing the real-time behavior of systems and predicting non-simulated scenarios has been inherently limited.

Recent developments in machine learning have fundamentally transformed many classical engineering approaches and provided suitable alternative methods to solve complex tasks, including the simulation of complex fluid motion and heat transfer. Motivated by the potential of machine learning for speeding up numerical simulation without compromising accuracy, a number of studies have focused on developing a surrogate model that aims to replace or accelerate CFD simulations. The most widely used surrogate models in chemical process engineering include polynomial approximations [4], Gaussian process (GP) [5], and deep learning models [6]. Polynomial approximation is the most commonly used model; however, it is simple and its applications are limited to less complex underlying models and local regions. GP is a nonparametric Bayesian approach that works well on small datasets and can provide a measure of uncertainty for predictions. GP is suitable for modeling small datasets where some prior knowledge of the generative process exists. Deep learning techniques have been widely used to develop surrogate models for fluid flow systems owing to their scalability with a large amount of data and hierarchical feature learning, which are suitable for large-scale and highly nonlinear systems, respectively. Ling et al. [7] used a deep neural network (NN) with embedded invariance for Reynolds-averaged turbulence modeling. Na et al. [8] proposed a non-linear surrogate model for toxic gas release scenarios using a variational autoencoder with deep convolutional layers and a deep NN. To obtain a reliable model, these deep learning approaches require a large amount of data to avoid overfitting and undue extrapolation. Although the feature extraction characteristic of deep learning is known to be advantageous, these extracted features are often inconsistent with physical implications. Hence, these purely data-based surrogate models cannot be reliably used for inference in untrained spaces [9].

To overcome these limitations, several approaches to combine scientific knowledge and deep learning models have been developed. Karpatne et al. [10] proposed a physics-guided NN that uses scientific knowledge as physics-based loss functions in the learning objective of NNs in order to obtain a physically consistent model. Raissi et al. [11] proposed a more advanced model called the PINN,

†To whom correspondence should be addressed.
E-mail: jgna@ewha.ac.kr, jongmin@snu.ac.kr

in which the constraint term of the physical law described by general non-linear partial differential equations is incorporated into NN training. PINN is suitable for systems where the first principles are given in the form of partial differential equations, such as fluid flow systems governed by mass and momentum conservation laws. PINN has numerous advantages over purely data-driven models: it can be effectively trained with small datasets; it shows high performance in regions without observable data because it can be trained in any region to satisfy the governing equation without actual data. It can also solve the inverse problem of finding unknown parameters that converge toward their true values during training. Owing to these advantages, the PINN approach has been applied to several physical systems such as high-speed aerodynamic flows described by the Euler equation [12], wall modeling in large-eddy simulations [13], and the incompressible Navier-Stokes equation [14]. Previous studies have mainly dealt with fluid flow systems only, and none of them have solved reaction kinetics and heat transfer together, which is essential in the model development of chemical engineering systems.

In this study, we investigated the feasibility of applying PINN to a chemical reactor system, which is a continuous stirred tank reactor (CSTR) with a van de Vusse reaction. A method for a training system, including mass conservation equations, momentum conservation equations, heat conservation equations, and species transport equations, is suggested. In particular, because the system requires multi-reference frames owing to the mixing domain, we present how the model framework is conceived for implementation. Further, as the system operates by several different types of physical equations and the number of system variables increases, reliable performance cannot be obtained with existing PINN methodology. Therefore, we suggest three strategies to improve model performance. First, a large number of training data points are needed to train the system; thus, mini-batch training is introduced to prevent memory errors and increase model accuracy. Second, because empirical and physics errors for each governing equation are of different scales and hinder proper learning, we adopt the concept of the weighted loss function. Third, a similarity-based sampling strategy is introduced, which can improve the performance of the model when a sudden change in the system state occurs.

PINN

PINN, first proposed by Raissi et al. [11], is effective in finding a robust solution from data derived from a system whose governing equations are in the form of partial differential equations. PINN embraces a partial differential form of physical law in machine learning, making it possible to learn not only the patterns of the data but also their underlying physical laws. PINN is based on the same algorithm as that of the feedforward NN, except for the loss function. While the gradients of the output value errors are back-propagated to update the weights and bias in the feedforward NN, additional loss terms regarding the physical law determined using automatic differentiation are included in PINN. Therefore, the loss function consists of an empirical error term (MSE_y) and a physics error term (MSE_f), as presented in Eq. (1).

$$MSE = MSE_y + MSE_f \tag{1}$$

As presented in Eq. (2), the empirical error term is defined as the mean squared error of the output variables, where $\{x_y^i, y^i\}$ $(i=1, \ldots, N_y)$ denotes the training data on y(x), N_y is the number of training data points, x_y^i is mostly defined as a set of independent space and time variables, and y^i are the system state variables such as velocity components and pressure. The physics error term in Eq. (3) is the mean squared error of the governing equations, which can be calculated from NNs through automatic differentiation, where $\{x_f^i\}$ specifies the collocation points for f(x), N_f is the number of collocation points, and f is a set of physical equations.

$$MSE_y = \frac{1}{N_y}\sum_{i=1}^{N_y}\left|y(x_y^i) - y^i\right|^2 \tag{2}$$

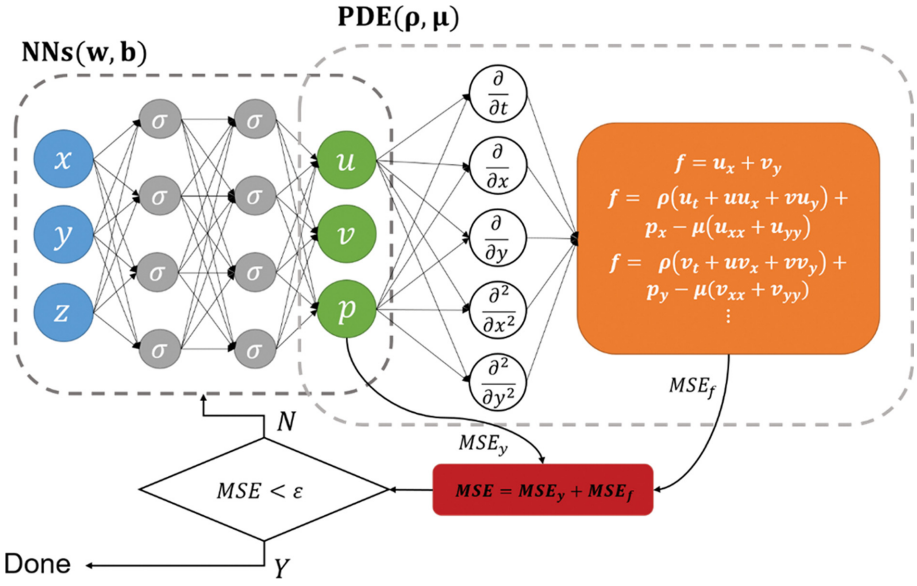

Fig. 1. Schematic of PINN.

$$MSE_f = \frac{1}{N_f}\sum_{fi=1}^{N_f}\left|f(x_f^i)\right|^2 \quad (3)$$

As shown in Fig. 1, the model outputs from the NNs are fed into the first-principles model, and the error in the output variables and the error in the physical equation are summed and backpropagated to update the hyperparameters of the NNs.

PINNs are suitable for simulating a chemical reactor system which is represented by a set of partial differential equations, such as mass and momentum balances, heat conservation equations, and species transport equations. However, because of the properties of the reactor system, where multi-reference frames exist and are governed by various types of physical equations, PINN structures need to be modified, and several strategies are needed to improve model performance. In the subsequent section, we suggest a loss function and model structure relevant to the target system.

MODEL DESCRIPTION

In this section, we introduce the modeling method of a CSTR with a van de Vusse reaction using a model framework based on CFD and PINNs. The CSTR with the van de Vusse reaction has been widely used as a benchmark problem owing to its nonlinearity [15]. There are two types of reactions commonly found in the literature: isothermal [16,17] and non-isothermal [18,19]. A non-isothermal reactor was used in this study to consider fluid flow, heat transfer, and reaction kinetics together.

1. CFD Modeling

Fig. 2 depicts the reactor geometry: a flat-bottomed, fully baffled tank, with a diameter of 0.288 m and a height of 0.288 m. It has one inlet, one outlet, and centrally located pitched-blade impellers that rotate at a speed of 10 rpm.

Van de Vusse reactions are referred to from Ridlehoover et al. [20], and the kinetic parameters that we used for the target system are presented in Table 1. The model component properties and the initial and boundary conditions are also listed in Table 1.

The reactor is divided into two domains: the inner and outer. The inner domain is assigned around the impeller, and the outer domain is the rest of the domain. The inner domain refers to the moving reference frame, which rotates in the opposite direction of the impeller, and the outer domain refers to the stationary frame.

Table 1. Reaction kinetics, component properties, and initial and boundary conditions

Reactions	Kinetic parameters	
	A (1/s)	E_a (J/mol)
$A \xrightarrow{k_1} B$	$3.575 * 10^9$	$8.109 * 10^7$
$B \xrightarrow{k_2} C$	$3.575 * 10^9$	$8.109 * 10^7$
$2A \xrightarrow{k_3} D$	$2.512 * 10^9$	$7.113 * 10^7$
Component properties		
ρ	934.6 kg/m^3	
μ	$1.0 * 10^{-3}$ kg/m·s	
C_p	3,010 J/K·kg	
K	0.6 W/m·K	
Mw_A	155.3 kg/mol	
D	$2.88 * 10^{-5}$ m^2/s	
Initial conditions		
C_{a_0}	$2.23 * 10^{-3}$ mol/L	
C_{b_0}	$1.04 * 10^{-3}$ mol/L	
C_{c_0}	$0.91 * 10^{-3}$ mol/L	
C_{d_0}	$0.92 * 10^{-3}$ mol/L	
T_0	79.591 °C	
T_f	104.9 °C	
$C_{a,f}$	$5.1 * 10^{-3}$ mol/L	
T_k	77.5 °C	
Q_f	$3.89 * 10^{-2}$ L/sec	

Therefore, the governing equation for each domain varies according to the reference frame. A set of governing equations that satisfy each domain is explained in the section on the governing equation, and the PINN model architecture that integrates two different domains into one model is introduced in the section on the PINN architecture.

The mesh structure is composed of 0.3 M cells. We performed a mesh convergence test on eight different meshes. The graph on the left in the Fig. 3 shows the average velocity value at the outlet according to the number of meshes and the graph on the right shows the average mole fraction of d at the outlet according to the number of meshes. When the number of meshes is more than

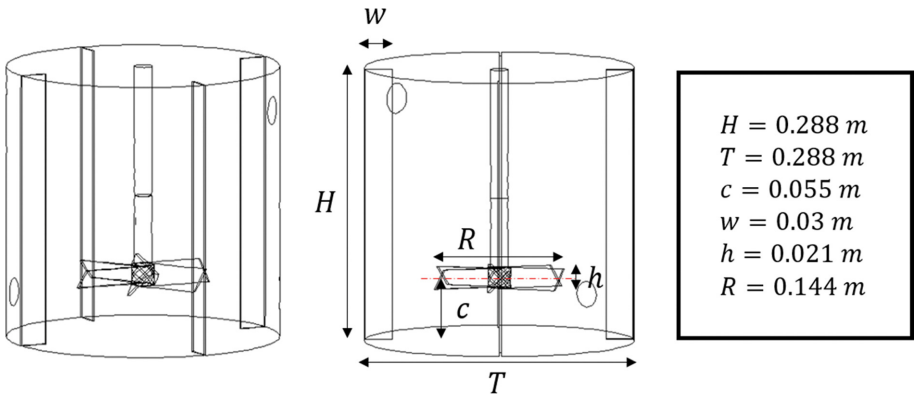

Fig. 2. Reactor system geometry and tank specifications.

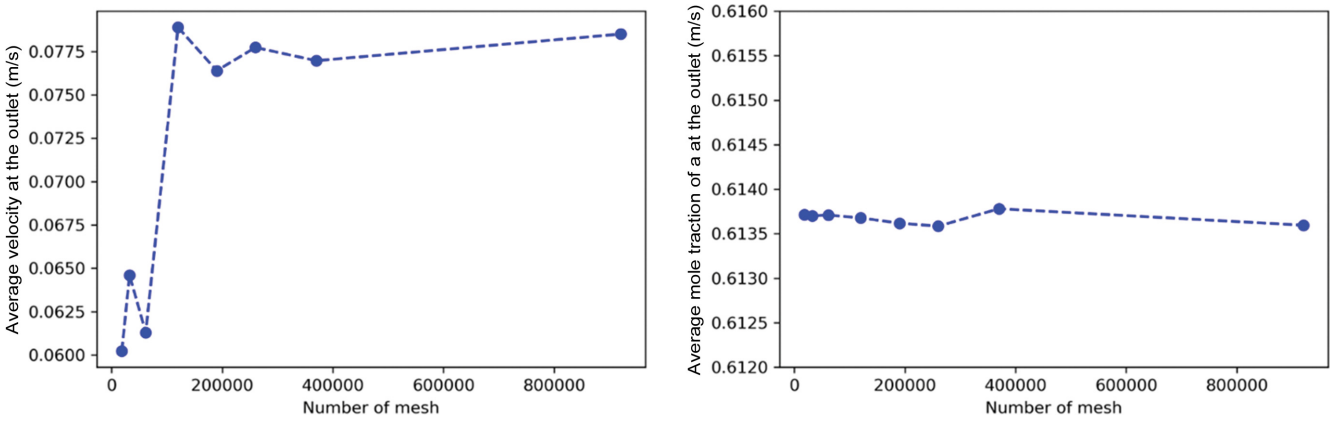

Fig. 3. Mesh convergence test results.

0.2 M, the simulation results converge. We used ANSYS-Fluent v19.2 to generate the model training data calculated through CFD. After the simulations were completed, the values of the state variables (velocity components, pressure, and temperature) and the mole fraction of each component according to time and position were exported.

2. Governing Equations

The target system is governed by the mass, momentum, and energy conservation laws and the species transport equations. We used a laminar model for the viscous model and assumed a compressible flow. Specifically, for our system, we consider two domains with different reference frames, and thus each domain follows a different form of physics law. The velocity vector is converted from a moving reference frame to an absolute frame using

$$\vec{v} = \vec{v}_r + (\vec{w} \times \vec{r}). \tag{4}$$

The term $\vec{v}$ is the absolute velocity vector, $\vec{v}_r$ is the relative velocity vector, $\vec{w}$ is the rotating angular velocity, and $\vec{r}$ is the position vector from the origin of the moving system. The governing equations for fluid flow in an absolute frame can be written as follows:

The equation for the conservation of mass is

$$\frac{\delta \rho}{\delta t} + \nabla \cdot (\rho \vec{v}) = 0, \tag{5}$$

where ρ is the density.

The equation for the conservation of momentum is

$$\frac{\delta}{\delta t}(\rho \vec{v}) + \nabla \cdot (\rho \vec{v}\vec{v}) = -\nabla p + \nabla \cdot (\bar{\bar{\tau}}), \tag{6}$$

where $\bar{\bar{\tau}}$ is the stress tensor.

The equation for the conservation of energy is

$$\frac{\delta}{\delta t}(\rho E) + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot (k \nabla T + \bar{\bar{\tau}} \cdot \vec{v}), \tag{7}$$

where E is the internal energy.

The equation for species transport is

$$\frac{\delta}{\delta t}(\rho Y_i) + \nabla \cdot (\rho \vec{v} Y_i) = -(\rho D \nabla Y_i) + R_i, \tag{8}$$

where Y_i is the local mass fraction of the i^{th} species, and R_i is the net rate of the production of species i by a chemical reaction.

For the relative velocity formulation, the governing equations of the fluid flow in a moving reference frame can be written as follows:

The equation for the conservation of mass is

$$\frac{\delta \rho}{\delta t} + \nabla \cdot (\rho \vec{v}_r) = 0. \tag{9}$$

The equation for the conservation of momentum is

$$\frac{\delta}{\delta t}(\rho \vec{v}_r) + \nabla \cdot (\rho \vec{v}_r \vec{v}_r) + \rho(2\vec{w} \times \vec{v}_r + \vec{w} \times \vec{w} \times \vec{r}) = -\nabla p + \nabla \cdot (\bar{\bar{\tau}}_r). \tag{10}$$

The momentum equation contains two additional acceleration terms. The $(2\vec{w} \times \vec{v}_r)$ term is the Coriolis acceleration, and the $(\vec{w} \times \vec{w} \times \vec{r})$ term is the centripetal acceleration.

The equation for the conservation of energy is

$$\frac{\delta}{\delta t}(\rho E_r) + \nabla \cdot (\rho \vec{v}_r H_r) = \nabla \cdot (k \nabla T + \bar{\bar{\tau}}_r \cdot \vec{v}_r), \tag{11}$$

where E_r is the relative internal energy and H_r is the relative total enthalpy.

The equation for species transport is

$$\frac{\delta}{\delta t}(\rho Y_i) + \nabla \cdot (\rho \vec{v}_r Y_i) = -\nabla(\rho D \nabla Y_i) + R_i. \tag{12}$$

These equations will contribute to PINN training.

3. PINN Architecture

We designed the PINN to infer all the states of interest, i.e., velocity components (u, v, w), temperature (T), and mole fractions of a, b, c, and d (f_a, f_b, f_c, f_d). $\mathbf{x} :=$ (x, y, z, t) is introduced as a set of independent space and time variables and $\mathbf{y} :=$ (u, v, w, T, f_a, f_b, f_c, p) represents the set of state variables. Based on the PINN framework [11], the output variable is approximated using a feed-forward multi-layer NN:

$$\mathbf{y}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \theta), \tag{13}$$

where $\mathcal{N}$ is an $\mathcal{L}$-layer NN with a set of input variables $\mathbf{x}$ and a network parameter $\boldsymbol{\theta}$. This network represents a compositional function, which is

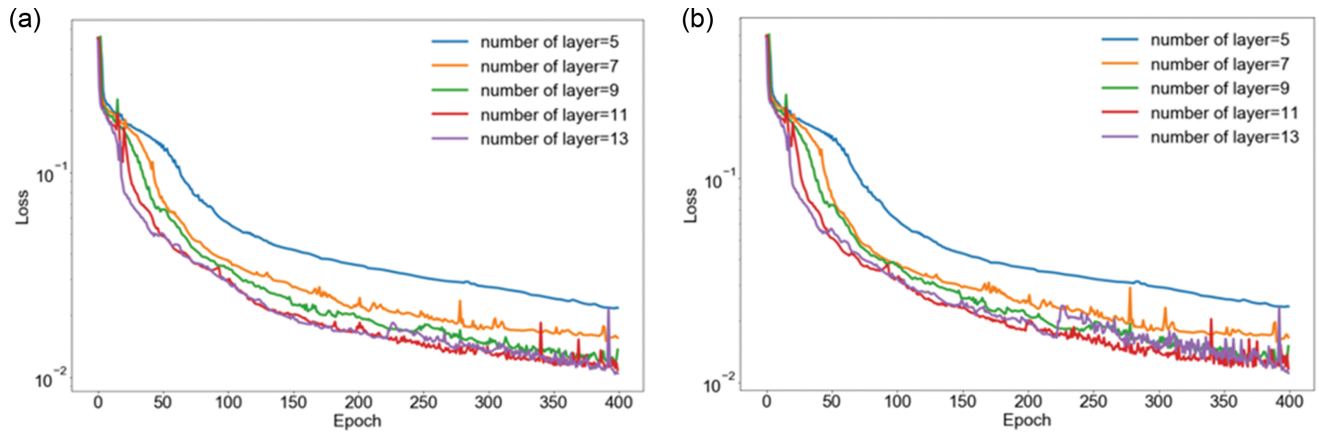

Fig. 4. Loss graph for various numbers of layers: (a) Training data loss, (b) test data loss.

$$\mathcal{N}(\mathbf{x};\theta)=\Sigma^{\mathcal{L}}\circ\Sigma^{\mathcal{L}-1}\circ\cdots\circ\Sigma^{1}(\mathbf{x}) \tag{14}$$

$$\Sigma^{\lambda}(\mathrm{z}^{\lambda-1}):=\mathrm{z}^{\lambda}=\sigma^{\lambda}(\mathbf{W}^{\lambda}\cdot\mathrm{z}^{\lambda-1}+\mathbf{b}^{\lambda})\ \text{ where } \lambda=1,\ldots,\mathcal{L} \tag{15}$$

where the symbol ∘ is the composition operation, λ is the layer number, $\mathrm{z}^0:=\mathbf{x}$ is the input to the network, $\mathrm{z}^{\mathcal{L}}:=\mathbf{y}$ is the output of the network, $\mathbf{W}^{\lambda}$ and $\mathbf{b}^{\lambda}$ are the network parameters of layer λ known as weight and bias, and σ^{λ} is the activation function.

The architecture of a PINN consists of 11 layers, including 100 neurons for each layer, whose numbers are determined empirically by simulating several cases. Too many layers prevent a model from being generalized, and a small number of layers is not sufficient to represent the system. Fig. 4 shows training and test loss according to the learning epoch for various number of layer. The number of epochs means how many times the entire dataset is passed forward and backward through the neural network. As shown in Fig. 4, when the number of layers exceeds 11, model accuracy does not improve anymore. Training and test data loss oscillate more when the number of layers is 13, reducing model stability. Each node is fully connected to the nodes of the previous layer and contributes to the next layer's nodes with the tangent hyperbolic activation function.

As mentioned, the loss function consists of an empirical loss term, which is the mean squared error between the real and predicted values of the output variables, and a physics loss term, which is the model error of the governing equations consisting of derivative terms calculated through automatic differentiation from the networks. A weighted loss function is used here. In particular, for our system, the total loss of the network, MSE, includes a loss term for the output variables (MSE_u, MSE_T and MSE_Y) and the physics equations ($\mathrm{MSE}_{f_{cont}}$), MSE_{f_u}, MSE_{f_k} and $\mathrm{MSE}_{f_{spec}}$).

$$\begin{aligned}\mathrm{MSE}=\mathrm{w}_{exp}(\mathrm{MSE}_u+\mathrm{MSE}_T+\mathrm{MSE}_Y)+\mathrm{w}_{phy_1}\mathrm{MSE}_{f_{cont}}\\+\mathrm{w}_{phy_2}\mathrm{MSE}_{f_u}+\mathrm{w}_{phy_3}\mathrm{MSE}_{f_k}+\mathrm{w}_{phy_4}\mathrm{MSE}_{f_{spec}},\end{aligned} \tag{16}$$

where the empirical loss term is evaluated at training points $\{\mathrm{x}_u^i, \mathrm{y}_u^i, \mathrm{z}_u^i, \mathrm{t}_u^i, \mathrm{u}^i, \mathrm{v}^i, \mathrm{w}^i, \mathrm{T}^i, \mathrm{f}_a^i, \mathrm{f}_b^i, \mathrm{f}_c^i\}$ i=1, …, N_u, and the physics loss term is evaluated at a collocation point $\{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}$ i=1, …, N_f. The empirical loss terms are expressed as

$$\mathrm{MSE}_u=\frac{1}{\mathrm{N}_u}\sum_{i=1}^{N_u}\left(\left|\mathrm{u}(\mathbf{x})^i-\mathrm{u}^i\right|^2+\left|\mathrm{v}(\mathbf{x})^i-\mathrm{v}^i\right|^2+\left|\mathrm{w}(\mathbf{x})^i-\mathrm{w}^i\right|^2\right) \tag{17}$$

$$\mathrm{MSE}_T=\frac{1}{\mathrm{N}_u}\sum_{i=1}^{N_u}\left(\left|\mathrm{T}(\mathbf{x})^i-\mathrm{T}^i\right|^2\right) \tag{18}$$

$$\mathrm{MSE}_Y=\frac{1}{\mathrm{N}_u}\sum_{i=1}^{N_u}\left(\left|\mathrm{f}_a(\mathbf{x})^i-\mathrm{f}_a^i\right|^2+\left|\mathrm{f}_b(\mathbf{x})^i-\mathrm{f}_b^i\right|^2+\left|\mathrm{f}_c(\mathbf{x})^i-\mathrm{f}_c^i\right|^2\right) \tag{19}$$

The physics loss terms differ depending on whether the collocation point belongs to the inner or outer domains. The loss term for the mass conservation equation is

$$\mathrm{MSE}_{f_{cont}}=\frac{1}{\mathrm{N}_f}\sum_{fi=1}^{N_f}\left(\left|\mathrm{f}_{cont}^i\right|^2\right). \tag{20}$$

The loss term for the momentum conservation equation is

$$\mathrm{MSE}_{f_u}=\begin{cases}\dfrac{1}{\mathrm{N}_{in}}\displaystyle\sum_{i=1}^{N_{in}}\left(\left|\mathrm{f}_{u_{in}}^i\right|^2+\left|\mathrm{f}_{u_{in}}^i\right|^2+\left|\mathrm{f}_{\mathrm{w}}^i\right|^2\right) & \{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}\in\mathbf{X}_{in}\\ \dfrac{1}{\mathrm{N}_{out}}\displaystyle\sum_{i=1}^{N_{out}}\left(\left|\mathrm{f}_{u_{out}}^i\right|^2+\left|\mathrm{f}_{u_{out}}^i\right|^2+\left|\mathrm{f}_{\mathrm{w}}^i\right|^2\right) & \{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}\in\mathbf{X}_{out}\end{cases}, \tag{21}$$

where $\mathbf{X}_{in}$ denotes the inner domain and $\mathbf{X}_{out}$ denotes the outer domain.

The loss term for the energy conservation equation is

$$\mathrm{MSE}_{f_k}=\begin{cases}\dfrac{1}{\mathrm{N}_{in}}\displaystyle\sum_{i=1}^{N_{in}}\left(\left|\mathrm{f}_{k_{in}}^i\right|^2\right) & \{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}\in\mathbf{X}_{in}\\ \dfrac{1}{\mathrm{N}_{out}}\displaystyle\sum_{i=1}^{N_{out}}\left(\left|\mathrm{f}_{k_{out}}^i\right|^2\right) & \{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}\in\mathbf{X}_{out}\end{cases}. \tag{22}$$

The loss term for the species transport equation is

$$\mathrm{MSE}_{f_{spec}} \tag{23}$$

$$=\begin{cases}\dfrac{1}{\mathrm{N}_{in}}\displaystyle\sum_{i=1}^{N_{in}}\left(\left|\mathrm{f}_{spec_{a_{in}}}^i\right|^2+\left|\mathrm{f}_{spec_{b_{in}}}^i\right|^2+\left|\mathrm{f}_{spec_{b_{in}}}^i\right|^2\right) & \{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}\in\mathbf{X}_{in}\\ \dfrac{1}{\mathrm{N}_{out}}\displaystyle\sum_{i=1}^{N_{out}}\left(\left|\mathrm{f}_{spec_{a_{out}}}^i\right|^2+\left|\mathrm{f}_{spec_{a_{out}}}^i\right|^2+\left|\mathrm{f}_{spec_{b_{out}}}^i\right|^2\right) & \{\mathrm{x}_f^i, \mathrm{y}_f^i, \mathrm{z}_f^i, \mathrm{t}_f^i\}\in\mathbf{X}_{out}\end{cases},$$

The detailed forms of all governing equations are listed in the Appendix. The network is trained by minimizing the total loss using the backpropagation algorithm, where every weight is updated by calculating the gradient in each weight with respect to a loss function with Eq. (24). It is repeated until the total loss is smaller than the stopping criterion. The model was trained by the optimization package, AdamOptimizer in TensorFlow.

$$w_{ij}^{*} = w_{ij} + \alpha * \frac{\partial MSE}{\partial w_{ij}} \tag{24}$$

where the w_{ij} is theweight for neuron j in layer i in the current training iteration, w_{ij}^{*} is theupdated weight for neuron j in layer i, and α is the learning rate.

As shown in Fig. 5, the entire training data set is composed of D_{emp} and D_{phy}. D_{emp} is the training data set used to calculate the empirical error, which is the set of the input and output variables of the network. D_{phy} is the collocation data set used to calculate the physics error, which is the set of only the input variables of the network. Collocation data are divided into two domains, and the physics losses are calculated separately using Eqs. (20)-(23). These physics losses are summed and combined with the empirical loss calculated by Eqs. (17)-(19) to determine the total loss, and the

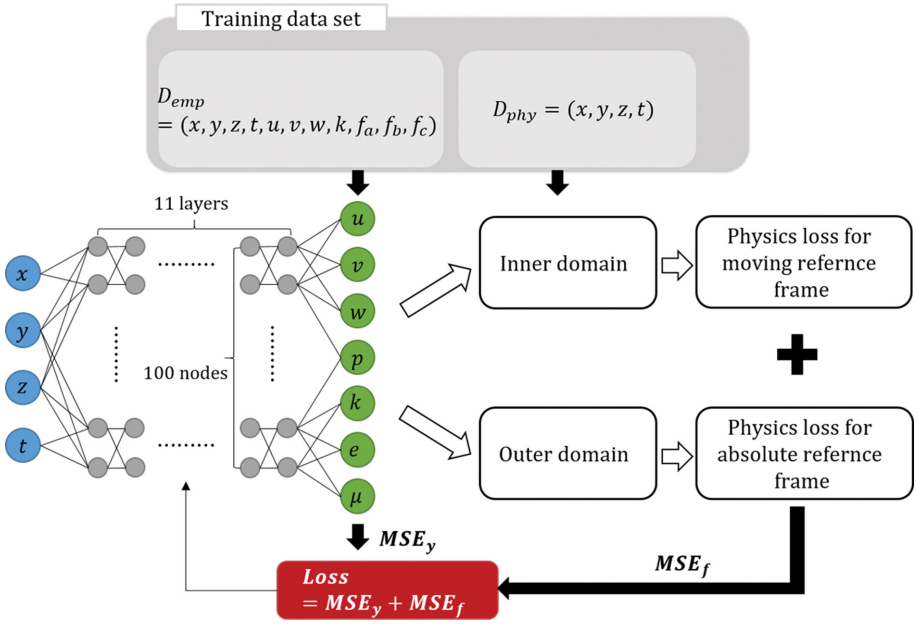

Fig. 5. PINN model structure for CSTR with Van de Vusse reaction.

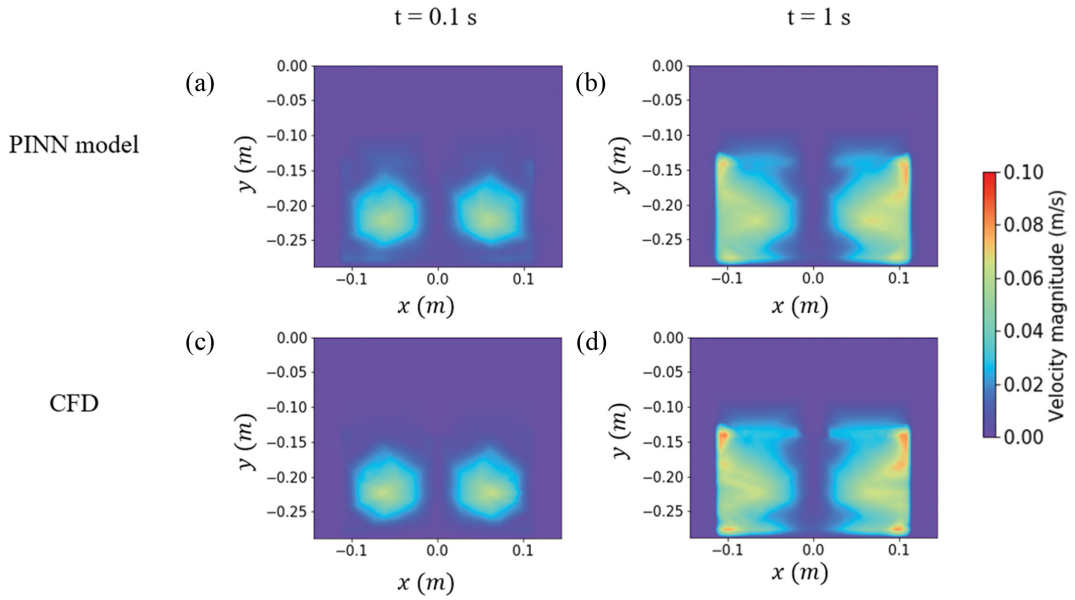

Fig. 6. Comparison of PINN model and CFD for velocity magnitude: PINN model solutions of the velocity magnitude at 0.0005 s and 0.005 s (a), (b), CFD solutions of the velocity magnitude at 0.0005 s and 0.005 s (c), (d).

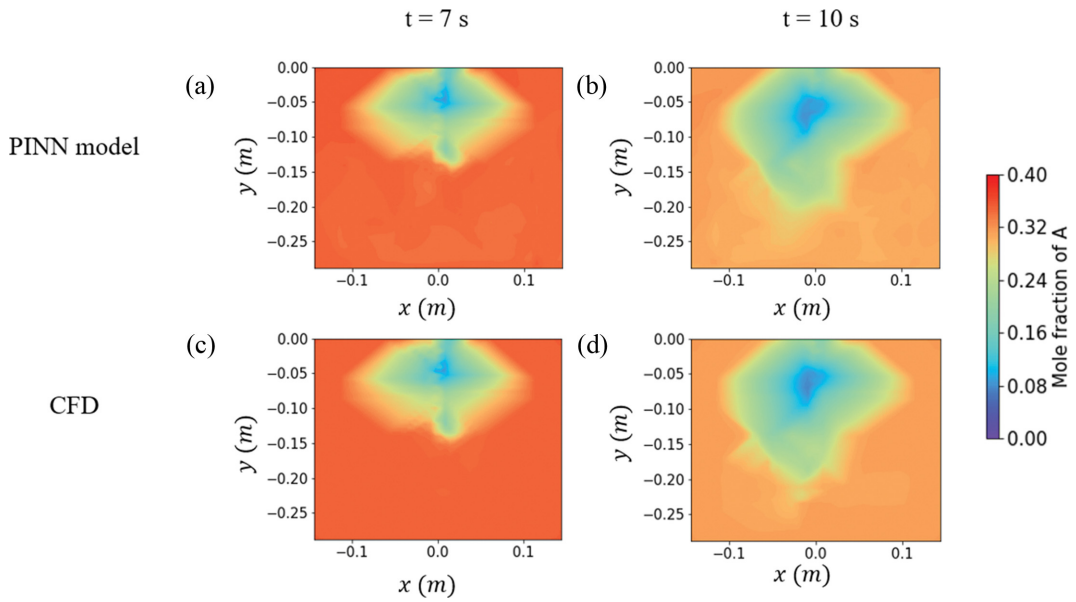

Fig. 7. Comparison of PINN model and CFD for mole fraction of A: PINN model solutions of the mole fraction of component A at 0.035 s and 0.05 s (a), (b), CFD solutions of the mole fraction of component A at 0.035 s and 0.05 s (c), (d).

network is trained by minimizing the total loss using the back-propagation algorithm. A total of $2 * 10^5$ data points were used for training data and $2 * 10^5$ data points were used for collocation points. The PINNs were trained using Tensorflow v1.10.0 with a NVIDIA GTX Titan GPU.

RESULT AND DISCUSSION

1. Model Verification

Fig. 6 and Fig. 7 show contour plots of the velocity magnitude and mole fraction of component A for the CFD model and the PINN model in a vertically sliced cross section of the reactor. The PINN model shows good agreement with the CFD model for all state variables, extracting the proper flow pattern. The L_2 relative error as presented in Eq. (25) of the PINN model is $1.98 * 10^{-1}$ for the velocity magnitude and $3.661 * 10^{-2}$ for the mole fraction of component A (Table 2). The first reason for this large L_2 relative error is that the CFD simulation does not provide an exact solution with a numerical error, which leads to errors in the PINN model. The second reason is that the system has many state variables to be learned and governed by highly nonlinear physical equations, which makes it difficult for the surrogate model to find the global optimum.

Table 2. L_2 relative error of PINN model for states variables

State variable	L_2 Relative error
Velocity magnitude	$1.98 * 10^{-1}$
Mole fraction of A	$3.66 * 10^{-2}$
Mole fraction of B	$2.21 * 10^{-2}$
Mole fraction of C	$1.25 * 10^{-2}$
Temperature	$1.22 * 10^{-1}$

$$L_2 \text{ relative error} = \sqrt{\frac{\sum_{i=1}^{N}(y_{real,\,i} - y_{pred,\,i})^2}{\sum_{i=1}^{N} y_{real,\,i}^2}} \tag{25}$$

Fig. 8 shows a comparison between the performance of PINN and NN in developing the surrogate model. We used the same network structure for the NN as the PINN, and this network is trained with the loss function as follows:

$$\text{MSE} = w_{emp}(\text{MSE}_u + \text{MSE}_T + \text{MSE}_Y), \tag{26}$$

where the weight was assigned the same value as in the PINN model.

NN shows a higher accuracy in predicting output values, with an empirical loss lower than that of the PINN. However, the physics loss of the NN is much larger than that of the PINN because the NN does not learn physical laws. In Fig. 9, the images of h) and i) represent the z-component of the learned velocity vector of the PINN and NN's, respectively, for the cross section in xz plane. While the PINN can extract the representative characteristic of the flow, NN shows noise that does not have any physical meaning because of overfitting. For a simple system, the empirical error of the NN is sometimes larger than that of the PINN; however, for the CSTR with an impeller, which has a complex geometry, PINN mostly has a larger empirical error than the NN. This is because the boundary condition, such as the non-slip condition on the surface of the impeller, that imposes non-linearity on the system makes it difficult for the PINN to obtain the optimum solution. Furthermore, the error of the governing equation, for example, the continuity equation, increases in regions where the velocity varies significantly, such as the boundary. Therefore, the loss term for the continuity equation implies that the model does not change significantly at the boundary condition, which yields a large empirical error. For this reason, the empirical error of PINN is larger

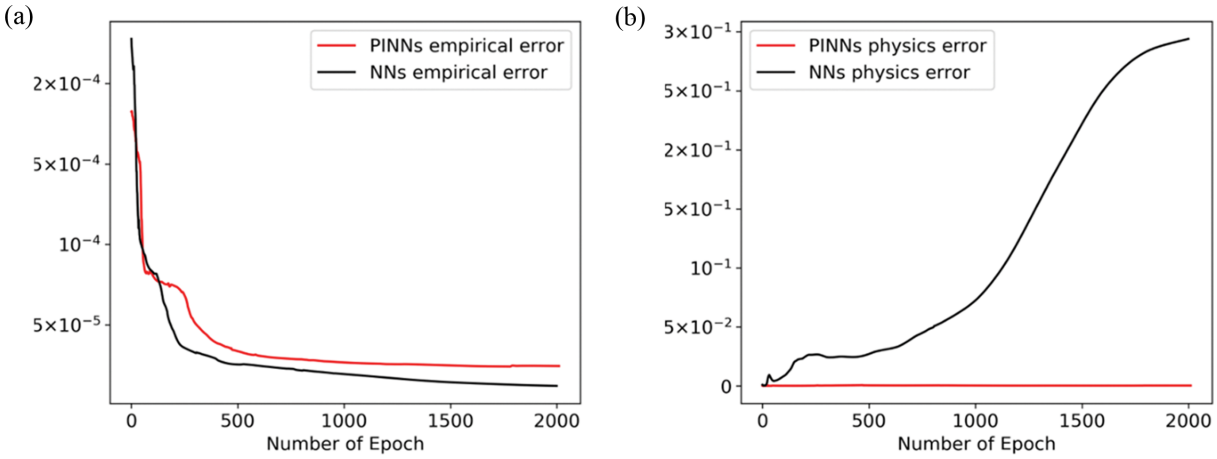

Fig. 8. Loss graph of PINN and NN: (a) Empirical loss of PINN and NN against number of epochs and (b) physics loss of PINN and NN against number of epochs.

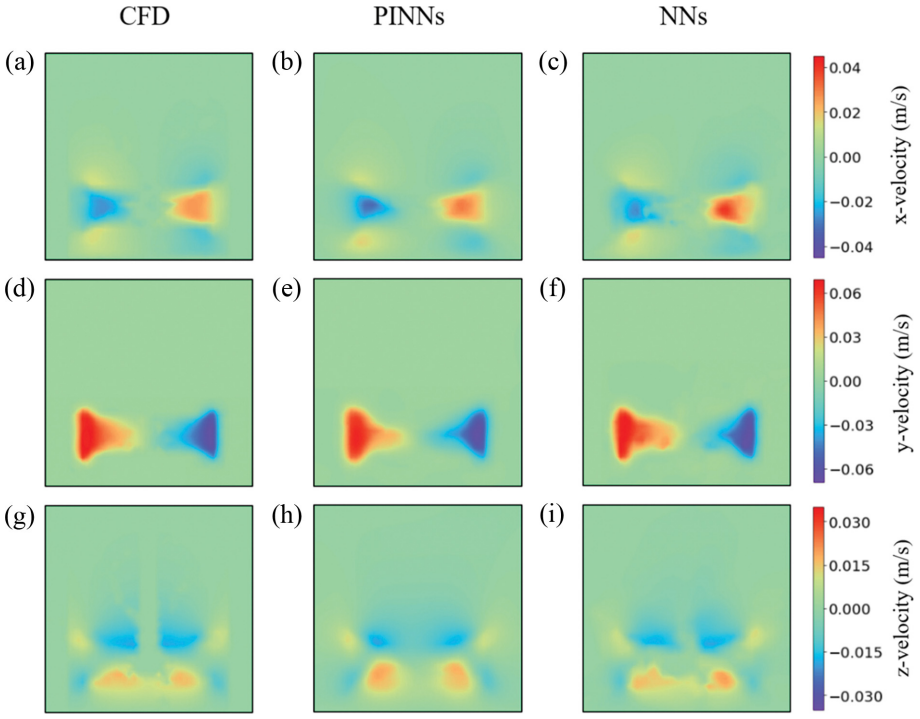

Fig. 9. Comparison of PINN model, NN model, and CFD for velocity components: x-direction of velocity solutions obtained using CFD, PINN model, and NN model (a), (b), (c), y-direction of velocity solutions obtained using CFD, PINN model, and NN model (d), (e), (f), z-direction of velocity solutions obtained using CFD, PINN model, and NN model (g), (h), (i).

than that of the NN, even though its physics error is much smaller than that of the NN.

2. Improvement of Model Performance

2-1. Mini-batch Training

We encountered a memory allocation error when more than approximately 5,000 data were trained at once using a 16G GPU. This is because TensorFlow pre-allocates the entire memory of the GPU [21]. Mini-batch training can resolve the memory error issue by dividing the entire dataset into several batches, making it possible to require memory only for the batch size, and not the total data size. The parameter is updated for every mini-batch of n training data points. When the batch size is 1, the so-called stochastic gradient descent, the model encounters the issue of high fluctuation of parameters during training [22]. A proper batch size should be specified to avoid model performance degradation and reduce the variance of the updated parameters for more stable convergence. Batch training not only resolves the memory allocation error but also improves the learning speed of the model. To find optimal batch size, eight cases with batch sizes ranging from 10 to 50,000 were simulated and learning performance was compared.

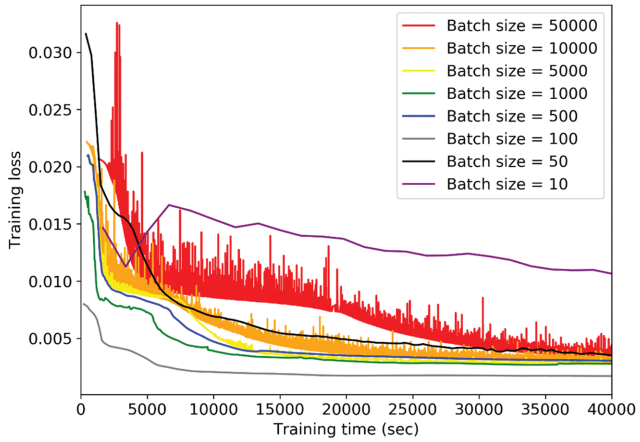

Fig. 10. Total loss of PINN model for different batch sizes against Training time (sec).

Fig. 10 is the result of comparing the training loss according to the learning time. As shown in following figure, the best learning performance is shown when the batch size is 100.

2-2. Weighted Loss Function

The loss function consists of two parts: empirical and physics terms. The value of each term should be maintained at a similar level for proper learning. To maintain the size of each term, we introduced a weight for every loss term. The weight of each physics loss term is presented in Table 3. Because the loss values of the momentum and energy conservation equations are very high, the weight of each term was set to a low value. The terms constituting the energy conservation equation are, for example, kVT, inherently larger than the terms of the momentum conservation equation because the scale of the temperature variable is much larger than that of the velocity variables. After setting the initial weights, we adjusted the weights further. We found that each weight affects the performance of the model. As shown in Fig. 11, when the weight of the empirical term is set to a high value, the contour plots show more obvious and detailed patterns. This is because the model behaves similarly to an NN as the weight of the empirical error increases. Accordingly, the empirical loss decreases and the physics loss increases as the weight of the empirical error is set to a high value, as shown in Fig. 12.

Table 3. Weight of each loss term

w_{phy_1}	1
w_{phy_2}	$1 * 10^{-6}$
w_{phy_3}	$1 * 10^{-18}$
w_{phy_4}	1

2-3. Similarity-based Sampling Strategy

When the data are selected randomly based on time and position, we face the problem of poor model accuracy when the state changes rapidly. Because our system starts from a stationary state,

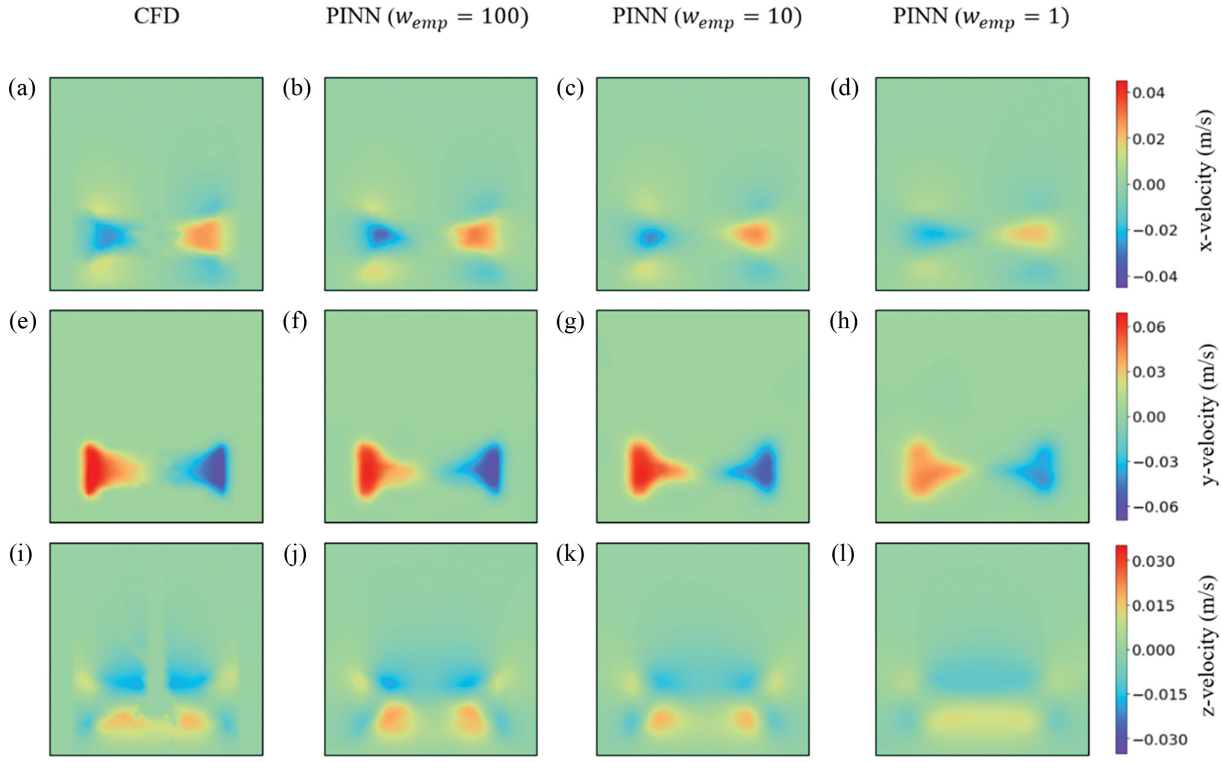

Fig. 11. Comparison of PINN models for several empirical term weights: x-direction of velocity solutions obtained using CFD and PINN model for empirical term weights of 100, 10, and 1 (a), (b), (c), (d), y-direction of velocity solutions obtained using CFD and PINN model for empirical term weights of 100, 10, and 1 (e), (f), (g), (h), z-direction of velocity solutions obtained using CFD and PINN model for empirical term weights of 100, 10, and 1 (i), (j), (k), (l).

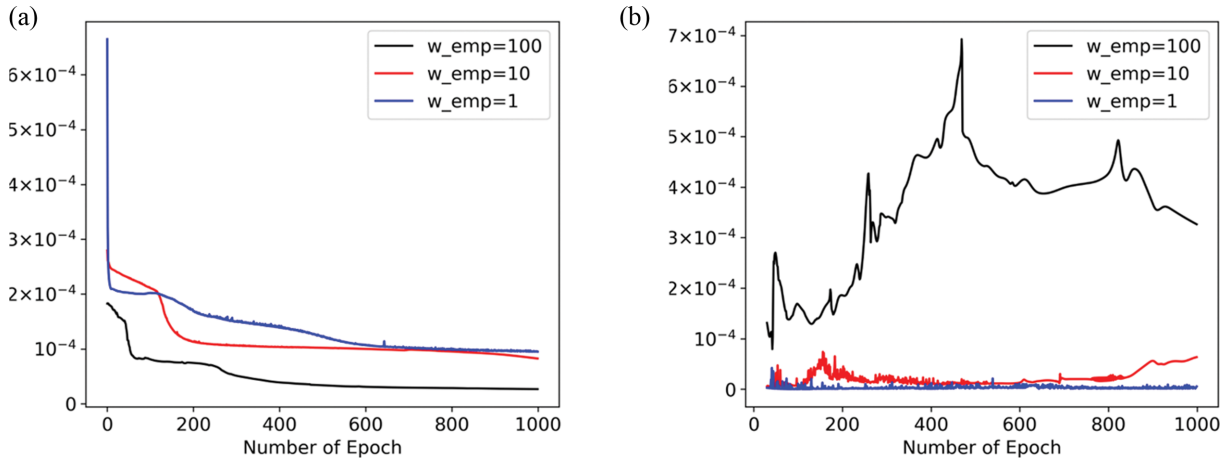

Fig. 12. Loss graph for various empirical term weights: (a) Empirical loss of PINN model for several empirical term weights against number of epochs, (b) physics loss of PINN model for several empirical term weights against number of epochs.

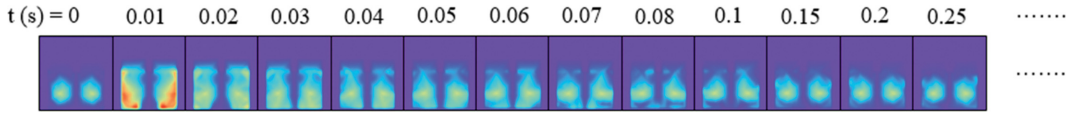

Fig. 13. Velocity magnitude with time.

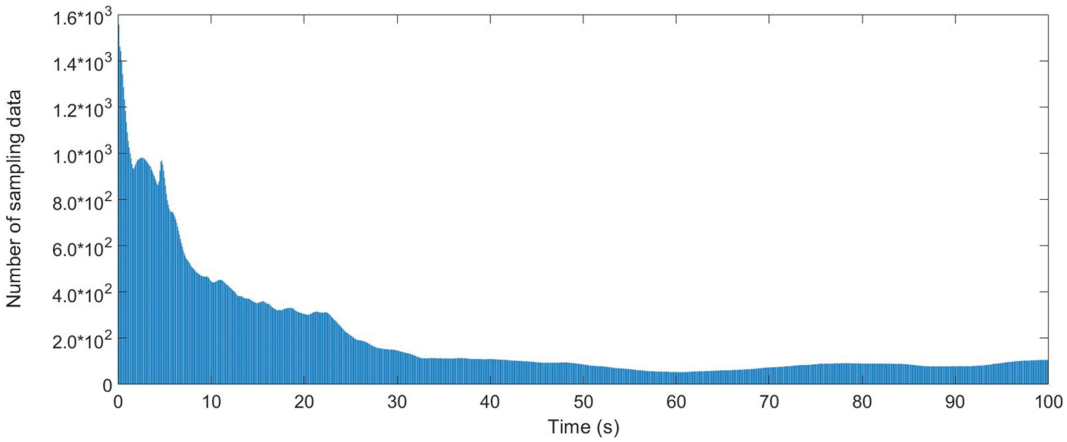

Fig. 14. Number of sampling data over time (total number data sets).

it changes rapidly at the beginning until it reaches a stationary state. This leads to degradation in the model performance, as shown in Fig. 13. To address this issue, we introduced a similarity-based sampling strategy. This sampling strategy is based on weighted sampling proposed by Kalal et al. [24], where the data are sampled based on given weights. In this study, we defined such weights with data similarities.

In a similarity-based sampling strategy, the number of sampled data in a certain domain is determined based on their similarity. We sampled more significantly changing data, and we reduced the number of samples where the data points at different time points showed a similar pattern. The similarities between the states are measured by the mean absolute error between adjacent times t_i and t_{i+1} for every state variable that is located at the same position as shown in Eq. (27) n is the number of data points for which similarity is calculated. The number of sampling data N_i for time index i is calculated using Eq. (28) and the distribution of the number of sampling data over time for the system is presented in Fig. 14. As expected, most of the sampling data were populated during the early stage in time, and the number of samples decreased as the system reached a steady state.

$$S_{i,j} = \frac{1}{\frac{1}{n}\Sigma\left|y^{j}_{i+1} - y^{j}_{i}\right|} \qquad \text{j=u, v, w, } f_a, f_t, f_c, \text{ k, i=time index} \tag{27}$$

$$N_i = N * \frac{\Sigma^{n}_{i=1}\Sigma^{N_{var}}_{j=1} S_{i,j}}{\Sigma^{N_{var}}_{j=1} S_{i,j}} \tag{28}$$

As shown in Fig. 15, the model with random sampling showed poor performance between 0 and 0.02 s because the number of sampled data points was insufficient compared to the high variety in the data pattern. Therefore, the model with the weighted sampling showed a considerably higher accuracy during the time interval of 0-0.02 s compared with the random sampling model. Numerically, as shown in Fig. 16, the model with similarity-based sampling achieved an L_2 relative error of up to five times lower than that of the random sampling model at $5 * 10^{-4}$ s.

3. Comparison of PINN Model with 1-D ODE Model

In the previous section, we investigated the performance of PINN, which finds a solution comparable to CFD with training data and governing equations. In particular, in the reactor model, the inhomogeneous state due to the internal flow plays an import-

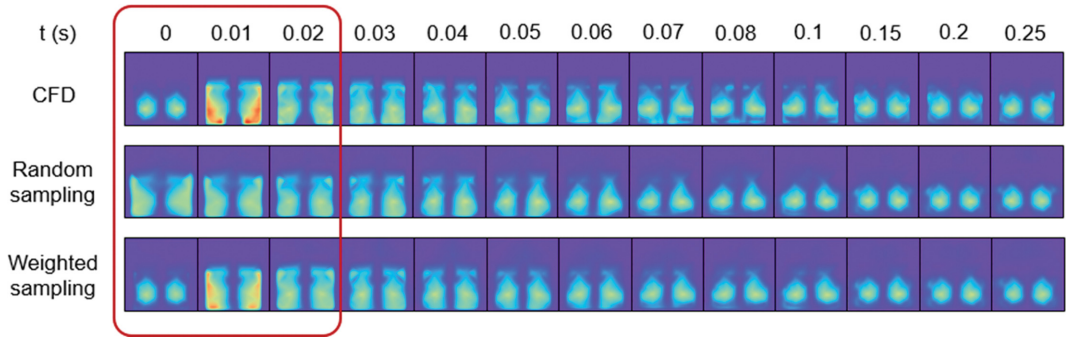

Fig. 15. Comparison of random sampling and weighted sampling.

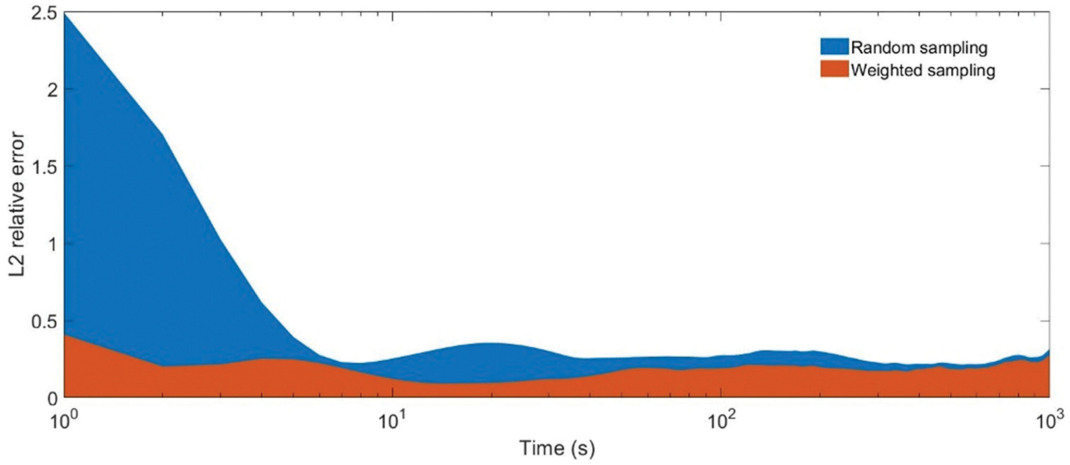

Fig. 16. L_2 relative error of random sampling and weighted sampling over time for velocity magnitude.

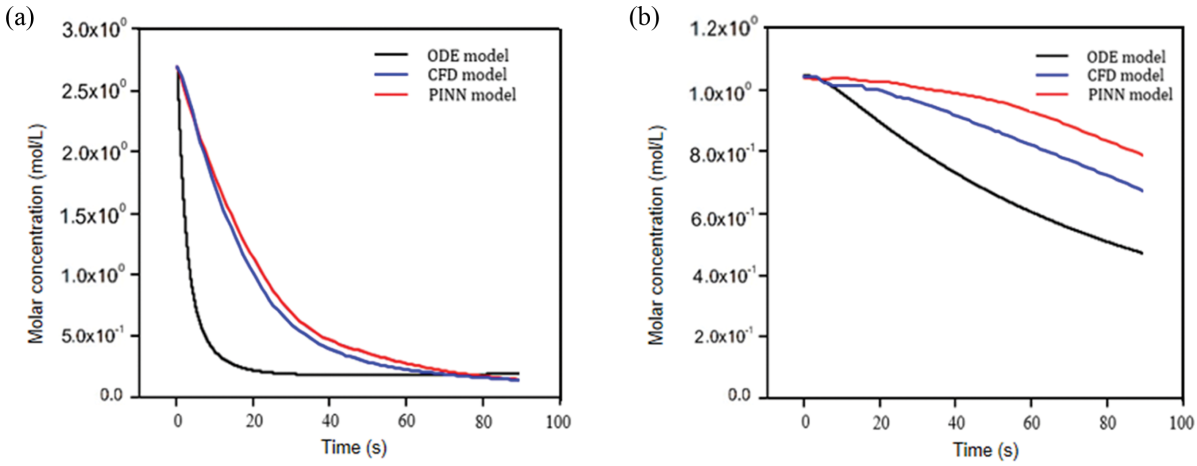

Fig. 17. Comparison of ODE, CFD, and PINN model: (a) Average mole concentration of component A and (b) average mole concentration of component B.

Table 4. Calculation times of ODE, PINN, and CFD (Calculation time: total elapsed time during simulation)

Model	ODE	PINN	CFD
Calculation time (s)	$7.21 * 10^{-2}$	$2.86 * 10^{2}$	$3.72 * 10^{5}$

ant role in determining product quality. The CFD model can simulate the fluid flow, but it has a limitation in industrial applications because of its long calculation time. Although a simple one-dimensional (1-D) ODE model can be simulated in real time, it cannot consider the fluid flow inside the reactor. PINN resolves both the problems of the CFD and the ODE model.

In this section, we investigate the difference in the results of the 1-D ODE, CFD, and PINN models, and how each model has different implementation times. As shown in Fig. 17, the molar concentration of component A calculated by the 1-D ODE model is smaller than that determined by the CFD and PINN models. This is because the components B, C, and D produced from component A are accumulated in the upper part of the reactor that cannot be stirred, hindering the reaction of component A. In the 1-D ODE model, states that are different from the actual phenomena are predicted because of the assumption of the homogeneous phase. From the results in Fig. 17, it is obvious that fluid flow should not be ignored because it has a large effect on the reactor system state, such as temperature and product specification. However, as presented in Table 4, the CFD model takes $3.72 * 10^{5}$ s using 16 CPU cores of Intel®Xeon® CPU E5-2697 v2 @2.70GHz to implement the model and that prevents its use in real-time applications. Even though it takes about $4 * 10^{4}$ s to train $2 * 10^{5}$ data sets, it takes very short time to implement the model after training. Therefore, the PINN model can be used for real-time optimization or control of a system that includes fluid flow. PINN is 1,300 times faster to implement than CFD, making it suitable for use in real-time problems; the accuracy of the PINN is also similar to that of CFD, as shown in Fig. 17.

CONCLUSIONS

We investigated the possibility of using PINN to develop a surrogate model of CSTRs with the van de Vusse reactions. We could find a solution that satisfies all the governing equations of mass, momentum, energy conservation, and species transport.

Because the system was governed by various types of physical equations and contained a multi-reference frame, we could not find a proper solution with the default setting of the PINN, which has been presented by Raissi [11]. Thus, a modified model framework that can implement a multi-reference frame system was introduced. In addition, we suggested three ways to enhance the performance of the PINN. First, we introduced mini-batch training. A full batch can only be used when the number of data is small because it requires a large amount of memory. In addition, it is known that a large batch size leads to poor generalization. A small batch can provide a model with a stochastic effect that allows the model to explore a wider range. For our system, we confirmed that PINN also has the advantages of a mini-batch that contributes to faster convergence and better generalization. Second, we used a weighted loss function. This was necessary because the scale of the physics loss was considerably larger than the empirical loss term, which led to poor learning performance. After we set each term to a similar size, we performed a more detailed adjustment of the weight. We tested the effects of weights on the training results. The model showed a more detailed pattern when the weight of the empirical term was larger, while the model likely learned a representative pattern when the weight of the empirical term was smaller. This could be adjusted according to the purpose of the model. However, it has a problem in that priori knowledge is required for determining the value of the weight because the scale of the physics loss terms is not known before the simulation. In future work, we will develop an adaptive learning method that can adaptively determine the weight of each term during training. Third, we introduced a similarity-based sampling strategy. The concept of "similarity" was proposed and the number of sampling data was allocated in inverse proportion to the similarity. Using similarity-based sampling considerably improved the model accuracy in cases when a sudden change in the system state occurred.

The solution from PINN has the same fidelity as that of CFD with similar accuracy. CFD is the only way to simulate a system that involves fluid flow. However, CFD has a critical problem of a high computational cost, which makes it difficult to apply to various problems such as system design and optimization. We confirmed that the PINN can dramatically reduce the calculation time while maintaining the fidelity of the CFD. However, it is difficult to obtain as high accuracy as CFD with PINN, especially for the more complex system. Therefore, the application of PINN is inevitably limited in application that requires as much accuracy as CFD.

In this work, we trained and developed the surrogate model for a single CFD case. We expect the model can be extended to a general surrogate model that can simulate various operating conditions with training on various cases. Like any other machine learning technique, PINN is hard to predict the completely outside of training domain. However, because the PINN learns physics equations, it will have a higher accuracy and would be more reliable in untrained regions than a black box model. This work shows that a system with complex flows can be simulated in real time and suggests the possibility that such a model can be used for system optimization and control.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2020R1A2C100550311).

NOTATION

A : pre-exponential factor
b^{λ} : bias parameter at layer λ
C_p : specific heat
C_{a_0} : initial concentration of component A
C_{b_0} : initial concentration of component B
C_{c_0} : initial concentration of component C

C_{d_0} : initial concentration of component D
C_{a_f} : concentration of A at input flow
D : diffusivity
E : internal energy
E_r : relative internal energy
E_a : activation energy
f_a : mole fraction of component A
f_b : mole fraction of component B
f_c : mole fraction of component C
f_d : mole fraction of component D
f_i : governing equations
{f_1: f_{cont}, f_2: $f_{u_{in}}$, f_3: $f_{u_{out}}$, f_4: $f_{v_{in}}$, f_5: $f_{v_{out}}$, f_6: f_w, f_7: $f_{k_{in}}$, f_8: $f_{k_{out}}$, f_9: $f_{spec_{a_{in}}}$, f_{10}: $f_{spec_{a_{in}}}$, f_{11}: $f_{spec_{b_{in}}}$, f_{12}: $f_{spec_{c_{in}}}$, f_{13}: $f_{spec_{a_{out}}}$, f_{14}: $f_{spec_{b_{out}}}$, f_{15}: $f_{spec_{c_{out}}}$}
f_{cont} : continuity equation
$f_{u_{in}}$: momentum equation for x-direction for inner domain
$f_{u_{out}}$: momentum equation for x-direction for outer domain
$f_{v_{in}}$: momentum equation for y-direction for inner domain
$f_{v_{out}}$: momentum equation for y-direction for outer domain
f_w : momentum equation for z-direction
$f_{k_{in}}$: energy conservation equation for inner domain
$f_{k_{out}}$: energy conservation equation for outer domain
$f_{spec_{a_{in}}}$: species transport equation for component A for inner domain
$f_{spec_{b_{in}}}$: species transport equation for component B for inner domain
$f_{spec_{c_{in}}}$: species transport equation for component C for inner domain
$f_{spec_{a_{out}}}$: species transport equation for component A for outer domain
$f_{spec_{b_{out}}}$: species transport equation for component B for outer domain
$f_{spec_{c_{out}}}$: species transport equation for component C for outer domain
H_r : relative total enthalpy
K : thermal conductivity
MSE : total loss
MSE_f : total physics loss
$MSE_{f_{cont}}$: continuity equation error
MSE_{f_u} : momentum conservation equation error
MSE_{f_k} : energy conservation equation error
$MSE_{f_{spec}}$: species transport equation error
MSE_T : output variable of temperature error
MSE_u : output variable of velocity error
MSE_Y : output variable of components mole fraction error
MSE_y : total empirical error
Mw_A : molecular weight
N_u : number of data points
N_f : number of collocation points for f
N_{var} : number of output variables
N : number of total training data
p : pressure
Q_f : input mass flowrate
R_i : net rate of the production of species *i*
$\vec{r}$: position vector from the origin of the moving system
$S_{i,j}$: similarity index at time *i* and component *j*
T : temperature
T_0 : initial temperature of reactor
T_f : input flow temperature
T_k : wall temperature
u : x-direction velocity component
u_j : output variables {u_1: u, u_2: v, u_3: w, u_4: k, u_5: f_a, u_6: f_b, u_7: f_c}
v : y-direction velocity component
$\vec{v}$: absolute velocity vector
$\vec{v}_r$: relative velocity vector
w : z-direction velocity component
$\vec{w}$: rotating angular velocity
$\mathbf{W}^{\lambda}$: weight parameter at layer λ
w_{emp} : weight for empirical loss term
w_{phy} : weight for physics loss term
X_{in} : inner domain
X_{out} : outer domain
x : x-direction location
Y_i : mass fraction of i^{th} species
y : y-direction location
$\mathbf{y}(\mathbf{x}^j)$: predicted output from the input x^j
$\mathbf{y}^j$: exact solution of output variables at x^j
z : z-direction location
ρ : density
μ : viscosity
$\bar{\tau}$: stress tensor
$\bar{\tau}_r$: relative stress tensor
$\mathcal{N}$: feed forward multi-layer NN
θ : network parameter
σ^{λ} : activation function at layer λ

REFERENCES

1. B. Sanderse, S. P. Van der Pijl and B. Koren, *Wind Energy*, **14**, 799 (2011).
2. F. J. Zajaczkowski, S. E. Haupt and K. J. Schmehl, *J. Wind Eng. Ind. Aerodyn.*, **99**, 320 (2011).
3. C. K. Harris, D. Roekaerts, F. J. J. Rosendal, F. G. J. Buitendijk, P. Daskopoulos, A. J. N. Vreenegoor and H. Wang, *Chem. Eng. Sci.*, **51**, 1569 (1996).
4. A. I. Forrester and A. J. Keane, *Prog. Aerosp. Sci.*, **45**, 50 (2009).
5. S. Park, J. Na, M. Kim and J. M. Lee, *Comput. Chem. Eng.*, **119**, 25 (2018).
6. Y. D. Lang, A. Malacina, L. T. Biegler, S. Munteanu, J. I. Madsen and S. E. Zitney, *Energy Fuels*, **23**, 1695 (2009).
7. J. Ling, A. Kurzawski and J. Templeton, *J. Fluid Mech.*, **807**, 155 (2016).
8. J. Na, K. Jeon and W. B. Lee, *Chem. Eng. Sci.*, **181**, 68 (2018).
9. G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto and L. Zdeborová, *Rev. Mod. Phys.*, **91**, 045002 (2019).
10. X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach and V. Kumar, arXiv preprint 2017, arXiv:1710.11431.
11. M. Raissi, P. Perdikaris and G. E. Karniadakis, *J. Comp. Phys.*, **378**, 686 (2019).
12. Z. Mao, A. D. Jagtap and G. E. Karniadakis, *Comput. Methods Appl. Mech. Eng.*, **360**, 112789 (2020).
13. X. I. A. Yang, S. Zafar, J. X. Wang and H. Xiao, *Phys. Rev. Fluids*, **4**, 034602 (2019).
14. X. Jin, S. Cai, H. Li and G. E. Karniadakis, *J. Comp. Phys.*, **426**, 109951 (2021).
15. S. Kuntanapreeda and P. M. Marusak, *Comput. Chem. Eng.*, **41**, 10 (2012).
16. C. T. Chen and S. T. Peng, *J. Process Control*, **15**, 515 (2005).

17. F. J. Doyle III, B. A. Ogunnaike and R. K. Pearson, *Automatica*, **31**, 697 (1995).
18. B. M. Åkesson and H. T. Toivonen, *J. Process Control*, **16**, 937 (2006).
19. K. Graichen, V. Hagenmeyer and M. Zeitz, *Comput. Chem. Eng.*, **33**, 473 (2009).
20. G. A. Ridlehoover and R. C. Seagrave, *Ind. Eng. Chem. Fund.*, **12**, 444 (1973).
21. C. Meng, M. Sun, J. Yang, M. Qiu and Y. Gu, In: Proc. of ML Systems Workshop in NIPS, December, 7 (2017).
22. J. Latz, *Stat Comput*, **31**, 1 (2021).
23. N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy and P. T. P. Tang, arXiv Preprint 2016, ArXiv:1609.04836.
24. Z. Kalal, J. Matas and K. Mikolajczyk, *BMVC*, **2**, 5 (2008).

APPENDIX

$$f_{cont} = u_x + v_y + w_z \tag{A1}$$

$$f_{u_{in}} = \rho(u_t + u_x u + u_y v + u_z w - \omega(u_x y - u_y x - v)) + p_x - \mu(u_{xx} + u_{yy} + u_{zz}) \tag{A2}$$

$$f_{u_{out}} = \rho(u_t + u_x u + u_y v + u_z w) + p_y - \mu(u_{xx} + u_{yy} + u_{zz}) \tag{A3}$$

$$f_{v_{in}} = \rho(v_t + v_x u + v_y v + v_z w - \omega(v_x y - v_y x + v)) + p_y - \mu(v_{xx} + v_{yy} + v_{zz}) \tag{A4}$$

$$f_{v_{out}} = \rho(v_t + v_x u + v_y v + v_z w) + p_y - \mu(v_{xx} + v_{yy} + v_{zz}) \tag{A5}$$

$$f_w = \rho(w_t + w_x u + w_y v + w_z w) + p_z - \mu(w_{xx} + w_{yy} + w_{zz}) \tag{A6}$$

$$f_{k_{in}} = \rho C_p\left(T_t - \frac{p_t}{\rho} + (uu_t - \omega u_t y + vv_t + \omega v_t x + ww_t) + uT_x + vT_y + wT_z\right) + \rho(u(uu_x - \omega u_x y + vv_x + \omega v + ww_x) + v(uu_y - \omega u_y y - \omega u + vv_y + \omega v_y x + ww_y) + w(uu_z - \omega u_z y + vv_z + \omega v_z x + ww_z) - K(T_{xx} + T_{yy} + T_{zz})) - \mu((u - \omega y)(u_{xx} + u_{yy} + u_{zz}) + (v + \omega x)(v_{xx} + v_{yy} + v_{zz}) + w(u_{xx} + u_{yy} + u_{zz}) + (u_y + v_x)^2 + (u_z + w_x)^2 + (v_z + w_y)^2 + 2(u_x^2 + v_y^2 + w_z^2)) \tag{A7}$$

$$f_{k_{out}} = \rho C_p(k_t + uT_x + vT_y + wT_z) + up_x + vp_y + wp_z - K(T_{xx} + T_{yy} + T_{zz}) - \mu(u(u_{xx} + u_{yy} + u_{zz}) + v(v_{xx} + v_{yy} + v_{zz}) + w(u_{xx} + u_{yy} + u_{zz}) + (u_y + v_x)^2 + (u_z + w_x)^2 + (v_z + w_y)^2 + 2(u_x^2 + v_y^2 + w_z^2)) \tag{A8}$$

$$f_{spec_{a_{in}}} = (f_{a_t} + (u - \omega y)f_{a_x} + (v + \omega x)f_{a_y} + wf_{a_z}) - D(f_{a_{xx}} + f_{a_{yy}} + f_{a_{zz}}) + f_a r_1 + \frac{r_3 \rho}{M w_a f_a^2 (2 - f_a - f_b - f_c)} \tag{A9}$$

$$f_{spec_{b_{in}}} = (f_{b_t} + (u - \omega y)f_{b_x} + (v + \omega x)f_{b_y} + wf_{b_z}) - D(f_{b_{xx}} + f_{b_{yy}} + f_{b_{zz}}) - f_a r_1 + f_b r_2 \tag{A10}$$

$$f_{spec_{c_{in}}} = (f_{c_t} + (u - \omega y)f_{c_x} + (v + \omega x)f_{c_y} + wf_{c_z}) - D(f_{c_{xx}} + f_{c_{yy}} + f_{c_{zz}}) - f_b r_2 \tag{A11}$$

$$f_{spec_{a_{out}}} = (f_{a_t} + uf_{a_x} + vf_{a_y} + wf_{a_z}) - D(f_{a_{xx}} + f_{a_{yy}} + f_{a_{zz}}) + f_a r_1 + \frac{r_3 \rho}{M w_a f_a^2 (2 - f_a - f_b - f_c)} \tag{A12}$$

$$f_{spec_{b_{out}}} = (f_{b_t} + uf_{b_x} + vf_{b_y} + wf_{b_z}) - D(f_{b_{xx}} + f_{b_{yy}} + f_{b_{zz}}) - f_a r_1 + f_b r_2 \tag{A13}$$

$$f_{spec_{c_{out}}} = (f_{c_t} + uf_{c_x} + vf_{c_y} + wf_{c_z}) - D(f_{c_{xx}} + f_{c_{yy}} + f_{c_{zz}}) - f_b r_2 \tag{A14}$$

$$y_x = \frac{\partial y}{\partial x} \tag{A15}$$

$$y_{xz} = \frac{\partial^2 y}{\partial x \partial z} \tag{A16}$$

$$r_1 = k_1 C_A \tag{A17}$$

$$r_2 = k_2 C_B \tag{A18}$$

$$r_3 = k_3 C_A^2 \tag{A19}$$