Nuclear Engineering and Technology 55 (2023) 2305–2314

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

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

Original Article

Solving partial differential equation for atmospheric dispersion of radioactive material using physics-informed neural network

Gibeom Kim, Gyunyoung Heo*

Department of Nuclear Engineering, Kyung Hee University, 1732 Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do, 17104, Republic of Korea

ARTICLE INFO

Article history:
Received 31 October 2022
Received in revised form
3 February 2023
Accepted 5 March 2023
Available online 20 May 2023

Keywords:
Atmospheric dispersion modeling
Physics-informed neural network (PINN)
Solutions of a partial differential equation

ABSTRACT

The governing equations of atmospheric dispersion most often taking the form of a second-order partial differential equation (PDE). Currently, typical computational codes for predicting atmospheric dispersion use the Gaussian plume model that is an analytic solution. A Gaussian model is simple and enables rapid simulations, but it can be difficult to apply to situations with complex model parameters. Recently, a method of solving PDEs using artificial neural networks called physics-informed neural network (PINN) has been proposed. The PINN assumes the latent (hidden) solution of a PDE as an arbitrary neural network model and approximates the solution by optimizing the model. Unlike a Gaussian model, the PINN is intuitive in that it does not require special assumptions and uses the original equation without modifications. In this paper, we describe an approach to atmospheric dispersion modeling using the PINN and show its applicability through simple case studies. The results are compared with analytic and fundamental numerical methods to assess the accuracy and other features. The proposed PINN approximates the solution with reasonable accuracy. Considering that its procedure is divided into training and prediction steps, the PINN also offers the advantage of rapid simulations once the training is over.

1. Introduction

It is important to well estimate the atmospheric dispersion of radioactive materials to prevent damage to residents and the environment. It is one of the essential parts of the environmental impact assessment of a nuclear power plant accident and emergency preparedness and response planning as well [1,2]. There is a governing equation describing atmospheric dispersion that can be derived from the law of mass conservation and takes the form of a second-order partial differential equation (PDE). Basically, estimating the atmospheric dispersion is performed by solving the PDE [3]. Methods for solving PDEs can be divided into analytic and numerical methods. The analytic method is applied primarily to a linear PDE that can apply the separation of variables; otherwise, various numerical methods such as finite differential methods (FDMs), Fourier analysis and computational fluid dynamics (CFD) can be applied. A representative analytic solution for the atmospheric dispersion is the Gaussian plume model, which is widely used in computational codes [3–5]. The Gaussian plume model is derived by simplifying the problem with multiple assumptions for fast computation. It predicts time-averaged concentrations of materials emitted from a point source with a constant emission rate and spread under a constant wind field. It cannot simulate dispersion over time, and it is restrictive to apply temporally and spatially varying parameters. The Gaussian puff model, which is a complemented alternative, can address other conditions and time-varying wind fields but is still based on the Gaussian plume model with several modified assumptions. However, such models can be useful when rapid decision-making is important, and it is widely used for local-scale estimation for instance air pollution near a roadway (line source) and atmospheric dispersion of hazardous materials due to accidents in the vicinity of a chemical plant or a nuclear power plant [5]. Numerical methods can be classified into Lagrangian and Eulerian models according to the way of kinematic descriptions. Lagrangian models track particles and describe their physical properties (e.g., position, velocity), including the puff model, CFD [6], and HYSPLIT (hybrid single-particle Lagrangian integrated trajectory) [7]. Eulerian models observe physical properties at fixed points in space including FDMs [8] and Fourier analysis [9]. The numerical methods require more computational cost than the analytic method but can carry out more precise calculations as conditions become more complex or when the analytic

* Corresponding author.
E-mail address: gheo@khu.ac.kr (G. Heo).

https://doi.org/10.1016/j.net.2023.03.009

methods are not available. Unlike analytic methods, numerical methods require fewer assumptions to approximate a solution and are relatively less restrictive to apply complicated parameters. In particular, they are used primarily to estimate the trajectory of air pollutants at an intercontinental or global scale, to which a Gaussian model is difficult to apply. Recently, a novel method called physics-informed neural network (PINN) has been proposed to approximate a solution of a PDE by optimizing a neural network [10]. The PINN is more intuitive in that it uses the original PDE without modifications or assumptions than other numerical methods and the analytic method as well. After the appearance of the PINN, many application studies were conducted. There are some nuclear engineering related studies applying the PINN. Regarding the reactor physics, Wang et al. [11], Dong et al. [12], and Yang et al. [13] solved different group neutron diffusion equations with different geometry of the reactor, and Schiassi et al. [14] solved the point kinetics equation. In addition, Zhang et al. [15] predicted the creep-fatigue life of 316 stainless steel, and Zhao et al. [16] predicted critical heat flux corresponding to the departure from nucleate boiling using the PINN approach.

In this paper, we present a method to approximate a solution for the atmospheric dispersion governing equation using the PINN. To verify the applicability of the PINN, we performed a comparative analysis with an exact solution (analytic method) and other fundamental numerical methods for a simple one-dimensional atmospheric dispersion problem. The PINN is an Eulerian model that describes properties using spatial coordinates. For comparative analysis, fundamental FDM and Fourier analysis belonging to the Eulerian model were used. We then expanded the problem to two dimensions and addressed the PINN applicability by qualitative analysis using several cases of model parameters. It should be noted that the main purpose of this paper is not to explore the best solution but to demonstrate how to solve an atmospheric dispersion PDE with the PINN and analyze its potential advantages and disadvantages.

Remainder of this paper is structured as follows. Section 2 provides an overview of the atmospheric dispersion model and reviews the analytic and numerical methods, including the PINN. Section 3 presents the results of applying the methods to the atmospheric dispersion model and compares their characteristics. Section 4 provides a summary of our findings.

2. Atmospheric dispersion models and solutions

2.1. Overview of atmospheric dispersion model

In this section, we briefly review the governing equation of atmospheric dispersion. Although many parameters affect atmospheric dispersion, such as chemical reactions, radioactive decay, and the vertical structure of the atmosphere, in this paper, we address the most basic form of the equation.

The governing equation of atmospheric dispersion can be derived as follows [17]. The mass concentration $C(\hat{r},t)$ $[kg/m^3]$ of a contaminant at location $\hat{r}=(x,y,z)\in\mathbb{R}^3$ $[m]$ and time $t\geq 0$ $[s]$ can be derived from the law of conservation of mass:

$$\frac{\partial C}{\partial t}+\nabla\bullet\vec{J}=S, \tag{1}$$

where $\vec{J}(\hat{r},t)$ $[kg/m^2s]$ represents the mass flux, and $S(\hat{r},t)$ $[kg/m^3s]$ is a source or sink that includes dry or wet deposition and chemical or radioactive decay of the contaminant.

In atmospheric dispersion, mass flux $\vec{J}$ is caused by the combined effects of diffusion and advection. The diffusion flux arises from turbulent eddy motion in the atmosphere and can be assumed to follow Fick's law. The advection flux is due to wind. The diffusion, advection and total flux can be expressed, respectively, as

$$\vec{J}_D=-K\nabla C \tag{2}$$

$$\vec{J}_A=C\vec{u} \tag{3}$$

$$\vec{J}=\vec{J}_D+\vec{J}_A=-K\nabla C+C\vec{u} \tag{4}$$

where $K=diag(K_x,K_y,K_z)$ $[m^2/s]$ is a diffusion coefficient that is a diagonal matrix of the turbulent eddy diffusivities, and $\vec{u}=(u_x,u_y,u_z)$ $[m/s]$ is a vector of wind velocities.

By substituting equation (4) into equation (1), we can obtain the governing equation of atmospheric dispersion, which has a PDE form as

$$\frac{\partial C}{\partial t}+\nabla\bullet\left(C\vec{u}\right)=\nabla\bullet(K\nabla C)+\mathrm{S}. \tag{5}$$

The following sections describe analytic and numerical methods for solving the PDE.

2.2. Analytic solution: Gaussian dispersion model

The Gaussian plume model is widely applied in simulation tools used by governments, industries, and environmental organizations due to its simplicity and computational speed. Several popular tools are based on this model, including AERMOD [18] and ADMS [19] (Atmospheric Dispersion Modeling System), which were developed by the US Environmental Protection Agency (EPA) and Cambridge Environmental Research Consultants (CERC), respectively. In the nuclear field, MACCS2 [20], and RASCAL [21] for risk assessment are representative tools based on the Gaussian model.

The Gaussian plume model is an analytic solution of equation (5) and is accompanied by many assumptions to simplify the problem resulting in rapid computation. For example.

- The source is a point source located at $(0,0,h)$ with a constant emission rate Q.$S=Q\delta(x)\delta(y)\delta(z-\mathrm{h})$.
- The wind velocity is constant, and its direction is to the positive x-axis.$\vec{u}=(u,0,0)$.
- For the x-axis, the wind velocity is sufficiently large such that the advection is dominant, and the diffusion effect is negligible.$K_x\frac{\partial^2 C}{\partial x^2}=0$.
- Parameters such as wind velocity and diffusivities do not change with time, and the time scale of interest is long enough to produce a steady-state solution.

With these assumptions, the derived Gaussian plume model (see derivation in Ref. [17]) is

$$\overline{C}(x,y,z)=\frac{Q}{2\pi\sigma_y\sigma_z\overline{u}}\exp\left(\frac{-y^2}{2\sigma_y^2}\right)\times\left(\exp\left(\frac{-(z-h)^2}{2\sigma_z^2}\right)+\exp\left(\frac{-(z+h)^2}{2\sigma_z^2}\right)\right), \tag{6}$$

where $\overline{C}(x,y,z)$ is the time-averaged mass concentration, Q is the emission rate, $\overline{u}=(u,0,0)$ is the time-averaged wind velocity at the height of release h, and σ_y and σ_z are horizontal and vertical diffusion parameters, respectively.

The Gaussian plume model predicts the average mass

concentration distribution over a sufficiently long period. Once the analytic solution is derived, it is simple and fast to calculate, although it might be difficult to apply to a problem that does not satisfy the assumptions or one with complicated wind or diffusivity parameters.

2.3. Numerical solutions

2.3.1. Finite difference method (FDM)

The rest of section 2 presents a brief review of the numerical methods for solving a PDE with its application on a simplified one-dimensional atmospheric dispersion model as equation (7). It is assumed that wind velocity and diffusivity are constant, and there is no source or sink term. Although multiple improved numerical methods are available, in this paper, we selected the most fundamental methods to solve the PDE for the purpose of a comparative study with the PINN.

$$\frac{\partial C}{\partial t}+u\frac{\partial C}{\partial x}-K\frac{\partial^2 C}{\partial x^2}=0 \tag{7}$$

$$C(x,0)=f(x)$$

$$C(-\infty,t)=0$$

$$C(\infty,t)=0$$

The FDM is one of the most common approaches to solve a PDE. It converts the PDE into a linear equation by approximating the derivatives to finite differences. FDMs can be divided into explicit and implicit methods, and the proper method should be chosen considering the type of PDE, convergence, and computational cost. Because equation (7) is a parabolic and linear PDE and includes first-order and second-order partial derivatives with respect to time t and space x, respectively, the forward time-centered space (FTCS) method was chosen that is the simplest FDM method for solving the parabolic form [22].

In this section, C is replaced by $C_{i,j}$ to describe the FTCS method. The subscript i represents the time index, and j represents the space index. The FTCS method uses the central difference in space and the forward difference in time. Thus, the derivative terms in equation (7) are replaced by equations (8)–(10).

$$\frac{\partial C}{\partial t}=\frac{C_{i+1,j}-C_{i,j}}{\Delta t} \tag{8}$$

$$\frac{\partial C}{\partial x}=\frac{C_{i,j+1}-C_{i,j-1}}{\Delta x} \tag{9}$$

$$\frac{\partial^2 C}{\partial x^2}=\frac{C_{i,j+1}-2C_{i,j}+C_{i,j-1}}{(\Delta x)^2} \tag{10}$$

Substituting equations 8–10 into equation (7) results in

$$\frac{C_{i+1,j}-C_{i,j}}{\Delta t}=K\frac{C_{i,j+1}-2C_{i,j}+C_{i,j-1}}{(\Delta x)^2}-u\frac{C_{i,j+1}-C_{i,j-1}}{\Delta x}. \tag{11}$$

Finally, equation (11) can be rewritten as

$$C_{i+1,j}=DC_{i,j+1}+EC_{i,j}+FC_{i,j-1} \tag{12}$$

$$D=\frac{2K\Delta t-u\Delta x\Delta t}{2(\Delta x)^2},\ E=\frac{2(\Delta x)^2-4K\Delta t}{2(\Delta x)^2},\ F=\frac{2K\Delta t+u\Delta x\Delta t}{2(\Delta x)^2}.$$

Equation (12) shows that the concentration C at time $i+1$ and position j can be calculated by the weighted sum of concentrations at previous time step i and positions $j-1$, j, and $j+1$, with some constant coefficients. For an advection-diffusion equation similar to the atmospheric dispersion equation, the stability conditions are known as $0\le\frac{K\Delta t}{(\Delta x)^2}\le\frac{1}{2}$ and $0\le\frac{u\Delta t}{\Delta x}\le 2\left(1-\frac{K\Delta t}{(\Delta x)^2}\right)$ [23]. Therefore, the stability conditions should be considered when determining Δt and Δx.

2.3.2. Fourier analysis

Fourier analysis is a useful method for solving a PDE because it can represent the PDE as an ordinary differential equation (ODE), which is much easier to solve. The solution can be obtained by applying a Fourier transform followed by an inverse Fourier transform, and it can be conducted efficiently by a fast Fourier transform (FFT) numerical algorithm. Fourier analysis is a method of representing an equation by the sum of orthogonal bases (e.g., eigenvalues and eigenfunctions) by transforming the original coordinates into more tractable coordinates. The Fourier transform is defined as equation (13) and (14) (see derivation in Ref. [24]).

$$f(x)=\mathscr{F}^{-1}(\widehat{f}(\omega))=\frac{1}{2\pi}\int_{-\infty}^{\infty}\widehat{f}(\omega)e^{i\omega x}d\omega \tag{13}$$

$$\widehat{f}(\omega)=\mathscr{F}(f(x))=\int_{-\infty}^{\infty}f(x)e^{-i\omega x}dx \tag{14}$$

where ω is a frequency.

Using a Fourier transform in space, $C(x,t)$ and its time derivatives in equation (7) can be represented as

$$\mathscr{F}(C(x,t))=\widehat{C}(\omega,t) \tag{15}$$

$$\mathscr{F}\left(\frac{\partial C}{\partial x}\right)=i\omega\widehat{C} \tag{16}$$

$$\mathscr{F}\left(\frac{\partial^2 C}{\partial x^2}\right)=\omega^2\widehat{C} \tag{17}$$

By substituting equation (15)–(17) into equation (7), we can obtain a simple first-order ODE:

$$\frac{\partial\widehat{C}}{\partial t}=-ui\omega\widehat{C}-K\omega^2\widehat{C}=-\left(ui\omega+K\omega^2\right)\widehat{C}. \tag{18}$$

The solution for equation (18) is as equation (19), and the solution of the original equation can be obtained through an inverse Fourier transform of equation (19).

$$\widehat{C}=\widehat{C}_0e^{-(ui\omega+K\omega^2)t} \tag{19}$$

where $\widehat{C}_0$ is the Fourier transform of the initial condition $C(x,0)$.

When applying a Fourier transform numerically, an FFT that dramatically reduces computational costs is commonly used with a large number of discretized data points and, as with an FDM, the

accuracy of the prediction increases with the number of data points.

2.3.3. Physics-informed neural network (PINN)

The PINN is a method that assumes a PDE as an arbitrary neural network and approximates the solution by optimizing the assumed neural network [11]. Because it uses the original form of the PDE without assumptions, it is less restrictive when applying parameters such as wind and a diffusion coefficient.

To solve the PDE, the latent solution of equation (7) $C(x,t)$ is assumed to be an arbitrary neural network for which the inputs are a location x and time t, and the output is concentration C.

$$C = NN_C(x,t) \tag{20}$$

We added a Python code corresponding to the equations in this section to improve comprehension. An arbitrary neural network for the solution $C(x,t)$ can be modeled using a TensorFlow 2.0 library function [25]. Here, the neural network was modeled with 11 layers, including input and output layers, and each hidden layer had 64 nodes. In addition, the input is normalized by the value between −1.0 and 1.0 to minimize the bias due to the different scales. Because concentration C was a positive value, for the last layer, the activation function was set to a sigmoid function, and the weights and bias were constrained to have a positive value.

```
#Modeling neural network
C_model = tf.keras.Sequential()
C_model.add(tf.keras.layers.InputLayer(input_shape=(2,)))        #input layer
C_model.add(tf.keras.layers.Lambda(lambda X: 2.0 * (X - lb)/(ub-lb) - 1.0))
#Normalization of input into (- 1.0 ≤ x < 1.0) (ub: upper bound, lb: lower bound)
…                                                                #hidden layer
C_model.add(tf.keras.layers.Dense(1, activation = 'sigmoid',
Kernel_initializer = 'glorot_normal',
Kernel_constraint = tf.keras.constraints.NonNeg(),
Bias_constraint = tf.keras.constraints.NonNeg()))                #ouput layer
```

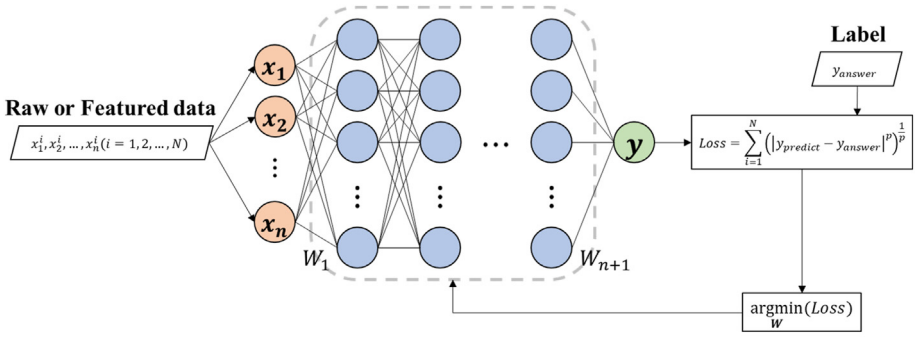
```
#Define C as an arbitrary neural network
def NN_C(x,t):
   C=C_model(x,t)
   Return C
```

The loss function for optimizing (training) the neural network is defined as follows. First, using equation (7), $f(x,t)$ is defined as

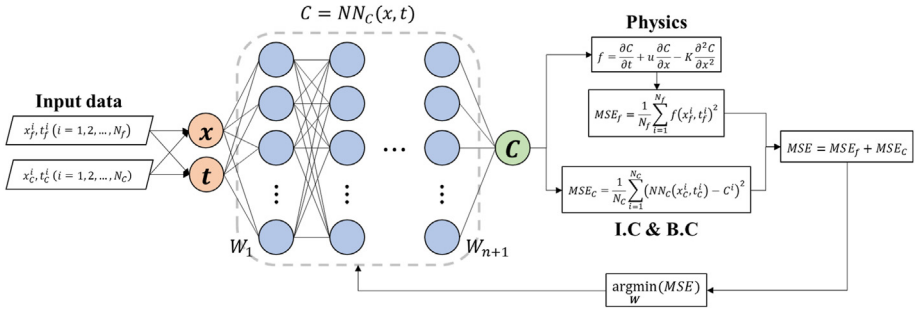
$$f = \frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} - K\frac{\partial^2 C}{\partial x^2} = \frac{\partial NN_C(x,t)}{\partial t} + u\frac{\partial NN_C(x,t)}{\partial x} - K\frac{\partial^2 NN_C(x,t)}{\partial x^2}. \tag{21}$$

Equation (21) can be written in Python code as follows, and the differential value of $NN_C(x,t)$ can be calculated by the "gradient" function in TensorFlow.

```
#Define f
def f(x,t):
   with tf.GradientTape(persistent = True) as tape:
      C=C_model(x,t)
      C_t=tape.gradient(C,t)
      C_x=tape.gradient(C,x)
      C_xx=tape.gradient(C_x,x)
   del tape
   Return C_t + u * C_x - K * C_xx
```



(a)



(b)

Fig. 1. The learning processes of a general neural network (a) and the PINN (b).

Rather than an FDM or Fourier analysis, the PINN is more intuitive in that it uses the original form of the PDE. $NN_C(x,t)$ can be optimized by finding its weights that minimize $f(x,t)$ (that is making $f(x,t)$ equal to zero) calculated with randomized inputs x and t.

Adding the error related to the initial and boundary conditions,

$$MSE_C = \frac{1}{N_C}\sum_{i=1}^{N_C}\left(NN_C\left(x_C^i, t_C^i\right) - C^i\right)^2 \tag{24}$$

where $\{x_f^i, t_f^i\}$ $(i = 1,2,3,\ldots,N_f)$ and $\{x_C^i, t_C^i, C^i\}$ $(i = 1,2,3,\ldots,N_C)$ are training datasets that consist of random values and the initial and boundary conditions, respectively.

```
#Define loss function
def loss(self, x_C, t_C, X_f, t_f, C_train):
    C_pred = NN_C(x_C,t_C)
    f_pred = f(x_f,t_f)
    return tf.reduce_mean(tf.square(f_pred))
       + tf.reduce_mean(tf.square(C_train - C_pred))
```

the loss function becomes

$$MSE = MSE_f + MSE_C \tag{22}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} f\left(x_f^i, t_f^i\right)^2 \tag{23}$$

In conclusion, the latent solution $C(x,t)$ can be obtained by optimizing the weights of $NN_C(x,t)$. In other words, it is to find the weights that minimize the loss function *MSE*. While neural network (supervised) learning in general constructs a loss function as an error between the predictions of the neural network and the labeled dataset that is an answer given by the user, the answer for the PINN is given by physics, i.e., it is physics-informed (Fig. 1).

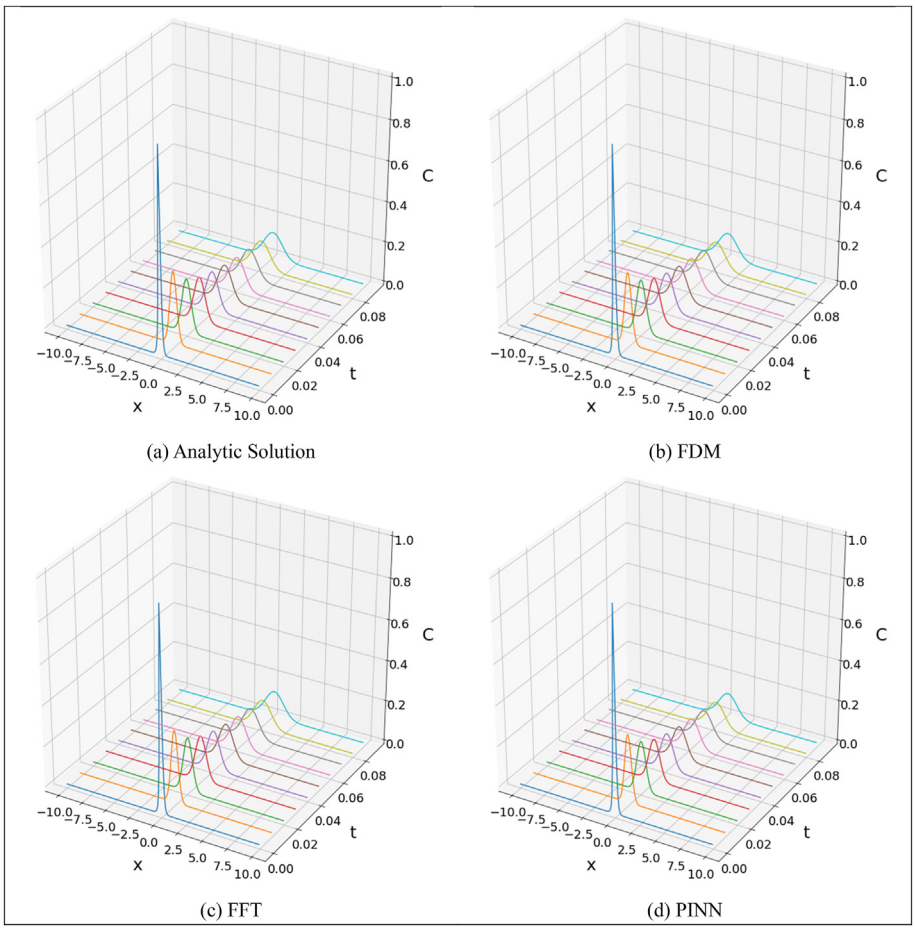


Fig. 2. Solutions for a 1-D atmospheric dispersion equation using analytic and numerical methods.

Table 1
Peak values over time of the results for each method.

Time(sec)	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
Analytic	0.354	0.265	0.219	0.192	0.173	0.159	0.147	0.138	0.130
FDM	0.350	0.260	0.214	0.188	0.168	0.155	0.143	0.134	0.127
FFT	0.354	0.265	0.219	0.193	0.173	0.159	0.147	0.138	0.130
PINN	0.330	0.248	0.205	0.180	0.163	0.151	0.140	0.132	0.124

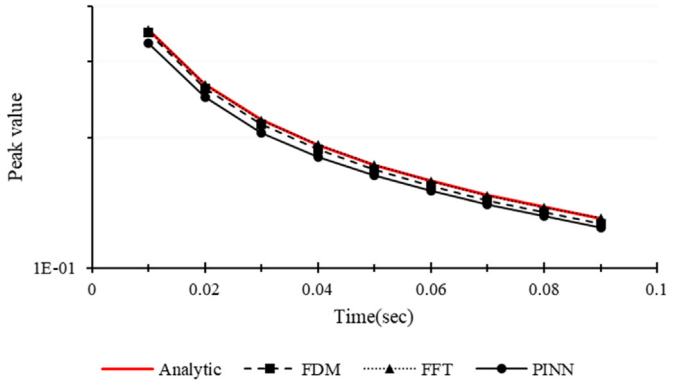


Fig. 3. Peak values over time of the results for each method.

3. Results

This section presents the applicability of the PINN through a comparative study with other methods and simple case studies. First, for a comparative study, we applied the aforementioned methods to a simple one-dimensional problem for which it is relatively easy to observe a correct answer and show the results intuitively to compare. Although there is a variety of improved numerical algorithms, most fundamental methods are used for the purpose of comparative study. How well the PINN performs was explored in the comparative study, after which we expanded the problem to two dimensional and applied the PINN with several model parameter cases to see whether it could reasonably estimate the qualitative effects of the model parameters.

3.1. Solving a 1-D atmospheric dispersion model

A comparative study of the analytic and numerical methods was conducted with a simplified one-dimensional atmospheric dispersion equation (7). We predicted atmospheric dispersion in a 20 [m] ($-10 < x < 10$) one-dimensional space for 0.1 [s]. The initial condition was assumed to be $C(x,0) = \frac{1}{\cosh(10x)}$, in which the distribution was concentrated in the center. The wind speed u and the diffusion coefficient K were assumed to be a constant that is regardless of space and time as 5 m/s and 5 m^2/s, respectively. The space interval Δx and time step size Δt were set to 0.05 $[m]$ and 0.001 $[s]$, respectively, given the stability condition of the FDM. All

Table 2
RMSE between analytic solution and PINN according to the configuration of the neural network.

					(* Number of hidden layers: 9)
Number of nodes	32	48	64	80	96
RMSE	3.088E-04	1.480E-04	1.529E-04	1.270E-04	**1.257E-04**
					(* Number of nodes layers: 64)
Number of hidden layers	3	6	9	12	15
RMSE	3.032E-04	**1.251E-04**	1.529E-04	1.597E-04	1.631E-04

codes for the results are available at https://github.com/gibeom92/atmospheric_dispersion_PINN.

The exact solution (see derivation in Ref. [26]) for equation (7) is

$$C(x,t)=e^{\frac{u}{2K}\left(x-\frac{u}{2}t\right)}A(x,t) \tag{25}$$

$$A(x,t)=\sum_{n=1}^{\infty}\left[D_{2n}\sin n\frac{\pi}{L}xe^{-K\left(n\frac{\pi}{L}\right)^2 t}+E_{2n-1}\cos\frac{2n-1}{2}\frac{\pi}{L}xe^{-K\left(\frac{2n-1}{2}\frac{\pi}{L}\right)^2 t}\right]$$

$$D_{2n}=\frac{1}{L}\int_{-L}^{L}f(x)e^{-\frac{u}{2K}x}\times\sin n\frac{\pi}{L}xdx$$

$$E_{2n-1}=\frac{1}{L}\int_{-L}^{L}f(x)e^{-\frac{u}{2K}x}\times\cos\frac{2n-1}{2}\frac{\pi}{L}xdx$$

where $f(x)$ is the initial condition $\frac{1}{\cosh(10x)}$, and L is 10 $[m]$, which is half the size of the space domain in this case.

As shown in equation (25), the exact solution is complicated unless the assumptions mentioned in section 2.2 are not applied. It has a Fourier series form, which implies that the accuracy varies with the number of terms used for the approximation (in this case, 500 terms are used). For the PINN, the training data consisted of 30,000 randomly sampled inputs for function f (equation (21)) and 10,000 inputs related to the initial and boundary conditions.

Fig. 2 provides the results of predictions using the analytic and numerical methods. The concentration distribution is plotted every 0.01 s for 0.1 s. Table 1 and Fig. 3 show the peak values over time of the results to facilitate the comparison between the results. The numerical methods including the PINN predicted well the qualitative trend in which the concentration decreases exponentially, and the center moves slightly to the right (a positive direction) over time, as shown in equation (25). Fig. 4 shows the root mean square errors between the exact (analytic) and the numerical solution over the prediction time step. The error of the PINN decreases over the prediction time step. It is presumed that this was because the PINN smoothly predicted the initial peak of the exact solution, which exponentially decreases over time. It is also possible that this error can be reduced by tuning the hyperparameters of the PINN, including the number of layers and nodes. Table 2 shows the RMSE between the analytic solution and PINN according to the configuration of the neural network. The RMSE is calculated for 2000 coordinate points (200 ($-10\le x<10, dx=0.1$) $\times$ 10 ($0\le t<0.1, dt=0.01$)). It is shown that the RMSE decreases as the number of nodes increases. For the number of layers, the RMSE is minimized by 6 layers. Other than the number of nodes and layers, there are lots of hyperparameters such as weight initializer, optimizer, activation functions, and so on. The performance of the model can be improved by the hyperparameters. It should be noted that because the neural network is a black box model, it is difficult to forecast the performance according to the hyperparameter. Considering the disadvantage such as overfitting, computational cost and time, appropriate hyperparameters should be found and selected.

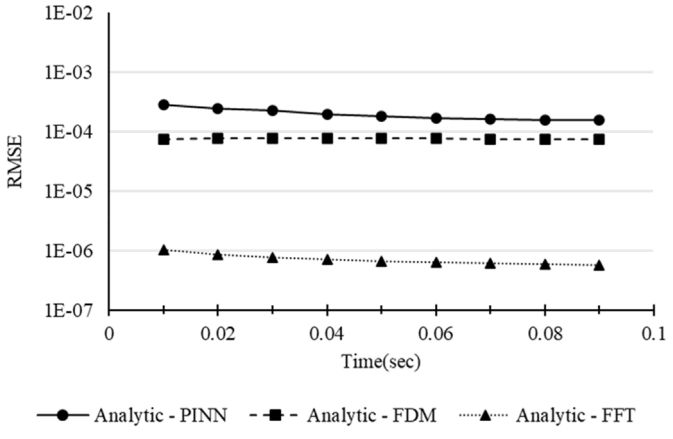


Fig. 4. Root mean square error (RMSE) between the analytic and numerical solutions.

Given that the PINN procedure is separated into training and prediction steps, the PINN can offer an advantage in terms of computational cost compared with the other two numerical methods. The training step may require considerable computational cost and time depending on the configuration of the neural network. However, the time it takes to predict is very short once the training is complete. In addition, FDM and Fourier analysis need to calculate all coordinates (time and space) even if we want to predict only one point. On the other hand, PINN is a function whose inputs are only the coordinates of the point of interest ($NN_C(x,t)$). It is useful in that, for instance, in the event of a radiological emergency, PINN can predict the concentration of radioactive materials only for the population concentrated areas. Furthermore, unlike the other numerical methods, the PINN uses a large number of randomly sampled coordinate points, not discretized coordinate points. Therefore, PINN works as a continuous function that can predict any points regardless of Δx and Δt.

3.2. Solving a 2-D air dispersion model using a PINN

In section 3.1, it was shown that the PINN has an accuracy comparable to that of the analytic and numerical solutions. Based on the reasonable performance of PINN, as confirmed in the one-dimensional atmospheric dispersion, we expanded the study case to a two-dimensional problem. In this case, we predicted atmospheric dispersion in a 20 $\times$ 20 $[m^2]$ ($-10<x<10$, $-10<y<10$) two-dimensional space for 3 [s]. The initial condition was assumed to be $C(x,0)=\frac{1}{\cosh(10\sqrt{x^2+y^2})}$. The diffusion coefficient K was assumed to increase over distance according to U.S.NRC [21]. The

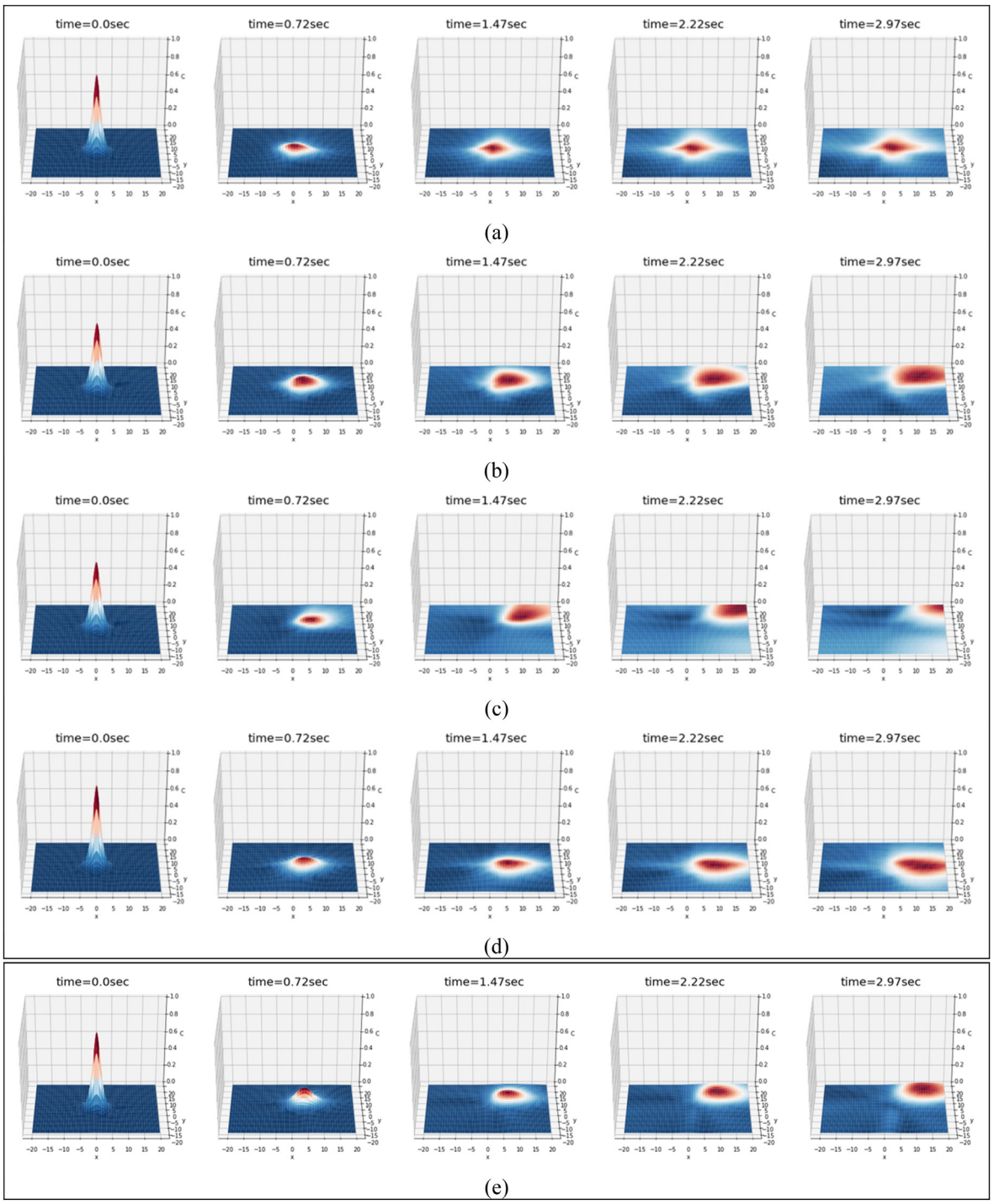
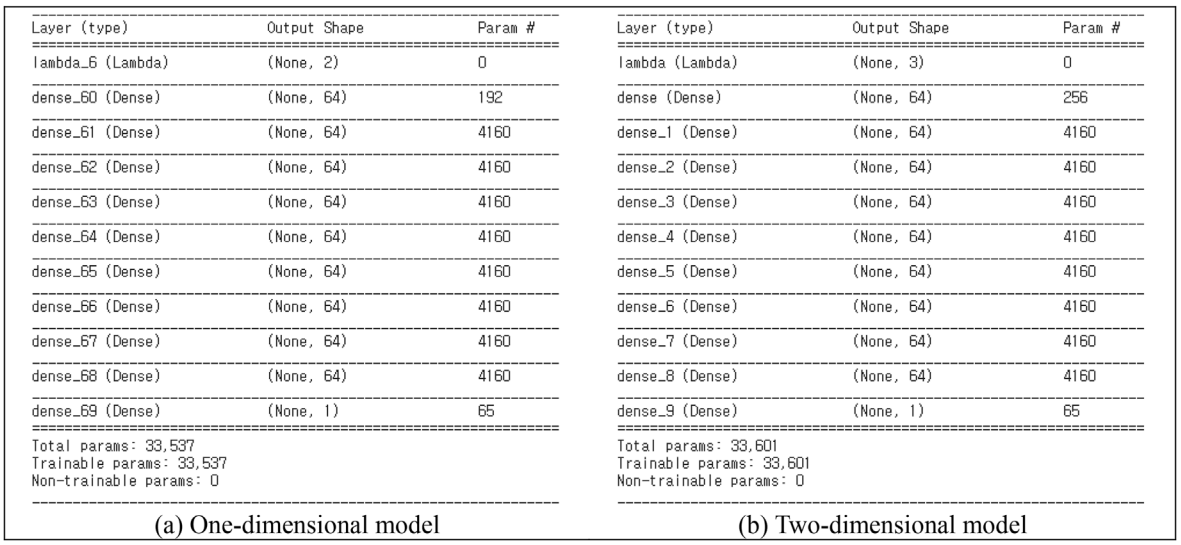


Fig. 5. Solutions for a 2-D atmospheric dispersion equation using PINN with different parameters.

training data consisted of 50,000 randomly sampled inputs for the function f, as shown in equations (21), and 10,000 inputs related to the initial and boundary conditions.

The prediction was conducted with five cases of model parameters. Fig. 5 (a)–(d) provide the results of the prediction with a wind vector $\vec{u}$ equal to $(2,2)$, $(5,5)$, $(10,10)$, and $(5,0)$ $[m/s]$, respectively, and Fig. 5 (e) shows the results with a wind vector $\vec{u}$ equal to $(5,5)$ and a smaller diffusion coefficient K.

As expected, in cases (a)–(c), the stronger the wind, the faster the distribution shift. Case (d) shows the change in direction of the distribution by wind vector. While the distribution of cases (a)–(c) moved to the diagonal direction, the distribution of case (d) moved to the right. A comparison of cases (b) and (e) shows the effect of diffusion coefficient K. The distribution of Case (e) for which the diffusion coefficient is smaller spreads slower than Case (b). We concluded that the PINN model appropriately learned and reflected the effects of the parameters involved in atmospheric dispersion. Although, for the two-dimensional problem, we did not present comparisons with other methods or errors with the correct answer (analytic solution), a qualitative analysis of the model parameters



Layer (type)	Output Shape	Param #
lambda_6 (Lambda)	(None, 2)	0
dense_60 (Dense)	(None, 64)	192
dense_61 (Dense)	(None, 64)	4160
dense_62 (Dense)	(None, 64)	4160
dense_63 (Dense)	(None, 64)	4160
dense_64 (Dense)	(None, 64)	4160
dense_65 (Dense)	(None, 64)	4160
dense_66 (Dense)	(None, 64)	4160
dense_67 (Dense)	(None, 64)	4160
dense_68 (Dense)	(None, 64)	4160
dense_69 (Dense)	(None, 1)	65

Total params: 33,537
Trainable params: 33,537
Non-trainable params: 0

(a) One-dimensional model

Layer (type)	Output Shape	Param #
lambda (Lambda)	(None, 3)	0
dense (Dense)	(None, 64)	256
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 64)	4160
dense_5 (Dense)	(None, 64)	4160
dense_6 (Dense)	(None, 64)	4160
dense_7 (Dense)	(None, 64)	4160
dense_8 (Dense)	(None, 64)	4160
dense_9 (Dense)	(None, 1)	65

Total params: 33,601
Trainable params: 33,601
Non-trainable params: 0

(b) Two-dimensional model

Fig. 6. Comparison of one-dimensional and two-dimensional PINN models.

was performed through several representative cases, and reasonable results were presented. Furthermore, if the correct answer is given by other methods, there is a possibility that the model can be improved by tuning various (hyper) parameters involved in the training process of the PINN. Meanwhile, the time it took to train the model was approximately 3700 and 4000 s for the one-dimensional and two-dimensional problems, respectively. This differed from our expectation that the training time would increase steeply (e.g., quadratic growth) with the number of dimensions. We can find one reason in the configuration of the neural network model. When we applied the PINN to the two-dimensional problem, we added only one node to the input layer of the one-dimensional model, and the number of weights to be trained increased by only 64, which is the number of nodes in the first hidden (dense) layer (Fig. 6). This was negligible relative to the total number of weights. In other words, the number of dimensions is not a dominant factor in the training time. The training time depends primarily on the size of the training data and hyperparameters such as the learning rate and the number of layers and nodes. This represents one advantage relative to other numerical methods for which the computational cost is dependent on the number of points to be calculated. However, a PINN also suffers from the limitations of neural networks in that it is less explainable and may require multiple trial-and-error to determine the optimal hyperparameters.

4. Conclusions

For the prediction of atmospheric dispersion, many computational codes use a Gaussian dispersion model, which is an analytic solution. However, because a Gaussian dispersion model accompanies various assumptions to simplify the problem, it can be difficult to properly consider parameters that affect the atmosphere, such as wind and diffusion coefficients in cases of being against the assumptions. Here, we presented a method for predicting atmospheric dispersion using a recently proposed PINN method. To see its applicability, we conducted a comparative analysis of analytic and fundamental numerical methods by applying them to a simple one-dimensional atmospheric dispersion problem. We found that the PINN achieved reasonable accuracy relative to other numerical methods. Furthermore, the PINN accuracy can be improved by tuning the hyperparameters. Although the training may be time-consuming, considering the characteristic of machine learning that the training and prediction process are divided, there is an advantage in terms of the time it takes to perform predictions. Once the training is over, the prediction time of the PINN model is relatively negligible. In addition, the PINN is able to predict an interested point only, while the other methods need to calculate all coordinate points. In addition, we expanded the problem to two dimensions and performed a qualitative analysis with several representative model parameter cases. The PINN well predicted atmospheric dispersion patterns according to the parameters regarding the advection and diffusion. It should be noted that this study is not to provide the best solution but to demonstrate the applicability of the PINN and analyze its potential advantages and disadvantages. To this end, a simple case study was set to facilitate the comparison and observation of prediction pattern changes. For the practical application, the performance of the PINN model needs to be validated and verified in the larger scale of time and space. In the future study, more complex model parameters will be addressed using the PINN model in the larger scale of time and space. It can be said that the major advantage of the PINN over existing data-driven approaches is that it can reflect not only the information on data but also physics. In this work, the PINN model is mainly trained by physics. We are planning to train the PINN model that performs stable prediction even on the more complex situation and larger scales using real data or simulation data from other programs, such as RASCAL, as well as physics. In addition, to find optimal hyperparameters efficiently and automatically, a training pipeline including data collection, preprocessing, model validation step, and so on will be constructed. Regarding the training time, we are also planning to study applying transfer learning, which uses a general pre-trained model that enables to reduce the time and enhance the training efficiency, to the PINN.

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.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government

(MSIP:Ministry of Science, ICT and Future Planning) (No. NRF-2021M2D2A1A02044210).

References

[1] US Nuclear Regulatory Commission, Standard review plans for environmental reviews for nuclear power plants, Supplement 1: Operating License Renewal (2013). NUREG-1555, Supplement 1, Revision 1.
[2] IAEA, Preparedness and Response for a Nuclear or Radiological Emergency, 7, IAEA Safety Standards Series No. GSR Part, Vienna, 2015. IAEA.
[3] A. De Visscher, Air Dispersion Modeling: Foundations and Applications, John Wiley & Sons, 2013.
[4] N.S. Holmes, L. Morawska, A review of dispersion modelling and its application to the dispersion of particles: an overview of different dispersion models available, Atmos. Environ. 40 (30) (2006) 5902–5928.
[5] Á. Leelőssy, F. Molnár, F. Izsák, Á. Havasi, I. Lagzi, R. Mészáros, Dispersion modeling of air pollutants in the atmosphere: a review, Cent. Eur. J. Geosci. 6 (3) (2014) 257–278.
[6] A. Baklanov, Application of CFD methods for modelling in air pollution problems: possibilities and gaps, in: Urban Air Quality: Measurement, Modelling and Management, Springer, Dordrecht, 2000, pp. 181–189.
[7] A.F. Stein, R.R. Draxler, G.D. Rolph, B.J. Stunder, M.D. Cohen, F. Ngan, NOAA's HYSPLIT atmospheric transport and dispersion modeling system, Bull. Am. Meteorol. Soc. 96 (12) (2015) 2059–2077.
[8] N. Sanín, G. Montero, A finite difference model for air pollution simulation, Adv. Eng. Software 38 (6) (2007) 358–365.
[9] R. Cervantes-Muratalla, L.P. Ramírez-Rodríguez, T. Mendívil-Reynoso, C.R.A. Murguia-Romero, D. García-Bedoya, Application of fourier transform for distribution of pollutants in air, World J. Environ.l Biosci. 9 (2) (2020) 1–7.
[10] 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.
[11] J. Wang, X. Peng, Z. Chen, B. Zhou, Y. Zhou, N. Zhou, Surrogate modeling for neutron diffusion problems based on conservative physics-informed neural networks with boundary conditions enforcement, Ann. Nucl. Energy 176 (2022), 109234.
[12] L. Dong, L. Qi, T. Lei, A. Ping, Y. Fan, Solving multi-dimensional neutron diffusion equation using deep machine learning technology based on pinn model, 核动力工程 43 (2) (2022) 1–8.
[13] Y. Yang, H. Gong, S. Zhang, Q. Yang, Z. Chen, Q. He, Q. Li, A Data-Enabled Physics-Informed Neural Network with Comprehensive Numerical Study on Solving Neutron Diffusion Eigenvalue Problems, 2022 arXiv preprint arXiv: 2208.13483.
[14] E. Schiassi, M. De Florio, B.D. Ganapol, P. Picca, R. Furfaro, Physics-informed neural networks for the point kinetics equations for nuclear reactor dynamics, Ann. Nucl. Energy 167 (2022), 108833.
[15] X.C. Zhang, J.G. Gong, F.Z. Xuan, A physics-informed neural network for creep-fatigue life prediction of components at elevated temperatures, Eng. Fract. Mech. 258 (2021), 108130.
[16] X. Zhao, K. Shirvan, R.K. Salko, F. Guo, On the prediction of critical heat flux using a physics-informed machine learning-aided framework, Appl. Therm. Eng. 164 (2020), 114540.
[17] J.M. Stockie, The mathematics of atmospheric dispersion modeling, SIAM Rev. 53 (2) (2011) 349–372.
[18] A.J. Cimorelli, S.G. Perry, A. Venkatram, J.C. Weil, R.J. Paine, R.B. Wilson, et al., AERMOD: a dispersion model for industrial source applications. Part I: general model formulation and boundary layer characterization, J. Appl. Meteorol. 44 (5) (2005) 682–693.
[19] CERC, ADMS 5 Atmospheric Dispersion Modelling System User Guide, 2016.
[20] U.S.NRC, Code Manual for MACCS2: Volume 1, User's Guide, 1998 (NUREG/CR-6613).
[21] U.S.NRC, RASCAL 4: Description of Models and Methods, 2012. NUREC-1940).
[22] R.H. Pletcher, J.C. Tannehill, D. Anderson, Computational Fluid Mechanics and Heat Transfer, CRC press, 2012.
[23] T.M.A.K. Azad, L.S. Andallah, Stability analysis of finite difference schemes for an advection diffusion equation, Bangladesh J. Sci. Res. 29 (2) (2016) 143–151.
[24] S.L. Brunton, J.N. Kutz, Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press, 2019.
[25] TensorFlow Core, TensorFlow core tutorials. https://www.tensorflow.org/tutorials, 2020. (Accessed 27 October 2022).
[26] G. Canbolat, H. Köse, A. Yildizeli, S. Cadirci, Analytical and numerical solutions of the 1D advection-diffusion equation, in: 5th International Conference on Advances in Mechanical Engineering Istanbul, 2019.