COMPUTATIONAL FLUID DYNAMICS ITERATION DRIVEN BY DATA

by

Zhijun ZHOU*, Qi ZHANG, Xichuan CAI, Kun LI, and Jingwei ZHAO

State Key Laboratory of Clean Energy Utilization, College of Energy Engineering, Zhejiang University, Hangzhou, Zhejiang, China

Original scientific paper
https://doi.org/10.2298/TSCI210313227Z

Data-driven approaches have achieved remarkable success in different applications, however, their use in solving PDE has only recently emerged. Herein, we present the potential fluid method, which uses existing data to nest physical meanings into mathematical iterative processes. Potential fluid method is suitable for PDE, such as CFD problems, including Burgers' equation. Potential fluid method can iteratively determine the steady-state space distribution of PDE. For mathematical reasons, we compare the potential fluid method with the finite difference method and give a detailed explanation.

Key words*: data driven, artificial neural network, PDE, CFD, iteration*

Introduction

In recent years, with the advancement of computer technology, we have witnessed the emergence of an era driven by data. In this era, *probability theory* and *mathematical statistics* have been the focus of technological developments such as machine learning [1]. At the same time, researchers face a huge challenge. They want to uncover the black box and combine these different physical phenomena with data. This will form a framework that can be applied to solve practical problems. Before the advent of this era, in order to solve these practical problems, researchers constructed corresponding PDE using some specific laws of physics. Then, they approximated PDE using finite difference methods (FDM) to solve equations iteratively. It is an innovative idea to figure out whether there is a hidden model between these data and physical phenomena

Nevertheless, this paper proposes a model that is more in line with CFD:

$$U_{i-1,j}, U_{i,j}, U_{i+1,j} \rightarrow \text{PFM} \rightarrow U_{i,j+1} \tag{1}$$

where $U_{i-1,j}, U_{i,j}, U_{i+1,j}$ represent the velocity of three adjacent points at $t = j$, $U_{i,j+1}$ – the velocity of a point at $t = j + 1$, and PFM – of the potential fluid method. The details of the calculation process will be described in detail later.

In addition, many deep learning concepts can be combined with classical methods in mathematical physics to help us to solve problems with complex data and inverse problems in complex dynamic systems. This combination of non-linear dynamics and machine learning

* Corresponding author, e-mail: zhouzj@zju.edu.cn

offers the potential to conduct predictive modeling [2, 3]. Differential equations are the cornerstone of various applied science and engineering fields. However, their application in statistics and machine learning is rarely discussed. Perhaps the most important related work in this area is the potential model. These models use differential equations to generalize potential variable models [4-6]. Different from the potential model mentioned in [5], the current work circumvents the need for analytical or numerical solutions to differential equations by placing the neural network at velocity, U, rather than on the governing equation, and the results are still consistent.

With respect to designing closed models for turbulence, Beck *et al.* [7] used a deep neural network to design a turbulence closure model for large-eddy simulations, and Ling *et al.* [8] used deep neural networks to model the Reynolds averaged turbulence. With respect to optimal designs, Viquerat *et al.* [9] used deep reinforcement learning for direct shape optimization, Yan *et al.* [10] optimized aerodynamic shapes using a new optimizer based on deep learning, and Cai *et al.* [11] estimated the motion particle velocity based on deep learning.

However, in the study of using ANN to solve PDE, there is still a lack of iterative applications. In particular, the new iterative ideas presented in this paper fill this gap.

In this paper, we develop a new neural network framework to develop the interface between these data and the physical model, which allows learning from data and equations in a specific way. In addition, we also assess the correctness of the method through an example. Therefore, our contributions to this work are:

– We design a new data-driven, iterative method of CFD.
– This method achieves some results in 1-D problems.
– This method can give a reasonable discrete distribution of steady-state velocity field with rough mesh.

Problem set-up and solution methodology

Problem set-up

In the traditional CFD calculation process, when solving a certain problem, we use the FDM to perform finite differences on the equations and conduct iterative calculations based on the initial conditions [12]. The time advancement and space advancement process of the iterative solution are shown in fig. 1 (taking the 1-D solution as an example).

Taking the 1-D as an example, the i on the horizontal axis in fig. 1 represents the node after meshing on the 1-D problem space, and the 1^{st}, 2^{nd}, 3^{rd}, ..., i^{th}, ..., and n^{th} nodes are represented from left to right. The j on the vertical axis in the figure represents the time step in which the 1-D problem advances, and from bottom to top are the 1^{st}, 2^{nd}, 3^{rd}, ..., j^{th}, ..., and t^{th} iteration time steps. Based on relationship of the four blue points in the figure, the physical information in the three points where $i = 1$, 2, and 3 at $j = 0$ is input into the difference equation obtained by the FDM method, and the physical information of this point with $j = 1$, $i = 2$ is obtained. As shown by the four orange points and the four red points in the figure, the physical information transfers in the same way. The space propels along the horizontal axis to obtain the physical information, and time advances along the vertical axis until the residual reaches a certain number, which is considered to be convergent, and the calculated physical information meets the requirements.

Now we develop a new idea. Based on the solution process of the FDM method, this paper built a neural network, and solved the PDE. The algorithm is shown in fig. 2.

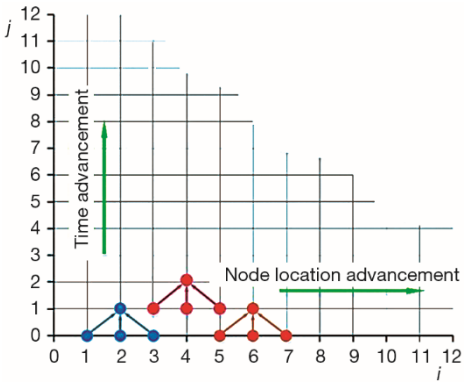

Figure 1. Schematic diagram of a CFD iterative process

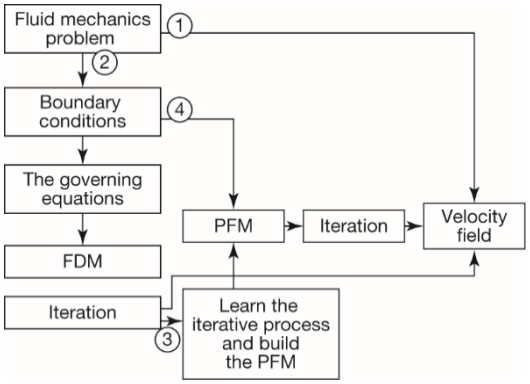

Figure 2. Diagram of the differences in the PFM and FDM when solving fluid mechanics problems

As shown in fig. 2, for CFD, Process 1 is our solution route, which means that the solution route starts from the problem and reaches the result (velocity field). Process 2 is a traditional CFD method that utilizes boundary conditions, solves equations based on governing equations, and uses the FDM to select equations suitable for CFD and an iterative solution. In this paper, a new algorithm is developed. The mentioned iterative process is used to conduct Process 3. According to a large amount of data generated by the iterative process, the hidden fluid model existing in the data is mined, and the interface framework between the data and the physical model is constructed. This paper builds this framework based on a neural network and constructs the PFM. When encountering fluid mechanics problems, according to the boundary conditions, we perform the work of Process 4 rather than Process 2. We input the data into the PFM, and then we get the corresponding output. After the iterations, the velocity field will be obtained. This paper will focus on the establishment and operation of the PFM and will compare the results between the FDM and PFM.

Solution methodology

In the iterative process of the FDM, the velocity value of the corresponding point of the input is changed. As shown in fig. 1, when the velocity value of each point at $j = 1$ is required, the velocity values of each point at $j = 0$ are used to solve the difference equations. As shown by the blue dots in fig. 1, $U_{2.1}$ is obtained by inputting $U_{1,0}, U_{2,0}, U_{3,0}$ into the equation, namely:

$$U_{2,1} = f(U_{1,0}, U_{2,0}, U_{3,0}) \tag{2}$$

In the conventional method, $U_{i,j+1} = f(U_{i-1,j}, U_{i,j}, U_{i+1,j})$ is obtained by $y = f(x_i)$. The orange and red dots in the figure also follow the same iterative logic. Then, in the calculation process, there will be many such correspondences. We will use this method to build a physical framework using neural network methods so that we can form a data-to-physical model interface, which is called the PFM. Then, in the iterative process, we can use the PFM instead of $y = f(x_i)$ to iteratively solve it. The neural network is shown in fig. 3.

We must normalize the input data before performing the operation. For a single neuron, we assume that the input is $x_1, x_2, \ldots, x_i, \ldots, x_n$; the corresponding neuron connection weights are $\omega_1, \omega_2, \ldots, \omega_i, \ldots, \omega_n$; and the threshold of the neuron is θ. The output value is:

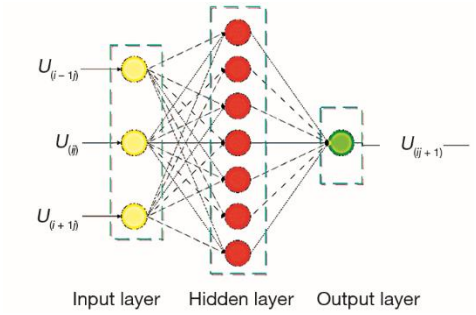

Figure 3. Physical information delivery framework

$$y = f\left(\sum_{i=1}^{n}\omega_i x_i - \theta\right) \tag{3}$$

In the process of setting up the PFM interface, the input data we use is the data of each time step obtained by the FDM, which is relatively easy to obtain. The output data are the solution of the corresponding difference equation, which is the specific point of the next time step's data.

Test cases of numerical results.

This section takes two 1-D problems as an example to introduce the practical application of this method.

Burgers' equation

Burgers' equation appears in various fields of engineering, including fluid mechanics, non-linear acoustics, gas dynamics, and traffic flow. It is a fundamental PDE and can be derived from the Navier-Stokes equations for the velocity field by dropping the pressure gradient term. Burgers' equation leads to the formation of impacts, which makes it difficult to solve using classical numerical methods. This paper will start with a simple Burgers' equation:

$$\frac{\partial U}{\partial t} + u\frac{\partial U}{\partial x} = \mu\frac{\partial^2 U}{\partial x^2} \tag{4}$$

The initial conditions are: $U_{x,0} = -0.5x, \quad x \in [-1,1]$.

The boundary conditions are:

$$\begin{cases} U_{-1,t} = 0.5 \\ U_{1,t} = -0.5 \end{cases}$$

One of the traditional basic CFD method is used to solve this equation, and the 1-D Burgers' equation is solved by using the Lax-Wendroff method:

$$\frac{\partial U}{\partial t} + \frac{\partial f}{\partial x} = \mu\frac{\partial^2 U}{\partial x^2} \tag{5}$$

where $f = U^2/2$. The solution is written as a conservation form. The difference equation obtained according to the space propulsion and time advancement method of fig. 1 is:

$$U_{i,j+1} = U_{i,j} + \left[\frac{\mu(U_{i+1,j} - 2U_{i,j} + U_{i-1,j})}{(\Delta x)^2} - \frac{1}{2} \frac{F_{i+1,j} - F_{i-1,j}}{\Delta x} \right] \mathrm{d}t \tag{6}$$

In the formula:

$$F_{i+1,j} = U^2{}_{i+1,j} \tag{7}$$

$$F_{i-1,j} = U^2{}_{i-1,j} \tag{8}$$

For readability, we swap the horizontal and vertical co-ordinates. Figure 4 is an FDM iterative process cloud diagram where the ordinate represents the position, the abscissa represents the advancement of time, and the depth of the color represents the velocity.

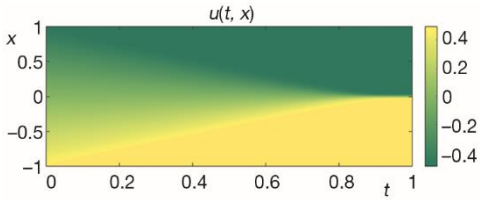

Figure 4. Process cloud diagram of the iterative difference equation
(*for color image see journal web site*)

Subsequently, this paper finds an iterative solution based on the FDM values, records the three-point input and one-point output values of each space advancement and time advancement, and establishes a training database of the neural network. We randomly selected the corresponding value (3 to 1) to form the training data set:

$$U_{i-1,j},\ U_{i,j},\ U_{i+1,j} \rightarrow \mathrm{PFM} \rightarrow U_{i,j+1} \tag{9}$$

As described in fig. 3, we take the adjacent three-point velocity at time t as an input and the velocity at the same position $t + \Delta t$ as the output for neural network training. The amount entered is normalized and the sigmoid is used as the activation function.

The neural network consists of l input layer neurons, m hidden layer neurons, and n output layer neurons, which is written as ANN (l, m, n), and the weights between neurons are W_{lm} and W_{mn}. During the training process in which the predicted output value is compared to the known output value, the initial estimated weight value is stepwise corrected and any errors are propagated back to determine the appropriate weight adjustments that are needed to minimize the error. Throughout the ANN simulation process, adaptive learning rates are used to increase the training speed and solve local minima problems.

For each iteration, if the performance approaches the goal, the learning speed increases as the learning increment increases. If performance is improved, the learning rate is adjusted using a factor learning reduction. In practical applications, the simple trial and error method is used to find the number of neurons in the hidden layer, and the Levenberg-Marquardt (L-M) technique [13], which is more powerful than the traditional gradient descent technique, is used to train the ANN. In this paper, the L-M training algorithm is used to adjust the weights. The logarithmic sigmoid transfer function is used as the activation function of the hidden layer and the output layer. The number of hidden layer neurons is found by simple trial and error. The deviation criterion during training is usually expressed using the mean square error (MSE) between the actual output and the predicted output:

$$MSE = \frac{1}{2q} \sum_{i=1}^{q} (y_i - y_{i,t})^2 \tag{10}$$

where q is the number of samples used to train the ANN, and y_i and $y_{i,t}$ – the i^{th} neural network predictor and actual value, respectively.

Second, the decision parameter, R^2, is also applied to the statistical data, and its formula is:

$$R^2 = \frac{\sum_{i=1}^{n}[U(i,j+1)_{\text{observed}} - \overline{U(i,j+1)}_{\text{observed}}]^2 - \sum_{i=1}^{n}[U(i,j+1)_{\text{observed}} - U(i,j+1)_{\text{predicted}}]^2}{\sum_{i=1}^{n}[U(i,j+1)_{\text{observed}} - \overline{U(i,j+1)}_{\text{observed}}]^2} \quad (11)$$

where the overbar means average.

We use the data generated by the 100 iterations of the FDM as the training dataset. In the training dataset containing 9900 sets of data, 10%, 30%, 50%, and other different proportions of data are selected for training. For each training group, 70% of the data were used for training, 15% of the data were used for validation and 15% of the data were used for testing. The calculated MSE is the error during the test. Furthermore, we change the number of hidden layer neurons. Here, p is the database containing the data ratio, and m is the number of hidden layer neurons, tab. 1.

Table 1. The square root of MSE of different ANN

p \ m	5	10	15	20
10%	$1.98\cdot10^{-3}$	$9.24\cdot10^{-4}$	$2.35\cdot10^{-4}$	$4.79\cdot10^{-4}$
30%	$7.69\cdot10^{-5}$	$5.48\cdot10^{-5}$	$1.43\cdot10^{-5}$	$3.24\cdot10^{-4}$
50%	$4.43\cdot10^{-5}$	$3.50\cdot10^{-5}$	$1.18\cdot10^{-5}$	$7.50\cdot10^{-5}$
70%	$3.40\cdot10^{-5}$	$3.05\cdot10^{-5}$	$3.66\cdot10^{-5}$	$1.30\cdot10^{-5}$
90%	$4.31\cdot10^{-5}$	$2.68\cdot10^{-5}$	$1.50\cdot10^{-5}$	$8.62\cdot10^{-6}$

We train at p = 90% and m = 20. The training results are shown in fig. 5.

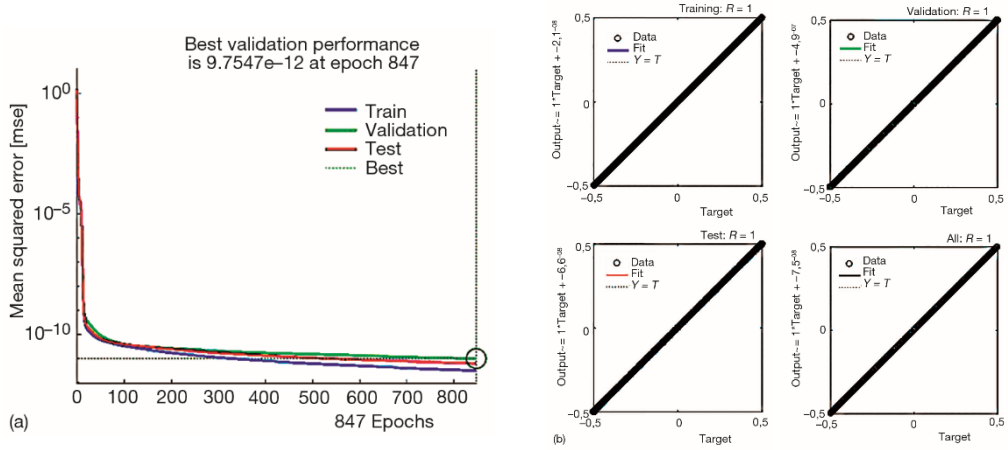

Figure 5. Training results; (a) iterative process diagram of MSE and (b) iterative process diagram of *R* (*for color image see journal web site*)

Rebuild

Subsequently, we replaced the governing equations in the PFM with the ANN for the iterative calculations. Figure 6 is the PFM iterative process cloud diagram, where the ordinate represents the position, the abscissa represents the propulsion time, and the depth of the color represents the velocity.

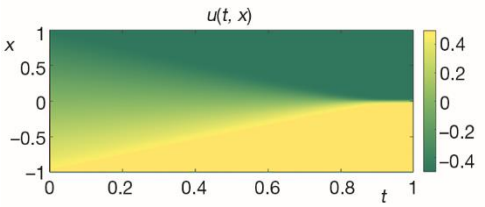

Figure 6. Process cloud diagram of the iterative PFM

As shown in the cloud diagram, the iterative process cloud diagram of the FDM is basically consistent with the iterative process cloud diagram of the PFM. We extracted four moments from it. The four-line diagrams after extraction are shown in fig. 7. At the times of $t = 0.25$, 0.5, 0.75, and 1 seconds, the velocity field distributions of Burgers' equation calculated by the two methods are consistent. In addition, the calculated values of the two methods are consistent, indicating that the PFM can achieve the same computational efficiency as the FDM in 1-D problems. The steady-state calculated results are unified and the aforementioned data-driven conjectures are verified, which achieves the computational purpose of this paper.

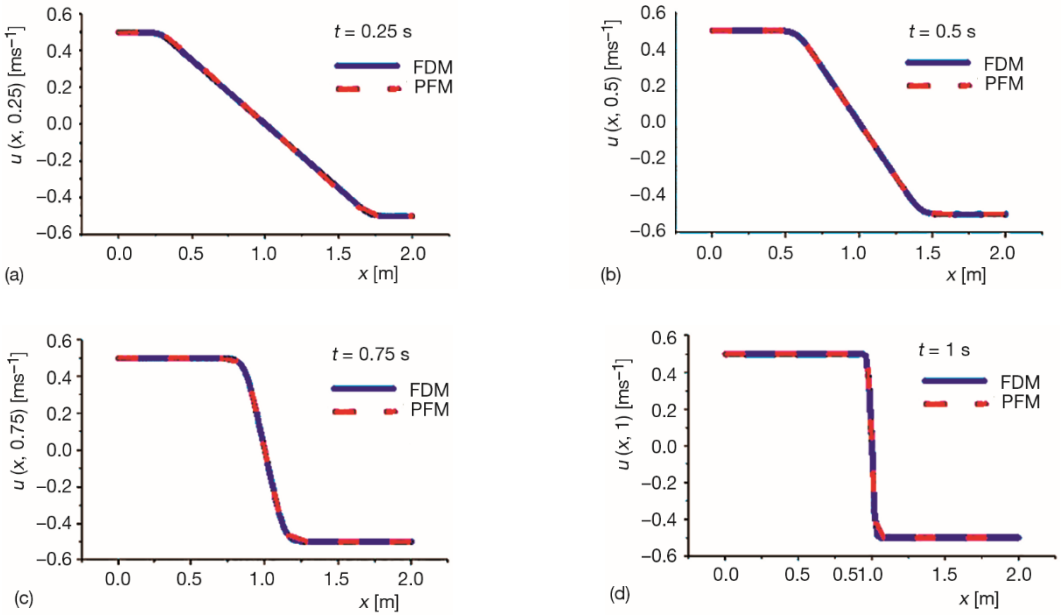

Figure 7. Speed distribution line graph at different time steps

Subsequently, we performed the calculation of the relative $\mathcal{L}_1$ error. The N_f is defined as the number of collected points. At the same position, we define the U calculated by the FDM as U_{lax} and define the U calculated by the PFM as U_{ANN}:

$$\mathcal{L}_1 = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|U_i^{lax} - U_i^{\mathrm{ANN}}\right| \tag{12}$$

In this way, we can compare the velocity values of the two clouds at the same location and make the tab. 2. We use the following formula to make the error have no specific physical meaning. Here, $\bar{U}_t$ is the average velocity at the time of t.

$$\mathcal{L}_1' = \frac{\mathcal{L}_1}{U_t^{lax}} \tag{13}$$

Table 2. Relative $\mathcal{L}_1$ error between U_{lax} and U_{ANN}

t	$\mathcal{L}_1'$
0.25	$8.91\cdot10^{-3}$
0.5	$8.68\cdot10^{-5}$
0.75	$5.89\cdot10^{-5}$
1	$5.39\cdot10^{-5}$

Different condition

Only the boundary conditions are changed and the other conditions remain unchanged.

The boundary conditions are:

$$\begin{cases} U_{-1,t} = 0.4 \\ U_{1,t} = -0.4 \end{cases}$$

$$\text{Originally:} \begin{cases} U_{-1,t} = 0.5 \\ U_{1,t} = -0.5 \end{cases}$$

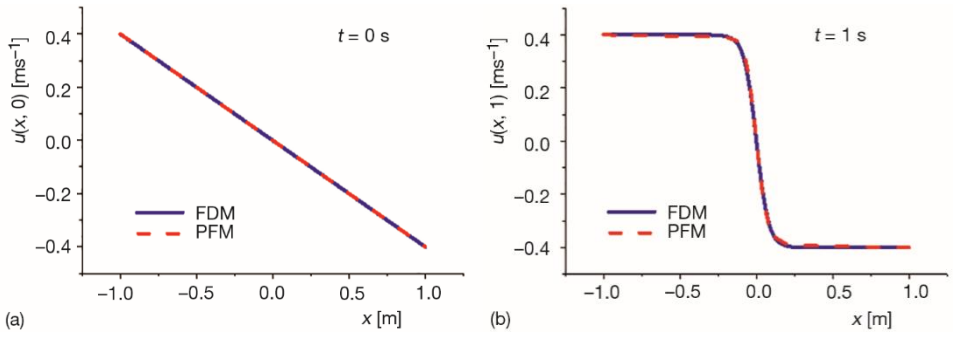

Figure 8. Speed distribution line graph under different condition

The PFM still provides the reliable steady-state distribution shown in fig. 8. This distribution is consistent with the steady-state calculation results of FDM under the same scale grid. Relative $\mathcal{L}_1^{'}$ error, tab. 3, between U_{lax} and U_{ANN}:

$$\mathcal{L}_1^{'} = 0 \quad (t = 0)$$

$$\mathcal{L}_1^{'} = 1.22E \cdot 10^{-2} \quad (t = 1)$$

Table 3. Relative $\mathcal{L}_1^{'}$ error

t	PFM	FDM	PFM/FDM
1	$1.04 \cdot 10^{-2}$	$3.56 \cdot 10^{-1}$	$2.9 \cdot 10^{-1}$

Different Δx

All the other conditions are the same, just increase Δx by a factor. The red solid line in fig. 9 is the steady-state numerical solution under the fine grid, we use this as a reference. We find that the PFM can still give a reliable steady-state distribution under rough grids, as shown by the blue line. But under rough mesh, FDM cannot give a reliable steady-state distribution.

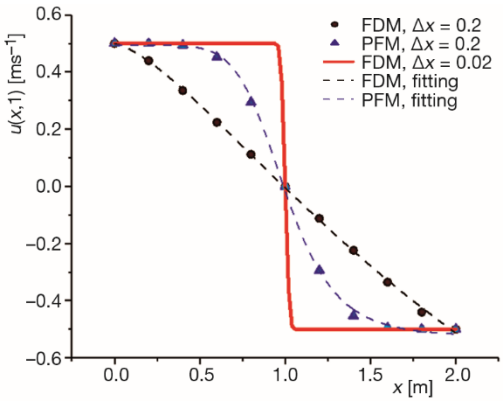

Figure 9. Speed distribution line graph with different Δx

Conclusions

In summary, this paper introduces a new method for solving the velocity distribution. In 1-D, it uses the FDM to construct a data-driven PFM framework to solve the velocity distribution iteratively.

In Burgers' equation, the performance of the two models on the boundary conditions and different Δx is compared. The performance of PFM is reasonable, especially under larger Δx. The solution process and the solution result are in line with the actual situation. The re-established velocity distribution conforms to the laws of physics and successfully realizes the expectations of this paper.

The expected results show that the PFM can be used to calculate the velocity distribution under different boundary conditions or Δx. The PFM can well uncover the black box process in CFD and complete the core processing of the new iterative calculation method proposed in this paper. (After publication, the relevant code will be uploaded to https://github.com/ zhangqi1996456).

Acknowledgment

This work was supported by the Ningxia Science and Technology Department (No. 2018BCE01004) and the National Nature Science Foundation of China (No. 51976186).

References

[1] Krizhevsky, A., *et al.,* ImageNet Classification with Deep Convolutional Neural Networks, *Proceedings,* Advances in Neural Information Processing Systems 25 (NIPS 2012), pp. 1097-1105, http://papers.nips.cc/paper/ 4824-imagenet-classification-w%5Cnpapers3://publication/uuid/1ECF396A-CEDA-45CD-9A9F-03344449DA2A
[2] Raissi, M., *et al.,*Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, *Machine Learning,* (2017), On-line first, arXiv:1711.105611v1
[3] Raissi, M., *et al.,* Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations, *Machine Learning* (2017), On-line first, arXiv:1711.10566v1
[4] Lawrence, N., Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models, *Journal of Machine Learning Research, 6* (2005), 11, pp. 1783-1816
[5] Bishop, C. M., Tipping, M. E., Latent Variable Models and Data Visualisation, in: *Stat. Neural Networks*: Advance at the Interfoce, (Bishop, M., Michael, E.), 2000, pp. 147-164
[6] Depeweg, S., *et al.,* Uncertainty Decomposition in Bayesian Neural Networks with Latent Variables, *Machine Learning,* On-line first, arXiv:1706.08495v2, 2017
[7] Beck, A. D., *et al.,* Deep Neural Networks for Data-Driven LES Closure Models, *Journal of Computational Physics, 398* (2019), Dec., 108910
[8] Ling, J., *et al.,* Reynolds Averaged Turbulence Modelling Using Deep Neural Networks with Embedded Invariance, J. *Fluid Mech., 807* (2016), Nov., pp. 155-166
[9] Viquerat, J., *et al.,* Direct Shape Optimization Through Deep Reinforcement Learning, *Journal of Computational Physics, 428* (2021), Mar., 110080
[10] Yan, X., *et al.,* Aerodynamic Shape Optimization Using a Novel Optimizer Based on Machine Learning Techniques, *Aerosp. Sci. Technol.*, *86* (2019), Mar., pp. 826-835
[11] Cai, S., *et al.,* Dense Motion Estimation of Particle Images *Via* a Convolutional Neural Network, *Exp. Fluids.*, *60* (2019), 4, pp. 1-16
[12] Abadie, L. M., Chamorro, J. M., Finite Difference Methods, in: *Lect. Notes Energy,* Springer, London, UK, 2013
[13] Watson, G. A., *Numerical Analytics, Proceedings,* Biennial Conference, Dundee, Scotland, 1977, Springer, New York, USA, 1978

Paper submitted: March 13, 2021
Paper revised: May 23, 2021
Paper accepted: May 24, 2021