The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

PREDICTION OF THE BIOMASS PARTICLES THROUGH THE PHYSICS-INFORMED NEURAL NETWORK

Fateme Darlik[1], Prasad Adhav[2], and Bernhard Peters[3]

email:[1]fateme.darlik@uni.lu, [2]prasad.adhav@uni.lu, [3]bernhard.peters@uni.lu

Key words: Physics informed neural network, Discrete element method, Biomass furnace, Surrogate model

Abstract. Woody biomass energy is a kind of renewable energy that contributes to the reduction of greenhouse gas emissions, the creation of healthier forests, and the reduction of wildfire danger. Generally speaking, simulations of the motion of biomass particles are a time-consuming process due to a large number of particles and required simulation time. We used a physics-informed neural network (PINN) model to predict the motion of particles by including their equations of motion to reconstruct the velocity fields and reduce the processing effort. compare to the discrete element methods, the PINNs methods have the advantage of predicting the velocity fields without the knowledge of the simulation's boundary and initial conditions as well as geometry. It has shown that the proposed model has reliable prediction results with a mean percentage error in time less than 1 percent.

1 INTRODUCTION

A biomass furnace uses a well-known process of gasification to turn biomass into flue gases. These flue gases are then in turn used to generate heat, which is used for different purposes, usually to generate steam and run steam turbines to generate electricity. In the day and age of climate change, biomass furnace is catching a lot of attention as an alternative renewable energy source [5]. The fuels often used are high in moisture content, a variety of minor constituents such as chlorine, sulfur, phosphorous, and a variety of ash-forming metals [14]. Hence, it is essential to study and predict these operating parameters for different fuels, especially new fuels such as domestic waste. These parameters are heavily dependent on the conversion of the fuel into flue gases and the burning of the flue gases. As performing experiments for biomass simulation is prohibitively expensive and time-consuming, numerical simulations are used to solve the problem.

The Discrete Element Method (DEM) is used to simulate the biomass moving bed. The DEM treats biomass fuels as particles. The DEM handles particle motion, interparticle collisions, and solid-state reactions. Such a numerical model of the biomass furnace contains thousands of particles. These particles then travel across the biomass furnace using fixed and moving grates. The numerical simulation for such particle movement is computationally costly, especially when a large number of particles for a long simulation time is used. Hence this becomes a major hurdle when modeling biomass furnaces with long grates and small size fuel "particles". For a given biomass furnace design, the particle bed movement does not change much for a given size and shape. The composition of the particle bed can surely change, which is the interesting

part when considering the biomass furnace operating parameters (apart from the particle bed movement). Thus it could be very beneficial if this particle bed movement could be predicted in a computationally cheaper way. In the present study, Physics-Informed Neural Networks (PINNs) are employed to predict the moving particle bed motion and are intended to be used as a surrogate model for the computationally costly DEM model. Neural network (NN) models are increasingly being used in a wide range of areas since they are effective tools for resolving, improving, and optimizing scientific problems. Neural networks can understand the implicit, complicated and vague information and based on that, are able to recognize the pattern behind it and predict the assigned outputs. Altogether, the neural network is proven to be a useful tool in solving engineering problems [4], especially in scenarios where there is not sufficient data to use for doing simulation [16], the simulation is time-consuming [3], and where optimization, decision making, and data analysis are the concerns.

Generally speaking, there are different types of neural networks such as fully connected neural networks [1], feed-forward artificial neural networks [2], convolution neural networks [10], recurrent neural networks [8], and physics-informed neural networks (PINNs) [17] which are based on the need of the problem. Among all of these, in this study, the PINNs are used as neural network tools. The main advantage of the PINNs [21], is that they are more suitable for the physical concerns of the engineering problems and their governing equation [20]. In contrast, when using a non-physics-informed neural network, the existence of a sufficient number of data is mandatory to have a well-trained neural network, however, in PINNs, the govern equations of the problem will give the neural network a better understanding of the problem itself [15]. Following that, PINNs are practical in problems where data generation via simulation is computationally expensive, or the only data available are the sparse data from experimental tests [18, 26]. PINNs are aware of data limitations and will predict the outputs by using extensive physical knowledge of the problem. Additionally, PINNs can be used with or without adding the information regarding boundary conditions, initial conditions [24, 25], and the geometry [23, 27], which should be decided based on the user concerns.

The motion of the particles in the moving bed is extracted from simulation data in this study. The PINN is constructed by adding the motion equations as the governing equation to the neural network. The PINN predicts and reconstructs the velocity fields of particles moving on moving grate beds while taking into account unexpected changes in the particles' downward motion caused by gravity. It is demonstrated here that PINNs are effective instruments capable of predicting the motion fields. Additionally, the time-average percent error of the velocity does not exceed one percent error, which explains the capability of our neural network. The paper is organized as follows: In section 2 we present the mathematical modeling of DEM and PINN training data generation. In section 3, the methodology for developing the PINN is described. In section 4 we present the results from the PINN. Finally, in section 5 we conclude the findings of the paper.

2 DEM SIMULATION FOR DATA GENERATION

Generally, real-world data or data based on the real world is used to train a NN. In the presented work, we use the data obtained from numerical simulations of the Biomass furnaces using the eXtended Discrete Element Method (XDEM) [19] software. XDEM software is based

on the Discrete Element Method (DEM), which can simulate particle motion, interparticle collisions, etc. This DEM software is then extended to account for inter-particle and intra-particle thermodynamics as well as the influence of fluid flow. XDEM is thoroughly verified and validated against experimental studies [7, 9, 11–13, 19], including validation for biomass furnace and processes involved in it. The mathematical modeling of XDEM is presented in the following sections.

The dynamics module of XDEM software predictions particle position, velocity, and acceleration. It uses a soft sphere model, where particles are assumed to be deformable and can overlap each other. The inter-particle contact forces are computed through the magnitude of overlap between the particles, based on the force-displacement law chosen. The particle hardness is described with Young's modulus, whereas the particle energy dissipation is modeled with a dampener and/or dash-pot. The transnational and rotational movements of individual particles are tracked using the classical mechanics equations as described in Eq 1 and Eq 2 respectively.

$$m_i \frac{d\vec{v_i}}{dt} = m_i \frac{d^2\vec{X}_i}{dt^2} = \vec{F_i^c} + \vec{F_i^g} + \vec{F_i^{ext}} \tag{1}$$

$$I_i \frac{d\vec{\omega_i}}{dt} = \sum_{j=1}^{n} \vec{M}_{i,j} \tag{2}$$

where $\vec{F_i^{ext}}$ is the sum of all the external forces acting on the particle, such as particle impact forces, buoyancy forces $\vec{F_B}$ and drag forces $\vec{F_D}$. The inter-particle impact energy dissipation is modeled as a spring and dashpot.

2.1 Biomass furnace simulation set up

In the present study, the dynamics module of the XDEM software is used to simulate a 2D moving bed in the combustion chamber of a biomass furnace. The biomass fuel is modeled as spherical particles. The biomass furnace case is set up as shown in figure 1. The moving bed composition consists of Beechwood and Stringy wood (agricultural waste) with properties described in table 1. The biomass fuel is fed into the furnaces through the 2 particle sources (for each type of fuel) from the left as shown in the figure 1.

Table 1: Example of the construction of one table

Properties	Beechwood	Stringy-wood
Particle Radius $[m]$	0.015	0.015
Particle bed mix fraction [%]	79.5	20.5
Mass Flow rate $[kg/s]$	0.0155	0.0038
Number of particles	1762	462
Apparent Density $[kg/m^3]$	648.52	610.47
Mean porosity [-]	0.685196	0.6375

The biomass furnace consists of grates, fixed (black) and moving (red). These moving grates are used to move the particles along the length of the biomass furnace. As time progresses the

particles will eventually fall from the outlet opening. Partial walls of the biomass furnace are added to show a general outline of the furnace.

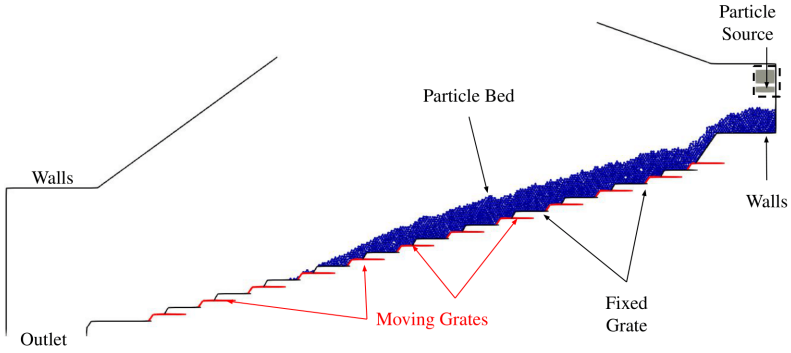

Figure 1: XDEM Biomass furnace Case setup

2.2 XDEM Results

The XDEM simulation is run from 0 s to 100 s. As seen in figure 2, we can see the moving (red) grates pushing particles along. This grate movement dictates the traverse speed of particles along the length. This speed in turn then dictates the residence time of particles in particular zones of the furnace (such as drying, pyrolysis, char burning, etc.). The moving bed comes into regular motion after around 40 s, after which the moving bed motion does not vary much. Considering this the particle position and velocities between 40 s and 100 s are used as training data for the PINN.

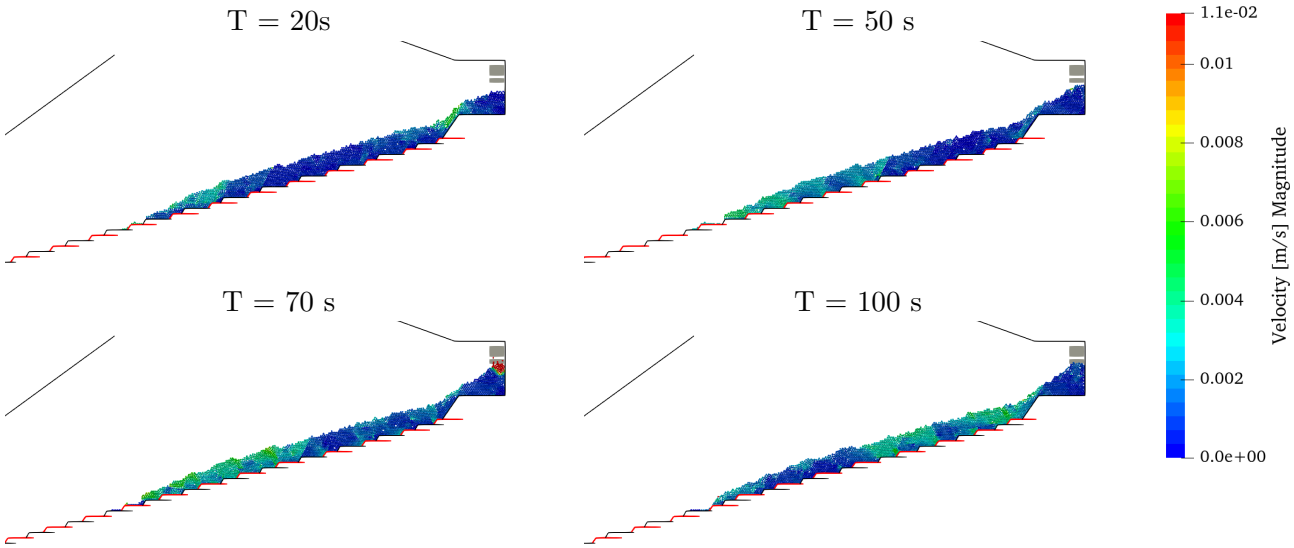

Figure 2: XDEM simulation results showing the progression of moving bed in Biomass furnace

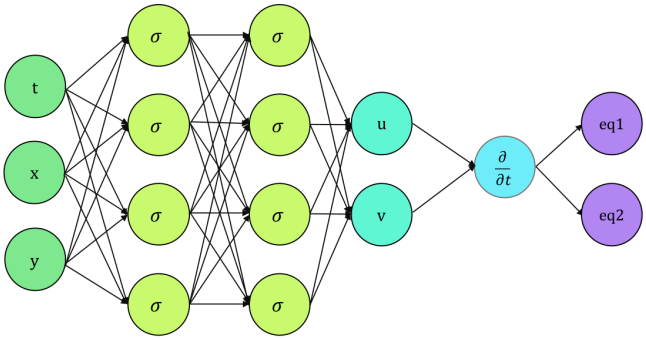

Figure 3: Physics-Informed neural network architecture. The inputs are the coordinates (x and y) and the simulation time. The output is the velocity in the x and y direction. The governing equations are the equation of the motion.

3 PHYSICS INFORMED NEURAL NETWORK

PINNs [17] is an extended type of scientific neural network that takes model equations such as partial differential equations (PDEs) into account, as part of its neural network. Nowadays, the application of PINNs [20]is expanding due to their efficiency in satisfying the equations in both supervised and unsupervised learning. It can approximate the solution of PDEs equations while training the neural network. Additionally, by using this methodology one can evaluate the boundary conditions and initial conditions needed for solving PDEs [26]. In other words, PINNs are categorized in deep learning networks that can produce an approximate solution in the integration domain for each point included in the PDEs equations. The residuals of PDEs are included in the loss function of its neural network which will directly affect the efficiency of the process. Another main advantage of the PINNs is that they can be a mesh-free methodology that can solve the PDEs by using a direct problem solution and changing it to the optimization problem. It reinforces the loss function by residual terms of the governing equations which can be interpreted as a penalizing term that forces the output of the neural network to be in an acceptable solution. Generally speaking by using this framework, the neural network not only predicts the outputs but also checks if the predicted variables can satisfy the governing equations [20]. It has proven to be practical, especially in mechanical engineering data-driven studies. PINNs consider underlying PDE which introduces the physics of the problem rather than just predicting the solution only based on the data, by reducing the error between the predicted data and actual data. One of the areas in that PINNs has been used widely is in the inverse problem to reconstruct the fluid field by using the data from the sensors [20]. By feeding the neural network the sparse data that is available from one domain, and adding the governing equations to the neural network, the PINNs can reconstruct the domain fields. This approach has been implemented in different fluid dynamics problems [22]. In this study, we aim to use the PINN to reconstruct the velocity fields for particles moving in the moving beds of fluids in the two-dimensional domain. In the figure 3 the architecture of the physics-informed neural network is depicted. The equations of the motion are added to the loss function to assure the neural not only predicts the output data but takes care of the fact that the output data should satisfy the equations.

4 RESULTS AND DISCUSSIONS

The PINN methodology is used to predict the velocity of the particles in a moving bed. The combustion chamber case study that is used in this article is thoroughly detailed in the section 2 above. Utilizing XDEM, the simulation is done, and the required data is then collected for usage as train and test data to feed the neural network. After fine-tuning the neural network's hyperparameters, a network with 10 layers and 200 neurons per layer is employed, yielding 323203 parameters. The activation function rectified linear unit (ReLU) 3 is applied for the hidden layer and the activation function sigmoid 4 is used for the output layer.

$$ReLU(x) := max(0, x) \tag{3}$$

$$\sigma(x) := \frac{1}{1 + e^{-x}} \tag{4}$$

The simulation time steps and the coordinates are used as the input of the neural network, and the velocities are the output of the neural network. As the order of the magnitude of the data is different from each other, a normalization between 0 to 1 is applied to all data including inputs and outputs. The mean squared error (MSE) is used for the loss function. The Adam optimization method is chosen as the optimization method as it is a stochastic gradient descent approach that is highly efficient, requires low memory, and is appropriate for large data problems [6]. An adaptive learning rate that started from $1e-3$ and decreases to $1e-5$ is used to control how quickly the model is adapted to the problem. The batch size is chosen to be equal to the total number of particles in each time step. To avoid overfitting, the total number of epochs is not defined from the start; training of the neural network continues until we get a minimum for the validation dataset. Of the whole number of data, 70 percent of it randomly is chosen as the training dataset, and the other 30 percent is considered as the test data set. And 10 percent of the training dataset is used as the validation dataset. After training for about 20,000 epochs, and monitoring the changes in the loss function, we stopped the training.

The equations that are added to neural networks are the equations of the motion (equations 5 and 6) in the both x and y direction.

$$e_1 := f_x(t, x, y) - m\frac{du(t, x, y)}{dt} \tag{5}$$

$$e_2 := f_y(t, x, y) - m\frac{dv(t, x, y)}{dt} \tag{6}$$

Where $u(t, x, y)$ is the particle velocity in x direction, $v(t, x, y)$ is the particle velocity in y direction, m is particles mass, and $f_x(t, x, y)$ and $f_y(t, x, y)$ are the forces applied on the particles in x and y direction respectively. The particles' applied forces are consists of interaction force, and the weights of the particles are all calculated by using the XDEM simulation. The mentioned equations are added to the PINN's loss function.

$$MSE := \sum_{n=1}^{N} \Big(|u(t^n, x^n, y^n) - \hat{u}^n|^2 + |v(t^v, x^n, y^n) - \hat{v}^n|^2 + \sum_{i=1}^{2} \sum_{n=1}^{N} (|e_i(t^n, x^n, y^n)|^2) \tag{7}$$

Where N is the total number of the particles, $\hat{u}$ and $\hat{v}$ are the learned velocity. In order to illustrate the effectiveness of our neural network, the time mean squared error (equation 8), and time mean squared error (equation 9) are reported for each output field.

$$MSE^t(u_i^t, \hat{u}_i^t) := \frac{1}{N}\sum_{i=1}^{N}(u_i^t - \hat{u}_i^t)^2 \tag{8}$$

$$MRPE^t(u_i^t, \hat{u}_i^t) := \frac{1}{N}\sum_{i=1}^{N}\frac{u_i^t - \hat{u}_i^t}{u_i^t} \times 100 \tag{9}$$

MSE^t calculates the average squared error between the exact and predicted data in each time step t, and the $MRPE^t$ calculates the relative percentage error between the exact and he predicted data for all the available particles in each time step t. Figure 4 shows a diagram of the MSE values over time. With the results obtained within the plot, the maximum value is less than $10e-3$, the average value for the velocity in both directions is $10e-6$, and the majority of the errors are some number between $10e-8$ and $10e-$, which altogether represents a good agreement between the predicted data and the exact data. Another intriguing result is depicted in figure 5. It can be seen from the chart that the proposed framework is capable of reconstructing the velocity fields with a percent error of less than 1 present. By having a closer look at the figures 4 and 5, one anticipated finding will be the fact that while the error for the velocity in the y direction has a lower value compared to the error for the velocity in the y direction, however, the maximum error comes from the velocity in u direction. We can attribute this to the fact that the existence of gravity in the y direction will lead to sudden changes in the value f the velocity which the neural networks would not be able to capture. On the other hand, if the particle is on the grates the velocity in the y direction are mainly the same as each other which will make it easier for the neural network to anticipate the behavior of the particles in these directions.

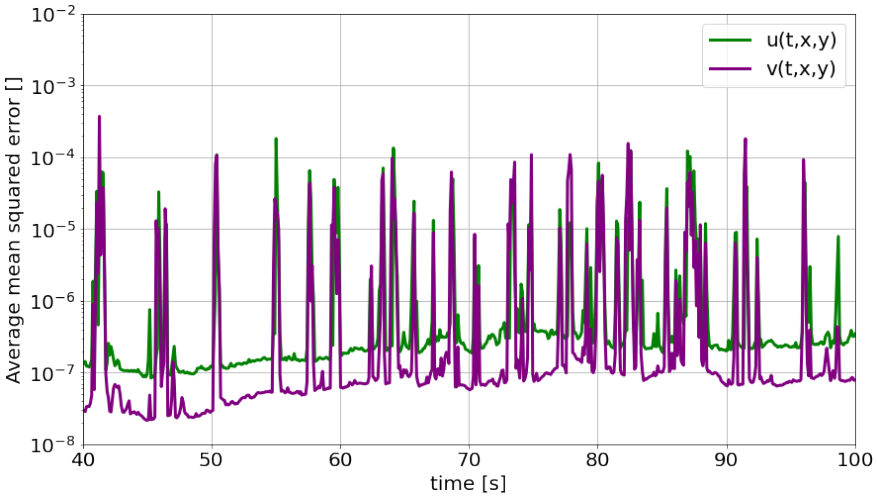

Figure 4: The average of the mean squared error of all the particles available in each time step.

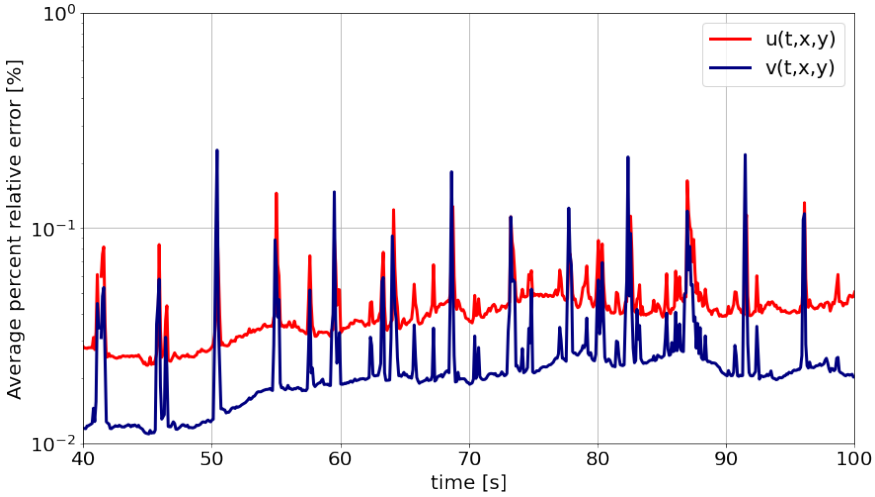

Figure 5: The average of the percentage error of all the particles available in each time step.

After a detailed study of the figures 4 and 5, we find out that at time 50 seconds, the maximum error value for both plots happened at time 50 seconds. Therefore, to validate our neural network, the representative snapshots of the exact data, learned data, and the relative percent error at a time of 50 seconds are depicted in figure 6. The first row of the figure 6 presents the contour of the exact value of velocity in both x and y directions that are derived from the DEM simulation. In the next row, the velocity predicted by PINNs is shown. Additionally, the last row shows the percent relative error between exact and predicted data which is calculated as equation 10:

$$RPE(u_i, \hat{u}_i) := \frac{u_i - \hat{u}_i}{u_i} \times 100 \tag{10}$$

Looking at figure 6 demonstrates that when comparing the predicted results to the exact results, the neural network will give us a smother change in the velocity. The present frameworks are capable of predicting the velocity with a maximum percent error of 0.35 percent in the x direction and 0.16 in the y direction. As these chosen time steps have maximum MRPE compared to other time steps, it is expected that the neural network in the other time steps will predict with higher accuracy.

To get more detail about the performance of the used methodology the percentage error between the exact learned velocity for three different time steps is represented in figure 7. The time 50, 70, and 100 seconds is chosen to be used in this section. One anticipated finding is that the value of the error in the edge of grates is generally larger than in other parts of the simulation. Additionally, in the inlet of the domain, where we are having mass flows, due to lack of enough information, it consists of a larger value of error when compares to the middle part of the simulation. The same behavior can be seen in the outlet of the domain, And the explanation is that in these sections as the particles are moving forward, it contains a fewer amount of particles, and in some locations, it might not have any particles at all. That causes the results, where when we move further in time, the maximum error value increase. Taken together, these results of predicting the velocity of the moving particles in the combustion

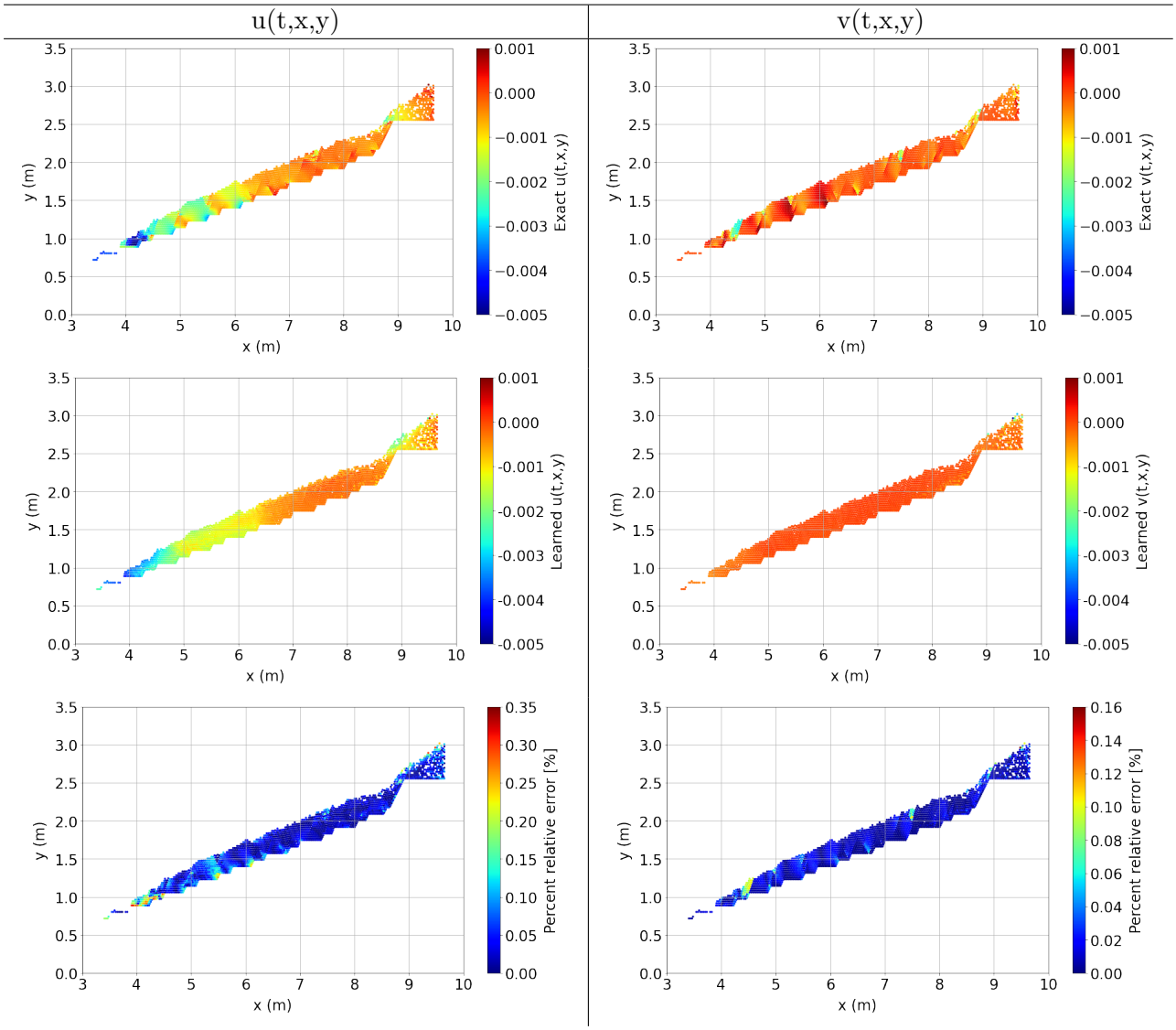

Figure 6: A snapshot of the velocity in the x and y directions at simulation time equal to 50 seconds. The first row displays the exact velocity value. The second row shows the predicted value for the velocity obtained from PINNs, and the last row shows the error between the exact value and the learned value, which will be used to demonstrate the framework's effectiveness.

chamber are remarkable illustrations of PINNS's advanced qualities and highlight its potential in resolving and reconstructing the physical fields.

5 CONCLUSIONS

In this study, the velocity of the particles moving in the bed of the biomass combustion chamber is studied by training PINNs. Traditionally, similar problems are solved by employing the DEM approach. However, in this study, we used a neural network to reconstruct the velocity

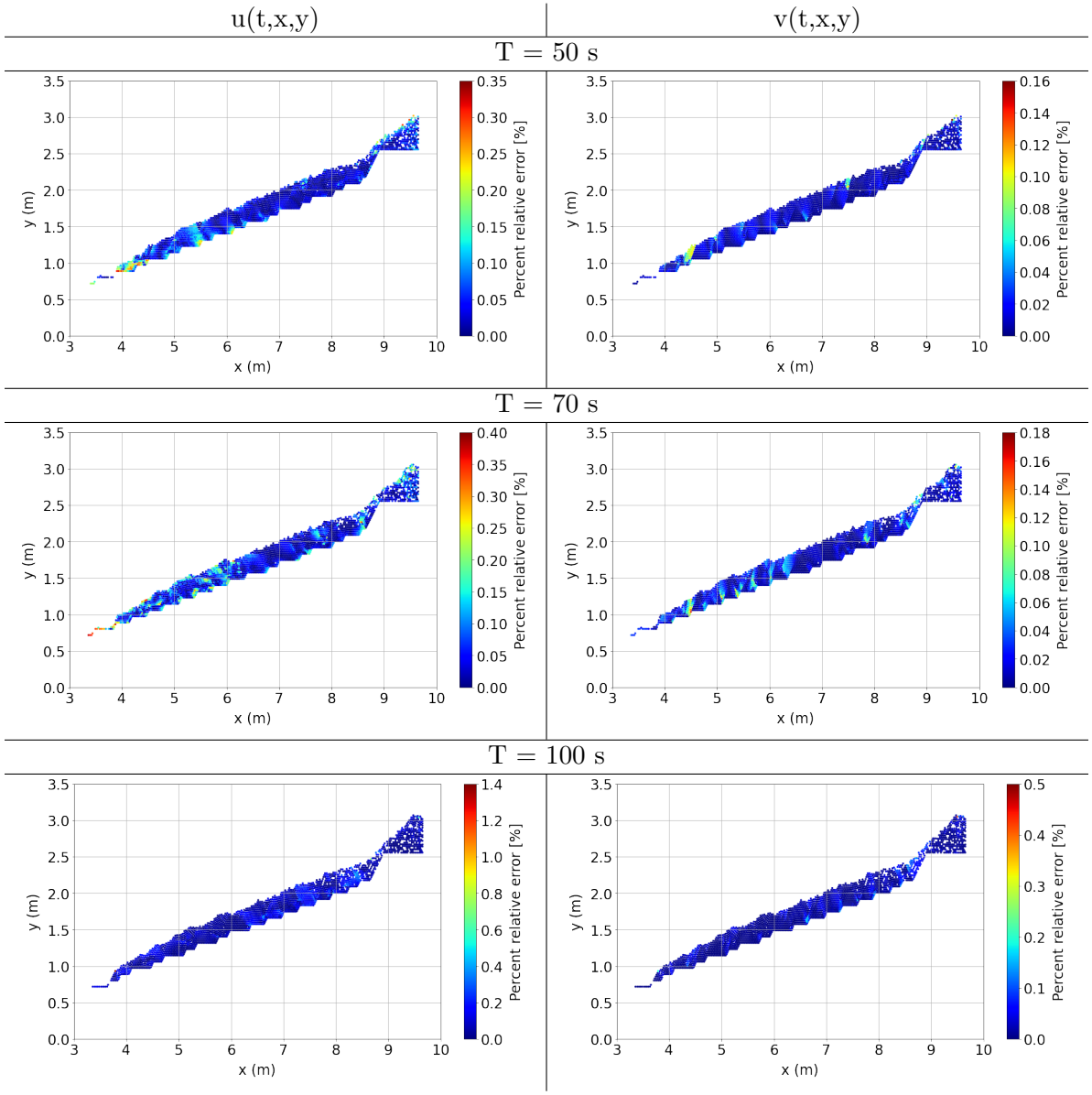

Figure 7: Percent error between the exact and learned velocity in the x and y direction at times 50, 70, and 100 seconds.

fields by having sparse random data of the fields. It's worth noting that a standard neural network method does not have a prior understanding of the problem's physics and its equation of the motions. To overcome this, we used the PINN method that adds the governing equations in the loss function and used the output of the neural network to satisfy the loss function. We used the scattered data set in the spatial-temporal domain using The velocity, pressure, and density fields coming from DEM simulation. Then, we define the PINN which gets the coordinates (x and y) and the simulation time as a neural network's input and outputting two fields of velocity in the x and y direction. To avoid overfitting and under-fitting of the neural

network, we tuned the neural network architecture and architecture of 10 hidden layers, and 200 neurons per hidden layer are chosen to train our neural network. The total number of the data includes approximately 2224 particles in 600-time steps. The simulation time is started at 40 seconds to avoid unstable changes at the beginning of the simulation, and it ends at 100 seconds, which gives us a total time of 60 seconds. the time step is 0.1, altogether we would have 600-time steps. The trained neural network is evaluated by comparing the learned data predicted from the neural network to the exact data provided by using DEM simulations. To do so, the plot of average percent relative error, and mean squared error in time is plotted for both velocities. The average percent error doesn't exceed the value of 1 percent and barely exceeds 0.1 percent, which means that the prediction of the neural network has a good agreement with the actual data. The same trend is observed in the mean squared error which validates the reliability of the PINN in predicting and reconstructing the velocity fields.

References

[1] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[2] George Bebis and Michael Georgiopoulos. "Feed-forward neural networks". In: *IEEE Potentials* 13.4 (1994), pp. 27–31.

[3] Osama Moselhi and Tariq Shehab-Eldeen. "Classification of defects in sewer pipes using neural networks". In: *Journal of infrastructure systems* 6.3 (2000), pp. 97–104.

[4] Magali RG Meireles, Paulo EM Almeida, and Marcelo Godoy Simões. "A comprehensive review for industrial applicability of artificial neural networks". In: *IEEE transactions on industrial electronics* 50.3 (2003), pp. 585–601.

[5] Ayhan Demirbas. "Potential applications of renewable energy sources, biomass combustion problems in boiler power systems and combustion related environmental issues". In: *Progress in energy and combustion science* 31.2 (2005), pp. 171–192.

[6] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[7] Amir Houshang Mahmoudi, Florian Hoffmann, and Bernhard Peters. "Detailed numerical modeling of pyrolysis in a heterogeneous packed bed using XDEM". In: *Journal of analytical and applied pyrolysis* 106 (2014), pp. 9–20.

[8] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization". In: *arXiv preprint arXiv:1409.2329* (2014).

[9] Amir Houshang Mahmoudi et al. "An experimental and numerical study of wood combustion in a fixed bed using Euler–Lagrange approach (XDEM)". In: *Fuel* 150 (2015), pp. 573–582.

[10] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[11] Amir Houshang Mahmoudi, Florian Hoffmann, and Bernhard Peters. "Semi-resolved modeling of heat-up, drying and pyrolysis of biomass solid particles as a new feature in XDEM". In: *Applied Thermal Engineering* 93 (2016), pp. 1091–1104.

[12] Amir Houshang Mahmoudi et al. "Modeling of the biomass combustion on a forward acting grate using XDEM". In: *Chemical engineering science* 142 (2016), pp. 32–41.

[13] Mohammad Mohseni and Bernhard Peters. "Effects of particle size distribution on drying characteristics in a drum by XDEM: A case study". In: *Chemical Engineering Science* 152 (2016), pp. 689–698.

[14] Mikko Hupa, Oskar Karlström, and Emil Vainio. "Biomass combustion technology development–It is all about chemical details". In: *Proceedings of the Combustion institute* 36.1 (2017), pp. 113–134.

[15] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations". In: *arXiv preprint arXiv:1711.10561* (2017).

[16] Zhi-Hua Zhou and Ji Feng. "Deep Forest: Towards An Alternative to Deep Neural Networks." In: *IJCAI*. 2017, pp. 3553–3559.

[17] Maziar Raissi. "Deep hidden physics models: Deep learning of nonlinear partial differential equations". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 932–955.

[18] Maziar Raissi and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357 (2018), pp. 125–141.

[19] Bernhard Peters et al. "XDEM multi-physics and multi-scale simulation technology: Review of DEM–CFD coupling, methodology and engineering applications". In: *Particuology* 44 (2019), pp. 176–193.

[20] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[21] Chengping Rao, Hao Sun, and Yang Liu. "Physics-informed deep learning for incompressible laminar flows". In: *Theoretical and Applied Mechanics Letters* 10.3 (2020), pp. 207–212.

[22] Jared Willard et al. "Integrating physics-based modeling with machine learning: A survey". In: *arXiv preprint arXiv:2003.04919* 1.1 (2020), pp. 1–34.

[23] Han Gao, Luning Sun, and Jian-Xun Wang. "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain". In: *Journal of Computational Physics* 428 (2021), p. 110079.

[24] Yo Nakamura et al. "Physics-Informed Neural Network with Variable Initial Conditions". In: *Proceedings of the* 7^{th} *World Congress on Mechanical, Chemical, and Material Engineering (MCM'21)* (2021). DOI: DOI:10.11159/htff21.113.

[25] Li Wang and Zhenya Yan. "Data-driven rogue waves and parameter discovery in the defocusing nonlinear Schrödinger equation with a potential using the PINN deep learning". In: *Physics Letters A* 404 (2021), p. 127408.

[26] Shengze Cai et al. "Physics-informed neural networks (PINNs) for fluid mechanics: A review". In: *Acta Mechanica Sinica* (2022), pp. 1–12.

[27] Jan Oldenburg et al. "Geometry aware physics informed neural network surrogate for solving Navier-Stokes equation (GAPINN)". In: (2022).