Physics-Informed Deep Learning for Modelling Particle Aggregation and Breakage Processes

Xizhong Chen[a], Li Ge Wang[b,c*], Fanlin Meng[d], and Zheng-hong Luo[e]

[a]Process and Chemical Engineering, School of Engineering,
University College Cork, Cork, Ireland
[b]Process Systems Enterprise, Hammersmith, London, UK
[c]Department of Chemical and Biological Engineering, University of Sheffield, UK
[d]Department of Mathematical Sciences, University of Essex, Colchester, UK
[e]Department of Chemical Engineering, School of Chemistry and Chemical Engineering, State Key Laboratory of Metal Matrix Composites, Shanghai Jiao Tong University, Shanghai 200240, P. R. China
*Email: L.G.Wang@psenterprise.com

Abstract

Particle aggregation and breakage phenomena are widely found in various industries such as chemical, agricultural and pharmaceutical processes. In this study, a physics-informed neural network is developed for solving both the forward and inverse problems of particle aggregation and breakage processes. In this method, the population balance equation is directly embedded in the loss function of a neural network so that the network can be trained efficiently and fulfil physical constraints. For the forward problems, solutions of population balance equations are obtained through the optimization of the neural network where the predictions well match the analytical solutions. In the inverse modelling, the data-driven discovery of model parameters of population balance equations is investigated. The sensitivity regarding the selection of different neural network structures is also investigated. The developed population balance equations embedded with neural network approach is promising for solving inverse problems of particle aggregation and breakage processes with noisy observation data.

Keywords: Physics-Informed Neural Network; Population balance equation; Aggregation; Breakage; Inverse problem; Parameter estimation

1. Introduction

Population balance model (PBM) is widely used for modelling particulate systems in a variety of applications such as crystallizations, millings, polymerization and granulations [1-6]. PBM can quantify how a distribution of particle descriptors changes over the studied time based on kernel functions representing the relevant mechanisms that affect the particle evolutions. It can also be coupled to computational fluid dynamics where the spatial evolution of particle descriptors can be tracked [7-10]. However, the model consists of highly nonlinear integral-partial different equations, which usually needs to be solved numerically. Until now, various numerical techniques are therefore developed to solve it, including the method of class [11], moment methods [12, 13], Monte Carlo methods [14] and meshless radial basis method [15]. Hounslow et al. (1988) and Litster et al. (1995) developed a discretization procedure for the aggregation problems using the method of class. This method was extended to solve breakage problem by Vanni (2000). Kumar and Ramkrishna (1996) developed a method of discretization with a pivot technique that involves a selective refinement of a relatively coarse grid. The number of sections is kept to a minimum and they have shown that the method can be successfully applied to agglomeration/breakage problems. A recent review of the solutions of population balance equations for simulating disperse multiphase flows is given by Shiea, Buffo, Vanni and Marchisio [16]. However, the existing numerical schemes are designed to solve the forward problems of population balance equation but are difficult to deal with the inverse problems. Many real-world applications of PBM need to be formulated as inverse modelling problems, which are to identify a set of parameters or functions to make the outputs of forward analysis match the desired results or measurements. The calibration or search of the optimal kernel parameters of the population balance model using the existing schemes still needs massive additional efforts.

Over the past few years, there has been a revolution in the successful application of deep learning in various fields such as computer vision, natural language processing, image classification and recognition. Despite its great success in these fields, applying deep learning in the field of physics-based particulate systems is relatively scarce. Physical processes can usually be described by conservation laws with dynamic and kinematic constraints represented through a set of ordinary or partial differential equations. Machine learning has the capabilities in learning complex representations of data and approximating physical processes, which provides the potential to solve the challenging open problems of physical systems. For example, in the wet granulation process, the underlying physics are not completely available (e.g. the breakage and agglomeration mechanisms). However,some extra information from the experimental data could be available for machine learning methods to explore. By integrating the information from both physics and data exploitation through machine learning methods, it will be well positioned to recover and identify the particle kinetic in a complex process such as granulation, milling and crystallisation. In recent years, the scientific machine learning method [17] is being actively explored to solve these problems. The method can be trained with both experimental and numerical data, which shows a great potential to optimize the complicated industrial processes. Particularly, the physics informed neural network (PINN) method has attracted much attention [18-20]. PINN endows a neural network model with known equations that govern the physics of a system [19, 21]. In addition to training data, the governing equations, as well as initial and boundary conditions of a mechanistic model can be embedded in the cost function of PINN. The method has been proven to be very successful in solving systems of ordinary differential equations and partial differential equations, such as Schrodinger, Allen-Cahn and Navier-Stokes equations [22]. Moreover, the method is also shown to be applicable in solvingboth forward and inverse problems of differential equations sytems, which can overcome the limitations of existing numerical methods and is promising to be applied in a wide range of physical systems [19].

In this study, we develop a physics-informed deep learning framework to solve both forward and inverse problems of the population balance model that describes and predicts the particle distribution undergoing aggregation and breakage. The governing equations of the population balance model is directly integrated into the loss function of a neural network to penalise the unphysical predictions so that the framework can be trained efficiently and fulfils physical constraints. The key advantage of such model structure is its interpretability and generalization for various particulate processes. To demonstrate the feasibility of physics-informed deep learning framework, cases with simple kernels are studied in the forward problems since analytical solutions are available for training. A laboratory ball mill with sparse experimental data will be used to test the performance of the framework for solving inverse problems. The paper is structured as follows. The mathematical theory is described in section 2. Exemplar forward solutions and model parameter discovery of population balance equations with simulation results are given in section 3. Finally, conclusions and future work are given in section 4.

2. Mathematical model

2.1 Population balance equations

Considering a well-mixed system of particles that undergo aggregation and breakage, the population balance equation can be written as below

$$\frac{dN(v,t)}{dt} = \frac{1}{2}\int_0^v N(v-v',t)N(v',t)\,\beta(v-v',v')dv' - \int_0^\infty N(v,t)N(v',t)\,\beta(v,v')dv' - \gamma(v)N(v,t) + \int_v^\infty \alpha(v,v')\gamma(v')N(v',t)\,dv' , \tag{1}$$

where $N(v,t)$ is the number density distribution function representing the number of particles per system volume with particle volume v at time *t*. $\beta(v,v')$ is the aggregation kernel. $\gamma(v)$ is the breakage rate kernel, $\alpha(v,v')$ is the breakage daughter size distribution function which represents the probability of making a daughter volume v from a parent volume v'. The first term on the right-hand side of the Eq. 1 calculates the birth rate of particle volume *v* due to aggregation of particles smaller than particle volume *v*. The second term calculates the death rate of particle volume v due to aggregation with other particles. The third term calculates the death rate of particle volume v due to fragmentation. The final term calculates the birth rate of particle volume v due to fragmentation of particles larger than particle volume v. A linear grid ($v_i = iv_0$) can be used to discretize the Eq.1, which leads to the following formula [23-25],

$$\frac{dN_i}{dt} = \frac{1}{2}\sum_{j=1}^{i-1} N_j N_{i-j}\,\beta_{j,i-j} - N_i \sum_{j=1}^{n} N_j\,\beta_{i,j} - \gamma_i N_i + \sum_{j=i+1}^{n} \alpha_{ij}\gamma_j N_j \tag{2}$$

where the subscript *i* and *j* represent *ith* and *jth* discretized grid, respectively.

2.2 Deep neural network

Deep neural networks have been proven to be a powerful metamodeling tool and complex function approximator [26]. Recent studies have shown that deep neural networks are a promising and effective method to establish metamodel for approximating temporal response of dynamical systems and approximating material constitute laws, which outperforms traditional metamodeling techniques in terms of both the capability of capturing nonlinear input-output relationships and prediction accuracy [27, 28]. In this work, a fully connected deep feedforward neural network was used. In this network, the information passes forwardly from the input layer through several hidden layers to the output layer. Despite many other types of neural networks have been developed in the past decades, the feedforward neural network is simple yet effective. It has been shown to be sufficient for dealing with most of the partial differential equation problems [29, 30]. A L-layer neural network $f^L(\boldsymbol{x})$ can be written as follow,

$$Input\ layer: f^0(\boldsymbol{x}) = \boldsymbol{x} \in \boldsymbol{R}^{d_{in}}, \tag{3}$$

$$Hidden\ layer: f^l(\boldsymbol{x}) = \sigma(\boldsymbol{W}^l f^{l-1}(\boldsymbol{x}) + \boldsymbol{b}^l) \in \boldsymbol{R}^{f_l}, \tag{4}$$

$$Output\ layer: f^L(\boldsymbol{x}) = \boldsymbol{W}^L f^{L-1}(\boldsymbol{x}) + \boldsymbol{b}^L \in \boldsymbol{R}^{d_{out}}, \tag{5}$$

where $\boldsymbol{x}$ is the input of the neural network, $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ are the weight matrix and bias vector in the l-th layer, respectively. σ is a non-linear activation function applied element-wisely. The feed forward neural network here consists of one input layer, l -1 hidden layers and one output layer.

2.3 Physics-informed Neural network

The schematic of the population balance embedded neural networks (PBNN) used in this work is shown in Figure 1. In this framework, a neural network $\widehat{N}(\boldsymbol{x}; \boldsymbol{\theta})$ is firstly constructed to be a surrogate of the solution $N(\boldsymbol{x})$. Here $\boldsymbol{\theta}$ is the hyperparameters containing the set of all weight matrices $\boldsymbol{W}$ and bias vectors $\boldsymbol{b}$ in the neural network $\widehat{N}$. An important feature of PBNN is that it respects the physical laws described by the population balance equations. This is achieved through modifying the loss function of the neural network. The governing equations are included as an additional term to punish the physical inconsistency of the output results of the neural network. One advantage of using a neural network as a surrogate model is that the derivative can be obtained through the chain rule for differentiating compositions of functions in the optimization processes. The derivatives can be evaluated using backpropagation in a deep neural network [31]. Herein, we use an automatic differentiation routine to compute the derivatives of the network's outputs with respect to the network's inputs. It firstly applies a forward pass to compute the values of all variables and then applies a backward pass to compute the derivatives. Automatic differentiation provides an accurate and efficient way to calculate derivatives, which has been implemented as a standard modular in most of the modern deep learning frameworks [32]. On the one hand, the automatic differentiation algorithm provides a robust way to update the hyperparameters in the neural network for optimization the trainings. On the other hand, it also offers an opportunity to construct a physical constraint loss function by directly incorporating the governing ordinary differential equations. Therefore, it has gained significant attention in both the machine learning and physical modelling community.

In this work, the loss function of the neural network is constructed as follows,

$$L(\boldsymbol{\theta}) = \lambda_1\left(\frac{1}{M_u}\sum_{i=1}^{M_u}\left|\widehat{N}^i(\boldsymbol{\theta}) - N\left(x^i, t^i\right)\right|^2\right) + \lambda_2\left(\frac{1}{M_f}\sum_{i=1}^{M_f}\left|F(N^f)\right|^2\right) \tag{6}$$

$$F(N_i) = \frac{dN_i}{dt} - \frac{1}{2}\sum_{j=1}^{i-1} N_j N_{i-j}\,\beta_{j,i-j} + N_i\sum_{j=1}^{n} N_j\,\beta_{i,j} + \gamma_i N_i - \sum_{j=i+1}^{n} \alpha_{ij}\gamma_j N_j \tag{7}$$

Here, M_u is the number of obversation points that are used to train the neural network; M_f is the number of discretized grids that are used to approximate the particle size distribution. The first term in the right-hand side of Eq. 6 represents the mean squared error of training data on initial and observation training data. The second term is to punish the physics inconsistency enforced by the population balance equation in Eq. 2. λ_1 and λ_2 are the regulation coefficients to balance the weight of the two mean squared errors, respectively. They are chosen to be one in this work. Note that the initial particle size distribution is imposed by hard constraint, i.e.

the output of the neural network is modified to the given initial particle size distribution during each optimization step. It is found that hard constraints of initial and boundary conditions significantly accelerate the optimization process and are very important for ensuring the uniqueness of the learned solutions [33]. The optimization is first done through the widely used stochastic gradient descent ADAM algorithm [34] to minimize the loss function. Later, a second-order method L-BFGS is chosen in further optimization steps [35]. The L-BFGS optimization is found to greatly improve training efficiency, particularly in solving inverse problems.

3. Simulation results

In this section, two distinct problems will be studied. The first problem is to solve the population balance equation in forward problem, which is to estimate the unknown hidden state of the system given fixed model parameters. The second problem is data-driven discovery of the population balance equation, which is to predict the model parameters that can best describe the observation data. Analytical solutions of the studied cases are used to train the PBNN. The effects of the neural network in PBNN will also be investigated. In general, a good combination of the architecture of neural networks is important for achieving a robust prediction. After some preliminary trials, a neural network architecture with 8 hidden layers (each layer has 20 neurons) has been chosen for most of the studied cases. Hyperbolic tangent (tanh) is used as the activation function of the neurons and Xavier normal initializer is used to set the initial random weights of layers where initial weights are drawn from a truncated normal distribution [36]. The effects of the network on the performance will also be briefly discussed in the following section.

3.1 Solving forward problem of the aggregation and breakage processes

Scott [37] presented an analytical solution for aggregation processes with a constant kernel and various initial conditions. Here, the aggregation case with a Gaussian-like distribution as an initial condition is solved. The initial condition is given as follows,

$$n(v,0) = \frac{N_0}{v_0}\left(\frac{v}{v_0}\right)e^{-v/v_0} \tag{8}$$

The value N_0 and v_0 are the initial number of particles per unit volume and the initial mean volume of the particles. The analytical solution for the number density function is given by Eq 9

$$n(v,t) = \frac{N_0}{v_0}\frac{4e^{-2\xi}}{\xi(T+2)^2}\sum_{k=1}^{\infty}\frac{(2\xi)^{2(k+1)}}{\Gamma(2(k+1))}\left(\frac{T}{T+2}\right)^k \tag{9}$$

where $\xi = v/v_0$. T is the non-dimensional time given by Eq 10

$$T = \beta_0 N_0 t \tag{10}$$

Here, β_0 is the aggregation rate constant. The analytical solution of the total number of particles is provided as below.

$$N(t) = 2N_0/(T+2) \tag{11}$$

Since the analytical formulation is general, a benchmark case with $N_0 = 1, \beta_0 = 1$ and $v_0 = 1$ is investigated here. The volume of the smallest particle is considered to be 0.25 and a grid with 40 particle size bins is used. The simulation time step is set to 0.01 and the aggregation time between 0 and 1 second is modelled.

Figure 2 shows the loss of PBNN during the optimization process. It can be seen that the loss continues to decrease as expected. A two-stage optimization process is adopted here. The first

2000 epoch optimization steps are achieved through the stochastic gradient descent ADAM algorithm with learning rate of 0.1. It is found that there is a substantial decrease of the loss at the beginning of the optimization while the decreasing rate becomes quite slow after 1000 epochs. To further speed up the optimization process, the optimization algorithm is switched to a second-order L-BFGS method after 2000 epochs where thereis a small jump of the loss right after switching. Nevertheless, the loss drops sharply during the first 100 epochs of the L-BFGS. After that, the decreasing rate of the loss becomes very slow due to the convergence of the solutions, which is further confirmed throughcomparisons between the analytical solution and the predicted number density function of the final time step as shown in Figure 3. It can be seen that the predictions are getting closer to the analytical solution along with the optimization process. The final prediction at 3000 epochs has excellent agreement with the analytical solution.

The impacts of the depth of the neural network architecture on the loss and prediction are shown in Figure 4. The neural networks have 20 neurons in each layer and three depths (i.e. 4 layers, 8 layers and 12 layers) are compared. In general, all three networks generate satisfactory predictive performance, which indicates the robustness of the proposed approach. It is observed that the 4 layers network results in a high oscillation during the first phase ADAM optimization and produces a relatively less accurate prediction on the final particle size number density. In comparison, a 12 layers network seems to produce the best prediction results. However, it contains more parameters and thus needs more computational resources for training. As a rule of thumb, it is usually preferable to choose the simplest network structure that can still achieve a desired prediction performance. It is found that the 8 layers network produces almost the same results as the 12 layers network. Table 1 lists a series of tests with different combinations of numbers of hidden layers and neurons. The relative L2 error is calculated as $\epsilon =$

$\frac{\sqrt{\sum_{i=1}^{M}\left\|n_{pred}^{i}-n_{ana}^{i}\right\|^{2}}}{\sqrt{\sum_{i=1}^{M}\left\|n_{ana}^{i}\right\|^{2}}}$. In general, the error deceases when the network becomes wider (more neurons per layer) and deeper (more hidden layers). However, the accuracy does not change too much when the number of hidden layers is increasing from 8 to 12. Furthermore, there is no significant improvement when using more than 20 hidden neurons per layer. Therefore, a network of 8 hidden layers with 20 hidden neurons architecture is preferred in this work.

Figure 5 illustrates the final convergence of the number density function solution in terms of a three-dimensional surface and a contour plot. Note that the solution from the traditional Artificial Neural Network (ANN) method is also plotted. The ANN has the same neural network setting as PBNN except that population balance equations are not encoded in the loss function. The loss function of ANN only contains the root mean square error between the neural network output and the analytical solution of the particle size distribution. The training is performed by minimizing the loss function over multiple time steps. It can be seen that both ANN and PBNN can successfully approximate a similar shape as the analytical solution.The number of small particles decreases, and the number of larger particles increases with the aggregation time. As shown in the contour plot, the aggregation process looks like the diffusion of concentration field in a temporal evolution. However, it is obvious that there are some discrepancies between the ANN and the analytical solution whereas the solution from PBNN is closer to the analytical solution. To validate the prediction, Figure 6 compares the time evolution of the total number of particles predicted by PBNN, ANN, method of class with 10 discretized bins and the analytical solution. It can be seen that the PBNN predicts that the number of particles decreases with the aggregation time, which is in a very good agreement with the analytical solution. In comparision, the solution with the method of class slightly overestimates the total number of particles through the aggregation process. In addition, the prediction of ANN deviates significantly from the analytical solution, whichindicates that the conservation law is difficult to maintain using ANN although it can approximate the shape of the solution. By training with more numerical results obtained from different sets of parameters may help improve the predictions of ANN. However, this requires more computational resources and it still could not guarantee that the physical constraints of the distribution are met. Figure 7 further shows that the predictions of the particle size distribution at different aggregation time agree with the analytical solution, highlighting that the dynamic evolution of the aggregation process is correctly captured.

Ziff and McGrady [38] derived an analytical solution for a uniform binary breakup case. The breakage kernel and the daughter particle distribution function are given as below.

$$\gamma = v^2 \tag{12}$$

$$\alpha(v, v') = 2/v' \tag{13}$$

In this particular case, the initial particle number distribution function is set to be monodispersed, i.e. $n(v, 0) = \delta(v - v_0)$. The analytical solution of particle number density distribution is given by

$$n(v, t) = e^{-tv^2}[\delta(v - v_0) + 2tv_0\Theta(v_0 - v)] \tag{14}$$

where Θ is the step function. The total number of particles can be obtained by integrating the number density distribution. The initial particle size v_0 is set to 1 in this case. The volume of the smallest particle is considered to be 0.25 and a grid of 40 particle size bins is used.

Figure 8 presents a comparison of the numerical and analytical solutions for the variation of the total number of particles. It is comfirmed that the total number of particles is increasing during a breakage process and the predictions of PBNN almost overlap with the analytical solutions. The predictions from the method of class with 10 discretized bins slightly overestimate the total number of particles throughout the breakage process and there is a distinct deviation between the predictions from ANN and analytical solution. Furthermore, Figure 9 shows that the predictions of the trajectories of particle number density function by PBNN are in a good agreement with the analytical results. These results indicate that the time evolution of particle number density function during the breakage process is well predicted by the present PBNN method.

3.2 Data-driven discovery of model parameters

As mentioned in the introduction, inverse problems of particle aggregation and breakage process are challenging to handle using exsiting methods. Therefore, it is worth exploring the applicability of using PBNN to solve inverse problems. The problem studied here can also be stated as the identifications of the parameters that best describe the provided observation data. To estimate the parameters, the unknow parameters in the equation are treated as network parameters that change during the optimization phase. This is achieved by defining them as trainable variable objects in TensorFlow [39].

To demonstrate the applicability of the constructed framework, the analytical solutions of the cases in section 3.1 are used as training data. Meanwhile, the aggregation and breakage rate constants are not given in advance but are defined as variables for the program to estimate. Figure 10 shows the evolution of the convergence in the model parameter identification process. Notably, the estimated parameters converge to the true values of kernel parameters in both aggregation and breakage cases. Meanwhile, it can be seen that the optimization is quite efficient, i.e. the true kernel parameters are estimated with around 1000 epochs.

In a realistic process, there are inherently noises in the experimental observation data due to the uncertainties in the measurement equipment, intrinsic variety of the particulate properties, and complexity of the processes. To further consider this effect and test the power of the proposed PBNN framework, we deliberately add some Gaussian white noise $G\ (0, \sigma)$ to the training data of the aggregation case for increasing the estimation difficulty of the kernel parameters. Here, G is a normal distribution with a standard deviation σ. Table 2 presents the comparisons between the true value and the final estimated values with different levels of Gaussian noises. It is found that the accuracy of the prediction decreases with the increase of the magnitude of the noises. Nevertheless, the aggregation rate constant can still be estimated reasonably well even with large noisy data. Theoretically, the framework can also be used to denoise the experimental measurements and estimate the uncertainties. However, this is beyond the scope of the present study but will be investigated in our future work.

To further demonstrate the model's applicability, the proposed PBNN approach is used to determine the milling parameters of a platinum group minerals ore material in a laboratory test. The material was milled in a Wits pilot under a rotating speed of 60 rpm [40]. The mono size feed samples were prepared by sieving 2 kg from the 15 kg platinum ore batches. The size of the feed material is in the range of 600-850 micron. To investigate the milling particle size evolution, some kernel function closures in PBNN need to be defined. An empirical selection function proposed by Austin, Klimpel and Luckie [41] is used to calculate the breakage rate. The formula is given as follows,

$$\gamma = a \cdot v^{\varepsilon} \cdot \frac{1}{1 + \left(\frac{v}{\mu}\right)^{\Lambda}} \tag{15}$$

where a and μ are two parameters that depend on the mill operational conditions, ε and Λ are two parameters that depend on the properties of milling materials. The breakage daughter size distribution is calculated as follows [41],

$$B(v,v') = \emptyset\left(\frac{v}{v'}\right)^{\varphi} + (1-\emptyset)\left(\frac{v}{v'}\right)^{\omega} \tag{16}$$

$$\alpha(v,v') = \frac{\partial B(v,v')}{\partial v} \tag{17}$$

where $B(v,v')$ is the cumulative breakage distribution function, φ and ω are two power index related to material characteristics, $\emptyset$ is also a material-dependent parameter representing the fraction of fines that will be produced in a single fracture step. In total, there are seven parameters to be estimated, which includes four breakage rate parameters (a, μ , ε and Λ) and three breakage daughter size distribution parameters ($\emptyset, \varphi$ and ω). All the seven parameters are set to be trainable variables in PBNN. The value $\emptyset$ is limited to vary from 0 to 1 while the other parameters are constrained to able to change from 0 to 10. The milling particle size distributions from 0 to 30 minutes are used to estimate the parameters.

Th final estimated parameters from PBNN are listed in Table 3. Figure 11 shows the comparison between measured and predicted particle size distribution evolution. It is found that there are some quantitative discrepancies of the particle size distribution in the early milling time, which may be due to the uncertainties in the function form of breakage kernel or the variability in the experiments. In general, a good match between the predicted and the experimentally measured particle size distribution evolution is observed. Generally, the physics-informed neural-network approach is deemed as a data-efficient machine learning method since the underlying laws of physics are explicitly encoded [19, 42, 43]. In practice, with the increase of the complexity of the kernel functions, PBNN may need more data to improve the model capability.

4. Conclusion

In this work, a physics-informed deep learning approach is developed for solving and discovering particle aggregation and breakage processes. The governing equations of the population balance model are incorporated into the loss function of a neural network by taking advantage of the recently proposed advanced automatic differentiation algorithm. The embedding of physical constraints makes it more efficient to train the neural network and achieve reliable predictions. Compared with classical methods for modeling particle aggregation and breakage processes, the proposed population balance neural network approach can be used to solve both forward and inverse problems. The impact of the depth of the network on the final predictions is also analysed. A shallow network is found to produce an oscillation of loss during the optimization process and a deeper network can help increase the accuracy of the predictions. It is demonstrated that the proposed approach can match well the analytical solutions available in the literature. It is also shown that the kernel parameters in the reverse problem can be successfully identified even with noisy observation data. Furthermore, the method is used to determine the preliminary parameters from a pilot-scale milling experiment. A good match between the predicted and the experimentally measured particle size distribution evolution is observed.

Although training data is not a prerequisite in the forward problem of a physics-informed neural network based approach, most of the current applications still use labeled data to help the network converge [19, 29, 42, 43]. In practice, we also found that the network converges faster with the aid of analytical solutions as labeled data in the forward problem. Noteworthy, some very recent works have been able to use physics-informed neural network based approach without labeled data [33, 44, 45]. Therefore, such methods will be explored for dealing with more complexed kernels in our future work. Another limitation of the current PBNN is that it can only be used to predict the undetermined parameters of several candidate kernel equations. As for future work, it would be beneficial to use the method to discover closure formulation for kernel functions, which could be achieved with the aid of recently proppposed advanced symbolic regression [46] or sparse identification of nonlinear dynamics framework [47].

Furthermore, the physics-informed neural network approach has shown to be successful in solving fluid flow problems like a meshless CFD solver [19, 33, 43, 44]. Therefore, we will investigate the possibility to extend the current work to implement a physics-informed neural network based CFD-PBM solver to deal with more complicate particulate processes.

Acknowledgments

The first author would like to acknowledge the startup funding from University College Cork. The corresponding author would like to thank Innovate UK for funding the Knowledge Transfer Partnership between University of Sheffield and Process Systems Enterprise. The MIT xPRO course Machine Learning, Modelling and Simulation Principles deliverd by Prof. Youssef Marzouk from MIT is highly appreciated and inspirational to conceptualize this work.

Nomenclature

a	Fitting parameter, -
b	Bias vector of neural network, -
B	Cumulative breakage distribution function, -
f	Neural network, -
F	Physics inconsistency of the surrogate solution, -
L	Loss function, -
M_u	Number of obversation points, -
M_f	Number of discretized grids, -

N	Number density distribution function, m^{-6}
$\widehat{N}$	Surrogate solution, m^{-6}
t	Time, s
T	Non-dimensional time, -
v	Particle volume, m^{3}
W	weight matrix of neural network, -
x	Input of neural network, m^{-6}

Greek letters

α	Breakage daughter size distribution
β	Aggregation kernel
γ	Breakage rate kernel
λ	Regulation coefficient
μ	Fitting parameter
ε	Material fiting parameter
ϵ	Relative error
φ	Material fiting parameter

ω	Material fiting parameter
θ	Hyperparameters
ξ	Non-dimemsional particle volume
δ	Delta function
Λ	Material fitting parameter
$\emptyset$	Material fitting parameter
Γ	Gamma function
Θ	Step function

Reference

[1] D. Ramkrishna, Population balances: Theory and applications to particulate systems in engineering, Elsevier2000.
[2] C.D. Immanuel, F.J. Doyle III, Solution technique for a multi-dimensional population balance model describing granulation processes, Powder Technology 156 (2005) 213-225.
[3] X. Chen, L.G. Wang, J.Y. Ooi, A DEM-PBM multiscale coupling approach for the prediction of an impact pin mill, Powder Technology (2020).
[4] B. Olaleye, C.-Y. Wu, L.X. Liu, Impact of feed material properties on the milling of pharmaceutical ribbons: A PBM analysis, International Journal of Pharmaceutics (2020) 119954.
[5] C.Y. Ma, X.Z. Wang, K.J. Roberts, Morphological population balance for modeling crystal growth in face directions, AIChE journal 54 (2008) 209-222.
[6] Z. Li, L.G. Wang, W. Chen, X. Chen, C. Liu, D. Yang, Scale-up procedure of parameter estimation in selection and breakage functions for impact pin milling, Advanced Powder Technology 31 (2020) 3507-3520.
[7] X.Z. Chen, Z.H. Luo, W.C. Yan, Y.H. Lu, I.S. Ng, Three-dimensional CFD-PBM coupled model of the temperature fields in fluidized-bed polymerization reactors, AIChE Journal 57 (2011) 3351-3366.
[8] T. Wang, J. Wang, Y. Jin, Population balance model for gas− liquid flows: Influence of bubble coalescence and breakup models, Industrial & engineering chemistry research 44 (2005) 7540-7549.

[9] Z. Gao, D. Li, A. Buffo, W. Podgórska, D.L. Marchisio, Simulation of droplet breakage in turbulent liquid–liquid dispersions with CFD-PBM: comparison of breakage kernels, Chemical Engineering Science 142 (2016) 277-288.
[10] X. Yu, M.J. Hounslow, G.K. Reynolds, A. Rasmuson, I. Niklasson Björn, P.J. Abrahamsson, A compartmental CFD-PBM model of high shear wet granulation, AIChE Journal 63 (2017) 438-458.
[11] M. Hounslow, R. Ryall, V. Marshall, A discretized population balance for nucleation, growth, and aggregation, AIChE journal 34 (1988) 1821-1832.
[12] D.L. Marchisio, J.T. Pikturna, R.O. Fox, R.D. Vigil, A.A. Barresi, Quadrature method of moments for population-balance equations, AIChE Journal 49 (2003) 1266-1276.
[13] J. Su, Z. Gu, Y. Li, S. Feng, X.Y. Xu, Solution of population balance equation using quadrature method of moments with an adjustable factor, Chemical Engineering Science 62 (2007) 5897-5911.
[14] Y. Lin, K. Lee, T. Matsoukas, Solution of the population balance equation using constant-number Monte Carlo, Chemical Engineering Science 57 (2002) 2241-2252.
[15] S. Alzyod, S. Charton, A meshless Radial Basis Method (RBM) for solving the detailed population balance equation, Chemical Engineering Science (2020) 115973.
[16] M. Shiea, A. Buffo, M. Vanni, D. Marchisio, Numerical Methods for the Solution of Population Balance Equations Coupled with Computational Fluid Dynamics, Annual Review of Chemical and Biomolecular Engineering 11 (2020) 339-366.
[17] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence, USDOE Office of Science (SC), Washington, DC (United States), 2019.
[18] S. Rudy, A. Alla, S.L. Brunton, J.N. Kutz, Data-driven identification of parametric partial differential equations, SIAM Journal on Applied Dynamical Systems 18 (2019) 643-660.
[19] 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, Journal of Computational Physics 378 (2019) 686-707.
[20] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences 115 (2018) 8505-8510.
[21] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, Journal of Computational Physics 397 (2019) 108850.
[22] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proceedings of the National Academy of Sciences 116 (2019) 15344-15349.
[23] R. Charls, Energy-size reduction relationships in comminution, Trans. AIME 9 (1957) 80-88.
[24] D. Fuerstenau, A.-Z. Abouzeid, The energy efficiency of ball milling in comminution, International Journal of Mineral Processing 67 (2002) 161-185.
[25] G.M. Hidy, On the theory of the coagulation of noninteracting particles in Brownian motion, Journal of Colloid Science 20 (1965) 123-144.
[26] S. Chen, S. Billings, Neural networks for nonlinear dynamic system modelling and identification, International journal of control 56 (1992) 319-346.
[27] D.J. Fonseca, D.O. Navaresse, G.P. Moynihan, Simulation metamodeling through artificial neural networks, Engineering applications of artificial intelligence 16 (2003) 177-183.
[28] K. Wang, W. Sun, Meta-modeling game for deriving theory-consistent, microstructure-based traction–separation laws via deep reinforcement learning, Computer Methods in Applied Mechanics and Engineering 346 (2019) 216-241.

[29] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, arXiv preprint arXiv:1907.04502 (2019).
[30] Z. Liu, Y. Yang, Q.-D. Cai, Solving Differential Equation with Constrained Multilayer Feedforward Network, arXiv preprint arXiv:1904.06619 (2019).
[31] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, nature 323 (1986) 533-536.
[32] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, The Journal of Machine Learning Research 18 (2017) 5595-5637.
[33] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Computer Methods in Applied Mechanics and Engineering 361 (2020) 112732.
[34] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[35] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical programming 45 (1989) 503-528.
[36] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026-1034.
[37] W.T. Scott, Analytic studies of cloud droplet coalescence I, Journal of the atmospheric sciences 25 (1968) 54-65.
[38] R.M. Ziff, E. McGrady, The kinetics of cluster fragmentation and depolymerisation, Journal of Physics A: Mathematical and General 18 (1985) 3027.
[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, TensorFlow: Large-scale machine learning on heterogeneous systems, (2015).
[40] N. Chimwani, D. Glasser, D. Hildebrandt, M.J. Metzger, F.K. Mulenga, Determination of the milling parameters of a platinum group minerals ore to optimize product size distribution for flotation purposes, Minerals Engineering 43 (2013) 67-78.
[41] L.G. Austin, R.R. Klimpel, P.T. Luckie, Process engineering of size reduction: ball milling, Society of Mining Engineers of the American Institute of Mining, Metallurgical and Petroleum Engineers1984.
[42] M. Raissi, G.E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics 357 (2018) 125-141.
[43] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (2020) 1026-1030.
[44] C. Rao, H. Sun, Y. Liu, Physics-informed deep learning for incompressible laminar flows, Theoretical and Applied Mechanics Letters 10 (2020) 207-212.
[45] O. Hennigh, S. Narasimhan, M.A. Nabian, A. Subramaniam, K. Tangsali, M. Rietmann, J.d.A. Ferrandis, W. Byeon, Z. Fang, S. Choudhry, NVIDIA SimNet^{TM}: an AI-accelerated multi-physics simulation framework, arXiv preprint arXiv:2012.07938 (2020).
[46] S.-M. Udrescu, M. Tegmark, AI Feynman: A physics-inspired method for symbolic regression, Science Advances 6 (2020) eaay2631.
[47] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (2016) 3932-3937.

Figure Captions

Figure 1. Diagram of a population balance neural network (PBNN) model for solving the forward and inverse problems of particle aggregation and breakage systems.

Figure 2 The optimization process of PBNN for solving the aggregation case

Figure 3 The prediction of the particle number density during the optimization process. (a) 200 epochs (b) 1000 epochs (c) 2000 epochs (d) 3000 epochs

Figure 4 Effect of the depth of the neural network architecture on the loss and prediction results in the aggregation case. (a) Loss function during optimization (b) Comparisons between the predictions and the analytical solution of final size number density.

Figure 5. Solution of the number density function in the aggregation case

Figure 6 Comparison between the prediction of total number of particles evolution and the analytical solution in aggregation case.

Figure 7. Comparison of the predictions and the analytical solutions of the time evolution of particle number density in aggregation case

Figure 8 Comparison between the prediction of total number of particles evolution and the analytical solution in the breakage case.

Figure 9 Comparison of the predictions and the analytical solutions of the time evolution of particle number density in the breakage case

Figure 10 Kernel parameter estimations by PBNN (a) aggregation rate constant estimation (b) breakage rate constant estimation

Figure 11 Measured and predicted particle size distributions in a ball milling laboratory test

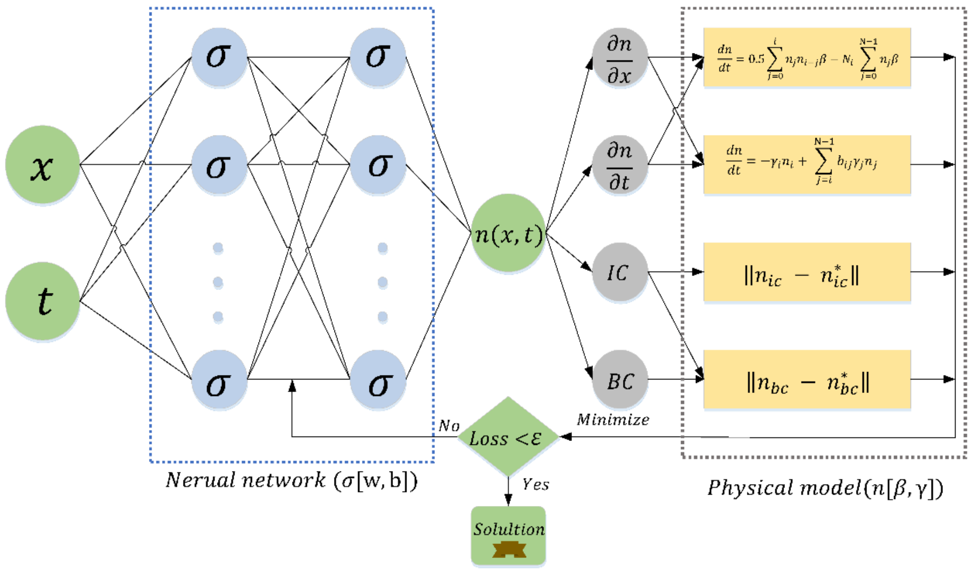

x
t
σ
$n(x,t)$
$\frac{\partial n}{\partial x}$
$\frac{\partial n}{\partial t}$
IC
BC
$\frac{dn}{dt} = 0.5 \sum_{j=0}^{i} n_j n_{i-j} \beta - N_i \sum_{j=0}^{N-1} n_j \beta$
$\frac{dn}{dt} = -\gamma_i n_i + \sum_{j=i}^{N-1} b_{ij} \gamma_j n_j$
$\|n_{ic} - n_{ic}^*\|$
$\|n_{bc} - n_{bc}^*\|$
Minimize
No
$Loss < \varepsilon$
Yes
Solultion
Nerual network (σ[w, b])
Physical model($n[\beta, \gamma]$)

Figure 1. Diagram of a population balance neural network (PBNN) model for solving the forward and inverse problems of particle aggregation and breakage systems.

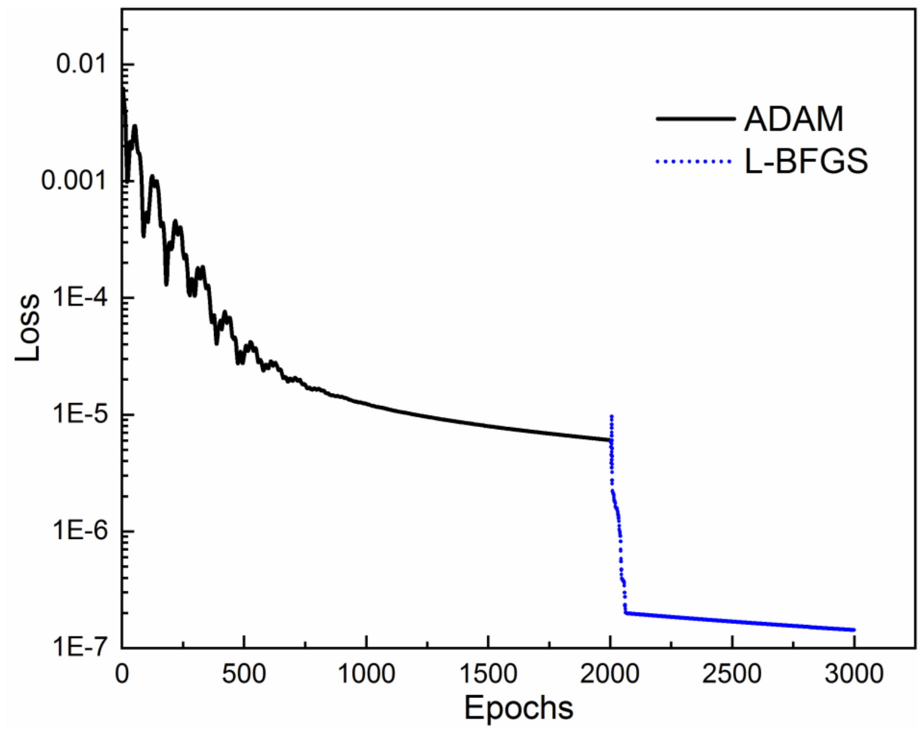


Figure 2 The optimization process of PBNN for solving the aggregation case

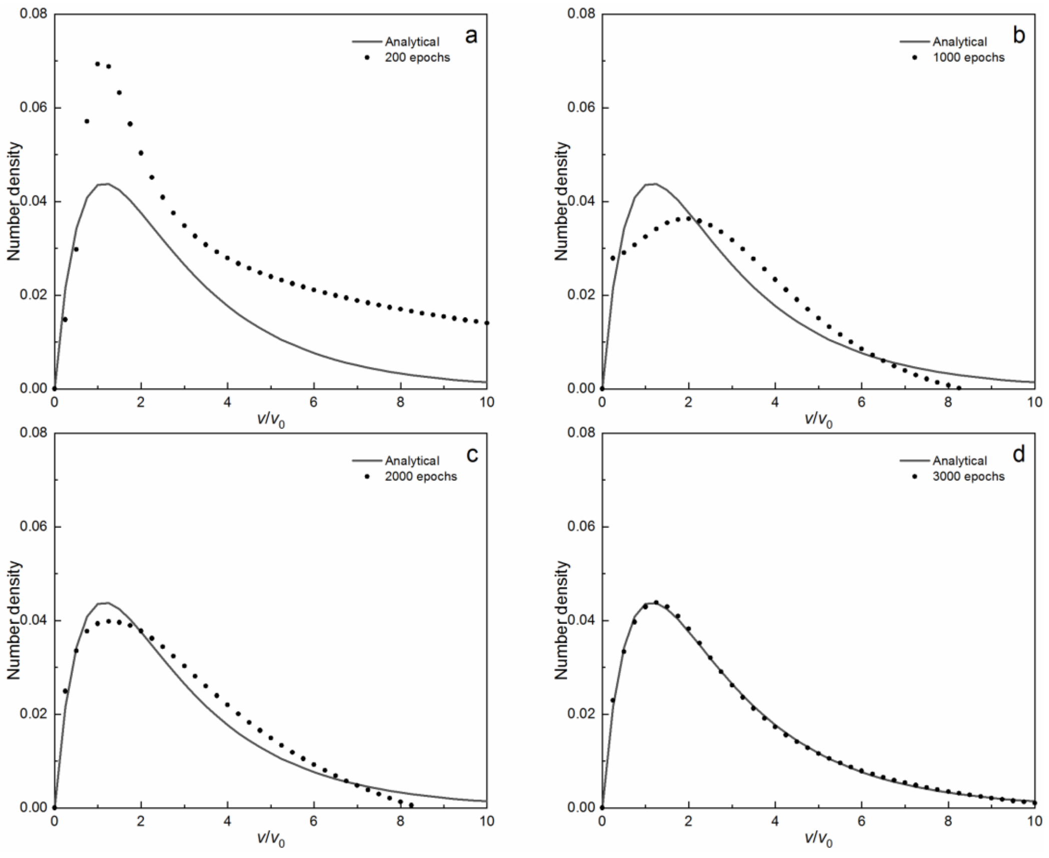


Figure 3 The prediction of the particle number density during the optimization process. (a) 200 epochs (b) 1000 epochs (c) 2000 epochs (d) 3000 epochs

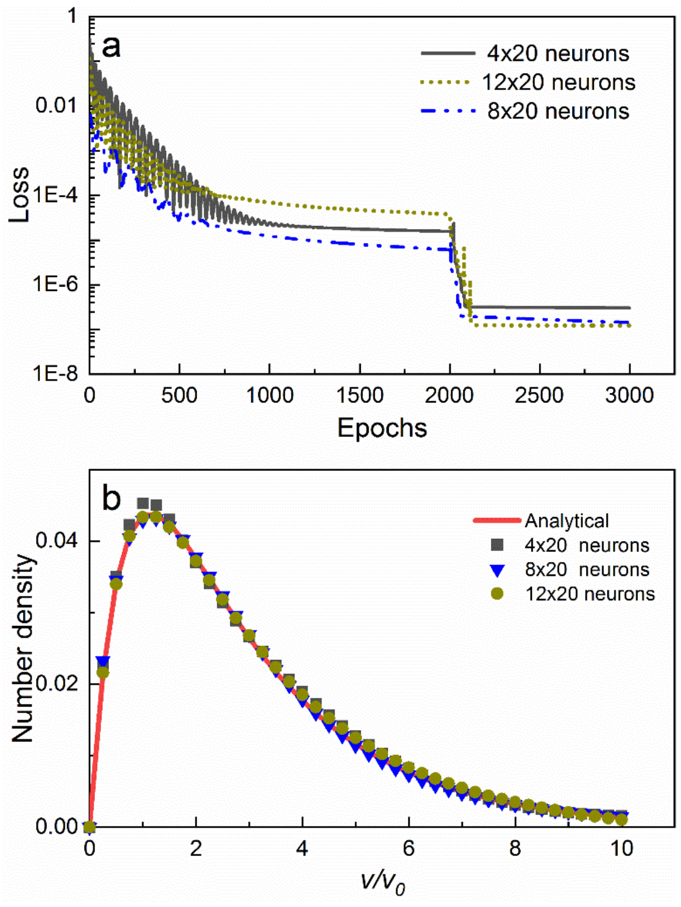
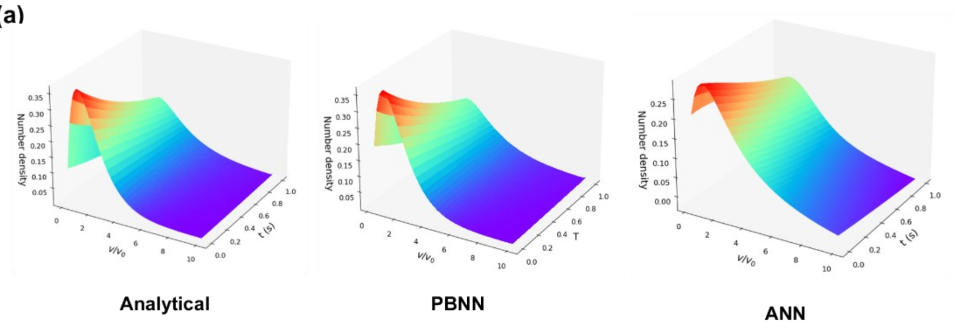


Figure 4 Effect of the depth of the neural network architecture on the loss and prediction results in the aggregation case. (a) Loss function during optimization (b) Comparisons between the predictions and the analytical solution of final size number density.

(a)



(b)

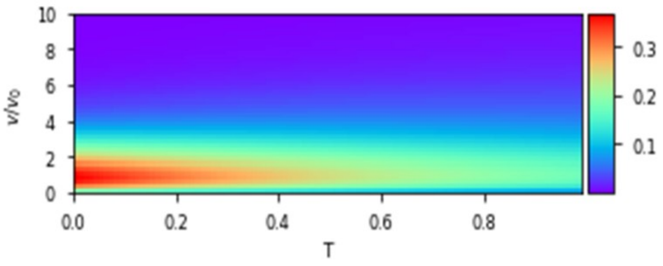


Figure 5. Solution of the number density function in the aggregation case

(a) surface plot from Analytical solution, PBNN and ANN (b) contour plot from PBNN

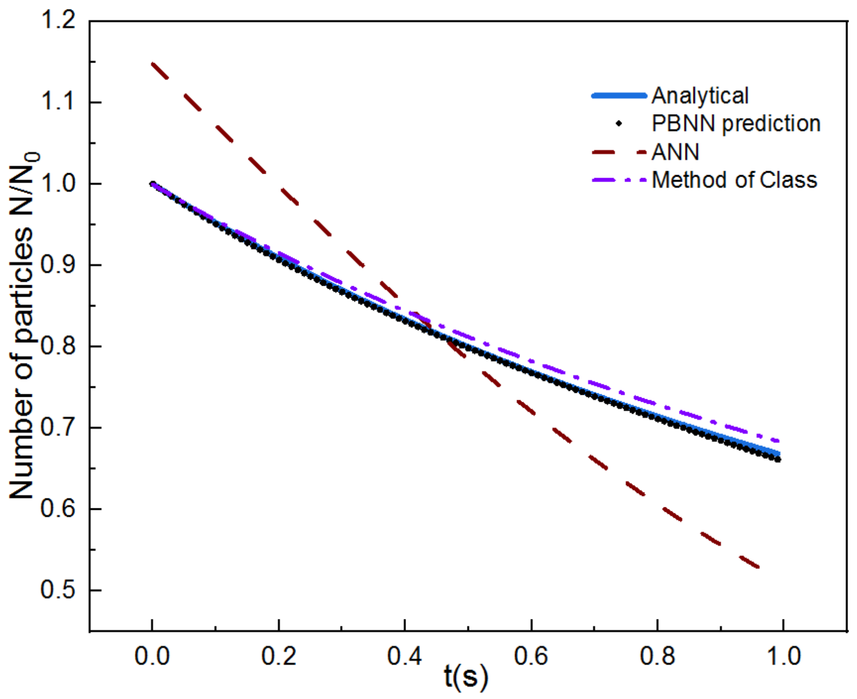


Figure 6 Comparison between the prediction of total number of particles evolution and the analytical solution in aggregation case.

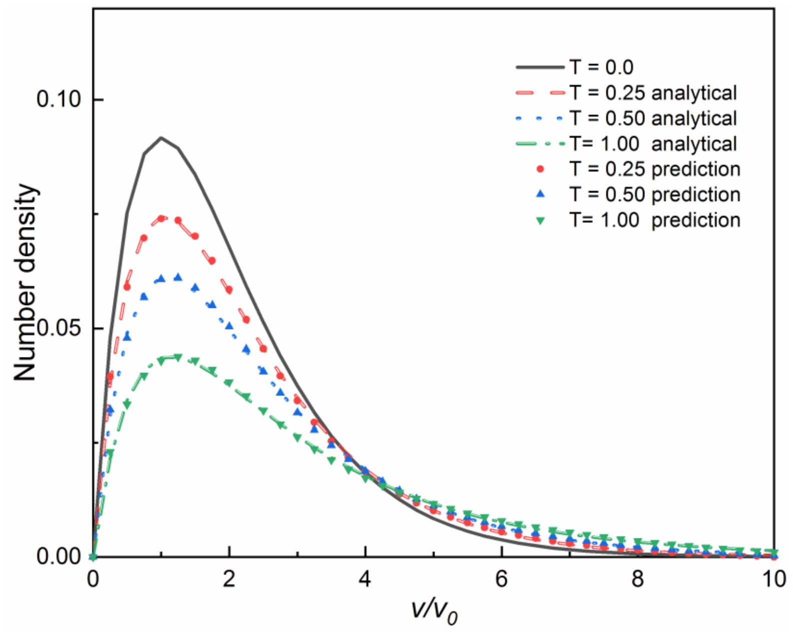


Figure 7. Comparison of the predictions and the analytical solutions of the time evolution of particle number density in aggregation case

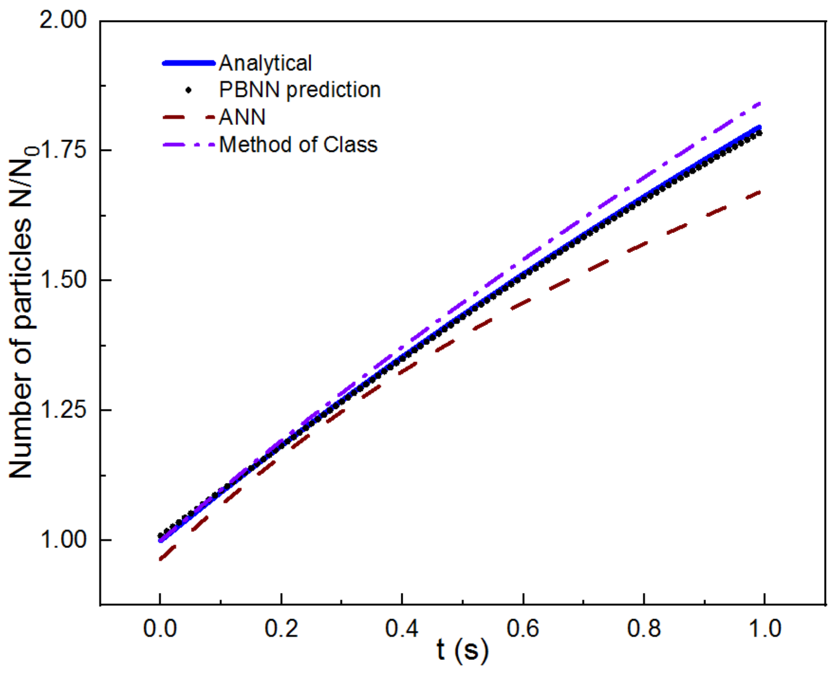


Figure 8 Comparison between the prediction of total number of particles evolution and the analytical solution in the breakage case.

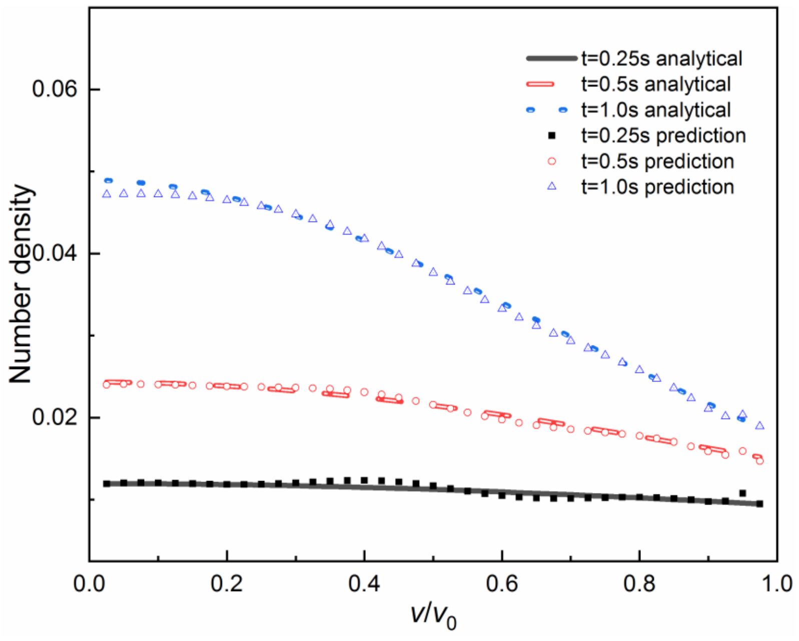


Figure 9 Comparison of the predictions and the analytical solutions of the time evolution of particle number density in the breakage case

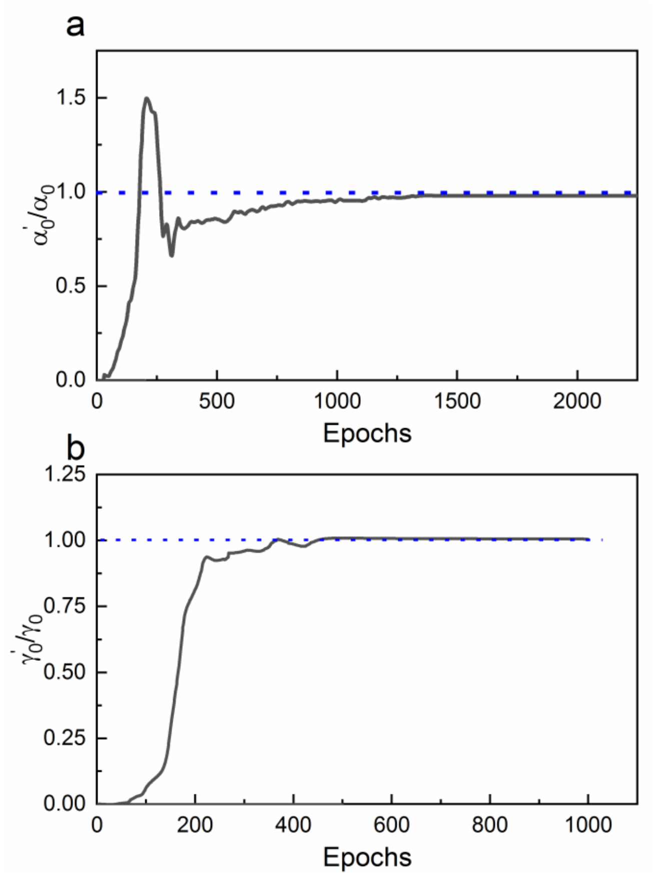


Figure 10 Kernel parameter estimations by PBNN (a) aggregation rate constant estimation (b) breakage rate constant estimation

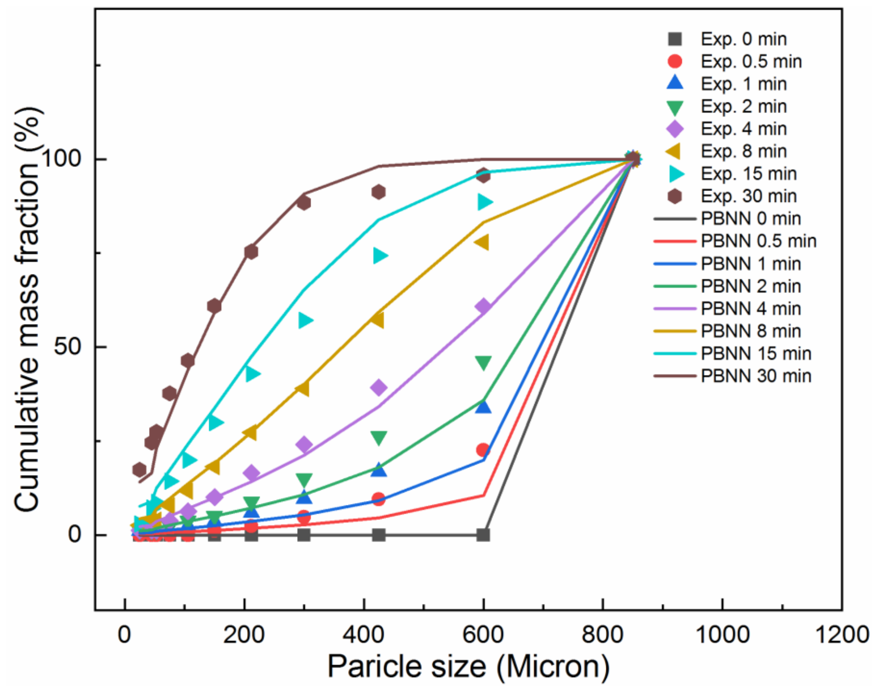


Figure 11 Measured and predicted particle size distributions in a ball milling laboratory test

Table captions

Table 1 Relative L2 errors of the predictions with varying layer widths and depths of the neural network

Table 2 Kernel parameter estimation with noise training data (true value =1)

Table 3 Kernel parameter estimation in the ball milling laboratory test

Table 1 Relative L2 errors of the predictions with varying layer widths and depths of the neural network

Depth / Width	4	8	12
10	4.92×10^{-2}	2.34×10^{-2}	2.45×10^{-2}
20	1.96×10^{-2}	1.26×10^{-2}	1.24×10^{-2}
40	1.67×10^{-2}	1.21×10^{-2}	1.22×10^{-2}

Table 2 Kernel parameter estimation with noise training data (true value =1)

Noise	$\sigma = 0$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.07$	$\sigma = 0.1$
PBNN Estimation	1.000	0.998	0.992	0.972	0.94

Table 3 Kernel parameter estimation in the ball milling laboratory test

variable	a	ε	Λ	μ	$\emptyset$	φ	ω
PBNN Estimation	0.27	1.14	7.95	7.26	0.57	4.88	0.95