Advances in Continuous and Discrete Models

RESEARCH Open Access

Data-driven machine learning approach based on physics-informed neural network for population balance model

Ishtiaq Ali[1*]

*Correspondence:
iamirzada@kfu.edu.sa
[1]Department of Mathematics and Statistics, College of Science, King Faisal University, P.O. Box 400, Al-Ahsa, 31982, Saudi Arabia

Abstract

Population balance models (PBM) are a fundamental tool in the field of process engineering and materials science for understanding and forecasting particulate system dynamics. These models are essential for the design and optimization of a wide range of industrial processes because they capture the evolution of particle size distribution in response to different phenomena such as nucleation, growth, aggregation, and breakage. The complex and non-linear nature of these models makes it difficult to find the analytical solution and even sometimes the approximate solution. This paper introduces a novel machine learning approach based on physics-informed neural network (PINN) for the approximate solution of PBM. Our strategy utilizes the use of a customized neural network framework that has been developed and trained on data generated from simulated PBM by using a finite difference method for the differential operators to identify the underlying dynamics and patterns controlling the evolution of particle distribution. In general, PINN uses automatic differentiation for the computation of differential operators, which is based on the chain rule and needs several matrix operations for computing, which reduces the processing efficiency during the training. The PINN approach provides a flexible, effective and highly adaptable solution framework by utilizing neural networks to approximate the solution and defining the loss function as the sum of the differential equation's residuals at specific random or regular points inside the domain, as well as the residuals of the initial and boundary conditions. It is shown numerically that this approach does not need a diffusion term for a stable solution, which is often needed in most numerical methods for solving these models. For further validation, we compare the results with the exact solution and find them with a very good agreement with each other.

Keywords: Population balance model; Data-driven machine learning approach; Physics-informed neural network; Numerical simulations

1 Introduction

Population balances (PB) are crucial for industrial crystallization since they simulate the formation and features of crystals within a crystallizer, eventually generating the final solid product. These balances offer a mathematical framework for monitoring changes,

especially particle characteristics, such as size, shape, and density, through the crystallization process, as well as for tracking the preservation and transformation of particle numbers. PB are structured analogous to mass and energy balances, highlighting the nonconservation of specific particle characteristics and aiming to figure out the relative contributions of multiple factors to these fluctuations. These PB can combine other variables from the particle phase space, a phrase referring to the multidimensional array of particle attributes, to describe other significant aspects, although they typically focus on tracking particle numbers based on linear size. This procedure uses multiple separate variables to model the changing properties of particles, allowing for a more detailed analysis that involves modeling changes in particle form in addition to size. In order to study the dynamics of particle populations, Hulburt and Katz developed the population equilibrium modeling technique [1]. Since then, it has been widely used in a wide range of scientific fields, such as engineering, physics, chemistry, biology, and meteorology. This methodology is especially common in the field of crystallization, where it is the primary method used to describe the size, shape, and other properties of particles. The approach has been thoroughly investigated through a combination of theoretical and empirical research. To enable real-time optimization and control, process engineers need a model that can predict outcomes much faster than the process occurs. Currently, in the field of particulate science, controlling characteristics like particle size distribution (PSD) and form is essential because of their significant influence on the final product and further processing steps.

The PBE is recognized as a key tool for modeling the dynamics of crystallization processes and mathematically can be expressed as a partial differential equation, which may include integral components to accommodate for secondary processes such as agglomeration and particle breaking. These PBE make a substantial contribution to practical applications by making it possible to optimize industrial processes and enhance the efficiency of dynamic particle system operations. For processes like crystallization, drug manufacturing, and material engineering, it estimates particle size distributions under different circumstances, supporting strong design strategies. Solving PBE accurately to obtain the crystal size distribution (CSD) is crucial. Numerous numerical studies have explored PBE, and each has advantages and disadvantages of its own [2–4]. Among these, the most common numerical approaches to solving PBE are the method of moments, the method of characteristics, the weight residuals, the orthogonal collocation method, the Monte Carlo method, finite-difference schemes, and discrete population balances, respectively [5–8]. Ramkrishna's review offers valuable insights into these methods [9]. The method of moments estimates CSD through its moments, facing limitations under complex conditions. The method of characteristics simplifies PDEs to ODEs along characteristic lines, suitable for simpler physics, but falters with complexity. Weight residuals and orthogonal collocation are based on basis functions, with their effectiveness contingent on the choice of these functions. The Monte Carlo method, though versatile, incurs high computational costs. Finite-difference and finite-volume methods, adaptable to complex scenarios, demand extensive grid points for precision. For other notable methods for approximate solutions of these models, we refer the reader to [10–23].

Gunawan et al.'s development of the high-resolution finite-volume methods (HR-FVM) is notable for its capacity to compute the particle size distribution (PSD) with accuracy and to solve PB equations involving agglomeration and breakage processes numerically, all without the problems of numerical diffusion and dispersion [24]. From a modeling

perspective, this makes HR-FVM the perfect option for optimization and control applications, while it's important to keep in mind that the computational requirements can be very high. A thorough discussion of these techniques may be found in the work by Qamar et al. in [25]. To improve the high-resolution finite volume methods' (HR-FVM) computing efficiency, a number of initiatives have been implemented. To reduce mesh size, Qamar et al. devised an adaptive mesh approach [26]. A master/slave CPU cluster architecture was proposed by Gunawan and his team as a parallel computing strategy [27]. The Fast HR-FVM, created by Majumder and colleagues, uses coordinate transformation to speed up simulations without sacrificing accuracy. Prakash and associates utilized the Matlab Parallel Computing Toolbox and Distributed Computing Server to execute HR-FVM algorithms in parallel on CPUs [28]. Recent advancements have introduced highly accurate numerical methods for solving the PBE, such as the HR-van method with a flux limiter, the Lattice–Boltzmann method, the weighted essentially non-oscillatory (WENO) method, and spectral methods. Notably, spectral methods, particularly the spectral collocation method, stand out for their efficiency and lower grid point requirements compared to other approaches [29–31].

Given the PBE's nature as a convection–reaction equation with pronounced hyperbolic characteristics, these features pose challenges to stability in numerical solutions. To avoid this, a diffusive term is introduced, enhancing stability at the expense of some deviation from the exact solution. This adjustment, while affecting accuracy, ensures the reliability of the numerical outcomes. Population balances are a typical engineering process used in drug crystal formation, pollutant production in fires, and the growth of microbial and cell populations [32]. The aim of this work is to introduce a novel and stable technique that does not need the addition or deletion of diffusion terms to be stable. To this end, we apply a data-driven technique based on PINN for the approximate solution of PBE. The main advantage of the PINN method over other traditional approaches is that it produces a prediction function over the entire computational domain rather than a discrete solution on the meshes as in mesh-based methods; they are applicable to high-dimensional problems; and they are mesh-free, avoiding the problems of mesh generation on complex regions and the construction of high-precision discrete schemes on meshes of poor geometric quality. The primary goal of this work is to optimize the loss function made up of PBM residuals by training neural networks to approximate the equation's solution.

The general form of the Population Balance Equation (PBE) can be expressed as:

$$\frac{\partial \mathcal{F}(\mathcal{L},t)}{\partial t} + \frac{\partial}{\partial \mathcal{L}}\left[\mathcal{F}(\mathcal{L},t)\mathcal{G}(\mathcal{L},t)\right] + \epsilon \frac{\partial^2 \mathcal{F}}{\partial \mathcal{L}^2}(\mathcal{L},t) = \mathcal{H}(\mathcal{L},t,\mathcal{F}). \tag{1}$$

The function $\mathcal{F}(\mathcal{L},t)$ represent the population density, which describes how the distribution of a given characteristic $\mathcal{L}$ (which may be age, size, or some other measurable property) changes with time t. The expression $\frac{\partial \mathcal{F}}{\partial t}$ indicates the rate of variation of this density with respect to time, giving insight on the population's dynamic behavior. A growth or velocity function is included via the inclusion of $\mathcal{G}(\mathcal{L},t)$ in the equation, specifically in the term $\frac{\partial}{\partial \mathcal{L}}\left[\mathcal{F}(\mathcal{L},t)\mathcal{G}(\mathcal{L},t)\right]$, which accounts for changes in the characteristic $\mathcal{L}$ brought about by growth or other processes and $\epsilon > 0$ is a positive constant. For the purpose of simulating events like cell growth, particle aggregation, or the dissemination of features within a population, this section of the equation is crucial. In addition, when taking into

consideration the variability and distribution of features, the diffusion term $\frac{\partial^2 \mathcal{F}}{\partial \mathcal{L}^2}$ represents the dispersion or spread in the characteristic $\mathcal{L}$ throughout the population. It also takes into consideration the dispersion or fluctuation of particle properties, including size, within a system, help in modeling the inherent unpredictability in particle systems, and depicts dispersion effects resulting from stochastic or random processes. Furthermore, especially in situations where convection predominates, it stabilizes numerical solutions by reducing abrupt gradients or discontinuities. In order to preserve computational stability and appropriately represent physical dispersion, this term is essential. In the final analysis, depending on the particular application, the function $\mathcal{H}(\mathcal{L}, t, \mathcal{F})$ acts as a source or sink term, such as external variables that influence the population density, for example, migration, births and deaths, or chemical reactions. This expression makes it possible to include complex connections between outside variables, which significantly improves the model's adaptability to a wide range of scientific investigations.

2 Physics-informed neural network technique for PBE

Artificial neural networks have been utilized to solve issues in many different application domains over the past few decades, including computer vision and natural language processing, among many others. The scientific machine learning (ML) community has recently seen the emergence of another extremely promising application: the use of artificial neural networks, commonly known as PINN, to solve partial differential equations (PDE). Since its initial introduction in the landmark study in [33], PINN have been gaining significant importance in both academia and industry. In PINN, in addition to training the model with data-driven supervised neural networks, the model is taught physics equations to promote consistency with the system's understood physics. They have the advantage of being able to reliably extrapolate beyond the existing data and being data-driven in learning a model, while still ensuring consistency with physics. Because of this, PINN can produce stronger models with fewer data points. It combines the flexibility of artificial neural networks with the benefits of classical numerical methods to provide a potential tool for solving PDE. By adding the physical aspects of a PDE into the training process, PINN are designed to learn the solution to a PDE. This method can greatly increase training speed and accuracy by integrating known data points with a physics-based loss function.

PINN has shown great promise as a viable substitute for more conventional numerical techniques like Finite Element Methods (FEMs). It uses the physical principles inherent in the PDE to direct the learning process. This makes it possible to include known data points to the training process, which can improve the solution's accuracy and effectiveness even further. The capacity of PINN to integrate the physical characteristics of the PDE into the learning process is its main benefit. By doing this, the network is guaranteed to understand the fundamental physical linkages and restrictions, which results in more precise and dependable solutions. PINN is an invaluable tool for resolving a variety of issues in a variety of disciplines, including engineering, physics, and finance. They can also handle complex and nonlinear PDE. To futher explore PINN, we refer the reader to [34–40].

To apply PINN to the given model in equation (1), a neural network architecture that can approximate the solution was designed, as shown in Fig. 1.

This network takes inputs of spatial and temporal coordinates and outputs an approximation of the function $\mathcal{F}(\mathcal{L}, t)$, which represents the exact population density. Let us denote the neural network as a function $\mathcal{F}(x, t; \theta)$, where x and t are the spatial and temporal

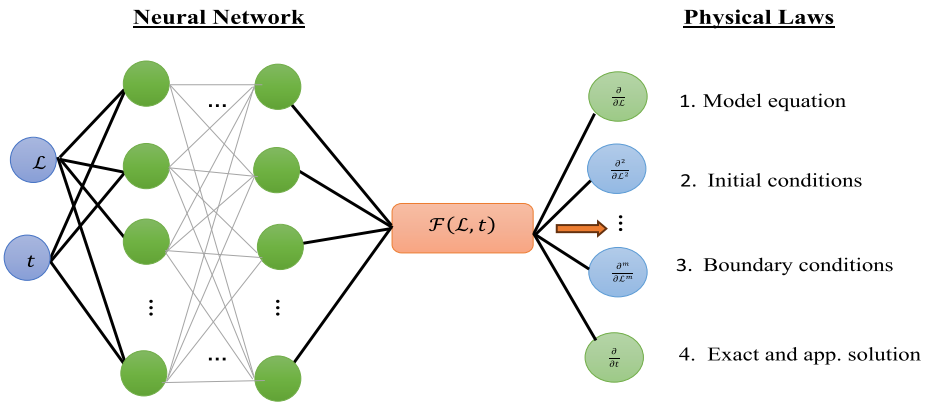

Figure 1 Schematic of the proposed physics-informed neural network for solving the model equation (1)

coordinates, respectively, and θ represents the network approximation of $\mathcal{F}(\mathcal{L},t)$. The network approximate the solution to the model equation of the form:

$$\mathcal{F}(x,t;\theta) \approx \mathcal{F}(\mathcal{L},t). \tag{2}$$

Define the loss function to ensure the network satisfies the physical laws, boundary and initial conditions, and observational data of the form:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + \lambda_1\mathcal{L}_{\text{BC}}(\theta) + \lambda_2\mathcal{L}_{\text{IC}}(\theta) + \lambda_3\mathcal{L}_{\text{data}}(\theta). \tag{3}$$

Here the term $\mathcal{L}_{\text{PDE}}$ enforces the satisfaction of the PBE:

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_{\text{PDE}}}\sum_{i=1}^{N_{\text{PDE}}}\left|\frac{\partial\mathcal{F}_\theta}{\partial t} + \frac{\partial}{\partial\mathcal{L}}\big(\mathcal{G}(\mathcal{L},t)\mathcal{F}_\theta\big) + \epsilon\frac{\partial^2\mathcal{F}_\theta}{\partial\mathcal{L}^2} - \mathcal{H}(\mathcal{L},t,\mathcal{F}_\theta)\right|^2, \tag{4}$$

The term $\mathcal{L}_{\text{BC}}$ enforces boundary conditions and is given by:

$$\mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N_{\text{BC}}}\sum_{i=1}^{N_{\text{BC}}}\left|\mathcal{F}_\theta(\mathcal{L}_{\text{BC}},t_{\text{BC}}) - g(\mathcal{L}_{\text{BC}},t_{\text{BC}})\right|^2, \tag{5}$$

where $g(\mathcal{L}_{\text{BC}},t_{\text{BC}})$ specifies the boundary values, while the term $\mathcal{L}_{\text{IC}}$ enforces adherence to the initial condition and is defined by:

$$\mathcal{L}_{\text{IC}}(\theta) = \frac{1}{N_{\text{IC}}}\sum_{i=1}^{N_{\text{IC}}}\left|\mathcal{F}_\theta(\mathcal{L}_{\text{IC}},0) - h(\mathcal{L}_{\text{IC}})\right|^2, \tag{6}$$

where $h(\mathcal{L}_{\text{IC}})$ is the initial distribution. The last term $\mathcal{L}_{\text{data}}$ matches observational or simulated data and is given by:

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}}\left|\mathcal{F}_\theta(\mathcal{L}_{\text{data}},t_{\text{data}}) - \mathcal{F}_{\text{FDM}}(\mathcal{L}_{\text{data}},t_{\text{data}})\right|^2. \tag{7}$$

Here $\mathcal{F}_{\text{FDM}}$ is the approximate solution using FDM. For our PDE, we discretize the domain into a grid of points over $\mathcal{L}$ and t, and approximate the derivatives as follows:

For time derivative approximation using a forward difference scheme, the time derivative can be approximated by:

$$\frac{\partial \mathcal{F}}{\partial t} \approx \frac{\mathcal{F}(\mathcal{L}, t + \Delta t) - \mathcal{F}(\mathcal{L}, t)}{\Delta t}. \tag{8}$$

Similarly, the first spatial derivative, using a central difference scheme, is approximated by:

$$\frac{\partial}{\partial \mathcal{L}} [\mathcal{F}(\mathcal{L}, t)\mathcal{G}(\mathcal{L}, t)] \approx \frac{\mathcal{F}(\mathcal{L} + \Delta\mathcal{L}, t)\mathcal{G}(\mathcal{L} + \Delta\mathcal{L}, t) - \mathcal{F}(\mathcal{L} - \Delta\mathcal{L}, t)\mathcal{G}(\mathcal{L} - \Delta\mathcal{L}, t)}{2\Delta\mathcal{L}}. \tag{9}$$

The second spatial derivative is given by:

$$\epsilon \frac{\partial^2 \mathcal{F}}{\partial \mathcal{L}^2}(\mathcal{L}, t) \approx \epsilon \left(\frac{\mathcal{F}(\mathcal{L} + \Delta\mathcal{L}, t) - 2\mathcal{F}(\mathcal{L}, t) + \mathcal{F}(\mathcal{L} - \Delta\mathcal{L}, t)}{(\Delta\mathcal{L})^2} \right). \tag{10}$$

The fully discretized form of the model equation (1) using FDM is given by:

$$\frac{F_i^{n+1} - F_i^n}{\Delta t} + \frac{\left[F_i^n G_i^n + F_i^{n+1} G_i^{n+1}\right]}{2\Delta\mathcal{L}} + \epsilon \frac{F_{i+1} - 2F_i + F_{i-1}}{\Delta\mathcal{L}^2} - \mathcal{H}_i^n = 0. \tag{11}$$

The terms λ_1, λ_2 and λ_3 are the weights for BC, IC, and data loss, respectively. The simple approach is to choose $\lambda_i = 1$, for all i, that is, all loss components are treated equally initially, and adjustments can be made based on validation errors using normalization, if needed. The training dataset is generated using a FDM to solve the PBE over a grid of spatial and temporal coordinates, incorporating predefined boundary and initial conditions. The validation dataset is derived similarly but uses different resolutions or parameter variations to evaluate the generalization of the model during training. Testing datasets are constructed to assess the model's performance on unseen scenarios, often involving altered parameters, broader domains, or challenging edge cases. By comparing PINN predictions against exact solutions or independent numerical methods, we ensure the model's robustness and accuracy beyond the training conditions.

3 Numerical examples

In this section, we present two numerical examples for demonstrating the strength and effectiveness of PINN. From these examples, we are able to see directly how PINN use the underlying physical rules to direct the learning process, ensuring both accuracy and generalizability in different situation. This practical strategy demonstrates how the models can solve complicated differential equations and accurately anticipate physical processes, frequently outperforming traditional numerical methods in terms of speed and scalability. Furthermore, by comparing the results of these examples with the exact solution, it becomes clear how efficient and flexible PINN are to a range of difficult-to-solve problems. Both below examples are chosen for [29].

3.1 Example 1

Consider the classical example of crystal growth in a batch process under specific assumptions. We consider a system where the crystal growth rate $\mathcal{G}$ is constant and independent

of crystal size $\mathcal{L}$. Processes like aggregation, nucleation, and breakage are ignored, simplifying the system to totally focus on crystal growth. The crystal growth rate is given as $\mathcal{G}$ = 1.0 mm/s, and the CSD follows a Gaussian distribution defined by:

$$\mathcal{F}(\mathcal{L},0) = \frac{10^{10}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\mathcal{L}-\mu)^2}{2\sigma^2}\right), \tag{12}$$

This Gaussian distribution is chosen because it accurately reflects symmetric changes around a mean size, which is common in crystallization processes. It is simpler to compute and evaluate. The parameters μ and σ are the mean and standard deviation of the crystal size distribution, respectively, with $\mu = 20\mu m$ and $\sigma = 3$. The simulation parameters include a time step dt = 0.001 s, a parameter ϵ = 0.005, and the end time of the simulation t_{end} = 60 s. Given that the crystal growth rate $\mathcal{G}$ is constant, the exact solution for the CSD at any time t can be found by shifting the initial distribution by $\mathcal{G}t$, that is:

$$\mathcal{F}(\mathcal{L},t) = \mathcal{F}(\mathcal{L}-\mathcal{G}t,0). \tag{13}$$

This equation states that the entire distribution of crystal sizes shifts uniformly as time progresses, with no change in the shape of the distribution, due to the constant growth rate applied equally to all crystals. To illustrate this, we simulate the process and plot the initial CSD and the CSD at t = 30 s and at t = 60 s to show the shift in the crystal growth process as shown in Fig. 2 and Fig. 3 for N = 200 and N = 600, respectively. A complete horizontal shift without any change in shape or spread is shown by comparison with the exact solution for growth that is independent of size. The exact solution and PINN are compared to indicate how well the PINN trained on FDM data follows this ideal behavior. This comparison may reveal small differences, such as a tiny numerical diffusion or a larger spread, inherited from the FDM training data, as shown in Figs. 2–3. These images illustrate the

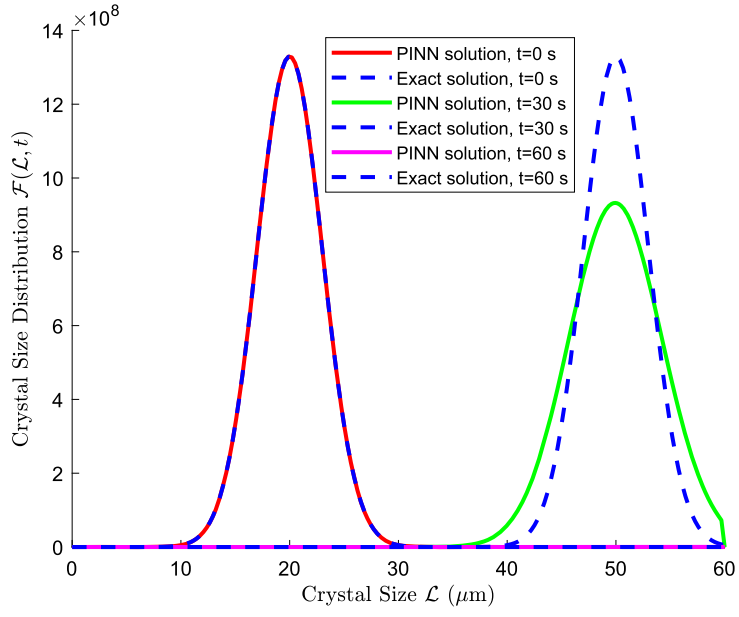

Figure 2 *Example 1:* Crystal size distribution over different time with N = 200 grid points

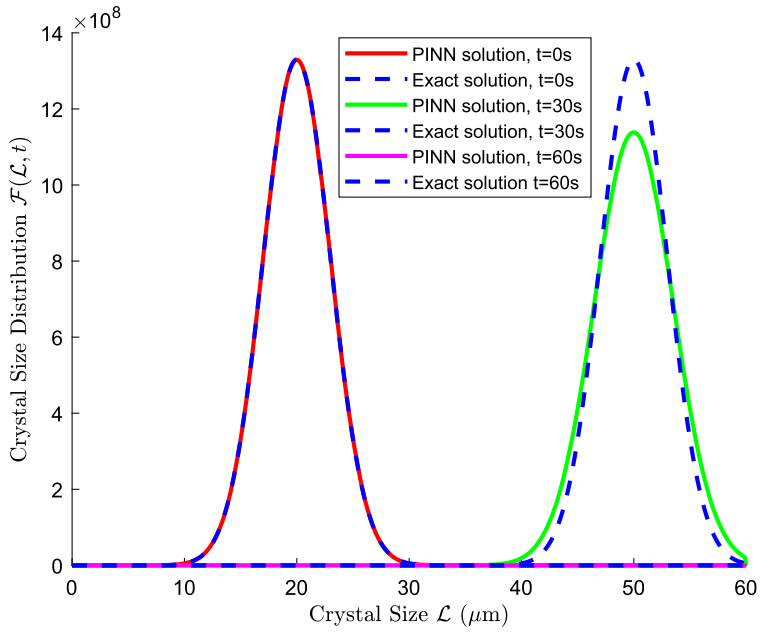

Figure 3 *Example 1:* Crystal size distribution over different time with $N = 600$ grid points

ability of PINNs to handle hyperbolic dynamics typical of convection-dominated systems in size-independent growth situations, where the crystal growth rate $\mathcal{G}$ is constant. As can be seen from the model's stability in these figures, dispersion and oversmoothing—two prominent problems in numerical simulations of hyperbolic equations—are not captured by the neural network when simulating the linear advection of crystal sizes. How well the PINN handles edge cases is shown by closely examining figures near borders (e.g., small or big crystal sizes). Given how difficult it can be to enforce boundary criteria in neural networks, effective border behavior free of oscillations or instabilities indicates that the network has learned the required boundary conditions implicitly. A reliable and safe crystallization process depends on precisely modeling the whole range of crystal sizes, which is ensured by good boundary behavior. This is especially true when scaling up from lab to industrial scales.

We observe that while increasing the grid points, one can get closer and closer to the exact solution. For the clear understanding the comparison between exact and PINN solution, we also provide subplot at different time, as shown in Fig. 4. In order to validate our PINN solution, we also train our neural network using the exact initial conditions and the result is shown in Fig. 5. The distribution of errors between expected and actual values is visually represented by a neural network training error histogram, which aids in identifying underfitting or overfitting tendencies during the model training process. This is shown in Fig. 6. How well a neural network model generalizes from its training data to new data depends on how well it performs during training. This is usually evaluated utilizing metrics like accuracy and precision. In case of Example 1, the best validation performance is epoch 10, as shown in Fig. 7. When a neural network is trained for regression tasks, it learns to predict continuous output values. The mean squared error or mean absolute error between the predicted and actual values is often used to evaluate the network's performance, as shown in Fig. 8. Training states of neural networks, including weights and

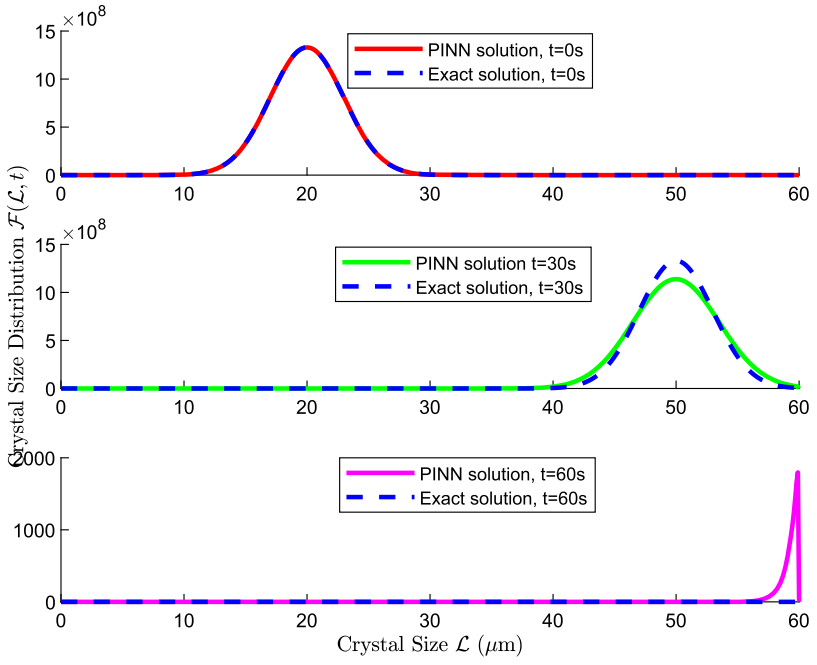

Figure 4 *Example 1:* Subplot of crystal size distribution over different time with $N = 600$ grid points

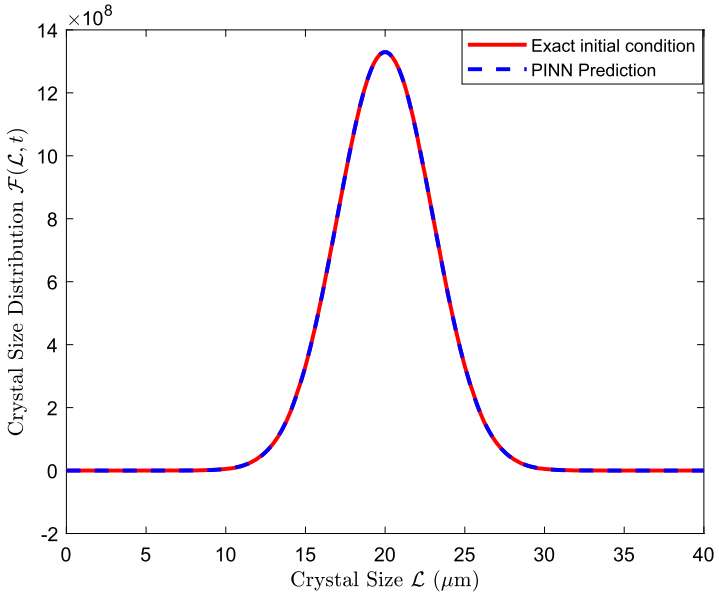

Figure 5 *Example 1:* Comparison of exact initial condition and predicted by PINN

biases, change as a result of techniques like optimization algorithms and backpropagation, which seek to effectively decrease the loss function, as shown in Fig. 9.

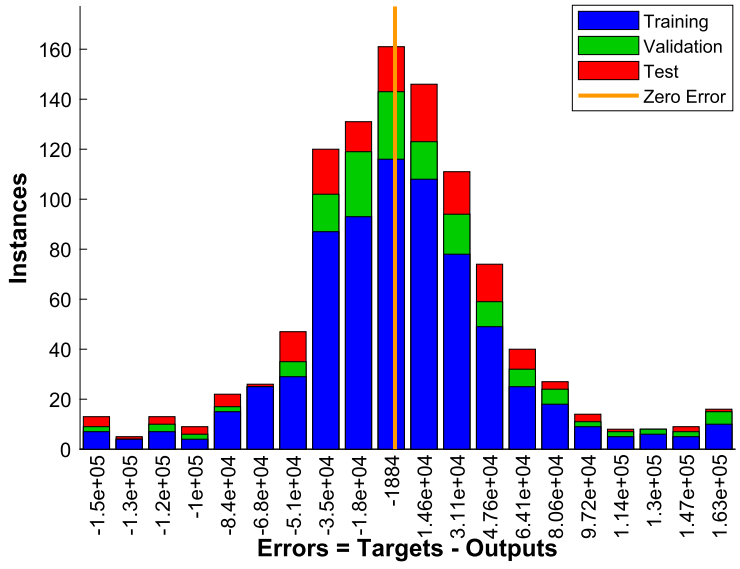

Figure 6 *Example 1:* Neural network training error histogram

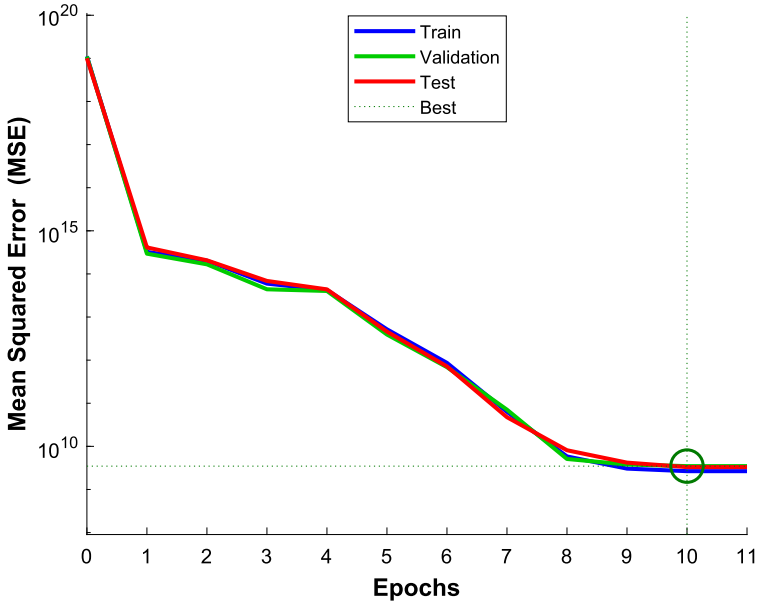

Figure 7 *Example 1:* Neural network training performance

3.2 Example 2

In the second example, crystallization takes place in a batch process, where the crystal size $\mathcal{L}$ determines the crystal growth rate $\mathcal{G}$. The crystal growth rate $\mathcal{G}(\mathcal{L},t) = \mathcal{G}_0\mathcal{L}$ is a linear function of crytal size. Again, $\mathcal{H}(\mathcal{L},t,\mathcal{F}) = 0$ as in Example 1 and the following equation is satified by CSD:

$$\mathcal{F}(\mathcal{L},0) = \frac{N_0}{\mathcal{L}} \exp\left(-\frac{\mathcal{L}}{\bar{\mathcal{L}}}\right),$$

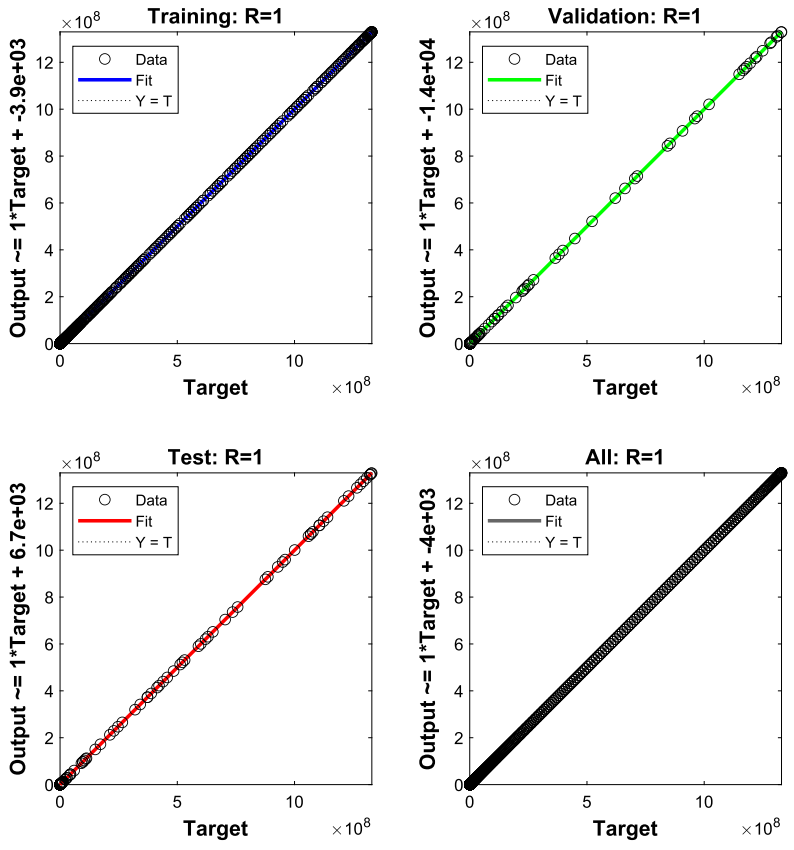

Figure 8 *Example 1:* Neural network training regression

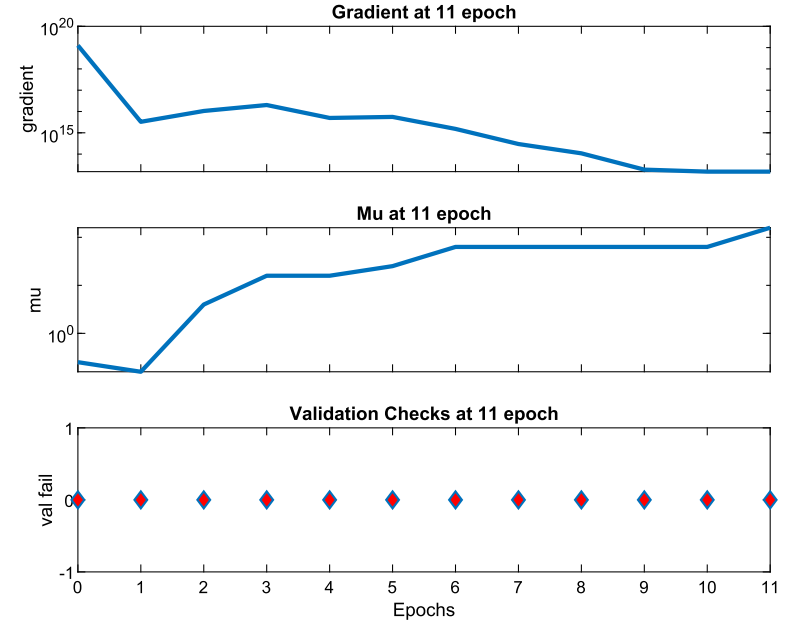

Figure 9 *Example 1:* Neural network training states

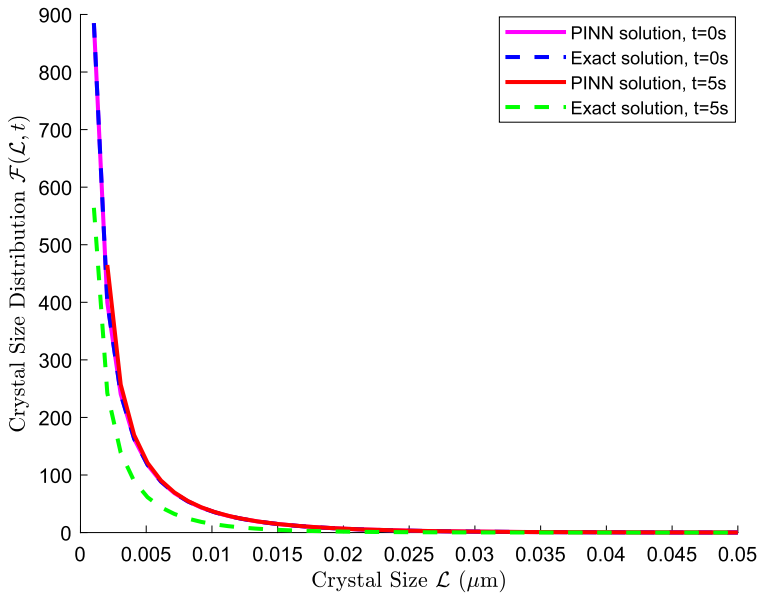

Figure 10 *Example 2:* Crystal size distribution over different time for $N = 50$ grid points

where N_0 and $\bar{\mathcal{L}}$ are constants. The exact solution to this equation is given by:

$$\mathcal{F}(\mathcal{L},t) = \frac{N_0}{\mathcal{L}} \exp\left(-\frac{\mathcal{L}}{\bar{\mathcal{L}}e^{-\mathcal{G}_0 t}}\right) \exp(-\mathcal{G}_0 t),$$

with parameters values, $\bar{\mathcal{L}} = 0.01\,\mu m^3$, $N_0 = 1$, $\mathcal{G}_0 = 0.1\,(\mu m^3)/s$, $\epsilon = 0.002$, and $dt = 0.00001\,s$. This example deals with size-dependent growth in a batch process, where the difficulty rises as the size-dependent growth rate changes. Over time, the size distribution becomes distorted and broader due to the size-dependent growth rate, which makes larger particles grow more quickly. In addition to making numerical approaches more difficult, this non-linear behavior creates steeper gradients and shifts in the distribution, proving that the PINN can properly capture intricate, size-dependent dynamics. With the help of FDM data, PINN is trained to reproduce how the FDM handles these fluctuations; this is often demonstrated by a shift and distortion (broadening or stretching) of the CSD. Larger crystals develop more quickly, as shown in Fig. 10–11. This also shows that evolution of the distribution and any numerical distortions common to FDM, such as small oscillations or inconsistencies at locations where the growth rate varies quickly. Comparing the exact solution, a more theoretically ideal transformation of the CSD with a clear representation of the non-linear growth effects can be seen in the exact solution for size-dependent growth. The neural network's ability to capture the complex dynamics trained on FDM data is demonstrated by comparing this to the PINN predictions, which highlights deviations such as over-smoothing and underestimating steep gradients. In order to see the comparison between exact and PINN predicted solution, a more clear view is given in the subplot consist of Fig. 12. The validation of the PINN performace is evaluated against the exact initials condition and the output is given in the form of Fig. 13.

These results show how accurate predictions are for size-dependent growth, where the growth rate $\mathcal{G}(\mathcal{L})$ varies with crystal size, which further confirm that the PINN can man-

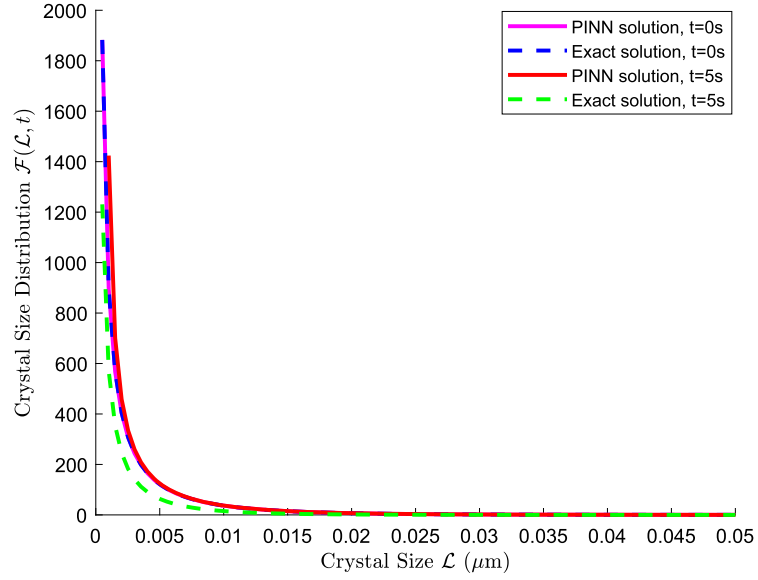

Figure 11 *Example 2:* Crystal size distribution over different time for $N = 100$ grid points

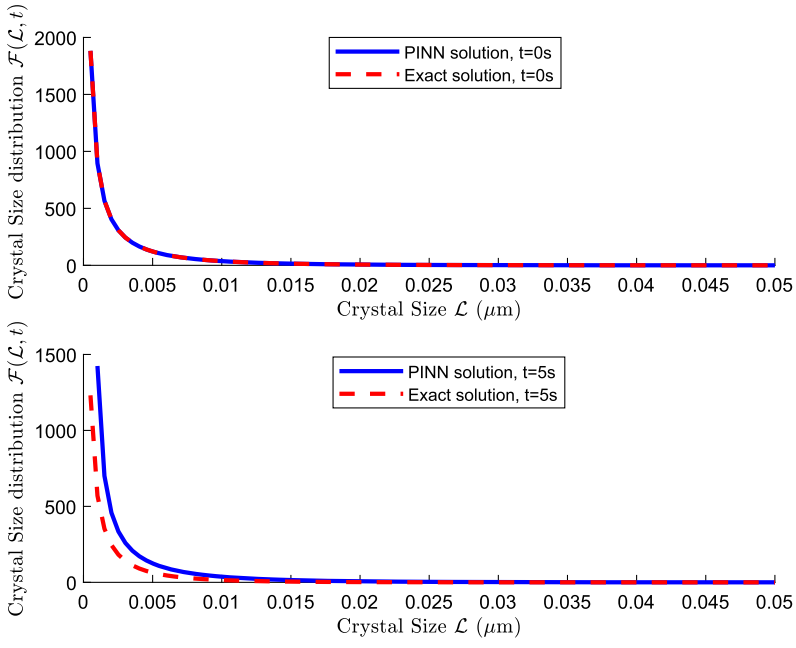

Figure 12 *Example 2:* subplot of crystal size distribution over different time for $N = 100$ grid points

age non-linear dependencies and interactions within the system. This demonstrates the network's capacity to discover intricate linkages and patterns from the data and ingrained physical rules rather than from explicit programming. The robustness of PINNs against parameter uncertainty can be demonstrated by looking at how different factors (such as growth rates or initial circumstances) affect the model's output. Figures 10–13 also illustrate that long-term forecasts over protracted times or until steady state is reached can

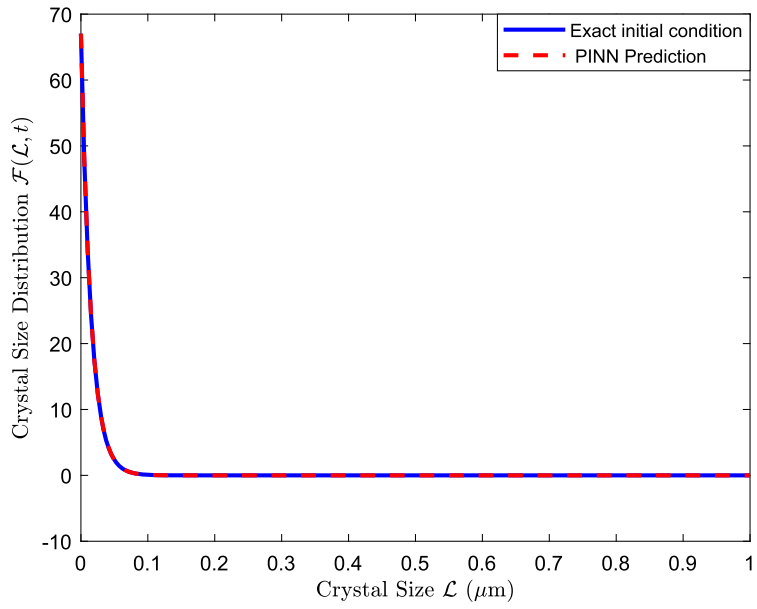

Figure 13 *Example 2:* Comparison of exact initial condition and predicted by PINN

show how the PINN can continue to be accurate and stable over time without straying from or drifting from physical expectations. Understanding the efficacy of neural networks in practical applications requires evaluating their training performance. Performance metrics reveal how well the network predicts new data. The architecture of the network, which includes the quantity and size of layers as well as the activation functions employed, has an impact on training performance as well. Furthermore, the quantity and caliber of training data are critical factors in determining how well a neural network learns. One can use sophisticated methods such as cross-validation to assess how resilient the network is to variations in data subsets. The best performance result is at epoch 7, which is shown in Fig. 14. An error histogram for diagnosing model behavior, especially when it comes to determining bias and variance problems is given in Fig. 15. Figure 16 indicates the training states for Example 2 examine how well the model reflects the data trends. Techniques such as feature scaling, correct network parameter setup, and regularization to avoid overfitting improve the performance of the regression. Plotting anticipated vs. actual values is a common method of evaluating regression models in order to visually examine how well the model reflects the data trends is shown in Fig. 17.

4 Results and discussion

First, we approximate the solutions of model equation (1) over a discrete grid using the FDM. By solving the equation step-by-step through the domain, FDM produces data points that show the state of the system at different moments in time. Despite being an approximation with underlying numerical faults, such as discretization error, this data offers a comprehensive dataset reflecting the numerical features of the FDM as well as the underlying physics. The neural network learn from this training dataset, which comprises both the ideal physical rules and the real-world numerical behaviors frequently observed in classical simulations. The training dataset is then used as training data for PINN. This dual feature of the data aids PINN in creating a solid model sensitive to both computational

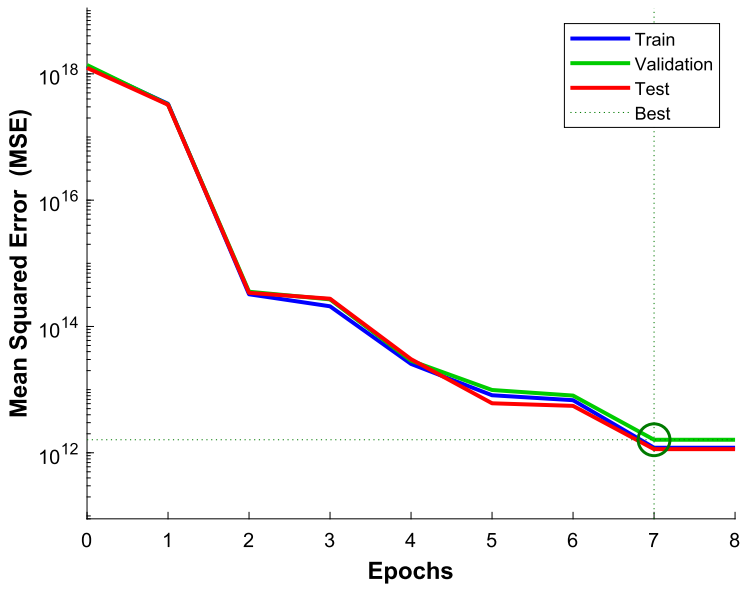

Figure 14 *Example 2:* Neural network training performance

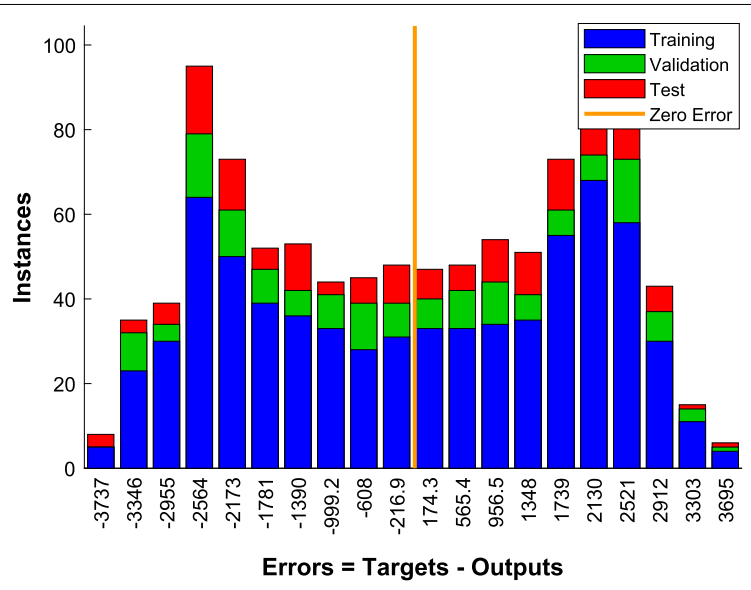

Figure 15 *Example 2:* Neural network training error histogram

and theoretical sensitivities. These PINN are designed so that their architecture immediately benefits from an understanding of the governing differential equations by including the differential equation into the loss function. Consequently, the network is trained to satisfy the physical principles governing the process in addition to fitting the training data (from FDM). The neural network weights must be adjusted during the training phase to guarantee that the predictions are consistent with the physical equations and to reduce the discrepancy between the network's predictions and the FDM data. In addition to im-

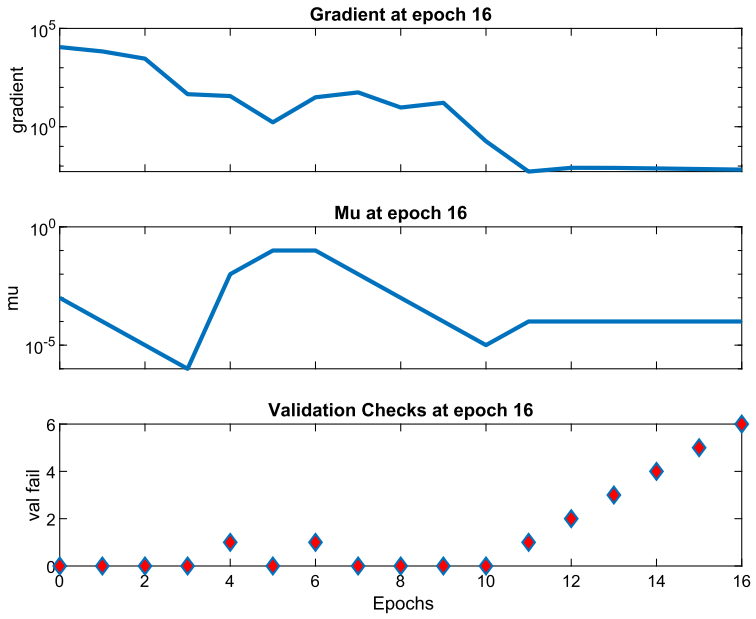

Figure 16 *Example 2:* Neural network training states

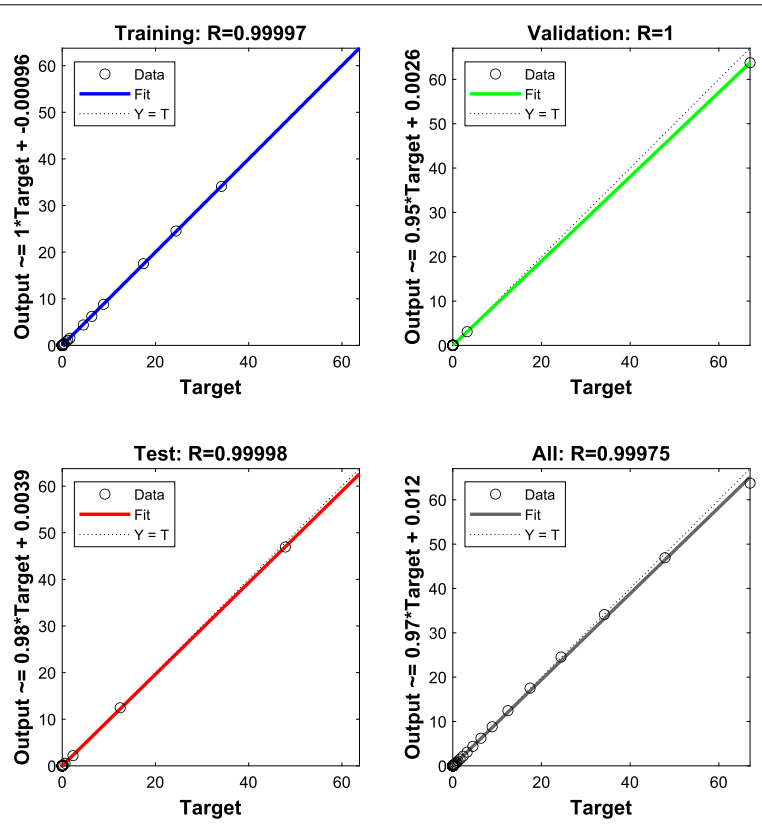

Figure 17 *Example 2:* Neural network training regression

proving generalization, this dual purpose also help PINN address some of the numerical errors in the FDM data.

5 Conclusion

In this study, the FDM is employed in order to generate training data, which is then used to train a PINN and predict how the particle distribution will change over time. To verify their accuracy and efficiency, the outcomes from the PINN are compared with those from the exact solution. It has been demonstrated that more complex and optimal crystallization techniques, which may dynamically modify parameters in response to changes in the crystal size distribution, require accurate modeling of size-dependent growth. In industrial applications where process conditions may alter or be unpredictable, this robustness is very valuable. Models that function well under a variety of circumstances lower the possibility of process failures and boost overall effectiveness. For continuous crystallization techniques to work and for the development of strategies that predict how process modifications will impact the end product over time, long-term stability is essential. In conclusion, the integration of numerical simulation and machine learning to PBMs through the use of finite difference methods to produce training data for PINNs enables the creation of reliable models that can produce precise predictions under a variety of circumstances. If the model configuration, training procedure, and data quality are carefully considered, it presents an achievable path toward addressing complex systems efficiently and dynamically.

Acknowledgements
This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Grant No. KFU250109].

Author contributions
The author read and approved the final manuscript.

Data availability
Not applicable.

Declarations

Competing interests
The author declares no competing interests.

Received: 31 October 2024 Accepted: 6 January 2025 Published online: 22 January 2025

References

1. Hulburt, H.M., Katz, S.: Some problems in particle technology: a statistical mechanical formulation. Chem. Eng. Sci. **19**, 555–574 (1964)
2. Rasmuson, Å.C.: Crystallization process analysis by population balance modeling. In: Myerson, A.S., Erdemir, D., Lee, A.Y. (eds.) Handbook of Industrial Crystallization, pp. 172–196. Cambridge University Press, Cambridge (2019)
3. Lin, F., Yang, Y., Yang, X.: Exact solutions of population balance equation with aggregation, nucleation, growth and breakage processes, using scaling group analysis. Symmetry **16**, 65 (2024)
4. Le Minh, T., Phan Thanh, T., Nguyen Thi Hong, N., Phan Minh, V.: A simple population balance model for crystallization of L-lactide in a mixture of n-hexane and tetrahydrofuran. Crystals **12**, 221 (2022)
5. Inguva, P.K., Schickel, K.C., Braatz, R.D.: Efficient numerical schemes for population balance models. Comput. Chem. Eng. **162**, 107808 (2022)
6. Su, J., Le, W., Gu, Z., Chen, C.: Local fixed pivot quadrature method of moments for solution of population balance equation. Processes **6**, 209 (2018)
7. Wang, K., Yu, S., Peng, W.: A new method for solving population balance equations using a radial basis function network. Aerosol Sci. Technol. **54**, 644–655 (2020)
8. Röhl, S., Hohl, L., Stock, S., Zhan, M., Kopf, T., von Klitzing, R., Kraume, M.: Application of population balance models in particle-stabilized dispersions. Nanomaterials **13**, 698 (2023)
9. Ramkrishna, D., Singh, M.R.: Population balance modeling: current status and future prospects. Annu. Rev. Chem. Biomol. Eng. **5**, 123–146 (2014)

10. Immanuel, C.D., Doyle, F.J. III: Computationally efficient solution of population balance models incorporating nucleation, growth and coagulation: application to emulsion polymerization. Chem. Eng. Sci. **58**, 3681–3698 (2003)
11. Tiong, S.I.X., Ahamed, F., Sitaraman, H., Leong, S.L., Ho, Y.K.: Modeling simultaneous particle shrinkage, dissolution and breakage using the modified moving grid technique. Powder Technol. **421**, 118439 (2023)
12. Huang, Y., Zhu, H., Yang, X., Tu, J., Jiang, S.: Population balance modeling for air–water bubbly flow in a vertical U-bend. J. Comput. Multiph. Flows **10**, 170–177 (2018)
13. Li, D., Li, Z., Gao, Z.: Quadrature-based moment methods for the population balance equation: an algorithm review. Chin. J. Chem. Eng. **27**, 483–500 (2019)
14. Qamar, S., Angelov, I., Elsner, M.P., Ashfaq, A., Seidel-Morgenstern, A., Warnecke, G.: Numerical approximations of a population balance model for coupled batch preferential crystallizers. Appl. Numer. Math. **59**(3–4), 739–753 (2009)
15. Qamar, S., Ashfaq, A., Angelov, I., Elsner, M.P., Warnecke, G., Seidel-Morgenstern, A.: Numerical solutions of population balance models in preferential crystallization. Chem. Eng. Sci. **63**, 1342–1352 (2008)
16. Solsvik, J., Jakobsen, H.A.: Evaluation of weighted residual methods for the solution of a population balance model describing bubbly flows: the least-squares, Galerkin, tau, and orthogonal collocation methods. Ind. Eng. Chem. Res. **52**, 15988–16013 (2013)
17. Sewerin, F., Rigopoulos, S.: An explicit adaptive grid approach for the numerical solution of the population balance equation. Chem. Eng. Sci. **168**, 250–270 (2017)
18. Shah, B.H., Ramkrishna, D., Borwanker, J.D.: Simulation of particulate systems using the concept of the interval of quiescence. AIChE J. **23**, 897–904 (1977)
19. Gooch, J.R.P., Hounslow, M.J.: Monte Carlo simulation of size-enlargement mechanisms in crystallization. AIChE J. **42**, 1864–1874 (1996)
20. Kumar, S., Ramkrishna, D.: On the solution of population balance equations by discretization—I. A fixed pivot technique. Chem. Eng. Sci. **51**, 1311–1332 (1996)
21. Ma, D.L., Tafti, D.K., Braatz, R.D.: Ind. Eng. Chem. Res. **41**, 6217–6223 (2002)
22. Majumder, A., Kariwala, V., Ansumali, S., Rajendran, A.: Lattice Boltzmann method for multi-dimensional population balance models in crystallization. Chem. Eng. Sci. **70**, 121–134 (2012)
23. Ruan, C.L., Liang, K.F., Chang, X.J., Zhang, L.: Weighted essentially nonoscillatory method for two-dimensional population balance equations in crystallization. Math. Probl. Eng. **2013**, 125128 (2013)
24. Gunawan, R., Fusman, I., Braatz, R.D.: High resolution algorithms for multidimensional population balance equations. AIChE J. **50**, 2738–2749 (2004)
25. Qamar, S., Elsner, M.P., Angelov, I.A., Warnecke, G., Seidel-Morgenstern, A.: A comparative study of high resolution schemes for solving population balances in crystallization. Comput. Chem. Eng. **30**, 1119–1131 (2006)
26. Qamar, S., Warnecke, G.: Solving population balance equations for two-component aggregation by a finite volume scheme. Chem. Eng. Sci. **62**, 679–693 (2007)
27. Gunawan, R., Fusman, I., Braatz, R.D.: Parallel high-resolution finite volume simulation of particulate processes. AIChE J. **54**, 1449–1458 (2008)
28. Prakash, A.V., Chaudhury, A., Barrasso, D., Ramachandran, R.: Simulation of population balance model-based particulate processes via parallel and distributed computing. Chem. Eng. Res. Des. **91**, 1259–1271 (2013)
29. Ruan, C.: Chebyshev spectral collocation method for population balance equation in crystallization. Mathematics **7**, 317 (2019)
30. Mantzaris, N.V., Daoutidis, P., Srienc, F.: Numerical solution of multi-variable cell population balance models. II. Spectral methods. Comput. Chem. Eng. **25**, 1441–1462 (2001)
31. Buck, A., Klaunick, G., Kumar, J., Peglow, M., Tsotsas, E.: Numerical simulation of particulate processes for control and estimation by spectral methods. AIChE J. **58**, 2309–2319 (2012)
32. Ramkrishna, D.: Population Balances: Theory and Applications to Particulate Systems in Engineering. Academic Press, San Diego (2000)
33. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
34. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics informed machine learning. Nat. Rev. Phys. **3**, 422–440 (2021)
35. Zhu, Y., Zabaras, N., Koutsourelakis, P.S., Perdikaris, P.: Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. J. Comput. Phys. **394**, 56–81 (2019)
36. Faroughi, S.A., Pawar, N.M., Fernandes, C., Raissi, M., Das, S., Kalantari, N.K., Mahjour, K.: Physics-guided, physics-informed, and physics-encoded neural networks and operators in scientific computing: fluid and solid mechanics. ASME J. Comput. Inf. Sci. Eng. **24**, 040802 (2024)
37. Mattey, R., Ghosh, S.: A novel sequential method to train physics informed neural networks for Allen–Cahn and Cahn–Hilliard equations. Comput. Methods Appl. Mech. Eng. **390**, 114474 (2022)
38. Cuomo, S., Di Cola, V.S., Giampaolo, F., et al.: Scientific machine learning through physics–informed neural networks: where we are and what's next. J. Sci. Comput. **92**, 88 (2022)
39. Raissi, M., Karniadakis, G.E.: Hidden physics models: machine learning of nonlinear partial differential equations. J. Comput. Phys. **357**, 125–141 (2018)
40. Yang, L., Meng, X., Karniadakis, G.E.: B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. J. Comput. Phys. **425**, 109913 (2021)

Publisher's Note