Neuroevolving monotonic PINNs for particle breakage analysis

Abhishek Gupta
School of Mechanical Sciences
Indian Institute of Technology Goa
Goa, India
abhishekgupta@iitgoa.ac.in

B. K. Mishra
School of Chemical and Materials Science
Indian Institute of Technology Goa
Goa, India
bk@iitgoa.ac.in

***Abstract*—Artificial intelligence has the potential to positively impact various facets of today's minerals industry. This paper is a first showcase of *physics-informed neural networks* (PINNs) for simulating the breakage of large particles into smaller fragments in a grinding mill, which is undeniably one of the most energy-intensive phases in the processing of mineral ores. The breakage is governed by a population balance integro-differential equation, whose accurate solution is crucial to support precise planning and control of processes to meet product specifications and sustainability goals. However, solutions derived using existing PINN algorithms, while computationally efficient, are found to violate a basic mathematical property of *monotonicity* of the modelled cumulative distribution function over particle sizes. This renders the solution of little practical use. Guided by the implicit function theorem, we discover that a synergy of existing techniques with *neuroevolutionary algorithms* can lead to the desired creation of monotonic PINNs. Real-world data of a batch grinding mill is used to validate our method, establishing PINNs as a powerful tool for simulating particle breakage dynamics.**

Keywords—Industrial AI, physics-informed neural networks, neuroevolution, particle breakage analysis

I. Introduction

A grinding mill is used for breaking a system of large particles (e.g., mined ores) into smaller fragments by inducing compressive stresses through impact loads. Grinding mills in the minerals industry process 40,000 – 100,000 tons of material per day, consuming such enormous amounts of electrical energy that power costs can be as high as half the total processing costs [1]. Much research attention has therefore been devoted to the accurate modelling and simulation of the grinding phenomenon, with research outcomes expected to assist with precise planning of environmentally sustainable processes. A one-dimensional *population balance equation* (PBE) is widely studied in this regard, and can be posed in continuous form as the following integro-differential equation [2]:

$$\frac{\partial F(x,t)}{\partial t} = \int_x^\infty S(x',t) \cdot B(x,x') \cdot \frac{\partial F(x',t)}{\partial x'} \cdot \mathrm{d}x' \qquad (1)$$

governing the evolution of the cumulative distribution function $F(x,t)$ over particle sizes. $F(x,t)$ represents the fraction of particles in the system by mass whose size is less than x at time t. *By definition,* $F(x,t)$ *must increase monotonically from 0 to 1 for particle sizes between 0 and* x_{max}, where x_{max} is the maximum size in the initial feed distribution $F(x,t=0)$. $S(x',t)$ is the selection function that specifies the rate at which particles of size x' break at time t. Eq. (1) is rendered nonlinear when the selection function depends on the time dependent state of the entire particulate system [3]. Finally, $B(x,x')$ is the breakage distribution function which gives the cumulative fraction of particles by mass less than size x that break out of particles of size x', with $x' \geq x$.

While several numerical schemes have been considered for solving Eq. (1), they usually involve some kind of discretization of time or particle size (aka, a *mesh*) that limits the resolution/precision at which particle size distributions can be modelled [4]. Such (coarse) discretization leads to inadequacies in industrial particulate system applications where the evolution of the full distribution, spanning particles that may vary by several orders of magnitude in size, is needed to precisely control the physiochemical and mechanical product properties. It is here that the emerging techniques of *physics-informed neural networks* (PINNs) offer a powerful alternative. PINNs – where governing equations of physical phenomena are embedded in the training objective of the model [5, 6] – serve as a *mesh-free* approach to solve differential equations central to diverse applications in science and engineering. In comparison to conventional deep learning, PINN generated outputs are more likely to be physics-compliant, while simultaneously making predictions of arbitrarily high resolution on arbitrarily fine grids without the need for any model retraining. In the context of grinding, the differentiability of a PINN allows the particle size density function to be derived at any snapshot in time. To the best of our knowledge, this work is among the first to study the applicability of PINNs in the particulate processing industry, making possible the accurate modelling and optimal control of energy-intensive grinding processes governed by PBEs.

II. PINNs for the breakage PBE

A PINN uses a neural network representation $\hat{F}(x,t;\boldsymbol{W},\boldsymbol{B})$ to approximate $F(x,t)$ in Eq. (1) for particles of size $x \in \Omega = [0, x_{max}]$ and time $t \in [0, t_{max}]$. $\boldsymbol{W} = \{W_1, W_2, \ldots, W_L\}$ and $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_L\}$ are the list of weights and biases, respectively, in a depth-L network where $L \geq 2$. If the i-th layer has n_i neurons then $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and $\boldsymbol{b}_i \in \mathbb{R}^{n_i}$; $n_0 = 2$ since there are only two inputs x and t in the considered PBE; $n_L = 1$ since the cumulative distribution function is the single output of interest. The output layer of a PINN is typically

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00181

linear with zero bias and the identity activation function, while the hidden layers employ infinitely differentiable neural activations such as the tanh, sigmoid, or Fourier functions [7].

Given this representation, the network parameters can be tuned by minimizing the following loss function:

$$\text{argmin}_{\boldsymbol{W},\boldsymbol{B}}\, \mathcal{L}_{PINN} = \mathcal{L}_R + \lambda_{IC}\mathcal{L}_{IC} + \lambda_{BC_1}\mathcal{L}_{BC_1} + \lambda_{BC_2}\mathcal{L}_{BC_2}, \quad (2)$$

where,

$$\mathcal{L}_R = \left\| \frac{\partial F(x,t)}{\partial t} - \int_x^{\infty} S(x',t)\cdot B(x,x')\cdot \frac{\partial F(x',t)}{\partial x'}\cdot \mathrm{d}x' \right\|^2_{L^2(\Omega\times[0,t_{max}])}, \quad (3a)$$

$$\mathcal{L}_{IC} = \left\| F(x,t=0) - \hat{F}(x,0;\boldsymbol{W},\boldsymbol{B}) \right\|^2_{L^2(\Omega)}, \quad (3b)$$

$$\mathcal{L}_{BC_1} = \left\| 0 - \hat{F}(0,t;\boldsymbol{W},\boldsymbol{B}) \right\|^2_{L^2([0,t_{max}])}, \quad (3c)$$

$$\mathcal{L}_{BC_2} = \left\| 1 - \hat{F}(x_{max},t;\boldsymbol{W},\boldsymbol{B}) \right\|^2_{L^2([0,t_{max}])}. \quad (3d)$$

Eq. (3a) sums the squared *residual* of the PBE over the simulation domain, with the integral term substituted by its Riemann quadrature formula. Eq. (3b) refers to the satisfaction of the initial feed distribution. Eqs. (3c) and (3d) refer to the satisfaction of the extreme values of a cumulative distribution function. Note that although the loss is formulated over a continuous domain, in practice, the terms are computed using only a finite set of collocation points distributed in the domain. λ_{IC}, λ_{BC_1}, and λ_{BC_2} are hyperparameters that control the trade-off between the different terms in the PINN loss function.

Examinations of the loss functions of PINNs have shown that they are significantly more rugged than those encountered in conventional data-driven deep learning [8]. This makes PINNs difficult to train by gradient descent. Additionally, Proposition 1 in [7] emphasized the point that wide neural nets at initialization tend towards flat output functions with vanishing derivatives. That is, $\hat{F}(x,t;\boldsymbol{W},\boldsymbol{B})$ is a constant when $\boldsymbol{W}$ and $\boldsymbol{B}$ are initialized by the Xavier method common in the PINNs literature. Such flatness is especially problematic while finding solutions to PBEs. Since $\frac{\partial \hat{F}(x,t;\boldsymbol{W},\boldsymbol{B})}{\partial t} = \frac{\partial \hat{F}(x,t;\boldsymbol{W},\boldsymbol{B})}{\partial x} = 0$ everywhere, substitution into Eq. (1) gives zero residual error. This trivially and deceptively minimizes Eq. (3a), while still being far away from the true solution that must also minimize Eqs. (3b) – (3d). The fact that the PBE residual is already zero at initialization indicates that the objective of minimizing the residual error supplies no information whatsoever for a purely gradient-based model training.

A. Depth-2 Randomized PINNs

One way to bypass the challenge of gradient-based training of PINNs for PBEs is to take a gradient-free route. The theories of randomized neural nets [9] offer one such pathway.

Building on recent results that a shallow but wide network suffices for physics simulation [10], we consider *randomized PINNs* (rPINNs) of depth-2 with a list of weights $\boldsymbol{W} = \{W_1, W_2\}$. The bias $\boldsymbol{B}$ is left out of our implementation. Entries of W_1 are preset and fixed to random values sampled i.i.d. from some user-defined interval $[-r, r]$. Weights of the output layer, on the other hand, are derived by closed-form *least squares* (LSQ) computation, hugely speeding up model training by circumventing the long training times that result from iterative gradient-based optimization. Details of the LSQ computation procedure applicable to both linear and nonlinear differential equations are available in [11]. Since the same steps apply to PBEs as well, they are not repeated in this short paper.

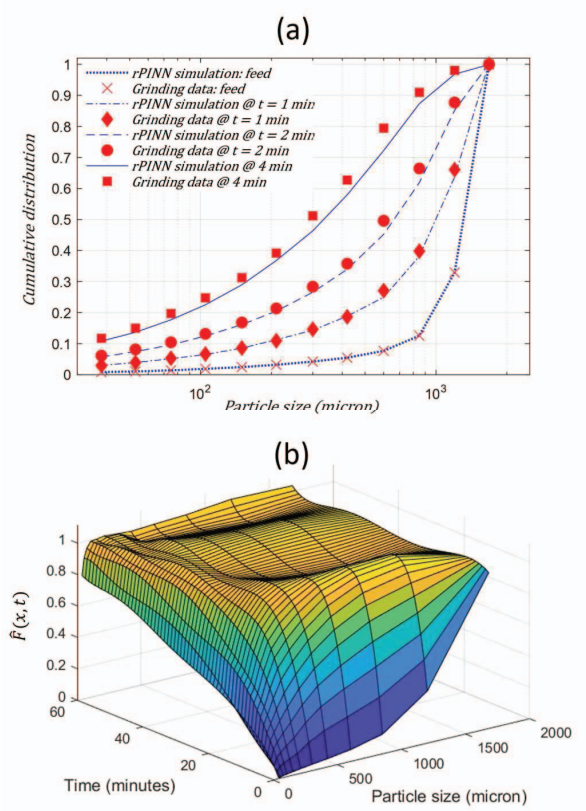

Fig. 1. (a) Depiction of the close agreement between the short time interval rPINN simulation and the real-world batch grinding data. (b) Surface plot of $\hat{F}(x,t)$ reveals severe monotonicity violations in long-time breakage simulation.

B. Applying rPINNs to Real-world Grinding Data

We employ the rPINN to solve a discrete version of Eq. (1) in the form of a system of ordinary differential equations. This formulation is sometimes encountered in real-world grinding processes where particles are categorized into discrete classes. Real data of particle size distribution was collected in [12]. The data, discrete system of equations, and discrete forms of the selection and breakage distribution functions are openly accessible in [3]. *The PINN method is unique in its ability to produce smooth, differentiable predictions of arbitrarily high resolution in particle size and time despite only having access to discrete data. Its differentiability allows density estimates to be readily derived from the modelled cumulative distribution function, setting PINNs apart from other numerical methods.*

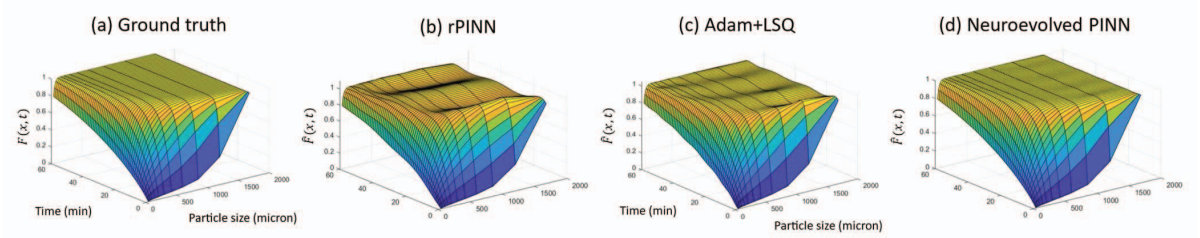

Fig. 2. (a) Surface plot of the ground truth solution to the system of equations governing the real-world batch grinding process. (b) rPINN simulation result whose pronounced waviness points to monotonicity violations of the modelled cumulative distribution function $\hat{F}(x,t)$. (c) Output of the Adam+LSQ optimization procedure showcases some reduction in waviness. (d) Neuroevolved PINN solution that not only matches well with the ground truth but also shows no apparent signs of monotonicity violation. The rPINN, Adam+LSQ and the neuroevolved PINN make use of the same LSQ computation procedure to determine the best-fit output layer weights.

Based on a parameter sweep, we configure the rPINN with 100 neurons, tanh neural activation, $r = 0.01$, $\lambda_{IC} = 100$, and $\lambda_{BC_1} = \lambda_{BC_2} = 1$. Fig. 1a shows close agreement between the resulting simulations and the short time interval batch grinding data (for $t_{max} = 4$ min) that's provided in [3]. However, when the total simulated time is prolonged to $t_{max} = 60$ min, for long-time breakage simulation, then monotonicity violations appear in the predicted cumulative distribution function. The violation is highlighted by the pronounced waviness of the surface plot of $\hat{F}(x,t)$ depicted in Fig. 1b. The prediction in Fig. 1b is therefore of little practical use, pointing to the need for more powerful training procedures to achieve monotonic PINNs.

III. Neuroevolution of PINNs for the PBE

Inspection of Eq. (1) confirms that the monotonicity of $F(x,t)$ is strictly implied by the PBE itself. Since minimization of the squared residual of Eq. (1) is a part of the loss function defined in Eq. (2), monotonicity hints are already contained in the loss function by construction. We therefore turn our attention from the loss function to its optimization algorithm.

The basic idea is to retain the fast LSQ computation for determining output layer weights while derandomizing the first layer of the rPINN. To this end, appealing to the classic implicit function theorem [13], we conclude that the LSQ computation provides best-fit output layer weights, denoted as W_2^*, that are a continuously differentiable function of W_1. That is, we may write $W_2^* = \varphi(W_1)$. Importantly, even though derivatives of the implicit function $\varphi(\cdot)$ exist, we may often be unable to write down explicit mathematical expressions for them.

On substituting W_2 with the best-fit $W_2^* = \varphi(W_1)$ in the PINN loss function, let the resulting reduced loss be written as $\mathcal{L}_{PINN}^*\big(W_1, \varphi(W_1)\big)$. This transforms Eq. (2) to:

$$\text{argmin}_{W_1}\, \mathcal{L}_{PINN}^*\big(W_1, \varphi(W_1)\big). \tag{4}$$

We then have,

$$\frac{d\mathcal{L}_{PINN}^*}{dW_1} = \frac{\partial \mathcal{L}_{PINN}^*}{\partial W_2^*} \cdot \left(\frac{dW_2^*}{dW_1}\right) + \frac{\partial \mathcal{L}_{PINN}^*}{\partial W_1}. \tag{5}$$

Since we generally don't have a formula for the derivative of W_2^*, the loss gradient becomes difficult to explicate.

Seeing the difficulty in accessing the true/exact gradients of the reduced loss function, we propose to tackle Eq. (4) by means of *natural evolution strategies* (NES) that make use of a population of candidate solutions to obtain stochastic estimates of the gradient descent direction [14]. Specifically, we let the population of candidates be sampled from a search distribution model $p_\theta(W_1)$. For mathematical convenience, p_θ is usually taken to be a multivariate Gaussian distribution with mean and variance encapsulated by θ. With this probabilistic viewpoint, the NES further transforms the reduced loss of Eq. (4) into the following *expected loss* under the search distribution:

$$J(\theta) = \mathbb{E}_\theta[\mathcal{L}_{PINN}^*(W_1)] = \int \mathcal{L}_{PINN}^*(W_1) \cdot p_\theta(W_1) \cdot \mathrm{d}W_1. \tag{6}$$

The gradient of the expected loss function with respect to θ can be readily estimated using the log-likelihood trick as:

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}_{PINN}^*\big(W_1^{(m)}\big) \cdot \nabla_\theta \log p_\theta\big(W_1^{(m)}\big), \tag{7}$$

where $W_1^{(1)}, W_1^{(2)}, \ldots, W_1^{(M)}$ is a population of M samples drawn from $p_\theta(W_1)$. Each $W_1^{(m)}$ points to a separate PINN solution with output layer $\varphi\big(W_1^{(m)}\big)$ and loss $\mathcal{L}_{PINN}^*\big(W_1^{(m)}\big)$. The gradient estimate in Eq. (7) provides a descent direction in the space of search distributions. In our implementation, we employ the state-of-the-art xNES algorithm [14] to make use of this estimation to gradually update the distribution parameters θ, and hence the population of solutions sampled from it, towards the optimum W_1^*. Alternate gradient-free optimizers like the CMA-ES could also be used [15] with no change to the overall procedure. The corresponding output layer weights are simply $\varphi(W_1^*)$, obtained by LSQ computation.

IV. Numerical Study

We carry out long-time breakage simulation of the discrete version of the PBE. Although real data of the batch grinding process is only available up to 4 min, the governing system of equations under special (time independent) forms of the selection function can be solved by the eigenvalue method to get more ground truth data for comparison. The simulation outcomes of the basic rPINN and the evolved PINNs can therefore be rigorously compared against the ground truth. Neuroevolution was performed with a population size of $M = 20$ for 1000 generations. The reduced loss function

evaluations of population individuals were parallelized on a single machine with a 14-core/20-thread CPU.

Our numerical study also includes an implementation of gradient-based optimization where Adam updates [16] of W_1 are hybridized with LSQ computations for determining W_2^*. Gradient descent is carried out for 1000 iterations. Adam+LSQ hybridization is made possible by approximating the gradient of the reduced PINN loss by dropping the first term on the right-hand side of Eq. (5). Danskin's theorem [17] suggests that this approximation is a descent direction of the reduced loss, and thus serves as a meaningful update direction for the weights W_1.

The comparison of cumulative distribution surface plots is shown in Fig. 2, where we have used the same neural network configuration as in Section II-B. The only difference is that we employ 200 neurons to demonstrate that the monotonicity violations of the rPINN persist even when the number of neurons is doubled. This is manifested in the pronounced waviness of the surface plot in Fig. 2b. The Adam+LSQ procedure showcases reduced waviness in comparison, but is unable to eliminate it; see Fig. 2c. In contrast, neuroevolution consistently leads to the creation of PINNs whose output shows no apparent signs of monotonicity violation; see Fig. 2d. The key enhancement of neuroevolution is that it not only derandomizes the input layer, but also provides a *globally* optimized basis space for subsequent synergy with LSQ computation at the output layer. Adam on the other hand mainly offers local optimization ability. The mean squared errors achieved by the rPINN, the Adam+LSQ algorithm, and the evolved PINN relative to the ground truth are given in Table I. *The neuroevolved solution is found to have almost three orders of magnitude lower error than the rPINN and two orders of magnitude lower error than Adam+LSQ.*

TABLE I

MEAN SQUARED ERRORS OF THE RPINN, ADAM+LSQ, AND THE NEUROEVOLVED PINN RELATIVE TO THE BATCH GRINDING GROUND TRUTH. BEST RESULT IS MARKED IN BOLD.

rPINN	**Adam+LSQ**	**Neuroevolution**
$4.70e-4$	$6.77e-5$	$\mathbf{8.46e-7}$

V. CONCLUSIONS

The experiments performed lead to the conclusion that synergizing neuroevolution (of a subset of the weights of a neural network) with LSQ computation (for determining the remaining weights) forms a powerful PINN optimizer. We apply the method for the first time to the problem of simulating particle breakage in a batch grinding mill, where we model the evolution of the monotonicity constrained cumulative distribution function over particle sizes. The synergized optimization algorithm is able to repeatably find highly accurate, physically/mathematically valid solutions that cannot be reliably found with other existing techniques. This paper thus adds to a growing line of research on neuroevolution as a noteworthy tool in the training of PINNs [8, 15, 18].

A common argument against evolutionary algorithms is the computational cost associated with sampling and evaluating populations of candidate solutions. However, we believe that once monotonicity preserving features have been discovered through neuroevolution, these can be transferred to a new PBE problem without the need to retrain the input layer from scratch. This could lead to considerable savings in compute while maintaining high levels of accuracy. This possibility will be further investigated in future work.

REFERENCES

[1] Mishra, B.K. and Rajamani, R.K., 1992. The discrete element method for the simulation of ball mills. *Appl. Math. Model.*, *16*(11), pp.598-604.

[2] Mishra, B.K., 2007. Monte Carlo method for the analysis of particle breakage. *Handbook of Powder Technology*, *12*, pp.637-660.

[3] Gupta, A. and Mishra, B.K., 2024. Multi-head neural networks for simulating particle breakage dynamics. *Theoretical and Applied Mechanics Letters*, *14*(2), p.100515.

[4] Müller, L., Klar, A. and Schneider, F., 2019. A numerical comparison of the method of moments for the population balance equation. *Mathematics and Computers in Simulation*, *165*, pp.26-55.

[5] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S. and Yang, L., 2021. Physics-informed machine learning. *Nature Reviews Physics*, *3*(6), pp.422-440.

[6] Sharma, R. and Shankar, V., 2022. Accelerated Training of Physics-Informed Neural Networks (PINNs) using Meshless Discretizations. In *NeurIPS Proceedings*, *35*, pp.1034-1046.

[7] Wong, J.C., Ooi, C.C., Gupta, A. and Ong, Y.S., 2022. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, *5*(3), pp.985-1000.

[8] Sung, N., Wong, J.C., Ooi, C.C., Gupta, A., Chiu, P.H. and Ong, Y.S., 2023, July. Neuroevolution of Physics-Informed Neural Nets: Benchmark Problems and Comparative Results. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (pp. 2144-2151).

[9] Li, M., Sonoda, S., Cao, F., Wang, Y.G. and Liang, J., 2023, July. How powerful are shallow neural networks with bandlimited random weights?. In *ICML* (pp. 19960-19981). PMLR.

[10] Wang, Y. and Zhong, L., 2023. NAS-PINN: Neural architecture search-guided physics-informed neural network for solving PDEs. *arXiv preprint arXiv:2305.10127*.

[11] Dong, S. and Li, Z., 2021. Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations. *Comput. Methods Appl. Mech. Eng.*, *387*, p.114129.

[12] Hoşten, Ç. and Avşar, Ç., 2004. Variation of back-calculated breakage rate parameters in Bond-mill grinding. *Scandinavian journal of metallurgy*, *33*(5), pp.286-293.

[13] Krantz, S.G. and Parks, H.R., 2002. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media.

[14] Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J. and Schmidhuber, J., 2014. Natural evolution strategies. *The Journal of Machine Learning Research*, *15*(1), pp.949-980.

[15] Wong, J.C., Ooi, C.C., Gupta, A., Chiu, P.H., Low, J.S.Z., Dao, M.H. and Ong, Y.S., 2023. Generalizable Neural Physics Solvers by Baldwinian Evolution. *arXiv preprint arXiv:2312.03243*.

[16] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[17] Al-Dujaili, A., Srikant, S., Hemberg, E. and O'Reilly, U.M., 2019, February. On the application of Danskin's theorem to derivative-free minimax problems. In *AIP Conference Proceedings* (Vol. 2070, No. 1). AIP Publishing.

[18] Wong, J.C., Gupta, A. and Ong, Y.S., 2021. Can transfer neuroevolution tractably solve your differential equations?. *IEEE Computational Intelligence Magazine*, *16*(2), pp.14-30.