Dynamic Curriculum Regularization for Enhanced Training of Physics-Informed Neural Networks

Callum Duffy
Centre for Data Intensive Science and Industry (DISI)
University College London
Gower Street, London WC1E 6BT, United Kingdom
callum.duffy.22@ucl.ac.uk

Gergana V. Velikova
PASQAL SAS
2 av. Augustin Fresnel, 91120 Palaiseau, France
gergana.velikova@pasqal.com

Abstract

Physics-informed neural networks (PINNs) are increasingly used to solve differential and integral equations, particularly nonlinear partial differential equations (PDEs), but often struggle with complex industrially-relevant problems. We introduce a new curriculum regularisation scheme for training PINNs, addressing two key limitations in existing methods. First, we implement early stopping, based on the maximum residual of the PINN solution, to prevent high-residual regions from propagating. Second, we propose a dynamic adjustment of the PDE parameter step size, guided by the $\mathcal{L}_2$ distance between solutions, to handle varying solution complexities in the PDE and PINN solution quality. Our approach improves both the $\mathcal{L}_2$ relative error and convergence speeds across three PDEs, demonstrating greater robustness and efficiency over traditional curriculum regularisation. We predict this technique to generalise well to other PDEs where vanilla and causal PINNs fail to converge.

1 Introduction

Curriculum learning (CL) is a sample efficient method of training deep learning models where a meaningful ordering is enforced on the samples provided to the neural network, instead of being randomly shuffled. The ordering is such that easier to learn samples are seen at the start of training and progressively more complex samples are introduced during training, this is inspired by how humans learn [Ben+09]. CL utility has been explored in Reinforcement Learning, Computer Vision, and Natural Language Processing [HW19; Por+20; Xu+20].

Physics-Informed Neural Networks (PINNs) are Deep Learning models used to represent differential, integral, and integro-differential equations applying problem-specific regularisation via the loss terms, or through encoding symmetries within the model architecture [LLF98; RPK19]. CL has been applied recently in the context of PINNs [MA23], as researchers working in the domain are exploring strategies for more efficient training of PINNs and solving well-known failure modes related to non-convergence during training [Kri+21], when solving partial differential equations (PDEs). These methods have proven to be valuable when the PINN is provided with a limited number of collocation points.

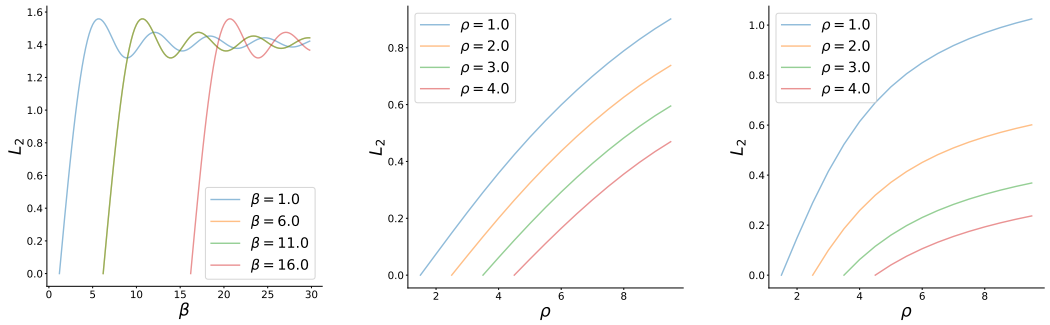
Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

Algorithm 1 Algorithm for the Dynamic CL regularisation method.

Initialise M (PINN model) with parameters θ
Define initial and target PDE parameters p_i, p_f
Define training points D_i (initial conditions), D_b (boundary conditions), D_f (collocation points using sampling function S_f with N_f points) and test points D_{test}.
Define optimizer Opt and training hyperparameters: maximum epochs $Epochs$, loss function L, early stopping metric E_m, and criterion E_c
Define new PDE parameter finder C and PDE Solver S (analytic or numerical)
Initialize PDE parameter $p \leftarrow p_i$
procedure DYNAMICCURRICULUM($M, Opt, p_i, p_f, D_i, D_b, D_f, Epochs, L, E_m, E_c, C, S$)
  while $p_i \neq p_f$ **do**
    for $i \leftarrow 1$ to $Epochs$ **do**
      Forward pass: $y \leftarrow M(D_i, D_b, D_f)$
      Compute loss: $l \leftarrow L(Y)$
      Update parameters: $\theta \leftarrow Opt(\theta, l)$
      Evaluate PINN: $l_{test} \leftarrow E_m(M(D_{test}))$
      if $E_c(l_{test})$ is true then **then** ▷ early stopping check
        Update PDE parameter: $p \leftarrow C(M, p, D_{test}, S)$ ▷ e.g. using L_2 distance
        Break ▷ Exit inner loop
      end if
      if Dynamic Sampling **then**
        Update collocation points: $D_f \leftarrow S_f(N_f)$
      end if
    end for
  end while
end procedure

2 Algorithm

In CL for PINNs, training begins with a manageable PDE complexity, gradually increasing complexity to the target parameter of interest during training. Traditional methods select evenly spaced PDE parameters, training on each for an equal number of fixed epochs, with extended training on the final parameter. While effective for challenging cases of the convection and reaction-diffusion equations where vanilla PINNs often fail [MA23], this approach has two main limitations. First, an equal number epochs per PDE parameter inaccurately assumes uniform training difficulty, neglecting failure modes; more complex parameters may require more training. Second, fixed parameter increments overlook solution quality and problem-specific characteristics of each PDE parameter. Our CL scheme resolves these issues, enhancing PINN accuracy and convergence speed. To address the first issue, we apply early stopping based on the maximum residual of the PINN during training, halting if the maximum residual falls below a threshold. This choice prevents training on already well-formed solutions, conserving the training budget for higher-complexity PDEs. The maximum residual criterion was selected to mitigate the risk of high residual regions persisting through training an effect that could destabilize solutions [Lu+21; Wu+23; NGM21; Wan+22; Daw+23]. Choosing a metric that monitors the maximum residual rather than the mean can prevent the early stopping criteria from being met prematurely, as the mean may obscure small, localized regions of high residual error by averaging them with lower residual values across the domain. For the second issue, we use the $\mathcal{L}_2$ distance between solutions to determine the step size in the PDE parameters. This distance as a function of a PDE parameter is shown in Figure 1, serves as a proxy for solution difficulty. For the convection equation with parameter β, PINNs may fail to converge at high values. From Figure 1a, we select the first minimum after the $\mathcal{L}_2$ maximum, it can be seen as the point at which the traveling waves in the solution become most in phase again after the maximum, to β. In contrast, for the reaction and reaction-diffusion equations Figures 1b and 1c, show $\mathcal{L}_2$ increases monotonically with ρ, consequently the $\mathcal{L}_2$ distance becomes a hyperparameter for step size. In practice, we calculate the $\mathcal{L}_2$ distance between the current PINN solution and target solutions (analytical or numerical) for subsequent PDE parameters, adjusting the PDE parameters until the $\mathcal{L}_2$ criterion is met. This dynamic selection allows for poorly performing PINNs to make smaller steps, thereby stabilizing the training process by adapting to solution difficulty. Refer to the pseudocode in 1 for the full procedure.



(a) (b) (c)

Figure 1: Curves representing the $\mathcal{L}_2$ distance between a solution and subsequent PDE parameters. (a) Convection equation, parameterised by β. (b) Reaction equation, parameterised by ρ. (c) Reaction-diffusion equation, parameterised by ρ.

3 Results

We explore the setting where we have $N_f = 1000$ collocation points, a resource limited regime that vanilla and causal PINNs have failed to converge, for the PDEs we considered [Daw+23], over ten different seeds for reproducibility. All reported models are the result of a hyperparameter sweep where the best of each type were selected. The neural networks used all share the same architecture a 4-layer fully connected neural network with 50 neurons per layer with a hyperbolic-tangent activation functions and a single neuron in the output layer. To ensure reproducibility of experiments, each of the models were run ten times for ten different random seeds, where all quoted errors are standard deviations. More details on the implementation can be found in Appendix A. Models labeled as "fixed step" follow the procedure outlined in Ref. [MA23].

3.1 Solving the convection equation

The convection equation, models heat transfer and other transport phenomena. We consider the one-dimensional case, a hyperbolic PDE is described by:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in [0, 2\pi], t \in [0, 1], \quad u(x, 0) = \sin(x), \tag{1}$$

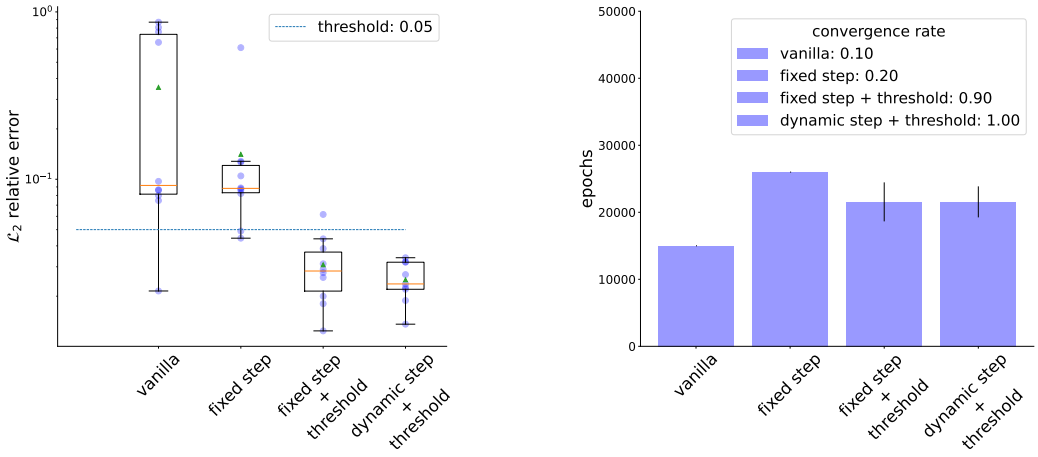
β is the convection coefficient and we enforce periodic boundary conditions. Previous studies show that for $\beta > 10$ PINNs fail to converge [Kri+21]. We set $\beta = 30$, to replicate a previous study into CL for the convection equation and compare it with our new technique [MA23]. Each model was trained for 27,500 epochs, with a learning rate of 5×10^{-4} using the Adam optimizer. We use a fixed set of collocation points drawn from the uniform distribution and define convergence as an $\mathcal{L}_2$ relative error falling below 0.05. For evaluating the loss at the boundary and initial conditions as well as evaluating the PINN at test time we discretize the time and space domain into 256 and 512 points respectively. The experimental results are presented in Table 1 and Figure 2 (Figure 2a, shows the $\mathcal{L}_2$ relative error for each model, while Figure 2b displays the epochs needed for convergence, with convergence rates in the legend). The vanilla PINN has the lowest convergence rate and highest $\mathcal{L}_2$ relative error. The CL scheme from [MA23], where β increases by 6 every 4000 epochs (7500 epochs for $\beta = 30$), improves on the vanilla PINN, but adding a 0.09 maximum residual threshold significantly improves performance. The best results are achieved by the model which dynamically adjusts the step in β while maintaining the same residual threshold, excelling in both average and worst-case scenarios.

3.2 Solving the reaction-diffusion equation

The next equation we consider is the reaction-diffusion equation, often used to model chemical systems. It has been previously shown that vanilla PINNs generally fail to solve this system [Kri+21]. The system can be described as follows

Table 1: The performance metrics related to the convection equation using multiple PINN training methods as described in the main text with CL standing for curriculum learning schemes. We define the epochs at convergence as an $\mathcal{L}_2$ relative error 0.05. The vanilla PINN is used to describe the original model described in [RPK19].

Method	$\mathcal{L}_2$ relative error (mean and SD)	Epochs at convergence
vanilla PINN	0.3540 ± 0.3465	15 ± 0k
CL Fixed step	0.1409 ± 0.1591	26 ± 0k
CL Fixed step + threshold	0.0308 ± 0.0136	21 ± 3k
CL Dynamic step + threshold	0.0251 ± 0.0064	21 ± 2k



(a) (b)

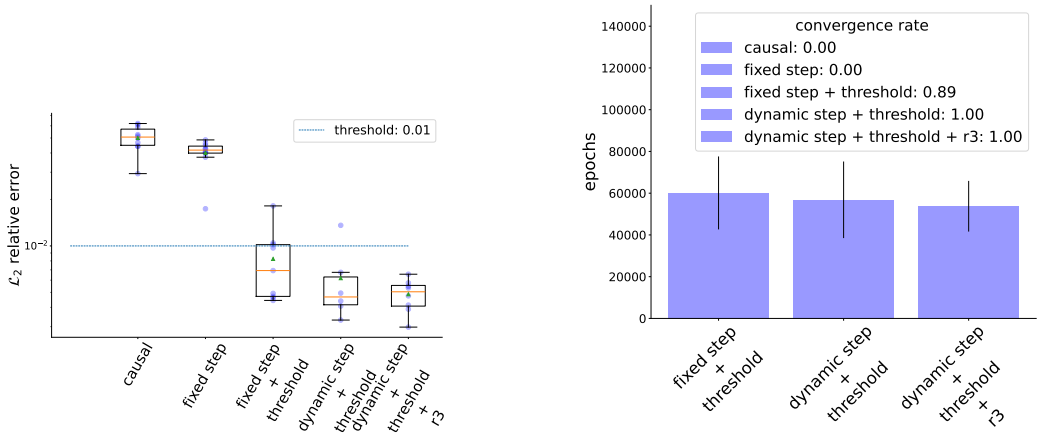
Figure 2: Results for the convection equation. (a) A boxplot of the $\mathcal{L}_2$ relative error for various models solving the convection equation. (b) A bar chart of the number of epochs to convergence for various models and rate of convergence.

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} - \rho u(1-u) = 0, \quad x \in [0, 2\pi], t \in [0, 1], \quad u(x, 0) = \mathcal{N}(\pi, \pi/4), \tag{2}$$

here, ν and ρ represent the diffusion and reaction components, with $\mathcal{N}$ a Gaussian distribution with mean π and standard deviation $\pi/4$. We select ρ to vary during CL. We find a low value of ρ is easier to solve than those greater. All models in this scenario are trained with a learning rate of 0.001 with the Adam optimizer, $80,000$ epochs and collocation points are resampled every epoch from the uniform distribution to aid convergence. We set the threshold for convergence with an $\mathcal{L}_2$ relative error of 0.01. For evaluating the loss at the boundary and initial conditions as well as evaluating the PINN at test time we discretize the time and space domain into 100 and 256 points respectively. We first solved the reaction equation for $\rho = 6$ and $\nu = 0$, a challenging case for vanilla PINNs. Each CL model began at $\rho = 1$ and ended at $\rho = 6$. A causal PINN was used as a baseline, as vanilla PINNs resulted in flat solutions across all trials as observed in [WSP22]. Results are shown in Table 2 and Figure 3, where the figures follow the same format as in section 3.1). The causal PINN failed to converge and had the highest $\mathcal{L}_2$ relative error. The fixed-step CL scheme, increased ρ by 2.5 every 25000 epochs ($30,000$ epochs for $\rho = 6$), reduced the $\mathcal{L}_2$ relative error but still did not converge. Adding a 0.05 maximum residual threshold, led to converges in some trials. The best results came from dynamically adjusting ρ, with the most performant model using R3 sampling (focusing on high residual regions) [Daw+23] instead of uniform sampling, converging in the fewest number of epochs and incurring lowest $\mathcal{L}_2$ relative error. These results demonstrate that coupling our CL scheme with advanced sampling techniques like $R3$ further enhances performance. Now, we shall briefly discuss the application of our CL scheme on the reaction-diffusion equation ($\rho = 6, \nu = 5$). Our baseline which was a causal PINN which achieved an $\mathcal{L}_2$ relative error of 0.05 ± 0.0092, failing to converge for all trials. In contrast, all CL schemes converged to $\mathcal{L}_2$ relative errors far below 0.01, what is of

Table 2: The performance metrics related to the reaction equation using multiple PINN training methods as described in the main text with CL standing for curriculum learning schemes. We define the epochs at convergence as an $\mathcal{L}_2$ relative error of 0.01. The causal PINN model is described in [WSP22].

Method	$\mathcal{L}_2$ relative error (mean and SD)	Epochs at convergence
causal PINN	0.0500 ± 0.0092	-
CL Fixed step	0.0401 ± 0.0082	-
CL Fixed step + threshold	0.0082 ± 0.0042	61 ± 17k
CL Dynamic step + threshold	0.0062 ± 0.0035	56 ± 18k
CL Dynamic step + threshold + R3	0.0049 ± 0.0011	53 ± 12k



(a) (b)

Figure 3: Results for the reaction equation. (a) A boxplot of the $\mathcal{L}_2$ relative error for various models solving the reaction equation. (b) A bar chart of the number of epochs to convergence for various models and rate of convergence.

note is the epochs to convergence. For the original scheme where ρ increases by 2.5 every 25000 epochs we converge on average in 51000 epochs, when adding in a maximum residual threshold of 0.02 this comes down to 45000 epochs, and when we allow for a dynamic step in ρ with an $\mathcal{L}_2$ distance of 0.3 along with a maximum residual threshold of 0.3 convergence occurs on average in 42000 epochs.

4 Discussion and limitations

We demonstrated that our CL techniques improve solution quality in terms of $\mathcal{L}_2$ relative error and convergence time, both in average and worst cases, for the convection, reaction and reaction-diffusion equations, outperforming previous approaches. Additionally, our method generalises well to different PINN architectures, including quantum PINNs, which show even faster convergence compared to industry standards (see Appendix B for details). However, this work has some limitations. We only tested three equations, so its effectiveness on other PDEs remains unknown. We also expect that if the number of collocation points sampled were to increase the performance gap in $\mathcal{L}_2$ relative error may decrease. Furthermore, while the PDEs we used were difficult due to residual-induced instabilities—making the maximum residual a natural threshold for early stopping, the initial and boundary conditions were relatively easy to solve. For equations where these conditions are more challenging, our method may require adjustments to perform effectively.

References

[LLF98] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". In: *IEEE transactions on neural networks* 9.5 (1998), pp. 987–1000.

[Ben+09] Yoshua Bengio et al. "Curriculum learning". In: *ACM International Conference Proceeding Series* 382 (2009). DOI: 10.1145/1553374.1553380. URL: https://dl.acm.org/doi/10.1145/1553374.1553380.

[HW19] Guy Hacohen and Daphna Weinshall. *On The Power of Curriculum Learning in Training Deep Networks*. 2019. arXiv: 1904.03626 [cs.LG]. URL: https://arxiv.org/abs/1904.03626.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems* 32 (Dec. 2019). ISSN: 10495258. URL: https://arxiv.org/abs/1912.01703v1.

[RPK19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[Por+20] Rémy Portelas et al. *Automatic Curriculum Learning For Deep RL: A Short Survey*. 2020. arXiv: 2003.04664 [cs.LG]. URL: https://arxiv.org/abs/2003.04664.

[Xu+20] Benfeng Xu et al. "Curriculum Learning for Natural Language Understanding". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 6095–6104. DOI: 10.18653/v1/2020.acl-main.542. URL: https://aclanthology.org/2020.acl-main.542.

[Kri+21] Aditi S. Krishnapriyan et al. *Characterizing possible failure modes in physics-informed neural networks*. 2021. arXiv: 2109.01050 [cs.LG]. URL: https://arxiv.org/abs/2109.01050.

[KPE21] Oleksandr Kyriienko, Annie E Paine, and Vincent E Elfving. "Solving nonlinear differential equations with differentiable quantum circuits". In: *Physical Review A* 103.5 (2021), p. 052416.

[Lu+21] Lu Lu et al. "DeepXDE: A Deep Learning Library for Solving Differential Equations". In: *SIAM Review* 63.1 (Jan. 2021), pp. 208–228. ISSN: 1095-7200. DOI: 10.1137/19m1274067. URL: http://dx.doi.org/10.1137/19M1274067.

[NGM21] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. "Efficient training of physics-informed neural networks via importance sampling". In: *Computer-Aided Civil and Infrastructure Engineering* 36.8 (Apr. 2021), pp. 962–977. ISSN: 1467-8667. DOI: 10.1111/mice.12685. URL: http://dx.doi.org/10.1111/mice.12685.

[Wan+22] Chuwei Wang et al. *Is L^2 Physics-Informed Loss Always Suitable for Training Physics-Informed Neural Network?* 2022. arXiv: 2206.02016 [cs.LG]. URL: https://arxiv.org/abs/2206.02016.

[WSP22] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. *Respecting causality is all you need for training physics-informed neural networks*. 2022. arXiv: 2203.07404 [cs.LG]. URL: https://arxiv.org/abs/2203.07404.

[Daw+23] Arka Daw et al. *Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling*. 2023. arXiv: 2207.02338 [cs.LG]. URL: https://arxiv.org/abs/2207.02338.

[MA23] Simone Monaco and Daniele Apiletti. "Training physics-informed neural networks: One learning to rule them all?" In: *Results in Engineering* 18 (2023), p. 101023. ISSN: 2590-1230. DOI: https://doi.org/10.1016/j.rineng.2023.101023. URL: https://www.sciencedirect.com/science/article/pii/S2590123023001500.

[PEK23] Annie E Paine, Vincent E Elfving, and Oleksandr Kyriienko. "Physics-Informed Quantum Machine Learning: Solving nonlinear differential equations in latent spaces without costly grid evaluations". In: *arXiv preprint arXiv:2308.01827* (2023).

[Wu+23] Chenxi Wu et al. “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 403 (Jan. 2023), p. 115671. ISSN: 0045-7825. DOI: 10.1016/j.cma.2022.115671. URL: http://dx.doi.org/10.1016/j.cma.2022.115671.

[Jad+24] Ben Jaderberg et al. “Let quantum neural networks choose their own frequencies”. In: *Phys. Rev. A* 109 (4 Apr. 2024), p. 042421. DOI: 10.1103/PhysRevA.109.042421. URL: https://link.aps.org/doi/10.1103/PhysRevA.109.042421.

[Sei+24] Dominik Seitz et al. “Qadence: a differentiable interface for digital-analog programs.” In: *arXiv:2401.09915* (2024). URL: https://github.com/pasqal-io/qadence.

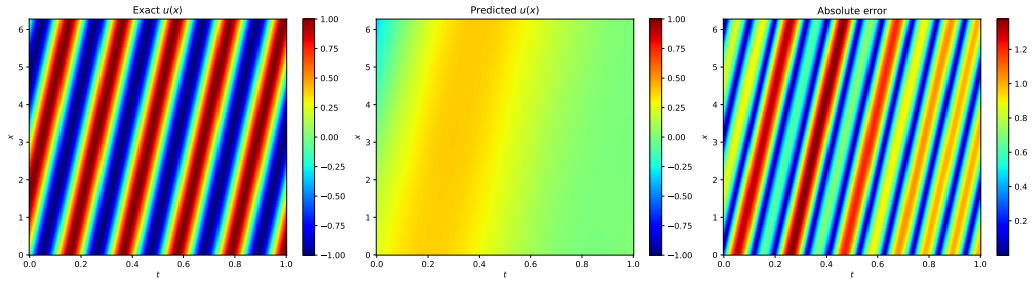


Figure 4: Results obtained for training a DQC on the convection equation with $\beta = 30$, without curriculum regularisation. (left) The exact solution obtained analytically. (middle) Solution obtained by the DQC algorithm. (right) The mean absolute error of the DQC solution.

A Implementation and resources

The curriculum learning scheme proposed was written in Python with the PINN models defined using Pytorch [Pas+19], and the quantum PINN models were implemented using Qadence [Sei+24].

All experiments were run on an internal high-performance computing cluster over CPUs. To run the hyperparameter sweeps 5 CPUs were required with 16GB of RAM and 250GB of storage was required to store the corresponding outputs from these runs.

B Quantum Physics-Informed Machine Learning

We have tested the applicability of our curriculum learning approach to a quantum physics-informed algorithm known as Differential Quantum Circuits (DQC) that are used to solve ordinary and partial differential equations using quantum neural networks (QNN) [KPE21; PEK23]. Our approach to CL mitigates propagation failures in DQC for various differential equations, drawing its initial motivation from solving the 1D convection equation with PINNs. Prior to using curriculum learning PINNs trained on the 1D convection equation for sufficiently high values of beta converge to a flat solution, failing to ascertain the high-frequency true solution. Similar pattern of non-convergence is observed in DQC.

The DQC model architecture used in these experiments is described in detail in [Jad+24].

B.1 Experimental Design and Results

The DQC models used for the study consisted of 2 qubits. The feature encoding gates are RY gates where the data is encoded into the angle. The ansatz block of the model consists of $RX(\theta)RZ(\phi)RX(\omega)$, where θ, ϕ, ω are trainable parameters, with nearest neighbouring entangling CNOT gates. Each DQC was trained for $100,000$ epochs and aimed to solve the convection equation for $\beta = 30$. Without curriculum regularization Figure 4, shows how the DQC failed to converge to an adequate solution. Introducing CL in the case of DQC where we increase β by 3 every $10,000$ epochs, with a threshold of 0.06 on the maximum residual, we obtain good convergence which can be seen in Figure 5.

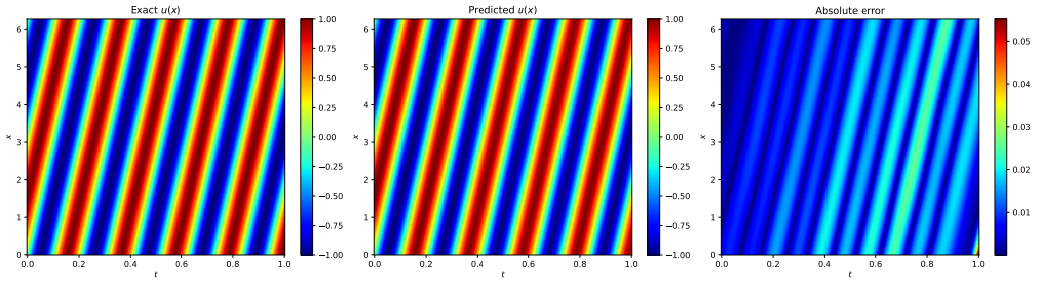


Figure 5: Results obtained for training a DQC on the convection equation with $\beta = 30$, with curriculum regularisation and an early stopping threshold. (left) The exact solution obtained analytically. (middle) Solution obtained by the DQC algorithm with curriculum regularisation. (right) The mean absolute error of the DQC solution with curriculum regularisation.