Applied Quantum Algorithms - Solving Nonlinear Differential Equations with Differentiable Quantum Circuits

Egor Sementul (6160689), Matylda Likus(6163912)

November 14, 2025

1 Introduction

This report takes case B Differential equation with highly non-trivial dynamics from the given paper [1] replicates its result, and as a novelty compares the DQC method against PINN (Physics-informed neural networks). The number of qubits and the depth of QNN was chosen after performing grid search with the objective of optimizing it (minimize them while ensuring the highest quality of result). To ensure comparability of methods, PINN number of parameters was set to match QNN and other setting were set to match each other.

Research questions:

1. What is optimal, best tradeoff between efficiency and effectiveness, number of qubits and depth of QNN for the given problem?
2. Can we gain power in terms of using less resources while achieving the same quality of solution using regularization with sigmoid scheduling?
3. How does a PINN with same number of trainable parameters compare to QNN in terms of solution quality, convergence speed, clock time per epoch/full training?
4. Under LBFGS finetuning of both models, can PINN outperform QNN in terms of solution quality, convergence speed, clock time? If, yes, where can it gain advantage and under which setting?
5. How do both models generalize in a scenario of limited training data? Can further fine-tuning improve generalization power?
6. How does the amount of training data affect the performance of QNN and PINN in terms of final solution quality?

2 Experiments

For the experiments we used case B Differential equation with highly non-trivial dynamics from the paper [1].

$$\frac{du}{dx} - 4u + 6u^2 - \sin(50x) - u\cos(25x) + \frac{1}{2} = 0, \quad u(0) = 0.75$$

To answer the research questions the following experiments were performed. It is also worth mentioning that all experiments were performed using Chebishev tower feature map, hardware efficient variational ansatz, total magnetization observable and floating boundary condition handling.

2.1 Experiment 1

As first experiment we performed grid search on number of qubits and depth for QNN to find optimal settings for it. We want to minimize the number of qubits and depth while maximizing the quality of the solution (mean square error between the shifted QNN output and a high accuracy reference solution). A full grid search was conducted over number of qubits from 2 up to 11 and depth from

4 up to 18. The result was was visualized on a logarithmic scale using a heatmap, where each cell represents a single QNN configuration, with lower values indicating better performance.

For this experiment we used QNN with the following parameters:

- 250 epochs
- 20 training points from 0 to 0.99
- Adam with adaptive learning rate and gradient clipping

2.1.1 Experiment 1.1

The goal of this experiment is to identify optimal combinations that minimize parameter space while maximizing quality of the solution when approximating the solution to a nonlinear differential equation with use of grid search with parameters explored above.

2.1.2 Experiment 1.2

This experiment extends the earlier grid search by incorporating a regularization with sigmoid scheduling into the QNN training pipeline, to explore how regularization affects the solution quality of QNN across different combinations of number of qubits and ansatz depths.

2.2 Experiment 2

This experiment is a direct performance comparison between PINN and QNN. The two models are parameter-matched to ensure a fair baseline for evaluating their expressiveness, training efficiency, and solution quality on a chosen case. For this experiment we used QNN and PINN with the following parameters:

- 500 epochs
- 20 training points from 0 to 0.99
- Adam with adaptive learning rate and gradient clipping
- For QNN 10 qubits with depth 5 (150 trainable parameters), and 153 for PINN (with following architecture: 1, 10, 11, 1)

2.2.1 Experiment 2.1

This experiment compares PINN and QNN to provide a baseline for how quantum models perform relative to classical PINN when controlled for parameter count.

2.2.2 Experiment 2.2

This experiment explores the impact of the LBFGS finetuning applied to the QNN and PINN, if further solution quality gains can be achieved. It compares PINN and QNN with LBFGS finetuning of 20, 100 and 250 epochs and 150 training points.

2.3 Experiment 3

This experiment aims to study sample efficiency and model generalization, where both the QNN and PINN are first trained on a reduced domain [0.0,0.5] using 20 points. This mimics scenarios where only partial information is available. After initial training, the models are finetuned using LBFGS optimization (with 20 epochs), still restricted to this reduced domain, before being evaluated across the full interval. For this experiment we used QNN and PINN with the following parameters:

- 500 epochs
- 20 training points from 0 to 0.5

- Adam with adaptive learning rate and gradient clipping
- For QNN 10 qubits with depth 5 (150 trainable parameters), and 153 for PINN (with following architecture: 1, 10, 11, 1)
- further finetuning for 20 epochs and 150 training points from 0 to 0.5.

2.4 Experiment 4

This experiment investigates efficiency of the QNN and PINN by evaluating how the number of training collocation points affects the solution quality. For this experiment we used QNN and PINN with the following parameters:

- 250 epochs
- 5 to 150 training points from 0 to 0.99
- Adam with adaptive learning rate and gradient clipping
- For QNN 10 qubits with depth 5 (150 trainable parameters), and 153 for PINN (with following architecture: 1, 10, 11, 1)

3 Results

3.1 Results for experiment 1

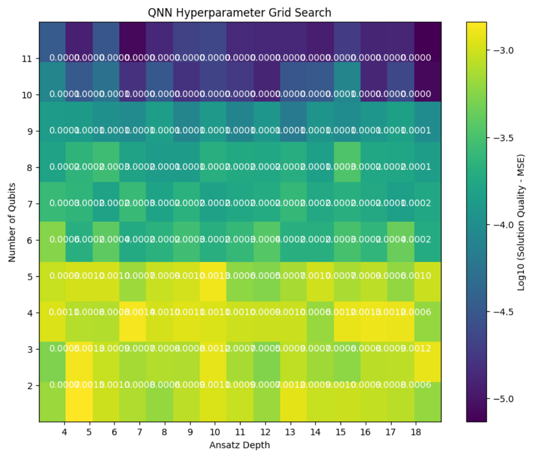

Figure 1: Grid search on number off qubits and depth for QNN

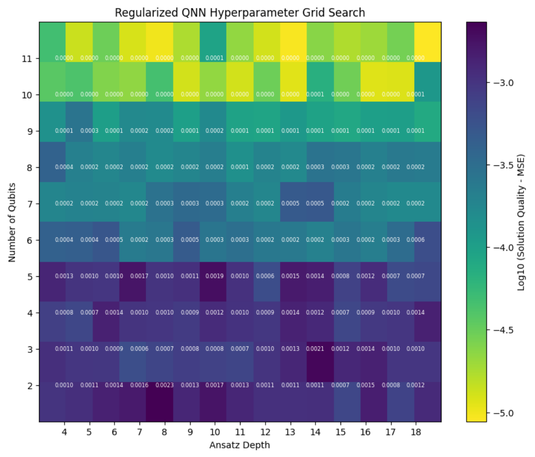

Figure 2: Regularised grid search on number off qubits and depth for QNN

3.2 Results for experiment 2

3.2.1 Results for experiment 2.1

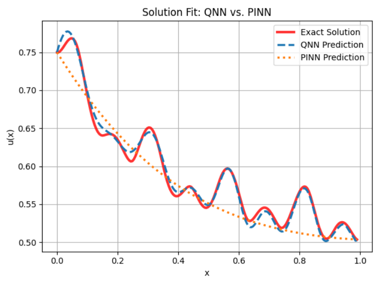

Figure 3: Solution for PINN with the same number of parameters as QNN

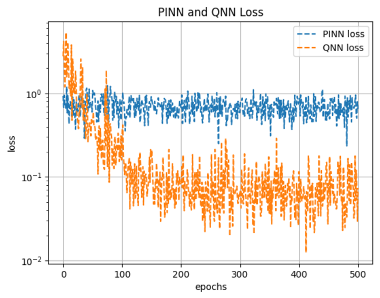

Figure 4: Loss for PINN with the same number of parameters as QNN

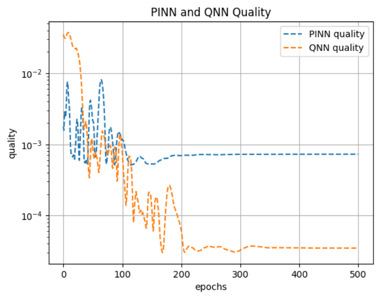

Figure 5: Quality of solution for PINN with the same number of parameters as QNN

3.2.2 Results for experiment 2.2

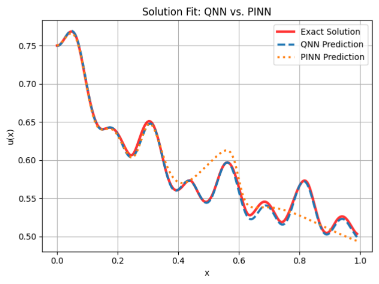

Figure 6: Solution with 20 epochs of finetuning

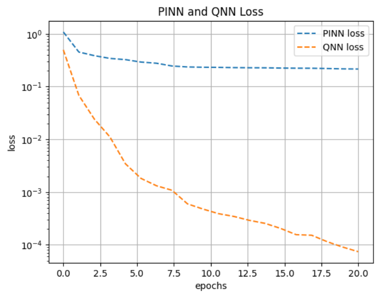

Figure 7: Loss with 20 epochs of finetuning

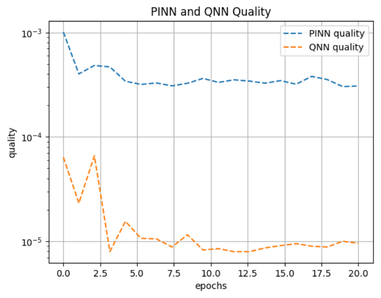

Figure 8: Quality of solution with 20 epochs of finetuning

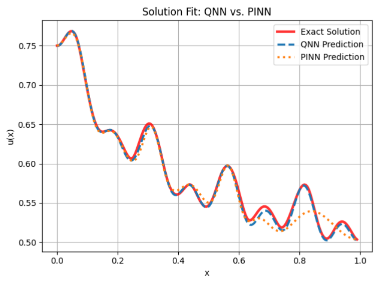

Figure 9: Solution with 100 epochs of finetuning

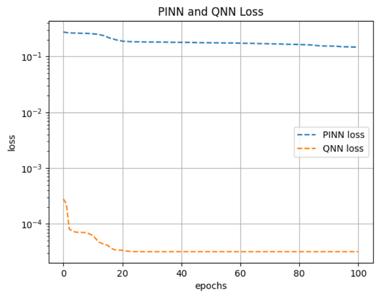

Figure 10: Loss with 100 epochs of finetuning

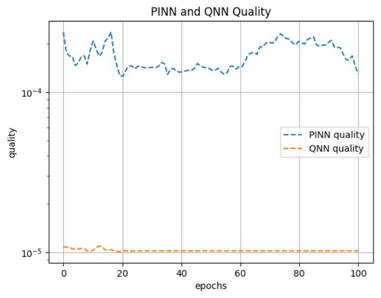

Figure 11: Quality with 100 epochs of finetuning

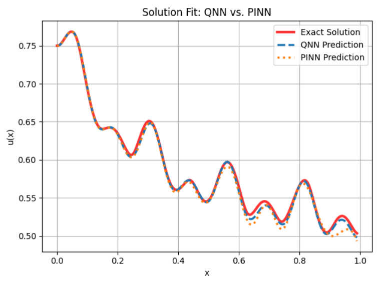

Figure 12: Solution with 250 epochs of finetuning

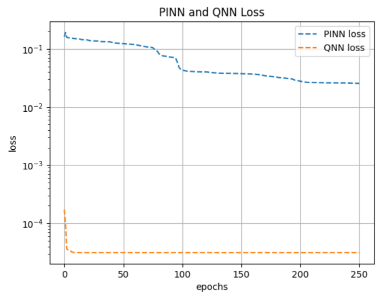

Figure 13: Loss with 250 epochs of finetuning

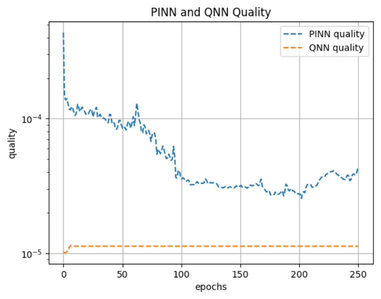

Figure 14: Quality with 250 epochs of finetuning

3.3 Results for experiment 3

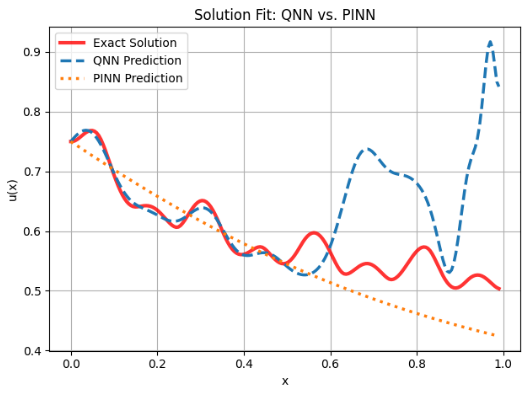

Figure 15: Fit for QNN and PINN with finetuning

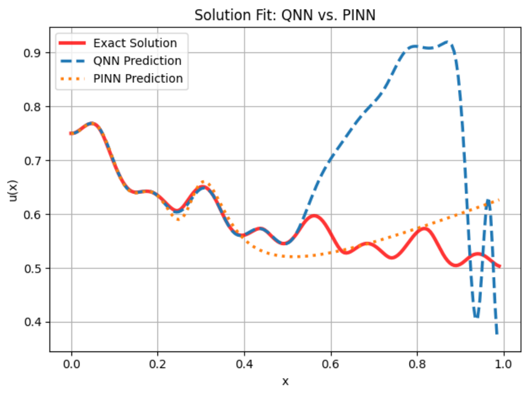

Figure 16: Fit for QNN and PINN with finetuning

3.4 Results for experiment 4

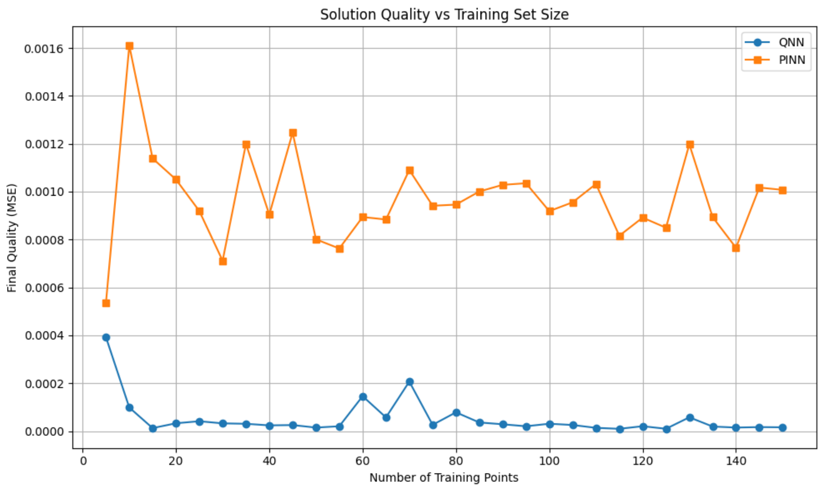

Figure 17: Quality of solution given number of training points

4 Discussion

Given the results we can say following things on the experiments.

From the first experiment, we can see very small impact of regularization on QNN, with two heatmaps having similar structure. The main insight is that the number of qubits plays a much larger role then the depth of the variational ansatz in the expressivity power of QNN. Considering the version without regularization we can see that first hight quality solution of value lower or equal to -4.5 is for 10 qubits and ansatz depth of 5. Given relatively low ansatz depth, that is why we used it for further experiments.

For the first part of the second experiment we see significant advantage of QNN solution, which is close to exact solution, while PINN only follows the solutions trend. It is also reflected on loss and solution quality graphs. However, considering the runtime PINN with is being 1 second is significantly better than QNN which took 6 minutes and 55 seconds for full training.

The second part of the second experiment explores the impact of LBFGS finetuning on QNN and PINN.
Considering finetuning of 20 epochs we see improvement in QNN performance mostly overlapping with exact solution. The loss function achieves loss of 10^{-4} (before it was 10^{-2}), and the quality (mean square error) improves from 10^{-4} to 10^{-5}. For the higher epochs we do not see many changes, the solution fit stays the same, while loss function and quality converges to previously mentioned values faster.
LBFGS finetuning has greater impact on PINN. For each value of finetuning epochs we see improvement in fit, loss and quality, and for 250 epochs they get close to QNN values, with biggest difference in loss. However, the QNN proves to perform better even after finetuning.
However, finetuning also impacts the runtime. QNN originally took 6 minutes and 55 seconds, and with 20 epochs of finetuning, its runtime increased to 13 minutes. With 100 epochs of finetuning it increased to 17 minutes and 31 seconds, and for 250 epochs it took 13 minutes and 44 seconds. On the other hand PINN runtime stays withing acceptable bounds being 2 seconds for 20 epochs, 11 seconds for 100 epochs and 26 seconds for 250 epochs. Therefore, even though PINN performs worse than QNN it takes significantly less time.

In the third experiment we wanted to test the generalization ability of QNN and PINN. We trained them on [0.0, 0.5] for 500 epochs, and test on full function. Furthermore, we tried with applying finetuning on [0.0, 0.5] before evaluating on the whole function period. As seen from the resulting plots, both models fail to generalize miserably. We only see a positive impact of training on [0.0, 0.5], where both QNN and PINN fit the function well.

In the last experiment we investigated the impact of the size of the training set on the quality of solution without the LBFGS finetuning. QNN quality improves slightly with the number of training points, converging near 0 MSE. We can see that increasing the number of training points beyond 20 does not improve the solution significantly. While for PINN it has smaller impact, the quality of solution just oscillates less with the number of training points.

5 Conclusion

In this section we would like to address the research questions stated in the beginning.
RQ1: for QNN best tradeoff between efficiency and effectiveness is observed for 10 qubits and ansatz depth of 5, which is 150 trainable parameters.
RQ2: applying regularization with sigmoid scheduling does not improve the performance QNN significantly. The general trend is observed, showing that the number of qubits play a larger role then the depth of the variational ansatz in the expressivity power of the quantum model, which is expected since it enriches the Chebishev polynomial basis and allows fitting wider range of functions.
RQ3: with the same number of trainable parameters for PINN and QNN, QNN performs significantly better in terms of solution quality, however, it takes more time to train. QNN converges after around 150 epochs and PINN does so almost immediately but that also reflects the bad, almost linear, initial

fit of that model.
RQ4:applying LBFGS finetuning QNN improves slightly, while PINN is strongly impacted by it, improving with the number of finetuning iterations. It still does not surpass QNN in terms of quality or loss, however, for 250 epochs PINN gets only slightly worse result than QNN while taking significantly less time.
RQ5: the models do not generalize well in scenario of limited training data, and LBFGS finetuning improves only fitting of known data.
RQ6: The amount of training points does not impact the quality of solution of QNN significantly, the improvement is seen when transitioning from 5 to 20 points. However, increasing the number further will only result in computational overhead of QNN training process. While, for PINN it reduces quality value oscillations with slight improvement.

Given the results of the experiments, QNN outperforms PINN, the same number of trainable parameters, with the cost of a long training time. Considering that QNN without finetuning has runtime of 6 minutes and 55 seconds and PINN with 250 epochs of LBFGS finetuning has runtime of 26 seconds, while bearing similar results, at the moment PINN with finetuning would benefit us more. That is also because PINN with even more finetuning epochs would give a better fit while still having a shorter runtime than QNN. However, as we used a perfect state vector simulator, it's runtime grows exponentially with number of qubits. On real hardware on the other hand, the noise should be taken into the account. We did attempt to use noise simulator with error mitigation, however, in qadence they only allow adding readout noise with mitigation through sampling which took too much time and the result we received differentiated too much from state vector simulator. We did consider switching to qiskit with mitiq, but it does not support differentiable circuit. In the future when the hardware constraints on quantum computers improve, the QNN could improve changing the turnouts.

References

[1] Oleksandr Kyriienko, Annie E. Paine, and Vincent E. Elfving. Solving nonlinear differential equations with differentiable quantum circuits. *Physical Review A*, 103(5), May 2021.