Week 1 – 26/05/2025

In the first week of this trimester, I began my work from the best model from last trimester, which is the moving quantum dot model. My initial plan was to try and create a more generalized model with variables such as vqd (speed of quantum dot movement) and ω (curvature of the potential), etc., as model inputs. The idea was to create a more flexible framework that could adapt to different quantum dot dynamics without needing to retrain from scratch.

The first experiment involved training a model with vqd as an input parameter. I choose the range of vqd between 10 nm/ps and 20 nm/ps. The model correctly predicted the quantum dot's movement and resulting electron's oscillations. However, I noticed that its accuracy dropped towards the end of the simulation.

Next, I trained the second model with ω as input. I trained it for omega in the range between 1 / ħ and 2 / ħ. This model's performance was better than the previous model, but it still faces some loss of accuracy issues, once again towards the end of the simulation. The results from the generalized models made me think about the improvement of my current moving quantum dot model.

This led me to explore causal training. In our context, causality means that a model's training should happen in a time-ordered manner. This means the predicted wavefunction or physics loss at a given time should depend on earlier times. Causal training clearly improved the performance of the vanilla moving quantum dot model, as it reduced the loss of accuracy towards the end of the simulation.

Week 2 – 02/06/2025

From last week, I saw clear improvement to the vanilla quantum dot model when done with causal training. This led me to try applying causal training to the generalized model with omega as input. As expected, this improved the result of the generalized model, as this made the model maintain its accuracy throughout the simulation.

After confirming these improvements, I started exploring methods to further refine the causal training strategy. In my current method, I am using a cumulative sum approach where the loss of a segment will be the sum of that segment’s loss and the cumulative loss of all previous segments. This ensures that the model respects the time-ordering constraint.

As part of trying a different approach, instead of adding the cumulative loss, I multiply the exponential of the sum of previous losses, along with a weight parameter to control the impact of causality. However, this approach does not improve my original cumulative sum method.

Week 3 – 09/06/2025

After confirming the success of causal training, my next goal was to explore a different training strategy that will improve the vanilla quantum dot model. This week, I began implementing normalization enforcement by introducing an additional loss term during training. The main purpose of this experiment was to check if this helps the model to maintain its accuracy.

To achieve this, I had to find the integral of the probability density ($|\Psi|^2$) over x to find the wavefunction normalization at a selected time. To calculate this integral during training, I decided to use Monte Carlo approximation using 1000 $|\Psi|^2$ samples. While using 10000 samples produced the best results, I had to stick with 1000 samples because otherwise the training time would increase significantly.

I start by enforcing normalization at individual time points such as $t = 20$ ps and combinations such as $t = 7$ ps and $t = 20$ ps. The results were consistent; the model stayed normalized at points where normalization is enforced. This led me to try and enforce normalization at multiple points at once to see if there are any improvements.

Week 4 – 16/06/2025

This week, I tried to implement different ways of normalization enforcement during model training and compared the results to see which one is the optimal approach. In all these experiments, I started applying normalization enforcement only after two-thirds of the training was completed. This is to ensure that the model will get an idea about the probability density of the problem before enforcing normalization. As in the previous week, I used Monte Carlo approximation to calculate the integrals.

The different combinations I used to calculate norm loss include:

1. 20 equally separated time points between 1 and 20 and 1000 $|\Psi|^2$ samples
2. 100 equidistant time points and 1000 $|\Psi|^2$ samples
3. 10 equidistant time points and 1000 $|\Psi|^2$ samples

Note that, for all these combinations, I calculated the norm loss at different points, and later I took their mean to create a final normalization loss, which I will then use to calculate the total loss. From these experiments, I got the best result when I used 20 equidistant time points to calculate the normalization loss. But the overall result of this approach was not better than the vanilla quantum dot model.

Week 5 – 23/06/2025

This week, I focused on modifying the way I calculated the final normalization loss. Previously, I calculated the final normalization loss for each epoch by taking the mean of all norm losses across the chosen time points. This week, I tried a different approach; instead of averaging, I summed up all the norm losses to get the final norm loss for that epoch. The idea was to increase the sensitivity of the norm loss and test whether this improved stability.

I tried this summation method with different numbers of time points, such as 10, 20, and 100, while keeping the number of probability density samples fixed at 1000. The results showed that this new approach did not outperform the previous method. This confirmed that the summation method added more computational load without delivering improvements in accuracy.

Week 6 – 30/06/2025

I continued my effort to find an optimal method to calculate the normalization loss during training. This week, I used the cumulative sum of norm losses at individual segments to find the final norm loss at that epoch. This approach is inspired by the way I did the causal training. Once again, I tried this week's strategy using 10, 20, and 100 time points. But the results I got were not better than the method from week 4.

While running these tests, I began reconsidering my earlier decision to only apply normalization enforcement for the last third of the training. For me, it became clear that this delayed application might limit the effectiveness of normalization enforcement. As a result, I experimented with enforcing normalization from the very beginning of the training process. Interestingly, this adjustment improved the vanilla model’s accuracy and stability. The trade-off, however, was an increase in training time.

Week 7 – 07/07/2025

Last week, I finally managed to create a new training strategy that will improve the vanilla quantum dot model, similar to causal training. This week, my first task was to create a hybrid model that combines both causal training and normalization enforcement. This further improved the overall performance of the vanilla quantum dot model.

After confirming the improvement of the hybrid model, I shifted my focus to further improving the normalization model. My goal is to find an optimal number of time points needed to calculate the normalization loss. As of now, I am using 20 equidistant time points to enforce normalization, but that was a trivial choice.

To investigate this, I tested different numbers of time points while also varying the curvature of the potential (ω). The idea was to find a relationship between the number of points and ω. But the results did not show any consistent trend.

Week 8 – 14/07/2025

I continued my effort to find a proper justification for the number of points used to calculate the normalization loss. This week, I tried to find a relation between the number of points and the time between two consecutive wave crests of the electron's oscillation. My idea was that the oscillation period might guide the spacing of time points. But unfortunately, I couldn't find any relationship. This suggests that the number of time points remains an empirical choice rather than one based on physical reasoning.

In addition to this, I tried to train the vanilla model only up to 12.5 ps and compare the results with the vanilla model trained up to 20 ps. The main idea was to check how much the oscillation is affecting the model’s performance. The model trained up to 12.5 ps clearly outperformed the other model. This motivated me to try to completely avoid the oscillations part and train only for the quantum dot's movement.

Week 9 – 21/07/2025

This week I trained a model for only the dynamic window, which is the time in which the quantum dot moves in space. This adjustment led to a noticeable improvement in the model’s performance, while significantly reducing its complexity and training time compared to the full window model.

I tested this concept with different ω and vqd values, and the improvement remained consistent in all cases. Later, I also tried and implemented causal training and normalization enforcement in this system, but I couldn't see any major improvements over the vanilla model, and I think this is because the vanilla model already is performing well.

After witnessing the performance boost of the dynamic-window model, I decided to try implementing the same in two dimensions. This way, I will get to know how well both PINN and Crank-Nicholson scale their complexity as I scale the dimension.

Week 10 – 28/07/2025

This week, I started by trying to check whether the Crank-Nicholson solution of the dynamic-window model breaks at any point. To check this, I increased the velocity of the quantum dot (vqd) so that the quantum dot's movement will be so quick it might take a lower time step (dt) for the Crank-Nicholson method. But this week I was unable to achieve the goal.

Another system I tried was the two-dimensional dynamic-window problem. This week I tried to solve the Schrodinger equation using the Crank-Nicholson method. This experiment demonstrated the difficulty of applying Crank-Nicholson to solve higher-dimensional problems. It took around 7 hours of simulation, whereas the same problem in 1-D took only 4 seconds. In addition to this, the output size of the 2-D solution took around 16 GB.

Week 11 – 04/08/2025

This week, I continued my attempts to make the Crank-Nicholson solution of the one-dimensional dynamic-window system break. To achieve this, I increased the v_{qd} almost up to the speed of light, the fastest the quantum dot can practically move. But unfortunately Crank-Nicholson was able to give a solution for that without decreasing its time step (dt). This was because, if we increase the v_{qd}, it actually reduces the total time required for the quantum dot's movement. This led to the dt of the Crank-Nicholson solution remaining the same, irrespective of the v_{qd}.

Later I solved the two-dimensional dynamic-window system using PINN. I managed to get good results with 6 hours of training, and the output file only took around 50 MB. So, in 2-D, PINN outperformed Crank-Nicholson both in time and space complexity.

Week 12 – 11/08/2025

This week, I wrapped up my work on this research project. Throughout this course, I noticed clear improvements in my academic writing, critical analysis, professional communication, teamwork, and ability to present technical ideas in simple language. In terms of technical skills, I developed a good understanding of the PyTorch framework and physics-informed neural network architecture.

I am satisfied with the results of my work. One major challenge I encountered was balancing accuracy with computational efficiency, which I addressed by experimenting with different loss formulations and model designs. Looking ahead, I aim to build on this research by exploring PINNs for even more complex quantum systems, and focusing on professional goals related to quantum computing and machine learning applications.