8th and 9th of February, Universidade do Porto, Portugal

This document details the problem to solve for PQHack 2025. Read it carefully. Good luck!

Useful references

To get started, you will find some useful information below:

- Qadence github: https://github.com/pasqal-io/qadence.
- Installation: https://pasqal-io.github.io/qadence/latest/getting_started/installation/.
- Useful QML constructors: https://pasqal-io.github.io/qadence/latest/content/qml_constructors/.
- 1D ODE example: https://pasqal-io.github.io/qadence/latest/tutorials/qml/dqc_1d/.
- Reference paper for the Helmholtz Equation problem: https://arxiv.org/abs/2001.04536
- Reference paper for PSR: https://arxiv.org/pdf/1905.13311

Also, do not forget to look at the tutorial notebook designed for this Hackathon!

Delivering your solutions

Each task will specify a deliverable in the form a python script that solves the task:

- You are expected to solve all tasks using trainable quantum circuits designed in Qadence. Alternative methods will not be accepted.
- Make sure you are using Qadence `v1.10.2`.
- Please follow the naming guidelines for each deliverable.
- Each script should be executable by running `python script.py`, and output the answer to the task.
- When generating random numbers, it's best that you fix a seed in your scripts so when we run the scripts we can reproduce exactly the same results you got.
- Certain tasks will require you to read some data files. Make sure you use relative paths that will work on any computer.
- You will not be evaluated on code quality, but do your best to keep the code clean. Remove unnecessary prototype code from the final deliverable, and provide useful comments to highlight your reasoning / organization where possible.

Finally, you should write a report explaining your reasoning or anything you think is important to understand your solution to each task. It can be as simple as a single text or pdf file with a section for each task. If a task asks for an image, please output it when running the script, there is no need to include them in the report.

For the final delivery combine all scripts, input data and the report in a single compressed folder `team_name.zip`. Send your delivery in the private discord channel for your team.

Part I: Learning simple functions

Now that you know about Qadence, let us go into our first task!
In `dataset_1_a.txt` you will find a dataset of pairs (x, y) generated by

$$x \to y = \cos(x + \phi) + \epsilon,$$

where ϕ is an unknown phase and ϵ represents some noise. You can probably think about many different ways to determine ϕ, but the goal here is to use variational quantum circuits (also known as quantum neural networks, or QNNs).
In this task, you should create a quantum circuit composed of two parts:

- a quantum feature map block $U(x)$ that encodes the input variable x as a feature parameter; and
- a variational block U_θ depending on a set of trainable parameters $\{\theta\}$.

We suggest that you start with a single qubit. When you run your circuit, you will get a state of the form

$$|f_\theta(x)\rangle = U_\theta U(x)\,|0\rangle\,,$$

We suggest that you set your output as

$$f_\theta(x) = \langle f_\theta(x)|\,\hat{O}\,|f_\theta(x)\rangle$$

for some observable $\hat{O}$. Naturally, you then want to optimize the variational parameters θ such that the pairs $(x, f_\theta(x))$ approximate the pairs (x, y) in your data set, which you can visualize with a plot, as in the tutorial.

Task 1A

Estimate the value of ϕ that was used to generate `dataset_1_a.txt`.

Deliverable: Write a script `task_1_a.py` that prints the value of ϕ at the end.

What if the signal is composed of more than one frequency? In `dataset_1_a.txt` you can find a list of pairs (x, y) that were generated according to the rule

$$x \longrightarrow y = \frac{1}{2}\Big(\sin(x + \phi_1) + \sin(2x + \phi_2)\Big) + \epsilon.$$

Bear in mind that the solutions with the smaller circuits and fewer qubits will receive the most points!

Task 1B

Estimate the values of ϕ_1 and ϕ_2 underlying `dataset_1_b.txt`.

Deliverable: Write a script `task_1_B.py` that prints the values ϕ_1 and ϕ_2 at the end.

Now we want you to analyze a dataset generated from the *fast-oscillating function*

$$x \longrightarrow y = \frac{1}{2}\Big(\sin(8x + \phi_1) + \sin(16x + \phi_2)\Big) + \epsilon,$$

which you can find in `dataset_1_c.txt`. How many qubits can your computer handle?

Task 1C

Estimate the values of ϕ_1 and ϕ_2 that generated `dataset_1_c.txt`.

Deliverable: Write a script `task_1_C.py` that prints the values ϕ_1 and ϕ_2 at the end.

Part II: Towards more flexible models

In the previous tasks you were trying to learn the phase of a harmonic signal. But that model can be too restrictive... For example, in `dataset_2_a.txt` the pairs (x, y) were generated by a function of the form

$$x \longrightarrow y = A_1 \sin(x + \phi_1) + A_2 \sin(2x + \phi_2) + \epsilon,$$

where A_1, A_2, ϕ_1 and ϕ_2 are unknown. How can you add expressiveness to your QNN? What kind model do you need to be able to fit for both amplitudes and phases?

Task 2A

Estimate the values of A_1, A_2, ϕ_1 and ϕ_2 that were used to generate `dataset_2_a.txt`.

Deliverable: Write a script `task_2_a.py` that prints the requested parameter values at the end.

One thing that all problems so far have in common is that you know the underlying frequencies before hand. But what if your data is generated by a model such as

$$x \longrightarrow y = A \cos(fx + \phi) + \epsilon,$$

where all the parameters A, f and ϕ are unknown? This is what you get in `dataset_2_b.txt`

Task 2B

Estimate the values of A, f, ϕ that were used to generate `dataset_2_b.txt`.

Deliverable: Write a script `task_2_B.py` that prints the requested parameter values.

In `dataset_2_c.txt` you can find a list of pairs (x, y) generated by a sum of three unknown frequencies,

$$x \longrightarrow y = A_1 \sin(f_1 x + \phi_1) + A_2 \sin(f_2 x + \phi_2) + A_3 \sin(f_3 x + \phi_3) + \epsilon.$$

This time, you are not asked to fit the unknown parameters. Instead, your task is to predict y for the new values of x found in `dataset_2_c_test.txt`.

Task 2C

Based on the training dataset of `dataset_2_c.txt`, predict y for each value of x found in `dataset_2_c_test.txt`.

Deliverable: Write a script `task_2_c.py` that plots your solution at the end, and generates a CSV file, `solution_2_c.csv`. For every x value in the test file, the CSV must contain a line with a pair x, y, with y being the model prediction for the input x.

Part III: Differential Equations

So far, you learned how to train a model to reproduce some data—great! However, there are cases where we want to learn a model *without* any training data. A simple example is a system defined by a differential equation like

$$\frac{d^2x}{dt^2} = -kx,$$

where k is a constant. This equation defines (implicitly) the motion of a harmonic oscillator. But now we have no data to guide us. In fact, it would be great to generate some data of our own!
So, how can you go about it? You can start, as before, with some model (a parameterized quantum circuit) that is flexible enough to encode the behavior of the harmonic oscillator. But now, instead of comparing its output $x(t)$ to some training data, you have to use the structure of the differential equation.

Task 3A

Choosing $k = 2.3$, solve the differential equation for the harmonic oscillator in the range $t \in [0, 10]$. Ensure that your solution respects the following boundary conditions: $x(0) = 1.0$ and $x'(0) = 1.2$.

Deliverable: Write a script `task_3_a.py` that plots your solution at the end, and generates a CSV file, `solution_3_a.csv`. For every t value in the test file `dataset_3_test.txt`, the CSV must contain a line with a pair t, x, with x being the model prediction for the input t.

Of course, a forever-oscillating mass is not really physical, is it? Find the solution to the *damped* harmonic oscillator, whose motion obeys the differential equation

$$\frac{d^2x}{dt^2} = -kx - d\frac{dx}{dt}.$$

Task 3B

Choosing $k = 3.3$ and $c = 0.8$, solve the differential equation for the damped harmonic oscillator in the range $t \in [0, 10]$. Ensure that your solution respects the following boundary conditions: $x(0) = 1.0$ and $x'(0) = 1.2$.

Deliverable: Write a script `task_3_b.py` that plots your solution at the end, and generates a CSV file, `solution_3_b.csv`. For every t value in the test file `dataset_3_test.txt`, the CSV must contain a line with a pair t, x, with x being the model prediction for the input t.

Part IV: Solving the Helmholtz Equation with a QNN

What's better than solving a one dimensional wave with a quantum computer? Solving a TWO dimensional wave with a quantum computer! The Helmholtz equation is the following partial differential equation (PDE),

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + k^2 u = q(x, y).$$

You should consider the domain $x, y \in [-1, 1]$. For the boundary conditions consider

$$u(x, \pm 1) = u(\pm 1, y) = 0.$$

For the forcing term use

$$q(x, y) = \left(k^2 - \omega_1^2 - \omega_2^2\right) \sin(\omega_1 x) \sin(\omega_2 y)$$

with $k = 1$.

Task 4

Choosing $\omega_1 = \pi$, $\omega_2 = 2\pi$, solve the Helmholtz equation for $x, y \in [-1, 1]$. Ensure that your solution respects the boundary conditions given.

Deliverable: Write a script `task_4.py` that plots your solution at the end, generates a CSV file, `solution_4.csv`. For every (x, y) value in the test file `dataset_4_test.txt`, the CSV must contain a line with the values x, y, u, with u being the model prediction for the input (x, y).

Part V: Derivatives and more derivatives!

By now you are a professional in Quantum Neural Networks! It's simple, you just add a bunch of parameters, define your loss function, and let PyTorch do all the work. Right? Well hang on... That works great on a simulation, but how do you think that will work on real quantum hardware? There is no such thing as automatic differentiation in quantum computing, and thus you need other techniques. Now you will learn a simplified *Parameter Shift Rule*

(PSR). PSRs allow you to get derivatives of quantum circuits with respect to circuit parameters by combining evaluations of that same circuit with shifts in the parameter values. Consider a QNN representing a function

$$f_\theta(t) = \langle f_\theta(t)| \hat{O} |f_\theta(t)\rangle$$

If you use a feature map of the type

$$U(t) = \bigotimes_{i=0}^{n-1} R_x^i(t)$$

for n qubits, then the derivative is given by

$$\frac{df_\theta(t)}{dt} = \frac{1}{2}\sum_{i=0}^{n}\left(\langle f_\theta^{i+}(t)| \hat{O} |f_\theta^{i+}(t)\rangle - \langle f_\theta^{i-}(t)| \hat{O} |f_\theta^{i-}(t)\rangle\right)$$

where the notation $i+$ and $i-$ indicate that the i-th R_x gate is evaluated as $R_x(t \pm \pi/2)$, respectively.
Your goal now will be to solve a differential equation without using automatic differentiation. Let's keep it simple, the equation you want to solve is

$$\frac{dx}{dt} = kx,$$

for $k = -2.5$. You should easily recognize that the solution is $x(t) = e^{kt}$. You should make use of PSR to compute model derivatives with respect to the input. To make it more realistic, you also aren't allowed to use PyTorch gradient-based optimizers, since those utilize automatic differentiaton to compute model derivatives with respect to the trainable parameters.

Task 5

Solve the differential equation above for $t \in [0, 1]$ with $x(0) = 1$ without using automatic differentiation.

Deliverable: Write a script `task_5.py` that plots your solution at the end, and generates a CSV file, `solution_5.csv`. For every t value in the test file `dataset_5_test.txt`, the CSV must contain a line with a pair t, x, with x being the model prediction for the input t.

If you want to go the extra mile, remember that quantum computers do not allow computing exact expectation values, as you have been doing so far, but only estimates based on sampling...

Part VI: Quantum Predators

A quantum computing scientist spent his whole career trying to find quantum advantage. Frustrated, after many years of training variational circuits without success, he decided to retire to the country side and bought a nice little farm. Over the course of the year, he spent his days drinking tea and counting all the rabbits and foxes running around in his farm. He meticulously counted the number of rabbits and foxes every day for 365 days, and registered them in `dataset_6.txt` in three columns (t, x, y), corresponding to the number of rabbits x and foxes y for each day t.

At the end of the year, he had a brilliant idea! All those years of training variational quantum circuits in Qadence could now put them to good use! He noticed that the populations of rabbits and foxes were well described by the famous Lotka-Volterra equations,

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = -\gamma y + \delta xy,$$

where $\alpha, \beta, \gamma, \delta$ are free parameters. He could characterize all the parameters that govern these populations with his quantum computer. Surely, this would be the quantum advantage he had been looking for all his career...

Task 6

Find the values for α, β, γ and δ that better describe the `dataset_6.txt`.

Deliverable: Write a script `task_6.py` that prints the requested values.