**Ion Trap Quantum Reinforcement Leaning Double-Qubit Circuit using complex manifolds.**

Please cite me as T.N.Toseland (The original author), in any future work where this gets used.

**Abstract.**

Up until now quantum deep reinforcement learning algorithms, have never actually been tested using entanglement rotation gates, and using complex valued tensors on the agent involved in this process.

Within a given environment these have all been typically either single qubit rotation gates, such as the qiskit rx gates or they have used some alternative ones, and therefore it is being proposed to measure this and hence see what does happen if that is attempted.

This is being done using a complex valued probabilistic agent, to see just what would happen if that is attempted.

**Introduction.**

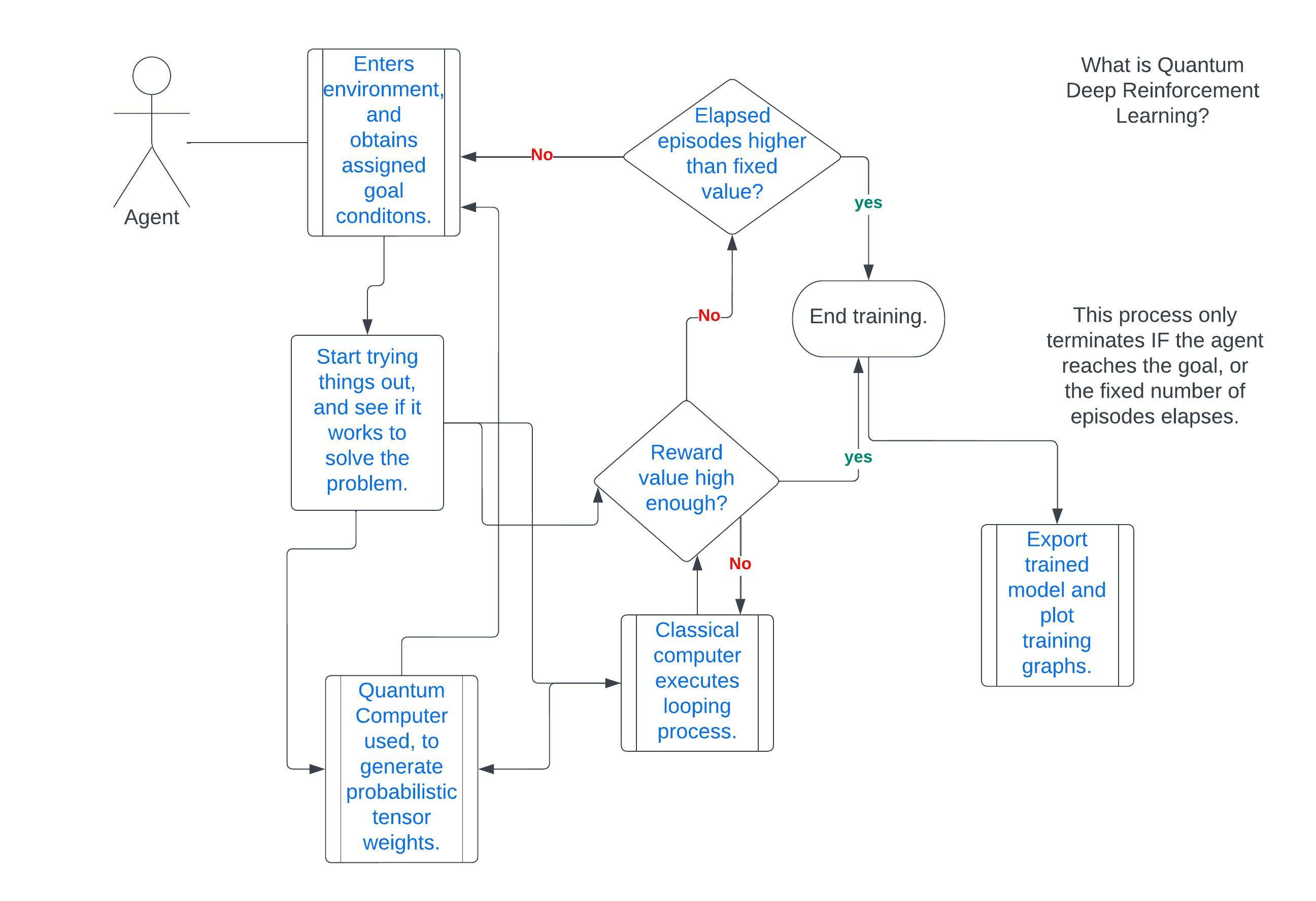
In this report a new type of circuit has been tested on quantum deep reinforcement learning using complex manifolds, rather than real-valued ones in the agent’s algorithm and it is being tested, on an Ion-Trap quantum computer. In particular this circuit uses entangling rotation gates rather than single qubit ones used in prior work,

To give an example of where exactly this has been used in the past and why, this has been attempted on fusion energy research in particular on magnetic containment, which is needed to try and contain the plasma reaction inside of a Tokomak. (22).

This is being done using a discrete approximation to continuous complex manifolds, as unfortunately at the time of writing this the hardware being used to try it, does not support openqasm 3.0 and therefore does not work with continuous gates.

**What is deep reinforcement learning?**

See below diagram for a basic idea of what reinforcement learning does, figure 1.

Deep reinforcement learning is a form of unsupervised machine learning, which unlike other networks such as a CNN (Convolutional Neural Network), does not rely on training data such as images for the purpose of making decisions about data. (23). Instead this relies heavily on training an agent within a specific environment which defines the rules, and it does this by defining a specific ‘reward function’ that the network is trying to optimise towards, so that once it is trained then it can achieve the required goals. It is open ended and it can try out different methods within the environmental constraints.

It has been suggested before that Quantum Computers can be used to accelerate complicated problems in machine learning, such as reinforcement learning for the purpose of solving difficult problems, but at the time of writing this there has been very little work on attempting to use a reinforcement learning algorithm with double-qubit gates on a Fourier transform instead of single qubit gates.

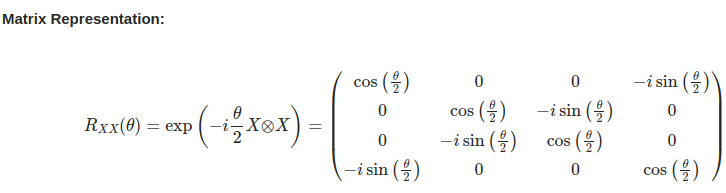
Therefore it is being proposed here that this be tested, and therefore some results obtained to find out just what does happen if that it attempted, or if multiple agents/critics are used on such an algorithm. Note that this will change the analysis aspects of this problem, as the double-qubit rotation gates will produce a complex valued output when used, instead of a real one.

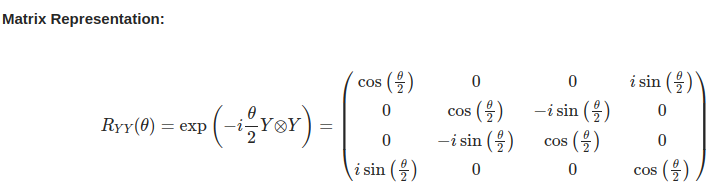
**Methods used.**

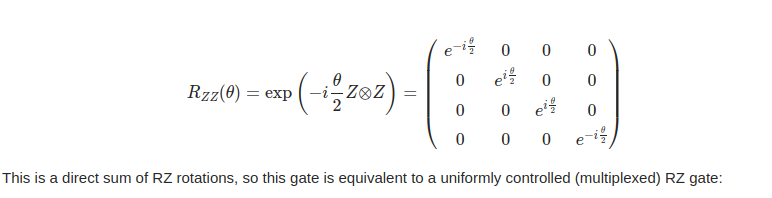
This has been attempted by modifying an existing notebook rather than creating a brand new one, as preliminary work to see what would happen if this is attempted, using alternative sets of qubit gates.

The gate configuration using on the encoding circuit was changed to be RXX, RYY and RZZ instead of the single qubit gates, RX, RY and RZ. This has been tested to find out what would happen, if this were to be attempted.

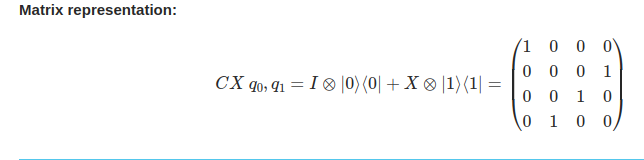
See below for the matrix representations of this.

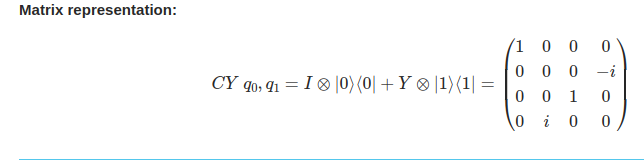


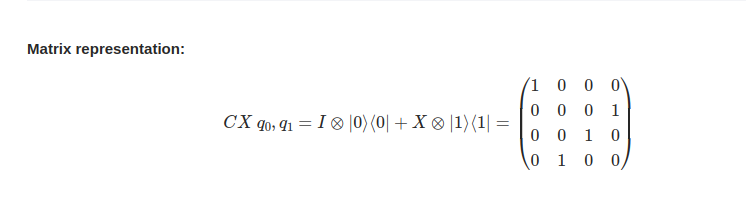




CX. CY and CZ gates shown below.







In the original notebook that this is based on the tensor used in Pytorch was real valued, which has now been changed to Numpy complex128 value date type. Note that without using this type of tensor, the algorithm will not actually run on IonQ’s simulator or on Harmony at all, and that this has actually been tested.

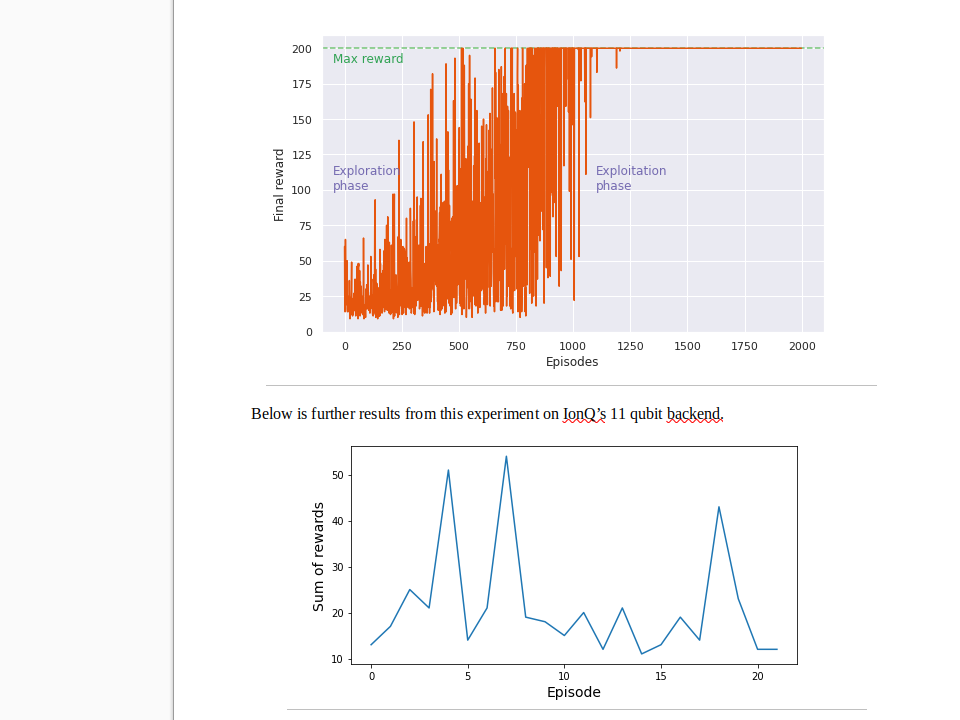
This will evaluate the required derivatives using a Wirtinger derivative to do it, which is admittedly crude but effective in simpler cases, like with the activation function used here selu, (the original had hyperbolic tangent).

Also the above gates are used in place of the original cz ones on the entanglement circuit, so although this has not been done entirely from scratch it is not the same as the original.

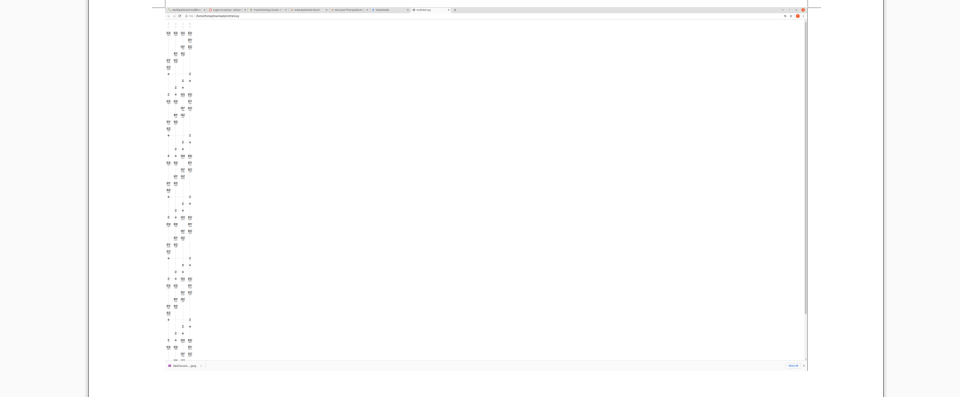
The loss function was set to hyperbolic tangent, as this seemed sensible.

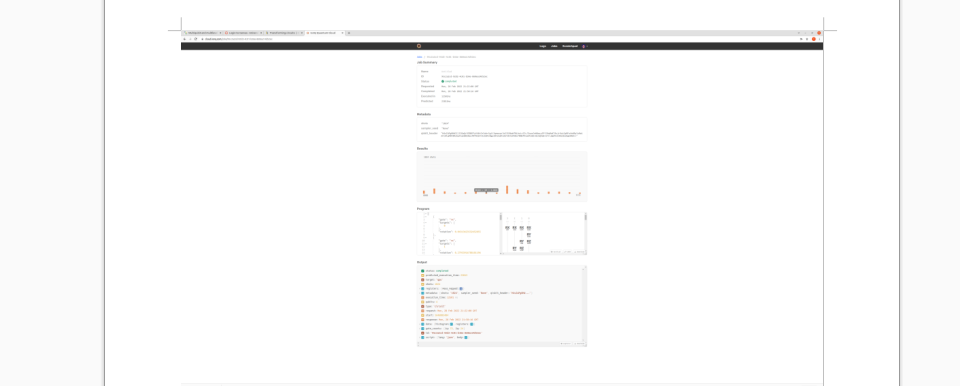
**Results obtained.**

Below are results obtained from a prior experiment using a single qubit algorithm on the cart pole problem, which was designed by another (1). Note that this was created by a different author, and all that has been done here is that it was tested on an actual quantum computer, namely IonQ Harmony 11-qubit Ion-Trap device, instead of simulating it. (run for 38 minutes total).





****

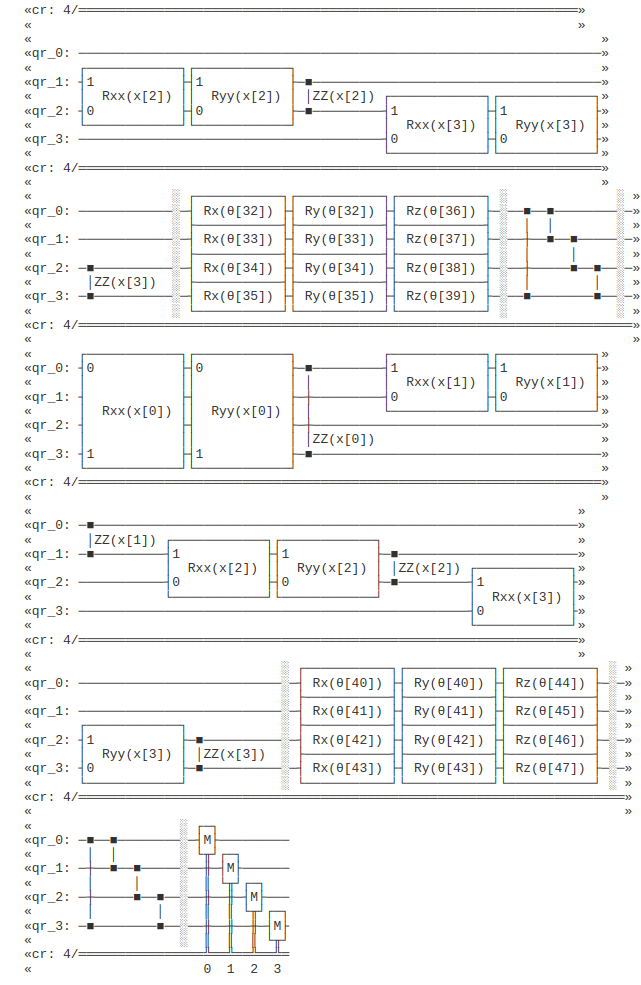
****

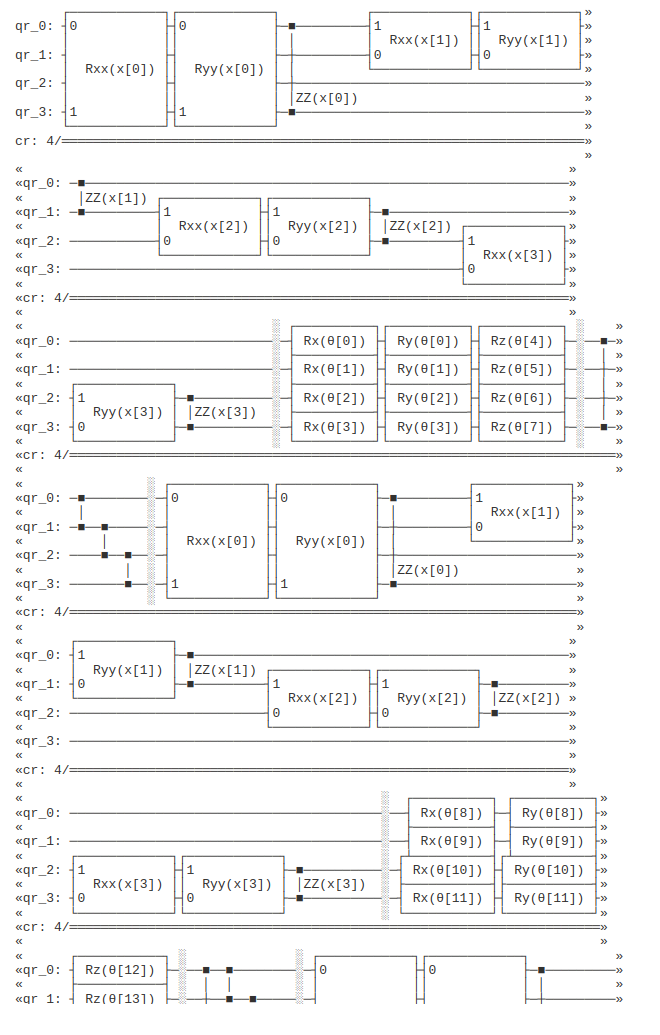
Above: Circuit diagrams and results obtained from the IonQ Experiment with only single qubit

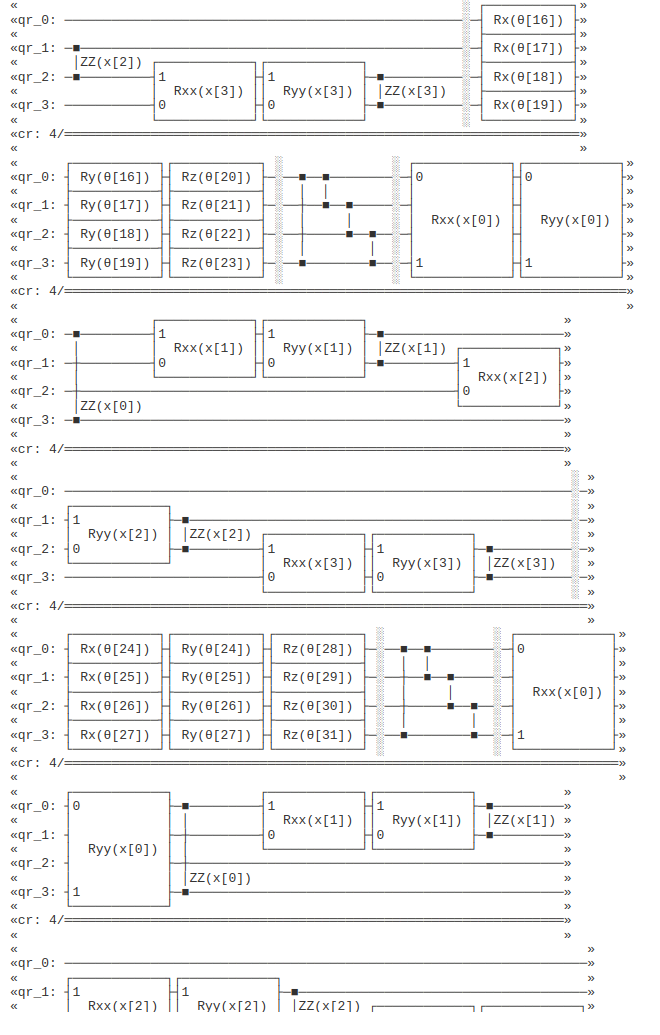
gates used, on quantum Fourier transform circuit and parametric encoding ones, using Pauli gates.

Below are results from attempts to make this work with double-qubit gates.

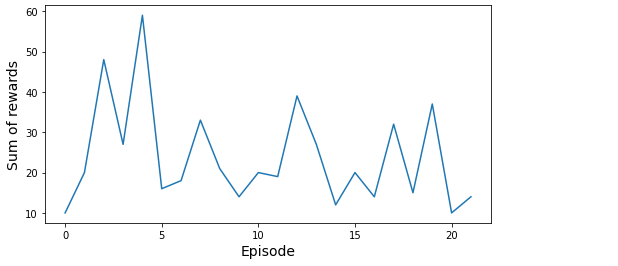
Circuit diagram shown below. (Figure 2a).







Note that in the above attempt the result was that the training didn’t actually finish, but the results shown below are what has been obtained on this, using the quantum computer mentioned below.

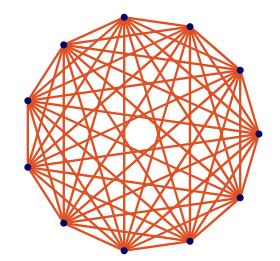


See below for output from Microsoft Azure, note that this was only simulated on Azure, but that this took a matter of less than an hour to complete, instead of 24 hours in the original.



Note that the reason that this is claimed, is because in the modified source code the only breaking condition for the training loop, is that the reward level reaches it’s maximum at 200, and although there were errors in the attempt to make this work that particular cell in the notebook did run successfully.

Note that the error shown below only occurred until the tensor data type used on the reinforcement learning agent, was changed to be complex valued instead of real floating point or integer. This occurred due to errors in open qasm 2.0 when it was tested using Qiskit on an IonQ platform, and also on classical simulations in an attempt to test it.

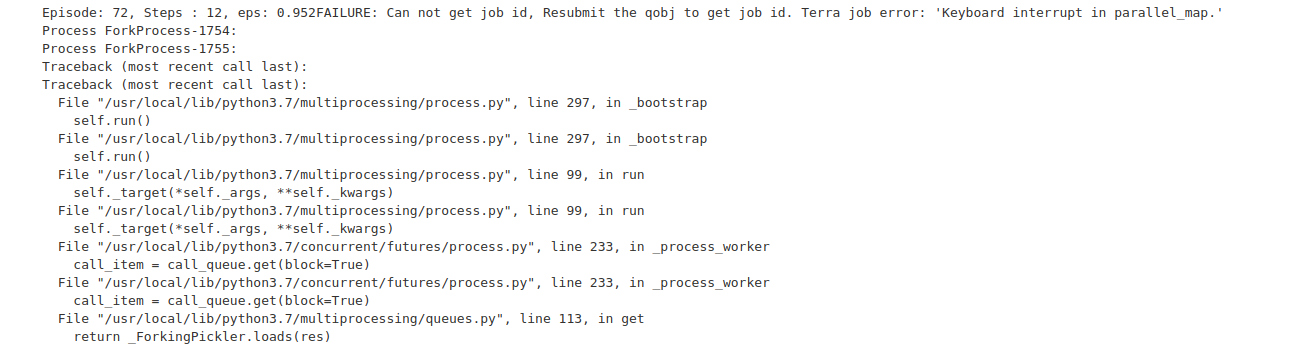
See above for an example of this, when using real variables. This does not happen with complex ones on the same notebook, if implemented using a pytorch complex valued tensor. Like for example with numpy complex 128.

But at the time of writing this, the reason why the graph shown above could be obtained at all from 21 episodes on the exact same notebook as this, is unknown.

Topology used for IonQ Harmony (in this experiment) shown below, using fully connected Ion-Trap qubits. (3).

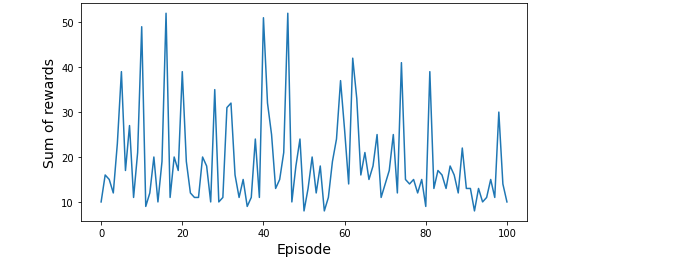
This was run for about 38 minutes to get the results shown above, please see the images for exact details.

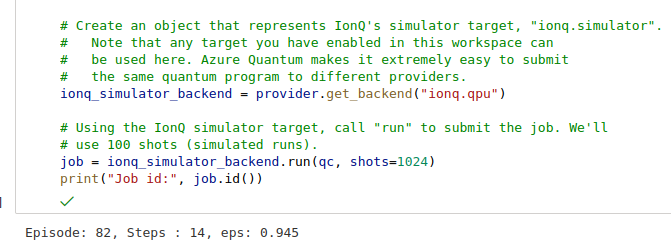
Note that the training above appears to be incomplete, but this simulation was working on a normal classical machine when last tested, and it only worked on IonQ’s simulator IF the input tensor used to create the agent is set to be complex valued. Otherwise qiskit will report an error on it as shown below.



Above is the latest results obtained on this, with around 72 episodes competed in 3 hours 23 minutes, on IonQ’s simulator. Note that this can realistically be tested on the actual quantum computer Harmony that was mentioned earlier, with very small changes to the source code.

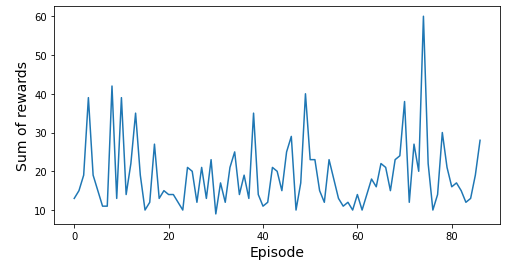
Below are the final results obtained from this experiment using IonQ’s Quantum simulator, unintentionally at the time of attempting to run this. (It was intended to be run on the actual hardware, but it took a total of about 4 hours to complete).





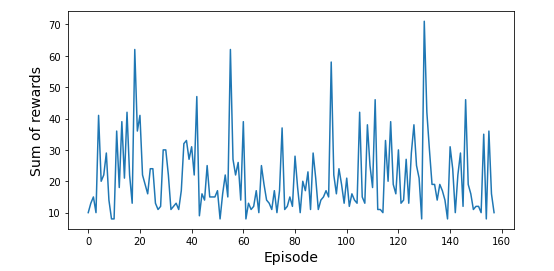
Note that it is puzzling as to why this didn’t actually run on the hardware at the time of testing this algorithm, at that this is also being tested using a different API to send the job across to Harmony instead of using the built in Microsoft Azure Quantum Qiskit API to do it.

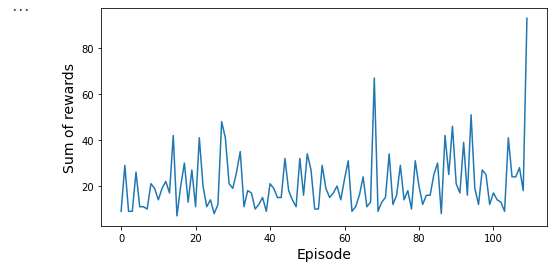
Actual result from Harmony shown below:



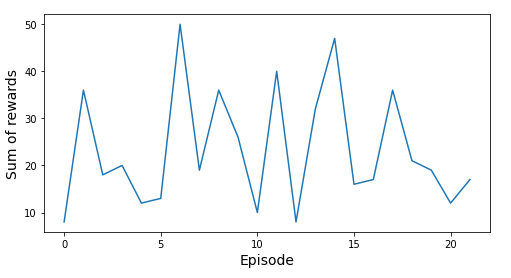
Unlike the simulated result shown above, this took less than 2 hours to complete, even though the training is strictly speaking incomplete.

Below are further results obtained from a modified circuit. Directly below: GPU result (classical simulation).



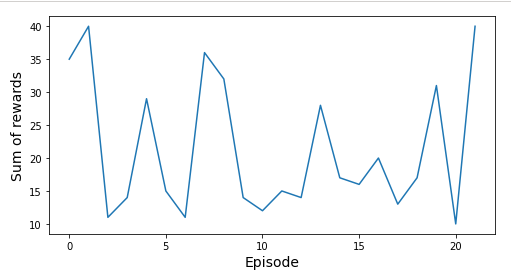
****

Above: result obtained from Quantum computer using modified algorithm, as that one has a LSTM memory layer attached. Note that the classical one shown above was not done using the IonQ simulator, but it was done using a local copy of qiskit and that this was not setup to use the IonQ simulator software, as it wasn’t known how to do at at the time of testing (on the local hardware).

****

**IonQ Simulated result above, using model with 3 LSTM layers using IonQ Simulator.**

**IonQ Quantum Computer result shown below.**

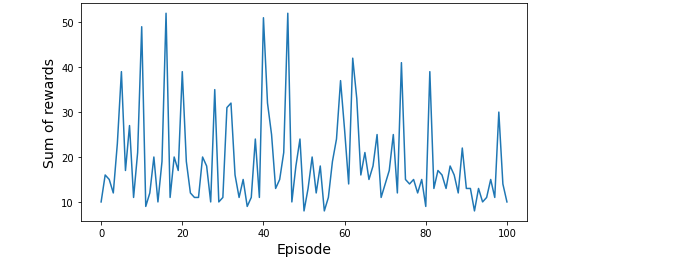
****

Note that in more than 1 of these examples, the training is not actually completed due to timing constraints at the time of testing it, and that there is more work to do on that.

It is not yet known why, but running the exact same circuit on 2 different pieces of hardware with the same software and the same API, is giving different results as shown above even over the same number of shots and the same circuits.

Final results obtained (with this running on the actual hardware as much as possible) shown below:

Note that this took about 41 to 45 seconds to run on IonQ Harmony.



**Discussion.**

It is not yet known why this happened, but in the above results obtained the simulated circuit ran for 4 hours in total on this cart pole problem, instead of more like 24 hours on the original circuit. (2).

Further testing on slightly modified circuits has also resulted in somewhat different graphs as shown above, even though these were tested using the exact same parameters and the exact same circuits, even the same python notebooks. Note that some of these have got a LSTM memory built into them.

However, what was different was that the one shown directly above was run on Harmony via Microsoft Azure where as the one above that, was run locally on a Nvidia Geforce RTX3090 graphics card.

It is quite possible that the reason why there are different results on this is because one version of this notebook, uses complex valued tensors where as the other one uses real valued floating point tensors.

At peak performance the quantum computing one appears to have higher performance (on Harmony), but on average the classical simulation appears to have higher reward values. One possible theory is that quite simply the quantum computer has more noise in it, and it will execute the qubits in an entangled way (circular in this case), where as a graphics card will only be able to simulate this.

It could be potentially argued that because GPU’s don’t have the same ‘noise’ problems that an actual quantum computer does, the results may not be entirely the same as each other although that does not seem like an explanation, for why the values of ‘Sums of rewards’ would be higher on 1 platform than on the other given the same software installed.

There has been other work attempted before now, to try and run a reinforcement learning agent on a quantum computer that does use an alternative architecture, such as photonics but this has not actually been done using entanglement gates, that are mathematically the same as these particular ones are. (20). Especially as that particular one relies heavily on single qubit gates, and so it doesn’t entangle them in the same way, as this work does.

Finally it can be said that the quantum hardware because of it’s nature being an Ion-Trap device instead of superconducting qubits, may not necessarily behave in exactly the same way as the hardware in which qiskit’s simulation software was originally, based on (superconducting qubits that is).

**Conclusions.**

Using complex valued tensors on entanglement rotation gates such as the ones used here, would appear to produce results that are much smoother mathematically on the training graphs, when used on an Ion Trap quantum computer such as the one that was used here.

It has been suggested that this may be down to the fact that complex manifolds used to calculate the probabilities, from the probability distributions used inside of this network, are continuous instead of semi continuous in the case of real valued manifolds.

These results also demonstrate that at the time of testing using openqasm 2.0, it is not actually possible to use these particular circuits with a real valued agent in this context, because of limitations on the quantum assembly language used here.

There is also evidence here that a quantum computer if used correctly, can at least partially train a neural network such as this one on reinforcement learning, if it is used correctly.

Finally this would suggest that while the original notebook in which this was based on does take around 24 hours to train, that if entanglement gates are used and complex valued tensors, then it may be possible to reduce that by a factor of 6 so that it only takes around 4 hours to do instead.

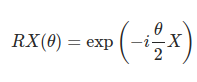
**Proposed future work.**

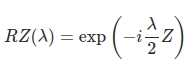
There were other errors shown when this was attempted, which are a bit inconsistent between IonQ and IBM Quantum Experience, both of which have been used in this work.

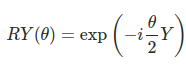
At the time of writing this, this particular circuit has not been tested on a photonic device and so the results may be different, especially as at the time of writing this the Ion trap device used to test this doesn’t support a continuous form of qubits. Therefore if this were to be attempted using continuous complex valued gates, rather than using the discrete model attempted here the results may well be different.

It may also be possible to use this as part of a larger semi-supervised learning system, such as a GAN combined with this reinforcement learning algorithm, which could well use entanglement gates and complex valued tensors too.

**Circuit gates used on single qubit tests (Original circuit only, NOT this one).**

****

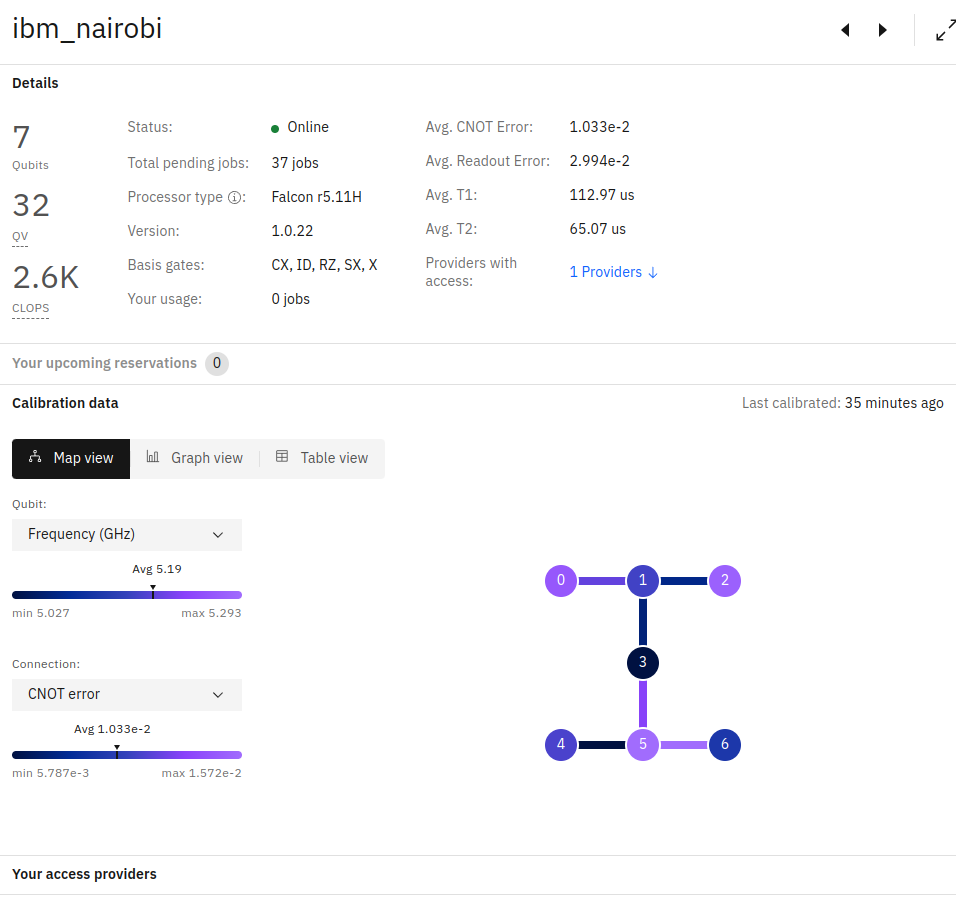
****

****

For double-qubit gates, please see further up.

**Topological Configuration on IBM Quantum Experience (Superconducting Qubits).**

Note that on this particular platform, it has not actually been possible to make this algorithm work with the hardware, even on testing the circuit using IBM’s circuit composer. Because quite simply the hardware does not seem to support all of these particular gates.



**References.**

<http://bayesiandeeplearning.org/2021/papers/22.pdf> [online] (28/02/2022). (1).

[**https://github.com/LauraGentini/QRL**](https://github.com/LauraGentini/QRL) **[online] (accessed 28/02/2022). (2).**

[**https://ionq.com/best-practices**](https://ionq.com/best-practices) **[online] (accessed 05/07/2022). (3).**

[**https://pytorch.org/docs/stable/generated/torch.complex.html**](https://pytorch.org/docs/stable/generated/torch.complex.html) [online] (accessed 09/07/2022). (4).

<https://ionq.com/docs/get-started-with-qiskit> [online] (accessed 09/07/2022). (5).

<https://qiskit.org/documentation/partners/ionq/guides/usage.html> [online] (accessed 09/07/2022). (6).

<https://qiskit.org/documentation/stubs/qiskit.circuit.library.RZZGate.html> [online] (accessed 14/07/2022). (7).

<https://qiskit.org/documentation/stubs/qiskit.circuit.library.RXXGate.html> [online] (accessed 14/07/2022). (8).

<https://qiskit.org/documentation/stubs/qiskit.circuit.library.RZZGate.html> [online] (accessed 14/07/2022). (9).

<https://arxiv.org/abs/1611.02779> [online] (accessed 14/07/2022). (10).

<https://ionq.com/docs/getting-started-with-native-gates> [online] (accessed 16/07/2022). (11).

appropriate resources from <https://qiskit.org/> [online] (accessed 16/07/2022). (12).

<https://www.nature.com/articles/s41467-019-13534-2.pdf> [online] (accessed 20/07/2022). (13).

S.Russell.P.Norvig. Artificial Intelligence: A Modern Approach. (14).

<https://github.com/openqasm/openqasm/tree/OpenQASM2.x> [online] (accessed 21/07/2022). (15).

<https://arxiv.org/pdf/2104.14722.pdf> [online] (accessed 21/07/2022). (16).

<https://arxiv.org/pdf/2109.00506.pdf> [online] (accessed 21/07/2022). (17).

<https://strawberryfields.readthedocs.io/en/stable/> [online] (accessed 21/07/2022). (18).

<https://pennylane.ai/qml/> [online] (accessed 21/07/2022). (19).

<https://arxiv.org/pdf/2108.12926.pdf> [online] (accessed 22/07/2022). (20).

<https://strawberryfields.readthedocs.io/en/stable/code/sf_ops.html> [online] (accessed 21/07/2022). (21).

<https://www.nature.com/articles/s41586-021-04301-9> [online] (accessed 28/07/2022). (22).

<https://link.springer.com/content/pdf/10.1007/s13244-018-0639-9.pdf> [online] (accessed 28/07/2022). (23).

Special thanks to Dr Doug Williamson from Cambridge University, for reviewing this work.