

# Quantum amplitude estimation with error mitigation for time-evolving probabilistic networks

M.C. Braun<sup>1</sup>, T. Decker<sup>\*1</sup>, N. Hegemann<sup>1</sup>, S.F. Kerstan<sup>1</sup>, C. Maier<sup>2</sup>, and J. Ulmanis<sup>2</sup>

<sup>1</sup>JoS QUANTUM GmbH, c/o Tech Quartier, Platz der Einheit 2,  
60327 Frankfurt am Main, Germany

<sup>2</sup>Alpine Quantum Technologies GmbH, Technikerstrasse 17 / 1,  
A-6020 Innsbruck, Austria

March 30, 2023

## Abstract

We present a method to model a discretized time evolution of probabilistic networks on gate-based quantum computers. We consider networks of nodes, where each node can be in one of two states: good or failed. In each time step, probabilities are assigned for each node to fail (switch from good to failed) or to recover (switch from failed to good). Furthermore, probabilities are assigned for failing nodes to trigger the failure of other, good nodes. Our method can evaluate arbitrary network topologies for any number of time steps. We can therefore model events such as cascaded failure and avalanche effects which are inherent to financial networks, payment and supply chain networks, power grids, telecommunication networks and others. Using quantum amplitude estimation techniques, we are able to estimate the probability of any configuration for any set of nodes over time. This allows us, for example, to determine the probability of the first node to be in the good state after the last time step, without the necessity to track intermediate states. We present the results of a low-depth quantum amplitude estimation on a simulator with a realistic noise model. We also present the results for running this example on the AQT quantum computer system PINE. Finally, we introduce an error model that allows us to improve the results from the simulator and from the experiments on the PINE system.

## 1 Introduction

The analysis of networks and their time evolution has applications in several industries, for example in financial risk management where counterparty risk needs to be evaluated

---

<sup>\*</sup>For contact email: thomas.decker@jos-quantum.de or sven.kerstan@jos-quantum.de. Authors are listed in alphabetical order.

to calculate regulatory capital requirements. Value at risk (VaR) or expected shortfall (ES) are common risk measures that need to be calculated by banks, asset managers, regulators and insurers on a regular basis. For example, we can think of a network of companies and banks that have financial obligations like credit contracts or that have the same assets on the balance sheet. A bank needs to control its overall credit exposure by estimating the likelihood of each counterparty defaulting, called probability of default. As the financial network is highly correlated, systemic risk needs to be considered as cascading effects can destabilize the system or parts of it [1–3].

In this publication, we present a method to model a discretized time evolution of probabilistic networks on gate-based quantum computers. First, we present the model that evolves over time in section 2 and then extend it to a quantum version in section 3. The quantum version can be run on gate-based quantum computers, which might lead to computational benefits, when the quantum computing hardware is available in the required size and precision. Compared to our previous work for the sensitivity analysis of business risk models [4], the model for networks and the model for business risks have in common that there are intrinsic and trigger probabilities. The extended network model, which we present in this paper, allows for arbitrary network topologies including cycles as the states in each time step depend only on the states of the previous time steps. Furthermore, each node comes with a recovery probability that allows a node to recover after a failure with a certain probability. To evaluate probabilities of default, Monte Carlo methods can be used to evaluate the model by randomly sampling results. The error of classical Monte Carlo methods decreases with  $\sqrt{1/N}$  for  $N$  samples. Thus calculating results with a 10 times higher accuracy, requires 100 times the computational effort. Depending on the size of the network this can get computational prohibitive.

We can use Monte Carlo methods on quantum computers, which refer to a quantum algorithm called quantum amplitude estimation (QAE) [5], to calculate risk measures for the probability of the failure of one node or a group of nodes. Various applications of QAE have been shown within finance, e.g. risk analysis [4, 6], the pricing of financial derivatives [7, 8] and many more [9, 10]. Using QAE would achieve a quadratic speedup compared to its classical counterpart and therefore requires only 10 times the computational effort for 10 times higher accuracy.

The standard amplitude estimation procedure [5] has hardware requirements that are challenging for current quantum devices due to noise. An example for a real world business problem was analyzed in [4]. For that example, it was estimated that the successful execution on a quantum computer would require it to perform at least 100 million gate operations before a single error occurs, on average. Furthermore, several assessments of the necessary overhead for quantum error correction have led to discouraging predictions for what would be required to achieve solutions that are faster or more cost-efficient than classical algorithms [11]. Reducing the requirements of a QAE is currently an active area of research [12–18] offering the possibility to implement QAE with lower requirements towards hardware fidelity [19].

We assess the viability of the analysis of a small network model with a variant of low-depth QAE methods in section 4. We introduce a noise model, which can be fitted to measurement results by gradient descent methods. For example, this allows us to improve the results of QAE for determining the probabilities of the failure of nodes in a network. We implement amplitude estimation for the quantum model on a simulator

with a realistic hardware noise model as well as on real hardware from AQT<sup>1</sup>. The result shows, that low-depth versions of QAE are currently feasible for small networks with one or two nodes and up to three or four time steps.

To summarize, the main contributions of this paper are the following:

1. The implementation of the time evolution of a probabilistic network model as a quantum circuit.
2. The execution of low-depth quantum amplitude estimation (QAE) of the model on a physical quantum computer: AQT's PINE system.
3. Evaluating the low depth QAE results with a simple noise model to improve the estimation of the probabilities, which we want to determine.

## 2 The network model

In this section, we define the network model. For a simple example, we show the results of a direct calculation of probabilities and compare these to the results of classical Monte Carlo evaluations. The exact and Monte Carlo evaluations for the network model are implemented in the Pygrnd library [20] and both methods are explained in detail in appendix A and B.

### 2.1 Definition of the model

We model a network by a set  $N = \{1, \dots, k\}$  of  $k$  nodes. We consider  $T$  time steps and for each time step  $t \in \{1, \dots, T\}$  a node  $n \in N$  can have the state 0 or 1. We use the functions  $s_n(t)$  for  $n \in N$  and  $t \in \{1, \dots, T\}$  to describe the state of node  $n$  in time step  $t$ . We set  $s_n(t) = 0$  if node  $n \in N$  is good and  $s_n(t) = 1$  if the node has failed. A configuration  $c \in \{0, 1\}^k$  at a time step is the sequence  $c = (s_k(t), \dots, s_1(t))$  of the states of all nodes<sup>2</sup>. We write  $p_c(t)$  for the probability for configuration  $c$  in time step  $t$ .

For each node  $n$ , we have the intrinsic probability  $p_n^{\text{fail}}$  to fail in a time step if the node was good in the previous time step. Each node has also the probability  $p_n^{\text{recover}}$  to recover if the state was 1 in the previous time step. Furthermore, if a node  $m \in N$  has failed in time step  $t$  then it might trigger the default of another node  $n \in N$  in the next time step with a certain probability. We write this probability as  $p_{m,n}^{\text{trigger}}$ .

We now complete the description of the time evolution of the system. When we assume that a node  $n \in N$  has failed in time step  $t$ , then we only consider the recovery probability for setting  $s_n(t+1)$  and no intrinsic or trigger probabilities are taken into account. If a node  $n \in N$  is good in time step  $t$ , then we set it to the failed state with the probability  $p_n^{\text{fail}}$ . We also iterate through all nodes  $m \in N$  in the model with  $p_{m,n}^{\text{trigger}} > 0$  and we set  $n$  to the failed state with this probability. If in this procedure the state turns to the failed state, then we do not set it to good again in this time step.

Note that the states in a time step depend only on the states of the previous time step. This separation of time steps allows us to have arbitrary dependencies between the

<sup>1</sup>AQT's quantum computer system PINE.

<sup>2</sup>We use this order of elements to be consistent with the qubit ordering of Qiskit.

nodes without ambiguities, e.g. it is possible that two nodes can trigger each other. We assume that time step  $t = 1$  is the initial time step and that all nodes have the state 1 with the corresponding intrinsic probability of failure. This can be seen as the result of a time step when for  $t = 0$  all nodes are in state 0.

## 2.2 Network example

In this section, we define the example that we use for most of the following discussions. We consider a model with two nodes and three time steps, i.e. we have  $k = 2$  and  $T = 3$ . The probabilities of the model are defined in table 1. We would like to find the probabilities for  $p_c(t)$  for the configurations  $c \in \{00, 10, 01, 11\}$  and time step  $t = 3$ . These are calculated in the following sections with a classical calculation of the probabilities and Monte Carlo methods and with a quantum circuit.

Table 1: The parameters of the example with 2 nodes.

node	$p_n^{\text{fail}}$	$p_n^{\text{recover}}$	$p_{m,n}^{\text{trigger}}$	$n = 1$	$n = 2$
1	0.2	0.3	$m = 1$	-	0.2
2	0.7	0.8	$m = 2$	0.8	-

## 2.3 Classical evaluation of the model

Following the rules of section 2.1, we can evaluate the probabilities of configurations in a time step with an exact calculation. The method is described in appendix A. Although the implemented method is not efficient and cannot be used for models with many nodes and time steps, we can use it to determine the probabilities of configurations for small examples. This allows us to assess the precision of the results of Monte Carlo methods in section 2.4 and to evaluate the performance of the quantum circuits that we define in section 3. The results<sup>3</sup> of the evaluation for the example of section 2.2 with 2 nodes for all time steps up to 3 are shown in table 2.

Table 2: Time evolution of the probabilities of the configurations of the network model from section 2.2 with 2 nodes.

t	$p_{00}(t)$	$p_{10}(t)$	$p_{01}(t)$	$p_{11}(t)$
0	1.0	0.0	0.0	0.0
1	0.240	0.560	0.060	0.140
2	0.167	0.174	0.479	0.179
3	0.140	0.219	0.308	0.333

---

<sup>3</sup>In this paper, we round all numerical values to three decimal places to simplify the notation.

## 2.4 Monte Carlo evaluation of the model

Besides the exact evaluation method of section 2.3, we can obtain the probabilities  $p_c(t)$  for each possible configuration  $c \in \{0,1\}^k$  after  $t$  time steps by Monte Carlo simulations of the model. A possible implementation is outlined in appendix B.

The results of several Monte Carlo evaluations for the example of section 2.2 can be seen in figure 1. It shows the results for calculating  $p_c(t)$  for all configurations  $c \in \{0,1\}^2$  for time step  $t = 3$ . The Monte Carlo evaluations were done with  $10^3$ ,  $10^4$ ,  $10^5$  and  $10^6$  runs. For each number of runs, we performed 20 independent calculations and the stability of the results depending on the number of runs can be seen in the spread of the data points. The horizontal lines correspond to the probabilities that are calculated by the method described in section 2.3.

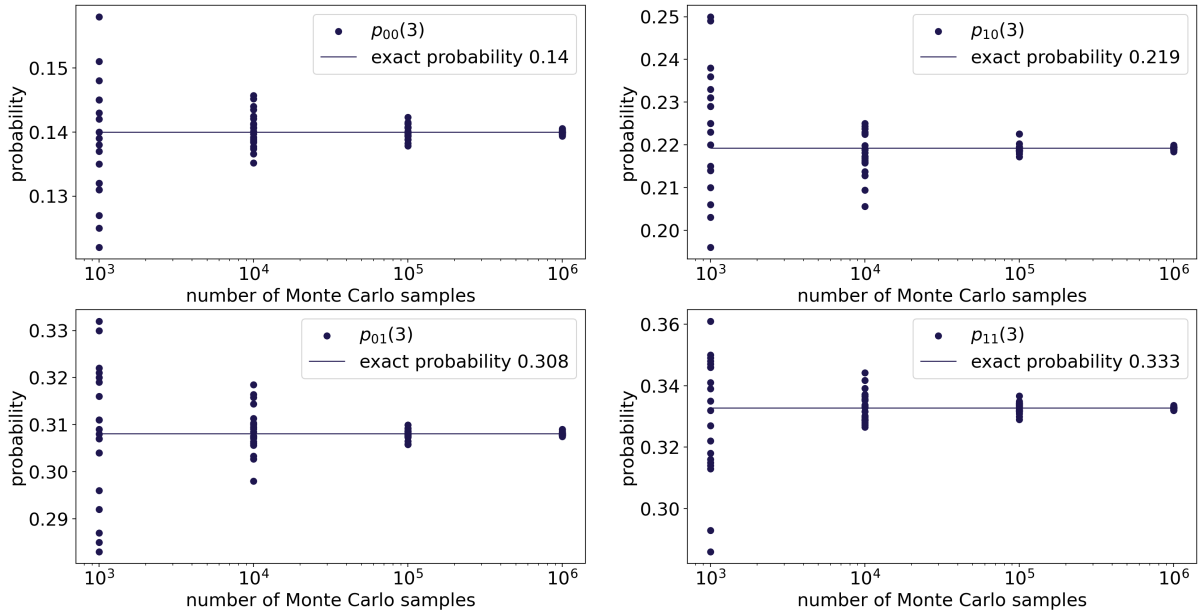


Figure 1: Results of Monte Carlo simulations for the example of section 2.2. The probabilities of the configurations calculated by the classical method can be found in table 2.

## 3 Time evolution of networks as quantum circuits

The network model of section 2 can be used to construct quantum circuits, which reproduce the probabilities  $p_c(t)$  for all configurations  $c \in \{0,1\}^k$  and for all time steps  $t$ . The definition of the networks and the transitions between time steps allows us to build quantum circuits for any network topology, including cycles in the dependency graph of the nodes.

### 3.1 Construction of quantum circuit for a model

A network with  $k$  nodes is represented by  $k$  qubits for each time step. We write  $q_n(t)$  for the qubit corresponding to node  $n \in N$  in time step  $t$ . The states  $|0\rangle$  and  $|1\rangle$  of

the qubit correspond directly to the states 0 and 1 of a node. We consider an iterative construction with an operator  $U_{\text{init}}$  to initialize a register and an operator  $U_{\text{time}}$  that connects two registers for consecutive time steps  $t$  and  $t + 1$  and that calculates the probability distribution for  $t + 1$ . The structure of the circuit corresponding to a model with  $k$  nodes and 3 time steps after the initial time step is shown in figure 2.

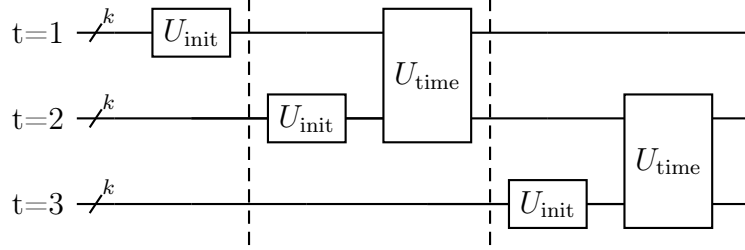


Figure 2: A circuit that implements three time steps after the initial time step for a network with  $k$  nodes, i.e. we have  $t = 3$ . The unitary  $U_{\text{init}}$  initializes the  $k$  qubits corresponding to the intrinsic probabilities  $p_n^{\text{fail}}$  to fail for each node. The unitary  $U_{\text{time}}$  implements the operator for one time step and it generates the states for the output on the lower register.

The operator  $U_{\text{init}}$  initializes a register for a time step with the intrinsic probabilities  $p_n^{\text{fail}}$  for each node  $n \in N$ . After applying this operator, each qubit would be measured in the state 1 with probability  $p_n^{\text{fail}}$ . A circuit for this consists of an  $R_y(\theta_n)$  gate<sup>4</sup> on each qubit as shown in figure 3. A probability of  $p_n^{\text{fail}}$  corresponds to the angle

$$\theta_n = 2 \arcsin \left( \sqrt{p_n^{\text{fail}}} \right). \quad (1)$$

The operator  $U_{\text{time}}$  acts on two consecutive registers. The first register corresponds to time step  $t$  and the second register corresponds to time step  $t + 1$ . We use controlled operations to change the qubits corresponding to the nodes.

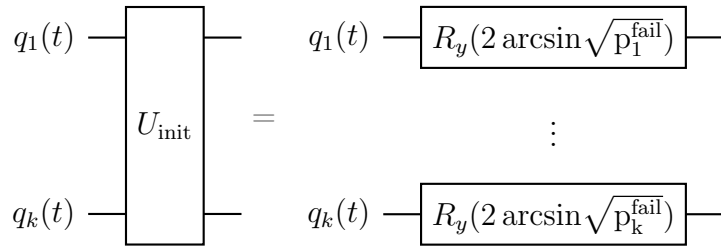


Figure 3: The implementation of  $U_{\text{init}}$  on the register with  $k$  qubits for time step  $t$ . When we measure the generated state of this gate then we obtain the result 1 with probability  $p_k^{\text{fail}}$  to fail for each node  $k$ . We use the notation of Qiskit for the gates  $R_y(\theta)$ .

The operator  $U_{\text{time}}$  can be constructed as follows:

---

<sup>4</sup>We use the notation of Qiskit [21] for the  $R_y(\theta)$  gates.

- If the qubit  $q_n(t)$  corresponding to node  $n$  is in state 1 in time step  $t$ , then we apply the operation  $R_y(\theta)$  with

$$\theta = 2 \arcsin \left( \sqrt{1 - p_n^{\text{recover}}} \right) - 2 \arcsin \left( \sqrt{p_n^{\text{fail}}} \right) \quad (2)$$

to the corresponding node in time step  $t + 1$ . This can be done with an 1-controlled  $R_y$  operation, i.e. the operation is performed if the control qubit is in state 1. After this operation, if the qubit for node  $n$  is in state 1 in time step  $t$  then the probability to measure it in state 0 is  $p_n^{\text{recover}}$  for a given configurations of the previous time steps as expected from the classical model. Note that the second part of the angle reverts the rotation of a qubit that is performed by  $U_{\text{init}}$ .

- If a node  $n$  is in state 0 in time step  $t$ , then we apply several controlled operations  $R_y(\theta)$  on qubit  $q_n(t + 1)$ . We define the set

$$M = \{m \in N : p_{m,n}^{\text{trigger}} > 0\} \quad (3)$$

of nodes with non-zero probability to trigger node  $n$  and we consider all possible configurations  $c \in \{0, 1\}^{|M|}$  of the nodes in  $M$ . For a configuration we calculate the probability  $p_{\text{off}}$  that  $n$  stays in state 0 by the product of the probabilities  $1 - p_{m,n}^{\text{trigger}}$  for all  $m \in M$  and  $1 - p_n^{\text{fail}}$ , i.e. the node is not triggered by another node and also not intrinsically. Then we add an  $R_y(\theta)$  gate on qubit  $q_n(t)$  with angle

$$\theta = 2 \arcsin(\sqrt{1 - p_{\text{off}}}) - 2 \arcsin(\sqrt{p_n^{\text{fail}}}) \quad (4)$$

and the operation is controlled by the state 0 or 1 of qubit  $q_m(t - 1)$  depending on the state of the configuration  $c$ . After this operation, if the qubit for node  $n$  is in state 0 in time step  $t$  then the probability to measure it in state 1 is  $1 - p_{\text{off}}$  for the given configurations of the previous time steps as expected from the classical model.

The construction that is outlined above can be optimized in several places, e.g. the separation between  $U_{\text{init}}$  and  $U_{\text{time}}$ , which makes the description of the construction simpler, is not necessary and introduces controlled operations in  $U_{\text{time}}$  that revert operations of  $U_{\text{init}}$ .

### 3.2 Quantum circuit for example

For the example of section 2.2, the operator  $U_{\text{time}}$  is shown in figure 4: The first gate corresponds to the recovery probability of the first node and the angle is determined by

$$2 \arcsin \left( \sqrt{1 - p_1^{\text{recover}}} \right) - 2 \arcsin \left( \sqrt{p_1^{\text{fail}}} \right) \quad (5)$$

$$= 2 \arcsin \left( \sqrt{0.7} \right) - 2 \arcsin \left( \sqrt{0.2} \right) = 1.055. \quad (6)$$

The second controlled gate with the angle  $\theta_2$  corresponds to the first node that is in state 0 in time step  $t$  and is triggered in time step  $t + 1$  by the second node when its state is 1. The probability that node 1 is not triggered is

$$(1 - p_1^{\text{fail}})(1 - p_{2,1}^{\text{trigger}}) = (1 - 0.2)(1 - 0.8) = 0.16. \quad (7)$$

Therefore, the probability that node 1 is triggered is 0.84 and this leads to the angle

$$\theta_2 = 2 \arcsin(\sqrt{0.84}) - 2 \arcsin(\sqrt{0.2}) = 1.391. \quad (8)$$

The values for  $\theta_3$  and  $\theta_4$  can be calculated by replacing node 1 with node 2 and vice versa. The full circuit for the example is shown in figure 5. When we measure the last two qubits of the circuit then we obtain the results that are shown in figure 6. The probabilities from the measurements and the probabilities of the classical calculations are the same up to the deviations that we expect, because we only performed 1000 repetitions, i.e. we executed the circuit 1000 times on an error-free simulator. With an increasing number of repetitions, the probabilities from the measurements would get closer to the exact values.

If we run the circuit many times and calculate the proportions of the result configurations then we are not more efficient than classical Monte Carlo methods in terms of model evaluations. However, the quantum circuit allows us to use QAE or low-depth QAE methods to evaluate the probability of a configuration with less evaluations of the model on a quantum computer. This is described in the following section.

## 4 Quantum amplitude estimation

We can use quantum amplitude estimation (QAE, see [5]) to measure the configuration probabilities of the network time evolution. On error free hardware, this would lead to a quadratic speedup compared to Monte Carlo simulations in the number of model evaluations. In the following, we consider the standard version of QAE and a low-depth version of it for the example with 2 nodes and 3 time steps that is introduced in section 2.2. The low-depth version is more suited for Noisy Intermediate-Scale Quantum (NISQ) devices as it does not use controlled versions of the Grover operators.

### 4.1 Construction of the Grover operator

The standard version of the quantum amplitude estimation is a phase estimation of the eigenvalues of a Grover operator. In our case, the network and its time evolution is given by a unitary  $U$  that produces the probabilities of configurations after a number of time steps. Based on this operator, we can construct the Grover operator by  $G = -US_0U^\dagger S_\chi$  where  $S_0$  applies the phase  $-1$  to the state  $|0\dots 0\rangle$  and  $S_\chi$  applies the phase  $-1$  to all

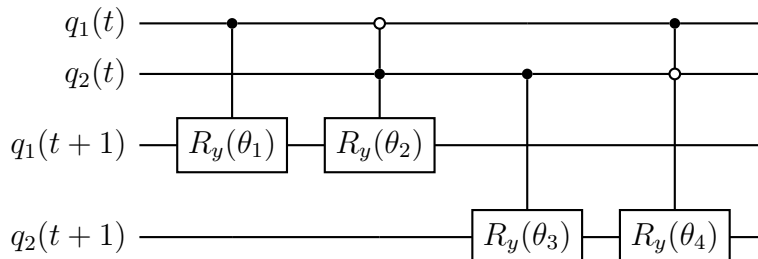


Figure 4: The operator  $U_{\text{time}}$  for the example from section 2.2. Here, we have the parameters  $\theta_1 = 1.055$ ,  $\theta_2 = 1.391$ ,  $\theta_3 = -1.055$  and  $\theta_4 = 0.135$ .



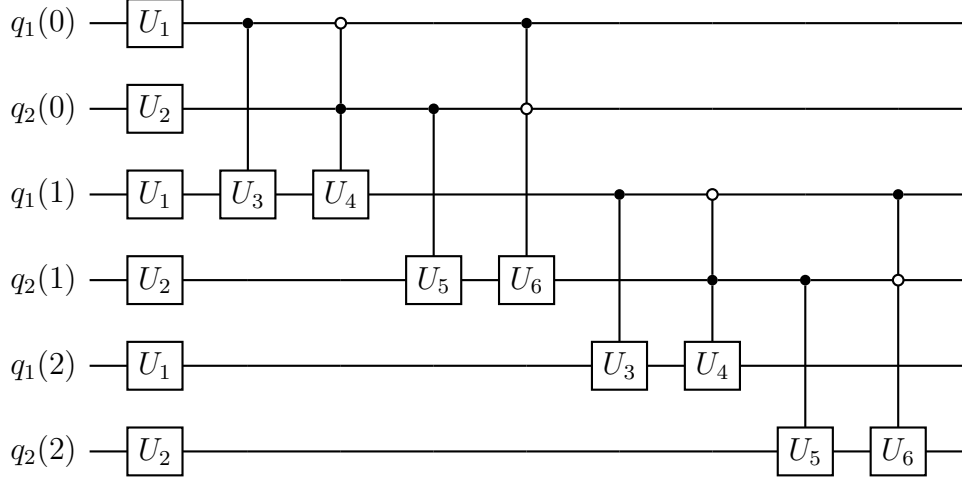


Figure 5: The quantum circuit for the network in section 2.2. We use the unitaries  $U_1 = R_y(0.927)$ ,  $U_2 = R_y(1.982)$ ,  $U_3 = R_y(1.055)$ ,  $U_4 = R_y(1.391)$ ,  $U_5 = R_y(-1.055)$  and  $U_6 = R_y(0.135)$ . The operators  $U_{\text{init}}$  are moved to the beginning of the circuit to make the drawing of the circuit more compact.

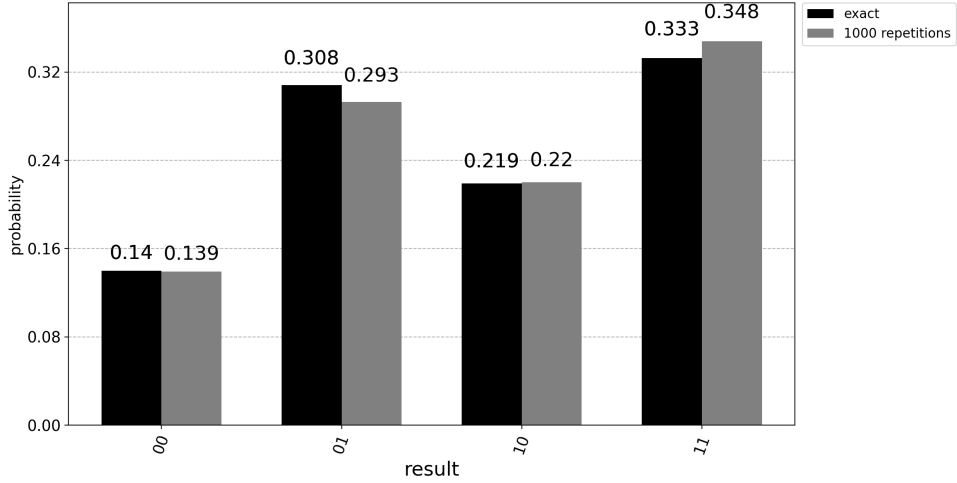


Figure 6: Result of the quantum circuit of figure 5 on an error-free simulator when we measure the qubits  $q_1(2)$  and  $q_2(2)$ . The probabilities of the measurement results correspond to the probabilities of the configuration of the model in time step 2.

Table 3: The eigenvalues of the Grover operators corresponding to the 4 possible configurations 00, 01, 10 and 11 for time step  $t = 3$ .

$c$	$\lambda_{\pm}$	$\theta$	$p_c(3)$
00	$0.720 \pm 0.694i$	0.767	0.140
01	$0.384 \pm 0.923i$	1.177	0.308
10	$0.562 \pm 0.827i$	0.975	0.220
11	$0.335 \pm 0.942i$	1.230	0.333

states that we search. For instance, if we want to find out the probability of a configuration  $c \in \{0, 1\}^k$  after  $t$  time steps, then we mark all states  $|c(1) \dots c(t)\rangle$  with  $c(t) = c$  with the phase  $-1$ . This phase operation acts only on the register corresponding to time step  $t$ .

For our example from section 2.2, we have 6 qubits for 2 nodes and 3 time steps after the initial configuration. We consider the Grover operators for all 4 possible configurations 00, 01, 10 and 11 after the last time step. The eigenvalues of the Grover operators<sup>5</sup> besides  $+1$  and  $-1$  are given in table 3. We can write

$$\lambda_{\pm} = \cos(\theta) \pm i \cdot \sin(\theta) \quad (9)$$

with the values in table 3 and we obtain the corresponding probabilities

$$p_c(t) = \sin^2(\theta/2). \quad (10)$$

We see that for each configuration  $c$  the non-trivial eigenvalues of the Grover operator lead to the correct probability of the configuration.

## 4.2 Standard quantum amplitude estimation

We can use the standard version of QAE to estimate the probability of the states that are marked by the operator  $S_{\chi}$ . The circuit for an example for the standard version of QAE is given in figure 7.

The unitary operation  $U$  is the model and it is used to initialize the registers, on which the controlled Grover operators  $G^{\ell}$  act. The measurements after the Fourier transform lead to a binary encoding of the probability, which we want to find. For an increasing number of qubits we obtain an increasingly better approximation of the result. For a resolution of three qubits as in figure 7 we obtain eight possible binary results along with the angles  $\theta$  and the corresponding probabilities  $\sin^2(\theta/2)$  that are shown in table 4.

In figure 8, we show the results of a QAE depending on the number of qubits that are used for the resolution of the output. The resulting probabilities are the values that correspond to the binary result with the highest number of counts. We see that with an increasing number of qubits the results are getting closer to the exact value.

The disadvantage of this method for amplitude estimation is that we need controlled operators, which are the powers of Grover operators. Without finding an exploitable structure of a Grover operator, we can only apply a Grover operator  $\ell$  times for performing the operator  $G^{\ell}$ . This means that for a QAE with  $b$  qubits of precision we have in total

---

<sup>5</sup>We used the function `numpy.linalg.eig` to calculate the eigenvector from the unitary, which we obtained with the unitary simulator of quantum circuits from Qiskit.

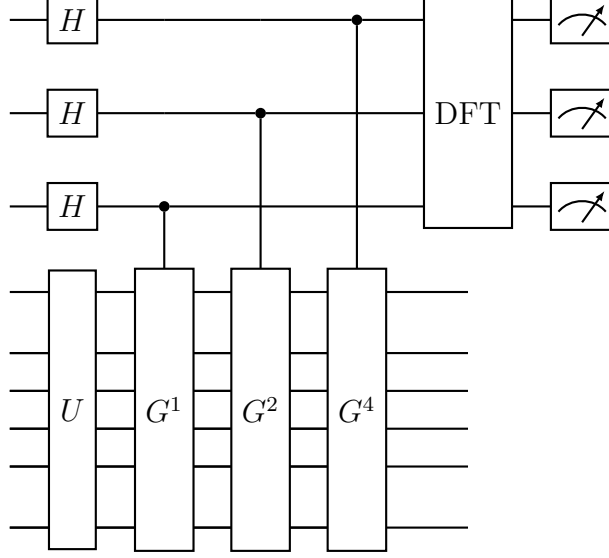


Figure 7: The quantum circuit for a standard QAE with a resolution of 3 qubits. Here, the unitary  $U$  is the model of the network and  $G$  is the corresponding Grover operator that marks all states depending on the configuration, which we want to analyze.

Table 4: The binary results with their corresponding angles  $\theta$  and probabilities for a QAE with 3 bits resolution.

res	000	001	010	011	100	101	110	111
$\theta$	0.0	0.785	1.571	2.356	3.142	3.927	4.712	5.498
prob	0.0	0.146	0.500	0.854	1.0	0.854	0.500	0.146

$2^b - 1$  controlled Grover operators in the circuit. Each Grover operator contains the model and its inverse along with additional operators to mark states with the phase  $-1$ . The model typically contains operations on qubits that are controlled by several control qubits and the controlled version of the Grover operator adds an additional qubit to this. As a consequence, the implementation of a QAE for even very simple models and for very few time steps on currently available hardware appears to be impossible.

For small circuits, the replacement of operators, which have several control qubits, with more elementary gates adds a significant overhead to a circuit. For circuits with many control qubits, the problem is less severe as there are techniques to reduce gates with several control qubits to more elementary gates with a linear overhead [22].

An alternative to the QAE is given by low-depth versions of QAE. These methods try to avoid the additional control qubit for the Grover operators and they trade this reduction of quantum complexity for a higher post-processing complexity. One of such methods is described and applied in the following section.

### 4.3 Low-depth quantum amplitude estimation

In this section we use a method for amplitude estimation that is inspired by [12] to reduce the necessary hardware requirements. The main advantage of this construction is

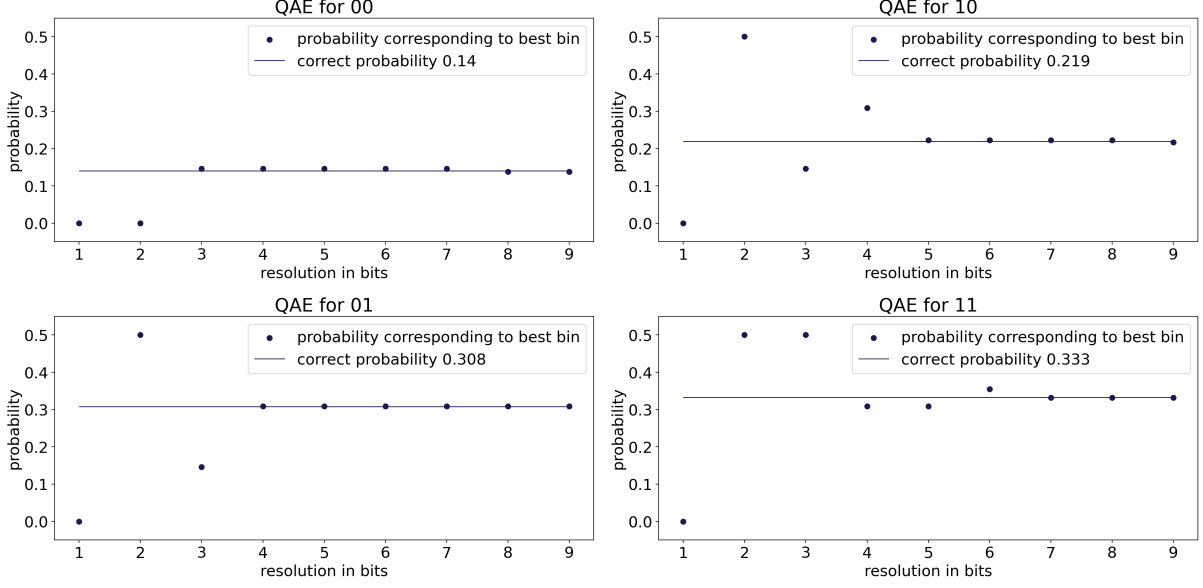


Figure 8: The probabilities obtained by QAE depending on the number of qubits of the precision.

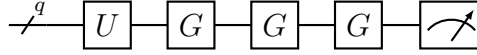


Figure 9: Circuit for low-depth QAE for a model  $U$  and a Grover operator  $G$  on a register with  $q$  qubits. The number of Grover operators depends on the chosen specific low-depth method. We use the measurement results to determine the proportion of states, which are marked by  $S_\chi$  of the Grover operator.

that controlled versions of the Grover operators are not needed. Besides this, we also do not need the additional qubits for the controlled operations and the Fourier transform. We execute several Grover operators directly on a qubit register after an initialization with the model and measure the number of good results. The fact that we do not need controlled Grover operators makes it reasonable to run such quantum circuits on current hardware.

A simple schematics for a low-depth QAE is shown in figure 9. As an example, we calculate the probabilities  $p_c(t)$  of section 2.2 of all four possible configurations after 3 time steps. For this, we use the 4 different Grover operators from section 4.1 and we run the series of experiments for each case on an error-free simulator. We execute the Grover operators  $G^\ell$  for all  $\ell \in \{0, 1, \dots, 8\}$  after the initialization of the register with  $U$  and for each such power we run the circuit 30 times<sup>6</sup>. In table 5 we list the obtained counts  $m_\ell$  for all configurations.

For instance, the value of  $m_2$  denotes the number of measurements of marked configurations out of the 30 repetitions when we apply two Grover operators after the initialization. For an angle  $\theta$ , which we want to determine, and the power  $\ell$  of a Grover operator

<sup>6</sup>Note that this simple choice of the powers of the Grover operators and the constant number of repetitions is chosen to give a good overview of the method.

Table 5: The counts for the low-depth QAE with 0 to 8 Grover operators after the initialization with  $U$  for all possible configurations of the example with 2 nodes and 3 time steps of section 2.2 on a noise-free simulator.

c	$m_0$	$m_1$	$m_2$	$m_3$	$m_4$	$m_5$	$m_6$	$m_7$	$m_8$	$\theta$	$p_c(3)$
00	5	29	26	5	4	21	27	6	6	0.390	0.144
01	8	27	2	22	19	0	30	10	7	0.588	0.308
10	6	29	10	1	25	18	0	24	25	0.487	0.219
11	7	28	0	22	16	7	30	2	23	0.612	0.330

we obtain the corresponding probability

$$\sin^2((2\ell + 1)\theta/2) \quad (11)$$

for measuring the good result. Therefore, for any angle  $\theta$  we can compare the deviation of the expected values  $m_\ell$  from the measured results. This deviation can be used to perform a gradient descent search for the angle  $\theta$ . The angles and probabilities in table 5 were obtained by this method<sup>7</sup>. In figure 10 we show the measured points together with the probabilities based on the correct angle  $\theta$ , which we derived from the exact classical calculation. The results show that the method works for an error-free simulation of the quantum circuits for low-depth QAE.

Note that the function of equation (11) has two solutions for each probability value, e.g. we obtain the probability 0.3 for  $\theta = 1.159$  and  $\theta = 1.982$ . If  $\theta$  is an angle, then  $2\pi - \theta$  is the other angle that leads to the same probability. In the following, we always take the smaller angle of both.

#### 4.4 Low-depth quantum amplitude estimation with noise

We can try to extract a useful result from the output of a noisy quantum computer by using our knowledge of the output functions. An example for such an approach, which differs from ours, was presented in [26]. The authors derived the shape of the distributions of the measurement results for low-depth QAE on a noisy machine by assuming that each rotation angle of a Grover picks up a normally distributed error. Other results for the output of low-depth QAE on noisy hardware were presented in [23] for parallel versions of QAE. An approach to noise that is similar to the one in this section was discussed in [16] for three different low-depth QAE algorithms. Two of those algorithms were tested on quantum hardware in [27].

Here, we want to model the number of results, which are marked by  $S_\chi$ , that we measure at the end of a noisy low-depth QAE circuit. We start from the assumption, that even a single error in a Grover operator will cause the output of the Grover operator to be essentially random. Therefore, when one or more errors occur, we expect that the ratio of marked states, which we measure at the end of the circuit, is the fraction  $f$  of states, which are marked by  $S_\chi$ , among all possible states.

<sup>7</sup>The corresponding functions for the gradient descent method are implemented in the Pygrnd module `pygrnd.qc.lowDepthQAEgradientDescent`.

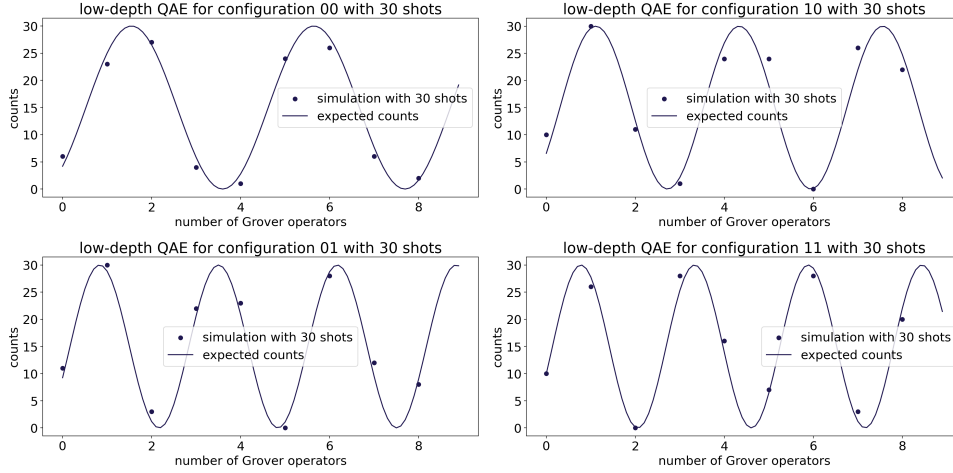


Figure 10: The results of the low-depth QAE with 30 repetitions for  $\ell$  Grover oracles with  $\ell \in \{0, \dots, 8\}$  on an error-free simulator. We consider the Grover oracles for the configurations 00, 01, 10 and 11 separately. The solid curves are  $\sin^2((2\ell + 1)\theta/2)$  for the correct values of  $\theta$  from the exact classical calculation. We show the correct values for real-valued  $\ell$  instead of integer values to show the origin of the values more clearly.

Furthermore, we assume that for circuits of the low-depth QAE as in figure 9 with only the  $M$  operator and no Grover operators the effect of noise can be neglected, i.e. we expect the results from equation (11). We also know that the probability that at least one error occurs during the execution of a circuit grows exponentially with the gate count. Therefore, we add an exponential decay factor to the expected output probability of equation (11). For a circuit QAE with angle  $\theta$  and  $\ell$  Grover operators we denote by  $r(\theta, \ell, a, f)$  the expected probability for measuring a marked result, where  $a$  approximates the probability of one Grover operator to incur an error when executed. This probability is given by the following equation:

$$r(\theta, \ell, a, f) = e^{-a\ell} \sin^2((2\ell + 1)\theta/2) + (1 - e^{-a\ell})f \quad (12)$$

We can use equation (12) to fit the output of a quantum computer and extract the angle  $\theta$  and therefore the probability to find good states. We apply this method to an example in section 4.6.

The expected benefits of using this model of a dampened oscillation over the naive approach of fitting the results to a pure sine wave are two-fold: First, the fitter should have an advantage due to the fact that the data points are generally closer to the fit, due to the dampening. Second, the dampening changes the wave length of the oscillation and this introduces an error in the naive fitting. The wavelength we find is too long and the corresponding probabilities are too low. For example, this effect can be seen in figure 13). With the error model, this problem should not arise.

## 4.5 Implications of the noise model

It is instructive to think of the simple error model of section 4.4 as the sum of two different error models. The first one determines if one or more errors do occur in a circuit. This is

modeled by the  $e^{-a\ell}$  terms in equation (12). The second model determines the output in case there is an error. If we assume that an error makes the output completely random, which is the simplest possible approximation, then the factor  $f$  in equation (12) is the expectation value of our second error model.

When we explicitly model these errors as a sum over Bernoulli random variable with expected value  $f$ , then we see that the factor  $f$  leads to the standard deviation

$$\sigma = \sqrt{\frac{f(1-f)}{N}}. \quad (13)$$

We want to determine the number  $\ell$  of consecutive Grover operators, which we can expect to execute and which allows us to still measure a useful signal at the end of the circuit. This number depends on the noise level and we can simply compare the amplitude  $\max_{\ell} e^{-a\ell}$  of the signal in equation (12) with the mean absolute deviation  $\bar{\sigma}$ , which is

$$\bar{\sigma} = \sqrt{\frac{2}{\pi}} \sigma, \quad (14)$$

where  $N$  denotes the number of repetitions, which we perform for a circuit. If the mean absolute deviation is bigger than the amplitude of the signal, then we cannot hope to extract the sine wave from our measurements anymore. This means that if the inequality

$$\sqrt{\frac{2}{\pi}} \cdot \sqrt{\frac{f(1-f)}{N}} < e^{-a\ell} \quad (15)$$

does not hold, then we cannot expect to measure useful results. Solving this for  $N$  tells us how many repetitions we need to find a useful signal for a given noise level  $a$  and number of Grover operators  $\ell$ . If we solve it for  $\ell$ , we find how many Grover operators we can have in a low-depth QAE for a given noise level  $a$  and number of repetitions  $N$ .

We can now try and use those insights to obtain optimal results on actual quantum hardware. This means building a protocol which, after the initial circuit with just the model evaluation, determines the optimal number of Grover operators and repetitions for the next experiment, based on the results known so far. We can also determine an estimation for an upper limit for the circuit with the largest number of Grover operators already after the second experiment. Some results related to such ideas have been worked out in [16].

## 4.6 Evaluation of results from simulator with noise model

The example of section 2.2 with 2 nodes and 3 time steps is too complex for current hardware due to the need of several gates in each Grover operator that are controlled by more than one qubit. For most of the currently available hardware types, these operators have to be decomposed into single-qubit operators, which are uncontrolled or which are controlled by a single qubit [22]. This decomposition introduces an overhead, which makes the evaluation of the low-depth QAE infeasible.

We analyze the behavior of the low-depth version of QAE as described in section 4.4. For this, we consider a simpler network and we run it on the AQT simulator [24], which uses a realistic noise model for ion-trap quantum computers [25]. The model has only

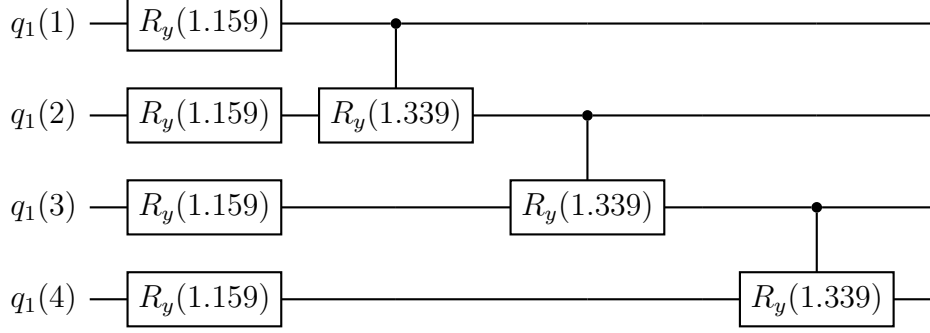


Figure 11: Circuit for the network model with 1 node and 3 time steps after the initialization. We are interested in the probability that the qubit  $q_1(4)$  is in state 1.

Table 6: The probabilities that we have state 1 after  $t$  time steps for the example with one node from the exact classical calculation.

t	1	2	3	4
probability	0.300	0.480	0.588	0.653

one node and we consider up to 3 time steps after the initial step. The node has the probabilities  $p_1^{\text{fail}} = 0.3$  and  $p_1^{\text{recover}} = 0.1$ . The probability that the node is in state 1 after  $t$  time steps is shown in table 6. The quantum circuit for the model is shown in figure 11.

We construct the Grover operators for all time steps  $t \in \{1, 2, 3, 4\}$  and we run the low-depth version along with the gradient descent search for the parameters  $a$  and  $f$  and the probabilities as described in section 4.4. We use the AQT simulator with noise and we make 2000 repetitions for each operator and each number of Grover operators in the low-depth circuit. The results are shown in table 7.

Table 7: The counts for the low-depth QAE with 0 to 8 Grover operators after the initialization with  $U$  for the example with 1 node and different time steps on the AQT simulator with noise model with 2000 repetitions for each  $t$  and  $m_\ell$ . The values  $a$  and  $f$  and the probabilities result from the gradient descent when we have the measured values  $m_\ell$ .

t	$m_0$	$m_1$	$m_2$	$m_3$	$m_4$	$m_5$	$m_6$	$m_7$	$m_8$	a	f	prob
1	617	1958	121	1265	1516	12	1817	928	344	-0.001	0.003	0.300
2	981	1181	760	1365	595	1384	717	1283	807	0.162	0.513	0.459
3	1236	841	1153	931	1033	988	983	1018	979	0.977	0.504	0.596
4	1423	969	991	1026	979	1002	983	1012	986	1.813	0.506	0.703

We see that for one or two time steps the values for the probability are accurate. However, for 3 and 4 time steps the exponential error factor  $a$  is much higher and this shows that there is a significant amount of error. This can be explained by the fact that in the circuit with 3 time steps we have the operator  $S_0$  that is an  $X$  gate that is controlled by two qubits. After compiling to 1- and 2-qubit gates, this leads to a much longer circuit



compared to the circuit for 2 time steps. The measurement results with the values of equation (11) for the correct values without error model, which are calculated with the exact classical method, are shown in figure 12 along with the values of equation (12) for the values, which we found with gradient descent for the error model.

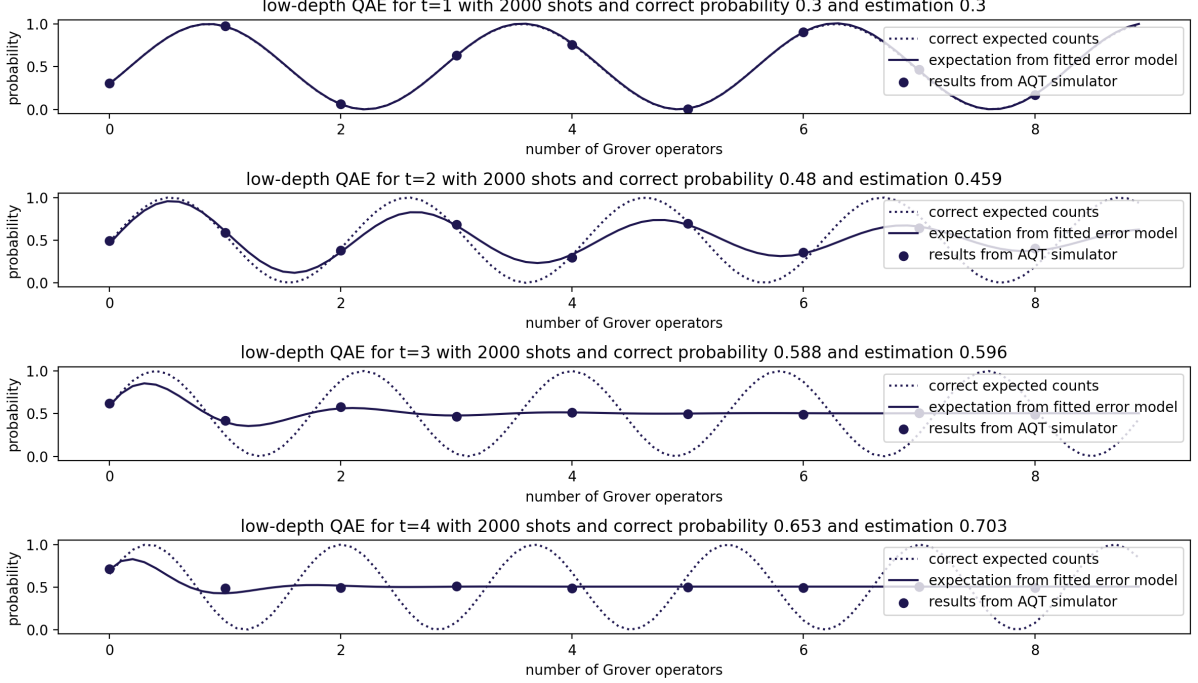


Figure 12: Measurement results (simulator with noise model) for the model with 1 node (filled circles) and expected probabilities to find the state 1 for the values, which are calculated by the exact classical method (dotted curves). The solid curves are the expected probabilities from equation (12) with values  $\theta$ ,  $a$  and  $f$  after gradient descent optimization. For  $t = 1$ , only one curve can be seen as both are very close.

## 4.7 Evaluation of results from hardware

In addition to the evaluation of a network model on a simulator in section 4.6, we also evaluated the model on the AQT ion trap quantum computer PINE. The PINE system was configured to work with a register of 8 qubits for our implementation. We used the Qiskit function `transpile` with optimization level 3 to convert the circuits into native gates for this machine.<sup>8</sup> For  $t = 1$  and  $t = 2$  we considered up to 8 Grover operators after the initialization and for  $t = 3$  and  $t = 4$  we used up to 4 Grover operators. For each number  $t$  of time steps and number  $\ell$  of Grover operator we gathered statistics from executing a circuit 8000 times. We used the simple error model from section 4.4 to obtain an estimation of the probabilities. The counts and the values  $a$  and  $f$  of the model, which we fitted with gradient descent, and the resulting estimated probabilities to find the node in state 1 after  $t$  time steps are shown in table 8.

<sup>8</sup>With this optimization level, the circuits for  $t = 1$  consist of a single uncontrolled gate.

Table 8: The counts for the low-depth QAE with 0 to 8 Grover operators for  $t = 1$  and  $t = 2$  and with 0 to 4 operators for  $t = 3$  and  $t = 4$  after the initialization with  $U$  for the example with 1 node on the AQT system PINE. The values  $a$  and  $f$  and the probabilities result from the gradient descent when we have the given values  $m_\ell$ . The values in the column  $p_{\text{cor}}$  are the results from the exact calculation.

t	$m_0$	$m_1$	$m_2$	$m_3$	$m_4$	$m_5$	$m_6$	$m_7$	a	f	prob	$p_{\text{cor}}$
1	2386	7867	494	5107	6283	84	7450	3689	-0.006	0.060	0.300	0.300
2	4545	4275	3133	5118	3110	5316	1639	6446	-0.076	0.526	0.487	0.480
3	5211	2505	5171	2336	5353	-	-	-	0.286	0.476	0.581	0.588
4	5609	2695	4678	3510	4171	-	-	-	0.884	0.497	0.664	0.653

We see that the estimation based on the error model is quite close to the correct values from the exact classical evaluation, which are in the first table in section 4.6. The data points, the exact values and the function of the simple error model, which is fitted with gradient descent, are shown in figure 13. Note that for  $t = 2$ , we have negative probabilities<sup>9</sup> from the model after fitting. Despite this result, the estimation of the probability is very close.

We tried to improve the fitting of the error model to the data by setting  $f = 0.5$ , which corresponds to the proportion of states, which are marked by  $S_\chi$ , to all states, but this did not improve the estimation.

As can be seen from table 8, for  $t = 3$  and  $t = 4$ , our noise model results were within about 1 and 2 percentage points of the exact results. In contrast to this, without noise model we could not even get to within 5 and 14 percentage points (for details see the notebook `probabilisticNetworks.ipynb` in Pygrnd [20]). This illustrates the value added by the noise model.

## 5 Conclusion and outlook

We showed how a simple model for the discrete time evolution of networks can be formulated as quantum circuits. We used the circuits to apply quantum amplitude estimation and a low-depth version of it to find the probabilities of different network configurations and their time evolution for simple examples. We introduced a simple error model for the expected measurement results on noisy hardware, which can be used to improve the post-processing of the measurement results. We used this approach to analyze a simple example on a simulator with a realistic noise model as well as on real hardware.

It is straightforward to extend the model to more dependencies between the nodes, e.g. a recovered node could lead to the recovery of another node with a certain probability. Another useful extension would be to allow dependencies over several time steps.

It is encouraging to see that already today, simple but non-trivial models can be evaluated not just on simulators, but on real life quantum computing hardware, AQT’s PINE

---

<sup>9</sup>The assumption that the complexity of the circuits increases with the number of Grover operators does not hold for  $t = 1$  and  $t = 2$ . The reason for this is that the optimizer is able to reduce the number of gates to a constant number.

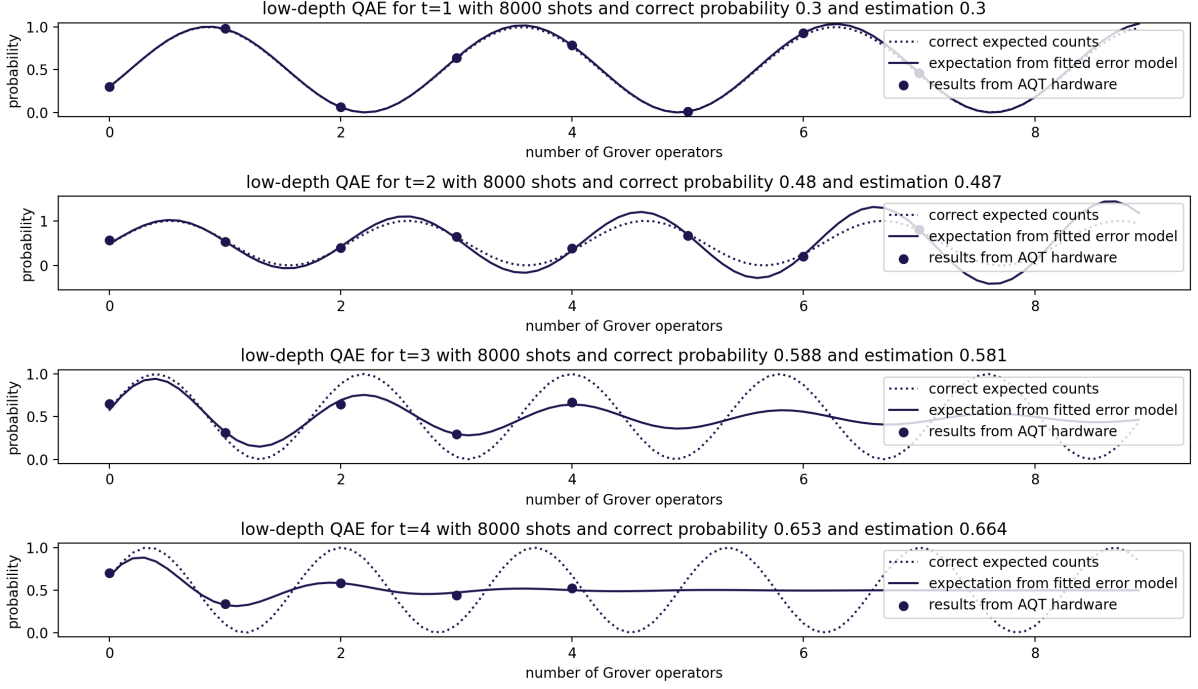


Figure 13: Measurement results (on PINE) for the model with 1 node (filled circles) and expected probabilities to find the node in state 1 for the correct value (dotted curves). The solid curves are the expected probabilities from equation (12) with values  $\theta$ ,  $a$  and  $f$  after gradient descent optimization. For  $t = 1$ , only one curve can be seen as both are very close. Note that the wavelength of the oscillation of the solid curves for  $t = 3$  and  $t = 4$  are shorter, which illustrates the advantage of the error model, see section 4.4.

machine in this case. The impact of the error model seems worth further investigations, even though significant hardware advances will be required before quantum computing will find practical applications for evaluating models like the one presented here.

## Acknowledgement

This work is supported by the German Federal Ministry of Education and Research through project 'Quantencomputer mit gespeicherten Ionen für Anwendungen (ATIQ)', sub-project 'Modellierung von Kreditrisiken mit Quantenalgorithmen' (FKL: 13N10124).

We also acknowledge support by the Austrian Research Promotion Agency (FFG) through grants ELQO (FFG-No 884471), HPQC (FFG-No 897481) and ITAQC (FFG-No 896213), and by the EU Quantum Technology Flagship under the MILLENION Grant Agreement with no. 101114305.

## Appendix A

### Pygrnd implementation of the exact evaluation

In this section, we describe the algorithm of Pygrnd [20], which calculates the exact probabilities with a classical algorithm. This evaluation is implemented in the function `classicalEvaluation` in the module `pygrnd.qc.probablisticNetworks`.

We can simplify the description of the algorithm by defining the following data structure. For a subset  $M \subseteq N$  and for a time step  $t$ , we denote by  $C_M(t)$  a data structure, which stores pairs  $(b, c) \in \{0, 1\}^k \times \{0, 1\}^k$  along with their corresponding probabilities after processing the nodes in  $M$ . A pair can occur more than once with different probability values. Here, the parameter  $M$  denotes the structure after all the nodes in  $M$  are already processed. The structure  $C_\emptyset(t)$  denotes the data before processing a node in this time step. The configuration  $b$  in the pair  $(b, c)$  is used to keep track of the history of a configuration during calculations, e.g. if we start with the configuration 00 and we calculate the probabilities of 00 and 10 for the next time step by considering first the probabilities for the first node, then we need to know the original configuration of 10 because the probabilities of the second node depend only on the configuration of the previous time step. The value  $b$  of the pair gives us this information.

The following iterative method makes use of this data structure and the method allows us to calculate the probabilities  $p_c(t+1)$  when we have the probabilities  $p_c(t)$  for all possible configurations  $c \in \{0, 1\}^k$ :

- We start with the configuration  $c = (0, \dots, 0)$  and assign the probability  $p_c(0) = 1$  for time step  $t = 0$  to it. This is the initialization of the procedure. It means that in  $C_M(0)$  we have only the element  $(c, c)$  with probability 1.
- For each new time step  $t + 1$ , we iterate over all configurations  $(b, c) \in C_N(t)$ , i.e. the data structure after processing all nodes in time step  $t$ . We store the pairs  $(c, c)$  along with the probability of  $p_{(b,c)}(t)$  in  $C_\emptyset(t+1)$ . If during the iteration over all elements of  $C_N(t)$  a pair  $(c, c)$  is already in  $C_\emptyset(t+1)$  then we add the corresponding probability to the existing value.
- When the nodes in the current subset  $M \subseteq N$  are already processed, then we consider the next node  $n \in N$ . We start with an empty  $C_{M'}(t+1)$  where  $M' = M \cup \{n\}$ .
- We iterate through all elements  $(b, c) \in C_M(t+1)$ . The elements have the form  $(b, c)$  and  $b$  is the original configuration from the previous time step.
  - If node  $n$  is set to 1 in the value  $b$ , then we add  $(b, c)$  to  $C_{M'}(t+1)$  with the probability value  $1 - p_n^{\text{recover}}$  and the value  $(b, c')$  with probability  $p_n^{\text{revocer}}$ , where  $c' = (s'_1(t), \dots, s'_k(t))$  with  $s'_n(t) = 0$  and  $s'_m(t) = s_m(t)$  for  $m \in N \setminus \{n\}$ .
  - If node  $n$  is set to 0 in the value  $b$ , then we calculate the probability  $p_{\text{off}}$  that the node stays off in time step  $t + 1$ . This is the product of  $1 - p_n^{\text{fail}}$  and the probabilities  $1 - p_{a,n}^{\text{trigger}}$  for all nodes  $a$  that are set to 1 in value  $b$ . We add

$(b, c)$  to  $C_{M'}(t + 1)$  with probability value  $p_{\text{off}}$  and  $(b, c')$  with  $1 - p_{\text{off}}$  where  $c' = (s'_1(t), \dots, s'_k(t))$  with  $s'_n(t) = 1$  and  $s'_m(t) = s_m(t)$  for  $m \in N \setminus \{n\}$ .

- If all entries of  $C_M(t + 1)$  are processed then we move to the next node. If all nodes are processed for a time step  $t$  then we can set  $C_{\Omega}(t + 1)$  to  $C_M(t)$  and start with iterating over the nodes again.
- Note that for a pair  $(b, c)$  of a configuration the element  $b$  stays constant through all nodes for a time step. We use  $b$  to keep track of the configuration of the nodes in the previous time step and we only change the values for the current time step while iterating over the nodes.

## Appendix B

### Pygrnd implementation of the Monte Carlo evaluation

In this section, we describe the Monte Carlo implementation of Pygrnd [20] for calculating the probabilities of the configurations. The function is `monteCarloEvaluation` and it is in the module `pygrnd.qc.probabilisticNetworks`.

- We start with the configuration  $(0, \dots, 0)$  for time step  $t = 0$ .
- For each new time step we iterate through all nodes.
- If a node had state 1 in the previous time step, we set it to 0 for the next time step with probability  $p_k^{\text{recover}}$ .
- If a node had state 0 in the previous time step, we set it to 1 for the next time step with probability  $p_k^{\text{fail}}$ .
- Furthermore, we iterate over all ancestor nodes  $m$  in the network that had state 1 in the previous time step and we set the node  $n$  to state 1 in the next time step with probability  $p_{m,n}^{\text{trigger}}$ .
- After iterating over all nodes this gives us a configuration for time step  $t + 1$ .
- We repeat the procedure above until we obtain a configuration for the desired time step.
- Once we reach the desired time step we store the configuration.
- We repeat the procedure above to generate statistics on generated configurations.
- We approximate the probability of a configuration by dividing the number of occurrences of a configuration by the total number of repetitions.

## References

- [1] S. Battiston, M. Puliga, R. Kaushik, P. Tasca, G. Caldarelli: "DebtRank: Too central to fail? Financial networks, the FED and systemic risk". *Scientific Reports* **2**, 541 (2012).
- [2] M. E. J. Newman: "The structure and function of complex networks". *SIAM Review* **45**, 167-256 (2003).
- [3] S. Battiston, G. Caldarelli, R. M. May, T. Roukny, J. E. Stiglitz: "The price of complexity in financial networks". *Proc. Natl. Acad. Sci.* **113**, 10031–10036 (2016).
- [4] M. C. Braun, T. Decker, N. Hegemann, S. F. Kerstan, C. Schaefer: "A quantum algorithm for the sensitivity analysis of business risks". Preprint at arXiv:2103.05475 (2021).
- [5] G. Brassard, P. Høyer, M. Mosca, A. Tapp: "Quantum amplitude amplification and estimation". *Quantum Computation and Quantum Information*, Samuel J. Lomonaco, Jr. (editor), AMS Contemporary Mathematics, 305:53-74 (2002).
- [6] S. Woerner, D. J. Egger: "Quantum risk analysis", *npj Quantum Inf.* **5**, 15 (2019).
- [7] P. Rebentrost, B. Gupt, T. R. Bromley: "Quantum computational finance: Monte Carlo pricing of financial derivatives". *Phys. Rev. A* **98**, 022321 (2018).
- [8] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. She, S. Woerner: "Option pricing using quantum computers", *Quantum* **4**, 291 (2020).
- [9] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, E. Yndurain: "Quantum computing for finance: State-of-the-art and future prospects". *IEEE Trans. Quantum Eng.* **1**, 1 (2020).
- [10] A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, A. Prakash: "Prospects and challenges of quantum finance". Preprint at arXiv:2011.06492 (2020).
- [11] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, H. Neven: "Focus beyond quadratic speedups for error-corrected quantum advantage". *PRX Quantum* **2**, 010103 (2021).
- [12] Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, N. Yamamoto: "Amplitude estimation without phase estimation". *Quantum Inf. Process.* **19**, 75 (2020).
- [13] T. Tanaka, Y. Suzuki, S. Uno, R. Raymond, T. Onodera, N. Yamamoto: "Amplitude estimation via maximum likelihood on noisy quantum computer", arXiv:2006.16223 (2020).
- [14] S. Aaronson, P. Rall: "Quantum approximate counting", simplified, in *Symposium on Simplicity in Algorithms* (SIAM, 2020) pp. 24–32.
- [15] D. Grinko, J. Gacon, C. Zoufal, S. Woerner: "Iterative quantum amplitude estimation", *npj Quantum Inf.* **7**, 1 (2021).

- [16] T. Giurgica-Tiron, I. Kerenidis, F. Labib, A. Prakash, W. Zeng: "Low depth algorithms for quantum amplitude estimation", arXiv:2012.03348 (2020).
- [17] S. Herbert: "Quantum Monte-Carlo integration: The full advantage in minimal circuit depth", arXiv:2105.09100 (2021).
- [18] K. Plekhanov, M. Rosenkranz, M. Fiorentini, M. Lubasch: "Variational quantum amplitude estimation". Preprint at <https://arxiv.org/abs/2109.03687> (2021).
- [19] T. Giurgica-Tiron, S. Johri, I. Kerenidis, J. Nguyen, N. Pseni, A. Prakash, K. Sosnova, K. Wright, W. Zeng: "Low depth amplitude estimation on a trapped ion quantum computer" Preprint at <https://arxiv.org/abs/2109.09685> (2021).
- [20] Pygrnd, JoS QUANTUM GmbH, <https://github.com/JoSQUANTUM/pygrnd>
- [21] Qiskit: An open-source framework for quantum computing, <https://qiskit.org>, 2021.
- [22] A. Barenco, C. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, H. Weinfurter: "Elementary gates for quantum computation", Physical Review **A52**, 3457 (1995).
- [23] M. C. Braun, T. Decker, N. Hegemann, S. F. Kerstan: "Error resilient quantum amplitude estimation from parallel quantum phase estimation". Preprint at arXiv:2204.01337 (2022).
- [24] Qiskit AQT Provider, <https://github.com/Qiskit-Partners/qiskit-aqt-provider>
- [25] The definition of the AQT noise model for ion trap quantum computers can be found in the file `circ_aqt/aqt_device.py` of the GitHub project <https://github.com/quantumlib/Cirq/tree/master/cirq-aqt>
- [26] S. Herbert, R. Guichard, Roland, D. Ng: "Noise-Aware Quantum Amplitude Estimation", arXiv: 2109.04840 (2021).
- [27] T. Giurgica-Tiron, S. Johri, I. Kerenidis, J. Nguyen, N. Pseni, A. Prakash, K. Sosnova, K. Wright, W. Zeng: "Low depth amplitude estimation on a trapped ion quantum computer", arXiv: 2109.09685 (2021).