Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

Instytut Informatyki

PRACA DYPLOMOWA

Nash equilibria searching and analysis in quantum games

Wyszukiwanie i analiza równowag Nasha w grach kwantowych

Autor: **Tomasz Zawadzki**

Kierunek studiów: **Informatyka**

Opiekun pracy: **dr inż. Katarzyna Rycerz**

Kraków, 2022

Abstract

Quantum game theory is a relatively new field of study that combines quantum mechanics and classical game theory. Theoretical analysis of quantum games may be challenging due to complex matrix calculations involving numerous trigonometric expressions, which scale exponentially with increasing number of players. In this thesis, we focus on software-aided analysis of quantum games in Eisert-Wilkens-Lewenstein scheme as well as introduce, evaluate and compare symbolic and numerical approaches towards such analysis. We propose algorithms for finding best responses and Nash equilibria in pure strategies as well as best response cycles which can be used for constructing mixed strategies with probabilistic Nash equilibria. We also demonstrate the functionalities of our library `ewl`, which can be utilized for deriving complex formulas in symbolic form, on the example of Quantum Prisoner's Dilemma with different parametrizations. Finally, we execute two- and three-player variants of this game on quantum simulators as well as on a real quantum IBM Q device, compare the results with theoretical expectations, and draw conclusions on the current state of quantum gate-based devices.

Streszczenie

Kwantowa teoria gier jest stosunkowo nową dziedziną nauki, która łączy mechanikę kwantową i klasyczną teorię gier. Teoretyczna analiza gier kwantowych może być trudna ze względu na skomplikowane obliczenia macierzowe z wieloma wyrażeniami trygonometrycznymi, które rosną wykładniczo wraz ze wzrostem liczby graczy. W niniejszej pracy skupimy się na wspomaganej programowo analizie gier kwantowych w schemacie Eiserta-Wilkensa-Lewensteina oraz opiszemy i porównamy symboliczne i numeryczne podejścia do takiej analizy. Zaproponujemy algorytmy do znajdowania najlepszych odpowiedzi i równowag Nasha w strategiach czystych, a także cykli najlepszych odpowiedzi, które mogą być wykorzystywane do konstruowania strategii mieszanych posiadających probabilistyczne równowagi Nasha. Na przykładzie kwantowego dylematu więźnia z różnymi parametryzacjami zademonstrujemy również funkcjonalności autorskiej biblioteki `ewl`, która może być wykorzystana do wyprowadzania złożonych formuł w postaci symbolicznej. Uruchomimy również dwu- i trzyosobowe warianty tej gry na symulatorach kwantowych oraz na rzeczywistym urządzeniu kwantowym IBM Q, porównamy wyniki z przewidywaniami teoretycznymi i sformułujemy wnioski na temat obecnego stanu urządzeń opartych o bramki kwantowe.

Acknowledgements

I would like to express my deepest and most sincere gratitude to my supervisor, dr inż. Katarzyna Rycerz for her guidance, advice and patience as well as immense knowledge and experience in the field of quantum computing that she is always willing to share. Her continuous support and motivation were a huge help for me while writing this thesis.

I would also like to thank prof. dr hab. Marek Szopa from University of Economics in Katowice and dr hab. Piotr Frąckiewicz, prof. AP from Pomeranian University in Słupsk for sharing their vast knowledge in the field of quantum games, suggesting the possible directions of research, and exchanging ideas.

I would also like to thank dr inż. Marian Bubak for his constructive remarks, fruitful discussions and motivation during seminars that helped me accomplish this work.

Last but not least, I would like to thank inż. Piotr Kotara for excellent cooperation and enthusiasm during the work on `ewl` library.

Contents

1 Introduction

The subject of this thesis is software-aided analysis and search of Nash equilibria in quantum games. However, this task is only the tip of the iceberg, as it requires the knowledge of best reply correspondence, which depends on the expected payoff function, which in turn depends on the distribution of game outcomes due to the probabilistic nature of quantum computing. In the first chapter, the motivation for this thesis as well as research objectives will be discussed.

1.1 Motivation

Quantum game theory is a relatively new field of study that combines quantum mechanics and classical game theory. Generalization of classical games to the quantum domain significantly expands the space of possible game states and players' strategies, revealing properties not found in the classical counterparts, and therefore becoming an intriguing subject of research.

In recent years there has been an increasing number of publications that introduce various ideas for quantum extensions of classical games to the quantum domain such as quantum poker [14], quantum tic-tac-toe [16], or quantum blackjack [27]. A more general approach towards so-called *quantization* is the EWL scheme [8] which is applicable to any two-player binary choice symmetric game, resulting in quantum games such as Quantum Prisoner's Dilemma [8, 6, 35, 36] or Quantum Absent-Minded Driver [13].

Theoretical analysis of quantum games relies on complex matrix computations with trigonometric expressions of multiple variables, which can be particularly prone to human error when performed manually. Moreover, since quantum operators acting on a n-qubit system are represented as unitary matrices of complex numbers of shape $2^n \times 2^n$, the problem scales exponentially with increasing number of players, which makes the formulas almost impossible to derive manually for three or more players.

The general motivation of this thesis is to explore possible ways to facilitate the analysis of quantum games in the EWL protocol on classical computers, propose and implement symbolic and numerical methods of finding Nash equilibria, then evaluate their performance on several sample quantum games, and finally compare the formulas against the results from publications.

1.2 Research questions

The following research questions provide a starting point for the research conducted in this thesis:

- Which properties of quantum games would be interesting or useful to analyze automatically?
- Can existing software for scientific computing reproduce various formulas related to quantum games as they are published?
- If yes, can these tools be utilized for analysis of more general cases of quantum games, for instance with more players?
- Is there a generic method for finding best reply correspondence or Nash equilibria in pure strategies for arbitrary quantum games in the EWL protocol?
- Are the results of running quantum games on a real quantum device consistent with theoretical expectations?

1.3 Research hypothesis

In this thesis we will try to demonstrate that existing software for scientific computing may be successfully utilized for the purpose of theoretical analysis of various properties of quantum games in the EWL protocol, including finding best responses for arbitrary strategies of the opponent, finding Nash equilibria in pure strategies, or proving the lack of existence of such strategy profiles.

1.4 Research objectives

The objective of this thesis is to conduct research in the field of software-aided analysis of quantum games, in particular:

- prepare software tools for calculating expected payoffs of individual players in arbitrary quantum game in the EWL protocol
- verify the correctness of obtained formulas related to quantum games by comparing them with the theoretical results from existing publications
- execute sample quantum games in the EWL protocol on IBM Q device using `Operators` library instead of manually decomposing quantum entanglement operators J and $J^\dagger$ into simpler quantum gates
- experimentally verify theoretical expectations of quantum game outcomes with the results from a real quantum device
- propose, evaluate and compare symbolic and numerical methods for finding best response in quantum games in the EWL protocol
- check the applicability of fixed-point definition of Nash equilibrium to find such type of equilibria in pure states

1.5 Related works

In [8], Eisert, Wilkens and Lewenstein propose a quantization scheme of two-player binary choice symmetric games which can also involve quantum entanglement (named the EWL protocol after the initials of the originators' names) and demonstrate its use on the example of Prisoner's Dilemma. It is shown that for maximally entangled initial state, the quantum variant of the game has more favourable Nash equilibrium than its classical counterpart. Moreover, there exists a particular quantum strategy called *miracle move* which always gives at least reward if played against any classical strategy which gives the quantum player an advantage over the classical player.

In [15], a quantum version of Prisoner's Dilemma is executed on a real IBM Q quantum device using Qiskit framework. The work focuses mainly on the construction of entanglement operator J known from EWL protocol which was non-trivial at the time due to a limited set of available gates, therefore manual decomposition of J and $J^\dagger$ operators into basic quantum gates supported by a specific IBM Q backend was necessary. The paper also yields valuable results on quantum noise present due to decoherence phenomenon occurring in quantum computers, as well as describes, implements and compares two error mitigation techniques.

In [6], a full $\mathrm{SU}(2)$ parametrization with 3 degrees of freedom is used instead of the original EWL parametrization from [8]. The analysis is continued in [35], confirming that the quantum version of the game is more profitable for both players, as well as showing that in such variant of Quantum Prisoner's Dilemma there always exists a counter-strategy that gives the highest possible payoff. In the following part of the work, mixed strategies that consist of strategies from best response cycles are introduced, leading to the existence of mixed Nash equilibrium located in the saddle point of expected payoff function. Also, some economical examples of utilizing quantum game Nash equilibria are proposed.

In [36], other types of equilibria are introduced, both for pure and mixed states. It is also shown that the Nash equilibria of these games in quantum mixed Pauli strategies are closer to Pareto optimal results than their classical counterparts. The paper also provides examples of equlibria for a few popular 2×2 quantum games.

In [34], instead of using original angle-based parametrizations and trigonometric functions, the authors introduce algebra of quaternions, which cleverly facilitates calculation of probabilities of possible game outcomes. Based on the knowledge of best reply correspondence in analytic form for the game of chicken (meaning *coward*), various types of Nash equilibria both in pure and mixed states are found and analyzed. However, the paper contains no instructions on how to obtain the best response function for a generic 2×2 quantum game.

Apart from publications, there also exist various software tools related to game theory such as Nashpy [37] or Gambit [24], implementing a wide variety of efficient algorithms for numerical analysis of classical games in terms of the existence of Nash equilibria, Pareto efficiency and other various properties, however with no support for quantum games.

1.6 Structure of the work

The thesis consists of nine chapters and an appendix.

Chapter 2 introduces basic terms and definitions related to quantum computing and quantum gate-based devices. Chapter 3 provides the background in the field of classical game theory. Chapter 4 presents the overview of quantum games, introduces the Eisert-Wilkens-Lewenstein (EWL) protocol and defines several popular parametrizations. Chapter 5 introduces a number of variants of Quantum Prisoner's Dilemma with various parametrizations. Chapter 6 presents `ewl` library, a Python tool for symbolic analysis of quantum games in EWL protocol with IBM-Q intergration. Chapter 7 describes proposed algorithms for finding best responses and Nash equilibria in pure states. Chapter 8 presents the results of the experiments with running quantum games on quantum simulators and real quantum devices as well as symbolic and numerical approaches towards software-aided analysis of quantum games. Chapter 9 discusses achieved goals, conclusions and future works.

Appendix A presents various attempts to find best response function symbolically using Mathematica with source code listings. Appendix B contains the abstract submitted for PPAM 2022 conference.

2 Quantum computing

In this chapter we will introduce basic terms and definitions related to quantum computing which are necessary for proper comprehension of quantum games.

2.1 Introduction

In a classical computer, the information is represented using bits which can be either 0 or 1. The quantum equivalent of a simplest unit of information is *quantum bit* or *qubit*, which is a *superposition* (linear combination) of two base states, $|0\rangle$ and $|1\rangle$ [29].

2.2 Dirac notation

In quantum mechanics, *Dirac notation* (also known as *bra-ket notation*) introduced by Paul Dirac is used to mathematically represent quantum states and operators.

A *ket* denotes a column vector:

$$|\psi\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{bmatrix}, \ \alpha_i \in \mathbb{C}, \tag{2.1}$$

while *bra* represents its Hermitian conjugate as a row vector:

$$\langle\psi| = |\psi\rangle^\dagger = \begin{bmatrix} \alpha_0^* & \alpha_1^* & \dots & \alpha_{n-1}^* \end{bmatrix} \tag{2.2}$$

where z^* denotes complex conjugate of $z \in \mathbb{C}$.

Inner product, *scalar product* or *dot product* of two vectors is denoted with *bra-ket* (hence the name of the notation) and returns a scalar:

$$\langle\psi|\phi\rangle = \begin{bmatrix} \alpha_0^* & \alpha_1^* & \dots & \alpha_{n-1}^* \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix} = \sum_{i=0}^{n-1} \alpha_i^* \beta_i \in \mathbb{C}. \tag{2.3}$$

Outer product of two vectors is a matrix and can be written as *ket-bra*:

$$|\psi\rangle\langle\phi| = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{bmatrix} \begin{bmatrix} \beta_0^* & \beta_1^* & \dots & \beta_{n-1}^* \end{bmatrix} \in \mathbb{C}_{n\times n} \tag{2.4}$$

where $(|\psi\rangle\langle\phi|)_{ij} = \alpha_i \beta_j^* \in \mathbb{C}$.

2.3 Tensor product

In order to represent multi-qubit systems, *tensor product* or *Kronecker product* of two vectors is defined as

$$|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{bmatrix} \otimes \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix} = \begin{bmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \vdots \\ \alpha_0\beta_{n-1} \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \\ \vdots \\ \alpha_1\beta_{n-1} \\ \vdots \\ \alpha_{m-1}\beta_0 \\ \alpha_{m-1}\beta_1 \\ \vdots \\ \alpha_{m-1}\beta_{n-1} \end{bmatrix} \tag{2.5}$$

and can be also abbreviated as $|\psi\rangle\,|\phi\rangle$ or $|\psi\phi\rangle$.

Moreover, *tensor product* or *Kronecker product* of two matrices $A_{m\times n} = [a_{ij}]$ and $B_{p\times q} = [b_{ij}]$ is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{C}_{mp\times nq} \tag{2.6}$$

which expands to

$$\begin{bmatrix} a_{11}b_{11} & \dots & a_{11}b_{1q} & \dots & a_{1n}b_{11} & \dots & a_{1n}b_{1q} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & \dots & a_{11}b_{pq} & \dots & a_{1n}b_{p1} & \dots & a_{1n}b_{pq} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} & \dots & a_{m1}b_{1q} & \dots & a_{mn}b_{11} & \dots & a_{mn}b_{1q} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & \dots & a_{m1}b_{pq} & \dots & a_{mn}b_{p1} & \dots & a_{mn}b_{pq} \end{bmatrix}. \tag{2.7}$$

2.4 Quantum states

Quantum states are represented as vectors from Hilbert space over the field of complex numbers. Two base quantum states are

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{2.8}$$

which form an orthonormal basis in single-qubit states called *computational basis*. In general, the arbitrary state of a single qubit can be expressed as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.9}$$

which means the quantum state $|\psi\rangle$ is a *superposition* (linear combination) of the base states $|0\rangle$ and $|1\rangle$ with *amplitudes* α and β, respectively, where $\alpha, \beta \in \mathbb{C}$ must satisfy the normalization condition

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.10}$$

An example state in quantum superposition is

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,. \tag{2.11}$$

Moreover, each quantum state can expressed using three angles $\theta, \phi, \gamma \in \mathbb{R}$ as

$$|\psi\rangle = e^{i\gamma}\left(\cos\tfrac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\tfrac{\theta}{2}\,|1\rangle\right) \tag{2.12}$$

where $e^{i\gamma}$ is the *global phase*, irrelevant due to the fact that has no physically observable consequences. Therefore, quantum states can be represented with only two parameters $\theta, \phi \in \mathbb{R}$ as

$$|\psi\rangle = \cos\tfrac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\tfrac{\theta}{2}\,|1\rangle \tag{2.13}$$

where $e^{i\phi}$ physically significant *relative phase* [29].

We can visualize the quantum state on the Bloch sphere as a unit vector that forms an angle θ with positive z-axis as well as angle ϕ with positive x-axis, as visualized in Fig. 2.1.

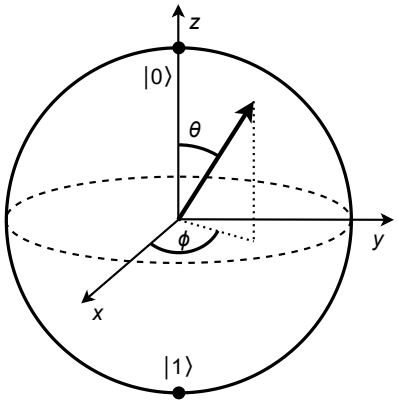

Figure 2.1: The Bloch sphere

In the general case of multi-qubit systems, $|00\ldots0\rangle, |00\ldots1\rangle, \ldots, |11\ldots1\rangle$ form an orthonormal basis also called computational basis. Therefore, an arbitrary n-qubit pure state can be expressed as a superposition of 2^n possible classical states:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^n-1} \end{bmatrix} \tag{2.14}$$

where the amplitudes $a_i \in \mathbb{C}$ must satisfy the normalization condition

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1. \tag{2.15}$$

Apart from pure quantum states, there also exist *mixed states* which are classical probabilistic mixtures of pure quantum states, fundamentally different from superposition. Since such states cannot be expressed as state vectors, a generalization of state vectors for mixed states is the *density matrix* defined as

$$\rho = \sum_{i=1}^{n} p_i |\psi_i\rangle \langle\psi_i| . \tag{2.16}$$

A density matrix represents a pure state if and only if it can be written as an outer product of a state vector $|\psi\rangle$ with itself:

$$\rho = |\psi\rangle \langle\psi| . \tag{2.17}$$

2.5 Quantum entanglement

Two or more qubits are *entangled* if and only if the quantum state of one of the qubits determines the quantum state of another qubit. Moreover, if the quantum state of a single qubit determines the state of all remaining qubits, such system is said to be *maximally entangled*. For instance,

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \tag{2.18}$$

is one of Bell states, which are maximally entangled states among two-qubit states.

An entangled state cannot be expressed as a tensor product of single qubits, or alternatively, there exist no single-qubit quantum states $|\psi_i\rangle$ such that

$$|\psi\rangle = \bigotimes_{i=1}^{n} |\psi_i\rangle . \tag{2.19}$$

Quantum entanglement the foundation of quantum communication protocols such as superdense coding or quantum teleportation.

2.6 Quantum operators

Quantum state of n qubits can be transformed into another n-qubit quantum state using *quantum operators* also known as *quantum gates*, which are represented as unitary matrices of size $2^n \times 2^n$. A matrix U is unitary if and only if

$$U^\dagger U = UU^\dagger = I \tag{2.20}$$

where † (dagger) denotes the Hermitian conjugate of matrix $U_{m\times n} = [u_{ij}]$ defined as

$$U^\dagger = (U^*)^\mathsf{T} = \begin{bmatrix} u_{11}^* & u_{21}^* & \dots & u_{m1}^* \\ u_{12}^* & u_{22}^* & \dots & u_{m2}^* \\ \vdots & \vdots & \ddots & \vdots \\ u_{1n}^* & u_{2n}^* & \dots & u_{mn}^* \end{bmatrix}. \tag{2.21}$$

Since unitarity implies reversibility, quantum computations (except for measurement) are reversible.

Application of quantum operator U on a pure quantum state $|\psi\rangle$ is written as $U|\psi\rangle$. For a quantum state represented with a density matrix ρ, the application of quantum operator U is realized as $A\rho A^\dagger$.

Some of the most commonly used single-qubit quantum operators are:

- identity (analogous to *no-op* instruction in classical computers):

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.22}$$

- Pauli-X (also called NOT gate, since it changes $|0\rangle \mapsto |1\rangle$ and $|1\rangle \mapsto |0\rangle$):

$$X = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.23}$$

- Pauli-Y:

$$Y = \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.24}$$

- Pauli-Z:

$$Z = \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.25}$$

- Sqrt-X (such that $(\sqrt{X})^2 = X$):

$$\sqrt{X} = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} \tag{2.26}$$

- Hadamard:

$$H = \frac{1}{\sqrt{2}}(X+Z) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.27}$$

- Phase shift (leaves $|0\rangle$ untouched and changes $|1\rangle \mapsto e^{i\varphi}|1\rangle$):

$$P(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \tag{2.28}$$

- Rotation-X:

$$R_x(\theta) = e^{-i\frac{\theta}{2}X} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.29}$$

- Rotation-Y:

$$R_y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.30}$$

- Rotation-Z:

$$R_z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{bmatrix} e^{-\frac{i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{bmatrix} \tag{2.31}$$

Pauli matrices X, Y, Z together with identity matrix I form a basis in the space of 2×2 quantum operators, meaning that each single-qubit quantum gate can be expressed as a linear combination of these four basis gates. Also, the rotation gates represent rotations on the Bloch sphere along three main axes.

Since quantum gates are linear operators, their behavior can be fully described only with the output of their application on quantum states from the computational basis:

$$U|\psi\rangle = U\left(\sum_{i=0}^{n-1} \psi_i |i\rangle\right) = \sum_{i=0}^{n-1} \psi_i U|i\rangle\,. \tag{2.32}$$

CNOT gate is a two-qubit quantum operator that swaps the amplitudes of states $|10\rangle$ and $|11\rangle$, or alternatively, conditionally applies NOT gate on the target qubit if the control qubit is $|1\rangle$:

$$\text{CNOT}_{10} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.33}$$

There also exists another variant of CNOT gate that interchanges the amplitudes of quantum states $|01\rangle$ and $|11\rangle$:

$$\text{CNOT}_{01} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.34}$$

Another commonly used two-qubit quantum operator is SWAP gate that simply swaps two qubits involved in the operation:

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.35}$$

2.7 Measurement

Since quantum states cannot be directly observed, in order to obtain a numerical result, a quantum system needs to be manipulated or tested in the process called *measurement*. During measurement, the wave function representing the quantum state of qubits collapses to a new quantum state from the measurement basis, with probabilities depending on the amplitudes of the original state.

For instance, measurement in the computational basis of a quantum state

$$|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.36}$$

will collapse either to $|0\rangle$ or $|1\rangle$ as classical bits with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. In general, an arbitrary pure n-qubit quantum state

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle\,, \tag{2.37}$$

when measured, will collapse to $|i\rangle$ with probability

$$p_i = |\alpha_i|^2\,. \tag{2.38}$$

According to no-cloning theorem, a quantum state cannot be copied, therefore it is not possible to perform multiple measurements of the same quantum state. Also, when qubits are entangled, measurement of one qubit affects other qubits as well.

2.8 Quantum circuits

A sequence of quantum operators applied on a system of qubits can be represented as a quantum circuit where qubits are depicted as horizontal wires and quantum gates are denoted as rectangular boxes labeled with the symbol of the gate. Quantum circuits terminated with measurement blocks are called *quantum algorithms*. The double lines represent classical registers that store the result of measurements as classical bits. Apart from being the output of the algorithm, they can also be used to classically control other quantum gates. An example quantum circuit is shown in Fig. 2.2.

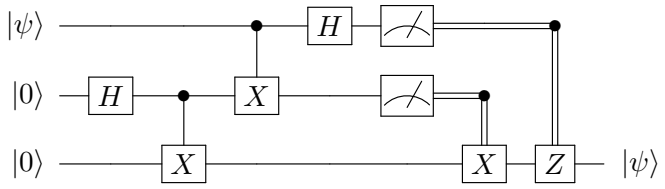

Figure 2.2: Quantum circuit for quantum teleportation

2.9 Quantum gate devices

Currently, there are two kinds of quantum computers available: *quantum gate-based devices* also known as *universal quantum computers*, and *quantum annealers*, for instance D-Wave[1], designed specifically for solving optimization problems by minimizing an objective function in a particular form. In this thesis, however, we focus on quantum gate-based devices, as they allow direct implementation of quantum circuits realizing quantum games in the EWL scheme.

The most popular provider of access to quantum gate-based devices is IBM, up to 127 qubits[2]. However, as of 2022, only 6 systems with at most 7 qubits are available for public use, free of charge. Parameters of IBM Q systems are summarized in Tab. 2.1. Running the quantum circuit on a real quantum device is described in detail in section 8.2 of this thesis.

Table 2.1: Comparision of parameters of IBM Q systems (as of June 18, 2022)

Backend name	Number of qubits	Quantum volume	CLOPS[3]	Processor type	Available for public use
ibmq_armonk	1	1	n/d	Canary r1.2	Yes
ibmq_lima	5	8	2.7K	Falcon r4T	Yes
ibmq_belem	5	16	2.5K	Falcon r4T	Yes
ibmq_quito	5	16	2.5K	Falcon r4T	Yes
ibmq_manila	5	32	2.8K	Falcon r5.11L	Yes
ibmq_oslo	7	32	2.6K	Falcon r5.11H	Yes
ibmq_santiago	5	32	n/d	Falcon r4L	No
ibmq_bogota	5	32	2.3K	Falcon r4L	No
ibmq_jakarta	7	16	2.4K	Falcon r5.11H	No
ibm_nairobi	7	32	2.6K	Falcon r5.11H	No
ibm_lagos	7	32	2.7K	Falcon r5.11H	No
ibm_perth	7	32	2.9K	Falcon r5.11H	No
ibmq_guadalupe	16	32	2.4K	Falcon r4P	No
ibm_peekskill	27	1	n/d	Falcon r8	No
ibmq_toronto	27	32	1.8K	Falcon r4	No
ibm_hanoi	27	64	2.3K	Falcon r5.11	No
ibm_auckland	27	64	2.4K	Falcon r5.11	No
ibm_cairo	27	64	2.4K	Falcon r5.11	No
ibmq_mumbai	27	128	1.8K	Falcon r5.1	No
ibmq_montreal	27	128	2K	Falcon r4	No
ibmq_kolkata	27	128	2K	Falcon r5.11	No
ibmq_brooklyn	65	32	1.5K	Hummingbird r2	No
ibm_washington	127	64	850	Eagle r1	No

[1] https://www.dwavesys.com/
[2] https://quantum-computing.ibm.com/services?services=systems
[3] circuit layer operations per second

2.10 Summary

In this chapter, we introduced the mathematical formalism used to describe quantum states and operators, as well as explained fundamental concepts in quantum mechanics such as superposition or quantum entanglement.

3 Classical game theory

In this chapter we will introduce basic terms and definitions related to classical game theory that will be frequently used in the later part of the thesis.

3.1 Introduction

Formally, game theory is the branch of mathematics focused on the study of optimal strategies of involved parties (*players*) in case of a conflict of interest. Each player chooses a certain action (*strategy*) and depending on strategies of all players, receives an appropriate *payoff*, represented as a numerical value (the higher, the better) [30].

3.2 Definitions

Assume that N is a finite set of n players and each player is denoted by i. Let S_i denote the set of possible pure strategies available to player i. A pure strategy provides a complete information how player plays the game.

Definition 3.2.1 (Pure strategy profile)**.** A pure strategy profile is a n-tuple composed of strategies s_i of all players, i.e.

$$\mathbf{s} = (s_1, s_2, \ldots, s_n) \tag{3.1}$$

such that $\forall\, i \in N : s_i \in S_i$.

Definition 3.2.2 (Payoff function)**.** A payoff function is a mapping

$$p_i : S_1 \times S_2 \times \ldots \times S_n \to \mathbb{R} \tag{3.2}$$

that maps strategy profile to outcome of the game (payoff) for player i.

Definition 3.2.3 (Normal-form game)**.** A game in normal form is a triple $\Gamma = (N, \mathbf{S}, \mathbf{p})$ where

- $N = \{1, 2, \ldots, n\}$ is a finite set of n players,
- $\mathbf{S} = S_1 \times S_2 \times \ldots \times S_n$ is a set of strategy profiles,
- $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ is a tuple of payoff functions.

For readability purposes, a strategy profile of all players excluding player i will be denoted as s_{-i}. Analogously, $S_{-i} = \bigtimes_{j=1, j\neq i}^{n} S_j$ denotes the set of all strategy profiles excluding player i. Moreover, strategy profile $\mathbf{s}$ will be represented as (s_i, s_{-i}).

Definition 3.2.4 (Mixed strategy)**.** A mixed strategy of player i is a classical probability distribution $m_i : S_i \to [0, 1]$ of pure strategies available to that player where

$$\sum_{s_i \in S_i} m_i(s_i) = 1. \tag{3.3}$$

Each pure strategy s_i can be interpreted as a mixed strategy m_i such that

$$m_i(s_i') = \begin{cases} 1 & s_i = s_i' \\ 0 & s_i \neq s_i' \end{cases}. \tag{3.4}$$

3.3 Classification

Among the classic games, we can distinguish the following classes of games:

Definition 3.3.1 (Finite game)**.** A game in normal form is said to be finite if and only if $\mathbf{S}$ is a finite set.

Definition 3.3.2 (Constant-sum game)**.** In a constant-sum game, for each strategy profile, the sum of payoffs for each player is always sequal to $C \in R$, i.e.

$$\forall\, (s_1, s_2, \ldots, s_n) \in \mathbf{S} : \sum_{i=1}^{n} p_i(s_i) = C. \tag{3.5}$$

Definition 3.3.3 (Zero-sum game)**.** A zero-sum game is a constant-sum game where $C = 0$, i.e.

$$\forall\, (s_1, s_2, \ldots, s_n) \in \mathbf{S} : \sum_{i=1}^{n} p_i(s_i) = 0. \tag{3.6}$$

Definition 3.3.4 (Bimatrix game)**.** In a bimatrix game, there are two players and each has a finite number of possible strategies, p and q, respectively. In such game, payoffs can be represented by two matrices $A, B \in \mathbb{R}_{p \times q}$ for player 1 and 2, respectively:

$$p_1(s_i, s_j) = A_{ij} \text{ and } p_2(s_i, s_j) = B_{ij}. \tag{3.7}$$

hence the name. Alternatively, the payoffs in a bimatrix game can be represented as a single matrix $P = \mathbb{R}_{p \times q \times n}$, so that $p_k(s_i, s_j) = P_{ijk}$.

Definition 3.3.5 (Symmetric game)**.** A game is symmetric if $S_1 = S_2 = \ldots = S_n$ and for each permutation $\pi : N \to N$

$$p_{\pi(i)}(s_1, s_2, \ldots, s_n) = p_i(s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(n)}). \tag{3.8}$$

In other words, the payoffs depend only on the strategies used in the strategy profile and not on who is playing them.

3.4 Nash equilibrium

Definition 3.4.1 (Best response)**.** A pure strategy s_i of player i is a best response to a strategy profile of all other players s_{-i} if

$$\forall s_i' \in S_i : p_i(s_i, s_{-i}) \geq p_i(s_i', s_{-i}). \tag{3.9}$$

Definition 3.4.2 (Nash equilibrium in pure states)**.** A strategy profile s^* is a Nash equilibrium in pure states if and only if

$$\forall\, i \in N\ \forall\, s_i \in S_i : p_i(s_i^*, s_{-i}^*) \geq p_i(s_i, s_{-i}^*). \tag{3.10}$$

Alternatively, a strategy profile s^* is a Nash equilibrium if s_i* is a best response to s_{-i}^* for each $i \in I$. In other words, no player can increase his payout by unilaterally (i.e. without changing the strategy of all the other players) changing his strategy.

Definition 3.4.3 (Nash equilibrium in pure states in a two-player game)**.** In a two-player game, a strategy profile (s_1, s_2) is a Nash equilibrium in pure states if and only if s_1 is a best response to s_2 and vice versa.

3.5 Prisoner's Dilemma

In general form, Prisoner's Dilemma is a non-cooperative symmetric bimatrix game with the following payoff matrix

$$P = \begin{bmatrix} (r,r) & (s,t) \\ (t,s) & (p,p) \end{bmatrix} \in \mathbb{R}_{2\times 2\times 2} \tag{3.11}$$

where r is the reward payoff if both players decide to cooperate, p is the punishment payoff when both players defect, t is temptation payoff and s is sucker's payoff in case the first player defects while the second cooperates, respectively.

For the dilemma to exists, these values need to satisfy the conditions $s < p < r < t$ and $2r > s + t$. By substituting the concrete common payoff values $r = 3$, $p = 1$, $t = 5$ and $s = 0$ we obtain

$$\begin{bmatrix} (3,3) & (0,5) \\ (5,0) & (1,1) \end{bmatrix}. \tag{3.12}$$

Note that these values are not meant to represent the number of years to be spent in prison – instead, the higher the payoff, the milder the sentence.

3.6 Summary

In this chapter we introduced the basic concepts of game theory, defined the major classes of games, provided a definition of Nash equilibrium, and introduced a classical game called Prisoner's Dilemma.

4 Quantum games

In this chapter we will introduce basic concepts related to quantum game theory.

4.1 Introduction

Quantum game combines classical game theory with quantum mechanics and focuses primarily on design, classification and analysis of quantum games, which are an extension of classical games to the quantum domain. Recently, a particularly interesting topic is the quantization of existing classical games, resulting in quantum games such as quantum poker [14], quantum tic-tac-toe [16], or quantum blackjack [27].

4.2 Characteristics

In order for a game to be considered quantum, it must exhibit three following properties as presented in [15]:

1. **Quantum game utilizes the concepts of quantum computing.**

   The state of the game is expressed as a quantum state. Players act by applying quantum operators on their qubits. The result of the game is a measurement of quantum state of the game.

2. **Quantum game is reducible to its classical counterpart.**

   If we limit the set of available pure strategies to base strategies, the game reduces to its classical counterpart.

3. **Quantum game extends the properties of its classical counterpart.**

   There must exist some kind of quantum advantage, for instance new kinds of equilibria not found in the classical counterpart, for instance arising from superposition of quantum states, the presence of quantum entanglement, or the probabilistic nature of quantum measurement.

4.3 Eisert-Wilkens-Lewenstein protocol

EWL protocol is a quantization scheme of classical games proposed by Eisert, Wilkens and Lewenstein that applies to any two-player binary choice symmetric game [8]. Figure 4.1 shows the general quantum circuit of a quantum game in the EWL protocol.

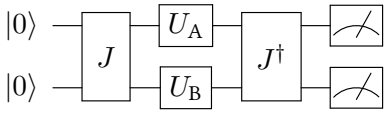

Figure 4.1: Quantum circuit of a quantum game in the EWL protocol

The result statevector of the quantum state can be calculated as

$$|\psi\rangle = J^\dagger \left(U_\mathrm{A} \otimes U_\mathrm{B}\right) J v \tag{4.1}$$

where the initial vector v is usually $|00\rangle$, and the entanglement operator J is publicly known to all players. The amplitudes of the final statevector can be easily converted to observational basis probabilities:

$$\mathbf{p} = |\psi\rangle \, |\psi\rangle^\dagger = ||\psi\rangle|^2 . \tag{4.2}$$

As mentioned in Eq. 3 in [12], the EWL protocol can be generalized to arbitary number of players. In case of n-player game, $n \in \mathbb{N}$, the following formula is used:

$$|\psi\rangle = J^\dagger \left(\bigotimes_{i=1}^{n} U_i \right) J \, |0\rangle^{\otimes n} \tag{4.3}$$

where $\{U_i\}_{i=1}^n$ is the sequence of players' strategies and J is the n-qubit entanglement operator, e.g.

$$J = \frac{1}{\sqrt{2}} \left(I^{\otimes n} + i\sigma_x^{\otimes n} \right) \tag{4.4}$$

where I is the identity matrix and σ_x is Pauli matrix X.

The expected payoff for player i can be calculated as

$$\$_\mathrm{i} = \sum_j p_j \$_{i,j}. \tag{4.5}$$

4.4 Parametrizations

In quantum games in the EWL protocol, players' strategies are single-qubit quantum operators. The only mathematical requirement is that they must be unitary matrices. Since the universal quantum gate has 3 degrees of freedom, strategies can be parametrized with three angles. However, it is possible to limit the space of available strategies using parametrizations with fewer degrees of freedom.

The original parametrization proposed by the authors of the EWL scheme in [8] is

$$U(\theta,\phi) = \begin{bmatrix} e^{i\phi}\cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & e^{-i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix},\ \theta \in [0,\pi],\ \phi \in [0,\tfrac{\pi}{2}]. \tag{4.6}$$

However, Benjamin and Hayden commented that "it seems unlikely that the restriction can reflect any reasonable physical constraint (limited experimental resources, say) because this set is not closed under composition" [2].

In order to overcome that issue, alternative parametrizations may be utilized, which are assumed to include set $\{\, U(\theta,0,0) : \theta \in [0,\pi] \,\}$ as well as the base strategies of the quantum game. However, as shown in later part of the thesis, the choice of parametrization used in the quantum game strongly affects its properties such as best responses or the existence of Nash equilibria.

Another parametrization with only 2 degrees of freedom is Frąckiewicz-Pykacz parametrization introduced in [12] and used in [36]:

$$U(\theta,\phi) = \begin{bmatrix} e^{i\phi}\cos\left(\frac{\theta}{2}\right) & ie^{i\phi}\sin\left(\frac{\theta}{2}\right) \\ ie^{-i\phi}\sin\left(\frac{\theta}{2}\right) & e^{-i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix},\ \theta \in [0,\pi],\ \phi \in [0,2\pi]. \tag{4.7}$$

This parametrization provides a strong isomorphism of the quantum game and gives the same Nash equilibria in mixed strategies as full $\mathrm{SU}(2)$ parametrization of EWL.

Apart from parametrizations with 2 angles, there are several frequently used parametrizations with 3 degrees of freedom that fully cover the set of all unitary matrices, for instance $U(\theta,\phi,\alpha)$ parametrization defined as

$$U(\theta,\phi,\alpha) = \begin{bmatrix} e^{-i\phi}\cos\left(\frac{\theta}{2}\right) & e^{i\alpha}\sin\left(\frac{\theta}{2}\right) \\ -e^{-i\alpha}\sin\left(\frac{\theta}{2}\right) & e^{i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix},\ \theta \in [0,\pi],\ \phi,\alpha \in [-\pi,\pi]. \tag{4.8}$$

Another parametrization is mentioned in Eq. 2 in [11] and [12]:

$$U(\theta,\alpha,\beta) = \begin{bmatrix} e^{i\alpha}\cos\left(\frac{\theta}{2}\right) & ie^{i\beta}\sin\left(\frac{\theta}{2}\right) \\ ie^{-i\beta}\sin\left(\frac{\theta}{2}\right) & e^{-i\alpha}\cos\left(\frac{\theta}{2}\right) \end{bmatrix},\ \theta \in [0,\pi],\ \alpha,\beta \in [0,2\pi). \tag{4.9}$$

Another parametrization from Eq. 2.24 in [15]:

$$U(\theta,\phi,\lambda) = \begin{bmatrix} e^{-\frac{i(\phi+\lambda)}{2}}\cos\left(\frac{\theta}{2}\right) & -e^{-\frac{i(\phi-\lambda)}{2}}\sin\left(\frac{\theta}{2}\right) \\ e^{\frac{i(\phi-\lambda)}{2}}\sin\left(\frac{\theta}{2}\right) & e^{\frac{i(\phi+\lambda)}{2}}\cos\left(\frac{\theta}{2}\right) \end{bmatrix},\ \theta \in [0,\pi],\ \phi,\lambda \in [0,4\pi). \tag{4.10}$$

Another parametrization from Eq. 45 in [33]:

$$U(\theta,\gamma,\delta) = \begin{bmatrix} e^{i(\gamma+\delta)}\cos\left(\frac{\theta}{2}\right) & ie^{i(\gamma-\delta)}\sin\left(\frac{\theta}{2}\right) \\ ie^{-i(\gamma-\delta)}\sin\left(\frac{\theta}{2}\right) & e^{-i(\gamma+\delta)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{4.11}$$

Another parametrization as described in IBM Qiskit Textbook [1]:

$$U(\theta,\phi,\lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\phi)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{4.12}$$

However, the analysis of quantum games with full $\mathrm{SU}(2)$ parametrization does not yield interesting properties because of the fact that for each strategy, there always exists a counter-strategy which gives the player the highest payout.

4.5 Summary

In this section we introduced quantum game theory, presented the rules that a game must follow in order to be considered quantum, as well as defined several parametrizations with 2 or 3 degrees of freedom that will be extensively used in the later part of the thesis.

5 Quantum Prisoner's Dilemma

In this chapter we will introduce the quantum version of Prisoner's Dilemma as well as define four variants of Quantum Prisoner's Dilemma with different parametrizations.

5.1 Introduction

Quantum Prisoner's Dilemma is a generalization of Prisoner's Dilemma introduced in section 3.5 to the quantum domain using the EWL protocol described in section 4.3.

Assuming there are two players, traditionally named Alice and Bob, the first qubit is assigned to Alice while the second is assigned to Bob. Two base strategic decisions, cooperate and defeat, are associated with two base quantum states $|0\rangle$ and $|1\rangle$, respectively. In order to enable quantum properties of the game, the qubits are entangled using publicly-known entanglement operator J. Then, players individually play their strategies by applying quantum gates on their respective qubits. Finally, the measurement is the outcome of the game.

Similarly to the original version of the game, there are exactly 4 possible game outcomes $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ which directly represent *CC*, *CD*, *DC* and *DD* from the classical counterpart, respectively, with analogous payoff values. However, an important difference is the randomness due to the nature of the measurement, therefore expected value of payoff should be analyzed.

5.2 Variant with original EWL parametrization

This variant was originally described in [8] and uses parametrization from Eq. 4.6:

$$U(\theta, \phi) = \begin{bmatrix} e^{i\phi}\cos\left(\frac{\theta}{2}\right) & \sin\left(\frac{\theta}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right) & e^{-i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{5.1}$$

where $\theta \in [0, \pi]$ and $\phi \in [0, \frac{\pi}{2}]$. For $\phi_A = \phi_B = 0$ the game reduces to its classical counterpart with mixed strategy of C and D with probabilities $\cos\left(\frac{\theta}{2}\right)^2$ and $\sin\left(\frac{\theta}{2}\right)^2$, respectively. Two base strategies, C (cooperate) and D (defeat), are defined as in Eq. 4 and Eq. 5 in [8]:

$$\begin{aligned} C &= U(0,0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ D &= U(\pi,0) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \end{aligned} \tag{5.2}$$

Additionally, paper [8] also introduces parameter $\gamma \in [0, \frac{\pi}{2}]$ which is a measure of quantum entanglement of the game with J matrix being defined as

$$J = \exp(-i\gamma D \otimes D/2). \tag{5.3}$$

For $\gamma = 0$, the game is separable and $D \otimes D$ is the equilibrium in dominant strategies just like in the classical variant. For $\gamma = \frac{\pi}{2}$, the game is maximally entangled. In such case, $Jv = [1, 0, 0, i]/\sqrt{2}$, or equivalently

$$|\psi\rangle = \frac{\sqrt{2}\,(|00\rangle + i|11\rangle)}{2}. \tag{5.4}$$

There are two players, Alice and Bob, who use single-qubit strategies $U_{\mathrm{A}} = U(\theta_{\mathrm{A}}, \phi_{\mathrm{A}})$ and $U_{\mathrm{B}} = U(\theta_{\mathrm{B}}, \phi_{\mathrm{B}})$, respectively. In contrast to the classical counterpart, $D \otimes D$ is no longer a Nash equilibrium. Instead, there exists an unique Nash equilibrium $Q \otimes Q$ where

$$Q = U(0, \tfrac{\pi}{2}) = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} \tag{5.5}$$

with payoffs $\$_{\mathrm{A}}(Q, Q) = \$_{\mathrm{B}}(Q, Q) = 3$ whereas $\$_{\mathrm{A}}(D, D) = \$_{\mathrm{B}}(D, D) = 1$. Additionally, $Q \otimes Q$ turns out to be Pareto optimal as well.

In a case where Alice can use quantum strategy but Bob is limited to classical strategies, Alice can play the following so-called *miracle move* from Eq. 9 in [8]:

$$M = U(\tfrac{\pi}{2}, \tfrac{\pi}{2}) = \frac{1}{\sqrt{2}} \begin{bmatrix} i & -1 \\ 1 & -i \end{bmatrix}, \tag{5.6}$$

which gives her expected payoff

$$\$_{\mathrm{A}}(M, U(\theta_{\mathrm{B}}, 0)) \geq 3 \tag{5.7}$$

for any $\theta_{\mathrm{B}} \in [0, \pi]$.

5.3 Variant with $U(\theta, \phi, \alpha)$ parametrization

This variant has been described in [6]. The most important difference from the previously discussed variant of the game is that this version uses parametrization with 3 degrees of freedom from Eq. 4.8:

$$U(\theta, \phi, \alpha) = \begin{bmatrix} e^{-i\phi}\cos\left(\frac{\theta}{2}\right) & e^{i\alpha}\sin\left(\frac{\theta}{2}\right) \\ -e^{-i\alpha}\sin\left(\frac{\theta}{2}\right) & e^{i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{5.8}$$

where $\theta \in [0, \pi]$ and $\phi, \alpha \in [-\pi, \pi]$. This parametrization produces arbitrary single-qubit gates from SU(2) group. Two base strategies, C (cooperate) and D (defeat), are defined as follows:

$$\begin{aligned} C &= U(0, 0, 0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ D &= U(\pi, 0, \tfrac{\pi}{2}) = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix}. \end{aligned} \tag{5.9}$$

Two players, Alice and Bob, use single-qubit strategies $U_\mathrm{A} = U(\theta_\mathrm{A}, \phi_\mathrm{A}, \alpha_\mathrm{A})$ and $U_\mathrm{B} = U(\theta_\mathrm{B}, \phi_\mathrm{B}, \alpha_\mathrm{B})$, respectively.

In Eq. 3 we can find formula for J gate:

$$J = \frac{1}{\sqrt{2}}(I + i\sigma_x \otimes \sigma_x) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & i \\ 0 & 1 & i & 0 \\ 0 & i & 1 & 0 \\ i & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

where σ_x represents Pauli matrix (quantum NOT gate).

Using formula from Eq. 4.1 and converting amplitudes to observational basis probabilities using Eq. 4.2 we can obtain the equations for the four possible outcomes of the game:

$$\begin{aligned} &\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) - \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\ &\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right) - \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2 \\ &\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2 \\ &\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) + \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2. \end{aligned} \tag{5.11}$$

As mentioned in [6], this variant of Quantum Prisoner's Dilemma has no single strategy Nash equilibrium because there is no dominant strategy. Eq. 5 shows the general formula for Bob's best response for arbitrary strategy $U(\theta_A, \phi_A, \alpha_A)$ of Alice:

$$\begin{aligned} \theta_B &= \theta_A + \pi \\ \phi_B &= \alpha_A \\ \alpha_B &= \phi_A - \tfrac{\pi}{2}. \end{aligned} \tag{5.12}$$

From this substitution we get $\$_A = 0$ and $\$_B = 5$ regardless of the parameters of the initial Alice's strategy. Analogous substitutions are also valid for Alice's best response.

An interesting property of this variant of the game is the existence of cycles of best responses for arbitrary initial strategy. Without loss of generality, let us assume that Alice plays

$$U_A = U(\theta_A, \phi_A, \alpha_A). \tag{5.13}$$

According to Eq. 5.12, Bob's best response for U_A is

$$U_B = U(\theta_A + \pi, \alpha_A, \phi_A - \tfrac{\pi}{2}). \tag{5.14}$$

We can also apply the same formula to obtain Alice's best response for U_B,

$$U'_A = U(\theta_A, \phi_A - \tfrac{\pi}{2}, \alpha_A - \tfrac{\pi}{2}). \tag{5.15}$$

Apparently, $U'_A \neq U_A$, so let us apply the formula again to get

$$U'_B = U(\theta_A + \pi, \alpha_A - \tfrac{\pi}{2}, \phi_A - \pi). \tag{5.16}$$

Finally, let us calculate Alice's best response one more time

$$U''_A = U(\theta_A, \phi_A - \pi, \alpha_A - \pi), \tag{5.17}$$

which according to [6] and [35] happens to be equal to the original Alice's strategy, revealing the following cycle of best responses:

$$U_A \to U_B \to U'_A \to U'_B \to U_A \tag{5.18}$$

where $X \to Y$ indicates that Y is the best response for X. Moreover, the property holds for arbitrary $\theta_A, \phi_A, \alpha_A$ so it fully covers the set of possible strategies.

As shown in [36], these cycles can be used to form mixed strategies $\{U_A, U'_A\}$ and $\{U_B, U'_B\}$ where Nash equilibria may exist. Mixed strategies are the subject of research performed by Piotr Kotara's for his master's thesis [20].

A similar variant of the game is described in [35] with the only difference being the base strategy D defined as

$$D = U(\pi, 0, 0) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{5.19}$$

5.4 Variant with $U(\theta, \alpha, \beta)$ parametrization

Another variant of Quantum Prisoner's Dilemma has been described in [12] with another full SU(2) parametrization from Eq. 4.9:

$$U(\theta, \alpha, \beta) = \begin{bmatrix} e^{i\alpha}\cos\left(\frac{\theta}{2}\right) & ie^{i\beta}\sin\left(\frac{\theta}{2}\right) \\ ie^{-i\beta}\sin\left(\frac{\theta}{2}\right) & e^{-i\alpha}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \ \theta \in [0, \pi], \ \alpha, \beta \in [0, 2\pi). \tag{5.20}$$

Two base strategies, C (cooperate) and D (defeat), are defined in Eq. 4 and Eq. 5 in [8]:

$$\begin{aligned} C = U(0,0,0) = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ D = U(\pi,0,0) = i\sigma_x = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix}. \end{aligned} \tag{5.21}$$

Alice plays $U_{\mathrm{A}} = U(\theta_1, \alpha_1, \beta_1)$ and Bob plays $U_{\mathrm{B}} = U(\theta_2, \alpha_2, \beta_2)$ (using digits instead of name initial to preserve the original notation). Eq. 7 shows the observational basis probabilities:

$$\begin{aligned} &\left(\cos(\alpha_1 + \alpha_2)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) + \sin(\beta_1 + \beta_2)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2 \\ &\left(\cos(\alpha_1 - \beta_2)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) + \sin(\alpha_2 - \beta_1)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2 \\ &\left(\sin(\alpha_1 - \beta_2)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) + \cos(\alpha_2 - \beta_1)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2 \\ &\left(\sin(\alpha_1 + \alpha_2)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) - \cos(\beta_1 + \beta_2)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2. \end{aligned} \tag{5.22}$$

5.5 Variant with Frąckiewicz-Pykacz parametrization

This variant was originally described in [12] and uses parametrization from Eq. 28:

$$U(\theta,\phi) = \begin{bmatrix} e^{i\phi}\cos\left(\frac{\theta}{2}\right) & ie^{i\phi}\sin\left(\frac{\theta}{2}\right) \\ ie^{-i\phi}\sin\left(\frac{\theta}{2}\right) & e^{-i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{5.23}$$

where $\theta \in [0,\pi]$ and $\phi \in [0,2\pi]$. Similarly as in previous cases, two base strategies for cooperation and defeat are defined as

$$\begin{aligned} C &= U(0,0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ D &= U(\pi,0) = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix}. \end{aligned} \tag{5.24}$$

As justified in [12], the parametrization is designed so that the quantum scheme is invariant with respect to isomorphic transformations of an input game. Simultaneously, it does not provide the players with the counter strategies like full SU(2) parametrizations do, allowing for Nash equilibria to exist in a wider class of quantum games. Eq. 10 in [36] describes the observational basis probabilities:

$$\begin{aligned} &\left(\cos\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A+\phi_B\right)+\sin\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A+\phi_B\right)\right)^2 \\ &\left(\cos\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A-\phi_B\right)-\sin\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A-\phi_B\right)\right)^2 \\ &\left(\cos\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A-\phi_B\right)+\sin\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A-\phi_B\right)\right)^2 \\ &\left(\cos\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A+\phi_B\right)-\sin\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A+\phi_B\right)\right)^2. \end{aligned} \tag{5.25}$$

5.6 Summary

In this chapter we presented several variants of Quantum Prisoner's Dilemma from various publications that will be further analysed in the later part of the thesis.

6 EWL library

This chapter introduces `ewl` library [39] which is a Python tool for symbolic analysis of quantum games in EWL protocol with IBM Q integration. The library has been developed by the author in cooperation with Piotr Kotara who also uses it in his master's thesis [20]. In this chapter we will discuss the motivation behind developing such tool as well as present its core functionalities and additional features.

6.1 Motivation

Quantum game analysis heavily relies on complex matrix computations where the elements are sums and products of trigonometric expressions involving multiple variables. Many relevant conclusions are drawn from the form and other characteristics of these trigonometric expressions, which needed to be appropriately simplified or otherwise transformed.

However, this kind of calculations is particularly prone to human error because of the length and complexity of such mathematical expressions. Moreover, some valuable simplification opportunities may be easily missed. Finally, with the increasing number of players, the sizes of the matrices grow exponentially, making them nearly impossible to analyze manually.

6.2 Functionalities

The library provides a layer of abstraction for generalized EWL circuits for arbitrary number of players with customizable base strategies representing the possible moves from the classical counterpart of the game. The initial quantum state can be passed in Dirac notation as a linear combination of kets, e.g. $\left(|00\rangle + i\,|11\rangle\right)/\sqrt{2}$. Based on this information, the library automatically derives the corresponding entanglement operator J and its Hermitian conjugate, $J^\dagger$.

The next step is to define the strategies used by each respective player. Each strategy must be a unitary matrix as it represents a single-qubit quantum gate. The library comes with several built-in parametrizations, including the original one from EWL paper [8] as well as other popular parametrizations with 2 or 3 degrees of freedom as described in [12, 33, 36]. The library has been designed to be easily extendable, allowing users to define and use custom parametrizations.

Among many features of the library, the most useful functionality is the ability to calculate the result quantum state as a vector of amplitudes in the computational basis. Optionally, the result can be further processed by symbolic expressions engine which

cleverly applies well-known trigonometric identities, reduction formulas and Euler's identity in order to simplify the expression.

From the amplitudes one can easily derive the probabilities of possible game results in the observational basis. Similarly to the previous cases, the library also performs expression simplification, often taking advantage of the fact that the parameters are real-valued.

Finally, based on the payoff matrix and previously mentioned probabilities, the library calculates the payoff functions as a linear combination of probabilities and their respective payoffs in a symbolic form.

The library has also been extended to support mixed strategies and apply Kraus operators on density matrices representing quantum state in analytic form. Using the library we were able to successfully recreate the calculations from [35] and obtain function shown in Eq. 7 in a symbolic form as well as visualize the payoff on a 3D plot identical to the one presented in Fig. 4.

6.3 Symbolic calculations and parameters

The true power of the library lies in the symbolic expression engine used, SymPy [25], which is written in Python, open-sourced and licensed under the New BSD license. Its elegant and convenient API makes it easy to handle matrices of symbolic expressions with multiple possibly real-valued parameters.

The `ewl` library directly uses SymPy expressions as input and output interface as well as for simplification of statevector amplitudes and payoff functions, with particular emphasis on trigonometric identities implemented under the hood of SymPy.

Furthermore, the library allows for parametrization of not only player strategies but also initial quantum state, base strategies as well as payoff matrix, so that the analysis can be generalized to an entire family of quantum games. It is also possible to substitute a specific value of parameter at any time for further analysis of a certain case of the quantum game.

The only drawback seems to be the insufficient performance and high memory consumption of SymPy when used for simplification of probabilities of possible outcomes of quantum game with three or more players. This is due to the fact that SymPy is written entirely in Python which is an interpreted language running on a virtual machine. Moreover, SymPy expressions are implemented as immutable objects and thus need to be copied with each transformation. A common practice is to implement computationally expensive operations in compiled languages such as C, C++ or Rust to achieve best possible performance, implement wrapper functions that synchronously call native functions and hide them behind a thin layer of abstraction.

Apart from its functionalities, a particularly noteworthy feature is support for rendering mathematical expressions in graphical form in Jupyter Notebook environment as well as possibility of converting formulas to LaTeX form.

6.4 Qiskit integration

The library integrates with Qiskit, the open-source software development kit created and developed by IBM specifically for interacting with quantum computers and simulators, allowing for verification of theoretical results on numerous available real quantum devices.

Running the Quantum Prisoner's Dilemma on IBM Q was the topic of Filip Galas' master's thesis [15]. At the early stage, Qiskit only offered a set of basic gates which did not include the universal one-bit quantum gate. The main part of the work was the decomposition of J gate into a product of matrices, each of which would be a quantum gate supported by Qiskit.

In 2020, IBM introduced `Operators` class[1], which makes it possible to create universal quantum gates directly from arbitrary NumPy arrays of floating-point numbers, while performing the complex decomposition under the hood.

This particular feature allows `ewl` library to convert arbitrary quantum games in the EWL protocol into a corresponding Qiskit quantum circuits simply by passing NumPy arrays of the matrices of entanglement operator J and Hermitian conjugate $J^\dagger$ along with pure strategies $\{U_i\}_{i=1}^n$ of n players.

The library also greatly simplifies visualizing quantum circuit of the game after transpiling and optimization for a selected backend. Apart from running quantum games on IBM Q devices, it is also possible to execute them on quantum simulators such as statevector simulator or QASM simulator, optionally with specified noise model for quantum gates such as bit flip, phase flip etc. applied during simulation.

6.5 Testing

The implementation of the library is thoroughly tested and covered with unit and functional tests written using PyTest framework [21]. In particular, all quantum games described in chapter 5 are embedded in the library for demonstrative purposes and used as test examples, including verification of payoffs for each combination of base strategies, probabilities in the observational basis as well as known best responses.

It is also worth mentioning that while preparing a set of unit tests to verify the correctness of the calculations performed by the library, we were able to found serious inconsistencies in at least two papers which unfortunately affected further calculations, as well as detect a typo in quite relevant formula in yet another article. These inconsistencies will be described in more detail and corrected in chapter 8 of the thesis.

[1] https://qiskit.org/documentation/tutorials/circuits_advanced/02_operators_overview.html

6.6 Usage

The library can be used both from regular Python scripts as well as in Jupyter Notebook environment (either hosted locally or using IBM Quantum Lab[2]). The package is published on PyPI[3] and can be easily installed in user's environment with a single command. The repository with source code of the library and numerous example notebooks is also available on GitHub[4]. The library is licensed under the MIT License.

6.7 Author contribution

The library was developed in cooperation with Piotr Kotara, who in particular enriched the library with support for mixed strategies as well as added the possibility to specify a custom noise model. The following functionalities were designed and implemented by the author of this thesis:

- symbolic calculation of amplitudes of the state vector, probabilities of game outcomes and expected payoffs for each player for pure strategies
- implementation of popular parametrizations
- support for symbolic parameters and substitutions
- utilities for plotting expected payoff function
- integration with IBM Q simulators and devices via Qiskit
- verification of results and unit testing
- library setup, CI, deployment on PyPI, example notebooks

Based on the results of the experiments performed for this thesis, an abstract for PPAM 2022 conference was submitted, which is attached in Appendix B. We would also like to thank the KQIS organizers for the invitation and the opportunity to present the functionalities of the library as well as our research results[5].

6.8 Summary

In this section we have described `ewl` library which serves as a very useful tool for analysis of quantum games in the EWL protocol, mostly due to its ability to perform complex symbolic calculations as well as seamless integration with real quantum devices. In particular, the library was used to obtain and verify formulas for probabilities of possible quantum game outcomes as well as payoff functions in analytic form that will be presented in the following chapters of the thesis.

[2] https://lab.quantum-computing.ibm.com
[3] https://pypi.org/project/ewl/
[4] https://github.com/tomekzaw/ewl/
[5] https://www.informatyka.agh.edu.pl/pl/blog/kqis-58-piotr-kotara-and-tomasz-zawadzki-software-aided-analysis-of-quantum-games/

7 Algorithms

In this chapter we will present the mathematical foundations of proposed algorithms for finding best responses in parametrized 2×2 quantum games which is preliminary to finding Nash equilibria in pure strategies.

7.1 Finding best responses

Let us denote U as the currently used parametrization function as described in chapter 5 and X as a space of its valid parameter values. Let $\mathbf{x}_\mathrm{A}, \mathbf{x}_\mathrm{B} \in X$ denote vectors of strategy parameters for Alice and Bob, respectively, such that

$$U_\mathrm{A} = U(\mathbf{x}_\mathrm{A}) \text{ and } U_\mathrm{B} = U(\mathbf{x}_\mathrm{B}). \tag{7.1}$$

Let us define Bob's best response for arbitrary Alice's move $\mathbf{x}_\mathrm{A}$ as a function in terms of parameter vectors as

$$\text{best response}_\mathrm{B}(\mathbf{x}_\mathrm{A}) = \underset{\mathbf{x}_\mathrm{B}^* \in X}{\operatorname{argmax}} \$_\mathrm{B}(U(\mathbf{x}_\mathrm{A}), U(\mathbf{x}_\mathrm{B}^*)). \tag{7.2}$$

Alice's best response function with respect to Bob's move $\mathbf{x}_\mathrm{B}$ can be defined analogously as

$$\text{best response}_\mathrm{A}(\mathbf{x}_\mathrm{B}) = \underset{\mathbf{x}_\mathrm{A}^* \in X}{\operatorname{argmax}} \$_\mathrm{A}(U(\mathbf{x}_\mathrm{A}^*), U(\mathbf{x}_\mathrm{B})). \tag{7.3}$$

There always exists at least one best response for given strategy. In a general case, there may exist more than one best response, so the above functions are assumed to return the set of all best responses.

The function maximization problem can be approached symbolically (algorithm 1) or numerically (algorithm 2).

Algorithm 1: Finding best response function symbolically

Input: $U(\mathbf{x})$ – parametrization of players' strategies
X – space of valid parameter values
$\$_\mathrm{B}(\mathbf{x}_\mathrm{A}, \mathbf{x}_\mathrm{B})$ – Bob's expected payoff function
Output: $\text{best response}_\mathrm{B}(\mathbf{x}_\mathrm{A})$ – Bob's best response function
$expr \leftarrow \$_\mathrm{A}(U(\mathbf{x}_\mathrm{A}), U(\mathbf{x}_\mathrm{B}))$
return *SymbolicArgMax*$(expr, \boldsymbol{x}_B, X)$

Algorithm 2: Finding best response numerically

Input: $U(\mathbf{x})$ – parametrization of players' strategies
X – space of valid parameter values
$\$_{\mathrm{B}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Bob's expected payoff function
$\mathbf{x}_{\mathrm{A}}$ – Alice's strategy
Output: $\mathbf{x}_{\mathrm{B}}^{*}$ – Bob's best response for $\mathbf{x}_{\mathrm{A}}$
$p(\mathbf{x}_{\mathrm{B}}) := \$_{\mathrm{A}}(U(\mathbf{x}_{\mathrm{A}}), U(\mathbf{x}_{\mathrm{B}}))$
return *NumericMaximize*$(p, \boldsymbol{x}_B, X)$

In the specific case of two-player quantum game with full $\mathrm{SU}(2)$ parametrization, for arbitrary strategy of the opponent, there always exists a winning strategy that affects the outcome of the game in such a way as to reach the quantum state with the largest payoff from the payoff matrix. In such case, it is sufficient to find parameters such that the probability of the game outcome being the one with largest payoff is equal to one, similarly as shown in [35]. Thanks to this observation, we can significantly decrease the complexity and thus difficulty of the task, reducing the problem to a single equation, or alternatively to a system of equations with the zero vector as the right-hand side.

For instance, in Quantum Prisoner's Dilemma, the state with maximum payoff for Bob is $|01\rangle$. Therefore, in order to obtain the formula for Bob's best response, it is sufficient to find the solutions of the equation

$$p_{01}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 1 \tag{7.4}$$

or, alternatively, solve the following system of equations

$$\begin{cases} p_{00}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 0 \\ p_{10}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 0 \\ p_{11}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 0 \end{cases} \tag{7.5}$$

for variables $\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}$ as a function of parameters $\theta_{\mathrm{A}}, \phi_{\mathrm{A}}, \alpha_{\mathrm{A}}$ representing arbitrary but known Alice's move.

In the general case of a quantum game using parametrization with 1 or 2 degrees of freedom, the method described above cannot be directly applied. This is due to the fact that the set of possible strategies is a proper subspace of $\mathrm{SU}(2)$ group and therefore may not include the winning strategy. In other words, for some strategies of the opponent, there may not exist a valid vector of parameters that gives the best possible payoff from the payoff matrix.

A fairly natural idea would be to first maximize the probability corresponding to the quantum state representing the outcome of the game with the highest payoff from the payoff matrix, then maximize the probability of the second best state, and so on. However, this algorithm does not always produce optimal solutions. Consider the following counterexample with $[0, 0.9, 0.1, 0]$ and $[0.2, 0.8, 0, 0]$ representing two

vectors of probabilities of possible game outcomes with payoffs $[3, 5, 0, 1]$, respectively. Although the probability of the state with the best payout is higher in the first case since $0.9 > 0.8$, the expected payoff is worse as $4.5 < 4.6$.

Therefore, in the general case, the only valid method so far seems to be the maximization of the entire expected payoff function, which is a linear combination of the probabilities of all possible game outcomes and the corresponding payoffs from the game matrix. As it turns out, symbolic maximization of such parametrized trigonometric expressions is not a simple task. Numerical methods, on the other hand, not only may miss a global maximum, but also heavily depend on the choice of starting point.

7.2 Nash equilibria as fixed points

In a two-player game, Nash equilibrium is a pair of strategies $(U_\mathrm{A}, U_\mathrm{B})$ such that U_A is the best response for U_B and simultaneously U_B is the best response for U_A.

A pair of strategies $(U(\mathbf{x}_\mathrm{A}), U(\mathbf{x}_\mathrm{B}))$ is a Nash equilibrium if

$$\mathbf{x}_\mathrm{B} \in \text{best response}_\mathrm{B}(\mathbf{x}_\mathrm{A}) \quad \text{and} \quad \mathbf{x}_\mathrm{A} \in \text{best response}_\mathrm{A}(\mathbf{x}_\mathrm{B}). \tag{7.6}$$

Assuming that only a single best response exists, the following pair of statements can be merged into

$$\mathbf{x}_\mathrm{A} = \text{best response}_\mathrm{A}(\text{best response}_\mathrm{B}(\mathbf{x}_\mathrm{A})), \tag{7.7}$$

which can be expressed as a fixed point equation

$$\mathbf{x}_\mathrm{A} = f(\mathbf{x}_\mathrm{A}) \tag{7.8}$$

where $f = (\text{best response}_\mathrm{A} \circ \text{best response}_\mathrm{B})$ is a composition of best response functions, $\mathbf{x}_\mathrm{A}$ is a fixed point of f, and $\mathbf{x}_\mathrm{B} = \text{best response}_\mathrm{B}(\mathbf{x}_\mathrm{A})$.

In case of a symmetric 2×2 game, $\text{best response}_\mathrm{A} = \text{best response}_\mathrm{B}$, so effectively $f = \text{best response}^2$, or alternatively $\mathbf{x}_\mathrm{A}$ is a 2-periodic point of best response function.

Similarly to the previously described task of finding best response functions, this problem can also be approached symbolically or numerically. If the analytic form of best response function is known, strategies that may be part of Nash equilibria can be easily obtained as symbolic expressions by solving the fixed point equation

$$\mathbf{x}_\mathrm{A} - f(\mathbf{x}_\mathrm{A}) = \mathbf{0}. \tag{7.9}$$

In a general case, if there exists m and n different best responses of Alice and Bob, respectively, it is necessary to solve the equation for each combination of best response functions and combine the results as shown in algorithm 3, i.e.

$$\bigcup_{i=1}^{m} \bigcup_{j=1}^{n} \left\{ \mathbf{x}_\mathrm{A} \in X \mid \mathbf{x}_\mathrm{A} - \text{best response}_\mathrm{A}^{(i)}(\text{best response}_\mathrm{B}^{(j)}(\mathbf{x}_\mathrm{A})) = \mathbf{0} \right\}. \tag{7.10}$$

Otherwise, if the analytic form of the best response function is unknown, numerous numerical methods of fixed-point search can be employed in order to find Nash equilibria. However, due to discrete nature of numerical computing, this requires searching the entire grid of valid parameter values constrained by appropriate boundaries or running the algorithm multiple number of times with randomizing the initial strategy, as proposed in algorithm 4.

Algorithm 3: Finding Nash equilibria symbolically

Input: $U(\mathbf{x})$ – parametrization of players' strategies
X – space of valid parameter values
$\$_{\mathrm{A}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Alice's expected payoff function
$\$_{\mathrm{B}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Bob's expected payoff function
Output: E – list of Nash equilibria in pure strategies

```
E ← []
for br_A ∈ SymbolicBestResponseFunctions(U, X, $_A) do
    for br_B ∈ SymbolicBestResponseFunctions(U, X, $_B) do
        expr ← br_A(br_B(x_A)) − x_A
        s ← SymbolicSolveEquation(expr, 0, x_A)
        E.insert(s)
    end
end
return E
```

Algorithm 4: Finding Nash equilibria numerically

Input: $U(\mathbf{x})$ – parametrization of players' strategies
X – space of valid parameter values
$\$_{\mathrm{A}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Alice's expected payoff function
$\$_{\mathrm{B}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Bob's expected payoff function
N – number of iterations of the algorithm
Output: E – list of found Nash equilibria in pure strategies

```
E ← []
i ← 0
while i < N do
    x_A ← RandomStrategy(X)
    x_B ← NumericBestResponse(U, X, x_A, $_B)
    x'_A ← NumericBestResponse(U, X, x_B, $_A)
    if U(x'_A) ≈ U(x_A) then
        E.insert((x_A, x_B))
    end
    i ← i + 1
end
return E
```

7.3 Mixed strategies based on best response cycles

Assuming that $U_\mathrm{A} \to U_\mathrm{B} \to U_\mathrm{A}$, a pure Nash equilibrium can be seen as a special case of best response cycle. However, in some quantum games there are no such cycles of length 2 and thus no Nash equilibria in pure strategies. The concept two mutual best responses in two-player games may be easily generalized to cycles of length $2n$ such that

$$U_\mathrm{A}^{(1)} \xrightarrow{\mathrm{B}} U_\mathrm{B}^{(1)} \xrightarrow{\mathrm{A}} U_\mathrm{A}^{(2)} \xrightarrow{\mathrm{B}} U_\mathrm{B}^{(2)} \xrightarrow{\mathrm{A}} \ldots \xrightarrow{\mathrm{A}} U_\mathrm{A}^{(n)} \xrightarrow{\mathrm{B}} U_\mathrm{B}^{(n)} \xrightarrow{\mathrm{A}} U_\mathrm{A}^{(1)} \tag{7.11}$$

where $X \xrightarrow{\mathrm{A}} Y$ denotes that $Y \in \text{best response}_\mathrm{A}(X)$ and analogously for $\xrightarrow{\mathrm{B}}$.

A proposed symbolic approach towards finding best response cycles symbolically is shown in algorithm 5, whereas the numerical approach is presented as algorithm 6.

Algorithm 5: Finding best response cycles symbolically

Input: $U(\mathbf{x})$ – parametrization of players' strategies
X – space of valid parameter values
$\$_\mathrm{A}(\mathbf{x}_\mathrm{A}, \mathbf{x}_\mathrm{B})$ – Alice's expected payoff function
$\$_\mathrm{B}(\mathbf{x}_\mathrm{A}, \mathbf{x}_\mathrm{B})$ – Bob's expected payoff function
L – length of best response cycle
Output: C – list of best response cycles in pure strategies

```
C ← []
brs_A[m] ← SymbolicBestResponseFunctions(U, X, $_A)
brs_B[n] ← SymbolicBestResponseFunctions(U, X, $_B)
for seq_A ∈ (ℤ_m)^{L/2} do
    for seq_B ∈ (ℤ_n)^{L/2} do
        expr ← x_A
        i ← 0
        while i < L/2 do
            br_A ← brs_A[seq_A[i]]
            br_B ← brs_B[seq_B[i]]
            expr ← br_A(br_B(expr))
            i ← i + 1
        end
        expr ← expr − x_A
        s ← SymbolicSolveEquation(expr, 0, x_A)
        C.insert(s)
    end
end
return C
```

Algorithm 6: Finding best response cycle numerically

Input: $U(\mathbf{x})$ – parametrization of players' strategies
X – space of valid parameter values
$\$_{\mathrm{A}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Alice's expected payoff function
$\$_{\mathrm{B}}(\mathbf{x}_{\mathrm{A}}, \mathbf{x}_{\mathrm{B}})$ – Bob's expected payoff function
Output: C – best response cycle found

```
i ← 0
x_A ← RandomStrategy(X)
prev ← [x_A]
while True do
    x_B ← NumericBestResponse(U, X, x_A, $_B)
    prev.insert(x_B)
    x_A ← NumericBestResponse(U, X, x_B, $_A)
    prev.insert(x_A)
    i ← i + 1
    j ← 0
    while j ≤ i do
        if U(x_A) ≈ U(prev[2 * j]) then
            return prev[2 * j : 2 * i]
        end
        j ← j + 1
    end
end
```

Having found a cycle of best responses, one can construct the following mixed quantum strategies

$$\left\{\left(p_i, U_{\mathrm{A}}^{(i)}\right)\right\}_{i=1}^{n} \quad \text{and} \quad \left\{\left(q_i, U_{\mathrm{B}}^{(i)}\right)\right\}_{i=1}^{n} \tag{7.12}$$

for Alice and Bob, respectively, each consisting of n pure strategies, where $p_i, q_i \in [0, 1]$ are mixing probabilities such that

$$\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i = 1. \tag{7.13}$$

As shown in [6], in Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization there exists a best response cycle of length 4. As described in [35], alternating pairs of these pure strategies form two mixed strategies of Alice and Bob with a mixed quantum Nash equilibrium located on the saddle point of the payoff function where $p_i = q_i = \frac{1}{2}$. This specific example is discussed in detail and further analyzed by Piotr Kotara in his master's thesis [20].

This work, however, focuses primarily on algorithmic methods of finding such best response cycles, either assuming that the best response function is already known in analytic form or without such knowledge, taking into account that there may exist more than one best response.

7.4 Analysis of the complexity

When proposing algorithms, one should also estimate its computational as well as space complexity in terms of size of the input.

As for symbolic algorithms, their computational and space complexity heavily depends on the size and difficulty of symbolic expressions used for global optimization or equation solving as well as on the complexity of specific implementations used to solve these problems.

If there exists m and n unique best responses of Alice and Bob, respectively, the computational complexity of algorithm 3 is $O(mn \cdot f(p))$ where $f(p)$ is the computational complexity function of solving a single equation symbolically. For finding best response cycles of length L, the computational complexity of algorithm 5 is $O(m^{L/2}n^{L/2} \cdot f(p))$.

For numerical algorithms, a reasonable parameter of the complexity function seems to be the number of variables (i.e. strategy parameters). However, single-qubit parametrizations have at most 3 degrees of freedom. Moreover, the numerical methods of finding best response or Nash equilibria proposed in this chapter are strongly based on Powell or Nelder-Mead optimization algorithms, so they naturally inherit their computational complexity.

If N denotes the number of iterations to find a Nash equilibrium, the computational complexity of algorithm 4 is $O(N \cdot f(p))$ where $f(p)$ is the computational complexity of finding a global maximum of payoff function numerically.

If $g(p)$ represents the length of numerical best response cycle and, the computational complexity of algorithm 6 is $O(f(p) \cdot g(p)^2)$, but can be reduced to $O(f(p) \cdot g(p))$ using a hash table.

7.5 Summary

In this chapter we presented symbolic and numerical algorithms for finding best responses, Nash equilibria in pure states as well as best response cycles in two-player quantum games. These algorithms which will be evaluated and compared later in the experimental part of the thesis.

8 Experiments

In this chapter, we will describe the experiments that were conducted as part of our research on software-aided analysis quantum games. We will start from running specific examples of quantum games on quantum simulators and real quantum devices in order to verify the theoretical results. Next, we will analyze and evaluate symbolic and numerical methods for finding best responses, best response cycles and Nash equilibria.

8.1 Environment

Unless stated otherwise, all experiments described in this chapter of the thesis were carried out on a 14-inch MacBook Pro (2021) with 8-core Apple M1 Pro CPU, 14-core GPU, 16-core Neural Engine, 16GB unified memory while connected to 67W USB-C Power Adapter plugged to the power supply, running macOS Monterey 12.2.1.

The computational part was carried out using Python 3.9.2 via Jupyter Notebook environment with SymPy 1.9 or, independently, using Mathematica 13.0.1. Due to the fact that one of the low-level dependencies of Qiskit are not yet compatible with M1-based Macs, some of the experiments interfacing with IBM Q simulators or devices were executed by Qiskit 0.34.2 running on Python 3.8.9 via dynamic binary translator Rosetta 2.

8.2 Running two-player variant of Quantum Prisoner's Dilemma on IBM Q

As a first experiment, we will run a quantum game on a real quantum device and compare theoretical results with actual measurements. Running Quantum Prisoner's Dilemma on IBM Q was the topic of Filip Galas' master's thesis [15]. At the time, the main limitation was the set of available quantum gates, which meant that Filip had to manually decompose the matrix of entanglement operator J as well as its Hermitian conjugate $J^\dagger$ into products of simple unitary matrices, each of which would be convertible into a quantum gate supported by Qiskit.

As mentioned in section 6.4, as of 2020, Qiskit supports creating quantum operators from arbitrary unitary matrices of floating-point numbers. The purpose of the experiment was to test this particular feature while using the convenient interface provided by the `ewl` library as well as compare the results afterwards.

Figure 8.1 shows the quantum circuit of a two-player quantum game in the EWL protocol. Two horizontal lines represent two individual qubits, q_0 and q_1, both initialized

to $|0\rangle$. The purple rectangles represent the quantum gates of entanglement operator J, two players' strategies U_0 and U_1, and Hermitian conjugate of previously mentioned entanglement operator, $J^\dagger$, respectively. The vertical light gray bars represent the barriers which divide the circuit into chunks, preventing the transpiler from applying optimizations between different parts of the circuit. Finally, the black squares represent the measurement blocks which collapse the quantum state into either $|0\rangle$ or $|1\rangle$ and then copy the results from individual qubits into the corresponding bits of the classical register marked as double horizontal line.

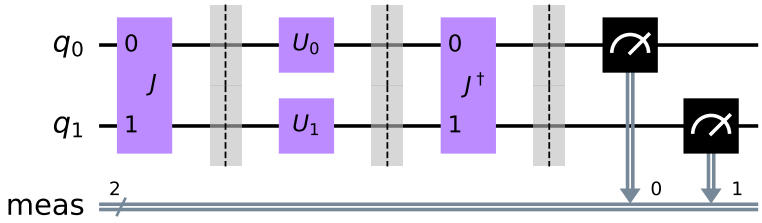

Figure 8.1: Visualization of the quantum circuit of a two-player quantum game in the EWL protocol created using Qiskit

Before a quantum circuit can be run on a real quantum device, first it must be transpiled, i.e. adapted to its architecture. During transpilation, a generic quantum circuit is converted into an equivalent one that uses only the set of quantum gates available on a particular quantum device. This process usually results in a noticeable increase in the number of quantum gates resulting from the decomposition of more complex gates into the basic ones. Transpilation is performed automatically when the job is submitted but can be also triggered manually, for instance for demonstrative purposes. Figure 8.2 shows the quantum circuit from figure 8.1 after transpiling for `ibmq_quito` quantum device. Since this backend is a 5-qubit system, the other three qubits are marked as *ancilla* (additional).

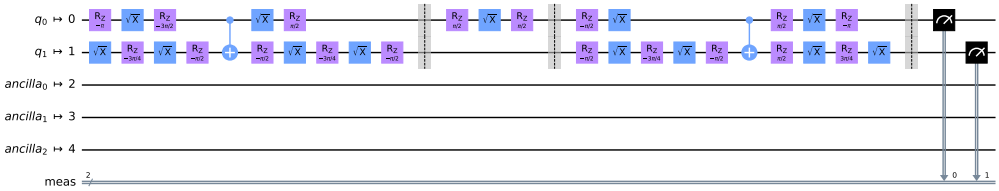

Figure 8.2: Visualization of the quantum circuit of a two-player quantum game in the EWL protocol after transpilation for `ibmq_quito` backend

In order to execute a quantum circuit on a real quantum device, one needs to submit a job using the secret API key assigned to his IBM Quantum[1] account to enqueue the experiment. Depending on the workload, the waiting time can range from a few minutes up to several hours.

Since quantum circuits are terminated with measurement blocks, the result will always be a classical bitstring (in this case 00, 01, 10 or 11, with different probabilities). Thus, in order to experimentally validate theoretical results, quantum circuits need to be run certain number of times to obtain the number of occurrences of each final state. The default number of shots is 1024.

Apart from executing a quantum circuit on a real quantum device, Qiskit also provides various tools for quantum simulation. Statevector simulator numerically calculates the exact state of the quantum system as Schrödinger wavefunction after running the circuit and returns the observational basis probabilities as a vector of 2^n floating-point numbers where n is the number of qubits. In general, for more complex quantum circuits, the computations can be performed using GPU instead of CPU.

Another quantum simulation tool available in Qiskit is QASM simulator. By default, it behaves like a perfect quantum computer with randomness but without noises. However, it can be also configured to mimic the behaviour of a real quantum device by using the provided noise model.

Table 8.1 shows the results of playing the Quantum Prisoner's Dilemma using $U(\theta, \alpha, \beta)$ parametrization with Alice playing quantum strategy $U(\frac{\pi}{2}, \frac{\pi}{2}, 0)$ while Bob plays classical strategy $U(0, 0, 0) = C$. The results obtained from the statevector simulator exactly match the probabilities calculated by the `ewl` library. The distribution of game results obtained from QASM simulator also seems to be reasonable since for final states 01 and 11, deviations from the expected value of 0.5 occur due to randomness – similarly like flipping a coin 100 times does not always result in heads falling exactly 50 times. For states 00 and 10, the results are also both equal to zero, beacuse the ideal QASM simulator (i.e. without noise model) does not produce any quantum errors and thus cannot reach a state with zero probability. Finally, the results obtained from a real quantum device `ibmq_quito` are quite close to the theoretical results, but include quantum errors from decoherence, which are especially noticeable in the results for states 00 and 10. For convenience, the results are visualized in Fig. 8.3.

Table 8.1: Probabilities and distribution of results of running the Quantum Prisoner's Dilemma on two different quantum simulators and on a real quantum device

Final state	**Theoretical results**	**Statevector simulator**	**QASM simulator**	**Quantum device**
00	0	0	0	0.030
01	0	0	0	0.038
10	0.5	0.5	0.507	0.479
11	0.5	0.5	0.493	0.453

[1] https://quantum-computing.ibm.com/

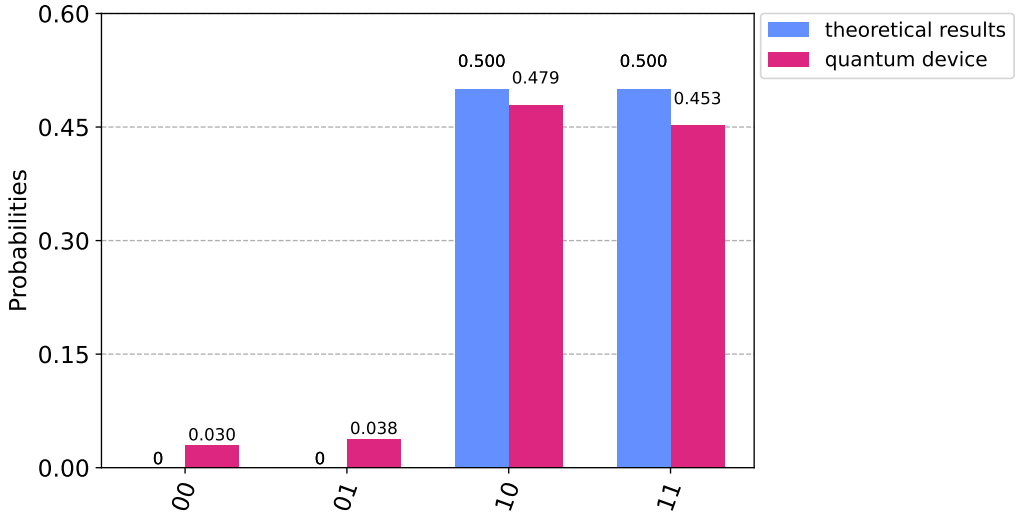

Figure 8.3: Comparision of the results of running the Quantum Prisoner's Dilemma on QASM simulator and `ibmq_quito` backend

8.3 Visualization of payoff function

As mentioned in chapter 5, a particularly useful functionality of the `ewl` library is the ability to symbolically calculate the observational basis probabilities of the outcomes and thus obtain the payoff function in analytic form. The objective of this experiment was to test out the symbolic capabilities of the library when it comes to computing payoff function in a quantum game with a custom conditional parametrization.

As the use case for this experiment we have chosen Alice's expected payoff function in Quantum Prisoner's Dilemma with a custom single-degree-of-freedom parametrization defined in Eq. 5.2 in [15] as

$$f(s) = \begin{cases} U(\pi - 2s\pi, 0, 0) & \text{for } s \in [0, \frac{1}{2}] \\ U(0, \left(s - \frac{1}{2}\right)\pi, \left(s - \frac{1}{2}\right)\pi) & \text{for } s \in (\frac{1}{2}, 1] \end{cases} \tag{8.1}$$

where U is the parametrization from Eq. 4.10. In particular, for certain values of parameter s, the strategy becomes

$$\begin{aligned} f(0) &= U(\pi, 0, 0) = X \\ f(\tfrac{1}{2}) &= U(0, 0, 0) = I \\ f(1) &= U(0, \tfrac{\pi}{2}, \tfrac{\pi}{2}) = Z. \end{aligned} \tag{8.2}$$

As mentioned in [15], the payoff values for the graph were calculated numerically by running IBM Q simulator for each point of the 20×20 grid. In this experiment we aim at obtaining the expected Alice's payoff function in analytic form as well as visualizing it on a three-dimensional plot.

Let us try to define this game using `ewl` library in order to obtain the payoff function in symbolic form. Assuming that Alice plays $U_\mathrm{A} = f(s_1)$ and Bob plays $U_\mathrm{B} = f(s_2)$ where $s_1, s_2 \in [0, 1]$, the first player's payoff function is, after simplification

$$\$_\mathrm{A}(s_1, s_2) = \begin{cases} -\sin^2(\pi s_1)\sin^2(\pi s_2) + 4\sin^2(\pi s_1) - \sin^2(\pi s_2) + 1 & \text{for } s_1 \leq \frac{1}{2} \wedge s_2 \leq \frac{1}{2} \\ 7\sin^2(\pi s_1)\sin^2(\pi s_2) - 4\sin^2(\pi s_1) - 5\sin^2(\pi s_2) + 5 & \text{for } s_1 \leq \frac{1}{2} \\ -3\sin^2(\pi s_1)\sin^2(\pi s_2) + 5\sin^2(\pi s_1) + \sin^2(\pi s_2) & \text{for } s_2 \leq \frac{1}{2} \\ 3\sin^2(\pi s_1)\sin^2(\pi s_2) + \sin^2(\pi(s_1 + s_2)) & \text{for } s_1 \leq \frac{1}{2} \vee s_2 \leq \frac{1}{2} \\ \cos(\pi(2s_1 + 2s_2)) + 2 & \text{otherwise} \end{cases} . \tag{8.3}$$

It is worth noting that similarly to the used parametrization, the resulting payoff function is also defined by the cases. Nevertheless, SymPy engine still manages to gracefully apply trigonometric simplifications. Finally, based on the payoff function in the symbolic form, the library is able to reproduce the 3D plot from Fig 5.1 in [15].

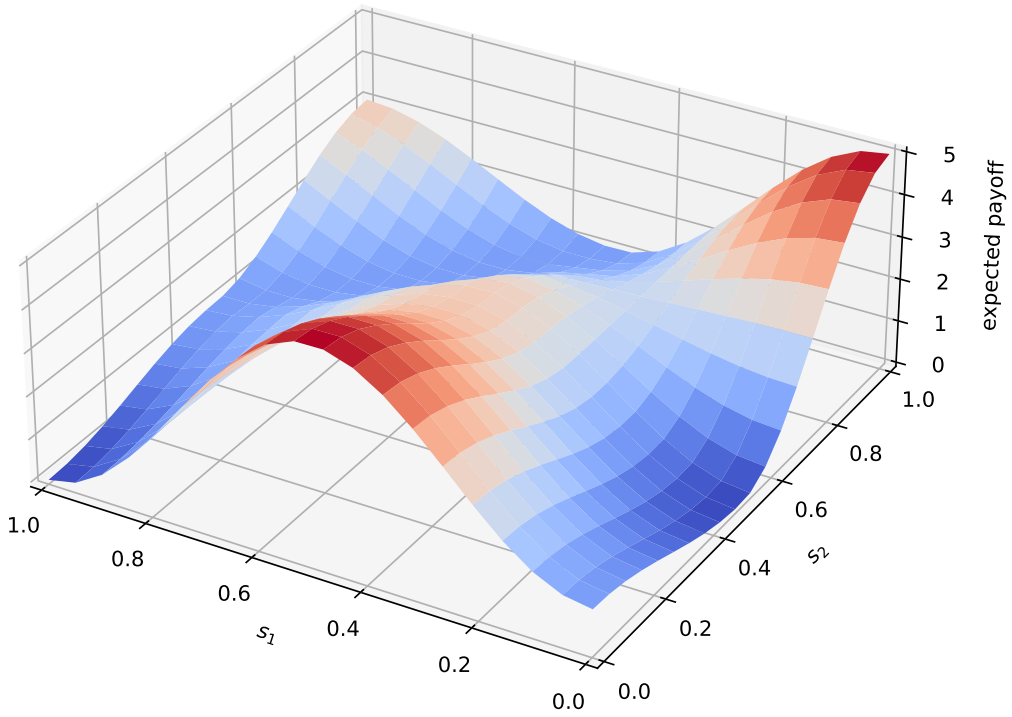

Figure 8.4: Alice's expected payoff function plot in Quantum Prisoner's Dilemma with respect to strategy parameters s_1, s_2 (compare to Fig. 5.1 in [15])

8.4 Analysis of the case with three players

Since the library has proven itself for the two-player variant of the quantum game, let us observe how it behaves when used for analysis of the case with three players. Assume that Alice, Bob and Charlie play the following strategies:

$$\begin{aligned} U_1 &= U(\theta_1, \alpha_1, \beta_1) \\ U_2 &= U(\theta_2, \alpha_2, \beta_2) \\ U_3 &= U(\theta_3, \alpha_3, \beta_3) \end{aligned} \tag{8.4}$$

where U is parametrization from Eq. 4.8. Apparently, the library was able to successfully calculate the amplitudes of the final statevector and convert them to observational basis probabilities of 8 possible game outcomes (i.e. $|000\rangle, |001\rangle, \ldots, |111\rangle$, respectively) as well as apply exponential and trigonometric simplifications using SymPy:

$$\begin{bmatrix} \sin^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\beta_1+\beta_2+\beta_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\alpha_1+\alpha_2+\alpha_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\beta_1+\beta_2-\alpha_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\alpha_1+\alpha_2-\beta_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\beta_1-\alpha_2+\beta_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\alpha_1-\beta_2+\alpha_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\beta_1-\alpha_2-\alpha_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\alpha_1-\beta_2-\beta_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\beta_1-\alpha_2-\alpha_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\alpha_1-\beta_2-\beta_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\beta_1-\alpha_2+\beta_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\alpha_1-\beta_2+\alpha_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\beta_1+\beta_2-\alpha_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\alpha_1+\alpha_2-\beta_3\right) \\ \sin^2\left(\frac{\theta_1}{2}\right)\sin^2\left(\frac{\theta_2}{2}\right)\sin^2\left(\frac{\theta_3}{2}\right)\cos^2\left(\beta_1+\beta_2+\beta_3\right)+\cos^2\left(\frac{\theta_1}{2}\right)\cos^2\left(\frac{\theta_2}{2}\right)\cos^2\left(\frac{\theta_3}{2}\right)\sin^2\left(\alpha_1+\alpha_2+\alpha_3\right) \end{bmatrix}. \tag{8.5}$$

Currently, the validity of the above formula cannot be verified independently due to the fact that no articles on three-player quantum games in the EWL protocols have been found by the author. However, for each possible combination of the classical strategies, $\{C, D\}^3$, the actual results match the expected ones. Moreover, the probabilities sum up to 1 for arbitrary values of parameters $\{\theta_i, \alpha_i, \beta_i\}_{i=1}^3$.

It is worth mentioning that it took 5 min 8 s on average to calculate the observational basis probabilities for three-player variant of the quantum game, compared to only 1.6 s for the two-player case. However, the actual symbolic operations, including tensor product calculation, matrix multiplication and squaring the statevector amplitudes, took only 55 ms on average. The remaining 99.9% of the execution time was spent on exponential and trigonometric simplification. However, given the complexity of this computation, the execution time of a few minutes is still acceptable, especially considering how long it would take a human as well as how prone to human error it would be. Moreover, the solution in the form of an analytic function depending on the strategy parameters carries more information than a numerical solution for specific values. The scalability of the symbolic part of `ewl` library with respect to the number of players for separable, partially entangled and maximally entangled initial quantum states was measured in Piotr Kotara's master thesis [20].

8.5 Running three-player variant of Quantum Prisoner's Dilemma on IBM Q

Having calculated the formulas for the observational basis probabilities of all possible game outcomes depending on the parameters of strategies of three players in section 8.4, let us run a specific case of such quantum game on a real quantum device. Figure 8.5 shows the general quantum circuit of a quantum game in the EWL protocol with three players.

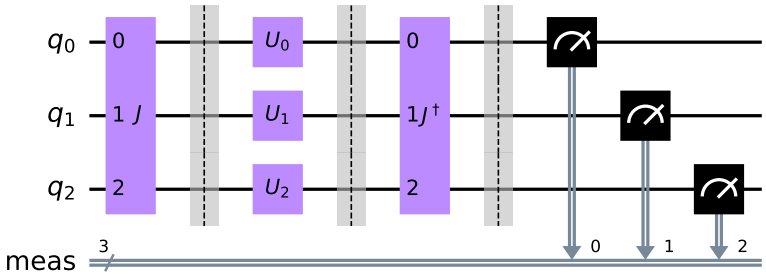

Figure 8.5: Visualization of the quantum circuit of a three-player quantum game in the EWL protocol created using Qiskit

Similarly as in section 8.2, the quantum game was executed on statevector simulator, QASM simulator and `ibmq_quito` backend. Table 8.2 shows the results of playing using $U(\theta, \alpha, \beta)$ parametrization where Alice plays $U(\frac{\pi}{2}, \frac{\pi}{2}, 0)$, Bob plays $U(\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{4})$ and Charlie plays $U(0, 0, 0) = C$. For convenience, the results are also visualized in a bar plot presented in Fig. 8.6.

Table 8.2: Probabilities and distribution of results of running the three-player quantum game in the EWL protocol on two different quantum simulators and on a real quantum device

Final state	Theoretical results	Statevector simulator	QASM simulator	Quantum device
000	0.25	0.25	0.265	0.155
001	0.125	0.125	0.146	0.159
010	0.125	0.125	0.115	0.138
011	0.25	0.25	0.236	0.135
100	0	0	0	0.085
101	0.125	0.125	0.121	0.135
110	0.125	0.125	0.117	0.105
111	0	0	0	0.089

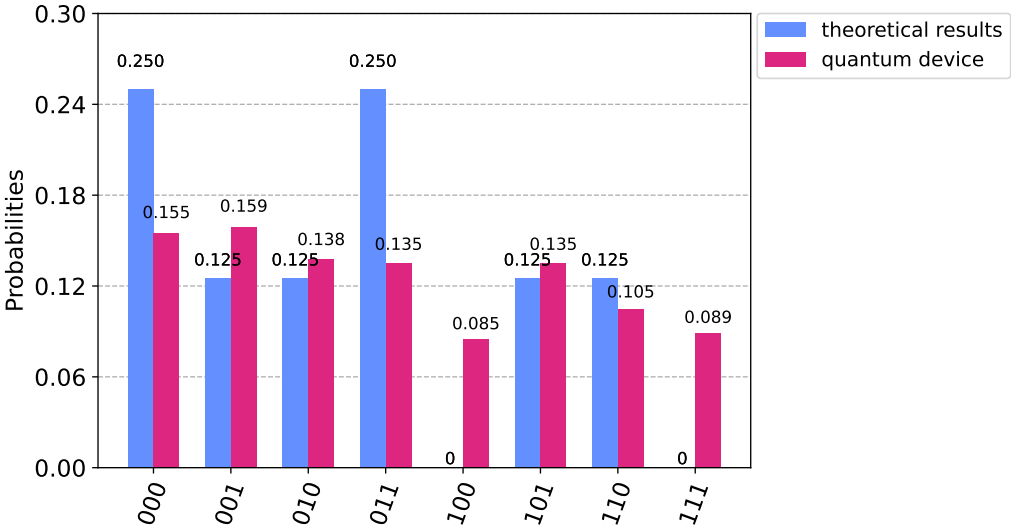

Figure 8.6: Comparision of the results of running the three-player quantum game on QASM simulator and `ibmq_quito` backend

Like previously, the output of statevector simulator exactly matches the probabilities calculated by the `ewl` library, confirming the correctness of the formulas for this particular case. The distribution of game outcomes obtained from QASM simulator is close to theoretical results and the deviations occur due to randomness which is naturally incorporated into this simulation technique.

However, when it comes to running three-player quantum game on a real quantum device, the distribution of game outcomes is far from ideal. For instance, let us consider two final states, $|000\rangle$ and $|100\rangle$, whose theoretical probabilities are 0.250 and 0, respectively. On `ibmq_quito` the frequencies of these two game outcomes were 0.195 and 0.085, correspondingly. In both cases, the results are off by approximately 0.1 which is a significant error.

Let us also consider final states $|001\rangle$ and $|011\rangle$ with expected probabilities of 0.125 and 0.25, respectively. On `ibmq_quito`, the actual rates of occurrence were 0.159 and 0.135, correspondingly, meaning that the game outcome with twice the probability was actually reached fewer times during the experiment.

Unlike the two-player case, running the three-player variant of quantum game on a real quantum device `ibmq_quito` did not yield satisfactory results. Most likely, the root cause of such significant errors is high decoherence due to the length of the transpiled version of the quantum circuit in terms of number of quantum gates. In [20] it is shown that the default transpiler produces a lengthy, non-optimal quantum circuit with even for a separable entanglement operator J. For reference, the actual quantum circuit after transpiling and optimization is presented in Fig. 8.7.

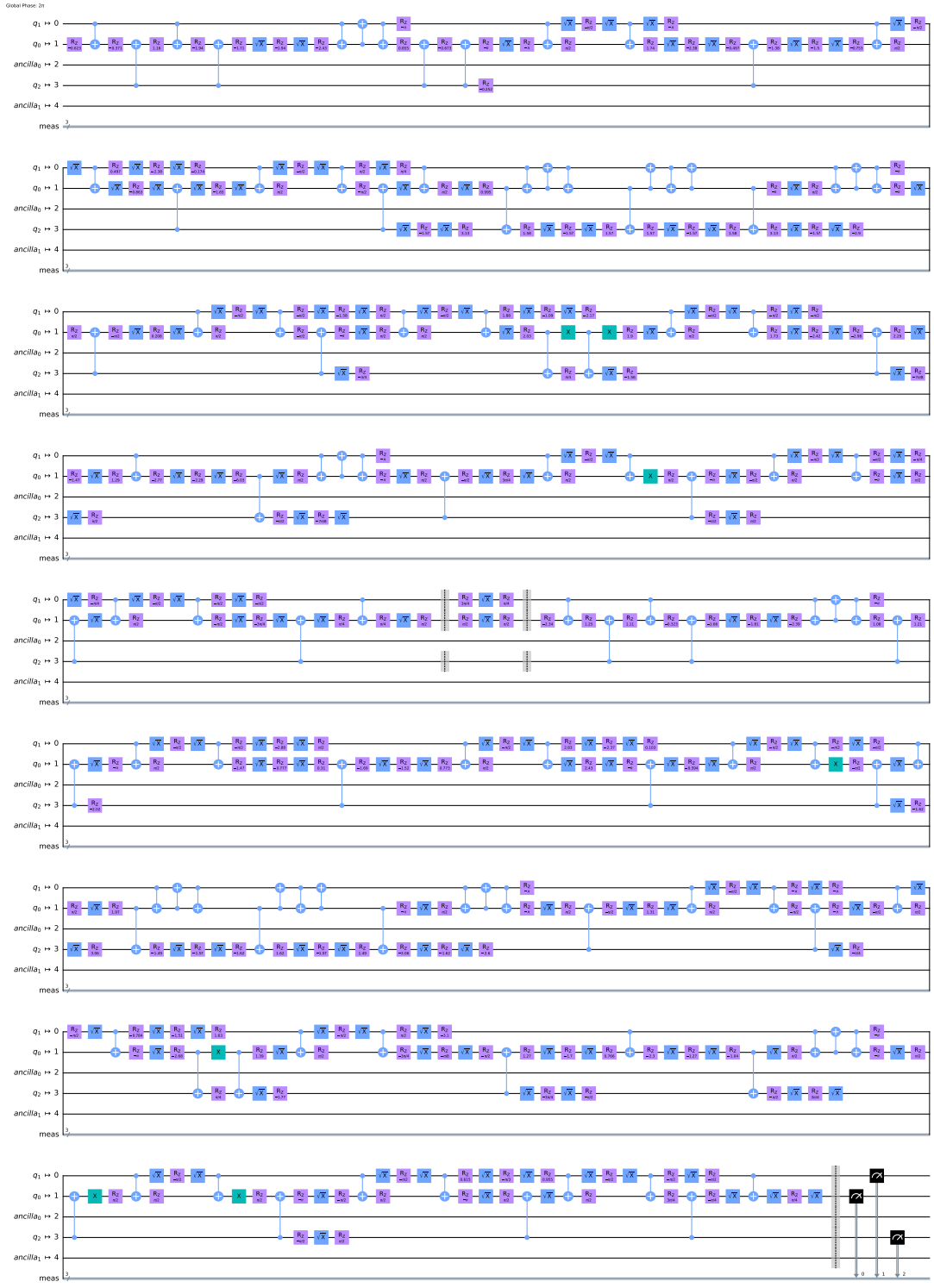

Figure 8.7: Visualization of the quantum circuit of a three-player quantum game in the EWL protocol after transpilation for `ibmq_quito` backend

Another important factor is the architecture of the quantum device itself. During transpilation, quantum gates operating on 3 or more qubits are decomposed into 1- or 2-qubit gates supported by the underlying architecture of the quantum device. For instance, `ibmq_quito` supports only the following 5 gates:

- `CX` – controlled-X (CNOT) gate $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$,
- `ID` – identity gate $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,
- `RZ` – rotation along z-axis, $RZ(\lambda) = \begin{bmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{bmatrix}$,
- `SX` – Sqrt-X gate, $\sqrt{X} = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}$,
- `X` – Pauli-X gate $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

Moreover, there are some limitations related to CNOT gate. Figure 8.8 shows the topology of three different 5-qubit IBM Q systems, including `ibmq_quito`, where qubits are represented with graph nodes and edges between them denote the possibility of applying CNOT gate on a specific pair of qubits. As shown in Fig. 8.7, while transpiling for `ibmq_quito`, qubits 0, 1 and 3 are chosen to represent the quantum state of the game, whereas qubits 2 and 4 are marked as auxiliary. As a consequence, CNOT gate cannot be directly applied between qubits 0 and 3. Instead, CNOT(q_0, q_3) gets decomposed into SWAP(q_0, q_1), CNOT(q_1, q_3) and SWAP(q_0, q_1). Then, each SWAP(q_0, q_1) can be further transpiled into CNOT(q_0, q_1), CNOT(q_1, q_0) and CNOT(q_0, q_1).

For such purposes, `ibmq_yorktown` backend would be definitely a better choice because of the fact that its qubit layout contains three nodes which form a triangle. Unfortunately, it is no longer available for public use.

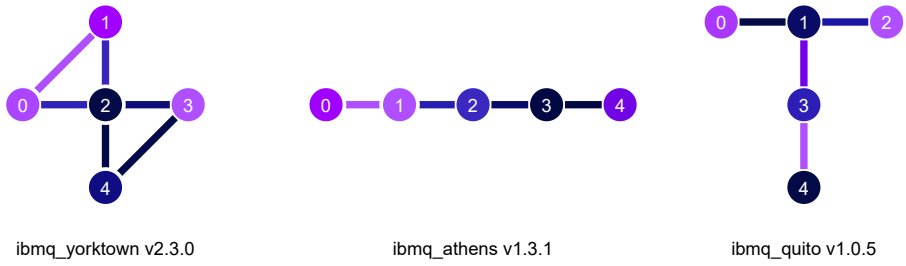

Figure 8.8: Topology diagrams of three 5-qubit IBM Q systems (`ibmq_yorktown`, `ibmq_athens` and `ibmq_quito`) with non-isomorphic qubit layouts

8.6 Verification of published formulas

One of the features of `ewl` library is the ability to symbolically calculate formulas which are critial for further processing such as observational basis probabilities of game outcomes or payoff functions for each respective player. Therefore, before we proceed to finding best responses and Nash equilibria for sample quantum games introduced in chapter 5, let us verify the correctness of the formulas calculated by `ewl` library in comparision to the results published in the references. The results were summarized in Tab. 8.3.

Table 8.3: Consistency of formulas calculated using `ewl` library with results from publications for different variants of Quantum Prisoner's Dilemma

Parametrization	Reference	Equality of formulas
$U(\theta, \alpha, \beta)$	[36]	✓
$U(\theta, \phi, \alpha)$	[6]	✗
$U(\theta, \phi, \alpha)$	[35]	✗
$U(\theta, \phi)$	[12]	✓*

Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization was described in section 5.3 with the probabilities of game outcomes presented on page 5 in [6]:

$$\begin{aligned}
&\left(\cos\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A+\phi_B\right)-\sin\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\sin\left(\alpha_A+\alpha_B\right)\right)^2\\
&\left(\sin\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\cos\left(\alpha_A-\phi_B\right)-\cos\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A-\alpha_B\right)\right)^2\\
&\left(\sin\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\sin\left(\alpha_A-\phi_B\right)+\cos\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A-\alpha_B\right)\right)^2\\
&\left(\cos\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A+\phi_B\right)+\sin\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\cos\left(\alpha_A+\alpha_B\right)\right)^2.
\end{aligned} \tag{8.6}$$

However, the results from `ewl` library are as follows:

$$\begin{aligned}
&\left(\cos\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A+\phi_B\right)-\sin\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\sin\left(\alpha_A+\alpha_B\right)\right)^2\\
&\left(\sin\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\cos\left(\alpha_A+\phi_B\right)+\cos\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A+\alpha_B\right)\right)^2\\
&\left(\sin\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\sin\left(\alpha_A+\phi_B\right)+\cos\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\cos\left(\phi_A+\alpha_B\right)\right)^2\\
&\left(\cos\left(\frac{\theta_A}{2}\right)\cos\left(\frac{\theta_B}{2}\right)\sin\left(\phi_A+\phi_B\right)-\sin\left(\frac{\theta_A}{2}\right)\sin\left(\frac{\theta_B}{2}\right)\cos\left(\alpha_A+\alpha_B\right)\right)^2.
\end{aligned} \tag{8.7}$$

While the probabilities are identical for state $|00\rangle$, the formulas do not match for states $|01\rangle$, $|10\rangle$ and $|11\rangle$ due to sign changes in several places. Specifically, the differences (after simplification) are as follows:

$$\begin{aligned}
\Delta p_{01} = &- \left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right) - \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2 \\
&+ \left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2, \\
\Delta p_{10} = &- \left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2 \\
&+ \left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2, \\
\Delta p_{11} = &- \left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) + \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\
&+ \left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) - \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2.
\end{aligned} \tag{8.8}$$

which are non-zero for some inputs. In order to determine which formula is correct, let us reproduce the calculations from the original publication. The initial vector is given by formula on page 6:

$$v = [1,0,0,0]\,. \tag{8.9}$$

The matrix form of the entanglement operator J is defined in Eq. 3 on page 7:

$$J = \frac{1}{\sqrt{2}}(I + i\sigma_x \otimes \sigma_x) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 0 & i \\ 0 & 1 & i & 0 \\ 0 & i & 1 & 0 \\ i & 0 & 0 & 1 \end{bmatrix}. \tag{8.10}$$

The parametrization of players' strategies is defined in Eq. 4 on page 7:

$$U(\theta,\phi,\alpha) = \begin{bmatrix} e^{-i\phi}\cos\left(\frac{\theta}{2}\right) & e^{i\alpha}\sin\left(\frac{\theta}{2}\right) \\ -e^{-i\alpha}\sin\left(\frac{\theta}{2}\right) & e^{i\phi}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{8.11}$$

Assume that $U_\mathrm{A} = U(\theta_\mathrm{A},\phi_\mathrm{A},\alpha_\mathrm{A})$ and $U_\mathrm{B} = U(\theta_\mathrm{B},\phi_\mathrm{B},\alpha_\mathrm{B})$, or equivalently:

$$\begin{aligned}
U_\mathrm{A} &= \begin{bmatrix} e^{-i\phi_\mathrm{A}}\cos\left(\frac{\theta_\mathrm{A}}{2}\right) & e^{i\alpha_\mathrm{A}}\sin\left(\frac{\theta_\mathrm{A}}{2}\right) \\ -e^{-i\alpha_\mathrm{A}}\sin\left(\frac{\theta_\mathrm{A}}{2}\right) & e^{i\phi_\mathrm{A}}\cos\left(\frac{\theta_\mathrm{A}}{2}\right) \end{bmatrix}, \\
U_\mathrm{B} &= \begin{bmatrix} e^{-i\phi_\mathrm{B}}\cos\left(\frac{\theta_\mathrm{B}}{2}\right) & e^{i\alpha_\mathrm{B}}\sin\left(\frac{\theta_\mathrm{B}}{2}\right) \\ -e^{-i\alpha_\mathrm{B}}\sin\left(\frac{\theta_\mathrm{B}}{2}\right) & e^{i\phi_\mathrm{B}}\cos\left(\frac{\theta_\mathrm{B}}{2}\right) \end{bmatrix}.
\end{aligned} \tag{8.12}$$

Let us calculate the final statevector according to Eq. 2 from page 6:

$$\psi = J^\dagger \left(U_\mathrm{A} \otimes U_\mathrm{B}\right) J v. \tag{8.13}$$

After substituting the matrices as well as performing exponential and trigonometric simplifications, the result statevector is equal to:

$$\begin{bmatrix} -\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) \\ i\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right) \\ i\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right) \\ \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right) - \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) \end{bmatrix}. \tag{8.14}$$

Finally, after converting the statevector to observational basis probabilities by multiplying each coordinate by its Hermitian conjugate (or alternatively, squaring its module) we obtain the following vector of probabilities of game outcomes:

$$\begin{bmatrix} \left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) - \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\ \left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\ \left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\ \left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) - \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \end{bmatrix}. \tag{8.15}$$

which is identical to Eq. 8.7, confirming the correctness of the probabilities calculated by `ewl` library. The above matrix calculations as well as exponential and trigonometric simplifications were performed independently using two different symbolic engines, SymPy and Mathematica. Consequently, the correct Bob's expected payoff function is

$$\begin{aligned} \$_\mathrm{B} = {} & 3\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) - \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\ & + 5\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right) + \cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2 \\ & + \left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right) - \sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2. \end{aligned} \tag{8.16}$$

Let us check if this error affects the later part of the publication. Eq. 5 on page 8 in [6] shows the parameters of Bob's best response for arbitrary strategy of Alice:

$$\begin{aligned} \theta_\mathrm{B} &= \theta_\mathrm{A} + \pi \\ \phi_\mathrm{B} &= \alpha_\mathrm{A} \\ \alpha_\mathrm{B} &= \phi_\mathrm{A} - \tfrac{\pi}{2}. \end{aligned} \tag{8.17}$$

Applying this substitution to the original probabilities from the article we obtain $[0, 1, 0, 0]$ and therefore $\$_B = 5$ which is the maximum possible payoff. However, if we substitute this formula to the correct observational basis probabilities we obtain

$$\begin{bmatrix} 0 \\ \left(\sin^2\left(\frac{\theta_A}{2}\right)\cos\left(2\alpha_A\right) + \cos^2\left(\frac{\theta_A}{2}\right)\cos\left(2\phi_A\right)\right)^2 \\ \left(\sin^2\left(\frac{\theta_A}{2}\right)\sin\left(2\alpha_A\right) - \cos^2\left(\frac{\theta_A}{2}\right)\sin\left(2\phi_A\right)\right)^2 \\ 4\sin^2\left(\frac{\theta_A}{2}\right)\cos^2\left(\frac{\theta_A}{2}\right)\sin^2\left(\alpha_A + \phi_A\right) \end{bmatrix} \tag{8.18}$$

which is not equal to $[0, 1, 0, 0]$ for some inputs. In particular, Bob's expected payoff is

$$\begin{aligned} \$_B = 5\left(\sin^2\left(\frac{\theta_A}{2}\right)\cos\left(2\alpha_A\right) + \cos^2\left(\frac{\theta_A}{2}\right)\cos\left(2\phi_A\right)\right)^2 \\ + 4\sin^2\left(\frac{\theta_A}{2}\right)\cos^2\left(\frac{\theta_A}{2}\right)\sin^2\left(\alpha_A + \phi_A\right), \end{aligned} \tag{8.19}$$

which also depends on Alice's strategy parametrs and is not equal to 5 for some inputs. The correct formula for Bob's best response should be as follows:

$$\begin{aligned} \theta_B &= \theta_A + \pi \\ \phi_B &= -\alpha_A \\ \alpha_B &= -\phi_A - \tfrac{\pi}{2}. \end{aligned} \tag{8.20}$$

Substituting Eq. 8.20 into Eq. 8.7 we obtain $[0, 1, 0, 0]$, or alternatively $\$_B = 5$. However, this is not the only best response. Consider the following formula:

$$\begin{aligned} \theta_B &= \pi - \theta_A \\ \phi_B &= -\alpha_A \\ \alpha_B &= -\phi_A + \tfrac{\pi}{2}. \end{aligned} \tag{8.21}$$

Applying the substitution Eq. 8.21 into Eq. 8.7 we also get $[0, 1, 0, 0]$ and thus $\$_B = 5$, regardless of Alice's move. It is worth mentioning that the above best response substitutions were found manually by trial and error. In the following sections we will try to find best responses automatically using symbolic analysis of the payoff function.

In section 5.3 we also introduced a similar variant of Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization, originally described in [35]. The only difference from the previously discussed version of the game is the base strategy D defined as $U(\pi, 0, 0)$ instead of $U(\pi, 0, \frac{\pi}{2})$. The observational basis probabilities presented on

page 178 in [35] are cited from [6] and thus identical to those from Eq. 8.6:

$$\begin{aligned}
&\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right)-\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2\\
&\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right)-\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2\\
&\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right)+\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2\\
&\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right)+\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2.
\end{aligned} \tag{8.22}$$

However, using `ewl` library to calculate the probabilities we obtain:

$$\begin{aligned}
&\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right)-\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2\\
&\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right)+\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2\\
&\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right)+\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2\\
&\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right)-\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2.
\end{aligned} \tag{8.23}$$

Similarly to the previous case, the probabilities from the original publication and from `ewl` library match only for state $|00\rangle$. For the following states $|01\rangle$, $|10\rangle$ and $|11\rangle$ the differences after simplification are as follows:

$$\begin{aligned}
\Delta p_{01} = &-\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right)-\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2\\
&+\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right)+\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2,
\end{aligned}$$

$$\begin{aligned}
\Delta p_{10} = &-\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\alpha_\mathrm{A}-\phi_\mathrm{B}\right)+\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\phi_\mathrm{A}-\alpha_\mathrm{B}\right)\right)^2\\
&+\left(\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\phi_\mathrm{B}\right)+\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2,
\end{aligned}$$

$$\begin{aligned}
\Delta p_{11} = &-\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right)+\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2\\
&+\left(\cos\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(\frac{\theta_\mathrm{B}}{2}\right)\sin\left(\phi_\mathrm{A}+\phi_\mathrm{B}\right)-\sin\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(\frac{\theta_\mathrm{B}}{2}\right)\cos\left(\alpha_\mathrm{A}+\alpha_\mathrm{B}\right)\right)^2.
\end{aligned} \tag{8.24}$$

which again are non-zero for some inputs. Moreover, the probabilities in Eq. 8.23 differ from those presented in Eq. 8.7, showing that the choice of the base strategy D indeed affects the quantum game.

Analogously to the previously discussed variant of the quantum game, let us check how this error affects the later part of the article. On page 181 in [35] there is a formula for Bob's best response depending on the parameters of Alice's strategy identical to Eq. 5 in [6]. Substituting it into the original probabilities from Eq. 8.22 we obtain (after simplification):

$$\begin{bmatrix} 4\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\sin^2\left(\alpha_\mathrm{A}+\phi_\mathrm{A}\right) \\ 0 \\ \cos^2\left(\theta_\mathrm{A}\right) \\ 4\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos^2\left(\alpha_\mathrm{A}+\phi_\mathrm{A}\right) \end{bmatrix} \tag{8.25}$$

which is not equal to $[0, 1, 0, 0]$ for some inputs. Bob's expected payoff is

$$\$_\mathrm{B} = \left(8 - 4\cos\left(2\alpha_\mathrm{A} + 2\phi_\mathrm{A}\right)\right)\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos^2\left(\frac{\theta_\mathrm{A}}{2}\right). \tag{8.26}$$

Applying the best response substitution from the publication into probabilities calculated by `ewl` library presented in Eq. 8.23 we obtain:

$$\begin{bmatrix} 4\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\sin^2\left(\alpha_\mathrm{A}+\phi_\mathrm{A}\right) \\ \left(\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(2\alpha_\mathrm{A}\right) + \cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(2\phi_\mathrm{A}\right)\right)^2 \\ \left(\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(2\alpha_\mathrm{A}\right) - \cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\sin\left(2\phi_\mathrm{A}\right)\right)^2 \\ 0 \end{bmatrix} \tag{8.27}$$

which again is not equal to $[0, 1, 0, 0]$ for some inputs. Bob's expected payoff is

$$\begin{aligned} \$_\mathrm{B} = 5&\left(\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(2\alpha_\mathrm{A}\right) + \cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos\left(2\phi_\mathrm{A}\right)\right)^2 \\ &+ 12\sin^2\left(\frac{\theta_\mathrm{A}}{2}\right)\cos^2\left(\frac{\theta_\mathrm{A}}{2}\right)\sin^2\left(\alpha_\mathrm{A}+\phi_\mathrm{A}\right). \end{aligned} \tag{8.28}$$

The correct substitution for Bob's best response is:

$$\begin{aligned} \theta_\mathrm{B} &= \theta_\mathrm{A} + \pi \\ \phi_\mathrm{B} &= -\alpha_\mathrm{A} - \tfrac{\pi}{2} \\ \alpha_\mathrm{B} &= -\phi_\mathrm{A}. \end{aligned} \tag{8.29}$$

Similarly to the previous case, there also exists another best response:

$$\begin{aligned} \theta_\mathrm{B} &= \pi - \theta_\mathrm{A} \\ \phi_\mathrm{B} &= -\alpha_\mathrm{A} + \tfrac{\pi}{2} \\ \alpha_\mathrm{B} &= -\phi_\mathrm{A}. \end{aligned} \tag{8.30}$$

Substituting Eq. 8.29 or Eq. 8.30 into Eq. 8.23 we obtain $[0, 1, 0, 0]$ and therefore $\$_\mathrm{B} = 5$ which is the expected result.

In section 5.5 we presented another variant of Quantum Prisoner's Dilemma with 2-parameter strategies. The probabilities of game outcomes are presented in Eq. 7 in [12]:

$$\begin{aligned}
&\left(\cos\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)+\sin\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2\\
&\left(\cos\left(\alpha_1-\beta_2\right)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)+\sin\left(\alpha_2-\beta_1\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2\\
&\left(\sin\left(\alpha_1-\beta_2\right)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)+\cos\left(\alpha_2-\beta_1\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2\\
&\left(\sin\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)-\cos\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2.
\end{aligned} \tag{8.31}$$

However, these probabilities do not sum up to 1. Using `ewl` library to calculate the observational basis probabilities we obtain:

$$\begin{aligned}
&\left(\cos\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)+\sin\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2\\
&\left(\cos\left(\alpha_1-\beta_2\right)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)+\sin\left(\alpha_2-\beta_1\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2\\
&\left(\sin\left(\alpha_1-\beta_2\right)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)+\cos\left(\alpha_2-\beta_1\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2\\
&\left(\sin\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)-\cos\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2.
\end{aligned} \tag{8.32}$$

The inconsistency only occurs for state $|11\rangle$:

$$\begin{aligned}
\Delta p_{11} = &-\left(\sin\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)-\cos\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\right)^2\\
&+\left(\sin\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)-\cos\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2.
\end{aligned} \tag{8.33}$$

More precisely, the only difference is the use of $\cos$ function instead of $\sin$ at the very end of the expression. So we conclude that the correct payoff function is

$$\begin{aligned}
\$_{\mathrm{B}} = &\,3\left(\cos\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)+\sin\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2\\
&+5\left(\sin\left(\alpha_2-\beta_1\right)\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)+\cos\left(\alpha_1-\beta_2\right)\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2\\
&+\left(\sin\left(\alpha_1+\alpha_2\right)\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)-\cos\left(\beta_1+\beta_2\right)\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\right)^2.
\end{aligned} \tag{8.34}$$

8.7 Finding best response symbolically

In order to be able to find Nash equilibria as fixed points of the composition of best response functions as described in section 7.2, first we need to know the best response substitutions in analytic form.

The purpose of this experiment was to utilize existing symbolic calculations engine such as SymPy or Mathematica in order to obtain Bob's best response functions for two different variants of Quantum Prisoner's Dilemma (either with 2 or 3 parameters per each player) as a symbolic expression with respect to the parameters of Alice's strategy, for instance

$$(\theta_{\mathrm{B}}^*, \phi_{\mathrm{B}}^*, \alpha_{\mathrm{B}}^*) = \text{best response}_{\mathrm{B}}(\theta_{\mathrm{A}}, \phi_{\mathrm{A}}, \alpha_{\mathrm{A}}). \tag{8.35}$$

The source code of all attempts to find best response function in symbolic form using Mathematica which are described in this section can be found in Appendix A.

The first approach was to directly use built-in functions for global symbolic optimization such as `Maximize` or `ArgMax`, i.e.

$$\text{best response}_{\mathrm{B}}(\theta_{\mathrm{A}}, \phi_{\mathrm{A}}, \alpha_{\mathrm{A}}) = \underset{(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) \in X}{\operatorname{argmax}} \$_{\mathrm{B}}(U(\theta_{\mathrm{A}}, \phi_{\mathrm{A}}, \alpha_{\mathrm{A}}), U(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}})). \tag{8.36}$$

Under the hood, for global non-linear constrained optimization problems Mathematica uses efficient methods based on Karush–Kuhn–Tucker (KKT) conditions. However, it turned out that Mathematica does not support symbolic parameters inside trigonometric expressions simultaneously when performing symbolic optimization.

The next attempt to finding best response function was to compute the gradient of the payoff function as partial derivatives of all parameters and solve a system of equations with the zero vector on the right side, i.e.

$$\nabla \$_{\mathrm{B}}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = \mathbf{0} \tag{8.37}$$

in real numbers using the built-in `Solve` function. Unfortunately, this time Mathematica raised an error with message informing that the problem cannot be solved with methods available to `Solve` function.

In the specific case of quantum game with full $\mathrm{SU}(2)$ parametrization, instead of trying to maximize the payoff function, we can maximize the probability of the game outcome with the highest payoff by solving an equation

$$p_{01} = 1 \tag{8.38}$$

or, alternatively, by finding a solution of the following system of equations

$$p_{00} = p_{10} = p_{11} = 0. \tag{8.39}$$

However, using the built-in `Solve` function we get the same error again, most likely due to multiple trigonometric expressions present in the input.

As a next step, we focused on possible ways to eliminate trigonometric expressions from the input. For instance, for each variable or parameter x in the original formula, we could replace $\sin x$ with a new symbol, y. However, it is not clear how to replace $\cos x$, which can be either $\sqrt{1-x^2}$ or its negation. Another idea would be to expand expressions with $\sin$ or $\cos$ functions into Taylor series and operate on polynomials which are way easier to handle when it comes to solving equations. Alternatively, we could utilize various polynomial approximation techniques, e.g. using Chebyshev polynomials, but unforunately most of the methods are designed specifically for single-dimensional cases.

The final approach was to substitute $\sin$ and $\cos$ of each parameter θ with two new variables, x and y, i.e.

$$\begin{aligned} x &= \sin\theta \\ y &= \cos\theta \end{aligned} \tag{8.40}$$

as well as capture the relationship between x and y from Pythagorean trigonometric identity $\sin^2\theta + \cos^2\theta = 1$ as the following constraint:

$$x^2 + y^2 = 1. \tag{8.41}$$

As a consequence, we reduced the problem from function maximization to solving the following system of equations without trigonometric expressions:

$$\begin{bmatrix} -a_1a_5x_1x_6 - a_1a_6x_1x_5 - a_2a_3x_2x_3 + a_2a_4x_2x_4 \\ a_1a_5x_2x_4 + a_1a_6x_2x_3 - a_2a_3x_1x_5 + a_2a_4x_1x_6 \\ a_1a_5x_1x_5 - a_1a_6x_1x_6 + a_2a_3x_2x_4 + a_2a_4x_2x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{8.42}$$

This time Mathematica did not raise any erorrs but still could not solve this task in a reasonable time of less than 12 hours.

In summary, all attempts to find Bob's best response function in analytic form using Mathematica either timed out or failed due to lack of direct support for certain types of equations. All of the attempts mentioned above were also implemented and executed in SymPy, however with no significant improvements.

Apart from the analysis of Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization, analogous experiments were conducted on the example of Quantum Prisoner's Dilemma with Frąckiewicz-Pykacz parametrization (as described in section 5.5) which involves only 2 degrees of freedom, meaning there are only 2 variables and 2 symbolic parameters in the formulas. However, this parametrization does fully cover $\mathrm{SU}(2)$ group, thus the only valid method for finding Bob's best response is to maximize the expected payoff function. In particular, the task cannot be reduced to solving a system of equations with observational basis probabilities on the left-hand side and zeros on the right-hand side due to reasons discussed in section 7.1.

8.8 Finding best response numerically

Since all attempts to find Bob's best response function in symbolic form were unsuccessful, let us use numerical methods instead. Similarly as in the previous section, we will be searching for global maxima of Bob's payoff function from Eq. 8.16, i.e.

$$\begin{aligned}\$_{\mathrm{B}} = 3&\left(\cos\left(\frac{\theta_{\mathrm{A}}}{2}\right)\cos\left(\frac{\theta_{\mathrm{B}}}{2}\right)\cos\left(\phi_{\mathrm{A}}+\phi_{\mathrm{B}}\right)-\sin\left(\frac{\theta_{\mathrm{A}}}{2}\right)\sin\left(\frac{\theta_{\mathrm{B}}}{2}\right)\sin\left(\alpha_{\mathrm{A}}+\alpha_{\mathrm{B}}\right)\right)^2\\ +5&\left(\sin\left(\frac{\theta_{\mathrm{A}}}{2}\right)\cos\left(\frac{\theta_{\mathrm{B}}}{2}\right)\cos\left(\alpha_{\mathrm{A}}+\phi_{\mathrm{B}}\right)+\cos\left(\frac{\theta_{\mathrm{A}}}{2}\right)\sin\left(\frac{\theta_{\mathrm{B}}}{2}\right)\sin\left(\phi_{\mathrm{A}}+\alpha_{\mathrm{B}}\right)\right)^2\\ +&\left(\cos\left(\frac{\theta_{\mathrm{A}}}{2}\right)\cos\left(\frac{\theta_{\mathrm{B}}}{2}\right)\sin\left(\phi_{\mathrm{A}}+\phi_{\mathrm{B}}\right)-\sin\left(\frac{\theta_{\mathrm{A}}}{2}\right)\sin\left(\frac{\theta_{\mathrm{B}}}{2}\right)\cos\left(\alpha_{\mathrm{A}}+\alpha_{\mathrm{B}}\right)\right)^2\end{aligned} \tag{8.43}$$

where $\theta_{\mathrm{B}} \in [0, \pi]$ and $\phi_{\mathrm{B}}, \alpha_{\mathrm{B}} \in [-\pi, \pi]$ are variables representing Bob's strategy while the parameters of Alice's move are denoted with $\theta_{\mathrm{A}} \in [0, \pi]$ and $\phi_{\mathrm{A}}, \theta_{\mathrm{A}} \in [-\pi, \pi]$. For this variant of Quantum Prisoner's Dilemma we already know that there are exactly two best responses, each with expected payoff equal to 5.

In this experiment we will evaluate the performance of two different derivative-free optimization methods:

- Powell [31]
- Nelder-Mead [28]

For each algorithm we will choose the initial point in three different ways:

- zero – starting point is always $[0, 0, 0]$
- Alice – starting point is the strategy of the opponent
- random – starting point is randomly selected with uniform distribution

Since the parametrization uses trigonometric functions, we can either narrow the search to one period of the function in each dimension or search the entire space $\mathbb{R}^3$ and normalize the coordinates afterwards:

- True – search bounds are enabled and set to $\Omega = [0, \pi] \times [-\pi, \pi] \times [-\pi, \pi]$
- False – search bounds are disabled but the result is normalized to Ω afterwards

The experiment consisted of $N = 10000$ runs for each combination of settings described above. For each iteration, Alice's strategy was randomly selected with uniform distribution. Next, Bob's best response was found using according numerical optimization method using implementations provided by `scipy.optimize`[2] module from SciPy library [38]. A single run was considered successful if the resulting payoff was greater than or equal to 4.9995 ($\varepsilon = 0.0005$). Success rate was calculated as the ratio of successful runs to all runs and expressed as a percentage. The results of the experiment are summarized in Tab. 8.4.

[2] https://docs.scipy.org/doc/scipy/tutorial/optimize.html

Table 8.4: Comparision of success rates of 12 different variants of a numerical method of searching for best response in Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization

Optimization method	Start point	Bounds	Success rate
Powell	zero	False	99.94%
Powell	random	False	99.23%
Powell	Alice	False	99.35%
Nelder-Mead	zero	False	96.60%
Nelder-Mead	random	False	99.52%
Nelder-Mead	Alice	False	99.01%
Powell	zero	True	86.94%
Powell	random	True	76.34%
Powell	Alice	True	79.14%
Nelder-Mead	zero	True	45.89%
Nelder-Mead	random	True	41.47%
Nelder-Mead	Alice	True	43.55%

Clearly, the methods with bounds disabled performed much better. In case of non-bounded optimization, the choice of the starting point has little effect on success rate. The highest success rate of 99.94% was achieved by Powell method with bounds disabled and zero as initial point. The performance of this method is even better than initially expected and hopefully will generalize to other variants of Quantum Prisoner's Dilemma as well as other quantum games.

8.9 Finding Nash equilibria symbolically

So far, the only quantum game for which we know the best response substitutions in analytic form is Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization. Specifically, for each arbitrary strategy of the opponent, there are exactly two different best responses, shown in Eq. 8.20 and Eq. 8.21.

In this section we will use this information in order to prove that this variant of the game does not have any Nash equilibria in pure states. However, instead of performing the calculations manually, we will use symbolic calculations engine, SymPy, to automate the process and eliminate the risk of human error.

Let us define the best response substitution from Eq. 8.20 as the following function:

$$\mathrm{br}_1(\theta, \phi, \alpha) = \left(\theta + \pi,\ -\alpha,\ -\phi - \tfrac{\pi}{2}\right). \tag{8.44}$$

Similarly, let us rewrite the second best response substitution Eq. 8.21 as another function:

$$\mathrm{br}_2(\theta, \phi, \alpha) = \left(\pi - \theta,\ -\alpha,\ -\phi + \tfrac{\pi}{2}\right). \tag{8.45}$$

As described in section 7.2, Nash equilibria in pure states are fixed points of a function composition of best response functions. Since there are two different best responses, regardless of the player, there are exactly $2^2 = 4$ combinations and thus 4 function compositions that need to be checked.

Assume that the initial strategy of Alice, expressed as parameter values, is

$$x_0 = (\theta, \phi, \alpha). \tag{8.46}$$

Imagine that Bob plays the first best response, which is

$$x_1 = \mathrm{br}_1(x_0) = \left(\theta + \pi,\ -\alpha,\ -\phi - \tfrac{\pi}{2}\right). \tag{8.47}$$

Then, Alice also responds with $\mathrm{br}_1(x_1)$, or equivalently

$$x_{11} = \mathrm{br}_1(\mathrm{br}_1(x_0)) = \left(\theta + 2\pi, \phi + \tfrac{\pi}{2}, \alpha - \tfrac{\pi}{2}\right). \tag{8.48}$$

Since the space of parameter values is periodic, let us define a relation $\equiv\ \subset \mathbb{R}^3 \times \mathbb{R}^3$ that holds when two points match when normalized to $\Omega = [0, \pi] \times [-\pi, \pi] \times [-\pi, \pi]$ which is the original boundary for parameter values that contains a single period of underlying trigonometric function in each respective dimension:

$$\mathbf{x}_1 \equiv \mathbf{x}_2 \iff \exists\, a, b, c \in \mathbb{Z} : \mathbf{x}_2 - \mathbf{x}_1 = (a\pi, 2b\pi, 2c\pi). \tag{8.49}$$

For readability purposes, instead of checking the equivalence with the initial strategy, we will calculate the difference of the final and the initial strategy in terms of parameter values and then check whether is it equivalent to a zero vector:

$$\mathbf{x}_1 \equiv \mathbf{x}_2 \iff \mathbf{x}_2 - \mathbf{x}_1 \equiv (0, 0, 0). \tag{8.50}$$

In this case the difference is

$$\Delta x_{11} = x_{11} - x_0 = \left(2\pi, \tfrac{\pi}{2}, -\tfrac{\pi}{2}\right) \not\equiv (0,0,0)\,, \tag{8.51}$$

meaning that playing the first best response twice, once by Bob and then once by Alice, does not return to the original strategy. However, there are still 3 remaining combinations that need to be checked:

$$\begin{aligned} x_{12} &= \mathrm{br}_2(\mathrm{br}_1(x_0)) = \left(-\theta, \phi + \tfrac{\pi}{2}, \alpha + \tfrac{\pi}{2}\right) \\ x_{21} &= \mathrm{br}_1(\mathrm{br}_2(x_0)) = \left(-\theta + 2\pi, \phi - \tfrac{\pi}{2}, \alpha - \tfrac{\pi}{2}\right) \\ x_{22} &= \mathrm{br}_2(\mathrm{br}_2(x_0)) = \left(\theta, \phi - \tfrac{\pi}{2}, \alpha + \tfrac{\pi}{2}\right). \end{aligned} \tag{8.52}$$

The differences are as follows:

$$\begin{aligned} \Delta x_{12} &= x_{12} - x_0 = \left(-2\theta, \tfrac{\pi}{2}, \tfrac{\pi}{2}\right) && \not\equiv (0,0,0) \\ \Delta x_{21} &= x_{21} - x_0 = \left(-2\theta + 2\pi, -\tfrac{\pi}{2}, -\tfrac{\pi}{2}\right) && \not\equiv (0,0,0) \\ \Delta x_{22} &= x_{22} - x_0 = \left(0, -\tfrac{\pi}{2}, \tfrac{\pi}{2}\right) && \not\equiv (0,0,0) \end{aligned} \tag{8.53}$$

Since none of the combinations of best response functions results in a strategy that is equivalent to the initial move, we proved that Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization has no Nash equilibria in pure states.

The most relevant conclusion of this experiment is that knowing all the best response substitutions for a specific variant of a quantum game in analytic form, we can automatically find all Nash equilibria or prove that there are no Nash equilibria in pure states simply by solving a fixed-point equation symbolically.

8.10 Finding best response cycles symbolically

The method of finding Nash equilibria in pure states described in section 7.2 and evaluated in section 8.9 can be also generalized to best response cycles. The only difference is that instead of checking each pair of best response functions, we need to check each combination of the desired length of the cycle.

In this section we will search for all best response cycles of length 4, i.e. consisting of two moves per each player, for instance

$$U_{\mathrm{A1}} \to U_{\mathrm{B1}} \to U_{\mathrm{A2}} \to U_{\mathrm{B2}} \to U_{\mathrm{A1}} \tag{8.54}$$

where the first and the last element are identical strategies, equivalent in terms of parameter values.

Similarly to the previous section, we will perform the analysis on the example of Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization as described in section 5.3. Again, we assume that the first player's initial strategy is

$$x_0 = (\theta, \phi, \alpha). \tag{8.55}$$

Since there are exactly 2 best response substitutions and we search for best response cycles of length 4, there is exactly $2^4 = 16$ following combinations that need to be checked:

$$\begin{aligned}
x_{1111} &= \mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_1(x_0)))) = (\theta + 4\pi,\ \phi + \pi,\ \alpha - \pi) \\
x_{1112} &= \mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_1(x_0)))) = (-\theta - 2\pi,\ \phi + \pi,\ \alpha) \\
x_{1121} &= \mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_1(x_0)))) = (-\theta,\ \phi,\ \alpha - \pi) \\
x_{1122} &= \mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_1(x_0)))) = (\theta + 2\pi,\ \phi,\ \alpha) \\
x_{1211} &= \mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_1(x_0)))) = (-\theta + 2\pi,\ \phi + \pi,\ \alpha) \\
x_{1212} &= \mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_1(x_0)))) = (\theta,\ \phi + \pi,\ \alpha + \pi) \\
x_{1221} &= \mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_1(x_0)))) = (\theta + 2\pi,\ \phi,\ \alpha) \\
x_{1222} &= \mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_1(x_0)))) = (-\theta,\ \phi,\ \alpha + \pi) \\
x_{2111} &= \mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_2(x_0)))) = (-\theta + 4\pi,\ \phi,\ \alpha - \pi) \\
x_{2112} &= \mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_2(x_0)))) = (\theta - 2\pi,\ \phi,\ \alpha) \\
x_{2121} &= \mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_2(x_0)))) = (\theta,\ \phi - \pi,\ \alpha - \pi) \\
x_{2122} &= \mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_2(x_0)))) = (-\theta + 2\pi,\ \phi - \pi,\ \alpha) \\
x_{2211} &= \mathrm{br}_1(\mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_2(x_0)))) = (\theta + 2\pi,\ \phi,\ \alpha) \\
x_{2212} &= \mathrm{br}_2(\mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_2(x_0)))) = (-\theta,\ \phi,\ \alpha + \pi) \\
x_{2221} &= \mathrm{br}_1(\mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_2(x_0)))) = (-\theta + 2\pi,\ \phi - \pi,\ \alpha) \\
x_{2222} &= \mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_2(\mathrm{br}_2(x_0)))) = (\theta,\ \phi - \pi,\ \alpha + \pi)\,.
\end{aligned} \tag{8.56}$$

Like in the previous section, we calculate the differences as follows:

$$\begin{aligned}
\Delta x_{1111} &= x_{1111} - x_0 = (4\pi,\ \pi,\ -\pi) && \not\equiv (0,0,0)\\
\Delta x_{1112} &= x_{1112} - x_0 = (-2\theta - 2\pi,\ \pi,\ 0) && \not\equiv (0,0,0)\\
\Delta x_{1121} &= x_{1121} - x_0 = (-2\theta,\ 0,\ -\pi) && \not\equiv (0,0,0)\\
\Delta x_{1122} &= x_{1122} - x_0 = (2\pi,\ 0,\ 0) && \equiv (0,0,0)\ \checkmark\\
\Delta x_{1211} &= x_{1211} - x_0 = (-2\theta + 2\pi,\ \pi,\ 0) && \not\equiv (0,0,0)\\
\Delta x_{1212} &= x_{1212} - x_0 = (0,\ \pi,\ \pi) && \not\equiv (0,0,0)\\
\Delta x_{1221} &= x_{1221} - x_0 = (2\pi,\ 0,\ 0) && \equiv (0,0,0)\ \checkmark\\
\Delta x_{1222} &= x_{1222} - x_0 = (-2\theta,\ 0,\ \pi) && \not\equiv (0,0,0)\\
\Delta x_{2111} &= x_{2111} - x_0 = (-2\theta + 4\pi,\ 0,\ -\pi) && \not\equiv (0,0,0)\\
\Delta x_{2112} &= x_{2112} - x_0 = (-2\pi,\ 0,\ 0) && \equiv (0,0,0)\ \checkmark\\
\Delta x_{2121} &= x_{2121} - x_0 = (0,\ -\pi,\ -\pi) && \not\equiv (0,0,0)\\
\Delta x_{2122} &= x_{2122} - x_0 = (-2\theta + 2\pi,\ -\pi,\ 0) && \not\equiv (0,0,0)\\
\Delta x_{2211} &= x_{2211} - x_0 = (2\pi,\ 0,\ 0) && \equiv (0,0,0)\ \checkmark\\
\Delta x_{2212} &= x_{2212} - x_0 = (-2\theta,\ 0,\ \pi) && \not\equiv (0,0,0)\\
\Delta x_{2221} &= x_{2221} - x_0 = (-2\theta + 2\pi,\ -\pi,\ 0) && \not\equiv (0,0,0)\\
\Delta x_{2222} &= x_{2222} - x_0 = (0,\ -\pi,\ \pi) && \not\equiv (0,0,0)
\end{aligned} \tag{8.57}$$

Among 16 possible combinations of best response chains of length 4 (i.e. 2 moves per each player), we found exactly 4 best response cycles:

$$\begin{aligned}
&U(\theta,\phi,\alpha) \xrightarrow{\text{br}_1} U\big(\theta+\pi, -\alpha, -\phi-\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_1} U\big(\theta+2\pi, \phi+\tfrac{\pi}{2}, \alpha-\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_2} U\big(-\theta-\pi, -\alpha+\tfrac{\pi}{2}, -\phi\big) \xrightarrow{\text{br}_2} U(\theta+2\pi,\phi,\alpha)\\
&U(\theta,\phi,\alpha) \xrightarrow{\text{br}_1} U\big(\theta+\pi, -\alpha, -\phi-\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_2} U\big(-\theta, \phi+\tfrac{\pi}{2}, \alpha+\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_2} U\big(\theta+\pi, -\alpha-\tfrac{\pi}{2}, -\phi\big) \xrightarrow{\text{br}_1} U(\theta+2\pi,\phi,\alpha)\\
&U(\theta,\phi,\alpha) \xrightarrow{\text{br}_2} U\big(\pi-\theta, -\alpha, -\phi+\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_1} U\big(2\pi-\theta, \phi-\tfrac{\pi}{2}, \alpha-\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_1} U\big(3\pi-\theta, -\alpha+\tfrac{\pi}{2}, -\phi\big) \xrightarrow{\text{br}_2} U(\theta-2\pi,\phi,\alpha)\\
&U(\theta,\phi,\alpha) \xrightarrow{\text{br}_2} U\big(\pi-\theta, -\alpha, -\phi+\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_2} U\big(\theta, \phi-\tfrac{\pi}{2}, \alpha+\tfrac{\pi}{2}\big) \xrightarrow{\text{br}_1} U\big(\theta+\pi, -\alpha-\tfrac{\pi}{2}, -\phi\big) \xrightarrow{\text{br}_1} U(\theta+2\pi,\phi,\alpha).
\end{aligned} \tag{8.58}$$

The final strategies are equivalent to the initial strategy because

$$U(\theta-2\pi,\phi,\alpha) \equiv U(\theta,\phi,\alpha) \equiv U(\theta+2\pi,\phi,\alpha). \tag{8.59}$$

Moreover, since θ, ϕ, α are not fixed, all 4 best response cycles hold for arbitrary initial strategy of Alice or Bob.

8.11 Finding Nash equilibria numerically

In section 8.7 we made a number of attempts towards finding the best response function in a symbolic form using Mathematica and SymPy for two different variants of Quantum Prisoner's Dilemma. However, no such attempt was successful. As a consequence, we changed the approach from symbolic to numerical optimization. In section 8.8 we implemented and evaluated 12 different variants of a numerical method for finding best response to a given strategy and selected the one with highest success rate.

In this section we will apply the algorithm of finding best response numerically in order to find Nash equilibria in pure states on the example of Quantum Prisoner's Dilemma with the EWL original parametrization $U(\theta, \phi)$ with 2 degrees of freedom as introduced in [8] for which the best response substitutions are not known in a symbolic form.

In the first example we will search for a Nash equilibrium in pure states starting from strategy $U(0, 0)$ as described in the publication. The results[3] are presented in Tab. 8.5. Each consecutive row represents a move of Alice or Bob which is the best response found numerically for the previous move of the opponent from the row above, along with the expected payoff of the current player.

Table 8.5: A sequence of best responses found numerically in Quantum Prisoner's Dilemma with original EWL parametrization starting from strategy $U(0, 0)$

Player	θ	ϕ	**Expected payoff**
Alice	0.000000	0.000000	n/a
Bob	3.141593	3.137909	5.0
Alice	0.000011	1.570802	3.0
Bob	0.000000	1.570791	3.0
Alice	6.283185	1.570802	3.0
Bob	6.283185	1.570791	3.0
Alice	6.283185	1.570802	3.0
Bob	6.283185	1.570791	3.0
Alice	6.283185	1.570802	3.0
Bob	6.283185	1.570791	3.0

After a few moves, the sequence of best responses collapses to a cycle of length 2 where Alice plays $U_A = U(6.283185, 1.570802)$ while Bob responds with $U_B = U(6.283185, 1.570791)$. Since $U_A \approx U_B \approx U(2\pi, \frac{\pi}{2}) \equiv U(0, \frac{\pi}{2})$, we conclude that we found a Nash equilibrium

$$\big(U(0, \tfrac{\pi}{2}),\, U(0, \tfrac{\pi}{2})\big) \tag{8.60}$$

with payoffs $\$_A = \$_B = 3$, which is the expected result as mentioned in the original publication [8] on the page 3.

[3] A single-precision floating number has about 7 decimal digits of precision ($\log_{10} 2^{23} \approx 6.92$). While the precision of the method has not been tested yet, the results shown in Tab. 8.7 sum to $3.141593 \approx \pi$ and all 6 decimal digits are correct.

As the second example, instead of starting from $U(0,0)$, we will randomize the initial strategy of Alice and run the experiment multiple number of times. The results of a single run are shown in Tab. 8.6.

Table 8.6: A sequence of best responses found numerically in Quantum Prisoner's Dilemma with original EWL parametrization starting from random strategy

Player	θ	ϕ	**Expected payoff**
Alice	2.030715	2.818488	n/a
Bob	1.044621	1.510837	4.933883
Alice	6.208956	1.609381	3.499695
Bob	6.278414	1.532301	3.002761
Alice	0.000299	1.609292	3.000011
Bob	0.000019	1.532301	3.0
Alice	6.283184	1.609292	3.0
Bob	6.283185	1.532301	3.0
Alice	6.283185	1.609292	3.0
Bob	6.283185	1.532301	3.0

For each run of the experiment, the final values of θ_A and θ_B were always close to $6.283185 \approx 2\pi$. However, ϕ_A and ϕ_B varied significantly, as shown in Tab. 8.7.

Table 8.7: Final values of parameters ϕ_A and ϕ_B of best response cycles of length 2 in Quantum Prisoner's Dilemma with original EWL parametrization starting with random initial strategy for 10 sample runs of the experiment

$\theta_\mathbf{A}$	$\phi_\mathbf{A}$	$\theta_\mathbf{B}$	$\phi_\mathbf{B}$	$\phi_\mathbf{A} + \phi_\mathbf{B}$
6.283185	1.723622	6.283185	1.417971	3.141593
6.283185	1.576331	6.283185	1.565262	3.141593
6.283185	1.603082	6.283185	1.538511	3.141593
6.283185	1.511496	6.283185	1.630097	3.141593
6.283185	1.570684	6.283185	1.570909	3.141593
6.283185	1.531750	6.283185	1.609842	3.141593
6.283185	1.813776	6.283185	1.327817	3.141593
6.283185	1.702410	6.283185	1.439183	3.141593
6.283185	1.570834	6.283185	1.570758	3.141593
6.283185	1.549452	6.283185	1.592141	3.141593

Apparently, for each execution of the experiment, $\phi_\mathrm{A} + \phi_\mathrm{B} \approx 3.141593 \approx \pi$, so we conclude that we found a whole family of best response cycles of length 2 in form of

$$\{\, (U(0,\, x),\, U(0,\, \pi - x)) : x \in [0, \pi] \,\}\,. \tag{8.61}$$

8.12 Finding best response cycles numerically

In this section we will evaluate the numerical method from section 8.11 on the example of Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization from [6] which was described in section 5.3. In this variant of the game there are no Nash equilibria, but for arbitrary initial startegy there exist exactly 4 different best response cycles of length 4 (i.e. 2 moves per each player) as shown symbolically using SymPy library in section 8.10 of the thesis.

Table 8.8: A sequence of best responses found numerically in Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization starting from random strategy

Player	θ	ϕ	α	**Expected payoff**
Alice	2.271152	-0.118018	-0.081848	n/a
Bob	-0.870404	0.081856	-1.452776	5.000000
Alice	-2.270383	1.453049	1.487888	4.999995
Bob	0.871299	1.653691	0.117743	5.000000
Alice	2.270338	-0.117733	-0.082893	5.000000
Bob	-0.871261	0.082902	-1.452838	5.000000
Alice	-2.269523	1.453107	1.486841	4.999995
Bob	0.872155	1.654738	0.117685	5.000000
Alice	2.269483	-0.117675	-0.083941	5.000000
Bob	-0.872115	0.083949	-1.452903	5.000000
Alice	-2.268748	1.453130	1.485877	4.999996
Bob	0.872926	1.655703	0.117663	5.000000
Alice	2.268713	-0.117652	-0.084905	5.000000
Bob	-0.872885	0.084913	-1.452932	5.000000
Alice	-2.268020	1.453131	1.484945	4.999996
Bob	0.873651	1.656635	0.117661	5.000000
Alice	2.267989	-0.117650	-0.085838	5.000000
Bob	-0.873609	0.085846	-1.452939	5.000000
Alice	-2.267319	1.453127	1.484048	4.999996
Bob	5.408823	-1.484063	0.117665	5.000000
Alice	2.267279	-0.117653	-0.086732	5.000000
Bob	-0.874319	0.086739	-1.452943	5.000000

After only a few iterations, the numerical method was able to find a best response cycle $U_{A1} \to U_{B1} \to U_{A2} \to U_{B2} \to U_{A1}$ composed of the following strategies:

$$\begin{aligned} U_{A1} &= U(-2.267319, 1.453127, 1.484048) \\ U_{B1} &= U(5.408823, -1.484063, 0.117665) \\ U_{A2} &= U(2.267279, -0.117653, -0.086732) \\ U_{B2} &= U(-0.874319, 0.086739, -1.452943). \end{aligned} \tag{8.62}$$

As the second example we will evaluate this method on the example of Quantum Prisoner's Dilemma with Frąckiewicz-Pykacz parametrization with 2 degrees of freedom where the best response function in analytic form is unknown. The results of a single run of the experiment are presented in Tab. 8.9.

Table 8.9: A sequence of best responses found numerically in Quantum Prisoner's Dilemma with Frąckiewicz-Pykacz parametrization starting from random strategy

Player	θ	ϕ	**Expected payoff**
Alice	4.731055	4.626624	n/a
Bob	1.755219	0.797327	4.003305
Alice	0.020601	5.500531	4.183335
Bob	3.141649	5.500531	4.999788
Alice	6.283185	0.788142	5.0
Bob	3.141595	0.788142	5.0
Alice	6.283185	5.500531	5.0
Bob	3.141595	5.500531	5.0
Alice	6.283185	0.788142	5.0
Bob	3.141595	0.788142	5.0
Alice	6.283185	5.500531	5.0
Bob	3.141595	5.500531	5.0
Alice	6.283185	0.788142	5.0
Bob	3.141595	0.788142	5.0
Alice	6.283185	5.500531	5.0
Bob	3.141595	5.500531	5.0

Similarly to the previous example, there is a pattern of 4 consecutive strategies that starts to repeat. The numerical method was able to detect a best response cycle $U_{\text{A1}} \to U_{\text{B1}} \to U_{\text{A2}} \to U_{\text{B2}} \to U_{\text{A1}}$ composed of the following strategies:

$$\begin{aligned} U_{\text{A1}} &= U(6.283185, 0.788142) \\ U_{\text{B1}} &= U(3.141595, 0.788142) \\ U_{\text{A2}} &= U(6.283185, 5.500531) \\ U_{\text{B2}} &= U(3.141595, 5.500531). \end{aligned} \tag{8.63}$$

with the maximum possible payoff $\$_{\text{A}} = \$_{\text{B}} = 5$ after each move.

To sum up, the main advantage of using numerical methods for finding Nash equilibria as well as best response cycles is the fact that it does not require knowing the best response functions in analytic form. However, this kind of analysis is strictly dependent on the choice of the initial strategy and it is necessary to run the experiment a number of times and manually generalize the results afterwards in order to draw reasonable conclusions. Moreover, the current solution only finds a single global maximum, while in general there may exist more than one best response.

8.13 Summary

In this chapter we presented the results of a number of experiments related to software-aided analysis of various properties of quantum games on the example of multiple variants of Quantum Prisoner's Dilemma with different parametrizations.

In section 8.1 we presented the environment in terms of hardware and software used for experimental part of the thesis.

In section 8.2 we executed the two-player variant of Quantum Prisoner's Dilemma on a quantum simulator and a real quantum device, compared the results with theoretical expectations as well as explained how quantum circuit is prepared for such execution.

In section 8.3 we demonstrated the abilities of `ewl` library as well as the underlying symbolic calculations engine, SymPy, by re-creating a three-dimensional plot of payoff function for a quantum game with a custom parametrization described in [15].

In section 8.4 we used `ewl` library to obtain formulas for a generalized variant of Quantum Prisoner's Dilemma with three players which we executed on a quantum simulator and a real quantum device in section 8.5.

In section 8.6 we compared the results obtained using the `ewl` library with formulas from other publications, found several calculation errors in three independent publications and provided the correct formulas which are necessary for further processing.

In section 8.7 we described numerous attempts to find best response substitutions in a symbolic form using Mathematica. Due to lack of support for certain types of equations, the problem cannot be solved directly. Therefore, we also presented a few approaches towards reducing the complexity of the input, however the task still could not be solved in a reasonable time. In section 8.8 we evaluated 12 different variants of a numerical method of find best responses numerically.

In section 8.9, based on the known best response substitutions, using SymPy we proved that a variant of Quantum Prisoner's Dilemma does not have any Nash equilibria in pure states. Then, in section 8.10 we symbolically showed that there exist exactly 4 best response cycles of length 4 in that variant of quantum game.

In section 8.11 and section 8.12 we performed analogous experiments using numerical methods for variants of Quantum Prisoner's Dilemma for which the best response functions are unknown.

General conclusions drawn from the experiments will be presented in section 9.2.

9 Summary

In this chapter, we will summarize the achieved goals in terms of research objectives, formulate general conclusions, and suggest possible directions for further research.

9.1 Achieved goals

Realizing the complexity of the calculations related to the analysis of quantum games, we began our work by designing and implementing a utility tool for performing symbolic calculations of various properties of quantum games in the EWL protocol. As a result, we created a Python library named `ewl` that combines quantum games with symbolic calculations and interfaces with IBM Q systems, offering the following functionalities:

- symbolic calculation of entanglement operator J, amplitudes of the state vector, distribution of possible game outcomes and payoff functions in analytic form for arbitrary initial state of the game and arbitrary number of players
- implementation of several popular quantum strategy parametrizations as well as possibility to define custom ones
- integration with IBM Q quantum simulators as well as real quantum devices using Qiskit framework

Symbolic expressions calculated by the library were critical for further experiments. Moreover, thanks to the library, it was possible to find minor inconsistencies in quite relevant formulas in 3 independent publications as described in section 8.6.

Being able to execute arbitrary quantum games in the EWL protocol on IBM Q simulators and devices, we successfully performed the following experiments as an extension of Filip Galas' master thesis [15], in particular we:

- executed two-player variant of Quantum Prisoner's on statevector simulator, QASM simulator as well as a real quantum device `ibmq_quito` and compared the results with theoretical expectations
- repeated the experiment for a generalized case of the game with three players and explained the influence of the underlying logical architecture of the quantum device as well as the transpiling process

In the theoretical part of the thesis we focused on the derivation and description of algorithms for searching for best responses and Nash equilibria. More specifically, we:

- formulated the best response search task as an optimization problem
- proposed a numerical algorithm for finding best response based on existing optimization methods such as Powell or Nelder-Mead
- evaluated the performance of 12 different variants of this numerical algorithm
- presented an alternative way of finding the best response in the special case of quantum games with full SU(2) parametrizations by solving a system of equations and explained why it cannot be applied for other classes of parametrizations
- described a symbolic approach towards finding Nash equilibria in pure states
- reduced the task of finding best response strategy from trigonometric function maximization to solving a simple system of equations

Finally, we tested the algorithms on real-world examples of quantum games and:

- symbolically proved that Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ parametrization has no Nash equilibria in pure states
- symbolically found 4 best response cycles of length 4 in the aforementioned variant of quantum game
- numerically found Nash equilibrium in Quantum Prisoner's Dilemma with original $U(\theta, \phi)$ parametrization when starting from $U(\pi, 0)$ strategy
- numerically found a whole family of best response cycles of length 2 in the aforementioned game starting from a random strategy
- numerically found a best response cycle of length 4 in Quantum Prisoner's Dilemma with $U(\theta, \phi, \alpha)$ as well as Frąckiewicz-Pykacz parametrization

9.2 Conclusions

The research conducted for this thesis leads to the following conclusions:

- Existing software for scientific computing may be successfully utilized for the purpose of theoretical analysis of various properties of quantum games in the EWL protocol. In particular, `ewl` library is a useful tool for deriving complex formulas describing generalized variants of such quantum games, for instance with more players.
- When implementing quantum games for IBM Q, `Operators` library can be used to construct arbitrary quantum gates instead of manual decomposition of entanglement operator.
- For three-player variant of Quantum Prisoner's Dilemma, the results of the experiment are far from ideal. Most likely, high decoherence is caused by the length of the transpiled version of the quantum circuit in terms of number of quantum layers due to the underlying logical architecture of quantum device.
- Knowing the best response function in analytic form we can find or prove the lack existence of Nash equilibria in pure strategies by solving a fixed-point equation symbolically.
- Due to lack of direct support for certain types of parametrized functions and equations, Mathematica was not able to find best response function symbolically. Despite numerous attempts of reducing the complexity of the input, the task still could not be solved in a reasonable time, similarly using SymPy.
- Among 12 tested variants of the numerical algorithm of finding best response, Powell method with bounds disabled as zero as starting point achieved the highest hit rate of 99.94%.
- When it comes to symbolic calculations, Mathematica is a clear winner in terms of performance, because its kernel is implemented in C/C++, whereas SymPy is written entirely in Python, which is an interpreted language. Moreover, SymPy expressions are immutable and thus have a larger memory footprint. On the other hand, SymPy is open-source, free of charge, and offers an elegant Python interface and therefore can be easily and directly integrated with Qiskit, as opposed to Mathematica which uses custom WolframScript language and has closed-source codebase.
- Finding best responses and Nash equilibria using numerical methods is far more efficient than using symbolic algorithms, however involves numerical errors and requires many iterations. On the other hand, the major advantage of symbolic approach is the exactness of the solution and the possibility to draw conclusions from its analytic form.

9.3 Future works

Among many possible ideas for future research, the most desirable direction would be definitely related to finding best response function symbolically. Despite numerous attempts involving simplification of the input as well as reduction to other kinds of problems, we were not able to obtain a generic formula for the best reply to arbitrary strategy of the opponent using Mathematica or SymPy. If we knew the best response function in analytic form, we could find whole families of Nash equilibria in symbolic form for arbitrary quantum game in the EWL scheme, which is ineffective and challenging with the numerical approach.

Another possible improvement is related to the numerical algorithms of finding best response, which currently are limited to finding only one global maximum, while in general there may exist more such points. Instead, methods such as SHGO [9] may be utilized to find multiple global maxima to avoid the possibility of missing some best responses or equilibria strategy profiles.

Finally, the `ewl` library, which was developed as part of the work, greatly facilitates the analysis more general variants of quantum games in the EWL protocol, for instance involving symbolic parameters or simply with more players, and thus provides new opportunities for quantum game theory researchers. A particularly interesting topic seems to be the study of influence of the underlying quantum computer architecture, especially connections between qubits, on the noise levels in quantum games with three or more players.

A Finding best response function symbolically using Mathematica

Mathematica provides a number of functions specifically for symbolic global optimization, including `Maximize`, `ArgMax` and `MaxValue`. A simple example of using Mathematica to find global maximum of a real-valued scalar function in a symbolic form is shown in Fig. A.1.

```
In[1]:= Maximize[a * x^2 + b * x + c, {x}, Reals]
Out[1]= { { c                (b == 0 && a == 0) || (b == 0 && a < 0)
          { -b^2+4 a c / 4 a  (b > 0 && a < 0) || (b < 0 && a < 0)   ,
          { ∞                True

          { x → { -b / 2 a     (b > 0 && a < 0) || (b < 0 && a < 0)
                { 0            (b == 0 && a == 0) || (b == 0 && a < 0) } }
                { Indeterminate True
```

Figure A.1: Symbolic global maximization of function $f(x) = ax^2 + bx + c$ of a single variable $x \in \mathbb{R}$ with three real-valued parameters a, b, c in Mathematica

The first approach towards finding the best response function in symbolic form using Mathematica was simply to maximize Bob's expected payoff function from Eq. 8.16 for variables $\theta_B, \phi_B, \alpha_B$ with respect to parameters $\theta_A, \phi_A, \alpha_A$ using built-in `Maximize` function. The expected result was to obtain the best response substitutions presented in Eq. 8.20 or Eq. 8.21. However, as presented in Fig. A.2, the attempt was unsuccessful.

```
In[1]:= Maximize[
          3 * (Cos[θA / 2] * Cos[θB / 2] * Cos[ϕA + ϕB] - Sin[θA / 2] * Sin[θB / 2] * Sin[αA + αB]) ^ 2
           + 5 * (Sin[θA / 2] * Cos[θB / 2] * Cos[αA + ϕB] + Cos[θA / 2] * Sin[θB / 2] * Sin[ϕA + αB]) ^ 2
           + (Cos[θA / 2] * Cos[θB / 2] * Sin[ϕA + ϕB] - Sin[θA / 2] * Sin[θB / 2] * Cos[αA + αB]) ^ 2,
          {θB, αB, ϕB}, Reals]
```

$$\text{Out[1]= } \text{Maximize}\left[3\left(\cos\left[\frac{\theta_A}{2}\right]\cos\left[\frac{\theta_B}{2}\right]\cos[\phi_A+\phi_B]-\sin[\alpha_A+\alpha_B]\sin\left[\frac{\theta_A}{2}\right]\sin\left[\frac{\theta_B}{2}\right]\right)^2+ 5\left(\cos\left[\frac{\theta_B}{2}\right]\cos[\alpha_A+\phi_B]\sin\left[\frac{\theta_A}{2}\right]+\cos\left[\frac{\theta_A}{2}\right]\sin\left[\frac{\theta_B}{2}\right]\sin[\alpha_B+\phi_A]\right)^2+ \left(-\cos[\alpha_A+\alpha_B]\sin\left[\frac{\theta_A}{2}\right]\sin\left[\frac{\theta_B}{2}\right]+\cos\left[\frac{\theta_A}{2}\right]\cos\left[\frac{\theta_B}{2}\right]\sin[\phi_A+\phi_B]\right)^2, \{\theta_B, \alpha_B, \phi_B\}, \mathbb{R}\right]$$

Figure A.2: An attempt to symbolically find a global maximum of Bob's expected payoff function using `Maximize` function in Mathematica

In order to simplify the input as well as for readability purposes, let us introduce new symbols, denoting the variables with lower-case letters and the parameters with capital letters in the following way:

$$\begin{aligned} A &:= \frac{\theta_{\mathrm{A}}}{2} & x &:= \frac{\theta_{\mathrm{B}}}{2} \\ B &:= \phi_{\mathrm{A}} & y &:= \phi_{\mathrm{B}} \\ C &:= \alpha_{\mathrm{A}} & z &:= \alpha_{\mathrm{B}}. \end{aligned} \tag{A.1}$$

As shown in Fig. A.3, this subtle simplification of the input to eliminate division by 2 inside trigonometric functions does not introduce any improvements.

```
In[1]:= Maximize[3 * (Cos[A] * Cos[x] * Cos[B + y] - Sin[A] * Sin[x] * Sin[C + z]) ^ 2
          + 5 * (Sin[A] * Cos[x] * Cos[C + y] + Cos[A] * Sin[x] * Sin[B + z]) ^ 2
          + (Cos[A] * Cos[x] * Sin[B + y] - Sin[A] * Sin[x] * Cos[C + z]) ^ 2,
        {x, y, z}, Reals]
Out[1]= Maximize[ (-Cos[C + z] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + y])^2 +
          5 (Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z])^2 +
          3 (Cos[A] Cos[x] Cos[B + y] - Sin[A] Sin[x] Sin[C + z])^2, {x, y, z}, ℝ]
```

Figure A.3: Another attempt to symbolically find a global maximum of Bob's expected payoff function using `Maximize` function in Mathematica

As it turned out, none of the previously mentioned functions supports symbolic parameters along with trigonometric expressions, although they are supported separately, as shown in Fig. A.4. When Mathematica kernel cannot find the answer, it simply returns the input.

```
In[1]:= Maximize[-x^2 + A, x]
Out[1]= {A, {x → 0}}

In[2]:= Maximize[Cos[x / 2], x]
Out[2]= {1, {x → 0}}

In[3]:= Maximize[Cos[x + A], x]
Out[3]= Maximize[Cos[A + x], x]
```

Figure A.4: An attempt to symbolically find global maximum of a simple parametrized trigonometric expression using `Maximize` in Mathematica

Since Mathematica cannot solve the problem directly, we need to research other approaches. In the documentation we can find `Solve`[1] function for solving systems of equations or inequalities which also supports symbolic parameters, as shown on the example of quadratic equation in Fig. A.5.

[1] https://reference.wolfram.com/language/ref/Solve.html

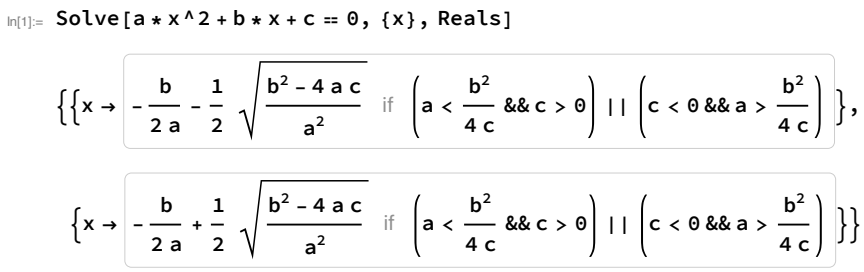

Figure A.5: Symbolic solution of equation $ax^2 + bx + c = 0$ with single variable $x \in \mathbb{R}$ and three real-valued parameters a, b, c using `Solve` function in Mathematica

Because this variant of Quantum Prisoner's Dilemma uses full $\mathrm{SU}(2)$ parametrization with 3 degrees of freedom that fully covers the set of unitary strategies, there always exists a response with expected payoff equal to 5 which is the maximum value from the payoff matrix. Therefore, for this specific variant of the quantum game, instead of maximizing the expected payoff function, we can solve the following equation:

$$\$_{\mathrm{B}}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 5. \tag{A.2}$$

Unfortunately, as presented in Fig. A.6, this equation cannot be solved in real numbers with the methods available to `Solve` function from Mathematica.

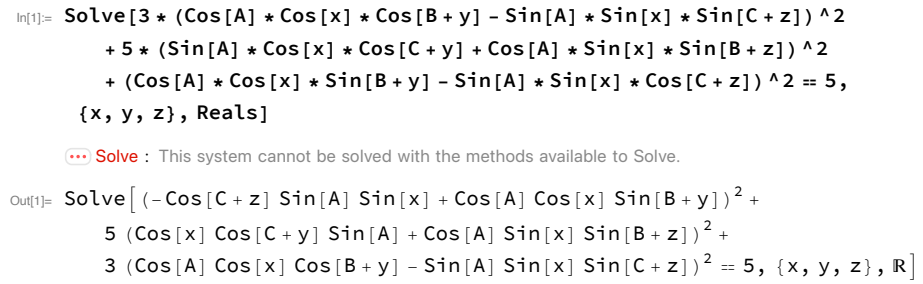

Figure A.6: An attempt to symbolically solve equation from Eq. A.2 using `Solve` function in Mathematica

Another approach would be to find all local optima of Bob's expected payoff function by solving the following system of equations involving partial derivatives:

$$\begin{cases} \dfrac{\partial \$_{\mathrm{B}}}{\partial \theta_{\mathrm{B}}}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 0 \\ \dfrac{\partial \$_{\mathrm{B}}}{\partial \phi_{\mathrm{B}}}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 0 \\ \dfrac{\partial \$_{\mathrm{B}}}{\partial \alpha_{\mathrm{B}}}(\theta_{\mathrm{B}}, \phi_{\mathrm{B}}, \alpha_{\mathrm{B}}) = 0 \end{cases} \tag{A.3}$$

or, alternatively, $\nabla \$_{\mathrm{B}} = \mathbf{0}$, and then find the global maximum among the results.

As shown in Fig. A.7, Mathematica is able to symbolically calculate partial derivatives, however it still fails to find a solution the system of equations which would lead to Bob's best response substitution. Although the right-hand sides are now equal to zero, it is still non-trivial to solve the system of equations as each left-hand side is a sum of multiple components composed of products of trigonometric expressions.

```
In[1]:= f = 3 * (Cos[A] * Cos[x] * Cos[B + y] - Sin[A] * Sin[x] * Sin[C + z]) ^ 2 +
          5 * (Sin[A] * Cos[x] * Cos[C + y] + Cos[A] * Sin[x] * Sin[B + z]) ^ 2 +
          (Cos[A] * Cos[x] * Sin[B + y] - Sin[A] * Sin[x] * Cos[C + z]) ^ 2
Out[1]= (-Cos[C + z] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + y])^2 +
        5 (Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z])^2 +
        3 (Cos[A] Cos[x] Cos[B + y] - Sin[A] Sin[x] Sin[C + z])^2

In[2]:= D[f, x]
Out[2]= 2 (-Cos[C + z] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + y])
          (-Cos[x] Cos[C + z] Sin[A] - Cos[A] Sin[x] Sin[B + y]) +
        10 (-Cos[C + y] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + z])
          (Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z]) +
        6 (-Cos[A] Cos[B + y] Sin[x] - Cos[x] Sin[A] Sin[C + z])
          (Cos[A] Cos[x] Cos[B + y] - Sin[A] Sin[x] Sin[C + z])

In[3]:= Solve[D[f, x] == 0 && D[f, y] == 0 && D[f, z] == 0, {x, y, z}, Reals]
        Solve : This system cannot be solved with the methods available to Solve.
Out[3]= Solve[2 (-Cos[C + z] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + y])
                (-Cos[x] Cos[C + z] Sin[A] - Cos[A] Sin[x] Sin[B + y]) +
              10 (-Cos[C + y] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + z])
                (Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z]) +
              6 (-Cos[A] Cos[B + y] Sin[x] - Cos[x] Sin[A] Sin[C + z])
                (Cos[A] Cos[x] Cos[B + y] - Sin[A] Sin[x] Sin[C + z]) == 0 &&
            2 Cos[A] Cos[x] Cos[B + y] (-Cos[C + z] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + y]) -
              10 Cos[x] Sin[A] Sin[C + y] (Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z]) -
              6 Cos[A] Cos[x] Sin[B + y]
                (Cos[A] Cos[x] Cos[B + y] - Sin[A] Sin[x] Sin[C + z]) == 0 &&
            10 Cos[A] Cos[B + z] Sin[x] (Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z]) +
              2 Sin[A] Sin[x] (-Cos[C + z] Sin[A] Sin[x] + Cos[A] Cos[x] Sin[B + y]) Sin[C + z] -
              6 Cos[C + z] Sin[A] Sin[x]
                (Cos[A] Cos[x] Cos[B + y] - Sin[A] Sin[x] Sin[C + z]) == 0, {x, y, z}, ℝ]
```

Figure A.7: An attempt to symbolically solve system of equations from Eq. A.3 using `Solve` function in Mathematica

Instead of trying to maximize Bob's expected payoff function, let us focus on the observational basis probabilities. Substitutions Eq. 8.20 and Eq. 8.21 for already known Bob's best responses were fitted manually so that $p_{01} = 1$ and $p_{00} = p_{10} = p_{11} = 0$ to maximize the probability of the state with the highest payoff, thus resulting in the best possible response. Note that this method is only applicable to quantum games with full SU(2) parametrization that fully covers the set of all unitary strategies.

Two attempts to solve equation $p_{01} = 1$ and system of equations $p_{00} = p_{10} = p_{11} = 0$ are shown in Fig. A.8 and Fig. A.9, respectively. Similarly to all previous attempts, those tasks cannot be solved with the methods available to `Solve` function.

```
In[1]:= Solve[(Sin[A] * Cos[x] * Cos[C + y] + Cos[A] * Sin[x] * Sin[B + z]) ^ 2 == 1,
  {x, y, z}, Reals]

... Solve : This system cannot be solved with the methods available to Solve.

Out[1]= Solve[(Cos[x] Cos[C + y] Sin[A] + Cos[A] Sin[x] Sin[B + z])^2 == 1, {x, y, z}, ℝ]
```

Figure A.8: An attempt to symbolically solve equation $p_{01} = 1$ using `Solve` function in Mathematica

```
In[1]:= Solve[
  (Sin[A] * Sin[x] * Sin[C + z] - Cos[A] * Cos[x] * Cos[B + y]) ^ 2 == 0
   && (Sin[A] * Sin[C + y] * Cos[x] + Sin[x] * Cos[A] * Cos[B + z]) ^ 2 == 0
   && (Sin[A] * Sin[x] * Cos[C + z] - Sin[B + y] * Cos[A] * Cos[x]) ^ 2 == 0,
  {x, y, z}, Reals]

... Solve : This system cannot be solved with the methods available to Solve.

Out[1]= Solve[(-Cos[A] Cos[x] Cos[B + y] + Sin[A] Sin[x] Sin[C + z])^2 == 0 &&
   (Cos[A] Cos[B + z] Sin[x] + Cos[x] Sin[A] Sin[C + y])^2 == 0 &&
   (Cos[C + z] Sin[A] Sin[x] - Cos[A] Cos[x] Sin[B + y])^2 == 0, {x, y, z}, ℝ]
```

Figure A.9: An attempt to symbolically solve system of equations $p_{00} = p_{10} = p_{11} = 0$ using `Solve` function in Mathematica

It is worth mentioning that in the latter case, instead of using observational basis probabilities which introduce squares of trigonometric expressions, we can use statevector amplitudes directly, as shown in Fig. A.10.

```
In[1]:= Solve[
  Sin[A] * Sin[x] * Sin[C + z] - Cos[A] * Cos[x] * Cos[B + y] == 0
   && Sin[A] * Sin[C + y] * Cos[x] + Sin[x] * Cos[A] * Cos[B + z] == 0
   && Sin[A] * Sin[x] * Cos[C + z] - Sin[B + y] * Cos[A] * Cos[x] == 0, {x, y, z}, Reals]

... Solve : This system cannot be solved with the methods available to Solve.

Out[1]= Solve[-Cos[A] Cos[x] Cos[B + y] + Sin[A] Sin[x] Sin[C + z] == 0 &&
   Cos[A] Cos[B + z] Sin[x] + Cos[x] Sin[A] Sin[C + y] == 0 &&
   Cos[C + z] Sin[A] Sin[x] - Cos[A] Cos[x] Sin[B + y] == 0, {x, y, z}, ℝ]
```

Figure A.10: An attempt to symbolically solve system of equations $\psi_{00} = \psi_{10} = \psi_{11} = 0$ using `Solve` function in Mathematica

Apparently, `Solve` function does not support such complex equations or systems of equations consisting of multiple trigonometric expressions. Therefore, we need to somehow simplify the input in order to avoid trigonometric functions if possible.

Among many ideas how to convert the input, the most promising approach so far is to introduce two new symbols, x and y, for each variable or parameter θ, and substitute

$$\begin{aligned} x &= \sin\theta \\ y &= \cos\theta. \end{aligned} \tag{A.4}$$

In order to capture the relationship between x and y, we can utilize Pythagorean trigonometric identity $\sin^2\theta + \cos^2\theta = 1$ as the following constraint:

$$x^2 + y^2 = 1. \tag{A.5}$$

Let us apply the following substitutions, using x_i for new variables and a_i for newly introduced parameters:

$$\begin{array}{llll} a_1 := \sin\frac{\theta_\mathrm{A}}{2} & a_2 := \cos\frac{\theta_\mathrm{B}}{2} & x_1 := \sin\frac{\theta_\mathrm{A}}{2} & x_2 := \cos\frac{\theta_\mathrm{B}}{2} \\ a_3 := \sin\phi_\mathrm{A} & a_4 := \cos\phi_\mathrm{B} & x_3 := \sin\phi_\mathrm{A} & x_4 := \cos\phi_\mathrm{B} \\ a_5 := \sin\alpha_\mathrm{A} & a_6 := \cos\alpha_\mathrm{B} & x_5 := \sin\alpha_\mathrm{A} & x_6 := \cos\alpha_\mathrm{B} \end{array} \tag{A.6}$$

along with the following contraints:

$$\begin{array}{ll} {a_1}^2 + {a_2}^2 = 1 & {x_1}^2 + {x_2}^2 = 1 \\ {a_3}^2 + {a_4}^2 = 1 & {x_3}^2 + {x_4}^2 = 1 \\ {a_5}^2 + {a_6}^2 = 1 & {x_5}^2 + {x_6}^2 = 1. \end{array} \tag{A.7}$$

After applying this reduction to system of equations $p_{00} = p_{10} = p_{11} = 0$ we obtain:

$$\begin{bmatrix} -a_1a_5x_1x_6 - a_1a_6x_1x_5 - a_2a_3x_2x_3 + a_2a_4x_2x_4 \\ a_1a_5x_2x_4 + a_1a_6x_2x_3 - a_2a_3x_1x_5 + a_2a_4x_1x_6 \\ a_1a_5x_1x_5 - a_1a_6x_1x_6 + a_2a_3x_2x_4 + a_2a_4x_2x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{A.8}$$

Surprisingly, this time Mathematica does not raise an error but unfortunately still fails to solve this system of equations in a reasonable time of less than 12 hours, as shown in Fig. A.11.

```
In[1]:= Solve[
  -a1 * a5 * x1 * x6 - a1 * a6 * x1 * x5 - a2 * a3 * x2 * x3 + a2 * a4 * x2 * x4 == 0 &&
   a1 * a5 * x2 * x4 + a1 * a6 * x2 * x3 - a2 * a3 * x1 * x5 + a2 * a4 * x1 * x6 == 0 &&
   a1 * a5 * x1 * x5 - a1 * a6 * x1 * x6 + a2 * a3 * x2 * x4 + a2 * a4 * x2 * x3 == 0 &&
   a1 ^ 2 + a2 ^ 2 == 1 && a3 ^ 2 + a4 ^ 2 == 1 && a5 ^ 2 + a6 ^ 2 == 1 &&
   x1 ^ 2 + x2 ^ 2 == 1 && x3 ^ 2 + x4 ^ 2 == 1 && x5 ^ 2 + x6 ^ 2 == 1,
  {x1, x2, x3, x4, x5, x6}, Reals]
```

Figure A.11: An unfinished attempt to symbolically solve system of equations $\psi_{00} = \psi_{10} = \psi_{11} = 0$ using `Solve` function in Mathematica

B Abstract for PPAM 2022

Based on the results of the performed experiments, an abstract has been prepared and submitted for 14th International Conference on Parallel Processing and Applied Mathematics[1] in cooperation with dr inż. Katarzyna Rycerz and inż. Piotr Kotara.

[1] https://ppam.edu.pl/

Software-aided analysis of EWL-based quantum games

Piotr Kotara[1], Tomasz Zawadzki[1], and Katarzyna Rycerz[1,2]

[1] AGH, Institute of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland
[2] Academic Computer Centre Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, Poland
{kotara,tzawadzki}@student.agh.edu.pl, kzajac@agh.edu.pl

1 Introduction

In this paper, we present the library supporting analysis of Eisert–Wilkens–Lewenstein (EWL) scheme [1] proposed as a quantum extension for 2×2 games on the example of Prisoners Dilemma. The proposed solution is based on modern approach combining symbolic and numerical calculations with the actual access to quantum simulators and real devices provided by IBM-Q. In particular, the library provides high-level functions for searching Nash equilibria in pure strategies as well as finding the best response cycles which lead to the existence of Nash equilibria in mixed states.

2 EWL library

EWL abstraction. EWL scheme is presented in the Fig. 1. Each player (represented by one qubit) applies his strategy as a unitary matrix U_A or U_B. The J gate is used to introduce quantum correlations between qubits. The payoff function is calculated as an expectation value of the measurement output after applying $J^\dagger$ gate.

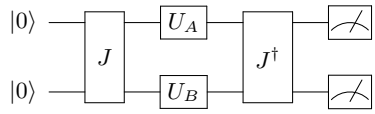

Fig. 1. Quantum circuit for EWL protocol

The library provides a layer of abstraction for generalized EWL circuits for arbitrary number of players with customizable base strategies representing the possible moves from the classical counterpart of the game. It automatically derives the corresponding entanglement operator J and its Hermitian conjugate, $J^\dagger$ from the given quantum state initial to users moves. The library comes with several built-in parametrizations [1,2], also allowing for using custom ones.

Symbolic calculations are provided by integration with SymPy package[3]. This allows for direct usage of symbolic expressions as an interface and also for simplification of statevector amplitudes and payoff functions. It also allows for easy switching from symbolic to numerical approach if necessary.
Qiskit[4] integration allows for verification of theoretical results on numerous available real IBM-Q quantum devices or simulators including noise models.
Algorithms. The library includes algorithms calculating the unitary matrix J of the EWL scheme entanglement operator, symbolic and numerical approach to finding best responses in parametrized 2×2 quantum games and, based on that, finding pure Nash equilibria or cycles of best responses.

3 Experiment Results

The example comparison of numerical search for the best response is shown in Tab. 1. Success rate was calculated with 0.01% tolerance. More examples of library usage can be found on GitHub[5].

Table 1. Comparison of numerical search for best response

Optimization method	Start point	Bounds	Success rate
Powell	zero	True	86.94%
Powell	zero	False	99.94%
Powell	random	True	76.34%
Powell	random	False	99.23%
Powell	Alice	True	79.14%
Powell	Alice	False	99.35%
Nelder-Mead	zero	True	45.89%
Nelder-Mead	zero	False	96.60%
Nelder-Mead	random	True	41.47%
Nelder-Mead	random	False	99.52%
Nelder-Mead	Alice	True	43.55%
Nelder-Mead	Alice	False	99.01%

References

1. Quantum games and quantum strategies **83**, `http://dx.doi.org/10.1103/PhysRevLett.83.3077`
2. Frackiewicz, P., Pykacz, J.: Quantum games with strategies induced by basis change rules. International Journal of Theoretical Physics **56**(12), 4017–4028 (2017)

[3] `https://www.sympy.org/`
[4] `http://qiskit.org`
[5] `https://github.com/tomekzaw/ewl/blob/master/examples/example.ipynb`

List of Figures

List of Tables

List of Algorithms

List of Symbols

General

$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers
$\mathbb{Z}$	set of integers
$\mathbb{Z}_n$	group of integers modulo n, i.e. $\{0, 1, \ldots, n\}$
i	imaginary unit, $i^2 = -1$
z^*	complex conjugate of z
$\lvert z \rvert$	absolute value of z
Δx	difference of two values
$x \approx y$	x is approximately equal to y
$\exp(x), e^x$	exponential function
$f : A \to B$	function f mapping elements from A to B
$\operatorname*{argmax}_{x \in X} f(x)$	arguments of the maxima

Linear algebra

$\mathbf{v}$	row vector
$\mathbf{v}^\top$	column vector
a_{ij}	element of matrix in row i and column j
A_{ij}	element of matrix A in row i and column j
$A_{m \times n} = [a_{ij}]$	matrix of shape $m \times n$ composed of elements a_{ij}
$\mathbb{C}_{m \times n}$	set of matrices of complex numbers of shape $m \times n$
$A + B$	sum of two matrices
AB	product of two matrices
A^*	complex conjugate of matrix
$A^\top$	matrix transposition of matrix
$A^\dagger$	Hermitian conjugate of matrix
$\exp(A), e^A$	matrix exponential

Set theory

$\forall$	universal quantifier
$\exists$	existential quantifier
$x \in X$	x is an element of set X, x belongs to set X
$\bigcup_{i=1}^{n} X_i$	set sum of sets $X_1, X_2, \ldots, X_n$
$A \setminus B$	set difference
$A \times B$	Cartesian product of sets A and B
A^n	n-th Cartesian product of set A
$\bigtimes_{i=1}^{n} A_i$	Cartesian product of sets A_i
$\{a_i\}_{i=1}^{n}$	sequence of $(a_1, a_2, \ldots, a_n)$
$\{1, 2, \ldots, n\}$	set of consecutive integers starting from 1 to n

Quantum theory

$\|\psi\rangle$	ket, column vector
$\langle\psi\|$	bra, row vector
$\langle\phi\|\psi\rangle$	bra-ket, inner product, scalar product or dot product
$\|\phi\rangle \langle\psi\|$	ket-bra, outer product
$A \otimes B$	tensor (Kronecker) product of two matrices
$\bigotimes_{i=1}^{n} A_i$	tensor product of matrices A_i
$\|\psi\rangle^{\otimes n}$	tensor product of $\|\psi\rangle$ with itself n times
$\sum_{i=1}^{n} \|\psi_i\rangle$	superposition of quantum states $\|\psi_i\rangle$
$\sigma_x, \sigma_y, \sigma_z$	Pauli matrices X, Y, Z
ρ	density matrix

Game theory

$\$_{\mathrm{X}}$	payoff function of player X
br_{X}	best response function of player X
θ_{X}	strategy parameter of player X
$A \to B$	strategy B is a best response for A
$\mathbf{u} \equiv \mathbf{v}$	equivalence of two vectors of strategy parameters
s_{-i}	sequence s without i-th element, i.e. $\{s_j\}_{i=1, i \neq j}^{n}$

Bibliography

[1] MD SAJID ANIS et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[2] Simon C Benjamin and Patrick M Hayden. "Comment on "Quantum Games and Quantum Strategies"". In: *Physical Review Letters* 87.6 (2001), p. 069801. DOI: 10.1103/PhysRevLett.87.069801.

[3] Simon C Benjamin and Patrick M Hayden. "Multiplayer quantum games". In: *Physical Review A* 64.3 (2001), p. 030301. DOI: 10.1103/PhysRevA.64.030301.

[4] John Bostanci and John Watrous. "Quantum game theory and the complexity of approximating quantum Nash equilibria". In: *arXiv preprint arXiv:2102.00512* (2021). DOI: arXiv:2102.00512.

[5] Bryan Randolph Bruns. "Names for games: locating 2×2 games". In: *Games* 6.4 (2015), pp. 495–520. DOI: 10.3390/g6040495.

[6] Kay-Yut Chen and Tad Hogg. "How well do people play a quantum prisoner's dilemma?" In: *Quantum Information Processing* 5.1 (2006), pp. 43–67. DOI: 10.1007/s11128-006-0012-7.

[7] Jens Eisert and Martin Wilkens. "Quantum games". In: *Journal of Modern Optics* 47.14-15 (2000), pp. 2543–2556. DOI: 10.1080/09500340008232180.

[8] Jens Eisert, Martin Wilkens, and Maciej Lewenstein. "Quantum games and quantum strategies". In: *Physical Review Letters* 83.15 (1999), p. 3077. DOI: 10.1103/PhysRevLett.83.3077.

[9] Stefan C Endres, Carl Sandrock, and Walter W Focke. "A simplicial homology algorithm for Lipschitz optimisation". In: *Journal of Global Optimization* 72.2 (2018), pp. 181–217. DOI: 10.1007/s10898-018-0645-y.

[10] Adrian P Flitney and Lloyd CL Hollenberg. "Nash equilibria in quantum games with generalized two-parameter strategies". In: *Physics Letters A* 363.5-6 (2007), pp. 381–388. DOI: 10.1016/j.physleta.2006.11.044.

[11] Piotr Frackiewicz. "Application of the EWL protocol to decision problems with imperfect recall". In: *arXiv preprint arXiv:1012.0806* (2010). DOI: 10.48550/arXiv.1012.0806.

[12] Piotr Frąckiewicz and Jarosław Pykacz. "Quantum games with strategies induced by basis change rules". In: *International Journal of Theoretical Physics* 56.12 (2017), pp. 4017–4028. DOI: 10.1007/s10773-017-3423-6.

[13] Piotr Frąckiewicz, Katarzyna Rycerz, and Marek Szopa. "Quantum absentminded driver problem revisited". In: *Quantum Information Processing* 21.1 (2022), pp. 1–21. DOI: 10.1007/s11128-021-03377-6.

[14] Franz G Fuchs, Vemund Falch, and Christian Johnsen. "Quantum Poker—a game for quantum computers suitable for benchmarking error mitigation techniques on NISQ devices". In: *The European Physical Journal Plus* 135.4 (2020), p. 353. DOI: 10.1140/epjp/s13360-020-00360-5.

[15] Filip Galas. "Quantum games on IBM-Q". In: (2019).

[16] Allan Goff. "Quantum tic-tac-toe: A teaching metaphor for superposition in quantum mechanics". In: *American Journal of Physics* 74.11 (2006), pp. 962–973. DOI: 10.1119/1.2213635.

[17] Faisal Shah Khan. "Calculating Nash equilibrium and Nash bargaining solution on quantum annealers". In: *arXiv preprint arXiv:2112.12583* (2021). DOI: arXiv:2112.12583.

[18] Faisal Shah Khan et al. "Quantum games: a review of the history, current state, and interpretation". In: *Quantum Information Processing* 17.11 (2018), pp. 1–42. DOI: 10.1007/s11128-018-2082-8.

[19] Vassili Kolokoltsov. "Quantum games: a survey for mathematicians". In: *arXiv preprint arXiv:1909.04466* (2019). DOI: arXiv:1909.04466.

[20] Piotr Kotara. "Analiza użycia schematu EWL na urządzeniach kwantowych typu NISQ". In: (2022).

[21] Holger Krekel et al. *pytest*. 2004.

[22] Steven Landsburg. "Nash equilibria in quantum games". In: *Proceedings of the American Mathematical Society* 139.12 (2011), pp. 4423–4434. DOI: arXiv:1110.1351.

[23] Carlton E Lemke and Joseph T Howson Jr. "Equilibrium points of bimatrix games". In: *Journal of the Society for industrial and Applied Mathematics* 12.2 (1964), pp. 413–423. DOI: 10.1137/0112033.

[24] Richard D McKelvey, Andrew M McLennan, and Theodore L Turocy. "Gambit: Software tools for game theory". In: (2006).

[25] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103.

[26] David A Meyer. "Quantum strategies". In: *Physical Review Letters* 82.5 (1999), p. 1052. DOI: 10.1103/PhysRevLett.82.1052.

[27] Yushi Mura and Hiroki Wada. "Quantization of blackjack: Quantum basic strategy and advantage". In: *Progress of Theoretical and Experimental Physics* 2021.10 (2021), 103A02. DOI: 10.1093/ptep/ptab125.

[28] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313. DOI: 10.1093/comjnl/7.4.308.

[29] Michael A Nielsen and Isaac L Chuang. "Quantum computation and quantum information". In: *Phys. Today* 54.2 (2001), p. 60. DOI: 10.1017/CBO9780511976667.

[30] Tadeusz Płatkowski. "Wstęp do teorii gier". In: *Uniwersytet Warszawski* (2012).

[31] Michael JD Powell. "An efficient method for finding the minimum of a function of several variables without calculating derivatives". In: *The computer journal* 7.2 (1964), pp. 155–162. DOI: 10.1093/comjnl/7.2.155.

[32] David Robinson and David Goforth. *The topology of the 2x2 games: a new periodic table*. Vol. 3. Psychology Press, 2005. DOI: 10.4324/9780203340271.

[33] Katarzyna Rycerz and Piotr Frąckiewicz. "A quantum approach to twice-repeated 2×2 game". In: *Quantum Information Processing* 19.8 (2020), pp. 1–20. DOI: 10.1007/s11128-020-02743-0.

[34] Azharuddin Shaik and Aden Ahmed. "Best Response Analysis in Two Person Quantum Games". In: *Advances in Pure Mathematics* 2014 (2014). DOI: 10.4236/apm.2014.47045.

[35] Marek Szopa. "Dlaczego w dylemat więźnia warto grać kwantowo?" In: *Studia Ekonomiczne* 178178 (2014), pp. 174–189. ISSN: 2083-8611.

[36] Marek Szopa. "Efficiency of classical and quantum games equilibria". In: *Entropy* 23.5 (2021), p. 506. DOI: 10.3390/e23050506.

[37] The Nashpy project developers. *Nashpy: v0.0.34*. DOI: 10.5281/zenodo.6620830.

[38] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[39] Tomasz Zawadzki and Piotr Kotara. *A Python tool for symbolic analysis of quantum games in EWL protocol with IBM Q integration*. https://github.com/tomekzaw/ewl.