Simultaneous Determination of Three Different Spin Projections

Using Entanglement and Weak Measurement

Joshua Matern

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Jean-François S. Van Huele, Advisor

Department of Physics and Astronomy

Brigham Young University

June 2014

ABSTRACT

Simultaneous Determination of Three Different Spin Projections
Using Entanglement and Weak Measurement

Joshua Matern
Department of Physics and Astronomy
Bachelor of Science

Uncertainty relations constrain knowledge about incompatible (non-commuting) quantum observables, such as spin components of particles. I study a technique by Vaidman, Aharonov, and Albert [Phys. Rev. Lett. 58, 1385 (1987)] to overcome these limitations using weak measurement and particle entanglement of all three Cartesian spin components. I verify that this method, which is referred to as the Buddy Method in this work, is applicable to all four Bell (maximally entangled) states. I also show that the Buddy Method gives significantly better than random probabilities for partially entangled states. Finally, I generalize the Buddy Method for arbitrary spin direction.

Keywords: Quantum, Spin, Entanglement, Weak Measurement, Incompatible Observables

ACKNOWLEDGMENTS

I would like to acknowledge Dr. Van Huele for his guidance and patience as well as Amanda Lin for her unwavering support.

Contents

List of Figures

List of Tables

Chapter 1

Motivation and Background

1.1 Introduction to Quantum Measurement

Can we fully understand quantum systems or are there fundamental limits to our knowledge? Incompatible quantum observables, corresponding to non-commuting operators, define an area of study with measurement constraints. This is due to the uncertainty inherent in such observables. I will define strong measurement as definite measurements that give precise information. Strong measurement of one of the observables loses information about the other observables. This is due to the system collapsing down to one particular eigenstate of whichever observable was measured. Because non-commuting operators do not share their eigenvectors, the strong measurement of one operator leaves multiple possible outcomes for the incompatible second measurement. A new method is needed to get more information and see if it is possible to overcome these limitations. Vaidman *et al* [1] proposes such a method using weak measurement and entangled states. This method will hereafter be called the Buddy Method in order to emphasize the importance played by an entangled, external system.

Entangled states are superpositions of quantum states of two subsystems where the subsystem can only be measured in their relation to the system as a whole and not individually. For example, if we have the entangled state

$$|00\rangle_{AB} + |11\rangle_{AB},$$

then all there is to know about the system is $A+B$. We know nothing about A or B individually because we ignore the value of each of them. Measurements need to be taken without collapsing entanglement, but as previously stated, strong measurement causes collapse with loss of information.

A fundamental property of all particles is spin, and its three components are incompatible observable. As such, following the lead of Vaidman, spin makes an ideal choice for testing the buddy method. Electron spin was chosen by Vaidman so we will do the same. Electrons are prevalent throughout the universe and their spin can take only two possible values of spin: up or down.

Spin, $\vec{S}$, is a vector quantity and an operator in quantum mechanics. It thus has three Cartesian projections, usually labeled as the S_x, S_y, and S_z components, corresponding to the orthogonal axes x, y, and z in 3-D space. These projections can be represented as the spin matrices σ_x , σ_y, and σ_z, where these components are quantum operators, which can be measured for any particular state and thus describe spin outcome measurements in space.

1.2 Quantum Techniques

1.2.1 Weak Measurement

Weak measurement allows for a form of measurement without collapsing states including entangled states. This is accompanied by choosing a state, called the pre-state, when preparing the system and selecting specific outcomes called post-states by discarding all other measurement outcomes. The system is then weakly probed in between these pre-selection and post-selection states in order to measure it. This weak interaction avoids a collapse of the wavefunction while simultaneously giving a value of the measures observable. A trade-off between accuracy in the measurement and lack of disturbance in the weak probe reflects quantum complementary. Data not matching the post-states are discarded, so as to gain a more accurate outcome. Performing this weak measurement many times allows us to know with increasing certainty which state the system is in.

Current experimental techniques [2] make weak measurement possible and enable us to determine two of the three spin projections. The pre-selected and post-selected states correspond to eigenstates of incompatible measurements such as σ_x and σ_z. It has been shown that using weak measurement allows for inferences for two measurements on multiple copies of a single system at a given time [3]. This allows for the accurate determination of two of the three spin components of a spin-$\frac{1}{2}$ particle. That still leaves no commonly accepted nor experimentally verified method to infer with certainty all three Cartesian components of a spin-$\frac{1}{2}$ particle.

1.2.2 Particle Entanglement

So far it does not seem possible to determine all three projections of particle spin. The standard measurement techniques used for quantum-mechanical systems cannot yield dispersion-free inferences (i.e. predictions without uncertainty) of more than two non-commuting observable of a single system at a single time. Attempting to measure all three spin components even using weak measurement with pre- and post-selection still leads to collapsing the spin system and losing information.

The use of entanglement with a second particle in conjunction with weak measurement is now proposed as a method for getting all three spin projections [1]. The system that we wish to know all three components for is entangled with another system. The physicist does weak measurement on the newly entangled system with newly selected pre- and post-states. Ignoring the components of

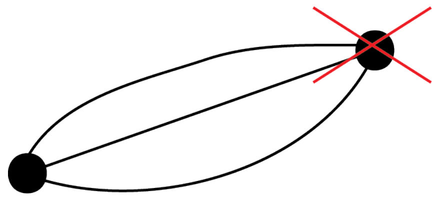

Figure 1.1 A particle entangled with another particle. The components for the second particle are ignored, being used only to obtain more information about the original particle.

the external, or buddy, system introduced for the entanglement as in Figure 1.1, the physicist is then able to infer with certainty the possible measurement outcome of any of the Cartesian components of the spin-$\frac{1}{2}$ system. As we consider entangled states, we will also consider joint measurement operators, which are operators on the full system rather than on the two subsystems individually.

This entanglement is a purely mathematical construct. No actual particle needs be actually introduced into the system. We use it as a point of reference in probability calculations. Two of the spin components are found using the previously outlined weak measurement method. Entanglement then allows us to infer the remaining spin component. The Buddy Method Vaidman *et al.* presented, however, is limited to a maximally entangled (or Bell) state. In the following, I study to what extent this method can be expanded to states other than this Bell state.

Chapter 2

Theoretical Methods

2.1 The Buddy Method

2.1.1 Background

As seen in Chapter One, Vaidman *et al.* developed a method to get all three projections of spin for a spin-$\frac{1}{2}$ by using entanglement with weak measurement. In their paper, the authors use the conventional language of quantum mechanics for easier access although their inspiration derives from their two-state formalism of quantum mechanics [1]. This is the only method I am aware of that can theoretically infer all three spin projections. Verification of this method will show that it can be used to lead to more accurate analyses of incompatible quantum observables.

In its original version, this method was demonstrated using the $|\psi_+\rangle$ Bell state

$$|\psi_+\rangle = \frac{1}{2}\sqrt{2}(|\uparrow\rangle_{ext}|\uparrow\rangle + |\downarrow\rangle_{ext}|\downarrow\rangle)$$

as the normalized pre-state where $|\uparrow\rangle$ and $|\downarrow\rangle$ refer to spin up and spin down in the z axis. The subscript *ext* refers to the buddy or auxiliary spin system. This method was not extended to other Bell states or states that were not maximally entangled.

A set of eigenvalues a_i and normalized eigenstates Φ_i as shown in Table 2.1 are chosen for the post-selected states. The eigenstates and the pre-state $|\psi_+\rangle$ are expressed using $|\uparrow\rangle$ and $|\downarrow\rangle$ with the external particle showing up in front. The eigenvalues for these states, $|\Phi_i\rangle$, are determined by $A|\Phi_i\rangle = a_i|\Phi_i\rangle$, $i = 1,...,4$. It is assumed that there is no degeneracy $a_i \neq a_j$. The operator used for the measurement was not given. Through reverse-engineering of the eigenstates used in the post-states, I determine the operator A, which can be found in Appendix A.

$$
\begin{aligned}
|\Phi_1\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\uparrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle e^{i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle e^{-i\pi/4})\\
|\Phi_2\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\uparrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle e^{i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle e^{-i\pi/4})\\
|\Phi_3\rangle &= \tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\downarrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle e^{-i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle e^{i\pi/4})\\
|\Phi_4\rangle &= \tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\downarrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle e^{-i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle e^{i\pi/4})
\end{aligned}
$$

Table 2.1 The post-state eigenstates for the $|\psi_+\rangle$ pre-state.

The Buddy Method is outlined as follows: Each joint operator measurement corresponds to $\sigma_i \otimes 1$ or $1 \otimes \sigma_i$ meaning either the external or first particle being measured as an operator state and then entangled with another particle, the buddy, on which no measurement is performed, represented by an identity matrix. These are labeled as First particle and External particle respectively in Table 2.2. This joining is done mathematically by taking the tensor product of the operator state with the identity matrix, both with the operator state first and the identity state first. Each operator yields only two different eigenvalues c_n, namely, ± 1 each with multiplicity two. This results in two different projection matrices. Using the variable C, the probability for any given result $C = c_n$ at any time between two pre- and post-measurements on the entangled state is

$$p(C = c_n) = \frac{|\langle\Phi_i|P_{C=c_n}|\psi\rangle|^2}{\Sigma_j|\langle\Phi_i|P_{C=c_n}|\psi\rangle|^2} \tag{2.1}$$

with $P_{C=C_n}$ being the projection matrices on the subspace of the two eigenvectors corresponding to eigenvalue c_n. After inputting the projection matrices into this equation with the selected pre- and post-states, it is claimed the probabilities will be either one or zero for the spin up or spin down state for each of the four post-states for the chosen σ_i as shown in Table 2.2. Probabilities of 100% or 0% mean we have 100% knowledge of the spin state eigenvalues along the three axes.

	First particle			External particle		
A	x	y	z	x	y	z
a_1	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$
a_2	$\downarrow$	$\downarrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$
a_3	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$
a_4	$\downarrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$

Table 2.2 The possible outcomes of the spin measurement for any of the three Cartesian components of the $|\psi_+\rangle$ Bell state given specific eigenstates indicating post-selected states

2.1.2 Verification

In order to check this method, we replicate the technique outlined in order to see if we obtain the same results. We first find the two projection matrices for each eigenstate and apply these to the probability equation. We then determine the probabilities of possible spin measurements of spin-$\frac{1}{2}$ particles in the $|\psi_+\rangle$ entangled state, from which we also derived the results in Table 2.2. These results match the probabilities in [1] and confirm the validity of the method.

2.2 Application of the Buddy Method

2.2.1 Extension

We now extend this method to the three other Bell states listed in Table 2.3. We want to check if the Buddy Method is applicable and equally successful for any maximally entangled state. Initial attempts using the $|\psi_-\rangle$ give 50% probabilities when using the post-states of Table 2.1. We conclude a new operator tailored to this pre-state needs to be created with new eigenstates as the post-states. These eigenstates are found in Table 2.4. The eigenvalues have the same structure as the $|\psi_-\rangle$ pre-state. This indicates that we expect the method needs to use appropriately worked pre- and post-states because Bell states should be equivalent.

$$
\begin{aligned}
|\psi_-\rangle &= \tfrac{1}{2}\sqrt{2}(|\uparrow\rangle_{ext}|\uparrow\rangle - |\downarrow\rangle_{ext}|\downarrow\rangle)\\
|\phi_+\rangle &= \tfrac{1}{2}\sqrt{2}(|\uparrow\rangle_{ext}|\downarrow\rangle + |\downarrow\rangle_{ext}|\uparrow\rangle)\\
|\phi_-\rangle &= \tfrac{1}{2}\sqrt{2}(|\uparrow\rangle_{ext}|\downarrow\rangle - |\downarrow\rangle_{ext}|\uparrow\rangle)
\end{aligned}
$$

Table 2.3 The Bell states other than the $|\psi_+\rangle$ pre-state.

$$
\begin{aligned}
|\Gamma_1\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\uparrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle ie^{i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle ie^{-i\pi/4})\\
|\Gamma_2\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\uparrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle ie^{i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle ie^{-i\pi/4})\\
|\Gamma_3\rangle &= -\tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\downarrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle ie^{-i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle ie^{i\pi/4})\\
|\Gamma_4\rangle &= -\tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\downarrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\downarrow\rangle ie^{-i\pi/4} + |\downarrow\rangle_{ext}|\uparrow\rangle ie^{i\pi/4})
\end{aligned}
$$

Table 2.4 The post-state eigenstates for the $|\psi_-\rangle$ pre-state.

The eigenstates are found in an artificial manner by realizing that a negative value can be obtained by the square of an imaginary number. As such, whenever a $|\downarrow\rangle$ appears in the eigenstates for the $|\psi_+\rangle$ eigenstates, we multiply it by i. After implementing the method with these eigenstates, we determine they are valid and so the artificial method works as expected.

After showing the method can be extended to one other Bell state, we can be confident it extends to the rest. For the sake of completeness, we carry out the method on the $|\phi_+\rangle$ and $|\phi_-\rangle$ pre-states with eigenstates found in Table 2.5 and Table 2.6 respectively. The transformation is achieved by flipping the second arrow for each of the corresponding $|\psi\rangle$ states.

$$\begin{aligned}
|\Delta_1\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\downarrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle e^{i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle e^{-i\pi/4})\\
|\Delta_2\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\downarrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle e^{i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle e^{-i\pi/4})\\
|\Delta_3\rangle &= \tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\uparrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle e^{-i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle e^{i\pi/4})\\
|\Delta_4\rangle &= \tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\uparrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle e^{-i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle e^{i\pi/4})
\end{aligned}$$

Table 2.5 The post-state eigenstates for the $|\phi_+\rangle$ pre-state.

$$\begin{aligned}
|\xi_1\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\downarrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle ie^{i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle ie^{-i\pi/4})\\
|\xi_2\rangle &= \tfrac{1}{2}\sqrt{2}|\uparrow\rangle_{ext}|\downarrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle ie^{i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle ie^{-i\pi/4})\\
|\xi_3\rangle &= -\tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\uparrow\rangle + \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle ie^{-i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle ie^{i\pi/4})\\
|\xi_4\rangle &= -\tfrac{1}{2}\sqrt{2}|\downarrow\rangle_{ext}|\uparrow\rangle - \tfrac{1}{2}(|\uparrow\rangle_{ext}|\uparrow\rangle ie^{-i\pi/4} + |\downarrow\rangle_{ext}|\downarrow\rangle ie^{i\pi/4})
\end{aligned}$$

Table 2.6 The post-state eigenstates for the $|\phi_-\rangle$ pre-state.

We now apply the method to partially entangled states for further extension of the Buddy Method. This is done by taking the $|\psi_+\rangle$ Bell state but generalizing it outside of maximum entanglement to

$$|\psi_+\rangle = \varepsilon|\uparrow\rangle_{ext}|\uparrow\rangle + \sqrt{1-\varepsilon^2}|\downarrow\rangle_{ext}|\downarrow\rangle$$

where ε is a real number between 0 and 1. When epsilon is zero or one, we have product (or non-entangled) states. When epsilon is $\frac{1}{\sqrt{2}}$, we have recovered a Bell state. We apply this transformation to the post-states as well and get the results in Table 2.7. As long as we are consistent, it does not matter which order we use for ε and $\sqrt{1-\varepsilon^2}$. We then carry out the same method as before.

$$
\begin{aligned}
|\Lambda_1\rangle &= \tfrac{\sqrt{1-\varepsilon^2}}{k}\tfrac{\sqrt{2}}{2}|\uparrow\rangle_{ext}|\uparrow\rangle + \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\uparrow\rangle_{ext}|\downarrow\rangle e^{i\pi/4} + \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\downarrow\rangle_{ext}|\uparrow\rangle e^{-i\pi/4}\\
|\Lambda_2\rangle &= \tfrac{\sqrt{1-\varepsilon^2}}{k}\tfrac{\sqrt{2}}{2}|\uparrow\rangle_{ext}|\uparrow\rangle - \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\uparrow\rangle_{ext}|\downarrow\rangle e^{i\pi/4} - \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\downarrow\rangle_{ext}|\uparrow\rangle e^{-i\pi/4}\\
|\Lambda_3\rangle &= \tfrac{\varepsilon}{k}\tfrac{\sqrt{2}}{2}|\downarrow\rangle_{ext}|\downarrow\rangle + \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\uparrow\rangle_{ext}|\downarrow\rangle e^{-i\pi/4} + \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\downarrow\rangle_{ext}|\uparrow\rangle e^{i\pi/4}\\
|\Lambda_4\rangle &= \tfrac{\varepsilon}{k}\tfrac{\sqrt{2}}{2}|\downarrow\rangle_{ext}|\downarrow\rangle - \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}(|\uparrow\rangle_{ext}|\downarrow\rangle e^{-i\pi/4} + \tfrac{\sqrt{\varepsilon}(1-\varepsilon^2)^{\frac{1}{4}}}{2k}|\downarrow\rangle_{ext}|\uparrow\rangle e^{i\pi/4})
\end{aligned}
$$

Table 2.7 The post-state eigenvalues with the normality factor k for the partially entangled $|\psi_+\rangle$ pre-state.

To know how far this method can be extended with the operators and to try and determine the range, we make linear combinations of both the x and y directions as well as the x, y, and z directions using polar notation. We then graph the first linear combination using a standard plot and the second using a spherical $3D$ plot as shown in Chapter 3.1.3.

Chapter 3

Results

3.1 Data

3.1.1 Bell States

The $|\psi_{-}\rangle$, $|\phi_{+}\rangle$, and $\phi_{-}\rangle$ pre-state probability results are shown in Table 3.1, Table 3.2, and Table 3.3 respectively.

	First particle			External particle		
A	x	y	z	x	y	z
a_1	$\uparrow$	$\downarrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$
a_2	$\downarrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
a_3	$\downarrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\downarrow$	$\downarrow$
a_4	$\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\downarrow$

Table 3.1 The possible outcomes of the spin measurement for any of the three Cartesian components of the $|\psi_{-}\rangle$ Bell state given specific eigenvalues indicating post-selected states

	First particle			External particle		
A	x	y	z	x	y	z
a_1	↑	↑	↑	↑	↑	↓
a_2	↓	↓	↑	↓	↓	↓
a_3	↑	↓	↓	↑	↓	↑
a_4	↓	↑	↓	↓	↑	↑

Table 3.2 The possible outcomes of the spin measurement for any of the three Cartesian components of the $|\phi_+\rangle$ Bell state given specific eigenvalues indicating post-selected states

	First particle			External particle		
A	x	y	z	x	y	z
a_1	↑	↓	↑	↓	↑	↓
a_2	↓	↑	↑	↑	↓	↓
a_3	↓	↓	↓	↑	↑	↑
a_4	↑	↑	↓	↓	↓	↑

Table 3.3 The possible outcomes of the spin measurement for any of the three Cartesian components of the $|\phi_-\rangle$ Bell state given specific eigenvalues indicating post-selected states

3.1.2 Partially Entangled States

The partially entangled states resulted in continuous functions for each post-state. The probability is always one or zero for σ_z. The results for σ_x partially entangled states are shown in Figure 3.1. The results for σ_y partially entangled states are shown in Figure 3.2. The line that reaches probability one and the line that reaches probability zero indicates spin up or spin down depending on which post-state is being used. The first and fourth post-state are the same and the second and third are the same for σ_y. For σ_x, the first and third post-state are identical and the second and fourth are identical.

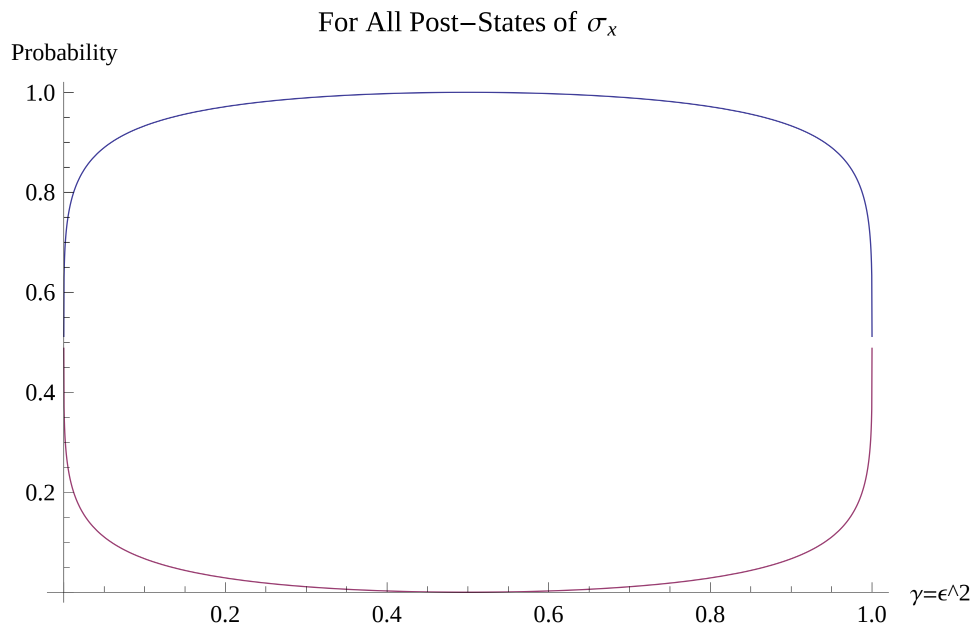

Figure 3.1 Partially entangled state with σ_x for all post-states. The line for up or down differs with each post-state.

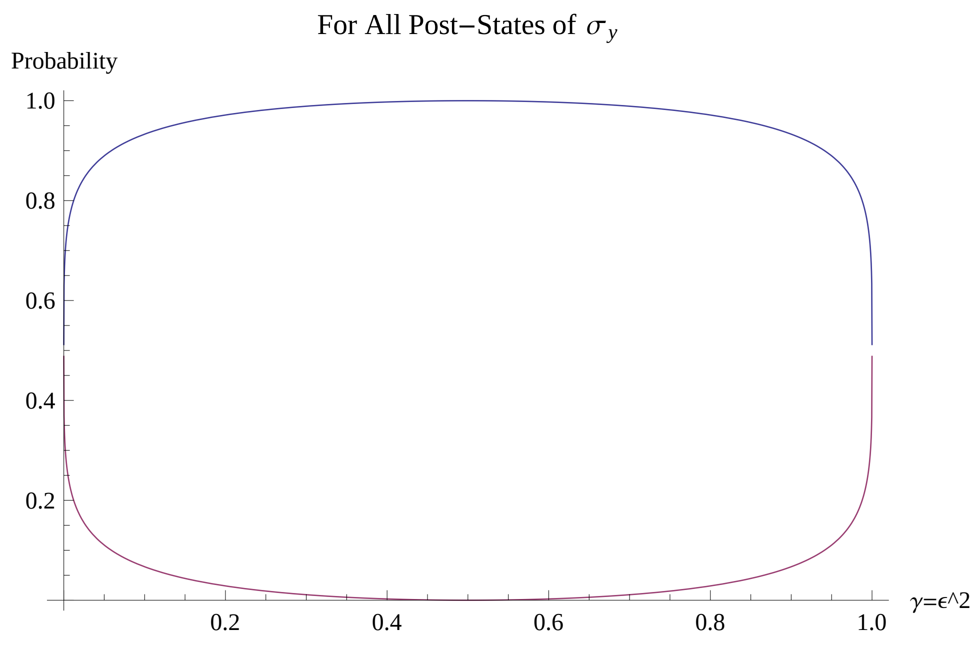

Figure 3.2 Partially entangled state with σ_y for all post-states. The line for up or down differs with each post-state.

3.1.3 Linear Combination of Operators

The plots for arbitrary linear combinations of σ_x and σ_y in the equatorial plane

$$\sigma_\phi \otimes \mathbb{1} = (\cos\phi\,\sigma_x + \sin\phi\,\sigma_x) \otimes \mathbb{1}$$

are shown in Figure 3.3 and Figure 3.4. The line corresponding to spin-up and spin-down flip between post-states such that the spin-up line for $|\Phi_1\rangle$ is the same as the spin-down result for $|\Phi_2\rangle$. The plots for arbitrary linear combinations in space

$$\sigma_{\theta,\phi} \otimes \mathbb{1} = (\sin\theta\cos\phi\,\sigma_x + \sin\theta\sin\phi\,\sigma_x + \cos\theta\,\sigma_z) \otimes \mathbb{1}$$

are shown in Figure 3.5-Figure 3.8. The sphere surrounding the plots is a representation of probability one for all spin directions.

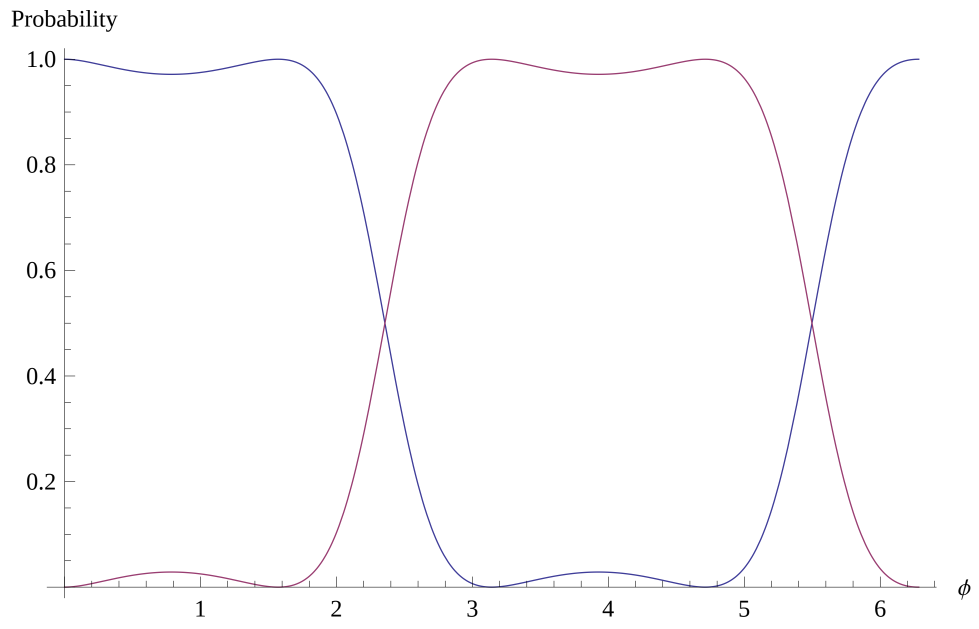

Figure 3.3 Linear combination of operators σ_x and σ_y for post-states $|\Phi_1\rangle$ and $|\Phi_2\rangle$. The line for up or down differs with each post-state.

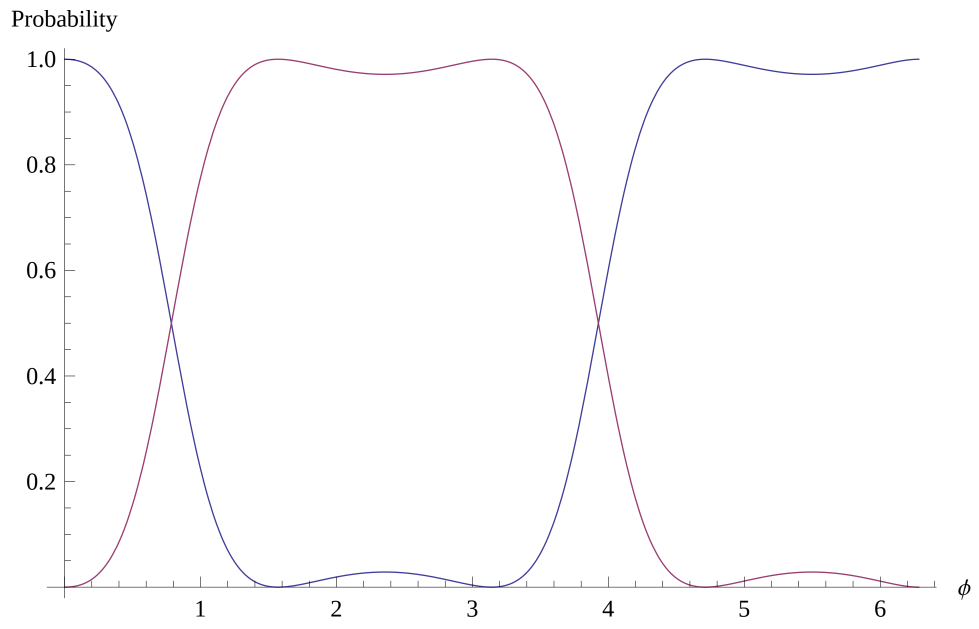

Figure 3.4 Linear combination of operators σ_x and σ_y for post-states $|\Phi_3\rangle$ and $|\Phi_4\rangle$. The line for up or down differs with each post-state.

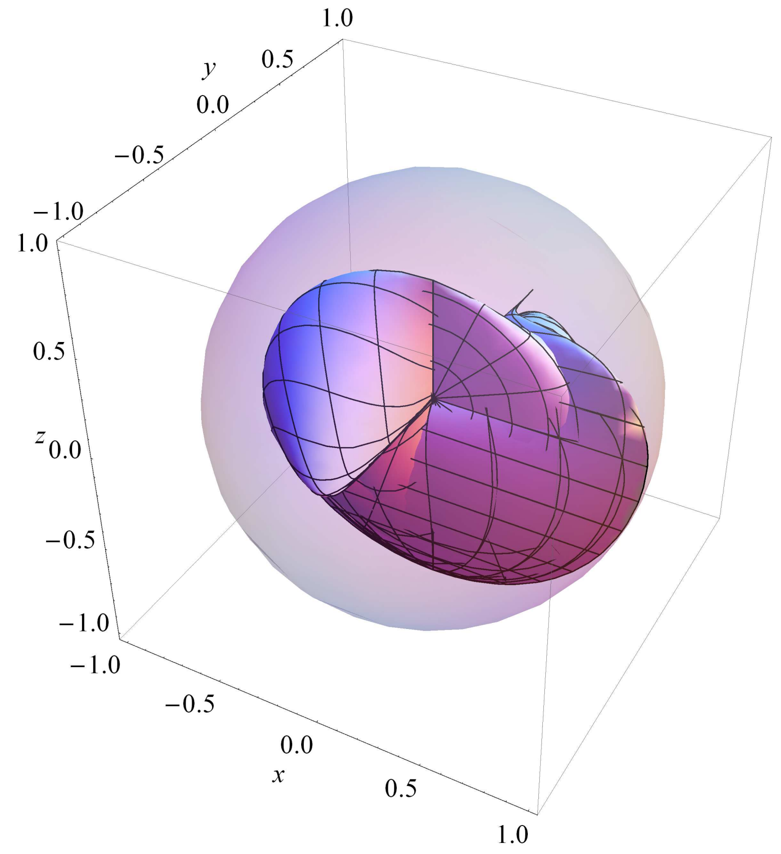

Figure 3.5 Linear combination of operators σ_x, σ_y, and σ_z for the first $|\psi_+\rangle$ post-state.

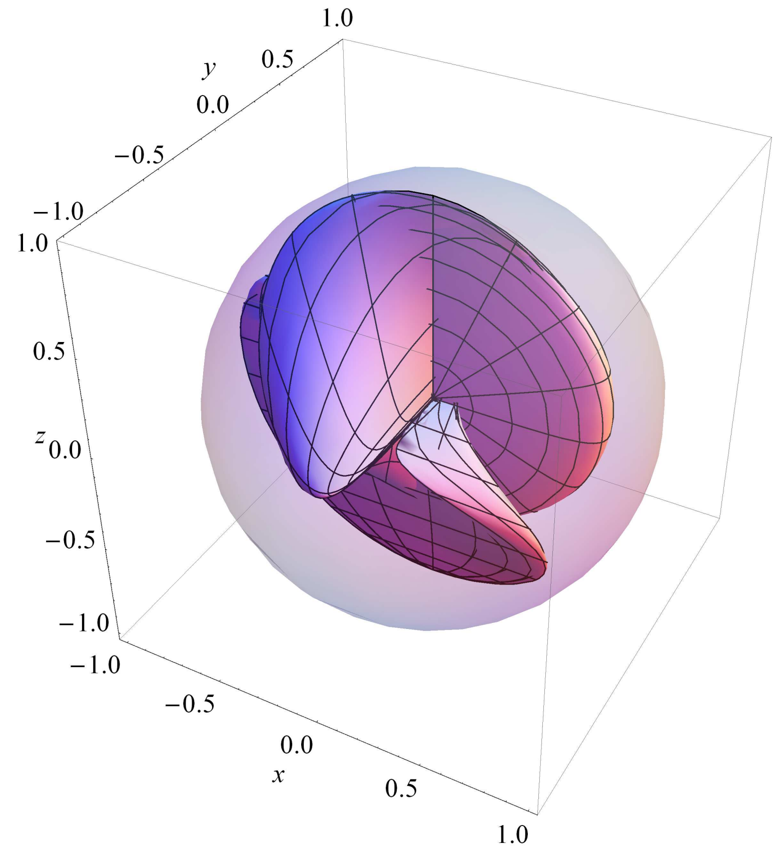

Figure 3.6 Linear combination of operators σ_x, σ_y, and σ_z for the second $|\psi_+\rangle$ post-state.

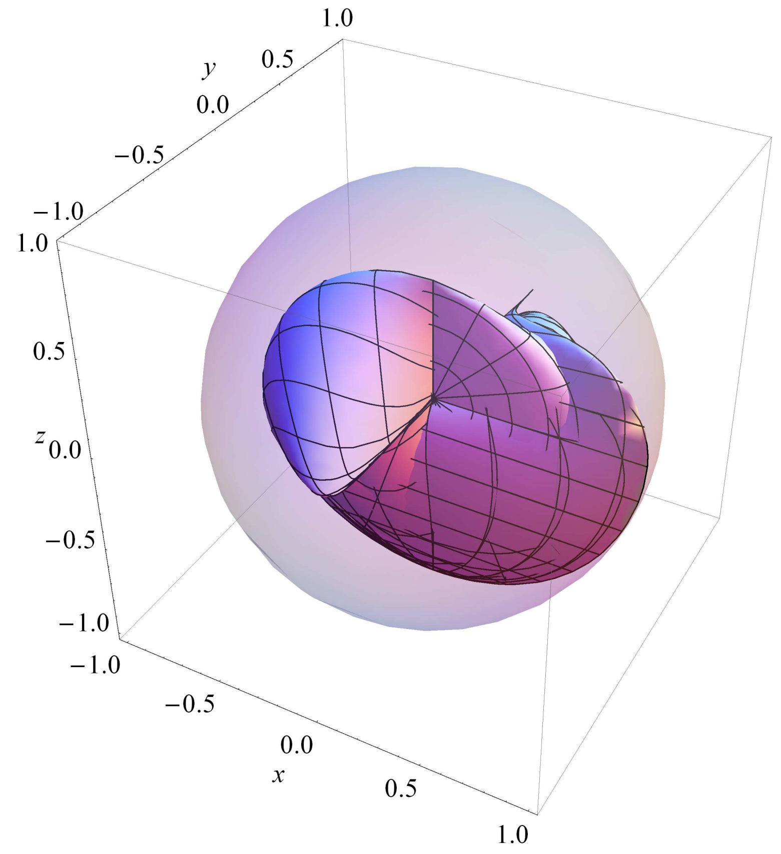

Figure 3.7 Linear combination of operators σ_x, σ_y, and σ_z for the third $|\psi_+\rangle$ post-state.

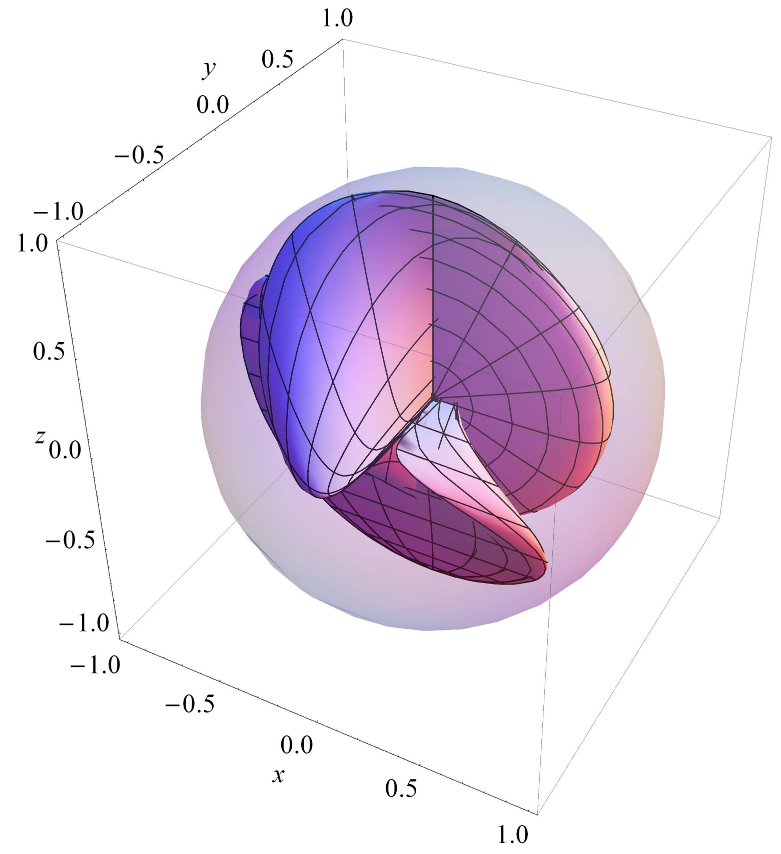

Figure 3.8 Linear combination of operators σ_x, σ_y, and σ_z for the fourth $|\psi_+\rangle$ post-state.

3.2 Analysis

The results for the $|\psi_-\rangle$ pre-selected state are different from the $|\psi_+\rangle$ pre-selected state. For each of the four Bell pre-states, the post-state eigenstate has unique values of the spin projections. As such, knowing the eigenstate tells us what components to expect and conversely knowing the components tells us which eigenstate we are in. In the $|\psi_+\rangle$ pre-state, the external particle's y component differs from the initial particle in that they are flipped. In the $|\psi_-\rangle$ pre-state, it is the x component that differs and is flipped. For the $|\psi_+\rangle$ pre-state, the z component is flipped while the $|\psi_-\rangle$ state has both the y and z component flipped.

The plots of the partially entangled states confirms what we already knew about the maximally entangled state, found in the center of the plots, but it also shows that we can still have very high probabilities of knowing the particle spin. We have a greater than 90% probability of knowing until ε is approximately 0.1 and 0.9. This indicates that not only is the Buddy Method useful for maximally entangled states, but it is also useful for most partially entangled states. It is only when we lose entanglement that we get equal probabilities for both outcomes of the spin projection values.

The plots of arbitrary linear combinations in the equatorial plane show that, as expected, there is a period of 2π for the combinations, and that there is a high probability of knowing the spin direction except at certain values of ϕ marked at regular intervals and approximately $\frac{\pi}{4}$ radians around them as shown in Figure 3.3 and Figure 3.4. For the first and second post-states, the intervals are at $\frac{3\pi}{4}$ and $\frac{7\pi}{4}$. For the third and fourth post-states, the intervals are at $\frac{\pi}{4}$ and $\frac{5\pi}{4}$. The probability of up or down being the probability closer to one differs depending on which post-state is being used.

The plots of arbitrary linear combinations in space show that certain directions have 100% probability of being known, but there are large dead areas where no information better than random probability is known. There is a large circle in the xz-plane in Cartesian space when y is zero that we get almost probability one for each post-state. The probability also extends in a cone on either side of the circle up to the y-axis in Cartesian space. This would seem to indicate the cone of possible information we can obtain from the Buddy Method.

3.3 Conclusions

3.3.1 Application

The Buddy Method can be extended to all maximally entangled states and most partially entangled states. Although Vaidman et al. only demonstrated this method for the $|\psi_{+}\rangle$ pre-state, we showed this method works for any maximally entangled state with the other three Bell states explicitly worked out. We also showed that except for the extreme ends away from the maximally entangled states, partially entangled states can also be known with a high degree of certainty, although not 100% certainty. There are limitations to the Buddy Method, as was demonstrated by using linear combinations which showed areas no better than random probability, but this method still gives more information than was previously thought possible for incompatible observables.

3.3.2 Further Exploration

We propose here three possible extensions of this work. First, the Buddy Method could be extended to three particle states to see if that changes any information we can obtain. Next, we studied spin-$\frac{1}{2}$ states but this method could also be applied to spin-1 states to see if it is possible to determine with certainty all of their spin components. Finally, this method could also be applied to other incompatible quantum observables besides spin.

3.3.3 Final Thoughts

The language of quantum mechanics is very versatile. Using it, we were able to perform this theoretical experiment. Furthermore, it did not require the use of difficult mathematics but only an understanding of linear algebra. This field should be easily accessible for new physics researchers who could possibly look into other incompatible quantum observables.

It is notable that through a mathematical construct, in this case entanglement with a particle whose information we do not care about, we are able to take existing data and infer with certainty additional information. The Buddy Method is limited to being retrodictive rather than predictive. It can be applied when a weak measurement has already been made in order to find the value of the remaining observables. It is unable to predict the values for an incompatible observable. Such techniques and constructs could possibly be created and used in other fields of science.

I hope that the Buddy Method gains widespread focus so that it becomes the subject of more study. We have shown that though there are limitations to our analysis of incompatible observables, some of these limitations have been made much smaller. Perhaps through further extensions of the Buddy Method, we will eliminate further limitations to studying incompatible observables.

Appendix A

Operator

.

$$\begin{pmatrix}
\frac{1}{2}a_1+\frac{1}{4}a_2+\frac{1}{2}a_3 & \frac{1}{2}a_1-\frac{1}{4}a_2-\frac{1}{4}a_3 & \frac{1}{4}ia_2-\frac{1}{4}ia_3 & -\frac{1}{4}ia_2+\frac{1}{4}ia_3 \\
\frac{1}{2}a_1-\frac{1}{4}a_2-\frac{1}{4}a_3 & \frac{1}{2}a_1+\frac{1}{4}a_2+\frac{1}{4}a_3 & -\frac{1}{4}ia_2+\frac{1}{4}ia_3 & \frac{1}{4}ia_2-\frac{1}{4}ia_3 \\
-\frac{1}{4}ia_2+\frac{1}{4}ia_3 & \frac{1}{4}ia_2-\frac{1}{4}ia_3 & \frac{1}{4}a_2+\frac{1}{4}a_3+\frac{1}{2}a_4 & -\frac{1}{4}a_2-\frac{1}{4}a_3+\frac{1}{2}a_4 \\
\frac{1}{4}ia_2-\frac{1}{4}ia_3 & -\frac{1}{4}ia_2+\frac{1}{4}ia_3 & -\frac{1}{4}a_2-\frac{1}{4}a_3+\frac{1}{2}a_4 & \frac{1}{4}a_2+\frac{1}{4}a_3+\frac{1}{2}a_4
\end{pmatrix}$$

Appendix B

Mathematica Code

In this appendix, I include all of the relevant code used to complete my research.

```
Z = .5 {{Sqrt[2], Exp[I Pi/4], Exp[-I Pi/4],
     0}, {Sqrt[2], -Exp[I Pi/4], -Exp[-I Pi/4], 0}, {0, Exp[-I Pi/4],
     Exp[I Pi/4], Sqrt[2]}, {0, -Exp[-I Pi/4], -Exp[I Pi/4], Sqrt[2]}};
V = ConjugateTranspose[Z];
S = Inverse[Z];
Abs[Det[Z]];
Z.V // FullSimplify;
L = {{a1, 0, 0, 0}, {0, a2, 0, 0}, {0, 0, a3, 0}, {0, 0, 0, a4}};


A = Z.L.V;
A // Chop // FullSimplify // MatrixForm
```

Begin with first pre and post states

```
psi = Sqrt[2]/2 SparseArray[{{1, 1} -> 1, {4, 1} -> 1}, {4, 4}];
```

```
phi1 = SparseArray[{{1, 1} -> Sqrt[2]/2, {2, 1} ->
     Exp[I Pi/4]/2, {3, 1} -> Exp[-I Pi/4]/2}, {4, 4}];
phi2 = SparseArray[{{1, 1} ->
     Sqrt[2]/2, {2, 1} -> -Exp[I Pi/4]/2, {3, 1} -> -Exp[-I Pi/4]/
      2}, {4, 4}];
phi3 = SparseArray[{{4, 1} -> Sqrt[2]/2, {2, 1} ->
     Exp[-I Pi/4]/2, {3, 1} -> Exp[I Pi/4]/2}, {4, 4}];
phi4 = SparseArray[{{4, 1} ->
     Sqrt[2]/2, {2, 1} -> -Exp[-I Pi/4]/2, {3, 1} -> -Exp[I Pi/4]/
      2}, {4, 4}];
phi1t = SparseArray[{{1, 1} -> Sqrt[2]/2, {1, 2} ->
     Exp[-I Pi/4]/2, {1, 3} -> Exp[I Pi/4]/2}, {4, 4}];
phi2t = SparseArray[{{1, 1} ->
     Sqrt[2]/2, {1, 2} -> -Exp[-I Pi/4]/2, {1, 3} -> -Exp[I Pi/4]/
      2}, {4, 4}];
phi3t = SparseArray[{{1, 4} -> Sqrt[2]/2, {1, 2} ->
     Exp[I Pi/4]/2, {1, 3} -> Exp[-I Pi/4]/2}, {4, 4}];
phi4t = SparseArray[{{1, 4} ->
     Sqrt[2]/2, {1, 2} -> -Exp[I Pi/4]/2, {1, 3} -> -Exp[-I Pi/4]/
      2}, {4, 4}];
```

Construct matrices for sigmas x, y, and z. Find eigenvalues, eigenstates. Make projection matrices.

```
Mz1 = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, -1}};
Eigenvalues[Mz1]
Eigenvectors[Mz1]
```

```
Mz1e = SparseArray[{{1, 1} -> 1}, {4, 4}];

Mz1et = Normal[SparseArray[{{1, 1} -> 1}, {4, 4}]];

Mz1e2 = SparseArray[{{2, 1} -> 1}, {4, 4}];

Mz1e2t = SparseArray[{{1, 2} -> 1}, {4, 4}];

Mz1p = Mz1e.Mz1et + Mz1e2.Mz1e2t;

MatrixForm[%]


Mz1e3 = SparseArray[{{3, 1} -> 1}, {4, 4}];

Mz1e3t = Normal[SparseArray[{{1, 3} -> 1}, {4, 4}]];

Mz1e4 = SparseArray[{{4, 1} -> 1}, {4, 4}];

Mz1e4t = SparseArray[{{1, 4} -> 1}, {4, 4}];

Mz1n = Mz1e3.Mz1e3t + Mz1e4.Mz1e4t;

MatrixForm[%]


My1 = {{0, 0, I, 0}, {0, 0, 0, I}, {-I, 0, 0, 0}, {0, -I, 0, 0}};

Eigenvalues[My1]

Eigenvectors[My1]

My1e = SparseArray[{{1, 1} -> I, {3, 1} -> 1}, {4, 4}];

My1et = SparseArray[{{1, 1} -> -I, {1, 3} -> 1}, {4, 4}];

My1e2 = SparseArray[{{2, 1} -> I, {4, 1} -> 1}, {4, 4}];

My1e2t = SparseArray[{{1, 2} -> -I, {1, 4} -> 1}, {4, 4}];

My1p = My1e.My1et + My1e2.My1e2t;

MatrixForm[%]


My1e3 = SparseArray[{{1, 1} -> -I, {3, 1} -> 1}, {4, 4}];
```

```
My1e3t = SparseArray[{{1, 1} -> I, {1, 3} -> 1}, {4, 4}];

My1e4 = SparseArray[{{2, 1} -> -I, {4, 1} -> 1}, {4, 4}];

My1e4t = SparseArray[{{1, 2} -> I, {1, 4} -> 1}, {4, 4}];

My1n = My1e3.My1e3t + My1e4.My1e4t;

MatrixForm[%]


Mx1 = {{0, 0, 1, 0}, {0, 0, 0, 1}, {1, 0, 0, 0}, {0, 1, 0, 0}};

Eigenvalues[Mx1]

Eigenvectors[Mx1]

Mx1e = SparseArray[{{1, 1} -> 1, {3, 1} -> 1}, {4, 4}];

Mx1et = SparseArray[{{1, 1} -> 1, {1, 3} -> 1}, {4, 4}];

Mx1e2 = SparseArray[{{2, 1} -> 1, {4, 1} -> 1}, {4, 4}];

Mx1e2t = SparseArray[{{1, 2} -> 1, {1, 4} -> 1}, {4, 4}];

Mx1p = Mx1e.Mx1et + Mx1e2.Mx1e2t;

MatrixForm[%]


Mx1e3 = SparseArray[{{1, 1} -> -1, {3, 1} -> 1}, {4, 4}];

Mx1e3t = SparseArray[{{1, 1} -> -1, {1, 3} -> 1}, {4, 4}];

Mx1e4 = SparseArray[{{2, 1} -> -1, {4, 1} -> 1}, {4, 4}];

Mx1e4t = SparseArray[{{1, 2} -> -1, {1, 4} -> 1}, {4, 4}];

Mx1n = Mx1e3.Mx1e3t + Mx1e4.Mx1e4t;

MatrixForm[%]


M1y = {{0, I, 0, 0}, {-I, 0, 0, 0}, {0, 0, 0, I}, {0, 0, -I, 0}};

Eigenvalues[M1y]
```

```
Eigenvectors[M1y]

M1ye = SparseArray[{{1, 1} -> I, {2, 1} -> 1}, {4, 4}];

M1yet = SparseArray[{{1, 1} -> -I, {1, 2} -> 1}, {4, 4}];

M1ye2 = SparseArray[{{3, 1} -> I, {4, 1} -> 1}, {4, 4}];

M1ye2t = SparseArray[{{1, 3} -> -I, {1, 4} -> 1}, {4, 4}];

M1yp = M1ye.M1yet + M1ye2.M1ye2t;

MatrixForm[%]


M1ye3 = SparseArray[{{1, 1} -> -I, {2, 1} -> 1}, {4, 4}];

M1ye3t = SparseArray[{{1, 1} -> I, {1, 2} -> 1}, {4, 4}];

M1ye4 = SparseArray[{{3, 1} -> -I, {4, 1} -> 1}, {4, 4}];

M1ye4t = SparseArray[{{1, 3} -> I, {1, 4} -> 1}, {4, 4}];

M1yn = M1ye3.M1ye3t + M1ye4.M1ye4t;

MatrixForm[%]


M1x = {{0, 1, 0, 0}, {1, 0, 0, 0}, {0, 0, 0, 1}, {0, 0, 1, 0}};

Eigenvalues[M1x];

Eigenvectors[M1x]

M1xe = SparseArray[{{1, 1} -> 1, {2, 1} -> 1}, {4, 4}];

M1xet = SparseArray[{{1, 1} -> 1, {1, 2} -> 1}, {4, 4}];

M1xe2 = SparseArray[{{3, 1} -> 1, {4, 1} -> 1}, {4, 4}];

M1xe2t = SparseArray[{{1, 3} -> 1, {1, 4} -> 1}, {4, 4}];

M1xp = M1xe.M1xet + M1xe2.M1xe2t;

MatrixForm[%]
```

```
M1xe3 = SparseArray[{{1, 1} -> -1, {2, 1} -> 1}, {4, 4}];

M1xe3t = SparseArray[{{1, 1} -> -1, {1, 2} -> 1}, {4, 4}];

M1xe4 = SparseArray[{{3, 1} -> -1, {4, 1} -> 1}, {4, 4}];

M1xe4t = SparseArray[{{1, 3} -> -1, {1, 4} -> 1}, {4, 4}];

M1xn = M1xe3.M1xe3t + M1xe4.M1xe4t;

MatrixForm[%]


M1z = {{1, 0, 0, 0}, {0, -1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, -1}};

Eigenvalues[M1z]

Eigenvectors[M1z]

M1ze = SparseArray[{{1, 1} -> 1}, {4, 4}];

M1zet = SparseArray[{{1, 1} -> 1}, {4, 4}];

M1ze2 = SparseArray[{{3, 1} -> 1}, {4, 4}];

M1ze2t = SparseArray[{{1, 3} -> 1}, {4, 4}];

M1zp = M1ze.M1zet + M1ze2.M1ze2t;

MatrixForm[%]


M1ze3 = SparseArray[{{2, 1} -> 1}, {4, 4}];

M1ze3t = Normal[SparseArray[{{1, 2} -> 1}, {4, 4}]];

M1ze4 = SparseArray[{{4, 1} -> 1}, {4, 4}];

M1ze4t = SparseArray[{{1, 4} -> 1}, {4, 4}];

M1zn = M1ze3.M1ze3t + M1ze4.M1ze4t;

MatrixForm[%]
```

Combine the sigma matrices

```
My1mx1 = {{0, 0, (I - 1)/2, 0}, {0, 0, 0, (I - 1)/2}, {(-I - 1)/2, 0,
    0, 0}, {0, (-I - 1)/2, 0, 0}};
Eigenvalues[My1mx1]
Eigenvectors[My1mx1]
My1mx1e =
  SparseArray[{{1, 1} -> -(1 - I)/Sqrt[2], {3, 1} -> 1}, {4, 4}];
My1mx1et =
  SparseArray[{{1, 1} -> -(1 + I)/Sqrt[2], {1, 3} -> 1}, {4, 4}];
My1mx1e2 =
  SparseArray[{{2, 1} -> -(1 - I)/Sqrt[2], {4, 1} -> 1}, {4, 4}];
My1mx1e2t =
  SparseArray[{{1, 2} -> -(1 + I)/Sqrt[2], {1, 4} -> 1}, {4, 4}];
My1mx1p = My1mx1e.My1mx1et + My1mx1e2.My1mx1e2t;
MatrixForm[%]


My1mx1e3 =
  SparseArray[{{1, 1} -> (1 - I)/Sqrt[2], {3, 1} -> 1}, {4, 4}];
My1mx1e3t =
  SparseArray[{{1, 1} -> (1 + I)/Sqrt[2], {1, 3} -> 1}, {4, 4}];
My1mx1e4 =
  SparseArray[{{2, 1} -> (1 - I)/Sqrt[2], {4, 1} -> 1}, {4, 4}];
My1mx1e4t =
  SparseArray[{{1, 2} -> (1 + I)/Sqrt[2], {1, 4} -> 1}, {4, 4}];
My1mx1n = My1mx1e3.My1mx1e3t + My1mx1e4.My1mx1e4t;
MatrixForm[%]
```

```
My1px1 = {{0, 0, (I + 1)/2, 0}, {0, 0, 0, (I + 1)/2}, {(-I + 1)/2, 0,
    0, 0}, {0, (-I + 1)/2, 0, 0}};
Eigenvalues[My1px1]
Eigenvectors[My1px1]
My1px1e =
  SparseArray[{{1, 1} -> (1 + I)/Sqrt[2], {3, 1} -> 1}, {4, 4}];
My1px1et =
  SparseArray[{{1, 1} -> (1 - I)/Sqrt[2], {1, 3} -> 1}, {4, 4}];
My1px1e2 =
  SparseArray[{{2, 1} -> (1 + I)/Sqrt[2], {4, 1} -> 1}, {4, 4}];
My1px1e2t =
  SparseArray[{{1, 2} -> (1 - I)/Sqrt[2], {1, 4} -> 1}, {4, 4}];
My1px1p = My1px1e.My1px1et + My1px1e2.My1px1e2t;
MatrixForm[%]


My1px1e3 =
  SparseArray[{{1, 1} -> -(1 + I)/Sqrt[2], {3, 1} -> 1}, {4, 4}];
My1px1e3t =
  SparseArray[{{1, 1} -> -(1 - I)/Sqrt[2], {1, 3} -> 1}, {4, 4}];
My1px1e4 =
  SparseArray[{{2, 1} -> -(1 + I)/Sqrt[2], {4, 1} -> 1}, {4, 4}];
My1px1e4t =
  SparseArray[{{1, 2} -> -(1 - I)/Sqrt[2], {1, 4} -> 1}, {4, 4}];
My1px1n = My1px1e3.My1px1e3t + My1px1e4.My1px1e4t;
MatrixForm[%]
```

```
My1mx1mz1 = {{-1, 0, (I - 1)/2, 0}, {0, -1,
    0, (I - 1)/2}, {(-I - 1)/2, 0, 1, 0}, {0, (-I - 1)/2, 0, 1}};
Eigenvalues[My1mx1mz1]
Eigenvectors[My1mx1mz1]
My1mx1mz1e =
  SparseArray[{{1, 1} -> (-1/2 + I/2) (-2 + Sqrt[6]), {3, 1} ->
     1}, {4, 4}];
My1mx1mz1et =
  SparseArray[{{1, 1} -> (-1/2 - I/2) (-2 + Sqrt[6]), {1, 3} ->
     1}, {4, 4}];
My1mx1mz1e2 =
  SparseArray[{{2, 1} -> (-1/2 + I/2) (-2 + Sqrt[6]), {4, 1} ->
     1}, {4, 4}];
My1mx1mz1e2t =
  SparseArray[{{1, 2} -> (-1/2 - I/2) (-2 + Sqrt[6]), {1, 4} ->
     1}, {4, 4}];
My1mx1mz1p = My1mx1mz1e.My1mx1mz1et + My1mx1mz1e2.My1mx1mz1e2t;
MatrixForm[%]


My1mx1mz1e3 =
  SparseArray[{{1, 1} -> (1/2 - I/2) (2 + Sqrt[6]), {3, 1} -> 1}, {4,
    4}];
My1mx1mz1e3t =
  SparseArray[{{1, 1} -> (1/2 + I/2) (2 + Sqrt[6]), {1, 3} -> 1}, {4,
    4}];
My1mx1mz1e4 =
  SparseArray[{{2, 1} -> (1/2 - I/2) (2 + Sqrt[6]), {4, 1} -> 1}, {4,
    4}];
My1mx1mz1e4t =
  SparseArray[{{1, 2} -> (1/2 + I/2) (2 + Sqrt[6]), {1, 4} -> 1}, {4,
    4}];
My1mx1mz1n = My1mx1mz1e3.My1mx1mz1e3t + My1mx1mz1e4.My1mx1mz1e4t;
MatrixForm[%]
```

Determine the probabilities

```
pz1p1 = (Abs[phi1t.Mz1p.psi]^2)[[1]][[1]];
pz1p2 = (Abs[phi2t.Mz1p.psi]^2)[[1]][[1]];
pz1p3 = (Abs[phi3t.Mz1p.psi]^2)[[1]][[1]];
pz1p4 = (Abs[phi4t.Mz1p.psi]^2)[[1]][[1]];
pz1n1 = (Abs[phi1t.Mz1n.psi]^2)[[1]][[1]];
pz1n2 = (Abs[phi2t.Mz1n.psi]^2)[[1]][[1]];
pz1n3 = (Abs[phi3t.Mz1n.psi]^2)[[1]][[1]];
pz1n4 = (Abs[phi4t.Mz1n.psi]^2)[[1]][[1]];

pz11sum = pz1p1 + pz1n1;
pz12sum = pz1p2 + pz1n2;
pz13sum = pz1p3 + pz1n3;
pz14sum = pz1p4 + pz1n4;

{pz11p = pz1p1/pz11sum, pz11n = pz1n1/pz11sum}
```

```
{pz12p = pz1p2/pz12sum, pz12n = pz1n2/pz12sum}

{pz13p = pz1p3/pz13sum, pz13n = pz1n3/pz13sum}

{pz14p = pz1p4/pz14sum, pz14n = pz1n4/pz14sum}


py1p1 = (Abs[phi1t.My1p.psi]^2)[[1]][[1]] // Simplify;

py1p2 = (Abs[phi2t.My1p.psi]^2)[[1]][[1]] // Simplify;

py1p3 = (Abs[phi3t.My1p.psi]^2)[[1]][[1]] // Simplify;

py1p4 = (Abs[phi4t.My1p.psi]^2)[[1]][[1]] // Simplify;

py1n1 = (Abs[phi1t.My1n.psi]^2)[[1]][[1]] // Simplify;

py1n2 = (Abs[phi2t.My1n.psi]^2)[[1]][[1]] // Simplify;

py1n3 = (Abs[phi3t.My1n.psi]^2)[[1]][[1]] // Simplify;

py1n4 = (Abs[phi4t.My1n.psi]^2)[[1]][[1]] // Simplify;


py11sum = py1p1 + py1n1;

py12sum = py1p2 + py1n2;

py13sum = py1p3 + py1n3;

py14sum = py1p4 + py1n4;


{py11p = py1p1/py11sum, py11n = py1n1/py11sum}

{py12p = py1p2/py12sum, py12n = py1n2/py12sum}

{py13p = py1p3/py13sum, py13n = py1n3/py13sum}

{py14p = py1p4/py14sum, py14n = py1n4/py14sum}


px1p1 = (Abs[phi1t.Mx1p.psi]^2)[[1]][[1]] // Simplify;

px1p2 = (Abs[phi2t.Mx1p.psi]^2)[[1]][[1]] // Simplify;
```

```
px1p3 = (Abs[phi3t.Mx1p.psi]^2)[[1]][[1]] // Simplify;

px1p4 = (Abs[phi4t.Mx1p.psi]^2)[[1]][[1]] // Simplify;

px1n1 = (Abs[phi1t.Mx1n.psi]^2)[[1]][[1]] // Simplify;

px1n2 = (Abs[phi2t.Mx1n.psi]^2)[[1]][[1]] // Simplify;

px1n3 = (Abs[phi3t.Mx1n.psi]^2)[[1]][[1]] // Simplify;

px1n4 = (Abs[phi4t.Mx1n.psi]^2)[[1]][[1]] // Simplify;


px11sum = px1p1 + px1n1;

px12sum = px1p2 + px1n2;

px13sum = px1p3 + px1n3;

px14sum = px1p4 + px1n4;


{px11p = px1p1/px11sum, px11n = px1n1/px11sum}

{px12p = px1p2/px12sum, px12n = px1n2/px12sum}

{px13p = px1p3/px13sum, px13n = px1n3/px13sum}

{px14p = px1p4/px14sum, px14n = px1n4/px14sum}


p1xp1 = (Abs[phi1t.M1xp.psi]^2)[[1]][[1]];

p1xp2 = Simplify[(Abs[phi2t.M1xp.psi]^2)[[1]][[1]]];

p1xp3 = (Abs[phi3t.M1xp.psi]^2)[[1]][[1]];

p1xp4 = Simplify[(Abs[phi4t.M1xp.psi]^2)[[1]][[1]]];

p1xn1 = Simplify[(Abs[phi1t.M1xn.psi]^2)[[1]][[1]]];

p1xn2 = (Abs[phi2t.M1xn.psi]^2)[[1]][[1]];

p1xn3 = Simplify[(Abs[phi3t.M1xn.psi]^2)[[1]][[1]]]

p1xn4 = (Abs[phi4t.M1xn.psi]^2)[[1]][[1]];
```

```
p1x1sum = p1xp1 + p1xn1;

p1x2sum = p1xp2 + p1xn2;

p1x3sum = p1xp3 + p1xn3;

p1x4sum = p1xp4 + p1xn4;


{p1x1p = p1xp1/p1x1sum, p1x1n = p1xn1/p1x1sum}

{p1x2p = p1xp2/p1x2sum, p1x2n = p1xn2/p1x2sum}

{p1x3p = p1xp3/p1x3sum, p1x3n = p1xn3/p1x3sum}

{p1x4p = p1xp4/p1x4sum, p1x4n = p1xn4/p1x4sum}


p1yp1 = (Abs[phi1t.M1yp.psi]^2)[[1]][[1]] // Simplify

p1yp2 = (Abs[phi2t.M1yp.psi]^2)[[1]][[1]] // Simplify;

p1yp3 = (Abs[phi3t.M1yp.psi]^2)[[1]][[1]] // Simplify;

p1yp4 = (Abs[phi4t.M1yp.psi]^2)[[1]][[1]] // Simplify;

p1yn1 = (Abs[phi1t.M1yn.psi]^2)[[1]][[1]] // Simplify;

p1yn2 = (Abs[phi2t.M1yn.psi]^2)[[1]][[1]] // Simplify

p1yn3 = (Abs[phi3t.M1yn.psi]^2)[[1]][[1]] // Simplify;

p1yn4 = (Abs[phi4t.M1yn.psi]^2)[[1]][[1]] // Simplify;


p1y1sum = p1yp1 + p1yn1; p1y2sum = p1yp2 + p1yn2; p1y3sum =
 p1yp3 + p1yn3; p1y4sum = p1yp4 + p1yn4;


{p1y1p = p1yp1/p1y1sum, p1y1n = p1yn1/p1y1sum}

{p1y2p = p1yp2/p1y2sum, p1y2n = p1yn2/p1y2sum}
```

```
{p1y3p = p1yp3/p1y3sum, p1y3n = p1yn3/p1y3sum}
{p1y4p = p1yp4/p1y4sum, p1y4n = p1yn4/p1y4sum}

p1zp1 = (Abs[phi1t.M1zp.psi]^2)[[1]][[1]];
p1zp2 = (Abs[phi2t.M1zp.psi]^2)[[1]][[1]];
p1zp3 = (Abs[phi3t.M1zp.psi]^2)[[1]][[1]];
p1zp4 = (Abs[phi4t.M1zp.psi]^2)[[1]][[1]];
p1zn1 = (Abs[phi1t.M1zn.psi]^2)[[1]][[1]];
p1zn2 = (Abs[phi2t.M1zn.psi]^2)[[1]][[1]];
p1zn3 = (Abs[phi3t.M1zn.psi]^2)[[1]][[1]];
p1zn4 = (Abs[phi4t.M1zn.psi]^2)[[1]][[1]];

p1z1sum = p1zp1 + p1zn1;
p1z2sum = p1zp2 + p1zn2; p1z3sum = p1zp3 + p1zn3; p1z4sum =
 p1zp4 + p1zn4;

{p1z1p = p1zp1/p1z1sum, p1z1n = p1zn1/p1z1sum}
{p1z2p = p1zp2/p1z2sum, p1z2n = p1zn2/p1z2sum}
{p1z3p = p1zp3/p1z3sum, p1z3n = p1zn3/p1z3sum}
{p1z4p = p1zp4/p1z4sum, p1z4n = p1zn4/p1z4sum}

Mn1 = {{Cos[a], 0, 0, 0}, {0, Cos[a], 0, 0}, {0, 0, -Cos[a], 0}, {0,
    0, 0, -Cos[a]}};
Eigenvalues[Mn1]
Eigenvectors[Mn1]
```

```
Mn1e = SparseArray[{{1, 1} -> 1}, {4, 4}];

Mn1et = SparseArray[{{1, 1} -> 1}, {4, 4}];

Mn1e2 = SparseArray[{{2, 1} -> 1}, {4, 4}];

Mn1e2t = SparseArray[{{1, 2} -> 1}, {4, 4}];

Mn1p = Mn1e.Mn1et + Mn1e2.Mn1e2t;

MatrixForm[%]


Mn1e3 = SparseArray[{{3, 1} -> 1}, {4, 4}];

Mn1e3t = SparseArray[{{1, 3} -> 1}, {4, 4}];

Mn1e4 = SparseArray[{{4, 1} -> 1}, {4, 4}];

Mn1e4t = SparseArray[{{1, 4} -> 1}, {4, 4}];

Mn1n = Mn1e3.Mn1e3t + Mn1e4.Mn1e4t;

MatrixForm[%]


M1n = {{Cos[a], Sin[a] E^-(I b), 0, 0}, {Sin[a] E^(I b), -Cos[a], 0,

    0}, {0, 0, Cos[a], Sin[a] E^-(I b)}, {0, 0,

    Sin[a] E^(I b), -Cos[a]}};

Eigenvalues[M1n]

Eigenvectors[M1n]

M1ne = SparseArray[{{1, 1} -> E^-(I b) (1 + Cos[a]) Csc[a], {2, 1} ->

     1}, {4, 4}];

M1net = SparseArray[{{1, 1} -> E^(I b) (1 + Cos[a]) Csc[a], {1, 2} ->

     1}, {4, 4}];

M1ne2 = SparseArray[{{3, 1} -> E^-(I b) (1 + Cos[a]) Csc[a], {4, 1} ->

      1}, {4, 4}];
```

```
M1ne2t = SparseArray[{{1, 3} -> E^(I b) (1 + Cos[a]) Csc[a], {1, 4} ->
      1}, {4, 4}];
M1ne.M1net + M1ne2.M1ne2t;
M1np = FullSimplify[%];
MatrixForm[%]


M1ne3 = SparseArray[{{1, 1} ->
     E^-(I b) (-1 + Cos[a]) Csc[a], {2, 1} -> 1}, {4, 4}];
M1ne3t = SparseArray[{{1, 1} ->
     E^(I b) (-1 + Cos[a]) Csc[a], {1, 2} -> 1}, {4, 4}];
M1ne4 = SparseArray[{{3, 1} ->
     E^-(I b) (-1 + Cos[a]) Csc[a], {4, 1} -> 1}, {4, 4}];
M1ne4t = SparseArray[{{1, 3} ->
     E^(I b) (-1 + Cos[a]) Csc[a], {1, 4} -> 1}, {4, 4}];
M1ne3.M1ne3t + M1ne4.M1ne4t;
M1nn = FullSimplify[%];
MatrixForm[%]


pn1p1 = (Abs[phi1t.Mn1p.psi]^2)[[1]][[1]];
pn1p2 = (Abs[phi2t.Mn1p.psi]^2)[[1]][[1]];
pn1p3 = (Abs[phi3t.Mn1p.psi]^2)[[1]][[1]];
pn1p4 = (Abs[phi4t.Mn1p.psi]^2)[[1]][[1]];
pn1n1 = (Abs[phi1t.Mn1n.psi]^2)[[1]][[1]];
pn1n2 = (Abs[phi2t.Mn1n.psi]^2)[[1]][[1]];
pn1n3 = (Abs[phi3t.Mn1n.psi]^2)[[1]][[1]];
```

```
pn1n4 = (Abs[phi4t.Mn1n.psi]^2)[[1]][[1]];


pn11sum = pn1p1 + pn1n1;

pn12sum = pn1p2 + pn1n2; pn13sum = pn1p3 + pn1n3; pn14sum =

 pn1p4 + pn1n4;


{pn11p = pn1p1/pn11sum, pn11n = pn1n1/pn11sum}

{pn12p = pn1p2/pn12sum, pn12n = pn1n2/pn12sum}

{pn13p = pn1p3/pn13sum, pn13n = pn1n3/pn13sum}

{pn14p = pn1p4/pn14sum, pn14n = pn1n4/pn14sum}


Mzx = {{0, 1, 0, 0}, {1, 0, 0, 0}, {0, 0, 0, -1}, {0, 0, -1, 0}};

Eigenvalues[Mzx]

Eigenvectors[Mzx]

Mzxe = SparseArray[{{1, 1} -> 1, {2, 1} -> 1}, {4, 4}];

Mzxet = SparseArray[{{1, 1} -> 1, {1, 2} -> 1}, {4, 4}];

Mzxe2 = SparseArray[{{3, 1} -> -1, {4, 1} -> 1}, {4, 4}];

Mzxe2t = SparseArray[{{1, 3} -> -1, {1, 4} -> 1}, {4, 4}];

Mzxp = Mzxe.Mzxet + Mzxe2.Mzxe2t;

MatrixForm[%]


Mzxe3 = SparseArray[{{1, 1} -> -1, {2, 1} -> 1}, {4, 4}];

Mzxe3t = SparseArray[{{1, 1} -> -1, {1, 2} -> 1}, {4, 4}];

Mzxe4 = SparseArray[{{3, 1} -> 1, {4, 1} -> 1}, {4, 4}];

Mzxe4t = SparseArray[{{1, 3} -> 1, {1, 4} -> 1}, {4, 4}];
```

```
Mzxn = Mzxe3.Mzxe3t + Mzxe4.Mzxe4t;
MatrixForm[%]


Mzy = {{0, -I, 0, 0}, {I, 0, 0, 0}, {0, 0, 0, I}, {0, 0, -I, 0}};
Eigenvalues[Mzy]
Eigenvectors[Mzy]
Mzye = SparseArray[{{1, 1} -> -I, {2, 1} -> 1}, {4, 4}];
Mzyet = SparseArray[{{1, 1} -> -I, {1, 2} -> 1}, {4, 4}];
Mzye2 = SparseArray[{{3, 1} -> I, {4, 1} -> 1}, {4, 4}];
Mzye2t = SparseArray[{{1, 3} -> I, {1, 4} -> 1}, {4, 4}];
Mzyp = Mzye.Mzyet + Mzye2.Mzye2t;
MatrixForm[%]


Mzye3 = SparseArray[{{1, 1} -> I, {2, 1} -> 1}, {4, 4}];
Mzye3t = SparseArray[{{1, 1} -> I, {1, 2} -> 1}, {4, 4}];
Mzye4 = SparseArray[{{3, 1} -> -I, {4, 1} -> 1}, {4, 4}];
Mzye4t = SparseArray[{{1, 3} -> -I, {1, 4} -> 1}, {4, 4}];
Mzyn = Mzye3.Mzye3t + Mzye4.Mzye4t;
MatrixForm[%]


pzxp1 = (Abs[phi1t.Mzxp.psi]^2)[[1]][[1]];
pzxp2 = (Abs[phi2t.Mzxp.psi]^2)[[1]][[1]];
pzxp3 = (Abs[phi3t.Mzxp.psi]^2)[[1]][[1]];
pzxp4 = (Abs[phi4t.Mzxp.psi]^2)[[1]][[1]];
pzxn1 = (Abs[phi1t.Mzxn.psi]^2)[[1]][[1]];
```

```
pzxn2 = (Abs[phi2t.Mzxn.psi]^2)[[1]][[1]];

pzxn3 = (Abs[phi3t.Mzxn.psi]^2)[[1]][[1]];

pzxn4 = (Abs[phi4t.Mzxn.psi]^2)[[1]][[1]];


pzx1sum = pzxp1 + pzxn1;

pzx2sum = pzxp2 + pzxn2; pzx3sum = pzxp3 + pzxn3; pzx4sum =

 pzxp4 + pzxn4;


{pzx1p = pzxp1/pzx1sum, pzx1n = pzxn1/pzx1sum}

{pzx2p = pzxp2/pzx2sum, pzx2n = pzxn2/pzx2sum}

{pzx3p = pzxp3/pzx3sum, pzx3n = pzxn3/pzx3sum}

{pzx4p = pzxp4/pzx4sum, pzx4n = pzxn4/pzx4sum}


pzyp1 = (Abs[phi1t.Mzyp.psi]^2)[[1]][[1]] // FullSimplify;

pzyp2 = (Abs[phi2t.Mzyp.psi]^2)[[1]][[1]] // FullSimplify;

pzyp3 = (Abs[phi3t.Mzyp.psi]^2)[[1]][[1]] // FullSimplify;

pzyp4 = (Abs[phi4t.Mzyp.psi]^2)[[1]][[1]] // FullSimplify;

pzyn1 = (Abs[phi1t.Mzyn.psi]^2)[[1]][[1]] // FullSimplify;

pzyn2 = (Abs[phi2t.Mzyn.psi]^2)[[1]][[1]] // FullSimplify;

pzyn3 = (Abs[phi3t.Mzyn.psi]^2)[[1]][[1]] // FullSimplify;

pzyn4 = (Abs[phi4t.Mzyn.psi]^2)[[1]][[1]] // FullSimplify;


pzy1sum = pzyp1 + pzyn1;

pzy2sum = pzyp2 + pzyn2; pzy3sum = pzyp3 + pzyn3; pzy4sum =

 pzyp4 + pzyn4;
```

```
{pzy1p = pzyp1/pzy1sum, pzy1n = pzyn1/pzy1sum}

{pzy2p = pzyp2/pzy2sum, pzy2n = pzyn2/pzy2sum}

{pzy3p = pzyp3/pzy3sum, pzy3n = pzyn3/pzy3sum}

{pzy4p = pzyp4/pzy4sum, pzy4n = pzyn4/pzy4sum}


psin = Sqrt[2]/2 SparseArray[{{1, 1} -> 1, {4, 1} -> -1}, {4, 4}];


pz1p1n = (Abs[phi1t.Mz1p.psin]^2)[[1]][[1]];

pz1p2n = (Abs[phi2t.Mz1p.psin]^2)[[1]][[1]];

pz1p3n = (Abs[phi3t.Mz1p.psin]^2)[[1]][[1]];

pz1p4n = (Abs[phi4t.Mz1p.psin]^2)[[1]][[1]];

pz1n1n = (Abs[phi1t.Mz1n.psin]^2)[[1]][[1]];

pz1n2n = (Abs[phi2t.Mz1n.psin]^2)[[1]][[1]];

pz1n3n = (Abs[phi3t.Mz1n.psin]^2)[[1]][[1]];

pz1n4n = (Abs[phi4t.Mz1n.psin]^2)[[1]][[1]];


pz11nsum = pz1p1n + pz1n1n;

pz12nsum = pz1p2n + pz1n2n;

pz13nsum = pz1p3n + pz1n3n;

pz14nsum = pz1p4n + pz1n4n;


{pz11pn = pz1p1n/pz11nsum, pz11nn = pz1n1n/pz11nsum}

{pz12pn = pz1p2n/pz12nsum, pz12nn = pz1n2n/pz12nsum}

{pz13pn = pz1p3n/pz13nsum, pz13nn = pz1n3n/pz13nsum}
```

```
{pz14pn = pz1p4n/pz14nsum, pz14nn = pz1n4n/pz14nsum}

py1p1n = (Abs[phi1t.My1p.psin]^2)[[1]][[1]] // Simplify;
py1p2n = (Abs[phi2t.My1p.psin]^2)[[1]][[1]] // Simplify;
py1p3n = (Abs[phi3t.My1p.psin]^2)[[1]][[1]] // Simplify;
py1p4n = (Abs[phi4t.My1p.psin]^2)[[1]][[1]] // Simplify;
py1n1n = (Abs[phi1t.My1n.psin]^2)[[1]][[1]] // Simplify;
py1n2n = (Abs[phi2t.My1n.psin]^2)[[1]][[1]] // Simplify;
py1n3n = (Abs[phi3t.My1n.psin]^2)[[1]][[1]] // Simplify;
py1n4n = (Abs[phi4t.My1n.psin]^2)[[1]][[1]] // Simplify;

py11nsum = py1p1n + py1n1n;
py12nsum = py1p2n + py1n2n;
py13nsum = py1p3n + py1n3n;
py14nsum = py1p4n + py1n4n;

{py11pn = py1p1n/py11nsum, py11nn = py1n1n/py11nsum}
{py12pn = py1p2n/py12nsum, py12nn = py1n2n/py12nsum}
{py13pn = py1p3n/py13nsum, py13nn = py1n3n/py13nsum}
{py14pn = py1p4n/py14nsum, py14nn = py1n4n/py14nsum}

px1p1n = (Abs[phi1t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1p2n = (Abs[phi2t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1p3n = (Abs[phi3t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1p4n = (Abs[phi4t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
```

```
px1n1n = (Abs[phi1t.Mx1n.psin]^2)[[1]][[1]] // Simplify;

px1n2n = (Abs[phi2t.Mx1n.psin]^2)[[1]][[1]] // Simplify;

px1n3n = (Abs[phi3t.Mx1n.psin]^2)[[1]][[1]] // Simplify;

px1n4n = (Abs[phi4t.Mx1n.psin]^2)[[1]][[1]] // Simplify;


px11nsum = px1p1n + px1n1n;

px12nsum = px1p2n + px1n2n;

px13nsum = px1p3n + px1n3n;

px14nsum = px1p4n + px1n4n;


{px11pn = px1p1n/px11nsum, px11nn = px1n1n/px11nsum}

{px12pn = px1p2n/px12nsum, px12nn = px1n2n/px12nsum}

{px13pn = px1p3n/px13nsum, px13nn = px1n3n/px13nsum}

{px14pn = px1p4n/px14nsum, px14nn = px1n4n/px14nsum}
```

New post-states for Psi-negative bell state

```
New Post-state


gam1 = SparseArray[{{1, 1} -> Sqrt[2]/2, {2, 1} ->
     I Exp[I Pi/4]/2, {3, 1} -> I Exp[-I Pi/4]/2}, {4, 4}];
gam2 = SparseArray[{{1, 1} ->
     Sqrt[2]/2, {2, 1} -> -I Exp[I Pi/4]/2, {3,
      1} -> -I Exp[-I Pi/4]/2}, {4, 4}];
gam3 = SparseArray[{{4, 1} -> -Sqrt[2]/2, {2, 1} ->
```

```
     I Exp[-I Pi/4]/2, {3, 1} -> I Exp[I Pi/4]/2}, {4, 4}];
gam4 = SparseArray[{{4, 1} -> -Sqrt[2]/2, {2,
      1} -> -I Exp[-I Pi/4]/2, {3, 1} -> -I Exp[I Pi/4]/2}, {4, 4}];
gam1t = SparseArray[{{1, 1} ->
     Sqrt[2]/2, {1, 2} -> -I Exp[-I Pi/4]/2, {1,
      3} -> -I Exp[I Pi/4]/2}, {4, 4}];
gam2t = SparseArray[{{1, 1} -> Sqrt[2]/2, {1, 2} ->
     I Exp[-I Pi/4]/2, {1, 3} -> I Exp[I Pi/4]/2}, {4, 4}];
gam3t = SparseArray[{{1, 4} -> -Sqrt[2]/2, {1,
      2} -> -I Exp[I Pi/4]/2, {1, 3} -> -I Exp[-I Pi/4]/2}, {4, 4}];
gam4t = SparseArray[{{1, 4} -> -Sqrt[2]/2, {1, 2} ->
     I Exp[I Pi/4]/2, {1, 3} -> I Exp[-I Pi/4]/2}, {4, 4}];

gam1 // MatrixForm
gam2 // MatrixForm
gam3 // MatrixForm
gam4 // MatrixForm

pz1p1n = (Abs[gam1t.Mz1p.psin]^2)[[1]][[1]];
pz1p2n = (Abs[gam2t.Mz1p.psin]^2)[[1]][[1]];
pz1p3n = (Abs[gam3t.Mz1p.psin]^2)[[1]][[1]];
pz1p4n = (Abs[gam4t.Mz1p.psin]^2)[[1]][[1]];
pz1n1n = (Abs[gam1t.Mz1n.psin]^2)[[1]][[1]];
pz1n2n = (Abs[gam2t.Mz1n.psin]^2)[[1]][[1]];
pz1n3n = (Abs[gam3t.Mz1n.psin]^2)[[1]][[1]];
```

```
pz1n4n = (Abs[gam4t.Mz1n.psin]^2)[[1]][[1]];


pz11nsum = pz1p1n + pz1n1n;

pz12nsum = pz1p2n + pz1n2n;

pz13nsum = pz1p3n + pz1n3n;

pz14nsum = pz1p4n + pz1n4n;


{pz11pn = pz1p1n/pz11nsum, pz11nn = pz1n1n/pz11nsum}

{pz12pn = pz1p2n/pz12nsum, pz12nn = pz1n2n/pz12nsum}

{pz13pn = pz1p3n/pz13nsum, pz13nn = pz1n3n/pz13nsum}

{pz14pn = pz1p4n/pz14nsum, pz14nn = pz1n4n/pz14nsum}


py1p1n = (Abs[gam1t.My1p.psin]^2)[[1]][[1]] // Simplify;

py1p2n = (Abs[gam2t.My1p.psin]^2)[[1]][[1]] // Simplify;

py1p3n = (Abs[gam3t.My1p.psin]^2)[[1]][[1]] // Simplify;

py1p4n = (Abs[gam4t.My1p.psin]^2)[[1]][[1]] // Simplify;

py1n1n = (Abs[gam1t.My1n.psin]^2)[[1]][[1]] // Simplify;

py1n2n = (Abs[gam2t.My1n.psin]^2)[[1]][[1]] // Simplify;

py1n3n = (Abs[gam3t.My1n.psin]^2)[[1]][[1]] // Simplify;

py1n4n = (Abs[gam4t.My1n.psin]^2)[[1]][[1]] // Simplify;


py11nsum = py1p1n + py1n1n;

py12nsum = py1p2n + py1n2n;

py13nsum = py1p3n + py1n3n;

py14nsum = py1p4n + py1n4n;
```

```
{py11pn = py1p1n/py11nsum, py11nn = py1n1n/py11nsum}
{py12pn = py1p2n/py12nsum, py12nn = py1n2n/py12nsum}
{py13pn = py1p3n/py13nsum, py13nn = py1n3n/py13nsum}
{py14pn = py1p4n/py14nsum, py14nn = py1n4n/py14nsum}

px1p1n = (Abs[gam1t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1p2n = (Abs[gam2t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1p3n = (Abs[gam3t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1p4n = (Abs[gam4t.Mx1p.psin]^2)[[1]][[1]] // Simplify;
px1n1n = (Abs[gam1t.Mx1n.psin]^2)[[1]][[1]] // Simplify;
px1n2n = (Abs[gam2t.Mx1n.psin]^2)[[1]][[1]] // Simplify;
px1n3n = (Abs[gam3t.Mx1n.psin]^2)[[1]][[1]] // Simplify;
px1n4n = (Abs[gam4t.Mx1n.psin]^2)[[1]][[1]] // Simplify;

px11nsum = px1p1n + px1n1n;
px12nsum = px1p2n + px1n2n;
px13nsum = px1p3n + px1n3n;
px14nsum = px1p4n + px1n4n;

{px11pn = px1p1n/px11nsum, px11nn = px1n1n/px11nsum}
{px12pn = px1p2n/px12nsum, px12nn = px1n2n/px12nsum}
{px13pn = px1p3n/px13nsum, px13nn = px1n3n/px13nsum}
{px14pn = px1p4n/px14nsum, px14nn = px1n4n/px14nsum}
```

```
p1xp1n = (Abs[gam1t.M1xp.psin]^2)[[1]][[1]] // FullSimplify;

p1xp2n = (Abs[gam2t.M1xp.psin]^2)[[1]][[1]] // FullSimplify;

p1xp3n = (Abs[gam3t.M1xp.psin]^2)[[1]][[1]] // FullSimplify;

p1xp4n = FullSimplify[(Abs[gam4t.M1xp.psin]^2)[[1]][[1]]];

p1xn1n = FullSimplify[(Abs[gam1t.M1xn.psin]^2)[[1]][[1]]];

p1xn2n = (Abs[gam2t.M1xn.psin]^2)[[1]][[1]] // FullSimplify;

p1xn3n = FullSimplify[(Abs[gam3t.M1xn.psin]^2)[[1]][[1]]];

p1xn4n = (Abs[gam4t.M1xn.psin]^2)[[1]][[1]] // FullSimplify;


p1x1nsum = p1xp1n + p1xn1n;

p1x2nsum = p1xp2n + p1xn2n;

p1x3nsum = p1xp3n + p1xn3n;

p1x4nsum = p1xp4n + p1xn4n;


{p1x1pn = p1xp1n/p1x1nsum, p1x1nn = p1xn1n/p1x1nsum}

{p1x2pn = p1xp2n/p1x2nsum, p1x2nn = p1xn2n/p1x2nsum}

{p1x3pn = p1xp3n/p1x3nsum, p1x3nn = p1xn3n/p1x3nsum}

{p1x4pn = p1xp4n/p1x4nsum, p1x4nn = p1xn4n/p1x4nsum}


p1yp1n = (Abs[gam1t.M1yp.psin]^2)[[1]][[1]] // Simplify;

p1yp2n = (Abs[gam2t.M1yp.psin]^2)[[1]][[1]] // Simplify;

p1yp3n = (Abs[gam3t.M1yp.psin]^2)[[1]][[1]] // Simplify;

p1yp4n = (Abs[gam4t.M1yp.psin]^2)[[1]][[1]] // Simplify;

p1yn1n = (Abs[gam1t.M1yn.psin]^2)[[1]][[1]] // Simplify;

p1yn2n = (Abs[gam2t.M1yn.psin]^2)[[1]][[1]] // Simplify;
```

```
p1yn3n = (Abs[gam3t.M1yn.psin]^2)[[1]][[1]] // Simplify;
p1yn4n = (Abs[gam4t.M1yn.psin]^2)[[1]][[1]] // Simplify;

p1y1nsum = p1yp1n + p1yn1n; p1y2nsum = p1yp2n + p1yn2n; p1y3nsum =
 p1yp3n + p1yn3n; p1y4nsum = p1yp4n + p1yn4n;

{p1y1pn = p1yp1n/p1y1nsum, p1y1nn = p1yn1n/p1y1nsum}
{p1y2pn = p1yp2n/p1y2nsum, p1y2nn = p1yn2n/p1y2nsum}
{p1y3pn = p1yp3n/p1y3nsum, p1y3nn = p1yn3n/p1y3nsum}
{p1y4pn = p1yp4n/p1y4nsum, p1y4nn = p1yn4n/p1y4nsum}

p1zp1n = (Abs[gam1t.M1zp.psin]^2)[[1]][[1]];
p1zp2n = (Abs[gam2t.M1zp.psin]^2)[[1]][[1]];
p1zp3n = (Abs[gam3t.M1zp.psin]^2)[[1]][[1]];
p1zp4n = (Abs[gam4t.M1zp.psin]^2)[[1]][[1]];
p1zn1n = (Abs[gam1t.M1zn.psin]^2)[[1]][[1]];
p1zn2n = (Abs[gam2t.M1zn.psin]^2)[[1]][[1]];
p1zn3n = (Abs[gam3t.M1zn.psin]^2)[[1]][[1]];
p1zn4n = (Abs[gam4t.M1zn.psin]^2)[[1]][[1]];

p1z1nsum = p1zp1n + p1zn1n;
p1z2nsum = p1zp2n + p1zn2n; p1z3nsum = p1zp3n + p1zn3n; p1z4nsum =
 p1zp4n + p1zn4n;

{p1z1pn = p1zp1n/p1z1nsum, p1z1nn = p1zn1n/p1z1nsum}
```

```
{p1z2pn = p1zp2n/p1z2nsum, p1z2nn = p1zn2n/p1z2nsum}
{p1z3pn = p1zp3n/p1z3nsum, p1z3nn = p1zn3n/p1z3nsum}
{p1z4pn = p1zp4n/p1z4nsum, p1z4nn = p1zn4n/p1z4nsum}
```

Phi positive Bell State

```
Phi positive Bell State

phip = Sqrt[2]/2 SparseArray[{{2, 1} -> 1, {3, 1} -> 1}, {4, 4}];
del1 = SparseArray[{{2, 1} -> Sqrt[2]/2, {1, 1} ->
     Exp[I Pi/4]/2, {4, 1} -> Exp[-I Pi/4]/2}, {4, 4}];
del2 = SparseArray[{{2, 1} ->
     Sqrt[2]/2, {1, 1} -> -Exp[I Pi/4]/2, {4, 1} -> -Exp[-I Pi/4]/
      2}, {4, 4}];
del3 = SparseArray[{{3, 1} -> Sqrt[2]/2, {1, 1} ->
     Exp[-I Pi/4]/2, {4, 1} -> Exp[I Pi/4]/2}, {4, 4}];
del4 = SparseArray[{{3, 1} ->
     Sqrt[2]/2, {1, 1} -> -Exp[-I Pi/4]/2, {4, 1} -> -Exp[I Pi/4]/
      2}, {4, 4}];
del1t = SparseArray[{{1, 2} -> Sqrt[2]/2, {1, 1} ->
     Exp[-I Pi/4]/2, {1, 4} -> Exp[I Pi/4]/2}, {4, 4}];
del2t = SparseArray[{{1, 2} ->
     Sqrt[2]/2, {1, 1} -> -Exp[-I Pi/4]/2, {1, 4} -> -Exp[I Pi/4]/
      2}, {4, 4}];
del3t = SparseArray[{{1, 3} -> Sqrt[2]/2, {1, 1} ->
```

```
     Exp[I Pi/4]/2, {1, 4} -> Exp[-I Pi/4]/2}, {4, 4}];
del4t = SparseArray[{{1, 3} ->
     Sqrt[2]/2, {1, 1} -> -Exp[I Pi/4]/2, {1, 4} -> -Exp[-I Pi/4]/
      2}, {4, 4}];

del1 // MatrixForm
del2 // MatrixForm
del3 // MatrixForm
del4 // MatrixForm

ppz1p1 = (Abs[del1t.Mz1p.phip]^2)[[1]][[1]];
ppz1p2 = (Abs[del2t.Mz1p.phip]^2)[[1]][[1]];
ppz1p3 = (Abs[del3t.Mz1p.phip]^2)[[1]][[1]];
ppz1p4 = (Abs[del4t.Mz1p.phip]^2)[[1]][[1]];
ppz1n1 = (Abs[del1t.Mz1n.phip]^2)[[1]][[1]];
ppz1n2 = (Abs[del2t.Mz1n.phip]^2)[[1]][[1]];
ppz1n3 = (Abs[del3t.Mz1n.phip]^2)[[1]][[1]];
ppz1n4 = (Abs[del4t.Mz1n.phip]^2)[[1]][[1]];

ppz11sum = ppz1p1 + ppz1n1;
ppz12sum = ppz1p2 + ppz1n2;
ppz13sum = ppz1p3 + ppz1n3;
ppz14sum = ppz1p4 + ppz1n4;

{ppz11p = ppz1p1/ppz11sum, ppz11n = ppz1n1/ppz11sum}
```

```
{ppz12p = ppz1p2/ppz12sum, ppz12n = ppz1n2/ppz12sum}

{ppz13p = ppz1p3/ppz13sum, ppz13n = ppz1n3/ppz13sum}

{ppz14p = ppz1p4/ppz14sum, ppz14n = ppz1n4/ppz14sum}


ppy1p1 = (Abs[del1t.My1p.phip]^2)[[1]][[1]] // FullSimplify;

ppy1p2 = (Abs[del2t.My1p.phip]^2)[[1]][[1]] // FullSimplify;

ppy1p3 = (Abs[del3t.My1p.phip]^2)[[1]][[1]] // FullSimplify;

ppy1p4 = (Abs[del4t.My1p.phip]^2)[[1]][[1]] // FullSimplify;

ppy1n1 = (Abs[del1t.My1n.phip]^2)[[1]][[1]] // FullSimplify;

ppy1n2 = (Abs[del2t.My1n.phip]^2)[[1]][[1]] // FullSimplify;

ppy1n3 = (Abs[del3t.My1n.phip]^2)[[1]][[1]] // FullSimplify;

ppy1n4 = (Abs[del4t.My1n.phip]^2)[[1]][[1]] // FullSimplify;


ppy11sum = ppy1p1 + ppy1n1;

ppy12sum = ppy1p2 + ppy1n2;

ppy13sum = ppy1p3 + ppy1n3;

ppy14sum = ppy1p4 + ppy1n4;


{ppy11p = ppy1p1/ppy11sum, ppy11n = ppy1n1/ppy11sum}

{ppy12p = ppy1p2/ppy12sum, ppy12n = ppy1n2/ppy12sum}

{ppy13p = ppy1p3/ppy13sum, ppy13n = ppy1n3/ppy13sum}

{ppy14p = ppy1p4/ppy14sum, ppy14n = ppy1n4/ppy14sum}


ppx1p1 = (Abs[del1t.Mx1p.phip]^2)[[1]][[1]] // FullSimplify;

ppx1p2 = (Abs[del2t.Mx1p.phip]^2)[[1]][[1]] // FullSimplify;
```

```
ppx1p3 = (Abs[del3t.Mx1p.phip]^2)[[1]][[1]] // FullSimplify;

ppx1p4 = (Abs[del4t.Mx1p.phip]^2)[[1]][[1]] // FullSimplify;

ppx1n1 = (Abs[del1t.Mx1n.phip]^2)[[1]][[1]] // FullSimplify;

ppx1n2 = (Abs[del2t.Mx1n.phip]^2)[[1]][[1]] // FullSimplify;

ppx1n3 = (Abs[del3t.Mx1n.phip]^2)[[1]][[1]] // FullSimplify;

ppx1n4 = (Abs[del4t.Mx1n.phip]^2)[[1]][[1]] // FullSimplify;


ppx11sum = ppx1p1 + ppx1n1;

ppx12sum = ppx1p2 + ppx1n2;

ppx13sum = ppx1p3 + ppx1n3;

ppx14sum = ppx1p4 + ppx1n4;


{ppx11p = ppx1p1/ppx11sum, ppx11n = ppx1n1/ppx11sum}

{ppx12p = ppx1p2/ppx12sum, ppx12n = ppx1n2/ppx12sum}

{ppx13p = ppx1p3/ppx13sum, ppx13n = ppx1n3/ppx13sum}

{ppx14p = ppx1p4/ppx14sum, ppx14n = ppx1n4/ppx14sum}


pp1zp1 = (Abs[del1t.M1zp.phip]^2)[[1]][[1]];

pp1zp2 = (Abs[del2t.M1zp.phip]^2)[[1]][[1]];

pp1zp3 = (Abs[del3t.M1zp.phip]^2)[[1]][[1]];

pp1zp4 = (Abs[del4t.M1zp.phip]^2)[[1]][[1]];

pp1zn1 = (Abs[del1t.M1zn.phip]^2)[[1]][[1]];

pp1zn2 = (Abs[del2t.M1zn.phip]^2)[[1]][[1]];

pp1zn3 = (Abs[del3t.M1zn.phip]^2)[[1]][[1]];

pp1zn4 = (Abs[del4t.M1zn.phip]^2)[[1]][[1]];
```

```
pp1z1sum = pp1zp1 + pp1zn1;
pp1z2sum = pp1zp2 + pp1zn2;
pp1z3sum = pp1zp3 + pp1zn3;
pp1z4sum = pp1zp4 + pp1zn4;

{pp1z1p = pp1zp1/pp1z1sum, pp1z1n = pp1zn1/pp1z1sum}
{pp1z2p = pp1zp2/pp1z2sum, pp1z2n = pp1zn2/pp1z2sum}
{pp1z3p = pp1zp3/pp1z3sum, pp1z3n = pp1zn3/pp1z3sum}
{pp1z4p = pp1zp4/pp1z4sum, pp1z4n = pp1zn4/pp1z4sum}

pp1yp1 = (Abs[del1t.M1yp.phip]^2)[[1]][[1]] // FullSimplify;
pp1yp2 = (Abs[del2t.M1yp.phip]^2)[[1]][[1]] // FullSimplify;
pp1yp3 = (Abs[del3t.M1yp.phip]^2)[[1]][[1]] // FullSimplify;
pp1yp4 = (Abs[del4t.M1yp.phip]^2)[[1]][[1]] // FullSimplify;
pp1yn1 = (Abs[del1t.M1yn.phip]^2)[[1]][[1]] // FullSimplify;
pp1yn2 = (Abs[del2t.M1yn.phip]^2)[[1]][[1]] // FullSimplify;
pp1yn3 = (Abs[del3t.M1yn.phip]^2)[[1]][[1]] // FullSimplify;
pp1yn4 = (Abs[del4t.M1yn.phip]^2)[[1]][[1]] // FullSimplify;

pp1y1sum = pp1yp1 + pp1yn1;
pp1y2sum = pp1yp2 + pp1yn2;
pp1y3sum = pp1yp3 + pp1yn3;
pp1y4sum = pp1yp4 + pp1yn4;
```

```
{pp1y1p = pp1yp1/pp1y1sum, pp1y1n = pp1yn1/pp1y1sum}
{pp1y2p = pp1yp2/pp1y2sum, pp1y2n = pp1yn2/pp1y2sum}
{pp1y3p = pp1yp3/pp1y3sum, pp1y3n = pp1yn3/pp1y3sum}
{pp1y4p = pp1yp4/pp1y4sum, pp1y4n = pp1yn4/pp1y4sum}

pp1xp1 = (Abs[del1t.M1xp.phip]^2)[[1]][[1]] // FullSimplify;
pp1xp2 = (Abs[del2t.M1xp.phip]^2)[[1]][[1]] // FullSimplify;
pp1xp3 = (Abs[del3t.M1xp.phip]^2)[[1]][[1]] // FullSimplify;
pp1xp4 = (Abs[del4t.M1xp.phip]^2)[[1]][[1]] // FullSimplify;
pp1xn1 = (Abs[del1t.M1xn.phip]^2)[[1]][[1]] // FullSimplify;
pp1xn2 = (Abs[del2t.M1xn.phip]^2)[[1]][[1]] // FullSimplify;
pp1xn3 = (Abs[del3t.M1xn.phip]^2)[[1]][[1]] // FullSimplify;
pp1xn4 = (Abs[del4t.M1xn.phip]^2)[[1]][[1]] // FullSimplify;

pp1x1sum = pp1xp1 + pp1xn1;
pp1x2sum = pp1xp2 + pp1xn2;
pp1x3sum = pp1xp3 + pp1xn3;
pp1x4sum = pp1xp4 + pp1xn4;

{pp1x1p = pp1xp1/pp1x1sum, pp1x1n = pp1xn1/pp1x1sum}
{pp1x2p = pp1xp2/pp1x2sum, pp1x2n = pp1xn2/pp1x2sum}
{pp1x3p = pp1xp3/pp1x3sum, pp1x3n = pp1xn3/pp1x3sum}
{pp1x4p = pp1xp4/pp1x4sum, pp1x4n = pp1xn4/pp1x4sum}
```

Phi negative Bell State

Phi negative Bell state

```
phin = Sqrt[2]/2 SparseArray[{{2, 1} -> 1, {3, 1} -> -1}, {4, 4}];
xi1 = SparseArray[{{2, 1} -> Sqrt[2]/2, {1, 1} ->
     I Exp[I Pi/4]/2, {4, 1} -> I Exp[-I Pi/4]/2}, {4, 4}];
xi2 = SparseArray[{{2, 1} ->
     Sqrt[2]/2, {1, 1} -> -I Exp[I Pi/4]/2, {4,
      1} -> -I Exp[-I Pi/4]/2}, {4, 4}];
xi3 = SparseArray[{{3, 1} -> -Sqrt[2]/2, {1, 1} ->
     I Exp[-I Pi/4]/2, {4, 1} -> I Exp[I Pi/4]/2}, {4, 4}];
xi4 = SparseArray[{{3, 1} -> -Sqrt[2]/2, {1,
      1} -> -I Exp[-I Pi/4]/2, {4, 1} -> -I Exp[I Pi/4]/2}, {4, 4}];
xi1t = SparseArray[{{1, 2} ->
     Sqrt[2]/2, {1, 1} -> -I Exp[-I Pi/4]/2, {1,
      4} -> -I Exp[I Pi/4]/2}, {4, 4}];
xi2t = SparseArray[{{1, 2} -> Sqrt[2]/2, {1, 1} ->
     I Exp[-I Pi/4]/2, {1, 4} -> I Exp[I Pi/4]/2}, {4, 4}];
xi3t = SparseArray[{{1, 3} -> -Sqrt[2]/2, {1,
      1} -> -I Exp[I Pi/4]/2, {1, 4} -> -I Exp[-I Pi/4]/2}, {4, 4}];
xi4t = SparseArray[{{1, 3} -> -Sqrt[2]/2, {1, 1} ->
     I Exp[I Pi/4]/2, {1, 4} -> I Exp[-I Pi/4]/2}, {4, 4}];

xi1 // MatrixForm
xi2 // MatrixForm
```

```
xi3 // MatrixForm
xi4 // MatrixForm

pnz1p1 = (Abs[xi1t.Mz1p.phin]^2)[[1]][[1]];
pnz1p2 = (Abs[xi2t.Mz1p.phin]^2)[[1]][[1]];
pnz1p3 = (Abs[xi3t.Mz1p.phin]^2)[[1]][[1]];
pnz1p4 = (Abs[xi4t.Mz1p.phin]^2)[[1]][[1]];
pnz1n1 = (Abs[xi1t.Mz1n.phin]^2)[[1]][[1]];
pnz1n2 = (Abs[xi2t.Mz1n.phin]^2)[[1]][[1]];
pnz1n3 = (Abs[xi3t.Mz1n.phin]^2)[[1]][[1]];
pnz1n4 = (Abs[xi4t.Mz1n.phin]^2)[[1]][[1]];

pnz11sum = pnz1p1 + pnz1n1;
pnz12sum = pnz1p2 + pnz1n2;
pnz13sum = pnz1p3 + pnz1n3;
pnz14sum = pnz1p4 + pnz1n4;

{pnz11p = pnz1p1/pnz11sum, pnz11n = pnz1n1/pnz11sum}
{pnz12p = pnz1p2/pnz12sum, pnz12n = pnz1n2/pnz12sum}
{pnz13p = pnz1p3/pnz13sum, pnz13n = pnz1n3/pnz13sum}
{pnz14p = pnz1p4/pnz14sum, pnz14n = pnz1n4/pnz14sum}

pnx1p1 = (Abs[xi1t.Mx1p.phin]^2)[[1]][[1]] // FullSimplify;
pnx1p2 = (Abs[xi2t.Mx1p.phin]^2)[[1]][[1]] // FullSimplify;
pnx1p3 = (Abs[xi3t.Mx1p.phin]^2)[[1]][[1]] // FullSimplify;
```

```
pnx1p4 = (Abs[xi4t.Mx1p.phin]^2)[[1]][[1]] // FullSimplify;

pnx1n1 = (Abs[xi1t.Mx1n.phin]^2)[[1]][[1]] // FullSimplify;

pnx1n2 = (Abs[xi2t.Mx1n.phin]^2)[[1]][[1]] // FullSimplify;

pnx1n3 = (Abs[xi3t.Mx1n.phin]^2)[[1]][[1]] // FullSimplify;

pnx1n4 = (Abs[xi4t.Mx1n.phin]^2)[[1]][[1]] // FullSimplify;


pnx11sum = pnx1p1 + pnx1n1;

pnx12sum = pnx1p2 + pnx1n2;

pnx13sum = pnx1p3 + pnx1n3;

pnx14sum = pnx1p4 + pnx1n4;


{pnx11p = pnx1p1/pnx11sum, pnx11n = pnx1n1/pnx11sum}

{pnx12p = pnx1p2/pnx12sum, pnx12n = pnx1n2/pnx12sum}

{pnx13p = pnx1p3/pnx13sum, pnx13n = pnx1n3/pnx13sum}

{pnx14p = pnx1p4/pnx14sum, pnx14n = pnx1n4/pnx14sum}


pny1p1 = (Abs[xi1t.My1p.phin]^2)[[1]][[1]] // FullSimplify;

pny1p2 = (Abs[xi2t.My1p.phin]^2)[[1]][[1]] // FullSimplify;

pny1p3 = (Abs[xi3t.My1p.phin]^2)[[1]][[1]] // FullSimplify;

pny1p4 = (Abs[xi4t.My1p.phin]^2)[[1]][[1]] // FullSimplify;

pny1n1 = (Abs[xi1t.My1n.phin]^2)[[1]][[1]] // FullSimplify;

pny1n2 = (Abs[xi2t.My1n.phin]^2)[[1]][[1]] // FullSimplify;

pny1n3 = (Abs[xi3t.My1n.phin]^2)[[1]][[1]] // FullSimplify;

pny1n4 = (Abs[xi4t.My1n.phin]^2)[[1]][[1]] // FullSimplify;
```

```
pny11sum = pny1p1 + pny1n1;
pny12sum = pny1p2 + pny1n2;
pny13sum = pny1p3 + pny1n3;
pny14sum = pny1p4 + pny1n4;

{pny11p = pny1p1/pny11sum, pny11n = pny1n1/pny11sum}
{pny12p = pny1p2/pny12sum, pny12n = pny1n2/pny12sum}
{pny13p = pny1p3/pny13sum, pny13n = pny1n3/pny13sum}
{pny14p = pny1p4/pny14sum, pny14n = pny1n4/pny14sum}

pn1zp1 = (Abs[xi1t.M1zp.phin]^2)[[1]][[1]] // FullSimplify;
pn1zp2 = (Abs[xi2t.M1zp.phin]^2)[[1]][[1]] // FullSimplify;
pn1zp3 = (Abs[xi3t.M1zp.phin]^2)[[1]][[1]] // FullSimplify;
pn1zp4 = (Abs[xi4t.M1zp.phin]^2)[[1]][[1]] // FullSimplify;
pn1zn1 = (Abs[xi1t.M1zn.phin]^2)[[1]][[1]] // FullSimplify;
pn1zn2 = (Abs[xi2t.M1zn.phin]^2)[[1]][[1]] // FullSimplify;
pn1zn3 = (Abs[xi3t.M1zn.phin]^2)[[1]][[1]] // FullSimplify;
pn1zn4 = (Abs[xi4t.M1zn.phin]^2)[[1]][[1]] // FullSimplify;

pn1z1sum = pn1zp1 + pn1zn1;
pn1z2sum = pn1zp2 + pn1zn2;
pn1z3sum = pn1zp3 + pn1zn3;
pn1z4sum = pn1zp4 + pn1zn4;

{pn1z1p = pn1zp1/pn1z1sum, pn1z1n = pn1zn1/pn1z1sum}
```

```
{pn1z2p = pn1zp2/pn1z2sum, pn1z2n = pn1zn2/pn1z2sum}
{pn1z3p = pn1zp3/pn1z3sum, pn1z3n = pn1zn3/pn1z3sum}
{pn1z4p = pn1zp4/pn1z4sum, pn1z4n = pn1zn4/pn1z4sum}

pn1xp1 = (Abs[xi1t.M1xp.phin]^2)[[1]][[1]] // FullSimplify;
pn1xp2 = (Abs[xi2t.M1xp.phin]^2)[[1]][[1]] // FullSimplify;
pn1xp3 = (Abs[xi3t.M1xp.phin]^2)[[1]][[1]] // FullSimplify;
pn1xp4 = (Abs[xi4t.M1xp.phin]^2)[[1]][[1]] // FullSimplify;
pn1xn1 = (Abs[xi1t.M1xn.phin]^2)[[1]][[1]] // FullSimplify;
pn1xn2 = (Abs[xi2t.M1xn.phin]^2)[[1]][[1]] // FullSimplify;
pn1xn3 = (Abs[xi3t.M1xn.phin]^2)[[1]][[1]] // FullSimplify;
pn1xn4 = (Abs[xi4t.M1xn.phin]^2)[[1]][[1]] // FullSimplify;

pn1x1sum = pn1xp1 + pn1xn1;
pn1x2sum = pn1xp2 + pn1xn2;
pn1x3sum = pn1xp3 + pn1xn3;
pn1x4sum = pn1xp4 + pn1xn4;

{pn1x1p = pn1xp1/pn1x1sum, pn1x1n = pn1xn1/pn1x1sum}
{pn1x2p = pn1xp2/pn1x2sum, pn1x2n = pn1xn2/pn1x2sum}
{pn1x3p = pn1xp3/pn1x3sum, pn1x3n = pn1xn3/pn1x3sum}
{pn1x4p = pn1xp4/pn1x4sum, pn1x4n = pn1xn4/pn1x4sum}

pn1yp1 = (Abs[xi1t.M1yp.phin]^2)[[1]][[1]] // FullSimplify;
pn1yp2 = (Abs[xi2t.M1yp.phin]^2)[[1]][[1]] // FullSimplify;
```

```
pn1yp3 = (Abs[xi3t.M1yp.phin]^2)[[1]][[1]] // FullSimplify;

pn1yp4 = (Abs[xi4t.M1yp.phin]^2)[[1]][[1]] // FullSimplify;

pn1yn1 = (Abs[xi1t.M1yn.phin]^2)[[1]][[1]] // FullSimplify;

pn1yn2 = (Abs[xi2t.M1yn.phin]^2)[[1]][[1]] // FullSimplify;

pn1yn3 = (Abs[xi3t.M1yn.phin]^2)[[1]][[1]] // FullSimplify;

pn1yn4 = (Abs[xi4t.M1yn.phin]^2)[[1]][[1]] // FullSimplify;


pn1y1sum = pn1yp1 + pn1yn1;

pn1y2sum = pn1yp2 + pn1yn2;

pn1y3sum = pn1yp3 + pn1yn3;

pn1y4sum = pn1yp4 + pn1yn4;


{pn1y1p = pn1yp1/pn1y1sum, pn1y1n = pn1yn1/pn1y1sum}

{pn1y2p = pn1yp2/pn1y2sum, pn1y2n = pn1yn2/pn1y2sum}

{pn1y3p = pn1yp3/pn1y3sum, pn1y3n = pn1yn3/pn1y3sum}

{pn1y4p = pn1yp4/pn1y4sum, pn1y4n = pn1yn4/pn1y4sum}
```

Creating partially entangled states

```
Partially Entangled state (Check with values of epsilon just removed from 1/Sqrt[2])


Clear[\[Epsilon]]

Clear[\[Gamma]]

\[Epsilon] = Sqrt[\[Gamma]];

(*pes= SparseArray[{{1,1}\[Rule]Sqrt[1-\[Epsilon]^2],{4,1}\[Rule]\
\[Epsilon]},{4,4}];
k=Sqrt[\[Epsilon]^2+2 \[Epsilon]^2 Sqrt[1-\[Epsilon]^2]];
lam1=SparseArray[{{1,1}\[Rule]\[Epsilon]/k \
Sqrt[2]/2,{2,1}\[Rule](Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[I Pi/4]/2,{3,1}\[Rule](Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[-I Pi/4]/2},{4,4}];
lam2=SparseArray[{{1,1}\[Rule]\[Epsilon]/k \
Sqrt[2]/2,{2,1}\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[I Pi/4]/2,{3,1}\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[-I Pi/4]/2},{4,4}];
lam3=SparseArray[{{4,1}\[Rule]Sqrt[1-\[Epsilon]^2]/k Sqrt[2]/2,{2,1}\
\[Rule](Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[-I \
Pi/4]/2,{3,1}->(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[I \
Pi/4]/2},{4,4}];
lam4=SparseArray[{{4,1}\[Rule]Sqrt[1-\[Epsilon]^2]/k Sqrt[2]/2,{2,1}\
\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[-I \
Pi/4]/2,{3,1}\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[I \
Pi/4]/2},{4,4}];
lam1t=SparseArray[{{1,1}\[Rule]\[Epsilon]/k \
Sqrt[2]/2,{1,2}\[Rule](Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[-I Pi/4]/2,{1,3}->(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[I Pi/4]/2},{4,4}];
lam2t=SparseArray[{{1,1}\[Rule]\[Epsilon]/k \
Sqrt[2]/2,{1,2}\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[-I Pi/4]/2,{1,3}\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/\
k Exp[I Pi/4]/2},{4,4}];
lam3t=SparseArray[{{1,4}\[Rule]Sqrt[1-\[Epsilon]^2]/k Sqrt[2]/2,{1,2}\
\[Rule](Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[I \
Pi/4]/2,{1,3}->(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[-I \
Pi/4]/2},{4,4}];
lam4t=SparseArray[{{1,4}\[Rule]Sqrt[1-\[Epsilon]^2]/k Sqrt[2]/2,{1,2}\
\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k Exp[I \
Pi/4]/2,{1,3}\[Rule]-(Sqrt[\[Epsilon]](1-\[Epsilon]^2)^(1/4))/k \
Exp[-I Pi/4]/2},{4,4}];*)


pes = SparseArray[{{1, 1} -> \[Epsilon], {4, 1} ->
     Sqrt[1 - \[Epsilon]^2]}, {4, 4}];
k = Sqrt[1 - \[Epsilon]^2 + 2 (1 - \[Epsilon]^2) \[Epsilon]];
lam1 = SparseArray[{{1, 1} ->
     Sqrt[1 - \[Epsilon]^2]/k Sqrt[2]/2, {2,
      1} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/k Exp[I Pi/4]/
       2, {3, 1} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2}, {4, 4}];
lam2 = SparseArray[{{1, 1} ->
     Sqrt[1 - \[Epsilon]^2]/k Sqrt[2]/2, {2,
      1} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[I Pi/4]/2, {3,
      1} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2}, {4, 4}];
lam3 = SparseArray[{{4, 1} -> \[Epsilon]/k Sqrt[2]/2, {2,
      1} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2, {3,
      1} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/k Exp[I Pi/4]/
       2}, {4, 4}];
lam4 = SparseArray[{{4, 1} -> \[Epsilon]/k Sqrt[2]/2, {2,
      1} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2, {3,
      1} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[I Pi/4]/2}, {4, 4}];
lam1t = SparseArray[{{1, 1} ->
     Sqrt[1 - \[Epsilon]^2]/k Sqrt[2]/2, {1,
      2} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2, {1,
      3} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/k Exp[I Pi/4]/
       2}, {4, 4}];
lam2t = SparseArray[{{1, 1} ->
     Sqrt[1 - \[Epsilon]^2]/k Sqrt[2]/2, {1,
      2} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2, {1,
      3} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[I Pi/4]/2}, {4, 4}];
lam3t = SparseArray[{{1, 4} -> \[Epsilon]/k Sqrt[2]/2, {1,
      2} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/k Exp[I Pi/4]/
       2, {1, 3} -> (Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2}, {4, 4}];
```

```
lam4t = SparseArray[{{1, 4} -> \[Epsilon]/k Sqrt[2]/2, {1,
      2} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[I Pi/4]/2, {1,
      3} -> -(Sqrt[\[Epsilon]] (1 - \[Epsilon]^2)^(1/4))/
       k Exp[-I Pi/4]/2}, {4, 4}];

       pez1p1 = (Abs[lam1t.Mz1p.pes]^2)[[1]][[1]];
pez1p2 = (Abs[lam2t.Mz1p.pes]^2)[[1]][[1]];
pez1p3 = (Abs[lam3t.Mz1p.pes]^2)[[1]][[1]];
pez1p4 = (Abs[lam4t.Mz1p.pes]^2)[[1]][[1]];
pez1n1 = (Abs[lam1t.Mz1n.pes]^2)[[1]][[1]];
pez1n2 = (Abs[lam2t.Mz1n.pes]^2)[[1]][[1]];
pez1n3 = (Abs[lam3t.Mz1n.pes]^2)[[1]][[1]];
pez1n4 = (Abs[lam4t.Mz1n.pes]^2)[[1]][[1]];

pez11sum = pez1p1 + pez1n1;
pez12sum = pez1p2 + pez1n2;
pez13sum = pez1p3 + pez1n3;
pez14sum = pez1p4 + pez1n4;

pez1 = {pez11p = pez1p1/pez11sum, pez11n = pez1n1/pez11sum}
pez2 = {pez12p = pez1p2/pez12sum, pez12n = pez1n2/pez12sum}
pez3 = {pez13p = pez1p3/pez13sum, pez13n = pez1n3/pez13sum}
pez4 = {pez14p = pez1p4/pez14sum, pez14n = pez1n4/pez14sum}
```

```
pey1p1 = (Abs[lam1t.My1p.pes]^2)[[1]][[1]] // FullSimplify;

pey1p2 = (Abs[lam2t.My1p.pes]^2)[[1]][[1]] // FullSimplify;

pey1p3 = (Abs[lam3t.My1p.pes]^2)[[1]][[1]] // FullSimplify;

pey1p4 = (Abs[lam4t.My1p.pes]^2)[[1]][[1]] // FullSimplify;

pey1n1 = (Abs[lam1t.My1n.pes]^2)[[1]][[1]] // FullSimplify;

pey1n2 = (Abs[lam2t.My1n.pes]^2)[[1]][[1]] // FullSimplify;

pey1n3 = (Abs[lam3t.My1n.pes]^2)[[1]][[1]] // FullSimplify;

pey1n4 = (Abs[lam4t.My1n.pes]^2)[[1]][[1]] // FullSimplify;


pey11sum = pey1p1 + pey1n1;

pey12sum = pey1p2 + pey1n2;

pey13sum = pey1p3 + pey1n3;

pey14sum = pey1p4 + pey1n4;


pey1 = FullSimplify[{pey11p = pey1p1/pey11sum,

   pey11n = pey1n1/pey11sum}]

pey2 = FullSimplify[{pey12p = pey1p2/pey12sum,

   pey12n = pey1n2/pey12sum}]

pey3 = FullSimplify[{pey13p = pey1p3/pey13sum,

   pey13n = pey1n3/pey13sum}]

pey4 = FullSimplify[{pey14p = pey1p4/pey14sum,

   pey14n = pey1n4/pey14sum}]
```

Figures 3.2 code

```
   pey1 /. \[Gamma] -> .1
pey1 /. \[Gamma] -> .9
plotpey1 =
 Plot[pey1, {\[Gamma], 0, 1},
  PlotLabel ->
   "For All Post-States of \!\(\*SubscriptBox[\(\[Sigma]\), \(y\)]\)",
   AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "peyone.eps", plotpey1];
plotpey2 =
 Plot[pey2, {\[Gamma], 0, 1},
  PlotLabel ->
   "Second Post-State of \!\(\*SubscriptBox[\(\[Sigma]\), \(y\)]\)",
  AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "peytwo.eps", plotpey2];
plotpey3 =
 Plot[pey3, {\[Gamma], 0, 1},
  PlotLabel ->
   "Third Post-State of \!\(\*SubscriptBox[\(\[Sigma]\), \(y\)]\)",
  AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "peythree.eps", plotpey3];
plotpey4 =
 Plot[pey4, {\[Gamma], 0, 1},
  PlotLabel ->
   "Fourth Post-State of \!\(\*SubscriptBox[\(\[Sigma]\), \(y\)]\)",
  AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "peyfour.eps", plotpey4];
```

Figure 3.1 code

```
pex1p1 = (Abs[lam1t.Mx1p.pes]^2)[[1]][[1]] // FullSimplify;
pex1p2 = (Abs[lam2t.Mx1p.pes]^2)[[1]][[1]] // FullSimplify;
pex1p3 = (Abs[lam3t.Mx1p.pes]^2)[[1]][[1]] // FullSimplify;
pex1p4 = (Abs[lam4t.Mx1p.pes]^2)[[1]][[1]] // FullSimplify;
pex1n1 = (Abs[lam1t.Mx1n.pes]^2)[[1]][[1]] // FullSimplify;
pex1n2 = (Abs[lam2t.Mx1n.pes]^2)[[1]][[1]] // FullSimplify;
pex1n3 = (Abs[lam3t.Mx1n.pes]^2)[[1]][[1]] // FullSimplify;
pex1n4 = (Abs[lam4t.Mx1n.pes]^2)[[1]][[1]] // FullSimplify;

pex11sum = pex1p1 + pex1n1;
pex12sum = pex1p2 + pex1n2;
pex13sum = pex1p3 + pex1n3;
pex14sum = pex1p4 + pex1n4;

pex1 = FullSimplify[{pex11p = pex1p1/pex11sum,
   pex11n = pex1n1/pex11sum}]
pex2 = FullSimplify[{pex12p = pex1p2/pex12sum,
   pex12n = pex1n2/pex12sum}]
pex3 = FullSimplify[{pex13p = pex1p3/pex13sum,
   pex13n = pex1n3/pex13sum}]
```

```
pex4 = FullSimplify[{pex14p = pex1p4/pex14sum,
   pex14n = pex1n4/pex14sum}]


   plotpex1 =
 Plot[pex1, {\[Gamma], 0, 1},
  PlotLabel ->
   "For All Post-States of \!\(\*SubscriptBox[\(\[Sigma]\), \(x\)]\)",
   AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "pexone.eps", plotpex1];
plotpex2 =
 Plot[pex2, {\[Gamma], 0, 1},
  PlotLabel ->
   "Second Post-State of \!\(\*SubscriptBox[\(\[Sigma]\), \(x\)]\)",
  AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "pextwo.eps", plotpex2];
plotpex3 =
 Plot[pex3, {\[Gamma], 0, 1},
  PlotLabel ->
   "Third Post-State of \!\(\*SubscriptBox[\(\[Sigma]\), \(x\)]\)",
  AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "pexthree.eps", plotpex3];
plotpex4 =
 Plot[pex4, {\[Gamma], 0, 1},
  PlotLabel ->
   "Fourth Post-State of \!\(\*SubscriptBox[\(\[Sigma]\), \(x\)]\)",
  AxesLabel -> {"\[Gamma]=\[Epsilon]^2", "Probability"}]
Export[NotebookDirectory[] <> "pexfour.eps", plotpex4];


Find the circle of x and y adding or subtracting together.  Plot operator as moves aroun
```

Figure 3.3 and 3.4 code

```
xi1 = SparseArray[{{1, 1} -> Sqrt[2]/2, {2, 1} ->
     Exp[I Pi/4]/2, {3, 1} -> Exp[-I Pi/4]/2}, {4, 4}];
xi2 = SparseArray[{{1, 1} ->
     Sqrt[2]/2, {2, 1} -> -Exp[I Pi/4]/2, {3, 1} -> -Exp[-I Pi/4]/
      2}, {4, 4}];
xi3 = SparseArray[{{4, 1} -> Sqrt[2]/2, {2, 1} ->
     Exp[-I Pi/4]/2, {3, 1} -> Exp[I Pi/4]/2}, {4, 4}];
xi4 = SparseArray[{{4, 1} ->
     Sqrt[2]/2, {2, 1} -> -Exp[-I Pi/4]/2, {3, 1} -> -Exp[I Pi/4]/
      2}, {4, 4}];
xi1t = SparseArray[{{1, 1} -> Sqrt[2]/2, {1, 2} ->
     Exp[-I Pi/4]/2, {1, 3} -> Exp[I Pi/4]/2}, {4, 4}];
xi2t = SparseArray[{{1, 1} ->
     Sqrt[2]/2, {1, 2} -> -Exp[-I Pi/4]/2, {1, 3} -> -Exp[I Pi/4]/
      2}, {4, 4}];
xi3t = SparseArray[{{1, 4} -> Sqrt[2]/2, {1, 2} ->
     Exp[I Pi/4]/2, {1, 3} -> Exp[-I Pi/4]/2}, {4, 4}];
xi4t = SparseArray[{{1, 4} ->
```

```
     Sqrt[2]/2, {1, 2} -> -Exp[I Pi/4]/2, {1, 3} -> -Exp[-I Pi/4]/
      2}, {4, 4}];


M2dpol1 = {{0, 0, E^(I \[Phi]), 0}, {0, 0, 0,
    E^(I \[Phi])}, {E^-(I \[Phi]), 0, 0, 0}, {0, E^-(I \[Phi]), 0, 0}};
Eigenvalues[M2dpol1]
Eigenvectors[M2dpol1]
M2dpol1e = SparseArray[{{1, 1} -> E^(I \[Phi]), {3, 1} -> 1}, {4, 4}];
M2dpol1et =
  SparseArray[{{1, 1} -> E^-(I \[Phi]), {1, 3} -> 1}, {4, 4}];
M2dpol1e2 = SparseArray[{{2, 1} -> E^(I \[Phi]), {4, 1} -> 1}, {4, 4}];
M2dpol1e2t =
  SparseArray[{{1, 2} -> E^-(I \[Phi]), {1, 4} -> 1}, {4, 4}];
M2dpol1p = M2dpol1e.M2dpol1et + M2dpol1e2.M2dpol1e2t;
MatrixForm[%]


M2dpol1e3 =
  SparseArray[{{1, 1} -> -E^(I \[Phi]), {3, 1} -> 1}, {4, 4}];
M2dpol1e3t =
  SparseArray[{{1, 1} -> -E^-(I \[Phi]), {1, 3} -> 1}, {4, 4}];
M2dpol1e4 =
  SparseArray[{{2, 1} -> -E^(I \[Phi]), {4, 1} -> 1}, {4, 4}];
M2dpol1e4t =
  SparseArray[{{1, 2} -> -E^-(I \[Phi]), {1, 4} -> 1}, {4, 4}];
M2dpol1n = M2dpol1e3.M2dpol1e3t + M2dpol1e4.M2dpol1e4t;
```

```
MatrixForm[%]


M2dpol1p1 = (Abs[xi1t.M2dpol1p.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1p2 = (Abs[xi2t.M2dpol1p.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1p3 = (Abs[xi3t.M2dpol1p.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1p4 = (Abs[xi4t.M2dpol1p.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1n1 = (Abs[xi1t.M2dpol1n.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1n2 = (Abs[xi2t.M2dpol1n.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1n3 = (Abs[xi3t.M2dpol1n.psi]^2)[[1]][[1]] // FullSimplify;

M2dpol1n4 = (Abs[xi4t.M2dpol1n.psi]^2)[[1]][[1]] // FullSimplify;


M2dpol11sum = M2dpol1p1 + M2dpol1n1;

M2dpol12sum = M2dpol1p2 + M2dpol1n2;

M2dpol13sum = M2dpol1p3 + M2dpol1n3;

M2dpol14sum = M2dpol1p4 + M2dpol1n4;


M2dpol11 =

 FullSimplify[{M2dpol11p = M2dpol1p1/M2dpol11sum,

   M2dpol11n = M2dpol1n1/M2dpol11sum}]

M2dpol12 =

 FullSimplify[{M2dpol12p = M2dpol1p2/M2dpol12sum,

   M2dpol12n = M2dpol1n2/M2dpol12sum}]

M2dpol13 =

 FullSimplify[{M2dpol13p = M2dpol1p3/M2dpol13sum,

   M2dpol13n = M2dpol1n3/M2dpol13sum}]
```

```
M2dpol14 =
 FullSimplify[{M2dpol14p = M2dpol1p4/M2dpol14sum,
   M2dpol14n = M2dpol1n4/M2dpol14sum}]


M2dplot1 =
 Plot[M2dpol11, {\[Phi], 0, 2 Pi},
  AxesLabel -> {"\[Phi]", "Probability"}]
Export[NotebookDirectory[] <> "mtplotone.eps", M2dplot1];
M2dplot2 =
 Plot[M2dpol12, {\[Phi], 0, 2 Pi},
  AxesLabel -> {"\[Phi]", "Probability"}]
Export[NotebookDirectory[] <> "mtplottwo.eps", M2dplot2];
M2dplot3 =
 Plot[M2dpol13, {\[Phi], 0, 2 Pi},
  AxesLabel -> {"\[Phi]", "Probability"}]
Export[NotebookDirectory[] <> "mtplotthree.eps", M2dplot3];
M2dplot4 =
 Plot[M2dpol14, {\[Phi], 0, 2 Pi},
  AxesLabel -> {"\[Phi]", "Probability"}]
Export[NotebookDirectory[] <> "mtplotfour.eps", M2dplot4];
```

Figures 3.5-3.8 code

```
M3dpol1 = {{Cos[\[Theta]], 0, Sin[\[Theta]] E^(I \[Phi]), 0}, {0,
    Cos[\[Theta]], 0,
    Sin[\[Theta]] E^(I \[Phi])}, {Sin[\[Theta]] E^-(I \[Phi]),
    0, -Cos[\[Theta]], 0}, {0, Sin[\[Theta]] E^-(I \[Phi]),
    0, -Cos[\[Theta]]}};
Eigenvalues[M3dpol1]
Eigenvectors[M3dpol1]
M3dpol1e =
  SparseArray[{{1, 1} ->
     E^(I \[Phi]) (1 + Cos[\[Theta]]) Csc[\[Theta]], {3, 1} -> 1}, {4,
     4}];
M3dpol1et =
  SparseArray[{{1, 1} ->
     E^(-I \[Phi]) (1 + Cos[\[Theta]]) Csc[\[Theta]], {1, 3} ->
     1}, {4, 4}];
M3dpol1e2 =
  SparseArray[{{2, 1} ->
     E^(I \[Phi]) (1 + Cos[\[Theta]]) Csc[\[Theta]], {4, 1} -> 1}, {4,
     4}];
M3dpol1e2t =
  SparseArray[{{1, 2} ->
     E^(-I \[Phi]) (1 + Cos[\[Theta]]) Csc[\[Theta]], {1, 4} ->
     1}, {4, 4}];
M3dpol1p = M3dpol1e.M3dpol1et + M3dpol1e2.M3dpol1e2t;
MatrixForm[%]


M3dpol1e3 =
  SparseArray[{{1, 1} ->
     E^(I \[Phi]) (-1 + Cos[\[Theta]]) Csc[\[Theta]], {3, 1} ->
     1}, {4, 4}];
M3dpol1e3t =
  SparseArray[{{1, 1} ->
     E^(-I \[Phi]) (-1 + Cos[\[Theta]]) Csc[\[Theta]], {1, 3} ->
     1}, {4, 4}];
M3dpol1e4 =
  SparseArray[{{2, 1} ->
     E^(I \[Phi]) (-1 + Cos[\[Theta]]) Csc[\[Theta]], {4, 1} ->
     1}, {4, 4}];
M3dpol1e4t =
  SparseArray[{{1, 2} ->
     E^(-I \[Phi]) (-1 + Cos[\[Theta]]) Csc[\[Theta]], {1, 4} ->
     1}, {4, 4}];
M3dpol1n = M3dpol1e3.M3dpol1e3t + M3dpol1e4.M3dpol1e4t;
MatrixForm[%]


M3dpol1p1 = (Abs[xi1t.M3dpol1p.psi]^2)[[1]][[1]] // FullSimplify;
M3dpol1p2 = (Abs[xi2t.M3dpol1p.psi]^2)[[1]][[1]] // FullSimplify;
M3dpol1p3 = (Abs[xi3t.M3dpol1p.psi]^2)[[1]][[1]] // FullSimplify;
M3dpol1p4 = (Abs[xi4t.M3dpol1p.psi]^2)[[1]][[1]] // FullSimplify;
M3dpol1n1 = (Abs[xi1t.M3dpol1n.psi]^2)[[1]][[1]] // FullSimplify;
M3dpol1n2 = (Abs[xi2t.M3dpol1n.psi]^2)[[1]][[1]] // FullSimplify;
M3dpol1n3 = (Abs[xi3t.M3dpol1n.psi]^2)[[1]][[1]] // FullSimplify;
```

```
M3dpol1n4 = (Abs[xi4t.M3dpol1n.psi]^2)[[1]][[1]] // FullSimplify;


M3dpol11sum = M3dpol1p1 + M3dpol1n1;

M3dpol12sum = M3dpol1p2 + M3dpol1n2;

M3dpol13sum = M3dpol1p3 + M3dpol1n3;

M3dpol14sum = M3dpol1p4 + M3dpol1n4;


M3dpol11 =

 FullSimplify[{M3dpol11p = M3dpol1p1/M3dpol11sum,

   M3dpol11n = M3dpol1n1/M3dpol11sum}]

M3dpol12 =

 FullSimplify[{M3dpol12p = M3dpol1p2/M3dpol12sum,

   M3dpol12n = M3dpol1n2/M3dpol12sum}]

M3dpol13 =

 FullSimplify[{M3dpol13p = M3dpol1p3/M3dpol13sum,

   M3dpol13n = M3dpol1n3/M3dpol13sum}]

M3dpol14 =

 FullSimplify[{M3dpol14p = M3dpol1p4/M3dpol14sum,

   M3dpol14n = M3dpol1n4/M3dpol14sum}]


base = SphericalPlot3D[1, {\[Theta], 0, Pi}, {\[Phi], 0, 2 Pi},

   AxesLabel -> {x, y, z}, PlotStyle -> Directive[Opacity[0.3]],

   Mesh -> None];


cm = 72/2.54;
```

```
sphdpol1 =
  SphericalPlot3D[M3dpol11, {\[Theta], 0, Pi}, {\[Phi], 0, 2 Pi},
   AxesLabel -> {x, y, z}];
sprime1 = Show[base, sphdpol1, PlotRange -> All]
sphone = Rasterize[Show[sprime1, ImageSize -> 10 cm],
   ImageResolution -> 600];
Export[NotebookDirectory[] <> "sphone.eps", sphone];


sphdol2 =
  SphericalPlot3D[M3dpol12, {\[Theta], 0, Pi}, {\[Phi], 0, 2 Pi},
   AxesLabel -> {x, y, z}];
sprime2 = Show[base, sphdol2, PlotRange -> All]
sphtwo = Rasterize[Show[sprime2, ImageSize -> 10 cm],
   ImageResolution -> 600];
Export[NotebookDirectory[] <> "sphtwo.eps", sphtwo];


sphdpol3 =
  SphericalPlot3D[M3dpol13, {\[Theta], 0, Pi}, {\[Phi], 0, 2 Pi},
   AxesLabel -> {x, y, z}];
sprime3 = Show[base, sphdpol3, PlotRange -> All]
sphthree =
  Rasterize[Show[sprime3, ImageSize -> 10 cm], ImageResolution -> 600];
Export[NotebookDirectory[] <> "sphthree.eps", sphthree];


sphdpol4 =
  SphericalPlot3D[M3dpol14, {\[Theta], 0, Pi}, {\[Phi], 0, 2 Pi},
   AxesLabel -> {x, y, z}];
sprime4 = Show[base, sphdpol4, PlotRange -> All]
sphfour =
  Rasterize[Show[sprime4, ImageSize -> 10 cm], ImageResolution -> 600];
Export[NotebookDirectory[] <> "sphfour.eps", sphfour];
```

Bibliography

[1] L. Vaidman, Y. Aharonov, and D. Albert, “How to Ascertain the Values of σ_x, σ_y, and σ_z of a Spin-$\frac{1}{2}$ Particle,” Phys. Rev. Lett. **58,** 1385 (1987).

[2] A. Lund and H. Wiseman, “Measuring measurementâĂŞdisturbance relationships with weak values,” New J. Phys **12,** 093011 (2010).

[3] Y. Aharonov and L. Vaidman, “Complete description of a quantum system at a given time,” J. Phys. A: Math Gen **24,** 2315 (1991).