STATE ESTIMATION OF N-LEVEL QUANTUM SYSTEMS

A. Magyar[1,2,4], L. Ruppert[2], K. M. Hangos[1], D. Petz[2,3]

Technical report SCL-007/2006

[1] Systems and Control Laboratory, Computer and Automation Research Institute, Budapest, Hungary
[2] Department of Analysis, Budapest University of Technology and Economics, Budapest, Hungary
[3] Alfréd Rényi Institute of Mathematics, Budapest, Hungary
[4] Department of Computer Science, University of Pannonia, Veszprém, Hungary

Contents

1 Introduction **2**
- 1.1 Motivation and aim . . . 2
- 1.2 Quantum mechanics - system theoretical point of view . . . 2
  - 1.2.1 Axioms of quantum mechanics . . . 2
  - 1.2.2 State estimation . . . 3
  - 1.2.3 Parameter estimation . . . 3

2 Measurement scheme **4**
- 2.1 Measurement of the Pauli-tensor basis elements . . . 4
  - 2.1.1 Bloch-parametrization . . . 4
  - 2.1.2 Observables . . . 5
- 2.2 A more general measurement scheme for N-level systems . . . 6
  - 2.2.1 Parametrization of the density operator . . . 6
  - 2.2.2 Observables . . . 6

3 Estimation scheme **8**
- 3.1 Unconstrained state estimation . . . 8
  - 3.1.1 Unconstrained point estimate of a qubit . . . 8
  - 3.1.2 Unconstrained estimator for N-level systems . . . 9
- 3.2 Comparison of different unconstrained estimation schemes . . . 9
  - 3.2.1 Estimator (3.1) . . . 9
  - 3.2.2 Minimal qubit tomography . . . 10
  - 3.2.3 Maximum likelihood estimate for minimal qubit tomography . . . 10
  - 3.2.4 Standard tomography . . . 10
  - 3.2.5 Estimator based on 5 mutually unbiased measurements for 2 qubits . . . 11
  - 3.2.6 Estimator based on 3 mutually unbiased measurements (also for 2 qubits) . . . 12
  - 3.2.7 Differences . . . 12
- 3.3 Constrained state estimation . . . 13
  - 3.3.1 Geometrical constraint . . . 13
  - 3.3.2 Eigenvalue-based constraint . . . 13
  - 3.3.3 Properties of the constrained estimators . . . 14

4 Simulation results **15**
- 4.1 Simulation environment . . . 15
- 4.2 Comparison of unconstrained and constrained estimation schemes . . . 15
  - 4.2.1 Measures of goodness . . . 15
  - 4.2.2 Three level system . . . 15

5 Conclusions **20**

6 Appendix **22**
- 6.1 Matlab source . . . 22
  - 6.1.1 Function FidelPlot.m . . . 22
  - 6.1.2 Function qls.m . . . 27

Chapter 1

Introduction

1.1 Motivation and aim

One of the biggest challenges of nowadays is the building of the quantum computer. It would be more effective than the classical one in solving certain problems, e.g. *prime factorization*. While the classical information of a classical computer is being stored as *bit*s the quantum information in a quantum computer will be stored as quantum bits, or *qubits* in short.
In order to have an operating quantum computer one must be able to transfer, modify, and *read* the quantum information of it.

The aim of this work is to develop and examine methods for guessing the real state of a quantum mechanical systems. The engineering literature uses the word state estimation for this task, while the physicist society terms it tomography.

The report is structured as follows. This chapter gives a brief introduction to quantum mechanical systems. The basic problem statements of state estimation and parameter estimation is also introduced here.
In chapter 2 measurement of quantum systems is discussed. Two different approach is used: the first uses numbers of quantum bits, i.e. quantum words, while the other assumes a d-level quantum system, or a *qudit*, in short. Of course, the quantities to be measured are somewhat different for the two case.

Once the measurement is done and measurement data is available from the system, it is possible to give a guess of the quantum system's actual state. A simple but effective method is used here, and it is compared with other estimators available in the literature. A useful extension of the estimator is also done in Chapter 3.

Chapter 4 shows simulation results made with the help of a simulator of our own making. Different types of experiments were performed to get answers to various questions about the previously introduced estimator.

Finally, chapter 5 concludes the work.

1.2 Quantum mechanics - system theoretical point of view

In this section we try to give a connection between quantum mechanics and system theory. First, the postulates of quantum mechanics is given in brief. Afterwards, system theoretical interpretation of the important elements are given.

1.2.1 Axioms of quantum mechanics

A1 Every quantum mechanical system **S** is there is a Hilbert-space $\mathcal{H}$. The *states* of **S** are statistical operators ρ acting on $\mathcal{H}$, they can be conveniently represented by positive (semi-)definite self-adjoint matrices having unit trace:

$$\rho \in \mathbb{C}^n, \quad \rho = \rho^*, \quad \mathrm{Tr}\rho = 1 \tag{1.1}$$

A2 The measurable physical quantities of quantum systems (so called *observables*) are represented by self-adjoint operators of $\mathcal{H}$ (i.e. self-adjoint matrices of $\mathbb{C}^{n\times n}$).

A3 The *measurement of an observable* A has a probabilistic nature. The possible outcomes of the measurement are the different λ eigenvalues of A, the corresponding probability is

$$\mathrm{Prob}(\lambda_i) = \mathrm{Tr}E_i\rho E_i^*,$$

where E_i are the eigenprojections corresponding to λ_i, i.e.

$$\sum_i E_i = I, \quad E_i^2 = E_i.$$

Moreover, the state of the system **S** after measuring A, and having the outcome λ_i changes to

$$\rho' = \frac{E_i\rho E_i^*}{\mathrm{Tr}E_i\rho E_i^*}$$

that means that the measurement has lost its good property of being a passive operation known from classical physics. *The measurement changes the actual state of the quantum system.*

A4 A quantum system **S** created as the composition of quantum systems $\mathbf{S_1}$, and $\mathbf{S_2}$ is described by the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$, where $\mathcal{H}_1$, and $\mathcal{H}_2$ are the Hilbert-spaces corresponding to $\mathbf{S_1}$ and $\mathbf{S_2}$, respectively.

A5 The *dynamical change* of the system **S** in the time interval $[s,t]$ is described by unitary propagators $U_{t,s}$:

$$\rho(t) = U_{t,s}\rho_s U_{t,s}^*$$

where unitary $U_{t,s}$ has the following properties:

$$U_{t,s}U_{s,r} = U_{t,r} \quad \text{and} \quad (s,t) \mapsto U_{s,t} \ \text{ is strongly continuous.}$$

1.2.2 State estimation

1.2.3 Parameter estimation

Chapter 2

Measurement scheme

As it was mentioned in section 1.2.1, the measurement has very special properties in the quantum mechanical domain. That's why the typical scenario of state estimation (i.e. one measures the output signal of *one* certain system, and from the measurement data containing the measured output signals, and the input signals introduced to the system one determines an estimate of the state signal) doesn't work. One difficulty is the probabilistic nature, i.e. there are different possible outcomes with certain probabilities. On the other hand, the state of interest damages immediately with the first measurement.

In order to avoid the difficulty caused by the fact that measurement has this strong effect on the system's state a very strong assumption is made: it is assumed that there are sufficiently *many systems* being in the *same state* (i.e. they are described by the same density matrix ρ), so the repeated von Neumann measurements are performed on different systems, but their state is the same. This way the state of a certain system after the measurement is irrelevant.

2.1 Measurement of the Pauli-tensor basis elements

The first approach of measurement is based on the so called Bloch-parametrization of the state space. The advantage of this parametrization is that it can be straightforwardly generalized from the simplest case of the qubit. On the other hand, the drawback is that it's only available for 2^N level quantum systems.

2.1.1 Bloch-parametrization

The Pauli-matrices

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

together with the unit matrix

$$\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

form a basis of the 2×2 density matrices.

Two level systems

For qubits, the Bloch-parametrization gives a convenient geometric view of the state space:

$$\rho = \frac{1}{2}(\sigma_0 + \theta_1\sigma_1 + \theta_2\sigma_2 + \theta_3\sigma_3),$$

this way the state of a qubit can be represented as a vector $\theta = (\theta_1, \theta_2, \theta_3)^T$ of $\mathbb{R}^3$. Taking the properties (1.1) into account the state space of a 2-level system is restricted to the unit ball of $\mathbb{R}^3$, this is the so called Bloch ball (see Fig.2.1). The pure states of the system are on the sphere, while the mixed states are inside the ball.

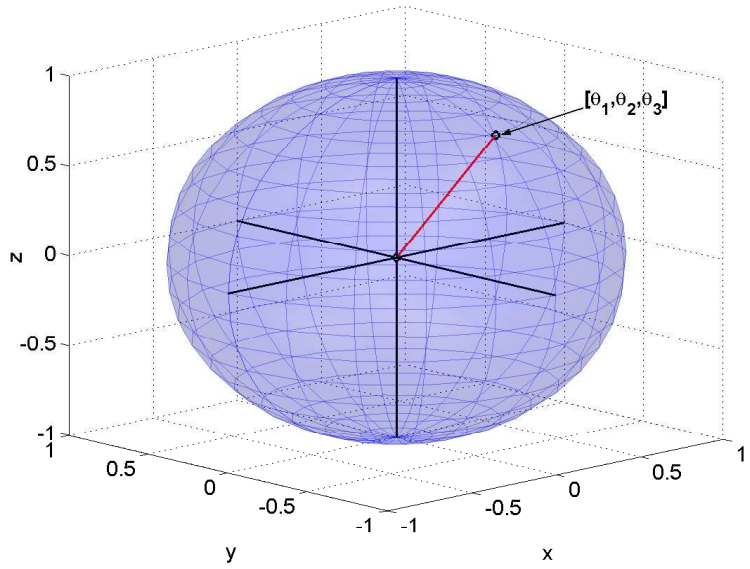

Figure 2.1: *Bloch vector of a 2 level quantum system - the state space is the Bloch ball.*

Generalization of for 2^N level systems

If the dimension of the system is 2^N, then a natural way of parameterizing the state uses the N-fold tensor product of the Pauli base

$$\rho(\theta) = \frac{1}{2^N} \sum_{i_1,i_2,\ldots,i_N=0}^{3} \theta_{i_1,i_2,\ldots,i_N} \cdot \sigma_{i_1} \otimes \sigma_{i_2} \otimes \ldots \otimes \sigma_{i_N}, \qquad \theta_{0,0,\ldots,0} = 1$$

i.e it is a multidimensional extension of the Bloch-vector. For 4 level systems the dimension of the Bloch-matrix is 4×4:

$$\Theta = \begin{bmatrix} 1 & \theta_{01} & \theta_{02} & \theta_{03} \\ \theta_{10} & \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{20} & \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{30} & \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix}$$

That is, the state has $4 \times 4 - 1 = 15$ parameters [1].
The natural structure for the coefficients $\theta_{i_1,\ldots,i_N}$ of the basis elements is an N-dimensional hyper-matrix of size $4 \times 4 \times \ldots \times 4$, i.e. a *Bloch-hyper-matrix*:

$$\Theta = (\theta_{i_1,\ldots,i_N})_{i_1,\ldots,i_N=0}^{3}$$

As before, the Bloch-parametrization ensures unit trace for $\rho(\Theta)$ since all the Pauli-tensor matrices are traceless except for the unity. The positivity of the density operator of size $2^N \times 2^N$ is still a constraint to be respected.

2.1.2 Observables

A natural set of observables for the above introduced 2^N-level systems are the $2^N - 1$ elements of the Pauli basis. All of them is self adjoint, and has the eigenvalues $+1$ and -1 (with the corresponding multiplicities).

The simplest case is the qubit since the observables are the Pauli spins in the 3 direction. If the total number of measurements if denoted by n, then $m = 3r$. It means, that the three independent von Neumann measurements are performed on three different copies of the system in each measurement step r. The probability of having the outcome ± 1 while measuring σ_i in state $\rho(\theta)$ is given by the formula

$$\text{Prob}(\pm 1) = \frac{1}{2}(1 \pm \theta_i), \quad i = 1, 2, 3.$$

Similarly, the probability of obtaining ± 1 while measuring observables

$$\sigma_{i_1} \otimes \sigma_{i_2} \otimes \ldots \otimes \sigma_{i_N} \tag{2.1}$$

in state $\rho(\Theta)$ is

$$\mathrm{Prob}(\pm 1) = \frac{1}{2}(1 \pm \theta_{i_1,\ldots,i_N}). \tag{2.2}$$

2.2 A more general measurement scheme for N-level systems

The use of Pauli(-tensor) matrices as basis for parametrization and observables for measurement is very useful for N-qubit systems (i.e. 2^N-level systems) but another approach is necessary if one is to deal with N-level systems (e.g. *qutrit*s - three level quantum systems).

2.2.1 Parametrization of the density operator

The most general parametrization of the density matrix that suits for all quantum systems uses the matrix elements as parameters:

$$\rho(\theta) = \sum_{k=1}^{N} \theta_{kk} E_{kk} + \sum_{i<j} \left(\theta_{ij}(E_{ij} + E_{ji}) + \theta_{ji}(iE_{ij} - iE_{ji})\right),$$

where E_{ij} are the matrix units (full of zeros except for the i,j-th element which is one). This way, the state of an N-level system can be given with N^2 real parameters. It can be seen that in contrast with the Bloch-parametrization this parametrization does not ensure the unit trace for the density, so a reasonable modification is:

$$\rho(\theta) = \sum_{k=1}^{N-1} \theta_{kk} E_{kk} + \left(1 - \sum_{k=1}^{N-1} \theta_{kk}\right) E_{NN} + \sum_{i<j} \left(\theta_{ij}(E_{ij} + E_{ji}) + \theta_{ji}(iE_{ij} - iE_{ji})\right).$$

This offers the use of a generalization of the Bloch-vector space [1]. The Bloch-vector of an N-level quantum system is a vector of $\mathbb{R}^{N^2-1}$.
For 3-level systems, the density $\rho(\theta)$ can be written up as

$$\rho(\theta) = \begin{bmatrix} \theta_{11} & \theta_{12} - i\theta_{21} & \theta_{13} - i\theta_{31} \\ \theta_{12} + i\theta_{21} & \theta_{22} & \theta_{23} - i\theta_{32} \\ \theta_{13} + i\theta_{31} & \theta_{23} + i\theta_{32} & 1 - (\theta_{11} + \theta_{22}) \end{bmatrix}$$

The state of a qutrit thus can be represented by a vector $\theta \in \mathbb{R}^8$.

2.2.2 Observables

The measurement connected to the above parametrization means the measuring of observables of three types:

- *Observables for the off-diagonal real entries*

  $$B_{ij} = E_{ij} + E_{ji}, \quad i,j = 1\ldots,N, \quad i < j \tag{2.3}$$

  it has the spectral decomposition

  $$B_{ij} = 1 \cdot \frac{1}{2}(E_{ii} + E_{ij} + E_{ji} + E_{jj}) + 0 \cdot \sum_{i \neq m \neq j} E_{mm} - 1 \cdot \frac{1}{2}(E_{ii} - E_{ij} - E_{ji} + E_{jj})$$

  The probabilities of the three different outcomes are as follows:

  $$\mathrm{Prob}(B_{ij} = \pm 1) = \frac{1}{2}(\theta_{ii} \pm (\theta_{ij} - i\theta_{ji}) \pm (\theta_{ij} + i\theta_{ji}) + \theta_{jj}) = \frac{1}{2}(\theta_{ii} + \theta_{jj}) \pm \theta_{ij}$$

  $$\mathrm{Prob}(B_{ij} = 0) = \sum_{i \neq m \neq j} \theta_{mm}$$

- *Observables for the off-diagonal imaginary entries*

$$C_{ij} = iE_{ij} - iE_{ji}, \quad i,j = 1\ldots,N, \quad i<j \tag{2.4}$$

it has the spectral decomposition

$$C_{ij} = 1\cdot\frac{1}{2}(E_{ii} - iE_{ij} + iE_{ji} + E_{jj}) + 0\cdot\sum_{i\neq m\neq j} E_{mm} - 1\cdot\frac{1}{2}(E_{ii} + iE_{ij} - iE_{ji} + E_{jj})$$

The probabilities of the three different outcomes are as follows:

$$\mathrm{Prob}(C_{ij} = \pm 1) = \frac{1}{2}(\theta_{ii} \mp i(\theta_{ij} - i\theta_{ji}) \pm i(\theta_{ij} + i\theta_{ji}) + \theta_{jj}) = \frac{1}{2}(\theta_{ii} + \theta_{jj}) \mp \theta_{ji}$$

$$\mathrm{Prob}(C_{ij} = 0) = \sum_{i\neq m\neq j} \theta_{mm}$$

- *Observables for the diagonal entries*

$$A_{ii} = E_{ii}, \quad i = 1,\ldots,N \tag{2.5}$$

it has the spectral decomposition

$$A_{ii} = 1\cdot E_{ii} + 0\cdot\sum_{i\neq m} E_{mm}$$

The probabilities of the two different outcomes are as follows:

$$\mathrm{Prob}(A_{ii} = +1) = \theta_{ii}$$

$$\mathrm{Prob}(A_{ii} = 0) = \sum_{i\neq m} \theta_{mm}$$

Chapter 3

Estimation scheme

This chapter starts with a simple version of the state estimation problem where the output of the estimator is not necessarily physically meaningful. This kind or estimators is termed *unconstrained estimator* in the sequel. Unconstrained estimators have fortunate statistical properties, this enables one to compare different unconstrained estimators using statistical tools.
Finally, a more sophisticated, the so called *constrained estimator* is introduced and examined for which the resulting state-estimates have physical meaning.

3.1 Unconstrained state estimation

In what follows it is assumed that we have measurement records from measurements of observables (2.1), or observables (2.3-2.5). Since the measurements have outcomes ± 1 (and 0 in the latter case), the records can be regarded as strings of these three symbols.
If M is the measurement which has the outcome t, then $\nu(r, M, t)$ denotes the relative frequency of t, when the measurement is performed r times.

3.1.1 Unconstrained point estimate of a qubit

Given a state, or equivalently a Bloch-vector, the probability of the outcome $+1$ of the Pauli-spin measurements can be given by formula (2.2). The basis of the following point estimation methods is the fact that increasing the number of measurements r the relative frequency of the outcome $+1$ goes to it's real probability:

$$\nu(r, \sigma_i, +1) \to \frac{1}{2}(1 + \theta_i), \quad \text{if} \quad r \to \infty, \quad i = 1, 2, 3.$$

Assuming that the number of measurement r is large enough, θ_i can be expressed using $\nu(r, \sigma_i, +1)$:

$$\Phi = \begin{bmatrix} 2\nu(r, \sigma_1, +1) - 1 \\ 2\nu(r, \sigma_2, +1) - 1 \\ 2\nu(r, \sigma_3, +1) - 1 \end{bmatrix} \tag{3.1}$$

It is important to note that the range (i.e. the set of all possible values) of the estimator (3.1) is not the Bloch ball, but the cube which is tangential to the Bloch ball at the centers of the six bordering squares. It means that the estimator (3.1) tends to give physically meaningless results typically for small numbers of measurements, or for states which are near (or on) the surface of the ball.

3.1.2 Unconstrained estimator for N-level systems

A more general version of the estimator is obtained by using data coming from measurements of observables (2.3 - 2.5). The same assumption is used as in the qubit case, so the estimator has the form:

$$\begin{aligned} \Phi_{kk} &= \nu(r, A_{kk}, +1), \quad k = 1, \ldots, N-1 \\ \Phi_{NN} &= 1 - \textstyle\sum_{k=1}^{N-1} \nu(r, A_{kk}, +1), \\ \Phi_{ij} &= \tfrac{1}{2}(\nu(r, B_{ij}, +1) - \nu(r, B_{ij}, +1)), \quad i < j \\ \Phi_{ji} &= \tfrac{1}{2}(\nu(r, C_{ij}, +1) - \nu(r, C_{ij}, +1)), \quad i < j \end{aligned} \tag{3.2}$$

Just like in the previous case, the state space is a real subset of the estimator's range.

3.2 Comparison of different unconstrained estimation schemes

In what follows different qubit state estimation schemes available in the literature are compared by means of their *mean squared error matrix*. If the state of the system is parameterized by $\theta = (\theta_1, \ldots, \theta_m)$, then the mean squared error matrix is an $m \times m$ matrix with entries defined as follows:

$$V(\theta)_{ij} = E\left[(\Phi(x)_i - \theta_i)(\Phi(x)_j - \theta_j)\right], \quad i, j = 1, \ldots, m.$$

For two level systems, $V_n(\theta)$ is a 3×3 matrix. In case of unbiased estimator, the mean squared error matrix is the covariance matrix otherwise it is corrected with the bias, i.e.

$$V(\theta) = Cov(\theta) + b(\theta)b(\theta)^T,$$

where

$$b(\theta)_i = E\left[\Phi(x)_i - \theta_i\right], \quad i = 1, \ldots, m$$

is the bias. The estimator Φ is termed unbiased if the bias is zero i.e. its' expectation value is the real value of the parameter θ.

3.2.1 Estimator (3.1)

This is the well known case, where the estimator has the form (3.1).
The expectation value of the estimator based on 1 triplet of measurement can be computed easily from the possible values of Φ and their probabilities:

$$\Phi_i = \begin{cases} +1, & p_1 = \frac{1}{2}(1 + \theta_i) \\ -1, & p_2 = \frac{1}{2}(1 - \theta_i) \end{cases}, \quad i = 1, 2, 3.$$

The expectation is

$$E\left[\Phi_i\right] = +1p_1 - 1p_2 = \frac{1}{2}(1 + \theta_i) - \frac{1}{2}(1 - \theta_i) = \theta_i, \quad i = 1, 2, 3,$$

i.e. the estimator is unbiased. Using the above estimator the mean squared error matrix is

$$V_{3\times r}(\theta) = \frac{1}{r}\begin{bmatrix} 1 - \theta_1^2 & 0 & 0 \\ 0 & 1 - \theta_2^2 & 0 \\ 0 & 0 & 1 - \theta_3^2 \end{bmatrix}.$$

It is apparent that the matrix is diagonal, but it naturally comes from the fact that the measurements of σ_1, σ_2, and σ_3 are performed independently.

3.2.2 Minimal qubit tomography

This estimator can be found in more details in [2]. Here only a short summary is given. Let

$$a_1 = \frac{1}{\sqrt{3}}(1,1,1),\ a_2 = \frac{1}{\sqrt{3}}(1,-1,-1),\ a_3 = \frac{1}{\sqrt{3}}(-1,1,-1),\ a_4 = \frac{1}{\sqrt{3}}(-1,-1,1)$$

and define the measurement with the operators

$$F_i = \frac{1}{4}(\sigma_0 + a_i \cdot \sigma),\ i = 1,\ldots,4,\ \sum F_i = I.$$

The probabilities of the 4 outcomes are

$$p_i = \mathrm{Tr} F_i \rho_\theta$$

The estimator is defined as

$$\Phi_n^{min} = 3\sum_{j=1}^{4} \frac{n_j}{n} \cdot a_j \tag{3.3}$$

where n_j is the number of the outcome j from n measurements. The mean squared error matrix is:

$$V_n^{min}(\theta) = \frac{1}{n}\begin{bmatrix} 3-\theta_1^2 & \sqrt{3}\theta_3 - \theta_1\theta_2 & \sqrt{3}\theta_2 - \theta_1\theta_3 \\ \sqrt{3}\theta_3 - \theta_1\theta_2 & 3-\theta_2^2 & \sqrt{3}\theta_1 - \theta_2\theta_3 \\ \sqrt{3}\theta_2 - \theta_1\theta_3 & \sqrt{3}\theta_1 - \theta_2\theta_3 & 3-\theta_3^2 \end{bmatrix}.$$

3.2.3 Maximum likelihood estimate for minimal qubit tomography

Let a_1, a_2, a_3 and a_4 be as above,

$$F_i = \frac{1}{4}(\sigma_0 + a_i \cdot \sigma),\ i = 1,\ldots,4,\ \sum F_i = I.$$

The probabilities of the 4 outcomes are

$$p_i = \mathrm{Tr} F_i \rho_\theta = \frac{1}{4}(1 + a_i \cdot \theta)$$

The estimator is

$$\Phi(i) = \frac{1}{2}(\sigma_0 + a_i \cdot \sigma) = 2F_i,\ i = 1,\ldots,4 \tag{3.4}$$

The estimator is biased, since its' expectation value (in Bloch-vector representation) is $\frac{1}{3}\theta$. The mean squared error matrix is:

$$V_1^{ML}(\theta) = \frac{1}{9}\begin{bmatrix} 3-\theta_1^2 & \sqrt{3}\theta_3 - \theta_1\theta_2 & \sqrt{3}\theta_2 - \theta_1\theta_3 \\ \sqrt{3}\theta_3 - \theta_1\theta_2 & 3-\theta_2^2 & \sqrt{3}\theta_1 - \theta_2\theta_3 \\ \sqrt{3}\theta_2 - \theta_1\theta_3 & \sqrt{3}\theta_1 - \theta_2\theta_3 & 3-\theta_3^2 \end{bmatrix}.$$

3.2.4 Standard tomography

Let $\sigma_i = P_i - Q_i$. Define the Positive Operator Valued Measurement (POVM)

$$F_i = \frac{1}{3}P_i, \quad F_{i+3} = \frac{1}{3}Q_i, \quad i = 1,2,3.$$

The probabilities of the six outcomes are

$$p_i = \frac{1}{6}(1+\theta_i), \quad \frac{1}{6}(1-\theta_i), \quad i = 1,2,3.$$

The estimator is defined using the notation from the previous section. Let

$$\begin{aligned} a_1 &= (1,0,0), & a_4 &= (-1,0,0), \\ a_2 &= (0,1,0), & a_5 &= (0,-1,0), \\ a_3 &= (0,0,1), & a_6 &= (0,0,-1), \end{aligned}$$

and the estimator is

$$\Phi_n^{stand} = 3\sum_{j=1}^{6} \frac{n_j}{n} \cdot a_j. \tag{3.5}$$

This way the mean squared error matrix is

$$V_n^{stand}(\theta) = \frac{1}{n}\begin{bmatrix} 3-\theta_1^2 & -\theta_1\theta_2 & -\theta_1\theta_3 \\ -\theta_1\theta_2 & 3-\theta_2^2 & -\theta_2\theta_3 \\ -\theta_1\theta_3 & -\theta_2\theta_3 & 3-\theta_3^2 \end{bmatrix}.$$

3.2.5 Estimator based on 5 mutually unbiased measurements for 2 qubits

The basic idea of the following estimator is to measure all the qubits in one step and handle the state estimation of one qubit based on m measurements as the estimation a 2^m-level system based on one measurement. The state of the 2^2-level system is:

$$\rho(\theta) = \frac{1}{2}(I + \theta\cdot\sigma) \otimes \frac{1}{2}(I + \theta\cdot\sigma)$$

It is parameterized by $\theta = (\theta_1, \theta_2, \theta_3)$. Five mutually unbiased measurements are performed on the above qubit-pair. The measurements are defined by the bases [3]:

$$\mathcal{B}_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathcal{B}_1 = \tfrac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{pmatrix}$$

$$\mathcal{B}_2 = \tfrac{1}{2}\begin{pmatrix} 1 & i & i & -1 \\ 1 & -i & -i & -1 \\ 1 & i & -i & 1 \\ 1 & -i & i & 1 \end{pmatrix} \quad \mathcal{B}_3 = \tfrac{1}{2}\begin{pmatrix} 1 & 1 & -i & i \\ 1 & -1 & i & i \\ 1 & 1 & i & -i \\ 1 & -1 & -i & -i \end{pmatrix}$$

$$\mathcal{B}_4 = \tfrac{1}{2}\begin{pmatrix} 1 & -i & 1 & i \\ 1 & i & -1 & i \\ 1 & i & 1 & -i \\ 1 & -i & -1 & -i \end{pmatrix}$$

The projection matrices of the measurements are denoted by $P_a(x)$, where $a = 0, \ldots, 5$ denotes the observable, x is for the outcome.
The estimator of the Bloch-vector is

$$\Phi_2^{MUB} = \begin{pmatrix} Tr\left[\left(-I + \sum_{x,a} P_a(x)\nu_a(x)\right)\cdot(I\otimes\sigma_1)\right] \\ Tr\left[\left(-I + \sum_{x,a} P_a(x)\nu_a(x)\right)\cdot(I\otimes\sigma_2)\right] \\ Tr\left[\left(-I + \sum_{x,a} P_a(x)\nu_a(x)\right)\cdot(I\otimes\sigma_3)\right] \end{pmatrix}. \tag{3.6}$$

where $\nu_a(x)$ stands for the relative frequency of the outcome x ($x = 1, 2, 3, 4$) in measurement a ($a = 0, 1, 2, 3, 4, 5$). Using the above estimator the mean squared error matrix is

$$V_{5\times 2}(\theta) = \begin{bmatrix} 1-\theta_1^2 & 0 & 0 \\ 0 & 1-\theta_2^2 & 0 \\ 0 & 0 & 1-\theta_3^2 \end{bmatrix}.$$

3.2.6 Estimator based on 3 mutually unbiased measurements (also for 2 qubits)

In the previous section 5 measurements were performed on qubit pairs being in the same state θ.
In what follows,we are using only $\mathcal{B}_0, \mathcal{B}_1$ and $\mathcal{B}_2$. The estimation works component-wise, $\mathcal{B}_0$ gives information about θ_3, $\mathcal{B}_1$ about θ_1, and $\mathcal{B}_2$ about θ_2, respectively.
The probabilities of the above estimator's different outcomes have the form

$$p_1 = \left(\frac{1}{2}(1+\theta_i)\right)^2, \quad p_2 = 2\cdot\left(\frac{1}{2}(1+\theta_i)\right)\cdot\left(\frac{1}{2}(1-\theta_i)\right), \quad p_3 = \left(\frac{1}{2}(1-\theta_i)\right)^2$$

As it can be seen they are identical to the probability distribution of the 3 independent spin measurements of the first section using 2×3 qubits, i.e. measuring $\{+1,+1\}$, measuring $\{+1,-1\}$, or $\{-1,+1\}$, and finally, $\{-1,-1\}$.
The estimator works similarly:

$$\Phi_2^{MUB2}{}_i = \begin{cases} +1 & \text{if the outcome corresponds to } p_1 \\ 0 & \text{if the outcome corresponds to } p_2 \\ -1 & \text{if the outcome corresponds to } p_3 \end{cases}, i = 1,2,3.$$

The MSE matrix of this estimator is:

$$V_{3\times 2}(\theta) = \frac{1}{2}\begin{bmatrix} 1-\theta_1^2 & 0 & 0 \\ 0 & 1-\theta_2^2 & 0 \\ 0 & 0 & 1-\theta_3^2 \end{bmatrix}.$$

It seems, that this kind of estimator is (practically) identical to the first one.

3.2.7 Differences

The question is which estimation scheme is the most effective. To answer this question we first compare (3.1) and (3.5). To do this we need to use the same numbers of qubits in both cases. So the question is wether

$$V_n^{stand}(\theta) > V_{3\times r}(\theta)$$

or not? The difference $V_n^{stand}(\theta) - V_{3\times r}(\theta)$ has the form

$$\tfrac{1}{n}\begin{bmatrix} 3-\theta_1^2 & -\theta_1\theta_2 & -\theta_1\theta_3 \\ -\theta_1\theta_2 & 3-\theta_2^2 & -\theta_2\theta_3 \\ -\theta_1\theta_3 & -\theta_2\theta_3 & 3-\theta_3^2 \end{bmatrix} - \tfrac{3}{n}\begin{bmatrix} 1-\theta_1^2 & 0 & 0 \\ 0 & 1-\theta_2^2 & 0 \\ 0 & 0 & 1-\theta_3^2 \end{bmatrix} =$$

$$\tfrac{1}{n}\begin{bmatrix} 2\theta_1^2 & -\theta_1\theta_2 & -\theta_1\theta_3 \\ -\theta_1\theta_2 & 2\theta_2^2 & -\theta_2\theta_3 \\ -\theta_1\theta_3 & -\theta_2\theta_3 & 2\theta_3^2 \end{bmatrix} = \tfrac{1}{n}\cdot\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \circ \left(\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \cdot \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}\right)$$

where $\circ$ stands for the Hadamard product. Since the two matrices are positive, the difference is also, i.e. (3.1) is more effective, than (3.5).
To decide about the minimal qubit tomography (3.3) and the standard one (3.5) one has to check the positivity of $V_n^{min}(\theta) - V_n^{stand}(\theta)$:

$$V_n^{min}(\theta) - V_n^{stand}(\theta) = \tfrac{1}{n}\begin{bmatrix} 3-\theta_1^2 & \sqrt{3}\theta_3 - \theta_1\theta_2 & \sqrt{3}\theta_2 - \theta_1\theta_3 \\ \sqrt{3}\theta_3 - \theta_1\theta_2 & 3-\theta_2^2 & \sqrt{3}\theta_1 - \theta_2\theta_3 \\ \sqrt{3}\theta_2 - \theta_1\theta_3 & \sqrt{3}\theta_1 - \theta_2\theta_3 & 3-\theta_3^2 \end{bmatrix} -$$

$$-\tfrac{1}{n}\begin{bmatrix} 3-\theta_1^2 & -\theta_1\theta_2 & -\theta_1\theta_3 \\ -\theta_1\theta_2 & 3-\theta_2^2 & -\theta_2\theta_3 \\ -\theta_1\theta_3 & -\theta_2\theta_3 & 3-\theta_3^2 \end{bmatrix} = \tfrac{1}{n}\begin{bmatrix} 0 & \sqrt{3}\theta_3 & \sqrt{3}\theta_2 \\ \sqrt{3}\theta_3 & 0 & \sqrt{3}\theta_1 \\ \sqrt{3}\theta_2 & \sqrt{3}\theta_1 & 0 \end{bmatrix}$$

which is indefinite, so they cannot be compared.

3.3 Constrained state estimation

As it was shown so far the unconstrained estimator (3.1) has nice statistical properties however, a regularization step would be useful in order to avoid faulty estimates possibly due to small number of measurements, or a state being on the boundary of the state space (for 2-level systems they are the pure states).

3.3.1 Geometrical constraint

For the qubit case, using the Bloch parametrization, the positive semi-definite density matrices correspond to the Bloch vectors of length less than, or equal to 1. The word *faulty estimate* means a Bloch vector being out of the Bloch ball, i.e. longer, than 1.
The simplest way of restricting the estimator to the unit ball is

$$\Phi_n^{\mathrm{con}}(\theta) = \begin{cases} \Phi_n(\theta), & \text{if } ||\Phi_n(\theta)|| \leq 1 \\ \\ \frac{\Phi_n(\theta)}{||\Phi_n(\theta)||}, & \text{if } ||\Phi_n(\theta)|| > 1 \end{cases} \tag{3.7}$$

Practically, it means that the Bloch vector is shrunk onto the boundary of the Bloch-vector space (the ball in this case).

In the N-level case [1], the Bloch-vector space is a proper convex, asymmetric subset of a ball in $\mathbb{R}^{N^2-1}$ having radius

$$\sqrt{\frac{2(N-1)}{N}}.$$

It means, that the constraining cannot be handled as easy as in the 2-level case. The structure of states is also more difficult since the pure state - mixed state distinction is not so straightforward. Instead, the concept invertible and non-invertible states will be used. Clearly, the set of pure states is identical to the set of non-invertible states only for 2-level systems - in higher dimensions the pure states are a proper subset of the non-invertible states. The boundary of the Bloch-vector space contains non-invertible states.
The idea is the same as in the qubit case, i.e. to replace the wrong estimates with states being on the boundary. Formally, the estimator is

$$\Phi_n^{\mathrm{con}}(\theta) = \begin{cases} \Phi_n(\theta), & \text{if } \Phi_n(\theta) \in \mathcal{B} \\ \\ \Gamma_n(\theta), & \text{otherwise} \end{cases} \tag{3.8}$$

where $\mathcal{B}$ stands for the Bloch-vector space. $\Gamma_n(\theta)$ is determined similarly to the simple case, i.e. the Bloch vector estimate is replaced with the Bloch vector having the same direction and being on the boundary of $\mathcal{B}$. It can be found easily, since the density matrices are positive definite inside $\mathcal{B}$, positive semi-definite on the boundary, and indefinite outside $\mathcal{B}$. It is enough to find the Bloch vector (having a given direction) for which the corresponding density matrix's minimal eigenvalue changes its sign. Using a bisector algorithm, the boundary of $\mathcal{B}$ in a given direction can be found easily.

3.3.2 Eigenvalue-based constraint

The other way of constraining the estimate onto $\mathcal{B}$ does not use the Bloch-vector representation. If the matrix corresponding to the estimate has negative eigenvalue, it is outside the set of density matrices.
The method works for diagonal matrices, so the basis should be chosen according to it

$$\rho(\Phi_n) = U^* \mathrm{Diag}(\lambda_1, \ldots, \lambda_n) U.$$

If the faulty estimate has k negative values in the diagonal form: $\lambda_1, \ldots, \lambda_k < 0$ and $\lambda_{k+1}, \ldots, \lambda_n \geq 0$, then define $\mathrm{Diag}(\mu_1, \ldots, \mu_n)$ such that:

$$\begin{aligned} \mu_i &= 0, \quad i = 1, \ldots, k \\ \mu_j &= \lambda_j + c, \quad j = k+1, \ldots, n, \quad c = \tfrac{1}{n-k} \textstyle\sum_{i=1}^{k} \lambda_i \end{aligned}$$

After finitely many steps, the algorithm results in a proper density matrix, and the constrained estimate is

$$\rho(\Phi_n) = U^* \mathrm{Diag}(\mu_1, \ldots, \mu_n) U.$$

3.3.3 Properties of the constrained estimators

Although they are slightly different in the method of computing the estimate, they perform very similarly in practice. The main drawback of the constrained estimators compared to the unconstrained one is that they are not unbiased, only in the asymptotic sense.

Chapter 4

Simulation results

4.1 Simulation environment

The simulations were performed in Matlab environment. The probabilistic behavior of the measurement was simulated by the built-in random number generator of Matlab. The simulator is also capable to simulate the dynamical change of 2, 3, and 4 level quantum systems but as it was assumed previously, the dynamics has not been taken into account.
The measurement data is collected once per experiment, i.e. the different estimation methods used the same data. Increasing the number of measurements the data of the previous measurements was supplemented with the data of new measurements.

4.2 Comparison of unconstrained and constrained estimation schemes

4.2.1 Measures of goodness

The quantities carrying information about the quality of the estimation are the *fidelity* and the L_2 *distance* between the estimate and the real state.

- *The fidelity* between two density matrices ρ and σ is

$$F(\rho, \sigma) = \mathrm{Tr}(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}});$$

  Note, that $F(\rho, \sigma)$ is 1, if the 2 density is the same and 0 if the are the the most different. For indefinite matrices, the fidelity can be greater, than 1, and complex as well.

- L_2 *distance* between two Bloch-vectors r and s is

$$d(r, s) = |r - s| = \sqrt{\sum_{i=1}^{m} (r_i - s_i)^2}.$$

4.2.2 Three level system

Two pairs of experiments were performed: one for estimating a mixed state and another for a rank-deficient state.

Invertible state

In the first case an invertible state was estimated based on different numbers of measurements $n = 8 \times r$. The state was

$$\rho = \begin{pmatrix} 0.3170 & -0.2136 - 0.0182i & 0.0483 - 0.0535i \\ -0.2136 + 0.0182i & 0.4048 & 0.0354 + 0.0398i \\ 0.0483 + 0.0535i & 0.0354 - 0.0398i & 0.2782 \end{pmatrix}$$

with eigenvalues $\lambda_1 = 0.1186, \lambda_2 = 0.2871, \lambda_3 = 0.5943$. Another density having the same eigenvalues was also used:

$$\sigma = \begin{pmatrix} 0.2443 & 0.0921 + 0.0346i & 0.0716 + 0.00391i \\ 0.0921 - 0.0346i & 0.2244 & 0.0612 - 0.0929i \\ 0.0716 - 0.0391i & 0.0612 + 0.0929i & 0.5313 \end{pmatrix}$$

Figures 4.1-4.4 shows the L_2 norm and the fidelity between the real state and the estimate for various numbers (n) of measurements.
It is apparent in figure 4.1 that although a mixed state is estimated, for a small number of measurements the unconstrained estimator $\Phi^{un}(\rho)$ gives indefinite estimates. In these cases the constrained estimator uses the nearest rank-deficient state (from the boundary of the Bloch-vector space). After $n \approx 200$ (i.e. $r \approx 25$) measurements, the unconstrained estimates lay inside the state space so the constrained estimator "switches" to the unconstrained one.

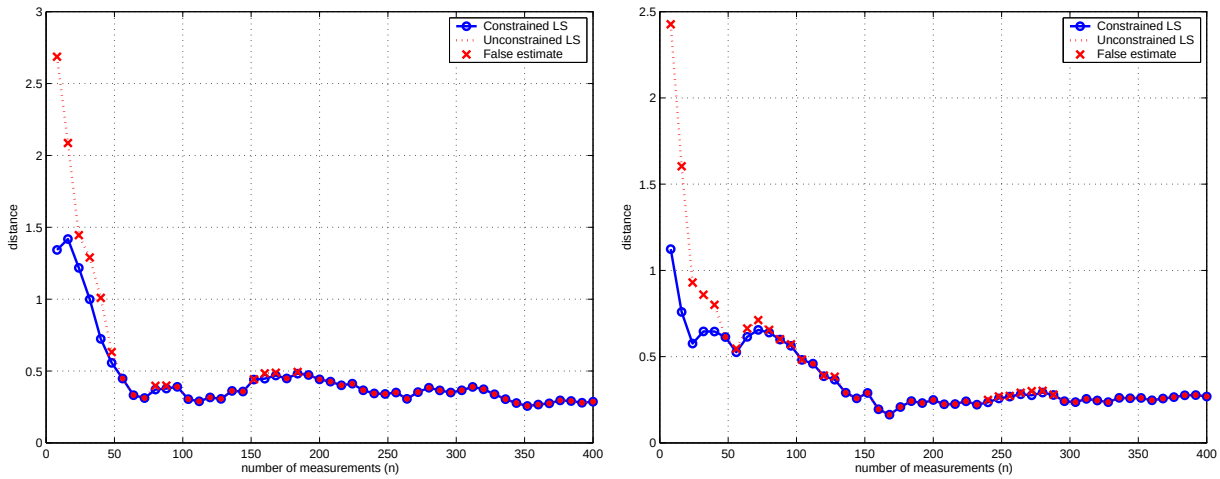

Figure 4.1: L_2 distance as a function of n for a mixed state ρ (left), and σ (right)

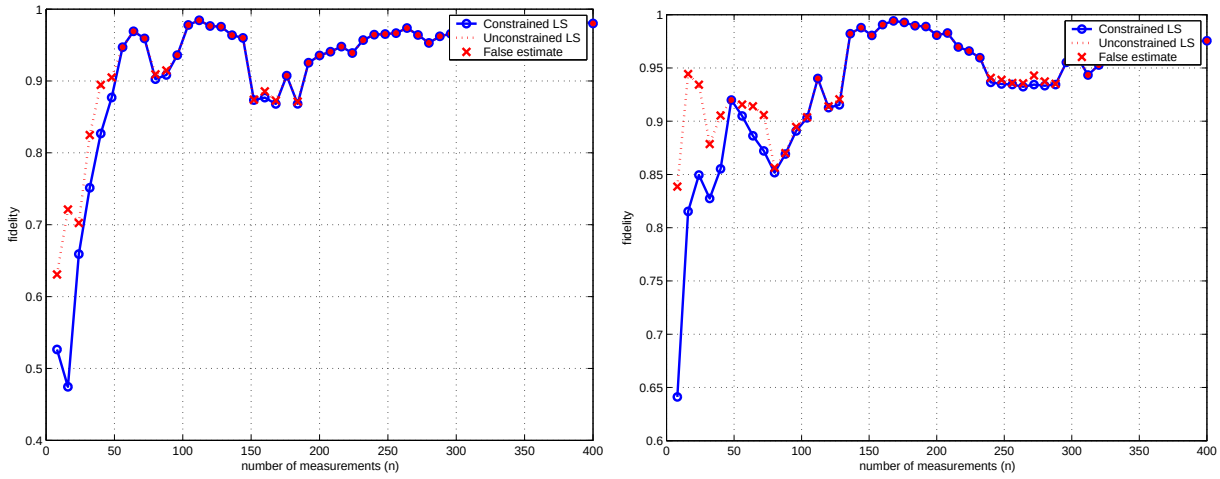

Figure 4.2: Fidelity as a function of n for a mixed state ρ (left), and σ (right)

Figures 4.3 and 4.4 shows the results for a longer interval of n, they tells us the same as the previous two pictures.

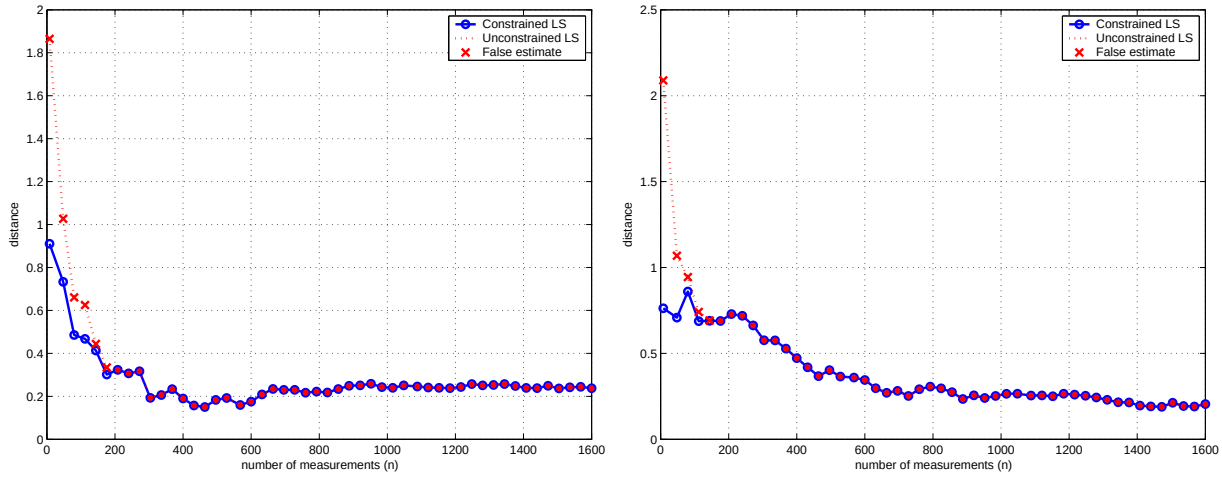

Figure 4.3: L_2 distance as a function of n for a mixed state ρ (left), and σ (right)

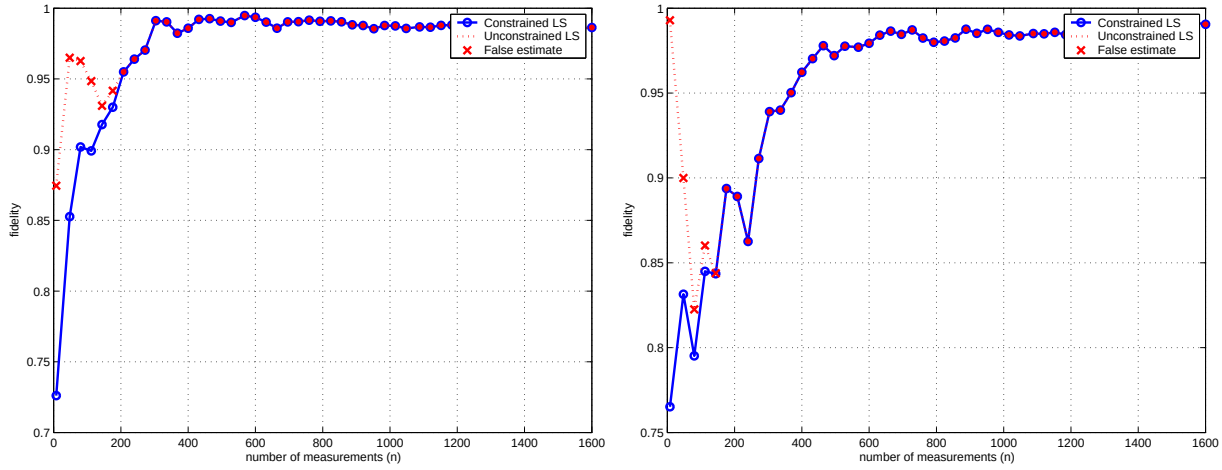

Figure 4.4: Fidelity as a function of n for a mixed state ρ (left), and σ (right)

Rank-deficient case

For the rank-deficient case the results are expected to be different from the previous case since the probability of an indefinite estimate of the unconstrained estimator does not vanish so fast as in the case of a mixed state. The state to be estimated was

$$\rho = \begin{pmatrix} 0.5588 & 0.3049 + 0.0963i & -0.1087 + 0.0055i \\ 0.3049 - 0.0963i & 0.3823 & 0.0273 + 0.0347i \\ -0.1087 - 0.0055i & 0.0273 - 0.0347i & 0.0589 \end{pmatrix}$$

with eigenvalues $\lambda_1 = 0; \lambda_2 = 0.1886, \lambda_3 = 0.8114$. Another density matrix with the same eigenvalues was used:

$$\sigma = \begin{pmatrix} 0.2124 & -0.2795 - 0.1155i & -0.1226 - 0.0702i \\ -0.2795 + 0.1155i & 0.4395 & 0.2362 - 0.0041i \\ -0.1226 + 0.0702i & 0.2362 + 0.0041i & 0.3481 \end{pmatrix}$$

Figures 4.5-4.8 shows the results of the experiment. Figure 4.5 verifies our expectations since the unconstrained estimator gives a meaningful estimate only a few cases. So the constrained estimator is seldom allowed to use the constrained estimate, it has to use a state from the boundary of the Bloch-vector space. As for the fidelity (figure 4.6), the curve of the unconstrained estimator does not hold any information since it is greater 1 and/or complex. Bit the constrained estimator's one approaches 1 quite well.

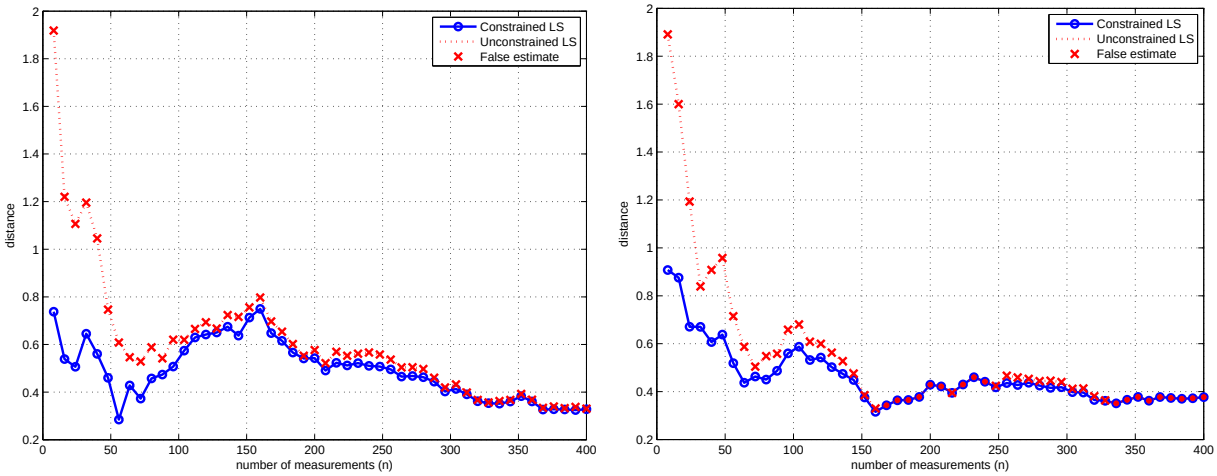

Figure 4.5: L_2 distance as a function of n for a rank deficient state ρ (left), and σ (right)

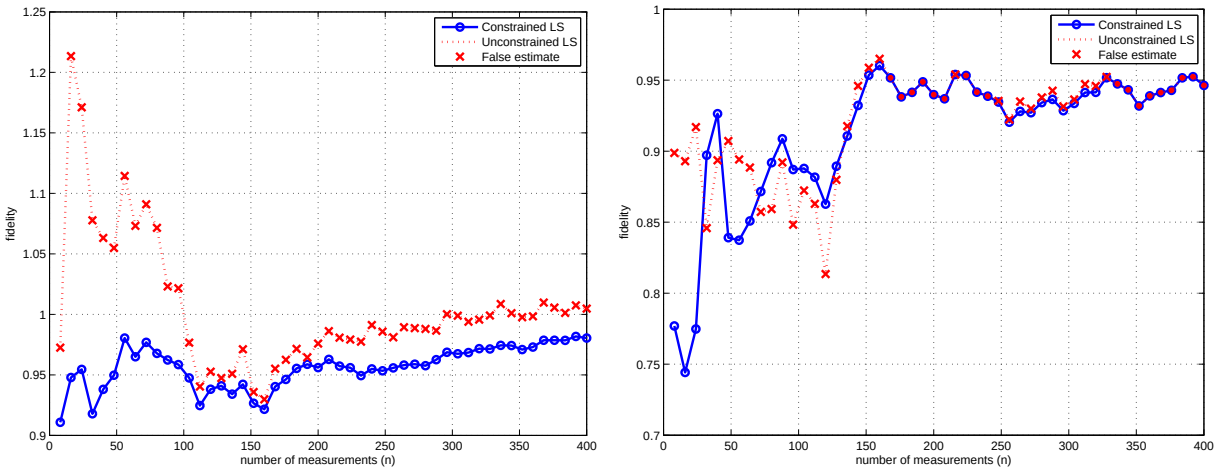

Figure 4.6: Fidelity as a function of n for a rank deficient state ρ (left), and σ (right)

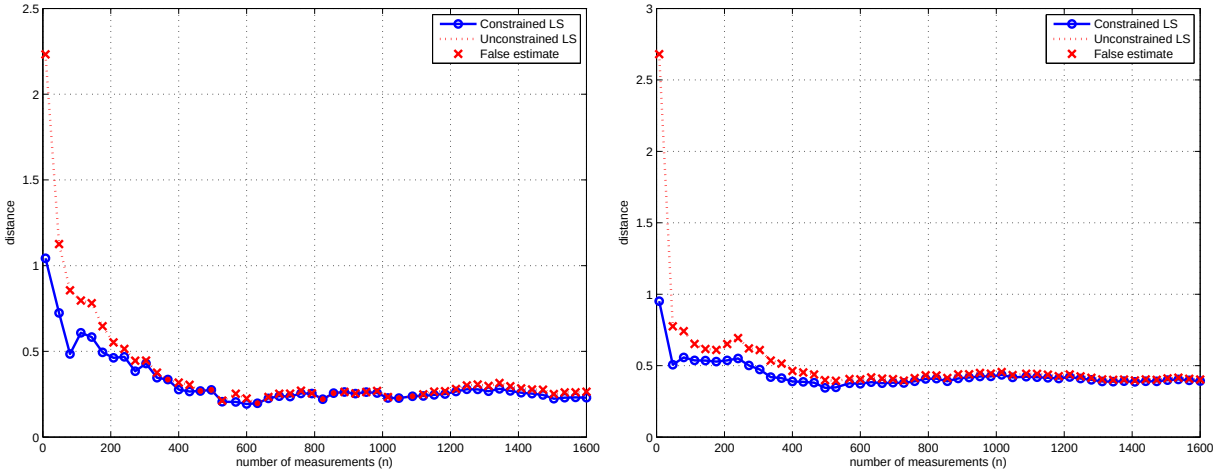

Figure 4.7: L_2 distance as a function of n for a rank deficient state ρ (left), and σ (right)

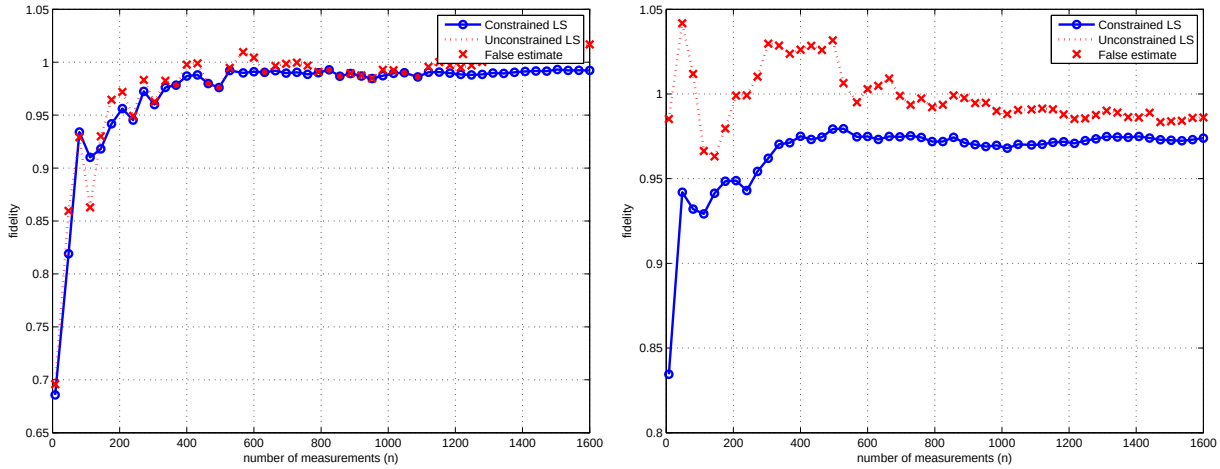

Figure 4.8: Fidelity as a function of r for a rank deficient state ρ (left), and σ (right)

Chapter 5

Conclusions

Essentially two different kinds state estimators for quantum mechanical systems were examined: an unconstrained one, and a constrained estimator. The unconstrained estimator is unbiased, effective, and last but not least is easy to compute. The only drawback of it is the possibility of resulting in a state estimate without any physically meaning.
On the other hand, the constrained estimator is better in the sense that it maps exactly to the state space, but it's statistical properties are not so fortunate, e.g. it is just asymptotically unbiased.
Computer simulations verified the theoretical results. For small number of measurements, or estimating a state being near the boundary of the state space the two estimators give different results, but as the number of measurements increase, the estimate of a relatively mixed state is the same for both methods. It means that the constrained estimator should be used only in circumstances when the number of measurements should be minimal, e.g. the price of measurement is high, or when the states to be estimated are non invertible states. However, if it is known, that the state to be estimated, completely different (simpler) methods should be used.

Bibliography

[1] Gen Kimura. The bloch vector for n-level systems. *arXiv:quant-ph/0301152 v2*, 2003.

[2] J. Rehácek, B. Englert, and D. Kaszlikowski. Minimal quit tomography. *Physical Review A*, 70:052321, 2004.

[3] W.K. Wooters and B.D. Fields. Optimal state determination by mutually unbiased measurements. *Annals of Physics*, 191:363–381, 1989.

Chapter 6

Appendix

6.1 Matlab source

The source of the most important Matlab functions are given below. The simulation results as well as the attached figures are created with these functions.

6.1.1 Function FidelPlot.m

```
function FidelPlot3LSmod(D,n) global de de=D;
%
% Syntax:
% FidelPlot3LS(D,n)
%
% D -   3x3 density matrix
% n -   grid points (the values of the elements of n must be positive integers between 1 and 1000)

% error handling
% checking 'D'
if (size(D) == [3 3])
    if(D-D' > 1e-6)
        error('Input argument "D" is not a density matrix! It should be self adjoint!')
    end
    if((trace(D) > 1+1e-5) || (trace(D) < 1-1e-5))
        error('Input argument "D" is not a density matrix! It should be of trace 1!')
    end
elseif(size(D) ~= [3 3])
    error('Input argument "rho" must be a 3x3 density matrix!');
end

% n
if min(size(n)) ~= 1
    error('Input argument n should be a vector with positive integer entries!');
end
n = ceil(n);    % rounding up the values
n = sort(n);    % sorting the elements
if min(n) < 1
    error('The elements of n must be positive integers between 1 and 1000!');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The measurement record should be the same for both cases so the
% measurement is performed here:
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The projection matrices of the observables Aii, Bij, and Cij for the measurements
% A11:
A11_p = [1 0 0;0 0 0;0 0 0]; A11_n = [0 0 0;0 0 0;0 0 0]; A11_z = [0
0 0;0 1 0;0 0 1];
% A22:
A22_p = [0 0 0;0 1 0;0 0 0]; A22_n = [0 0 0;0 0 0;0 0 0]; A22_z = [1
0 0;0 0 0;0 0 1];
% A33:
A33_p = [0 0 0;0 0 0;0 0 1]; A33_n = [0 0 0;0 0 0;0 0 0]; A33_z = [1
0 0;0 1 0;0 0 0];
% B12:
B12_p = 0.5*[1 1 0;1 1 0;0 0 0]; B12_n = 0.5*[1 -1 0;-1 1 0;0 0 0];
B12_z = [0 0 0;0 0 0;0 0 1];
% B13:
B13_p = 0.5*[1 0 1;0 0 0;1 0 1]; B13_n = 0.5*[1 0 -1;0 0 0;-1 0 1];
B13_z = [0 0 0;0 1 0;0 0 0];
% B23:
B23_p = 0.5*[0 0 0;0 1 1;0 1 1]; B23_n = 0.5*[0 0 0;0 1 -1;0 -1 1];
B23_z = [1 0 0;0 0 0;0 0 0];
% C12:
C12_p = 0.5*[1 -i 0;i 1 0;0 0 0]; C12_n = 0.5*[1 i 0;-i 1 0;0 0 0];
C12_z = [0 0 0;0 0 0;0 0 1];
% C13:
C13_p = 0.5*[1 0 -i;0 0 0;i 0 1]; C13_n = 0.5*[1 0 i;0 0 0;-i 0 1];
C13_z = [0 0 0;0 1 0;0 0 0];
% B23:
C23_p = 0.5*[0 0 0;0 1 -i;0 i 1]; C23_n = 0.5*[0 0 0;0 1 i;0 -i 1];
C23_z = [1 0 0;0 0 0;0 0 0];
%
meas_A11=[]; meas_A22=[]; meas_A33=[]; meas_B12=[]; meas_B13=[];
meas_B23=[]; meas_C12=[]; meas_C13=[]; meas_C23=[];
%
for j=1:n(end)                          % Measurements
    [d_f,rA11]=measure3(D,A11_p,A11_n,A11_z);
    [d_f,rA22]=measure3(D,A22_p,A22_n,A22_z);
    [d_f,rA33]=measure3(D,A33_p,A33_n,A33_z);
    [d_f,rB12]=measure3(D,B12_p,B12_n,B12_z);
    [d_f,rB13]=measure3(D,B13_p,B13_n,B13_z);
    [d_f,rB23]=measure3(D,B23_p,B23_n,B23_z);
    [d_f,rC12]=measure3(D,C12_p,C12_n,C12_z);
    [d_f,rC13]=measure3(D,C13_p,C13_n,C13_z);
    [d_f,rC23]=measure3(D,C23_p,C23_n,C23_z);
    meas_A11=[meas_A11 , rA11];
    meas_A22=[meas_A22 , rA22];
    meas_A33=[meas_A33 , rA33];
    meas_B12=[meas_B12 , rB12];
    meas_B13=[meas_B13 , rB13];
    meas_B23=[meas_B23 , rB23];
    meas_C12=[meas_C12 , rC12];
    meas_C13=[meas_C13 , rC13];
    meas_C23=[meas_C23 , rC23];
end
```

```
warning off MATLAB:sqrtm:SingularMatrix     % turn off warning for singular matrices

% Making a process bar to show the remaining time:
av = 1;% the averaging window size
s=[]; for k=1:av*max(size(n))
    s = [s '_'];
end
fprintf('\n%s\n',s);

warning off MATLAB:sqrtm:SingularMatrix

fid_ls = []; fid_lsb = []; fid_lsu = []; fid_lsc = [];
%
fidim_ls = []; fidim_lsb = []; fidim_lsu = [];
%
L2_ls = []; L2_lsb = []; L2_lsu = []; L2_lsc = [];
%
BL_ls = []; BL_lsb = []; BL_lsu = []; BL_lsc = [];
%
me_ls = []; me_lsb = []; me_lsu = [];
%

for i = 1:max(size(n))
    ls_f=0;lsb_f=0;lsu_f=0;lsc_f=0;
    ls_fim=0;lsb_fim=0;lsu_fim=0;
    ls_L2=0;lsb_L2=0;lsu_L2=0;lsc_L2=0;
    ls_BL=0;lsb_BL=0;lsu_BL=0;lsc_BL=0;
    ls_me=0;lsb_me=0;lsu_me=0;
    %
    for j=1:av   % we are averaging av experiments
        % The function experiment is included here:
        dat{j} = [];  % structure meas will contain all the data about the experiments

        % Data on the measurement situation
        %D = vector2density(bloch);
        dat{j}.NumberOfMeasurement = n(i);

        % Data of the LS estimation
        [D_ls] = qls3(n(i),D,meas_A11(:,1:n(i)),meas_A22(:,1:n(i)),meas_A33(:,1:n(i)),meas_B12(:,1:n(i
        dat{j}.LSDens = D_ls;
        %dat{j}.LSVariance = sigma_ls;
        dat{j}.LSFidelity = fidelity3(D,D_ls);
        dat{j}.LSFidelity_imag = fidelity3imag(D,D_ls);
        dat{j}.LSNormL2 = norm(real(D-D_ls))+norm(imag(D-D_ls));
        dat{j}.LSMinEig = min(real(eig(D_ls)));
        dat{j}.LSNormBL = norm((density2vector3(D)-density2vector3(D_ls)));

        % Data of the unconstrained LS estimation
        [D_lsu] = qls3_un(n(i),D,meas_A11(:,1:n(i)),meas_A22(:,1:n(i)),meas_A33(:,1:n(i)),meas_B12(:,1
        dat{j}.LSUDens = D_lsu;
        %dat{j}.LSUVariance = sigma_lsu;
        dat{j}.LSUFidelity = fidelity3(D,D_lsu);
        dat{j}.LSUFidelity_imag = fidelity3imag(D,D_lsu);
        dat{j}.LSUNormL2 = norm(real(D-D_lsu))+norm(imag(D-D_lsu));
        dat{j}.LSUMinEig = min(real(eig(D_lsu)));
```

```
    dat{j}.LSUNormBL = norm((density2vector3(D)-density2vector3(D_lsu)));

    % Data of the second type constrained LS estimation
    [D_lsb] = qls3b(n(i),D,meas_A11(:,1:n(i)),meas_A22(:,1:n(i)),meas_A33(:,1:n(i)),meas_B12(:,1:n
    dat{j}.LSBDens = D_lsb;
    %dat{j}.LSUVariance = sigma_lsu;
    dat{j}.LSBFidelity = fidelity3(D,D_lsb);
    dat{j}.LSBFidelity_imag = fidelity3imag(D,D_lsb);
    dat{j}.LSBNormL2 = norm(real(D-D_lsb))+norm(imag(D-D_lsb));
    dat{j}.LSBMinEig = min(real(eig(D_lsb)));
    dat{j}.LSBNormBL = norm((density2vector3(D)-density2vector3(D_lsb)));

    % Data of the always constrained  LS estimation (nearest pure state)
    [D_lsc] = qls3_con(n(i),D,meas_A11(:,1:n(i)),meas_A22(:,1:n(i)),meas_A33(:,1:n(i)),meas_B12(:,
    dat{j}.LSCDens = D_lsc;
    %dat{j}.LSVariance = sigma_ls;
    dat{j}.LSCFidelity = fidelity3(D,D_lsc);
    dat{j}.LSCFidelity_imag = fidelity3imag(D,D_lsc);
    dat{j}.LSCNormL2 = norm(real(D-D_lsc))+norm(imag(D-D_ls));
    dat{j}.LSCMinEig = min(real(eig(D_lsc)));
    dat{j}.LSCNormBL = norm((density2vector3(D)-density2vector3(D_lsc)));

    %dat{j} = experiment3(n(i),D);    % performs an experiment with n(i) measurements
    % fidelity
    ls_f = ls_f+dat{j}.LSFidelity;
    lsu_f = lsu_f+dat{j}.LSUFidelity;
    lsb_f = lsb_f+dat{j}.LSBFidelity;
    lsc_f = lsc_f+dat{j}.LSCFidelity;
    % imag of fidelity
    ls_fim = ls_fim+dat{j}.LSFidelity_imag;
    lsu_fim = lsu_fim+dat{j}.LSUFidelity_imag;
    lsb_fim = lsb_fim+dat{j}.LSBFidelity_imag;
    % minimal eigenvalue of the estimate
    ls_me = ls_me+dat{j}.LSMinEig;
    lsu_me = lsu_me+dat{j}.LSUMinEig;
    lsb_me = lsb_me+dat{j}.LSBMinEig;
    % L2 norm
    ls_L2 = ls_L2+dat{j}.LSNormL2;
    lsu_L2 = lsu_L2+dat{j}.LSUNormL2;
    lsb_L2 = lsb_L2+dat{j}.LSBNormL2;
    lsc_L2 = lsc_L2+dat{j}.LSCNormL2;
    % Bloch vector norm
    ls_BL = ls_BL+dat{j}.LSNormBL;
    lsu_BL = lsu_BL+dat{j}.LSUNormBL;
    lsb_BL = lsb_BL+dat{j}.LSBNormBL;
    lsc_BL = lsc_BL+dat{j}.LSCNormBL;
    fprintf('*');   % Updating the process bar
end
% fidelity
fid_ls = [fid_ls ls_f/av];
fid_lsu = [fid_lsu lsu_f/av];
fid_lsb = [fid_lsb lsb_f/av];
fid_lsc = [fid_lsc lsc_f/av];
% imagfidelity
fidim_ls = [fidim_ls ls_fim/av];
```

```
    fidim_lsu = [fidim_lsu lsu_fim/av];
    fidim_lsb = [fidim_lsb lsb_fim/av];
    % minimal eigenvaule
    me_ls = [me_ls ls_me/av];
    me_lsu = [me_lsu lsu_me/av];
    me_lsb = [me_lsb lsb_me/av];
    % L2 norm
    L2_ls = [L2_ls ls_L2/av];
    L2_lsu = [L2_lsu lsu_L2/av];
    L2_lsb = [L2_lsb lsb_L2/av];
    L2_lsc = [L2_lsc lsc_L2/av];
    % Bloch vector norm
    BL_ls = [BL_ls ls_BL/av];
    BL_lsu = [BL_lsu lsu_BL/av];
    BL_lsb = [BL_lsb lsb_BL/av];
    BL_lsc = [BL_lsc lsc_BL/av];
    % finding the faulty estimates of the unconstrained estimator
    err = [];
    fid_err = [];
    L2_err = [];
    BL_err = [];
    for k=1:max(size(me_lsu))
        if me_lsu(k) < 1e-6
            err = [err n(k)];
            fid_err = [fid_err fid_lsu(k)];
            L2_err = [L2_err L2_lsu(k)];
            BL_err = [BL_err BL_lsu(k)];
        end
    end
    %
    % finding the good estimates of the unconstrained estimator
    good = [];
    fid_good = [];
    L2_good = [];
    BL_good = [];
    for k=1:max(size(me_lsu))
        if me_lsu(k) > 1e-6
            good = [good n(k)];
            fid_good = [fid_good fid_lsu(k)];
            L2_good = [L2_good L2_lsu(k)];
            BL_good = [BL_good BL_lsu(k)];
        end
    end
    %
    dat=[];
end

% Plotting the fidelity as a function of the number of measurements
figure('Name','Fidelity as the function of the number of
measurements');
%plot(n,fid_lsc,'-go','LineWidth',2,'MarkerFaceColor','g');
%hold on
plot(8*n,fid_ls,'-bo','LineWidth',2); hold on
plot(8*n,fid_lsb,'-go','LineWidth',2);
plot(8*n,fid_lsu,':r','LineWidth',2);%,'MarkerFaceColor','r');
```

```
plot(8*err,fid_err,'rx','LineWidth',2,'MarkerSize',9);
plot(8*good,fid_good,'ro','MarkerSize',4,'MarkerFaceColor','r');
legend('Constrained LS','Constrained LS #2','Unconstrained
LS','False estimate') grid on xlabel('number of measurements (n)');
ylabel('fidelity');
%title('Fidelity as a function of the number of measurements');
print -depsc fidelity

% Plotting the Bloch vector length as a function of the number of measurements
figure('Name','Distance as the function of the number of
measurements');
%plot(n,BL_lsc,'-go','LineWidth',2,'MarkerFaceColor','g');
%hold on
plot(8*n,BL_ls,'-bo','LineWidth',2); hold on
plot(8*n,BL_lsb,'-go','LineWidth',2);
plot(8*n,BL_lsu,':r','LineWidth',2);%,'MarkerFaceColor','r');
plot(8*err,BL_err,'rx','LineWidth',2,'MarkerSize',9);
plot(8*good,BL_good,'ro','MarkerSize',4,'MarkerFaceColor','r');
legend('Constrained LS','Constrained LS #2','Unconstrained
LS','False estimate') grid on xlabel('number of measurements (n)');
ylabel('distance');
%title('distance as a function of the number of measurements');
print -depsc blochlength

%end of file
%Magyar Attila
```

6.1.2 Function qls.m

```
function [est] =
qls3_con(n,D,meas_A11,meas_A22,meas_A33,meas_B12,meas_B13,meas_B23,meas_C12,meas_C13,meas_C23)
global sigma global direction
%Function qls performs a least squares parameter estimation on a quantum
%system (spin 1/2) having no dynamics. The function computes and plots the
%least squares point estimates for the three spin component Sx, Sy and Sz.
%
%Syntax:
%est = qls(n,D)
%n          - number of measurements
%D          - density matrix before the measurement
%est        - estimated Bloch-vector
%
%Important! The function needs puredist3.m, and puredist3_con.m for the
%optimization phase.

%d=D;
 %d_bloch=density2vector(d);

% A11
pA11 = 0; pA11_rel = 0; nA11 = 0; for j = 1:size(meas_A11,2)
    if meas_A11(j)==1
        pA11 = pA11+1;
    elseif meas_A11(j)==-1
        nA11 = nA11+1;
    end
end
```

```
% pA11: number of +1 measurements
% nA11: number of -1 measurements (it is always 0)
% zA11: number of 0 measurements
zA11 = n-pA11-nA11; pA11_rel = pA11 / n; nA11_rel = nA11/n; zA11_rel
= zA11/n;
% pB12_rel: relative frequency of +1 measurements
% nB12_rel: relative frequency of -1 measurements
% zB12_rel: relative frequency of 0 measurements
s11_re = pA11_rel;
%s33_re = zA11_rel;
% x=(pB12+1)/(n+2);
% sigma_x2=x*(1-x)/(n+3);

% A22
pA22 = 0; pA22_rel = 0; nA22 = 0; for j = 1:size(meas_A22,2)
    if meas_A22(j)==1
        pA22 = pA22+1;
    elseif meas_A22(j)==-1
        nA22 = nA22+1;
    end
end
% pA22: number of +1 measurements
% nA22: number of -1 measurements (it is always 0)
% zA22: number of 0 measurements
zA22 = n-pA22-nA22; pA22_rel = pA22 / n; nA22_rel = nA22/n; zA22_rel
= zA22/n;
% pA22_rel: relative frequency of +1 measurements
% nA22_rel: relative frequency of -1 measurements
% zA22_rel: relative frequency of 0 measurements
s22_re = pA22_rel;
%s33_re = zA11_rel;
% x=(pB12+1)/(n+2);
% sigma_x2=x*(1-x)/(n+3);

% A33
pA33 = 0; pA33_rel = 0; nA33 = 0; for j = 1:size(meas_A33,2)
    if meas_A33(j)==1
        pA33 = pA33+1;
    elseif meas_A33(j)==-1
        nA33 = nA33+1;
    end
end
% pA33: number of +1 measurements
% nA33: number of -1 measurements (it is always 0)
% zA33: number of 0 measurements
zA33 = n-pA33-nA33; pA33_rel = pA33 / n; nA33_rel = nA33/n; zA33_rel
= zA33/n;
% pA33_rel: relative frequency of +1 measurements
% nA333_rel: relative frequency of -1 measurements
% zA33_rel: relative frequency of 0 measurements
s33_re = pA33_rel;
%s33_re = zA11_rel;
% x=(pB12+1)/(n+2);
% sigma_x2=x*(1-x)/(n+3);
```

```
% B12
pB12 = 0; pB12_rel = 0; nB12 = 0; for j = 1:size(meas_B12,2)
    if meas_B12(j)==1
        pB12 = pB12+1;
    elseif meas_B12(j)==-1
        nB12 = nB12+1;
    end
end
% pB12: number of +1 measurements
% nB12: number of -1 measurements
% zB12: number of 0 measurements
zB12 = n-pB12-nB12; pB12_rel = pB12 / n; nB12_rel = nB12/n; zB12_rel
= zB12/n;
% pB12_rel: relative frequency of +1 measurements
% nB12_rel: relative frequency of -1 measurements
% zB12_rel: relative frequency of 0 measurements
s12_re = 0.5*(pB12_rel - nB12_rel);
%s33_re = zB12_rel;
% x=(pB12+1)/(n+2);
% sigma_x2=x*(1-x)/(n+3);


% B13
pB13 = 0; nB13 = 0; for j = 1:size(meas_B13,2)
    if meas_B13(j)==1
        pB13 = pB13+1;
    elseif meas_B13(j)==-1
        nB13 = nB13+1;
    end
end
% pB13: number of +1 measurements
% nB13: number of -1 measurements
% zB13: number of 0 measurements
zB13 = n-pB13-nB13; pB13_rel = pB13/n; nB13_rel = nB13/n; zB13_rel =
zB13/n;
% pB13_rel: relative frequency of +1 measurements
% nB13_rel: relative frequency of -1 measurements
% zB13_rel: relative frequency of 0 measurements
s13_re = 0.5*(pB13_rel - nB13_rel);
%s22_re = zB13_rel;
% y=(py+1)/(n+2);
% sigma_y2=y*(1-y)/(n+3);

% B23
pB23 = 0; nB23 = 0; for j = 1:size(meas_B23,2)
    if meas_B23(j)==1
        pB23 = pB23+1;
    elseif meas_B23(j)==-1
        nB23 = nB23+1;
    end
end
% pB23: number of +1 measurements
% nB23: number of -1 measurements
% zB23: number of 0 measurements
zB23 = n-pB23-nB23; pB23_rel = pB23/n; nB23_rel = nB23/n; zB23_rel =
```

```
zB23/n;
% pB23_rel: relative frequency of +1 measurements
% nB23_rel: relative frequency of -1 measurements
% zB23_rel: relative frequency of 0 measurements
s23_re = 0.5*(pB23_rel - nB23_rel);
%s11_re = zB23_rel;
% z=(pz+1)/(n+2);
% sigma_z2=z*(1-z)/(n+3);

% C12
pC12 = 0; nC12 = 0; for j = 1:size(meas_C12,2)
    if meas_C12(j)==1
        pC12 = pC12+1;
    elseif meas_C12(j)==-1
        nC12 = nC12+1;
    end
end
% pC12: number of +1 measurements
% nC12: number of -1 measurements
% zC12: number of 0 measurements
zC12 = n-pC12-nC12; pC12_rel = pC12/n; nC12_rel = nC12/n; zC12_rel =
zC12/n;
% pC12_rel: relative frequency of +1 measurements
% nC12_rel: relative frequency of -1 measurements
% zC12_rel: relative frequency of 0 measurements
s12_im = 0.5*(pC12_rel - nC12_rel);
% x=(pB12+1)/(n+2);
% sigma_x2=x*(1-x)/(n+3);


% C13
pC13 = 0; nC13 = 0; for j = 1:size(meas_C13,2)
    if meas_C13(j)==1
        pC13 = pC13+1;
    elseif meas_C13(j)==-1
        nC13 = nC13+1;
    end
end
% pC13: number of +1 measurements
% nC13: number of -1 measurements
% zC13: number of 0 measurements
zC13 = n-pC13-nC13; pC13_rel = pC13/n; nC13_rel = nC13/n; zC13_rel =
zC13/n;
% pC13_rel: relative frequency of +1 measurements
% nC13_rel: relative frequency of -1 measurements
% zC13_rel: relative frequency of 0 measurements
s13_im = 0.5*(pC13_rel - nC13_rel);
% y=(py+1)/(n+2);
% sigma_y2=y*(1-y)/(n+3);

% C23
pC23 = 0; nC23 = 0; for j = 1:size(meas_C23,2)
    if meas_C23(j)==1
        pC23 = pC23+1;
    elseif meas_C23(j)==-1
```

```
        nC23 = nC23+1;
    end
end
% pC23: number of +1 measurements
% nC23: number of -1 measurements
% zC23: number of 0 measurements
zC23 = n-pC23-nC23; pC23_rel = pC23/n; nC23_rel = nC23/n; zC23_rel =
zC23/n;
% pC23_rel: relative frequency of +1 measurements
% nC23_rel: relative frequency of -1 measurements
% zC23_rel: relative frequency of 0 measurements
s23_im = 0.5*(pC23_rel - nC23_rel);
% z=(pz+1)/(n+2);
% sigma_z2=z*(1-z)/(n+3);

%sigma_est = [sigma_x2;sigma_y2;sigma_z2];    % Giving back the variance


sigma = [s11_re   s12_re-i*s12_im   s13_re-i*s13_im ;...
       s12_re+i*s12_im   s22_re   s23_re-i*s23_im ;...
       s13_re+i*s13_im   s23_re+i*s23_im   1- s11_re - s22_re];

%constraint:
s = density2vector3(sigma); direction = 1/norm(s)*s; x =
bisector(direction,0,2/sqrt(3),1e-6); est = vector2density3(x);

% Magyar Attila
%end of file
```