Quantum Software Models: Quantum Modules Tomography and Recovery Theorem

Iaakov Exman
School of Computer Science, Faculty of Sciences
HIT – Holon Institute of Technology
Holon, Israel
iaakov@hit.ac.il

Ariel Zvulunov
Software Engineering
The Jerusalem College of Engineering, Azrieli
Jerusalem, Israel
ariel.zvulun@gmail.com

***Abstract*—Quantum Tomography partially measures and then recovers the remaining density matrix quantum state, in order to verify that a certain device – processor or detector – indeed outputs the intended quantum state. However, single matrix element measurements rapidly increase in number with the density matrix dimension and are error-prone. This work proposes a novel *quantum software viewpoint* on quantum tomography. Instead of individual matrix elements, measurements and density matrix recovery are performed on higher-abstraction *modules*, i.e. semantically meaningful groups of matrix elements.**

***Quantum Modules Tomography* potentially reduce the necessary number of explicit measurements, while still allowing recovery of the whole density matrix. A *Recovery Theorem* is formulated and proved: density matrix diagonal measurements suffice to recover the whole system density matrix, in terms of modules. The recovery procedure is illustrated by a few case studies.**

Keywords-quantum software; quantum modules tomography; density matrix; software modules; density matrix recovery theorem

I. INTRODUCTION

Quantum Software Models deal with quantum software systems represented by density matrices. In a previous paper of this series [7], it has been shown that software system modules span sub-spaces, obtained by projectors acting on the whole software system state.

Quantum Tomography (QT) is a set of techniques to check behavior correctness of quantum devices, i.e. whether they output the intended quantum state, represented by a density matrix. QT measures some matrix elements, recovering the remaining density matrix elements, from the measured ones.

This paper argues that quantum *modules* tomography may reduce the number of necessarily measured matrix elements, and enables recovery of the whole density matrix of quantum software systems, from the matrix diagonal.

A. Quantum Software is Measurable

Quantum software is represented by a Density Matrix, whose most important characteristic is to be modularizable, as described in the next section. Quantum Software is a generic concept, that may refer to any of three types – quantum systems processing qubits, classical systems processing classical bits, or hybrid systems transitioning back and forth between quantum and classical sub-systems, due to the representation similarity of these three kinds of systems. Quantum software systems will be shown to comply with quantum computing measurements (see e.g. Nielsen and Chuang [12]). They are measurable, which is essential for Quantum Tomography.

B. Quantum Software Modules Tomography

There are two kinds of Quantum Tomography: *quantum state tomography* (e.g. Gross et al. [10]) reconstructing quantum states, and quantum process tomography which reconstructs processes by physical systems ([4],[13]).

Quantum Modules Tomography, proposed in this paper, is a novel kind of Quantum Tomography. Measurements and recovery of the whole software system density matrix are based upon system modules, instead of individual matrix elements. This is enabled by quantum software algebraic constraints.

C. Paper Organization

The rest of the paper is organized as follows. Section II characterizes Quantum Software. Section III deals with quantum software measurement. Section IV details Quantum Modules Tomography, then formulates and proves the Recovery Theorem for the whole density matrix. Section V illustrates the procedures with case studies and a simulation. Section VI concisely refers to related work. Section VII is a Conclusion of the paper.

II. QUANTUM SOFTWARE MODELS

This section introduces the reader to the main characteristics of Quantum Software Systems. It explains Density Matrix generation and modularization for these systems.

A. Conceptual Semantics and Bipartite Graph

Quantum Software systems display two main properties, structure and behavior, fitting two kinds of entities: *Structors & Functionals*. Structors generalize classes in Object Oriented Design (OOD). Functionals generalize OOD class methods.

DOI: 10.18293/SEKE2023-214

Structors & Functionals have a double role. They are the concepts conveying meaning of the Quantum Software system. In the other role, their associated indices stand for vertices of graphs, which generate density matrices – the underlying linear algebraic representation of quantum software theory.

The Command Design Pattern ([9], page 233) is a running example in this paper, to explain basic ideas. Its goal is to abstract typical commands – copy, paste, delete, save – found in a variety of useful applications, into a reusable generic pattern. Behind every Structor ***Sj*** or Functional ***Fk*** indices there is a real meaningful concept understandable by software engineers.

The Command Design Pattern conceptual semantics is collected in Fig. 1. The *Command* module contains the command abstraction. Its Structors are ICommand, a generic interface, and Concrete Command standing for any commonly used command. Their Functionals enable to specify or execute commands. The *Invoker* module stands for menu-items or buttons used to invoke commands. The *Receiver* module refers to either a file or a formatted document receiving the outcome of a command execution.

	Modules	Module Size		Structors		Functionals
M1	Command	2-by-2	S1	ICommand	F1	Execute
			S2	Concrete Command	F2	Specify Command
M2	Invoker	1-by-1	S3	Action Invoker	F3	Invoke
M3	Receiver	2-by-2	S4	IFile Receiver	F4	Receiver Action
			S5	Concrete Receiver	F5	Specify Receiver

Figure 1. Command Design Pattern Modules, Structors & Functionals – It has 3 modules, Command, Invoker, Receiver, with sizes 2-by-2, 1-by-1, 2-by-2. For instance, the Command Module Structors & Functionals set is {S1, S2, F1, F2}. (Figures in color online).

A simple depiction of a quantum software system is a bipartite graph [15]. These graphs have two vertex sets, where a vertex is linked only to vertices in the other set. One set has Structor vertices {S1, S2, …, Sj} and another set has Functional vertices {F1, F2, …, Fk}. A Structor providing a Functional – say, a class providing a method definition, in OOD parlance – is linked to the Functional.

The Command Design Pattern bipartite graph is seen in Fig. 2. Algebraic manipulations for modularization of quantum software systems, are totally independent of the specific conceptual semantics of the system indices.

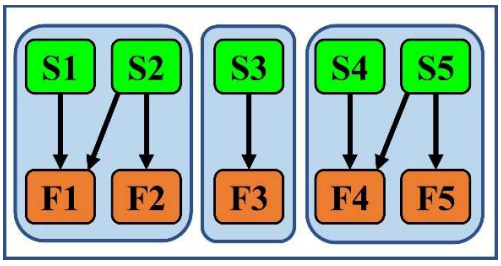

Figure 2. Command Design Pattern Bipartite Graph– Its 10 vertices are: 5 (green) Structors {S1, S2, …, S5} and 5 (orange) Functionals {F1, F2, …, F5}. Also seen 3 (blue) background Modules. The middle module has {S3, F3} as vertices. Arrows link Structors to provided Functionals, e.g. S2 provides F1 & F2.

B. *From Bipartite Graph to Density Matrix*

A Laplacian matrix $\boldsymbol{L}$ [11], [18] is defined upon any graph, in particular a bipartite graph, according to equation (1).

$$\boldsymbol{L = D - A} \tag{1}$$

where $\boldsymbol{D}$ is the diagonal Degree matrix – showing the degree $\mathbf{d_{jj}}$ of vertex **j** – and $\boldsymbol{A}$ is the Adjacency matrix – showing the neighbors of each vertex with a 1-valued matrix element, with negative sign due to equation (1).

It has been observed by Braunstein et al. [3] that a Density Matrix $\boldsymbol{\rho}$ can be obtained from a Laplacian $\boldsymbol{L}$ normalized by its Trace, the sum of the diagonal degrees, as in eq. (2).

$$\boldsymbol{\rho = L / Trace(L)} \tag{2}$$

A Density matrix $\boldsymbol{\rho}$ generated from the bipartite graph in Fig. 2, by equation (1) and normalized as in eq. (2) can be seen in Fig. 3.

$\rho = 1/14*$

		0000⟩		0001⟩		0010⟩		0011⟩		0100⟩		0101⟩		0110⟩		0111⟩		1010⟩		1001⟩
		F1	F2	F3	F4	F5	S1	S2	S3	S4	S5
⟨0000|	F1	2	0	0	0	0	-1	-1	0	0	0
⟨0001|	F2	0	1	0	0	0	0	-1	0	0	0
⟨0010|	F3	0	0	1	0	0	0	0	-1	0	0
⟨0011|	F4	0	0	0	2	0	0	0	0	-1	-1
⟨0100|	F5	0	0	0	0	1	0	0	0	0	-1
⟨0101|	S1	-1	0	0	0	0	1	0	0	0	0
⟨0110|	S2	-1	-1	0	0	0	0	2	0	0	0
⟨0111|	S3	0	0	-1	0	0	0	0	1	0	0
⟨1000|	S4	0	0	0	-1	0	0	0	0	1	0
⟨1001|	S5	0	0	0	-1	-1	0	0	0	0	2

Figure 3. Command Design Pattern Density Matrix – One can see the Laplacian matrix $\boldsymbol{L}$ within square brackets, normalized by the Trace = 14. Within the Laplacian, one perceives the Diagonal Degree matrix $\boldsymbol{D}$ (green) and the two quadrants of the Adjacency matrix $\boldsymbol{A}$ (blue), above the diagonal and reflected below the diagonal. Above the column indices one sees *kets*, and to the left of the row indices the *bras* of this software system basis set.

Algebraic features of software systems represented by a Density Matrix derived from the Laplacian are essential for this paper's goal, viz. to facilitate recovery of the quantum software system state. One can easily show, from eq. (1) that each of the Laplacian rows and columns sum to zero. This is preserved in the Density Matrix – and perceived e.g. in Fig. 3.

The Adjacency matrix of a Quantum Software system, also appearing in eq. (1), has two quadrants, as seen in Fig. 3. The upper-right quadrant is above the diagonal and reflected into the lower-left quadrant below the diagonal. Further algebraic properties of relevance to Quantum Module Tomography are collected in Definition 1, in sub-section **IV**.***B.***

C. *Modules from the Density Matrix*

Modules of Quantum Software systems can be obtained directly from the Laplacian eigenvectors fitting zero-valued eigenvalues (see [6]). Modules can also be obtained by partition of the sum of projection operators fitting kets of the system basis set, into disjoint sets of projection operators, as explained in [7]. Modularized matrices of a Software system have mutually orthogonal modules.

III. QUANTUM SOFTWARE MEASUREMENT

This section shows that Quantum Software systems comply with quantum computing projective measurements [12] for modules, illustrated with the running example software system.

A. Quantum Software Projective Measurement

Module projectors span sub-spaces of a whole Quantum Software system. A natural choice for modules is projective measurement, defined by its observables and final state.

An observable is a Hermitian operator $\boldsymbol{M_m}$ on the state space of the whole system Density Matrix $\boldsymbol{\rho}$. Its properties are: mutually orthogonal observables and spectral decomposition as in eq. (3).

$$\boldsymbol{M_m} = \sum_{\boldsymbol{n}} \boldsymbol{n} * \boldsymbol{P_n} \tag{3}$$

$\boldsymbol{P_n}$ is a projector onto the $\boldsymbol{M_m}$ eigenspace with eigenvalue $\boldsymbol{n}$.

The final state of the system $\boldsymbol{S_m}$ after measurement is obtained by equation (4).

$$\boldsymbol{S_m} = (\boldsymbol{M_m}\,\boldsymbol{\rho}\boldsymbol{M_m}^{\dagger})/\boldsymbol{Trace}(\boldsymbol{M_m}^{\dagger}\boldsymbol{M_m}\,\boldsymbol{\rho}) \tag{4}$$

where $\boldsymbol{M_m}^{\dagger}$ is the conjugate transpose of $\boldsymbol{M_m}$.

B. Quantum Software Module Observables

The upper module {S1, S2, F1, F2} in each quadrant of Fig. 3 exemplifies eq. (3) by its observable as a projectors' sum:

$$\boldsymbol{M_1} = |\mathbf{0000}\rangle\langle\mathbf{0000}| + |\mathbf{0001}\rangle\langle\mathbf{0001}| + |\mathbf{0101}\rangle\langle\mathbf{0101}| + |\mathbf{0110}\rangle\langle\mathbf{0110}| \tag{5}$$

This observable is seen as a full-matrix in Fig. 4.

$M_1 =$

		0000⟩		0001⟩					0101⟩		0110⟩			
		F1		F2				S1		S2			
⟨0000|	F1	1	0	0	0	0	0	0	0	0	0	0	0
⟨0001|	F2	0	1	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0
⟨0101|	S1	0	0	0	0	0	1	0	0	0	0	0	0
⟨0110|	S2	0	0	0	0	0	0	1	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	0	0	0	0	0	0	0	0

Figure 4. Observable Matrix of Upper Module of the Command Design Pattern in Fig. 3 – The only non-zero matrix elements in this observable are in the diagonal, fitting the eq. (5) projectors. Matrix element indices, kets and bras of this Module are seen. The module locations in $\boldsymbol{\rho}$ are marked as blue background.

B. Quantum Software Module Measurement Final State

Now we insert the Observable $\boldsymbol{M_1}$ into eq. (4) to exemplify the final state after measurement of the Command Design Pattern. Equation (4) is simplified, as $\boldsymbol{M_1}$ is real, diagonal and has only 1-valued matrix elements. Also the normalizing factor (1/14) of $\boldsymbol{\rho}$ in Fig. 3 cancels out, as it appears in both numerator and denominator of eq. (4). The resulting final state is in Fig. 5.

$S_1 = 1/6*$

		0000⟩		0001⟩					0101⟩		0110⟩			
		F1		F2		F3	F4	F5	S1		S2	S3	S4	S5
⟨0000|	F1	2	0	0	0	0	0	0	-1	-1	0	0	0
⟨0001|	F2	0	1	0	0	0	0	0	0	-1	0	0	0
	F3	0	0	0	0	0	0	0	0	0	0	0	0
	F4	0	0	0	0	0	0	0	0	0	0	0	0
	F5	0	0	0	0	0	0	0	0	0	0	0	0
⟨0101|	S1	-1	0	0	0	0	0	0	1	0	0	0	0
⟨0110|	S2	-1	-1	0	0	0	0	0	0	2	0	0	0
	S3	0	0	0	0	0	0	0	0	0	0	0	0
	S4	0	0	0	0	0	0	0	0	0	0	0	0
	S5	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5. Final state after measurement of the Command Design Pattern upper module in Fig. 3 – It is a decreased size Density Matrix of the Module itself, embedded in its exact location of the whole system density matrix. The normalizing factor 1/6 is adjusted to preserve Density Matrix properties.

This final state is an interesting result, since the whole system Density Matrix can be obtained from a bidirectional direct sum of the modules reduced density matrices (see [8]).

IV. QUANTUM MODULES TOMOGRAPHY

This section states the idea of Quantum Modules Tomography and presents its overall procedure. The Recovery Theorem enabling a more efficient Quantum Software Density Matrix recovery is formulated and proved.

A. Idea of Quantum Modules Tomography

Abstraction is the basic idea behind software modules and any of their applications. Abstraction means reducing the number of concepts necessary to formulate the purpose and functionality of a system and its sub-systems. The benefits are deeper understanding and increased efficiency.

Quantum Modules Tomography is based upon two assumptions:

- *Reduction of measurements number* – to at most proportional to the size of the diagonal Degree matrix;
- *Constraining of Density Matrix completion* – based upon the Laplacian standard algebraic features collected below in Definition 1, in the next sub-section.

An important observation is that the number and sizes of Modules in a software system is finite and small. This is justified by dealing with software systems in a hierarchical way. The same is true concerning the numerical values of the diagonal Degree matrix elements which express the relative dimensions of the modules.

B. Towards the Recovery Theorem: Density Matrix Algebraic Features

The standard algebraic features of the Laplacian are preserved by Density Matrices. These features are collected in Definition 1, in the next text box.

Definition 1 – **Standard Algebraic Features of software systems' Laplacian**

These features are:

- *Sum-to-zero* – *Laplacian* rows & columns sum to zero.
- *Square-Quadrants* – *Adjacency Matrix* quadrants containing modules are square.
- *Adjacency Matrix Linear Independence* – Adjacency Matrix rows are mutually linearly independent; Adjacency matrix columns are also mutually linearly independent.
- Modules Orthogonality – A given Module rows and columns are orthogonal to rows and columns of all other modules of the same software system. Thus, modules are block-diagonal.

C. The Recovery Theorem

After the needed measurements are done for a quantum software system, the next theorem assures that one can recover lacking values to complete the system density matrix.

Theorem 1 – **Recovery of whole Density Matrix from its Diagonal**

Assuming:

a- ***quantum software system*** – is describable by a modular density matrix $\boldsymbol{r}$ obtained by normalizing a Laplacian $\boldsymbol{L}$ as $\boldsymbol{r = L / Trace(L)}$, where $\boldsymbol{L}$ complies with the Standard Laplacian algebraic features in Definition 1;

b- ***strictly necessary measurements*** – were performed to obtain the Density Matrix diagonal;

Then:

The whole quantum software system Density Matrix is completely recoverable by a finite small set of additions to the off-diagonal density matrix elements.

Proof:

The number of the off-diagonal density matrix additions is obviously finite. It is enough to complete the set of additions to the upper-right quadrant of the Adjacency matrix. The lower-left quadrant is the reflection of the upper-right quadrant. The exact number of additions to the upper-right quadrant of the Adjacency matrix is half of the Trace of the whole matrix.

For each diagonal 1-valued matrix element one should add one negative 1-valued matrix element in the same row and/or column. For each diagonal 2-valued matrix element one should add two negative 1-valued matrix elements in the same row and/or column. And so on for higher valued diagonal matrix elements. All the remaining matrix elements above the whole matrix diagonal are zero-valued, and reflected below the diagonal.

The above unequivocally determines all the density matrix element numerical values to be added. The exact addition locations are constrained by the standard Laplacian algebraic features. □

We emphasize an essential aspect of the above theorem:

- the number of additions is *small* as module sizes are limited by collecting only related concepts in each module, to facilitate human understanding.

D. Quantum Modules Tomography Procedure

The Quantum Modules Tomography Procedure is:

1) Measure whole matrix degree diagonal elements;
2) Fill the Adjacency matrix diagonal of the upper-right quadrant with negative 1-valued elements;
3) Add other non-zero elements in the Adjacency matrix upper-right quadrant according to the Density Matrix degree diagonal;
4) Complete with zero-valued elements above the Density Matrix degree diagonal;
5) Reflect the upper-right quadrant to elements below the Density Matrix degree diagonal.

V. Case Studies

This section illustrates the Quantum Modules Tomography procedure by means of two Quantum Software systems: the Grover Search design, a strictly quantum system, and a simulation of a simplified classical system.

A. Quantum Grover Search

Grover Search (see e.g. [2] page 166) is a well-known quantum algorithm which performs search of unstructured (unsorted) $\boldsymbol{N}$ data items with a reduced complexity of $O(\sqrt[2]{\boldsymbol{N}})$, instead of the classical complexity of $O(\boldsymbol{N})$.

Modules, Structors and Functionals of strictly quantum systems (processing only qubits) are obtained from quantum circuits, as in Fig. 6.

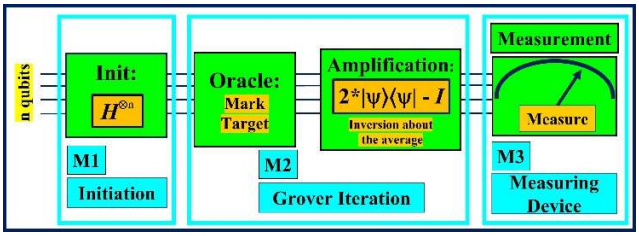

Figure 6. Grover Search quantum circuit - Structors are "boxes" (green) containing quantum gates (orange), except for the typical "measurement" that inputs qubits and outputs classical bits. Modules (blue) are Structors grouped by some logical reason. For instance, the Grover Iteration is a single cycle of a loop consisting of an Oracle and an Amplification Structor.

The Grover Search Modules, Structors & Functionals are shown in Fig. 7.

	Modules	Module Size		Structors		Functionals
M1	Initiation	1-by-1	S1	Init	F1	$H^{\otimes n}$ Equal-Superposition
M2	Grover Iteration	2-by-2	S2	Oracle	F2	Mark-Target
			S3	Amplification	F3	Inversion about the average
M3	Measuring Device	1-by-1	S4	Measurement	F4	Measure

Figure 7. Grover Search Modules, Sructors & Functionals – It has the same color conventions as in Fig. 1. ***H*** stands for the Hadamard operator, here to the n[th] tensor product power.

Once one has the Structors & Functionals for the quantum software system – be it purely classical, strictly quantum, or hybrid – from a bipartite graph one generates and normalizes the Laplacian to a Density Matrix.

The Grover Search quantum software system Density Matrix is in Fig. 8. It differs from purely classical systems, as the Command Pattern, by the number of basis set qubits needed to describe the system, and the module sizes.

$\rho = 1/10^*$

		000⟩		001⟩		010⟩		011⟩		100⟩		101⟩		110⟩		111⟩
		F1	F2	F3	F4	S1	S2	S3	S4
⟨000|	F1	1	0	0	0	-1	0	0	0
⟨001|	F2	0	2	0	0	0	-1	-1	0
⟨010|	F3	0	0	1	0	0	0	-1	0
⟨011|	F4	0	0	0	1	0	0	0	-1
⟨100|	S1	-1	0	0	0	1	0	0	0
⟨101|	S2	0	-1	0	0	0	1	0	0
⟨110|	S3	0	-1	-1	0	0	0	2	0
⟨111|	S4	0	0	0	-1	0	0	0	1

Figure 8. Grover Search Density Matrix – This software system Density Matrix has a 3 qubits basis set and 3 modules, only one with a 2-by-2 size.

The Grover Search case study illustrates that:

- The same algebraic techniques are applicable to all quantum software systems – pure classical, strictly quantum or hybrid (classical and quantum).
- Ignoring Structors & Functionals location and semantics, a quantum Grover Search 2-by-2 module in Fig. 8 or a classical Command design pattern 2-by-2 module in Fig. 3 are identical and measurable by an observable comparable to that in Fig. 4.

B. Simulation of a Simple Classical System

The purpose of this simulation is to illustrate the idea that Quantum Module Tomography indeed may recover the whole Density Matrix of a Quantum Software system, while reducing the number of needed matrix element measurements. The simulation was done on the simplified Software System, whose density matrix is seen in Fig. 9.

$\rho = 1/8^*$

	|000⟩	|001⟩	|010⟩	|011⟩	|100⟩	|101⟩
		F1	F2	F3	S1	S2	S3
⟨000|	F1	2	0	0	-1	-1	0
⟨001|	F2	0	1	0	0	-1	0
⟨010|	F3	0	0	1	0	0	-1
⟨011|	S1	-1	0	0	1	0	0
⟨100|	S2	-1	-1	0	0	2	0
⟨101|	S3	0	0	-1	0	0	1

Figure 9. Simplified Quantum Software System Density Matrix – it is obtained from Fig. 8 rows/columns {F2,F3,F4} and {S2,S3,S4} suitably renumbered. It uses the same color conventions as in previous figures. It has two modules.

The measurements projectors are in Fig. 10 with their probabilities.

Projector	Probability
|000⟩⟨000|	1/4
|001⟩⟨001|	1/8
|010⟩⟨010|	1/8
|011⟩⟨011|	1/8
|100⟩⟨100|	1/4
|101⟩⟨101|	1/8

Figure 10. Projectors used for measurements of the Simplified Quantum Software System seen in Fig. 9. – The respective probabilities are shown.

The overall simulation – implemented in qiskit, an IBM quantum computing oriented language – performed "projectors" sampling upon the Density Matrix in Fig. 9, using the **DensityMatrix.sample_memory** function. Once the sampling achieved a stable probability distribution, one obtains the values of each of the diagonal matrix elements, as specified in the Quantum Modules Tomography Procedure in sub-section IV.*D*. These values confirmed the expectations of Fig. 10.

When does the measured probability distribution reach stability? The answer is given in Fig. 11.

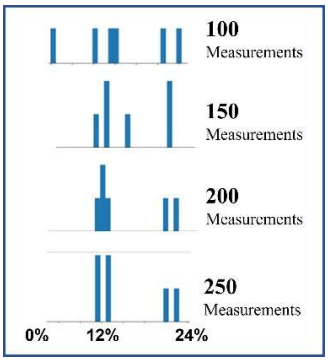

Figure 11. Convergence of sampling percentages – one perceives that by increasing measurement numbers, graphs converge to two values 12% and 24% already for 200 Measurements and is stable for 250 Measurements. When samplings are stable, one may stop the simulation. These percentages respectively fit the 1/8 and ¼ probabilities.

For instance, looking at the 250 measurements, since the heights of the 12% results are twice those of 24% results, then the total value for 12% is 12*4= 48% and the total value for 24% is 24*2=48%. This fits the facts that there are 4 projectors with probability 1/8, i.e. 4*(1/8)= ½ and only two projectors with probability ¼, i.e. 2*(1/4)= ½. In other words, the simulation results confirm the expectations of Fig. 10.

VI. Related Work

Here are very concise references to topics relevant to this paper.

Braunstein and co-authors [3] focus on mixed states separability, represented by density matrices obtained from Laplacians. They state that specific graph types originate entangled or separable states independently of their labeling. Chai Wah Wu [19], [20], also considers Laplacian and density matrix separability.

The Quantum Tomography Measurements literature is very extensive. We provide a few entry points. The "Quantum State Estimation" book edited by Paris and Rehacek [13] has chapters relevant to Tomography. For instance, the D'Ariano et al. [4] chapter on "Quantum Tomographic Methods", and the chapter by Altpeter et al. on "Quantum State Tomography" [1].

The maximum-likelihood (MaxLik) algorithm, and its variants, e.g. the Diluted Maximum-Likelihood algorithm by Rehacek et al. [14] offer alternative ways to reconstruct a quantum state from tomographic measurements.

VII. Conclusion

A. Theory and Potential Applications

The potential applications refer to eventually reducing the number of measurements, and faster recovery of quantum software systems' density matrix, according to the Recovery Theorem enabling density matrix reconstruction from measurement of the density matrix diagonal only.

Two caveats are in place here. This paper theory focused on density matrices derived from Laplacians. Elsewhere, we discussed transformations of any density matrices to quantum software density matrices as derived from Laplacians.

Second, the ultimate testing of the ideas of the "Quantum Modules Tomography" theory will be in actual tomography measurements in a laboratory. This is an invitation for experimental researchers.

B. Future Work

Open issues to be dealt with in future work include different formulations of Recovery Theorems, solving particularly difficult modularizations of quantum software systems, and concurrent use of alternative reconstruction density matrix algorithms, such as maximum-likelihood, together with the Recovery Theorem of this paper.

C. Main Contribution

The main contribution of this paper is the *Modules* abstraction for Quantum Software Systems, resulting in the *Recovery Theorem*, with potential applications to Quantum Modules Tomography.

References

[1] J. B. Altpeter, D.F.V. James and P.G. Kwiat. 2004. Quantum State Tomography. In Lect. Notes Phys. 649. (Ref. [13]).

[2] Adriano Barenco. 1998. Quantum Computation: An Introduction. pp. 143-183. in Hoi-Kwong Lo, Sandu Popescu and Tim Spiller (eds.) *Introduction to Quantum Computation and Information*, World Scientific, Singapore

[3] Samuel L. Braunstein, Sibasish Ghosh and Simone Severini. 2006. The Laplacian of a graph as a density matrix: a basic combinatorial approach to separability of mixed states. arXiv:quant-ph/0406165.

[4] Giacomo M. D'Ariano, Matteo G.A. Paris and Massimiliano F. Sacchi. 2004. Quantum Tomographic Methods. In *Lect. Notes Phys.* **649**, pp. 7-58. (Ref. [13]).

[5] Iaakov Exman. 2014. Linear Software Models: Standard Modularity Highlights Residual Coupling. *Int. Journal on Software Engineering and Knowledge Engineering*, vol. 24, pp. 183-210. DOI: 10.1142/S0218194014500089.

[6] Iaakov Exman and Rawi Sakhnini. 2018. Linear Software Models: Bipartite Isomorphism between Laplacian Eigenvectors and Modularity Matrix Eigenvectors. Int. Journal on Software Engineering and Knowledge Engineering, vol. 28, pp. 897-935. DOI: 10.1142/S0218194018400107

[7] Iaakov Exman and Alon Tsalik Shmilovich. 2021. Quantum Software Models: The Density Matrix for Classical and Quantum Software Systems Design, in Proc. (Q-SE) IEEE/ACM 2nd International Workshop on Quantum Software Engineering, pp. 1-6. DOI: 10.1109/Q-SE52541.2021.00008

[8] Iaakov Exman and Alexey Nechaev. 2022. Quantum Software Models: Software Density Matrix is a Perfect Direct Sum of Module Matrices. In Proc. SEKE'2022 Int. Conference on Software Engineering and Knowledge Engineering, pp. 434-439. DOI: 10.18293/SEKE2022-158.

[9] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. 1995. *Design Patterns – Elements of Reusable Object-Oriented Software.* Addison-Wesley, Boston, MA, USA.

[10] David Gross, Yi-Kai Liu, Steven T. Flammia, Stephen Becker and Jens Eisert. 2010. Quantum state tomography via compressed sensing. Phys. Rev. Letters, 226-236. Also arXiv:0909.3304 [quant-ph].

[11] R. Merris. 1994. Laplacian matrices of graphs: A Survey. *Linear Algebr. Appl.*, **197-198,** pp. 143-176.

[12] Michael A. Nielsen and Isaac L. Chuang. 2000. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, UK.

[13] Matteo Paris and Jaroslav Rehacek (eds.). 2004. *Quantum State Estimation*. In *Lecture Notes in Physics*, Vol. **649**. Springer, Heidelberg, Germany. DOI: 10.1007/b986673.

[14] Jaroslav Rehacek, Zdenek Hradil, E. Knill and A.I. Lvovsky. 2007. Diluted maximum-likelihood algorithm for quantum tomography. arXiv:quant-ph/0611244.

[15] Eric W. Weisstein, Bipartite graph. 2022.

[16] http://mathworld.wolfram.com/Bipartite-Graph.html

[17] Eric W. Weisstein, Laplacian, 2022.

[18] http://mathworld.wolfram.com/LaplacianMatrix.html

[19] Chai Wah Wu. 2009. Multipartite Separability of Laplacian Matrices of graphs. *Electronic J. of Combinatorics*, **16**, #R61.

[20] Chai Wah Wu. 2016. Graphs whose normalized Laplacian matrices are separable as density matrices in quantum mechanics. *J. Discrete Mathematics*, 339, pp. 1377-1381. DOI: 10.1016/j.disc.2015.12.001