On Partially Unitary Learning

Mikhail Gennadievich Belov*

Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, GSP-1, Moscow, Vorob'evy Gory, 119991, Russia

Vladislav Gennadievich Malyshkin†

Ioffe Institute, Politekhnicheskaya 26, St Petersburg, 194021, Russia

(Dated: May, 14, 2024)

`$Id: PartiallyUnitaryLearning.tex,v 1.267 2024/05/16 08:49:47 mal Exp $`

The problem of an optimal mapping between Hilbert spaces IN of $|\psi\rangle$ and OUT of $|\phi\rangle$ based on a set of wavefunction measurements (within a phase) $\psi_l \to \phi_l$, $l = 1 \ldots M$, is formulated as an optimization problem maximizing the total fidelity $\sum_{l=1}^{M} \omega^{(l)} \left|\langle\phi_l|\mathcal{U}|\psi_l\rangle\right|^2$ subject to probability preservation constraints on $\mathcal{U}$ (partial unitarity). Constructed operator $\mathcal{U}$ can be considered as a IN to OUT quantum channel; it is a partially unitary rectangular matrix of the dimension $\dim(OUT) \times \dim(IN)$ transforming operators as $A^{OUT} = \mathcal{U} A^{IN} \mathcal{U}^\dagger$. An iteration algorithm finding the global maximum of this optimization problem is developed and it's application to a number of problems is demonstrated. A software product implementing the algorithm is available from the authors.

* mikhail.belov@tafs.pro

† malyshki@ton.ioffe.ru

I. INTRODUCTION

The progress in machine learning (ML) knowledge representation from linear regression coefficients, perceptron weights[1], statistical learning[2], and logical approaches[3] to support vector machines[4], rules and decision trees[5], fuzzy logic[6, 7], and deep learning[8] has been the direction of ML development within the last four decades. Recently knowledge representation in the form of a unitary operator started to attract a lot of attention[9–11]. The problem of learning unitary matrices is also useful in other fields. For example in quantum mechanics inverse problem[12, 13], quantum computing[14], market dynamics[15] and others.

The techniques used for unitary learning differ in unitary matrix representation, input data, and quality criteria. A substantial number of existing works[16, 17] use Frobenius L^2 norm of a difference between target and current matrix $\|\mathcal{U}-\mathcal{V}\|^2$. A limitation of this type of approach is that it cannot determine operator $\mathcal{U}$ from wavefunction measurements — they are measured without phase information. A better option is to use fidelity of the source/target states $|\langle\phi\,|\,\mathcal{U}\,|\,\psi\rangle|^2$; this approach is used in [9, 18] and many others. An important advantage of it is that multiplication of source $|\psi\rangle$ or target $|\phi\rangle$ by a random phase does not change the fidelity. A disadvantage — the operator $\mathcal{U}$ itself can be determined only within a phase.

In our previous work[19] the problem of unitary learning was generalized to partially unitary operators. This operator maps two Hilbert spaces of different dimensions (a unitary operator maps a Hilbert space into itself). The problem is to maximize the fidelity of a mapping between Hilbert spaces IN of $|\psi\rangle$ and OUT of $|\phi\rangle$ based on a set of wavefunction measurement (within a phase) observations $\psi_l \to \phi_l$, $l = 1 \ldots M$, as an optimization problem maximizing the total fidelity $\sum_{l=1}^{M} \omega^{(l)} \,|\langle\phi_l\,|\,\mathcal{U}\,|\,\psi_l\rangle|^2$ subject to probability preservation constraints on $\mathcal{U}$ (partial unitarity). This problem is reduced to a problem of maximization of a quadratic form on $\mathcal{U}$ subject to quadratic form constraints. This is a variant of QCQP problem. This problem is non-convex with local extremums and multiple saddle points. In this work an iteration algorithm finding the global maximum of this optimization problem is developed and it's application to a number of problems is demonstrated. For software availability see Appendix B; all references to a code in the paper correspond to this software.

II. FORMULATION OF THE PROBLEM

Consider quantum mechanics inverse problem. It may be broadly described as a problem of determining the internal structure (e.g. Hamiltonian) of a system from wavefunction measurements. A number of other problems in statistics, machine learning, data analysis, etc. can be converted to a problem of the following form: Let there are two Hilbert spaces of the dimensions n and D, corresponding states are $\psi(\mathbf{x})$ and $\phi(\mathbf{f})$ in some $\mathbf{x}$- and $\mathbf{f}$- bases x_k and f_j of $n \geq D$

$$\psi(\mathbf{x}) = \sum_{k=0}^{n-1} \alpha_k x_k \tag{1a}$$

$$\phi(\mathbf{f}) = \sum_{j=0}^{D-1} \beta_j f_j \tag{1b}$$

There is a scalar product operation $\langle\cdot\rangle$ in each Hilbert space allowing to calculate scalar product inside the basis $\langle\psi\,|\,\psi'\rangle = \sum_{k,q=0}^{n-1} \alpha_k^* \langle x_k\,|\,x_q\rangle\, \alpha'_q$ and $\langle\phi\,|\,\phi'\rangle = \sum_{j,i=0}^{D-1} \beta_j^* \langle f_j\,|\,f_i\rangle\, \beta'_i$, but *not across the bases*: the $\langle x_k\,|\,f_j\rangle$ cannot be calculated! Assume we have $l = 1 \ldots M$ wavefunction pairs (typically $M \gg n$) as "observations":

$$\psi_l(\mathbf{x}) \to \phi_l(\mathbf{f}) \qquad\qquad \text{weight } \omega^{(l)} \tag{2}$$

$$1 = \langle\psi_l\,|\,\psi_l\rangle = \langle\phi_l\,|\,\phi_l\rangle \tag{3}$$

these are M mapping from Hilbert space $\mathbf{x}$ to Hilbert space $\mathbf{f}$. The inverse problem is to find partially unitary operator $\mathcal{U}$ (a matrix u_{jk} of the dimension $D \times n$) acting from $\mathbf{x}$ to $\mathbf{f}$

$$f_j = \sum_{k=0}^{n-1} u_{jk} x_k \qquad\qquad j = 0 \ldots D-1 \tag{4}$$

that maximizes the total probability (fidelity)

$$\mathcal{F} = \sum_{l=1}^{M} \omega^{(l)} \Big| \langle\phi_l\,|\,\mathcal{U}\,|\,\psi_l\rangle \Big|^2 \underset{\mathcal{U}}{\longrightarrow} \max \tag{5}$$

$$\langle f_j\,|\,f_{j'}\rangle = \sum_{k,k'=0}^{n-1} u_{jk} \langle x_k\,|\,x_{k'}\rangle\, u^*_{j'k'} \qquad\qquad j, j' = 0 \ldots D-1 \tag{6}$$

subject to scalar product preservation (6). The operator $\mathcal{U}$ is acting from Hilbert space $\mathbf{x}$ to Hilbert space $\mathbf{f}$, it can be viewed as a memoryless quantum channel. The functional (5) has matrix element $\langle\phi_l\,|\,\mathcal{U}\,|\,\psi_l\rangle$ absolute value squared, this allows to express the total probability

(fidelity) only with $\langle f_j \,|\, f_{j'} \rangle$, $\langle x_k \,|\, x_{k'} \rangle$ and fourth order terms $\langle f_j x_k \,|\, f_{j'} x_{k'} \rangle$, without using unavailable "cross-moments" $\langle x_k \,|\, f_j \rangle$. The functional also has proper wavefunction phase invariance. A specific of quantum measurement is that a wavefunction cannot be measured itself, only wavefunction squared (probability density) can be possibly obtained from a measurement operation. The ψ^2 is observable whereas the ψ is not. Measured wavefunction pairs (2) should be considered as measured ψ^2 and ϕ^2 with the square root applied, obtained result is possibly multiplied by an unknown random phase $\exp(i\xi_l)$. The optimization problem (5) of finding optimal quantum channel $\mathcal{U}$ is invariant with respect to random phases introduced to measured wavefunctions (2). Without loss of generality we can consider Hilbert space bases orthogonal:

$$\delta_{jj'} = \langle f_j \,|\, f_{j'} \rangle \tag{7a}$$

$$\delta_{kk'} = \langle x_k \,|\, x_{k'} \rangle \tag{7b}$$

if this is not the case — an orthogonalizing procedure can be applied, see Appendices A and E of [20] or plain apply an orthogonalization of Gram–Schmidt type, see `com/polytechnik/kgo/DemoRecoverMapping.java:GramSchmidtTest` for an implementation. For orthogonal bases the condition (6) is exactly unitary condition if $n = D$, and if $n > D$ we name it — the condition of partial unitarity.

$$\delta_{jj'} = \sum_{k=0}^{n-1} u_{jk} u^*_{j'k} \qquad\qquad j, j' = 0 \ldots D-1 \tag{8}$$

The meaning of $\mathcal{F}$ (5) is weighted (with $\omega^{(l)}$ weights) sum of possible similarity between $|\psi_l\rangle$ and $|\phi_l\rangle$. By construction we cannot directly project the states belonging to different Hilbert spaces as $\langle \phi_l \,|\, \psi_l \rangle$, not to mention that such a direct projection will not be invariant with respect to random phases of measured wavefunctions. The only possible way to transform a state from $\mathbf{x}$ to $\mathbf{f}$ is to apply operator $\mathcal{U}$ (4), this is a quantum channel that links two different Hilbert spaces. The states $\psi_l(\mathbf{x})$ and $\phi_l(\mathbf{f})$ in (1) basis are defined with $l = 1 \ldots M$ coefficients $\alpha_k^{(l)}$ and $\beta_j^{(l)}$. For (7) orthogonalized basis the functional (5) takes the form

$$\mathcal{F} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} S_{jk;j'k'} u^*_{j'k'} \tag{9}$$

$$S_{jk;j'k'} = \sum_{l=1}^{M} \omega^{(l)} \beta_j^{(l)} \alpha_k^{(l)} \beta_{j'}^{(l)\,*} \alpha_{k'}^{(l)\,*} \tag{10}$$

this expression is obtained with (4) conversion of $\phi_l(\mathbf{f})$ to a function in $\mathbf{x}$-space and then projecting it to $\psi_l(\mathbf{x})$. For non-orthogonal bases (7) the expression for $S_{jk;j'k'}$ is more complicated and less convenient to use. The tensor $S_{jk;j'k'} = S^*_{j'k';jk}$ is Hermitian by construction. The original problem is now reduced to: maximize (9) functional subject to (8) partial unitarity constraint. There are a number of practical problems that can be reduced to this optimization problem.

A. A Quantum System Time Evolution

Consider a quantum system with time-independent Hamiltonian H. It's time evolution

$$\mathcal{U} = \exp\left[-i\frac{t}{\hbar}H\right] \tag{11}$$

$$\left|\psi^{(t)}\right\rangle = \left|\mathcal{U}|\psi^{(t=0)}\right\rangle \tag{12}$$

Assume we have a long sequence $l = 1 \dots M$ of system observations made equidistantly at time moments $t_l = \tau l$

$$\psi_l(\mathbf{x}) \to \psi_{l+1}(\mathbf{x}) \qquad\qquad \text{weight } \omega^{(l)} = 1 \tag{13}$$

measured wavefunctions can possibly have random phases. Given the basis (1a) measured sample is a sequence of $l = 1 \dots M$ coefficients $\alpha_k^{(l)}$ that define actual wavefunction within a random phase $\exp(i\xi_l)$. To obtain the optimization problem of previous section we put $D = n$, $\beta_j^{(l)} = \alpha_k^{(l+1)}$, and $\mathcal{U}$ is a unitary operator of wavefunction time shift τ.

$$|\psi_{l+1}\rangle = |\mathcal{U}|\psi_l\rangle \tag{14}$$

The $S_{jk;j'k'}$ is then

$$S_{jk;j'k'} = \sum_{l=1}^{M} \omega^{(l)} \alpha_j^{(l+1)} \alpha_k^{(l)} \alpha_{j'}^{(l+1)\,*} \alpha_{k'}^{(l)\,*} \tag{15}$$

The result of optimization problem (9) subject to unitary ($n = D$) constraint (8) is a unitary time shift operator $\mathcal{U}$. To obtain Hamiltonian from $\mathcal{U}$ — this requires taking the logarithm of a unitary matrix, this is a different problem that requires separate consideration[21].

$$H = i\frac{\hbar}{\tau}\ln\mathcal{U} \tag{16}$$

B. A Classic System $\mathbf{x} \to \mathbf{f}$ Mapping

Consider classic $\mathbf{x}^{(l)} \to \mathbf{f}^{(l)}$ vectors mapping, $l = 1 \dots M$, all measurements are real numbers.

$$(x_0, x_1, \dots, x_k, \dots, x_{n-1})^{(l)} \to (f_0, f_1, \dots, f_j, \dots, f_{D-1})^{(l)} \qquad \text{weight } \omega^{(l)} \tag{17}$$

This is a mapping of some observable to observable classic measurements of real vectors, for example let $\mathbf{x}$ be attributes and $\mathbf{f}$ being class label of some dataset used in machine learning classification problem of vector type. A few example of $\mathbf{f}(\mathbf{x})$ constructed model: linear regression, Radon-Nikodym approximation[20], a logical function, a neural network model, etc.

We want to convert this data to $\psi_l(\mathbf{x}) \to \phi_l(\mathbf{f})$ form (2) to construct partially unitary operation $\mathcal{U}$ converting a state from $\mathbf{x}$ to $\mathbf{f}$. Let us define an average in both Hilbert spaces as the sum over data sample (17)

$$\langle h \rangle = \sum_{l=1}^{M} \omega^{(l)} h_l \tag{18}$$

$$G^{\mathbf{x}}_{kk'} = \langle x_k x_{k'} \rangle \tag{19}$$

$$G^{\mathbf{f}}_{jj'} = \langle f_j f_{j'} \rangle \tag{20}$$

where h is a function on $\mathbf{x}$ or $\mathbf{f}$. When (18) is applied to $h_l = x^{(l)}_k x^{(l)}_{k'}$ or $f^{(l)}_j f^{(l)}_{j'}$ obtain Gram matrix (19) or (20). As we discussed above the $\mathbf{x}$ and $\mathbf{f}$ bases can be always orthogonalized (7) with basis linear transform; Gram matrix is equal to unit matrix in the case of an orthogonal basis. Note, that with this classic data we can calculate "cross-moments" $\langle x_k \,|\, f_j \rangle$, an example of a model that uses cross-moments is linear regression

$$\left\langle \left| f_j - \sum_{k=0}^{n-1} \gamma_k x_k \right|^2 \right\rangle \to \min \tag{21}$$

$$f_j(\mathbf{x}) \approx \sum_{k,k'=0}^{n-1} x_k G^{\mathbf{x};-1}_{kk'} \langle f_j x_{k'} \rangle \tag{22}$$

As we use quantum channel ideology, our model should not depend on "cross-moments", only partially unitary operator $\mathcal{U}$ (4) can link $\mathbf{x}$ and $\mathbf{f}$.

To construct a wavefunction $\psi_{\mathbf{y}}(\mathbf{x})$ localized at $\mathbf{x}=\mathbf{y}$ consider localized state $\psi_{\mathbf{y}}(\mathbf{x})$

$$\psi_{\mathbf{y}}(\mathbf{x}) = \sqrt{K(\mathbf{y})}\sum_{i,k=0}^{n-1} y_i G^{\mathbf{x};\,-1}_{ik} x_k = \frac{\sum\limits_{i,k=0}^{n-1} y_i G^{\mathbf{x};\,-1}_{ik} x_k}{\sqrt{\sum\limits_{i,k=0}^{n-1} y_i G^{\mathbf{x};\,-1}_{ik} y_k}} = \frac{\sum\limits_{i=0}^{n-1} \psi^{[i]}(\mathbf{y})\psi^{[i]}(\mathbf{x})}{\sqrt{\sum\limits_{i=0}^{n-1}\left[\psi^{[i]}(\mathbf{y})\right]^2}} \tag{23}$$

$$K(\mathbf{x}) = \frac{1}{\sum\limits_{i,k=0}^{n-1} x_i G^{\mathbf{x};\,-1}_{ik} x_k} = \frac{1}{\sum\limits_{i=0}^{n-1}\left[\psi^{[i]}(\mathbf{x})\right]^2} \tag{24}$$

The $\psi_{\mathbf{y}}(\mathbf{x})$ is a function on $\mathbf{x}$ localized at given $\mathbf{y}$, it is just normalized reproducing kernel, $1=\langle\psi_{\mathbf{y}}\,|\,\psi_{\mathbf{y}}\rangle$. In (23) it is written in two bases: original x_k, for which $\langle x_q x_k\rangle = G^{\mathbf{x}}_{qk}$, and in some orthogonalized basis $\left|\psi^{[i]}\right\rangle$ such that $\left\langle\psi^{[q]}\,\middle|\,\psi^{[k]}\right\rangle=\delta_{qk}$. The $K(\mathbf{x})$ is Christoffel function. Localized states in $\mathbf{f}$-space can be obtained with n to D and x to f replacement. With these localized states obtain $\psi_l(\mathbf{x})\to\phi_l(\mathbf{f})$ mapping for classic data (17), $l=1\ldots M$

$$\sqrt{K(\mathbf{x}^{(l)})}\sum_{i,k=0}^{n-1} x^{(l)}_i G^{\mathbf{x};\,-1}_{ik} x_k \to \sqrt{K(\mathbf{f}^{(l)})}\sum_{i,j=0}^{D-1} f^{(l)}_i G^{\mathbf{f};\,-1}_{ik} f_k \qquad \text{weight } \omega^{(l)} \tag{25}$$

This mapping is actually an original $\mathbf{x}\to\mathbf{f}$ mapping (17) with $\mathbf{x}$ and $\mathbf{f}$ linearly transformed and normalized. Contrary to an original mapping — the (25) can be considered as a mapping of two Hilbert space states with (18) scalar product. It is convenient to introduce the moments of Christoffel functions product:

$$\left\langle x_k f_j\,\middle|\,K^{(\mathbf{x})}K^{(\mathbf{f})}\,\middle|\,x_{k'}f_{j'}\right\rangle = \sum_{l=0}^{M}\omega^{(l)}\frac{x^{(l)}_k x^{(l)}_{k'}}{\sum\limits_{q,q'=0}^{n-1} x^{(l)}_q G^{\mathbf{x};\,-1}_{qq'} x^{(l)}_{q'}}\cdot\frac{f^{(l)}_j f^{(l)}_{j'}}{\sum\limits_{s,s'=0}^{D-1} f^{(l)}_s G^{\mathbf{f};\,-1}_{ss'} f^{(l)}_{s'}} \tag{26}$$

Assuming the $\mathbf{x}$ and $\mathbf{f}$ bases are orthogonalized (7), the tensor (10) is then

$$S_{jk;j'k'} = \left\langle x_k f_j\,\middle|\,K^{(\mathbf{x})}K^{(\mathbf{f})}\,\middle|\,x_{k'}f_{j'}\right\rangle \tag{27}$$

We obtained, now for classic data, an optimization problem (9) subject to (8) partial unitarity constraint. The result is operator $\mathcal{U}$ maximizing the $\mathcal{F}$. This problem has the same mathematical form as quantum problem of Section II A above. But there is one important difference: where Hilbert space inner product comes from. In quantum problem it is an original property of Hilbert spaces used to formulate the problem, but the functional $\mathcal{F}$ is obtained from measurement data. In classic problem *both* scalar product in Hilbert space and functional $\mathcal{F}$ are obtained from measurement data.

To evaluate the result at some given point $\mathbf{y}$ construct localized density (23) and transform it to $\mathbf{f}$-space with (4). Projecting it to a $\mathbf{g}$-localized state in $\mathbf{f}$-space obtain the probability of a given outcome $\mathbf{g}$ (here $\psi_{\mathbf{g}}(\mathbf{f})$ is a $\mathbf{f}$-space state localized at $\mathbf{f}=\mathbf{g}$, similarly to $\mathbf{x}$-state (23), i.e. in $g_i G^{\mathbf{f};-1}_{ij} f_j$ put f_j from (4) after that obtained expression is a function on x_k that can be coupled with (23); for orthogonal bases (7) obtain (27))

$$\langle \psi_{\mathbf{g}} \,|\, \mathcal{U} \,|\, \psi_{\mathbf{y}} \rangle^2 = \frac{\left| \sum\limits_{k=0}^{n-1} \sum\limits_{j,s=0}^{D-1} g_j G^{\mathbf{f};-1}_{js} u_{sk} y_k \right|^2}{\sum\limits_{j,j'=0}^{D-1} g_j G^{\mathbf{f};-1}_{jj'} g_{j'} \sum\limits_{k,k'=0}^{n-1} y_k G^{\mathbf{x};-1}_{kk'} y_{k'}} \tag{28}$$

The optimization result is u_{jk} matrix, $j=0\ldots D-1; k=0\ldots n-1$. This operator, given some input state (such as localized state $|\psi_{\mathbf{x}}\rangle$), uniquely (within a phase) finds the function in $\mathbf{f}$-space $|\mathcal{U}|\psi_{\mathbf{x}}\rangle$ (coefficients $a_j(\mathbf{x})$) that predicts the probability (28) of outcome $\mathbf{f}$:

$$P(\mathbf{f})\Big|_{\mathbf{x}} = \langle \psi_{\mathbf{f}} \,|\, \mathcal{U} \,|\, \psi_{\mathbf{x}} \rangle^2 = \frac{\left| \sum\limits_{j=0}^{D-1} a_j f_j \right|^2}{\sum\limits_{j,j'=0}^{D-1} f_j G^{\mathbf{f};-1}_{jj'} f_{j'}} \tag{29}$$

$$a_j = \sum_{s=0}^{D-1} \sum_{k=0}^{n-1} G^{\mathbf{f};-1}_{js} u_{sk} x_k \sqrt{K(\mathbf{x})} \tag{30}$$

the $\mathbf{f}$ is equal to the value of the outcome we are interested to determine the probability of. Given $\mathbf{x}$ the probability of an outcome $\mathbf{f}$ is squared linear function on f_j multiplied by Christoffel function $K(\mathbf{f})$. This form of probability, a linear function on f_j squared divided by a quadratic form on f_j, can be obtained from many different considerations; the difference between them is in coefficients a_j. The maping (25) is a pure states mapping. For mixed states mapping the probability will be a ratio of two general quadratic forms instead of rank one matrix in the nominator (29).

For a simple demonstration of creating $\mathbf{x} \to \mathbf{f}$ classic mapping see Section VI below.

C. Learning Unitary Dynamics

In nature most of dynamic equations are equivalent to a sequence of unitary transformations (Newton, Maxwell, Schrödinger equations). Consider a simple classic problem. Let there is an initial state vector $\mathbf{X}^{(0)}$ of unit L^2 norm and a unitary matrix $\mathcal{U}$. The operator is applied

to $\mathbf{X}^{(0)}$ M times:

$$\mathbf{X}^{(l+1)} = \mathcal{U}\mathbf{X}^{(l)} \tag{31}$$

From this sequence M observations $(\mathbf{x}, \mathbf{f})$ (17) are created by taking $(l, l+1)$ elements of the sequence and multiplying them by a random phase (or ± 1 for real space).

$$\mathbf{x}^{(l)} = \exp(i\xi_l)\mathbf{X}^{(l)} \tag{32a}$$

$$\mathbf{f}^{(l)} = \exp(i\zeta_l)\mathbf{X}^{(l+1)} \tag{32b}$$

The random phases make any $\mathbf{x} \leftrightarrow \mathbf{f}$ regression-type methods inapplicable. From unitary property of operator $\mathcal{U}$ we immediately get (6), it does not depend on random phases. Whereas in quantum problem we have a Hilbert space with an inner product $\langle\cdot\rangle$, in this classic problem there is no built in inner product available. The only available average is the sum (18) over M observations, there is no any other average such as ensemble, quantum measurement, etc. For unitary dynamic with $\omega^{(l)} = 1$ this sum is regular time-average. The problem becomes: to determine operator $\mathcal{U}$ from sample (32), $l = 0 \ldots M-1$, that undergoes unitary time-evolution (31). The goal is to determine a matrix u_{jk} (with $D = n$) maximizing quality criterion $\mathcal{F}$ (5).

Gram matrices $G^{\mathbf{x}}_{kk'}$ (19) and $G^{\mathbf{f}}_{jj'}$ (20) are time-average. The functional $\mathcal{F}$ is also time-average of (31) data

$$\mathcal{F} = \sum_{l=1}^{M} \omega^{(l)} \left| \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} X_j^{(l+1)} u_{jk} X_k^{(l)} \right|^2 \underset{u_{jk}}{\longrightarrow} \max \tag{33}$$

Using (32) obtain (for real space)

$$S_{jk;j'k'} = \sum_{l=1}^{M} \omega^{(l)} f_j^{(l)} x_k^{(l)} f_{j'}^{(l)} x_{k'}^{(l)} \tag{34}$$

Random phases (± 1 for real space) in (32) do not affect $S_{jk;j'k'}$ and Gram matrices $\langle x_k x_{k'} \rangle$, $\langle f_j f_{j'} \rangle$ as the phases cancel each other in probabilities. The Gram matrices are not necessary unit matrices. This can be changed by basis regularization. Introduce regularization transformations $R^{\mathbf{x}}$ and $R^{\mathbf{f}}$ such that

$$\mathfrak{x} = R^{\mathbf{x}}\mathbf{x} \tag{35a}$$

$$\mathfrak{f} = R^{\mathbf{f}}\mathbf{f} \tag{35b}$$

produce unit Gram matrices; this can be for example $R^{\mathbf{x}} = G^{\mathbf{x};-1/2}$, $R^{\mathbf{f}} = G^{\mathbf{f};-1/2}$ or any other, e.g. Gram-Schmidt with pivoting or QR decomposition. Then the transform (4) can be written in the form

$$\mathfrak{f} = R^{\mathbf{f}}\mathcal{U}R^{\mathbf{x};-1}\mathfrak{x} \tag{36}$$

and we can consider an optimization problem for operator $\widetilde{\mathcal{U}} = R^{\mathbf{f}}\mathcal{U}R^{\mathbf{x};-1}$ instead of for $\mathcal{U}$. The solution in the original basis is then $\mathcal{U} = R^{\mathbf{f};-1}\widetilde{\mathcal{U}}R^{\mathbf{x}}$, see `com/polytechnik/kgo/DemoRecoverUnitaryMatrixFromSeq.java:getUFromConfigGramMatrixChannel` for an implementation.

A question can be asked whether in the $D = n$ case of data (32) an operator of system dynamics $\mathcal{U}$ having the matrix u_{jk} maximizing $\mathcal{F}$ subject to (6) constraints will always be unitary? It depends on the data. For a data of unitary dynamics (31) Gram matrices $\langle x_k x_{k'}\rangle$ and $\langle f_j f_{j'}\rangle$ must be the same since unitary $\mathcal{U}$ preserves scalar product. But if the data contains non-unitary contribution — Gram matrices $\langle x_k x_{k'}\rangle$, $\langle f_j f_{j'}\rangle$ can be different and this difference contributes to non-unitarity of u_{jk}; the constraints (6) preserve Gram matrix passing through quantum channel. This is the meaning of the constraints — Gram matrix must to transform from Hilbert space IN into Hilbert space OUT as any other operator. The idea is to measure Gram matrix from sample in both Hilbert spaces, construct u_{jk} by solving optimization problem and then generalize the model stating that any other operator converts the same (37). If other than Gram matrix operators are available in both Hilbert spaces — they can be used to build a quantum channel in exactly the same manner. Consider unit matrix transformation what corresponds to traditional unitary learning.

D. Traditional Unitary Learning

Consider "traditional" unitary learning. For (32) data sample it is postulated that operator $\mathcal{U}$ (31) is unitary hence Gram matrix properties are irrelevant to the task. The problem becomes: maximize $\mathcal{F}$ (9) subject to unitary $\mathcal{U}$, i.e. (8) constraints with $D = n$. This is exactly the problem considered in Section II C above but with a quantum channel transforming (37) a unit matrix from Hilbert space IN into a unit matrix in Hilbert space OUT (instead of Gram matrix). Practically this unit matrix invariance can be implemented exactly as considered above, the only difference — no regularization (35) should be performed at all

since it is postulated that $\mathcal{U}$ must always be unitary. The $\mathbf{f}$ and $\mathbf{x}$ should be used directly as is when constructing $S_{jk;j'k'}$ (34), see `com/polytechnik/kgo/DemoRecoverUnitaryMatrixFromSeq.java:getUFromConfigUnitMatrixChannel` for an implementation. Contrary to Gram matrix quantum channel of previous section the $\mathcal{U}$ obtained with unit matrix quantum channel is always unitary.

For a simple demonstration of recovering u_{jk} from unitary dynamics data (32) see Section IV below.

E. Algebraic Structure of the Optimization Problem

Formulated optimization problem maximizes (9) subject to partial unitarity constraint (8). This is a variant of QCQP problem. We find an operator $\mathcal{U}$ optimally transforming (on given measurement data) a state $|\psi(\mathbf{x})\rangle$ from Hilbert space IN (of dimension n) into a state $|\phi(\mathbf{f})\rangle$ from Hilbert space OUT (of dimension D). The operator is a rectangular matrix of the dimension $D \times n$ transforming an operator A from IN to OUT as

$$A^{OUT} = \mathcal{U} A^{IN} \mathcal{U}^\dagger \tag{37}$$

This transform converts any operator between two Hilbert spaces, for example a pure state $A^{IN} = |\psi\rangle\langle\psi|$ into a pure state $A^{OUT} = |\mathcal{U}|\psi\rangle\langle\psi|\mathcal{U}|$, for a more general form see Kraus operators (77) below. There always should be an operator known in both Hilbert spaces, it is used to create the constraints on $\mathcal{U}$ when maximizing the fidelity — these constraints (probability preservation) determine the specific form of partial unitarity. We consider two such operators: Gram matrix (most of this paper) and unit matrix (traditional unitary learning in Section II D). Constrained optimization problem on $\mathcal{U}$ is reduced to a new algebraic problem (this is a variation of Lagrangian $\mathcal{L}$ (48) set to zero)

$$S\mathcal{U} = \lambda\mathcal{U} \tag{38}$$

which is remotely similar to an eigenvalue problem, but S is a Hermitian tensor, "eigenvector" $\mathcal{U}$ is partially unitary $D \times n$ matrix, and "eigenvalues" λ is a Hermitian $D \times D$ matrix; functional extremal value $\mathcal{F}$ is equal to λ spur (the sum of diagonal elements). The algebraic structure of this eigenvalue–like problem, let us call it an "**eigenoperator**" problem, requires a separate study. Currently we only have a numeric solution algorithm.

III. NUMERICAL SOLUTIOIN

The problem becomes: optimize (39) subject to (40) constraints

$$\mathcal{F} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} S_{jk;j'k'} u^*_{j'k'} \underset{u}{\longrightarrow} \max \tag{39}$$

$$\delta_{jj'} = \sum_{k=0}^{n-1} u_{jk} u^*_{j'k} \qquad\qquad j, j' = 0 \ldots D-1 \tag{40}$$

The tensor $S_{jk;j'k'} = S^*_{j'k';jk}$ is Hermitian, for simplicity we consider it and u_{jk} to be real and do not write complex conjugated * below and use the terms *unitary* and *orthogonal* interchangeably; a generalization to complex values is straightforward. If we consider a subset of all (40) constraints the optimization problem can be readily solved. Consider the squared Frobenius norm of matrix u_{jk} to be a "simplified constraint":

$$\sum_{j=0}^{D-1} \sum_{k=0}^{n-1} u^2_{jk} = D \tag{41}$$

This is a partial constraint (it is the sum of all (40) diagonal elements). For this partial constraint the optimization problem (39) is equivalent to an eigenvalue problem — it can be directly solved by considering a vector of Dn dimension obtained from u_{jk} operator by saving all it's components to a single vector, row by row.

The main idea of [19] was to modify partial constraint solution to satisfy the full set of the constraints (40). There are several options to perform solution adjustment from partial to full set of constraints. The one producing the minimal change to the solution is an application of Gram matrix

$$G^u_{jj'} = \sum_{k=0}^{n-1} u_{jk} u_{j'k} \tag{42}$$

inverse square root $G^{u;-1/2} = 1/\sqrt{G^u}$ to u_{jk}.[1] There are 2^D square roots of a positively definite symmetric matrix dimension D, different within the $\pm$ signs. The simplest method to calculate it — convert $G^u_{jj'}$ to diagonal form in the basis of $G^u_{jj'}$ eigenvectors

$$\left|G^u|g^{[i]}\right\rangle = \lambda^{[i]}_G \left|g^{[i]}\right\rangle \tag{43}$$

[1] See Appendix A of [19] for adjustments in different bases and for an approach that uses SVD instead of eigenproblem (43). Also note that both SVD-based and eigenproblem-based transforms convert a single u_{jk} state satisfying partial constraint (41) to the state satisfying the full set of the constraints (40). A problem of converting *several* $u^{[s]}_{jk}$ states satisfying partial constraint to a single state satisfying the full set of the constraints can greatly improve algorithm's initial convergence, this is a subject of future research.

then change the eigenvalues to $\pm 1 \Big/ \sqrt{\lambda_G^{[i]}}$ and convert the matrix back to initial basis

$$\left\| G^{u;-1/2} \right\| = \sum_{i=0}^{D-1} \frac{\pm 1}{\sqrt{\lambda_G^{[i]}}} \left| g^{[i]} \right\rangle \left\langle g^{[i]} \right| \tag{44}$$

By checking the result one can verify that for any u_{jk} producing non-degenerated Gram matrix (42) the vector

$$\widetilde{u}_{jk} = \sum_{i=0}^{D-1} G_{ji}^{u;-1/2} u_{ik} \tag{45}$$

satisfies all the constraints (40) (the most general form of an adjustment is an application of $BG^{u;-1/2}$ to u_{ik} where B is an arbitrary unitary operator, the $\pm$ square root signs can be included into B; below we consider all signs in (44) to be "+"), see `com/polytechnik/kgo/AdjustedStateToUnitaryWithEigenproblem.java` for an implementation. Thus multiple constraints optimization problem (39) can be reduced to an unconstrained optimization problem (we use $D = \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} u_{jk}^2 = \sum_{j,j'=0}^{D-1} \sum_{k=0}^{n-1} u_{jk} G_{jj'}^{u;-1} u_{j'k}$ identity):

$$\mathcal{F} = \frac{\sum\limits_{j,j',i,i'=0}^{D-1} \sum\limits_{k,k'=0}^{n-1} u_{jk} G_{ji}^{u;-1/2} S_{ik;i'k'} G_{j'i'}^{u;-1/2} u_{j'k'}}{\frac{1}{D} \sum\limits_{j=0}^{D-1} \sum\limits_{k=0}^{n-1} u_{jk}^2} \underset{u}{\longrightarrow} \max \tag{46}$$

However, this unconstrained problem[2]

- cannot be reduced to an eigenvalue problem since $G_{ji}^{u;-1/2}$ itself depends on u_{jk}.
- is degenerate: there are multiple u_{jk} producing the same $\mathcal{F}$. Convert a solution u_{jk} to Gram matrix basis (43), change the eigenvalues, then convert back to the initial basis, this is similar to (44) transform. Hence the Hessian matrix is degenerated, this prevents us from direct application of Newton's and quasi-Newton optimizations. One can possibly use a penalty function like $\sum 1/\lambda_G$

  $$\mathrm{Spur} G^{u;-1} = \sum_{i=0}^{D-1} \frac{1}{\lambda_G^{[i]}} \tag{47}$$

  that has the minimal value D when all the constraints (40) are satisfied.

[2] There are alternative ways to obtain an unconstrained optimization problem. In the unitary case $D = n$ one can use $D(D+1)/2$ parameters to parameterize a Hermitian matrix, a unitary matrix is then obtained with matrix exponentiation (11); functional optimization, however, requires the derivatives of matrix exponent what complicates the problem[22, 23]. There is an option to parameterize a unitary matrix with Cayley transform $U = (I - A)(I + A)^{-1}$ for a skew-Hermitian matrix $A^\dagger = -A$. Both methods are problematic to use, especially in the case of rectangular u_{jk}, with $D < n$. See also [24] about other ways of unitary parametrization.

Alternatively an iteration approach with Lagrange multipliers can be used. Iterations are required since we cannot simultaneously solve the equation for u_{jk} and $\lambda_{jj'}$. A single iteration consists in solving an eigenvalue problem, adjusting obtained solution to satisfy the full set of the constraints, and calculating the new values of Lagrange multipliers; there are three building blocks of the algorithm:

- Eigenproblem solution (56) to solve partially constrained optimization problem. An important feature is that any additional linear constraints on u_{jk} of (57) form can be easily incorporated — obtain the eigenproblem (63).

- Solution adjustment operation (45) that converts a partial constraint solution (41) to a full set one (40).

- Linear system solution (52) to obtain new values of Lagrange multipliers.

In it's naïve form a convergence of the iteration algorithm turned out to be poor. The major new result of the current paper is an iteration algorithm with a *good convergence*. A good convergence was achieved by considering additional *linear* constraints to eigenproblem at iteration step. Consider Lagrange multipliers $\lambda_{jj'}$, a matrix of $D \times D$ dimension, to approach optimization problem (39) with the constraints (40)

$$\mathcal{L} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} S_{jk;j'k'} u_{j'k'} + \sum_{j,j'=0}^{D-1} \lambda_{jj'} \left[\delta_{jj'} - \sum_{k'=0}^{n-1} u_{jk'} u_{j'k'} \right] \underset{u}{\longrightarrow} \max \tag{48}$$

Variating $\mathcal{L}$ over u_{sq} obtain (50)

$$b_{sq} = \sum_{j'=0}^{D-1} \sum_{k'=0}^{n-1} S_{sq;j'k'} u_{j'k'} \tag{49}$$

$$0 = b_{sq} - \sum_{j'=0}^{D-1} \lambda_{sj'} u_{j'q} \tag{50}$$

Here and below we consider Lagrange multipliers matrix to be Hermitian $\lambda_{jj'} = \lambda_{j'j}$, this is a condition of extremal solution existence[19]. The variation (50) contains Dn equations. The Hermitian Lagrange multipliers matrix $\lambda_{jj'}$ has D^2 real parameters ($D(D+1)/2$ independent ones) for real space and $2D^2$ real parameters (D^2 independent ones) for complex space. Thus for a general u_{jk} the variation (50) cannot be satisfied in full. The most straightforward way

to obtain Lagrange multipliers for a given u_{jk} is to take L^2 norm of variation (50) and obtain the λ_{ij} minimizing the sum of squares.

$$\sum_{i=0}^{D-1}\sum_{q=0}^{n-1}\left|b_{iq}-\sum_{j=0}^{D-1}\frac{\lambda_{ij}+\lambda_{ji}}{2}u_{jq}\right|^2 \underset{\lambda_{ij}}{\longrightarrow} \min \tag{51}$$

$$\lambda_{ij}=\mathrm{Herm}\sum_{k=0}^{n-1}u_{ik}b_{jk}=\mathrm{Herm}\sum_{j'=0}^{D-1}\sum_{k,k'=0}^{n-1}u_{ik}S_{jk;j'k'}u_{j'k'} \tag{52}$$

The minimization gives Hermitian matrix λ_{ij} which is obtained as a linear system solution; see `com/polytechnik/kgo/LagrangeMultipliersPartialSubspace.java:calculateRegularLambda` for an implementation. A more general form of λ_{ij} is presented in Appendix A below. The problem can now be considered as maximizing a quadratic form with the matrix $\mathcal{S}_{jk;j'k'}$

$$\mathcal{S}_{jk;j'k'}=S_{jk;j'k'}-\lambda_{jj'}\delta_{kk'} \tag{53}$$

$$\sum_{j,j'=0}^{D-1}\sum_{k,k'=0}^{n-1}u_{jk}\mathcal{S}_{jk;j'k'}u_{j'k'} \underset{u}{\longrightarrow} \max \tag{54}$$

subject to constraints (40). Consider the eigenproblem[3]

$$\frac{\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k,k'=0}^{n-1}u_{jk}\mathcal{S}_{jk;j'k'}u_{j'k'}}{\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k=0}^{n-1}u_{jk}Q_{jj'}u_{j'k}} \underset{u}{\longrightarrow} \max \tag{55}$$

$$\sum_{j'=0}^{D-1}\sum_{k'=0}^{n-1}S_{sq;j'k'}u^{[s]}_{j'k'}-\sum_{j'=0}^{D-1}\lambda_{sj'}u^{[s]}_{j'q}=\mu^{[s]}\sum_{j'=0}^{D-1}Q_{sj'}u^{[s]}_{j'q} \tag{56}$$

with additional N_d linear constraints added (their specific form providing a good convergence of the algorithm is discussed below in Appendix A 1)

$$0=\sum_{j=0}^{D-1}\sum_{k=0}^{n-1}C_{d;jk}u_{jk} \qquad\qquad d=0\ldots N_d-1 \tag{57}$$

A common method of solving eigenproblem (55) with homogeneous linear constraints (57) is Lagrange multipliers method[25]. This approach, however, creates a difficulty when both

[3] From the variation (50) it follows that the most interesting $u^{[s]}_{jk}$ states have the value of functional (55) close to zero thence the matrix $Q_{jj'}$ in the denominator can be chosen as an arbitrary positively definite Hermitian matrix to improve convergence. The $Q_{jj'}=\delta_{jj'}$ gives familiar results; the other choice can be $Q_{jj'}=\lambda_{jj'}$, i.e. have Lagrange multipliers in denominator instead of adding them to (53): consider a variation of $\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k,k'=0}^{n-1}u_{jk}S_{jk;j'k'}u_{j'k'}\Big/\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k=0}^{n-1}u_{jk}\lambda_{jj'}u_{j'k}$ over u_{jk} to obtain the same (50); see `com/polytechnik/kgo/KGOIterationalLagrangeMultipliersInDenominatorU.java`.

linear and quadratic constraints are present, especially when the number of constraints is large. A better approach to deal with linear constraints is to convert (57) to a form that expresses $\mathrm{rank}(C_{d;jk})$ components of u_{jk} via it's other components, the coefficients by the selected components $\widetilde{C}_{d;jk}$ are zero (we plain moved some of (57) terms from right to left hand side).

$$u_{j^{[d]}k^{[d]}} = \sum_{j=0}^{D-1}\sum_{k=0}^{n-1}\widetilde{C}_{d;jk}u_{jk} \qquad\qquad d = 0\ldots\mathrm{rank}(C_{d;jk})-1 \tag{58}$$

Then any u_{jk} satisfying (57) linear constraints can be expresses as a linear combination of selected components, let us denote them as some general variables V_p, $p = 0\ldots N_V - 1$

$$N_V = Dn - \mathrm{rank}(C_{d;jk}) \tag{59}$$

$$u_{jk} = \sum_{p=0}^{N_V-1} M_{jk;p}V_p \tag{60}$$

Substitute (60) back to (55) to obtain an unconstrained generalized eigenproblem (62) relatively N_V generalized variables V_p; linear constraints (57) are incorporated into new variables V_p, the number of variables is reduced by $\mathrm{rank}(C_{d;jk})$.[4] The (60) transform is actually a regular Gaussian elimination, a special form of LU decomposition. A simple implementation with row and column pivoting is used in `com/polytechnik/utils/EliminateLinearConstraints_HomegrownLUFactorization.java`; for a very large number of linear constraints a transform like RRQR factorization is probably more numerically stable. The new eigenproblem has the matrices $\mathcal{S}_{p;p'}$ and $Q_{p;p'}$ in nominator and denominator

$$\mathcal{S}_{p;p'} = \sum_{j,j'=0}^{D-1}\sum_{k,k'=0}^{n-1} M_{jk;p}\mathcal{S}_{jk;j'k'}M_{j'k';p'} \tag{61a}$$

$$Q_{p;p'} = \sum_{j,j'=0}^{D-1}\sum_{k=0}^{n-1} M_{jk;p}Q_{jj'}M_{j'k;p'} \tag{61b}$$

the $\lambda_{p;p'}$ converts the same. We can write the eigenproblem (55) in the form

$$\frac{\sum\limits_{p,p'=0}^{N_V-1} V_p\mathcal{S}_{p;p'}V_{p'}}{\sum\limits_{p,p'=0}^{N_V-1} V_pQ_{p;p'}V_{p'}} \underset{V}{\longrightarrow} \max \tag{62}$$

[4] Note that at this stage some of the vectors that were removed with the constraints can be possibly injected back into V_p basis; they can be directly added as additional column(s) to $M_{jk;p}$. This only changes the size of V_p basis $N_V = Dn - \mathrm{rank}(C_{d;jk}) + N_{inj}$. A better option, however, is to modify the $C_{d;jk}$ in the first place and avoid basis injection.

$$\sum_{p'=0}^{N_V-1} S_{p;p'} V_{p'}^{[s]} - \sum_{p'=0}^{N_V-1} \lambda_{p;p'} V_{p'}^{[s]} = \mu^{[s]} \sum_{p'=0}^{N_V-1} Q_{p;p'} V_{p'}^{[s]} \tag{63}$$

The cost of this conversion is that if originally we put $Q_{jj'} = \delta_{jj'}$ then in V_p basis the denominator in (62) is no longer a unit matrix. This is not an issue since any modern linear algebra package internally converts a generalized eigenproblem to a regular one, see e.g. DSYGST, DSPGST, DPBSTF and similar subroutines.

Let the original eigenproblem (56) be already solved and extremal $u_{jk}^{[s]}$ satisfying partial quadratic constraint (41) are obtained. Consider Lagrangian variation $\delta\mathcal{L}/\delta u_{sq}$. If the state u_{jk} is (55) extremal then the variation (50) is zero

$$0 = \sum_{j'=0}^{D-1} \sum_{k'=0}^{n-1} S_{sq;j'k'} u_{j'k'} - \sum_{j'=0}^{D-1} \breve{\lambda}_{sj'} u_{j'q} \tag{64}$$

$$\breve{\lambda}_{jj'} = \lambda_{jj'} + \mu^{[s]} Q_{jj'} \tag{65}$$

It is *exactly zero* (for all Dn equations) if u_{jk} is one of $u_{jk}^{[s]}$ and Lagrange multipliers are (65). This u_{jk}, however, may not satisfy the full set of the constraints (40). It is adjusted with (45) to satisfy all the constraints, and a new $\lambda_{jj'}$ is calculated to use in the problem (56). A difficulty we met in [19, 20] is that this iteration algorithm when $\lambda_{jj'}$ (52) is used — it does not converge to a solution, only given large enough iterations number it produces a good enough solution among iterations seen. In this paper this difficulty is overcame by solving the eigenproblem (56) with extra linear constraints (57). The improved iteration algorithm becomes:

1. Take initial λ_{ij} and linear constraints $C_{d;jk}$ to solve optimization problem (62) with respect to V_p. Solution method — an eigenvalue problem of N_V dimension, the number of columns in $M_{jk;p}$ matrix. A new u_{jk} is obtained from V_p using (60). The result: $s = 0 \ldots N_V - 1$ eigenvalues $\mu^{[s]}$ and corresponding matrices $u_{jk}^{[s]}$ reconstructed from $V_p^{[s]}$. The value of N_V is typically $Dn - (D-1)(D+2)/2$.

2. A heuristic is required to select the u_{jk} among all N_V eigenstates. Trying a number of them and selecting the maximum (i.e. from all $s = 0 \ldots N_V - 1$ select the best state among top eigenvalues $\mu^{[s]}$: try all positive and 10 highest negative) providing a large value of the original functional (46) typically gives only a local maximum. Numerical experiments show that the index of eigenstate is a good invariant: *always* selecting the

state $V_p^{[s]}$ of: largest $\mu^{[s]}$, second largest $\mu^{[s]}$, third largest $\mu^{[s]}$, etc. converges to a different solution of the original problem. The global maximum of $\mathcal{F}$ typically corresponds to the largest $\mu^{[s]}$ selection. A good heuristic is to run algorithm $\mathfrak{n}$ times, always selecting the state of $\mathfrak{n}$-th largest $\mu^{[s]}$. Then select the global maximum among these $\mathfrak{n}$ runs; the remaining $\mathfrak{n}-1$ solutions are also good — this way we managed to obtain up to a dozen of different solutions. At worst — a cycle without convergence (usually with a period of 2 iterations) is observed; this was a problem back in [19]. With the linear constraints technique of Appendix A 1 this difficulty is overcame.

3. Obtained u_{jk} is not partially unitary as the constraint (41) is a subset of the full ones (40). Apply the adjustment (45) and calculate the λ_{ij} (52) in the adjusted state, this is the Lagrange multipliers for the next iteration.

4. For a good convergence, in addition to λ_{ij}, we need to select a subspace for u_{jk} next iteration variation. Using full size Dn basis gives a poor convergence [19]. There are two feasible options to improve it: either use some advanced method of λ_{ij} calculation, see Appendix A below, or constraint the subspace of u_{jk} variation, see Appendix A 1 below; the later technique of additional linear constraints $C_{d;jk}$ (57) provides superior results.

5. Put this new λ_{ij} to (53) and, using the basis V_p obtained (60) from $C_{d;jk}$, calculate generalized eigenproblem nominator and denominator matrices (61) to be used for the next iteration. Repeat iteration process until converged to a maximum (presumably global) of $\mathcal{F}$ with u_{jk} satisfying the constraints (40). If a convergence is achieved the λ_{ij} stops changing from iteration to iteration and the $\mu^{[s]}$ of the selected state in step 2 above is close to zero. On the first iteration take initial values of Lagrange multipliers $\lambda_{ij}=0$ and no linear constraints.

Based on a number of numerical experiments we can possibly conclude that this iteration algorithm always converges. A determination of exact convergence domain is a subject of future research. A distinctive feature of this algorithm is that instead of usual iteration internal state in the form of a pair: approximation, Lagrange multipliers, (u_{jk},λ_{ij}) it uses iteration internal state in the form of a triple: approximation, Lagrange multipliers, homogeneous linear constraints, $(u_{jk},\lambda_{ij},C_{d;jk})$. Whereas most optimization algorithms have linear system

solution (Newtonian iteration) as a building block, the algorithm in question has eigenproblem solution as the building block. This allows us to develop a much more fine-grained solution selection in step 2 above, what makes the algorithm less sensitive to degeneracy and more likely to converge to the global maximum. The drawback of using eigenproblem solution as a building block is that it is more computationally costly than a linear system solution or a first order gradient method. However the main goal of the paper is to present a working proof of concept algorithm capable to solve a new algebraic "eigenoperator" problem (38); computational complexity of optimization is a separate task. Among evident optimizations — since in step 2 it is usually sufficient always to select the state of maximal eigenvalue a replacement of a general purpose eigenproblem solver by a one finding only the largest eigenvalue is expected greatly to increase algorithm performance.

A reference implementation of this algorithm is available at `com/polytechnik/kgo/KGOIterationalSubspaceLinearConstraints.java`. There are dozens of other algorithms implemented, but only this one provides iterations convergence. There is a driver in `com/polytechnik/kgo/KGOSolutionVectorXVectorF.java:main` class for an algorithm quick test. The driver generates a deterministic random sample from which the tensor $S_{jk;j'k'}$ is calculated and then given algorithm (taken from command argument) is applied to the problem of finding optimal partially unitary operator. Let us compare a convergence of this paper iteration algorithm and the result of [19] (vanilla Lagrange multipliers) on driver's deterministic randomly generated data with default settings: $D = 4$ and $n = 19$ of $M = 13540$ observations.

```
java com/polytechnik/kgo/KGOSolutionVectorXVectorF ITERATIONS_KGOIterationalSubspaceLinearConstraints 2>&1 | grep Selected
```

and

```
java com/polytechnik/kgo/KGOSolutionVectorXVectorF ITERATIONS_KGOIterationalSimpleOptimizationU 2>&1 | grep Selected
```

The output is `grep`-filtered to show only iteration status. The result is presented in Table I. From the table we see that the algorithm is cycling a number of initial iterations without a convergence, then when it hits an area of contraction mapping — the convergence becomes very fast. Plain vanilla Lagrange multipliers method does not converge at all: full Dn space optimization of (55) with λ_{ij} severely breaks the constraints (40) and problem degeneracy impede the convergence. This can be demonstrated by running the `KGOIterationalSubspa`

TABLE I. Convergence comparison for this paper algorithm and the one from our previous work [19] on data sample with $D = 4$, $n = 19$, $M = 13540$ for the first $i = 0 \dots 17$ iterations. We present: μ – the eigenvalue of selected state, $\mathcal{F}$ – the value of (46) functional, and an indicator of unitarity $\sum 1/\lambda_G$.

	KGOIterationalSubspaceLinearConstraints			KGOIterationalSimpleOptimizationU		
i	μ	$\mathcal{F}$	$\sum 1/\lambda_G$	μ	$\mathcal{F}$	$\sum 1/\lambda_G$
0	2296.97	7864.08	38.86	2296.97	7864.08	38.86
1	201.08	7936.32	20.02	239.84	8020.58	34.94
2	113.98	8010.69	27.04	110.62	8032.89	17.70
3	100.82	8178.92	7.07	53.94	8014.30	14.31
4	64.91	7890.37	26.06	150.60	8138.49	15.82
5	204.71	8078.93	32.47	57.33	8113.74	11.39
6	134.24	7873.87	34.30	109.05	8071.24	17.90
7	201.39	8112.94	24.76	38.24	8005.98	11.93
8	90.78	7869.54	34.38	92.91	8059.80	18.31
9	159.77	8057.35	12.58	49.79	8101.24	6.51
10	83.15	8212.67	8.06	148.46	8094.14	29.02
11	28.15	8194.82	6.22	86.11	8035.41	16.24
12	25.06	8292.05	4.03	156.60	8060.77	26.22
13	3.08	8302.52	4.02	95.37	8112.07	17.72
14	0.23	8303.46	4.00	73.73	8016.44	23.42
15	$5.63 \cdot 10^{-4}$	8303.46	4.00	202.11	8040.93	32.05
16	$1.05 \cdot 10^{-8}$	8303.46	4.00	93.91	7963.84	21.20
17	$2.39 \cdot 10^{-13}$	8303.46	4.00	192.02	8030.50	24.14

`ceLinearConstraints` in a standard way for 20 iterations to obtain a perfect solution, then starting at iteration 21 turn off linear constraints (A8), i.e. internally switch it to vanilla Lagrange multipliers method. The result — for the next 4 to 7 iterations the solution stays almost the same, but after 7-10 iterations it is starting to diverge (due to accumulated floating point errors) and an irregular cycling without convergence is observed. This shows a

critical importance of linear constraints (A8) for iteration algorithm convergence and intrinsic instability of vanilla methods.

There is a number of QCQP software packages, both commercial and open source: [26], python qcqp, NAG QCQP among many others. A question arize how well existing software can handle a QCQP optimization problem of this paper? We did not test third party software much (in the future we are going to test existing software much more extensively), but generally — the results were unsatisfactory, especially in finding the *global* maximum. The reason is that existing QCQP software packages were designed as general purpose solvers intended to solve any QCQP problem, many of them are heuristic Newton-style solvers with Lagrange multipliers often combined with some form of linear programming and convex optimization. The optimization problem considered in this paper is special: it has only quadratic equality constraints, the problem is non-convex, it has multiple solutions, local maximums, and multiple saddle points. For such a problem the solvers based on a single solution iteration (e.g. Newtonian type (second order), gradient type (first order), etc.) are unlikely to determine the global maximum. One may try solvers of Genetic programming type, but they lack the knowledge about underline algebraic structure of the problem. The problem of finding an operator optimally converting an operator from one Hilbert space to another requires special solver. Our use of generalized eigenvalue problem as algorithm building block has an advantage of obtaining at once many solution candidates (eigenvectors); with proper selection we can greatly increase the chances of finding the global maximum.

Compared to [19] the quality of our optimization algorithm was greatly improved and we are going to use it as a "black box" to apply to several problems for demonstration. Let us demonstrate the value of finding a partially unitary operator optimally transforming a vector from Hilbert space *IN* to Hilbert space *OUT*.

IV. A DEMONSTRATION OF RECOVERING UNITARY DYNAMICS FROM PHASE-STRIPPED DATA

Consider an inverse problem. Let a dynamic system to evolve with $\mathbf{X}^{(l+1)} = \mathcal{U}\mathbf{X}^{(l)}$ equation (31). The problem is to recover orthogonal operator $\mathcal{U}$ from an observed sample $\mathbf{X}^{(l)}$. The problem would be trivial if $\mathbf{X}^{(l)}$ are directly observed — a regression analysis can reveal $\mathcal{U}$, see e.g. Section "An application of LRR representation solution to dynamic

system identification problem" of [20]. What greatly complicates the matter — we consider observations of quantum channel type hence the $\mathbf{X}^{(l)}$ are observable only within a phase. This means that any wavefunction mapping (2) can be observed only within an unknown phase; this can be modeled by a multiplication of actually measured classic values by random phase factors. The problem becomes: from (32) sample to determine an operator $\mathcal{U}$.[5]

In this section a few simple examples are presented. All the calculations are performed in real space, thus a ± 1 factor is used instead of a complex phase factor. A timeserie is initially generated with (31) then obtained $\mathbf{X}^{(l)}$ are multiplied by random ± 1 factors, these new vectors (32) are now considered as observables. The problem is to recover $\mathcal{U}$ from observable data. We present a recovery with two quantum channels: transforming Gram matrix of Section II C and transforming unit matrix of Section II D. Since the unitarity of test data is exact — both methods recover underline operator $\mathcal{U}$ exactly, thus only Gram matrix quantum channel is presented below.

Let us start with a simple SO(3) rotation group. Rotation matrix can be represented via Euler angles (φ, θ, ψ) (in a space of dimension d there are total $d(d-1)/2$ angles).

$$g(\varphi,\theta,\psi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{66}$$

Take $\mathcal{U} = g(\varphi = 0.1, \theta = 0.4, \psi = 0.7)$

$$\mathcal{U} = \begin{pmatrix} 0.7018 & -0.7113 & 0.0389 \\ 0.6668 & 0.6366 & -0.3875 \\ 0.2509 & 0.2978 & 0.9211 \end{pmatrix} \tag{67}$$

and apply 1000 transformations to initial vector $(0.0921, 0.5523, 0.8285)$. The datafile `SO3.csv` is generated with

```
java com/polytechnik/algorithms/PrintOrthogonalSeq
Writing 1000 points to /tmp/SO3.csv .
```

There is a simple demo-program that recovers operator $\mathcal{U}$ from sampled data (32) with random phases ± 1 possibly introduced into observations; this program maximizes (33) with a

[5] The operator itself is defined within a phase. More degeneracy may come form the data: e.g. if some components X_j and X_{j+1} are always stay the same in the sample — then we cannot distinguish a unit matrix from a permutation matrix.

crude regularization (36) (numerical stability depends on regularization, the result is not[20]) and without taking care of possible data degeneracy, but it's simplicity makes it's internals clear.

```
java com/polytechnik/kgo/DemoRecoverUnitaryMatrixFromSeq --data_file_to_bui
ld_unitarymodel_from=/tmp/SO3.csv --data_cols=6:1,3:0
```

The program recovers the $\mathcal{U}$ identically

$$\mathcal{U} = \begin{pmatrix} -0.7018 & 0.7113 & -0.0389 \\ -0.6668 & -0.6366 & 0.3875 \\ -0.2509 & -0.2978 & -0.9211 \end{pmatrix} \tag{68}$$

As we discussed above operator $\mathcal{U}$ is defined within a ± 1 factor in real space. The same test was run on randomly generated orthogonal matrices of the dimensions $3, 5, 7, 17, 40$. All tests can be run automatically as

```
java com/polytechnik/algorithms/PrintOrthogonalSeq\$TestAuto
```

The maximal difference in matrix elements (for all the dimensions tried) is less that 10^{-13} what is about floating point errors. This makes us to conclude that developed numerical algorithm of Section III can recover system dynamics from wavefunction measurements (without a phase) for dynamic systems of high dimension. This algorithm is a powerful method to solve quantum mechanics inverse problem.

V. A DEMONSTRATION OF POLYNOMIAL MAPPING RECOVERING

Consider polynomials in $[-1 : 1]$ interval. Let there are $l = 1 \dots M$ points $y^{(l)}$ split equidistantly in the interval. Define the data

$$x_k^{(l)} = \xi^{(l)} T_k(y^{(l)}) \qquad k = 0 \dots n-1 \tag{69a}$$

$$f_j^{(l)} = \zeta^{(l)} P_j(y^{(l)}) \qquad j = 0 \dots D-1 \tag{69b}$$

Here T_k is Chebyshev polynomial and P_j is Legendre polynomial. The $\xi^{(l)}$ and $\zeta^{(l)}$ are deterministic random functions on l that take the value ± 1. The problem is to build u_{jk} matrix maximizing (5) subject to (6) constraints. The mapping (4) for (69) data has the meaning of finding the coefficients expanding D Legendre polynomials in n Chebyshev polynomials. The problem would be trivial (reduced to a linear system solution) were it not for random ± 1

factors $\xi^{(l)}$ and $\zeta^{(l)}$ in (69). With random phases presence any direct projection of one basis on another becomes unavailable, the only feasible way is to consider quantum channel (4) mapping. The solution is similar to the one of previous section. Calculate Gram matrices (19), (20) and $S_{jk;j'k'}$ tensor (34). After any regularization of input data, for example (36), apply the algorithm of Section III. Specifically, generate a datafile `ChebyshevLegendre.csv` by running

```
java com/polytechnik/algorithms/PrintChebyshevToLegendreMapping
Writing 500 points to /tmp/ChebyshevLegendre.csv
```

This file has 500 $[-1:1]$ points of (69) data with $T_0 \dots T_{10}$ and $P_0 \dots P_5$ randomly multiplied by ± 1 factors. Consider a simple task of converting $T_0 \dots T_4$ to $P_0 \dots P_4$. Use a simple demo-program that recovers the mapping (4) from sampled data (69) with random phases ± 1 possibly introduced into observations, the approach is phase-agnostic; this program maximizes $\mathcal{F}$; program's simplicity makes it's internals clear.

```
java com/polytechnik/kgo/DemoRecoverMapping --data_file_to_build_model_from
=/tmp/ChebyshevLegendre.csv --data_cols=21:2,6:14,18:-1:0
```

The program recovers the mapping identically (the values below 10^{-15} are replaces by 0).

$$\mathcal{U} = \begin{pmatrix} -1.0 & 0 & 0 & 0 & 0 \\ 0 & -1.0 & 0 & 0 & 0 \\ -0.25 & 0 & -0.75 & 0 & 0 \\ 0 & -0.375 & 0 & -0.625 & 0 \\ -0.140625 & 0 & -0.3125 & 0 & -0.546875 \end{pmatrix} \tag{70}$$

Exact values can be obtained using a polynomial library from [27], it is included with this paper software [28]. Run in jshell `new Legendre().convertBasisToPBASIS(5,new Chebyshev())` to obtain P_j over T_k expansion

$$\begin{pmatrix} 1.0 & & & & \\ 0 & 1.0 & & & \\ 0.25 & 0 & 0.75 & & \\ 0 & 0.375 & 0 & 0.625 & \\ 0.140625 & 0 & 0.3125 & 0 & 0.546875 \end{pmatrix} \tag{71}$$

i.e. $P_3 = 0.375T_1 + 0.625T_3$. This matches (70) within a $\pm$ sign. In this $n = D$ case we have a perfect recovery. Now consider the same problem of polynomial mapping for $D < n$, let us

run it with $D = 4$ and $n = 5$.

```
java com/polytechnik/kgo/DemoRecoverMapping --data_file_to_build_model_from
=/tmp/ChebyshevLegendre.csv --data_cols=21:2,6:14,17:-1:0
```

Obtained mapping

$$\mathcal{U} = \begin{pmatrix} 0 & -1.294340 & 0 & 0.560225 & 0 \\ -0.639496 & 0 & -0.449476 & 0 & 0.075960 \\ 0 & -0.676776 & 0 & -0.692739 & 0 \\ -0.149669 & 0 & -0.620827 & 0 & -0.502329 \end{pmatrix} \tag{72}$$

is not a subset of $D = 5$, $n = 5$ mapping (70), a subset can be obtained by running with $D = 4$, $n = 4$. We previously discussed [19] the difficulties of the $D < n$ case. A mapping between two Hilbert spaces of different dimensions sometimes leads to an unusual behavior since the probability, not the value, is maximized. The value of $\mathcal{F}$ increases when going from $D = 4$, $n = 4$ to $D = 4$, $n = 5$ but the mapping is no longer a subset. The behavior is determined by $S_{jk;j'k'}$ choice. In this section a product (34) is used. There are alternative options, for example (27) that uses Christoffel function factor; a few other are discussed in [19], see e.g. dimension adjusted Christoffel function in the Appendix C therein.

VI. A DEMONSTRATION OF FUNCTION INTERPOLATION

In two previous sections we considered examples of constructing u_{jk} when the data of $\mathbf{x} \to \mathbf{f}$ form has vectors $\mathbf{x}$ and $\mathbf{f}$ already to belong to corresponding Hilbert spaces. For classic measurement this is often not the case, a transformation $\mathbf{x} \to \psi$, $\mathbf{f} \to \phi$ is required to build Hilbert space states ψ and ϕ from original observations $\mathbf{x}$ and $\mathbf{f}$. The most straightforward method to construct such states is to consider localized at given $\mathbf{x}$ (or $\mathbf{f}$) states (23) to use them as wavefunctions mapping (25). Considered in section II B above localized states $\psi_{\mathbf{y}}(\mathbf{x})$ use for scalar product the same Gram matrix (19), which is obtained from $l = 1 \dots M$ sampled data. Effectively this is sampled average (18) with weight multiplied by a localized at $\mathbf{y}$ non-negative function $\psi^2_{\mathbf{y}}(\mathbf{x})$ (it is normalized as $1 = \left\langle \psi^2_{\mathbf{y}}(\mathbf{x}) \right\rangle$), it gives a Radon-Nikodym approximation of some characteristic g

$$g_{RN}(\mathbf{y}) = \left\langle g \psi^2_{\mathbf{y}} \right\rangle \tag{73}$$

Here approximated value is a superposition of observed g with positive density $\psi^2_{\mathbf{y}}(\mathbf{x})$. Familiar least squares interpolation (22) that is expanding the value, not a probability, has similar form

$$g_{LS}(\mathbf{y}) = \psi_{\mathbf{y}}(\mathbf{y}) \langle g\psi_{\mathbf{y}} \rangle \tag{74}$$

Here the average is taken with $\psi_{\mathbf{y}}(\mathbf{x})$ density — it is not always positive as the $\psi^2_{\mathbf{y}}(\mathbf{x})$ in Radon-Nikodym (73) is; the $\psi_{\mathbf{y}}(\mathbf{y})$ is normalizing factor $1/\sqrt{K(\mathbf{y})}$. The (73) and (74) differ in how they represent delta function $\delta(\mathbf{x}-\mathbf{y})$: as $\psi^2_{\mathbf{y}}(\mathbf{x})$ or as $\psi_{\mathbf{y}}(\mathbf{y})\psi_{\mathbf{y}}(\mathbf{x})$.

Consider scalar function interpolation problem in the form of two Hilbert space mapping with u_{jk}. Let there is a scalar $f(x)$ and there are M observation points $f^{(l)} = f(x^{(l)})$, $l = 1 \dots M$. Convert this scalar mapping to a vector one (the $\xi^{(l)}$ and $\zeta^{(l)}$ are deterministic random functions on l that take the value ± 1)

$$x^{(l)}_k = \xi^{(l)} \left(x^{(l)}\right)^k \qquad k = 0 \dots n-1 \tag{75a}$$

$$f^{(l)}_j = \zeta^{(l)} \left(f^{(l)}\right)^j \qquad j = 0 \dots D-1 \tag{75b}$$

For numerical stability it is better, instead of monomials (powers of argument), to use Chebyshev or Legendre polynomials as $x^{(l)}_k = \xi^{(l)} T_k(ax^{(l)} + b)$ and $f^{(l)}_j = \zeta^{(l)} T_j(cf^{(l)} + d)$ with a, b, c, d chosen to bring argument into $[-1 : 1]$ interval. This greatly increases numerical stability of calculations; the result itself, however, is invariant relatively polynomial basis choice — the result will be the same with any polynomial basis. From obtained vector to vector mapping construct $\mathbf{x}^{(l)}$ and $\mathbf{f}^{(l)}$ localized states $\psi_{\mathbf{x}^{(l)}}(\mathbf{x}) \to \psi_{\mathbf{f}^{(l)}}(\mathbf{f})$ mapping (2) with the former one defined in (23) and the later one obtained from it with argument and index replacement (25). Put them to (28) and obtain (27). After solving the problem for u_{jk} an evaluation of interpolated $f(x)$ can be performed as following: from a given x construct the vector $\mathbf{x}$ (75a). Substitute it to (29) to obtain the probability of a given vector of outcome $\mathbf{f}$. In scalar case the value of outcome can be evaluated, for example, by constructing (75b) vector $\mathbf{f}$ from f and then finding the value of f providing the maximal value of probability (29). For scalar f this problem can be reduced to finding roots of a single variable polynomial.

This approach creates a number of issues. An attempt to fit a scalar function $f(x)$ to Hilbert spaces mapping using the moments like $\left\langle f^j x^k f^{j'} x^{k'} K(\mathbf{x}) K(\mathbf{f}) \right\rangle$ (26) create difficulties both in computations and in mapping back from Hilbert space to function value. A simple demonstration Let $f(x) = x^p$ with $p = 2$. This requires to calculate the x-moments of maximal

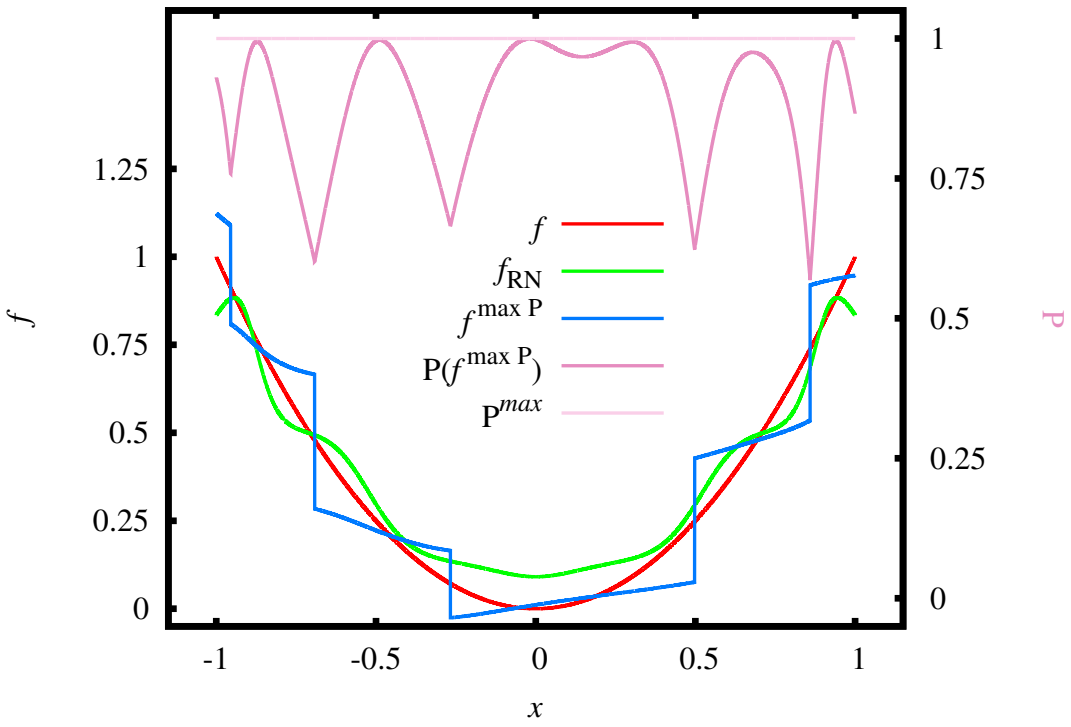

FIG. 1. Scalar function $f = x^2$ (red) interpolation with: green: Radon-Nikodym (73), blue: f corresponding to maximal probability (29), the dependence has discontinuities; some numerical instability presents even in this $D = n = 6$ case. Pink: probabilities corresponding to (29) and (76a).

degree $2(n-1)+2p(D-1)$ that for, say $n = D = 6$ gives the maximal polynomial degree of 30 what creates a numerical instability difficulty. Also the problem of converting back from Hilbert space to the value of f by checking all extremums of (29) to find the f providing the maximal probability is a "switching" function that creates a non-continuous solution when polynomial root changes. An example is presented in Fig. 1. For $f = x^2$ (red line) we calculated Radon-Nikodym approximation (73) (least squares is identical to the original f since the x^2 is in the basis thus it is not presented in the plot), and the f corresponding to the maximum of probability (blue line) and corresponding to it probability (pink). In the plot one can see staring numerical instability (plot asymmetry) and discontinuity in f corresponding to the changing of the selected root. As expected for the x of $P(f^{\max P}(x)) = 1$ the value of $f^{\max P}$ is equal to exact $f(x)$.

Alternatively[19] we can consider (29) without (75b) requirement that all components of

$\mathbf{f}$ are obtained from a single scalar f.

$$f_j^{\max P} = \sum_{j'=0}^{D-1} G^{\mathbf{f}}_{jj'} a_{j'} \tag{76a}$$

$$P(\mathbf{f}^{\max P})\Big|_{\mathbf{x}} = \sum_{j,j'=0}^{D-1} a_j G^{\mathbf{f}}_{jj'} a_{j'} \tag{76b}$$

The maximal value of probability (29) at (76a) is an important characteristic of obtained solution. If problem dimensions are balanced – the value is equal to 1 for all x (pink line P^{max} in Fig. 1), it can be lower in $D < n$ case and this requires separate consideration. In [19] we evaluated $f(x)$ from first two elements of vector $\mathbf{f}$: for a polynomial basis $P_j(f)$ (in (75b) $P_j(f) = f^j$) we can obtain the value of f from the ratio of $P_0 = f_0^{\max P}$ and $P_1(f) = f_1^{\max P}$. Back in [19] we thought that observed singularities are caused by a poor solution to the optimization problem. Whereas the algorithm developed in this paper provides a very good solution to the optimization problem of Hilbert spaces mapping, the method of obtaining the value of scalar f from basis functions ratio does not work well.

Currently we do not have a good solution to the problem of converting two Hilbert spaces mapping u_{jk} into a scalar function $f(x)$. This is a subject of future research.

VII. CONCLUSION

A novel QCQP problem of an optimal mapping between Hilbert spaces IN of $|\psi\rangle$ and OUT of $|\phi\rangle$ based on a set of wavefunction measurement (within a phase) observations $\psi_l \to \phi_l$, $l = 1 \dots M$, is formulated as an optimization problem maximizing the total fidelity $\sum_{l=1}^{M} \omega^{(l)} \left|\langle \phi_l \left| \mathcal{U} \right| \psi_l \rangle\right|^2$ subject to probability preservation constraints on $\mathcal{U}$. It is a linear operator transforming states between IN and OUT Hilbert spaces as (37). The operator $\mathcal{U}$, a rectangular matrix u_{jk} of $\dim(OUT) \times \dim(IN)$ dimensions with $D = \dim(OUT) \leq n = \dim(IN)$, can be viewed as a quantum channel. Time evolution is a special case of this problem. An optimization problem of a quadratic function on $\mathcal{U}$ with quadratic constraints is solved with a numerical method outlined in Section III. The method is an iteration method with generalized eigenproblem as it's building block, this is different from commonly used optimization methods that deploy Newtonian iteration or gradient iteration as a building block; this allows us to find the global maximum for almost any input. In addition, instead of usual iteration internal state in the form of a pair: approximation, Lagrange multipliers:

(u_{jk}, λ_{ij}), the algorithm uses iteration internal state in the form of a triple: approximation, Lagrange multipliers, homogeneous linear constraints $(u_{jk}, \lambda_{ij}, C_{d;jk})$; it is the very linear constraints that provide a fast convergence of the algorithm.

This numerical method can be applied to a number of classic and quantum problems, especially to quantum mechanics inverse problem: recover a Hamiltonian form a sequence of wavefunction observations; an operator $\mathcal{U}$ is obtained from observations then apply (16). The equation is quadratic on $\mathcal{U}$ what makes the results the same if measured wavefunctions are multiplied by random phase factors.

A demonstration of unitary dynamics $\mathbf{X}^{(l+1)} = \mathcal{U}\mathbf{X}^{(l)}$ system identification: determine $\mathcal{U}$ from measured $\widetilde{\mathbf{X}}^{(l)}$ that are actual $\mathbf{X}^{(l)}$ multiplied by random phase factors, is presented for a number of orthogonal operators $\mathcal{U}$ of the dimensions $n = D = \{3, 5, 7, 17, 40\}$. An exact recovery of $\mathcal{U}$ was observed. The technique was also applied to the problems of polynomial bases mapping and scalar function interpolation. For polynomial bases mapping an exact solution was obtained. The problem of scalar function interpolation is difficult to solve with the developed technique since there is no good equivalence between the problem of Hilbert spaces mapping and scalar function mapping.

Considered problem of two Hilbert spaces optimal mapping is reduced to a new algebraic "eigenoperator" problem (38), currently we have only a numerical solution to it. An important generalization of Unitary Learning was made in this paper by considering Hilbert spaces of different dimensions $D \leq n$. A question arise about further generalization. An important topic of future research can be Kraus' theorem that determines the most general form of Hilbert spaces mapping[29]:

$$A^{OUT} = \sum_s B_s A^{IN} B_s^\dagger \tag{77}$$

with Kraus operators B_s satisfying the constraints that unit A^{IN} is converted to unit A^{OUT}

$$\sum_s B_s B_s^\dagger = \mathbb{1} \tag{78}$$

We can further generalize Kraus operators B_s by considering them to be rectangular $D \times n$ matrices with a combined constraints sum similar to (8) for a single operator. This is a generalization of (37) transform; it is applicable to systems with quantum decoherence. In the Appendix I of [20] corresponding optimization problem is formulated, but it's numerical solution is a subject of future research.

ACKNOWLEDGMENTS

This research was supported by Autretech group[30], www.атретек.рф. We thank our colleagues from Autretech R&D department who provided insight and expertise that greatly assisted the research. Our grateful thanks are also extended to Mr. Gennady Belov for his methodological support in doing the data analysis.

Appendix A: On Lagrange Multipliers Calculation With Selected States Variation

In this work we do not use iterations for Lagrange multipliers. Instead, on every iteration, for current solution approximation u_{jk} we calculate Lagrange multipliers that correspond to an extremum in this state. This makes the algorithm more stable and allows a procedure of solution adjustment (45) from partial (41) to full set (40) of constraints to be directly applied. The variation of Lagrangian $\mathcal{L}$ (48) must be zero in the state u_{jk}

$$0 = \frac{1}{2}\frac{\delta\mathcal{L}}{\delta u_{iq}} = \sum_{j'=0}^{D-1}\sum_{k'=0}^{n-1} S_{iq;j'k'}u_{j'k'} - \sum_{j'=0}^{D-1}\lambda_{ij'}u_{j'q} \tag{A1}$$

These are Dn equations. Lagrange multipliers λ_{ij} is a Hermitian $D\times D$ matrix of $D(D+1)/2$ independent elements. Thus for a general u_{jk} all Dn equations (A1) cannot be simultaneously satisfied. They are satisfied in (64) for eigenstate $u^{[s]}_{jk}$ of (56), but after (45) adjustment this is no longer the case. A trivial solution is to take L^2 norm of variation vector (A1) and minimize the sum of squares (51) to obtain (52) as linear system solution. The problem which arise is that there are too many equations Dn, the variation is a vector of Dn components, and minimizing L^2 norm of it with just $D(D+1)/2$ parameters makes the convergence poor. The idea is to calculate the sum of squares only in specific states. Consider the problem

$$\sum_{s=0}^{N_v-1}\left|\sum_{i=0}^{D-1}\sum_{q=0}^{n-1} v^{[s]}_{iq}\left[\sum_{j'=0}^{D-1}\sum_{k'=0}^{n-1} S_{iq;j'k'}u_{j'k'} - \sum_{j=0}^{D-1}\frac{\lambda_{ij}+\lambda_{ji}}{2}u_{jq}\right]\right|^2 \underset{\lambda_{ij}}{\longrightarrow} \min \tag{A2}$$

The variation (A1) is projected on $v^{[s]}_{iq}$, $s=0\ldots N_v-1$, states and the sum of projection squares is taken. If $v^{[s]}_{iq}$ is a full basis, e.g. all the $s=0\ldots Dn-1$ eigenvectors $u^{[s]}_{jk}$ of (56) then the (A2) is exactly (51). One may think about (A1) as Lagrangian (48) variation $\delta\mathcal{L}/\delta u_{iq}$. The sum of squares (51) is $\left\langle \frac{\delta\mathcal{L}}{\delta u} \,\middle|\, \frac{\delta\mathcal{L}}{\delta u}\right\rangle$, the sum of squares in $v^{[s]}_{iq}$-projected states (A2) is $\sum_{s=0}^{N_v-1}\left\langle \frac{\delta\mathcal{L}}{\delta u} \,\middle|\, v^{[s]}\right\rangle\left\langle v^{[s]} \,\middle|\, \frac{\delta\mathcal{L}}{\delta u}\right\rangle$, for any full basis $v^{[s]}_{iq}$ they are the same. In practice the

$v_{iq}^{[s]}$ may not be necessary full or orthogonal. If they are a subset of original eigenstates $u_{jk}^{[s]}$ (56) of iteration algorithm – they are orthogonal, if (45) adjustment is applied to them – they are not. One may consider cross states with $Dn \times Dn$ Gram matrix, similar to (23) as $\sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} G^{-1}_{jk;j'k'} v_{j'k'}$, but this is not usually necessary since the selection of vectors $v_{iq}^{[s]}$ is performed only for algorithm's better convergence.

Once the vectors $v_{iq}^{[s]}$ are selected – in the problem (A2) the $D \times D$ Hermitian matrix λ_{ij} should be expressed via λ_r vector of the dimension $D(D+1)/2$. A variation over λ_r gives a linear system of the dimension $D(D+1)/2$ that can be readily solved. The number of $v_{iq}^{[s]}$ vectors should be at least $D(D+1)/2$, otherwise the linear system will be degenerated. For matrix to vector conversion and linear system solution see `com/polytechnik/kgo/LagrangeMultipliersPartialSubspace.java:getLambdaForSubspace` that implements this functionality to solve (A2) minimization problem and obtain λ_{ij} from a subspace chosen as the states of high eigenvalues of problem (56). The success of this approach is moderate: The number of top-$\mu^{[s]}$ states to select is not precisely clear, linear system may become degenerated, etc. One can check these our attempts at `com/polytechnik/kgo/LagrangeMultipliersPartialSubspace.java:getLambdaForSubspace` and usage in `com/polytechnik/kgo/KGOIterationalLagrangeMultipliersPartialSubspace.java`. This makes us to conclude that instead of considering a subspace for constructing λ_{ij} we should consider a subspace for variation of u_{jk}.

1. Linear Constraints On Variation

Degeneracy of the problem and quadratic constraints require not only a good approximation for Lagrange multipliers, but also a restricted subspace for variation of u_{jk} in (55) problem. The difficulty comes from partial unitarity constraints (40), optimization (55) preserves only partial constraint (41). Consider a full orthogonal basis $v_{jk}^{[s]}$, $\delta_{ss'} = \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} v_{jk}^{[s]} v_{jk}^{[s']}$, $s = 0 \dots Dn-1$, and a variation vector δu_{jk} expanded over this basis

$$\delta u_{jk} = \sum_{s=0}^{Dn-1} a_s v_{jk}^{[s]} \tag{A3}$$

$$a_s = \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} v_{jk}^{[s]} \delta u_{jk} \tag{A4}$$

Were we work in a regular vector space the only available operation is scalar product

$$\langle v \,|\, u \rangle = \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} v_{jk} u_{jk} \tag{A5}$$

Now, when we study partially unitary operators (37), we have a tensor

$$G_{ij}^{v|u} = \sum_{k=0}^{n-1} v_{ik} u_{jk} \tag{A6}$$

The scalar product corresponds to $\langle v \,|\, u \rangle = \mathrm{Spur} G^{v|u}$, Gram matrix (42) is $G_{ij}^{u} = G_{ij}^{u|u}$. Consider a variation $G_{ij}^{u+\delta u|u+\delta u}$. To preserve partial unitarity constraints (40) on u_{jk} within linear terms (on δu) we must have all off-diagonal elements of $\mathrm{Herm} G_{ij}^{u|\delta u}$ to be zero $0 = G_{ij}^{u|\delta u} + G_{ji}^{u|\delta u}$, $i \neq j$ (homogeneous) and diagonal elements to be one (inhomogeneous). The partial constraint (41) does preserve matrix spur thus it is sufficient to have diagonal elements just to be equal, and this is a homogeneous constraint $0 = G_{ii}^{u|\delta u} - G_{i-1\,i-1}^{u|\delta u}$, $i = 1 \ldots D-1$. Expanding δu in basis (A3) obtain the constraints

$$0 = \sum_{s=0}^{Dn-1} a_s \left(G_{ij}^{u|v^{[s]}} + G_{ji}^{u|v^{[s]}} \right) \qquad j < i,\, i = 0 \ldots D-1 \tag{A7a}$$

$$0 = \sum_{s=0}^{Dn-1} a_s \left(G_{ii}^{u|v^{[s]}} - G_{i-1\,i-1}^{u|v^{[s]}} \right) \qquad i = 1 \ldots D-1 \tag{A7b}$$

There are $(D-1)(D+2)/2$ total linear homogeneous constraints on expansion coefficients a_s. The a_s now became not all Dn independent, there are just $Dn - (D-1)(D+2)/2$ independent of them. The constraints (A7) are homogeneous

$$N_d = (D-1)(D+2)/2 \tag{A8a}$$

$$d_{offd} : 0 \ldots D(D-1)/2 - 1 \tag{A8b}$$

$$d_{diag} : D(D-1)/2 \ldots (D-1)(D+2)/2 - 1 \tag{A8c}$$

$$s : 0 \ldots Dn - 1 \tag{A8d}$$

$$C_{d;s} = \begin{cases} G_{ij}^{u|v^{[s]}} + G_{ji}^{u|v^{[s]}} & \text{if } d \in d_{offd} \text{ for } j < i,\, i = 0 \ldots D-1 \\ G_{ii}^{u|v^{[s]}} - G_{i-1\,i-1}^{u|v^{[s]}} & \text{if } d \in d_{diag} \text{ for } i = 1 \ldots D-1 \end{cases} \tag{A8e}$$

and can be put directly to (57) after basis transformation

$$C_{d;jk} = \sum_{s=0}^{Dn-1} C_{d;s} v_{jk}^{[s]} \tag{A9}$$

There are $D(D-1)/2$ constraints for zero off-diagonal elements (A8b) and $D-1$ constraints (one less the dimension) of diagonal elements equal to each other (A8c), there are total $(D-1)(D+2)/2$ homogeneous constraints. The constraints $C_{d;jk}$, the same as Lagrange multipliers λ_{ij}, are calculated solely from current iteration u_{jk}; see `com/polytechnik/kgo/LinearConstraints.java:getOrthogonalOffdiag0DiagEq` for an implementation.

The result of the consideration above is: if, instead of a full basis $v_{jk}^{[s]}$ of the dimension Dn, we take the basis V_p (60) that has $\mathrm{rank}(C_{d;jk}) = (D-1)(D+2)/2$ fewer elements — the variation of (62) will preserve partial unitarity of u_{jk} within first order. This drastically changes algorithm convergence. It starts perfectly converging to a true solution if both sets of the constraints (A7a) and (A7b) are satisfied; a single set does not provide convergence. Iteration algorithm finding a partially unitary operator optimally converting an operator from *IN* Hilbert space to *OUT* Hilbert space (37) is the main result of this paper. The result was achieved by considering on each iteration not a pair: approximation, Lagrange multipliers: (u_{jk}, λ_{ij}), but a triple: approximation, Lagrange multipliers, homogeneous linear constraints: $(u_{jk}, \lambda_{ij}, C_{d;jk})$. This is the cost of working with a quadratically constrained degenerate problem having multiple local extremums.

A problem of quadratic form optimization $\sum_{i,j=0}^{D-1} u_i M_{ij} u_j \underset{u}{\longrightarrow} \max$ with a single quadratic constraint $1 = \sum_{i,j=0}^{D-1} u_i Q_{ij} u_j$ with a positively definite matrix Q_{ij} can be reduced to an eigenvalue problem. In the Appendix F of [31] and later in [32] a quadratic form optimization problem with *two* quadratic form constraints was considered. An extra constraint was in the from $0 = \sum_{i,j=0}^{D-1} u_i C_{ij} u_j$. This is a much simpler problem than the one considered in the current paper. For this problem an algorithm with vanilla Lagrange multipliers iterations converges without extra linear constraints added, see `com/polytechnik/utils/IstatesConditionalV2.java` for an implementation. However, even in this simple case, a number of iteration starting values should be tried to find the global maximum. Numerical experiments show that adding on every iteration a single linear constraint on u_j: $0 = \sum_{i,j=0}^{D-1} u_i C_{ij} u_j^{(cur)}$, where $u_j^{(cur)}$ is the value of u_j used to calculate current iteration Lagrange multipliers, greatly increases the chances to find the global maximum. A reference implementation `com/polytechnik/utils/IstatesConditionalSubspaceLinearConstraints.java` on every iteration solves an eigenproblem of the dimension $D-1$ (a single constraint reduces the dimension by 1) and always selects the maximal eigenvalue, this is similar to the heuristic used in the algorithm above. The result is almost always better than that of `IstatesConditionalV2.j`

`ava` which solves an eigenproblem of the dimension D and tries a large number of vectors to select the next iteration from. This shows an advantage of considering iteration state as a triple (solution, Lagrange multipliers, linear constraints) even in a simple case of a single additional quadratic constraint.

Appendix B: Software description

- Install java 22 or later.
- Download the latest version of the source code `code_polynomials_quadratures.zip` from [28] or from alternative location.
- Decompress and recompile the program. Run a simple test recovering orthogonal matrices of the dimensions $3, 5, 7, 17, 40$.

  ```
  unzip code_polynomials_quadratures.zip
  javac -g com/polytechnik/*/*java
  java com/polytechnik/algorithms/PrintOrthogonalSeq\$TestAuto >/tmp/diag 2>&1
  ```

  The diagnostics is saved to the file `/tmp/diag`
- Check the maximal absolute difference between elements of original and recovered orthogonal matrices, do `grep DIFF /tmp/diag`

  ```
  GRAM DIFF for dim=3 is 2.3314683517128287E-15
  UNIT DIFF for dim=3 is 4.440892098500626E-16
  GRAM DIFF for dim=5 is 6.439293542825908E-15
  UNIT DIFF for dim=5 is 7.105427357601002E-15
  GRAM DIFF for dim=7 is 5.064698660461886E-14
  UNIT DIFF for dim=7 is 1.6431300764452317E-14
  GRAM DIFF for dim=17 is 4.6851411639181606E-14
  UNIT DIFF for dim=17 is 3.2807090377673376E-14
  GRAM DIFF for dim=40 is 4.5630166312093934E-14
  UNIT DIFF for dim=40 is 3.907985046680551E-14
  ```

  Since the unitarity of test data is exact — both quantum channels: invariant Gram matrix of Section II C and invariant unit matrix of Section II D recover the operator u_{jk} exactly.

[1] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review **65**, 386 (1958).

[2] V. Vapnik and A. Y. Chervonenkis, The method of ordered risk minimization, I, Avtomatika i Telemekhanika **8**, 21 (1974).

[3] P. Hájek and T. Havránek, On generation of inductive hypotheses, International Journal of Man-Machine Studies **9**, 415 (1977).

[4] V. Vapnik, *The nature of statistical learning theory* (Springer science & business media, 2013).

[5] I. H. Witten and E. Frank, Data mining: practical machine learning tools and techniques with Java implementations, Acm Sigmod Record **31**, 76 (2002).

[6] L. A. Zadeh, Fuzzy sets, Information and control **8**, 338 (1965).

[7] P. Hájek, Fuzzy logic and arithmetical hierarchy, Fuzzy sets and Systems **73**, 359 (1995).

[8] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence **35**, 1798 (2013).

[9] A. Bisio, G. Chiribella, G. M. D'Ariano, S. Facchini, and P. Perinotti, Optimal quantum learning of a unitary transformation, Physical Review A **81**, 032324 (2010).

[10] M. Arjovsky, A. Shah, and Y. Bengio, Unitary evolution recurrent neural networks, in *International conference on machine learning* (PMLR, 2016) pp. 1120–1128.

[11] S. Hyland and G. Rätsch, Learning unitary operators with help from u(n), in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31 (2017).

[12] M. Razavy, *An introduction to inverse problems in physics* (World Scientific, 2020).

[13] J. R. Johansson, P. D. Nation, and F. Nori, QuTiP: An open-source Python framework for the dynamics of open quantum systems, Computer Physics Communications **183**, 1760 (2012).

[14] S. B. Ramezani, A. Sommers, H. K. Manchukonda, S. Rahimi, and A. Amirlatifi, Machine learning algorithms in quantum computing: A survey, in *2020 International joint conference on neural networks (IJCNN)* (IEEE, 2020) pp. 1–8.

[15] V. G. Malyshkin and M. G. Belov, Market Directional Information Derived From (Time, Execution Price, Shares Traded) Sequence of Transactions. On The Impact From The Future, arXiv preprint arXiv:2210.04223 10.48550/arXiv.2210.04223 (2022).

[16] B. T. Kiani, S. Lloyd, and R. Maity, Learning unitaries by gradient descent, arXiv preprint

arXiv:2001.11897 10.48550/arXiv.2001.11897 (2020).

[17] B. Kiani, R. Balestriero, Y. LeCun, and S. Lloyd, projUNN: efficient method for training deep networks with unitary matrices, Advances in Neural Information Processing Systems **35**, 14448 (2022).

[18] S. Lloyd and R. Maity, Efficient implementation of unitary transformations, arXiv preprint arXiv:1901.03431 10.48550/arXiv.1901.03431 (2019).

[19] V. G. Malyshkin, On Machine Learning Knowledge Representation In The Form Of Partially Unitary Operator. Knowledge Generalizing Operator, arXiv preprint arXiv:2212.14810 10.48550/arXiv.2212.14810 (2022).

[20] V. G. Malyshkin, On The Radon-Nikodym Spectral Approach With Optimal Clustering, arXiv preprint arXiv:1906.00460 10.48550/arXiv.1906.00460 (2019).

[21] T. A. Loring, Computing a logarithm of a unitary matrix with general spectrum, Numerical Linear Algebra with Applications **21**, 744 (2014).

[22] I. Najfeld and T. F. Havel, Derivatives of the matrix exponential and their computation, Advances in applied mathematics **16**, 321 (1995).

[23] A. Raza, Differentiating exponentials of Hamiltonians (2020), `https://araza6.github.io/posts/hamiltonian-differentiation/`.

[24] A. Raza, Learning unitary matrices (2020), `https://araza6.github.io/posts/unitary-learning/`.

[25] G. H. Golub, Some modified matrix eigenvalue problems, Siam Review **15**, 318 (1973).

[26] G. Frison, J. Frey, F. Messerer, A. Zanelli, and M. Diehl, Introducing the quadratically-constrained quadratic programming framework in HPIPM, in *2022 European Control Conference (ECC)* (IEEE, 2022) pp. 447–453.

[27] V. G. Malyshkin and R. Bakhramov, Mathematical Foundations of Realtime Equity Trading. Liquidity Deficit and Market Dynamics. Automated Trading Machines, arXiv preprint arXiv:1510.05510 10.48550/arXiv.1510.05510 (2015).

[28] V. G. Malyshkin, The code for polynomials calculation (2014), `http://www.ioffe.ru/LNEPS/malyshkin/code.html` and an alternative location.

[29] K. Kraus, *States, Effects, and Operations: Fundamental Notions of Quantum Theory*, Lecture Notes in Physics, Vol. 190 (Springer-Verlag, 1983) Lectures in Mathematical Physics at the University of Texas at Austin.

[30] Autretech LLC, www.атретек.рф, is a resident of the Skolkovo Technopark. In May 2019, the company became the winner of the Skolkovo Cybersecurity Challenge competition.

[31] V. G. Malyshkin, Market Dynamics: On Directional Information Derived From (Time, Execution Price, Shares Traded) Transaction Sequences, arXiv preprint arXiv:1903.11530 10.48550/arXiv.1903.11530 (2019).

[32] L. Boudjemila, V. V. Davydov, and V. G. Malyshkin, On Quadratic Form Optimization Problem With Multiple Constraints of the Quadratic Form Type, in *The 5th International Conference on Future Networks & Distributed Systems* (2021) pp. 568–571.