# QuanEstimation: An open-source toolkit for quantum parameter estimation

Mao Zhang,[1] Huai-Ming Yu,[1] Haidong Yuan,[2] Xiaoguang Wang,[3] Rafał Demkowicz-Dobrzański,[4] and Jing Liu [1,*]

[1]*National Precise Gravity Measurement Facility, MOE Key Laboratory of Fundamental Physical Quantities Measurement, School of Physics, Huazhong University of Science and Technology, Wuhan 430074, China*

[2]*Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, China*

[3]*Zhejiang Institute of Modern Physics, Department of Physics, Zhejiang University, Hangzhou 310027, China*

[4]*Faculty of Physics, University of Warsaw, 02-093 Warszawa, Poland*

Quantum parameter estimation promises a high-precision measurement in theory; however, how to design the optimal scheme in a specific scenario, especially under a practical condition, is still a serious problem that needs to be solved case by case due to the existence of multiple mathematical bounds and optimization methods. Depending on the scenario considered, different bounds may be more or less suitable, both in terms of computational complexity and the tightness of the bound itself. At the same time, the metrological schemes provided by different optimization methods need to be tested against realization complexity, robustness, etc. Hence, a comprehensive toolkit containing various bounds and optimization methods is essential for the scheme design in quantum metrology. To fill this vacancy, here we present a Python-Julia-based open-source toolkit for quantum parameter estimation, which includes many well-used mathematical bounds and optimization methods. Utilizing this toolkit, all procedures in the scheme design, such as the optimizations of the probe state, control and measurement, can be readily and efficiently performed.

## I. INTRODUCTION

Quantum metrology is an emerging cross-disciplinary field between precision measurement and quantum technology, and has now become one of the most promising fields in quantum technology due to the general belief that it could step into the industrial-grade applications in a short time [1–5]. Meanwhile, its development not only benefits the applied technologies like the magnetometry, thermometry, and gravimetry, but also the studies in fundamental physics such as the detection of gravitational waves [6] and the search of dark matters [7,8]. As the theoretical support of quantum metrology, quantum parameter estimation started from 1960s [9], and has become an indispensable component of quantum metrology nowadays [10–18].

One of the key challenges in quantum parameter estimation is to design optimal schemes with quantum apparatuses and quantum resources, leading to enhanced precision when compared with their classical counterparts. A typical scheme in quantum parameter estimation usually contains four steps: (1) preparation; (2) parametrization; (3) measurement; and (4) classical estimation. The first step is the preparation of the probe state. The parameters to be estimated are involved in the second step, which is also known as sensing in the field of quantum sensing. With the parameterized state given in the second step, the third step is to perform the quantum measurement, which results in a set of probability distributions. Estimating the unknown parameters from the obtained probability distributions is finished in the last step. The design of an optimal scheme usually requires the optimizations of some or all of the steps above.

In quantum parameter estimation, there exist various mathematical bounds to depict the theoretical precision limit. Depending on the type of the bound considered, it will be more or less informative depending on the type of estimation scenario considered, be it: single-shot versus many-repetition scenario, single versus multiple-parameter scenario, etc. Moreover, by choosing different objective functions when optimizing quantum estimation schemes, one may arrive at solutions with contrastingly different robustness properties, complexity of practical implementation and so on. Hence, the design of optimal schemes has to be performed case by case most of the time. This is the reason why a general quantum parameter estimation toolkit is needed. In the meantime, thanks to the fast development of quantum metrology and its promising future, many scientists working on specific physical systems, such as the nitrogen-vacancy centers, quantum circuits, trapped ions, and atoms, are also eager to use the cutting-edge technologies in quantum parameter estimation for the design of metrological schemes on their platforms. The existence of a comprehensive toolkit will definitely reduce the technical difficulty for them to fulfill this mission and greatly improve the efficiency of research. Therefore, developing such a toolkit is the major motivation of this paper.

*liujingphys@hust.edu.cn

Currently, there exist many useful toolkits based on various coding platforms in quantum information. A famous one is the QuTiP developed by Johansson, Nation, and Nori [19,20] in 2012, which can execute many basic calculations in quantum information. In the field of quantum control, Machnes *et al.* [21] developed DYNAMO and Hogben *et al.* developed Spinach [22] based on Matlab. Goerz *et al.* developed Krotov [23], which owns three versions based on Fortran, Python, and Julia, respectively. Günther *et al.* developed Quandary [24] based on C++. Moreover, there exist other packages like Kwant [25] for quantum transport and ProjectQ [26] for quantum computing. In quantum metrology, Chabuda and Demkowicz-Dobrzański developed TNQMetro [27], a tensor-network based Python package to perform efficient quantum metrology computations.

Hereby we present a numerical toolkit, QuanEstimation, based on both Python and Julia for the quantum parameter estimation and provide some examples to demonstrate its usage and performance. QuanEstimation is designed to fill the lack of a general toolkit in quantum parameter estimation, not a general one in quantum information. Hence, it only focuses on the missions in quantum parameter estimation, and the main features are significantly different from the existing toolkits in quantum information. Specifically, it contains several widely-used metrological tools, such as the asymptotic Fisher information based quantities as well as their Bayesian counterparts (including direct Bayesian cost minimization, Bayesian versions of the classical and quantum Cramér-Rao bounds as well as the quantum Ziv-Zakai bound). For the sake of scheme design, QuanEstimation can execute the optimizations of the probe state, control, and measurement, as well as the simultaneous optimizations among them with both gradient-based and gradient-free methods. Due to the fact that most of the time adaptive measurement schemes are the best practical way to realize the asymptotic advantage indicated by the quantum Fisher information, QuanEstimation can also execute online adaptive measurement schemes, such as the adaptive phase estimation, and provide the real-time values of the tunable parameters that can be directly used in an experiment.

## II. OVERVIEW

QuanEstimation is a scientific computing package focusing on the calculations and optimizations in quantum parameter estimation. It is based on both Python and Julia. The interface is written in Python due to the fact that nowadays Python is one of the most popular platforms for scientific computing. However, QuanEstimation contains many optimization processes, which need to execute massive numbers of elementary processes such as the loops. These elementary processes could be very time-consuming in Python, and thus strongly affect the efficiency of the optimizations. This is why Julia is involved in this package. Julia has many wonderful features, such as optional typing and multiple dispatch, and these features let the loop and other calculation processes cost way less time than those in Python. Hence, the optimizations in QuanEstimation are all performed in Julia. Nevertheless, currently the community of Julia is not comparable to that of Python, and the hybrid structure of this package would allow

the people who are not familiar with Julia use the package without any obstacle. In the meantime, QuanEstimation has a full Julia version for the users experienced in Julia.

The package structure of QuanEstimation is illustrated in Fig. 1. The blue boxes and the light-blue boxes represent the folders and the files. The orange boxes and the gray boxes represent the classes and the functions/methods. The boxes circled by the dotted lines represents the wrapped Julia methods, which are solved in Julia, namely, this part of calculation are sent to Julia to execute.

The functions for the calculation of the parametrization process and dynamics are in the folder named "Parameterization". In this folder, the file "GeneralDynamics.py" contains the functions to solve the Lindblad-type master equation. Currently, the master equation can be solved directly, i.e., solving the corresponding ordinary differential equation, or via the matrix exponential. To improve the efficiency, the calculation of the dynamics via the matrix exponential are executed in Julia and when the calculation is finished, the data is sent back to Python for further use. The file "NonDynamics.py" contains the nondynamical methods for the parametrization, which currently includes the description via Kraus operators. Details and the usage of these functions will be thoroughly introduced in Sec. III.

The functions for the calculation of the metrological tools and bounds are distributed in two folders named "AsymptoticBound" and "BayesianBound". In the folder "AsymptoticBound", the file "CramerRao.py" contains the functions to calculate the quantities related to the quantum Cramér-Rao bounds, and the file "AnalogCramerRao.py" contains those to calculate the Holevo-type quantum Cramér-Rao bound and Nagaoka-Hayashi bound. In the folder "BayesianBound", the file "BayesCramerRao.py" contains the functions to calculate several versions of the Bayesian classical and quantum Cramér-Rao bounds and "ZivZakai.py" contains the function to calculate the quantum Ziv-Zakai bound. The file "BayesEstimation.py" contains the functions to execute the Bayesian estimation and the maximum likelihood estimation. The aforementioned metrological tools and the corresponding rules to call them will be given in Sec. IV.

The functions for the calculation of metrological resources are placed in the folder named "Resource". In this folder, the file "Resource.py" currently contains two types of resources, the spin squeezing and the target time to reach a given value of an objective function, which will be thoroughly introduced in Sec. V. The resources that can be readily calculated via QuTiP [19,20] are not included at this moment.

The scripts for the control optimization, state optimization, measurement optimization, and comprehensive optimization are in the folders named "ControlOpt", "StateOpt", "MeasurementOpt", and "ComprehensiveOpt", respectively. The structures of these folders are basically the same, and here we only take the folder of "ControlOpt" as an demonstration to explain the basic structure. In this folder, the file "ControlStruct.py" contains a function named *ControlOpt()* and a class named *ControlSystem()*. The function *ControlOpt()* is used to receive the initialized parameters given by the user, and then delivers them to one of the classes in the files "GRAPE_Copt.py", "PSO_Copt.py", "DE_Copt.py", and "DDPG_Copt.py" according to the user's choice of the
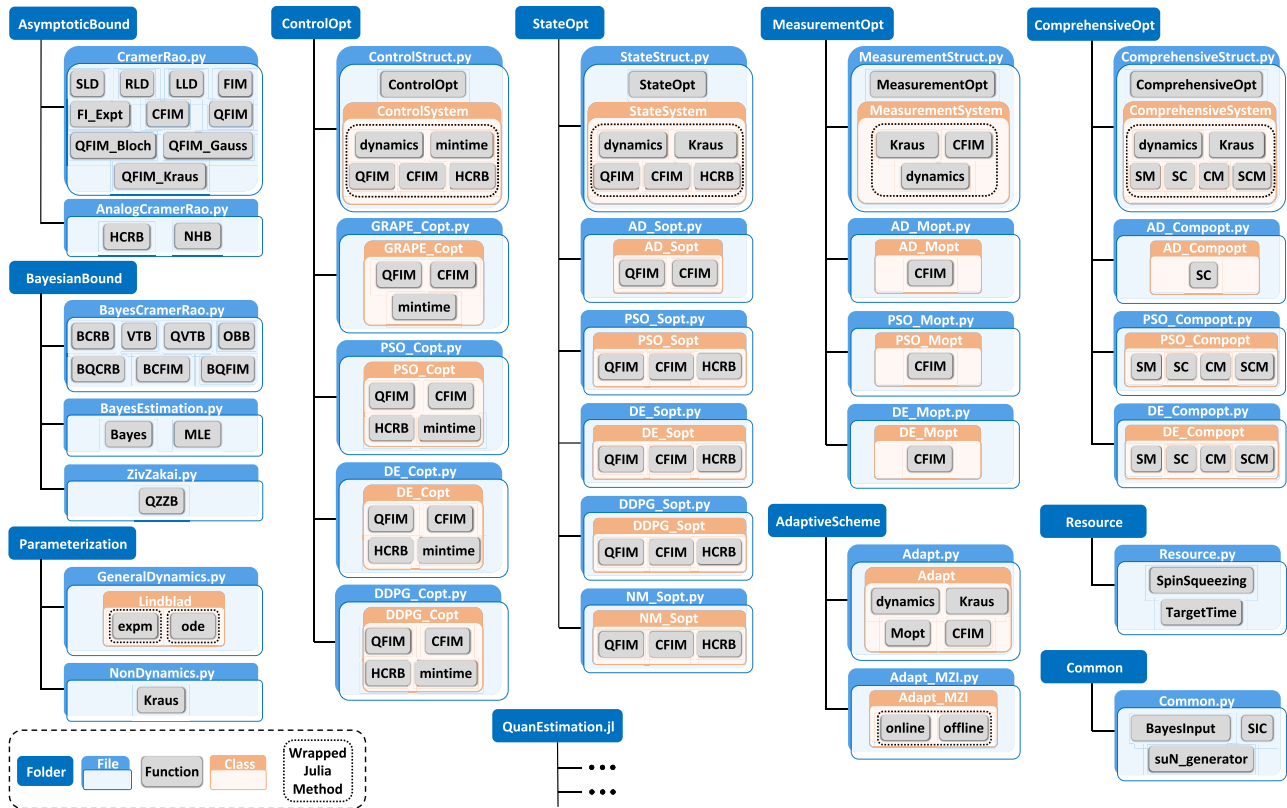
FIG. 1. Schematic of the package structure of QuanEstimation. The blue boxes and the light-blue boxes represent the folders and the files. The orange boxes and the gray boxes represent the classes and the functions/methods. The boxes circled by the dotted lines represents the wrapped Julia methods, which are solved in Julia scripts.

algorithm. These classes inherit the attributes in *ControlSystem()*. Then based on the choice of the objective function, the related parts in *ControlSystem()* is called in these classes to further run the scripts in Julia. *ControlSystem()* contains all the common parts that different algorithms would use and the interface with the scripts in Julia. This design is to avoid the repetition code in the algorithm files and let the extension neat and simple when more algorithms need to be included in the future. The usage of QuanEstimation for control optimization, state optimization, measurement optimization, and comprehensive optimization, as well as the corresponding illustrations will be thoroughly discussed in Secs. VI–IX, respectively.

The scripts for the adaptive measurement are in the folder named "AdaptiveScheme". In this folder, the file "Adapt.py" contains the class to execute the adaptive measurement scheme, and "Adapt_MZI.py" contains the class to generate online and off-line adaptive schemes in the Mach-Zehnder interferometer. The details of the adaptive scheme and how to perform it with QuanEstimation will be given in Sec. X.

The folder "Common" contains some common functions that are regularly called in QuanEstimation. Currently it contains three functions. *SIC()* is used to generate a set of rank-one symmetric informationally complete positive operator-valued measure. *suN_generator()* is used to generate a set of su($N$) generators. *BayesInput()* is used to generate a legitimate form of Hamiltonian (or a set of Kraus operators)

and its derivative, which can be used as the input in some functions in "BayesEstimation.py" and "Adapt.py".

All the Julia scripts are wrapped up as an independent Julia package named "QuanEstimation.jl", which has already been added in the official Julia registry and can be directly called in Julia via *using QuanEstimation*. One design principle of QuanEstimation for the optimizations is that once the calculation goes into the parts in Julia, it will stay in Julia until all the calculations are finished and data generated. Hence, "QuanEstimation.jl" also contains the scripts to calculate the metrological tools and resources for the sake of internal calling in Julia. To keep a high extendability, the optimizations are divided into four elements in Julia, including the scenario of optimization, the algorithm, the parametrization process and the objective function, which are distributed in the files "OptScenario.jl", "Algorithm.jl", "Parameterization.jl", and "ObjectiveFunc.jl" in the folders "OptScenario", "Algorithm", "Parameterization", and "ObjectiveFunc", respectively. Once the information and parameter settings of all elements are input by the user, they are sent to the file "run.jl", which is further used to execute the program.

Similar to other packages, the usage of QuanEstimation requires the existence of some other packages in the environment. In python it requires the preinstallation of numpy, scipy, sympy, cvxpy, and more-itertools. In Julia it requires the preinstallation of LinearAlgebra, Zygote, Convex, SCS, ReinforcementLearning, SparseArrays, DelimitedFiles,

StatsBase, BoundaryValueDiffEq, Random, Trapz, Interpolations, Printf, IntervalSets, StableRNGs, Flux, Distributions, DifferentialEquations, and QuadGK. The calling of the package in Python can be done with the following line of code:

```
from quanestimation import *
```

All the scripts demonstrated in the following are based on this calling form.

## III. PARAMETRIZATION PROCESS

The parametrization process is a key step in the quantum parameter estimation, and in physical terms this process corresponds to a parameter dependent quantum dynamics. Hence, the ability to solve the dynamics is an indispensable element of numerical calculations in quantum parameter estimation. In QuanEstimation, we mainly focus on the dynamics governed by the quantum master equation

$$\partial_t \rho = \mathcal{L}\rho$$
$$= -i[H, \rho] + \sum_i \gamma_i \left( \Gamma_i \rho \Gamma_i^\dagger - \frac{1}{2}\{\rho, \Gamma_i^\dagger \Gamma_i\} \right), \quad (1)$$

where $\rho$ is the evolved density matrix, $H$ is the Hamiltonian of the system, and $\Gamma_i$ and $\gamma_i$ are the $i$th decay operator and decay rate, respectively. Here $\gamma_i$ could either be fixed or time dependent. The total Hamiltonian $H$ includes two terms, the free Hamiltonian $H_0(\mathbf{x})$, which is a function of the parameters $\mathbf{x}$, and control Hamiltonian $H_c$. In the quantum parameter estimation, most calculations require the dynamical information of $\rho$ and its derivatives with respect to $\mathbf{x}$, which is denoted by $\partial_\mathbf{x}\rho := (\partial_0\rho, \partial_1\rho, \dots)$ with $\partial_a$ short for $\partial_{x_a}$. Hence, in the package $\rho$ and $\partial_\mathbf{x}\rho$ can be solved simultaneously via the code:

```
dynamics = Lindblad(tspan,rho0,H0,dH,
                    decay=[],Hc=[],ctrl=[])
rho,drho = dynamics.expm()
```

Here the input *tspan* is an array representing the time length for the evolution and *rho0* is a matrix representing the initial (probe) state. *H0* is a matrix or a list of matrices representing the free Hamiltonian. It is a matrix when the free Hamiltonian is time independent and a list (the length equals to that of *tspan*) when it is time dependent. *dH* is a list containing the derivatives of $H_0(\mathbf{x})$ on $\mathbf{x}$, i.e., $[\partial_a H_0, \partial_b H_0, \dots]$. *decay* is a list including both decay operators and decay rates, and its input rule is *decay=[[Gamma1,gamma1],[Gamma2,gamma2],...]*, where *Gamma1* (*Gamma2*) and *gamma1* (*gamma2*) represent $\Gamma_1$ ($\Gamma_2$) and $\gamma_1$ ($\gamma_2$), respectively. *gamma1* (*gamma2*) could be either a float number (representing a fixed decay rate) or an array (representing a time-dependent decay rate), and when it is an array, its length should be the same with *tspan*. Currently all the length of the decay rates should be the same, i.e., all be float numbers or arrays with the same length. The default value is empty, which means the dynamics is unitary. *Hc* is a list of matrices representing the control Hamiltonians and when it is empty, the dynamics is only governed by the free Hamiltonian. *ctrl* (default value is empty) is a list of arrays containing the control amplitudes with respect the control Hamiltonians in *Hc*. The output *rho* is a list representing

density matrices in the dynamics. *drho* is also a list and its $i$th entry is a list containing all derivatives $\partial_\mathbf{x}\rho$ at $i$th time interval. Moreover, *dynamics.expm()* in this demonstrating code means the dynamics is solved by the matrix exponential, i.e., the density matrix at $j$th time interval is calculated via $\rho_j = e^{\Delta t_j \mathcal{L}}\rho_{j-1}$ with $\Delta t_j$ a small time interval and $\rho_{j-1}$ the density matrix at the previous time interval. $\partial_\mathbf{x}\rho_j$ is solved by the iterative equation

$$\partial_\mathbf{x}\rho_j = \Delta t_j(\partial_\mathbf{x}\mathcal{L})\rho_j + e^{\Delta t_j \mathcal{L}}(\partial_\mathbf{x}\rho_{j-1})$$
$$= -i\Delta t_j[\partial_\mathbf{x}H_0, \rho_j] + e^{\Delta t_j \mathcal{L}}(\partial_\mathbf{x}\rho_{j-1}). \quad (2)$$

Here the decay operators and decay rates are assumed to be independent of $\mathbf{x}$. In this method $\Delta t_j$ is automatically obtained by calculating the difference between the $j$th and $(j-1)$th entries in *tspan*. The numerical accuracy of the equation above is limited by the set of $\{\Delta t_j\}$, indicating that a smaller $\{\Delta t_j\}$ would always benefit the improvement of the accuracy in general. However, a smaller $\{\Delta t_j\}$ also means a larger number of calculation steps for a fixed evolution time, resulting in a greater time consumption. Hence, in practice a reasonable values of $\{\Delta t_j\}$ should be chosen to balance the accuracy and time consumption.

Alternatively, the dynamics can also be solved by directly solving the ordinary differential equation (ODE) in Eq. (1), which can be realized by replacing *dynamics.expm()* with *dynamics.ode()* in the demonstrating code. In this method, $\partial_\mathbf{x}\rho$ is solved by the equation

$$\partial_t(\partial_\mathbf{x}\rho) = -i[\partial_\mathbf{x}H, \rho] + \mathcal{L}(\partial_\mathbf{x}\rho). \quad (3)$$

The calculation of metrological bounds, which will be discussed in the next section, does not rely on the calling of above intrinsic dynamics in the package as they only require the input of $\rho$ and $\partial_\mathbf{x}\rho$ (and other essential parameters), not any dynamical information. Hence, the dynamics can also be solved by other packages like QuTiP [19,20].

In certain cases, the parametrization process can be described by some nondynamical methods, such as the Kraus operators. In this case, the parameterized density matrix can be expressed by

$$\rho(\mathbf{x}) = \sum_i K_i(\mathbf{x})\rho_0 K_i^\dagger(\mathbf{x}), \quad (4)$$

where $K_i(\mathbf{x})$ is a Kraus operator satisfying $\sum_i K_i^\dagger K_i = \mathbb{1}$ with $\mathbb{1}$ the identity operator, $\rho_0$ is the probe state, which is independent of the unknown parameters. In QuanEstimation, $\rho$ and $\partial_\mathbf{x}\rho$ obtained from Kraus operators can be solved via the code:

```
rho,drho = Kraus(rho0,K,dK)
```

Here *rho0* is a matrix representing the probe state, *K* is a list of matrices with each entry a Kraus operator, and *dK* is a list with $i$th entry also a list representing the derivatives $\partial_\mathbf{x}K_i$.

The aforementioned functions only calculate $\rho$ and $\partial_\mathbf{x}\rho$ at a fixed point of $\mathbf{x}$. However, in the Bayesian scenarios, the values of $\rho$ and $\partial_\mathbf{x}\rho$ with respect to a regime of $\mathbf{x}$ may be in need. In this case, if the users can provide the specific functions of

$H$ and $\partial_{\mathbf{x}}H$, or Kraus operators $\{K_i\}$ and derivatives $\{\partial_{\mathbf{x}}K_i\}$, the variables $H$, $dH$ (or $K$, $dK$) can be generated by the function

```
H0,dH = BayesInput(x,func,dfunc,
                   channel = "dynamics")
```

Here $x$ is a list of arrays representing the regime of $\mathbf{x}$. $H0$ is a list of matrices representing the free Hamiltonian with respect to the values in $x$, and it is multidimensional in the case that $x$ has more than one entry. $dH$ is a (multidimensional) list with each entry also a list representing $\partial_{\mathbf{x}}H$ with respect to the values in $x$. $func$ and $dfunc$ are the handles of the functions $func()$ and $dfunc()$, which are defined by the users representing $H(\mathbf{x})$ and $\partial_{\mathbf{x}}H(\mathbf{x})$. Notice that the output of $dfunc()$ should also be a list representing $[\partial_0 H, \partial_1 H, \dots]$. The output of $BayesInput()$ can be switched between $H$, $dH$ and $K$, $dK$ by setting $channel = $ "dynamics" or $channel = $ "Kraus". After calling $BayesInput()$, $\rho$ and $\partial_{\mathbf{x}}\rho$ can be further obtained via the calling of $Lindblad()$ and $Kraus()$.

## IV. QUANTUM METROLOGICAL TOOLS

In this section, we will briefly introduce the metrological tools that have been involved in QuanEstimation and demonstrate how to calculate them with our package. Both asymptotic and Bayesian tools are included, such as the quantum Cramér-Rao bounds, Holevo Cramér-Rao bound, Nagaoka-Hayashi bound, Bayesian estimation, and Bayesian type of Cramér-Rao bounds like Van Trees bound and Tsang-Wiseman-Caves bound.

### A. Quantum Cramér-Rao bounds

Quantum Cramér-Rao bounds [28,29] are the most renown metrological tools in quantum parameter estimation. Let $\rho = \rho(\mathbf{x})$ be a parameterized density matrix and $\{\Pi_y\}$ a set of positive operator-valued measure (POVM), then the covariance matrix $\mathrm{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) := \sum_y \mathrm{Tr}(\rho \Pi_y)(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^{\mathrm{T}}$ for the unknown parameters $\mathbf{x} = (x_0, x_1, \dots)^{\mathrm{T}}$ and the corresponding unbiased estimators $\hat{\mathbf{x}} = (\hat{x}_0, \hat{x}_1, \dots)^{\mathrm{T}}$ satisfies the following inequalities [28,29]

$$\mathrm{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \frac{1}{n}\mathcal{I}^{-1}(\{\Pi_y\}) \geqslant \frac{1}{n}\mathcal{F}^{-1}, \tag{5}$$

where $n$ is the repetition of the experiment, $\mathcal{I}$ is the classical Fisher information matrix (CFIM) and $\mathcal{F}$ is the quantum Fisher information matrix (QFIM). Note that the estimators $\hat{\mathbf{x}}$ are in fact functions of the measurement outcomes $y$, and formally should always be written as $\hat{\mathbf{x}}(y)$. Still, we drop this explicit dependence on $y$ for conciseness of formulas. A thorough derivation of this bound can be found in a recent review [13].

For a set of discrete probability distribution $\{p(y|\mathbf{x}) = \mathrm{Tr}(\rho \Pi_y)\}$, the CFIM is defined by

$$\mathcal{I}_{ab} = \sum_y \frac{1}{p(y|\mathbf{x})}[\partial_a p(y|\mathbf{x})][\partial_b p(y|\mathbf{x})]. \tag{6}$$

Here $\mathcal{I}_{ab}$ is short for $\mathcal{I}_{x_a, x_b}$, the $ab$th entry of the CFIM. For a continuous probability density, the equation above becomes

$\mathcal{I}_{ab} = \int \frac{1}{p(y|\mathbf{x})}[\partial_a p(y|\mathbf{x})][\partial_b p(y|\mathbf{x})]dy$. The diagonal entry $\mathcal{I}_{aa}$ is the classical Fisher information (CFI) for $x_a$.

The QFIM does not depend on the actual measurement performed, and one can encounter a few equivalent definitions of this quantity. The one the most often used reads

$$\mathcal{F}_{ab} = \frac{1}{2}\mathrm{Tr}(\rho\{L_a, L_b\}) \tag{7}$$

with $\mathcal{F}_{ab}$ being the $ab$th entry of $\mathcal{F}$ and $L_{a(b)}$ the symmetric logarithmic derivative (SLD) operator for $x_{a(b)}$. $\{\cdot, \cdot\}$ represents the anticommutator. The SLD operator is Hermitian and determined by the equation

$$\partial_a \rho = \frac{1}{2}(\rho L_a + L_a \rho). \tag{8}$$

The mathematical properties of the SLD operator and QFIM can be found in a recent review [13]. The diagonal entry of $\mathcal{F}_{aa}$ is the quantum Fisher information (QFI) for $x_a$. Utilizing the spectral decomposition $\rho = \sum_i \lambda_i |\lambda_i\rangle\langle\lambda_i|$, the SLD operator can be calculated via the equation

$$\langle\lambda_i|L_a|\lambda_j\rangle = \frac{2\langle\lambda_i|\partial_a\rho|\lambda_j\rangle}{\lambda_i + \lambda_j}, \tag{9}$$

for $\lambda_i$ or $\lambda_j$ not equal to zero. For $\lambda_i = \lambda_j = 0$, the corresponding matrix entry of $L_a$ can be set to zero.

In QuanEstimation, the SLD operator can be calculated via the function:

```
SLD(rho,drho,rep = "original",eps = 1e-8)
```

Here the input $rho$ is a matrix representing the parameterized density matrix, and $drho$ is a list of matrices representing the derivatives of the density matrix on $\mathbf{x}$, i.e., $[\partial_0\rho, \partial_1\rho, \dots]$. When $drho$ only contains one entry ($[\partial_0\rho]$), the output of $SLD()$ is a matrix ($L_0$), and it is a list ($[L_0, L_1, \dots]$) otherwise. The basis of the output SLD can be adjusted via the variable $rep$. The default choice $rep = $ "original" means the basis is the same with that of the input density matrix. The other choice is $rep = $ "eigen", which means the SLD is written in the eigenspace of the density matrix. Due to the fact that the entries of SLD in the kernel are arbitrary, in the package they are just set to be zeros for simplicity. The default machine epsilon is $eps = 1e{-}8$, which can be modified as required. Here the machine epsilon means that if a eigenvalue of the density matrix is less than the given number ($10^{-8}$ by default), it will be treated as zero in the calculation of SLD.

Apart from the SLD operator, the QFIM can also be defined via other types of logarithmic derivatives. Some well-used ones are the right and left logarithmic derivatives (RLD, LLD) [29,30]. The RLD and LLD are determined by $\partial_a\rho = \rho\mathcal{R}_a$ and $\partial_a\rho = \mathcal{R}_a^{\dagger}\rho$, respectively. Utilizing the spectral decomposition, the entries of RLD and LLD can be calculated as

$$\langle\lambda_i|\mathcal{R}_a|\lambda_j\rangle = \frac{1}{\lambda_i}\langle\lambda_i|\partial_a\rho|\lambda_j\rangle, \quad \lambda_i \neq 0, \tag{10}$$

$$\langle\lambda_i|\mathcal{R}_a^{\dagger}|\lambda_j\rangle = \frac{1}{\lambda_j}\langle\lambda_i|\partial_a\rho|\lambda_j\rangle, \quad \lambda_j \neq 0. \tag{11}$$

The corresponding QFIM is $\mathcal{F}_{ab} = \mathrm{Tr}(\rho\mathcal{R}_a\mathcal{R}_b^{\dagger})$. In QuanEstimation, the LLD and RLD can be calculated via the functions $RLD()$ and $LLD()$. The inputs are the same with $SLD()$. Notice that the RLD and LLD only exist when the support of $\rho$

contains the the support of $\partial_a \rho$. Hence, if this condition is not satisfied, the calculation will be terminated and a line of reminder will arise to remind that *RLD()* and *LLD()* do not exist in this case.

In QuanEstimation, the QFIM and QFI can be calculated via the function:

```
QFIM(rho, drho, LDtype = "SLD", exportLD = False,
     eps = 1e-8)
```

Here *LDtype = " "* is the type of logarithmic derivatives, including *"SLD"*, *"RLD"*, and *"LLD"*. Notice that the values of QFIM based on RLD and LLD are actually the same when the RLD and LLD exist. If *exportLD = True*, apart from the QFIM, the corresponding values of logarithmic derivatives in the original basis will also be exported.

In the case that the parametrization is described via the Kraus operators, the QFIM can be calculated via the function:

```
QFIM_Kraus(rho0, K, dK, LDtype = "SLD",
           exportLD = False, eps = 1e-8)
```

The input *rho0* is a matrix representing the density matrix of the initial state. *K* is a list of matrices with each entry a Kraus operator, and *dK* is a list with *i*th entry being also a list representing the derivatives $\partial_{\mathbf{x}} K_i$.

The CFIM and CFI for a fully classical scenario can be calculated by the function

```
FIM(p, dp, eps = 1e-8)
```

The input *p* is an array representing the probability distribution and *dp* is a list with the *i*th entry being itself also a list containing the derivatives of $p_i$ on $\mathbf{x}$, i.e., $[\partial_0 p_i, \partial_1 p_i, \dots]$. In the realistic experiments, the derivatives of the conditional probability are difficult to obtain, and therefore the CFI cannot be measured directly. However, it is possible that a known small drift $\delta \mathbf{x}$ can be experimentally invoked into the system and let the true value of $\mathbf{x}$ moves slightly. In such cases, the CFI can be calculated once the distributions $p(y|\mathbf{x})$ and $p(y|\mathbf{x} + \delta \mathbf{x})$ are acquired, which are usually obtained via the distribution fitting. In the case of single-parameter estimation, the CFI can be expressed by $\mathcal{I} = 8[1 - \sum_y \sqrt{p(y|x)p(y|x + \delta x)}]/\delta^2 x$ due to the relation between the CFI and the classical fidelity $\sum_y \sqrt{p(y|\mathbf{x})q(y|\mathbf{x})}$. In QuanEstimation, if the users have two sets of data (results of *y*) with respect to the value *x* and $x + \delta x$ in experiments, then the CFI can be calculated via the following function:

```
FI_Expt(y1, y2, dx, ftype = "norm")
```

Here *y1* and *y2* are two arrays containing the data of *y* in experiments with respect to the values *x* and $x + \delta x$, and *dx* represents the value of $\delta x$. Currently, four types of distributions are available for distribution fitting, including the normal (Gaussian) distribution (*ftype = "norm"*), gamma distribution (*ftype = "gamma"*), rayleigh distribution (*ftype = "rayleigh"*), and poisson distribution (*ftype = "poisson"*).

In a quantum scenario, the CFIM can be calculated by

```
CFIM(rho, drho, M = [], eps = 1e-8)
```

The variable *M* is a list containing a set of POVM. The default measurement is a set of rank-one symmetric informationally complete POVM (SIC-POVM) [31–33]. A set of rank-one SIC-POVM $\{\frac{1}{d}|\phi_j\rangle\langle\phi_j|\}_{j=1}^{d^2}$ satisfies $|\langle\phi_j|\phi_k\rangle|^2 = (d\delta_{jk} + 1)/(d + 1)$ for any *j* and *k* with $|\phi_j\rangle$ being a normalized quantum state and *d* the dimension of the Hilbert space. One way to construct a set of SIC-POVM is utilizing the Weyl-Heisenberg operators [33,34], which is defined by $D_{ab} = (-e^{i\pi/d})^{ab} A^a B^b$. The operators *A* and *B* satisfy $A|k\rangle = |k + 1\rangle$, $B|k\rangle = e^{i2\pi k/d}|k\rangle$ with $\{|k\rangle\}_{k=0}^{d-1}$ an orthonormal basis in the Hilbert space. There exists a normalized fiducial vector $|\psi\rangle$ in the Hilbert space such that $\{\frac{1}{d} D_{ab}|\psi\rangle\langle\psi|D_{ab}^\dagger\}_{a,b=1}^d$ is a set of SIC-POVM. In the package, $|\psi\rangle$ is taken as the one numerically found by Fuchs *et al.* in Ref. [32]. If the users want to see the specific formula of the SIC-POVM, the function *SIC(n)* can be called. The input *n* is the dimension of the density matrix. Currently, the function *SIC(n)* only valid when $n \leqslant 151$.

In both functions *QFIM()* and *CFIM()*, the outputs are real numbers ($\mathcal{F}_{aa}$ and $\mathcal{I}_{aa}$) in the single-parameter case, namely, when *drho* only contains one entry, and they are real symmetric or Hermitian matrices in the multi-parameter scenarios. The basis of QFIM and CFIM are determined by the order of entries in *drho*. For example, when *drho* is $[\partial_0 \rho, \partial_1 \rho, \dots]$, the basis of the QFIM and CFIM is $\{x_0, x_1, \dots\}$.

For some specific scenarios, the calculation method in *QFIM()* may be not efficient enough. Therefore, we also provide the calculation of QFIM in some specific scenarios. The first one is the calculation in the Bloch representation. In this case, the function for the calculation of QFIM is of the form

```
QFIM_Bloch(r, dr, eps = 1e-8)
```

The input *r* is an array representing a Bloch vector and *dr* is a list of arrays representing the derivatives of the Bloch vector on $\mathbf{x}$. Gaussian states are very commonly used in quantum metrology, and the corresponding QFIM can be calculated by the function

```
QFIM_Gauss(R, dR, D, dD)
```

The input *R* is an array representing the first-order moment, i.e., the expected value $\langle \mathbf{R} \rangle := \mathrm{Tr}(\rho \mathbf{R})$ of the vector $\mathbf{R} = (q_1, p_1, q_2, p_2, \dots)^{\mathrm{T}}$, where $q_i = (a_i + a_i^\dagger)/\sqrt{2}$ and $p_i = (a_i - a_i^\dagger)/(i\sqrt{2})$ are the quadrature operators with $a_i$ ($a_i^\dagger$) the annihilation (creation) operator of *i*th bosonic mode. *dR* is a list with *i*th entry also a list containing the derivatives $\partial_{\mathbf{x}}\langle[\mathbf{R}]_i\rangle$. Here $[\cdot]_i$ represents the *i*th entry of the vector. *D* is a matrix representing the second-order moment, $D_{ij} = \langle[\mathbf{R}]_i[\mathbf{R}]_j + [\mathbf{R}]_j[\mathbf{R}]_i\rangle/2$, and *dD* is a list of matrices representing the derivatives $\partial_{\mathbf{x}} D$. Notice that *QFIM_Bloch()* and *QFIM_Gauss()* can only compute the SLD-based QFIM.

*Example.* Now we present an example to show the usage of these functions. Consider a single qubit Hamiltonian $H = \omega \sigma_3/2$ with $\sigma_3$ a Pauli matrix and $\omega$ the frequency. Take $\omega$ as the parameter to be estimated and assume its true value (denoted by $\omega_{\mathrm{tr}}$) is 1. Planck unit ($\hbar = 1$) is applied
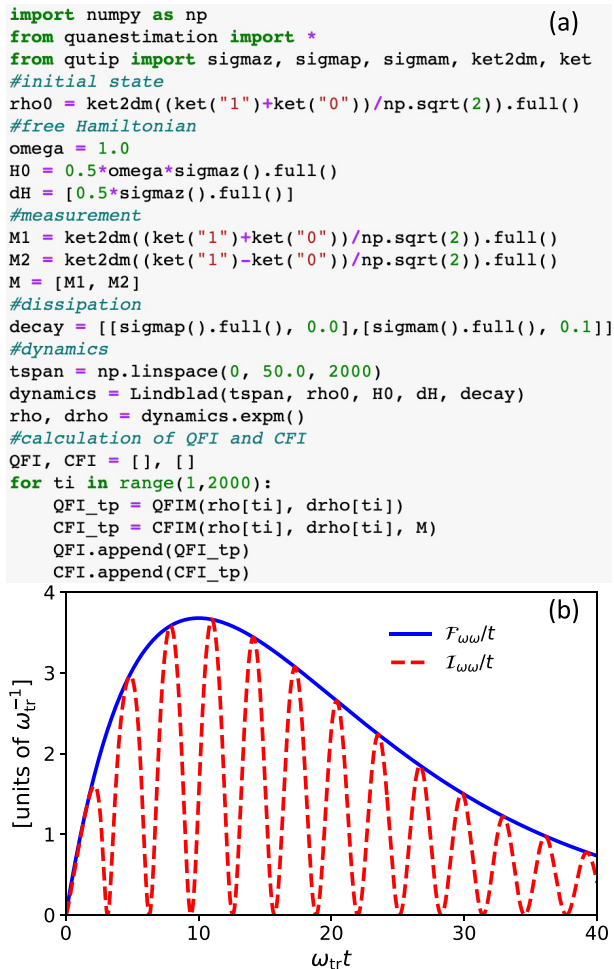
```python
import numpy as np
from quanestimation import *
from qutip import sigmaz, sigmap, sigmam, ket2dm, ket
#initial state
rho0 = ket2dm((ket("1")+ket("0"))/np.sqrt(2)).full()
#free Hamiltonian
omega = 1.0
H0 = 0.5*omega*sigmaz().full()
dH = [0.5*sigmaz().full()]
#measurement
M1 = ket2dm((ket("1")+ket("0"))/np.sqrt(2)).full()
M2 = ket2dm((ket("1")-ket("0"))/np.sqrt(2)).full()
M = [M1, M2]
#dissipation
decay = [[sigmap().full(), 0.0],[sigmam().full(), 0.1]]
#dynamics
tspan = np.linspace(0, 50.0, 2000)
dynamics = Lindblad(tspan, rho0, H0, dH, decay)
rho, drho = dynamics.expm()
#calculation of QFI and CFI
QFI, CFI = [], []
for ti in range(1,2000):
    QFI_tp = QFIM(rho[ti], drho[ti])
    CFI_tp = CFIM(rho[ti], drho[ti], M)
    QFI.append(QFI_tp)
    CFI.append(CFI_tp)
```

(a)



(b)

FIG. 2. (a) The demonstrating code for the calculation of QFI and CFI with QuanEstimation. (b) The evolution of $\mathcal{F}_{\omega\omega}/t$ (solid-blue line) and $\mathcal{I}_{\omega\omega}/t$ (dashed-red line). The initial state is $|+\rangle$. The true value of $\omega$ ($\omega_{\text{tr}}$) is set to be 1, and the decay rates are set to be $\gamma_+/\omega_{\text{tr}} = 0$ and $\gamma_-/\omega_{\text{tr}} = 0.1$. Planck units are applied here.

in the Hamiltonian. The dynamics is governed by the master equation

$$\partial_t \rho = -i[H, \rho] + \gamma_+ \left(\sigma_+ \rho \sigma_- - \tfrac{1}{2}\{\sigma_- \sigma_+, \rho\}\right) + \gamma_- \left(\sigma_- \rho \sigma_+ - \tfrac{1}{2}\{\sigma_+ \sigma_-, \rho\}\right), \quad (12)$$

where $\sigma_\pm = (\sigma_1 \pm \sigma_2)/2$ with $\sigma_1$, $\sigma_2$ also Pauli matrices. $\gamma_+$ and $\gamma_-$ are the decay rates. The measurement is taken as $\{|+\rangle\langle+|, |-\rangle\langle-|\}$ with

$$|\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle). \quad (13)$$

Here $|0\rangle$ ($|1\rangle$) is the eigenstate of $\sigma_3$ with respect to the eigenvalue 1 ($-1$). The specific code for the calculation of QFI/CFI are given in Fig. 2(a), and the corresponding evolution of $\mathcal{F}_{\omega\omega}/t$ (solid-blue line) and $\mathcal{I}_{\omega\omega}/t$ (dashed-red line) are shown in Fig. 2(b). The operators such as the density matrix and measurement can either be generated via QuTiP as in the demonstrating code or direct input.

### B. Holevo Cramér-Rao bound

Holevo Cramér-Rao bound (HCRB) is another useful asymptotic bound in quantum parameter estimation and tighter than the quantum Cramér-Rao bound in general. The HCRB can be expressed as [14,35–37]

$$\text{Tr}(W \text{cov}(\hat{\mathbf{x}}, \{\Pi_y\})) \geqslant \min_{\mathbf{X},V} \text{Tr}(WV) \quad (14)$$

with $W$ the weight matrix and $V$ a matrix satisfying $V \geqslant Z(\mathbf{X})$. Here $Z(\mathbf{X})$ is a Hermitian matrix and its $ab$th entry is defined by $[Z(\mathbf{X})]_{ab} := \text{Tr}(\rho X_a X_b)$, where $\mathbf{X} = [X_0, X_1, \cdots]$ is a vector of operators and its $a$th entry is defined by $X_a := \sum_y (\hat{x}_a - x_a)\Pi_y$ with $\hat{x}_a$ the $a$th entry of $\hat{\mathbf{x}}$. To let the local estimator $\hat{\mathbf{x}}$ unbiased, $\mathbf{X}$ needs to satisfy $\text{Tr}(X_a \partial_b \rho) = \delta_{ab}$, $\forall a, b$. Here $\delta_{ab}$ is the Kronecker delta function. An equivalent formulation of HCRB is [14,35–37]

$$\min_{\mathbf{X},V} \text{Tr}(WV) = \min_{\mathbf{X}} \text{Tr}(W\text{Re}(Z)) + \|\sqrt{W}\text{Im}(Z)\sqrt{W}\|, \quad (15)$$

where $\text{Re}(Z)$ and $\text{Im}(Z)$ represent the real and imaginary parts of $Z$, and $\|\cdot\|$ is the trace norm, i.e., $\|A\| := \text{Tr}\sqrt{A^\dagger A}$ for a matrix $A$. Numerically, in a specific matrix basis $\{\lambda_i\}$, which satisfies $\text{Tr}(\lambda_i\lambda_j) = \delta_{ij}$, the HCRB can be solved via the semidefinite programming as it can be reformulated into a linear semidefinite problem [38],

$$\min_{\mathbf{X},V} \text{Tr}(WV),$$

$$\text{subject to} \begin{cases} \begin{pmatrix} V & \Lambda^{\mathrm{T}}R^\dagger \\ R\Lambda & \mathbb{1} \end{pmatrix} \geqslant 0, \\ \sum_i [\Lambda]_{ai}\text{Tr}(\lambda_i \partial_b \rho) = \delta_{ab}. \end{cases} \quad (16)$$

Here the $ij$th entry of $\Lambda$ is obtained by decomposing $\mathbf{X}$ in the basis $\{\lambda_i\}$, $X_i = \sum_j [\Lambda]_{ij}\lambda_j$, and $R$ satisfies $Z = \Lambda^{\mathrm{T}}R^\dagger R\Lambda$. The semidefinite programming can be solved by the package CVXPY [39,40] in Python and Convex [41] in Julia. In QuanEstimation, the HCRB can be calculated via the function:

```
HCRB(rho, drho, W, eps = 1e-8)
```

The input $W$ is the weight matrix and $rho$, $drho$ have been introduced previously. Since $Z_{aa}$ is equivalent to the variance of the unbiased observable $O := \sum_y \hat{x}_a \Pi_y$ [unbiased condition is $\text{Tr}(\rho O) = x$], i.e., $Z_{aa} = \text{Tr}(\rho O^2) - [\text{Tr}(\rho O)]^2$, in the case of single-parameter estimation the optimal $V$ is nothing but $Z_{aa}$ itself. Furthermore, it can be proved that $Z_{aa} \geqslant 1/\mathcal{F}_{aa}$ and the equality is attainable asymptotically. Hence, one can see that $\min_{X_a} Z_{aa} = 1/\mathcal{F}_{aa}$, which means the HCRB is equivalent to the quantum Cramér-Rao bound in the single-parameter estimation. Due to better numerical efficiency of QFI computation, whenever $drho$ has only one entry, the calling of $HCRB()$ will automatically jump to $QFIM()$ in the package. Similarly, if $W$ is a rank-one matrix, the HCRB also reduces to $\text{Tr}(W\mathcal{F}^{-1})$ and thus in this case the calculation of HCRB will also be replaced by the calculation of QFIM.

*Example.* Now let us take a two-parameter estimation as an example to demonstrate the calculation of HCRB with QuanEstimation. Consider a two-qubit system with the XX coupling. The Hamiltonian of this system is

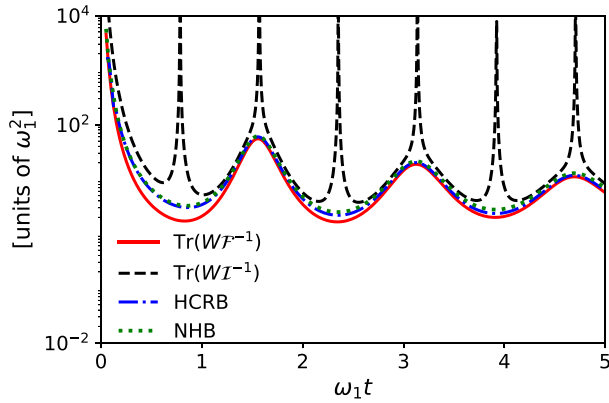$$H = \omega_1 \sigma_3^{(1)} + \omega_2 \sigma_3^{(2)} + g\sigma_1^{(1)}\sigma_1^{(2)}, \quad (17)$$

FIG. 3. Time evolution of $\mathrm{Tr}(W\mathcal{F}^{-1})$ (solid-red line), $\mathrm{Tr}(W\mathcal{I}^{-1})$ (dashed-black line), HCRB (dash-dotted-blue line), and NHB (dotted-green line) in the case of two-qubit system with the XX coupling. The probe state is $(|00\rangle + |11\rangle)/\sqrt{2}$. $W = \mathbb{1}$ and $\omega_1 = 1$. The true values of $\omega_2$ and $g$ are 1 and 0.1, respectively. The decay rates $\gamma_1 = \gamma_2 = 0.05\omega_1$. The POVM for $\mathrm{Tr}(W\mathcal{I}^{-1})$ is $\{\Pi_1, \Pi_2, \mathbb{1} - \Pi_1 - \Pi_2\}$ with $\Pi_1 = 0.85|00\rangle\langle 00|$ and $\Pi_2 = 0.4|++\rangle\langle ++|$. Planck units are applied here.

where $\omega_1$, $\omega_2$ are the frequencies of the first and second qubit, $\sigma_i^{(1)} = \sigma_i \otimes \mathbb{1}$, and $\sigma_i^{(2)} = \mathbb{1} \otimes \sigma_i$ for $i = 1, 2, 3$. $\mathbb{1}$ is the identity matrix. Planck units are applied here ($\hbar = 1$). The parameters $\omega_2$ and $g$ are the ones to be estimated. The dynamics is governed by the master equation

$$\partial_t \rho = -i[H, \rho] + \sum_{i=1,2} \gamma_i \big(\sigma_3^{(i)} \rho \sigma_3^{(i)} - \rho\big) \qquad (18)$$

with $\gamma_i$ the decay rate for $i$th qubit. The time evolutions of quantum Cramér-Rao bound [$\mathrm{Tr}(W\mathcal{F}^{-1})$], classical Cramér-Rao bound [$\mathrm{Tr}(W\mathcal{I}^{-1})$], and HCRB are shown in Fig. 3. The POVM for $\mathrm{Tr}(W\mathcal{I}^{-1})$ is $\{\Pi_1, \Pi_2, \mathbb{1} - \Pi_1 - \Pi_2\}$ with $\Pi_1 = 0.85|00\rangle\langle 00|$ and $\Pi_2 = 0.4|++\rangle\langle ++|$. The probe state is $(|00\rangle + |11\rangle)/\sqrt{2}$ and the weight matrix $W = \mathbb{1}$. As shown in this plot, HCRB (dash-dotted-blue line) is tighter than $\mathrm{Tr}(W\mathcal{F}^{-1})$ (solid-red line), which is in agreement with the fact that the HCRB is in general tighter than the quantum Cramér-Rao bound, unless the quantum Cramér-Rao bound is attainable, in which case the two bounds coincide [14].

### C. Nagaoka-Hayashi bound

Apart from the HCRB, the Nagaoka-Hayashi bound (NHB) [37,42,43] is another available bound for quantum multiparameter estimation, and is tighter than the HCRB in general. The expression of the NHB is

$$\mathrm{Tr}(W\mathrm{cov}(\hat{\mathbf{x}}, \{\Pi_y\})) \geqslant \min_{\mathbf{X}, \mathcal{Q}} \mathrm{Tr}((W \otimes \rho)\mathcal{Q}). \qquad (19)$$

Here $\mathcal{Q}$ is a symmetric block matrix with each block a Hermitian matrix, and it satisfies $\mathcal{Q} \geqslant \mathbf{X}^{\mathrm{T}}\mathbf{X}$ with $\mathbf{X}$ defined in Sec. IV B, namely, $\mathbf{X} = [X_0, X_1, \cdots]$ and $X_a = \sum_y (\hat{x}_a - x_a)\Pi_y$. Similar to the HCRB, the calculation of NHB can also be reformulated into a linear semidefinite problem [43]

as follows:

$$\min_{\mathbf{X}, \mathcal{Q}} \mathrm{Tr}((W \otimes \rho)\mathcal{Q}),$$

$$\text{subject to} \begin{cases} \begin{pmatrix} \mathcal{Q} & \mathbf{X}^{\mathrm{T}} \\ \mathbf{X} & \mathbb{1} \end{pmatrix} \geqslant 0, \\ \mathrm{Tr}(\rho X_a) = 0, \ \forall a, \\ \mathrm{Tr}(X_a \partial_b \rho) = \delta_{ab}, \ \forall a, b. \end{cases} \qquad (20)$$

In QuanEstimation, the NHB can be calculated via the function:

```
NHB(rho, drho, W)
```

The performance of the NHB is also demonstrated in Fig. 3 with the Hamiltonian in Eq. (17) and dynamics in Eq. (18). In this case the NHB (dotted-green line) is indeed slightly tighter than the HCRB, and thus also tighter than the quantum Cramér-Rao bound [$\mathrm{Tr}(W\mathcal{F}^{-1})$]. However, there still exist a gap between the classical Cramér-Rao bound [$\mathrm{Tr}(W\mathcal{I}^{-1})$] and the NHB, indicating that the chosen measurement may not be an optimal one.

### D. Bayesian estimation

Bayesian estimation is another well-used method in parameter estimation, in which the prior distribution is updated via the posterior distribution obtained by the Bayes' rule

$$p(\mathbf{x}|y) = \frac{p(y|\mathbf{x})p(\mathbf{x})}{\int p(y|\mathbf{x})p(\mathbf{x})d\mathbf{x}}, \qquad (21)$$

where $p(\mathbf{x})$ is the current prior distribution, $y$ is the result obtained in practice, and $\int d\mathbf{x} := \int dx_0 \int dx_1 \cdots$. The prior distribution is then updated with $p(\mathbf{x}|y)$, and the estimated value of $\mathbf{x}$ is obtained via a reasonable estimator, such as the expected value $\hat{\mathbf{x}} = \int \mathbf{x} p(\mathbf{x}|y)d\mathbf{x}$ or the maximum a posteriori estimation (MAP), $\hat{\mathbf{x}} = \mathrm{argmax}_{\mathbf{x}} \, p(\mathbf{x}|y)$.

In QuanEstimation, the Bayesian estimation can be performed via the function:

```
pout, xout = Bayes(x, p, rho, y, M = [],
            estimator = "mean", savefile = False)
```

The input $x$ is a list of arrays representing the regimes of $\mathbf{x}$, which is the same with the function *BayesInput()* discussed in Sec. III. Notice that in the package all the calculations of the integrals over the prior distributions are performed discretely. Hence, for now the input prior distribution is required to be an array, instead of a continuous function. $p$ is an array representing the values of $p(\mathbf{x})$ with respect to $\mathbf{x}$. It is multidimensional in the case of multiparameter estimation, i.e., the entry number of $x$ are at least two. The input *rho* is a (multidimensional) list of matrices representing the values of density matrix with respect to all values of $\mathbf{x}$, which can be alternatively generated via the function *BayesInput()* if specific functions of $H$ and $\partial_{\mathbf{x}} H$ on $\mathbf{x}$ can be provided. $M=[]$ is a list of matrices representing a set of POVM and its default setting is a SIC-POVM. $y$ is an array representing the results obtained in an experiment. The result corresponds to the POVM operator input in $M$, which means it is an integer between 0 and $d - 1$ with $d$ the entry number of the set of POVM. The type of estimator can be set via *estimator* = " " and currently it has
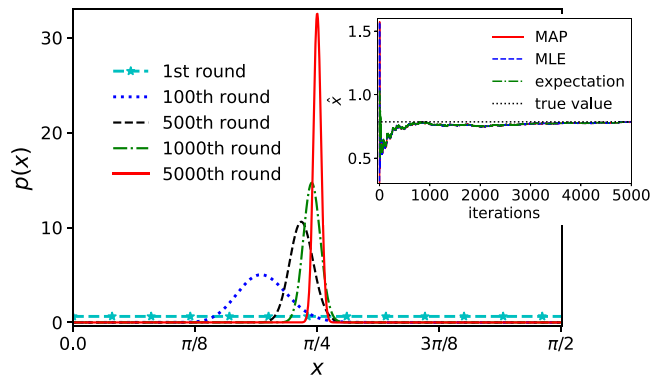
FIG. 4. Iteration of posterior distribution by the Bayes' rule. The inset shows the change of estimated value as a function of iteration for MAP (solid-red line), MLE (dashed-blue line), and expectation (dash-dotted-green line). The dotted-black line represents the true value.

two choices. When *estimator = "mean"* the estimator is the expected value, and when *estimator = "MAP"* the estimator is the MAP. The output *pout* (a multidimensional array) and *xout* (an array) are the final posterior distribution and estimated value of **x** obtained via the chosen estimator. When *savefile = True*, two files "pout.npy" and "xout.npy" will be generated, which include the updated $p(\mathbf{x})$ and the corresponding optimal **x** in all rounds. If the users call this function in the full-Julia package, the output files are "pout.csv" and "xout.csv".

*Example.* Now let us consider a simple example with the Hamiltonian

$$H = \frac{\kappa \omega_0}{2}(\sigma_1 \cos x + \sigma_3 \sin x), \qquad (22)$$

where $x$, $\kappa$ are two dimensionless parameters and $x$ is taken as the unknown one. Planck units are applied here ($\hbar = 1$) and $\omega_0$ is set to be 1. The initial state is taken as $|+\rangle$ and the target time $\omega_0 T = 1$. The prior distribution is assumed to be uniform in the regime $[0, \pi/2]$. The measurement is $\{|+\rangle\langle+|, |-\rangle\langle-|\}$. The results in experiment are simulated by a random generation according to the probabilities $p(\pm|x) = \langle\pm|\rho|\pm\rangle$ with respect to the value $x = \pi/4$. As shown in Fig. 4, with the growth of iteration number, the deviation decreases monotonously and the estimated value (center value of the distribution) approaches to $\pi/4$, which can also be confirmed by the convergence of estimated value (solid-red line) shown in the inset. As a matter of fact, here the maximum likelihood estimation (MLE) can also provide similar performance by taking the likelihood function with the MAP estimator $\hat{\mathbf{x}} = \text{argmax}_{\mathbf{x}} \prod_i p(y_i|\mathbf{x})$ (dashed-blue line in the inset). In QuanEstimation, this MLE can be calculated by the function

```
Lout, xout = MLE(x, rho, y, M=[], savefile=False)
```

When *savefile = True*, two files "Lout.npy" and "xout.npy" will be generated including all the data in the iterations.

In Bayesian estimation, another useful tool is the average Bayesian cost [44] for the quadratic cost, which is defined by

$$\bar{C} := \int p(\mathbf{x}) \sum_y p(y|\mathbf{x})(\mathbf{x} - \hat{\mathbf{x}})^{\mathrm{T}} W(\mathbf{x} - \hat{\mathbf{x}}) \, d\mathbf{x} \qquad (23)$$

with $W$ the weight matrix. In QuanEstimation, this average Bayesian cost can be calculated via the function:

```
BayesCost(x, p, xest, rho, M=[], eps=1e-8)
```

Here $x$ and $p$ are the same with those in *Bayes()*. *xest* is a list of arrays representing the estimator $\hat{\mathbf{x}}$. The $i$th entry of each array in *xest* represents the estimator with respect to $i$th result. In the case of the single-parameter scenario, $W$ is chosen to be 1 regardless of the input. The average Bayesian cost satisfies the inequality [14]

$$\bar{C} \geqslant \int p(\mathbf{x})(\mathbf{x}^{\mathrm{T}} W \mathbf{x}) d\mathbf{x} - \sum_{ab} W_{ab} \text{Tr}(\bar{\rho} \bar{L}_a \bar{L}_b), \qquad (24)$$

where $\bar{\rho} := \int p(\mathbf{x})\rho \, d\mathbf{x}$ and the operator $\bar{L}_a$ is determined by the equation $\int x_a p(\mathbf{x})\rho \, d\mathbf{x} = (\bar{L}_a\bar{\rho} + \bar{\rho}\bar{L}_a)/2$. In the case of the single-parameter scenario, the inequality above reduces to

$$\bar{C} \geqslant \int p(x)x^2 \, dx - \text{Tr}(\bar{\rho}\bar{L}^2) \qquad (25)$$

and represents a bound, which is always saturable—the optimal measurement correspond to projection measurement in the eigenbasis of $\bar{L}$, while the corresponding eigenvalues represent the estimated values of the parameter. If the mean value $\int p(x)x \, dx$ is subtracted to zero, then the inequality above can be rewritten into $\bar{C} \geqslant \delta^2 x - \text{Tr}(\bar{\rho}\bar{L}^2)$ with $\delta^2 x := \int p(x)x^2 \, dx - \int p(x)x \, dx$ the variance of $x$ under the prior distribution. In QuanEstimation, the bound given in Eq. (24) can be calculated via the following function:

```
BCB(x, p, rho, W=[], eps=1e-8)
```

Here the inputs $x$ and $p$ are the some with those in *Bayes()* and *BayesCost()*. $W$ represents the weight matrix and the default value is the identity matrix.

### E. Bayesian Cramér-Rao bounds

In the Bayesian scenarios, the quantum Cramér-Rao Bounds and Holevo Cramér-Rao bound are not appropriate to grasp the the ultimate precision limits as they are ignorant of the prior information. Still, Bayesian Cramér-Rao bounds can be used instead. In these scenarios, the covariance matrix is redefined as

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) = \int p(\mathbf{x}) \sum_y \text{Tr}(\rho \Pi_y)(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^{\mathrm{T}} d\mathbf{x}, \quad (26)$$

where the integral $\int d\mathbf{x} := \iiint dx_0 dx_1 \cdots$. In such cases, one version of the Bayesian Cramér-Rao bound (BCRB) is of the form

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \int p(\mathbf{x})(B\mathcal{I}^{-1}B + \mathbf{b}\mathbf{b}^{\mathrm{T}}) d\mathbf{x}, \qquad (27)$$

where $\mathcal{I}$ is the CFIM, and $\mathbf{b} = (b(x_0), b(x_1), \dots)^{\mathrm{T}}$ is the vector of biases, i.e., $b(x_a) = \sum_y \hat{x}_a p(y|\mathbf{x}) - x_a$ for each $x_a$ with $p(y|\mathbf{x})$ the conditional probability. $B$ is a diagonal matrix with the $a$th entry $B_{aa} = 1 + [\mathbf{b}']_a$. Here $\mathbf{b}' := (\partial_0 b(x_0), \partial_1 b(x_1), \dots)^{\mathrm{T}}$. The quantum correspondence of this

bound (BQCRB) reads

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \int p(\mathbf{x})(B\mathcal{F}^{-1}B + \mathbf{b}\mathbf{b}^{\mathrm{T}})d\mathbf{x}, \qquad (28)$$

where $\mathcal{F}$ is the QFIM of all types. As a matter of fact, there exists a similar version of Eq. (27), which can be expressed by

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \mathcal{B}\,\mathcal{I}_{\text{Bayes}}^{-1}\,\mathcal{B} + \int p(\mathbf{x})\mathbf{b}\mathbf{b}^{\mathrm{T}}d\mathbf{x}, \qquad (29)$$

where $\mathcal{I}_{\text{Bayes}} = \int p(\mathbf{x})\mathcal{I}d\mathbf{x}$ is the average CFIM with $\mathcal{I}$ the CFIM defined in Eq. (6). $\mathcal{B} = \int p(\mathbf{x})Bd\mathbf{x}$ is the average of $B$. Its quantum correspondence reads

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \mathcal{B}\,\mathcal{F}_{\text{Bayes}}^{-1}\,\mathcal{B} + \int p(\mathbf{x})\mathbf{b}\mathbf{b}^{\mathrm{T}}d\mathbf{x}, \qquad (30)$$

where $\mathcal{F}_{\text{Bayes}} = \int p(\mathbf{x})\mathcal{F}d\mathbf{x}$ is average QFIM with $\mathcal{F}$ the QFIM of all types.

Another version of the Bayesian Cramér-Rao bound is of the form

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \int p(\mathbf{x})\mathcal{G}(\mathcal{I}_p + \mathcal{I})^{-1}\mathcal{G}^{\mathrm{T}}d\mathbf{x}, \qquad (31)$$

and its quantum correspondence can be expressed by

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant \int p(\mathbf{x})\mathcal{G}(\mathcal{I}_p + \mathcal{F})^{-1}\mathcal{G}^{\mathrm{T}}d\mathbf{x}, \qquad (32)$$

where the entries of $\mathcal{I}_p$ and $\mathcal{G}$ are defined by

$$[\mathcal{I}_p]_{ab} := [\partial_a \ln p(\mathbf{x})][\partial_b \ln p(\mathbf{x})], \qquad (33)$$

and $\mathcal{G}_{ab} := [\partial_b \ln p(\mathbf{x})][\mathbf{b}]_a + B_{aa}\delta_{ab}$. The derivations and thorough discussions of these bounds will be further discussed in an independent paper, which will be announced in a short time.

The functions in QuanEstimation to calculate $\mathcal{I}_{\text{Bayes}}$ and $\mathcal{F}_{\text{Bayes}}$ are:

```
BCFIM(x, p, rho, drho, M = [], eps = 1e − 8)
BQFIM(x, p, rho, drho, LDtype = "SLD", eps = 1e − 8)
```

And the functions for the calculations of BCRBs and BQCRBs are:

```
BCRB(x, p, dp, rho, drho, M = [], b = [], db = [],
      btype = 1, eps = 1e − 8)
BQCRB(x, p, dp, rho, drho, b = [], db = [], btype = 1,
      LDtype = "SLD", eps = 1e − 8)
```

The input $x$ and $p$ are the same with those in the function *Bayes()*. $dp$ is a (multidimensional) list of arrays representing the derivatives of the prior distribution, which is only essential when $btype = 3$. In the case that $btype = 1$ and $btype = 2$, it could be set as *[]*. $rho$ and $drho$ are (multidimensional) lists representing the values of $\rho$ and $\partial_{\mathbf{x}}\rho$. For example, if the input $x$ includes three arrays, which are the values of $x_0$, $x_1$, and $x_2$ for the integral, then the $ijk$th entry of $rho$ and $drho$ are a matrix $\rho$ and a list $[\partial_0\rho, \partial_1\rho, \partial_2\rho]$ with respect to the values $[x_0]_i$, $[x_1]_j$, and $[x_2]_k$. Here $[x_0]_i$, $[x_1]_j$, and $[x_2]_k$ represent the $i$th, $j$th, and $k$th value in the first, second, and third array in $x$. As a matter of fact, if the users can provide specific functions of $H$ and $\partial_{\mathbf{x}}H$ on $\mathbf{x}$, $rho$ and $drho$ can be alternatively generated via the functions *BayesInput()* and *Lindblad()* [or *Kraus()*]. $b$ and $db$ are two lists of arrays representing $\mathbf{b}$ and $\mathbf{b}'$, and the

default settings for both of them are zero vectors (unbiased). In *BCRB()* the measurement is input via $M = []$, and if it is empty, a set of rank-one SIC-POVM will be automatically applied, similar to that in *CFIM()*. Moreover, $btype = 1$, $btype = 2$, and $btype = 3$ represent the calculation of Eqs. (27), (29), and (31). In the meantime, in *BQCRB()*, $btype = 1$, $btype = 2$, and $btype = 3$ represent the calculation of Eqs. (28), (30), and (32). Similar to *QFIM()*, $LDtype = $ " " here is the type of logarithmic derivatives, including three choices: *"SLD"*, *"RLD"*, and *"LLD"*. Recently, Ref. [45] provide an optimal biased bound based on the type-1 BQCRB in the case of single-parameter estimation, which can be calculated in QuanEstimation via the function:

```
OBB(x, p, dp, rho, drho, d2rho, LDtype = "SLD",
     eps = 1e − 8)
```

The input $dp$ is an array containing the derivatives $\partial_x p$. $d2rho$ is a list containing the second-order derivative of the density matrix on the unknown parameter.

Another famous Bayesian version of Cramér-Rao bound is introduced by Van Trees in 1968 [46], which is known as the Van Trees bound (VTB). The VTB is expressed by

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant (\mathcal{I}_{\text{prior}} + \mathcal{I}_{\text{Bayes}})^{-1}, \qquad (34)$$

where $\mathcal{I}_{\text{prior}} = \int p(\mathbf{x})\mathcal{I}_p d\mathbf{x}$ is the CFIM for $p(\mathbf{x})$ with $\mathcal{I}_p$ defined in Eq. (33). In the derivation, the assumption

$$\int \partial_a[b(x_b)p(\mathbf{x})]d\mathbf{x} = 0 \qquad (35)$$

is applied for all subscripts $a$ and $b$. In 2011, Tsang, Wiseman and Caves [47] provided a quantum correspondence of the VTB (QVTB). The Tsang-Wiseman-Caves bound is of the form

$$\text{cov}(\hat{\mathbf{x}}, \{\Pi_y\}) \geqslant (\mathcal{I}_{\text{prior}} + \mathcal{F}_{\text{Bayes}})^{-1}. \qquad (36)$$

The functions in QuanEstimation for the calculation of VTB and QVTB are:

```
VTB(x, p, dp, rho, drho, M = [], eps = 1e − 8)
QVTB(x, p, dp, rho, drho, LDtype = "SLD", eps = 1e − 8)
```

Here $dp$ is a (multidimensional) list of arrays representing the derivatives of the prior distribution. For example, if $x$ includes 3 arrays, which are the values of $x_0$, $x_1$, and $x_2$ for the integral, then the $ijk$th entry of $dp$ is an array $(\partial_0 p, \partial_1 p, \partial_2 p)$ with respect to values $[x_0]_i$, $[x_1]_j$, and $[x_2]_k$.

*Example.* Let us still take the Hamiltonian in Eq. (22) and initial state $|+\rangle$ as an example. $x$ is still the parameter to be estimated. The prior distribution is taken as a Gaussian distribution

$$p(x) = \frac{1}{c\eta\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\eta^2}} \qquad (37)$$

in a finite regime $[-\pi/2, \pi/2]$, where $\mu$ is the expectation, $\eta$ is the standard deviation, and $c = \frac{1}{2}[\text{erf}(\frac{\pi-2\mu}{2\sqrt{2}\eta}) + \text{erf}(\frac{\pi+2\mu}{2\sqrt{2}\eta})]$ is the normalized coefficient. Here $\text{erf}(x) := \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ is the error function. The measurement in the classical bounds is taken as a set of SIC-POVM. The performance of the classical and quantum Bayesian bounds are given in Figs. 5(a)
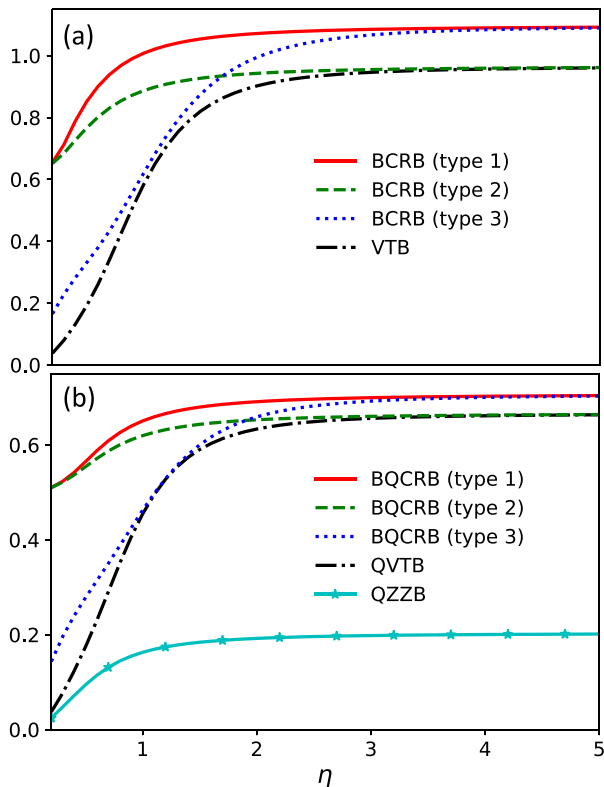
FIG. 5. (a) The performance of classical Bayesian bounds, including BCRB of type 1 (solid-red line), type 2 (dashed-green line), type 3 (dotted-blue line), and VTB (dash-dotted-black line). (b) The performance of quantum Bayesian bounds, including BQCRB of type 1 (solid-red line), type 2 (dashed-green line), type 3 (dotted-blue line), QVTB (dash-dotted-black line), and QZZB (solid-cyan-pentagram line). The parameters $\mu = 0$ and $\kappa = \pi/2$ in the plots. Planck units are applied here.

and 5(b). As shown in Fig. 5(a), in this case BCRB of type 1 (solid-red line) and type 2 (dashed-green line) are tighter than type 3 (dotted-blue line) and VTB (dash-dotted-black line) when the deviation $\eta$ is small. With the increase of $\eta$, BCRB of type 1 and type 3 coincide with each other, so do BCRB of type 2 and VTB. Furthermore, BCRB of type 1 and type 3 are always tighter than type 2 and VTB in this example. The performance of quantum Bayesian bounds are similar, as shown in Fig. 5(b). BQCRB (solid-red line for type 1 and dashed-green line for type 2) are tighter than type 3 (dotted-green line) and QVTB (dash-dotted-black line) when $\eta$ is small and BQCRB of type 1 (type 2) and type 3 (QVTB) coincide with each other for a large $\eta$.

### F. Quantum Ziv-Zakai bound

Apart from the Cramér-Rao bounds, the Ziv-Zakai bound is another useful bound in Bayesian scenarios. It was first provided by Ziv and Zakai in 1969 [48] for the single-parameter estimation and then extended to the linear combination of multiple parameters by Bell *et al.* [49], which is also referred to as the Bell-Ziv-Zakai bound. In 2012, Tsang provided a quantum correspondence of the Ziv-Zakai bound [50] (QZZB), and in 2015 Berry *et al.* [51] provided a quantum

correspondence of the Bell-Ziv-Zakai bound. In QZZB, the variance $\mathrm{var}(\hat{x}, \{\Pi_y\})$, a diagonal entry of the covariance matrix, satisfies the following inequality:

$$\mathrm{var}(\hat{x}, \{\Pi_y\}) \geqslant \frac{1}{2} \int_0^\infty d\tau \tau \mathcal{V} \int_{-\infty}^\infty dx \min\{p(x), p(x+\tau)\}$$
$$\times \left( 1 - \frac{1}{2} ||\rho(x) - \rho(x+\tau)|| \right), \qquad (38)$$

where $|| \cdot ||$ is the trace norm. $\mathcal{V}$ is the "valley-filling" operator satisfying $\mathcal{V}f(\tau) = \max_{h \geqslant 0} f(\tau + h)$. In the numerical calculations, the prior distribution has to be limited or truncated in a finite regime $[\alpha, \beta]$, i.e., $p(x) = 0$ when $x > \beta$ or $x < \alpha$, and then the QZZB reduces to

$$\mathrm{var}(\hat{x}, \{\Pi_y\}) \geqslant \frac{1}{2} \int_0^{\beta-\alpha} d\tau \tau \mathcal{V} \int_\alpha^\beta dx \min\{p(x), p(x+\tau)\}$$
$$\times \left( 1 - \frac{1}{2} ||\rho(x) - \rho(x+\tau)|| \right). \qquad (39)$$

The function in QuanEstimation for the calculation of QZZB is:

```
QZZB(x, p, rho, eps=1e-8)
```

The performance of QZZB is also demonstrated with the Hamiltonian in Eq. (22) and prior distribution in Eq. (37), as shown in Fig. 5(b). In this example, its performance (solid-cyan-pentagram line) is worse than BQCRB and QVTB. However, this tightness relation may dramatically change in other systems or with other prior distributions. Hence, in a specific scenario using QuanEstimation to perform a thorough comparison would be a good choice to find the tightest tool for the scheme design.

## V. METROLOGICAL RESOURCES

The improvement of precision usually means a higher consumption of resources. For example, the repetition of experiments will make the deviation of the unknown parameter to scale proportionally to $1/\sqrt{n}$ ($n$ the repetition number) in theory. The repetition number or the total time is thus the resource responsible for this improvement. Constraint on quantum resources is an important aspect in the study of quantum parameter estimation, and is crucial to reveal the quantum advantage achievable in practical protocols. The numerical calculations of some typical resources have been added in QuTiP, such as various types of entropy and the concurrence. Hence, we do not need to rewrite them in QuanEstimation. Currently, two additional metrological resources, spin squeezing and the time to reach a given precision limit are provided in the package. The spin squeezing can be calculated via the function:

```
SpinSqueezing(rho, basis="Dicke", output="KU")
```

Here the input *rho* is a matrix representing the state. The basis of the state can be adjusted via *basis* = " ". Two options *"Dicke"* and *"Pauli"* represent the Dicke basis and the original basis of each spin. *basis* = *"Pauli"* here is equivalent to choose *basis* = *"uncoupled"* in the function *jspin()* in QuTiP. Two types of spin squeezing can be calculated in this function.

*output = "KU"* means the output is the one given by Kitagawa and Ueda [52], and *output = "WBIMH"* means the output is the one given by Wineland *et al.* [53].

The time to reach a given precision limit can be calculated via the function:

```
TargetTime(f,tspan,func,*args,**kwargs)
```

Notice that the dynamics needs to be run first before using this function. For example, it is available to be called after the calling of both *Lindblad()* and *Lindblad.expm()*. The input *f* is a float number representing the given value of the precision limit. The time is searched within the regime defined by the input *tspan* (an array). *func* is the handle of a function *func()* depicting the precision limit. *\*args* is the corresponding input parameters, in which *rho* and *drho* should be the output of *Lindblad.expm()* [or *Lindblad.ode()* and any other method that may included in the future]. *\*\*kwargs* is the keyword arguments in *func()*. The difference between input parameters and keyword arguments in QuanEstimation is that the keyword arguments have default values and thus one does not have to assign values to them when calling the function. Currently, all the asymptotic bounds discussed in Sec. IV are available to be called here.

## VI. CONTROL OPTIMIZATION

Quantum control is a leading approach in quantum metrology to achieve the improvement of measurement precision and boost the resistance to decoherence. This is possible thanks to high controllability of typical quantum metrological setups. A paradigmatic controllable Hamiltonian is of the form

$$H = H_0(\mathbf{x}) + \sum_{k=1}^{K} u_k(t)H_k, \tag{40}$$

where $H_0(\mathbf{x})$ is the free Hamiltonian containing the unknown parameters $\mathbf{x}$ and $H_k$ is the $k$th control Hamiltonian with the corresponding control amplitude $u_k(t)$. In quantum parameter estimation, the aim of control is to improve the precision of the unknown parameters. Hence, natural choices for the the objective function $f$ are the various metrological bounds. The quantum Cramér-Rao bounds are easiest to calculate and hence will typically be the first choice. In the single-parameter estimation, the QFI or CFI can be taken as the objective function, depending whether the measurement can be optimized or is fixed. In the multiparameter scenario, the objective function can be $\mathrm{Tr}(W\mathcal{F}^{-1})$, $\mathrm{Tr}(W\mathcal{I}^{-1})$, or the HCRB.

Searching the optimal controls in order to achieve the maximum or minimum values of an objective function is the core task in quantum control. Most existing optimization algorithms, such as Gradient ascent pulse engineering [54–56], Krotov's method [57–59], and machine learning [60,61], are capable of providing useful control strategies in quantum parameter estimation. The gradient-based algorithms usually perform well in small-scale systems. For complex problems where the gradient-based methods are more challenging or even fail to work at all, gradient-free algorithms are a good alternative. Here we introduce several control algorithms in

quantum parameter estimation that have been added into our package and give some illustrations.

First, we present the specific code in QuanEstimation for the execution of the control optimization,

```
control = ControlOpt(savefile=False,
                method="auto−GRAPE",**kwargs)
control.dynamics(tspan,rho0,H0,dH,Hc,
                decay=[],ctrl_bound=[],
                dyn_method="expm")
control.QFIM(W=[],LDtype="SLD")
control.CFIM(M=[],W=[])
control.HCRB(W=[])
```

The input *tspan* is an array representing the time for the evolution. *rho0* is a matrix representing the density matrix of the initial state. *H0* is a matrix representing the free Hamiltonian $H_0(\mathbf{x})$ and *Hc* is a list containing the control Hamiltonians, i.e., $[H_1, H_2, \dots]$. *dH* is a list of matrices representing $\partial_{\mathbf{x}}H_0$. In the case that only one entry exists in *dH*, the objective functions in *control.QFIM()* and *control.CFIM()* are the QFI and CFI, and if more than one entries are input, the objective functions are $\mathrm{Tr}(W\mathcal{F}^{-1})$ and $\mathrm{Tr}(W\mathcal{I}^{-1})$. Different types of QFIM can be selected as the objective function via the variable *LDtype* = " ", which includes three options *"SLD"*, *"RLD"*, and *"LLD"*. The measurement for CFI/CFIM is input via *M = []* in *control.CFIM()* and the default value is a SIC-POVM. The weight matrix *W* can be manually input via *W = []*, and the default value is the identity matrix.

In some cases, the control amplitudes have to be limited in a regime, for example $[a, b]$, which can be realized by input *ctrl_bound = [a,b]*. If no value is input, the default regime is $[-\infty, \infty]$. *decay = []* is a list of decay operators and corresponding decay rates for the master equation in Eq. (1) and its input rule is *decay = [[Gamma_1,gamma_1],...]*. The dynamics is solved via the matrix exponential by default, which can be switched to ODE by setting *dyn_method = "ode"*. The default value for *savefile* is *False*, which means only the controls obtained in the final episode will be saved in the file named "controls.csv", and if it is set to be *True*, the controls obtained in all episodes will be saved in this file. The values of QFI, CFI, $\mathrm{Tr}(W\mathcal{F}^{-1})$ or $\mathrm{Tr}(W\mathcal{I}^{-1})$ in all episodes will be saved regardless of this setting in the file named "f.csv". Another file named "total_reward.csv" will also be saved to save the total rewards in all episodes when DDPG is chosen as the optimization method. Here the word "episode" is referred to as a round of update of the objective function in the scenario of optimization.

The switch of optimization algorithms can be realized by *method = " "*, and the corresponding parameters can be set via *\*\*kwargs*. All available algorithms in QuanEstimation are given in Table I together with the corresponding default parameter settings. Notice that in the case that *method = "auto-GRAPE"* is applied, *dyn_method = "ode"* is not available for now. In some algorithms maybe more than one set of guessed controls are needed, and if not enough sets are input then random-value controls will be generated automatically to fit the number. In the meantime, if excessive number of sets are input, only the suitable number of controls will be used. *LDtype = "SLD"* is the only choice when *method = "GRAPE"* as the QFIMs based on RLD and

TABLE I. Available control methods in QuanEstimation and corresponding default parameter settings. Notice that auto-GRAPE and GRAPE are not available when *control.HCRB()* is called.

| Algorithms | method= | **kwargs and default values | |
| --- | --- | --- | --- |
| auto-GRAPE | "auto-GRAPE" | "Adam" | True |
| (GRAPE) | ("GRAPE") | "ctrl0" | [] |
| | | "max_episode" | 300 |
| | | "epsilon" | 0.01 |
| | | "beta1" | 0.90 |
| | | "beta2" | 0.99 |
| PSO | "PSO" | "p_num" | 10 |
| | | "ctrl0" | [] |
| | | "max_episode" | [1000,100] |
| | | "c0" | 1.0 |
| | | "c1" | 2.0 |
| | | "c2" | 2.0 |
| | | "seed" | 1234 |
| DE | "DE" | "p_num" | 10 |
| | | "ctrl0" | [] |
| | | "max_episode" | 1000 |
| | | "c" | 1.0 |
| | | "cr" | 0.5 |
| | | "seed" | 1234 |
| DDPG | "DDPG" | "ctrl0" | [] |
| | | "max_episode" | 500 |
| | | "layer_num" | 3 |
| | | "layer_dim" | 200 |
| | | "seed" | 1234 |

LLD are unavailable to be the objective function for GRAPE in the package. All the aforementioned algorithms will be thoroughly introduced and discussed with examples in the following subsections.

Apart from the QFIM and CFIM, the HCRB can also be taken as the objective function in the case of multiparameter estimation, which can be realized by calling *control.HCRB()*. Notice that auto-GRAPE and GRAPE are not available in *method* = " " here as the calculation of HCRB is performed via optimizations (semidefinite programming), not direct calculations. Due to the equivalence between the HCRB and quantum Cramér-Rao bound in the single-parameter estimation, if *control.HCRB()* is called in this case, the entire program will be terminated and a line of reminder will arise to remind the users to invoke *control.QFIM()* instead.

### A. Gradient ascent pulse engineering

The gradient ascent pulse engineering algorithm (GRAPE) was developed by Khaneja *et al.* [54] in 2005 for the design of pulse sequences in the NMR systems, and then applied into the quantum parameter estimation for the generation of optimal controls [55,56], in which the gradients of the objective function $f(T)$ at a fixed time $T$ were obtained analytically. In the pseudocode given in Ref. [17], the propagators between any two time points have to be saved, which would occupy a large amount of memory during the computation and make it difficult to deal with high-dimensional Hamiltonians or long-time evolutions. To solve this problem, a modified pseudocode is provided as given in Algorithm 1. In this modified version, after obtaining the evolved state $\rho_t$ and $\partial_\mathbf{x}\rho_t$, the gradient

**Algorithm 1:** GRAPE

---

Initialize the control amplitude $u_k(t)$ for all $t$ and $k$;
**for** episode=1, $M$ **do**
  Receive initial state $\rho_0$ ($\rho_{\mathrm{in}}$);
  **for** $t = 1, T$ **do**
    Evolve with the control $\rho_t = e^{\Delta t \mathcal{L}_t}\rho_{t-1}$;
    Calculate the derivatives $\partial_\mathbf{x}\rho_t = -i\Delta t[\partial_\mathbf{x}H_0(\mathbf{x})]^\times \rho_t + e^{\Delta t \mathcal{L}_t}\partial_\mathbf{x}\rho_{t-1}$;
    Save $\rho_t$ and $\partial_\mathbf{x}\rho_t$;
    **for** $k = 1, K$ **do**
      Calculate $\frac{\delta\rho_t}{\delta u_k(t)} = -i\Delta t H_k^\times \rho_t$, $\partial_\mathbf{x}\left(\frac{\delta\rho_t}{\delta u_k(t)}\right) = -i\Delta t H_k^\times (\partial_\mathbf{x}\rho_t)$;
      **for** $j = t-1, 1$ **do**
        Calculate $\frac{\delta\rho_t}{\delta u_k(j)} = e^{\Delta t \mathcal{L}_t}\frac{\delta\rho_{t-1}}{\delta u_k(j)}$, $\partial_\mathbf{x}\left(\frac{\delta\rho_t}{\delta u_k(j)}\right) = \left(\partial_\mathbf{x}e^{\Delta t \mathcal{L}_t}\right)\frac{\delta\rho_{t-1}}{\delta u_k(j)} + e^{\Delta t \mathcal{L}_t}\partial_\mathbf{x}\left(\frac{\delta\rho_{t-1}}{\delta u_k(j)}\right)$;
      **end for**
      Save $\frac{\delta\rho_t}{\delta u_k(t)}$, $\partial_\mathbf{x}\left(\frac{\delta\rho_t}{\delta u_k(t)}\right)$ and all $\frac{\delta\rho_t}{\delta u_k(j)}$, $\partial_\mathbf{x}\left(\frac{\delta\rho_t}{\delta u_k(j)}\right)$;
    **end for**
  **end for**
  Calculate the SLDs for all $\mathbf{x}$ and the objective function $f(T)$;
  **for** $t = 1, T$ **do**
    **for** $k = 1, K$ **do**
      Calculate the gradient $\frac{\delta f(T)}{\delta u_k(t)}$ with $\frac{\delta\rho_T}{\delta u_k(t)}$ and $\partial_\mathbf{x}\left(\frac{\delta\rho_T}{\delta u_k(t)}\right)$;
      Update control $u_k(t) \leftarrow u_k(t) + \epsilon \frac{\delta f(T)}{\delta u_k(t)}$.
    **end for**
  **end for**
**end for**
Save the controls $\{u_k(t)\}$ and corresponding $f(T)$.

---

$\delta\rho_t/\delta u_k(t)$ and its derivatives with respect to **x** are then calculated via the equations

$$\frac{\delta\rho_t}{\delta u_k(t)} = -i\Delta t H_k^{\times}(\rho_t) \tag{41}$$

with $\Delta t$ a small time interval, $H_k^{\times}(\cdot) = [H_k, \cdot]$ the commutator between $H_k$ and other operators, and

$$\partial_{\mathbf{x}}\left(\frac{\delta\rho_t}{\delta u_k(t)}\right) = -i\Delta t H_k^{\times}(\partial_{\mathbf{x}}\rho_t), \tag{42}$$

The gradients $\delta\rho_t/\delta u_k(j)$ $(j < t)$ are calculated adaptively according to the equation

$$\frac{\delta\rho_t}{\delta u_k(j)} = e^{\Delta t \mathcal{L}_t}\frac{\delta\rho_{t-1}}{\delta u_k(j)} \tag{43}$$

and its derivatives are obtained via

$$\partial_{\mathbf{x}}\left(\frac{\delta\rho_t}{\delta u_k(j)}\right) = (\partial_{\mathbf{x}}e^{\Delta t \mathcal{L}_t})\frac{\delta\rho_{t-1}}{\delta u_k(j)} + e^{\Delta t \mathcal{L}_t}\partial_{\mathbf{x}}\left(\frac{\delta\rho_{t-1}}{\delta u_k(j)}\right). \tag{44}$$

In this process, only the gradients $\delta\rho_t/\delta u_k(t)$, $\delta\rho_t/\delta u_k(j)$ and their derivatives need to be saved for the further use in the next round, and will be deleted after that. This operation avoids the usage of propagators and thus saves a lot of memory during the computation. In this way, the gradients $\delta\rho_T/\delta u_k(t)$ ($t$ is any time here) are obtained adaptively and $\delta f(T)/\delta u_k(t)$ can then be calculated accordingly. The specific expression of $\delta f(T)/\delta u_k(t)$ can be found in Refs. [13,55,56]. Adam [62] is also applied in this algorithm for the further improvement of the computational efficiency.

Consider the dynamics governed by Eq. (12) with parameter settings in Fig. 2, and the control Hamiltonian

$$u_1(t)\sigma_1 + u_2(t)\sigma_2 + u_3(t)\sigma_3. \tag{45}$$

In the case of $\omega_{\mathrm{tr}}T = 5$ with zeros as the initial guess of the controls, the QFI converges in 35 rounds with Adam, yet it takes 780 rounds to converge when Adam is not applied.

### B. Auto-GRAPE

In the multiparameter estimation, the gradients of $\mathrm{Tr}(W\mathcal{F}^{-1})$ and $\mathrm{Tr}(W\mathcal{I}^{-1})$ are very difficult to obtain analytically when the length of **x** is large. This is because the analytical calculation of $\mathcal{F}^{-1}$ and $\mathcal{I}^{-1}$ are difficult, if not completely impossible. Hence, the functions $\sum_a W_{aa}/\mathcal{F}_{aa}$ and $\sum_a W_{aa}/\mathcal{I}_{aa}$, which are lower bounds of $\mathrm{Tr}(W\mathcal{F}^{-1})$ and $\mathrm{Tr}(W\mathcal{I}^{-1})$, or their inverse functions, are taken as the superseded objective functions in GRAPE [56]. Although it has been proved that these superseded functions show positive performance on the generation of controls, it is still possible that the direct use of $\mathrm{Tr}(W\mathcal{F}^{-1})$ and $\mathrm{Tr}(W\mathcal{I}^{-1})$ might bring better results. To investigate it, hereby we provide a new GRAPE algorithm based on the automatic differentiation technology. This algorithm is referred to as the auto-GRAPE in this paper.

Automatic differentiation (AD) is an emerging numerical technology in machine learning to evaluate the exact derivatives of an objective function [63]. Recently, Song *et al.* [64] used AD to generate controls with complex and optional constraints. AD decomposes the calculation of the objective function into some basic arithmetic and apply the chain rules

---

**Algorithm 2:** auto-GRAPE

Initialize the control amplitude $u_k(t)$ for all $t$ and $k$;
**for** episode=1, $M$ **do**
    Receive initial state $\rho_0$ ($\rho_{\mathrm{in}}$);
    **for** $t = 1, T$ **do**
        Evolve with the control $\rho_t = e^{\Delta t \mathcal{L}_t}\rho_{t-1}$;
        Calculate the derivatives
        $\partial_{\mathbf{x}}\rho_t = -i\Delta t[\partial_{\mathbf{x}}H_0(\mathbf{x})]^{\times}\rho_t + e^{\Delta t \mathcal{L}_t}\partial_{\mathbf{x}}\rho_{t-1}$;
        Save $\rho_t$ and $\partial_{\mathbf{x}}\rho_t$;
    **end for**
    Calculate the SLD and objective function $f(T)$.
    Calculate the gradient $\frac{\delta f(T)}{\delta u_k(t)}$ with the
    automatic differentiation method for all $t$ and $k$.
    **for** $t = 1, T$ **do**
        **for** $k = 1, K$ **do**
            Update control $u_k(t) \leftarrow u_k(t) + \epsilon\frac{\delta f(T)}{\delta u_k(t)}$.
        **end for**
    **end for**
**end for**
Save the controls $\{u_k(t)\}$ and corresponding $f(T)$.

---

to calculate the derivatives. AD not only provides high precision results of the derivatives, but its computing complexity is no more than the calculation of the objective function. Hence, it would be very useful to evaluate the gradient in GRAPE. Here we use a Julia package Zygote [65] to implement our auto-GRAPE algorithm. auto-GRAPE is available for all types of QFIM in the package. In the following we only discuss the SLD-based QFIM for simplicity. The pseudocode of auto-GRAPE for SLD-based QFIM is given in Algorithm 2. In Zygote, "array mutation" operations should be avoided as the evaluation of differentiation for such operations are not supported currently. However, the general numerical calculation of the SLD is finished via the calculation of its each entry, as shown in Eq. (9). This entry-by-entry calculation would inevitably cause mutations of the array, and cannot directly apply automatic differentiation for now. Here we introduce two methods to realize AD in our package. One method is to avoid the entry-by-entry calculation directly. Luckily, Šafránek provided a method for the calculation of SLD and QFIM in the Liouville space [66], which just avoids the entry-by-entry calculation. Denote vec($A$) as the column vector with respect to a $d$-dimensional matrix $A$ in Liouville space and vec($A$)$^{\dagger}$ as the conjugate transpose of vec($A$). The entry of $A$ is defined by $[\mathrm{vec}(A)]_{id+j} := A_{ij}$ ($i, j \in [0, d - 1]$ and the subscript of vec($A$) starts from 0). Then the SLD can be calculated via the equation [66]

$$\mathrm{vec}(L_a) = 2(\rho \otimes \mathbb{1} + \mathbb{1} \otimes \rho^*)^{-1}\mathrm{vec}(\partial_a\rho), \tag{46}$$

where $\rho^*$ is the conjugate of $\rho$. The above calculation procedure treats the array as an entirety and only contains basic linear algebra operations on the array. Hence, it is available to be used for the implementation of automatic differentiation with the existing tools like Zygote. In QuanEstimation, the vectorization of matrices are performed with the aforementioned method, i.e., $[\mathrm{vec}(A)]_{id+j} := A_{ij}$, in all Python scripts, yet in the Julia scripts, the vectorization is taken as $[\mathrm{vec}(A)]_{i+jd} := A_{ij}$ for the calculation convenience. Since all
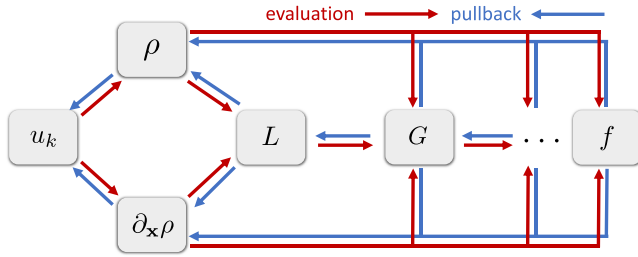
FIG. 6. The schematic of chain rules in automatic differentiation with the logarithmic derivative related functions as the objective function.

the outputs are converted back to the matrices, this difference would not affect the user experience. This method is easy to be implemented in coding, yet the dimension growth of the calculation in the Liouville space would significantly affect the computing efficiency and memory occupation. Therefore, we introduce the second method as follows.

The core of AD is utilizing the chain rules to evaluate the derivatives of the objective function. As illustrated in Fig. 6, in AD the value of the objective function $f$ is evaluated from left to right (red arrows), and the derivatives are calculated backwards (blue arrows), which is also called pullback in the language of AD. In our case, the differentiation of $f$ on a control amplitude $u_k$ needs to be evaluated through all three paths, from $f$ to $\rho$, from $f$ to $\partial_{\mathbf{x}}\rho$ (if $f$ is a function of $\partial_{\mathbf{x}}\rho$) and from $f$ to $G$ to $L$. Here $L$ represents the SLDs of all parameters and $G := G(L) = G(\rho, \partial_{\mathbf{x}}\rho)$ could be any intermediate function. For example, the contribution of the path from $f$ to $\rho$ to the derivative $df/du_k$ is $\frac{\partial f}{\partial \rho}\frac{\partial \rho}{\partial u_k}$. Notice that here $\partial f/\partial \rho$ is a formal derivative. The paths to $\rho$ and $\partial_{\mathbf{x}}\rho$ can be routinely solved in Zygote, however, the path to $L$ cannot be solved due to the entry-by-entry calculation of SLD in Eq. (9), which causes the difficulty to generate $\partial L/\partial \rho$ and $\partial L/\partial(\partial_{\mathbf{x}}\rho)$, and therefore $\partial G/\partial \rho$ and $\partial G/\partial(\partial_{\mathbf{x}}\rho)$ cannot be obtained. The chain rules in AD cannot be applied then. Hence, we need to manually provide $\partial G/\partial \rho$ and $\partial G/\partial(\partial_{\mathbf{x}}\rho)$ to let AD work in our case. To do it, one should first know that the total differentiation $dG_{\alpha\beta}$ (the $\alpha\beta$th entry of $dG$) can be evaluated via the equation

$$dG_{\alpha\beta} = \sum_{ij} \frac{\partial G_{\alpha\beta}}{\partial L_{ij}} dL_{ij} + \frac{\partial G_{\alpha\beta}}{\partial (L_{ij})^*} d(L_{ij})^*, \quad (47)$$

which can be written into a more compact matrix form

$$dG_{\alpha\beta} = \mathrm{Tr}\left(\left(\frac{\partial G_{\alpha\beta}}{\partial L}\right)^{\mathrm{T}} dL + \left(\frac{\partial G_{\alpha\beta}}{\partial L^*}\right)^{\mathrm{T}} dL^*\right). \quad (48)$$

Due to the fact that the SLD is a Hermitian matrix, one can have $dL^* = dL^{\mathrm{T}}$, and the equation above reduces to

$$dG_{\alpha\beta} = \mathrm{Tr}\left(\left(\frac{\partial G_{\alpha\beta}}{\partial L}\right)^{\mathrm{T}} dL + \frac{\partial G_{\alpha\beta}}{\partial L^{\mathrm{T}}} dL\right)$$

$$= 2\mathrm{Tr}\left(\left(\frac{\partial G_{\alpha\beta}}{\partial L}\right)^{\mathrm{T}} dL\right). \quad (49)$$

Now we introduce an auxiliary function $h$, which satisfies

$$\left(\frac{\partial G_{\alpha\beta}}{\partial L}\right)^{\mathrm{T}} = \rho h^{\mathrm{T}} + h^{\mathrm{T}}\rho. \quad (50)$$

This equation is a typical Lyapunov equation and can be numerically solved. Substituting the equation above into the expression of $dG_{\alpha\beta}$, one can find that

$$dG_{\alpha\beta} = 2\mathrm{Tr}(h^{\mathrm{T}} dL\rho + h^{\mathrm{T}}\rho dL). \quad (51)$$

Due to the fact that $\partial_{\mathbf{x}}\rho = (\rho L + L\rho)/2$, we have $\rho dL + (dL)\rho = 2d(\partial_{\mathbf{x}}\rho) - (d\rho)L - Ld\rho$, which means

$$dG_{\alpha\beta} = 2\mathrm{Tr}(2h^{\mathrm{T}} d(\partial_{\mathbf{x}}\rho)) - 2\mathrm{Tr}((Lh^{\mathrm{T}} + h^{\mathrm{T}}L)d\rho). \quad (52)$$

Next, since $G = G(\rho, \partial_{\mathbf{x}}\rho)$, $dG_{\alpha\beta}$ can also be expressed by

$$dG_{\alpha\beta} = 2\mathrm{Tr}\left(\left(\frac{\partial G_{\alpha\beta}}{\partial \rho}\right)^{\mathrm{T}} d\rho + \left(\frac{\partial G_{\alpha\beta}}{\partial(\partial_{\mathbf{x}}\rho)}\right)^{\mathrm{T}} d(\partial_{\mathbf{x}}\rho)\right). \quad (53)$$

This equation is derived through a similar calculation procedure for Eq. (49). Comparing this equation with Eq. (52), one can see that

$$\frac{\partial G_{\alpha\beta}}{\partial \rho} = 2h, \quad (54)$$

$$\frac{\partial G_{\alpha\beta}}{\partial(\partial_{\mathbf{x}}\rho)} = -hL^{\mathrm{T}} - L^{\mathrm{T}}h. \quad (55)$$

With these expressions, $\partial G/\partial \rho$ and $\partial G/\partial(\partial_{\mathbf{x}}\rho)$ can be obtained correspondingly. In this way, the entire path from $f$ to $L$ is connected. Together with the other two paths, AD can be fully applied in our case. The performance of computing time and memory allocation for the calculation of the gradient of QFI between these two realization methods of AD are compared with different dimensional density matrices. The dimension is denoted by $N$. As shown in the upper table in Table II, the computing time and memory allocation of the second method are better than the first one except for the case of $N = 2$, and this advantage becomes very significant when $N$ is large. Moreover, the computing time and memory allocation of the first method grow fast with the increase of dimension, which is reasonable as the calculations, especially the diagonalization, in the first method are performed in the $N^2$-dimensional space. There is no data of the first method when $N$ is larger than 7 as the memory occupation has exceeded our computer's memory. From this comparison, one can see that the second method performs better than the first one in basically all aspects and hence is chosen as the default auto-GRAPE method in QuanEstimation.

*Example.* Consider the dynamics in Eq. (12) and control Hamiltonian in Eq. (45). Now define

$$\delta_{\mathrm{c}}\omega := 1/\sqrt{\mathcal{I}_{\omega\omega}}, \quad (56)$$

$$\delta_{\mathrm{q}}\omega := 1/\sqrt{\mathcal{F}_{\omega\omega}} \quad (57)$$

as the theoretical optimal deviations with and without fixed measurement. The corresponding performance of controls generated via GRAPE and auto-GRAPE are shown in Figs. 7(a) and 7(b), which are obtained by 300 episodes in general. In QuanEstimation, the number of episodes can be set via the variable *max_episode = 300* in **kwargs in Table I. As shown in these plots, the values of $\sqrt{\omega_{\mathrm{tr}}T}\delta_{\mathrm{q}}\omega$ in (a)

TABLE II. Upper table: Comparison of the average computing time and memory allocation for the calculation of the gradient of QFI between two realization methods of AD. M1 and M2 represent the first and second methods. $N$ is the dimension of the density matrix. The density matrix and its derivative are generated randomly in the test. Lower table: Comparison of the average computing time per episode between GRAPE and auto-GRAPE with different target time $T$. Parallel computing is not applied here. KB, MB, and GB represent Kilobyte, Megabyte, and Gigabyte, respectively.

| | M1 | | M2 | |
| --- | --- | --- | --- | --- |
| $N$ | Computing time | Memory allocation | Computing time | Memory allocation |
| 2 | 4.46 $\mu$s | 2.99 KB | 5.14 $\mu$s | 2.24 KB |
| $2^2$ | 18.09 $\mu$s | 17.01 KB | 11.17 $\mu$s | 5.46 KB |
| $2^3$ | 257.65 $\mu$s | 217.63 KB | 35.84 $\mu$s | 18.79 KB |
| $2^4$ | 4.55 ms | 3.34 MB | 151.51 $\mu$s | 90.18 KB |
| $2^5$ | 174.61 ms | 53.01 MB | 962.17 $\mu$s | 501.85 KB |
| $2^6$ | 9.45 s | 846.18 MB | 11.05 ms | 3.31 MB |
| $2^7$ | 6151.51 s | 137.95 GB | 45.70 ms | 230.98 MB |
| $2^8$ | | | 347.50 ms | 1.73 GB |
| $2^9$ | | | 3.29 s | 13.36 GB |
| $2^{10}$ | | | 41.51 s | 105.08 GB |

| $\omega T$ | 5 | 10 | 15 | 20 | 30 | 40 |
| --- | --- | --- | --- | --- | --- | --- |
| GRAPE | 5.23 s | 21.75 s | 44.95 s | 71.00 s | 178.56 s | 373.89 s |
| auto-GRAPE | 0.32 s | 0.77 s | 1.45 s | 2.19 s | 4.14 s | 7.00 s |

and $\sqrt{\omega_{tr}T}\delta_c\omega$ in (b) obtained via GRAPE (red pentagrams) and auto-GRAPE (blue circles) basically coincide with each other, which is reasonable as they are intrinsically the same algorithm, just with different gradient calculation methods. However, auto-GRAPE shows a significant improvement on the computing time consumption, as given in the lower table in Table II, especially for a large target time $T$. The growth of average computing time per episode with the increase of $T$ in auto-GRAPE is quite insignificant compared to that in GRAPE. Adam can be applied by setting *Adam = True* in *\*\*kwargs*. For the sake of a good performance, one can set appropriate Adam parameters in *\*\*kwargs*, including the learning rate *epsilon*, the exponential decay rate for the first (second) moment estimates *beta1* (*beta2*). The default values of these parameters in the package are 0.01 and 0.90 (0.99). If *Adam = False*, the controls are updated with the constant step *epsilon*. Due to the convergence problem of Adam in some cases, several points in the figure are obtained by a second running of the code with a constant step, which takes the optimal control obtained in the first round (with Adam) as the initial guess.

In some scenarios, the time resolution of the control amplitude could be limited if the dynamics is too fast or the target time is too short. Hence, in the numerical optimization in such cases, the time steps of control cannot equal to that of the dynamics. Here we use the total control amplitude number $N_c = T/\Delta t_c$ with $\Delta t_c$ the control time step, to represent the time resolution of the control and we assume $\Delta t_c$ is fixed in the
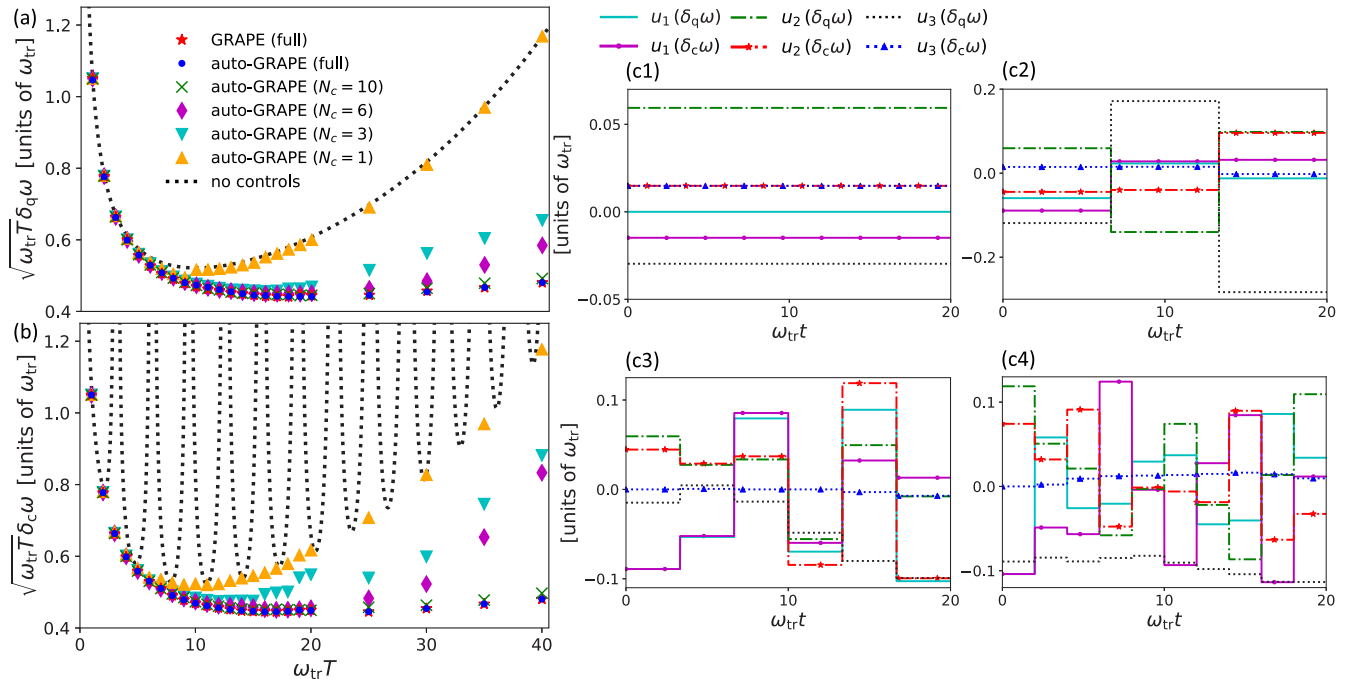


FIG. 7. The performance of control-enhanced (a) $\sqrt{\omega_{tr}T}\delta_q\omega$ and (b) $\sqrt{\omega_{tr}T}\delta_c\omega$ with different $N_c$. The optimal controls are generated via GRAPE and auto-GRAPE with the dynamics in Eq. (12) and control Hamiltonian in Eq. (45). The dotted-black lines in (a) and (b) represent $\sqrt{\omega_{tr}T}\delta_q\omega$ and $\sqrt{\omega_{tr}T}\delta_c\omega$ without control. The red pentagrams are those obtained via GRAPE with a full $N_c$, i.e., $N_c$ equals to the number of time steps. The blue circles, green crosses, purple diamonds, cyan-downward triangles, and orange-upward triangles represent those obtained via auto-GRAPE with $N_c$ being full, 10, 6, 3, and 1, respectively. Other parameters are set to be the same with those in Fig. 2. [(c1)–(c4)] The optimal controls in the case of $\omega_{tr}T = 20$ with $N_c$ being (c1) 1, (c2) 3, (c3) 6, and (c4) 10, respectively. The true value $\omega_{tr}$ is set to be 1. Planck units are applied here.
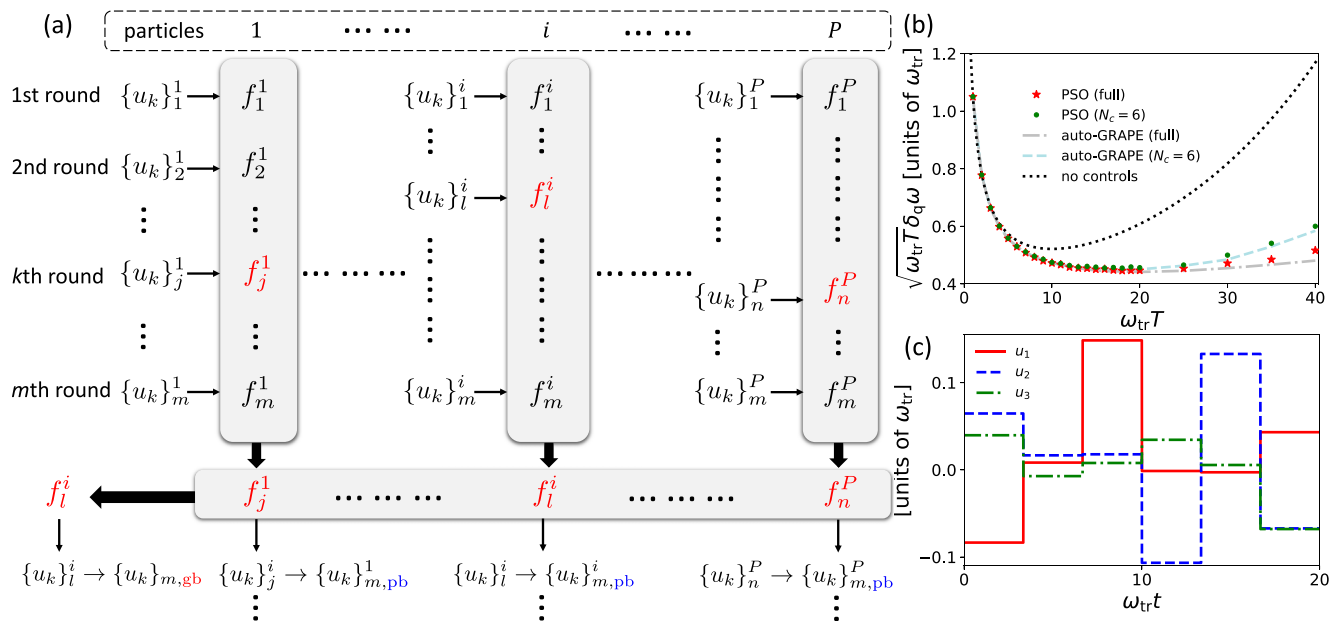
FIG. 8. (a) Illustration of the basic operation of PSO in $m$th round of episode. The personal best (with the blue subscript pb) for each particle in this round is obtained by comparing all the values of $f$ of this particle in all previous rounds including the current one. The global best (with the red subscript gb) is obtained by comparing the values of $f$ of all personal bests in this round. The light-gray areas represent the process of comparison, which takes the values of $\{u_k\}$ with respect to the maximum value of $f$. (b) The control-enhanced values of $\sqrt{\omega_{tr}T}\delta_q\omega$ with a full $N_c$ (red pentagrams) and $N_c = 6$ (green circles), where the controls are generated via PSO. (c) The optimal controls for $N_c = 6$ in the case of $\omega_{tr}T = 20$. The true value $\omega_{tr}$ is set to be 1. Planck units are applied here.

dynamics. A full $N_c$ in Figs. 7(a) and 7(b) means $\Delta t_c$ equals to the dynamical time step $\Delta t$. In the numerical calculation, it is possible that quotient of $\Delta t_c$ by $\Delta t$ is not an integer, indicating that the existing time of all control amplitudes cannot be equivalent. To avoid this problem, in QuanEstimation the input number ($N_t$) of dynamical time steps is automatically adjusted to $kN_c$ with $k$ the smallest integer to let $kN_c > N_t$, if it is not already an integer multiple of $N_c$. For example, if $N_c = 3$ and $N_t = 100$, then $N_t$ is adjusted to 102. Notice that in the package GRAPE is not available to deal with a nonfull $N_c$ scenario for a technical reason. If GRAPE is invoked in this case, it would automatically go back to auto-GRAPE. As a matter of fact, auto-GRAPE outperforms GRAPE in most aspects, therefore, we strongly suggest the users choose auto-GRAPE, instead of GRAPE, in practice.

The performance of controls with limited $N_c$ is also demonstrated in Figs. 7(a) and 7(b) with the dynamics in Eq. (12) and control Hamiltonian in Eq. (45). It can be seen that the constant-value controls ($N_c = 1$, orange-upward triangles) cannot reduce the values of $\delta_c\omega$ and $\delta_q\omega$. In the case of fixed measurement it can only suppress the oscillation of $\delta_c\omega$. The performance improves with the increase of $N_c$ and when $N_c = 10$, the values of $\delta_q\omega$ and $\delta_c\omega$ are very close to those with a full $N_c$. This fact indicates that inputting 10 control amplitudes is good enough in this case and a full $N_c$ control is unnecessary. A limited $N_c$ here could be easier to realize in practice and hence benefit the experimental realization.

### C. Particle swarm optimization

Particle swarm optimization (PSO) is a well-used gradient-free method in optimizations [67,68], and has been applied in the detection of gravitational waves [69], the characterization of open systems [70], the prediction of crystal structure [71], and in quantum metrology it has been used to generate adaptive measurement schemes in phase estimations [72,73].

A typical version of PSO includes a certain number (denoted by $P$) of parallel particles. In quantum control, these particles are just $P$ sets of controls $\{u_k\}$ labeled by $\{u_k\}^i$ for $i = 1, \ldots, P$. The value of $\{u_k\}$ of $i$th particle in $m$th round of episode is further denoted by $\{u_k\}_m^i$. The basic optimization philosophy of PSO is given in Fig. 8(a) and the pseudocode is given in Algorithm 3. In the pseudocode, $\{u_k\}_{0,\text{pb}}^i$ and $f(\{u_k\}_{0,\text{pb}}^i)$ are just formal notations representing the initialization of the personal bests. There exist two basic concepts in PSO, the personal best and global best. In the $m$th round of episode, the personal best of $i$th particle ($\{u_k\}_{m,\text{pb}}^i$) is assigned by the $\{u_k\}$ with respect to the maximum value of $f$ among all previous episodes of this particle, namely,

$$\{u_k\}_{m,\text{pb}}^i = \arg\left(\max_{n\in[1,m]} f\left(\{u_k\}_n^i\right)\right) \quad (58)$$

with $\arg(\cdot)$ the argument. For example, as illustrated in Fig. 8, if $f_j^1$ is the maximum in $\{f_1^1, f_2^1, \ldots, f_m^1\}$, then $\{u_k\}_{m,\text{pb}}^1$ is assigned by $\{u_k\}_j^1$. Once the personal bests are obtained for all particles, the global best is assigned by the $\{u_k\}$ with respect

**Algorithm 3:** PSO

Initialize the control $\{u_k\}_1^i$ for each $i \in [1, P]$;
Initialize $\{\delta u_k\}_1^i = 0$ for each $i \in [1, P]$;
Assign $f(\{u_k\}_{0,\mathrm{pb}}^i) = 0$ for each $i \in [1, P]$;
**for** $m = 1, M$ **do**
    **for** $i = 1, P$ **do**
        Receive the control $\{u_k\}_m^i$;
        Evolve the state with $\{u_k\}_m^i$ and calculate the objective function $f(\{u_k\}_m^i)$ at the target time $T$;
        Compare $f(\{u_k\}_m^i)$ with value of the personal best in last episode $f(\{u_k\}_{m-1,\mathrm{pb}}^i)$ and assign the new personal
          best $\{u_k\}_{m,\mathrm{pb}}^i = \arg\left(\max\left\{f(\{u_k\}_{m-1,\mathrm{pb}}^i), f(\{u_k\}_m^i)\right\}\right)$;
    **end for**

    Compare all $f(\{u_k\}_{m,\mathrm{pb}}^i)$ with $i \in [1, P]$ and assign the global best $\{u_k\}_{m,\mathrm{gb}} = \arg\left(\max_{i \in [1,P]} f(\{u_k\}_{m,\mathrm{pb}}^i)\right)$;
    **for** $i = 1, P$ **do**
        Calculate $\{\delta u_k\}_m^i = c_0 \{\delta u_k\}_{m-1}^i + \mathrm{rand}() \cdot c_1 (\{u_k\}_{m,\mathrm{pb}}^i - \{u_k\}_m^i) + \mathrm{rand}() \cdot c_2 (\{u_k\}_{m,\mathrm{gb}} - \{u_k\}_m^i)$;
        Update the control $\{u_k\}_{m+1}^i = \{u_k\}_m^i + \{\delta u_k\}_m^i$.
    **end for**
**end for**
Save the global best $\{u_k\}_{M,\mathrm{gb}}$ and corresponding $f$.

to the maximum value of $f$ among all personal bests, i.e.,

$$\{u_k\}_{m,\mathrm{gb}} = \arg\left(\max_{i \in [1,P]} f(\{u_k\}_{m,\mathrm{pb}}^i)\right). \quad (59)$$

With all personal bests and the global best, the velocity $\{\delta u_k\}_m^i$ for the $i$th particle is calculated by

$$\{\delta u_k\}_m^i = c_0 \{\delta u_k\}_{m-1}^i + \mathrm{rand}() \cdot c_1 (\{u_k\}_{m,\mathrm{pb}}^i - \{u_k\}_m^i)$$
$$+ \mathrm{rand}() \cdot c_2 (\{u_k\}_{m,\mathrm{gb}} - \{u_k\}_m^i), \quad (60)$$

where rand() represents a random number within [0,1] and $c_0$, $c_1$, $c_2$ are three positive constant numbers. In the package, these parameters can be adjusted in **kwargs*, shown in Table I, via the variables *c0*, *c1*, and *c2*. A typical choice for these constants is $c_0 = 1$, $c_1 = c_2 = 2$, which are also the default values in the package. *max_episode* in **kwargs* represents the episode number to run. If it is only set to be a number, for example *max_episode = 1000*, the program will continuously run 1000 episodes. However, if it is a list, for example *max_episode = [1000,100]*, the program will also run 1000 episodes in total but replace $\{u_k\}$ of all particles with the current global best every 100 episodes. *p_num* represents the particle number and is set to be 10 in default. The initial guesses of control can be input via *ctrl0* and the default choice *ctrl0 = []* means all the guesses are randomly generated. In the case that the number of input guessed controls is less than the particle number, the algorithm will generate the remaining ones randomly. On the other hand, if the number is larger than the particle number, only the suitable number of controls will be used. The optimization result can be realized repeatedly by fixing the value of the variable *seed*, and its default value is 1234 in the package.

*Example.* Here we also illustrate the performance of controls generated via PSO with the dynamics in Eq. (12) and control Hamiltonian in Eq. (45). $\delta_q \omega$ is defined in Eq. (57). The performance of controls with a full $N_c$ (red pentagrams) and $N_c = 6$ (green circles) are shown in Fig. 8(b), and the cor-

responding optimal controls for $N_c = 6$ are given in Fig. 8(c). Compared to the result obtained via auto-GRAPE (dash-dotted-gray line for a full $N_c$ and dashed-light-blue line for $N_c = 6$), the performance of PSO is worse than that of auto-GRAPE, especially in the case of a large target time $T$ with a full $N_c$. This is due to the fact that the search space is too large for PSO in such cases as the time step $\Delta t$ is fixed in the calculation and a larger $T$ means a larger value of $N_c$. For example, in the case of $\omega_{\mathrm{tr}} T = 40$ and $N_c = 10\,000$, the total parameter number in the optimization is 30 000. PSO can provide a good performance when the dimension of the search space is limited. In the case of $N_c = 6$, the result of PSO basically coincides with that of auto-GRAPE. Hence, for those large systems that the calculation of gradient is too time-consuming or the search space is limited, the gradient-free methods like PSO would show their powers.

### D. Differential evolution

Differential evolution (DE) is another useful gradient-free algorithm in optimizations [74]. It has been used to design adaptive measurements in quantum phase estimation [75,76], high-quality control pulses in quantum information [77,78], and help to improve the learning performance in quantum open systems [79,80]. Different with PSO, DE would not converge prematurely in general and its diversification is also better since the best solution does not affect other solutions in the population [81].

A typical DE includes a certain number (denoted by $P$) of $\{u_k\}^i$, which is usually referred to as the populations in the language of DE. In QuanEstimation, the population number and the guessed controls can be set via the variables *p_num* and *ctrl0* in **kwargs*, as shown in Table I. The rule for the usage of *ctrl0* here is the same as *ctrl0* in PSO. After the initialization of all $\{u_k\}^i$ ($i \in [1, P]$), two important processes in DE, mutation and crossover, are performed, as illustrated in Fig. 9(a) with the pseudocode in Algorithm 4. In the step
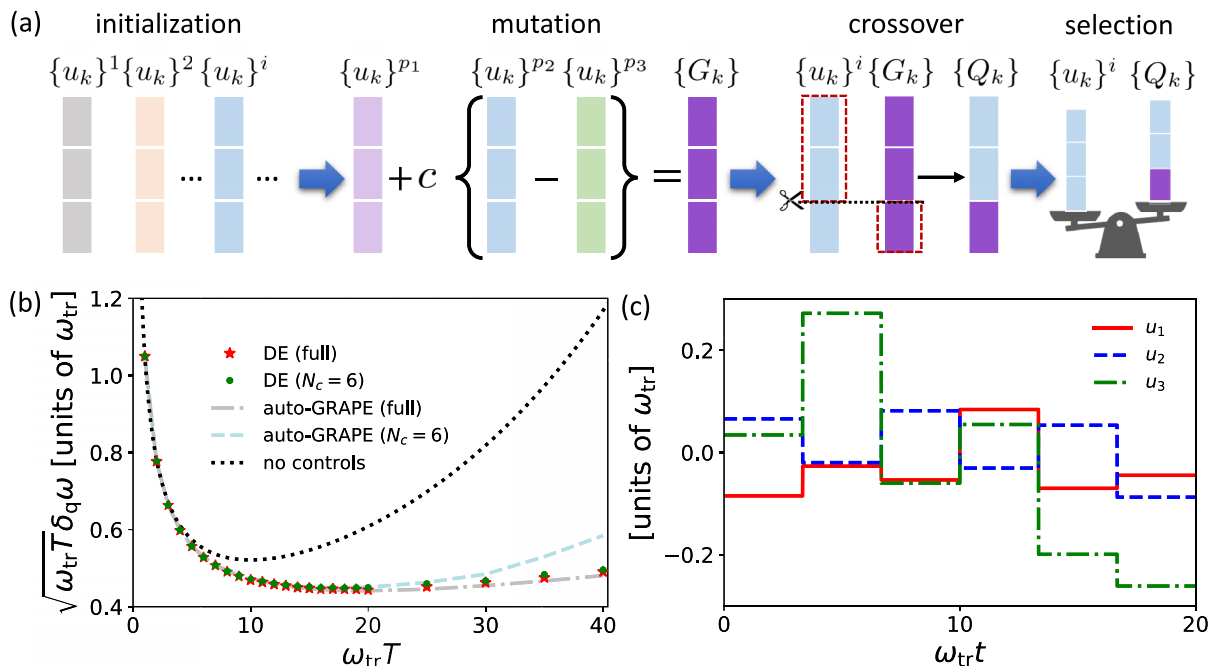
FIG. 9. (a) Illustration of the optimization process of DE, which includes four steps: initialization, mutation, crossover, and selection. (b) The control-enhanced values of $\sqrt{\omega_{tr}T}\delta_q\omega$ with a full $N_c$ (red pentagrams) and $N_c = 6$ (green circles), where the controls are generated via DE. (c) The optimal controls for $N_c = 6$ in the case of $\omega_{tr}T = 20$. The true value $\omega_{tr}$ is set to be 1. Planck units are applied here.

of mutation, three populations $\{u_k\}^{p_1}$, $\{u_k\}^{p_2}$ and $\{u_k\}^{p_3}$ are randomly picked from all $\{u_k\}^i$, and used to generate a new population $\{G_k\}$ via the equation

$$\{G_k\} = \{u_k\}^{p_1} + c(\{u_k\}^{p_2} - \{u_k\}^{p_3}) \tag{61}$$

with $c \in [0, 1]$ (or [0,2]) a constant number. The next step is the crossover. At the beginning of this step, a random integer $a$ is generated in the regime $[1, N_c]$, which is used to make sure

the crossover happens definitely. Then another new population $\{Q_k\}$ is generated for each $\{u_k\}^i$ utilizing $\{G_k\}$. Now we take $j$th entry of $Q_k$ ($[Q_k]_j$) as an example to show the generation rule. In the first, a random number $r$ is picked in the regime $[0,1]$. Then $[Q_k]_j$ is assigned via the equation

$$[Q_k]_j = \begin{cases} [G_k]_j, & \text{if } r \leqslant c_r \text{ or } j = a, \\ [u_k]_j, & \text{if } r > c_r \text{ and } j \neq a, \end{cases} \tag{62}$$

**Algorithm 4:** DE

---

Initialize the control $\{u_k\}^i$ for $i \in [1, P]$;
Evolve the state with $\{u_k\}^i$ and calculate the objective function $f(\{u_k\}^i)$ at the target time $T$ for $i \in [1, P]$;
**for** episode=1, $M$ **do**
    **for** $i = 1, P$ **do**
        Randomly generate three integers $p_1, p_2, p_3$ in the regime $[1, P]$;
        Generate $\{G_k\}$ via the equation $\{G_k\} = \{u_k\}^{p_1} + c(\{u_k\}^{p_2} - \{u_k\}^{p_3})$;
        **for** $k = 1, K$ **do**
            Generate a random integer $a \in [1, N_c]$;
            **for** $j = 1, N_c$ **do**
                Generate a random number $r \in [0, 1]$ and assign $[Q_k]_j = \begin{cases} [G_k]_j, & \text{if } r \leq c_r \text{ or } j = a, \\ [u_k]_j, & \text{if } r > c_r \text{ and } j \neq a; \end{cases}$
            **end for**
        **end for**
        Evolve the state with the control $\{Q_k\}$ and calculate $f(\{Q_k\})$ at time $T$;
        **if** $f(\{u_k\}^i) < f(\{Q_k\})$ **then**
            Replace $\{u_k\}^i$ with $\{Q_k\}$.
        **end if**
    **end for**
**end for**
Compare all $f(\{u_k\}^i)$ and save $\{u_k\}^i$ (and corresponding $f$) with respect to the largest $f$.

---

where $[G_k]_j$ is the $j$th entry of $G_k$ and $[u_k]_j$ is the $j$th entry of a $u_k$ in $\{u_k\}^i$. This equation means if $r$ is no larger than a given constant $c_r$ (usually called crossover constant in DE), then assign $[G_k]_j$ to $[Q_k]_j$, otherwise assign $[u_k]_j$ to $[Q_k]_j$. In the meantime, the $a$th entry of $Q_k$ always takes the value of $[G_k]_j$ regardless the value of $r$ to make sure at least one point mutates. After the crossover, the values of objective functions $f(\{u_k\}^i)$ and $f(\{Q_k\})$ are compared, and $\{u_k\}^i$ is replaced by $\{Q_k\}$ if $f(\{Q_k\})$ is larger. In the package, $c$ and $c_r$ can be adjusted via the variables c and cr in **kwargs, and the default values are 1.0 and 0.5.

*Example.* The performance of controls generated via DE is also illustrated with the dynamics in Eq. (12) and control Hamiltonian in Eq. (45). $\delta_q\omega$ is defined in Eq. (57). As shown in Fig. 9(b), different with PSO, the performance of DE with a full $N_c$ (red pentagrams) is very close to that of auto-GRAPE (dash-dotted-gray line), even for a large target time $T$, which indicates that DE works better than PSO in this example. More surprisingly, in the case of $N_c = 6$, DE (green circles) not only outperforms PSO, but also significantly outperforms auto-GRAPE (dashed-light-blue line). This result indicates that no algorithm has the absolute advantage in general. Comparison and combination of different algorithms are a better approach to design optimal controls in quantum metrology, which can be conveniently finished via QuanEstimation. The optimal controls obtained via DE for $N_c = 6$ are given in Fig. 9(c) in the case of $\omega_{tr}T = 20$. The results above are obtained with 1000 episodes, which can be adjusted via *max_episode = 1000* in **kwargs.

### E. Deep Deterministic Policy Gradients

Deep deterministic policy gradients (DDPG) is a powerful tool in machine learning [82] and has already been applied in quantum physics to perform quantum multiparameter estimation [61] and enhance the generation of spin squeezing [83]. The pseudocode of DDPG for quantum estimation and the corresponding flow chart can be found in Ref. [17], and the details will not be repeatedly addressed herein.

*Example.* The performance of controls generated via DDPG in the case of single-parameter estimation is also illustrated with the dynamics in Eq. (12) and control Hamiltonian in Eq. (45), as shown in Fig. 10(a). $\delta_q\omega$ is defined in Eq. (57). The reward is taken as the logarithm of the ratio between the controlled and noncontrolled values of the QFI at time $t$. It can be seen that the performance of DDPG with a full $N_c$ (red pentagrams) shows a significant disparity with that of auto-GRAPE (dash-dotted-gray line). A more surprising fact is that it is even worse than the performance of both auto-GRAPE (dashed-light-blue line) and DDPG (green circles) with $N_c = 6$. And the performance of DDPG with $N_c = 6$ also presents no advantage compared to PSO and DE. However, we cannot rashly say that PSO and DE outperform DDPG here as DDPG involves way more parameters and maybe a suitable set of parameters would let its performance comparable or even better than PSO and DE. Nevertheless, we can still safely to say that PSO and DE, especially DE, are easier to find optimal controls in this example and DDPG does not present a general advantage here. The total reward in the case of $\omega_{tr}T = 20$ with a full $N_c$ and $N_c = 6$ are given in Figs. 10(b) and 10(c),
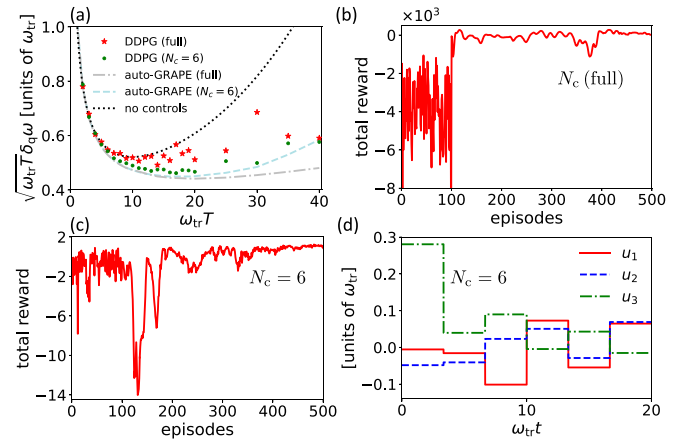


FIG. 10. (a) The control-enhanced values of $\sqrt{\omega_{tr}T}\delta_q\omega$ with a full $N_c$ (red pentagrams) and $N_c = 6$ (green circles), where the controls are generated via DDPG. [(b),(c)] The change of total reward in the episodes in the case of (b) a full $N_c$ and (c) $N_c = 6$. (d) The controls obtained via DDPG for $N_c = 6$ in the case of $\omega_{tr}T = 20$. The true value $\omega_{tr}$ is set to be 1. Planck units are applied here.

respectively. The total reward indeed increases and converges for a full $N_c$, but the final performance is only sightly better than the noncontrolled value [dotted-black line in Fig. 10(a)]. For $N_c = 6$, the total reward does not significantly increase, which means the corresponding performance of $\delta_q\omega$ basically comes from the average performance of random controls. The controls obtained via DDPG for $N_c = 6$ are shown in Fig. 10(d).

### F. Performance of the convergence speed

Apart from the improvement of the objective function, the convergence speed is also an important aspect of an algorithm to evaluate its performance. Here we illustrate the convergence performance of different algorithms in Fig. 11 in the single-parameter scenario discussed previously, namely, the dynamics in Eq. (12) and control Hamiltonian in Eq. (45) with a full $N_c$. As shown in Fig. 11(a), GRAPE (dashed-red line) and auto-GRAPE (dotted-black line) show higher convergence speed than PSO (solid-green line) and DE (dash-dotted-cyan line). This phenomenon coincides with the common understanding that the gradient-based methods converge faster than gradient-free methods in general. DE converges slower than GRAPE and auto-GRAPE, but the final performance of QFI basically coincides with them. PSO presents the slowest speed in this example and the final result of QFI is also worse than others. DDPG is not involved in this figure as its improvement on the QFI is not as significant as others.

The effect of Adam in auto-GRAPE is also illustrated in Fig. 11(b). Denote $\epsilon$ as the learning rate in Adam. In the case of constant-step update, auto-GRAPE with $\epsilon = 0.01$ (dotted-black line) converges faster than that with $\epsilon = 0.005$ (dash-dotted-green line), which is common and reasonable as a large step usually implies a higher convergence speed. However, when Adam is invoked, this difference becomes very insignificant and both lines (solid-gray line for $\epsilon = 0.01$ and dashed-blue line for $\epsilon = 0.005$) converge faster than constant-
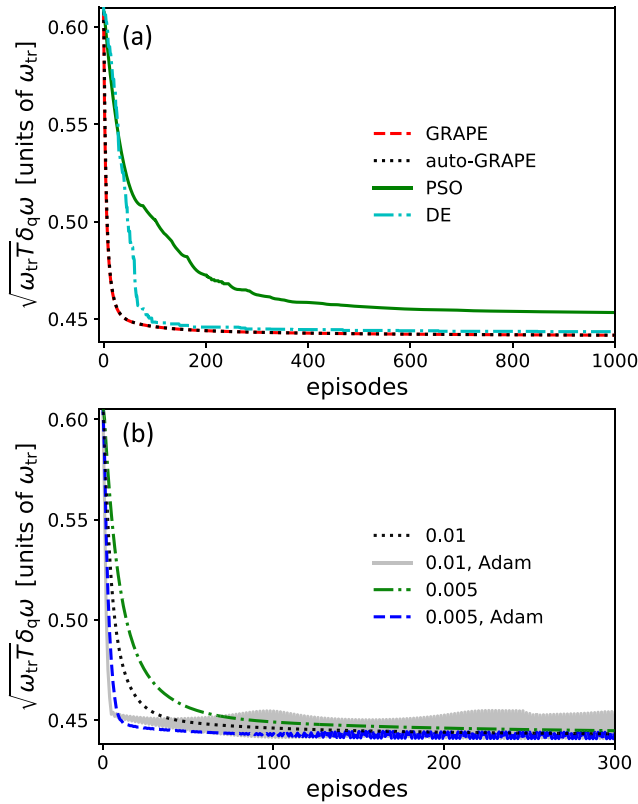
FIG. 11. (a) The convergence performance of different algorithms, including GRAPE (dashed-red line), auto-GRAPE (dotted-black line), PSO (solid-green line) and DE (dash-dotted-cyan line). (b) The convergence performance of auto-GRAPE with constant step $\epsilon = 0.01$ (dotted-black line), $\epsilon = 0.005$ (dash-dotted-green line), and with Adam (solid-gray line for $\epsilon = 0.01$ and dashed-blue line for $\epsilon = 0.005$). The target time $\omega_{tr}T = 20$, and the true value $\omega_{tr}$ is set to be 1. Planck units are applied here.

step updates. However, it should be noticed that a large $\epsilon$ in Adam may result in a strong oscillation of $\delta_q\omega$ in the episodes, and it should be adjusted to smaller values if one wants to avoid this phenomenon.

### G. Multiparameter estimation

Compared to the single-parameter estimation, multiparameter estimation is a more challenging problem in quantum metrology. In this case, $\text{Tr}(W\mathcal{F}^{-1})$ cannot be used as the objective function in the implementation of GRAPE as the analytical calculation of $\mathcal{F}^{-1}$ is very difficult, if not fully impossible, when the number of parameter is large. Hence, in GRAPE when $W = \mathbb{1}$, $\sum_a 1/\mathcal{F}_{aa}$, a lower bound of $\text{Tr}(\mathcal{F}^{-1})$, is taken as the superseded objective function [13,17,56]. Unfortunately, $\sum_a W_{aa}/\mathcal{F}_{aa}$ fails to be a valid lower bound for a general $W$. In this case, to keep $\sum_a W_{aa}/\mathcal{F}_{aa}$ a valid lower bound, the parameters for estimation have to be reorganized by the linear combination of the original ones to let $W$ be diagonal, which causes the inconvenience to implement GRAPE in such cases. Different with GRAPE, this problem naturally vanishes in auto-GRAPE as the inverse matrix $\mathcal{F}^{-1}$ is calculated automatically and so does the gradient. In the meantime,

PSO and DE would also not face such problems as they are gradient free.

*Example.* Here we take an electron-nuclear spin system, which can be readily realized in the nitrogen-vacancy centers, as an example to demonstrate and compare the performance of different algorithms included in QuanEstimation. The Hamiltonian of this system reads [84–86]

$$H_0/\hbar = DS_3^2 + g_S\vec{B}\cdot\vec{S} + g_I\vec{B}\cdot\vec{I} + \vec{S}^{\mathrm{T}}\mathcal{A}\vec{I}, \quad (63)$$

where $S_i = s_i \otimes \mathbb{1}$, and $I_i = \mathbb{1} \otimes \sigma_i$ ($i = 1, 2, 3$) represent the electron and nuclear ($^{15}$N) operators with $s_1$, $s_2$, and $s_3$ spin-1 operators. Their specific expressions are

$$s_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad s_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & -i & 0 \\ i & 0 & -i \\ 0 & i & 0 \end{pmatrix},$$

and $s_3 = \text{diag}(1, 0, -1)$. The vectors $\vec{S} = (S_1, S_2, S_3)^{\mathrm{T}}$, $\vec{I} = (I_1, I_2, I_3)^{\mathrm{T}}$ and $\mathcal{A}$ is the hyperfine tensor. In this case, $\mathcal{A} = \text{diag}(A_1, A_1, A_2)$ with $A_1$ and $A_2$ the axial and transverse magnetic hyperfine coupling coefficients. The hyperfine coupling between the magnetic field and electron are approximated to be isotopic. The coefficients $g_S = g_e\mu_B/\hbar$ and $g_I = g_n\mu_n/\hbar$. Here $g_e$ ($g_n$) is the g factor of the electron (nuclear), $\mu_B$ ($\mu_n$) is the Bohr (nuclear) magneton and $\hbar$ is the Plank's constant. The control Hamiltonian is

$$H_c/\hbar = \sum_{i=1}^{3}\Omega_i(t)S_i, \quad (64)$$

where $\Omega_i(t)$ is a time-varying Rabi frequency. In practice, the electron suffers from the noise of dephasing, which means the dynamics of the full system is described by the master equation

$$\partial_t\rho = -i[H_0 + H_c, \rho] + \frac{\gamma}{2}\left(S_3\rho S_3 - S_3^2\rho - \rho S_3^2\right), \quad (65)$$

with $\gamma$ the dephasing rate, which is usually inverse proportional to the dephasing time $T_2^*$.

Now we use this system as a controllable magnetometer to estimate the magnetic field $\vec{B}$, which is a three-parameter estimation problem. The optimal controls can be obtained via different algorithms. In this case, the initial state is taken as $(|1\rangle + |-1\rangle) \otimes |\uparrow\rangle/\sqrt{2}$, where $(|1\rangle + |-1\rangle)/\sqrt{2}$ is an electron state with $|1\rangle$ ($|-1\rangle$) the eigenstate of $s_3$ corresponding to the eigenvalue 1 ($-1$). $|\uparrow\rangle$ is a nuclear state and the eigenstate of $\sigma_3$ corresponding to the eigenvalue 1. $W$ is chosen to be $\mathbb{1}$. The systematic parameters are chosen as $D = 2\pi \times 2.87$ GHz, $g_S = 2\pi \times 28.03$ GHz/T, $g_I = 2\pi \times 4.32$ MHz/T, $A_1 = 2\pi \times 3.65$ MHz, $A_2 = 2\pi \times 3.03$ MHz, and the true values of $\vec{B}$ are $B_1 = B_2 = B_3 = 0.50$ mT. The dephasing rate $\gamma = 2\pi \times 1$ MHz. All the parameter values are selected according to Refs. [86,87].

The performance of controls given by different algorithms is given in Fig. 12. The control amplitude is limited in the regime $[-20 \text{ MHz}, 20 \text{ MHz}]$. In the case of a full $N_c$ $[N_c = 2000T/(0.01\,\mu\text{s})]$, as shown in Fig. 12(a), the performance of GRAPE (red pentagrams), auto-GRAPE (cyan triangles), PSO (blue crosses), DE (yellow circles) and DDPG (orange pluses) basically coincide for small target time ($T \leqslant 0.01\,\mu\text{s}$), and the reduction of $\text{Tr}(W\mathcal{F}^{-1})$ is limited
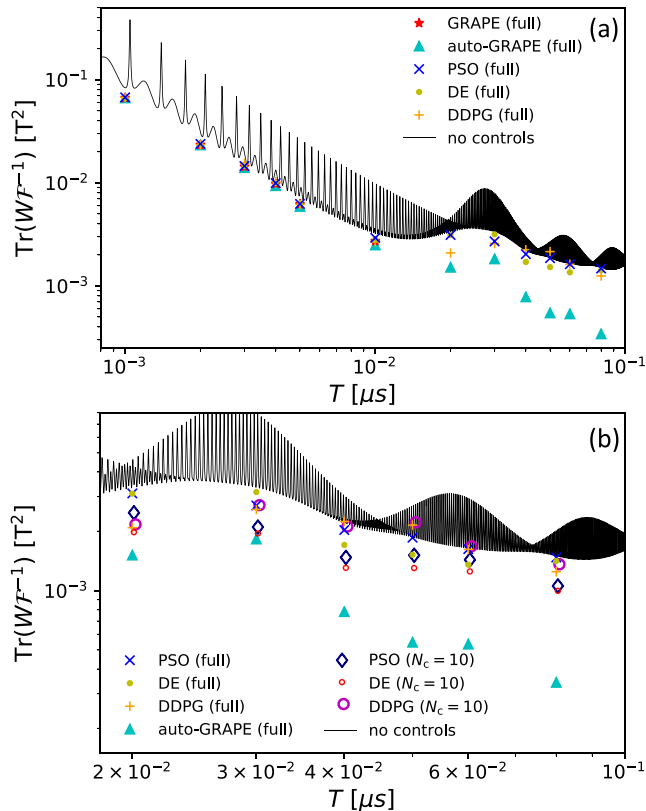
FIG. 12. (a) The performance of controls generated via different algorithms, including GRAPE (red pentagrams), auto-GRAPE (cyan triangles) with full $N_c$, PSO (blue crosses) with full $N_c$, DE (yellow circles) with full $N_c$ and DDPG (orange pluses) with full $N_c$. (b) The performance of controls generated via PSO (dark-blue diamonds), DE (small-red-hollow circles) and DDPG (large-purple-hollow circles) with limited $N_c$ ($N_c = 10$). $W$ is chosen to be $\mathbb{1}$.

compared to the noncontrolled values (solid-black line). In the regime of large target time ($T > 0.01\,\mu$s), auto-GRAPE shows the best performance. GRAPE is not applied in these points as its time consumption is too heavy for our computers. PSO and DE only find controls that provide slightly enhancement on $\mathrm{Tr}(W\mathcal{F}^{-1})$ in this regime. The different behaviors of the performance are due to the large search space in this case. For example, the total control number for $T = 0.08\,\mu$s is 48 000 including all three controls $\Omega_1$, $\Omega_2$, and $\Omega_3$. In such a large parameter space, different with the gradient-based methods, the gradient-free methods cannot promise to find optimal values. Hence, the gradient-based methods would be a good choice in such cases. However, one should notice that the gradient-based methods like auto-GRAPE could be more memory consuming than gradient-based methods. In the case that the computer memory is limited, one may have to choose gradient-free methods.

In the case of a small search space, for example $N_c = 10$, the performance of PSO and DE improve significantly, as shown in Fig. 12(b). Both PSO (dark-blue diamonds) and DE (smal-red-hollow circles) with $N_c = 10$ outperform the full $N_c$ cases, yet DDPG with $N_c = 10$ (large-purple-hollow circles) does not show this behavior. Similar to the single-

parameter scenario, DE provides a better performance than PSO and DDPG when the control number $N_c$ is limited. A more interesting fact is that for some target time, like $T = 0.03\,\mu$s, PSO and DE even provide comparable performance with auto-GRAPE with a full $N_c$, indicating that the control schemes given by PSO and DE in this case not only meet the best precision limit, but are also simpler toimplement in experiments than the full-$N_c$ scheme given by auto-GRAPE.

### H. Minimum parametrization time optimization

The control optimizations discussed in the previous subsections are performed with a fixed target time $T$. In some scenarios, the goal is not to achieve the highest precision within a fixed time, but to reach a given precision as soon as possible. This problem requires the search of minimum time to reach a given value of the objective function, which can be realized in QuanEstimation in the class *ControlOpt()*. After the callings of *control = ControlOpt()* and *control.dynamics()*, one can use the following code to solve this problem:

```
control.mintime(f, W = [], M = [], method = "binary",
                 target = "QFIM", LDtype = "SLD")
```

Here the input $f$ is a float number representing the given value of the objective function. The type of objective function can be adjusted via *target = " "*, which includes three options *"QFIM"* (default), *"CFIM"*, and *"HCRB"*. The measurement can be input via $M = []$ if necessary, and in this case the objective function will be chosen as the CFIM regardless of the setting in *target = " "*. In the case of *target = "QFIM"*, the type of QFIM can be changed via *LDtype = " "*. The choices include *"SLD"*, *"RLD"*, and *"LLD"*. *method = "binary"* represents the binary search (logarithmic search) and *method = "forward"* represents the forward search from the beginning of time. Choosing a suitable method may help to improve the calculation efficiency. For example, if the users already know that the minimum time is very small compared to $T$, the forward search would be more efficient than the binary search. Notice that the search is restricted in the regime $[0, T]$ where $T$ is given by the array *tspan* input in *ControlOpt()*, and the current code can only deal with full-$N_c$ controls. The outputs are two files "mtspan.csv" and "controls.csv" containing the array of time from the start to the searched minimum time and the corresponding length of optimal controls, respectively.

## VII. STATE OPTIMIZATION

Quantum resources like entanglement and squeezing are key in quantum parameter estimation to demonstrate a quantum enhanced precision. In contrast to the dynamical resources like time or control, entanglement and squeezing are usually embedded in the probe state, indicating that different probe states would present dramatically different performance on the precision. The search of optimal probe states is thus an essential step in the design of optimal schemes. Various methodologies, including direct analytical calculations [88–101], semianalytical [102–110] and full numerical approaches [111–115], have been proposed and discussed. More advances of the state optimization in quantum metrology can

TABLE III. Available methods for state optimization in QuanEstimation and corresponding default parameter settings. Notice that AD is not available when the HCRB are taken as the objective function.

| Algorithms | method= | **kwargs and default values | |
| --- | --- | --- | --- |
| AD | "AD" | "Adam" | False |
| | | "psi0" | [] |
| | | "max_episode" | 300 |
| | | "epsilon" | 0.01 |
| | | "beta1" | 0.90 |
| | | "beta2" | 0.99 |
| PSO | "PSO" | "p_num" | 10 |
| | | "psi0" | [] |
| | | "max_episode" | [1000,100] |
| | | "c0" | 1.0 |
| | | "c1" | 2.0 |
| | | "c2" | 2.0 |
| | | "seed" | 1234 |
| DE | "DE" | "p_num" | 10 |
| | | "psi0" | [] |
| | | "max_episode" | 1000 |
| | | "c" | 1.0 |
| | | "cr" | 0.5 |
| | | "seed" | 1234 |
| NM | "NM" | "p_num" | 10 |
| | | "psi0" | [] |
| | | "max_episode" | 1000 |
| | | "ar" | 1.0 |
| | | "ae" | 2.0 |
| | | "ac" | 0.5 |
| | | "as0" | 0.5 |
| | | "seed" | 1234 |
| RI | "RI" | "psi0" | [] |
| | | "max_episode" | 300 |
| | | "seed" | 1234 |
| DDPG | "DDPG" | "psi0" | [] |
| | | "max_episode" | 500 |
| | | "layer_num" | 3 |
| | | "layer_dim" | 200 |
| | | "seed" | 1234 |

be found in a recent review [17]. QuanEstimation includes the process of state optimization with various methods, including both gradient-based and gradient-free methods. The specific code in QuanEstimation for the execution of state optimization are as follows:

```
state = StateOpt(savefile=False, method="AD",
            **kwargs)
state.dynamics(tspan, H0, dH, Hc=[], ctrl=[],
            decay=[], dyn_method="expm")
state.QFIM(W=[], LDtype="SLD")
state.CFIM(M=[], W=[])
```

In the case that the parametrization is described by the Kraus operators, replace *state.dynamics()* with the code *state.Kraus(K,dK)*. The parameters above have already been introduced previously. The default settings *W = []* and *M = []* means $W = \mathbb{1}$ and the measurement is a SIC-POVM. The optimization method can be adjusted via *method = " "* and corresponding parameters can be set via **kwargs*. The

available optimization methods and corresponding default parameter settings are given in Table III. The dynamics can also be solved with ODE by setting *dyn_method = "ode"*. One exception is that when *method = "AD"* is applied. In this case ODE is not available for now. Two files "f.csv" and "states.csv" will be generated at the end of the program, which include the values of the objective function in all episodes and the final obtained optimal probe state. When *savefile = True*, the states obtained in all episodes will be saved in "states.csv". In the multiparameter estimation, the HCRB can also be chosen as the objective function by calling the code

```
state.HCRB(W=[])
```

Notice that if *method = "AD"*, *state.HCRB()* is not available. Similar to the control optimization, if the users invoke *state.HCRB()* in the single-parameter scenario, a warning will arise to remind them to call *state.QFIM()* instead.

---

**Algorithm 5:** AD for pure states

---

Receive the guessed probe state $\rho_0 = |\psi_{in}\rangle\langle\psi_{in}|$;

**for** episode=1, $M$ **do**

    Calculate the density matrix $\rho_T$ and its derivative $\partial_\mathbf{x}\rho_T$ at the target time $T$;

    Calculate the objective function $f(T)$ with $\rho_T$ and $\partial_\mathbf{x}\rho_T$;

    Calculate the gradients $\left\{\frac{\delta f(T)}{\delta c_i}\right\}$ with the automatic differentiation;

    Update the coefficients $\{c_i\} \leftarrow \{c_i\} + \epsilon\left\{\frac{\delta f(T)}{\delta c_i}\right\}$;

    Normalize the coefficients $\{c_i\} \leftarrow \frac{1}{\sqrt{\sum_j |c_j|^2}}\{c_i\}$.

**end for**

Save the coefficients and corresponding $f(T)$, and reconstruct the state.

---

In the previous section, we already showed the power of automatic differentiation (AD) in the construction of auto-GRAPE. Similarly, it can also be used in the state optimization. Due to the convexity of the QFI and QFIM [11,13], the optimal probe states are pure states in most scenarios. Hence, we first consider the state optimization within the set of pure states. The pseudocode of AD in state optimization for pure states is given in Algorithm 5. In a specific basis $\{|i\rangle\langle i|\}$, a probe state could be expanded by $|\psi\rangle = \sum_i c_i|i\rangle$, and the search of optimal probe states is equivalent to the search of a set of normalized complex coefficients $\{c_i\}$. In AD, a guessed probe state is first given or generated and evolved to the target time $T$ according to the given dynamics, during which the density matrices and corresponding derivatives with respect to $\mathbf{x}$ are calculated and saved. Then after calculating the objective function $f(T)$ at time $T$, all gradients $\{\delta f(T)/\delta c_i\}$ are evaluated via the automatic differentiation, and the coefficients $\{c_i\}$ are updated accordingly with the step $\epsilon$. This step can be adjusted via *epsilon* in **\*\*kwargs*. Finally, the updated coefficients are normalized as required by the quantum mechanics. In the package, Adam is not applied by default in AD and it can be turned on by setting *Adam=True* in **\*\*kwargs*.

Regarding the gradient-free methods, apart from the PSO, DE and DDPG, QuanEstimation also contains the Nelder-Mead algorithm (NM) [116], which has already been used by Fröwis *et al.* [112] to perform the state optimization in the case of collective spins. The detailed flow chart of NM to locate the minimum value of an objective function can be found in Ref. [17]. For the sake of self-consistency of the paper, here we present its pseudocode in Algorithm 6 for the search of the maximum value of $f$ at the target time $T$.

In NM, $n+1$ guessed states are input and sorted descendingly according to the corresponding values of $f$, namely, $f(|\psi_1\rangle) \geqslant \cdots \geqslant f(|\psi_{n+1}\rangle)$. In one episode of optimization, the average state $|\psi_a\rangle := \frac{1}{\sqrt{\mathcal{N}_a}}\sum_{k=1}^n |\psi_k\rangle$ and reflected state $|\psi_r\rangle := \frac{1}{\sqrt{\mathcal{N}_r}}[|\psi_a\rangle + a_r(|\psi_a\rangle - |\psi_{n+1}\rangle)]$ are first calculated. In the case that the reflected state is better than $|\psi_1\rangle$, i.e., $f(|\psi_r\rangle)$ is larger than $f(|\psi_1\rangle)$, the expanded state $|\psi_e\rangle := \frac{1}{\sqrt{\mathcal{N}_e}}[|\psi_a\rangle + a_e(|\psi_r\rangle - |\psi_a\rangle)]$ is then calculated and compared to the reflected state. If the reflected state is still better, then replace $|\psi_{n+1}\rangle$ with $|\psi_r\rangle$, otherwise replace $|\psi_{n+1}\rangle$ with $|\psi_e\rangle$. In the

---

**Algorithm 6:** NM for pure states

---

Receive a set of guessed states $|\psi_1\rangle, \cdots, |\psi_{n+1}\rangle$;

**for** episode=1, $M$ **do**

    Evolve all states according to the given dynamics and calculate the objective function $f$ at time $T$;

    Sort the states and reassign the indices to let $f(|\psi_1\rangle) \geq f(|\psi_2\rangle) \geq \cdots \geq f(|\psi_{n+1}\rangle)$;

    Calculate the average state $|\psi_a\rangle = \frac{1}{\sqrt{\mathcal{N}_a}}\sum_{k=1}^n |\psi_k\rangle$;

    Calculate the reflected state $|\psi_r\rangle = \frac{1}{\sqrt{\mathcal{N}_r}}[|\psi_a\rangle + a_r(|\psi_a\rangle - |\psi_{n+1}\rangle)]$;

    **if** $f(|\psi_r\rangle) > f(|\psi_1\rangle)$ **then**

        Calculate the expanded state $|\psi_e\rangle = \frac{1}{\sqrt{\mathcal{N}_e}}[|\psi_a\rangle + a_e(|\psi_r\rangle - |\psi_a\rangle)]$;

        **if** $f(|\psi_r\rangle) \geq f(|\psi_e\rangle)$ **then**

            Replace $|\psi_{n+1}\rangle$ with $|\psi_r\rangle$;

        **else**

            Replace $|\psi_{n+1}\rangle$ with $|\psi_e\rangle$;

        **end if**

    **else if** $f(|\psi_1\rangle) \geq f(|\psi_r\rangle) > f(|\psi_n\rangle)$ **then**

        Replace $|\psi_{n+1}\rangle$ with $|\psi_r\rangle$;

    **else if** $f(|\psi_n\rangle) \geq f(|\psi_r\rangle) > f(|\psi_{n+1}\rangle)$ **then**

        Calculate the outside contracted state $|\psi_{oc}\rangle = \frac{1}{\sqrt{\mathcal{N}_{oc}}}[|\psi_a\rangle + a_c(|\psi\rangle_r - |\psi_a\rangle)]$;

        **if** $f(|\psi_{oc}\rangle) \geq f(|\psi_r\rangle)$ **then**

            Replace $|\psi_{n+1}\rangle$ with $|\psi_{oc}\rangle$;

        **else**

            Replace all $|\psi_k\rangle$ for $k \in [2, n+1]$ with $\frac{1}{\sqrt{\mathcal{N}_k}}[|\psi_1\rangle + a_s(|\psi_k\rangle - |\psi_1\rangle)]$;

        **end if**

    **else**

        Calculate the inside contracted state $|\psi_{ic}\rangle = \frac{1}{\sqrt{\mathcal{N}_{ic}}}[|\psi_a\rangle - a_c(|\psi_a\rangle - |\psi_{n+1}\rangle)]$;

        **if** $f(|\psi_{ic}\rangle) > f(|\psi_{n+1}\rangle)$ **then**

            Replace $|\psi_{n+1}\rangle$ with $|\psi_{ic}\rangle$;

        **else**

            Replace all $|\psi_k\rangle$ for $k \in [2, n+1]$ with $\frac{1}{\sqrt{\mathcal{N}_k}}[|\psi_1\rangle + a_s(|\psi_k\rangle - |\psi_1\rangle)]$;

        **end if**

    **end if**

**end for**

Return the optimal state $|\psi_1\rangle$ and corresponding $f$.

---

case that the performance of the reflected state is in the middle of $|\psi_1\rangle$ and $|\psi_n\rangle$, just replace $|\psi_{n+1}\rangle$ with it. If its performance is between $|\psi_n\rangle$ and $|\psi_{n+1}\rangle$, then the outside contracted state $|\psi_{oc}\rangle := \frac{1}{\sqrt{\mathcal{N}_{oc}}}[|\psi_a\rangle + a_c(|\psi\rangle_r - |\psi_a\rangle)]$ is calculated and compared to the reflected state. $|\psi_{n+1}\rangle$ is replaced with $|\psi_{oc}\rangle$ if $|\psi_{oc}\rangle$ outperforms the reflected state, otherwise all states $\{|\psi_k\rangle\}$, except the best one $|\psi_1\rangle$, are replaced with the states $\frac{1}{\sqrt{\mathcal{N}_k}}[|\psi_1\rangle + a_s(|\psi_k\rangle - |\psi_1\rangle)]$ and the program goes to the next episode. In the case that $|\psi_r\rangle$ is no better than any state in $\{|\psi_k\rangle\}$, the inside contracted state $|\psi_{ic}\rangle := \frac{1}{\sqrt{\mathcal{N}_{ic}}}[|\psi_a\rangle - a_c(|\psi_a\rangle - |\psi_{n+1}\rangle)]$ is then calculated and compared to $|\psi_{n+1}\rangle$. If it is better than $|\psi_{n+1}\rangle$, replace $|\psi_{n+1}\rangle$ with it, otherwise perform the same replacement operation to all states as done previously. At the beginning of next round, all states are sorted in descending order again. $\mathcal{N}_a := \langle\psi_a|\psi_a\rangle$ is the normalization coefficient, same as $\mathcal{N}_r, \mathcal{N}_e, \mathcal{N}_{oc}$, and $\mathcal{N}_{ic}$. A general setting of

**Algorithm 7:** RI

---

Receive the guessed probe state $\rho_0$;
**for** $m=1, M$ **do**

  Evolve the state with $\rho = \sum_i K_i \rho_0 K_i^\dagger$;
  Calculate the derivative
  $\partial_a \rho = \sum_i (\partial_a K_i) \rho_0 K_i^\dagger + K_i \rho_0 (\partial_a K_i^\dagger)$;
  Calculate the QFI and the SLD $L$ with $\rho$ and $\partial_{\mathbf{x}} \rho$;
  Calculate the matrix $\mathcal{M}$;
  Find the eigenvector $|\psi_m\rangle$ of $\mathcal{M}$ corresponding to
    its largest eigenvalue;
  Replace $\rho_0$ with $|\psi_m\rangle\langle\psi_m|$.
**end for**
Return the optimal state $|\psi_M\rangle$ and the QFI.

---

the coefficients are $a_r = 1.0$, $a_e = 2.0$, $a_c = a_s = 0.5$, which are also the default values in the package. These coefficients can be adjusted in **\*\*kwargs** (shown in Table III) via *ar*, *ae*, *ac*, and *as0*. In the meantime, *p_num* in **\*\*kwargs** represents the state number $n+1$.

Apart from the aforementioned algorithms, there also exist dedicated algorithms for the state optimization in quantum parameter estimation. Here we introduce a reverse iterative algorithm (RI), which was first proposed in Refs. [117,118] in the Bayesian estimation context, and then applied to the QFI in Ref. [119]. In the case of single-parameter estimation, the QFI can be rewritten into

$$\mathcal{F}_{aa} = \sup_A [2\mathrm{Tr}(A\partial_a\rho) - \mathrm{Tr}(\rho A^2)]. \qquad (66)$$

This form is equivalent to the standard definition of the QFI as can be seen by solving the maximization problem $2\mathrm{Tr}(A\partial_a\rho) - \mathrm{Tr}(\rho A^2)$ with respect to $A$, which is formally a quadratic function in matrix $A$ and the resulting extremum condition yields the standard linear equation for $\partial_a\rho = \frac{1}{2}(A\rho + \rho A)$, i.e., the optimal $A = L_a$ is just the SLD operator. When this solution is plugged into the formula and it yields $\mathrm{Tr}(\rho L_a^2)$, which is in agreement with the standard definition of the QFI. Consider the parametrization process described by the Kraus operators given in Eq. (4), $\rho = \sum_i K_i(x)\rho_0 K_i^\dagger(x)$. Taking into account Eq. (66), we see that the problem of identifying the optimal input state $\rho_0$ that maximizes the QFI, can be written as a double maximization problem,

$$\sup_{\rho_0} \mathcal{F}_{aa} = \sup_{A, \rho_0} [2\mathrm{Tr}(A\partial_a\rho) - \mathrm{Tr}(\rho A^2)]. \qquad (67)$$

This observation leads to an effective iterative protocol, where for a fixed $\rho_0$ we find the optimal $A$ that maximizes the above expression, and then fixing the optimal $A$ found in the previous step we look for the optimal $\rho_0$. In order to implement the procedure, note that the QFI can be rewritten in the "Heisenberg picture" form, where the Kraus operators effectively act on the $L_a$ operators, as

$$\mathcal{F}_{aa} = \mathrm{Tr}(\rho_0 \mathcal{M}) \qquad (68)$$

with

$$\mathcal{M} = \sum_i 2[(\partial_a K_i^\dagger)L_a K_i + K_i^\dagger L_a(\partial_a K_i)] - K_i^\dagger L_a^2 K_i. \qquad (69)$$

This equation indicates that for a fixed $\mathcal{M}$ (i.e., fixed $A = L_a$), the optimal probe state is nothing but the eigenvector corresponding to the maximum eigenvalue of $\mathcal{M}$. The pseudocode of this algorithm is given in Algorithm 7. In one round of the optimization, $\mathcal{M}$ is calculated and its eigenvector with respect to the maximum eigenvalue of $\mathcal{M}$ is calculated and used as the probe state in the next round. In the package, this method
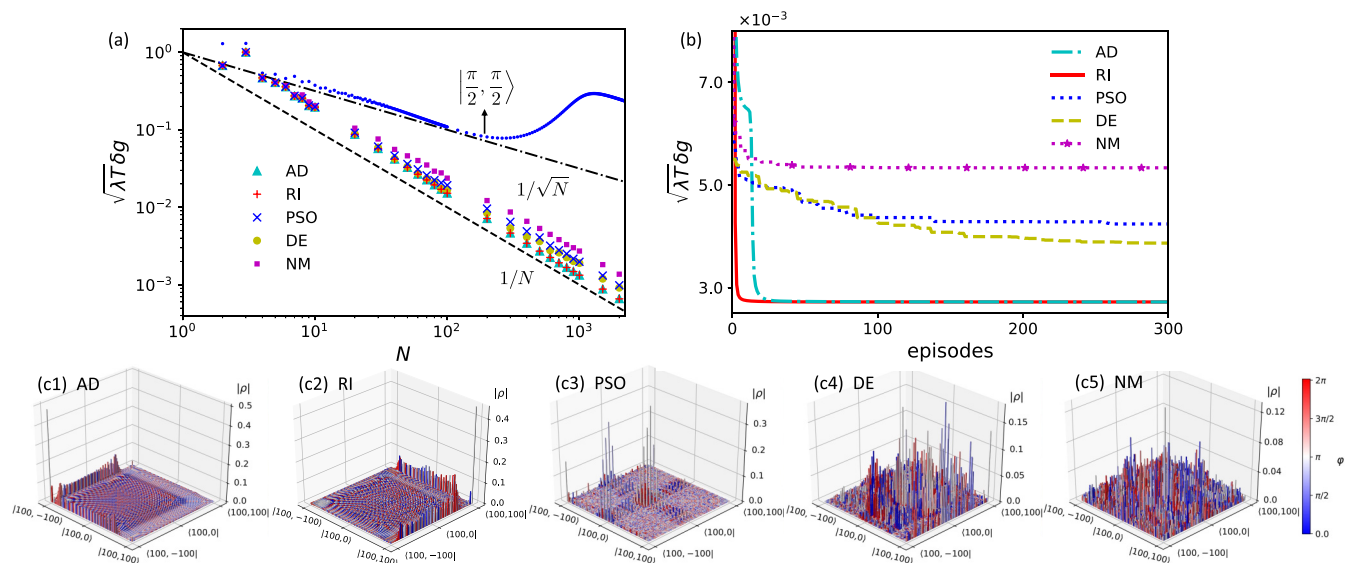


FIG. 13. (a) The performance of the optimal probe states searched via AD (cyan triangles), RI (red pluses), PSO (blue crosses), DE (yellow circles), and NM (purple squares) in the Lipkin-Meshkov-Glick model in the absence of noise. The blue dots represents the value of $\sqrt{\lambda T}\delta g$ for the coherent spin state $|\pi/2, \pi/2\rangle$, and the dash-dotted-black and dashed-black lines represent $1/\sqrt{N}$ and $1/N$, respectively. (b) The convergence performance of AD (dash-dotted-cyan line), RI (solid-red line), PSO (dotted-blue line), DE (dashed-yellow line), and NM (dotted-star-purple line) in the case of $N = 500$. [(c1)–(c5)] The searched optimal states with different algorithms in the case of $N = 100$. The target time is chosen as $\lambda T = 10$. The true value of $g$ is 0.5, and the value of $h/\lambda$ is set to be 0.1. Planck units are applied here.

can be invoked via *method = "RI"*. The number of episodes and the seed can be adjusted in ***kwargs** (shown in Table III) via *max_episode* and *seed*. Notice that this method is only available when *state.Kraus()* is invoked, and in the current version of the package, it only works for the single-parameter quantum estimation, i.e., the objective function is the QFI. The extension to the CFI and the case of multiparameter estimation will be thoroughly discussed in an independent paper.

*Example.* Here we use the Lipkin-Meshkov-Glick model as an example to show the state optimization with QuanEstimation. The Hamiltonian of this model is [120]

$$H_{\text{LMG}} = -\frac{\lambda}{N}\left(J_1^2 + gJ_2^2\right) - hJ_3, \qquad (70)$$

where $J_i = \frac{1}{2}\sum_{j=1}^{N}\sigma_i^{(j)}$ ($i = 1, 2, 3$) is the collective spin operator with $\sigma_i^{(j)}$ the $i$th Pauli matrix for the $j$th spin. $N$ is the total number of spins, $\lambda$ is the spin-spin interaction strength, $h$ is the strength of the external field and $g$ is the anisotropic parameter. All searches with different algorithms start from the coherent spin state $|\theta = \pi/2, \phi = \pi/2\rangle$, which is defined by [121]

$$|\theta, \phi\rangle = \exp\left(-\frac{\theta}{2}e^{-i\phi}J_+ + \frac{\theta}{2}e^{i\phi}J_-\right)|J, J\rangle, \qquad (71)$$

where $|J, J\rangle$ is a Dicke state with $J = N/2$ and $J_\pm = J_1 \pm iJ_2$. Here we consider the case that the search is constrained to pure states with fixed $J = N/2$, which can be expressed as $|\psi\rangle = \sum_{m=-J}^{J} c_m|J, m\rangle$ with $|J, m\rangle$ a general Dicke state and $c_m$ a complex coefficient. Let us first study the single-parameter scenario with $g$ the parameter to be estimated.

The performance of the optimal probe states searched via AD (cyan triangles), RI (red pluses), PSO (blue crosses), DE (yellow circles) and NM (purple squares) in the absence of noise are given in Fig. 13(a). Here $\delta g = 1/\sqrt{\mathcal{F}_{gg}}$ is the theoretical optimal deviation for $g$. The target time is taken as $\lambda T = 10$ (Planck units are applied). The performance of DDPG is not good enough and thus not shown in the figure. For a very small $N$, the searched optimal states do not show an obvious advantage than the state $|\pi/2, \pi/2\rangle$ (blue dots). However, when $N$ is large the advantage becomes significant, and the performance of all searched states outperform $|\pi/2, \pi/2\rangle$ and $1/\sqrt{N}$ (dash-dotted-black line) in the case that $N$ is larger than around 6. For a large $N$, the performance of the states obtained via AD and RI are the best and very close to $1/N$ (dashed-black line). The performance of DE and PSO basically coincide with each other (more accurately to say, the performance of DE is slightly better than that of PSO), but is worse than AD and RI. The performance of NM is the worst in this example. Please note that we cannot rashly say that the general performance of NM is worse than DE or PSO in the state optimization just based on this plot as different parameter settings in the algorithms sometimes could dramatically affect the behaviors, yet we basically use the general recommended settings in all algorithms. Nevertheless, different sensitivities of the parameter settings on the final result still indicates that DE and PSO are easier to locate optimal states than NM at least in this example.

Regarding the convergence performance in this example, as shown in Fig. 13(b), RI shows the fastest convergence speed

and the best optimized value. AD is slightly slower than RI but still way faster than the gradient-free methods. However, the disadvantage of AD is that occupation of memory grows very fast with the increase of $N$. Hence, RI would be the best choice to try first for the state optimization in the case of unitary parametrization. In the last, as a demonstration, the searched optimal states via different algorithms in the case of $N = 100$ are shown in Figs. 13(c1)–13(c5).

*Example.* When the collective dephasing is involved, the dynamics of this system is governed by the following master equation:

$$\partial_t\rho = -i[H_{\text{LMG}}, \rho] + \gamma\left(J_3\rho J_3 - \frac{1}{2}\{\rho, J_3^2\}\right) \qquad (72)$$

with $\gamma$ the decay rate. The performance of optimal probe states searched via AD (solid-red line), PSO (dashed-star-blue line), DE (dash-dotted-circle-cyan line), and NM (dashed-purple line) are illustrated with $N = 8$ and $N = 30$ in Figs. 14(a) and 14(c), respectively. The corresponding optimal probe states are given in Figs. 14(b1)–14(b4) for $N = 8$ and Figs. 14(d1)–14(d4) for $N = 30$. In both cases, the states obtained via AD, PSO, and DE basically present coincidental performance at time $T$, and outperform $|\pi/2, \pi/2\rangle$ (dotted-black lines). Similar to the unitary scenario, the state obtained via NM shows a worse performance at time $T$, and it even fails to find a better state than $|\pi/2, \pi/2\rangle$ in the case of $N = 30$. In this figure, the number of parallel sets (also called particles in PSO and populations in DE) are 10 for all NM, DE, and PSO. After increasing the number of parallel sets from 10 to 20 [labelled by NM (20) in the plot], the performance of NM (dash-dotted-green line) improves in the case of $N = 8$, which basically coincides with others. However, it still fails to find a better state when $N = 30$. More number of parallel sets may be requires for NM in this case. The states obtained via NM (20) are shown in Figs. 14(b5) and 14(d5) for $N = 8$ and $N = 30$, respectively.

Next we discuss the state optimization in multiparameter estimation. Consider the simultaneous estimation of $g$ and $h/\lambda$ in the Lipkin-Meshkov-Glick model with the dynamics in Eq. (72). Figures 15(a) and 15(b) show the performance of optimal states obtained via different algorithms for $W = \text{diag}(1/2, 1/2)$ and $W = \text{diag}(1/3, 2/3)$, respectively. In both cases AD (solid-red line) and DE (dash-dotted-circle-cyan line) present the best performance at the target time $\lambda T = 10$, and DE even slightly outperform AD in the case of $W = \text{diag}(1/2, 1/2)$. The performance of PSO (dashed-star-blue line) is worse than AD and DE, yet still better than NM (dashed-purple line) and NM with 20 parallel sets (dash-dotted-green line). The performance of NM does not even outperform the coherent spin state $|\pi/2, \pi/2\rangle$ (dotted-black line) in the case of $W = \text{diag}(1/2, 1/2)$. Hence, apart from gradient-based algorithm like AD, PSO, and DE would also be good choices for state optimizations. The optimal states obtained from AD and DE for $W = \text{diag}(1/2, 1/2)$ and $W = \text{diag}(1/3, 2/3)$ are demonstrated in Figs. 15(c1) and 15(c2) and Figs. 15(d1) and 15(d2), respectively. Although the performance on $\text{Tr}(W\mathcal{F}^{-1})$ are basically the same for these states, they may still have gaps on other properties like the difficulties of preparation, the robustness to the imperfect preparation and so on. Hence, in practice one needs to
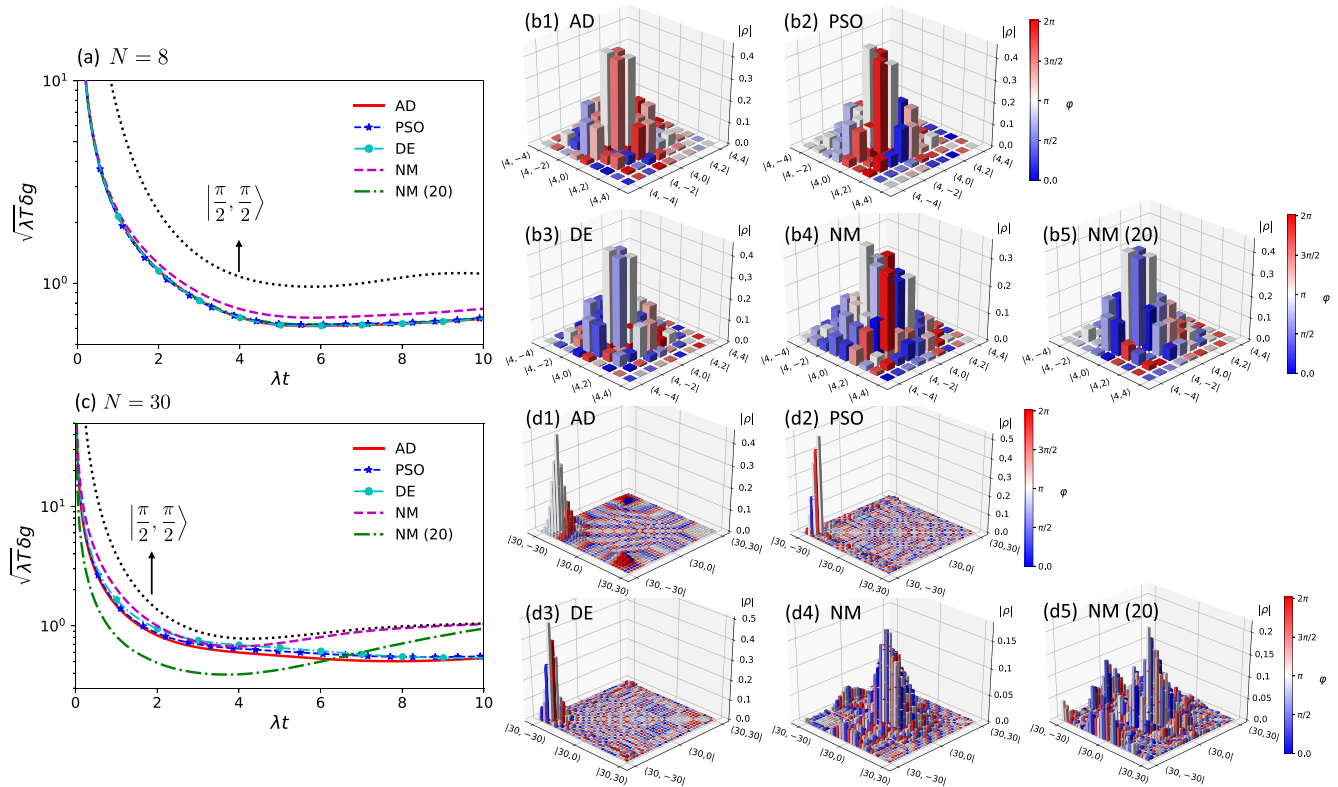
FIG. 14. The performance of probe states obtained via different algorithms for (a) $N = 8$ and (c) $N = 30$ when the collective dephasing exists. The solid-red line, dashed-star-blue line, dash-dotted-circle-cyan line, dashed-purple line represent the values of $\sqrt{\lambda T} \delta g$ for the searched states obtained via AD, PSO, DE, and NM, respectively. The dash-dotted-green line represents that of NM with 20 parallel sets. The dotted-black line represent the result of $|\pi/2, \pi/2\rangle$. [(b1)–(b5)] The searched optimal states for $N = 8$. [(d1)–(d5)] The searched optimal states for $N = 30$. The target time $\lambda T = 10$, and the true values of $g$ is 0.5. The value of $h/\lambda$ is set to be 0.1 and the decay rate $\gamma/\lambda = 0.1$. Planck units are applied here.

compare these optimal states comprehensively case by case to make wise choices.

## VIII. MEASUREMENT OPTIMIZATION

Measurement is critical in quantum parameter estimation [122–125]. On one hand, all asymptotic bounds require some optimal measurements to attain if it is attainable, and hence the search of optimal measurements is a natural requirement in theory to approach the ultimate precision limit. On the other hand, the choice of measurements is usually limited in practice, and how to find conditioned optimal measurements with the practical measurements in hand is an important step towards the design of a realizable scheme. QuanEstimation includes the optimization of measurements for several scenarios. The first one is the optimization of rank-one projective measurements. A set of projective measurements $\{\Pi_i\}$ satisfies $\Pi_i \Pi_j = \Pi_i \delta_{ij}$ and $\sum_i \Pi_i = \mathbb{1}$, and it can be rewritten into $\{|\phi_i\rangle\langle\phi_i|\}$ with $\{|\phi_i\rangle\}$ an orthonormal basis in the Hilbert space. In this way, the optimization of rank-one projective measurement is equivalent to identifying the optimal basis, which can be realized using PSO and DE in QuanEstimation. In this case the automatic differentiation is not working very well due to the Gram-Schmidt orthogonalization procedure after the update of $\{|\phi_i\rangle\}$ according to the gradients. In some cases, the realizable measurement has to be limited in the

linear combination of a given set of POVM, hence, the second scenario is to find the optimal linear combination of an input measurement. Moreover, in some cases the measurement $\{\Pi_i\}$ has to be fixed, but an arbitrary unitary operation can be invoked before performing the measurement, which is equivalent to a new measurement $\{U \Pi_i U^\dagger\}$. Based on this, the third scenario is to find the optimal rotated measurement of an input measurement.

The code in QuanEstimation for the execution of measurement optimization are as follows:

```
m = MeasurementOpt(mtype = "projection",
          minput = [], savefile = False,
          method = "DE", **kwargs)
m.dynamics(tspan, rho0, H0, dH, Hc = [], ctrl = [],
          decay = [], dyn_method = "expm")
m.CFIM(W = [])
```

In the case that the parametrization is described by the Kraus operators, replace *m.dynamics()* with the code *m.Kraus(rho0,K,dK)*. The optimization method can be adjusted via *method = " "* and corresponding parameters can be set via ***kwargs*. The available optimization methods and corresponding default parameter settings are given in Table IV. *dyn_method = "ode"* is also available here to invoke ODE for solving the dynamics, except the case that *method = "AD"*
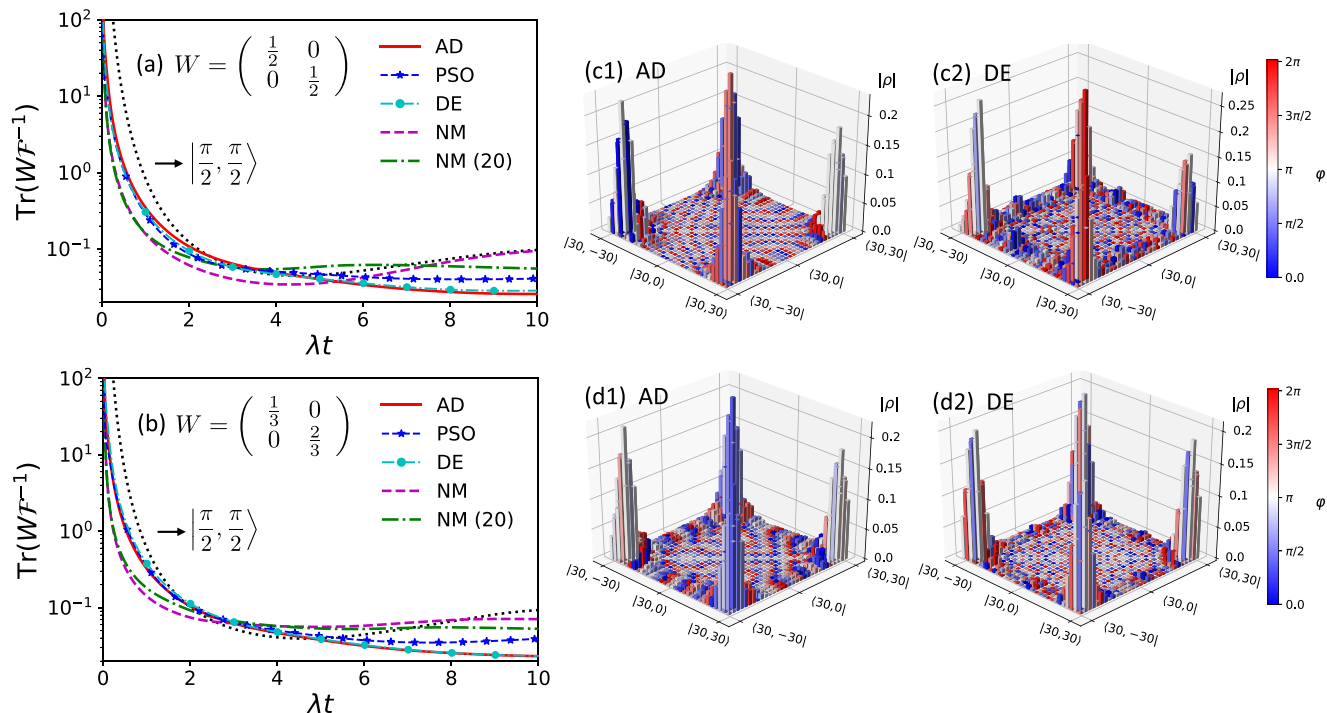
FIG. 15. The performance of different algorithms for the weight matrix (a) $W = \text{diag}(1/2, 1/2)$ and (b) $W = \text{diag}(1/3, 2/3)$. The solid-red line, dashed-star-blue line, dash-dotted-circle-cyan line, dashed-purple line, and dash-dotted-green line represent the results obtained via AD, PSO, DE, NM, and NM with 20 parallel sets, respectively. The dotted-black line represent the result of $|\pi/2, \pi/2\rangle$. [(c1),(c2)] The optimal states obtained from AD and DE for $W = \text{diag}(1/2, 1/2)$. [(d1),(d2)] The optimal states obtained from AD and DE for $W = \text{diag}(1/3, 2/3)$. The target time $\lambda T = 10$. The true values of $g$ and $h/\lambda$ are set to be 0.5 and 0.1. Planck units are applied here.

is applied. Two files "f.csv" and "measurements.csv" will be generated at the end of the program. When *savefile = True*,

TABLE IV. Available methods for measurement optimization in QuanEstimation and corresponding default parameter settings. Notice that AD is only available when *mtype = "input"*. Here *measurement0* is the initial guess of the measurement.

| Algorithms | method= | **kwargs and default values | |
|---|---|---|---|
| PSO | "PSO" | "p_num" | 10 |
| | | "measurement0" | [] |
| | | "max_episode" | [1000,100] |
| | | "c0" | 1.0 |
| | | "c1" | 2.0 |
| | | "c2" | 2.0 |
| | | "seed" | 1234 |
| DE | "DE" | "p_num" | 10 |
| | | "measurement0" | [] |
| | | "max_episode" | 1000 |
| | | "c" | 1.0 |
| | | "cr" | 0.5 |
| | | "seed" | 1234 |
| AD | "AD" | "Adam" | False |
| (available when | | "measurement0" | [] |
| *mtype = "input"*) | | "max_episode" | 300 |
| | | "epsilon" | 0.01 |
| | | "beta1" | 0.90 |
| | | "beta2" | 0.99 |

the measurements obtained in all episodes will be saved in "measurements.csv".

The variable *mtype = " "* defines the type of scenarios for the optimization, and currently it includes two options: *mtype = "projection"* and *mtype = "input"*. The first one means the optimization is performed in the first scenario, i.e., within the set of projective measurements. In this case, *minput = []* should keep empty. Since $|\phi_i\rangle$ in a rank-one projective measurement $\{|\phi_i\rangle\langle\phi_i|\}$ can be expended as $|\phi_i\rangle = \sum_j C_{ij}|j\rangle$ in a given orthonormal basis $\{|j\rangle\}$, the optimization of the rank-one projective measurement is equivalent to the optimization of a complex matrix $C$. When the gradient-free methods are applied, all entries in $C$ are updated via the given algorithm in each episode, then adjusted via the Gram-Schmidt orthogonalization procedure to make sure $\{|\phi_i\rangle\langle\phi_i|\}$ is a legitimate projective measurement, i.e., $\langle\phi_i|\phi_j\rangle = \delta_{ij}$, $\forall i, j$ and $\sum_i |\phi_i\rangle\langle\phi_i| = \mathbb{1}$. The second option *mtype = "input"* means the optimization is performed in the second and third scenarios. The input rule of *minput* for the second scenario is *minput = ["LC", [Pi1,Pi2,...], m]* and for the third one is *minput = ["rotation", [Pi1,Pi2,...]]*. Here *[Pi1,Pi2,...]* is a list of matrices representing the input measurement $[\Pi_1, \Pi_2, \dots]$. The variable *m* in the second scenario is an integer representing the number of operators of the output measurement, and thus should be no larger than that of the input measurement. For example, assume the input measurement is $\{\Pi_i\}_{i=1}^6$ and input 4 in the position of *m* means the the output measurement is $\{\Pi_i'\}_{i=1}^4$ where $\Pi_i' = \sum_{j=1}^6 B_{ij}\Pi_j$. The optimization is to find
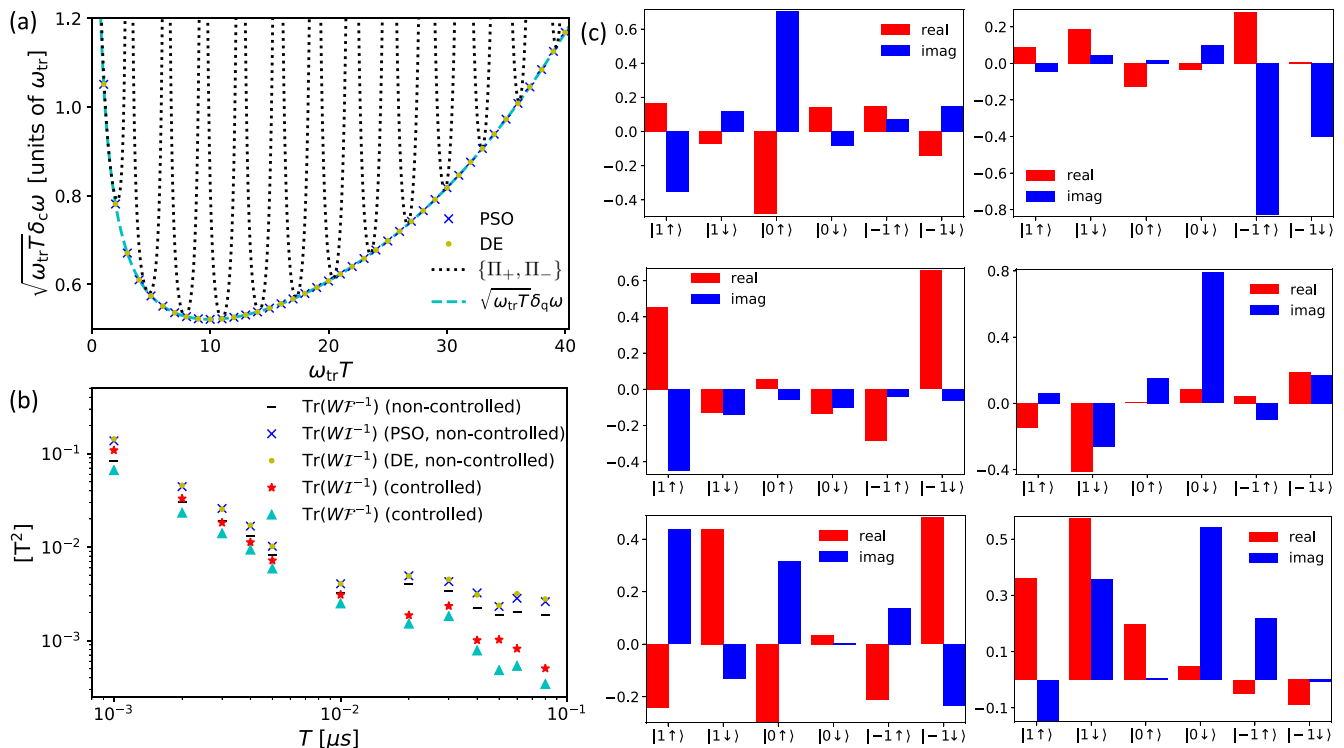
FIG. 16. (a) The performance of optimal projective measurements obtained via PSO (blue crosses) and DE (yellow circles) in the case of single-parameter estimation. The dashed-cyan line represents the values of $\sqrt{\omega_{tr}T}\delta_q\omega$ and the dotted-black line represents the values of $\sqrt{\omega_{tr}T}\delta_c\omega$ with respect to the projective measurement $\{\Pi_+ = |+\rangle\langle+|, \Pi_- = |-\rangle\langle-|\}$. The true value $\omega_{tr} = 1$. Planck units are applied in this plot. (b) The performance of optimal projective measurements obtained via PSO (blue crosses) and DE (yellow circles) in the case of multiparameter estimation in the absence of control. The black underlines and cyan triangles represent the values of $\mathrm{Tr}(W\mathcal{F}^{-1})$ without and with optimal control. The red pentagrams represent the controlled values of $\mathrm{Tr}(W\mathcal{I}^{-1})$ with the optimal measurements obtained in the noncontrolled scenario. (c) Demonstration of the optimal projective measurement obtained by DE in the multiparameter estimation at the target time $T = 0.04\,\mu$s. The red and blue bars represent the real and imaginary parts of the coefficients of the optimal measurement in the basis $\{|1\uparrow\rangle, |1\downarrow\rangle, |0\uparrow\rangle, |0\downarrow\rangle, |-1\uparrow\rangle, |-1\downarrow\rangle\}$.

an optimal real matrix $B$ for the optimal CFI or $\mathrm{Tr}(W\mathcal{I}^{-1})$. To make sure the updated measurement in each episode is still a legitimate POVM, all entries of $B$ are limited in the regime $[0,1]$ and $\sum_i B_{ij}$ is required to be 1, which is realized by the normalization process. In this scenario, apart from PSO and DE, AD can also be implemented. In the third scenario, the unitary operation is expressed by $U = \prod_k \exp(is_k\lambda_k)$ where $\lambda_k$ is a SU($N$) generator and $s_k$ is a real number in the regime $[0, 2\pi]$. The optimization is to find an optimal set of $\{s_k\}$ for the optimal CFI or $\mathrm{Tr}(W\mathcal{I}^{-1})$, and similar to the second scenario, AD is also available here besides PSO and DE. In the case that *mtype = "projection"*, each entry of *measurement0* in \*\*kwargs is a list of arrays, and in the case that *mtype = "input"*, each entry is an array.

*Example.* Now we consider two models to demonstrate the measurement optimizations in the first scenario. The first one is a single-parameter case with the single-qubit Hamiltonian $H = \omega\sigma_3/2$ and dynamics in Eq. (12). $\delta_c\omega$ and $\delta_q\omega$ are defined in Eqs. (56) and (57). As shown in Fig. 16(a), $\delta_c\omega$ for the projective measurement $\{\Pi_+ = |+\rangle\langle+|, \Pi_- = |-\rangle\langle-|\}$ (dotted-black line) can only reach $\delta_q\omega$ (dashed-cyan line) at some specific time points, which has already been shown in Sec. IV A. However, utilizing the optimal projective measurements obtained via PSO (blue crosses) and DE (yellow

circles), $\delta_c\omega$ saturates $\delta_q\omega$ for all target time. This performance coincides with the common understanding that the QFI can be theoretically attained by certain optimal measurements.

In the case of multiparameter estimation, we use the Hamiltonian in Eq. (63) and dynamics in Eq. (65) to demonstrate the performance of the optimal projective measurements. The magnetic field $\vec{B}$ is still the quantity to be estimated. Different with the single-parameter case, the values of $\mathrm{Tr}(W\mathcal{I}^{-1})$ for the optimal measurements found by PSO (blue crosses) and DE (yellow circles) cannot attain $\mathrm{Tr}(W\mathcal{F}^{-1})$ (black underlines) in the absence of control, as shown in Fig. 16(b). The gap between $\mathrm{Tr}(W\mathcal{F}^{-1})$ and $\mathrm{Tr}(W\mathcal{I}^{-1})$ is due to the fact that the quantum Cramér-Rao bound is not attainable here. Next, together with the optimal measurement, which gives the lowest $\mathrm{Tr}(W\mathcal{I}^{-1})$, the control is also invoked to further evaluate the reduction of $\mathrm{Tr}(W\mathcal{I}^{-1})$. Utilizing the optimal controls obtained via auto-GRAPE, the values of $\mathrm{Tr}(W\mathcal{I}^{-1})$ (red pentagrams) continue to reduce compared to the noncontrolled case, yet it is still unable to attain the controlled values of $\mathrm{Tr}(W\mathcal{F}^{-1})$ (cyan triangles) in general due to the attainability problem. Nevertheless, their differences are very insignificant for some target time, indicating that the combined performance of the optimal measurement and optimal control approaches to the ultimate precision limit.
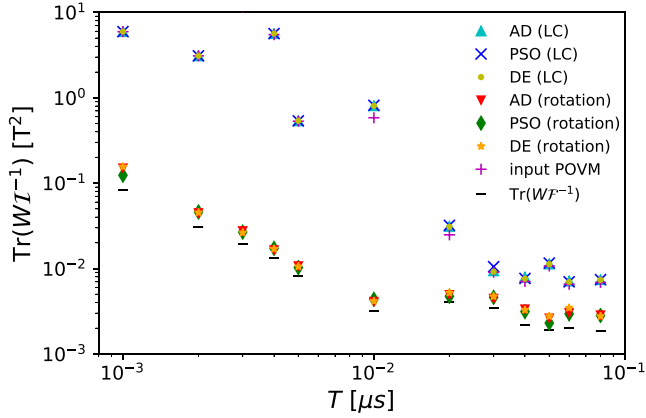
FIG. 17. Demonstration of the measurement optimization in the second (LC) and third scenarios (rotation). The cyan-upward triangles, blue crosses and yellow circles represent the performance of optimal measurements found by AD, PSO, and DE, respectively in the second scenario. The red-downward triangles, green diamonds and orange pentagrams represent the performance of optimal measurements found by AD, PSO, and DE in the third scenario.

The optimal measurement $\{|\phi_1\rangle\langle\phi_1|, \ldots, |\phi_6\rangle\langle\phi_6|\}$ obtained by DE in the absence of control are demonstrated in Fig. 16(c). The red and blue bars represent the real and imaginary parts of the coefficients of $|\phi_1\rangle$ to $|\phi_6\rangle$ in the basis $\{|1\uparrow\rangle, |1\downarrow\rangle, |0\uparrow\rangle, |0\downarrow\rangle, |-1\uparrow\rangle, |-1\downarrow\rangle\}$.

The optimizations in the second and third scenarios are also demonstrated with the Hamiltonian in Eq. (63) and dynamics in Eq. (65). The input measurement is taken as $\{|ij\rangle\langle ij|\}_{i=0,\pm1;j=\uparrow,\downarrow}$, which includes 6 operators. In the second scenario, the number of output POVM operators is set to be 4. As shown in Fig. 17, the performance of measurements found by AD (cyan-upward triangles), PSO (blue crosses) and DE (yellow circles) approach to and even reach that of the input measurement (magenta pluses). This fact indicates that in this case, an optimal 4-operator measurement can reach the performance of the original 6-operator measurement, and the reduction of operator numbers may benefit the practical precision of the measurements in experiments. In the third scenario, the performance of optimal measurements found by AD (red-downward triangles), PSO (green diamonds), and DE (orange pentagrams) not only significantly better than that of the input measurement, but also approach to the ultimate precision limit given by $\text{Tr}(W\mathcal{F}^{-1})$ (black underlines), indicating that the performance of these optimal measurements are very close to that of the global optimal measurements, if there exist any. The probe states, the true values of the parameters to be estimated and other parameters are set to be the same with those in Sec. VI G.

## IX. COMPREHENSIVE OPTIMIZATION

The previous sections focused on the univariate (single variable) optimizations. However, in a practical scenario the probe state, control (if available) and measurement may all need to be optimized. More importantly, the optimal results obtained for an univariate optimization may cease to be optimal when other variables are involved. For example, the

optimal probe state and measurement for the noncontrolled case may not be optimal anymore in the controlled case. Hence, sometimes a comprehensive optimization, i.e., simultaneous multivariate optimization, is in need.

QuanEstimation can deal with four types of multivariate optimizations, including the optimizations of the probe state and measurement (SM), the probe state and control (SC), control and measurement (CM), and all three together (SCM). In these scenarios, the key feature of comprehensive optimization is that all variables are optimized simultaneously. Regarding the objective function, in the cases of SM, CM, and SCM, namely, when the measurement is involved, it has to be dependent on the measurement. In current version of the package it is chosen as the CFI or $\text{Tr}(W\mathcal{I}^{-1})$. In the case of SC, the objective function could be either QFI/$\text{Tr}(W\mathcal{F}^{-1})$ or CFI/$\text{Tr}(W\mathcal{I}^{-1})$ for a flexible or fixed choice of measurement. The process of comprehensive optimizations and corresponding objective functions have been illustrated in the first lines (with gray background) in Figs. 18(a)–18(d). In QuanEstimation, the code for the execution of comprehensive optimization are:

```
com = ComprehensiveOpt(savefile=False,
                method="DE",**kwargs)
com.dynamics(tspan,H0,dH,Hc=[],ctrl=[],
                decay=[],ctrl_bound=[],
                dyn_method="expm")
com.SM(W=[])
com.SC(W=[],M=[],target="QFIM",LDtype="SLD")
com.CM(rho0,W=[])
com.SCM(W=[])
```

In the case that the parametrization is described by the Kraus operators, replace *com.dynamics()* with the code *com.Kraus(K,dK)*. All four types of comprehensive optimizations can be called through *com.SM()*, *com.SC()*, *com.CM()*, and *com.SCM()*. Notice that if *com.Kraus()* is invoked, only *com.SM()* is available as control is not suitable for the parameterization process described by the Kraus operators. In *com.CM()*, the input *rho0* is a matrix representing the fixed probe state. In *com.SC()*, the objective function can be set via *target ="  "*, including three choices *target = "QFIM"* (default), *target = "CFIM"*, and *target = "HCRB"*. If a set of measurement is input via *M = []*, the objective function will be automatically chosen as the CFIM regardless of the input in *target = "  "*. The type of QFIM can be adjusted via *LDtype = "  "* (*"SLD"*, *"RLD"*, *"LLD"*). The available methods for the comprehensive optimization and corresponding default parameter settings are given in Table V. Notice that AD is only available when *com.SC()* is called and the objective function is not the HCRB. At the end of the program, "f.csv" will be generated including the values of the objective function in all episodes. In the meantime, some or all of the files "controls.csv", "states.csv", and "measurements.csv" will also be generated according to the type of comprehensive optimization.

Alternatively, the multivariate optimization can also be finished by the combination of univariate optimizations, as shown in the second lines in Figs. 18(a)–18(d). In the case of SM (or CM) shown in Fig. 18(a) [Fig. 18(c)], one could first perform the state (control) optimization with QFI/$\text{Tr}(W\mathcal{F}^{-1})$ the objective function. Next, take the found optimal state
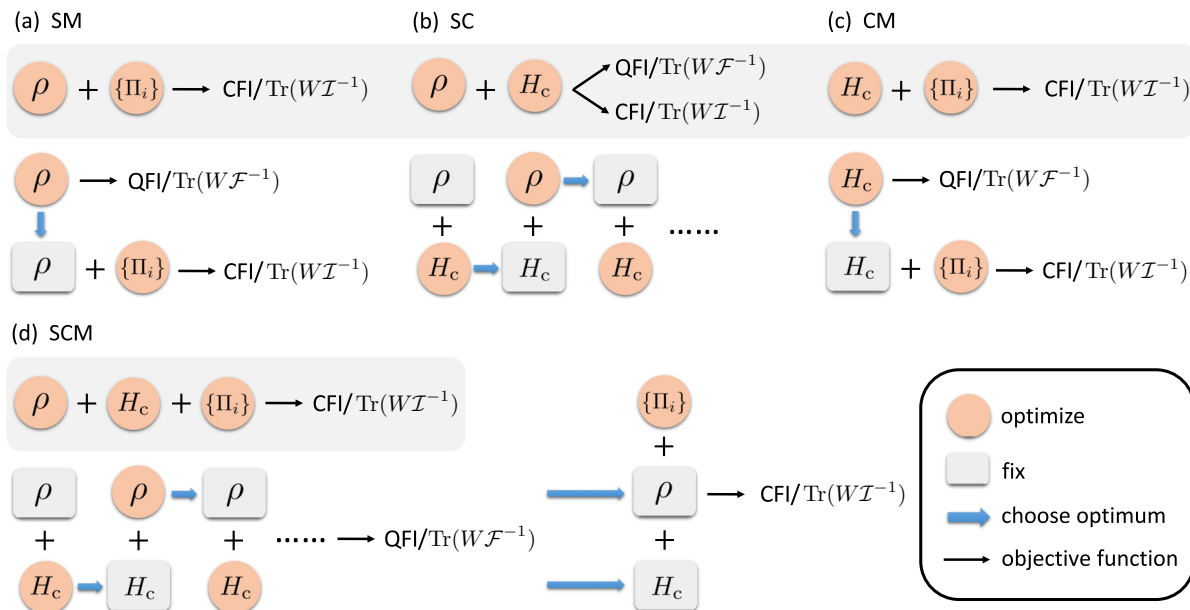
FIG. 18. Illustration of the comprehensive optimization (first lines with gray background) and combination of univariate optimizations (second lines) in four types of multivariate optimizations, including the optimizations of (a) the probe state and measurement (SM), (b) the probe state and control (SC), (c) control and measurement (CM), and (d) the probe state, control, and measurement (SCM).

(control) as the fixed input, and further optimize the measurement with $\mathrm{CFI}/\mathrm{Tr}(W\mathcal{I}^{-1})$ the objective function. If the optimized values of the $\mathrm{CFI}/\mathrm{Tr}(W\mathcal{I}^{-1})$ in the second process

TABLE V. Available methods for comprehensive optimization in QuanEstimation and corresponding default parameter settings. Notice that AD is only available when *com.SC()* is called.

| Algorithms | method= | **kwargs and default values | |
| --- | --- | --- | --- |
| PSO | "PSO" | "p_num" | 10 |
| | | "psi0" | [] |
| | | "ctrl0" | [] |
| | | "measurement0" | [] |
| | | "max_episode" | [1000,100] |
| | | "c0" | 1.0 |
| | | "c1" | 2.0 |
| | | "c2" | 2.0 |
| | | "seed" | 1234 |
| DE | "DE" | "p_num" | 10 |
| | | "psi0" | [] |
| | | "ctrl0" | [] |
| | | "measurement0" | [] |
| | | "max_episode" | 1000 |
| | | "c" | 1.0 |
| | | "cr" | 0.5 |
| | | "seed" | 1234 |
| AD (available for SC) | "AD" | "Adam" | False |
| | | "psi0" | [] |
| | | "ctrl0" | [] |
| | | "measurement0" | [] |
| | | "max_episode" | 300 |
| | | "epsilon" | 0.01 |
| | | "beta1" | 0.90 |
| | | "beta2" | 0.99 |

reaches the optimized values of the $\mathrm{QFI}/\mathrm{Tr}(W\mathcal{F}^{-1})$ in the first process, the entire scheme is then optimal. Things could be more complex in the multiparameter estimation due to the attainability problem. The existence of the gap between the optimized $\mathrm{Tr}(W\mathcal{I}^{-1})$ and $\mathrm{Tr}(W\mathcal{F}^{-1})$ does not necessarily mean the scheme is not optimal. Nevertheless, there is no doubt that a smaller gap always implies a better scheme at least in theory. In the case of SC, the state optimization and control optimization can be performed in turn with the optimal quantity found in the previous turn as the fixed input [Fig. 18(b)]. Same with the comprehensive optimization, both $\mathrm{QFI}/\mathrm{Tr}(W\mathcal{F}^{-1})$ and $\mathrm{CFI}/\mathrm{Tr}(W\mathcal{I}^{-1})$ can be taken as the objective function in this case. At last, in the case of SCM the combination strategy in SC could be performed first with $\mathrm{QFI}/\mathrm{Tr}(W\mathcal{F}^{-1})$ the objective function, and the measurement is further optimized with the found optimal state and control as the fixed input [Fig. 18(d)]. Same with the scenario of SM, if the optimized $\mathrm{CFI}/\mathrm{Tr}(W\mathcal{I}^{-1})$ obtained in the second process reaches the optimized $\mathrm{QFI}/\mathrm{Tr}(W\mathcal{F}^{-1})$ in the first process, the entire scheme is optimal.

*Example.* Now we provide some demonstrations on the comprehensive optimization with QuanEstimation and compare their performance with the combination strategy. First, consider a noncontrolled example with the single-qubit Hamiltonian $\omega\sigma_3/2$, which is a SM scenario. The dynamics is governed by Eq. (12) with decay rates $\gamma_-/\omega_{\mathrm{tr}} = 0$ and $\gamma_+/\omega_{\mathrm{tr}} = 0.1$. The target time $\omega_{\mathrm{tr}}T = 20$. In this case, the optimized values of $\sqrt{\omega_{\mathrm{tr}}T}\delta_{\mathrm{c}}\omega$ in the comprehensive optimization and combination strategy are both 0.608 (in the units of $\omega_{\mathrm{tr}}$, same below), equivalent to the optimal $\sqrt{\omega_{\mathrm{tr}}T}\delta_{\mathrm{q}}\omega$ obtained in the solely state optimization, indicating that the schemes found by both strategies are indeed optimal in theory. Next we invoke the controls described in Eq. (45). In the case
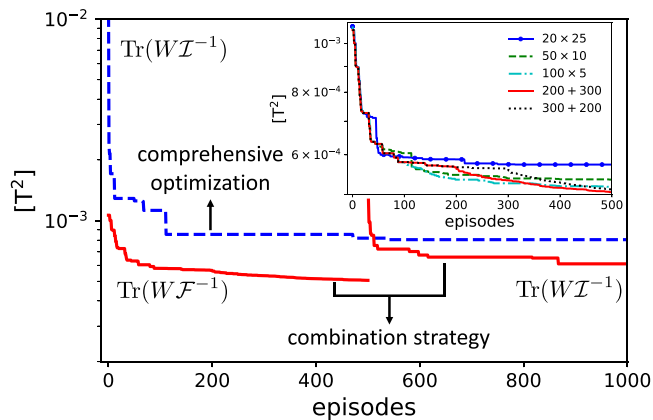
FIG. 19. Performance comparison between the comprehensive optimization and combination strategy in the multiparameter estimation in the case of SCM. The dashed-blue line represents the optimization of $\mathrm{Tr}(W\mathcal{I}^{-1})$ in the comprehensive optimization. The solid-red lines represent the optimization of $\mathrm{Tr}(W\mathcal{F}^{-1})$ in the SC (first 500 episodes) and that of $\mathrm{Tr}(W\mathcal{I}^{-1})$ in the measurement optimization (last 500 episodes) in the combination strategy. The inset shows the performance of different combination strategies in the SC part due to the episode number of each optimization. All the optimizations in the figure are finished by DE.

of SC, the optimized $\sqrt{\omega_{\mathrm{tr}}T}\delta_q\omega$ obtained in the combination strategy is 0.441, and that in the comprehensive optimization is 0.440. Furthermore, in the case of SCM, the optimized $\sqrt{\omega_{\mathrm{tr}}T}\delta_c\omega$ provided by the combination strategy is 0.441, equivalent to the optimal $\sqrt{\omega_{\mathrm{tr}}T}\delta_q\omega$ obtained in the SC, and that provided by the comprehensive optimization is 0.443. The performance of these two strategies basically coincide with each other in this example.

This equivalent performance may due to two facts: the example is simple and the QFI is attainable in theory. In the multiparameter estimation, these two strategies may show divergent performance as the QFIM is not always guaranteed to be attainable. For example, in the case of SCM, $\mathrm{Tr}(W\mathcal{F}^{-1})$ are first optimized in the SC. However, it is hard to say whether the optimal probe state and control for an unattainable $\mathrm{Tr}(W\mathcal{F}^{-1})$ can still provide a good $\mathrm{Tr}(W\mathcal{I}^{-1})$ and benefit the subsequent measurement optimization. To investigate it, we still take the nitrogen-vacancy center as an example. The free Hamiltonian, control Hamiltonian, and dynamics are described in Eqs. (63)–(65). The performance of comprehensive optimization and combination strategy in the SCM are shown in Fig. 19. The comprehensive optimization (dashed-blue line), which takes $\mathrm{Tr}(W\mathcal{I}^{-1})$ as the objective function, basically converges at around 110 episodes. The combination strategy (solid-red line) splits into two parts, the one in the first 500 episodes is the combination optimization of SC, and that in the last 500 episodes is the optimization of the measurement. The gap between these two lines is actually the gap between the optimal $\mathrm{Tr}(W\mathcal{F}^{-1})$ and the value of $\mathrm{Tr}(W\mathcal{I}^{-1})$ with a random measurement. In the SC part, the alternative optimizations of the probe state and control can be done in different ways due to the episode number of each optimization. As shown in the inset of Fig. 19, here we test several selections, including 20 episodes for each optimization (solid-circle-blue

line), 50 episodes for each optimization (dashed-green line), 100 episodes for each optimization (dash-dotted-cyan line), 200 episodes for state optimization and 300 episodes for control optimization (solid-red line), and 300 episodes for state optimization and 200 episodes for control optimization (dotted-black line). In these selections, the fourth one, 200 episodes for state optimization and 300 episodes for control optimization, shows the best performance at the end of the 500 episodes, and the corresponding optimal state and control are chosen for the subsequent measurement optimization. In this example, the final performance of the combination strategy is better than that of the simultaneous strategy, indicating that the unattainability of $\mathrm{Tr}(W\mathcal{F}^{-1})$ in the SC does not present negative effects on the final performance. However, this result does not mean the combination strategy is always better in general. In practice, the comparison of these two strategies might still be needed case by case in the scheme design.

## X. ADAPTIVE MEASUREMENT SCHEMES

Adaptive measurement is another common scenario in quantum parameter estimation. In this scenario, apart from the unknown parameters $\mathbf{x}$, the Hamiltonian also includes a set of tunable parameters $\mathbf{u}$. A typical case is that the tunable parameters are invoked by the same way with $\mathbf{x}$, resulting in the total Hamiltonian $H(\mathbf{x}+\mathbf{u})$. In the point estimation approach, the QFIM and CFIM computed at the true values of $\mathbf{x}$ may not always provide the practically achievable precision due to the fact that the actual working point may be slightly away from the true values. Hence, the tunable parameters $\mathbf{u}$ are invoked to let the Hamiltonian $H(\mathbf{x}+\mathbf{u})$ work at the optimal point $\mathbf{x}_{\mathrm{opt}}$. An obvious difficulty for the implementation of this scheme is that one actually does not known the true values in practice, which means $\mathbf{u}$ has to be given according to the estimated values $\hat{\mathbf{x}}$, and the entire scheme would only be useful when it is implemented adaptively. In the meantime, a preestimation of $\mathbf{x}$ is usually needed. The inaccuracy of $\hat{\mathbf{x}}$ would result in the inaccuracy of $\mathbf{u}$, and $\hat{\mathbf{x}}+\mathbf{u}$ is then inevitably far from $\mathbf{x}_{\mathrm{opt}}$, causing a lousy performance of the scheme. This scheme has been demonstrated by Berni *et al.* [126] in optical phase estimation with additional real-time feedback controls.

Now let us introduce in detail all steps required to implement this scheme. Consider the Hamiltonian $H(\mathbf{x})$ where $\mathbf{x}$ is restricted in a finite regime with a prior distribution $p(\mathbf{x})$. The first step is to find the optimal value $\mathbf{x}_{\mathrm{opt}}$ in this regime with respect to the minimum $\mathrm{Tr}(W\mathcal{I}^{-1})$ when the measurement is fixed. If the measurement can be altered flexibly in practice, $\mathbf{x}_{\mathrm{opt}}$, together with the corresponding optimal measurement, can be obtained with $\mathrm{Tr}(W\mathcal{F}^{-1})$ the objective function. Next, perform the preestimation via the Bayesian estimation with the fixed or optimal measurement and update the prior distribution with the posterior distribution in Eq. (21). When $p(\mathbf{x})$ has been updated to a reasonable narrow distribution, the tunable parameters $\mathbf{u}$ are then invoked into the system. In the $n$th round of this step, with the observed result $y^{(n)}$, the posterior distribution is obtained via the Bayes' rule as

$$p(\mathbf{x}, \mathbf{u}^{(n)}|y^{(n)}) = \frac{p(y^{(n)}|\mathbf{x}, \mathbf{u}^{(n)})p(\mathbf{x})}{\int p(y^{(n)}|\mathbf{x}, \mathbf{u}^{(n)})p(\mathbf{x})d\mathbf{x}}, \quad (73)$$

where $\mathbf{u}^{(n)}$ is obtained in the $(n-1)$th round. The estimated value $\hat{\mathbf{x}}^{(n)}$ can be obtained through the MAP,

$\hat{\mathbf{x}}^{(n)} = \mathrm{argmax}\, p(\mathbf{x}, \mathbf{u}^{(n)}|y^{(n)})$. The value of $\mathbf{u}$ used in the next round is obtained by $\mathbf{u}^{(n+1)} = \mathbf{x}_{\mathrm{opt}} - \hat{\mathbf{x}}^{(n)}$, and the prior distribution is also replaced by the current posterior distribution. This update method of $\mathbf{u}$ is referred to as the fixed optimal point method (FOP) in this paper. In QuanEstimation, the preestimation can be finished with the function *Bayes()* discussed in Sec. IV D, and the adaptive process can be executed with the code

```
apt = Adapt(x, p, rho0, method = "FOP",
            savefile = False, max_episode = 1000,
            eps = 1e-8)
apt.dynamics(tspan, H, dH, Hc = [], ctrl = [],
             decay = [], dyn_method = "expm")
apt.CFIM(M = [], W = [])
```

In the case that the parametrization process is described by the Kraus operators, replace *apt.dynamics()* with *apt.Kraus(K,dK)*. The inputs $x$ and $p$ are the same with those in *Bayes()*. The input $H$ is a list of matrices representing the Hamiltonian with respect to the values in $x$, and it is multidimensional in the multiparameter case. $dH$ is a (multidimensional) list with each entry also a list representing $\partial_{\mathbf{x}}H$ with respect to the values in $x$. In the case that specific functions of $H$ and $\partial_{\mathbf{x}}H$ can be provided, $H$ and $dH$ can be alternatively generated via the function *BayesInput()* discussed in Sec. III. In *apt.CFIM()*, $M$ is the input measurement and the default one is a set of SIC-POVM.

During the running of the code, three files "xout.csv", "y.csv", and "pout.csv" will be generated including the data of $\hat{\mathbf{x}}$, result $y$ in all rounds of iteration and final obtained $p(\mathbf{x})$. In the case that *savefile = True*, "pout.csv" contains the data of $p(\mathbf{x})$ in all rounds. If the choice of measurement is flexible in the experiment, before the invocation of *apt.CFIM()*, the optimal measurement with respect to $\mathbf{x}_{\mathrm{opt}}$ can be first obtained via calling $M = apt.Mopt(W = [])$. In the case that the users would like to run the preestimation with the optimal measurement, they can just call *apt.Mopt()* first and input the optimal measurement to *Bayes()* for the preestimation.

During the running of *apt.CFIM()*, the users should type the result $y$ obtained in practice on the screen and receive the values of $\mathbf{u}$ used for the next round of experiment. In the case that the users have already done the preestimation by themselves, they can directly use *Adapt()* without calling *Bayes()* first.

Apart from the FOP, $\mathbf{u}$ can also be updated via other strategies. One such choice is utilizing the optimization of the mutual information (MI), which is defined by

$$I(\mathbf{u}) = \int p(\mathbf{x}) \sum_y p(y|\mathbf{x}, \mathbf{u}) \log_2 \left[ \frac{p(y|\mathbf{x}, \mathbf{u})}{\int p(\mathbf{x})p(y|\mathbf{x}, \mathbf{u})d\mathbf{x}} \right] d\mathbf{x}. \tag{74}$$

In the $n$th round, the prior distribution $p(\mathbf{x})$ is updated via Eq. (73), and $\mathbf{u}^{(n+1)}$ is obtained by the equation $\mathbf{u}^{(n+1)} = \mathrm{argmax}\, I(\mathbf{u})$. In QuanEstimation, this method can be invoked by setting *method = "MI"* in *Adapt()*. Notice that in this method a good preestimation should let the prior distribution converges to zero at the boundary of the input $x$.

Let us still take the Hamiltonian in Eq. (22) as an example. The initial state is $|+\rangle$ and the target time $\omega_0 T = 1$
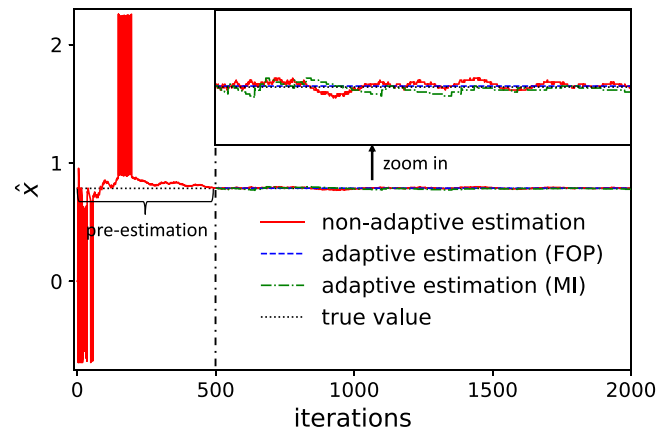


FIG. 20. Performance comparison between the adaptive schemes realized by FOP (dashed-blue line) and MI (dash-dotted-green line), and the nonadaptive schemes (solid-red line). The adaptive measurement starts after 500 rounds of preestimation. The nonadaptive scheme is a full Bayesian estimation. The dotted-black line represents the true value.

(Planck units are applied). The prior distribution is uniform in the regime $(-\pi/4, 3\pi/4)$. In this regime, zero is an optimal point and is chosen to be $x_{\mathrm{opt}}$. The measurement is $\{|+\rangle\langle+|, |-\rangle\langle-|\}$. The results are simulated by generating random values in the regime $[0,1]$. When it is smaller (larger) than $p(+|x)$, the posterior distribution is calculated with $p(+|x)$ $[p(-|x)]$. As shown in Fig. 20, after 500 rounds of preestimation, the adaptive schemes realized by FOP (dashed-blue line) and MI (dash-dotted-green line) indeed show better performance, namely, smaller variance, compared to the nonadaptive scheme (solid-red line), which is fully finished by the Bayesian estimation. Notice that this figure is only a one-time simulation of the experiment. The performance may be different when the results are different.

Another famous adaptive scheme is the online adaptive phase estimation, proposed by Berry *et al.* [127,128], in the scenario of Mach-Zehnder interferometer (MZI). In this scheme, after reading the result $y^{(n)}$ in the $n$th round, the value of the tunable phase $\Phi_{n+1}$ or phase difference $\Delta\Phi_{n+1}$ is generated. The relation between $\Phi_{n+1}$ and $\Delta\Phi_{n+1}$ can be taken as $\Phi_{n+1} = \Phi_n - (-1)^{y^{(n)}}\Delta\Phi_{n+1}$. Hentschel and Sanders [72,73] further provided an off-line strategy with PSO, and the optimization methods are further extended to DE [75] and genetic algorithm [129] in recent years. Apart from the original references, details of this scheme can also be found in a recent review [17]. In QuanEstimation, this scheme can be executed by the code

```
apt = Adapt_MZI(x, p, rho0)
apt.general()
apt.online(target = "sharpness", output = "phi")
```

The input *rho0* is a matrix representing the probe state. The output can be tuned between $\Phi$ and $\Delta\Phi$ by setting *output = "phi"* or *output = "dphi"* in *apt.online()* in the demonstrating code. *target = "sharpness"* means the tunable phase is obtained by the maximization of the sharpness function. The specific formula of the sharpness function can be found in Refs. [17,72,73,127,128]. Alternatively, the sharpness

function can also be replaced by the mutual information in Eq. (74) via setting *target = "MI"*, which has been both theoretically and experimentally discussed by DiMario and Becerra in 2020 [130].

The off-line strategy can also be executed by replacing *apt.online()* with the code

```
apt.offline(target = "sharpness", method = "DE",
            **kwargs)
```

PSO is also available here (*method = "PSO"*). When the entire program is finished, a file named "xout.csv" including the data of output in all rounds will be generated. In the case of online scheme, an additional file "y.csv" including the result *y* in all rounds will also be generated. The design of *apt.general()* here is to give us a space for the further inclusion of the adaptive phase estimation in other optical scenarios such as the SU(1,1) interferometers.

## XI. SUMMARY

In this paper, we present a new open-source toolkit, QuanEstimation, for the design of optimal schemes in the quantum parameter estimation. The source of the package, as well as the demonstrating code for the calculation of all examples discussed in this paper, can be download in GitHub [131] and the documentation is in Ref. [132]. This package is based on both platforms of Python and Julia. The combined structure is to guarantee the calculation efficiency of Julia is fully utilized, and in the meantime, the people who have no knowledge of Julia would have no obstacle in using this package. In the meantime, a full Julia version of the package is also available in GitHub [133], which is suitable for those familiar with Julia. QuanEstimation includes several well-studied metrological tools in quantum parameter estimation, such as the various types of Cramér-Rao bounds and their quantum correspondences, quantum Ziv-Zakai bound, and Bayesian estimation. To perform the scheme design, QuanEstimation can execute the optimizations of the probe state, control, measurement, and the comprehensive optimizations, namely, the simultaneous optimizations among them. General adaptive measurement schemes as well as the adaptive phase estimation can also be performed with this toolkit.

QuanEstimation is suitable for many practical quantum systems, especially those with finite-dimensional Hilbert spaces, such as the trapped ions, nitrogen-vacancy centers, and quantum circuits. Therefore, it is not only useful for the theorists working in the field of quantum parameter estimation, but could also be particularly useful for the experimentalists who are not familiar with the theories in this field yet intend to utilize them to design experimental schemes. More functions and features will be constantly input into the package and the calculation efficiency for certain specific scenarios will be further improved in the future. Moreover, the calculations in QuanEstimation are majorly based on the density matrices, which may cause inefficiency when the dimension of Hilbert space is large. More technologies targeting at the many-body systems, such as the sparse matrices and matrix product states, will be further involved in the package in the future. We believe that there is a good chance that this package would become a common toolkit in the field of quantum metrology for the numerical calculations and scheme designs.

## ACKNOWLEDGMENTS

[1] V. Giovannetti, S. Lloyd, and L. Maccone, Quantum-enhanced measurements: Beating the standard quantum limit, Science **306**, 1330 (2004).

[2] V. Giovannetti, S. Lloyd, and L. Maccone, Advances in quantum metrology, Nat. Photon. **5**, 222 (2011).

[3] C. L. Degen, F. Reinhard, and P. Cappellaro, Quantum sensing, Rev. Mod. Phys. **89**, 035002 (2017).

[4] D. Braun, G. Adesso, F. Benatti, R. Floreanini, U. Marzolino, M. W. Mitchell, and S. Pirandola, Quantum-enhanced measurements without entanglement, Rev. Mod. Phys. **90**, 035006 (2018).

[5] L. Pezzè, A. Smerzi, M. K. Oberthaler, R. Schmied, and P. Treutlein, Quantum metrology with nonclassical states of atomic ensembles, Rev. Mod. Phys. **90**, 035005 (2018).

[6] J. Aasi, J. Abadie, B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, C. Adams, T. Adams, P. Addesso, R. X. Adhikari *et al.*, Enhanced sensitivity of the LIGO gravitational wave detector by using squeezed states of light, Nat. Photon. **7**, 613 (2013).

[7] K. M. Backes, D. A. Palken, S. A. Kenany, B. M. Brubaker, S. B. Cahn, A. Droster, G. C. Hilton, S. Ghosh, H. Jackson, S. K. Lamoreaux *et al.*, A quantum enhanced search for dark matter axions, Nature (London) **590**, 238 (2021).

[8] M. Jiang, H. Su, A. Garcon, X. Peng, and D. Budker, Search for axion-like dark matter with spin-based amplifiers, Nat. Phys. **17**, 1402 (2021).

[9] C. W. Helstrom, Minimum mean-squared error of estimates in quantum statistics, Phys. Lett. A **25**, 101 (1967).

[10] M. G. A. Paris, Quantum estimation for quantum technology, Int. J. Quantum Inf. **07**, 125 (2009).

[11] G. Tóth and I. Apellaniz, Quantum metrology from a quantum information science perspective, J. Phys. A: Math. Theor. **47**, 424006 (2014).

[12] M. Szczykulska, T. Baumgratz, and A. Datta, Multi-parameter quantum metrology, Adv. Phys. X **1**, 621 (2016).

[13] J. Liu, H. Yuan, X.-M. Lu, and X. Wang, Quantum Fisher information matrix and multiparameter estimation, J. Phys. A: Math. Theor. **53**, 023001 (2020).

[14] R. Demkowicz-Dobrzański, W. Górecki, and M. Guţă, Multi-parameter estimation beyond Quantum Fisher Information, J. Phys. A: Math. Theor. **53**, 363001 (2020).

[15] J. S. Sidhu and P. Kok, Geometric perspective on quantum parameter estimation, AVS Quantum Sci. **2**, 014701 (2020).

[16] F. Albarelli, M. Barbieri, M. G. Genoni, and I. Gianani, A perspective on multiparameter quantum metrology: From theoretical tools to applications in quantum imaging, Phys. Lett. A **384**, 126311 (2020).

[17] J. Liu, M. Zhang, H. Chen, L. Wang, and H. Yuan, Optimal scheme for quantum metrology, Adv. Quantum Technol. **5**, 2100080 (2022).

[18] V. Erol, F. Ozaydin, and A. Altintas, Analysis of entanglement measures and LOCC maximized quantum Fisher information of general two qubit systems, Sci. Rep. **4**, 5422 (2014).

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

[20] J. R. Johansson, P. D. Nation, and F. Nori, QuTiP 2: A Python framework for the dynamics of open quantum systems, Comput. Phys. Commun. **184**, 1234 (2013).

[21] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework, Phys. Rev. A **84**, 022305 (2011).

[22] H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore, and I. Kuprov, Spinach-A software library for simulation of spin dynamics in large spin systems, J. Magn. Reson. **208**, 179 (2011).

[23] M. H. Goerz, D. Basilewitsch, F. Gago-Encinas, M. G. Krauss, K. P. Horn, D. M. Reich, and C. P. Koch, Krotov: A Python implementation of Krotov's method for quantum optimal control, SciPost Phys. **7**, 080 (2019).

[24] S. Günther, N. A. Petersson, and J. L. Dubois, Quandary: An open-source C++ package for high-performance optimal control of open quantum systems, arXiv:2110.10310.

[25] C. W. Groth, M. Wimmer, A. R. Akhmerov, and X. Waintal, Kwant: A software package for quantum transport, New J. Phys. **16**, 063065 (2014).

[26] D. S. Steiger, T. Häner, and M. Troyer, ProjectQ: An open source software framework for quantum computing, Quantum **2**, 49 (2018).

[27] K. Chabuda and R. Demkowicz-Dobrzański, TNQMetro: Tensor-network based package for efficient quantum metrology computations, Comput. Phys. Commun. **274**, 108282 (2022).

[28] C. W. Helstrom, *Quantum Detection and Estimation Theory* (Academic, New York, 1976).

[29] A. S. Holevo, *Probabilistic and Statistical Aspects of Quantum Theory* (North-Holland, Amsterdam, 1982).

[30] H. P. Yuen and M. Lax, Multiple-parameter quantum estimation and measurement of nonselfadjoint observables, IEEE Trans. Inf. Theory **19**, 740 (1973).

[31] G. Gour and A. Kalev, Construction of all general symmetric informationally complete measurements, J. Phys. A: Math. Theor. **47**, 335302 (2014).

[32] C. A. Fuchs, M. A. Hoang, and B. C. Stacey, The SIC question: History and state of play, Axioms **6**, 21 (2017).

[33] J. M. Renes, R. Blume-Kohout, A. J. Scott, and C. M. Caves, Symmetric informationally complete quantum measurements, J. Math. Phys. **45**, 2171 (2004).

[34] A. J. Scott and M. Grassl, Symmetric informationally complete positive-operator-valued measures: A new computer study, J. Math. Phys. **51**, 042203 (2010).

[35] A. S. Holevo, Statistical decision theory for quantum systems, J. Multivariate Anal. **3**, 337 (1973).

[36] M. Hayashi and K. Matsumoto, Asymptotic performance of optimal state estimation in qubit system, J. Math. Phys. **49**, 102101 (2008).

[37] H. Nagaoka, A new approach to Cramér-Rao bounds for quantum state estimation, in *Asymptotic Theory Of Quantum Statistical Inference: Selected Papers* (World Scientific, Singapore, 2005), pp. 100–112.

[38] F. Albarelli, J. F. Friel, and A. Datta, Evaluating the Holevo Cramér-Rao Bound for Multiparameter Quantum Metrology, Phys. Rev. Lett. **123**, 200503 (2019).

[39] S. Diamond and S. Boyd, CVXPY: A python-embedded modeling language for convex optimization, J. Mach. Learn. Res. **17**, 1 (2016).

[40] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, A rewriting system for convex optimization problems, J. Control Decis. **5**, 42 (2018).

[41] M. Udell, K. Mohan, D. Zeng, J. Hong, S. Diamond, and S. Boyd, Convex Optimization in Julia, in *Proceedings of the 2014 First Workshop for High Performance Technical Computing in Dynamic Languages* (IEEE, New Orleans, Louisiana 2014), pp. 18–28.

[42] M. Hayashi, *Asymptotic Theory of Quantum Statistical Inference: Selected Papers* (World Scientific, Singapore, 2005).

[43] L. O. Conlon, J. Suzuki, P. K. Lam, and S. M. Assad, Efficient computation of the Nagaoka-Hayashi bound for multiparameter estimation with separable measurements, npj Quantum Inf. **7**, 110 (2021).

[44] C. P. Robert, *The Bayesian Choice* (Springer, Berlin, 2007).

[45] J. Liu and H. Yuan, Valid lower bound for all estimators in quantum parameter estimation, New J. Phys. **18**, 093009 (2016).

[46] H. L. Van Trees, *Detection, Estimation, and Modulation Theory: Part I* (Wiley, New York, 1968).

[47] M. Tsang, H. M. Wiseman, and C. M. Caves, Fundamental Quantum Limit to Waveform Estimation, Phys. Rev. Lett. **106**, 090401 (2011).

[48] J. Ziv and M. Zakai, Some lower bounds on signal parameter estimation, IEEE Trans. Inf. Theory **15**, 386 (1969).

[49] K. L. Bell, Y. Steinberg, Y. Ephraim, and H. L. Van Trees, Extended Ziv-Zakai lower bound for vector parameter estimation, IEEE Trans. Inf. Theory **43**, 624 (1997).

[50] M. Tsang, Ziv-Zakai Error Bounds for Quantum Parameter Estimation, Phys. Rev. Lett. **108**, 230401 (2012).

[51] D. W. Berry, M. Tsang, M. J. W. Hall, and H. M. Wiseman, Quantum Bell-Ziv-Zakai Bounds and Heisenberg Limits for Waveform Estimation, Phys. Rev. X **5**, 031018 (2015).

[52] M. Kitagawa and M. Ueda, Squeezed spin states, Phys. Rev. A **47**, 5138 (1993).

[53] D. J. Wineland, J. J. Bollinger, W. M. Itano, F. L. Moore, and D. J. Heinzen, Spin squeezing and reduced quantum noise in spectroscopy, Phys. Rev. A **46**, R6797(R) (1992).

[54] N. Khaneja, T. Reiss, C. Hehlet, T. Schulte-Herbruggen, and S. J. Glaser, Optimal control of coupled spin dynamics: Design of NMR pulse sequences by gradient ascent algorithms, J. Magn. Reson. **172**, 296 (2005).

[55] J. Liu and H. Yuan, Quantum parameter estimation with optimal control, Phys. Rev. A **96**, 012117 (2017).

[56] J. Liu and H. Yuan, Control-enhanced multiparameter quantum estimation, Phys. Rev. A **96**, 042114 (2017).

[57] D. M. Reich, G. Gualdi, and C. P. Koch, Minimum number of input states required for quantum gate characterization, Phys. Rev. A **88**, 042309 (2013).

[58] D. M. Reich, G. Gualdi, and C. P. Koch, Optimal Strategies for Estimating the Average Fidelity of Quantum Gates, Phys. Rev. Lett. **111**, 200401 (2013).

[59] M. H. Goerz, D. M. Reich, and C. P. Koch, Optimal control theory for a unitary operation under dissipative evolution, New J. Phys. **16**, 055012 (2014).

[60] H. Xu, J. Li, L. Liu, Y. Wang, H. Yuan, and X. Wang, Generalizable control for quantum parameter estimation through reinforcement learning, npj Quantum Inf. **5**, 82 (2019).

[61] H. Xu, L. Wang, H. Yuan, and X. Wang, Generalizable control for multiparameter quantum metrology, Phys. Rev. A **103**, 042615 (2021).

[62] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[63] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 1 (2018).

[64] Y. Song, J. Li, Y.-J. Hai, Q. Guo, and X.-H. Deng, Optimizing quantum control pulses with complex constraints and few variables through autodifferentiation, Phys. Rev. A **105**, 012616 (2022).

[65] M. Innes, Don't Unroll Adjoint: Differentiating SSA-form programs, arXiv:1810.07951.

[66] D. Šafránek, Simple expression for the quantum Fisher information matrix, Phys. Rev. A **97**, 042322 (2018).

[67] J. Kennedy and R. Eberhar, Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks **4**, 1942 (1995).

[68] R. C. Eberhart and Y. Shi, Particle swarm optimization: developments, applications and resources, in *Proc. 2001 Congr. Evol. Comput.* (IEEE Cat. No.01TH8546) (IEEE, Piscataway, NJ, 2001), pp. 81-86.

[69] Y. Michimura, K. Komori, A. Nishizawa, H. Takeda, K. Nagano, Y. Enomoto, K. Hayama, K. Somiya, and M. Ando, Particle swarm optimization of the sensitivity of a cryogenic gravitational wave detector, Phys. Rev. D **97**, 122003 (2018).

[70] M. P. V. Stenberg, O. Köhn, and F. K. Wilhelm, Characterization of decohering quantum systems: Machine learning approach, Phys. Rev. A **93**, 012122 (2016).

[71] Y. Wang, J. Lv, L. Zhu, and Y. Ma, Crystal structure prediction via particle-swarm optimization, Phys. Rev. B **82**, 094116 (2010).

[72] A. Hentschel and B. C. Sanders, Machine Learning for Precise Quantum Measurement, Phys. Rev. Lett. **104**, 063603 (2010).

[73] A. Hentschel and B. C. Sanders, Efficient Algorithm for Optimizing Adaptive Quantum Metrology Processes, Phys. Rev. Lett. **107**, 233601 (2011).

[74] R. Storn and K. Price, Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. **11**, 341 (1997).

[75] N. B. Lovett, C. Crosnier, M. Perarnau-Llobet, and B. C. Sanders, Differential Evolution for Many-Particle Adaptive Quantum Metrology, Phys. Rev. Lett. **110**, 220501 (2013).

[76] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, Learning in quantum control: High-dimensional global optimization for noisy quantum dynamics, Neurocomputing **268**, 116 (2017).

[77] X. Yang, J. Li, and X. Peng, An improved differential evolution algorithm for learning high-fidelity quantum controls, Sci. Bull. **64**, 1402 (2019).

[78] X. Yang, C. Arenz, I. Pelczer, Q.-M. Chen, R.-B. Wu, X. Peng, and H. Rabitz, Assessing three closed-loop learning algorithms by searching for high-quality quantum control pulses, Phys. Rev. A **102**, 062605 (2020).

[79] H. Ma, D. Dong, C.-C. Shu, Z. Zhu, and C. Chen, Quantum learning control using differential evolution with equally-mixed strategies, Control Theory Technol. **15**, 226 (2017).

[80] D. Dong, X. Xing, H. Ma, C. Chen, Z. Liu, and H. Rabitz, Learning-based quantum robust control: Algorithm, applications, and experiments, IEEE Trans. Cybern. **50**, 3581 (2019).

[81] V. Kachitvichyanukul, Comparison of three evolutionary algorithms: GA, PSO, and DE, Ind. Eng. Manag. Syst. **11**, 215 (2012).

[82] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, arXiv:1509.02971.

[83] Q.-S. Tan, M. Zhang, Y. Chen, J.-Q. Liao, and J. Liu, Generation and storage of spin squeezing via learning-assisted optimal control, Phys. Rev. A **103**, 032601 (2021).

[84] I. Schwartz, J. Scheuer, B. Tratzmiller, S. Müller, Q. Chen, I. Dhand, Z.-Y. Wang, C. Müller, B. Naydenov, F. Jelezko, and M. B. Plenio, Robust optical polarization of nuclear spin baths using Hamiltonian engineering of nitrogen-vacancy center quantum dynamics, Sci. Adv. **4**, eaat8978 (2018).

[85] P. Rembold, N. Oshnik, M. M. Müller, S. Montangero, T. Calarco, and E. Neu, Introduction to quantum optimal control for quantum sensing with nitrogen-vacancy centers in diamond, AVS Quantum Sci. **2**, 024701 (2020).

[86] J. F. Barry, J. M. Schloss, E. Bauch, M. J. Turner, C. A. Hart, L. M. Pham, and R. L. Walsworth, Sensitivity optimization for NV-diamond magnetometry, Rev. Mod. Phys. **92**, 015004 (2020).

[87] S. Felton, B. L. Cann, A. M. Edmonds, S. Liggins, R. J. Cruddace, M. E. Newton, D. Fisher, and J. M. Baker, Electron paramagnetic resonance studies of nitrogen interstitial defects in diamond, J. Phys.: Condens. Matter **21**, 364212 (2009).

[88] C. M. Caves, Quantum-mechanical noise in an interferometer, Phys. Rev. D **23**, 1693 (1981).

[89] J. Liu, X. X. Jing, and X. Wang, Phase-matching condition for enhancement of phase sensitivity in quantum metrology, Phys. Rev. A **88**, 042316 (2013).

[90] M. Jarzyna and R. Demkowicz-Dobrzański, Quantum interferometry with and without an external phase reference, Phys. Rev. A **85**, 011801(R) (2012).

[91] M. D. Lang and C. M. Caves, Optimal Quantum-Enhanced Interferometry Using a Laser Power Source, Phys. Rev. Lett. **111**, 173601 (2013).

[92] M. D. Lang and C. M. Caves, Optimal quantum-enhanced interferometry, Phys. Rev. A **90**, 025802 (2014).

[93] K. Modi, H. Cable, M. Williamson, and V. Vedral, Quantum Correlations in Mixed-State Metrology, Phys. Rev. X **1**, 021022 (2011).

[94] L. J. Fiderer, J. M. E. Fraïsse, and D. Braun, Maximal Quantum Fisher Information for Mixed States, Phys. Rev. Lett. **123**, 250502 (2019).

[95] A. Monras, Optimal phase measurements with pure Gaussian states, Phys. Rev. A **73**, 033821 (2006).

[96] D. Šafránek and I. Fuentes, Optimal probe states for the estimation of Gaussian unitary channels, Phys. Rev. A **94**, 062313 (2016).

[97] S. Knysh, E. H. Chen, and G. A. Durkin, True limits to precision via unique quantum probe, arXiv:1402.0495.

[98] A. Fujiwara, Quantum channel identification problem, Phys. Rev. A **63**, 042304 (2001).

[99] D. Šafránek, Estimation of Gaussian quantum states, J. Phys. A: Math. Theor. **52**, 035304 (2019).

[100] K. F. Pál, G. Tóth, E. Bene, and T. Vértesi, Bound entangled singlet-like states for quantum metrology, Phys. Rev. Res. **3**, 023101 (2021).

[101] R. Trényi, Á. Lukács, P. Horodecki, R. Horodecki, T. Vértesi, and G. Tóth, Multicopy metrology with many-particle quantum states, arXiv:2203.05538.

[102] U. Dorner, R. Demkowicz-Dobrzański, B. J. Smith, J. S. Lundeen, W. Wasilewski, K. Banaszek, and I. A. Walmsley, Optimal Quantum Phase Estimation, Phys. Rev. Lett. **102**, 040403 (2009).

[103] R. Demkowicz-Dobrzański, U. Dorner, B. J. Smith, J. S. Lundeen, W. Wasilewski, K. Banaszek, and I. A. Walmsley, Quantum phase estimation with lossy interferometers, Phys. Rev. A **80**, 013825 (2009).

[104] A. Forsgren, P. E. Gill, and M. H. Wright, Interior methods for nonlinear optimization, SIAM Rev. **44**, 525 (2002).

[105] L. Maccone and G. De Cillis, Robust strategies for lossy quantum interferometry, Phys. Rev. A **79**, 023812 (2009).

[106] S. Knysh, V. N. Smelyanskiy, and G. A. Durkin, Scaling laws for precision in quantum interferometry and the bifurcation landscape of the optimal state, Phys. Rev. A **83**, 021804(R) (2011).

[107] H. Yuan and C.-H. F. Fung, Quantum metrology matrix, Phys. Rev. A **96**, 012310 (2017).

[108] G. Tóth and T. Vértesi, Quantum States with a Positive Partial Transpose are Useful for Metrology, Phys. Rev. Lett. **120**, 020506 (2018).

[109] G. Tóth, T. Vértesi, P. Horodecki, and R. Horodecki, Activating Hidden Metrological Usefulness, Phys. Rev. Lett. **125**, 020402 (2020).

[110] Á. Lukács, R. Trényi, T. Vértesi, and G. Tóth, Optimizing local Hamiltonians for the best metrological performance, arXiv:2206.02820.

[111] K. Chabuda, J. Dziarmaga, T. J. Osborne, and R. Demkowicz-Dobrzański, Tensor-network approach for quantum metrology in many-body quantum systems, Nat. Commun. **11**, 250 (2020).

[112] F. Fröwis, M. Skotiniotis, B. Kraus, and W. Dür, Optimal quantum states for frequency estimation, New J. Phys. **16**, 083010 (2014).

[113] P. A. Knott, A search algorithm for quantum state engineering and metrology, New J. Phys. **18**, 073033 (2016).

[114] D. Basilewitsch, H. Yuan, and C. P. Koch, Optimally controlled quantum discrimination and estimation, Phys. Rev. Res. **2**, 033396 (2020).

[115] A. Larrouy, S. Patsch, R. Richaud, J.-M. Raimond, M. Brune, C. P. Koch, and S. Gleyzes, Fast Navigation in a Large Hilbert Space Using Quantum Optimal Control, Phys. Rev. X **10**, 021058 (2020).

[116] J. A. Nelder and R. Mead, A simplex method for function minimization, Comput. J. **7**, 308 (1965).

[117] R. Demkowicz-Dobrzański, Optimal phase estimation with arbitrary a priori knowledge, Phys. Rev. A **83**, 061802(R) (2011).

[118] K. Macieszczak, M. Fraas, and R. Demkowicz-Dobrzański, Bayesian quantum frequency estimation in presence of collective dephasing, New J. Phys. **16**, 113002 (2014).

[119] K. Macieszczak, Quantum Fisher Information: Variational principle and simple iterative algorithm for its efficient computation, arXiv:1312.1356.

[120] H. J. Lipkin, N. Meshkov, and A. J. Glick, Validity of many-body approximation methods for a solvable model: (I). Exact solutions and perturbation theory, Nucl. Phys. **62**, 188 (1965).

[121] J. Ma, X. Wang, C. P. Sun, and F. Nori, Quantum spin squeezing, Phys. Rep. **509**, 89 (2011).

[122] M. Yu, D. Li, J. Wang, Y. Chu, P. Yang, M. Gong, N. Goldman, and J. Cai, Experimental estimation of the quantum Fisher information from randomized measurements, Phys. Rev. Res. **3**, 043122 (2021).

[123] A. Rath, C. Branciard, A. Minguzzi, and B. Vermersch, Quantum Fisher Information from Randomized Measurements, Phys. Rev. Lett. **127**, 260501 (2021).

[124] A. Zhang, J. Xie, H. Xu, K. Zheng, H. Zhang, Y.-T. Poon, V. Vedral, and L. Zhang, Experimental Self-Characterization of Quantum Measurements, Phys. Rev. Lett. **124**, 040402 (2020).

[125] L. Xu, H. Xu, T. Jiang, F. Xu, K. Zheng, B. Wang, A. Zhang, and L. Zhang, Direct Characterization of Quantum Measurements Using Weak Values, Phys. Rev. Lett. **127**, 180401 (2021).

[126] A. A. Berni, T. Gehring, B. M. Nielsen, V. Händchen, M. G. A. Paris, and U. L. Andersen, *Ab initio* quantum-enhanced optical phase estimation using real-time feedback control, Nat. Photon. **9**, 577 (2015).

[127] D. W. Berry and H. M. Wiseman, Optimal States and Almost Optimal Adaptive Measurements for Quantum Interferometry, Phys. Rev. Lett. **85**, 5098 (2000).

[128] D. W. Berry, H. M. Wiseman, and J. K. Breslin, Optimal input states and feedback for interferometric phase estimation, Phys. Rev. A **63**, 053804 (2001).

[129] K. Rambhatla, S. E. D'Aurelio, M. Valeri, E. Polino, N. Spagnolo, and F. Sciarrino, Adaptive phase estimation through a genetic algorithm, Phys. Rev. Res. **2**, 033078 (2020).

[130] M. T. DiMario and F. E. Becerra, Single-Shot Non-Gaussian Measurements for Optical Phase Estimation, Phys. Rev. Lett. **125**, 120505 (2020).

[131] https://github.com/QuanEstimation/QuanEstimation

[132] https://quanestimation.github.io/QuanEstimation/

[133] https://github.com/QuanEstimation/QuanEstimation.jl