Extendible C++ Application in Photonic Technologies based on Parallel Computing

Tigran, Gevorgyan
Institute for Physical Research, National Academy of Sciences, Ashtarak, Armenia
e-mail: t_gevorgyan@ysu.am

Anna, Shahinyan
Yerevan State University, Yerevan, Armenia
e-mail: anna_shahinyan@ysu.am

Gagik, Kryuchkyan
Yerevan State University, Yerevan, Armenia
e-mail: kryuchkyan@.ysu.am

ABSTRACT

We provide an application for numerical simulations and modelling of complex quantum systems in the presence of dissipation and decoherence in area of Quantum Optics. The basic idea of framework is to allow users to build arbitrarily complex interacting quantum systems from elementary free subsystems and interactions between them, and simulate their time evolution with a number of available time-evolution drivers. It is very sensitive to performance both in terms of computer resources and coding/design. In the latter aspect the goal, as always in software design, is to create maximally reusable code. The application is extendible for C++ programmers and is user-friendly, it does not require any programming knowledge as has a graphical user interface. The framework provides wide range of applications in quantum optics. It can be used for numerical calculation density matrix of the system, quasidistributions, Poincaré section, and fidelity both for pure and mixed states. Standard algebraic notation is used to build operators and to perform arithmetic operations on operators and states. States can be represented in different basis (Fock, coherent, etc.). The package is available also for clusters.

1. INTRODUCTION

In last twenty years quantum physics gained technological interest but theoretical science is not always able to answer questions that are possed by engineers as well as analytical solutions are not always exist. But the technological progress of last years in computer science opened a new area in physics: modelling of physical processes. Computers development till super-computers and clusters gets possible to solve problems numerically that were not possible to solve before. Theoretical results become more realistic with physical process modelling. In this paper, we provide a new application that gets possible to solve problems for quantum systems that are constructed from bosons, including laser and photonic processes in nonlinear media in the presence of quantum noise. There are some papers that solve the same problems [1], [2], [3], [4]. All of them are based on the numerical method of quantum trajectories for calculation of the master equation for a density matrix and require somehow programming knowledge. In this paper we essentially expand these results by adding the application as it has user-friendly graphical user interface on one side and we tried to solve not only master equation but also provide some other utilities of Quantum Optics to investigate physical systems like Poincaré section, quasi-distributions, fidelity on the other side. The only C++ knowledge is required if user wants to extend the application field of the program. There are provided some extension facilities, like input file format modification, adding new state definition for fidelity calculation. Behavior of physical objects can be explained and predicted with system of differential equations. In recent years quantum physics has taken on special significance and wide applications and it is no longer possible to neglect the environment interaction: dissipation and decoherence. These are treated as open systems that cannot be described by a Hilbert-space vector $|\Psi\rangle$ which evaluation is described by Schrodinger equation, but by a density matrix which time evaluation is described by master equation. In general the master equation has no analytical solution for arbitrary evaluation times. One of widely used approximations is Markovian dynamics described in terms of Lindblad master equation for the reduced density matrix ρ

$$\frac{d\rho}{dt} = \frac{-i}{\hbar}[H,\rho] + \sum_{i=1,2}\left(L_i\rho L_i^+ - \frac{1}{2}L_i^+L_i\rho - \frac{1}{2}\rho L_i^+L_i\right), \quad (1)$$

where H is the Hamiltonian of the system and L_i are Lindblad operators representing the environment interaction. The numerical solution of the Eq. 1 is also hard because of limited computer resources as if state requires D basis vectors in Hilbert space for representation, the density matrix will require D^2-1 basis. This problem is overcame by unravelling the density operator evaluation into quantum trajectories

$$\rho(t) = M(|\psi_\xi(t)\rangle\langle\psi_\xi(t)|) = \lim_{N\to\infty}\frac{1}{N}\sum_{\xi}^{N}|\psi_\xi(t)\rangle\langle\psi_\xi(t)| \quad (2)$$

In the current implementation quantum state diffusion (QSD) method is used [5] that is based on the stochastic equation for the state $|\psi_\xi(t)\rangle$ that involves both Hamiltonian and the Lindblad operators. We calculate the density operator using an expansion of the state vector $|\psi_\xi\rangle$ in a truncated basis of Fock's number states of a harmonic oscillator (photonic states)

$$|\psi_\xi(t)\rangle = \sum_n a_n^\xi(t)|n\rangle. \quad (3)$$

The library corresponding to QSD method has been used to investigate quantum dissipative chaos [6], stochastic resonance [7], quantum-to-classical transition [8], and long-lived quantum interference [9]. Electromagnetic field states are well described in phase space, because due to the Heinsenberg uncertainty principle it is impossible to have a point in phase space. Such description is usually done within framework of the quasidistributions that are measurable physical quantities which are fully calculated via density matrix. There is a number of quasidistributions that describe system in phase space [10]. We provide calculations of the Wigner (Eq. 4),that can be measure through the tomography mea-

surement and Husimi (Eq. 5) function

$$W(\alpha) = \frac{2}{\pi^2} e^{-2|\alpha|^2} \int d^2\beta \langle -\beta|\rho|\beta\rangle e^{-2(\beta\alpha^* - \beta^*\alpha)}. \quad (4)$$

$$Q(\alpha) = \frac{1}{\pi}\langle\alpha|\rho|\alpha\rangle \quad (5)$$

In these formulas α, β are coherent states, while ρ describes density matrix. Husimi function is the density matrix mean in coherent state, while Wigner function is the Fourier transformation of density matrix. These distributions have a wide application in quantum physics. We have chosen Wigner function for detailed calculations that has negative values for pure quantum states and it lets to obtain probability distribution for one of the conjugate variables. Note, that the tomography measurement lets to reconstruct state from Wigner function. The package also has mechanisms for calculation of the fidelity that is closeness of two quantum density operators. It was first introduced in [11] and now is widely used in quantum optics for investigation of quantum entanglement, quantum interference, quantum chaos and in area of quantum information theory [12], [13], [14], [15]. The expression that defines fidelity is:

$$F(\rho,\sigma) = (Tr(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}))^2 \quad (6)$$

where ρ and σ are density matrixes. The other quantity of interest in this paper is the Poincaré section. It is usually used to investigate periodic/quasi-periodic dynamical systems that exhibit a periodic cycle or chaotic attractor and hence chaotic behavior. In more details the calculations of these quantities will be presented in the next sections.

2. PROGRAM ARCHITECTURE

The Library software package for development of photonic technologies is designed according to the object oriented programming (OOP) paradigms. Its design keeps abstraction and multiple code usage. The basic idea of the framework is to allow users to build arbitrarily complex interacting quantum systems from elementary free subsystems and interactions between them , and simulate their time-evolution with a number of available time-evolution drivers. The package has three main internal layers: engine, mediator objects and graphical user interface (GUI) (see Fig (1)). Each of engines is independent from the others and can be run separately.

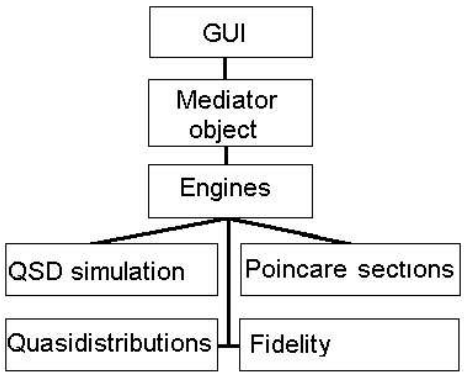

Figure 1: The program architecture diagram.

2.1 Quantum State Diffusion layer architecture

In this layer, are solved some problems of quantum optics like: density matrix calculation, calculations for the mean-value different boson operators of the quantum system which is described by any type of Hamiltonian. The current implementation of the layer keeps the basic ideas of library from [1]. Core is written in C++ software language and the high priority Boost library is used, which works well with C++. The engine core layer is a set of template classes representing Hermitian space objects [1]. State vector, different operator implementations (like a photon creation and annihilation, Linbland, etc), density matrix, and etc. are going to be the abstraction items. This layer also includes some numerical simulation algorithms, based on named objects, on of such algorithms in QSD method (Eq. (2)). In this version the incorrect input handling is done compile time using meta-programming concepts. The program will not run with wrong values time resources will be saved. The one of the modified parts of this layer is the output parsing process. We modified the previous version of this package mainly the parsing process to decrease simulation time. The parameters values of the Hamiltonian and output quantities are parsed for the first QSD trajectory simulation and are cached in memory. We reached this result by creating an abstract interface for data probing (measurement). An important part of this layer that the user can get results run time. For example, we below illustrate the situation for an open quantum system. One of the system of interest is the nondegenerate optical parametric oscillator (NOPO) with quarter plate inside the cavity recently proposed as an effective source of continuous variable entangled light beams with localized phases. The interaction Hamiltonian of this system reads as follows

$$H = \sum_{i=1}^{3} \hbar\triangle a_i^+ a_i + \imath\hbar E(t)\left(a_3^+ - a_3\right)$$
$$+\imath\hbar k\left(a_3 a_1^+ a_2^+ - a_3^+ a_1 a_2\right) + \hbar\chi(t)\left(a_1^+ a_2 + a_1 a_2^+\right), \quad (7)$$

where a_i are the boson operators for the cavity modes ω_i. The mode a_3 at frequency ω is driven by an external field with time modulated amplitude E(t), while a_1 and a_2 describe subharmonics of two orthogonal polarizations at degenerate frequencies $\omega/2$ generated in the process $\omega \to \omega/2+\omega/2$. The k determines the efficiency of the down-conversion process, while $\chi(t)$ describes the energy exchange between the subharmonics $\triangle_i$ and the cavity damping rates γ_i. In this layer it is possible to get numerically the density matrix and the photon excitation number for each of modes for this kind of systems.

2.2 Calculation of quasidistributions

This is purposed for calculation of quasidistributions (Wigner, Husimi) and tomography measurement for a given density matrix (Eqs (4) and (5)). The input file format can be specified by user from GUI. If the user requires more than one quasidistributions the simulation can run on different nodes, parallel and they will not be any synchronization problem with I/O streams as the density matrix is read from file at first and then simulation starts. The parameters needed for calculation of quasidistributions are: number of points in phase space, basis vector dimension, initial values in phase space, tomography angle if tomography measurement is required.

2.3 Poincaré section generation

We construct Poincaré section for periodic on time systems choosing x_0 and y_0 as an arbitrary initial phase-space point of the system at the time t_0. In this case, we define a constant phase map in the (X, Y) plane by the sequence of points $(X_n, Y_n) = (X(t_n), Y(t_n))$ at $t_n = t_0 + \frac{2\pi}{\delta}n$, for $n = 0, 1, 2, ...$, where δ is the frequency of the system. This means that for any $t = t_n$ the system is at one of the points

of the Poincaré section. The real and imagine parts of annihilation operator are considered as phase space axis. In general Poincaré section in quantum and classical limits are different. Quantum Poincaré map is obtained from one QSD trajectory. This method has been discussed in [16]. This is part of QSD layer. The difference between ordinary operator mean value calculation is the time range control and output format, the calculation time should be time of period. In the current implementation for obtaining classical Poincaré section Heinsenberg equation of annihilation operator in semiclassical limit should be entered from GUI. In future we plan to get Heinsenberg equation semiclassical equation programmatically from the Hamiltonian.

2.4 Fidelity calculation

This engine also is object oriented. It has a library for pure states that can be extended by user. To extend the pure states library user should inherit from states::state class and overload generate_norm() and generate_state_in_fock_basis() virtual functions. The only restriction in the current implementation is that the state should be represented in Fock basis. By default the library contains generalized coherent state, self-phase modulated (Kerr nonlinearity) state, coherent state in the presence of field, photon-phase squeezed states. For the input file user can specify his/her special format.

2.5 GUI and mediator object

The engines that simulate are separated from graphical user interface. The connection between GUI and engines is established by intermediate objects that holds data initialized from GUI by user and exports a file (source file) which is dynamically linked with the engines shared libraries for further simulation. The intermediate object is implemented to have one more level of abstraction and minimize GUI and engines relation. Simulation engines are independent from GUI and can work with other ones via objects::qsd, objects::quasidistribution, objects::fidelity objects. In the current implementation the GUI is realized by Qt 4.2 library. In GUI programming design generic programming concepts are not kept to have a richer interface. The connection between GUI and QSD engine is provided by objects::qsd that holds the Hamiltonian, number of modes, initial state, system parameters, evaluation time, vector dimension, number of trajectories to be run, number of processors on which program should run, the names of output files. The intermediate object purposed for quasidistributions holds user specified input file name, output files names, the quasidistribution type, initial state in phase space, the basis vectors dimension, number of points in phase space for keeping quasidistribution and tomography angle in the case of tomography measurement specification. By default the input file format is the same as generates QSD engine for density matrix, but the user can write his/her own from GUI. In this case user should be familiar with C++ I/O stream programming. The simulation success will depend on it.

3. PARALLELIZATION

Numerical simulations for system which have more than one dimension (number of freedom) may take long time. For this type of systems a state vector contains about 50000 complex numbers which is about of each trajectory for QSD takes minutes. And since at least 1000-5000 trajectories are required to get smooth results a full simulations takes days. To reduce execution time current version of program is developed which runs on cluster, and uses advantages of parallel programming. There are few ways to share the source code for parallel calculation. The QSD algorithm simulates the same equation many times with the same initial state and exactly same parameters. We choose one of them which have simple character and is effective as the maximum size of nodes (processors) is not limited. By using this sharing approach we avoid some problems like synchronization and time delay in massage passing process in interprocessors communication. The parallelism is realized by MPI (The Massage Passing Interface) library which work well with C++. For QSD layer in GUI part we give also the number of nodes N. After parsing program starts by calling run() function which is written in the exported main.cpp file. As parameter run() function takes also the number of trajectories ($nt = 5000$) and number of nodes

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    for(int i = 0; i < N; ++i)
    {
        if ( i == myid )
        x.run(tr, dt, nt, i);
    }
}
MPI_Finalize();
```

The $MPI\_Comm\_rank()$ function return the number of rank which get run() function like a parameter. While working in parallel mode equal number of trajectories is being simulated on different nodes (1-N) of the cluster. After simulation results from different processes are merged. In this way runtime of the simulation can be reduced by about many times, which explicitly depends on number of available nodes in the cluster.

4. USAGE

The application starts from ordinary main window. It contains menu bar, toolbar, status bar and project tree view. For starting simulation user should create a new project or open existing one using File->New Project or File->Open menu items. The *File* and *Edit* menu items are standard. The *Run* menu contains Run QSD, Run Quasidistribution, Run Poincaré, Run Fidelity items. Simulation menu contains the following items: Simulation error due to step, Simulation error due to trajectory count, Running trajectory number. The new project command will open a dialog shown in Fig. 2(a). It has four template for creating project. User should enter project name, and select any type for the project. By default the project is created in the current directory. If the user specifies "Simulate QSD" project then opened dialog shown in Fig. 2(b). In this tab user specifies parameters like number of simulation trajectories 2000, vector dimension size 500, the count of processors 8 for parallelization, time interval from 0 to 100, integration step 10^{-6}. To set physical system configurations user should change to the next tab "Physical System parameters" that is depicted in Fig. 2(c). In this tab user sets system Hamiltonian (Eq. 7), parameters $(E(t), k, \chi((t))$, specifies number of modes (a_i) in according GUI items. The parameters and operators added by user are displayed in list view to remind user about already added parameters. User cannot add parameters and creation/anihillation operators with the same names. To probe the simulation result user should also add files in this step. The type and name specification of parameter(operator) are required. Also user can add description to parameter that will be displayed as a tooltip in list view. During simulation user can require the error caused by integration step, simulation trajectories count. The error due to integration step corresponds to second-order Runge-Kutta integration algorithm error as the deterministic part of QSD equation is integrated by this algorithm.

5. ACKNOWLEDGEMENT

This paper is prepared in the framework of the ISTC projects: grants A-1451 and A-1606. We would like to acknowledge the developers of Boost, Qt, openMP.

REFERENCES

[1] H. H. Adamyan, N. H. Adamyan, N. T. Gevorgyan, T. V. Gevorgyan, and G. Yu. Kryuchkyan, "Software for numerical simulations in the field of quantum technologies based on parallel programming", *Physics of Particles and Nuclei Letters*, 161-163, 2008.

[2] R. Schack, T. A. Brunn, A C++ library using quantum trajectories to solve quantum master equations, Comp. Phys. Commun. **102**, 210-228, 1997

[3] S. M. Tan, A computational toolbox for quantum and atomic optics, J. Opt. B, 1, 424, (1999)

[4] A. Vukics, C++ QEDv2, arXiv:0904.4172 quant-ph (2009)

[5] N. Gisin and I. C. Percival, J. Phys. A **25**, 5677 (1992); **26**, 2233 (1993); **26**, 2245 (1993); I. C. Percival, Quantum State Diffusion(Cambridge Unoversity Press, Campridge, (2000)

[6] H. H. Adamyan, S. B. Manvelyan, and G. Yu. Kryuchkyan Phys. Rev. **E64**, 046219 (2001)

[7] H. H. Adamyan, S. B. Manvelyan, and G. Yu. Kryuchkyan Phys. Rev. **A63**, 022102 (2001)

[8] G. Yu. Kryuchkyan and S. B. Manvelyan, Phys. Rev. Lett. **88**, 094101 (2002); Phys. Rev. A**68**, 013823 (2003)

[9] T. V. Gevorgyan, A. R. Shahinyan, G. Yu. Kryuchkyan, Phys. Rev. A **79** 094101 (2009)

[10] W. P. Schleich, Quantum Optics in Phase Space, Wiley-VCH, (2001)

[11] R. Jozsa Fidelity for Mixed Quantum States. IN J. Mod. Opt. 41.23152324, 1994.

[12] H. F. Hofmann, T. Ide, and T. Kobayashi, Phys. Rev. A **62**, 062304 (2000)

[13] G. Benenti and G. Casati, Phys. Rev. E **65**, 066205 (2002)

[14] J. Emerson, Y. S. Weinstein, S. Lloyd, and D. G. Cory, Phys. Rev. Lett. **89**, 284102 (2002)

[15] H. Barnum, Ch. A. Fuchs, R. Jozsa, and B. Schumacher, arxiv:quant-ph/9603014v1, 1996.

[16] A. Kapulkin, A. K. Pattanayak, Phys. Rev. Lett. **101**, 074101 (2008)

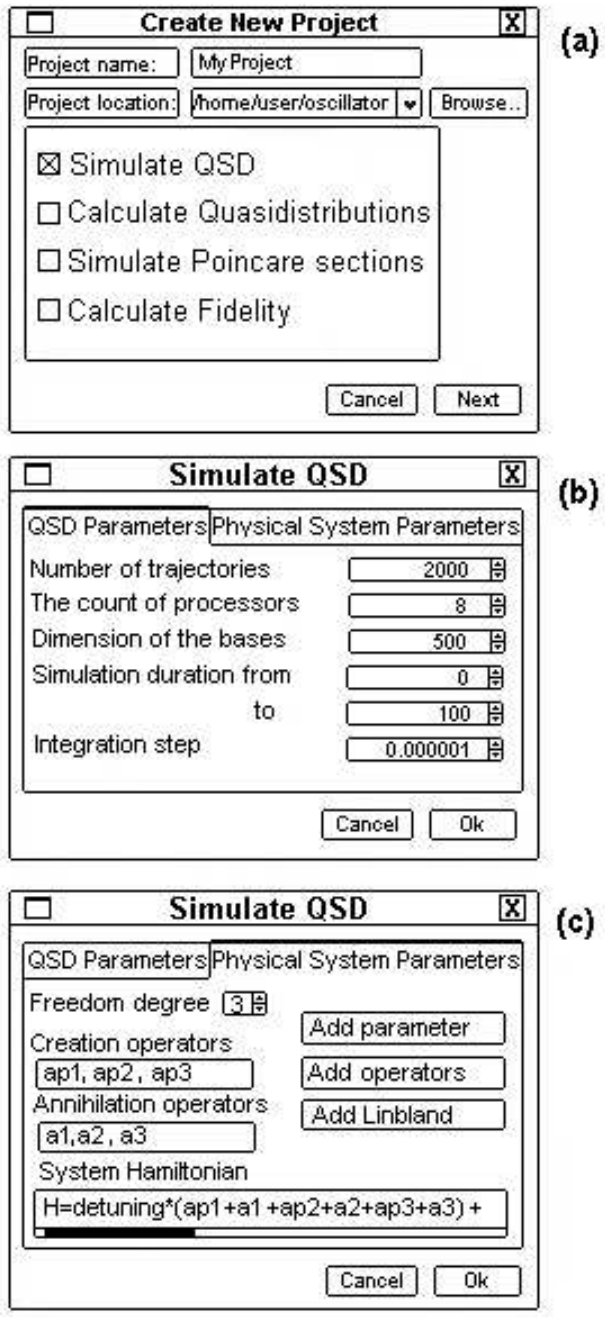

Figure 2: The dialogs to create new project. (a) New project dialog; (b) New QSD project dialog QSD parameters configuration dialog; (c) New QSD project physical system configuration dialog.