Письма в ЭЧАЯ. 2008. Т. 5, № 3(145). С. 289–293

SOFTWARE FOR NUMERICAL SIMULATIONS IN THE FIELD OF QUANTUM TECHNOLOGIES BASED ON PARALLEL PROGRAMMING

H. H. Adamyan [a], *N. H. Adamyan* [b], *N. T. Gevorgyan* [a,1], *T. V. Gevorgyan* [a,b], *G. Yu. Kryuchkyan* [a,b,2]

[a] Institute for Physical Research, National Academy of Sciences, Ashtarak-2, Armenia
[b] Yerevan State University, Yerevan, Armenia

We provide a software package for numerical simulations and modeling of complex quantum systems in the presence of dissipation and decoherence for a wide class of problems in the field of quantum technologies. This software is based on the method of quantum trajectories usually used for calculations of the density matrix. An important part of this toolkit is the universal user interface which is based on the TCL scripting language (Tool Command Language). It is elaborated in such a manner that the system description and system parameters should not be included in the source code. The core is implemented as a generic set of C++ classes which can be efficiently reused for modeling of a wide range of photonic systems. The code has been written so that it can facilitate optimization of the performance without breaking the object-orientedness of the design. We demonstrate that this software package is very useful for high-performance parallel calculations on the cluster.

Представлен пакет программ для проведения численных симуляций и моделирования сложных квантовых систем при наличии диссипации и нарушении когерентности для широкого класса задач в мире квантовых технологий. Программа основана на методе квантовых путей, обычно используемом для вычисления матрицы плотности. Важной частью данного программного обеспечения является универсальный пользовательский интерфейс, основанный на языке TCL. Он разработан таким образом, что нет необходимости включать описание системы и ее параметров в исходную программу. Ядро запускается как набор C++ классов, которые можно эффективно использовать снова для моделирования фотонных систем в широком диапазоне. Программа написана таким образом, что можно проводить оптимизацию производительности без нарушения объектно-ориентированного характера модели. Показано, что данный программный пакет весьма полезен при проведении параллельных вычислений на кластере.

PACS: 02.70.-c; 03.67.Lx

INTRODUCTION

Quantum state diffusion (QSD) is an efficient method of representing and computing the time evolution of quantum systems in the presence of dissipation and decoherence [1]. In most quantum systems, computation using QSD is orders of magnitude more economical in

[1] E-mail: ngevorg@server.physdep.r.am
[2] E-mail: gkryuchk@server.physdep.r.am

terms of computer storage space and computation time than numerical solution of master equation for the density operator. According to this method the density operator is calculated as the ensemble mean

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

over stochastic states $|\psi_\xi(t)\rangle$ which describe evolution of the quantum system along a quantum trajectory. Particularly, the corresponding equation of motion for photonic systems is

$$|d\psi_\xi\rangle = -\frac{i}{\hbar}H|\psi_\xi\rangle dt - \frac{1}{2}\left(L^+L - 2\langle L^+\rangle L + \langle L\rangle\langle L^+\rangle\right)|\psi_\xi\rangle dt + \left(L - \langle L\rangle\right)|\psi_\xi\rangle d\xi, \tag{2}$$

where H is a reference system Hamiltonian; $\hbar$ is the Planck constant; $L = \sqrt{\gamma}a$ is the Lindblad operator with γ being a dissipation rate; a is the annihilation photonic operator. Here ξ indicates the dependence on the stochastic process and the complex Wiener variables $d\xi$ satisfy the fundamental properties

$$M(d\xi) = 0, \quad M(d\xi, d\xi) = 0, \quad M(d\xi, d\xi^*) = dt. \tag{3}$$

As we see, in this method each trajectory can be evaluated independently of the others. This makes application of a cluster more preferable for computations.

We present software for numerical simulations of a wide class of complex quantum systems including elaboration of photonic devices using QSD and parallel calculations. This package has been arranged for high performed calculation on arm-cluster and allows us to perform calculations in reasonable time. The part of this activity has involved elaboration of the user interface for concretization of the quantum systems of interest as well as the core programs. This package has recently been applied for elaboration of new devices for long-distance quantum communication, for investigation of quantum dissipative chaos, and quantum critical phenomena [2–7]. Below, we describe the software architecture (Sec. 1), the user interface (Sec. 2), and peculiarities of parallel computation (Sec. 3).

1. SOFTWARE ARCHITECTURE

The software has several internal layers. A core layer is a set of template classes representing Hermitian space objects. These classes include such objects as state vector, different operator implementations, density matrix, etc. The number of mod's (number of freedom of the system) is also parameterized. A layer above contains named objects of state vectors, operators, numerical constants and variables, functions. This layer also includes some numerical simulation algorithms, based on named objects, one of such algorithms in QSD method. Defined variables, constants and functions can be used while defining operators. A layer above is a User Interface (UI). The UI is based on TCL, a well-known scripting language. TCL is widely used in scientific and engineering applications. It provides simple and easy interface as well as has advanced features. For simple computations a TCL programming knowledge is not required, a required system description can be built based on provided templates and examples.

1.1. Operators. Operators defined in the software as described above exist in two layers. The lower layer defines optimized template classed for built-in operators. These classes include: photon creation; photon annihilation; photon number; external pump operators.

Higher level operators are being built as mathematical expressions of built-in operators, functions and variables. Using operators which contain a lot of mathematical operations may decrease efficiency and increase runtime. Runtime of a simulation is critical for complex systems. For advanced users there are ways of avoiding this. One can define the desired operator in lower level of architecture and do optimizations there. A way of doing this is modifications/additions to source code and recompilation of the software, the other way is a creation of a shared library based on public headers and load as a plug-in into the system, it can easily be done by TCL «load» command. After this another operator can be built based on new defined operator.

1.2. Variables and Functions. Each system for simulation has some parameters, which are being represented as constants in operators. The description of a system may also depend on time. Time is also exported as a variable and can be referred to from within operators. Each algorithm which does time evaluation simulation updates this variable on each step. Also it is possible to define functions as named expressions. While constructing expressions some built-in functions may be used, such as sin, cos, exp, etc. It is also possible to define additional functions through public headers by constructing and loading shared libraries, essentially in the same way as for operators described in Subsec. 1.1.

2. USER INTERFACE

As noted above, user interface is based on TCL scripting language. A UI layer provides a set of classes for registration of commands, option parsing and error handling, as well as a set of commands for basic operation. This set includes commands for defining functional and operator expression, state vectors management, simulation algorithms.

The set of classes are kind of C++ wrapper around C level TCL API (Application Programming Interface). It provides easy and type-based parsing of command options. It is also possible to define user commands based on either this C++ API or direct TCL API. The defined commands can be integrated by loading shared libraries in the same way as operators and functions.

2.1. Basic Set of Commands. Here are provided basic commands in more detail.

2.1.1. Defined_operator. Creates an instance of a built-in operator giving a name.

Command: define_operator

Summary: Defines a named operator of built-in type.

Syntax: define_operator <name> type <type> -mode <int>

Example: define_operator P -type creation -mode 1

Defines a first mod creation operator named P.

2.1.2. Feval. Evaluates given string expression. This command can be used for creation of named expressions (functions), for defining constants and for printing values.

Command: feval

Summary: Evaluates given expression.

Syntax: feval <string>

Example: feval {omega = 0.1; pump = 1; F0 = 1.5;}

This example defines a few different constants.

feval {F_t = F0 * cos (omega * t); } Defines a time-dependent function.

2.1.3. Oeval. Evaluates given string operator expression. While parsing new names are assumed as operator names, while values can refer to operator or expression.

Command: oeval

Summary: Evaluates given operator expression.

Syntax: oeval <string>

Example: oeval { H = omega*N + pump * (A + P) }

This expression defines operator named «H» referring to operators N, A and P, and expressions/constants omega and pump.

2.1.4. Define_state. Creates a state vector with given name of given size.

Command: define_state

Summary: Creates a state vector.

Syntax: define_state <string> -size <list of int>

Example: define_state I -size 100

Creates a state vector named «I» of size 100. In this case the number of modes is 1.

2.1.5. qsd. This command executes Quantum State Diffusion algorithm.

Command: sqa

Summary: Executes Quantum State Diffusion algorithm.

Syntax: qsd -Hamiltonian <string> -Linblads <list of strings> -nTraj <int> -timeVar <string> -init <string> -state_prefix <string>

Example: qsd -Hamiltonian H -Linblads A -init I -nTraj 500 -state_prefix "har_"

2.2. A Simple Example. Here is a simple example of a TCL script which executes QSD for harmonic system, and exports mean values of number operator to a file.

```
define_operator A -type annihilation -mode 1
define_operator P -type creation -mode 1
define_state I -size 100
feval {omega = 0.1; pump = 1; tStart = 0; tEnd = 5; dt = 0.01;}
oeval { N = P * A }
oeval { H = omega*N + pump * (A + P) }
set prefix har
qsd -Hamiltonian H -Linblads A -init I -nTraj 500 -state_prefix$prefix
export_states -state_prefix $prefix -operator N
```

3. OPTIMIZATIONS (PARALLEL COMPUTATION)

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 the system, state vector may have a big size. As a result, a simulation of each trajectory for QSD takes minutes. And since at least 1000–5000 trajectories are required to get smooth results a full simulation takes days. To reduce execution time, a version of program is developed which runs on cluster and uses advantages of parallel programming. QSD algorithm simulates the same equation many times with the same initial state and exactly the same parameters. After simulation results from different runs are

merged. While working in parallel mode an equal number of trajectories are being simulated on different nodes of the cluster. In this way a runtime of the simulation can be reduced about many times, which explicitly depends on the number of available nodes in the cluster.

CONCLUSION

A program for numerical simulations in the field of quantum technologies is developed. The program interaction is built on user interface based on the scripting language TCL well known in technical field. The program comes with a set of basic TCL commands, but also is open for adding user defined operations. It has a set of built-in operators and also provides an interface for defining complex operators. For this a TCL interface and C++ interface are provided. It also provides interface for adding additional mathematical functions that use defined TCL commands. For optimal computations the program also supports parallel programming.

Acknowledgements. This work was supported by INTAS Grant 04-77-7289.

REFERENCES

1. *Percival I. C.* Quantum State Diffusion. Cambridge Univ. Press, 2000.
2. *Adamyan H. H., Manvelyan S. B., Kryuchkyan G. Yu.* Stochastic Resonance in Quantum Trajectories for an Anharmonic Oscillator // Phys. Rev. A. 2001. V. 63. P. 022102.
3. *Adamyan H. H., Manvelyan S. B., Kryuchkyan G. Yu.* Chaos in a Double Driven Dissipative Nonlinear Oscillator // Phys. Rev. E. 2001. V. 64. P. 046219.
4. *Kryuchkyan G. Yu., Manvelyan S. B.* Quantum Dissipative Chaos in the Statistics of Excitation Numbers // Phys. Rev. Lett. 2002. V. 88. P. 09410.
5. *Kryuchkyan G. Yu., Manvelyan S. B.* Sub-Poissonian Statistics in Order-to-Chaos Transition // Phys. Rev. A. 2003. V. 68. P. 0138114.
6. *Adamyan H. H., Kryuchkyan G. Yu.* Continuous Variable Entanglement of Phase Locked Light Beams // Phys. Rev. A. 2004. V. 69. P. 053814.
7. *Adamyan H. H. et al.* Quadrature Entanglement and Photon Correlations in the Presence of Phase-Locking // Phys. Rev. A. 2006. V. 73. P. 033810.