Quantum Construct

Version 1.0

- Design -

Shekhar Suresh Chandra[1]

September 23, 2005

[1]Monash University. Email: Shekhar.Chandra@spme.monash.edu.au

Contents

Preface

This document is intended to provide the framework and design of the *Quantum Construct C++ Toolkit* (henceforth refered to as $qC++$). This is the second of the three forms of documentation writen for $qC++$, the others are the Specifications document and the Code Documentation done in HTML format using Doxygen.

Acknowledgments

I wish to thank Dr. Rotha Yu for his experience and information pertaining to the numerical work of the project for which $qC++$ was originally designed. As well as my supervisor Associate Professor Michael Morgan for his insightful vision, critical input and guidance on this project. My honours colleague, Gary Ruben for his valuable information and ideas also regarding the development of the project and $qC++$. Finally to my Family, for their patience and care during my time in honours.

Chapter 1

Introduction

The QC Library is intended to be used to construct quantum systems and numerically evolve them using quantum mechanics. The various containers have to be identified for the type of field or system being evolved, as well as an overall class/object structure of any standard quantum system.

Chapter 2

Key Elements of Quantum Systems

The key elements of a quantum system will vary from application to application, however, in general a quantum system will have

1. Quantum Fields in various forms - such as scalar and vector fields.
2. Equilibrium State/Solver - the physics of the static form of the system.
3. Evolver - the physics of the system will tell the system how to evolve.
4. Phases - phase retrieval and manipulation are also desired in certain systems.
5. The System - The system to be modelled. Here the system comprises of all the above.

These classes will be together with the GUI and Graphing classes that will also be developed.

Chapter 3

Classes/Objects

For each of the key areas mentioned above, a class/object will be created. From these classes, the library will be built, as well as the actual project software. As a matter of convention, all classes will have the QC prefix to them.

3.1 Quantum Field Classes

These classes are mainly container classes with added data extraction for specific physical applications. The base is the QCField class (where the common elements of all fields reside) and then the more specific classes derived from the base class.

3.1.1 Base Class - QCField

This will be the wrapper of the Blitz++ Array data structure. It shall form the basis of other types of fields in physics (such as complex scalar fields, vector fields etc.), hence the reason of it being the base class for the quantum fields. The class will be implemented using templates for the type of field and dimension of the field.

3.1.2 Derived Class - QCComplexField

This class shall provide the necessary operations and containment for the complex scalar fields, required for neutral complex scalar fields (such as the electric scalar potential and neutral order parameters etc.). It is a derived class consisting of complex values. The class will be implemented using templates for the dimension of the field.

3.1.3 Derived Class - QCVectorField

This class shall provide the necessary operations and containment for vector fields, required for fields such as the magnetic vector potential. It is a derived class consisting of vectors. The class will be implemented using templates for the dimension of the field.

3.2 Numerical Classes

These classes are mainly numerical method related classes designed to complete one of two primary numerical objectives. These are to setup a *Steady State Solution* and to *Evolve the Solution*.

3.2.1 Steady State Class - QCSteadyState

The class is designed to take a field as input and setup a steady state (time-independent solution) according to various supported quantum mechanical equations (such as the Gross-Pitaevskii equation). This is essentially a boundary value problem solver and will utilize the respective numerical methods.

3.2.2 Evolving Class - QCEvolve

The class is designed to take a field as input and to evolve the field according to the supported equations (such as the Gross-Pitaevskii equation), in a similar fashion to the

unitary time evolution operator of quantum mechanics. This is essentially an initial value problem requiring unitary evolution. Consequently, the applicable methods will be used, such as the *Fast Semi-Implicit Method* utilizing *Approximate Factorization*.

3.2.3 Phase Class - QCPhase

The manipulation of phase is desired in some systems. This class is designed to allow various methods of phase retrieval and phase imprinting. Some methods of phase retrieval will be the Generalised Gerchberg-Saxton algorithm (GGS algorithm).

3.3 System Classes

This class is where the quantum system is encapsulated. It uses all the above classes, except perhaps those of the plotting and GUI classes. There will be a base class and then derived classes which will resemble the system being modelled more closely.

3.3.1 Base Class - QCSystem

This class is designed to encapsulate the base or the common elements of all quantum systems, mostly all the numerical classes and the members related to them. The specific systems have to be designed as a derived class from this class.

3.3.2 Derived Class - QCBEC

The derived class from the QCSystem class that specifically models a Bose-Einstein Condensate. All BEC specific related members are in this class.

3.4 Graphing Classes

The graphing classes are designed to display the output of the numerical simulations. Various methods are available, but since the GUI's are to be implemented using Trolltech QT, the plotting packages used will be those that are QT integratable. Such a package is the QwtPlot3D library. This is esstentially a widget plugin for QT.

3.4.1 3D Plotting Class - QCPlot

This class shall be a widget class that shall be able to read files, load them and plot them. Also be able to extract the necessary data from the fields and plot them also. It shall be done using QwtPlot3D, which utilizes OpenGL. Various support for files will be implemented as well as various types of plots. The surface plot shall be the first to be implemented.

3.5 GUI Classes

The GUI's classes are difficult to design to be general in terms of applications. However, a couple of general output windows can be designed. All GUI's will be done Trolltech QT, due to the maturity, acceptance in the software industry and portability of the GUI library.

3.5.1 Status Dialog Class - QCStatusDlg

The Status Dialog class is designed to be a dialog box that contains the general status output of the simulation (such as the stage of simulation and the total time taken etc.).

3.5.2 Plot Dialog Class - QCPlotDlg

The Plot Dialog class is designed to be a dialog box that contains the general options of the plot of the simulations (such as the plot range and colours etc.).