QUANTUM MECHANICS USING COMPUTER ALGEBRA

Includes Sample Programs in C++, SymbolicC++, Maxima, Maple, and Mathematica

2nd Edition

Willi-Hans Steeb • Yorick Hardy

$$[b, b^\dagger] = I$$

$$[c, c^\dagger]_+ = I$$

World Scientific

QUANTUM MECHANICS USING COMPUTER ALGEBRA

Includes Sample Programs in C++, SymbolicC++, Maxima, Maple, and Mathematica

2nd Edition

William Stein
University of Johannesburg, South Africa

Yorick Hardy
University of Johannesburg, South Africa

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

QUANTUM MECHANICS USING COMPUTER ALGEBRA
Includes Sample Programs in C++, SymbolicC++, Maxima, Maple, and Mathematica
(2nd Edition)

ISBN-13 978-981-4307-16-1
ISBN-10 981-4307-16-5

Printed in Singapore by World Scientific Printers

QUANTUM MECHANICS USING COMPUTER ALGEBRA

Includes Sample Programs in C++, SymbolicC++, Maxima, Maple, and Mathematica

2nd Edition

Preface

Solutions to problems in quantum mechanics are important for scientists, engineers and many others. This book gives a collection of most standard methods in quantum mechanics together with their programs in SymbolicC++, Maxima, Mathematica and Maple. Advanced topics in quantum mechanics are also included. In most cases the output of the programs is also displayed. Most of the problems are implement in SymbolicC++ and Maxima. For a number of selected problems the program are implemented in Mathematica, Maple and C++. In the first edition the programs had been implemented in Reduce.

SymbolicC++, Maxima, Reduce, Mathematica and Maple are the most widely available and simple to use computer algebra systems. They enable users to manipulate algebraic expressions and equations symbolically. For example, we can differentiate and integrate symbolically. Number crunching can also be done. Moreover, symbolic manipulation and number crunching can be combined in one program.

Beside the standard methods, modern developments in quantum mechanics are also included. These include Bose operators, Fermi operators, coherent states, squeezed state, gauge theory, quantum groups and super Lie algebras. All the special functions (such as Hermite, Chebyshev, Legendre) important in quantum theory and Hilbert space theory are also implemented.

The level of presentation is such that one can study the subject early on in ones education in science. There is a balance between practical computation and the underlying mathematical theory. The book is ideally suited for use in a quantum mechanics lecture.

The web sites for the different packages are:

Maxima: `http://maxima.sourceforge.net`

SymbolicC++: `http://issc.uj.ac.za`

Reduce: `http://reduce-algebra.sourceforge.net`

Maple: `http://www.maplesoft.com`

Mathematica: `http://www.wolfram.com`

Without doubt, this book can be extended. If you have comments or suggestions, we would be pleased to have them. The email addresses of the authors are:

`steebwilli@gmail.com`
`yorickhardy@gmail.com`

Contents

1. Introduction

Most of the programs are written in SymbolicC++ and Maxima. A few are written in Maple and Mathematica.

SymbolicC++ is case-sensitive. In SymbolicC++, variables are created with the `Symbolic("name")` constructor, for example `Symbolic x("x")`. Additional parameters specify dimensions for vectors, `Symbolic v("v",3)`, or matrices `Symbolic A("A",2,2)`. The tilde operator (`~`) makes a non-commutative variable from a commutative one and vice versa. The usual functions such as `sin(x)`, `cos(x)` and `exp(x)` are provided. Arbitrary functions $y(x)$ are written as `y[x]`. In SymbolicC++ the command for differentiation is `df` and for integration `integrate`. For example

```
cout << df((x^3)+2*x,x) << endl;       // 3*x^(2)+2
cout << integrate((x^2)+1,x) << endl; // 1/3*x^(3)+x
```

The command for solving equations is `solve`. For example

```
cout << solve((x^2)+(a+1)*x+a==0, x) << endl;
```

Substitutions are achieved with the indexing operator `[]` or the method `subst`. The method `subst_all` works like `subst` except that it tries to perform the substitution on the resulting expression until no further substitution is possible.

```
cout << (x*y + (x^2))[x==2] << endl;        // 2*y+4
cout << (x*y + (x^2)).subst(x==2) << endl; // 2*y+4
```

SymbolicC++ also includes methods for obtaining the coefficients of expressions (`coeff`). Manipulations with matrices are also implemented, for example matrix multiplication, the Kronecker product (`kron`), direct sum, trace, determinant and many other operations. For example the 3×3 identity matrix could be represented by

```
Symbolic I3 = Symbolic("",3,3).identity;
```

An example to represent the 2×2 matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

with noncommutative elements a, b, c, d would be

```
Symbolic a("a"), b("b"), c("c"), d("d");
a = ~a; b = ~b; c = ~c; d = ~d;  // noncommutative
Symbolic T = ((a, b),(c, d));
```

Since SymbolicC++ is embedded in C++ one can utilize the powerful programming techniques of C++ in SymbolicC++.

Maxima is case-sensitive. To differentiate $x^3 + 2x + 2$ with respect to x we write

```
diff(x^3+2*x+2,x)
```

To integrate $\exp(-x) + 2 * x$ with respect to x we write

```
integrate(exp(-x)+2*x,x)
```

When we want to integrate this expression between 0 and 2 we write

```
integrate(exp(-x)+2*x,x,0,2)
```

The `solve` command tries to solve equations. For example

```
solve(x^2+(a+1)*x+a=0,x)
```

Another important command is the substitution command **subst**. For example

```
subst(x=2,x*y+x^2)
```

with the output `2 y + 4`. Amongst others, Maxima includes the following mathematical functions `sqrt` (square root), `exp` (exponential functions), `log` (natural logarithm ln, and the trigonometric functions `sin`, `cos`, `tan`. Maxima reserves `%i` for $\sqrt{-1}$, `%pi` for the number π and `%e` for the number e. All the necessary matrix manipulations can also be done. Maxima can also be utilized as programming language.

Maple is also case-sensitive. In Maple the differentiation command **diff**. For example

```
diff(x^3+2*x,x)
```

Integration is the command `int`. For example

```
int(x^2+1,x)
```

and if we want to integrate between boundaries

```
int(x^2+1,x,0..2)
```

Maple has two different commands for solving equations. The command

```
solve(x^2+(1+a)*x+a=0,x)
```

solves the equation $x^2 + (1 + a)x + a = 0$ with respect to x and gives the result $x = -1$ and $x = -a$. The command

```
fsolve(x^2-x-1=0,x)
```

solves the quadratic equation $x^2 - x - 1 = 0$ and gives the output $-0.618...$ and $1.618....$ The substitution command is given by **subs**. For example the command

```
subs(x=2,x*y+x^2)
```

gives $2y + 4$. Among others, Maple includes the following mathematical functions: `sqrt` (square root), `exp` (exponential function), `log` (natural logarithm) and the trigonometric functions `sin`, `cos`, `tan` with the arguments in radians. Predefined constants are `Catalan`, `Pi`, `gamma`, `infinity` `false`, `true`. All the operations for matrix manipulations are also provided. Maple can also be utilized as programming language.

Mathematica is also case-sensitive. In Mathematica the differentiation command is `D`. For example

```
D[x^3+2*x,x]
```

Integration is the command `Integrate`. For example

```
Integrate[x^2+1,x]
```

and if we want to integrate between boundaries

```
Integrate[x^2+1,{ x,0,2 }]
```

The command `solve` in Mathematica can solve a number of algebraic equations and also systems of algebraic equations. For example

```
solve[x^2+(1+a)*x+a==0,x]
```

solves the equation $x^2 + (1 + a)x + a = 0$ with respect to x and gives the result $x = -1$ and $x = -a$. The replacement operator `/.` applies rules to expressions. Consider the expression $x * y + x * x$. Then

```
x*y+x*x /. x-> 2
```

gives $4+2y$. Among others, Mathematica includes the following mathematical functions: `Sqrt` (square root), `Exp` (exponential function), `Log` (natural logarithm) and the trigonometric functions `Sin`, `Cos`, `Tan` with the arguments in radians. Predefined constants are `I` (for $\sqrt{-1}$), `Catalan`, `E`, `Pi`, `Degree`, `GoldenRatio`, `EulerGamma`. All the operations for matrix manipulations are also provided. Mathematica can also be utilized as programming language.

2. Conservation Law and Schrödinger Equation

The wave function ψ satisfies the *Schrödinger equation*

$$i\hbar\frac{\partial\psi}{\partial t} = \hat{H}\psi \tag{1}$$

where the Hamilton operator $\hat{H}$ is given by

$$\hat{H} := -\frac{\hbar^2}{2m}\Delta + V(\mathbf{r}), \qquad \Delta := \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2}. \tag{2}$$

In the SymbolicC++ program and Maxima program we show that the *conservation law*

$$\frac{\partial\rho}{\partial t} + \mathrm{div}\mathbf{s} = 0, \qquad \mathrm{div}\mathbf{s} := \frac{\partial s_1}{\partial x_1} + \frac{\partial s_2}{\partial x_2} + \frac{\partial s_3}{\partial x_3} \tag{3}$$

holds, where the *probability density* ρ is defined by

$$\rho := \psi^*\psi \tag{4}$$

and the *probability current density*

$$\mathbf{s} := \frac{\hbar}{2mi}(\psi^*\nabla\psi - \psi\nabla\psi^*). \tag{5}$$

We also have

$$\int_{\mathbb{R}^3} \psi^*(\mathbf{x})\psi(\mathbf{x})dx_1dx_2dx_3 = 1 \tag{6}$$

i.e. the wave function ψ is normalized. The program is written for one-space dimension, but can easily be extended to higher dimensions. In one-space dimension we have

$$\Delta := \frac{\partial^2}{\partial x^2} \tag{7}$$

$$s := \frac{\hbar}{2mi}\left(\psi^*\frac{\partial\psi}{\partial x} - \psi\frac{\partial\psi^*}{\partial x}\right). \tag{8}$$

We take into account that

$$-i\hbar\frac{\partial\psi^*}{\partial t} = \hat{H}\psi^*. \tag{9}$$

```
// conservation.cpp

#include <iostream>
#include "symbolicc++.h"
using namespace std;

int main(void)
{
Symbolic psi("psi"), psis("psis"), x("x"), t("t"), V("V");
Symbolic hb("hb"), i("i"), m("m");

psi = psi[x,t]; psis = psis[x,t];
Symbolic rho = psis*psi;
Symbolic r1 = df(rho,t);
Symbolic r2 = r1[df(psi,t)==i*hb/(2*m)*df(psi,x,2)-i*V*psi];
Symbolic r3 = r2[df(psis,t)==-i*hb/(2*m)*df(psis,x,2)+i*V*psis];

Symbolic s = -i*hb/(2*m)*(psis*df(psi,x)-psi*df(psis,x));
Symbolic r4 = df(s,x);
cout << "result = " << r4 + r3 << endl;
return 0;
}
```

The output is

```
result = 0
```

The Maxima program is

```
/* conservation.mac */

depends (psi,x,psi,t);
depends (psis,x,psis,t);

rho: psis*psi;
r1: diff(rho,t);
r2: subst(%i*hb/(2*m)*diff(psi,x,2)-%i*V*psi,diff(psi,t),r1);
r3: subst(-%i*hb/(2*m)*diff(psis,x,2)+%i*V*psis,diff(psis,t),r2);
s: -%i*hb/(2*m)*(psis*diff(psi,x)-psi*diff(psis,x));
r4: diff(s,x);
result: expand(r4 + r3);
print(result);
```

3. Wave Packet and Free Schrödinger Equation

The Schrödinger equation for the free particle in one space dimension is given by

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}. \tag{1}$$

This equation admits the solution (wave packet)

$$\psi(x,t) = \frac{B}{(1+i\hbar t/ma^2)^{1/2}}\exp\left(-\frac{x^2}{2a^2\,(1+i\hbar t/ma^2)}\right) \tag{2}$$

where a has the dimension of a length and B is determined by normalizing ψ for a fixed t, i.e.,

$$\int_{-\infty}^{\infty}\psi^*(x,t)\psi(x,t)dx = 1. \tag{3}$$

For $t = 0$ we have

$$\psi(x,0) = B\exp\left(-\frac{x^2}{2a^2}\right) \tag{4}$$

with the density

$$\rho(x,0) = \psi^*(x,0)\psi(x,0) = |B|^2\exp\left(-\frac{x^2}{a^2}\right). \tag{5}$$

Thus at $t = 0$ the particle is localized with $|x| \leq a$. Using

$$\widetilde{\psi}(\widetilde{x}(x),\widetilde{t}(t)) = \psi(x,t), \quad \widetilde{x} = x/a, \quad \widetilde{t} = \hbar t/(ma^2) \tag{6}$$

we can cast the Schrödinger equation into the dimensionless form

$$i\frac{\partial\widetilde{\psi}}{\partial\widetilde{t}} = -\frac{1}{2}\frac{\partial^2\widetilde{\psi}}{\partial\widetilde{x}^2} \tag{7}$$

with

$$\widetilde{\psi}(\widetilde{x},\widetilde{t}) = \frac{B}{(1+i\widetilde{t})^{1/2}}\exp(-\widetilde{x}^2/2)\,. \tag{8}$$

In the SymbolicC++ program we show that (8) is a solution of (7). We also find ψ^*. The density is given by $\rho = \psi\psi^*$. We show that ψ^* satisfies the dimensionless Schrödinger equation

$$i\hbar\frac{\partial\widetilde{\psi}^*}{\partial\widetilde{t}} = \frac{1}{2}\frac{\partial^2\widetilde{\psi}^*}{\partial\widetilde{x}^2}\,. \tag{9}$$

In the Maxima program we utilize (7) and the ansatz (8).

```
// wavepacket.cpp

#include <iostream>
#include "symbolicc++.h"
using namespace std;

int main(void)
{
using SymbolicConstant::i;
Symbolic xt("xt"), tt("tt"), B("B");
Symbolic f1 = B/(sqrt(1+i*tt));
Symbolic f2 = exp(-(xt*xt)/(2*(1+i*tt)));
Symbolic psit = f1*f2;
Symbolic res1 = df(psit,xt,2)/2; res1 = res1/f2;
Symbolic res2 = i*df(psit,tt);   res2 = res2/f2;
Symbolic result1 = res1 + res2;
cout << "result1 = " << result1 << endl << endl;

Symbolic f1s = f1[i==-i]; Symbolic f2s = f2[i==-i];
Symbolic psist = f1s*f2s;
Symbolic res3 = df(psist,xt,2)/2;
res3 = res3/f2s;
Symbolic res4 = -i*df(psist,tt);
res4 = res4/f2s;
Symbolic result2 = res3 + res4;
cout << "result2 = " << result2 << endl << endl;
return 0;
}
```

The Maxima program is

```
/* wavepacket.mac */

depends (psi,x,psi,t);
depends (f1,x,f1,t);
depends (f2,x,f2,t);
f1: B/(sqrt(1+%i*hb*t/(m*a2)));
f2: exp(-(x*x)/(2*a2*(1+%i*hb*t/(m*a2))));
psi: f1*f2;
res1: hb*hb*diff(psi,x,2)/(2*m);
res2: %i*hb*diff(psi,t);
result1: (res1+res2)/f2;
result2: expand(result1);
print(result2);
```