PD Dr. Roman Schmied

Using Mathematica for Quantum Mechanics

A Student's Manual

University of Basel, Switzerland

contents

preface

> The limits of my language mean
> the limits of my world.
>
> Ludwig Wittgenstein

Learning quantum mechanics is difficult and counter-intuitive. The first lectures I heard were filled with strange concepts that had no relationship with the mechanics I knew, and it took me years of solving research problems until I acquired even a semblance of understanding and intuition. This process is much like learning a new language, in which a solid mastery of the concepts and rules is required before new ideas and relationships can be expressed fluently.

The major difficulty in bridging the chasm between introductory quantum lectures, on the one hand, and advanced research topics, on the other, was for me the lack of such a language, or of a technical framework in which quantum ideas could be expressed and manipulated. On the one hand, I had the hand tools of algebraic notation, which are understandable but only serve to express very small systems and ideas; on the other hand I had diagrams, circuits, and quasi-phenomenological formulae that describe interesting research problems, but which are difficult to grasp with the mathematical detail I was looking for.

This book is an attempt to help students transform all of the concepts of quantum mechanics into concrete computer representations, which can be constructed, evaluated, analyzed, and hopefully understood at a deeper level than what is possible with more abstract representations. It was written for a Master's and PhD lecture given yearly at the University of Basel, Switzerland. The goal is to give a language to the student in which to speak about quantum physics in more detail, and to start the student on a path of fluency in this language. We will revisit most of the problems encountered in introductory quantum mechanics, focusing on computer implementations for finding analytical as well as numerical solutions and their visualization. On our journey we approach questions such as:

- You already know how to calculate the energy eigenstates of a single particle in a simple one-dimensional potential. How can such calculations be generalized to non-trivial potentials, higher dimensions, and interacting particles?
- You have heard that quantum mechanics describes our everyday world just as well as classical mechanics does, but have you ever seen an example where such behavior is calculated in detail and where the transition from classical to quantum physics is evident?
- How can we describe the internal spin structure of particles? How does this internal structure couple to the particles' motion?
- What are qubits and quantum circuits, and how can they be assembled to simulate a future quantum computer?

Most of the calculations necessary to study and visualize such problems are too complicated to be done by hand. Even relatively simple problems, such as two interacting particles in a one-dimensional trap, do not have analytic solutions and require the use of computers for their solution and visualization. More complex problems scale exponentially with the number of degrees of freedom, and make the use of large computer simulations unavoidable.

The methods presented in this book do not pretend to solve large-scale quantum-mechanical problems in an efficient way; the focus here is more on developing a descriptive language. Once this language is

established, it will provide the reader with the tools for understanding efficient large-scale calculations better.

Why Mathematica?

This book is written in the *Wolfram language* of Mathematica (version 11); however, any other language such as Matlab or Python may be used with suitable translation, as the core ideas presented here are not specific to the Wolfram language.

There are several reasons why Mathematica was chosen over other computer-algebra systems:

- Mathematica is a very high-level programming environment, which allows the user to focus on *what* s?he wants to do instead of *how* it is done. The Wolfram language is extremely expressive and can perform deep calculations with very short and unencumbered programs.
- Mathematica supports a wide range of programming paradigms, which means that you can keep programming in your favorite style. See section 1.9 for a concrete example.
- The *Notebook* interface of Mathematica provides an interactive experience that holds programs, experimental code, results, and graphics in one place.
- Mathematica seamlessly mixes analytic and numerical facilities. For many calculations it allows you to push analytic evaluations as far as possible, and then continue with numerical evaluations by making only minimal changes.
- A very large number of algorithms for analytic and numerical calculations is included in the Mathematica kernel and its libraries.

Mathematica source code

Some sections of this book contain embedded Mathematica source code files, for direct evaluation by the reader (see page 121 for a list of embedded files). If your PDF reader supports embedded files, you will see a double-clickable orange link here: [code]. If all you see is a blank space between orange square brackets, or a non-clickable orange link, your PDF reader does not support embedded files; please switch to the Adobe® Acrobat® Reader®.

outline of discussed topics

In five chapters, this book takes the student all the way to relatively complex numerical simulations of quantum circuits and interacting particles with spin:

Chapter 1 gives an introduction to Mathematica and the Wolfram language, with a focus on techniques that will be useful for this book. This chapter can be safely skipped or replaced by an alternative introduction to Mathematica.

Chapter 2 makes the connection between quantum mechanics and vector/matrix algebra. In this chapter, the abstract concepts of quantum mechanics are converted into computer representations, which form the basis for the following chapters.

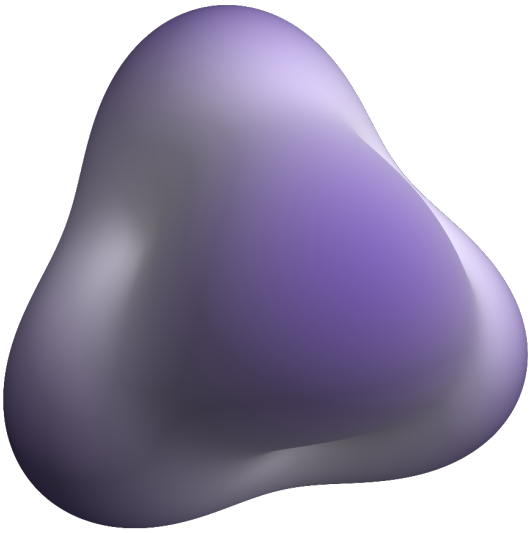
Chapter 3 discusses quantum systems with finite-dimensional Hilbert spaces, focusing on spin systems and qubits. These are the most basic quantum-mechanical elements and are ideal for making a first concrete use of the tools of chapter 2.

Chapter 4 discusses the quantum mechanics of particles moving in one- and several-dimensional space. We develop a real-space description of these particles' motion and interaction, and stay as close as possible to the classical understanding of particle motion in phase space.

Chapter 5 connects the topics of chapter 3 and chapter 4, describing particles with spin that move through space.

1

Wolfram language overview

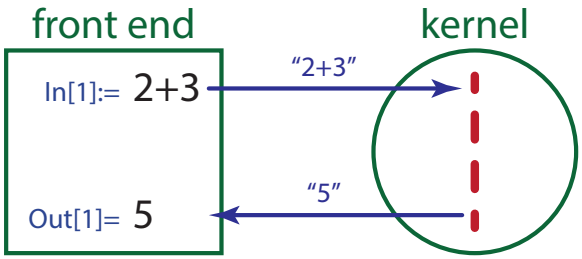


The Wolfram language is a beautiful and handy tool for expressing a wide variety of technical thoughts. Wolfram Mathematica is the software that implements the Wolfram language. In this chapter, we have a look at the most central parts of this language, without focusing on quantum mechanics yet. Students who are familiar with the Wolfram language may skip this chapter; others may prefer alternative introductions. Wolfram Research, the maker of Mathematica and the Wolfram language, provides many resources for learning:

- https://www.wolfram.com/mathematica/resources/ – an overview of Mathematica resources to learn at your own pace
- https://reference.wolfram.com/language/guide/LanguageOverview.html – an overview of the Wolfram language
- https://www.wolfram.com/language/ – the central resource for learning the Wolfram language
- https://reference.wolfram.com/language/ – the Mathematica documentation

1.1 introduction

Wolfram Mathematica is an interactive system for mathematical calculations. The Mathematica system is composed of two main components: the *front end*, where you write the input in the Wolfram language, give execution commands, and see the output, and the *kernel*, which does the actual calculations.



This distinction is important to remember because the kernel remembers all the operations in the order they are sent to it, and this order may have nothing to do with the order in which these commands are displayed in the front end.

When you start Mathematica you see an empty "notebook" in which you can write commands. These commands are written in a mixture of text and mathematical symbols and structures, and it takes a bit of practice to master all the special input commands. In the beginning you can write all your input in pure text mode, if you prefer. Let's try an example: add the numbers $2+3$ by giving the input

```
In[1]:= 2+3
```

and, with the cursor anywhere within the "cell" containing this text (look on the right edge of the notebook to see cell limits and groupings) you press "shift-enter". This sends the contents of this cell to the kernel, which executes it and returns a result that is displayed in the next cell:

```
Out[1]= 5
```

If there are many input cells in a notebook, they only get executed in order if you select "Evaluate Notebook" from the "Evaluation" menu; otherwise you can execute the input cells in any order you wish by simply setting the cursor within one cell and pressing "shift-enter".

The definition of any function or symbol can be called up with the `?` command:

```
In[2]:= ?Factorial
        n! gives the factorial of n. >>
```

The arrow ≫ that appears at the end of this informative text is a hyperlink into the documentation, where (usually) instructive examples are presented.

1.1.1 exercises

Do the following calculations in Mathematica, and try to understand their structure:

Q1.1 Calculate the numerical value of the Riemann zeta function $\zeta(3)$ with

```
In[3]:= N[Zeta[3]]
```

Q1.2 Square the previous result (`%`) with

```
In[4]:= %^2
```

Q1.3 Calculate $\int_0^\infty \sin(x)e^{-x}dx$ with

```
In[5]:= Integrate[Sin[x]*Exp[-x], {x, 0, Infinity}]
```

Q1.4 Calculate the first 1000 digits of π with

```
In[6]:= N[Pi, 1000]
```

or, equivalently, using the Greek symbol π=Pi,

```
In[7]:= N[π, 1000]
```

Q1.5 Calculate the analytic and numeric values of the Clebsch–Gordan coefficient $\langle 100, 10; 200, -12|110, -2\rangle$:

```
In[8]:= ClebschGordan[{100, 10}, {200, -12}, {110, -2}]
```

Q1.6 Calculate the limit $\lim_{x\to 0} \frac{\sin x}{x}$ with

```
In[9]:= Limit[Sin[x]/x, x -> 0]
```

Q1.7 Make a plot of the above function with

```
In[10]:= Plot[Sin[x]/x, {x, -20, 20}, PlotRange -> All]
```

Q1.8 Draw a Mandelbrot set with

```
In[11]:= F[c_, imax_] := Abs[NestWhile[#^2+c&, 0., Abs[#]<=2&, 1, imax]] <= 2
In[12]:= With[{n = 100, imax = 1000},
         Graphics[Raster[Table[Boole[!F[x+I*y,imax]],{y,-2,2,1/n},{x,-2,2,1/n}]]]]
```

Q1.9 Do the same with a built-in function call:

```
In[13]:= MandelbrotSetPlot[]
```

1.2 variables and assignments

https://reference.wolfram.com/language/howto/WorkWithVariablesAndFunctions.html

Variables in the Wolfram language can be letters or words with uppercase or lowercase letters, including Greek symbols. Assigning a value to a variable is done with the = symbol,

```
In[14]:= a = 5
Out[14]= 5
```

If you wish to suppress the output, then you must end the command with a semi-colon:

```
In[15]:= a = 5;
```

The variable name can then be used anywhere in an expression:

```
In[16]:= a + 2
Out[16]= 7
```

1.2.1 immediate and delayed assignments

https://reference.wolfram.com/language/tutorial/ImmediateAndDelayedDefinitions.html

Consider the two commands

```
In[17]:= a = RandomReal[]
Out[17]= 0.38953
In[18]:= b := RandomReal[]
```

(your random number will be different).

The first statement `a=...` is an *immediate assignment*, which means that its right-hand side is evaluated when you press shift-enter, produces a specific random value, and is assigned to the variable `a` (and printed out). From now on, every time you use the variable `a`, the *exact same* number will be substituted. In this sense, the variable `a` contains the number 0.389 53 and has no memory of where it got this number from. You can check the definition of `a` with `?a`:

```
In[19]:= ?a
         Global‘a
         a = 0.38953
```

The definition `b:=...` is a *delayed assignment*, which means that when you press shift-enter the right-hand side is not evaluated but merely stored as a definition of `b`. From now on, every time you use the variable `b`, its right-hand-side definition will be substituted and executed, resulting in a new random number each time. You can check the definition of `b` with

```
In[20]:= ?b
         Global‘b
         b := RandomReal[]
```

Let's compare the repeated performance of `a` and `b`:

```
In[21]:= {a, b}
Out[21]= {0.38953, 0.76226}
In[22]:= {a, b}
Out[22]= {0.38953, 0.982921}
In[23]:= {a, b}
Out[23]= {0.38953, 0.516703}
In[24]:= {a, b}
Out[24]= {0.38953, 0.0865169}
```

If you are familiar with computer file systems, you can think of an immediate assignments as a *hard link* (a direct link to a precomputed inode number) and a delayed assignment as a *soft link* (symbolic link, textual instructions for how to find the linked target).

1.2.2 exercises

Q1.10 Explain the difference between

```
In[25]:= x = u + v
```

and

```
In[26]:= y := u + v
```

In particular, distinguish the cases where `u` and `v` are already defined before `x` and `y` are defined, where they are defined only afterwards, and where they are defined before but change values after the definition of `x` and `y`.

1.3 four kinds of bracketing

https://reference.wolfram.com/language/tutorial/TheFourKindsOfBracketingInTheWolframLanguage.html

There are four types of brackets in the Wolfram language:

- parentheses for grouping, for example in mathematical expressions:

```
In[27]:= 2*(3-7)
```

- square brackets for function calls:

```
In[28]:= Sin[0.2]
```

- curly braces for lists:

```
In[29]:= v = {a, b, c}
```

- double square brackets for indexing within lists: (see section 1.10)

```
In[30]:= v[[2]]
```

1.4 prefix and postfix

There are several ways of evaluating a function call in the Wolfram language, and we will see most of them in this lecture. As examples of function calls with a single argument, the main ways in which sin(0.2) and $\sqrt{2+3}$ can be calculated are

standard notation (infinite precedence):

```
In[31]:= Sin[0.2]
Out[31]= 0.198669
In[32]:= Sqrt[2+3]
Out[32]= Sqrt[5]
```

prefix notation with `@` (quite high precedence, higher than multiplication):

```
In[33]:= Sin @ 0.2
Out[33]= 0.198669
In[34]:= Sqrt @ 2+3
Out[34]= 3+Sqrt[2]
```

Notice how the high precedence of the `@` operator effectively evaluates `(Sqrt@2)+3`, not `Sqrt@(2+3)`.

postfix notation with `//` (quite low precedence, lower than addition):

```
In[35]:= 0.2 //Sin
Out[35]= 0.198669
In[36]:= 2+3 //Sqrt
Out[36]= Sqrt[5]
```

Notice how the low precedence of the `//` operator effectively evaluates `(2+3)//N`, not `2+(3//N)`.

Postfix notation is often used to transform the output of a calculation:

- Adding `//N` to the end of a command will convert the result to decimal representation, if possible.

- Adding `//MatrixForm` to the end of a matrix calculation will display the matrix in a tabular form.
- Adding `//Timing` to the end of a calculation will display the result together with the amount of time it took to execute.

If you are not sure which form is appropriate, for example if you don't know the precedence of the involved operations, then you should use the standard notation or place parentheses where needed.

1.4.1 exercises

Q1.11 Calculate the decimal value of Euler's constant e (`E`) using standard, prefix, and postfix notation.

1.5 programming constructs

When you program in the Wolfram language you can choose between a number of different programming paradigms, and you can mix these as you like. Depending on the chosen style, your program may run much faster or much slower.

1.5.1 procedural programming

https://reference.wolfram.com/language/guide/ProceduralProgramming.html

A subset of the Wolfram language behaves very similarly to C, Python, Java, or other procedural programming languages. Be very careful to distinguish semi-colons, which separate commands within a single block of code, from commas, which separate different code blocks!

Looping constructs behave like in common programming languages:

```
In[37]:= For[i = 1, i <= 5, i++,
           Print[i]]
         1
         2
         3
         4
         5
```

Notice that `i` is now a globally defined variable, which you can check with

```
In[38]:= ?i
         Global`i
         i=6
```

The following, on the other hand, does not define the value of the variable `j` in the global context:

```
In[39]:= Do[Print[j], {j, 1, 5}]
         1
         2
         3
         4
         5
In[40]:= ?j
         Global`j
```

In this sense, `j` is a local variable in the `Do` context. The following, again, defines `k` as a global variable:

```
In[41]:= k = 1;
         While[k <= 5,
           Print[k];
           k++]
         1
         2
         3
         4
         5
In[42]:= ?k
         Global`k
         k=6
```

Conditional execution: The conditional statement `If[condition, do-when-true, do-when-false]` follows the same logic as in every other programming language,

```
In[43]:= If[5! > 100,
           Print["larger"],
           Print["smaller or equal"]]
         larger
```

Notice that the `If` statement has a return value, similar to the "?" statement of C and Java:

```
In[44]:= a = If[5! > 100, 1, -1]
Out[44]= 1
```

Apart from *true* and *false*, Mathematica statements can have a third state: *unknown*. For example, the comparison `x==0` evaluates to neither true nor false if `x` is not defined. The fourth slot in the `If` statement covers this case:

```
In[45]:= x == 0
Out[45]= x == 0
In[46]:= If[x == 0, "zero", "nonzero", "unknown"]
Out[46]= "unknown"
```

Modularity: code can use local variables within a *module*:

```
In[47]:= Module[{i},
           i = 1;
           While[i > 1/192, i = i/2];
           i]
Out[47]= 1/256
```

After the execution of this code, the variable `i` is still undefined in the global context.

1.5.2 exercises

Q1.12 Write a program that sums all integers from 123 to 9968. Use only local variables.

Q1.13 Write a program that sums consecutive integers, starting from 123, until the sum is larger than 10 000. Return the largest integer in this sum. Use only local variables.

1.5.3 functional programming

https://reference.wolfram.com/language/guide/FunctionalProgramming.html

Functional programming is a very powerful programming technique that can give large speedups in computation because it can often be parallelized over many computers or CPUs. In our context, we often use lists (vectors or matrices, see section 1.10) and want to apply functions to each one of their elements.

The most common functional programming constructs are

Anonymous functions: [1] you can quickly define a function with parameters `#1`, `#2`, `#3`, etc., terminated with the `&` symbol: (the symbol `#` is an abbreviation for `#1`)

```
In[48]:= f = #^2 &;
In[49]:= f[7]
Out[49]= 49
In[50]:= g = #1-#2 &;
In[51]:= g[88, 9]
Out[51]= 79
```

Functions and anonymous functions, for example `#^2&`, are first-class objects[2] just like numbers, matrices, etc. You can assign them to variables, as in `In[48]` and `In[50]` above; you can also use them directly as arguments to other functions, as for example in `In[55]` below; or you can use them as return values of other functions, as in `In[478]`.

The symbol `##` stands for the sequence of all parameters of a function:

```
In[52]:= f = {1,2,3,##,4,5,6} &;
In[53]:= f[7,a,c]
Out[53]= {1,2,3,7,a,c,4,5,6}
```

The symbol `#0` stands for the function itself. This is useful for defining recursive anonymous functions (see item 7 of section 1.9).

Map `/@`: apply a function to each element of a list.

```
In[54]:= a = {1, 2, 3, 4, 5, 6, 7, 8};
In[55]:= Map[#^2 &, a]
Out[55]= {1, 4, 9, 16, 25, 36, 49, 64}
In[56]:= #^2 & /@ a
Out[56]= {1, 4, 9, 16, 25, 36, 49, 64}
```

Notice how we have used the anonymous function `#^2&` here without ever giving it a name.

Apply `@@`: apply a function to an entire list and generate a single result. For example, applying `Plus` to a list will calculate the sum of the list elements; applying `Times` will calculate their product. This operation is also known as *reduce*.[3]

```
In[57]:= a = {1, 2, 3, 4, 5, 6, 7, 8};
In[58]:= Apply[Plus, a]
Out[58]= 36
In[59]:= Plus @@ a
Out[59]= 36
In[60]:= Apply[Times, a]
Out[60]= 40320
In[61]:= Times @@ a
Out[61]= 40320
```

[1] See https://en.wikipedia.org/wiki/Anonymous_functions.
[2] See https://en.wikipedia.org/wiki/First-class_citizen.
[3] See https://en.wikipedia.org/wiki/MapReduce.

1.5.4 exercises

Q1.14 Write an anonymous function with three arguments that returns the product of these arguments.

Q1.15 Given a list

```
In[62]:= a = {0.1, 0.9, 2.25, -1.9};
```

calculate $x \mapsto \sin(x^2)$ for each element of `a` using the `Map` operation.

Q1.16 Calculate the sum of all the results of **Q1.15**.

1.6 function definitions

https://reference.wolfram.com/language/tutorial/DefiningFunctions.html

Functions are assignments (see section 1.2) with parameters. As for parameter-free assignments, we distinguish between *immediate* and *delayed* function definitions.

1.6.1 immediate function definitions

We start with *immediate* definitions: a function $f(x) = \sin(x)/x$ is defined with

```
In[63]:= f[x_] = Sin[x]/x;
```

Notice the underscore `_` symbol after the variable name `x`: this underscore indicates a *pattern* (denoted by `_`) named `x`, not the symbol `x` itself. Whenever this function `f` is called with any parameter value, this parameter value is inserted wherever `x` appears on the right-hand side, as is expected for a function definition. You can find out how `f` is defined with the `?` operator:

```
In[64]:= ?f
         Global‘f
         f[x_] = Sin[x]/x
```

and you can ask for a function evaluation with

```
In[65]:= f[0.3]
Out[65]= 0.985067
In[66]:= f[0]
         Power: Infinite expression 1/0 encountered.
         Infinity: Indeterminate expression 0 ComplexInfinity encountered.
Out[66]= Indeterminate
```

Apparently the function cannot be evaluated for $x = 0$. We can fix this by defining a special function value:

```
In[67]:= f[0] = 1;
```

Notice that there is no underscore on the left-hand side, so there is no pattern definition. The full definition of `f` is now

```
In[68]:= ?f
         Global‘f
         f[0] = 1
         f[x_] = Sin[x]/x
```

If the function `f` is called, then these definitions are checked in order of appearance in this list. For example, if we ask for `f[0]`, then the first entry matches and the value 1 is returned. If we ask for `f[0.3]`, then the first entry does not match (since 0 and 0.3 are not strictly equal), but the second entry matches since anything can be plugged into the pattern named `x`. The result is $\sin(0.3)/0.3 = 0.985\,067$, which is what we expected.

1.6.2 delayed function definitions

Just like with delayed assignments (section 1.2.1), we can define delayed function calls. For comparison, we define the two functions

```
In[69]:= g1[x_] = x + RandomReal[]
Out[69]= 0.949868 + x
In[70]:= g2[x_] := x + RandomReal[]
```

Check their effective definitions with `?g1` and `?g2`, and notice that the definition of `g1` was executed immediately when you pressed shift-enter and its result assigned to the function `g1` (with a specific value for the random number, as printed out), whereas the definition of `g2` was left unevaluated and is executed each time anew when you use the function `g2`:

```
In[71]:= {g1[2], g2[2]}
Out[71]= {2.94987, 2.33811}
In[72]:= {g1[2], g2[2]}
Out[72]= {2.94987, 2.96273}
In[73]:= {g1[2], g2[2]}
Out[73]= {2.94987, 2.18215}
```

1.6.3 memoization: functions that remember their results

https://reference.wolfram.com/language/tutorial/FunctionsThatRememberValuesTheyHaveFound.html

When we define a function that takes a long time to evaluate, we may wish to store its output values such that if the function is called with identical parameter values again, then we do not need to re-evaluate the function but can simply remember the already calculated result.[4] We can make use of the interplay between patterns and values, and between immediate and delayed assignments, to construct such a function that remembers its values from previous function calls.

See if you can understand the following definition.

```
In[74]:= F[x_] := F[x] = x^7
```

If you ask for `?F` then you will simply see this definition. Now call

```
In[75]:= F[2]
Out[75]= 128
```

and ask for `?F` again. You see that the specific immediate definition of `F[2]=128` was added to the list of definitions, with the evaluated result 128 (which may have taken a long time to calculate in a more complicated function). The next time you call `F[2]`, the specific definition of `F[2]` will be found earlier in the definitions list than the general definition `F[x_]` and therefore the precomputed value of `F[2]` will be returned.

When you re-define the function `F` after making modifications to it, you must clear the associated remembered values in order for them to be re-computed at the next occasion. It is a good practice to prefix every definition of a memoizing function with a `Clear` command:

```
In[76]:= Clear[F];
In[77]:= F[x_] := F[x] = x^9
```

For function evaluations that take even longer, we may wish to save the accumulated results to a file in order to read them back at a later time. For the above example, we save all definitions associated with the symbol `F` to the file `Fdef.mx` with

[4] This is technically called *memoization*: https://en.wikipedia.org/wiki/Memoization. A similar functionality can be achieved with Mathematica's `Once` operator, which allows fine-grained control over the storage location, conditions, and duration of the persistent result.

```
In[78]:= SetDirectory[NotebookDirectory[]];
In[79]:= DumpSave["Fdef.mx", F];
```

The next time we wish to continue the calculation, we define the function `F` and load all of its already known values with

```
In[80]:= Clear[F];
In[81]:= F[x_] := F[x] = x^9
In[82]:= SetDirectory[NotebookDirectory[]];
In[83]:= Get["Fdef.mx"];
```

1.6.4 functions with conditions on their arguments

https://reference.wolfram.com/language/guide/Patterns.html

The Wolfram language contains a powerful pattern language that we can use to define functions that only accept certain arguments. For function definitions we will use three main types of patterns:

Anything-goes: A function defined as

```
In[84]:= f[x_] := x^2
```

can be called with any sort of arguments, since the pattern `x_` can match *anything*:

```
In[85]:= f[4]
Out[85]= 16
In[86]:= f[2.3-0.1I]
Out[86]= 5.28-0.46I
In[87]:= f[{1,2,3,4}]
Out[87]= {1,4,9,16}
In[88]:= f[y^2]
Out[88]= y^4
```

Type-restricted: A pattern like `x_Integer` will only match arguments of integer type. If the function is called with a non-matching argument, then the function is not executed:

```
In[89]:= g[x_Integer] := x-3
In[90]:= g[x_Rational] := x
In[91]:= g[x_Real] := x+3
In[92]:= g[x_Complex] := 0
In[93]:= g[7]
Out[93]= 4
In[94]:= g[7.1]
Out[94]= 10.1
In[95]:= g[2/3]
Out[95]= 2/3
In[96]:= g[2+3I]
Out[96]= 0
In[97]:= g[x]
Out[97]= g[x]
```

Conditional: Complicated conditions can be specified with the `/;` operator:

```
In[98]:= h[x_/;x<=3] := x^2
In[99]:= h[x_/;x>3] := x-11
In[100]:=h[2]
Out[100]=4
In[101]:=h[5]
Out[101]=-6
```

Conditions involving a single function call returning a Boolean value, for example `x_/;PrimeQ[x]`, can be abbreviated with `x_?PrimeQ`. Other useful "question" functions are `IntegerQ`, `NumericQ`, `EvenQ`, `OddQ`, etc. See https://reference.wolfram.com/language/tutorial/PuttingConstraintsOnPatterns.html for more information.

1.6.5 functions with optional arguments

https://reference.wolfram.com/language/tutorial/OptionalAndDefaultArguments.html

Function arguments can be optional, indicated with the `:` symbol. For each optional argument, a default value must be defined that is used whenever the function is called without the argument specified. The optional arguments must be the last ones in the arguments list. There can be arbitrarily many optional arguments.

As an example, the function

```
In[102]:=f[a_, b_:5] = {a,b}
```

uses the default value $b = 5$ whenever it is called with only one argument:

```
In[103]:=f[7]
Out[103]={7,5}
```

When called with two arguments, the second argument overrides the default value for b:

```
In[104]:=f[7,2]
Out[104]={7,2}
```

1.7 rules and replacements

https://reference.wolfram.com/language/tutorial/ApplyingTransformationRules.html

We will often use replacement rules in the calculations of this course. A replacement rule is an instruction `x -> y` that replaces any occurrence of the symbol (or pattern) `x` with the symbol `y`. We apply such a rule with the `/.` or `ReplaceAll` operator:

```
In[105]:=a + 2 /. a -> 7
Out[105]=9
In[106]:=ReplaceAll[a + 2, a -> 7]
Out[106]=9
In[107]:=c - d /. {c -> 2, d -> 8}
Out[107]=-6
In[108]:=ReplaceAll[c - d, {c -> 2, d -> 8}]
Out[108]=-6
```

Rules can contain patterns, in the same way as we use them for defining the parameters of functions (section 1.6):

```
In[109]:=a + b /. x_ -> x^2
Out[109]=(a + b)^2
```

Notice that here the pattern `x_` matched the entire expression `a + b`, not the subexpressions `a` and `b`. To be more specific and do the replacement only at level 1 of this expression, we can write

```
In[110]:=Replace[a + b, x_ -> x^2, {1}]
Out[110]=a^2 + b^2
```

Doing the replacement at level 0 gives again

```
In[111]:=Replace[a + b, x_ -> x^2, {0}]
Out[111]=(a + b)^2
```

At other instances, restricted patterns can be used to achieve a desired result:

```
In[112]:=a + 2 /. x_Integer -> x^2
Out[112]=4 + a
```

Many Wolfram language functions return their results as replacement rules. For example, the result of solving an equation is a list of rules:

```
In[113]:=s = Solve[x^2 - 4 == 0, x]
Out[113]={{x -> -2}, {x -> 2}}
```

We can make use of these solutions with the replacement operator `/.`, for example to check the solutions:

```
In[114]:=x^2 - 4 /. s
Out[114]={0, 0}
```

1.7.1 immediate and delayed rules

Just as for assignments (section 1.2.1) and functions (section 1.6), rules can be immediate or delayed. In an immediate rule of the form `x -> y`, the value of `y` is calculated once upon defining the rule. In a delayed rule of the form `x :> y`, the value of `y` is re-calculated every time the rule is applied. This can be important when the rule is supposed to perform an action. Here is an example: we replace `c` by `f` with

```
In[115]:={a, b, c, d, c, a, c, b} /. c -> f
Out[115]={a, b, f, d, f, a, f, b}
```

We do the same while counting the number of replacements with

```
In[116]:=i = 0;
In[117]:={a, b, c, d, c, a, c, b} /. c :> (i++; Echo[i, "replacement "]; f)
       » replacement 1
       » replacement 2
       » replacement 3
Out[117]={a, b, f, d, f, a, f, b}
In[118]:=i
Out[118]=3
```

In this case, the delayed rule `c :> (i++; Echo[i, "replacement "]; f)` is a list of commands enclosed in parentheses `()` and separated by semicolons. The first command increments the replacement counter `i`, the second prints a running commentary (see section 1.8), and the third gives the result of the replacement. The result of such a list of commands is always the last expression, in this case `f`.

1.7.2 repeated rule replacement

The /. operator uses the given list of replacement rules only once:

```
In[119]:= a /. {a -> b, b -> c}
Out[119]= b
```

The //. operator, on the other hand, uses the replacement rules repeatedly until the result no longer changes (in this case, after two applications):

```
In[120]:= a //. {a -> b, b -> c}
Out[120]= c
```

1.8 debugging and finding out how Mathematica expressions are evaluated

https://reference.wolfram.com/language/guide/TuningAndDebugging.html

https://www.wolfram.com/language/elementary-introduction/2nd-ed/47-debugging-your-code.html

The precise way Mathematica evaluates an expression depends on many details and can become very complicated.[5] For finding out more about particular cases, especially when they aren't evaluated in the way that you were expecting, the Trace command may be useful. This command gives a list of all intermediate results, which helps in understanding the way that Mathematica arrives at its output:

```
In[121]:= Trace[x - 3x + 1]
Out[121]= {{-(3x), -3x, -3x}, x-3x+1, 1-3x+x, 1-2x}
In[122]:= x = 5;
In[123]:= Trace[x - 3x + 1]
Out[123]= {{x, 5}, {{{x, 5}, 3×5, 15}, -15, -15}, 5-15+1, -9}
```

A more verbose trace is achieved with TracePrint:

```
In[124]:= TracePrint[y - 3y + 1]
      y-3 y+1
       Plus
       y
       -(3 y)
        Times
        -1
        3 y
         Times
         3
         y
       -3 y
       -3 y
        Times
        -3
        y
       1
      y-3 y+1
      1-3 y+y
      1-2 y
       Plus
       1
       -2 y
        Times
```

[5] See https://reference.wolfram.com/language/tutorial/EvaluationOfExpressionsOverview.html.

```
          -2
          y
Out[124]= 1 - 2 y
```

It is very useful to print out intermediate results in a long calculation via the `Echo` command, particularly during code development. Calling `Echo[x,label]` prints `x` with the given label, and returns `x`; in this way, the `Echo` command can be simply added to a calculation without perturbing it:

```
In[125]:= Table[Echo[i!, "building table: "], {i, 3}]
       » building table: 1
       » building table: 2
       » building table: 6
Out[125]= {1, 2, 6}
```

In order to run your code "cleanly" after debugging it with `Echo`, you can either remove all instances of `Echo`, or you can re-define `Echo` to do nothing:

```
In[126]:= Unprotect[Echo]; Echo = #1 &;
```

Re-running the code of `In[125]` now gives just the result:

```
In[127]:= Table[Echo[i!, "building table: "], {i, 3}]
Out[127]= {1, 2, 6}
```

Finally, it can be very insightful to study the "full form" of expressions, especially when it does not match a pattern that you were expecting to match. For example, the internal full form of ratios depends strongly on the type of numerator or denominator:

```
In[128]:= FullForm[a/b]
Out[128]= Times[a, Power[b, -1]]
In[129]:= FullForm[1/2]
Out[129]= Rational[1, 2]
In[130]:= FullForm[a/2]
Out[130]= Times[Rational[1, 2], a]
In[131]:= FullForm[1/b]
Out[131]= Power[b, -1]
```

1.8.1 exercises

Q1.17 Why do we need the `Unprotect` command in `In[126]`?

Q1.18 To replace a ratio a/b by the function `ratio[a,b]`, we could enter

```
In[132]:= a/b /. {x_/y_ -> ratio[x,y]}
Out[132]= ratio[a,b]
```

Why does this not work to replace the ratio 2/3 by the function `ratio[2,3]`?

```
In[133]:= 2/3 /. {x_/y_ -> ratio[x,y]}
Out[133]= 2/3
```

1.9 many ways to define the factorial function [code]

The following list of definitions of the factorial function is based on the Wolfram demo https://www.wolfram.com/training/videos/EDU002/. Try to understand as many of these definitions as possible. What this means in practice is that for most problems you can pick the programming paradigm that suits your way of thinking best, instead of being forced into one way or another. The different paradigms have different advantages and disadvantages, which may become clearer to you as you become more familiar with them.

You must call `Clear[f]` between different definitions!

1. Define the function `f` to be an alias of the built-in function `Factorial`: calling `f[5]` is now strictly the same thing as calling `Factorial[5]`, which in turn is the same thing as calling `5!`.

```
In[134]:= f = Factorial;
```

2. A call to `f` is forwarded to the function "`!`": calling `f[5]` triggers the evaluation of `5!`.

```
In[135]:= f[n_] := n!
```

3. Use the mathematical definition $n! = \Gamma(n+1)$:

```
In[136]:= f[n_] := Gamma[n+1]
```

4. Use the mathematical definition $n! = \prod_{i=1}^{n} i$:

```
In[137]:= f[n_] := Product[i, {i,n}]
```

5. Rule-based recursion, using the Wolfram language's built-in pattern-matching capabilities: calling `f[5]` leads to a call of `f[4]`, which leads to a call of `f[3]`, and so on until `f[1]` immediately returns the result 1, after which the program unrolls the recursion stack and does the necessary multiplications:

```
In[138]:= f[1] = 1;
In[139]:= f[n_] := n*f[n-1]
```

6. The same recursion but without rules (no pattern-matching):

```
In[140]:= f[n_] := If[n == 1, 1, n*f[n-1]]
```

7. Define the same recursion through functional programming: `f` is a function whose name is `#0` and whose first (and only) argument is `#1`. The end of the function definition is marked with `&`.

```
In[141]:= f = If[#1 == 1, 1, #1*#0[#1-1]]&;
```

8. procedural programming with a `Do` loop:

```
In[142]:= f[n_] := Module[{t = 1},
            Do[t = t*i, {i, n}];
            t]
```

9. procedural programming with a `For` loop: this is how you would compute factorials in procedural programming languages like C. It is a very precise step-by-step prescription of how exactly the computer is supposed to do the calculation.

```
In[143]:=f[n_] := Module[{t = 1, i},
          For[i = 1, i <= n, i++,
            t *= i];
          t]
```

10. Make a list of the numbers $1 \ldots n$ (with `Range[n]`) and then multiply them together at once, by applying the function `Times` to this list. This is the most elegant way of multiplying all these numbers together, because both the generation of the list of integers and their multiplication are done with internally optimized methods. The programmer merely specifies *what* he would like the computer to do, and not *how* it is to be done.

```
In[144]:=f[n_] := Times @@ Range[n]
```

11. Make a list of the numbers $1 \ldots n$ and then multiply them together one after the other.

```
In[145]:=f[n_] := Fold[Times, 1, Range[n]]
```

12. Functional programming: make a list of functions $\{t \mapsto t, t \mapsto 2t, t \mapsto 3t, \ldots, t \mapsto nt\}$, and then, starting with the number 1, apply each of these functions once.

```
In[146]:=f[n_] := Fold[#2[#1]&, 1, Array[Function[t, #1*t]&, n]]
```

13. Construct a list whose length we know to be $n!$:

```
In[147]:=f[n_] := Length[Permutations[Range[n]]]
```

14. Use repeated pattern-based replacement (`//.`, see section 1.7.2) to find the factorial: start with the object $\{1, n\}$ and apply the given rule until the result no longer changes because the pattern no longer matches.

```
In[148]:=f[n_] := First[{1,n} //. {a_,b_/;b>0} :> {b*a,b-1}]
```

15. Build a string whose length is $n!$:

```
In[149]:=f[n_] := StringLength[Fold[StringJoin[Table[#1, {#2}]]&, "A", Range[n]]]
```

16. Starting from the number n, repeatedly replace each number m by a list containing m times the number $m-1$. At the end, we have a list of lists of ... of lists that overall contains $n!$ times the number 1. Flatten it out and count the number of elements.

```
In[150]:=f[n_] := Length[Flatten[n //. m_ /; m > 1 :> Table[m - 1, {m}]]]
```

17. Analytically calculate $\frac{\mathrm{d}^n(x^n)}{\mathrm{d}x^n}$, the n^{th} derivative of x^n:

```
In[151]:=f[n_] := D[x^n, {x, n}]
```

1.9.1 exercises

Q1.19 In which ones of the definitions of section 1.9 can you replace a delayed assignment (:=) with an immediate assignment (=) or vice-versa? What changes if you do this replacement? (see section 1.2.1)

Q1.20 In which ones of the definitions of section 1.9 can you replace a delayed rule (:>) with an immediate rule (->) or vice-versa? What changes if you do this replacement? (see section 1.7.1)

Q1.21 Can you use the trick of section 1.6.3 for any of the definitions of section 1.9?

Q1.22 Write two very different programs that calculate the first hundred Fibonacci numbers $\{1, 1, 2, 3, 5, 8, \dots\}$, where each number is the sum of the two preceding ones.

1.10 vectors, matrices, tensors

In this lecture we will use vectors and matrices to represent quantum states and operators, respectively.

1.10.1 vectors

https://reference.wolfram.com/language/tutorial/VectorOperations.html

In the Wolfram language, vectors are represented as lists of objects, for example lists of real or complex numbers:

```
In[152]:= v = {1,2,3,2,1,7+I};
In[153]:= Length[v]
Out[153]= 6
```

You can access any element by its index, using double brackets, with the first element having index 1 (as in Fortran or Matlab), *not* 0 (as in C, Java, or Python):

```
In[154]:= v[[4]]
Out[154]= 2
```

Negative indices count from the end of the list:

```
In[155]:= v[[-1]]
Out[155]= 7+I
```

Lists can contain arbitrary elements (for example strings, graphics, expressions, lists, functions, etc.).

If two vectors $\vec{a}$ and $\vec{b}$ of equal length are defined, then their scalar product $\vec{a}^* \cdot \vec{b}$ is calculated with

```
In[156]:= a = {0.1, 0.2, 0.3 + 2I};
In[157]:= b = {-0.27I, 0, 2};
In[158]:= Conjugate[a].b
Out[158]= 0.6 - 4.027I
```

Vectors of equal length can be element-wise added, subtracted, multiplied etc. with the usual operators:

```
In[159]:= a + b
Out[159]= {0.1 - 0.27I, 0.2, 2.3 + 2.I}
In[160]:= 2 a
Out[160]= {0.2, 0.4, 0.6 + 4.I}
```

1.10.2 matrices

https://reference.wolfram.com/language/tutorial/BasicMatrixOperations.html

Matrices are lists of lists, where each sublist describes a row of the matrix:

```
In[161]:= M = {{3,2,7},{1,1,2},{0,-1,5},{2,2,1}};
In[162]:= Dimensions[M]
Out[162]= {4, 3}
```

In this example, `M` is a 4×3 matrix. Pretty-printing a matrix is done with the MatrixForm wrapper,

```
In[163]:= MatrixForm[M]
```

Accessing matrix elements is analogous to accessing vector elements:

```
In[164]:= M[[1,3]]
Out[164]= 7
In[165]:= M[[2]]
Out[165]= {1, 1, 2}
```

Matrices can be transposed with `Transpose[M]`.

Matrix–vector and matrix–matrix multiplications are done with the `.` operator:

```
In[166]:= M.a
Out[166]= {2.8 + 14.I, 0.9 + 4.I, 1.3 + 10.I, 0.9 + 2.I}
```

1.10.3 sparse vectors and matrices

https://reference.wolfram.com/language/guide/SparseArrays.html

Large matrices can take up enormous amounts of computer memory. In practical situations we are often dealing with matrices that are "sparse", meaning that most of their entries are zero. A much more efficient way of storing them is therefore as a list of only their nonzero elements, using the `SparseArray` function.

A given vector or matrix is converted to sparse representation with

```
In[167]:= M = {{0,3,0,0,0,0,0,0,0,0},
              {0,0,0,-1,0,0,0,0,0,0},
              {0,0,0,0,0,0,0,0,0,0}};
In[168]:= Ms = SparseArray[M]
Out[168]= SparseArray[<2>, {3, 10}]
```

where the output shows that `Ms` is a 3×10 sparse matrix with 2 non-zero entries. We could have entered this matrix more easily by giving the list of non-zero entries,

```
In[169]:= Ms = SparseArray[{{1, 2} -> 3, {2, 4} -> -1}, {3, 10}];
```

which we can find out from

```
In[170]:= ArrayRules[Ms]
Out[170]= {{1, 2} -> 3, {2, 4} -> -1, {_, _} -> 0}
```

which includes a specification of the default pattern `{_,_}`. This sparse array is converted back into a normal array with

```
In[171]:= Normal[Ms]
Out[171]= {{0,3,0,0,0,0,0,0,0,0},
           {0,0,0,-1,0,0,0,0,0,0},
           {0,0,0,0,0,0,0,0,0,0}}
```

Sparse arrays and vectors can be used just like full arrays and vectors (they are internally converted automatically whenever necessary). But for some linear algebra operations they can be much more efficient. A matrix multiplication of two sparse matrices, for example, scales only with the number of non-zero elements of the matrices, not with their size.

1.10.4 matrix diagonalization

"Solving" the time-independent Schrödinger equation, as we will be doing in section 2.2, involves calculating the eigenvalues and eigenvectors of Hermitian[6] matrices.

In what follows it is assumed that we have defined $\boldsymbol{H}$ as a Hermitian matrix. As an example we will use

```
In[172]:= H = {{0,   0.3, I,    0},
               {0.3, 1,   0,    0},
               {-I,  0,   1,    -0.2},
               {0,   0,   -0.2, 3}};
```

eigenvalues

The eigenvalues of a matrix `H` are computed with

```
In[173]:= Eigenvalues[H]
Out[173]= {3.0237, 1.63842, 0.998322, -0.660442}
```

Notice that these eigenvalues (energy values) are not necessarily sorted, even though in this example they appear in descending order. For a sorted list we use

```
In[174]:= Sort[Eigenvalues[H]]
Out[174]= {-0.660442, 0.998322, 1.63842, 3.0237}
```

For very large matrices `H`, and in particular for sparse matrices (see section 1.10.3), it is computationally inefficient to calculate all eigenvalues. Further, we are often only interested in the lowest-energy eigenvalues and eigenvectors. There are very efficient algorithms for calculating extremal eigenvalues,[7] which can be used by specifying options to the `Eigenvalues` function: if we only need the largest two eigenvalue, for example, we call

```
In[175]:= Eigenvalues[H, 2, Method -> {"Arnoldi",
                                       "Criteria" -> "RealPart",
                                       MaxIterations -> 10^6}]
Out[175]= {3.0237, 1.63842}
```

There is no direct way to calculate the *smallest* eigenvalues; but since the smallest eigenvalues of `H` are the largest eigenvalues of `-H` we can use

```
In[176]:= -Eigenvalues[-H, 2, Method -> {"Arnoldi",
                                         "Criteria" -> "RealPart",
                                         MaxIterations -> 10^6}]
Out[176]= {0.998322, -0.660442}
```

[6] A complex matrix $\boldsymbol{H}$ is *Hermitian* if $\boldsymbol{H} = \boldsymbol{H}^\dagger$. See https://en.wikipedia.org/wiki/Hermitian_matrix.
[7] Arnoldi–Lanczos algorithm: https://en.wikipedia.org/wiki/Lanczos_algorithm.

eigenvectors

The eigenvectors of a matrix `H` are computed with

```
In[177]:= Eigenvectors[H]
Out[177]= {{0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264},
          {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379},
          {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187},
          {0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821}}
```

In this case of a 4×4 matrix, this generates a list of four ortho-normal 4-vectors.

Usually we are interested in calculating the eigenvalues and eigenvectors at the same time:

```
In[178]:= Eigensystem[H]
Out[178]= {{3.0237, 1.63842, 0.998322, -0.660442},
           {{0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264},
            {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379},
            {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187},
            {0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821}}}
```

which generates a list containing the eigenvalues and the eigenvectors. The ordering of the elements in the eigenvalues list corresponds to the ordering in the eigenvectors list; but the sorting order is generally undefined. To generate a list of (eigenvalue, eigenvector) pairs in ascending order of eigenvalues, we calculate

```
In[179]:= Sort[Transpose[Eigensystem[H]]]
Out[179]= {{-0.660442, {0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821}},
           {0.998322, {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187}},
           {1.63842, {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379}},
           {3.0237, {0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264}}}
```

To generate a sorted list of eigenvalues `eval` and a corresponding list of eigenvectors `evec` we calculate

```
In[180]:= {eval,evec} = Transpose[Sort[Transpose[Eigensystem[H]]]];
In[181]:= eval
Out[181]= {-0.660442, 0.998322, 1.63842, 3.0237}
In[182]:= evec
Out[182]= {{0.-0.844772I, 0.+0.152629I, 0.512134, 0.0279821},
           {0.-0.0053472I, 0.+0.955923I, -0.292115, -0.029187},
           {0.+0.533642I, 0.+0.250762I, 0.799103, 0.117379},
           {0.-0.0394613I, 0.-0.00584989I, -0.117564, 0.992264}}
```

The trick with calculating only the lowest-energy eigenvalues can be applied to eigenvalue calculations as well, since the eigenvectors of `-H` and `H` are the same:

```
In[183]:= {eval,evec} = Transpose[Sort[Transpose[-Eigensystem[-H, 2,
           Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}]]]];
In[184]:= eval
Out[184]= {-0.660442, 0.998322}
In[185]:= evec
Out[185]= {{-0.733656+0.418794I, 0.132553-0.0756656I,
            -0.253889-0.444771I, -0.0138721-0.0243015 I},
           {-0.000575666-0.00531612I, 0.102912+0.950367I,
            -0.290417+0.0314484I, -0.0290174+0.0031422I}}
```

Notice that these eigenvectors are not the same as those calculated further above! This difference is due to arbitrary multiplications of the eigenvectors with phase factors $e^{i\varphi}$.

To check that the vectors in `evec` are ortho-normalized, we calculate the matrix product

```
In[186]:= Conjugate[evec].Transpose[evec] //Chop //MatrixForm
```

and verify that the matrix of scalar products is indeed equal to the unit matrix.

To check that the vectors in `evec` are indeed eigenvectors of `H`, we calculate all matrix elements of `H` in this basis of eigenvectors:

```
In[187]:= Conjugate[evec].H.Transpose[evec] //Chop //MatrixForm
```

and verify that the result is a diagonal matrix whose diagonal elements are exactly the eigenvalues `eval`.

1.10.5 tensor operations

https://reference.wolfram.com/language/guide/RearrangingAndRestructuringLists.html

We have seen above that in the Wolfram language, a vector is a list of numbers (section 1.10.1) and a matrix is a list of lists of numbers (section 1.10.2). Higher-rank tensors are correspondingly represented as lists of lists of . . . of lists of numbers. In this section we describe general tools for working with tensors, which extend the methods used for vectors and matrices. See section 2.4.3 for a concrete application of higher-rank tensors. We note that the sparse techniques of section 1.10.3 naturally extend to higher-rank tensors.

As an example, we start by defining a list (*i.e.*, a vector) containing 24 elements:

```
In[188]:= v = Range[24]
Out[188]= {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24}
In[189]:= Dimensions[v]
Out[189]= {24}
```

We have chosen the elements in this vector to indicate their position in order to make the following transformations easier to understand.

reshaping

We reshape the list `v` into a $2 \times 3 \times 4$ tensor with

```
In[190]:= t = ArrayReshape[v, {2,3,4}]
Out[190]= {{{1,2,3,4},{5,6,7,8},{9,10,11,12}},
          {{13,14,15,16},{17,18,19,20},{21,22,23,24}}}
In[191]:= Dimensions[t]
Out[191]= {2, 3, 4}
```

Notice that the order of the elements has not changed; but they are now arranged as a list of lists of lists of numbers. Alternatively, we could reshape `v` into a $2 \times 2 \times 3 \times 2$ tensor with

```
In[192]:= u = ArrayReshape[v, {2,2,3,2}]
Out[192]= {{{{1,2},{3,4},{5,6}},{{7,8},{9,10},{11,12}}},
          {{{13,14},{15,16},{17,18}},{{19,20},{21,22},{23,24}}}}
In[193]:= Dimensions[u]
Out[193]= {2, 2, 3, 2}
```

flattening

The reverse operation is called flattening:

```
In[194]:= Flatten[t] == Flatten[u] == v
Out[194]= True
```

Tensor flattening can be applied more specifically, without flattening the entire structure into a single list. As an example, in `u` we flatten indices 1&2 together and indices 3&4 together, to find a 4×6 matrix that we could have calculated directly with `ArrayReshape[v, {4,6}]`:

```
In[195]:=Flatten[u, {{1,2}, {3,4}}]
Out[195]={{1,2,3,4,5,6},{7,8,9,10,11,12},{13,14,15,16,17,18},{19,20,21,22,23,24}}
In[196]:=% == ArrayReshape[v, {4,6}]
Out[196]=True
```

We sometimes use the `ArrayFlatten` command, which is just a special case of `Flatten` with fixed arguments, flattening indices 1&3 together and indices 2&4 together:

```
In[197]:=ArrayFlatten[u] == Flatten[u, {{1,3}, {2,4}}]
Out[197]=True
```

transposing

A tensor transposition is a re-ordering of a tensor's indices. For example,

```
In[198]:=tt = Transpose[t, {2,3,1}]
Out[198]={{{1,5,9},{13,17,21}},{{2,6,10},{14,18,22}},
        {{3,7,11},{15,19,23}},{{4,8,12},{16,20,24}}}
In[199]:=Dimensions[tt]
Out[199]={4, 2, 3}
```

generates a $4 \times 2 \times 3$-tensor `tt`, where the first index of `t` is the second index of `tt`, the second index of `t` is the third index of `tt`, and the third index of `t` is the first index of `tt`; this order of index shuffling is given in the parameter list `{2,3,1}` meaning $\{1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}\} \mapsto \{2^{\text{nd}}, 3^{\text{rd}}, 1^{\text{st}}\}$. More explicitly,

```
In[200]:=Table[t[[i,j,k]] == tt[[k,i,j]], {i,2}, {j,3}, {k,4}]
Out[200]={{{True,True,True,True},{True,True,True,True},
          {True,True,True,True}},{{True,True,True,True},
          {True,True,True,True},{True,True,True,True}}}
```

contracting

As a generalization of a scalar product, indices of equal length of a tensor can be contracted. This is the operation of summing over an index that appears twice in the list of indices. For example, contracting indices 2 and 5 of the rank-6 tensor $X_{a,b,c,d,e,f}$ yields the rank-4 tensor with elements $Y_{a,c,d,f} = \sum_i X_{a,i,c,d,i,f}$.

For example, we can either contract indices 1&2 in `u`, or indices 1&4, or indices 2&4, since they are all of length 2:

```
In[201]:=TensorContract[u, {1, 2}]
Out[201]={{20, 22}, {24, 26}, {28, 30}}
In[202]:=TensorContract[u, {1, 4}]
Out[202]={{15, 19, 23}, {27, 31, 35}}
In[203]:=TensorContract[u, {2, 4}]
Out[203]={{9, 13, 17}, {33, 37, 41}}
```

1.10.6 exercises

Q1.23 Calculate the eigenvalues and eigenvectors of the Pauli matrices:
https://en.wikipedia.org/wiki/Pauli_matrices
Are the eigenvectors ortho-normal? If not, find an ortho-normal set.

Q1.24 After `In[203]`, try to contract indices 3&4 in the tensor `u`. What went wrong?

1.11 complex numbers

By default all variables in the Wolfram language are assumed to be complex numbers, unless otherwise specified. All mathematical functions can take complex numbers as their input, often by analytic continuation.[8]

The most commonly used functions on complex numbers are `Conjugate`, `Re`, `Im`, `Abs`, and `Arg`. When applied to numerical arguments they do what we expect:

```
In[204]:= Conjugate[2 + 3I]
Out[204]= 2 - 3I
In[205]:= Im[0.7]
Out[205]= 0
```

When applied to variable arguments, however, they fail and frustrate the inexperienced user:

```
In[206]:= Conjugate[x+I*y]
Out[206]= Conjugate[x] - I*Conjugate[y]
In[207]:= Im[a]
Out[207]= Im[a]
```

This behavior is due to Mathematica not knowing that `x`, `y`, and `a` in these examples are real-valued. There are several ways around this, all involving *assumptions*. The first is to use the `ComplexExpand` function, which assumes that all variables are *real*:

```
In[208]:= Conjugate[x+I*y] //ComplexExpand
Out[208]= x - I*y
In[209]:= Im[a] //ComplexExpand
Out[209]= 0
```

The second is to use explicit *local* assumptions, which may be more specific than assuming that all variables are real-valued:

```
In[210]:= Assuming[Element[x, Reals] && Element[y, Reals],
            Conjugate[x + I*y] //FullSimplify]
Out[210]= x - I*y
In[211]:= Assuming[Element[a, Reals], Im[a] //FullSimplify]
Out[211]= 0
```

The third is to use *global* assumptions (in general, global system variables start with the `$` sign):

```
In[212]:= $Assumptions = Element[x, Reals] && Element[y, Reals] && Element[a, Reals];
In[213]:= Conjugate[x+I*y] //FullSimplify
Out[213]= x - I*y
In[214]:= Im[a] //FullSimplify
Out[214]= 0
```

1.12 units

https://reference.wolfram.com/language/tutorial/UnitsOverview.html

The Wolfram language is capable of dealing with units of measure, as required for physical calculations. For example, we can make the assignment

```
In[215]:= s = Quantity[3, "m"];
```

[8] See https://en.wikipedia.org/wiki/Analytic_continuation.

to specify that `s` should be three meters. A large number of units can be used, as well as physical constants:

```
In[216]:= kB = Quantity["BoltzmannConstant"];
```

will define the variable `kB` to be Boltzmann's constant. Take note that complicated or slightly unusual quantities are evaluated through the online service *Wolfram Alpha*®, which means that you need an internet connection in order to evaluate them. For this and other reasons, unit calculations are very slow and to be avoided whenever possible.

If you are unsure whether your expression has been interpreted correctly, the full internal form

```
In[217]:= FullForm[kB]
Out[217]= Quantity[1, "BoltzmannConstant"]
```

usually helps. Alternatively, converting to SI units can often clarify a definition:

```
In[218]:= UnitConvert[kB]
Out[218]= Quantity[1.38065*10^-23, "kg m^2/(s^2 K)"]
```

In principle, we can use this mechanism to do all the calculations in this lecture with units; however, for the sake of generality (as many other computer programs cannot deal with units) when we do numerical calculations, we will convert every quantity into dimensionless form in what follows.

In order to eliminate units from a calculation, we must determine a set of units in which to express the relevant quantities. This means that every physical quantity x is expressed as the product of a *unit* and a *dimensionless multiplier*. The actual calculations are performed only with the dimensionless multipliers. A smart choice of units can help in implementing a problem.

As an example we calculate the acceleration of an A380 airplane ($m = 560\,\mathrm{t}$) due to its jet engines ($F = 4 \times 311\,\mathrm{kN}$). The easiest way is to use the Wolfram language's built-in unit processing:

```
In[219]:= F = Quantity[4*311, "kN"];
In[220]:= m = Quantity[560, "t"];
In[221]:= a = UnitConvert[F/m, "m/s^2"] //N
Out[221]= 2.22143 m/s^2
```

This method is, however, much slower than using purely numerical calculations, and furthermore cannot be generalized to matrix and vector algebra.

Now we do the same calculation with dimensionless multipliers only. For this, we first set up a consistent set of units, for example the SI units:

```
In[222]:= ForceUnit = Quantity["Newtons"];
In[223]:= MassUnit = Quantity["Kilograms"];
In[224]:= AccelerationUnit = UnitConvert[ForceUnit/MassUnit]
Out[224]= 1 m/s^2
```

It is important that these units are consistent with each other, *i.e.*, that the product of the mass and acceleration units gives the force unit. The calculation is now effected with a simple numerical division `a=F/m`:

```
In[225]:= F = Quantity[4*311, "kN"] / ForceUnit
Out[225]= 1244000
In[226]:= m = Quantity[560, "t"] / MassUnit
Out[226]= 560000
In[227]:= a = F/m //N
Out[227]= 2.22143
```

This result of 2.221 43 acceleration units, meaning $2.221\,43\,\mathrm{m/s^2}$, is the same as `Out[221]`.

We can do this type of calculation in any consistent unit system: as a second example, we use the unit definitions

```
In[228]:= ForceUnit = Quantity["KiloNewtons"];
In[229]:= MassUnit = Quantity["AtomicMassUnit"];
In[230]:= AccelerationUnit = UnitConvert[ForceUnit/MassUnit]
Out[230]= 6.022141*10^29 m/s^2
```

and calculate

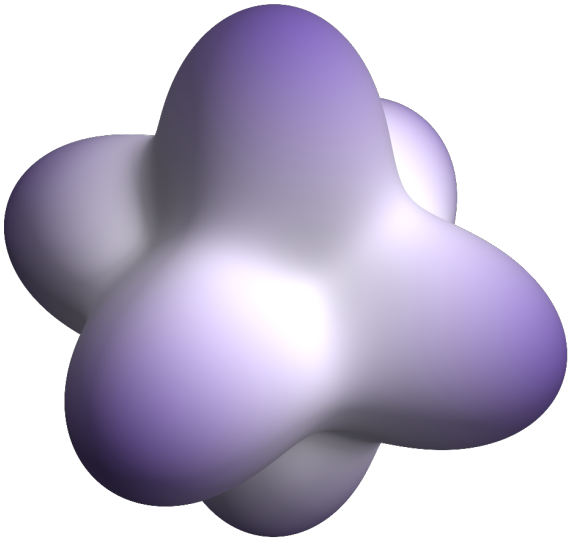
```
In[231]:= F = Quantity[4*311, "kN"] / ForceUnit
Out[231]= 1244
In[232]:= m = Quantity[560, "t"] / MassUnit
Out[232]= 3.3723989*10^32
In[233]:= a = F/m //N
Out[233]= 3.68877*10^-30
```

This result is again the same as `Out[221]`, because $3.688\,77 \times 10^{-30}$ acceleration units are $3.688\,77 \times 10^{-30} \times 6.022\,141 \times 10^{29}\,\mathrm{m/s^2}$.

It is not important which unit system we use. In practice, it is often convenient to use a system of units that yields dimensionless multipliers that are on the order of unity; but this is not a strict requirement.

2

quantum mechanics: states and operators



If you are like most students of quantum mechanics, then you have begun your quantum studies by hearing stories about experiments such as Young's double slit,[1] the Stern–Gerlach spin quantization,[2] and Heisenberg's uncertainty principle.[3] Many concepts and analogies are introduced to get an idea of what quantum mechanics is about and to begin to develop an intuition for it. Yet there is a large gap between this kind of qualitative understanding and being able to solve even the simplest quantum-mechanical problems on a computer, essentially because a computer only works with numbers, not with stories, analogies, or visualizations.

The goal of this chapter is to connect the fundamental quantum-mechanical concepts to representations that a computer can understand. We develop the tools that will be used in the remaining chapters to express and solve interesting quantum-mechanical problems.

[1] See https://en.wikipedia.org/wiki/Double-slit_experiment.
[2] See https://en.wikipedia.org/wiki/Stern-Gerlach_experiment.
[3] See https://en.wikipedia.org/wiki/Uncertainty_principle.

2.1 basis sets and representations

Quantum-mechanical problems are usually specified in terms of operators and quantum states. The quantum states are elements of a Hilbert space; the operators act on such vectors. How can these objects be represented on a computer, which only understands numbers but not Hilbert spaces?

In order to find a computer-representable form of these abstract objects, we assume that we know an ortho-normal[4] basis $\{|i\rangle\}_i$ of this Hilbert space, with scalar product $\langle i|j\rangle = \delta_{ij}$. In section 2.4 we will talk about how to construct such bases. For now we make the assumption that this basis is complete, such that $\sum_i |i\rangle\langle i| = \mathbb{1}$. We will see in section 2.1.1 how to deal with incomplete basis sets.

Given any operator $\hat{\mathcal{A}}$ acting on this Hilbert space, we use the completeness relation twice to find

$$\hat{\mathcal{A}} = \mathbb{1}\cdot\hat{\mathcal{A}}\cdot\mathbb{1} = \left[\sum_i |i\rangle\langle i|\right]\cdot\hat{\mathcal{A}}\cdot\left[\sum_j |j\rangle\langle j|\right] = \sum_{ij}\underbrace{\langle i|\hat{\mathcal{A}}|j\rangle}_{A_{ij}}\,|i\rangle\langle j|. \tag{2.1}$$

We define a numerical matrix $\boldsymbol{A}$ with elements $A_{ij} = \langle i|\hat{\mathcal{A}}|j\rangle \in \mathbb{C}$ to rewrite this as

$$\hat{\mathcal{A}} = \sum_{ij} A_{ij}\,|i\rangle\langle j|. \tag{2.2}$$

The same can be done with a state vector $|\psi\rangle$: using the completeness relation,

$$|\psi\rangle = \mathbb{1}\cdot|\psi\rangle = \left[\sum_i |i\rangle\langle i|\right]\cdot|\psi\rangle = \sum_i \underbrace{\langle i|\psi\rangle}_{\psi_i}\,|i\rangle, \tag{2.3}$$

and by defining a numerical vector $\vec{\boldsymbol{\psi}}$ with elements $\psi_i = \langle i|\psi\rangle \in \mathbb{C}$ the state vector is

$$|\psi\rangle = \sum_i \psi_i |i\rangle. \tag{2.4}$$

Both the matrix $\boldsymbol{A}$ and the vector $\vec{\boldsymbol{\psi}}$ are complex-valued objects which can be represented in any computer system. Equation (2.2) and Equation (2.4) serve to convert between Hilbert-space representations and number-based (matrix/vector-based) representations. These equations are at the center of what it means to find a computer representation of a quantum-mechanical problem.

2.1.1 incomplete basis sets

For infinite-dimensional Hilbert spaces we must usually content ourselves with finite basis sets that approximate the low-energy physics (or, more generally, the physically relevant dynamics) of the problem. In practice this means that an orthonormal basis set may not be complete,

$$\sum_i |i\rangle\langle i| = \hat{P}, \tag{2.5}$$

which is the projector onto that subspace of the full Hilbert space which the basis is capable of describing. We denote $\hat{Q} = \mathbb{1} - \hat{P}$ as the complement of this projector: $\hat{Q}$ is the projector onto the remainder of the Hilbert space that is left out of this truncated description. The equivalent of Equation (2.1) is then

$$\begin{aligned}\hat{\mathcal{A}} = \mathbb{1}\cdot\hat{\mathcal{A}}\cdot\mathbb{1} &= (\hat{P}+\hat{Q})\cdot\hat{\mathcal{A}}\cdot(\hat{P}+\hat{Q}) = \hat{P}\cdot\hat{\mathcal{A}}\cdot\hat{P} + \hat{P}\cdot\hat{\mathcal{A}}\cdot\hat{Q} + \hat{Q}\cdot\hat{\mathcal{A}}\cdot\hat{P} + \hat{Q}\cdot\hat{\mathcal{A}}\cdot\hat{Q} \\ &= \underbrace{\sum_{ij} A_{ij}\,|i\rangle\langle j|}_{\text{within described subspace}} + \underbrace{\hat{P}\cdot\hat{\mathcal{A}}\cdot\hat{Q} + \hat{Q}\cdot\hat{\mathcal{A}}\cdot\hat{P}}_{\text{neglected coupling to (high-energy) part}} + \underbrace{\hat{Q}\cdot\hat{\mathcal{A}}\cdot\hat{Q}}_{\text{neglected (high-energy) part}}\end{aligned} \tag{2.6}$$

In the same way, the equivalent of Equation (2.3) is

$$|\psi\rangle = \mathbb{1}\cdot|\psi\rangle = (\hat{P}+\hat{Q})\cdot|\psi\rangle = \underbrace{\sum_i \psi_i\,|i\rangle}_{\text{within described subspace}} + \underbrace{\hat{Q}|\psi\rangle}_{\text{neglected (high-energy) part}} \tag{2.7}$$

[4] The following calculations can be extended to situations where the basis is not ortho-normal. For the scope of this lecture we are however not interested in this complication.

Since $\hat{Q}$ is the projector onto the neglected subspace, the component $\hat{Q}|\psi\rangle$ of Equation (2.7) is the part of the quantum state $|\psi\rangle$ that is left out of the description in the truncated basis. In specific situations we will need to make sure that all terms involving $\hat{Q}$ in Equation (2.6) and Equation (2.7) can be safely neglected. See Equation (4.28) for a problematic example of an operator expressed in a truncated basis.

variational ground-state calculations

Calculating the ground state of a Hamiltonian in an incomplete basis set is a special case of the variational method.[5] As we will see for example in section 4.1.7, the variational ground-state energy is always larger than the true ground-state energy. When we add more basis functions, the numerically calculated ground-state energy decreases monotonically. At the same time, the overlap (scalar product) of the numerically calculated ground state with the true ground state monotonically increases to unity. These convergence properties often allow us to judge whether or not a chosen computational basis set is sufficiently complete.

2.1.2 exercises

Q2.1 We describe a spin-1/2 system in the basis $\mathcal{B}$ containing the two states

$$|\Uparrow_{\vartheta,\varphi}\rangle = \cos\left(\frac{\vartheta}{2}\right)|\uparrow\rangle + e^{i\varphi}\sin\left(\frac{\vartheta}{2}\right)|\downarrow\rangle$$
$$|\Downarrow_{\vartheta,\varphi}\rangle = -e^{-i\varphi}\sin\left(\frac{\vartheta}{2}\right)|\uparrow\rangle + \cos\left(\frac{\vartheta}{2}\right)|\downarrow\rangle \tag{2.8}$$

1. Show that the basis $\mathcal{B} = \{|\Uparrow_{\vartheta,\varphi}\rangle, |\Downarrow_{\vartheta,\varphi}\rangle\}$ is orthonormal.
2. Show that the basis $\mathcal{B}$ is complete: $|\Uparrow_{\vartheta,\varphi}\rangle\langle\Uparrow_{\vartheta,\varphi}| + |\Downarrow_{\vartheta,\varphi}\rangle\langle\Downarrow_{\vartheta,\varphi}| = \mathbb{1}$.
3. Express the states $|\uparrow\rangle$ and $|\downarrow\rangle$ as vectors in the basis $\mathcal{B}$.
4. Express the Pauli operators $\hat{\sigma}_x$, $\hat{\sigma}_y$, $\hat{\sigma}_z$ as matrices in the basis $\mathcal{B}$.
5. Show that $|\Uparrow_{\vartheta,\varphi}\rangle$ and $|\Downarrow_{\vartheta,\varphi}\rangle$ are eigenvectors of $\hat{\sigma}(\vartheta,\varphi) = \hat{\sigma}_x\sin(\vartheta)\cos(\varphi) + \hat{\sigma}_y\sin(\vartheta)\sin(\varphi) + \hat{\sigma}_z\cos(\vartheta)$. What are the eigenvalues?

Q2.2 The eigenstate basis for the description of the infinite square well of unit width is made up of the ortho-normalized functions

$$\langle x|n\rangle = \phi_n(x) = \sqrt{2}\sin(n\pi x) \tag{2.9}$$

defined on the interval $[0,1]$, with $n \in \{1,2,3,\dots\}$.

1. Calculate the function $P_\infty(x,y) = \langle x|\left[\sum_{n=1}^{\infty}|n\rangle\langle n|\right]|y\rangle$.
2. In computer-based calculations we limit the basis set to $n \in \{1,2,3,\dots,n_{\max}\}$ for some large value of $n_{\max}$. Using Mathematica, calculate the function $P_{n_{\max}}(x,y) = \langle x|\left[\sum_{n=1}^{n_{\max}}|n\rangle\langle n|\right]|y\rangle$ (use the `Sum` function). Make a plot for $n_{\max} = 10$ (use the `DensityPlot` function).
3. What does the function P represent?

2.2 time-independent Schrödinger equation

The time-independent Schrödinger equation is

$$\hat{\mathcal{H}}|\psi\rangle = E|\psi\rangle. \tag{2.10}$$

As in section 2.1 we use a computational basis to express the Hamiltonian operator $\hat{\mathcal{H}}$ and the quantum state ψ as

$$\hat{\mathcal{H}} = \sum_{ij} H_{ij}\,|i\rangle\langle j|, \qquad |\psi\rangle = \sum_i \psi_i\,|i\rangle. \tag{2.11}$$

[5] See https://en.wikipedia.org/wiki/Variational_method_(quantum_mechanics).

With these substitutions the Schrödinger equation becomes

$$\begin{aligned}\left[\sum_{ij} H_{ij}\ |i\rangle\langle j|\right]\left[\sum_{k}\psi_k\ |k\rangle\right] &= E\left[\sum_{\ell}\psi_\ell\ |\ell\rangle\right]\\ \sum_{ijk} H_{ij}\psi_k\ \underbrace{\langle j|k\rangle}_{=\delta_{jk}}\ |i\rangle &= \sum_{\ell} E\psi_\ell\ |\ell\rangle\\ \sum_{ij} H_{ij}\psi_j\ |i\rangle &= \sum_{\ell} E\psi_\ell\ |\ell\rangle \end{aligned} \tag{2.12}$$

Multiplying this equation by $\langle m|$ from the left, and using the orthonormality of the basis set, gives

$$\begin{aligned}\langle m|\sum_{ij} H_{ij}\psi_j\ |i\rangle &= \langle m|\sum_{\ell} E\psi_\ell\ |\ell\rangle\\ \sum_{ij} H_{ij}\psi_j\underbrace{\langle m|i\rangle}_{=\delta_{mi}} &= \sum_{\ell} E\psi_\ell\underbrace{\langle m|\ell\rangle}_{=\delta_{m\ell}}\\ \sum_{j} H_{mj}\psi_j &= E\psi_m \end{aligned} \tag{2.13}$$

In matrix notation this can be written as

$$\boxed{\boldsymbol{H}\cdot\vec{\boldsymbol{\psi}} = E\vec{\boldsymbol{\psi}}.} \tag{2.14}$$

This is the central equation of this lecture. It is the time-independent Schrödinger equation in a form that computers can understand, namely an eigenvalue equation in terms of numerical (complex) matrices and vectors.

If you think that there is no difference between Equation (2.10) and Equation (2.14), then I invite you to re-read this section as I consider it extremely important for what follows in this course. You can think of Equation (2.10) as an abstract relationship between operators and vectors in Hilbert space, while Equation (2.14) is a *numerical representation* of this relationship in a concrete basis set $\{|i\rangle\}_i$. They both contain the exact same information (since we converted one to the other in a few lines of mathematics) but they are conceptually very different, as one is understandable by a computer and the other is not.

2.2.1 diagonalization

The matrix form of Equation (2.14) of the Schrödinger equation is an eigenvalue equation as you know from linear algebra. Given a matrix of complex numbers $\boldsymbol{H}$ we can find the eigenvalues E_i and eigenvectors $\vec{\boldsymbol{\psi}}_i$ using Mathematica's built-in procedures, as described in section 1.10.4.

2.2.2 exercises

Q2.3 Express the spin-1/2 Hamiltonian

$$\hat{\mathcal{H}} = \sin(\vartheta)\cos(\varphi)\hat{\sigma}_x + \sin(\vartheta)\sin(\varphi)\hat{\sigma}_y + \cos(\vartheta)\hat{\sigma}_z \tag{2.15}$$

in the basis $\{|\uparrow\rangle, |\downarrow\rangle\}$, and calculate its eigenvalues and eigenvectors. NB: $\hat{\sigma}_{x,y,z}$ are the Pauli operators.

2.3 time-dependent Schrödinger equation

The time-dependent Schrödinger equation is

$$i\hbar\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = \hat{\mathcal{H}}(t)|\psi(t)\rangle, \tag{2.16}$$

where the Hamiltonian $\hat{\mathcal{H}}$ can have an explicit time dependence. This differential equation has the formal solution

$$|\psi(t)\rangle = \hat{\mathcal{U}}(t_0;t)|\psi(t_0)\rangle \tag{2.17}$$

in terms of the *propagator*

$$\hat{\mathcal{U}}(t_0;t) = \mathbb{1} - \frac{\mathrm{i}}{\hbar}\int_{t_0}^{t} \mathrm{d}t_1 \hat{\mathcal{H}}(t_1) - \frac{1}{\hbar^2}\int_{t_0}^{t}\mathrm{d}t_1\int_{t_0}^{t_1}\mathrm{d}t_2 \hat{\mathcal{H}}(t_1)\hat{\mathcal{H}}(t_2) + \frac{\mathrm{i}}{\hbar^3}\int_{t_0}^{t}\mathrm{d}t_1\int_{t_0}^{t_1}\mathrm{d}t_2\int_{t_0}^{t_2}\mathrm{d}t_3 \hat{\mathcal{H}}(t_1)\hat{\mathcal{H}}(t_2)\hat{\mathcal{H}}(t_3) + \frac{1}{\hbar^4}\int_{t_0}^{t}\mathrm{d}t_1\int_{t_0}^{t_1}\mathrm{d}t_2\int_{t_0}^{t_2}\mathrm{d}t_3\int_{t_0}^{t_3}\mathrm{d}t_4 \hat{\mathcal{H}}(t_1)\hat{\mathcal{H}}(t_2)\hat{\mathcal{H}}(t_3)\hat{\mathcal{H}}(t_4) + \cdots \quad (2.18)$$

that propagates any state from time t_0 to time t. An alternative form is given by the *Magnus expansion*[6]

$$\hat{\mathcal{U}}(t_0;t) = \exp\left[\sum_{k=1}^{\infty}\hat{\Omega}_k(t_0;t)\right] \quad (2.19)$$

with the contributions

$$\begin{aligned}
\hat{\Omega}_1(t_0;t) &= -\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t}\mathrm{d}t_1\hat{\mathcal{H}}(t_1)\\
\hat{\Omega}_2(t_0;t) &= -\frac{1}{2\hbar^2}\int_{t_0}^{t}\mathrm{d}t_1\int_{t_0}^{t_1}\mathrm{d}t_2[\hat{\mathcal{H}}(t_1),\hat{\mathcal{H}}(t_2)]\\
\hat{\Omega}_3(t_0;t) &= \frac{\mathrm{i}}{6\hbar^3}\int_{t_0}^{t}\mathrm{d}t_1\int_{t_0}^{t_1}\mathrm{d}t_2\int_{t_0}^{t_2}\mathrm{d}t_3\left([\hat{\mathcal{H}}(t_1),[\hat{\mathcal{H}}(t_2),\hat{\mathcal{H}}(t_3)]]+[\hat{\mathcal{H}}(t_3),[\hat{\mathcal{H}}(t_2),\hat{\mathcal{H}}(t_1)]]\right)\\
&\ldots
\end{aligned} \quad (2.20)$$

This expansion in terms of different-time commutators is often easier to evaluate than Equation (2.18), especially when the contributions vanish for $k > k_{\max}$ (see section 2.3.3 for the case $k_{\max} = 1$). Even if higher-order contributions do not vanish entirely, they (usually) decrease in importance much more rapidly with increasing k than those of Equation (2.18). Also, even if the Magnus expansion is artificially truncated (neglecting higher-order terms), the quantum-mechanical evolution is still unitary; this is not the case for Equation (2.18).

Notice that the exponential in Equation (2.19) haa an operator or a matrix as their argument: in Mathematica this matrix exponentiation is done with the `MatrixExp` function. It does not calculate the exponential element-by-element, but instead calculates

$$e^{\hat{A}} = \sum_{n=0}^{\infty}\frac{\hat{A}^n}{n!}, \qquad e^{\boldsymbol{A}} = \sum_{n=0}^{\infty}\frac{\boldsymbol{A}^n}{n!}. \quad (2.21)$$

2.3.1 time-independent basis

We express the quantum state again in terms of the chosen basis, which is assumed to be time-independent. This leaves the time-dependence in the expansion coefficients,

$$\hat{\mathcal{H}}(t) = \sum_{ij} H_{ij}(t)\ |i\rangle\langle j|, \qquad |\psi(t)\rangle = \sum_i \psi_i(t)\ |i\rangle. \quad (2.22)$$

Inserting these expressions into the time-dependent Schrödinger equation (2.16) gives

$$\mathrm{i}\hbar\sum_i \dot{\psi}_i(t)\ |i\rangle = \left[\sum_{jk} H_{jk}(t)\ |j\rangle\langle k|\right]\sum_\ell \psi_\ell(t)\ |\ell\rangle = \sum_{jk} H_{jk}(t)\psi_k(t)\ |j\rangle. \quad (2.23)$$

Multiplying with $\langle m|$ from the left:

$$\mathrm{i}\hbar\dot{\psi}_m(t) = \sum_k H_{mk}(t)\psi_k(t) \quad (2.24)$$

or, in matrix notation,

$$\boxed{\mathrm{i}\hbar\dot{\vec{\boldsymbol{\psi}}}(t) = \boldsymbol{H}(t)\cdot\vec{\boldsymbol{\psi}}(t).} \quad (2.25)$$

Since the matrix $\boldsymbol{H}(t)$ is supposedly known, this equation represents a system of coupled complex differential equations for the vector $\vec{\boldsymbol{\psi}}(t)$, which can be solved on a computer.

[6] See https://en.wikipedia.org/wiki/Magnus_expansion.

2.3.2 time-dependent basis: interaction picture

It can be advantageous to use a time-dependent basis. The most frequently used such basis is given by the interaction picture of quantum mechanics, where the Hamiltonian can be split into a time-independent principal part $\hat{\mathcal{H}}_0$ and a small time-dependent part $\hat{\mathcal{H}}_1$:

$$\hat{\mathcal{H}}(t) = \hat{\mathcal{H}}_0 + \hat{\mathcal{H}}_1(t). \tag{2.26}$$

Assuming that we can diagonalize $\hat{\mathcal{H}}_0$, possibly numerically, such that the eigenfunctions satisfy $\hat{\mathcal{H}}_0|i\rangle = E_i|i\rangle$, we propose the time-dependent basis

$$|i(t)\rangle = e^{-\mathrm{i}E_i t/\hbar}|i\rangle. \tag{2.27}$$

If we express any quantum state in this basis as

$$|\psi(t)\rangle = \sum_i \psi_i(t)\ |i(t)\rangle = \sum_i \psi_i(t) e^{-\mathrm{i}E_i t/\hbar}|i\rangle, \tag{2.28}$$

the time-dependent Schrödinger equation becomes

$$\begin{aligned}
\sum_i \left[\mathrm{i}\hbar\dot{\psi}_i(t) + E_i\psi_i(t)\right] e^{-\mathrm{i}E_i t/\hbar}|i\rangle &= \sum_j \psi_j(t) e^{-\mathrm{i}E_j t/\hbar} E_j\ |j\rangle + \sum_j \psi_j(t) e^{-\mathrm{i}E_j t/\hbar}\hat{\mathcal{H}}_1(t)\ |j\rangle \\
\sum_i \mathrm{i}\hbar\dot{\psi}_i(t) e^{-\mathrm{i}E_i t/\hbar}|i\rangle &= \sum_j \psi_j(t) e^{-\mathrm{i}E_j t/\hbar}\hat{\mathcal{H}}_1(t)\ |j\rangle
\end{aligned} \tag{2.29}$$

Multiply by $\langle k|$ from the left:

$$\begin{aligned}
\langle k|\sum_i \mathrm{i}\hbar\dot{\psi}_i(t) e^{-\mathrm{i}E_i t/\hbar}|i\rangle &= \langle k|\sum_j \psi_j(t) e^{-\mathrm{i}E_j t/\hbar}\hat{\mathcal{H}}_1(t)\ |j\rangle \\
\sum_i \mathrm{i}\hbar\dot{\psi}_i(t) e^{-\mathrm{i}E_i t/\hbar}\underbrace{\langle k|i\rangle}_{=\delta_{ki}} &= \sum_j \psi_j(t) e^{-\mathrm{i}E_j t/\hbar}\langle k|\hat{\mathcal{H}}_1(t)|j\rangle \\
\mathrm{i}\hbar\dot{\psi}_k(t) &= \sum_j \psi_j(t) e^{-\mathrm{i}(E_j - E_k)t/\hbar}\langle k|\hat{\mathcal{H}}_1(t)|j\rangle.
\end{aligned} \tag{2.30}$$

This is the same matrix/vector evolution expression as Equation (2.25), except that here the Hamiltonian matrix elements must be defined as

$$H_{ij}(t) = \langle i|\hat{\mathcal{H}}_1(t)|j\rangle e^{-\mathrm{i}(E_j - E_i)t/\hbar}. \tag{2.31}$$

We see immediately that if the interaction Hamiltonian vanishes [$\hat{\mathcal{H}}_1(t) = 0$], then the expansion coefficients $\psi_i(t)$ become time-independent, as expected since they are the coefficients of the eigenfunctions of the time-independent Schrödinger equation.

When a quantum-mechanical system is composed of different parts that have vastly different energy scales of their internal evolution $\hat{\mathcal{H}}_0$, then the use of Equation (2.31) can have great numerical advantages. It turns out that the relevant interaction terms $H_{ij}(t)$ in the interaction picture will have relatively slowly evolving phases $\exp[-\mathrm{i}(E_j - E_i)t/\hbar]$, on a time scale given by relative energy *differences* and not by *absolute* energies; this makes it possible to solve the coupled differential equations of Equation (2.25) numerically without using an absurdly small time step.

2.3.3 special case: $\left[\hat{\mathcal{H}}(t), \hat{\mathcal{H}}(t')\right] = 0\ \forall(t, t')$

If the Hamiltonian commutes with itself at different times, $\left[\hat{\mathcal{H}}(t), \hat{\mathcal{H}}(t')\right] = 0\ \forall(t, t')$, the propagator (2.19) of Equation (2.16) can be simplified to

$$\hat{\mathcal{U}}(t_0; t) = \exp\left[-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t} \hat{\mathcal{H}}(s)\mathrm{d}s\right], \tag{2.32}$$

and the corresponding solution of Equation (2.25) is

$$\vec{\boldsymbol{\psi}}(t) = \exp\left[-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t} \boldsymbol{H}(s)\mathrm{d}s\right] \cdot \vec{\boldsymbol{\psi}}(t_0). \tag{2.33}$$

Again, these matrix exponentials are calculated with `MatrixExp` in Mathematica.

2.3.4 special case: time-independent Hamiltonian

In the special (but common) case where the Hamiltonian is time-independent, the integral in Equation (2.33) can be evaluated immediately, and the solution is

$$\vec{\boldsymbol{\psi}}(t) = \exp\left[-\frac{\mathrm{i}(t-t_0)}{\hbar}\boldsymbol{H}\right]\cdot\vec{\boldsymbol{\psi}}(t_0). \tag{2.34}$$

If we have a specific Hamiltonian matrix `H` defined, for example the matrix of section 1.10.4, we can calculate the propagator $\boldsymbol{U}(\Delta t) = \exp[-\mathrm{i}\boldsymbol{H}\Delta t/\hbar]$ for $\Delta t = t - t_0$ with

```
In[234]:=U[Δt_] = MatrixExp[-I*H*Δt/ℏ]
```

The resulting expression for `U[Δt]` will in general be very long, and slow to compute. A more efficient definition is to matrix-exponentiate a numerical matrix for specific values of the propagation interval `Δt`, using a delayed assignment:

```
In[235]:=U[Δt_?NumericQ] := MatrixExp[-I*H*N[Δt]/ℏ]
```

2.3.5 exercises

Q2.4 Demonstrate that the propagator (2.32) gives a quantum state (2.17) that satisfies Equation (2.16).

Q2.5 Calculate the propagator of the Hamiltonian of **Q2.3**.

Q2.6 After `In[234]` and `In[235]`, check `?U`. Which definition of `U` comes first? Why?

2.4 basis construction

In principle, the choice of basis set $\{|i\rangle\}_i$ does not influence the way a computer program like Mathematica solves a quantum-mechanical problem. In practice, however, we always need a *constructive* way to find some basis for a given quantum-mechanical problem. A basis that takes the system's Hamiltonian into account may give a computationally simpler description; but in complicated systems it is often more important to find *any* way of constructing a usable basis set than finding the perfect one.

2.4.1 description of a single degree of freedom

When we describe a single quantum-mechanical degree of freedom, it is often possible to deduce a useful basis set from knowledge of the Hilbert space itself. This is what we will be doing in chapter 3 for spin systems, where the well-known Dicke basis $\{|S, M_S\rangle\}_{M_S=-S}^{S}$ turns out to be very useful.

For more complicated degrees of freedom, we can find inspiration for a basis choice from an associated Hamiltonian. Such Hamiltonians describing a single degree of freedom are often so simple that they can be diagonalized by hand. If this is not the case, real-world Hamiltonians $\hat{\mathcal{H}}$ can often be decomposed like Equation (2.26) into a "simple" part $\hat{\mathcal{H}}_0$ that is time-independent and can be diagonalized easily, and a "difficult" part $\hat{\mathcal{H}}_1$ that usually contains complicated interactions and/or time-dependent terms but is of smaller magnitude. A natural choice of basis set is the set of eigenstates of $\hat{\mathcal{H}}_0$, or at least those eigenstates below a certain cutoff energy since they will be optimally suited to describe the complete low-energy behavior of the degree of freedom in question. This latter point is especially important for infinite-dimensional systems (chapter 4), where any computer representation will necessarily truncate the dimensionality, as discussed in section 2.1.1.

examples of basis sets for single degrees of freedom:

- *spin degree of freedom:* Dicke states $|S, M_S\rangle$ (see chapter 3)
- *translational degree of freedom:* square-well eigenstates, harmonic oscillator eigenstates (see chapter 4)

- *rotational degree of freedom:* spherical harmonics
- *atomic system:* hydrogen-like orbitals
- *translation-invariant system:* periodic plane waves
- *periodic system (crystal):* periodic plane waves on the reciprocal lattice

2.4.2 description of coupled degrees of freedom

A broad range of quantum-mechanical systems of interest are governed by Hamiltonians of the form

$$\hat{\mathcal{H}}(t) = \left(\sum_{k=1}^{N} \hat{\mathcal{H}}^{(k)}(t)\right) + \hat{\mathcal{H}}_{\text{int}}(t), \tag{2.35}$$

where N individual degrees of freedom are governed by their individual Hamiltonians $\hat{\mathcal{H}}^{(k)}(t)$, while their interactions are described by $\hat{\mathcal{H}}_{\text{int}}(t)$. This is a situation we will encounter repeatedly as we construct more complicated quantum-mechanical problems from simpler parts. A few simple examples are:

- A set of N interacting particles: the Hamiltonians $\hat{\mathcal{H}}^{(k)}$ describe the individual particles, while $\hat{\mathcal{H}}_{\text{int}}$ describes their interactions (see section 3.4).
- A single particle moving in three spatial degrees of freedom: the three Hamiltonians $\hat{\mathcal{H}}^{(x)} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}$, $\hat{\mathcal{H}}^{(y)} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial y^2}$, $\hat{\mathcal{H}}^{(z)} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial z^2}$ describe the kinetic energy in the three directions, while $\hat{\mathcal{H}}_{\text{int}}$ contains the potential energy, which usually couples these three degrees of freedom (see section 4.4).
- A single particle with internal (spin) and external (motional) degrees of freedom, which are coupled through a state-dependent potential in $\hat{\mathcal{H}}_{\text{int}}$ (see chapter 5).

The existence of individual Hamiltonians $\hat{\mathcal{H}}^{(k)}$ assumes that the Hilbert space of the complete system has a tensor-product structure

$$V = V^{(1)} \otimes V^{(2)} \otimes \cdots \otimes V^{(N)}, \tag{2.36}$$

where each Hamiltonian $\hat{\mathcal{H}}^{(k)}$ acts only in a single component space,

$$\hat{\mathcal{H}}^{(k)} = \mathbb{1}^{(1)} \otimes \mathbb{1}^{(2)} \otimes \cdots \otimes \mathbb{1}^{(k-1)} \otimes \hat{h}^{(k)} \otimes \mathbb{1}^{(k+1)} \otimes \cdots \otimes \mathbb{1}^{(N)}. \tag{2.37}$$

Further, if we are able to construct bases $\{|i_k\rangle^{(k)}\}_{i_k=1}^{n_k}$ for all of the component Hilbert spaces $V^{(k)}$, as in section 2.4.1, then we can construct a basis for the full Hilbert space V by taking all possible tensor products of basis functions:

$$|i_1, i_2, \ldots, i_N\rangle = |i_1\rangle^{(1)} \otimes |i_2\rangle^{(2)} \otimes \cdots \otimes |i_N\rangle^{(N)}. \tag{2.38}$$

This basis will have $\prod_{k=1}^{N} n_k$ elements, which can easily become a very large number for composite systems.

quantum states

A product state of the complete system

$$|\psi\rangle = |\psi_1\rangle^{(1)} \otimes |\psi_2\rangle^{(2)} \otimes \cdots \otimes |\psi_N\rangle^{(N)} \tag{2.39}$$

can be described in the following way. First, each single-particle state is decomposed in its own basis as in Equation (2.4),

$$|\psi_k\rangle^{(k)} = \sum_{i_k=1}^{n_k} \psi_{i_k}^{(k)} |i_k\rangle^{(k)}. \tag{2.40}$$

Inserting these expansions into Equation (2.39) gives the expansion into the basis functions (2.38) of the full system,

$$\begin{aligned} |\psi\rangle = \left[\sum_{i_1=1}^{n_1} \psi_{i_1}^{(1)} |i_1\rangle^{(1)}\right] \otimes \left[\sum_{i_2=1}^{n_2} \psi_{i_2}^{(2)} |i_2\rangle^{(2)}\right] \otimes \cdots \otimes \left[\sum_{i_N=1}^{n_N} \psi_{i_N}^{(N)} |i_N\rangle^{(N)}\right] \\ = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_N=1}^{n_N} \left[\psi_{i_1}^{(1)} \psi_{i_2}^{(2)} \ldots \psi_{i_N}^{(N)}\right] |i_1, i_2, \ldots, i_N\rangle \end{aligned} \tag{2.41}$$

In Mathematica, such a state tensor product can be calculated as follows. For example, assume that ψ1 is a vector containing the expansion of $|\psi_1\rangle^{(1)}$ in its basis, and similarly for ψ2 and ψ3. The vector ψ of expansion coefficients of the full state $|\psi\rangle = |\psi_1\rangle^{(1)} \otimes |\psi_2\rangle^{(2)} \otimes |\psi_3\rangle^{(3)}$ is calculated with

```
In[236]:= ψ = Flatten[KroneckerProduct[ψ1, ψ2, ψ3]]
```

See **Q2.10** for a numerical example.

More generally, any state can be written as

$$|\psi\rangle = \sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N} \psi_{i_1,i_2,\ldots,i_N}|i_1, i_2, \ldots, i_N\rangle, \tag{2.42}$$

of which Equation (2.41) is a special case with $\psi_{i_1,i_2,\ldots,i_N} = \psi^{(1)}_{i_1}\psi^{(2)}_{i_2}\cdots\psi^{(N)}_{i_N}$.

operators

If the Hilbert space has the tensor-product structure of Equation (2.36), then the operators acting on this full space are often given as tensor products as well,

$$\hat{A} = \hat{a}^{(1)}_1 \otimes \hat{a}^{(2)}_2 \otimes \cdots \otimes \hat{a}^{(N)}_N, \tag{2.43}$$

or as a sum over such products. If every single-particle operator is decomposed in its own basis as in Equation (2.2),

$$\hat{a}^{(k)}_k = \sum_{i_k=1}^{n_k}\sum_{j_k=1}^{n_k} a^{(k)}_{i_k,j_k}|i_k\rangle^{(k)}\langle j_k|^{(k)}, \tag{2.44}$$

inserting these expressions into Equation (2.43) gives the expansion into the basis functions (2.38) of the full system,

$$\begin{aligned}\hat{A} &= \left[\sum_{i_1=1}^{n_1}\sum_{j_1=1}^{n_1} a^{(1)}_{i_1,j_1}|i_1\rangle^{(1)}\langle j_1|^{(1)}\right] \otimes \left[\sum_{i_2=1}^{n_2}\sum_{j_2=1}^{n_2} a^{(2)}_{i_2,j_2}|i_2\rangle^{(2)}\langle j_2|^{(2)}\right] \otimes \cdots \otimes \left[\sum_{i_N=1}^{n_N}\sum_{j_N=1}^{n_N} a^{(N)}_{i_N,j_N}|i_N\rangle^{(N)}\langle j_N|^{(N)}\right] \\ &= \sum_{i_1=1}^{n_1}\sum_{j_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{j_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\sum_{j_N=1}^{n_N}\left[a^{(1)}_{i_1,j_1}a^{(2)}_{i_2,j_2}\ldots a^{(N)}_{i_N,j_N}\right]|i_1, i_2, \ldots, i_N\rangle\langle j_1, j_2, \ldots, j_N|. \end{aligned} \tag{2.45}$$

In Mathematica, such an operator tensor product can be calculated similarly to In[236] above. For example, assume that a1 is a matrix containing the expansion of $\hat{a}^{(1)}_1$ in its basis, and similarly for a2 and a3. The matrix A of expansion coefficients of the full operator $\hat{A} = \hat{a}^{(1)}_1 \otimes \hat{a}^{(2)}_2 \otimes \hat{a}^{(3)}_3$ is calculated with

```
In[237]:= A = KroneckerProduct[a1, a2, a3]
```

Often we need to construct operators which act only on one of the component spaces, as in Equation (2.37). For example, in a 3-composite system the subsystem Hamiltonians $\hat{h}^{(1)}$, $\hat{h}^{(2)}$, and $\hat{h}^{(3)}$ are first expanded to the full Hilbert space,

```
In[238]:= H1 = KroneckerProduct[h1,
                                IdentityMatrix[Dimensions[h2]],
                                IdentityMatrix[Dimensions[h3]]];
In[239]:= H2 = KroneckerProduct[IdentityMatrix[Dimensions[h1]],
                                h2,
                                IdentityMatrix[Dimensions[h3]]];
In[240]:= H3 = KroneckerProduct[IdentityMatrix[Dimensions[h1]],
                                IdentityMatrix[Dimensions[h2]],
                                h3];
```

where IdentityMatrix[Dimensions[h1]] generates a unit matrix of size equal to that of h1. In this way, the matrices H1, H2, H3 are of equal size and can be added together, even if h1, h2, h3 all have different sizes (expressed in Hilbert spaces of different dimensions):

```
In[241]:= H = H1 + H2 + H3;
```

More generally, any operator can be written as

$$\hat{A} = \sum_{i_1=1}^{n_1}\sum_{j_1=1}^{n_1}\sum_{i_2=1}^{n_2}\sum_{j_2=1}^{n_2}\cdots\sum_{i_N=1}^{n_N}\sum_{j_N=1}^{n_N} a_{i_1,j_1,i_2,j_2,\ldots,i_N,j_N}|i_1, i_2, \ldots, i_N\rangle\langle j_1, j_2, \ldots, j_N|, \tag{2.46}$$

of which Equation (2.45) is a special case with $a_{i_1,j_1,i_2,j_2,\ldots,i_N,j_N} = a^{(1)}_{i_1,j_1}a^{(2)}_{i_2,j_2}\cdots a^{(N)}_{i_N,j_N}$.

2.4.3 reduced density matrices [code]

In this section we calculate reduced density matrices by partial tracing. We start with the most general tripartite case, and then specialize to the more common bipartite case.

Assume that our quantum-mechanical system is composed of three parts A, B, C, and that its Hilbert space is a tensor product of the three associated Hilbert spaces with dimensions d_A, d_B, d_C: $V = V^{(A)} \otimes V^{(B)} \otimes V^{(C)}$. Similar to Equation (2.46), any state of this system can be written as a density matrix

$$\hat{\rho}_\mathrm{ABC} = \sum_{i,i'=1}^{d_\mathrm{A}}\sum_{j,j'=1}^{d_\mathrm{B}}\sum_{k,k'=1}^{d_\mathrm{C}} \rho_{i,j,k,i',j',k'}|i_\mathrm{A}, j_\mathrm{B}, k_\mathrm{C}\rangle\langle i'_\mathrm{A}, j'_\mathrm{B}, k'_\mathrm{C}|, \tag{2.47}$$

where we use the basis states $|i_\mathrm{A}, j_\mathrm{B}, k_\mathrm{C}\rangle = |i\rangle^{(A)} \otimes |j\rangle^{(B)} \otimes |k\rangle^{(C)}$ defined in terms of the three basis sets of the three component Hilbert spaces.

We calculate a reduced density matrix $\hat{\rho}_\mathrm{AC} = \mathrm{Tr}_\mathrm{B}\,\hat{\rho}_\mathrm{ABC}$, which describes what happens to our knowledge of the subsystems A and C when we forget about subsystem B. For example, we could be studying a system of three particles, and take an interest in the state of particles A and C after we have lost particle B. This reduced density matrix is defined as a partial trace,

$$\begin{aligned}
\hat{\rho}_\mathrm{AC} &= \sum_{j''=1}^{d_\mathrm{B}}\langle j''_\mathrm{B}|\hat{\rho}_\mathrm{ABC}|j''_\mathrm{B}\rangle = \sum_{j''=1}^{d_\mathrm{B}}\langle j''_\mathrm{B}|\left[\sum_{i,i'=1}^{d_\mathrm{A}}\sum_{j,j'=1}^{d_\mathrm{B}}\sum_{k,k'=1}^{d_\mathrm{C}} \rho_{i,j,k,i',j',k'}|i_\mathrm{A}, j_\mathrm{B}, k_\mathrm{C}\rangle\langle i'_\mathrm{A}, j'_\mathrm{B}, k'_\mathrm{C}|\right]|j''_\mathrm{B}\rangle \\
&= \sum_{j''=1}^{d_\mathrm{B}}\sum_{i,i'=1}^{d_\mathrm{A}}\sum_{j,j'=1}^{d_\mathrm{B}}\sum_{k,k'=1}^{d_\mathrm{C}} \rho_{i,j,k,i',j',k'}\langle j''_\mathrm{B}|i_\mathrm{A}, j_\mathrm{B}, k_\mathrm{C}\rangle\langle i'_\mathrm{A}, j'_\mathrm{B}, k'_\mathrm{C}|j''_\mathrm{B}\rangle = \sum_{j''=1}^{d_\mathrm{B}}\sum_{i,i'=1}^{d_\mathrm{A}}\sum_{j,j'=1}^{d_\mathrm{B}}\sum_{k,k'=1}^{d_\mathrm{C}} \rho_{i,j,k,i',j',k'}\,[\delta_{j'',j}|i_\mathrm{A}, k_\mathrm{C}\rangle]\,[\delta_{j'',j'}\langle i'_\mathrm{A}, k'_\mathrm{C}|] \\
&= \sum_{i,i'=1}^{d_\mathrm{A}}\sum_{k,k'=1}^{d_\mathrm{C}}\left[\sum_{j=1}^{d_\mathrm{B}}\rho_{i,j,k,i',j,k'}\right]|i_\mathrm{A}, k_\mathrm{C}\rangle\langle i'_\mathrm{A}, k'_\mathrm{C}|,
\end{aligned} \tag{2.48}$$

which makes no reference to subsystem B. It only describes the joint system AC that is left after forgetting about subsystem B.

In Mathematica, we mostly use flattened basis sets, that is, our basis set for the joint Hilbert space of subsystems A, B, C is a flat list of length $d = d_\mathrm{A}d_\mathrm{B}d_\mathrm{C}$:

$$\{|1_\mathrm{A}, 1_\mathrm{B}, 1_\mathrm{C}\rangle, |1_\mathrm{A}, 1_\mathrm{B}, 2_\mathrm{C}\rangle, \ldots, |1_\mathrm{A}, 1_\mathrm{B}, d_\mathrm{C}\rangle, |1_\mathrm{A}, 2_\mathrm{B}, 1_\mathrm{C}\rangle, |1_\mathrm{A}, 2_\mathrm{B}, 2_\mathrm{C}\rangle, \ldots, |1_\mathrm{A}, 2_\mathrm{B}, d_\mathrm{C}\rangle, \ldots, |d_\mathrm{A}, d_\mathrm{B}, d_\mathrm{C}\rangle\}. \tag{2.49}$$

In section 1.10.5 we have seen how lists and tensors can be re-shaped. As we will see below, these tools are used to switch between representations involving indices (i, j, k) (*i.e.*, lists with three indices, rank-three tensors) corresponding to Equation (2.47), and lists involving a single flattened-out index corresponding more to Equation (2.49).

In practical calculations, any density matrix ρABC of the joint system is given as a $d \times d$ matrix whose element (u, v) is the prefactor of the contribution $|u\rangle\langle v|$ with the indices u and v addressing elements in the flat list of Equation (2.49). In order to calculate a reduced density matrix, we first reshape this $d \times d$ density matrix ρABC into a rank-six tensor R with dimensions $d_\mathrm{A} \times d_\mathrm{B} \times d_\mathrm{C} \times d_\mathrm{A} \times d_\mathrm{B} \times d_\mathrm{C}$, and with elements $r_{i,j,k,i',j',k'}$ of Equation (2.47):

```
In[242]:= R = ArrayReshape[ρABC, {dA,dB,dC,dA,dB,dC}]
```

Next, we contract indices 2 and 5 of R in order to do the partial trace over subsystem B, as is done in Equation (2.48) (effectively setting $j = j'$ and summing over j). We find a rank-4 tensor S with dimensions $d_A \times d_C \times d_A \times d_C$:

```
In[243]:= S = TensorContract[R, {2,5}]
```

Finally, we flatten out this tensor again (simultaneously combining indices 1&2 and 3&4) to find the $d_A d_C \times d_A d_C$ reduced density matrix ρAC:

```
In[244]:= ρAC = Flatten[S, {{1,2}, {3,4}}]
```

We assemble all of these steps into a generally usable function:

```
In[245]:= rdm[ρABC_?MatrixQ, {dA_Integer /; dA >= 1,
                              dB_Integer /; dB >= 1,
                              dC_Integer /; dC >= 1}] /;
          Dimensions[ρABC] == {dA*dB*dC, dA*dB*dC} :=
            Flatten[TensorContract[ArrayReshape[ρABC, {dA,dB,dC,dA,dB,dC}], {2,5}],
              {{1,2}, {3,4}}]
```

When our system is in a pure state, $\hat{\rho}_{ABC} = |\psi\rangle\langle\psi|$, this procedure can be simplified greatly. This is particularly important for large system dimensions, where calculating the full density matrix $\hat{\rho}_{ABC}$ may be impossible due to memory constraints. For this, we assume that $|\psi\rangle = \sum_{i=1}^{d_A} \sum_{j=1}^{d_B} \sum_{k=1}^{d_C} \psi_{i,j,k} |i_A, j_B, k_C\rangle$, and therefore $\rho_{i,j,k,i',j',k'} = \psi_{i,j,k} \psi^*_{i',j',k'}$. Again, in Mathematica the coefficients of a state vector ψABC are a flat list referring to the elements of the flat basis of Equation (2.49), and so we start by constructing a rank-3 tensor P with dimensions $d_A \times d_B \times d_C$, whose elements are exactly the $\psi_{i,j,k}$, similar to In[242]:

```
In[246]:= P = ArrayReshape[ψABC, {dA,dB,dC}]
```

We transpose this rank-three tensor into a $d_A \times d_C \times d_B$ tensor P1 and a $d_B \times d_A \times d_C$ tensor P2 by changing the order of the indices:

```
In[247]:= P1 = Transpose[P, {1, 3, 2}]
In[248]:= P2 = Transpose[P, {2, 1, 3}]
```

Now we can contract the index j_B by a dot product, to find a rank-4 tensor Q with dimensions $d_A \times d_C \times d_A \times d_C$:

```
In[249]:= Q = P1 . Conjugate[P2]
```

Finally we flatten Q into the $d_A d_C \times d_A d_C$ reduced density matrix ρAC by combining indices 1&2 and 3&4:

```
In[250]:= ρAC = Flatten[Q, {{1,2}, {3,4}}]
```

We assemble all of these steps into a generally usable function that extends the definition of In[245]:

```
In[251]:= rdm[ψABC_?VectorQ, {dA_Integer /; dA >= 1,
                              dB_Integer /; dB >= 1,
                              dC_Integer /; dC >= 1}] /;
          Length[ψABC] == dA*dB*dC :=
            With[{P = ArrayReshape[ψABC, {dA,dB,dC}]},
              Flatten[Transpose[P, {1,3,2}].ConjugateTranspose[P], {{1,2}, {3,4}}]]
```

Notice that we have merged the transposition of In[248] and the complex-conjugation of In[249] into a single call of the ConjugateTranspose function.

bipartite systems

Consider now the more common case of a bipartite system composed of only two subsystems A and B. We can still use the definitions developed above for tripartite (ABC) structures by introducing a trivial third subsystem with dimension $d_C = 1$. This trivial subsystem will not change anything since it must always be in its one and only possible state. Therefore, given a density matrix `ρAB` of the joint system AB, we calculate the reduced density matrices of subsystems A and B with

```
In[252]:= ρA = rdm[ρAB, {dA,dB,1}];
In[253]:= ρB = rdm[ρAB, {1,dA,dB}];
```

respectively, since it is always the middle subsystem of a given list of three subsystems that is eliminated through partial tracing. In typical Mathematica fashion, we define a `traceout` function that traces out the first d dimensions if `d` > 0 and the last d dimensions if `d` < 0:

```
In[254]:= traceout[ρ_?MatrixQ, d_Integer /; d >= 1] /;
            Length[ρ] == Length[Transpose[ρ]] && Divisible[Length[ρ], d] :=
              rdm[ρ, {1, d, Length[ρ]/d}]
In[255]:= traceout[ρ_?MatrixQ, d_Integer /; d <= -1] /;
            Length[ρ] == Length[Transpose[ρ]] && Divisible[Length[ρ], -d] :=
              rdm[ρ, {Length[ρ]/(-d), -d, 1}]
In[256]:= traceout[ψ_?VectorQ, d_Integer /; d >= 1] /; Divisible[Length[ψ], d] :=
              rdm[ψ, {1, d, Length[ψ]/d}]
In[257]:= traceout[ψ_?VectorQ, d_Integer /; d <= -1] /; Divisible[Length[ψ], -d] :=
              rdm[ψ, {Length[ψ]/(-d), -d, 1}]
```

2.4.4 exercises

Q2.7 Two particles of mass m are moving in a three-dimensional harmonic potential $V(r) = \frac{1}{2}m\omega^2 r^2$ with $r = \sqrt{x^2+y^2+z^2}$, and interacting via s-wave scattering $V_{\text{int}} = g\delta^3(\vec{r}_1 - \vec{r}_2)$.

1. Write down the Hamiltonian of this system.
2. Propose a basis set in which we can describe the quantum mechanics of this system.
3. Calculate the matrix elements of the Hamiltonian in this basis set.

Q2.8 Calculate `ψ` in `In[236]` without using `KroneckerProduct`, but using the `Table` command instead.

Q2.9 Calculate `A` in `In[237]` without using `KroneckerProduct`, but using the `Table` command instead.

Q2.10 Given two spin-1/2 particles in states

$$|\psi\rangle^{(1)} = 0.8|\uparrow\rangle - 0.6|\downarrow\rangle, \qquad |\psi\rangle^{(2)} = 0.6\mathrm{i}|\uparrow\rangle + 0.8|\downarrow\rangle, \tag{2.50}$$

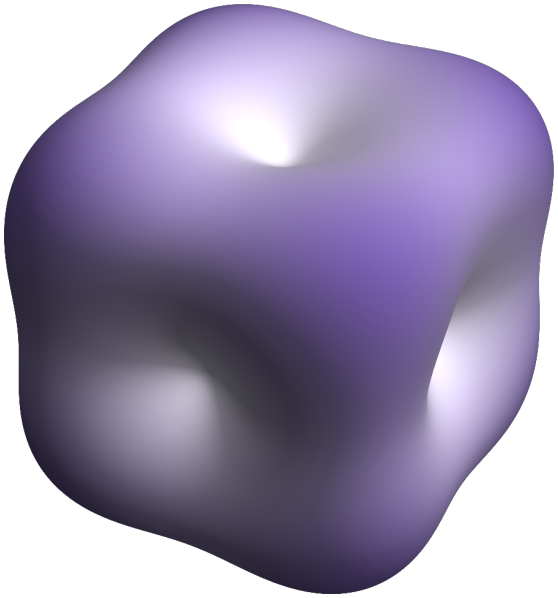
use the `KroneckerProduct` function to calculate the joint state $|\psi\rangle = |\psi\rangle^{(1)} \otimes |\psi\rangle^{(2)}$, and compare the result to a manual calculation. In which order do the coefficients appear in the result of `KroneckerProduct`?

Q2.11 For the state of Equation (2.50), calculate the reduced density matrices $\rho^{(1)}$ and $\rho^{(2)}$ by tracing out the other subsystem. Compare them to the density matrices $|\psi\rangle^{(1)}\langle\psi|^{(1)}$ and $|\psi\rangle^{(2)}\langle\psi|^{(2)}$. What do you notice?

See also **Q3.19** and **Q3.20**.

3

spin and angular momentum



In this chapter we put together everything we have studied so far—Mathematica, quantum mechanics, computational bases, units—to study simple quantum systems. We start our explorations of quantum mechanics with the description of angular momentum. The reason for this choice is that, in contrast to the mechanically more intuitive linear motion (chapter 4), rotational motion is described with finite-dimensional Hilbert spaces and thus lends itself as a relatively simple starting point. As applications we look at the hyperfine structure of alkali atoms, lattice spin models, and quantum circuits.

3.1 quantum-mechanical spin and angular momentum operators [code]

A classical rotational motion is described by its angular momentum, which is a three-dimensional pseudovector[1] whose direction indicates the rotation axis and whose length gives the rotational momentum. For an isolated system, the angular momentum is conserved and is thus very useful in the description of the system's state.

In quantum mechanics, angular momentum is equally described by a three-dimensional pseudovector operator $\hat{\boldsymbol{S}}$, with operator elements (in Cartesian coordinates) $\hat{\boldsymbol{S}} = (\hat{S}_x, \hat{S}_y, \hat{S}_z)$. The joint eigenstates of the squared angular momentum magnitude $\|\hat{\boldsymbol{S}}\|^2 = \hat{S}^2 = \hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2$ and of the z-component $\hat{S}_z$ are called the Dicke states $|S, M\rangle$, and satisfy

$$\hat{S}^2|S, M\rangle = S(S+1)\ |S, M\rangle \tag{3.1a}$$
$$\hat{S}_z|S, M\rangle = M\ |S, M\rangle \tag{3.1b}$$

For every integer or half-integer value of the angular momentum $S \in \{0, \frac{1}{2}, 1, \frac{3}{2}, 2, \ldots\}$, there is a set of $2S+1$ Dicke states $|S, M\rangle$ with $M \in \{-S, -S+1, \ldots, S-1, S\}$ that form a basis for the description of the rotation axis orientation. These states also satisfy the following relationships with respect to the x- and y-components of the angular momentum:

$$\hat{S}_+|S, M\rangle = \sqrt{S(S+1) - M(M+1)}\ |S, M+1\rangle \qquad \text{raising operator} \tag{3.2a}$$
$$\hat{S}_-|S, M\rangle = \sqrt{S(S+1) - M(M-1)}\ |S, M-1\rangle \qquad \text{lowering operator} \tag{3.2b}$$
$$\hat{S}_\pm = \hat{S}_x \pm \mathrm{i}\hat{S}_y \qquad \text{Cartesian components} \tag{3.2c}$$

As you know, quantum mechanics is not limited to spins or angular momenta of length $S = 1/2$.

In Mathematica we represent these operators in the Dicke basis as follows, with the elements of the basis set ordered with decreasing projection quantum number M:

```
In[258]:= SpinQ[S_] := IntegerQ[2S] && S>=0
In[259]:= splus[0] = {{0}} //SparseArray;
In[260]:= splus[S_?SpinQ] := splus[S] =
            SparseArray[Band[{1,2}] -> Table[Sqrt[S(S+1)-M(M+1)],
              {M,S-1,-S,-1}], {2S+1,2S+1}]
In[261]:= sminus[S_?SpinQ] := Transpose[splus[S]]
In[262]:= sx[S_?SpinQ] := sx[S] = (splus[S]+sminus[S])/2
In[263]:= sy[S_?SpinQ] := sy[S] = (splus[S]-sminus[S])/(2I)
In[264]:= sz[S_?SpinQ] := sz[S] = SparseArray[Band[{1,1}]->Range[S,-S,-1], {2S+1,2S+1}]
In[265]:= id[S_?SpinQ] := id[S] = IdentityMatrix[2S+1, SparseArray]
```

- Notice that we have defined all these matrix representations as *sparse* matrices (see section 1.10.3), which will make larger calculations much more efficient later on. Further, all definitions are memoizing (see section 1.6.3) to reduce execution time when they are used repeatedly.

- The function `SpinQ[S]` yields `True` only if `S` is a nonnegative half-integer value and can therefore represent a physically valid spin. In general, functions ending in `...Q` are *questions* on the character of an argument (see section 1.6.4).

- The operator $\hat{S}_+$, defined with `splus[S]`, contains only one off-diagonal band of non-zero values. The `SparseArray` matrix constructor allows building such banded matrices by simply specifying the starting point of the band and a vector with the elements of the nonzero band.

- The operator $\hat{S}_z$, defined with `sz[S]`, shows you the ordering of the basis elements since it has the projection quantum numbers on the diagonal.

[1] See https://en.wikipedia.org/wiki/Pseudovector.

- The last operator `id[S]` is the unit operator operating on a spin of length `S`, and will be used below for tensor-product definitions. Note that the `IdentityMatrix` function usually returns a full matrix, which is not suitable for large-scale calculations. By giving it a `SparseArray` option, it returns a sparse identity matrix of desired size.

- All these matrices can be displayed with, for example,

```
In[266]:= sx[3/2] //Normal
Out[266]= {{0, Sqrt[3]/2, 0, 0},
          {Sqrt[3]/2, 0, 1, 0},
          {0, 1, 0, Sqrt[3]/2},
          {0, 0, Sqrt[3]/2, 0}}
```

or, for a more traditional view,

```
In[267]:= sx[3/2] //MatrixForm
```

3.1.1 exercises

Q3.1 Verify that for $S = 1/2$ the above Mathematica definitions give the Pauli matrices: $\hat{S}_i = \frac{1}{2}\hat{\sigma}_i$ for $i = x, y, z$.

Q3.2 Verify in Mathematica that for given integer or half-integer S, the three operators (matrices) $\hat{\vec{S}} = \{\hat{S}_x, \hat{S}_y, \hat{S}_z\}$ behave like a quantum-mechanical pseudovector of length $\|\hat{\vec{S}}\| = \sqrt{S(S+1)}$:

1. Show that $[\hat{S}_x, \hat{S}_y] = \mathrm{i}\hat{S}_z$, $[\hat{S}_y, \hat{S}_z] = \mathrm{i}\hat{S}_x$, and $[\hat{S}_z, \hat{S}_x] = \mathrm{i}\hat{S}_y$.
2. Show that $\hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2 = S(S+1)\mathbb{1}$.
3. What is the largest value of S for which you can do these verifications within one minute (each) on your computer? *Hint:* use the `Timing` function.

Q3.3 The operators $\hat{S}_{x,y,z}$ are the generators of rotations: a rotation by an angle α around the axis given by a normalized vector $\vec{n}$ is done with the operator $\hat{R}_{\vec{n}}(\alpha) = \exp(-\mathrm{i}\alpha\vec{n}\cdot\hat{\vec{S}})$. Set $\vec{n} = \{\sin(\vartheta)\cos(\varphi), \sin(\vartheta)\sin(\varphi), \cos(\vartheta)\}$ and calculate the operator $\hat{R}_{\vec{n}}(\alpha)$ explicitly for $S = 0$, $S = 1/2$, and $S = 1$. Check that for $\alpha = 0$ you find the unit operator.

3.2 spin-1/2 electron in a dc magnetic field [code]

As a first example we look at a single spin $S = 1/2$. We use the basis containing the two states $|\uparrow\rangle = |\frac{1}{2}, \frac{1}{2}\rangle$ and $|\downarrow\rangle = |\frac{1}{2}, -\frac{1}{2}\rangle$, which we know to be eigenstates of the operators $\hat{S}^2$ and $\hat{S}_z$. The matrix expressions of the operators relevant for this system are given by the Pauli matrices divided by two,

$$\boldsymbol{S}_x = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \frac{1}{2}\boldsymbol{\sigma}_x \qquad \boldsymbol{S}_y = \frac{1}{2}\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix} = \frac{1}{2}\boldsymbol{\sigma}_y \qquad \boldsymbol{S}_z = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{2}\boldsymbol{\sigma}_z \tag{3.3}$$

In Mathematica we enter these as

```
In[268]:= Sx = sx[1/2]; Sy = sy[1/2]; Sz = sz[1/2];
```

using the general definitions of angular momentum operators given in section 3.1. Alternatively, we can write

```
In[269]:= {Sx,Sy,Sz} = (1/2) * Table[PauliMatrix[i], {i,1,3}];
```

As a Hamiltonian we use the coupling of this electron spin to an external magnetic field, $\hat{\mathcal{H}} = -\hat{\vec{\boldsymbol{\mu}}} \cdot \vec{\boldsymbol{B}}$. The magnetic moment of the electron is $\hat{\vec{\boldsymbol{\mu}}} = \mu_\mathrm{B} g_\mathrm{e} \hat{\vec{\boldsymbol{S}}}$ in terms of its spin $\hat{\vec{\boldsymbol{S}}}$, the Bohr magneton $\mu_\mathrm{B} = 9.274\,009\,68(20) \times 10^{-24}\,\mathrm{J/T}$, and the electron's g-factor $g_\mathrm{e} = -2.002\,319\,304\,362\,2(15)$.[2] The Hamiltonian is therefore

$$\hat{\mathcal{H}} = -\mu_\mathrm{B} g_\mathrm{e} (\hat{S}_x B_x + \hat{S}_y B_y + \hat{S}_z B_z). \tag{3.4}$$

In our chosen matrix representation this Hamiltonian is

$$\boldsymbol{H} = -\mu_\mathrm{B} g_\mathrm{e} (\boldsymbol{S}_x B_x + \boldsymbol{S}_y B_y + \boldsymbol{S}_z B_z) = -\frac{1}{2} \mu_\mathrm{B} g_\mathrm{e} \begin{pmatrix} B_z & B_x - \mathrm{i} B_y \\ B_x + \mathrm{i} B_y & -B_z \end{pmatrix}. \tag{3.5}$$

In order to implement this Hamiltonian, we first define a system of units. Here we express magnetic field strengths in Gauss and energies in MHz times Planck's constant (it is common to express energies in units of frequency, where the conversion is sometimes implicitly done via Planck's constant):

```
In[270]:= MagneticFieldUnit = Quantity["Gausses"];
In[271]:= EnergyUnit = Quantity["PlanckConstant"]*Quantity["MHz"] //UnitConvert;
```

In this unit system, the Bohr magneton is approximately 1.4 MHz/G:

```
In[272]:= μB = Quantity["BohrMagneton"]/(EnergyUnit/MagneticFieldUnit) //UnitConvert
Out[272]= 1.3996245
```

We define the electron's g-factor with

```
In[273]:= ge = UnitConvert["ElectronGFactor"]
Out[273]= -2.00231930436
```

The Hamiltonian of Equation (3.4) is then

```
In[274]:= H[Bx_, By_, Bz_] = -μB * ge * (Sx*Bx+Sy*By+Sz*Bz)
```

natural units

An alternative choice of units, called *natural units*, is designed to simplify a calculation by making the numerical value of the largest possible number of quantities equal to 1. In the present case, this would be achieved by relating the field and energy units to each other in such a way that the Bohr magneton becomes equal to 1:

```
In[275]:= MagneticFieldUnit = Quantity["Gausses"];
In[276]:= EnergyUnit = MagneticFieldUnit * Quantity["BohrMagneton"] //UnitConvert;
In[277]:= μB = Quantity["BohrMagneton"]/(EnergyUnit/MagneticFieldUnit) //UnitConvert
Out[277]= 1.0000000
```

In this way, calculations can often be simplified substantially because the Hamiltonian effectively becomes much simpler than it looks in other unit systems. We will be coming back to this point in future calculations.

[2] Notice that the magnetic moment of the electron is anti-parallel to its spin ($g_e < 0$). The reason for this is the electron's negative electric charge. When the electron spin is parallel to the magnetic field, the electron's energy is higher than when they are anti-parallel.

3.2.1 time-independent Schrödinger equation

The time-independent Schrödinger equation for our spin-1/2 problem is, from Equation (2.14),

$$-\frac{1}{2}\mu_B g_e \begin{pmatrix} B_z & B_x - iB_y \\ B_x + iB_y & -B_z \end{pmatrix} \cdot \vec{\psi} = E\vec{\psi} \tag{3.6}$$

The eigenvalues of the Hamiltonian (in our chosen energy units) and eigenvectors are calculated with:

```
In[278]:=Eigensystem[H[Bx,By,Bz]]
```

As described in section 1.10.4 the output is a list with two entries, the first being a list of eigenvalues and the second a list of associated eigenvectors. As long as the Hamiltonian matrix is Hermitian, the eigenvalues will all be real-valued; but the eigenvectors can be complex. Since the Hilbert space of this spin problem has dimension 2, and the basis contains two vectors, there are necessarily two eigenvalues and two associated eigenvectors of length 2. The eigenvalues can be called $E_\pm = \pm\frac{1}{2}\mu_B g_e \|\vec{B}\|$. The list of eigenvalues is given in the Mathematica output as $\{E'_-, E'_+\}$. Notice that these eigenvalues only depend on the magnitude of the magnetic field, and not on its direction. This is to be expected: since there is no preferred axis in this system, there cannot be any directional dependence. The choice of the basis as the eigenstates of the $\hat{S}_z$ operator was entirely arbitrary, and therefore the energy eigenvalues cannot depend on the orientation of the magnetic field with respect to this quantization axis.

The associated eigenvectors are

$$\vec{\psi}_\pm = \{\frac{B_z \pm \|\vec{B}\|}{B_x + iB_y}, 1\}, \tag{3.7}$$

which Mathematica returns as a list of lists, $\{\vec{\psi}_-, \vec{\psi}_+\}$. Notice that these eigenvectors are not normalized.

3.2.2 exercises

Q3.4 Calculate the eigenvalues (in units of J) and eigenvectors (ortho-normalized) of an electron spin in a magnetic field of 1 T in the x-direction.

Q3.5 Set $\vec{B} = B[\vec{e}_x \sin(\vartheta)\cos(\varphi) + \vec{e}_y \sin(\vartheta)\sin(\varphi) + \vec{e}_z \cos(\vartheta)]$ and calculate the eigenvalues and normalized eigenvectors of the electron spin Hamiltonian.

3.3 coupled spin systems: ^{87}Rb hyperfine structure [code]

Ground-state Rubidium-87 atoms consist of a nucleus with spin $I = 3/2$, a single valence electron (spin $S = 1/2$, orbital angular momentum $L = 0$, and therefore total spin $J = 1/2$), and 36 core electrons that do not contribute any angular momentum. In a magnetic field along the z-axis, the effective Hamiltonian of this system is[3]

$$\hat{\mathcal{H}} = \hat{\mathcal{H}}_0 + hA_{\text{hfs}}\,\hat{\vec{I}}\cdot\hat{\vec{J}} - \mu_B B_z(g_I\hat{I}_z + g_S\hat{S}_z + g_L\hat{L}_z), \tag{3.8}$$

where h is Planck's constant, μ_B is the Bohr magneton, $A_{\text{hfs}} = 3.417\,341\,305\,452\,145(45)$ GHz is the spin–spin coupling constant in the ground state of ^{87}Rb, $g_I = +0.000\,995\,141\,4(10)$ is the nuclear g-factor, $g_S = -2.002\,319\,304\,362\,2(15)$ is the electron spin g-factor, and $g_L = -0.999\,993\,69$ is the electron orbital g-factor.

The first part $\hat{\mathcal{H}}_0$ of Equation (3.8) contains all electrostatic interactions, core electrons, nuclear interactions etc. We will assume that the system is in the ground state of $\hat{\mathcal{H}}_0$, which means that the valence electron is in the $5^2S_{1/2}$ state and the nucleus is deexcited. This ground state is eight-fold degenerate and consists of the four magnetic sublevels of the $I = 3/2$ nuclear spin, the two sublevels of the $S = 1/2$ electronic spin, and the single level of the $L = 0$ angular momentum. The basis for the description of this atom is therefore the tensor product basis of a spin-3/2, a spin-1/2, and a spin-0.[4]

The spin operators acting on this composite system are defined as in section 2.4.2. For example, the nuclear-spin operator $\hat{I}_x$ is extended to the composite system by acting trivially on the electron spin and

[3] See http://steck.us/alkalidata/rubidium87numbers.pdf.

[4] The spin-0 subsystem is trivial and could be left out in principle. It is included here to show the method in a more general way.

orbital angular momenta, $\hat{I}_x \mapsto \hat{I}_x \otimes \mathbb{1} \otimes \mathbb{1}$. The electron-spin operators are defined accordingly, for example $\hat{S}_x \mapsto \mathbb{1} \otimes \hat{S}_x \otimes \mathbb{1}$. The electron orbital angular momentum operators are, for example, $\hat{L}_x \mapsto \mathbb{1} \otimes \mathbb{1} \otimes \hat{L}_x$. In Mathematica these operators are defined with

```
In[279]:= Ix = KroneckerProduct[sx[3/2], id[1/2], id[0]];
In[280]:= Iy = KroneckerProduct[sy[3/2], id[1/2], id[0]];
In[281]:= Iz = KroneckerProduct[sz[3/2], id[1/2], id[0]];
In[282]:= Sx = KroneckerProduct[id[3/2], sx[1/2], id[0]];
In[283]:= Sy = KroneckerProduct[id[3/2], sy[1/2], id[0]];
In[284]:= Sz = KroneckerProduct[id[3/2], sz[1/2], id[0]];
In[285]:= Lx = KroneckerProduct[id[3/2], id[1/2], sx[0]];
In[286]:= Ly = KroneckerProduct[id[3/2], id[1/2], sy[0]];
In[287]:= Lz = KroneckerProduct[id[3/2], id[1/2], sz[0]];
```

The total electron angular momentum is $\hat{\vec{J}} = \hat{\vec{S}} + \hat{\vec{L}}$:

```
In[288]:= Jx = Sx + Lx;   Jy = Sy + Ly;   Jz = Sz + Lz;
```

The total angular momentum of the ^{87}Rb atom is $\hat{\vec{F}} = \hat{\vec{I}} + \hat{\vec{J}}$:

```
In[289]:= Fx = Ix + Jx;   Fy = Iy + Jy;   Fz = Iz + Jz;
```

Before defining the system's Hamiltonian, we declare a system of units. Any system will work here, so we stay with units commonly used in atomic physics: magnetic fields are expressed in Gauss, while energies are expressed in MHz times Planck's constant. As time unit we choose the microsecond:

```
In[290]:= MagneticFieldUnit = Quantity["Gausses"];
In[291]:= EnergyUnit = Quantity["PlanckConstant"] * Quantity["Megahertz"];
In[292]:= TimeUnit = Quantity["Microseconds"];
```

The numerical values of the Bohr Magneton and the reduced Planck constant in these units are

```
In[293]:= μBn = Quantity["BohrMagneton"]/(EnergyUnit/MagneticFieldUnit)
Out[293]= 1.3996245
In[294]:= ℏn = Quantity["ReducedPlanckConstant"]/(EnergyUnit*TimeUnit)
Out[294]= 0.15915494
```

Using these definitions we define the hyperfine Hamiltonian with magnetic field in the z-direction as

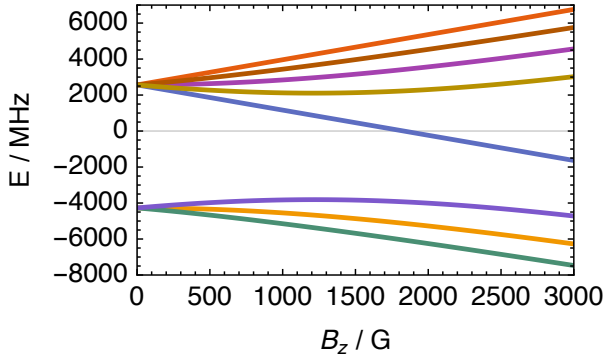
```
In[295]:= Hhf = A(Ix.Jx+Iy.Jy+Iz.Jz) - μB*Bz*(gI*Iz+gS*Sz+gL*Lz);
In[296]:= hfc = {μB -> μBn, ℏ -> ℏn,
          A->Quantity["PlanckConstant"]*Quantity[3.417341305452145,"GHz"]/EnergyUnit,
          gS -> -2.0023193043622,
          gL -> -0.99999369,
          gI -> +0.0009951414};
```

This yields the Hamiltonian as an 8×8 matrix, and we can calculate its eigenvalues and eigenvectors with

```
In[297]:= {eval, evec} = Eigensystem[Hhf] //FullSimplify;
```

We plot the energy eigenvalues with

```
In[298]:= Plot[Evaluate[eval /. hfc], {Bz, 0, 3000},
          Frame -> True, FrameLabel -> {"Bz / G", "E / MHz"}]
```



3.3.1 eigenstate analysis

In this section we analyze the results `eval` and `evec` from the Hamiltonian diagonalization above. For this we first need to define *ortho-normalized* eigenvectors since in general we cannot assume `evec` to be ortho-normalized.

In general we can always define an ortho-normalized eigenvector set with

```
In[299]:=nevec = Orthogonalize[evec]
```

The problem with this definition is, however, immediately apparent if you look at the output given by Mathematica: since no assumptions on the reality of the variables were made, the orthogonalization is done in too much generality and quickly becomes unwieldy. Even using `Assuming` and `ComplexExpand`, as in section 1.11, does not give satisfactory results. But if we notice that the eigenvectors in `evec` are all purely real-values, and are already orthogonal, then a simple vector-by-vector normalization is sufficient for calculating an ortho-normalized eigenvector set:

```
In[300]:=nevec = #/Sqrt[#.#] & /@ evec;
In[301]:=nevec . Transpose[nevec] //FullSimplify
```

The fact that `In[301]` finds a unit matrix implies that the vectors in `nevec` are ortho-normal.

field-free limit

In the field-free limit $B_z = 0$ the energy levels are

```
In[302]:=Assuming[A > 0, Limit[eval, Bz -> 0]]
Out[302]={3A/4, 3A/4, -5A/4, 3A/4, -5A/4, 3A/4, -5A/4, 3A/4}
```

We see that the level with energy $-\frac{5}{4}A$ is three-fold degenerate while the level with energy $\frac{3}{4}A$ is five-fold degenerate. This is also visible in the eigenvalue plot above. Considering that we have coupled two spins of lengths $I = \frac{3}{2}$ and $J = \frac{1}{2}$, we expect the composite system to have either total spin $F = 1$ (three sublevels) or $F = 2$ (five sublevels); we can make the tentative assignment that the $F = 1$ level is at energy $E_1 = -\frac{5}{4}A$ and the $F = 2$ level at $E_2 = \frac{3}{4}A$.

In order to demonstrate this assignment we express the matrix elements of the operators $\hat{F}^2$ and $\hat{F}_z$ in the field-free eigenstates, making sure to normalize these eigenstates before taking the limit $B_z \to 0$:

```
In[303]:=nevec0 = Assuming[A > 0, Limit[nevec, Bz -> 0]];
In[304]:=nevec0 . (Fx.Fx+Fy.Fy+Fz.Fz) . Transpose[nevec0]
In[305]:=nevec0 . Fz . Transpose[nevec0]
```

Notice that in this calculations we have used the fact that all eigenvectors are real, which may not always be the case for other Hamiltonians. We see that the field-free normalized eigenvectors `nevec0` are eigenvectors of both $\hat{F}^2$ and $\hat{F}_z$, and from looking at the eigenvalues we can identify them as

$$\{|2,2\rangle, |2,-2\rangle, |1,0\rangle, |2,0\rangle, |1,1\rangle, |2,1\rangle, |1,-1\rangle, |2,-1\rangle\} \tag{3.9}$$

in the notation $|F, M_F\rangle$. These labels are often used to identify the energy eigenstates even for small $B_z \neq 0$.

low-field limit

For small magnetic fields, we series-expand the energy eigenvalues to first order in B_z:

```
In[306]:= Assuming[A > 0, Series[eval, {Bz, 0, 1}] //FullSimplify]
```

From these low-field terms, in combination with the field-free level assignment, we see that the $F = 1$ and $F = 2$ levels have effective g-factors of $g_1 = (-g_S + 5g_I)/4 \approx 0.501\,824$ and $g_2 = -(-g_S - 3g_I)/4 \approx -0.499\,833$, respectively, so that their energy eigenvalues follow the form

$$E_{F,M_F}(B_z) = E_F(0) - \mu_B M_F g_F B_z + \mathcal{O}(B_z^2). \tag{3.10}$$

These energy shifts due to the magnetic field are called *Zeeman shifts*.

high-field limit

The energy eigenvalues in the high-field limit are infinite; but we can calculate their lowest-order series expansions with

```
In[307]:= Assuming[μB > 0 && gS < -gI < 0,
            Series[eval, {Bz, Infinity, 0}] //FullSimplify]
```

From these expansions we can already identify the states in the eigenvalue plot above.

In order to calculate the eigenstates in the high-field limit we must again make sure to normalize the states *before* taking the limit $B_z \to \infty$:[5]

```
In[308]:= nevecinf = Assuming[μB > 0 && gS < -gI < 0,
             FullSimplify[Limit[nevec, Bz -> Infinity], A > 0]]
Out[308]= {{1,  0, 0,  0, 0,  0, 0, 0},
           {0,  0, 0,  0, 0,  0, 0, 1},
           {0,  0, 0, -1, 0,  0, 0, 0},
           {0,  0, 0,  0, 1,  0, 0, 0},
           {0,  -1, 0,  0, 0, 0, 0, 0},
           {0,  0, 1,  0, 0,  0, 0, 0},
           {0, 0, 0,  0, 0,  -1, 0, 0},
           {0,  0, 0,  0, 0,  0, 1, 0}}
```

From this we immediately identify the high-field eigenstates as our basis states in a different order,

$$\{|\tfrac{3}{2}, \tfrac{1}{2}\rangle, |-\tfrac{3}{2}, -\tfrac{1}{2}\rangle, |\tfrac{1}{2}, -\tfrac{1}{2}\rangle, |-\tfrac{1}{2}, \tfrac{1}{2}\rangle, |\tfrac{3}{2}, -\tfrac{1}{2}\rangle, |\tfrac{1}{2}, \tfrac{1}{2}\rangle, |-\tfrac{1}{2}, -\tfrac{1}{2}\rangle, |-\tfrac{3}{2}, \tfrac{1}{2}\rangle\} \tag{3.11}$$

where we have used the abbreviation $|M_I, M_J\rangle = |\frac{3}{2}, M_I\rangle \otimes |\frac{1}{2}, M_J\rangle$. You can verify this assignment by looking at the matrix elements of the $\hat{I}_z$ and $\hat{J}_z$ operators with

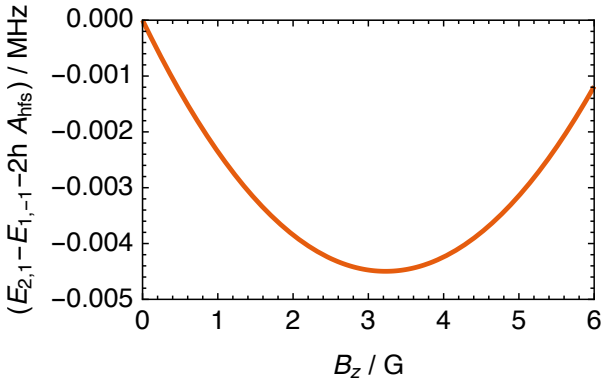
```
In[309]:= nevecinf . Iz . Transpose[nevecinf]
In[310]:= nevecinf . Jz . Transpose[nevecinf]
```

[5] Note that in In[308] we use two stages of assumptions, using the assumption $A > 0$ only in FullSimplify but not in Limit. This is done in order to work around an inconsistency in Mathematica 11.3.0.0, and may be simplified in a future edition.

3.3.2 "magic" magnetic field

The energy eigenvalues of the low-field states $|1,-1\rangle$ and $|2,1\rangle$ have almost the same first-order magnetic field dependence since $g_1 \approx -g_2$ (see low-field limit above). If we plot their energy difference as a function of magnetic field we find an extremal point:

```
In[311]:=Plot[eval[[6]]-eval[[7]]-2A /. hfc, {Bz, 0, 6}]
```



At the "magic" field strength $B_0 = 3.228\,96\,\mathrm{G}$ the energy difference is independent of the magnetic field (to first order):

```
In[312]:=NMinimize[eval[[6]] - eval[[7]] - 2 A /. hfc, Bz]
Out[312]={-0.00449737, {Bz -> 3.22896}}
```

This is an important discovery for quantum information science with ^{87}Rb atoms. If we store a qubit in the state $|\vartheta,\varphi\rangle = \cos(\vartheta/2)|1,-1\rangle + e^{i\varphi}\sin(\vartheta/2)|2,1\rangle$ and tune the magnetic field exactly to the magic value, then the experimentally unavoidable magnetic-field fluctuations will not lead to fluctuations of the energy difference between the two atomic levels and thus will not lead to qubit decoherence. Very long qubit coherence times can be achieved in this way.

For the present case where $|g_I| \ll |g_S|$, the magic field is approximately $B_z \approx \frac{16Ag_I}{3\mu_B g_S^2}$.

3.3.3 coupling to an oscillating magnetic field

In this section we study the coupling of a ^{87}Rb atom to a weak oscillating magnetic field. Such a field could be the magnetic part of an electromagnetic wave, whose electric field does not couple to our atom in the electronic ground state. This calculation is a template for more general situations where a quantum-mechanical system is driven by an oscillating field.

The ^{87}Rb hyperfine Hamiltonian in the presence of an oscillating magnetic field is

$$\hat{\mathcal{H}}(t) = \underbrace{hA_{\mathrm{hfs}}\,\hat{\boldsymbol{I}}\cdot\hat{\boldsymbol{J}} - \mu_B B_z(g_I\hat{I}_z + g_S\hat{S}_z + g_L\hat{L}_z)}_{\hat{\mathcal{H}}_0} - \cos(\omega t)\times\underbrace{\mu_B\vec{\boldsymbol{B}}^{\mathrm{ac}}\cdot(g_I\hat{\boldsymbol{I}} + g_S\hat{\boldsymbol{S}} + g_L\hat{\boldsymbol{L}})}_{-\hat{\mathcal{H}}_1} \tag{3.12}$$

where the static magnetic field is assumed to be in the z direction, as before. Unfortunately, $[\hat{\mathcal{H}}(t),\hat{\mathcal{H}}(t')] = [\hat{\mathcal{H}}_1,\hat{\mathcal{H}}_0]\,(\cos(\omega t)-\cos(\omega t')) \neq 0$ in general, so we cannot use the exact solution of Equation (2.33) of the time-dependent Schrödinger equation. In fact, the time-dependent Schrödinger equation of this system has no analytic solution at all. In what follows we will calculate approximate solutions.

Since we have diagonalized the time-independent Hamiltonian $\hat{\mathcal{H}}_0$ already, we use its eigenstates as a basis for calculating the effect of the oscillating perturbation $\hat{\mathcal{H}}_1(t)$. In general, calling $\{|i\rangle\}_{i=1}^{8}$ the set of eigenstates of $\hat{\mathcal{H}}_0$, with $\hat{\mathcal{H}}_0|i\rangle = E_i|i\rangle$ for $i \in \{1\ldots8\}$, we expand the general hyperfine state as in Equation (2.28),

$$|\psi(t)\rangle = \sum_{i=1}^{8}\psi_i(t)e^{-iE_it/\hbar}|i\rangle. \tag{3.13}$$

The time-dependent Schrödinger equation for the expansion coefficients $\psi_i(t)$ in this interaction picture is given in Equation (2.30): for $i = 1 \ldots 8$ we have

$$\mathrm{i}\hbar\dot{\psi}_i(t) = \sum_{j=1}^{8} \psi_j(t) e^{-\mathrm{i}(E_j - E_i)t/\hbar} \cos(\omega t) \langle i|\hat{\mathcal{H}}_1|j\rangle = \frac{1}{2}\sum_{j=1}^{8} \psi_j(t) \left[e^{-\mathrm{i}\left(\frac{E_j - E_i}{\hbar} - \omega\right)t} + e^{\mathrm{i}\left(\frac{E_i - E_j}{\hbar} - \omega\right)t} \right] T_{ij}, \quad (3.14)$$

where we have replaced $\cos(\omega t) = \frac{1}{2}e^{\mathrm{i}\omega t} + \frac{1}{2}e^{-\mathrm{i}\omega t}$ and defined

$$T_{ij} = \langle i|\hat{\mathcal{H}}_1|j\rangle = -\langle i| \left[\mu_\mathrm{B} \vec{B}^\mathrm{ac} \cdot (g_I \hat{\vec{I}} + g_S \hat{\vec{S}} + g_L \hat{\vec{L}}) \right] |j\rangle. \quad (3.15)$$

From Equation (3.14) we can proceed in various ways:

Transition matrix elements: The time-independent matrix elements T_{ij} of the perturbation Hamiltonian are called the *transition matrix elements* and describe how the populations of the different eigenstates of $\hat{\mathcal{H}}_0$ are coupled through the oscillating field. We calculate them in Mathematica as follows:

```
In[313]:= H0 = A*(Ix.Jx + Iy.Jy + Iz.Jz) - μB*Bz*(gS*Sz + gL*Lz + gI*Iz);
In[314]:= H1 = -μB*(gS*(Bacx*Sx + Bacy*Sy + Bacz*Sz)
                + gI*(Bacx*Ix + Bacy*Iy + Bacz*Iz)
                + gL*(Bacx*Lx + Bacy*Ly + Bacz*Lz));
In[315]:= H[t_] = H0 + H1*Cos[ω*t];
In[316]:= {eval, evec} = Eigensystem[H0] //FullSimplify;
In[317]:= nevec = Map[#/Sqrt[#.#] &, evec];
In[318]:= T = Assuming[A > 0, nevec.H1.Transpose[nevec] //FullSimplify];
```

Looking at this matrix $\boldsymbol{T}$ we see that not all energy levels are directly coupled by an oscillating magnetic field. For example, $T_{1,2} = 0$ indicates that the populations of the states $|1\rangle$ and $|2\rangle$ can only be coupled indirectly through other states, but not directly (hint: check `T[[1,2]]`).

Numerical solution: Equation (3.14) is a series of linear coupled differential equations, which we write down explicitly in Mathematica with

```
In[319]:= deqs = Table[I*ħ*Subscript[ψ,i]'[t] ==
                 Sum[Subscript[ψ,j][t]*Exp[-I*(eval[[j]]-eval[[i]])*t/ħ]
                   *Cos[ω*t]*T[[i,j]], {j, 8}], {i,8}];
```

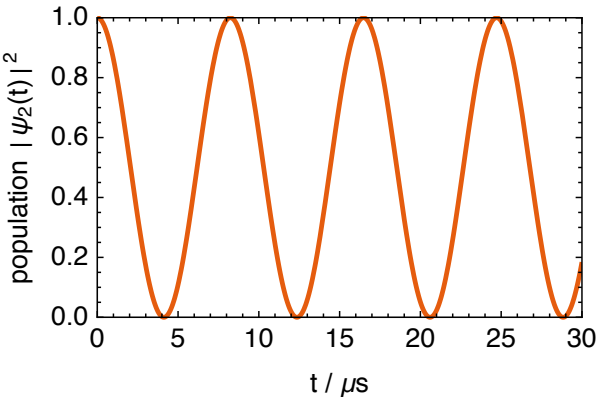
Assuming concrete conditions, for example the initial state $|\psi(t=0)\rangle = |F=2, M_F=-2\rangle$ which is the second eigenstate `nevec[[2]]` [see Equation (3.9)], and magnetic fields $B_z = 3.228\,96\,\mathrm{G}$, $B_x^\mathrm{ac} = 100\,\mathrm{mG}$, $B_y^\mathrm{ac} = B_z^\mathrm{ac} = 0$, and an ac field angular frequency of $\omega = 2\pi \times 6827.9\,\mathrm{MHz}$, we can find the time-dependent state $|\psi(t)\rangle$ with

```
In[320]:= S = NDSolve[Join[deqs /. hfc /.{Bz->3.22896, Bacx->0.1, Bacy->0, Bacz->0,
              ω->2*π*6827.9},
            {Subscript[ψ,1][0]==0,Subscript[ψ,2][0]==1,
             Subscript[ψ,3][0]==0,Subscript[ψ,4][0]==0,
             Subscript[ψ,5][0]==0,Subscript[ψ,6][0]==0,
             Subscript[ψ,7][0]==0,Subscript[ψ,8][0]==0}],
            Table[Subscript[ψ,i][t],{i,8}], {t, 0, 30},
            MaxStepSize->10^(-5), MaxSteps->10^7]
```

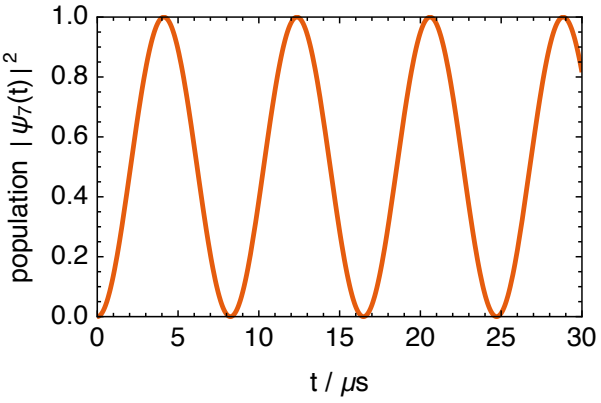
Notice that the maximum step size in this numerical solution is very small (10^{-5} time units or 10 ps), since it needs to capture the fast oscillations of more than 6.8 GHz. As a result, a large number of numerical steps is required, which makes this way of studying the evolution very difficult in practice.

We plot the resulting populations with

```
In[321]:= Plot[Evaluate[Abs[Subscript[ψ,2][t] /. S[[1]]]^2], {t, 0, 30}]
```




```
In[322]:=Plot[Evaluate[Abs[Subscript[ψ,7][t] /. S[[1]]]^2], {t, 0, 30}]
```



We see that the population is mostly sloshing between $\hat{\mathcal{H}}_0$-eigenstates $|2\rangle \approx |F=2, M_F=-2\rangle$ and $|7\rangle \approx |F=1, M_F=-1\rangle$ [see Equation (3.9)]. Each population oscillation takes about 8.2 µs (the Rabi period), and we say that the Rabi frequency is about 120 kHz.

Rotating-wave approximation: The time-dependent prefactor $\exp\left[-\mathrm{i}\left(\frac{E_j-E_i}{\hbar}-\omega\right)t\right]+\exp\left[\mathrm{i}\left(\frac{E_i-E_j}{\hbar}-\omega\right)t\right]$ of Equation (3.14) oscillates very rapidly unless either $\frac{E_j-E_i}{\hbar}-\omega \approx 0$ or $\frac{E_i-E_j}{\hbar}-\omega \approx 0$, where one of its terms changes slowly in time. The *rotating-wave approximation* (RWA) consists of neglecting all rapidly rotating terms in Equation (3.14). Assume that there is a single[6] pair of states $|i\rangle$ and $|j\rangle$ such that $E_i - E_j \approx \hbar\omega$, with $E_i > E_j$, while all other states have an energy difference far from $\hbar\omega$. The RWA thus consists of simplifying Equation (3.14) to

$$\begin{aligned}
\mathrm{i}\hbar\dot{\psi}_i(t) &\approx \frac{1}{2}\psi_j(t)e^{\mathrm{i}\left(\frac{E_i-E_j}{\hbar}-\omega\right)t}T_{ij} = \frac{1}{2}\psi_j(t)T_{ij}e^{-\mathrm{i}\Delta t}\\
\mathrm{i}\hbar\dot{\psi}_j(t) &\approx \frac{1}{2}\psi_i(t)e^{-\mathrm{i}\left(\frac{E_i-E_j}{\hbar}-\omega\right)t}T_{ji} = \frac{1}{2}\psi_i(t)T_{ji}e^{\mathrm{i}\Delta t}\\
\mathrm{i}\hbar\dot{\psi}_k(t) &\approx 0 \text{ for } k \notin \{i,j\}
\end{aligned} \tag{3.16}$$

with $T_{ji} = T_{ij}^*$ and the detuning $\Delta = \omega - (E_i - E_j)/\hbar$. All other terms in Equation (3.14) have been neglected because they rotate so fast in time that they "average out" to zero. This approximate system of differential equations has the exact solution

$$\begin{aligned}
\psi_i(t) &= e^{-\frac{\mathrm{i}}{2}\Delta t}\left[\psi_i(0)\cos\left(\frac{\Omega t}{2}\right)+\mathrm{i}\left(\frac{\Delta}{\Omega}\psi_i(0)-\frac{T_{ij}}{\hbar\Omega}\psi_j(0)\right)\sin\left(\frac{\Omega t}{2}\right)\right]\\
\psi_j(t) &= e^{\frac{\mathrm{i}}{2}\Delta t}\left[\psi_j(0)\cos\left(\frac{\Omega t}{2}\right)-\mathrm{i}\left(\frac{\Delta}{\Omega}\psi_j(0)+\frac{T_{ij}^*}{\hbar\Omega}\psi_i(0)\right)\sin\left(\frac{\Omega t}{2}\right)\right]\\
\psi_k(t) &= \psi_k(0) \text{ for } k \notin \{i,j\}
\end{aligned} \tag{3.17}$$

[6] The following derivation is readily extended to situations where several pairs of states have an energy difference approximately equal to $\hbar\omega$. In such a case we need to solve a larger system of coupled differential equations.

in terms of the generalized Rabi frequency $\Omega = \sqrt{|T_{ij}|^2/\hbar^2 + \Delta^2}$. We can see that the population sloshes back and forth ("Rabi oscillation") between the two levels $|i\rangle$ and $|j\rangle$ with angular frequency Ω, as we had seen numerically above.

We can verify this solution im Mathematica as follows. First we define

```
In[323]:= Δ = ω - (Ei-Ej)/ħ;
In[324]:= Ω = Sqrt[Tij*Tji/ħ^2 + Δ^2];
```

and the solutions

```
In[325]:= ψi[t_] = E^(-I*Δ*t/2)*(ψi0*Cos[Ω*t/2]+I*(Δ/Ω*ψi0-Tij/(ħ*Ω)*ψj0)
                  *Sin[Ω*t/2]);
In[326]:= ψj[t_] = E^(I*Δ*t/2)*(ψj0*Cos[Ω*t/2]-I*(Δ/Ω*ψj0+Tji/(ħ*Ω)*ψi0)
                  *Sin[Ω*t/2]);
```

With these definitions, we can check the Schrödinger equations (3.16):

```
In[327]:= FullSimplify[I*ħ*ψi'[t] == (1/2) * ψj[t] * Exp[-I*Δ*t]*Tij]
Out[327]= True
In[328]:= FullSimplify[I*ħ*ψj'[t] == (1/2) * ψi[t] * Exp[I*Δ*t]*Tji]
Out[328]= True
```

as well as the initial conditions

```
In[329]:= ψi[0]
Out[329]= ψi0
In[330]:= ψj[0]
Out[330]= ψj0
```

dressed states: If we insert the RWA solutions, Equation (3.17), into the definition of the general hyperfine state, Equation (3.13), and set all coefficients $\psi_k = 0$ for $k \notin \{i, j\}$, and then write $\sin(z) = (e^{iz} - e^{-iz})/(2i)$ and $\cos(z) = (e^{iz} + e^{-iz})/2$, we find the state

$$
\begin{aligned}
|\psi(t)\rangle &\approx \psi_i(t) e^{-iE_i t/\hbar}|i\rangle + \psi_j(t) e^{-iE_j t/\hbar}|j\rangle \\
&= \frac{1}{2} e^{-i\left(E_i - \frac{\hbar(\Omega-\Delta)}{2}\right)t/\hbar} \left\{ \left[\psi_i(0)\left(1+\frac{\Delta}{\Omega}\right) - \psi_j(0)\frac{T_{ij}}{\hbar\Omega}\right]|i\rangle + \left[\psi_j(0)\left(1-\frac{\Delta}{\Omega}\right) - \psi_i(0)\frac{T_{ij}^*}{\hbar\Omega}\right] e^{i\omega t}|j\rangle \right\} \\
&\quad + \frac{1}{2} e^{-i\left(E_i + \frac{\hbar(\Omega+\Delta)}{2}\right)t/\hbar} \left\{ \left[\psi_i(0)\left(1-\frac{\Delta}{\Omega}\right) + \psi_j(0)\frac{T_{ij}}{\hbar\Omega}\right]|i\rangle + \left[\psi_j(0)\left(1+\frac{\Delta}{\Omega}\right) + \psi_i(0)\frac{T_{ij}^*}{\hbar\Omega}\right] e^{i\omega t}|j\rangle \right\}. \qquad (3.18)
\end{aligned}
$$

In order to interpret this state more clearly, we need to expand our view of the problem to include the quantized driving field. For this we assume that the driving mode of the field (for example, the used mode of the electromagnetic field) in state $|n\rangle$ contains n quanta of vibration (for example, photons), and has an energy of $E_n = n\hbar\omega$. The two states $|i\rangle$ and $|j\rangle$ describing our system, with $E_i - E_j \approx \hbar\omega$, actually correspond to states in the larger system containing the driving field. In this sense, we can say that the state $|i, n\rangle$, with the system in state $|i\rangle$ and the driving field containing n quanta, is approximately resonant with the state $|j, n+1\rangle$, with the system in state $|j\rangle$ and the driving field containing $n+1$ quanta. A transition from $|i\rangle$ to $|j\rangle$ is actually a transition from $|i, n\rangle$ to $|j, n+1\rangle$, where one quantum is added simultaneously to the driving field in order to conserve energy (approximately). A transition from $|j\rangle$ to $|i\rangle$ corresponds to the system absorbing one quantum from the driving field.

The energy of the quantized driving field contributes an additional time dependence

$$
|i\rangle \mapsto |i, n\rangle e^{-in\omega t}, \qquad |j\rangle \mapsto |j, n+1\rangle e^{-i(n+1)\omega t}, \qquad (3.19)
$$

and Equation (3.18) thus becomes

$$
\begin{aligned}
&|\psi(t)\rangle \approx \\
&\frac{1}{2}e^{-\mathrm{i}\left(E_i+n\hbar\omega+\frac{\hbar(\Delta-\Omega)}{2}\right)t/\hbar}\left\{\left[\psi_i(0)\left(1+\frac{\Delta}{\Omega}\right)-\psi_j(0)\frac{T_{ij}}{\hbar\Omega}\right]|i,n\rangle+\left[\psi_j(0)\left(1-\frac{\Delta}{\Omega}\right)-\psi_i(0)\frac{T_{ij}^*}{\hbar\Omega}\right]|j,n+1\rangle\right\} \\
&+\frac{1}{2}e^{-\mathrm{i}\left(E_i+n\hbar\omega+\frac{\hbar(\Delta+\Omega)}{2}\right)t/\hbar}\left\{\left[\psi_i(0)\left(1-\frac{\Delta}{\Omega}\right)+\psi_j(0)\frac{T_{ij}}{\hbar\Omega}\right]|i,n\rangle+\left[\psi_j(0)\left(1+\frac{\Delta}{\Omega}\right)+\psi_i(0)\frac{T_{ij}^*}{\hbar\Omega}\right]|j,n+1\rangle\right\}\Bigg) \\
&\qquad\qquad = \frac{1}{2}e^{-\mathrm{i}E_-t/\hbar}|-\rangle+\frac{1}{2}e^{-\mathrm{i}E_+t/\hbar}|+\rangle \qquad (3.20)
\end{aligned}
$$

With this substitution, the state consists of two components, called *dressed states*,

$$
|\pm\rangle = \left[\psi_i(0)\left(1\mp\frac{\Delta}{\Omega}\right)\pm\psi_j(0)\frac{T_{ij}}{\hbar\Omega}\right]|i,n\rangle+\left[\psi_j(0)\left(1\pm\frac{\Delta}{\Omega}\right)\pm\psi_i(0)\frac{T_{ij}^*}{\hbar\Omega}\right]|j,n+1\rangle. \qquad (3.21)
$$

that are time-invariant apart from their energy (phase) prefactors. These energy prefactors correspond to the effective energy of the dressed states in the presence of the oscillating field,[7]

$$
E_\pm = E_i+n\hbar\omega+\frac{\hbar(\Delta\pm\Omega)}{2} = E_j+(n+1)\hbar\omega+\frac{\hbar(-\Delta\pm\Omega)}{2}. \qquad (3.22)
$$

We look at these dressed states in two limits:

- On resonance ($\Delta = 0$), we have $\hbar\Omega = |T_{ij}|$, and the dressed states of Equation (3.21) become

$$
\begin{aligned}
|\pm\rangle &= \left[\psi_i(0)\pm\psi_j(0)\frac{T_{ij}}{|T_{ij}|}\right]|i,n\rangle+\left[\psi_j(0)\pm\psi_i(0)\frac{T_{ij}^*}{|T_{ij}|}\right]|j,n+1\rangle \\
&= \left[\psi_i(0)\pm\psi_j(0)\frac{T_{ij}}{|T_{ij}|}\right]\left(|i,n\rangle\pm\frac{T_{ij}^*}{|T_{ij}|}|j,n+1\rangle\right), \qquad (3.23)
\end{aligned}
$$

  which are equal mixtures of the original states $|i,n\rangle$ and $|j,n+1\rangle$. They have energies

$$
E_\pm = E_i+n\hbar\omega\pm\frac{1}{2}|T_{ij}| = E_j+(n+1)\hbar\omega\pm\frac{1}{2}|T_{ij}| \qquad (3.24)
$$

  in the presence of a resonant ac coupling field: the degeneracy of the levels $|i,n\rangle$ and $|j,n+1\rangle$ is lifted, and the dressed states are split by $E_+-E_- = |T_{ij}|$.

- Far off-resonance ($\Delta\to\pm\infty$) we have $\Omega\approx|\Delta|+\frac{|T_{ij}|^2}{2\hbar^2|\Delta|}$, and Equation (3.20) becomes

$$
|\psi(t)\rangle \approx e^{-\mathrm{i}\left(E_i+n\hbar\omega-\frac{|T_{ij}|^2}{4\hbar\Delta}\right)t/\hbar}\psi_i(0)|i,n\rangle+e^{-\mathrm{i}\left(E_j+(n+1)\hbar\omega+\frac{|T_{ij}|^2}{4\Delta}\right)t/\hbar}\psi_j(0)|j,n+1\rangle. \qquad (3.25)
$$

  (Hint: to verify this, look at the cases $\Delta\to+\infty$ and $\Delta\to-\infty$ separately). The energy levels $|i,n\rangle$ and $|j,n+1\rangle$ are thus shifted by $\mp\frac{|T_{ij}|^2}{4\hbar\Delta}$, respectively, and there is no population transfer between the levels. That is, the dressed states become equal to the original states. Remember that we had assumed $E_i > E_j$:

  - For a blue-detuned drive ($\Delta\to+\infty$), the upper level $|i\rangle$ is *lowered* in energy by $\Delta E = \frac{|T_{ij}|^2}{4\hbar\Delta}$ while the lower level $|j\rangle$ is *raised* in energy by ΔE.
  - For a red-detuned drive ($\Delta\to-\infty$), the upper level $|i\rangle$ is *raised* in energy by $\Delta E = \frac{|T_{ij}|^2}{4\hbar|\Delta|}$ while the lower level $|j\rangle$ is *lowered* in energy by ΔE.

  These shifts are called *ac Zeeman shifts* in this case, or *level shifts* more generally. When the oscillating field is a light field, level shifts are often called *light shifts* or *ac Stark shifts*.

[7] The instantaneous energy of a state is defined as $E = \langle\hat{H}\rangle = \mathrm{i}\hbar\langle\frac{\partial}{\partial t}\rangle$. For a state $|\psi(t)\rangle = e^{-\mathrm{i}\omega t}|\phi\rangle$ the energy is $E = \mathrm{i}\hbar\langle\psi(t)|\frac{\partial}{\partial t}|\psi(t)\rangle = \mathrm{i}\hbar\langle\phi|e^{\mathrm{i}\omega t}\frac{\partial}{\partial t}e^{-\mathrm{i}\omega t}|\phi\rangle = \hbar\omega$.

3.3.4 exercises

Q3.6 Take two angular momenta, for example $I = 3$ and $J = 5$, and calculate the eigenvalues of the operators $\hat{I}^2$, $\hat{I}_z$, $\hat{J}^2$, $\hat{J}_z$, $\hat{F}^2$, and $\hat{F}_z$, where $\hat{\vec{F}} = \hat{\vec{I}} + \hat{\vec{J}}$.

Q3.7 In **Q3.6** you have coupled two angular momenta but you have not used any Clebsch–Gordan coefficients. Why not? Where do these coefficients appear?

Q3.8 For a spin of a certain length, for example $S = 100$, take the state $|S, S\rangle$ (a spin pointing in the $+z$ direction) and calculate the expectation values $\langle \hat{S}_x \rangle$, $\langle \hat{S}_y \rangle$, $\langle \hat{S}_z \rangle$, $\langle \hat{S}_x^2 \rangle - \langle \hat{S}_x \rangle^2$, $\langle \hat{S}_y^2 \rangle - \langle \hat{S}_y \rangle^2$, $\langle \hat{S}_z^2 \rangle - \langle \hat{S}_z \rangle^2$. *Hint:* the expectation value of an operator $\hat{A}$ is $\langle S, S|\hat{A}|S, S\rangle$.

Q3.9 Use `In[323]` and `In[324]` to calculate the detuning Δ and the generalized Rabi frequency Ω for the ^{87}Rb solution of `In[320]`, where the population oscillates between the levels $i = 2$ and $j = 7$. What is the oscillation period corresponding to Ω? Does it match the plots of `In[321]` and `In[322]`?

Q3.10 Do the presented alkali atom calculation for ^{23}Na: are there any magic field values?
http://steck.us/alkalidata/sodiumnumbers.pdf

Q3.11 Do the presented alkali atom calculation for ^{85}Rb: are there any magic field values?
http://steck.us/alkalidata/rubidium85numbers.pdf

Q3.12 Do the presented alkali atom calculation for ^{133}Cs: are there any magic field values?
http://steck.us/alkalidata/cesiumnumbers.pdf

Q3.13 Set $\vec{B} = 0$ and $\vec{B}^{\text{ac}} = B(\vec{e}_x + i\vec{e}_y)$ in the expression for `T` in `In[318]`. Which transitions are allowed for such circularly-polarized light around the quantization axis? *Hint:* use Equation (3.9) to identify the states.

Q3.14 Set $\vec{B} = 0$ and $\vec{B}^{\text{ac}} = B\vec{e}_z$ in the expression for `T` in `In[318]`. Which transitions are allowed for such linearly-polarized light along the quantization axis? *Hint:* use Equation (3.9) to identify the states.

3.4 coupled spin systems: Ising model in a transverse field [code]

We now turn to larger numbers of coupled quantum-mechanical spins. A large class of such coupled spin systems can be described with Hamiltonians of the form

$$\hat{\mathcal{H}} = \sum_{k=1}^{N} \hat{\mathcal{H}}^{(k)} + \sum_{k=1}^{N-1} \sum_{k'=k+1}^{N} \hat{\mathcal{H}}_{\text{int}}^{(k,k')}, \tag{3.26}$$

where the $\hat{\mathcal{H}}^{(k)}$ are single-spin Hamiltonians (for example couplings to a magnetic field) and the $\hat{\mathcal{H}}_{\text{int}}^{(k,k')}$ are coupling Hamiltonians between two spins. Direct couplings between three or more spins can usually be neglected.

As an example we study the dimensionless "transverse Ising" Hamiltonian

$$\hat{\mathcal{H}} = -\frac{b}{2} \sum_{k=1}^{N} \hat{S}_x^{(k)} - \sum_{k=1}^{N} \hat{S}_z^{(k)} \hat{S}_z^{(k+1)} \tag{3.27}$$

acting on a ring of N spin-S systems where the $(N+1)$st spin is identified with the first spin. We can read off three limits from this Hamiltonian:

- For $b \to \pm\infty$ the spin–spin coupling Hamiltonian can be neglected, and the ground state will have all spins aligned with the $\pm x$ direction,

$$|\psi_{+\infty}\rangle = |+x\rangle^{\otimes N}, \qquad |\psi_{-\infty}\rangle = |-x\rangle^{\otimes N}. \tag{3.28}$$

  The system is therefore in a product state for $b \to \pm\infty$, which means that there is no entanglement between spins. In the basis of $|S, M\rangle$ Dicke states, Equation (3.1) and Equation (3.2), the single-spin

states making up these product states are

$$|+x\rangle = 2^{-S} \sum_{M=-S}^{S} \sqrt{\binom{2S}{M+S}}\, |S, M\rangle, \tag{3.29a}$$

$$|-x\rangle = 2^{-S} \sum_{M=-S}^{S} (-1)^{M+S} \sqrt{\binom{2S}{M+S}}\, |S, M\rangle, \tag{3.29b}$$

which are aligned with the x-axis in the sense that $\hat{S}_x|+x\rangle = S\,|+x\rangle$ and $\hat{S}_x|-x\rangle = -S\,|-x\rangle$.

- For $b = 0$ the Hamiltonian contains only nearest-neighbor ferromagnetic spin–spin couplings $-\hat{S}_z^{(k)}\hat{S}_z^{(k+1)}$. We know that this Hamiltonian has two degenerate ground states: all spins pointing up or all spins pointing down,

$$|\psi_{0\uparrow}\rangle = |+z\rangle^{\otimes N}, \qquad |\psi_{0\downarrow}\rangle = |-z\rangle^{\otimes N}, \tag{3.30}$$

where in the Dicke-state representation of Equation (3.1) we have $|+z\rangle = |S, +S\rangle$ and $|-z\rangle = |S, -S\rangle$. While these two states are product states, for $|b| \ll 1$ the perturbing Hamiltonian $-\frac{b}{2}\sum_{k=1}^{N}\hat{S}_x^{(k)}$ is diagonal in the states $\frac{|\psi_{0\uparrow}\rangle \pm |\psi_{0\downarrow}\rangle}{\sqrt{2}}$, which are not product states. The exact ground state for $0 < b \ll 1$ is close to $\frac{|\psi_{0\uparrow}\rangle + |\psi_{0\downarrow}\rangle}{\sqrt{2}}$, and for $-1 \ll b < 0$ it is close to $\frac{|\psi_{0\uparrow}\rangle - |\psi_{0\downarrow}\rangle}{\sqrt{2}}$. These are both maximally entangled states ("Schrödinger cat states").

Now we calculate the ground state $|\psi_b\rangle$ as a function of the parameter b, and compare the results to the above asymptotic limits.

3.4.1 basis set

The natural basis set for describing a set of N coupled spins is the tensor-product basis (see section 2.4.2). In this basis, the spin operators $\hat{S}_{x,y,z}^{(k)}$ acting only on spin k are defined as having a trivial action on all other spins, for example

$$\hat{S}_x^{(k)} \mapsto \underbrace{\mathbb{1}\otimes\mathbb{1}\otimes\cdots\otimes\mathbb{1}}_{(k-1)} \otimes \hat{S}_x \otimes \underbrace{\mathbb{1}\otimes\cdots\otimes\mathbb{1}}_{(N-k)}. \tag{3.31}$$

In Mathematica such single-spin-S operators acting on spin k out of a set of N spins are defined as follows. First we define the operator acting as $\hat{a} =$ `a` on the `k`$^{\text{th}}$ spin out of a set of `n` spins, and trivially on all others:

```
In[331]:= op[S_?SpinQ, n_Integer, k_Integer, a_?MatrixQ] /;
            1<=k<=n && Dimensions[a] == {2S+1,2S+1} :=
            KroneckerProduct[IdentityMatrix[(2S+1)^(k-1), SparseArray],
                             a,
                             IdentityMatrix[(2S+1)^(n-k), SparseArray]]
```

Next, we specialize this to $\hat{a} = \hat{S}_x$, $\hat{S}_y$, $\hat{S}_z$:

```
In[332]:= sx[S_?SpinQ, n_Integer, k_Integer] /; 1<=k<=n := op[S, n, k, sx[S]]
In[333]:= sy[S_?SpinQ, n_Integer, k_Integer] /; 1<=k<=n := op[S, n, k, sy[S]]
In[334]:= sz[S_?SpinQ, n_Integer, k_Integer] /; 1<=k<=n := op[S, n, k, sz[S]]
```

Notice that we have used `n` $= N$ because the symbol `N` is already used internally in Mathematica.

From these we assemble the Hamiltonian:

```
In[335]:= H[S_?SpinQ, n_Integer/;n>=3, b_] := -b/2*Sum[sx[S, n, k], {k, n}] -
            Sum[sz[S, n, k].sz[S, n, Mod[k+1,n,1]], {k, n}]
```

The modulus `Mod[k+1,n,1]` represents the periodicity of the spin ring and ensures that the index remains within $1 \ldots N$ (*i.e.*, a modulus with offset 1).

3.4.2 asymptotic ground states

The asymptotic ground states for $b = 0$ and $b \to \pm\infty$ mentioned above are all product states of the form $|\psi\rangle = |\theta\rangle^{\otimes N}$ where $|\theta\rangle$ is the state of a single spin. We form an N-particle tensor product state of such single-spin states with

```
In[336]:= productstate[θ_?VectorQ, 1] = θ;
In[337]:= productstate[θ_?VectorQ, n_Integer/;n>=2] :=
            Flatten[KroneckerProduct @@ Table[θ, n]]
```

in accordance with In[236]; notice that the case $N = 1$ requires special attention.

The particular single-spin states $|+x\rangle$, $|-x\rangle$, $|+z\rangle$, $|-z\rangle$ we will be using are

```
In[338]:= xup[S_?SpinQ] := 2^(-S)*Table[Sqrt[Binomial[2S,M+S]],{M,S,-S,-1}]
In[339]:= xdn[S_?SpinQ] := 2^(-S)*Table[(-1)^(M+S)*Sqrt[Binomial[2S,M+S]], {M,S,-S,-1}]
In[340]:= zup[S_?SpinQ] := SparseArray[1 -> 1, 2S+1]
In[341]:= zdn[S_?SpinQ] := SparseArray[-1 -> 1, 2S+1]
```

We can check that these are correct with

```
In[342]:= Table[sx[S].xup[S] == S*xup[S], {S, 0, 4, 1/2}]
Out[342]= {True, True, True, True, True, True, True, True, True}
In[343]:= Table[sx[S].xdn[S] == -S*xdn[S], {S, 0, 4, 1/2}]
Out[343]= {True, True, True, True, True, True, True, True, True}
In[344]:= Table[sz[S].zup[S] == S*zup[S], {S, 0, 4, 1/2}]
Out[344]= {True, True, True, True, True, True, True, True, True}
In[345]:= Table[sz[S].zdn[S] == -S*zdn[S], {S, 0, 4, 1/2}]
Out[345]= {True, True, True, True, True, True, True, True, True}
```

From these we construct the product states

```
In[346]:= allxup[S_?SpinQ,n_Integer/;n>=1] := productstate[xup[S],n]
In[347]:= allxdn[S_?SpinQ,n_Integer/;n>=1] := productstate[xdn[S],n]
In[348]:= allzup[S_?SpinQ,n_Integer/;n>=1] := productstate[zup[S],n]
In[349]:= allzdn[S_?SpinQ,n_Integer/;n>=1] := productstate[zdn[S],n]
```

3.4.3 Hamiltonian diagonalization

We find the m lowest-energy eigenstates of this Hamiltonian with the procedures described in section 1.10.4: for example, with $S = 1/2$ and $N = 20$,[8]

```
In[350]:= With[{S = 1/2, n = 20},
          (* Hamiltonian *)
          h[b_] = H[S, n, b];
          (* two degenerate ground states for b=0 *)
          gs0up = allzup[S, n];
          gs0dn = allzdn[S, n];
          (* ground state for b=+Infinity *)
          gsplusinf = allxup[S, n];
          (* ground state for b=-Infinity *)
          gsminusinf = allxdn[S, n];
          (* numerically calculate lowest m eigenstates *)
          Clear[gs];
          gs[b_?NumericQ, m_Integer /; m>=1] := gs[b, m] = -Eigensystem[-h[N[b]], m,
```

[8] The attached Mathematica code uses $N = 14$ instead, since calculations with $N = 20$ take a long time.

```
14            Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}] //
15            Transpose //Sort //Transpose;
16        ]
```

Comments:

- `gs0up` $= |\psi_{0\uparrow}\rangle$ and `gs0dn` $= |\psi_{0\downarrow}\rangle$ are the exact degenerate ground states for $b = 0$; `gsplusinf` $= |\psi_{+\infty}\rangle$ and `gsminusinf` $= |\psi_{-\infty}\rangle$ are the exact nondegenerate ground states for $b = \pm\infty$.

- The function `gs`, which calculates the `m` lowest-lying eigenstates of the Hamiltonian, remembers its calculated values (see section 1.6.3): this is important here because such eigenstate calculations can take a long time when `n` is large.

- The function `gs` numerically calculates the eigenvalues using `h[N[b]]` as a Hamiltonian, which ensures that the Hamiltonian contains floating-point machine-precision numbers instead of exact numbers in case `b` is given as an exact number. Calculating the eigenvalues and eigenvectors of a matrix of exact numbers takes extremely long (please try: on line 13 of `In[350]` replace `-Eigensystem[-h[N[b]], ...` with `-Eigensystem[-h[b], ...` and compare the run time of `gs[1, 2]` with that of `gs[1.0, 2]`.).

- The operations `//Transpose //Sort //Transpose` on line 15 of `In[350]` ensure that the eigenvalues (and associated eigenvectors) are sorted in ascending energy order (see `In[180]`).

- When the ground state is degenerate, which happens here for $b \approx 0$, the Arnoldi algorithm has some difficulty finding the correct degeneracy. This means that `gs[0,2]` may return two non-degenerate eigenstates instead of the (correct) two degenerate ground states. This is a well-known problem that can be circumvented by calculating more eigenstates.

- A problem involving N spin-S systems leads to matrices of size $(2S+1)^N \times (2S+1)^N$. This scaling quickly becomes very problematic (even if we use sparse matrices) and is at the center of why quantum mechanics is difficult. Imagine a system composed of $N = 1000$ spins $S = 1/2$: its state vector is a list of $2^{1000} = 1.07 \times 10^{301}$ complex numbers! Comparing this to the fact that there are only about 10^{80} particles in the universe, we conclude that such a state vector could never be written down and therefore the Hilbert space method of quantum mechanics we are using here is fundamentally flawed. But as this is an introductory course, we will stick to this classical matrix-mechanics formalism and let the computer bear the weight of its complexity. Keep in mind, though, that this is not a viable strategy for large systems, as each doubling of computer capacity only allows us to add a single spin to the system, which, using Moore's law, allows us to add one spin every two years.[9]

  There are alternative formulations of quantum mechanics, notably the path-integral formalism, which partly circumvent this problem; but the computational difficulty is not eliminated, it is merely shifted. Modern developments such as tensor networks[10] try to limit the accessible Hilbert space by restricting calculations to a subspace where the entanglement between particles is bounded. This makes sense since almost all states of the huge Hilbert space are so complex and carry such complicated quantum-mechanical entanglement that (i) they would be extremely difficult to generate with realistic Hamiltonians, and (ii) they would decohere within very short time.

3.4.4 analysis of the ground state

energy gap

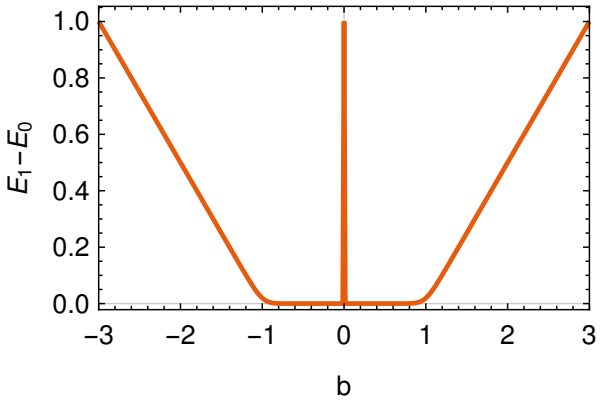
Much of the behavior of our Ising spin chain can be seen in a plot of the *energy gap*, which is the energy difference between the ground state and the first excited state. With `m` $= 2$ we calculate the two lowest-lying energy levels and plot their energy difference as a function of the parameter b:

[9] Moore's law is the observation that over the history of computing hardware, the number of transistors on integrated circuits doubles approximately every two years. From https://en.wikipedia.org/wiki/Moore's_law.

[10] Matrix product states and tensor networks: https://en.wikipedia.org/wiki/Matrix_product_state.

```
In[351]:= With[{bmax = 3, db = 1/64, m = 2},
          ListLinePlot[Table[{b, gs[b,m][[1,2]]-gs[b,m][[1,1]]},
            {b, -bmax, bmax, db}]]]
```

Notice how the fact that the `gs` function remembers its own results speeds up this calculation by a factor of 2 (see section 1.6.3).



Even in this small 20-spin simulation we can see that this gap is approximately

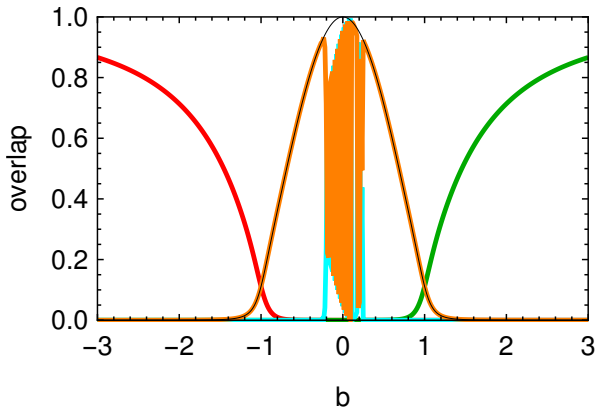
$$E_1 - E_0 \approx \begin{cases} 0 & \text{if } |b| < 1, \\ \frac{|b|-1}{2} & \text{if } |b| > 1. \end{cases} \tag{3.32}$$

This observation of a qualitative change in the excitation gap suggests that at $b = \pm 1$ the system undergoes a *quantum phase transition* (*i.e.*, a phase transition induced by quantum fluctuations instead of thermal fluctuations). We note that the gap of Equation (3.32) is independent of the particle number N and is therefore a *global* property of the Ising spin ring, not a property of each individual spin (in which case it would scale with N).

overlap with asymptotic states

Once a ground state $|\psi_b\rangle$ has been calculated, we compute its overlap with the asymptotically known states using scalar products. Notice that for $b = 0$ we calculate the scalar products with the states $\frac{|\psi_{0\uparrow}\rangle \pm |\psi_{0\downarrow}\rangle}{\sqrt{2}}$ as they are the approximate ground states for $|b| \ll 1$.

```
In[352]:= With[{bmax = 3, db = 1/64, m = 2},
          ListLinePlot[
            Table[{{b, Abs[gsminusinf.gs[b,m][[2,1]]]^2},
             {b, Abs[gsplusinf.gs[b, m][[2,1]]]^2},
             {b, Abs[((gs0up-gs0dn)/Sqrt[2]).gs[b,m][[2,1]]]^2},
             {b, Abs[((gs0up+gs0dn)/Sqrt[2]).gs[b,m][[2,1]]]^2},
             {b, Abs[((gs0up-gs0dn)/Sqrt[2]).gs[b,m][[2,1]]]^2 +
                 Abs[((gs0up+gs0dn)/Sqrt[2]).gs[b,m][[2,1]]]^2}},
              {b, -bmax, bmax, db}] //Transpose]]
```



Observations:

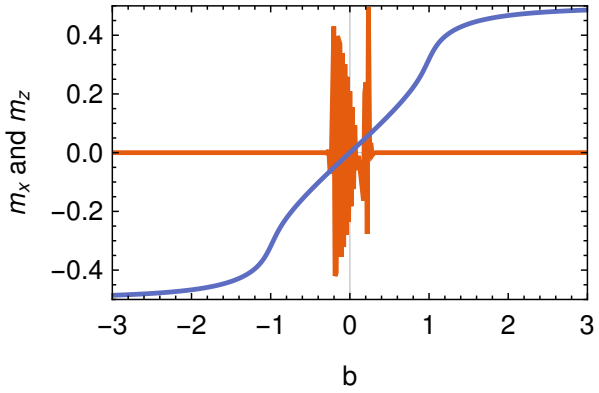
- The overlap $|\langle\psi_b|\psi_{-\infty}\rangle|^2$ (red) approaches 1 as $b \to -\infty$.
- The overlap $|\langle\psi_b|\psi_{+\infty}\rangle|^2$ (green) approaches 1 as $b \to +\infty$.
- The overlap $\left|\langle\psi_b|\frac{|\psi_{0\uparrow}\rangle-|\psi_{0\downarrow}\rangle}{\sqrt{2}}\right|^2$ (cyan) is mostly negligible.
- The overlap $\left|\langle\psi_b|\frac{|\psi_{0\uparrow}\rangle+|\psi_{0\downarrow}\rangle}{\sqrt{2}}\right|^2$ (orange) approaches 1 as $b \to 0$.
- The sum of these last two, $\left|\langle\psi_b|\frac{|\psi_{0\uparrow}\rangle-|\psi_{0\downarrow}\rangle}{\sqrt{2}}\right|^2 + \left|\langle\psi_b|\frac{|\psi_{0\uparrow}\rangle+|\psi_{0\downarrow}\rangle}{\sqrt{2}}\right|^2 = |\langle\psi_b|\psi_{0\uparrow}\rangle|^2 + |\langle\psi_b|\psi_{0\downarrow}\rangle|^2$ (thin black), approaches 1 as $b \to 0$ and is less prone to numerical noise.
- If you redo this calculation with an *odd* number of spins, you may find different overlaps with the $\frac{|\psi_{0\uparrow}\rangle\pm|\psi_{0\downarrow}\rangle}{\sqrt{2}}$ asymptotic states. Their sum, however, drawn in black, should be insensitive to the parity of N.
- For $|b| \lesssim 0.2$ the excitation gap (see above) is so small that the calculated ground-state eigenvector is no longer truly the ground state but becomes mixed with the first excited state due to numerical inaccuracies. This leads to the jumps in the orange and cyan curves (notice, however, that their sum, shown in black, is stable). If you redo this calculation with larger values for `m`, you may get better results.

magnetization

Studying the ground state coefficients list directly is of limited use because of the large amount of information contained in its numerical representation. We gain more insight by studying specific observables, for example the magnetizations $\langle\hat{S}_x^{(k)}\rangle$, $\langle\hat{S}_y^{(k)}\rangle$, and $\langle\hat{S}_z^{(k)}\rangle$. We add the following definition to the `With[]` clause in `In[350]`:

```
16      (* spin components expectation values *)
17      Clear[mx,my,mz];
18      mx[b_?NumericQ, m_Integer /; m >= 1, k_Integer] :=
19        mx[b, m, k] = With[{g = gs[b,m][[2,1]]},
20          Re[Conjugate[g].(sx[S, n, Mod[k, n, 1]].g)]];
21      my[b_?NumericQ, m_Integer /; m >= 1, k_Integer] :=
22        my[b, m, k] = With[{g = gs[b,m][[2,1]]},
23          Re[Conjugate[g].(sy[S, n, Mod[k, n, 1]].g)]];
24      mz[b_?NumericQ, m_Integer /; m >= 1, k_Integer] :=
25        mz[b, m, k] = With[{g = gs[b,m][[2,1]]},
26          Re[Conjugate[g].(sz[S, n, Mod[k, n, 1]].g)]];
27      ]
```

In our transverse Ising model only the x-component of the magnetization is nonzero. Due to the translational symmetry of the system we can look at the magnetization of any spin, for example the first one ($k = 1$): $m_x(b)$ (blue) and $m_z(b)$ (orange, non-zero due to numerical inaccuracies)



We see that in the phases of large $|b|$, the spins are almost entirely polarized, while in the phase $|b| < 1$ the x-magnetization is roughly proportional to b.

spin–spin fluctuation correlations

Quantum-mechanical spins always fluctuate around their mean direction. In the example of **Q3.8**, the state $|S, S\rangle$ points on average along the $+z$ direction in the sense that $\langle\hat{\vec{S}}\rangle = \langle S, S|\hat{\vec{S}}|S, S\rangle = \{0, 0, S\}$; but it fluctuates away from this axis as $\langle\hat{S}_x^2\rangle = \langle\hat{S}_y^2\rangle = S/2$.

By introducing the fluctuation operator $\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}} = \hat{\vec{S}} - \langle\hat{\vec{S}}\rangle$, we can interpret spin fluctuations through the expectation values $\langle\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}}\rangle = \{0, 0, 0\}$ (fluctuations always average to zero) and $\langle(\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}})^2\rangle = \langle\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}}\cdot\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}}\rangle = \langle\hat{\vec{S}}\cdot\hat{\vec{S}}\rangle - \langle\hat{\vec{S}}\rangle\cdot\langle\hat{\vec{S}}\rangle = S(S+1) - \|\langle\hat{\vec{S}}\rangle\|^2$. Since the spin magnetization has length $0 \le \|\langle\hat{\vec{S}}\rangle\| \le S$, these fluctuations satisfy $S \le \langle(\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}})^2\rangle \le S(S+1)$: they are positive for every spin state.

When two (or more) spins are present, their quantum-mechanical fluctuations can become correlated. We quantify such spin–spin fluctuation correlations between two spins k and k' with the measure

$$C_{k,k'} = \langle\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}}^{(k)}\cdot\hat{\boldsymbol{\delta}}\hat{\vec{\boldsymbol{S}}}^{(k')}\rangle = \langle\hat{\vec{S}}^{(k)}\cdot\hat{\vec{S}}^{(k')}\rangle - \langle\hat{\vec{S}}^{(k)}\rangle\cdot\langle\hat{\vec{S}}^{(k')}\rangle, \tag{3.33}$$

which has the form of a statistical covariance.[11] For any spin length S (assuming $S^{(k)} = S^{(k')}$), the first term of Equation (3.33) can be written as

$$\langle\hat{\vec{S}}^{(k)}\cdot\hat{\vec{S}}^{(k')}\rangle = \frac{\langle(\hat{\vec{S}}^{(k)} + \hat{\vec{S}}^{(k')})^2\rangle - \langle(\hat{\vec{S}}^{(k)})^2\rangle - \langle(\hat{\vec{S}}^{(k')})^2\rangle}{2} = \frac{1}{2}\langle(\hat{\vec{S}}^{(k)} + \hat{\vec{S}}^{(k')})^2\rangle - S(S+1), \tag{3.34}$$

which allows us to predict its expectation value as a function of the total-spin quantum number describing the two spins-S. As this quantum number can be anywhere between 0 and $2S$, we have $0 \le \langle(\hat{\vec{S}}^{(k)} + \hat{\vec{S}}^{(k')})^2\rangle \le 2S(2S+1)$. This expectation value is not restricted to integer values. As a result we make the following observations:

- $-S(S+1) \le C_{k,k'} \le S^2$: spin fluctuations can be correlated ($C_{k,k'} > 0$), anti-correlated ($C_{k,k'} < 0$), or uncorrelated ($C_{k,k'} = 0$).
- The strongest correlations $C_{k,k'} = S^2$ are found when the two spins-S form a joint spin-$2S$ and at the same time are unaligned ($\langle\hat{\vec{S}}^{(k)}\rangle\cdot\langle\hat{\vec{S}}^{(k')}\rangle = 0$).
- The strongest anti-correlations $C_{k,k'} = -S(S+1)$ are found when the two spins-S form a joint spin-0 (*i.e.*, a spin-singlet). In this case, the magnetizations always vanish: $\langle\hat{\vec{S}}^{(k)}\rangle = \langle\hat{\vec{S}}^{(k')}\rangle = \{0, 0, 0\}$.

[11] See https://en.wikipedia.org/wiki/Covariance.

For the specific case $S = 1/2$, which we use in the present calculations, two spins can form a joint singlet (total spin 0; $\langle(\hat{\vec{S}}^{(k)} + \hat{\vec{S}}^{(k')})^2\rangle = 0$), a joint triplet (total spin 1; $\langle(\hat{\vec{S}}^{(k)} + \hat{\vec{S}}^{(k')})^2\rangle = 2$), or a mixture of these ($0 \le \langle(\hat{\vec{S}}^{(k)} + \hat{\vec{S}}^{(k')})^2\rangle \le 2$), and the correlation is restricted to the values $-\frac{3}{4} \le C_{k,k'} \le +\frac{1}{4}$ for all states. Specific cases are:

- In the pure joint singlet state $\frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{\sqrt{2}}$ the correlation is precisely $C_{k,k'} = -\frac{3}{4}$. A fluctuation of one spin implies a counter-fluctuation of the other in order to keep them anti-aligned and in a spin-0 joint state. Remember that the spin monogamy theorem states that if spins k and k' form a joint singlet, then both must be uncorrelated with all other spins in the system.

- In a pure joint triplet state, *i.e.*, any mixture of the states $|\uparrow\uparrow\rangle$, $|\downarrow\downarrow\rangle$, and $\frac{|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle}{\sqrt{2}}$, the correlation is $0 \le C_{k,k'} \le +\frac{1}{4}$. A fluctuation of one spin implies a similar fluctuation of the other in order to keep them aligned and in a spin-1 joint state.

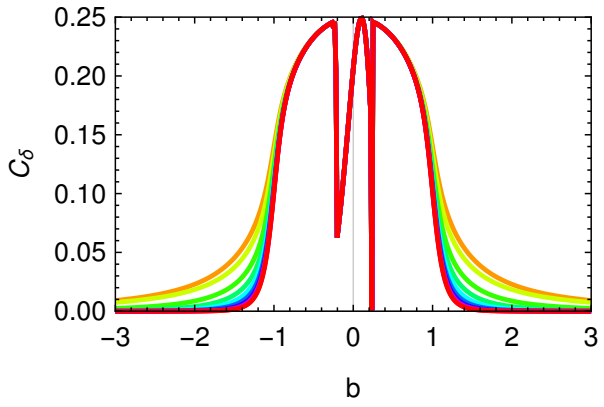
- The maximum correlation $C_{k,k'} = +\frac{1}{4}$ is reached for unaligned triplet states, *i.e.*, when $\langle\hat{\vec{S}}^{(k)}\rangle \cdot \langle\hat{\vec{S}}^{(k')}\rangle = 0$. Examples include the states $\frac{|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle}{\sqrt{2}}$, $\frac{|\uparrow\uparrow\rangle - |\downarrow\downarrow\rangle}{\sqrt{2}}$, and $\frac{|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle}{\sqrt{2}}$.

- In the fully parallel triplet states $|\uparrow\uparrow\rangle$ and $|\downarrow\downarrow\rangle$, the magnetizations are aligned but their fluctuations are uncorrelated: $C_{k,k'} = 0$, and hence $\langle\hat{\vec{S}}^{(k)} \cdot \hat{\vec{S}}^{(k')}\rangle = \langle\hat{\vec{S}}^{(k)}\rangle \cdot \langle\hat{\vec{S}}^{(k')}\rangle$.

In order to estimate these spin fluctuation correlations, we add the following definition to the `With[]` clause in `In[350]`:

```
      (* spin-spin correlation operator *)
      Clear[Cop];
      Cop[k1_Integer, k2_Integer] := Cop[k1, k2] =
        With[{q1 = Mod[k1,n,1], q2 = Mod[k2,n,1]},
          sx[S,n,q1].sx[S,n,q2] + sy[S,n,q1].sy[S,n,q2]
            + sz[S,n,q1].sz[S,n,q2]];
      (* spin-spin correlations *)
      Clear[c];
      c[b_?NumericQ,m_Integer/;m>=1,{k1_Integer,k2_Integer}] :=
        c[b,m,{k1,k2}] = With[{g = gs[b,m][[2,1]]},
          Re[Conjugate[g].(Cop[k1,k2].g)]-(mx[b,m,k1]*mx[b,m,k2]
            +my[b,m,k1]*my[b,m,k2]+mz[b,m,k1]*mz[b,m,k2])];
      ]
```

Since our spin ring is translationally invariant, we can simply plot $C_\delta = C_{1,1+\delta}$: for $N = 20$ and $\delta = 1 \ldots 10$ (top to bottom),



Observations:

- The spin fluctuations are maximally correlated ($C = +\frac{1}{4}$) for $b = 0$, in the ferromagnetic phase. They are all either pointing up or pointing down, so every spin is correlated with every other spin; keep in mind that the magnetization vanishes at the same time (page 58). It is only the spin–spin interactions that correlate the spins' directions and therefore their fluctuations.

- The spin fluctuations are uncorrelated ($C \to 0$) for $b \to \pm\infty$, in the paramagnetic phases. They are all pointing in the $+x$ direction for $b \gg 1$ or in the $-x$ direction for $b \ll -1$, but they are doing so in an independent way and would keep pointing in that direction even if the spin–spin interactions were switched off. This means that the fluctuations of the spins' directions are uncorrelated.

entropy of entanglement

We know now that in the limits $b \to \pm\infty$ the spins are polarized (magnetized) but their fluctuations are uncorrelated, while close to $b = 0$ they are unpolarized (unmagnetized) but their fluctuations are maximally correlated. Here we quantify these correlations with the *entropy of entanglement*, which measures the entanglement of a single spin with the rest of the spin chain.

In a system composed of two subsystems A and B, the entropy of entanglement is defined as the *von Neumann entropy* of the reduced density matrix (see section 2.4.3),

$$S_{\mathrm{AB}} = -\operatorname{Tr}\left(\hat{\rho}_{\mathrm{A}} \log_2 \hat{\rho}_{\mathrm{A}}\right) = -\sum_i \lambda_i \log_2 \lambda_i \tag{3.35}$$

where the λ_i are the eigenvalues of $\hat{\rho}_{\mathrm{A}}$ (or of $\hat{\rho}_{\mathrm{B}}$; the result is the same). Care must be taken with the case $\lambda_i = 0$: we find $\lim_{\lambda\to 0} \lambda \log_2 \lambda = 0$. For this we define the function

```
In[353]:= s[0|0.] = 0;
In[354]:= s[x_] = -x*Log[2, x];
```

that uses Mathematica's pattern matching to separate out the special case $x = 0$. Note that we use an alternative pattern[12] `0|0.` that matches both an analytic zero `0` and a numeric zero `0.`, which Mathematica distinguishes carefully.[13]

We define the entropy of entanglement of the first spin with the rest of the spin ring using the definition of `In[257]`, tracing out the last $(2S+1)^{N-1}$ degrees of freedom and leaving only the first $2S+1$ degrees of freedom of the first spin:

```
In[355]:= EE[S_?SpinQ, ψ_] :=
            Total[s /@ Re[Eigenvalues[traceout[ψ, -Length[ψ]/(2S+1)]]]]
```

Observations:

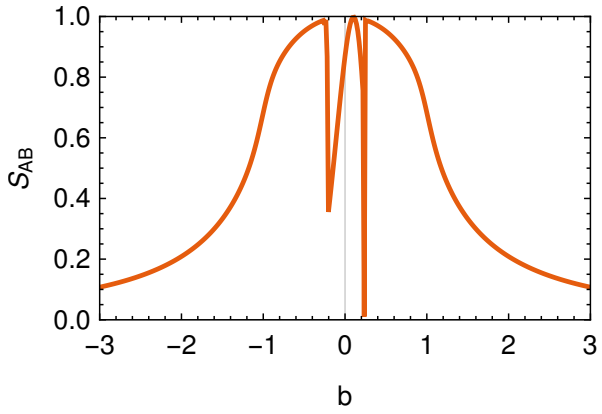
- Entanglement entropies of the known asymptotic ground states:

```
In[356]:= EE[1/2, (gs0up+gs0dn)/Sqrt[2]]
Out[356]= 1
In[357]:= EE[1/2, (gs0up-gs0dn)/Sqrt[2]]
Out[357]= 1
In[358]:= EE[1/2, gsplusinf]
Out[358]= 0
In[359]:= EE[1/2, gsminusinf]
Out[359]= 0
```

- Entanglement entropy as a function of b: again the calculation is numerically difficult around $b \approx 0$ because of the quasi-degeneracy.

[12] See https://reference.wolfram.com/language/tutorial/PatternsInvolvingAlternatives.html.
[13] Experiment: `0==0.` yields `True` (testing for semantic identity), whereas `0===0.` yields `False` (testing for symbolic identity).

```
In[360]:=With[{bmax = 3, db = 1/64, m = 2},
         ListLinePlot[Table[{b, EE[1/2, gs[b,m][[2,1]]]},
           {b, -bmax, bmax, db}], PlotRange -> {0, 1}]]
```



Notice that the quantum phase transitions at $b = \pm 1$ are not visible in this plot.

3.4.5 exercises

Q3.15 For $S = 1/2$, what is the largest value of N for which you can calculate the ground state of the transverse Ising model at the critical point $b = 1$?

Q3.16 Study the transverse Ising model with $S = 1$:

1. At which values of b do you find quantum phase transitions?
2. Characterize the ground state in terms of magnetization, spin–spin correlations, and entanglement entropy.

Q3.17 Study the transverse XY model for $S = 1/2$:

$$\hat{\mathcal{H}} = -\frac{b}{2}\sum_{k=1}^{N}\hat{S}_z^{(k)} - \sum_{k=1}^{N}\left(\hat{S}_x^{(k)}\hat{S}_x^{(k+1)} + \hat{S}_y^{(k)}\hat{S}_y^{(k+1)}\right) \tag{3.36}$$

1. Guess the shape of the ground states for $b \pm \infty$ [notice that the first term in the Hamiltonian of Equation (3.36) is in the z-direction!] and compare to the numerical calculations.
2. At which values of b do you find quantum phase transitions?
3. Characterize the ground state in terms of magnetization, spin–spin correlations, and entanglement entropy.

Q3.18 Study the Heisenberg model for $S = 1/2$:

$$\hat{\mathcal{H}} = -\frac{b}{2}\sum_{k=1}^{N}\hat{S}_z^{(k)} - \sum_{k=1}^{N}\hat{\vec{S}}^{(k)}\cdot\hat{\vec{S}}^{(k+1)} \tag{3.37}$$

1. Guess the shape of the ground states for $b \pm \infty$ [notice that the first term in the Hamiltonian of Equation (3.37) is in the z-direction!] and compare to the numerical calculations.
2. What is the ground-state degeneracy for $b = 0$?
3. At which values of b do you find quantum phase transitions?
4. Characterize the ground state in terms of magnetization, spin–spin correlations, and entanglement entropy.

Q3.19 Consider two spin-1/2 particles in the triplet state $|\psi\rangle = |\uparrow\uparrow\rangle$. Subsystem A is the first spin, and subsystem B is the second spin.

1. What is the density matrix $\hat{\rho}_{AB}$ of this system?
2. What is the reduced density matrix $\hat{\rho}_A$ of subsystem A (the first spin)? Is this a pure state? If yes, what state?
3. What is the reduced density matrix $\hat{\rho}_B$ of subsystem B (the second spin)? Is this a pure state? If yes, what state?
4. Calculate the von Neumann entropies of $\hat{\rho}_{AB}$, $\hat{\rho}_A$, and $\hat{\rho}_B$.

Q3.20 Consider two spin-1/2 particles in the singlet state $|\psi\rangle = \frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{\sqrt{2}}$. Subsystem A is the first spin, and subsystem B is the second spin.

1. What is the density matrix $\hat{\rho}_{AB}$ of this system?
2. What is the reduced density matrix $\hat{\rho}_A$ of subsystem A (the first spin)? Is this a pure state? If yes, what state?
3. What is the reduced density matrix $\hat{\rho}_B$ of subsystem B (the second spin)? Is this a pure state? If yes, what state?
4. Calculate the von Neumann entropies of $\hat{\rho}_{AB}$, $\hat{\rho}_A$, and $\hat{\rho}_B$.

3.5 coupled spin systems: quantum circuits [code]

The computational structure developed so far in this chapter can be used to simulate quantum circuits, such as they are used to run quantum algorithms leading all the way to quantum computers. In its simplest form, a quantum circuit contains a set of N spin-1/2 quantum objects called *qubits*, on which a sequence of operations called *quantum gates* is executed. In analogy to classical binary logic, the basis states of the qubits' Hilbert space are usually denoted as $|0\rangle$ (replacing the spin-1/2 state $|\uparrow\rangle$) and $|1\rangle$ (replacing $|\downarrow\rangle$).

In this section, we go through the steps of assembling quantum circuits and simulating their behavior on a classical computer. Naturally, the matrix representation of quantum gates and circuits constructed here is neither efficient nor desirable for building an actual quantum computer. It is merely useful for acquiring a detailed understanding of the workings of quantum circuits and algorithms.

In what follows, we adhere strictly to Chapter 5 of Nielsen&Chuang,[14] which provides many more details of the calculations, as well as further reading for the interested student.

3.5.1 quantum gates

Any quantum circuit can be constructed from a set of simple building blocks, similarly to a classical digital circuit. These building blocks are canonical quantum gates,[15] of which we implement a useful subset here.

single-qubit gates

Single-qubit gates act on one specific qubit in a set:

- The Pauli-X gate —[X]— acts like $\hat{\sigma}_x = |1\rangle\langle 0| + |0\rangle\langle 1|$ on the desired qubit, and has no effect on all other qubits. A single-qubit input state $|\psi_{\text{in}}\rangle$ entering the gate from the left is transformed into the output state $|\psi_{\text{out}}\rangle = \hat{\sigma}_x|\psi_{\text{in}}\rangle$ exiting the gate towards the right.
- The Pauli-Y gate —[Y]— acts like $\hat{\sigma}_y = \mathrm{i}|1\rangle\langle 0| - \mathrm{i}|0\rangle\langle 1|$ on the desired qubit, and has no effect on all other qubits.
- The Pauli-Z gate —[Z]— acts like $\hat{\sigma}_z = |0\rangle\langle 0| - |1\rangle\langle 1|$ on the desired qubit, and has no effect on all other qubits.

[14] Michael A. Nielsen and Isaac L. Chuang: *Quantum Computation and Quantum Information*, 10th Anniversary Edition, Cambridge University Press, Cambridge, UK (2010).

[15] See https://en.wikipedia.org/wiki/Quantum_logic_gate.

- The Hadamard gate —[H]— acts like $\frac{\hat{\sigma}_x+\hat{\sigma}_z}{\sqrt{2}} = \frac{|0\rangle\langle 0|+|0\rangle\langle 1|+|1\rangle\langle 0|-|1\rangle\langle 1|}{\sqrt{2}}$ on the desired qubit, and has no effect on all other qubits.

To implement these single-qubit gates in a general way, we proceed as in In[331] by defining a matrix that represents the operator $\hat{a}$ acting on the k^{th} qubit in a set of n qubits:

```
In[361]:= op[n_Integer, k_Integer, a_] /; 1<=k<=n && Dimensions[a]=={2,2} :=
            KroneckerProduct[IdentityMatrix[2^(k-1), SparseArray],
                             a,
                             IdentityMatrix[2^(n-k), SparseArray]]
```

This allows us to define the single-qubit Pauli and Hadamard gates with

```
In[362]:= {id, σx, σy, σz} = Table[SparseArray[PauliMatrix[i]], {i, 0, 3}];
In[363]:= X[n_Integer, k_Integer] /; 1<=k<=n := op[n, k, σx]
In[364]:= Y[n_Integer, k_Integer] /; 1<=k<=n := op[n, k, σy]
In[365]:= Z[n_Integer, k_Integer] /; 1<=k<=n := op[n, k, σz]
In[366]:= H[n_Integer, k_Integer] /; 1<=k<=n := op[n, k, (σx+σz)/Sqrt[2]]
```

as well as the corresponding rotation operators $\hat{R}_x(\phi) = \frac{1+e^{i\phi}}{2}\mathbb{1} + \frac{1-e^{i\phi}}{2}\sigma_x = e^{i\phi/2}e^{-i\phi\hat{\sigma}_x/2}$ etc. that are also known as phase gates,

```
In[367]:= RX[n_Integer, k_Integer, φ_] /; 1<=k<=n :=
            op[n, k, (1+Exp[I*φ])/2*id + (1-Exp[I*φ])/2*σx]
In[368]:= RY[n_Integer, k_Integer, φ_] /; 1<=k<=n :=
            op[n, k, (1+Exp[I*φ])/2*id + (1-Exp[I*φ])/2*σy]
In[369]:= RZ[n_Integer, k_Integer, φ_] /; 1<=k<=n :=
            op[n, k, (1+Exp[I*φ])/2*id + (1-Exp[I*φ])/2*σz]
```

two-qubit gates

Interesting quantum circuits require operations that involve more than one qubit.

The SWAP gate exchanges the state of qubits j and k in a set of n qubits:

j —×—
k —×—

Without going through complicated considerations over basis-set indices, we construct it through the definition $\text{SWAP}^{(jk)} = (\mathbb{1}^{(j)} \otimes \mathbb{1}^{(k)} + \hat{\sigma}_x^{(j)} \otimes \hat{\sigma}_x^{(k)} + \hat{\sigma}_y^{(j)} \otimes \hat{\sigma}_y^{(k)} + \hat{\sigma}_z^{(j)} \otimes \hat{\sigma}_z^{(k)})/2$ and building on the above Pauli gates:

```
In[370]:= SWAP[n_Integer, {j_Integer, k_Integer}] /; 1<=j<=n && 1<=k<=n && j!=k :=
            (IdentityMatrix[2^n, SparseArray] +
             X[n,j].X[n,k] + Y[n,j].Y[n,k] + Z[n,j].Z[n,k])/2
```

The matrix representation of a two-qubit SWAP takes on the familiar form

```
In[371]:= SWAP[2, {1,2}] //Normal
Out[371]= {{1, 0, 0, 0},
           {0, 0, 1, 0},
           {0, 1, 0, 0},
           {0, 0, 0, 1}}
```

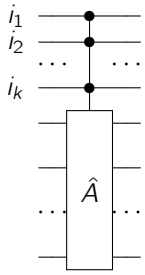
The square root of the SWAP gate is also sometimes used, and is defined similarly:

```
In[372]:= SQRTSWAP[n_Integer, {j_Integer, k_Integer}] /; 1<=j<=n && 1<=k<=n && j!=k :=
            (3+I)/4 * IdentityMatrix[2^n, SparseArray] +
            (1-I)/4 * (X[n,j].X[n,k] + Y[n,j].Y[n,k] + Z[n,j].Z[n,k])
```

To define the controlled-NOT or CNOT gate, we first make a general definition for controlled gates. The n-qubit operator `CTRL[n,λ,A]` acts like the operator $\hat{A}$ if all qubits in the list $\lambda = \{i_1, i_2, \ldots, i_k\}$ are in the $|1\rangle$ state, and has no action (acts like the identity operator on n qubits) if any of the qubits in the list λ are in the $|0\rangle$ state:

$$\text{CTRL} = \left[\bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}\right] \cdot \hat{A} + \left[\mathbb{1} - \bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}\right] \cdot \mathbb{1} = \mathbb{1} + \left[\bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}\right] \cdot (\hat{A} - \mathbb{1}) \tag{3.38}$$

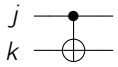
Its circuit representation is



The bracket in the last expression of Equation (3.38) is constructed with `Apply[Dot, op[n,#,P1]&/@λ]` that first constructs a list of projection operators $|1\rangle\langle 1|^{(i_j)}$ for the control qubits, and then applies the `Dot` operator to assemble them into the product $\bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}$.

```
In[373]:= P0 = (id + σz)/2 //SparseArray;  (* qubit projector |0⟩⟨0| *)
In[374]:= P1 = (id - σz)/2 //SparseArray;  (* qubit projector |1⟩⟨1| *)
In[375]:= CTRL[n_Integer, λ_ /; VectorQ[λ,IntegerQ], A_] /;
            (Unequal@@λ) && Min[λ]>=1 && Max[λ]<=n && Dimensions[A]=={2^n,2^n} :=
            IdentityMatrix[2^n, SparseArray] +
            Apply[Dot, op[n,#,P1]&/@λ].(A - IdentityMatrix[2^n, SparseArray])
```

With this definition, the CNOT operator $\text{CNOT}^{(jk)} = |0\rangle\langle 0|^{(j)} \otimes \mathbb{1}^{(k)} + |1\rangle\langle 1|^{(j)} \otimes \sigma_x^{(k)}$



is simply the CTRL operator with a single element in the list $\lambda = \{j\}$ and a single-qubit $\hat{A} = \hat{\sigma}_x^{(k)}$ operator,

```
In[376]:= CNOT[n_Integer, j_Integer -> k_Integer] /; 1<=j<=n && 1<=k<=n && j!=k :=
            CTRL[n, {j}, op[n, k, σx]]
```

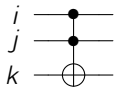
Notice that here we use the notation `CNOT[n, j->k]` to indicate that qubit j controls qubit k: this arrow notation `->` is purely for syntactic beauty and has no further effects (it is a pattern like any other, with no unintended side effects). The matrix representation of a two-qubit CNOT takes on the familiar form

```
In[377]:= CNOT[2, 1->2] //Normal
Out[377]= {{1, 0, 0, 0},
           {0, 1, 0, 0},
           {0, 0, 0, 1},
           {0, 0, 1, 0}}
```

three-qubit gates

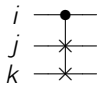
For completeness, we define three-qubit gates that are sometimes useful in the construction of general quantum circuits.

The CCNOT gate or Toffoli gate is a controlled-NOT gate `CCNOT[n, {i,j}->k]` with two controlling qubits, i and j, and is defined in analogy to the CNOT gate:



```
In[378]:= CCNOT[n_Integer, {i_Integer, j_Integer} -> k_Integer] /;
            1<=i<=n && 1<=j<=n && 1<=k<=n && Unequal[i,j,k] :=
              CTRL[n, {i,j}, op[n, k, σx]]
```

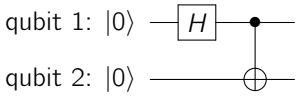
The controlled-SWAP gate or Fredkin gate `CSWAP[n, i->{j,k}]` conditionally swaps two qubits, j and k:




```
In[379]:= CSWAP[n_Integer, i_Integer -> {j_Integer, k_Integer}] /;
            1<=i<=n && 1<=j<=n && 1<=k<=n && Unequal[i,j,k] :=
              CTRL[n, {i}, SWAP[n, {j, k}]]
```

3.5.2 a simple quantum circuit

As a simple example, we study the quantum circuit



The unitary operation corresponding to this circuit is a Hadamard gate on qubit 1, followed by a controlled-NOT gate where qubit 1 controls the inversion of qubit 2. In Mathematica this gate sequence needs to be written from right to left, because the gates are represented by matrices that will be applied to a state vector on their right:

```
In[380]:= S = CNOT[2, 1->2] . H[2, 1];
In[381]:= Normal[S]
Out[381]= {{1/Sqrt[2], 0, 1/Sqrt[2], 0},
           {0, 1/Sqrt[2], 0, 1/Sqrt[2]},
           {0, 1/Sqrt[2], 0, -1/Sqrt[2]},
           {1/Sqrt[2], 0, -1/Sqrt[2], 0}}
```

The matrix representation of `Out[381]` refers to the two-qubit basis set $\mathcal{B}_2 = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, which we can inspect for any number of qubits with

```
In[382]:= B[n_Integer /; n>=1] := Tuples[{0, 1}, n]
In[383]:= B[2]
Out[383]= {{0,0}, {0,1}, {1,0}, {1,1}}
```

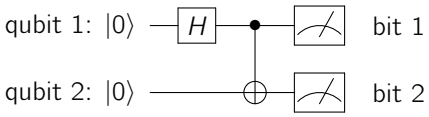
The input state of our circuit is the product state $|\psi_{\text{in}}\rangle = |0\rangle \otimes |0\rangle = |00\rangle$, which is the first element of $\mathcal{B}_2$:

```
In[384]:= ψin = {1,0,0,0};
```

The output state of our circuit follows from the application of `S`,

```
In[385]:= ψout = S . ψin
Out[385]= {1/Sqrt[2], 0, 0, 1/Sqrt[2]}
```

Looking at the basis set $\mathcal{B}_2$ we identify this output state with the maximally entangled state $|\psi_{\text{out}}\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Projective measurements on the two qubits,



give 50% probability of finding the classical result "00" and 50% probability of finding "11", whereas the bit combinations "01" and "10" never occur:

```
In[386]:= Abs[ψout]^2
Out[386]= {1/2, 0, 0, 1/2}
```

It is important to recognize that these four probabilities are insufficient to identify the state $|\psi_{\text{out}}\rangle$, even if many measurements are made, because any state whose diagonal density-matrix elements match `Out[386]` gives these probabilities. Generally, in order to identify a two-qubit output state fully, a quantum-state tomography (QST)[16] must be performed, which involves applying further phase gates (qubit rotations) before the projective measurements and measuring all sixteen (fifteen non-trivial) expectation values $\langle\psi_{\text{out}}|\hat{\sigma}_1 \otimes \hat{\sigma}_2|\psi_{\text{out}}\rangle$ for $\hat{\sigma}_1, \hat{\sigma}_2 \in \{\mathbb{1}, \hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z\}$, followed by an inversion procedure to estimate the density matrix:[17]

$$\begin{aligned}\hat{\rho} &= \frac{1}{4} \sum_{\hat{\sigma}_1 \in \{\mathbb{1}, \hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z\}} \sum_{\hat{\sigma}_2 \in \{\mathbb{1}, \hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z\}} \langle\psi_{\text{out}}|\hat{\sigma}_1 \otimes \hat{\sigma}_2|\psi_{\text{out}}\rangle \cdot \hat{\sigma}_1 \otimes \hat{\sigma}_2 \\ &= \frac{1}{4} \begin{bmatrix} \mathbb{1}\mathbb{1} + \mathbb{1}z + z\mathbb{1} + zz & \mathbb{1}x - \mathrm{i}\mathbb{1}y + zx - \mathrm{i}zy & x\mathbb{1} + xz - \mathrm{i}y\mathbb{1} - \mathrm{i}yz & xx - \mathrm{i}xy - \mathrm{i}yx - yy \\ \mathbb{1}x + \mathrm{i}\mathbb{1}y + zx + \mathrm{i}zy & \mathbb{1}\mathbb{1} - \mathbb{1}z + z\mathbb{1} - zz & xx + \mathrm{i}xy - \mathrm{i}yx + yy & x\mathbb{1} - xz - \mathrm{i}y\mathbb{1} + \mathrm{i}yz \\ x\mathbb{1} + xz + \mathrm{i}y\mathbb{1} + \mathrm{i}yz & xx - \mathrm{i}xy + \mathrm{i}yx + yy & \mathbb{1}\mathbb{1} + \mathbb{1}z - z\mathbb{1} - zz & \mathbb{1}x - \mathrm{i}\mathbb{1}y - zx + \mathrm{i}zy \\ xx + \mathrm{i}xy + \mathrm{i}yx - yy & x\mathbb{1} - xz + \mathrm{i}y\mathbb{1} - \mathrm{i}yz & \mathbb{1}x + \mathrm{i}\mathbb{1}y - zx - \mathrm{i}zy & \mathbb{1}\mathbb{1} - \mathbb{1}z - z\mathbb{1} + zz \end{bmatrix} \end{aligned} \tag{3.39}$$

(abbreviating $xy = \langle\psi_{\text{out}}|\hat{\sigma}_x \otimes \hat{\sigma}_y|\psi_{\text{out}}\rangle$ etc.) A full QST on n qubits requires measuring $4^n - 1$ such expectation values, which makes the QST infeasible in general.

3.5.3 application: the Quantum Fourier Transform

The discrete classical Fourier transform[18] (CFT) of a list of N complex numbers $\vec{x} = \{x_0, x_1, \ldots, x_{N-1}\}$ is given by the list $\vec{y} = \{y_0, y_1, \ldots, y_{N-1}\}$ with elements

$$y_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k e^{2\pi \mathrm{i} jk/N}. \tag{3.40}$$

It can be seen as a unitary matrix operation

$$\vec{y} = \boldsymbol{F} \cdot \vec{x} \qquad \text{with } F_{jk} = e^{2\pi \mathrm{i} jk/N}/\sqrt{N}. \tag{3.41}$$

With the Fast Fourier Transform (FFT) algorithm,[19] the computational effort of evaluating Equation (3.41) is of order $\mathcal{O}[N \log(N)]$.

The discrete Quantum Fourier Transform (QFT) is precisely the same transformation, except that the vectors $\vec{x}$ and $\vec{y}$ are encoded into quantum states. For this, a quantum system with Hilbert space dimension N is described by a basis set $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$, and the states $|x\rangle = \sum_{j=0}^{N-1} x_j|j\rangle$ and $|y\rangle = \sum_{j=0}^{N-1} y_j|j\rangle$ are seen as related by the unitary QFT operator $\hat{\mathcal{F}}$ such that

$$|y\rangle = \hat{\mathcal{F}}|x\rangle \qquad \text{with } \langle j|\hat{\mathcal{F}}|k\rangle = F_{jk}, \tag{3.42}$$

in analogy to Equation (3.41). The idea of this section is that the QFT can be evaluated much faster than the CFT, even though both are mathematically equivalent.

We assume that $N = 2^n$ is an integer power of two.[20] The Hilbert space of n qubits has exactly $2^n = N$ dimensions, and therefore we use these n qubits to encode the states $|x\rangle$ and $|y\rangle$ in the following way. The 2^n basis states $\mathcal{B}_n = \{|00\ldots00\rangle, |00\ldots01\rangle, |00\ldots10\rangle, \ldots, |11\ldots11\rangle\}$ are, in our usual construction through tensor products (section 2.4.2), listed in increasing order when interpreted as binary numbers (see `In[382]`). We give each basis state a new label equal to this binary number: $|00\ldots00\rangle = |0\rangle$,

[16] See https://en.wikipedia.org/wiki/Quantum_tomography.

[17] Equation (3.39) is a direct inversion that may not result in a positive semi-definite density matrix if experimental noise is present. In such cases, more elaborate inversion procedures are available.

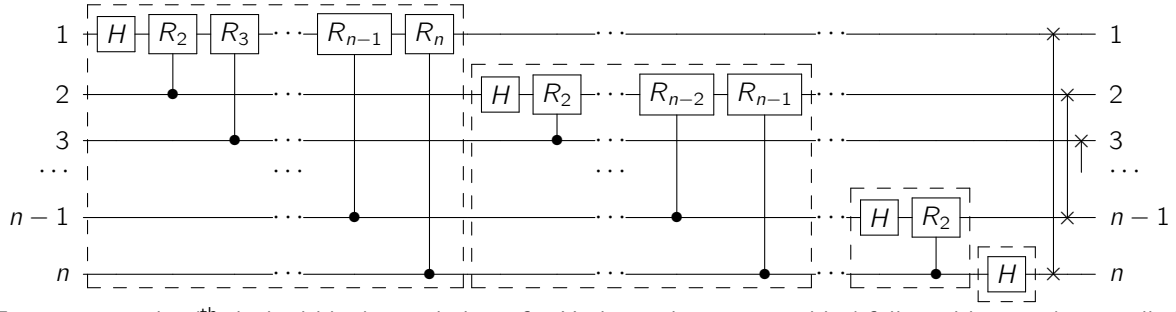
[18] See https://en.wikipedia.org/wiki/Discrete_Fourier_transform.

[19] See https://en.wikipedia.org/wiki/Fast_Fourier_transform.

[20] For all other cases, choose n as the smallest integer $\geq \log_2(N)$ and set $x_N \ldots x_{2^n-1}$ to zero.

$|00\ldots01\rangle = |1\rangle$, $|00\ldots10\rangle = |2\rangle$, $\ldots$, $|11\ldots11\rangle = |2^n - 1\rangle$, such that the state of the first qubit is the most significant bit (MSB) of the binary representation of the basis state's index, and the state of the n^{th} qubit is the least significant bit (LSB) of the binary representation of the basis state's index. What follows below is a quantum circuit operating on these n qubits that has the effect of the QFT operator $\hat{\mathcal{F}}$, as expressed in this binary basis.

The Quantum Fourier Transform circuit is assembled from single-qubit Hadamard gates and two-qubit controlled Z-phase gates, where $\hat{R}_k = \hat{R}_z(2\pi/2^k) = |0\rangle\langle 0| + e^{2\pi i/2^k}|1\rangle\langle 1|$ using `In[369]`:



To construct the i^{th} dashed block consisting of a Hadamard gate on qubit i followed by $n - i$ controlled Z-phase gates, we remember that the application of matrix operators happens from right to left, in the reverse order from that shown in the circit diagram above. We first construct a list of the controlled `RZ` operators and contract it by applying `Dot`:

```
In[387]:= QFTblock[n_Integer, i_Integer] /; 1<=i<=n :=
            Apply[Dot, Table[CTRL[n, {j}, RZ[n, i, 2π/2^(j+1-i)]], {j, n, i+1, -1}]].
            H[n,i]
```

We assemble the n-qubit QFT operator from these dashed `QFTblock` blocks and a set of SWAP operations that reverses the qubit order,

```
In[388]:= QFT[n_Integer] /; n>=1 :=
            Apply[Dot, Table[SWAP[n, {i, n+1-i}], {i, 1, n/2}]].
            Apply[Dot, Table[QFTblock[n, i], {i, n, 1, -1}]]
```

The matrix representation of this `QFT` operator is a $2^n \times 2^n$ matix with element (j, k) given by $2^{-n/2} e^{2\pi i jk/2^n}$, precisely as expected from Equation (3.42) with $N = 2^n$. We check this relation for $n = 1 \ldots 6$ with

```
In[389]:= Table[QFT[n] == 2^(-n/2)*Table[Exp[2π*I*j*k/2^n], {j,0,2^n-1}, {k,0,2^n-1}],
            {n, 6}] //FullSimplify
Out[389]= {True, True, True, True, True, True}
```

The resources used to construct this quantum circuit are

- n Hadamard gates,
- $\frac{n(n-1)}{2}$ controlled Z-phase gates, and
- $n/2$ swap gates.

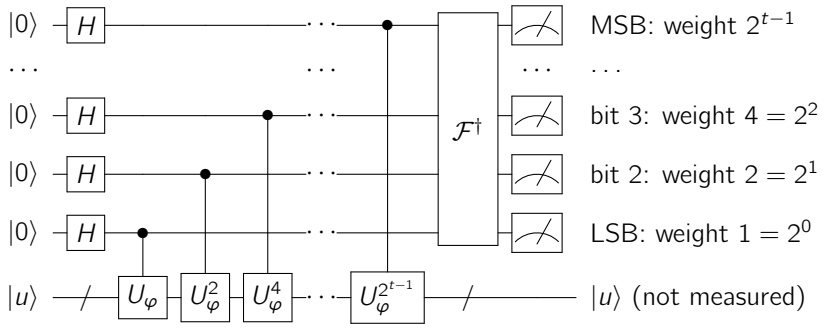
In the present classical simulation of quantum circuits, each quantum gate is a sparse $2^n \times 2^n$ matrix, usually containing $\mathcal{O}(2^n)$ nonzero matrix elements; applying such a simulated gate to a state therefore takes $\mathcal{O}(2^n)$ time, which makes the simulated QFT no faster than the classical FFT, which scales as $\mathcal{O}(2^n n)$. However, if we can construct a physical system in which these gates can be applied in a time that scales at most polynomially with n, then the QFT is a massive improvement over the scaling of the classical FFT. The development of such physical qubit/gate systems is the focus of much ongoing scientific research.

3.5.4 application: quantum phase estimation

The QFT circuit of section 3.5.3 cannot be used by itself in practice, because it requires the preparation of an arbitrary quantum state containing an exponential number of parameters x_j, as well as a full quantum-state tomography to read out an exponential number of parameters y_j describing the final state (see section 3.5.2). In this section we study a quantum circuit that uses the QFT as a component, and circumvents these exponential input/output bottlenecks.

Unitary matrices have eigenvalues that are of unit norm, and can be written as $e^{2\pi i\varphi}$ with $\varphi \in \mathbb{R}$. The question addressed here is: given a unitary operator $\hat{U}_\varphi$ and an eigenstate $|u\rangle$ such that $\hat{U}_\varphi|u\rangle = e^{2\pi i\varphi}|u\rangle$, can we estimate φ efficiently, that is, to t binary digits with an effort that scales polynomially with t?

The answer is yes, using the following quantum circuit that makes use of the Quantum Fourier Transform of section 3.5.3 but (i) starts with an initial product state that can be prepared with $\mathcal{O}(t)$ effort, and (ii) does not require a full quantum state tomography, but instead finishes with a simple projective measurement that takes $\mathcal{O}(t)$ effort.



To set up a quantum phase estimation in Mathematica, we begin by defining the unitary operator $\hat{U}_\varphi$ and its eigenstate $|u\rangle$ with

```
In[390]:= u = {1};
In[391]:= U[φ_] = {{Exp[2π*I*φ]}};
```

and check that they satisfy $\hat{U}_\varphi|u\rangle = e^{2\pi i\varphi}|u\rangle$ and $\langle u|u\rangle = 1$:

```
In[392]:= {U[φ].u === E^(2π*I*φ)*u, Norm[u] == 1}
Out[392]= {True, True}
```

Here we use a one-dimensional quantum system: the operator $\hat{U}_\varphi$ is a 1×1 matrix, and the state $|u\rangle$ is a list of length 1. More generally, the Hilbert space of the system under test (SUT) can be arbitrarily large (see **Q3.22**), and more complex quantum circuits can be substituted for $\hat{U}$ in more elaborate experiments.

In order to construct the phase estimation circuit, we will also need the unit operator acting on the SUT:

```
In[393]:= U0 = IdentityMatrix[Length[u], SparseArray];
```

The controlled version of the $\hat{U}_\varphi$ operator, where the i^{th} qubit out of a set of n qubits controls the application of $\hat{U}_\varphi$ to the SUT, is $\hat{U}_\varphi^{(i)} = |0\rangle\langle 0|^{(i)} \otimes \mathbb{1} + |1\rangle\langle 1|^{(i)} \otimes \hat{U}_\varphi$. We use the tensor-product techniques of section 2.4.2 to couple the qubits to the SUT:

```
In[394]:= CTRLU[n_Integer, i_Integer, φ_] /; 1<=i<=n :=
            KroneckerProduct[op[n,i,P0], U0] + KroneckerProduct[op[n,i,P1], U[φ]]
```

The initial state of the phase estimation circuit is $|\psi_0\rangle = |0\rangle^{\otimes t} \otimes |u\rangle$. We know that the state $|0\rangle^{\otimes t} = |00\ldots00\rangle$ is the first basis state in the computational basis $\mathcal{B}_n$ (eigen-basis of $\hat{\sigma}_z$), and construct it with `SparseArray[1->1, 2^t]`. As an example, we work with $t = 4$ qubits here:

```
In[395]:= t = 4;
In[396]:= ψ0 = Flatten[KroneckerProduct[SparseArray[1->1, 2^t], u]] //Normal;
```

Applying a Hadamard gate to each qubit gives the state

```
In[397]:= ψ1 = KroneckerProduct[Apply[Dot, Table[H[t, i], {i, t}]], U0] . ψ0;
```

Applying the controlled $\hat{U}_\varphi^m = \hat{U}_{m\varphi}$ operations sequentially then gives the state

```
In[398]:= ψ2[φ_] = Apply[Dot, Table[CTRLU[t, i, 2^(t-i)*φ], {i, t, 1, -1}]] . ψ1;
```

Finally, an inverse QFT yields the phase-estimation state $|\varepsilon_\varphi\rangle$. Remember that the QFT is a unitary operation, and therefore its inverse is its Hermitian conjugate:

```
In[399]:= ε[φ_] = KroneckerProduct[ConjugateTranspose[QFT[t]], U0] . ψ2[φ];
```

We use the techniques of section 2.4.3, in particular In[257], to drop the component $|u\rangle$ at the end of the quantum circuit and find the reduced density matrix of the qubits. The diagonal elements of this reduced density matrix are the probabilities of finding the various basis states of $\mathcal{B}_n$ in a projective measurement as shown on the right of the above circuit:

```
In[400]:= prob[φ_?NumericQ] := Re[Diagonal[traceout[ε[N[φ]], -Length[u]]]]
```

The first element of `prob[φ]` gives the probability of measurement outcomes $\{0,0,0,0\}$, that is, the probability that the qubits are in the joint state $|0000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle$. The second element of `prob[φ]` gives the probability of measurement outcomes $\{0,0,0,1\}$, that is, the probability that the qubits are in the joint state $|0001\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle$. And so forth: the j^{th} element of `prob[φ]` gives the probability of measurement outcomes corresponding to the binary representation of $j-1$.

The trick of this phase-estimation quantum circuit is that the information on φ is contained in the state $|\varepsilon_\varphi\rangle$ in a way that can be extracted from these probabilities without doing a full quantum-state tomography. We get an idea of what this means by looking at the probabilities for the different measurement outcomes when φ is an integer multiple of 2^{-t}:

```
In[401]:= Table[prob[φ], {φ, 0, 1, 2^(-t)}] //Chop
Out[401]= {{1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 1., 0, 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 1., 0, 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0, 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1., 0},
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.},
          {1., 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
```

Whenever φ is an integer multiple of $2^{-t} = 1/16$, we find that only one basis state is occupied, and therefore the outcomes of the projective measurements on the 4 qubits always give the same results, with no quantum fluctuation. A single projective measurement of all 4 qubits can be interpreted as a binary number $j \in \{0,1,2,\ldots,15\}$ that is related to the phase estimate as $\varphi = j/16$; no quantum-state tomography is required.

What happens when φ is not an integer multiple of 2^{-t}? It turns out that the basis state corresponding to the nearest integer multiple of 2^{-t} will be found most frequently in the projective measurements. For example, for $\varphi = 0.2$ the probabilities for projecting $|\varepsilon_{0.2}\rangle$ into the 16 basis states are

```
In[402]:= prob[0.2]
Out[402]= {0., 0.01, 0.02, 0.88, 0.06, 0.01, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}
```

(rounded here to two decimals). The fourth basis state, which is $|0011\rangle$ corresponding to $\varphi = 3/16$, will be found in about 88% of all experiments, and so a plurality vote most likely yields $\varphi \approx 3/16 = 0.1875$ as a fair estimate of the phase, with an upper bound on the error of $2^{-t-1} = 1/32$; no quantum-state tomography required. We extract the expected plurality-vote winner of a large number of experiments with

```
In[403]:= mostprobable[φ_?NumericQ] := (Ordering[prob[φ], -1][[1]] - 1)/2^t
```

where the `Ordering` function is used to give the position of the largest element:

```
In[404]:= mostprobable[0.2]
Out[404]= 3/16
```

It can be shown that `mostprobable[φ]==Mod[Round[φ, 2^(-t)], 1]`. Increasing the number of qubits t results in more precise estimates, while keeping the circuit complexity at $\mathcal{O}(t^2)$.

3.5.5 exercises

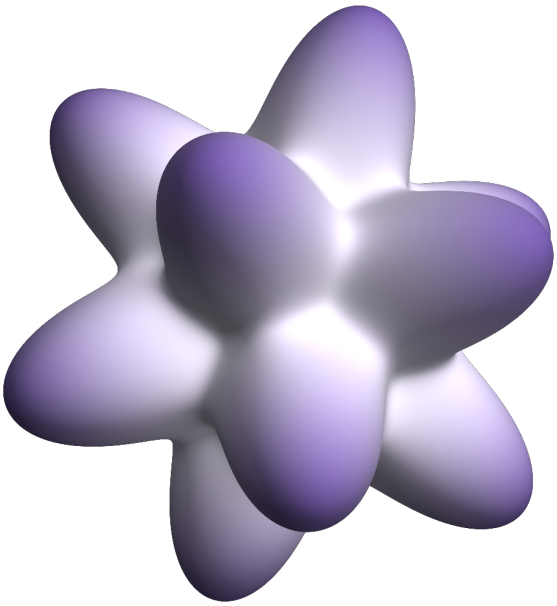
Q3.21 For the output state $|\psi_{\text{out}}\rangle$ of `Out[386]`, calculate all expectation values necessary to fill in Equation (3.39).

Q3.22 Multi-dimensional phase estimation: set $u = \{1, 1\}/\sqrt{2}$ and $\hat{U}_\varphi = e^{2\pi i\varphi}\{\{1, 0\}, \{0, 1\}\}$ (two-dimensional system under test) and show that the phase-estimation algorithm still works.

Q3.23 What happens if $|u\rangle$ is not an eigenstate of $\hat{U}_\varphi$? Set $u = \{1, 1\}/\sqrt{2}$ and $\hat{U}_\varphi = \{\{e^{2\pi i\varphi}, 0\}, \{0, e^{4\pi i\varphi}\}\}$ (two-dimensional system with two different evolution frequencies) and re-evaluate the attached Mathematica script. Plot `prob[φ]` for a range of frequencies φ using `ListDensityPlot` and interpret the resulting figure.

4

quantum motion in real space



So far we have studied the quantum formalism in the abstract (chapter 2) and in the context of rotational dynamics (chapter 3). In this chapter we work with the spatial motion of point particles, which represents a kind of mechanics that is much closer to our everyday experience. Here, quantum states are called *wavefunctions* and depend on the spatial coordinate(s). This apparent difference to the material covered in the previous chapters disappears when we express all wavefunctions in a basis set. We develop numerical methods for studying spatial dynamics that stay as close to a real-space description as quantum mechanics allows.

4.1 one particle in one dimension

A single particle moving in one dimension is governed by a Hamiltonian of the form

$$\hat{\mathcal{H}} = \hat{T} + \hat{V} \tag{4.1}$$

in terms of the kinetic operator $\hat{T}$ and the potential operator $\hat{V}$. These operators are usually expressed in the Dirac position basis set $\{|x\rangle\}_{x\in\mathbb{R}}$,[1] which diagonalizes the position operator in the sense that $\hat{x}|x\rangle = x|x\rangle$,[2] is ortho-normalized $\langle x|y\rangle = \delta(x-y)$, and complete $\int_{-\infty}^{\infty}|x\rangle\langle x|\mathrm{d}x = \mathbb{1}$. Using this Dirac basis, the explicit expressions for the operators in the Hamiltonian are

$$\hat{T} = -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x|x\rangle\frac{\mathrm{d}^2}{\mathrm{d}x^2}\langle x|, \qquad \hat{V} = \int_{-\infty}^{\infty}\mathrm{d}x|x\rangle V(x)\langle x|, \tag{4.2}$$

where m is the particle's mass and $V(x)$ is its potential. Single-particle states $|\psi\rangle$, on the other hand, are written in this basis as

$$|\psi\rangle = \int_{-\infty}^{\infty}\mathrm{d}x\psi(x)|x\rangle, \tag{4.3}$$

where $\psi(x) = \langle x|\psi\rangle$ is the wavefunction.

In what follows we restrict the freedom of the particle to a domain $x \in \Omega = [0, a]$, where a can be very large in order to approximately describe infinite systems (example: section 4.1.7). This assumes the potential to be

$$V(x) = \begin{cases} \infty & \text{for } x \le 0 \\ W(x) & \text{for } 0 < x < a \\ \infty & \text{for } x \ge a \end{cases} \tag{4.4}$$

This restriction is necessary in order to achieve a finite representation of the system in a computer.

exercises

Q4.1 Insert Equations (4.2) and Equation (4.3) into the time-independent Schrödinger equation $\hat{\mathcal{H}}|\psi\rangle = E|\psi\rangle$. Use the ortho-normality of the Dirac basis to derive the usual form of the Schrödinger equation for a particle's wavefunction in 1D: $-\frac{\hbar^2}{2m}\psi''(x) + V(x)\psi(x) = E\psi(x)$.

Q4.2 Use Equation (4.3) to show that the scalar product between two states is given by the usual formula $\langle\psi|\chi\rangle = \int_{-\infty}^{\infty}\psi^*(x)\chi(x)\mathrm{d}x$.

4.1.1 units

In order to proceed with implementing the Hamiltonian (4.1), we first need a consistent set of units (see section 1.12) in which to express length, time, mass, and energy. Of these four units, only three are independent: expressions like the classical kinetic energy $E = \frac{1}{2}mv^2$ indicate a fixed relationship between these four units.

A popular system of units is the International System of Units (SI),[3] in which this consistency is built in:

```
In[405]:= LengthUnit = Quantity["Meters"];     (* choose freely *)
In[406]:= TimeUnit = Quantity["Seconds"];      (* choose freely *)
In[407]:= MassUnit = Quantity["Kilograms"];    (* choose freely *)
In[408]:= EnergyUnit = MassUnit*LengthUnit^2/TimeUnit^2 //UnitConvert;
```

[1] To be exact, the Dirac position basis set spans a space that is much larger than the Hilbert space of square-integrable smooth functions used in quantum mechanics. This can be seen by noting that this basis set has an uncountably infinite number of elements $|x\rangle$, while the dimension of the Hilbert space in question is only countably infinite [see Equation (4.5) for a countably infinite basis set]. The underlying problem of the *continuum*, which quantum mechanics attempts to resolve, is discussed with some of its philosophical origins and implications by Erwin Schrödinger in his essay "Science and Humanism" (Cambridge University Press, 1951, ISBN 978-0521575508).

[2] This eigenvalue equation is tricky: remember that $\hat{x}$ is an operator, $|x\rangle$ is a state, and x is a real number.

[3] See https://en.wikipedia.org/wiki/International_System_of_Units.

The consistency of this set of definitions is seen in In[408], making the energy unit depend on the other units, which in turn can be chosen freely. Many other combinations are possible, as long as this consistency remains.

Another popular choice is to additionally couple the time and energy units through Planck's constant $\hbar$, and make both dependent on the length and mass units (thus reducing the system of units to only two degrees of freedom):

```
In[409]:= LengthUnit = Quantity["Meters"];     (* choose freely *)
In[410]:= MassUnit = Quantity["Kilograms"];    (* choose freely *)
In[411]:= TimeUnit =
            MassUnit*LengthUnit^2/Quantity["ReducedPlanckConstant"] //UnitConvert;
In[412]:= EnergyUnit = Quantity["ReducedPlanckConstant"]/TimeUnit //UnitConvert;
```

This latter set of units is what we will be using in what follows, without restriction of generality. We express the reduced Planck constant in these units with

```
In[413]:= ħ = Quantity["ReducedPlanckConstant"]/(EnergyUnit*TimeUnit) //UnitConvert //N
Out[413]= 1.
```

which is equal to unity because of our chosen coupling between energy and time units; in other unit systems the value will be different. Note the use of //N at the end of In[413] to force the result to be a pure machine-precision number instead of a variable-precision number that tracks the accuracy of the involved physical quantities.

To set the physical size a of the computational box, for example to $a = 5\,\mu\text{m}$, we execute

```
In[414]:= a = Quantity[5, "Micrometers"]/LengthUnit //UnitConvert //N;
```

and to set the particle's mass m, for example to the neutron's mass,

```
In[415]:= m = Quantity["NeutronMass"]/MassUnit //UnitConvert //N;
```

In the calculations that follow, we will not be explicit about the system of units and the physical quantities. Instead, we will use direct dimensionless definitions such as

```
In[416]:= a = 30;    (* calculation box size in units of length *)
In[417]:= m = 1;     (* particle mass in units of mass *)
In[418]:= ħ = 1;     (* value of ħ assuming In[412] *)
```

These are to be replaced by In[414], In[415], and In[413] in a more concrete physical situation.

4.1.2 computational basis functions

In order to perform quantum-mechanical calculations of a particle moving in one dimension, we need a basis set that is more practical than the Dirac basis used to define the relevant operators and states above. Indeed, Dirac states $|x\rangle$ are difficult to represent in a computer because they are uncountable, densely spaced, and highly singular.

The most generally useful basis sets for computations are the *momentum basis* and the *finite-resolution position basis*, which we will look at in turn, and which will be shown to be related to each other by a type-I discrete sine transform.

momentum basis

The simplest one-dimensional quantum-mechanical system of the type of Equation (4.1) is the infinite square well with $W(x) = 0$. Its energy eigenstates $\phi_n(x)$ for $n = 1, 2, 3, \ldots$ satisfy the Schrödinger

equation $-\frac{\hbar^2}{2m}\phi_n''(x) = E_n\phi_n(x)$ (see **Q4.1**) and the boundary conditions $\phi_n(0) = \phi_n(a) = 0$ necessitated by Equation (4.4). Their explicit normalized forms are

$$\langle x|n\rangle = \phi_n(x) = \sqrt{\frac{2}{a}}\sin\left(\frac{n\pi x}{a}\right) \tag{4.5}$$

with eigen-energies

$$E_n = \frac{n^2\pi^2\hbar^2}{2ma^2}. \tag{4.6}$$

We know from the Sturm–Liouville theorem[4] that these functions form a complete set (see **Q2.2**); further, we can use Mathematica to show that they are ortho-normalized:

```
In[419]:= φ[a_, n_, x_] = Sqrt[2/a]*Sin[n*π*x/a];
In[420]:= Table[Integrate[φ[a,n1,x]*φ[a,n2,x], {x, 0, a}],
            {n1, 10}, {n2, 10}] //MatrixForm
```

They are eigenstates of the squared momentum operator $\hat{p}^2 = \left(-i\hbar\frac{d}{dx}\right)^2 = -\hbar^2\frac{d^2}{dx^2}$:

$$\hat{p}^2|n\rangle = \frac{n^2\pi^2\hbar^2}{a^2}|n\rangle, \tag{4.7}$$

which we verify with

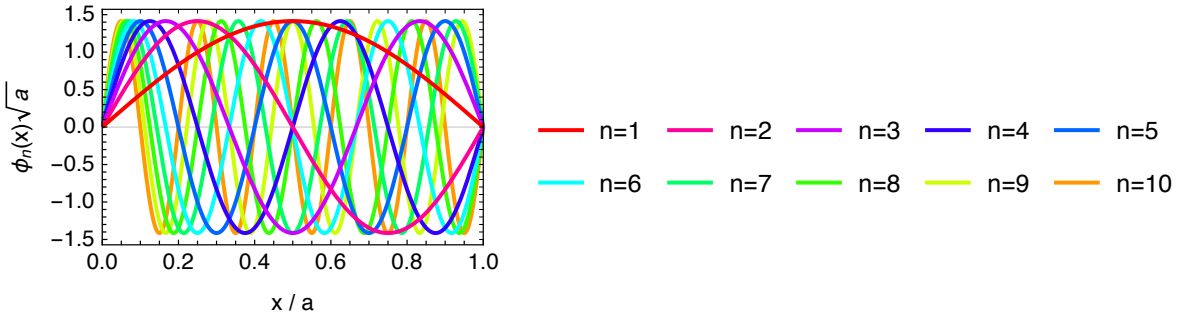
```
In[421]:= -ħ^2*D[φ[a,n,x], {x,2}] == (n^2*π^2*ħ^2)/a^2*φ[a,n,x]
Out[421]= True
```

This makes the kinetic operator $\hat{T} = \hat{p}^2/(2m)$ diagonal in this basis:

$$\langle n|\hat{T}|n'\rangle = E_n\delta_{nn'}, \qquad \hat{T} = \sum_{n=1}^{\infty}|n\rangle E_n\langle n|. \tag{4.8}$$

However, in general the potential energy, and most other operators that will appear later, are difficult to express in this momentum basis.

The momentum basis of Equation (4.5) contains a countably infinite number of basis functions, which is a great advantage over the uncountably infinite cardinality of the Dirac basis set. In practical calculations, we restrict the computational basis to $n \in \{1 \ldots n_{\max}\}$, which means that we only consider physical phenomena with excitation energies below $E_{n_{\max}} = \frac{\pi^2\hbar^2}{2ma^2}n_{\max}^2$ (see section 2.1.1). Here is an example of what these position-basis functions look like for $n_{\max} = 10$:



Using the approximate completeness of the momentum basis, $\sum_{n=1}^{n_{\max}}|n\rangle\langle n| \approx \mathbb{1}$ (see section 2.1.1), the kinetic Hamiltonian thus becomes

$$\hat{T} \approx \left[\sum_{n=1}^{n_{\max}}|n\rangle\langle n|\right]\hat{T}\left[\sum_{n'=1}^{n_{\max}}|n'\rangle\langle n'|\right] = \sum_{n,n'=1}^{n_{\max}}|n\rangle\langle n|\hat{T}|n'\rangle\langle n'| = \sum_{n=1}^{n_{\max}}|n\rangle E_n\langle n|. \tag{4.9}$$

We set up the kinetic Hamiltonian operator as a sparse diagonal matrix in the momentum basis with

[4] See https://en.wikipedia.org/wiki/Sturm-Liouville_theory.

```
In[422]:= nmax = 100;
In[423]:= TM = SparseArray[Band[{1,1}] -> Range[nmax]^2*π^2*ħ^2/(2*m*a^2)];
```

where $n_{\max} = 100$ was chosen as an example.

finite-resolution position basis

Given an energy-limited momentum basis set $\{|n\rangle\}_{n=1}^{n_{\max}}$ from above, we define a set of $n_{\max}$ equally-spaced points

$$x_j = j \cdot \Delta \tag{4.10}$$

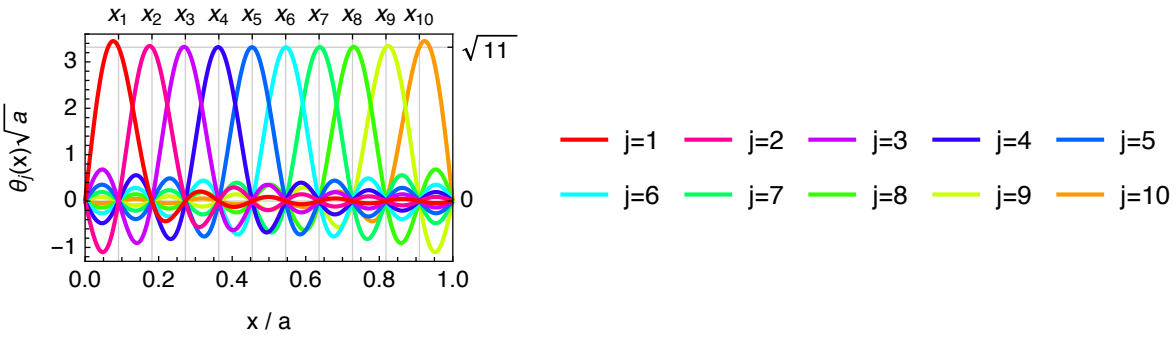
for $j \in \{1 \ldots n_{\max}\}$, with spacing $\Delta = a/(n_{\max}+1)$. These grid points fill the calculation range $x \in [0, a]$ uniformly without covering the end points. We then define a new basis set as the closest possible representations of delta-functions at these points: for $j \in \{1 \ldots n_{\max}\}$,

$$|j\rangle = \sqrt{\Delta} \sum_{n=1}^{n_{\max}} \phi_n(x_j)|n\rangle. \tag{4.11}$$

The spatial wavefunctions of these basis states are

$$\langle x|j\rangle = \vartheta_j(x) = \sqrt{\Delta} \sum_{n=1}^{n_{\max}} \phi_n(x_j)\phi_n(x). \tag{4.12}$$

Here is an example of what these position-basis functions look like for $n_{\max} = 10$:



This new basis set is also ortho-normal, $\langle j|j'\rangle = \delta_{jj'}$, and it is strongly local in the sense that only the basis function $\vartheta_j(x)$ is nonzero at x_j, while all others vanish:

$$\langle x_{j'}|j\rangle = \vartheta_j(x_{j'}) = \delta_{jj'}/\sqrt{\Delta}. \tag{4.13}$$

We define these basis functions in Mathematica with

```
In[424]:= nmax = 10;
In[425]:= Δ = a/(nmax+1);
In[426]:= xx[j_] = j*Δ;
In[427]:= θ[j_, x_] = Sqrt[Δ]*Sum[φ[n,xx[j]]*φ[n,x], {n, nmax}];
```

Since the basis function $\vartheta_j(x) =$ `θ[j,x]` is the only one which is nonzero at $x_j =$ `xx[j]`, and it is close to zero everywhere else (exactly zero at the $x_{j'\neq j}$), we can usually make several approximations:

- If a wavefunction is given as a vector $\vec{\boldsymbol{v}} =$ `v` in the position basis, $|\psi\rangle = \sum_{j=1}^{n_{\max}} v_j|j\rangle$, then by Equation (4.13) the wavefunction is known at the grid points:

$$\psi(x_j) = \langle x_j|\psi\rangle = \langle x_j| \sum_{j'=1}^{n_{\max}} v_{j'}|j'\rangle = \sum_{j'=1}^{n_{\max}} v_{j'}\langle x_j|j'\rangle = \sum_{j'=1}^{n_{\max}} v_{j'}\delta_{jj'}/\sqrt{\Delta} = \frac{v_j}{\sqrt{\Delta}}. \tag{4.14}$$

  The density profile is thus given by the values of $\rho(x_j) = |\psi(x_j)|^2 = |v_j|^2/\Delta$. This allows for very easy plotting of wavefunctions and densities by linearly interpolating between these grid points (*i.e.*, an interpolation of order 1):

```
In[428]:= ListLinePlot[Transpose[{Table[xx[j], {j, nmax}], Abs[v]^2/Δ}]]
```

By the truncation of the basis at $n_{\max}$, the wavefunction has no frequency components faster than one half-wave per grid-point spacing, and therefore we can be sure that this linear interpolation is a reasonably accurate representation of the wavefunction $\psi(x)$ and the density $\rho(x) = |\langle x|\psi\rangle|^2$, in particular as $n_{\max} \to \infty$.

- An even simpler interpolation of order zero assumes that the wavefunction is constant over intervals $[-\Delta/2, \Delta/2]$ centered at each grid point. This primitive interpolation is used, for example, to calculate the Wigner quasi-probability distribution in section 4.1.8.

- Similarly, if a density operator is given by $\hat\rho = \sum_{j,j'=1}^{n_{\max}} R_{j,j'}|j\rangle\langle j'|$, then the value of the density operator at a grid point $(x_j, x_{j'})$ is given by

$$\rho(x_j, x_{j'}) = \langle x_j|\hat\rho|x_{j'}\rangle = \langle x_j|\left[\sum_{j'',j'''=1}^{n_{\max}} R_{j'',j'''}|j''\rangle\langle j'''|\right]|x_{j'}\rangle = \sum_{j'',j'''=1}^{n_{\max}} R_{j'',j'''}\langle x_j|j''\rangle\langle j'''|x_{j'}\rangle$$
$$= \sum_{j'',j'''=1}^{n_{\max}} R_{j'',j'''}\frac{\delta_{j,j''}}{\sqrt{\Delta}}\frac{\delta_{j',j'''}}{\sqrt{\Delta}} = \frac{R_{j,j'}}{\Delta}. \quad (4.15)$$

That is, the coefficients $R_{j,j'}$ and the density values $\rho(x_j, x_{j'})$ are very closely related. The diagonal elements of this expression ($j = j'$) give the spatial density profile.

- For any function $f(x)$ that varies slowly (smoothly) over length scales of the grid spacing Δ, we can make the approximation

$$f(x)\vartheta_j(x) \approx f(x_j)\vartheta_j(x). \quad (4.16)$$

This approximation becomes exact on every grid point according to Equation (4.13), and the assumed smoothness of $f(x)$ makes it a good estimate for any x.

conversion between basis sets

Within the approximation of a truncation at maximum energy $E_{n_{\max}}$, we can express any wavefunction $|\psi\rangle$ in both basis sets of Equation (4.5) and Equation (4.12):

$$|\psi\rangle = \sum_{n=1}^{n_{\max}} u_n|n\rangle = \sum_{j=1}^{n_{\max}} v_j|j\rangle \quad (4.17)$$

Inserting the definition of Equation (4.11) into Equation (4.17) we find

$$\sum_{n=1}^{n_{\max}} u_n|n\rangle = \sum_{j=1}^{n_{\max}} v_j\left[\sqrt{\Delta}\sum_{n'=1}^{n_{\max}}\phi_{n'}(x_j)|n'\rangle\right] = \sum_{n'=1}^{n_{\max}}\left[\sqrt{\Delta}\sum_{j=1}^{n_{\max}} v_j\phi_{n'}(x_j)\right]|n'\rangle \quad (4.18)$$

and therefore, since the basis set $\{|n\rangle\}$ is ortho-normalized,

$$u_n = \sqrt{\Delta}\sum_{j=1}^{n_{\max}} v_j\phi_n(x_j) = \sum_{j=1}^{n_{\max}} X_{nj}v_j \quad (4.19)$$

with the basis conversion coefficients

$$X_{nj} = \langle n|j\rangle = \sqrt{\Delta}\phi_n(x_j) = \sqrt{\frac{a}{n_{\max}+1}}\sqrt{\frac{2}{a}}\sin\left(\frac{n\pi x_j}{a}\right) = \sqrt{\frac{2}{n_{\max}+1}}\sin\left(\frac{\pi nj}{n_{\max}+1}\right). \quad (4.20)$$

The inverse transformation is found from $|n\rangle = \sum_{j=1}^{n_{\max}}\langle j|n\rangle|j\rangle$ inserted into Equation (4.17), giving

$$v_j = \sum_{n=1}^{n_{\max}} X_{nj}u_n \quad (4.21)$$

in terms of the same coefficients of Equation (4.20). Thus the transformations relating the vectors $\vec{\boldsymbol{u}}$ (with components u_n) and $\vec{\boldsymbol{v}}$ (with components v_j) are $\vec{\boldsymbol{v}} = \boldsymbol{X} \cdot \vec{\boldsymbol{u}}$ and $\vec{\boldsymbol{u}} = \boldsymbol{X} \cdot \vec{\boldsymbol{v}}$ in terms of the same symmetric orthogonal matrix $\boldsymbol{X}$ with coefficients X_{nj}.

We could calculate these coefficients with

```
In[429]:= X = Table[Sqrt[2/(nmax+1)]*Sin[π*n*j/(nmax+1)], {n, nmax}, {j, nmax}] //N;
```

but this is not very efficient, especially for large n_{max}.

It turns out that Equation (4.19) and Equation (4.21) relate the vectors $\vec{\boldsymbol{u}}$ and $\vec{\boldsymbol{v}}$ by a *type-I discrete sine transform* (DST-I), which Mathematica can evaluate very efficiently via a fast Fourier transform.[5] Since the DST-I is its own inverse, we can use

```
In[430]:= v = FourierDST[u, 1];
In[431]:= u = FourierDST[v, 1];
```

to effect such conversions. We will see a very useful application of this transformation when we study the time-dependent behavior of a particle in a potential ("split-step method", section 4.1.9).

The matrix $\boldsymbol{X}$ is also useful for converting *operator* representations between the basis sets: the momentum representation $\boldsymbol{U}$ and the position representation $\boldsymbol{V}$ of the same operator satisfy $\boldsymbol{V} = \boldsymbol{X} \cdot \boldsymbol{U} \cdot \boldsymbol{X}$ and $\boldsymbol{U} = \boldsymbol{X} \cdot \boldsymbol{V} \cdot \boldsymbol{X}$. In practice, as above we can convert operators between the position and momentum representation with a two-dimensional type-I discrete sine transform:

```
In[432]:= V = FourierDST[U, 1];
In[433]:= U = FourierDST[V, 1];
```

This easy conversion is very useful for the construction of the matrix representations of Hamiltonian operators, since the kinetic energy is diagonal in the momentum basis, Equation (4.7), while the potential energy operator is approximately diagonal in the position basis, Equation (4.25).

4.1.3 the position operator

The position operator $\hat{x} = \int_{-\infty}^{\infty} \mathrm{d}x |x\rangle x \langle x|$ is one of the basic operators that is used frequently to construct Hamiltonians of moving particles. The exact expressions for the matrix elements of this operator in the momentum basis are

$$\langle n|\hat{x}|n'\rangle = \int_0^a \mathrm{d}x \sqrt{\frac{2}{a}} \sin\left(\frac{n\pi x}{a}\right) x \sqrt{\frac{2}{a}} \sin\left(\frac{n'\pi x}{a}\right) = \begin{cases} \frac{a}{2} & \text{if } n = n' \\ -\frac{8ann'}{\pi^2(n^2-n'^2)^2} & \text{if } n - n' \text{ is odd} \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

This allows us to construct the exact matrix representations of the operator $\hat{x}$ in both the momentum (`xM`) and the position (`xP`) bases:

```
In[434]:= xM = SparseArray[{
              Band[{1,1}] -> a/2,
              {n1_,n2_} /; OddQ[n1-n2] -> -8*a*n1*n2/(π^2*(n1^2-n2^2)^2)},
            {nmax,nmax}];
In[435]:= xP = FourierDST[xM, 1];
```

[5] See https://en.wikipedia.org/wiki/Discrete_sine_transform and https://en.wikipedia.org/wiki/Fast_Fourier_transform. The precise meaning of the DST-I can be seen from its equivalent definition through a standard discrete Fourier transform of doubled length: for a complex vector `v`, we can substitute `FourierDST[v,1]` by `DST1[v_?VectorQ]:=-I*Fourier[Join[{0},v,{0},Reverse[-v]]][[2;;Length[v]+1]]`. In this sense it is the discrete Fourier transform of a list `v` augmented with (i) zero boundary conditions and (ii) reflection anti-symmetry at the boundaries. Remember that the `Fourier[]` transform assumes periodic boundary conditions, which are incorrect in the present setup, and need to be modified into a DST-I.

A simple approximation of the position operator, which will be extremely useful in what follows, is found by observing that xP is almost a diagonal matrix, with approximately the grid coordinates $x_1, x_2, \ldots, x_{n_{\max}}$ on the diagonal. This approximate form can be proved by using the locality of the position basis functions, Equation (4.16):

$$x_{jj'} = \langle j|\hat{x}|j'\rangle = \langle j|\left[\int_{-\infty}^{\infty} \mathrm{d}x|x\rangle x\langle x|\right]|j'\rangle = \int_0^a \mathrm{d}x\langle j|x\rangle x\langle x|j'\rangle = \int_0^a \mathrm{d}x\vartheta_j^*(x)x\vartheta_{j'}(x)$$
$$\approx x_{j'}\int_0^a \mathrm{d}x\vartheta_j^*(x)\vartheta_{j'}(x) = \delta_{jj'}x_j. \quad (4.23)$$

The resulting approximate diagonal form of the position operator in the position basis, found from the approximate completeness relation $\sum_{j=1}^{n_{\max}}|j\rangle\langle j| \approx \mathbb{1}$, is

$$\hat{x} \approx \left[\sum_{j=1}^{n_{\max}}|j\rangle\langle j|\right]\hat{x}\left[\sum_{j'=1}^{n_{\max}}|j'\rangle\langle j'|\right] = \sum_{j,j'=1}^{n_{\max}}|j\rangle\langle j|\hat{x}|j'\rangle\langle j'| \approx \sum_{j,j'=1}^{n_{\max}}|j\rangle\delta_{jj'}x_j\langle j'| = \sum_{j=1}^{n_{\max}}|j\rangle x_j\langle j|. \quad (4.24)$$

```
In[436]:= Δ = a/(nmax+1);                          (* the grid spacing *)
In[437]:= xgrid = Range[nmax]*Δ;                   (* the computational grid *)
In[438]:= xP = SparseArray[Band[{1,1}] -> xgrid];  (* x operator, position basis *)
In[439]:= xM = FourierDST[xP, 1];                  (* x operator, momentum basis *)
```

4.1.4 the potential-energy operator

If a potential energy function $W(x)$ varies smoothly over length scales of the grid spacing Δ, then the trick of section 4.1.3 allows us to approximate the matrix elements of this potential energy in the position basis,

$$V_{jj'} = \langle j|\hat{V}|j'\rangle = \langle j|\left[\int_0^a \mathrm{d}x|x\rangle W(x)\langle x|\right]|j'\rangle = \int_0^a \mathrm{d}x\langle j|x\rangle W(x)\langle x|j'\rangle = \int_0^a \mathrm{d}x\vartheta_j^*(x)W(x)\vartheta_{j'}(x)$$
$$\approx W(x_{j'})\int_0^a \mathrm{d}x\vartheta_j^*(x)\vartheta_{j'}(x) = \delta_{jj'}W(x_j), \quad (4.25)$$

where we have used the definitions of Equation (4.2) and Equation (4.4). This is a massive simplification compared to the explicit evaluation of potential integrals for each specific potential energy function. The potential-energy operator thus becomes approximately

$$\hat{V} \approx \left[\sum_{j=1}^{n_{\max}}|j\rangle\langle j|\right]\hat{V}\left[\sum_{j'=1}^{n_{\max}}|j'\rangle\langle j'|\right] = \sum_{j,j'=1}^{n_{\max}}|j\rangle\langle j|\hat{V}|j'\rangle\langle j'| \approx \sum_{j,j'=1}^{n_{\max}}|j\rangle\delta_{jj'}W(x_j)\langle j'| = \sum_{j=1}^{n_{\max}}|j\rangle W(x_j)\langle j|. \quad (4.26)$$

```
In[440]:= Wgrid = Map[W, xgrid];            (* the potential on the computational grid *)
In[441]:= VP = SparseArray[Band[{1,1}]->Wgrid]; (* potential operator, position basis *)
In[442]:= VM = FourierDST[VP, 1];           (* potential operator, momentum basis *)
```

4.1.5 the kinetic-energy operator

The representation of the kinetic energy operator can be calculated very accurately with the description given above. We transform the definition of In[423] to the finite-resolution position basis with

```
In[443]:= TP = FourierDST[TM, 1];   (* kinetic operator, position basis *)
```

For large $n_{\max}$ and small excitation energies the exact kinetic-energy operator can be replaced by the position-basis form

$$\langle j|\hat{T}|j'\rangle \approx \frac{\hbar^2}{2m\Delta^2} \times \begin{cases} 2 & \text{if } j = j', \\ -1 & \text{if } |j-j'| = 1, \\ 0 & \text{if } |j-j'| \geq 2, \end{cases} \quad (4.27)$$

which corresponds to replacing the second derivative in the kinetic operator by the finite-differences expression $\psi''(x) \approx -\left[2\psi(x) - \psi(x-\Delta) - \psi(x+\Delta)\right]/\Delta^2$. While Equation (4.27) looks simple, it is ill suited for the calculations that will follow because (i) any matrix exponentials involving $\hat{T}$ will be difficult to calculate, and (ii) it is not very accurate (higher-order finite-differences expressions[6] are not much better). Thus we will not be using such approximations in what follows, and prefer the more useful and more accurate definition through `In[423]` and `In[443]`.

4.1.6 the momentum operator

The discussion has so far been conducted in terms of the kinetic-energy operator $\hat{T} = \hat{p}^2/(2m)$ without explicitly talking about the momentum operator $\hat{p} = -\mathrm{i}\hbar\frac{\mathrm{d}}{\mathrm{d}x}$. This was done because the matrix representation of the momentum operator is problematic. A direct calculation of the matrix elements in the momentum basis yields

$$\langle n|\hat{p}|n'\rangle = -\mathrm{i}\hbar\int_0^a \mathrm{d}x\phi_n(x)\frac{\mathrm{d}\phi_{n'}(x)}{\mathrm{d}x} = \frac{\hbar}{a}\times\begin{cases}\frac{4\mathrm{i}nn'}{n'^2-n^2} & \text{if } n-n' \text{ is odd,}\\ 0 & \text{if } n-n' \text{ is even.}\end{cases} \tag{4.28}$$

In Mathematica, this is implemented with the definition

```
In[444]:= pM = SparseArray[{n1_,n2_}/;OddQ[n1-n2]->(4*I*ħ*n1*n2)/(a*(n2^2-n1^2)),
                          {nmax,nmax}];   (* momentum operator, momentum basis *)
In[445]:= pP = FourierDST[pM, 1];         (* momentum operator, position basis *)
```

This result is, however, unsatisfactory, since (i) it generates a matrix that is not sparse, and (ii) for a finite basis size $n \le n_{\max} < \infty$ it does not exactly generate the kinetic-energy operator $\hat{T} = \hat{p}^2/(2m)$ (see **Q4.3**). We will avoid using the momentum operator whenever possible, and use the kinetic-energy operator $\hat{T}$ instead (see above). An example of the direct use of $\hat{p}$ is given in section 5.2.

For large $n_{\max}$ and small excitation energies the exact momentum operator can be replaced by the position-basis form

$$\langle j|\hat{p}|j'\rangle \approx \frac{\mathrm{i}\hbar}{2\Delta}\times\begin{cases}-1 & \text{if } j-j' = -1,\\ +1 & \text{if } j-j' = +1,\\ 0 & \text{if } |j-j'| \neq 1,\end{cases} \tag{4.29}$$

which corresponds to replacing the first derivative in the momentum operator by the finite-differences expression $\psi'(x) \approx \left[\psi(x+\Delta) - \psi(x-\Delta)\right]/(2\Delta)$. While Equation (4.29) looks simple, it is ill suited for the calculations that will follow because any matrix exponentials involving $\hat{p}$ will still be difficult to calculate; further, the same finite-differences caveats as in section 4.1.5 apply. Thus we will not be using such approximations in what follows, and prefer the more accurate definition through `In[444]`.

exercises

Q4.3 Using $n_{\max} = 100$, calculate the matrix representations of the kinetic-energy operator $\hat{T}$ and the momentum operator $\hat{p}$ in the momentum basis. Compare the spectra of $\hat{T}$ and $\hat{p}^2/(2m)$ and notice the glaring differences, even at low energies. *Hint:* use natural units such that `a=m=ħ=1` for simplicity.

Q4.4 Using $n_{\max} = 20$, calculate the matrix representations of the position operator $\hat{x}$ and the momentum operator $\hat{p}$ in the momentum basis. To what extent is the commutation relation $[\hat{x},\hat{p}] = \mathrm{i}\hbar$ satisfied? *Hint:* use natural units such that `a=m=ħ=1` for simplicity.

4.1.7 example: gravity well [code]

As an example of a single particle moving in one spatial dimension, we study the gravity well. This problem can be solved analytically, which helps us to determine the accuracy of our numerical methods.

We assume that a particle of mass m is free to move in the vertical direction x, where $x = 0$ is the earth's surface and $x > 0$ is up; the particle is forbidden from travelling below the earth's surface (*i.e.*, it

[6] See https://en.wikipedia.org/wiki/Finite_difference_coefficient for explicit forms of higher-order finite-differences expressions that can be used to approximate derivatives.

is restricted to $x > 0$ at all times). There is no dissipation or friction. The Hamiltonian of the particle's motion is

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + mg\hat{x}. \tag{4.30}$$

The wavefunction $\psi(x)$ of this particle must satisfy the boundary condition $\psi(x) = 0\ \forall x \leq 0$.

In what follows we use the length unit $\mathcal{L} = \left(\frac{\hbar^2}{m^2 g}\right)^{1/3}$, which is proportional to the size of the ground state of Equation (4.30), as well as the mass unit $\mathcal{M} = m$. Following section 4.1.1 we then define the time unit $\mathcal{T} = \mathcal{M}\mathcal{L}^2/\hbar = \left(\frac{\hbar}{mg^2}\right)^{1/3}$ and the energy unit $\mathcal{E} = \hbar/\mathcal{T} = (mg^2\hbar^2)^{1/3}$. These natural units lead to simple expressions for the mass: $\mathtt{m} = \frac{m}{\mathcal{M}} = 1$, $\hbar = \frac{\hbar}{\mathcal{E}\mathcal{T}} = 1$, and $\mathtt{g} = \frac{g}{\mathcal{L}/\mathcal{T}^2} = 1$. As a result, we can set up the Hamiltonian (4.30) without fixing the particle's mass and gravitational acceleration explicitly:

```
In[446]:= m = ħ = g = 1;
```

Other systems of units can be used in the same way by using the tools of section 4.1.1: first define a consistent set of units, and then express the physical quantities in terms of these units. The gravitational acceleration in particular would be set with

```
In[447]:= g = Quantity["StandardAccelerationOfGravity"]/(LengthUnit/TimeUnit^2);
```

analytic quantum energy eigenstates

The exact normalized eigenstates and associated energy eigenvalues of Equation (4.30) are

$$\psi_k(x) = \begin{cases} \left(\frac{2m^2 g}{\hbar^2}\right)^{1/6} \cdot \frac{\mathrm{Ai}\left[\alpha_k + x\cdot\left(\frac{2m^2 g}{\hbar^2}\right)^{1/3}\right]}{\mathrm{Ai}'(\alpha_k)} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \qquad E_k = -\alpha_k \cdot \left(\frac{mg^2\hbar^2}{2}\right)^{1/3} \tag{4.31}$$

for $k = 1, 2, 3, \ldots$, where $\mathrm{Ai}(z) =$ `AiryAi[z]` is the Airy function, $\mathrm{Ai}'(z)$ its first derivative, and $\alpha_k =$ `AiryAiZero[k]` its zeros: $\alpha_1 \approx -2.33811$, $\alpha_2 \approx -4.08795$, $\alpha_3 \approx -5.52056$, etc.

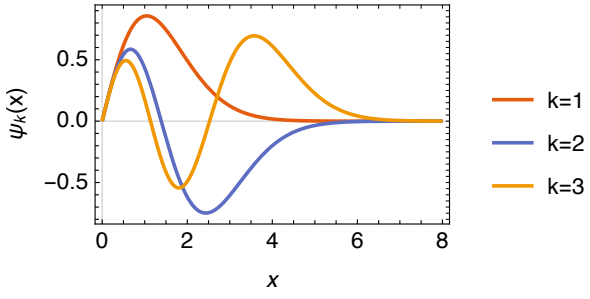
For comparison to numerical calculations below, we define the exact eigenstates and eigen-energies with

```
In[448]:= ψ[k_,x_] = (2*m^2*g/ħ^2)^(1/6)*AiryAi[AiryAiZero[k]+x*(2*m^2*g/ħ^2)^(1/3)]/
                     AiryAi'[AiryAiZero[k]];
In[449]:= ε[k_] = -AiryAiZero[k]*(m*g^2*ħ^2/2)^(1/3);
```

The ground-state energy is, in our chosen energy unit $\mathcal{E}$,

```
In[450]:= N[ε[1]]
Out[450]= 1.85576
```

The lowest three energy eigenstates look thus:



numerical solution (I): momentum basis

Our first numerical attempt to find the ground state of the gravity well relies on the momentum basis of states $|n\rangle$. For this approach we treat the Hamiltonian of the problem in the same way as discussed in chapter 2 and chapter 3: we express each term of the Hamiltonian as a matrix in a fixed basis set.

Since our calculation will take place in a finite box $x \in [0, a]$, we must choose the box size a large enough to contain most of the ground-state probability if we want to calculate it accurately. For the present calculation we choose $a = 10\mathcal{L}$, which is sufficient (see figure above) since we picked the length unit $\mathcal{L}$ similar to the ground-state size:

```
In[451]:= a = 10;
```

We only use a small number of basis functions here, to illustrate the method:

```
In[452]:= nmax = 12;
```

The matrix elements of the kinetic energy are set up following In[423]:

```
In[453]:= TM = SparseArray[Band[{1,1}] -> Range[nmax]^2*π^2*ℏ^2/(2*m*a^2)];
```

The matrix elements of the potential energy of Equation (4.30) are, from In[434],

```
In[454]:= xM = SparseArray[{
              Band[{1,1}] -> a/2,
              {n1_,n2_} /; OddQ[n1-n2] -> -8*a*n1*n2/(π^2*(n1^2-n2^2)^2)},
            {nmax,nmax}];
In[455]:= VM = m*g*xM;
```

The full Hamiltonian in the momentum representation is therefore

```
In[456]:= HM = TM + VM;
```

and the ground-state energy and wavefunction coefficients in the momentum representation

```
In[457]:= gsM = -Eigensystem[-N[HM], 1,
          Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}];
```

The ground state energy is

```
In[458]:= gsM[[1, 1]]
Out[458]= 1.85608
```

very close to the exact result of Out[450].

The ground state wavefunction is defined as a sum over basis functions,

```
In[459]:= φ[n_, x_] = Sqrt[2/a]*Sin[n*π*x/a];
In[460]:= ψ0[x_] = gsM[[2,1]] . Table[φ[n, x], {n, nmax}];
```

We can calculate the overlap of this numerical ground state with the exact one given in In[448], $|\langle\psi_0|\psi_1\rangle|^2$:

```
In[461]:= Abs[NIntegrate[ψ0[x]*ψ[1,x], {x, 0, a}]]^2
Out[461]= 0.999965
```

Even for $n_{\max} = 12$ this overlap is already very close to unity in magnitude. It quickly approaches unity as $n_{\max}$ increases, with the mismatch decreasing as $n_{\max}^{-9}$ for this specific system. The numerically calculated ground-state energy approaches the exact result from above, with the mismatch decreasing as $n_{\max}^{-7}$ for this specific system. These convergence properties, discussed in section 2.1.1, are very general and allow us to extrapolate many quantities to $n_{\max} \to \infty$ by polynomial fits of numerically calculated quantities as functions of $n_{\max}$.

numerical solution (II): mixed basis

The numerical method outlined above only works because we have an analytic expression for the matrix elements of the potential operator $\hat{V} = mg\hat{x}$, given in Equation (4.22). For a more general potential, the method of Equation (4.26) is more useful, albeit less accurate. Here we re-do the numerical ground-state calculation in the position basis. The computation is set up in the same way as above,

```
In[462]:= a = 10;
In[463]:= nmax = 12;
In[464]:= Δ = a/(nmax+1);          (* grid spacing *)
In[465]:= xgrid = Range[nmax]*Δ; (* the computational grid *)
```

The matrix elements of the kinetic-energy operator in the position basis are calculated with a discrete sine transform,

```
In[466]:= TM = SparseArray[Band[{1,1}] -> Range[nmax]^2*π^2*ℏ^2/(2*m*a^2)];
In[467]:= TP = FourierDST[TM, 1];
```

The matrix elements of the potential energy of in Equation (4.30) are, from In[440],

```
In[468]:= W[x_] = m*g*x;          (* the potential function *)
In[469]:= Wgrid = Map[W, xgrid];(* the potential on the computational grid *)
In[470]:= VP = SparseArray[Band[{1,1}] -> Wgrid];
```

The full Hamiltonian in the position representation is therefore

```
In[471]:= HP = TP + VP;
```

and the ground-state energy and wavefunction coefficients in the position representation

```
In[472]:= gsP = -Eigensystem[-N[HP], 1,
            Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}];
```

The ground state energy is now less close to the exact value than before, due to the additional approximation of Equation (4.26):

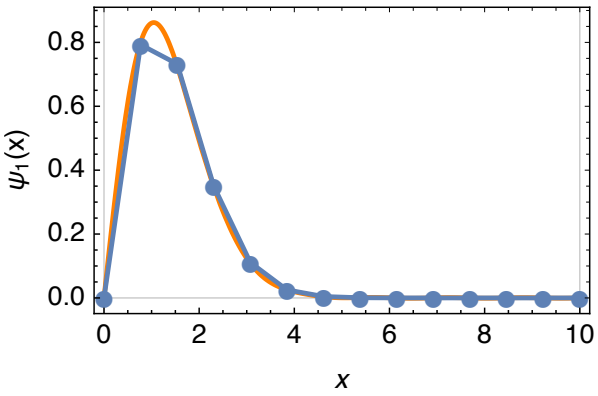
```
In[473]:= gsP[[1, 1]]
Out[473]= 1.86372
```

We therefore need a larger $n_{\max}$ to achieve the same accuracy as in the first numerical calculation. The great advantage of the present calculation is, however, that it is easily generalized to arbitrary potential-energy functions in In[468].

As shown in In[428], the wavefunction can be plotted approximately with

```
In[474]:= γ = Join[{{0,0}}, Transpose[{xgrid, gsP[[2,1]]/Sqrt[Δ]}], {{a,0}}];
In[475]:= ListLinePlot[γ]
```

where we have "manually" added the known boundary values $\gamma(0) = \gamma(a) = 0$ to the list of numerically calculated wave-function values.



You can see that even with $n_{\max} = 12$ grid points this ground-state wavefunction (blue lines interpolating between blue calculated points) looks remarkably close to the exact one (orange line, see plot on page 80).

If we need to go beyond linear interpolation, the precise wavefunction is calculated by converting to the momentum representation as in In[431] and multiplying with the basis functions as in In[460]:

```
In[476]:= φ[n_, x_] = Sqrt[2/a]*Sin[n*π*x/a];
In[477]:= ψ0[x_] = FourierDST[gs[[2,1]],1] . Table[φ[n, x], {n, nmax}];
```

exercises

Q4.5 What is the probability to find the particle below x = 1 (*i.e.*, below $x = \mathcal{L}$) when it is in the ground state of the gravity well, Equation (4.30)? Calculate analytically, with numerical method I, and with numerical method II.

Q4.6 Calculate the mean height $\langle \hat{x} \rangle$ in the ground state of the gravity well. How large is this quantity for a neutron in earth's gravitational field? *Hint:* see *Quantum states of neutrons in the Earth's gravitational field* by Valery V. Nesvizhevsky *et al.*, Nature 415, pages 297–299 (2002).

Q4.7 Calculate the energy levels and energy eigenstates of a particle in a harmonic potential, described by the Hamiltonian

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \frac{1}{2}m\omega^2\hat{x}^2. \tag{4.32}$$

Do the calculated energy levels match the analytically known values? *Hint:* use the system of units given in In[409]ff with a length unit of $\mathcal{L} = \sqrt{\hbar/(m\omega)}$, a mass unit $\mathcal{M} = m$, and an energy unit $\mathcal{E} = \hbar\omega$ (*i.e.*, the natural units). Choose the calculation box with size $a = 10\mathcal{L}$ and shift the minimum of the harmonic potential to the center of the calculation box.

4.1.8 the Wigner quasi-probability distribution [code]

The Wigner quasi-probability distribution[7] of a wavefunction $\psi(x)$ is a real-valued distribution in phase space defined as

$$W(x,k) = \frac{1}{\pi}\int_{-\infty}^{\infty} \mathrm{d}y \psi(x-y)\psi^*(x+y)e^{2iky}, \tag{4.33}$$

where $k = p/\hbar$ is the wavenumber, closely related to the momentum but in units of inverse length. W often makes it easier to interpret wavefunctions than simply plotting $\psi(x)$, especially when $\psi(x)$ is complex-valued. Time-dependent wavefunctions are often plotted as Wigner distribution movies, which makes it easier to track a particle as it moves through phase space. In the classical limit, the time-dependent Wigner distribution becomes the classical phase-space density that satisfies the Liouville equation.

For a quick and easy evaluation of the Wigner distribution, we approximate the wavefunction as piecewise constant, using Equation (4.13): $\psi(x) \approx \psi(x_{[x/\Delta]}) = v_{[x/\Delta]}/\sqrt{\Delta}$, where we have used the calculation grid spacing $\Delta = a/(n_{\max}+1)$ and the nearest-integer rounding function $[z] = \mathrm{round}(z)$. This approximation

[7] See https://en.wikipedia.org/wiki/Wigner_quasiprobability_distribution.

will be valid as long as $|k| \ll \pi/\Delta$. Inserting it into Equation (4.33), and assuming that $x = x_j = j\Delta$ is a grid point (*i.e.*, we will only sample the Wigner function on the spatial grid of the calculation), we can split the integral over y into integrals over segments of length Δ,

$$\begin{aligned} W(x_j,k) &= \frac{1}{\pi}\sum_{m=-\infty}^{\infty}\int_{-\Delta/2}^{\Delta/2} \mathrm{d}y\psi[x_j-(m\Delta+y)]\psi^*[x_j+(m\Delta+y)]e^{2ik(m\Delta+y)} \\ &\approx \frac{1}{\pi}\sum_{m=-\infty}^{\infty}\int_{-\Delta/2}^{\Delta/2} \mathrm{d}y\psi(x_{j-m})\psi^*(x_{j+m})e^{2ik(m\Delta+y)} = \frac{1}{\pi}\sum_{m=-\infty}^{\infty}\underbrace{\psi(x_{j-m})}_{\frac{v_{j-m}}{\sqrt{\Delta}}}\underbrace{\psi^*(x_{j+m})}_{\frac{v^*_{j+m}}{\sqrt{\Delta}}}e^{2ikm\Delta}\underbrace{\int_{-\Delta/2}^{\Delta/2}\mathrm{d}y e^{2iky}}_{\sin(k\Delta)/k} \\ &= \frac{\mathrm{sinc}(k\Delta)}{\pi}\sum_{m=-\min(j-1,n_{\max}-j)}^{\min(j-1,n_{\max}-j)} v_{j-m}v^*_{j+m}e^{2ikm\Delta}, \end{aligned} \tag{4.34}$$

where $\mathrm{sinc}(z) = \sin(z)/z$. The following Mathematica code converts a coefficient vector $\vec{\boldsymbol{v}}$ of length $n_{\max}$ into a function of the dimensionless momentum $\kappa = a \cdot k$ that calculates $W(x_j, k)$ for every grid point $j = 0, 1, \ldots, n_{\max}+1$ (including the boundary grid points that are usually left out of our calculations):

```
In[478]:=WignerDistribution[v_?VectorQ] := With[{nmax = Length[v]},
          Function[κ, Evaluate[Sinc[κ/(nmax+1)]/π*Table[
            Sum[v[[j-m]]*Conjugate[v[[j+m]]]*Exp[2*I*κ*m/(nmax+1)],
            {m,-Min[j-1,nmax-j],Min[j-1,nmax-j]}]//Re//ComplexExpand, {j,0,nmax+1}]]]
```

Notice that this function `WignerDistribution` returns an anonymous function (see section 1.5.3) of one parameter, which in turn returns a list of values. As an example of its use, we make a 2D plot of the Wigner distribution on the interval $x \in [x_{\min}, x_{\max}]$:[8]

```
In[479]:=WignerDistributionPlot[Y_,
              {xmin_?NumericQ, xmax_?NumericQ} /; xmax > xmin] :=
        Module[{nmax, qmax, w, W},
          (* number of grid points *)
          nmax = Length[Y];
          (* calculate the Wigner distribution *)
          w = WignerDistribution[Y];
          (* evaluate it on the natural dimensionless momentum grid *)
          qmax = Floor[nmax/2];
          W = Table[w[q*π], {q, -qmax, qmax}];
          (* make a plot *)
          ArrayPlot[W, FrameTicks->Automatic, AspectRatio -> 1/GoldenRatio,
            DataRange->{{xmin,xmax},qmax*π/(xmax-xmin)*{-1,1}},
            ColorFunctionScaling -> False,
            ColorFunction -> (Blend[{Blue, White, Red}, (π*#+1)/2]&)]]
```

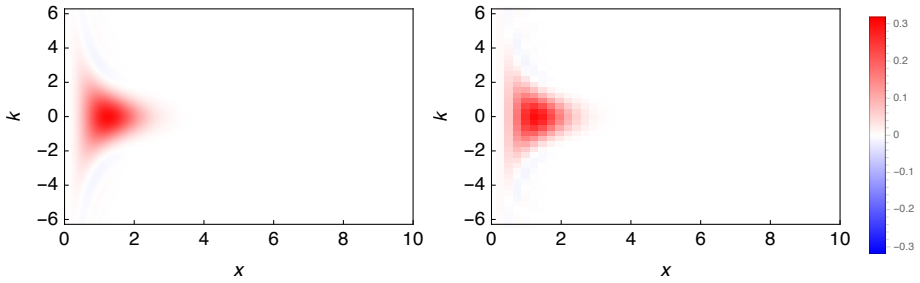
Notice that we evaluate the Wigner distribution only up to momenta $\pm n_{\max}\pi/(2a)$, which is the Nyquist limit in this finite-resolution system.[9] The color scheme is chosen such that the Wigner distribution values range $[-\frac{1}{\pi}, +\frac{1}{\pi}]$ is mapped onto the colors blended from blue, white, and red, such that negative Wigner values are shown in shades of blue while positive values are shown in shades of red.

As an example, we plot the Wigner distribution of the numerical ground-state wavefunction shown on page 82: on the left, the exact distribution from Equation (4.33); on the right, the grid evaluation of `In[478]` and `In[479]` (calculated with $n_{\max} = 40$) with

```
In[480]:=WignerDistributionPlot[gsP[[2, 1]], {0, a}]
```

[8] This procedure works for situations other than the usual $x_{\min} = 0$ and $x_{\max} = a$.

[9] See https://en.wikipedia.org/wiki/Nyquist_frequency.



extension to density operators

If the state of the system is not pure, but given as a density matrix $\rho(x,x') = \langle x|\hat{\rho}|x'\rangle$ instead of as a wavefunction $\psi(x) = \langle x|\psi\rangle$, then we do not have the option of plotting the wavefunction and we can only resort to the Wigner distribution for a graphical representation.

Noticing that Equation (4.33) contains the term

$$\psi(x-y)\psi^*(x+y) = \langle x-y|\psi\rangle\langle\psi|x+y\rangle = \langle x-y|\hat{\rho}|x+y\rangle = \rho(x-y,x+y) \tag{4.35}$$

in terms of the pure-state density operator $\hat{\rho} = |\psi\rangle\langle\psi|$, the definition of the Wigner distribution is generalized to

$$W(x,k) = \frac{1}{\pi}\int_{-\infty}^{\infty} \mathrm{d}y\rho(x-y,x+y)e^{2\mathrm{i}ky}. \tag{4.36}$$

We can make the same approximations as in Equation (4.34) to calculate the Wigner function on a spatial grid point $x = x_j$ [see Equation (4.15)]:

$$\begin{aligned} W(x_j,k) &= \frac{1}{\pi}\sum_{m=-\infty}^{\infty}\int_{-\Delta/2}^{\Delta/2}\mathrm{d}y\rho[x_j-(m\Delta+y),x_j+(m\Delta+y)]e^{2\mathrm{i}k(m\Delta+y)} \\ &\approx \frac{1}{\pi}\sum_{m=-\infty}^{\infty}\int_{-\Delta/2}^{\Delta/2}\mathrm{d}y\rho(x_{j-m},x_{j+m})e^{2\mathrm{i}k(m\Delta+y)} \\ &= \frac{\mathrm{sinc}(k\Delta)}{\pi}\sum_{m=-\min(j-1,n_{\max}-j)}^{\min(j-1,n_{\max}-j)} R_{j-m,j+m}e^{2\mathrm{i}km\Delta}. \end{aligned} \tag{4.37}$$

In analogy to In[478] we define

```
In[481]:=WignerDistribution[R_ /; MatrixQ[R, NumericQ] &&
          Length[R] == Length[Transpose[R]]] :=
        With[{n = Length[R]},
          Function[k, Evaluate[Sinc[k/(n+1)]/π*Table[
            Sum[R[[j-m,j+m]]*Exp[2*I*k*m/(n+1)],
            {m,-Min[j-1,n-j],Min[j-1,n-j]}]//Re//ComplexExpand, {j,0,n+1}]]]]
```

For a pure state, the density matrix has the coefficients $R_{j,j'} = v_j v_{j'}^*$, and the definitions of In[478] and In[481] thus give exactly the same result if we use

```
In[482]:=R = KroneckerProduct[v, Conjugate[v]]
```

In addition, the 2D plotting function of In[479] also works when called with a density matrix as first parameter.

exercises

Q4.8 Plot the Wigner distribution of the first excited state of the gravity well. What do you notice?

4.1.9 1D dynamics in the square well [code]

Assume again a single particle of mass m moving in a one-dimensional potential, with the time-independent Hamiltonian given in Equation (4.1). The motion is again restricted to $x \in [0, a]$. We want to study the time-dependent wavefunction $\psi(x,t) = \langle x|\psi(t)\rangle$ given in Equation (2.34),

$$|\psi(t)\rangle = \exp\left[-\frac{\mathrm{i}(t-t_0)}{\hbar}\hat{\mathcal{H}}\right]|\psi(t_0)\rangle. \tag{4.38}$$

The simplest way of computing this propagation is to express the wavefunction and the Hamiltonian in a particular basis and use a matrix exponentiation to find the time dependence of the expansion coefficients of the wavefunction. For example, if we use the finite-resolution position basis, we have seen on page 82 how to find the matrix representation of the Hamiltonian, `HP`. For a given initial wavefunction represented by a position-basis coefficient vector `v0` we can then define

```
In[483]:= v[Δt_?NumericQ] := MatrixExp[-I*HP*Δt/ħ].v0
```

as the propagation over a time interval $\Delta t = t - t_0$. If you try this, you will see that calculating $|\psi(t)\rangle$ in this way is not very efficient, because the matrix exponentiation is a numerically difficult operation.

A much more efficient method can be found by using the Trotter expansion

$$e^{\lambda(X+Y)} = e^{\frac{\lambda}{2}X}e^{\lambda Y}e^{\frac{\lambda}{2}X} \times e^{\frac{\lambda^3}{24}[X,[X,Y]]+\frac{\lambda^3}{12}[Y,[X,Y]]} \times e^{-\frac{\lambda^4}{48}[X,[X,[X,Y]]]-\frac{\lambda^4}{16}[X,[Y,[X,Y]]]-\frac{\lambda^4}{24}[Y,[Y,[X,Y]]]}\ldots$$
$$\approx e^{\frac{\lambda}{2}X}e^{\lambda Y}e^{\frac{\lambda}{2}X}, \tag{4.39}$$

where the approximation is valid for small λ since the neglected terms are of third and higher orders in λ (notice that there is no second-order term in λ!). Setting $\lambda = -\frac{\mathrm{i}(t-t_0)}{M\hbar}$ for some large integer M, as well as $X = \hat{V}$ and $Y = \hat{T}$, we find

$$|\psi(t)\rangle = e^{M\lambda\hat{\mathcal{H}}}|\psi(t_0)\rangle = \left[e^{\lambda\hat{\mathcal{H}}}\right]^M|\psi(t_0)\rangle = \left[e^{\lambda(\hat{T}+\hat{V})}\right]^M|\psi(t_0)\rangle \overset{\text{Trotter Equation (4.39)}}{\downarrow} \lim_{M\to\infty}\left[e^{\frac{\lambda}{2}\hat{V}}e^{\lambda\hat{T}}e^{\frac{\lambda}{2}\hat{V}}\right]^M|\psi(t_0)\rangle. \tag{4.40}$$

This can be evaluated very efficiently. We express the potential Hamiltonian in the finite-resolution position basis, Equation (4.26), the kinetic Hamiltonian in the momentum basis, Equation (4.9), and the time-dependent wavefunction in both bases of Equation (4.17):

$$|\psi(t)\rangle = \sum_{n=1}^{n_{\max}} u_n(t)|n\rangle = \sum_{j=1}^{n_{\max}} v_j(t)|j\rangle \tag{4.41a}$$

$$\hat{V} \approx \sum_{j=1}^{n_{\max}} W(x_j)|j\rangle\langle j| \tag{4.41b}$$

$$\hat{T} \approx \sum_{n=1}^{n_{\max}} \frac{n^2\pi^2\hbar^2}{2ma^2}|n\rangle\langle n| \tag{4.41c}$$

The expansion coefficients of the wavefunction are related by a type-I discrete sine transform, see Equation (4.19), Equation (4.21), `In[430]`, and `In[431]`.

The great advantage of the diagonal matrices of Equation (4.41)b and Equation (4.41)c is that algebra with diagonal matrices is as simple as algebra with scalars, but applied to the diagonal elements one-by-one. In particular, for any diagonal matrix $D = \sum_j d_j|j\rangle\langle j|$ the integer matrix powers are $D^k = \sum_j d_j^k|j\rangle\langle j|$, and matrix exponentionals are calculated by exponentiating each diagonal element separately: $\exp(D) = \sum_{k=0}^{\infty} D^k/k! = \sum_{k=0}^{\infty}(\sum_j d_j^k|j\rangle\langle j|)/k! = \sum_j(\sum_{k=0}^{\infty} d_j^k/k!)|j\rangle\langle j| = \sum_j \exp(d_j)|j\rangle\langle j|$. As a result,

$$e^{\frac{\lambda}{2}\hat{V}} = \sum_{j=1}^{n_{\max}} e^{\frac{\lambda}{2}W(x_j)}|j\rangle\langle j|, \tag{4.42}$$

and the action of the potential Hamiltonian thus becomes straightforward:

$$e^{\frac{\lambda}{2}\hat{V}}|\psi(t)\rangle = \left[\sum_{j=1}^{n_{\max}} e^{\frac{\lambda}{2}W(x_j)}|j\rangle\langle j|\right]\left[\sum_{j'=1}^{n_{\max}} v_{j'}(t)|j'\rangle\right] = \sum_{j,j'=1}^{n_{\max}} e^{\frac{\lambda}{2}W(x_j)}v_{j'}(t)|j\rangle\underbrace{\langle j|j'\rangle}_{\delta_{jj'}} = \sum_{j=1}^{n_{\max}}\underbrace{\left[e^{\frac{\lambda}{2}W(x_j)}v_j(t)\right]}_{v_j'}|j\rangle, \tag{4.43}$$

which is an element-by-element multiplication of the coefficients of the wavefunction with the exponentials of the potential—no matrix operations are required. The expansion coefficients (position basis) after propagation with the potential Hamiltonian for a "time" step $\lambda/2$ are therefore

$$v'_j = e^{\frac{\lambda}{2}W(x_j)} v_j. \tag{4.44}$$

The action of the kinetic Hamiltonian in the momentum representation is found in exactly the same way:

$$e^{\lambda \hat{T}}|\psi(t)\rangle = \left[\sum_{n=1}^{n_{\max}} e^{\lambda \frac{n^2\pi^2\hbar^2}{2ma^2}} |n\rangle\langle n|\right]\left[\sum_{n'=1}^{n_{\max}} u_{n'}(t)|n'\rangle\right] = \sum_{n=1}^{n_{\max}} \underbrace{\left[e^{\lambda \frac{n^2\pi^2\hbar^2}{2ma^2}} u_n(t)\right]}_{u'_n} |n\rangle. \tag{4.45}$$

The expansion coefficients (momentum basis) after propagation with the kinetic Hamiltonian for a "time" step λ are therefore

$$u'_n = e^{\lambda \frac{n^2\pi^2\hbar^2}{2ma^2}} u_n. \tag{4.46}$$

We know that a type-I discrete sine transform brings the wavefunction from the finite-resolution position basis to the momentum basis and vice-versa. The propagation under the kinetic Hamiltonian thus consists of

1. a type-I discrete sine transform to calculate the coefficients $v_j \mapsto u_n$,
2. an element-by-element multiplication, Equation (4.46), to find the coefficients $u_n \mapsto u'_n$,
3. and a second type-I discrete sine transform to calculate the coefficients $u'_n \mapsto v'_j$.

Here we assemble all these pieces into a program that propagates a state $|\psi(t_0)\rangle$, which is given as a coefficient vector $\vec{\boldsymbol{v}}$ in the finite-resolution position basis, forward in time to $t = t_0 + \Delta t$. First, for reference, a procedure for the exact propagation by matrix exponentiation and matrix–vector multiplication, as in In[483]:

```
In[484]:= VP = SparseArray[Band[{1,1}]->Wgrid];
In[485]:= TM = SparseArray[Band[{1,1}]->Range[nmax]^2*π^2*ħ^2/(2*m*a^2)];
In[486]:= TP = FourierDST[TM, 1];
In[487]:= HP = TP + VP;
In[488]:= propExact[Δt_?NumericQ, v0_ /; VectorQ[v0, NumericQ]] :=
            MatrixExp[-I*HP*N[Δt/ħ]].v0
```

Next, an iterative procedure that propagates by M small steps via the Trotter approximation, Equation (4.39):

```
In[489]:= propApprox[Δt_?NumericQ, M_Integer /; M >= 1,
                     v0_ /; VectorQ[v0, NumericQ]] :=
            Module[{λ, Ke, Pe2, propKin, propPot2, prop},
              (* compute the λ constant *)
              λ = -I*N[Δt/(M*ħ)];
              (* compute the diagonal elements of exp[λ*T] *)
              Ke = Exp[λ*Range[nmax]^2*π^2*ħ^2/(2*m*a^2)];
              (* propagate by a full time-step with T *)
              propKin[v_] := FourierDST[Ke*FourierDST[v, 1], 1];
              (* compute the diagonal elements of exp[λ*V/2] *)
              Pe2 = Exp[λ/2*Wgrid];
              (* propagate by a half time-step with V *)
              propPot2[v_] := Pe2*v;
              (* propagate by a full time-step by H=T+V *)
              (* using the Trotter approximation *)
              prop[v_] := propPot2[propKin[propPot2[v]]];
              (* step-by-step propagation *)
              Nest[prop, v0, M]]
```

Notice that there are no basis functions, integrals, etc. involved in this calculation; everything is done in terms of the values of the wavefunction on the grid $x_1 \ldots x_{n_{\max}}$. This efficient method is called *split-step propagation*.

The `Nest` command "nests" a function call: for example, `Nest[f,x,3]` calculates $f(f(f(x)))$. We use this on line 18 of `In[489]` to repeatedly propagate by small time steps via the Trotter approximation. Since this algorithm internally calculates the wavefunction at all the intermediate times $t = t_0 + \frac{m}{M}(t - t_0)$ for $m = 1, 2, 3, \ldots, M$, we can modify our program in order to follow this time evolution. To achieve this we simply replace the `Nest` command with `NestList`, which is similar to `Nest` but returns all intermediate results: for example, `NestList[f,x,3]` returns the list $\{x, f(x), f(f(x)), f(f(f(x)))\}$. We replace last line of the code above with

```
18          Transpose[{Range[0, M]/M*Δt, NestList[prop, v0, M]}]]
```

which now returns a list of pairs containing (i) the time and (ii) the wavefunction at the corresponding time.

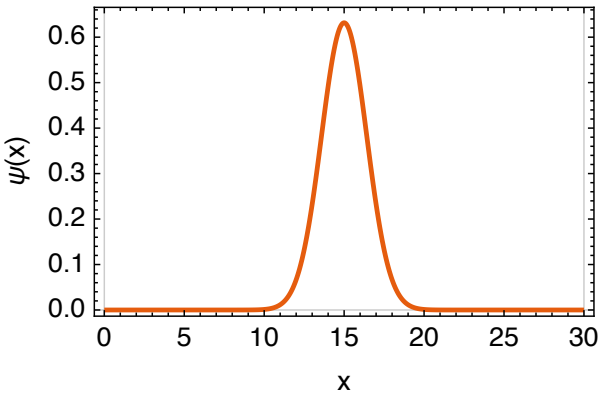
example: bouncing in the gravity well

As an example of particle dynamics, we return to the gravity well of section 4.1.7. Classically, if we drop a particle from height x_0 at $t = 0$ under the influence of gravity, then its trajectory is $x(t) = x_0 - \frac{1}{2}gt^2$, until it reaches the earth's surface ($x = 0$) at time $t_1 = \sqrt{2x_0/g}$. We plot this classical bouncing trajectory for a scaled starting height $x_0 = 15$ with

```
1  In[490]:=With[{x0 = 15, Δt = 50}, {t1 = Sqrt[2*x0/g]},
2           Plot[x0 - Mod[t, 2*t1, -t1]^2/2, {t, 0, Δt}]]
```

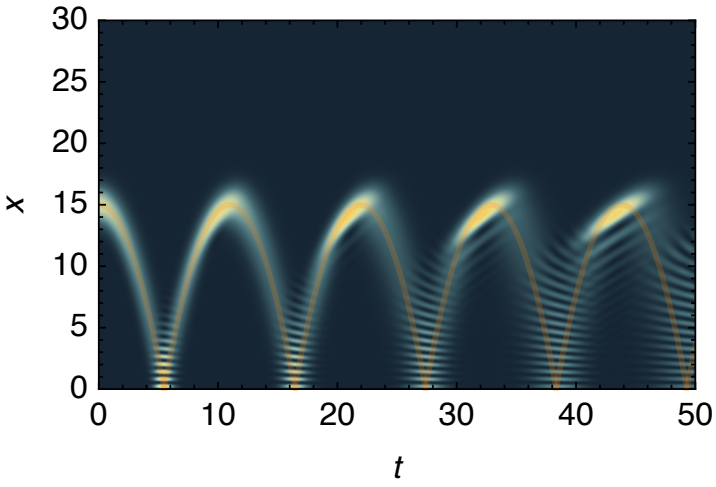
In order to simulate a quantum particle bouncing along this trajectory, we start at the same height $x_0 = 15$ but assume that the particle initially has a wavefunction of root-mean-square width $\sigma = 1$: the initial state in the position basis is

```
1  In[491]:=x0 = 15;                (* starting height *)
2  In[492]:=σ = 1;                  (* starting width *)
3  In[493]:=t1 = Sqrt[2*x0/g];      (* classical bounce time *)
4  In[494]:=vv = Normalize[N[Exp[-((xgrid-x0)/(2*σ))^2]]];  (* starting state *)
5  In[495]:=ListLinePlot[Join[{{0,0}},Transpose[{xgrid,vv/Sqrt[Δ]}],{{a,0}}],
6           PlotRange->All]
```



We propagate this particle in time for $\Delta t = 50$ time units, using $M = 1000$ time steps, and plot the time-dependent density $\rho(x, t) = |\psi(x, t)|^2 = |\langle x|\psi(t)\rangle|^2$ using the trick of Equation (4.13):

```
1  In[496]:=With[{Δt = 50, M = 1000},
2           ρ = ArrayPad[Abs[propApprox[Δt, M, vv][[All,2]]]^2/Δ, {{0, 0}, {1, 1}}];
3           ArrayPlot[Reverse[Transpose[ρ]], DataRange -> {{0, Δt}, {0, a}}]
```

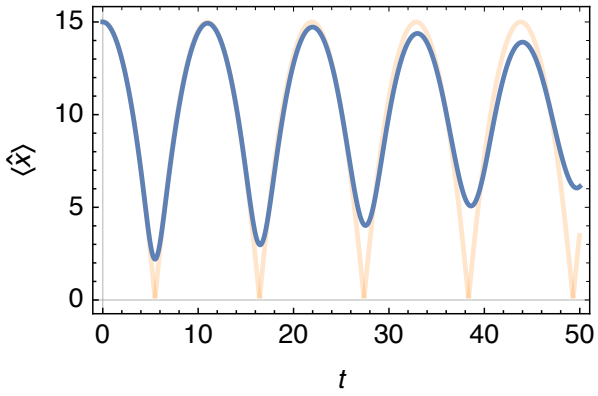


The orange overlaid curve shows the classical particle trajectory from In[490], which the quantum particle follows approximately while self-interfering during the reflections.

To study the correspondence between classical and quantum-mechanical motion more quantitatively, we can calculate and plot time-dependent quantities such as the time-dependent mean position: using Equation (4.24),

$$\langle \hat{x} \rangle(t) = \langle \psi(t) | \hat{x} | \psi(t) \rangle \approx \langle \psi(t) | \left[\sum_{j=1}^{n_{\max}} |j\rangle x_j \langle j| \right] |\psi(t)\rangle = \sum_{j=1}^{n_{\max}} x_j |v_j(t)|^2. \tag{4.47}$$

```
In[497]:= With[{Δt = 50, M = 1000},
            ListLinePlot[{#[[1]], Abs[#[[2]]]^2.xgrid} & /@ propApprox[Δt, M, vv]]]
```



Here the quantum deviations from the classical trajectory (orange) become apparent.

4.1.10 1D dynamics in a time-dependent potential

While the direct propagation of Equation (2.34) only works for time-independent Hamiltonians, the split-step method of In[489] can be extended to time-dependent Hamiltonians, in particular to time-dependent potentials $W(x, t)$. For this, we assume that the potential varies slowly enough in time that it is almost constant during a Trotter step $\Delta t/M$; this assumption usually becomes exact as $M \to \infty$.

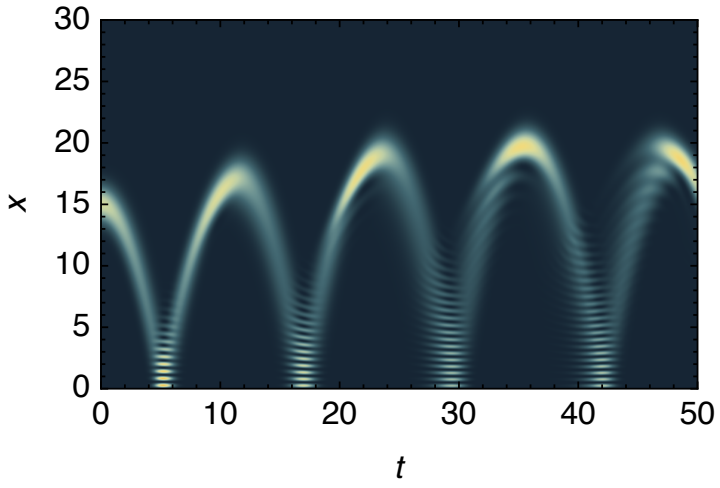
```
In[498]:= propApprox[Wt_,
                     Δt_?NumericQ, M_Integer /; M >= 1,
                     v0_ /; VectorQ[v0, NumericQ]] :=
           Module[{λ, Ke, propKin, propPot2, prop},
             (* compute the λ constant *)
```

```
6           λ = -I*N[Δt/(M*ħ)];
7           (* compute the diagonal elements of exp[λ*T] *)
8           Ke = Exp[λ*Range[nmax]^2*π^2/2];
9           (* propagate by a full time-step with T *)
10          propKin[v_] := FourierDST[Ke*FourierDST[v, 1], 1];
11          (* propagate by a half time-step with V *)
12          (* evaluating the potential at time t *)
13          propPot2[t_, v_] := Exp[λ/2*(Wt[#,t]&/@xgrid)]*v;
14          (* propagate by a full time-step by H=T+V *)
15          (* using the Trotter approximation *)
16          (* starting at time t *)
17          prop[v_, t_] := propPot2[t+3Δt/(4M), propKin[propPot2[t+Δt/(4M), v]]];
18          (* step-by-step propagation *)
19          Transpose[{Range[0, M]/M*Δt, FoldList[prop, v0, Range[0,M-1]/M*Δt]}]
```

- The definition of `propApprox` now needs a time-dependent potential `Wt[x,t]` that it can evaluate as the propagation proceeds. This potential must be specified as a pure function with two arguments, as in the example below.
- The exponentials for the potential propagation, calculated once-and-for-all on line 11 of `In[489]`, are now re-calculated in each call of the `propPot2` function.
- In the Trotter propagation step of Equation (4.40) we evaluate the potential twice in each propagation interval $[t, t + \Delta t/M]$: once at $t + \frac{1}{4}\Delta t/M$ for the first half-step with the potential operator $\hat{V}$, and once at $t + \frac{3}{4}\Delta t/M$ for the second half-step.
- On line 19 of `In[498]` we have replaced `NestList` by `FoldList`, which is more flexible: for example, `FoldList[f,x,{a,b,c}]` calculates the list $\{x, f(x,a), f(f(x,a),b), f(f(f(x,a),b),c)\}$. By giving the list of propagation interval starting times as the last argument of `FoldList`, the `prop` function is called repeatedly, with the current interval starting time as the second argument.

As an example, we calculate the time-dependent density profile under the same conditions as above, except that the gravitational acceleration is modulated periodically: $W(x,t) = W(x) \cdot [1 + A \cdot \sin(\omega t)]$. The oscillation frequency $\omega = \pi/t_1$ is chosen to drive the bouncing particle resonantly and enhance its amplitude. This time-dependent potential is passed as the first argument to `propApprox`:

```
In[499]:= With[{A = 0.1, ω = π/t1, Δt = 50, M = 1000},
            Wt[x_, t_] = W[x]*(1 + A*Sin[ω*t]);
            ρ = ArrayPad[Abs[propApprox[Wt,Δt,M,vv][[All,2]]]^2/Δ, {{0, 0}, {1, 1}}];
            ArrayPlot[Reverse[Transpose[ρ]], DataRange -> {{0, Δt}, {0, a}}]
```



The increase in bouncing amplitude can be seen clearly in this density plot.

exercises

Q4.9 Convince yourself that the Trotter expansion of Equation (4.39) is really necessary, *i.e.*, that $e^{X+Y} \neq e^X e^Y$ if X and Y do not commute. *Hint:* use two concrete non-commuting objects X and Y, for example two random 2×2 matrices as generated with `RandomReal[{0,1},{2,2}]`.

Q4.10 Given a particle moving in the range $x \in [0, a]$ with the Hamiltonian

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + W_0 \sin(10\pi x/a), \tag{4.48}$$

compute its time-dependent wavefunction starting from a "moving Gaussian" $\psi(t=0) \propto e^{-\frac{(x-a/2)^2}{4\sigma^2}} e^{\mathrm{i}kx}$ with $\sigma = 0.05a$ and $k = 100/a$. Study $\langle\hat{x}\rangle(t)$ using first $W_0 = 0$ and then $W_0 = 5000\frac{\hbar^2}{ma^2}$. *Hint:* use natural units such that `a=m=ħ=1` for simplicity.

4.2 Many particles in one dimension: dynamics with the non-linear Schrödinger equation

The advantage of the split-step evolution of Equation (4.40) becomes particularly clear when the system's energy depends on the wavefunction in a more complicated way than in the usual time-independent Schrödinger equation. A widely used example is the nonlinear energy functional[10]

$$E[\psi] = \underbrace{-\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x\psi^*(x)\psi''(x)}_{E_{\mathrm{kin}}[\psi]} + \underbrace{\int_{-\infty}^{\infty}\mathrm{d}x V(x)|\psi(x)|^2}_{E_{\mathrm{pot}}[\psi]} + \underbrace{\frac{\kappa}{2}\int_{-\infty}^{\infty}\mathrm{d}x|\psi(x)|^4}_{E_{\mathrm{int}}[\psi]}, \tag{4.49}$$

in which the last term describes the *mean-field interactions* between N particles that are all in wavefunction $\psi(x)$ (normalized to $\int_{-\infty}^{\infty}\mathrm{d}x|\psi(x)|^2 = 1$), and which are therefore in a joint product wavefunction $\psi(x)^{\otimes N}$ (see Equation (2.39)). Each particle sees a potential $V_{\mathrm{int}}(x) = \frac{\kappa}{2}|\psi(x)|^2$ generated by the average density $(N-1)|\psi(x)|^2$ of other particles with the same wavefunction, usually through collisional interactions. In three dimensions, the coefficient $\kappa = (N-1) \times 4\pi\hbar^2 a_s/m$ approximates the mean-field s-wave scattering between a particle and the $(N-1)$ other particles, with s-wave scattering length a_s (see section 4.4); in the present one-dimensional example, no such identification is made.

In order to find the ground state (energy minimum) of Equation (4.49) under the constraint of wavefunction normalization $\int_{-\infty}^{\infty}\mathrm{d}x|\psi(x)|^2 = 1$, we use the Lagrange multiplier[11] method: using the Lagrange multiplier μ called the *chemical potential*, we conditionally minimize the energy with respect to the wavefunction by setting its functional derivative[12]

$$\frac{\delta}{\delta\psi^*(x)}\left(E[\psi] - \mu\int_{-\infty}^{\infty}\mathrm{d}x|\psi(x)|^2\right) = -\frac{\hbar^2}{m}\psi''(x) + 2V(x)\psi(x) + 2\kappa|\psi(x)|^2\psi(x) - 2\mu\psi(x) = 0 \tag{4.50}$$

to zero. This yields the non-linear Schrödinger equation

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \underbrace{V(x) + \kappa|\psi(x)|^2}_{V_{\mathrm{eff}}(x)}\right]\psi(x) = \mu\psi(x), \tag{4.51}$$

also called the *Gross–Pitaevskii equation* for the description of dilute Bose–Einstein condensates. By analogy to the linear Schrödinger equation, it also has a time-dependent form for the description of Bose–Einstein condensate dynamics,

$$\mathrm{i}\hbar\frac{\partial\psi(x,t)}{\partial t} = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \underbrace{V(x,t) + \kappa|\psi(x,t)|^2}_{V_{\mathrm{eff}}(x,t)}\right]\psi(x,t). \tag{4.52}$$

[10] A functional is an operation that calculates a number from a given function. For example, $E[\psi] : L^2 \to \mathbb{R}$ converts a wavefunction $\psi \in L^2$ into an energy $E \in \mathbb{R}$. See https://en.wikipedia.org/wiki/Functional_(mathematics).

[11] See https://en.wikipedia.org/wiki/Lagrange_multiplier.

[12] See https://en.wikipedia.org/wiki/Functional_derivative.

For any $\kappa \neq 0$ there is no solution of the form of Equation (4.38). But the split-step method of Equation (4.40) can still be used to simulate Equation (4.52) because the wavefunction-dependent effective potential $V_{\text{eff}}(x,t)$ is still diagonal in the position representation. We extend the Mathematica code of In[498] by modifying the propPot2 method to include a non-linear term with prefactor κ (added as an additional argument to the propApprox function), and do not forget that the wavefunction at grid point x_j is $\psi(x_j) = v_j/\sqrt{\Delta}$:

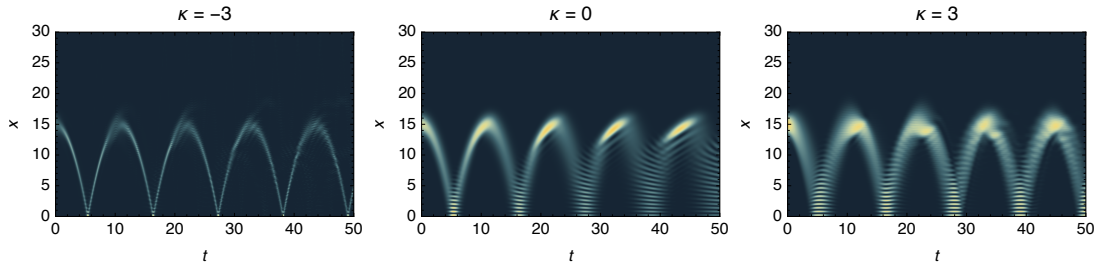
```
In[500]:= propApprox[Wt_, κ_?NumericQ, Δt_?NumericQ, M_Integer /; M >= 1,
                     v0_ /; VectorQ[v0, NumericQ]] :=
```

and

```
          propPot2[t_, v_] := Exp[λ/2*((Wt[#,t]&/@xgrid) + κ*Abs[v]^2/Δ)]*v;
```

As an example, we plot the time-dependent density for the time-independent gravitational well $W(x,t) = mgx$ and $\kappa = -3\cdot(g\hbar^4/m)^{1/3}$ (attractive interaction), $\kappa = 0$ (no interaction), $\kappa = +3\cdot(g\hbar^4/m)^{1/3}$ (repulsive interaction):

```
In[501]:= With[{κ = -3 * (g*ħ^4/m)^(1/3), Δt = 50, M = 10^3},
          ρ = ArrayPad[Abs[propApprox[W[#1]&,κ,Δt,M,vv][[All,2]]]^2/Δ,{{0,0},{1,1}}];
          ArrayPlot[Reverse[Transpose[ρ]], DataRange -> {{0, Δt}, {0, a}}]
```



Observations:

- The noninteractive case ($\kappa = 0$) shows a slow broadening and decoherence of the wavepacket.
- Attractive interactions ($\kappa < 0$) make the wavepacket collapse to a tight spot and bounce almost like a classical particle.
- Repulsive interactions ($\kappa > 0$) make the wavepacket broader, which slows down its decoherence.

exercises

Q4.11 Dimensionless problem (a=m=ħ=1): Given a particle moving in the range $x \in [0,1]$ with the non-linear Hamiltonian

$$\hat{\mathcal{H}} = -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + \underbrace{\Omega\left[\left(\frac{x-\frac{1}{2}}{\delta}\right)^2 - 1\right]^2}_{W(x)} + \kappa|\psi(x)|^2, \tag{4.53}$$

do the following calculations:

1. Plot the potential $W(x)$ for $\Omega = 1$ and $\delta = \frac{1}{4}$ (use $\kappa = 0$). What are the main characteristics of this potential? *Hint:* compute $W(\frac{1}{2})$, $W'(\frac{1}{2})$, $W(\frac{1}{2}\pm\delta)$, $W'(\frac{1}{2}\pm\delta)$.
2. Calculate and plot the time-dependent density $|\psi(x,t)|^2$ for $\Omega = 250$, $\delta = \frac{1}{4}$, and $\kappa = 0$, starting from $\psi_0(x) \propto \exp\left[-\left(\frac{x-x_0}{2\sigma}\right)^2\right]$ with $x_0 = 0.2694$ and $\sigma = 0.0554$. Calculate the probabilities for finding the particle in the left half ($x < \frac{1}{2}$) and in the right half ($x > \frac{1}{2}$) up to $t = 20$. What do you observe?
3. What do you observe for $\kappa = 0.5$? Why?

4.2.1 imaginary-time propagation for finding the ground state of the non-linear Schrödinger equation [code]

In the previous section we have looked at the dynamical evolution of a Bose–Einstein condensate with the time-dependent nonlinear Schrödinger equation (4.52), which could be performed with minimal modifications to previous Mathematica code. The time-independent nonlinear Schrödinger equation (4.51), on the other hand, seems at first sight inaccessible to our established methods: it is an operator eigenvalue equation, with the operator acting non-linearly on the wavefunction and thus invalidating the matrix diagonalization method of section 2.2. How can we determine the ground state of Equation (4.51)?

You may remember from statistical mechanics that at temperature T, the density operator of a system governed by a Hamiltonian $\hat{\mathcal{H}}$ is

$$\hat{\rho}(\beta) = \frac{e^{-\beta\hat{\mathcal{H}}}}{Z(\beta)} \tag{4.54}$$

with $\beta = 1/(k_\mathrm{B}T)$ the reciprocal temperature in terms of the Boltzmann constant $k_\mathrm{B} = 1.380\,648\,8(13) \times 10^{-23}\,\mathrm{J/K}$. The partition function $Z(\beta) = \mathrm{Tr}\, e^{-\beta\hat{\mathcal{H}}}$ ensures that the density operator has the correct norm, $\mathrm{Tr}\,\hat{\rho}(\beta) = 1$.

We know that at zero temperature the system will be in its ground state $|\gamma\rangle$,[13]

$$\lim_{\beta\to\infty} \hat{\rho}(\beta) = |\gamma\rangle\langle\gamma|. \tag{4.55}$$

If we multiply this equation by an arbitrary state $|\psi\rangle$ from the right, we find

$$\lim_{\beta\to\infty} \hat{\rho}(\beta)|\psi\rangle = |\gamma\rangle\langle\gamma|\psi\rangle. \tag{4.56}$$

Assuming that $\langle\gamma|\psi\rangle \neq 0$ (which is true for almost all states $|\psi\rangle$), the ground state is therefore

$$|\gamma\rangle = \frac{\lim_{\beta\to\infty}\hat{\rho}(\beta)|\psi\rangle}{\langle\gamma|\psi\rangle} = \frac{1}{\langle\gamma|\psi\rangle}\lim_{\beta\to\infty}\frac{1}{Z(\beta)} \times e^{-\beta\hat{\mathcal{H}}}|\psi\rangle. \tag{4.57}$$

This means that if we take almost any state $|\psi\rangle$ and calculate $\lim_{\beta\to\infty}\frac{1}{Z(\beta)}e^{-\beta\hat{\mathcal{H}}}|\psi\rangle$, we find a state that is proportional to the ground state (the prefactors $\frac{1}{\langle\gamma|\psi\rangle}$ and $\frac{1}{Z(\beta)}$ are merely scalar prefactors). But we already know how to do this: the wavefunction $e^{-\beta\hat{\mathcal{H}}}|\psi\rangle$ is calculated from $|\psi\rangle$ by *imaginary-time propagation*. In fact the split-step algorithm of section 4.1.9 remains valid if we replace $\mathrm{i}(t-t_0)/\hbar \mapsto \beta$. The advantage of Equation (4.57) over the matrix method of section 2.2 is that the former can be implemented even if the Hamiltonian depends on the wavefunction, as in Equation (4.51). The only caveat is that, while regular time propagation (section 4.1.9) is unitary, imaginary-time propagation is not. The wavefunction must therefore be re-normalized after each imaginary-time evolution step (with the `Normalize` function).

To implement this method of calculating the ground state by imaginary-time propagation, we set $\beta = M\cdot\delta\beta$ and modify Equation (4.57) to

$$|\gamma\rangle \propto \lim_{M\cdot\delta\beta\to\infty} e^{-M\,\delta\beta\,\hat{\mathcal{H}}}|\psi\rangle = \lim_{M\cdot\delta\beta\to\infty}\left[e^{-\delta\beta\,\hat{\mathcal{H}}}\right]^M|\psi\rangle \overset{\text{Trotter Equation (4.39)}}{\overset{\downarrow}{=}} \lim_{\delta\beta\to 0}\lim_{M\cdot\delta\beta\to\infty}\left[e^{-\frac{\delta\beta}{2}\hat{V}}e^{-\delta\beta\,\hat{T}}e^{-\frac{\delta\beta}{2}\hat{V}}\right]^M|\psi\rangle. \tag{4.58}$$

In practice we choose a small but finite "imaginary-time" step $\delta\beta$, and keep multiplying the wavefunction by $e^{-\frac{\delta\beta}{2}\hat{V}}e^{-\delta\beta\,\hat{T}}e^{-\frac{\delta\beta}{2}\hat{V}}$ until the normalized wavefunction no longer changes and the infinite-β limit ($M\cdot\delta\beta\to\infty$) has effectively been reached.

```
In[502]:= groundstate[g_?NumericQ, δβ_?NumericQ, tolerance_: 10^-10] :=
  Module[{Ke, propKin, propPot2, v0, γ},
    (* compute the diagonal elements of exp[-δβ*T] *)
    Ke = Exp[-δβ*Range[nmax]^2*π^2/2] //N;
    (* propagate by a full imaginary-time-step with T *)
    propKin[v_] := Normalize[FourierDST[Ke*FourierDST[v,1],1]];
    (* propagate by a half imaginary-time-step with V *)
    propPot2[v_] := Normalize[Exp[-δβ/2*(Wgrid + g*(nmax+1)*Abs[v]^2)]*v];
    (* propagate by a full imaginary-time-step by *)
```

[13] For simplicity we assume here that the ground state is non-degenerate.

```
10          (* H=T+V using the Trotter approximation *)
11          prop[v_] := propPot2[propKin[propPot2[v]]];
12          (* random starting point *)
13          v0 = Normalize@RandomComplex[{-1-I,1+I}, nmax];
14          (* propagation to the ground state *)
15          γ = FixedPoint[prop,v0,SameTest->Function[{v1,v2},Norm[v1-v2]<tolerance]];
16          (* return the ground-state coefficients *)
17          γ]
```

The last argument, `tolerance`, is optional and is given the default value 10^{-10} if not specified (see section 1.6.5). The `FixedPoint` function is used to apply the imaginary-time propagation until the result no longer changes (two consecutive results are considered equal if the function given as `SameTest` returns true when applied to these two results).

Multiplying Equation (4.51) by $\psi^*(x)$ and integrating over x gives

$$\mu = \underbrace{-\frac{\hbar^2}{2m}\int_{-\infty}^{\infty} \mathrm{d}x \psi^*(x)\psi''(x)}_{E_{\mathrm{kin}}[\psi]} + \underbrace{\int_{-\infty}^{\infty} \mathrm{d}x V(x)|\psi(x)|^2}_{E_{\mathrm{pot}}[\psi]} + \underbrace{g\int_{-\infty}^{\infty} \mathrm{d}x |\psi(x)|^4}_{2E_{\mathrm{int}}[\psi]}, \tag{4.59}$$

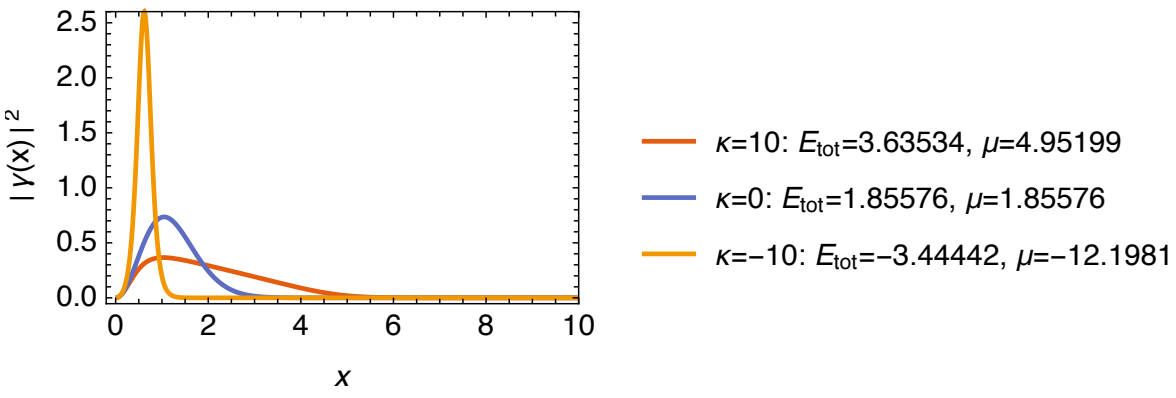
which is very similar to Equation (4.49) apart from a factor of two for E_{int}. We use this to calculate the total energy and the chemical potential in `In[502]` by replacing lines 16ff with

```
16          (* energy components *)
17          Ekin = π^2/2*Range[nmax]^2.Abs[FourierDST[γ,1]]^2;
18          Epot = Wgrid.Abs[γ]^2;
19          Eint = (g/2)(nmax+1)*Total[Abs[γ]^4];
20          (* total energy *)
21          Etot = Ekin + Epot + Eint;
22          (* chemical potential *)
23          μ = Ekin + Epot + 2*Eint;
24          (* return energy, chemical potential, coefficients *)
25          {Etot, μ, γ}]
```

and adding the local variables `Ekin`, `Epot`, `Eint`, `Etot`, and `μ` on line 2.

As an example we calculate the ground-state density for the gravity well of section 4.1.7 with three different values of the interaction strength κ [in units of $(g\hbar^4/m)^{1/3}$]:

```
In[503]:=With[{κ = 3 * (g*ħ^4/m)^(1/3), δβ = 10^-4},
          {Etot, μ, γ} = groundstate[δβ, κ];
          ListLinePlot[Join[{{0, 0}}, Transpose[{xgrid,Abs[γ]^2/Δ}], {{a, 0}}],
            PlotRange -> All, PlotLabel -> {Etot, μ}]]
```



Note that for $\kappa = 0$ the Gross–Pitaevskii equation is the Schrödinger equation, and the chemical potential is equal to the total energy, matching the exact result of `Out[450]`.

exercises

Q4.12 Dimensionless problem (a=m=ħ=1): Given a particle moving in the range $x \in [0,1]$ with the non-linear Hamiltonian

$$\hat{\mathcal{H}} = -\frac{1}{2}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + 2500\left(x-\frac{1}{2}\right)^2 + \kappa|\psi(x)|^2, \tag{4.60}$$

do the following calculations:

1. For $\kappa = 0$ calculate the exact ground state $|\zeta\rangle$ (assuming that the particle can move in the whole domain $x \in \mathbb{R}$) and its energy eigenvalue. *Hint:* assume $\zeta(x) = \exp\left[-\left(\frac{x-\frac{1}{2}}{2\sigma}\right)^2\right]/\sqrt{\sigma\sqrt{2\pi}}$ and find the value of σ that minimizes $\langle\zeta|\hat{\mathcal{H}}|\zeta\rangle$.
2. Calculate the ground state $\lim_{\beta\to\infty} e^{-\beta\hat{\mathcal{H}}}|\zeta\rangle$ and its chemical potential by imaginary-time propagation (with normalization of the wavefunction after each propagation step), using the code given above.
3. Plot the ground-state density for different values of κ.
4. Plot the total energy and the chemical potential as functions of κ.

4.3 several particles in one dimension: interactions

In section 4.2 we have studied a simple mean-field description of many-particle systems, with the advantage of simplicity and the disadvantage of not describing inter-particle correlations. Here we use a different approach that captures the full quantum mechanics of many-particle systems (including correlations), with the disadvantage of much increased calculation size.

We have seen in section 2.4.2 how to describe quantum-mechanical systems with more than one degree of freedom. This method can be used for describing several particles moving in one dimension. In the following we look at two examples of interacting particles.

When more than one particle is present in a system, we must distinguish between bosons and fermions. Whenever the Hamiltonian is symmetric under particle exchange (which is the case in this section), each one of its eigenstates can be associated with an irreducible representation of the particle permutation group. For two particles, the only available choices are the symmetric and the antisymmetric irreducible representations, and therefore every numerically calculated eigenstate can be labeled as either bosonic (symmetric) or fermionic (antisymmetric). For more particles, however, other irreducible representations exist,[14] meaning that some numerically calculated eigenstates of the Hamiltonian may not be physical at all because they are neither bosonic (fully symmetric) nor fermionic (fully antisymmetric).

4.3.1 two identical particles in one dimension with contact interaction [code]

We first look at two identical particles moving in a one-dimensional square well of width a and interacting through a contact potential $V_{\text{int}}(x_1,x_2) = \kappa \times \delta(x_1 - x_2)$. Such potentials are a good approximation of the s-wave scattering interactions taking place in cold dilute gases. The Hamiltonian of this system is[15]

$$\hat{\mathcal{H}} = \underbrace{-\frac{\hbar^2}{2m}\left[\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}\right]}_{\hat{T}} + \underbrace{V(x_1)+V(x_2)}_{\hat{V}} + \underbrace{\kappa\delta(x_1-x_2)}_{\hat{\mathcal{H}}_{\text{int}}}, \tag{4.61}$$

where $V(x)$ is the single-particle potential (as in section 4.1) and κ is the interaction strength, often related to the s-wave scattering length a_s. For the time being we do not need to specify whether the particles are bosons or fermions.

We describe this system with the tensor-product basis constructed from two finite-resolution position basis sets:

$$|j_1,j_2\rangle = |j_1\rangle \otimes |j_2\rangle \qquad \text{for } j_1,j_2 \in \{1,2,3,\ldots,n_{\max}\}. \tag{4.62}$$

[14] See https://en.wikipedia.org/wiki/Irreducible_representation.

[15] Notice that we write this Hamiltonian in an abbreviated form. The full operator form, with terms similar to Equation (4.2) but containing double integrals over space, is cumbersome to write (see Equation (4.63)).

Most of the matrix representations of the terms in Equation (4.61) are constructed as tensor products of the matrix representations of the corresponding single-particle representations since $\hat{T} = \hat{T}_1 \otimes \mathbb{1} + \mathbb{1} \otimes \hat{T}_2$ and $\hat{V} = \hat{V}_1 \otimes \mathbb{1} + \mathbb{1} \otimes \hat{V}_2$. The only new element is the interaction Hamiltonian $\hat{\mathcal{H}}_{\text{int}}$. Remembering that its formal operator definition is

$$\hat{\mathcal{H}}_{\text{int}} = \kappa \int_{-\infty}^{\infty} dx_1 dx_2 \Big[|x_1\rangle \otimes |x_2\rangle\Big] \delta(x_1 - x_2) \Big[\langle x_1| \otimes \langle x_2|\Big] \tag{4.63}$$

(while Equation (4.61) is merely a shorthand notation), we calculate its matrix elements in the finite-precision position basis with

$$\langle j_1, j_2|\hat{\mathcal{H}}_{\text{int}}|j_1', j_2'\rangle = \kappa \int_{-\infty}^{\infty} dx_1 dx_2 \langle j_1|x_1\rangle\langle j_2|x_2\rangle \delta(x_1 - x_2)\langle x_1|j_1'\rangle\langle x_2|j_2'\rangle = \kappa \int_0^a dx \vartheta_{j_1}(x)\vartheta_{j_2}(x)\vartheta_{j_1'}(x)\vartheta_{j_2'}(x). \tag{4.64}$$

These quartic overlap integrals can be calculated by a four-dimensional type-I discrete sine transform (see Equation (4.20) and **Q4.13**),

$$\int_0^a dx \vartheta_{j_1}(x)\vartheta_{j_2}(x)\vartheta_{j_3}(x)\vartheta_{j_4}(x) = \frac{1}{2a} \sum_{n_1,n_2,n_3,n_4=1}^{n_{\max}} X_{n_1 j_1} X_{n_2 j_2} X_{n_3 j_3} X_{n_4 j_4} \Big[\delta_{n_1+n_2,n_3+n_4} + \delta_{n_1+n_3,n_2+n_4} + \delta_{n_1+n_4,n_2+n_3} - \delta_{n_1,n_2+n_3+n_4} - \delta_{n_2,n_1+n_3+n_4} - \delta_{n_3,n_1+n_2+n_4} - \delta_{n_4,n_1+n_2+n_3}\Big], \tag{4.65}$$

which we evaluate in Mathematica very efficiently and all at once with

```
In[504]:= overlap4 = FourierDST[Table[KroneckerDelta[n1+n2,n3+n4]
          +KroneckerDelta[n1+n3,n2+n4]+KroneckerDelta[n1+n4,n2+n3]
          -KroneckerDelta[n1,n2+n3+n4]-KroneckerDelta[n2,n1+n3+n4]
          -KroneckerDelta[n3,n1+n2+n4]-KroneckerDelta[n4,n1+n2+n3],
          {n1,nmax},{n2,nmax},{n3,nmax},{n4,nmax}],1]/(2*a);
```

Mathematica code As before, we assume that the quantities a, m, and $\hbar$ are expressed in a suitable set of units (see section 4.1.1). First we define the grid size and the unit operator `id` acting on a single particle:

```
In[505]:= m = ħ = 1;    (* for example *)
In[506]:= a = 1;        (* for example *)
In[507]:= nmax = 50;    (* for example *)
In[508]:= Δ = a/(nmax+1);
In[509]:= xgrid = Range[nmax]*Δ;
In[510]:= id = IdentityMatrix[nmax, SparseArray];
```

The total kinetic Hamiltonian is assembled via a Kronecker product (tensor product) of the two single-particle kinetic Hamiltonians:

```
In[511]:= T1M = SparseArray[Band[{1,1}]->Range[nmax]^2*π^2*ħ^2/(2*m*a^2)];
In[512]:= T1P = FourierDST[T1M, 1];
In[513]:= TP = KroneckerProduct[T1P, id] + KroneckerProduct[id, T1P];
```

The same for the potential Hamiltonian (here we assume no potential, that is, a square well; but you may modify this):

```
In[514]:= W[x_] = 0;
In[515]:= Wgrid = W /@ xgrid;
In[516]:= V1P = SparseArray[Band[{1,1}]->Wgrid];
In[517]:= VP = KroneckerProduct[V1P, id] + KroneckerProduct[id, V1P];
```

The interaction Hamiltonian is constructed from In[504] with the ArrayFlatten command, which flattens the combination basis set $|j_1\rangle \otimes |j_2\rangle$ into a single basis set $|j_1, j_2\rangle$, or in other words, which converts the $n_{\max} \times n_{\max} \times n_{\max} \times n_{\max}$-matrix overlap4 into a $n_{\max}^2 \times n_{\max}^2$-matrix:

```
In[518]:=HintP = ArrayFlatten[overlap4];
```

The full Hamiltonian, in which the amplitude of the potential can be adjusted with the prefactor Ω and the interaction strength with g, is

```
In[519]:=HP[Ω_, κ_] = TP + Ω*VP + κ*HintP;
```

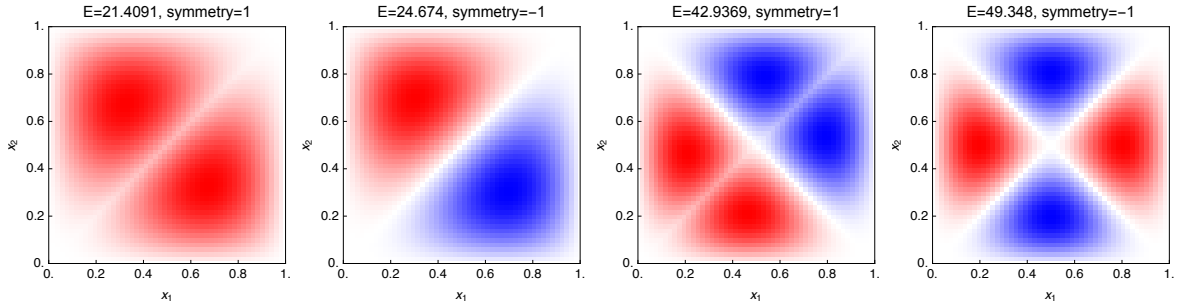
We calculate eigenstates (the ground state, for example) with the methods already described previously. The resulting wavefunctions are in the tensor-product basis of Equation (4.62), and they can be plotted with

```
In[520]:=plot2Dwf[ψ_] := Module[{ψ1,ψ2},
          (* make a square array of wavefunction values *)
          ψ1 = ArrayReshape[ψ, {nmax,nmax}];
          (* add a frame of zeros at the edges *)
          (* representing the boundary conditions *)
          ψ2 = ArrayPad[ψ1, 1];
          (* plot *)
          ArrayPlot[Reverse[Transpose[ψ2]]]
```

Assuming that a given wavefunction ψ is purely real-valued,[16] we can plot it with

```
In[521]:=plot2Dwf[v/Δ]
```

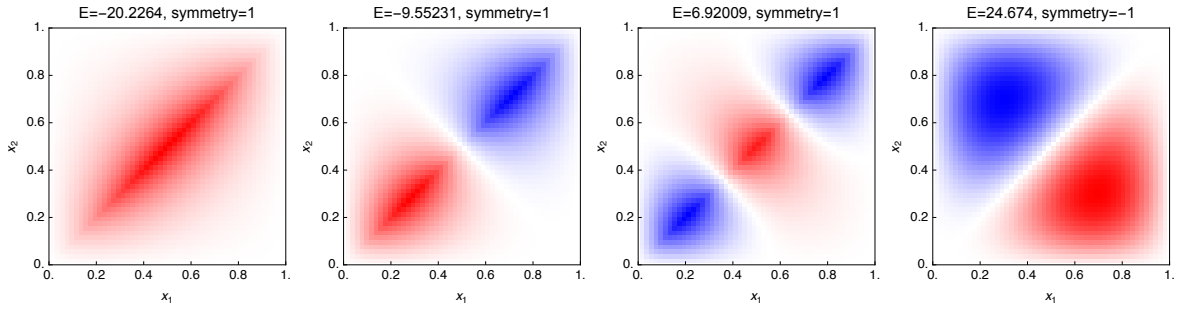
Here we plot the four lowest-energy wavefunctions for $\Omega = 0$ (no potential, the particles move in a simple infinite square well) and $\kappa = +25$ (repulsive interaction), using $n_{\max} = 50$ grid points, with the title of each panel showing the energy and the symmetry (see below). White corresponds to zero wavefunction, red is positive $\psi(x_1, x_2) > 0$, and blue is negative $\psi(x_1, x_2) < 0$.



We can see that in the ground state for $g > 0$ the particles avoid each other, *i.e.*, the ground-state wavefunction $\psi(x_1, x_2)$ is reduced whenever $x_1 = x_2$.

And here are the lowest four energy eigenstate wavefunctions for $\kappa = -10$:

[16] The eigenvectors of Hermitian operators can always be chosen to have real coefficients. Proof: Suppose that $\boldsymbol{H} \cdot \vec{\boldsymbol{\psi}} = E\vec{\boldsymbol{\psi}}$ for a vector $\vec{\boldsymbol{\psi}}$ with complex entries. Complex-conjugate the eigenvalue equation, $\boldsymbol{H}^\dagger \cdot \vec{\boldsymbol{\psi}}^* = E^*\vec{\boldsymbol{\psi}}^*$; but $\boldsymbol{H}^\dagger = \boldsymbol{H}$ and $E^* = E$, and hence $\vec{\boldsymbol{\psi}}^*$ is also an eigenvector of $\boldsymbol{H}$ with eigenvalue E. Thus we can introduce two real-valued vectors $\vec{\boldsymbol{\psi}}_\mathrm{r} = \vec{\boldsymbol{\psi}} + \vec{\boldsymbol{\psi}}^*$ and $\vec{\boldsymbol{\psi}}_\mathrm{i} = \mathrm{i}(\vec{\boldsymbol{\psi}} - \vec{\boldsymbol{\psi}}^*)$, representing the real and imaginary parts of $\vec{\boldsymbol{\psi}}$, respectively, which are both eigenvectors of $\boldsymbol{H}$ with eigenvalue E. Mathematica (as well as most other matrix diagonalization algorithms) automatically detect Hermitian matrices and return eigenvectors with real coefficients.



We can see that in the ground state for $\kappa < 0$ the particles attract each other, *i.e.*, the ground-state wavefunction $\psi(x_1, x_2)$ is increased whenever $x_1 = x_2$. We also notice that the second-lowest state for $\kappa = +25$ is exactly equal to the fourth-lowest state for $\kappa = -10$: its wavefunction vanishes whenever $x_1 = x_2$ and thus the contact interaction has no influence on this state.

In the above plots we have noted the symmetry of each eigenstate (symmetric or antisymmetric with respect to particle exchange), which is calculated with the integral

$$\mathcal{S}[\psi] = \int_0^a \mathrm{d}x_1 \mathrm{d}x_2 \psi^*(x_1, x_2)\psi(x_2, x_1) = \begin{cases} +1 & \text{for symmetric states } \psi(x_2, x_1) = \psi(x_1, x_2), \\ -1 & \text{for antisymmetric states } \psi(x_2, x_1) = -\psi(x_1, x_2). \end{cases} \tag{4.66}$$

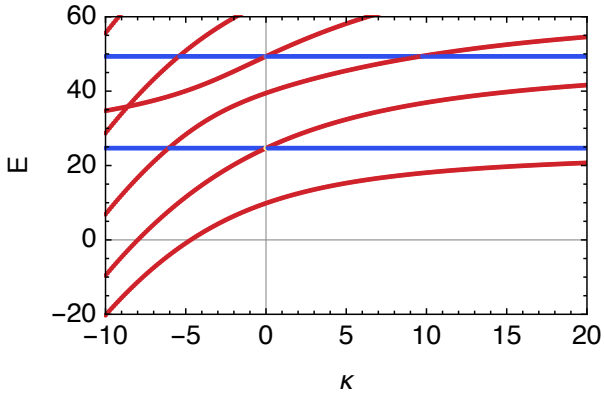
In Mathematica, the mirrored wavefunction $\psi(x_2, x_1)$ is calculated with the particle interchange operator $\hat{\Xi}$ defined as

```
In[522]:= Ξ = ArrayFlatten[SparseArray[{i_,j_,j_,i_} -> 1, {nmax, nmax, nmax, nmax}]];
```

such that $\psi(x_2, x_1) = \langle x_2, x_1|\psi\rangle = \langle x_1, x_2|\hat{\Xi}|\psi\rangle$. The symmetry of a state, defined in Equation (4.66), is therefore the expectation value of the $\hat{\Xi}$ operator:

```
In[523]:= symmetry[v_] := Re[Conjugate[v].(Ξ.v)]
```

Here we show the numerical energy eigenvalues of the contact interaction Hamiltonian, colored according to their symmetry: red dots indicate symmetric states ($\mathcal{S} = +1$), whereas blue dots indicate antisymmetric states ($\mathcal{S} = -1$).



In this representation it becomes even clearer that antisymmetric states are independent of the contact interaction because their wavefunction vanishes whenever $x_1 = x_2$ (see **Q4.16**).

bosons and fermions

The reason why every state in the above calculation is either symmetric or antisymmetric with respect to particle interchange is that the Hamiltonian `In[519]` commutes with the particle interchange operator `In[522]` (see **Q4.15**). As a result, $\hat{\mathcal{H}}$ and $\hat{\Xi}$ can be diagonalized simultaneously.

We notice that $\hat{\Xi}$ has only eigenvalues ± 1:

```
In[524]:= Ξ //Eigenvalues //Counts
Out[524]= <| -1 -> 1225, 1 -> 1275 |>
```

The $n_{\max}(n_{\max}+1)/2$ eigenvalues $+1$ correspond to eigenvectors that are symmetric under particle interchange and form a basis of the symmetric subspace of the full Hilbert space (bosonic states); the $n_{\max}(n_{\max}-1)/2$ eigenvalues -1 correspond to eigenvectors that are antisymmetric under particle interchange and form a basis of the antisymmetric subspace of the full Hilbert space (fermionic states). By constructing a matrix whose rows are the symmetric eigenvectors, we construct an operator $\hat{\Pi}_s$ that projects from the full Hilbert space onto the space of symmetric states,

```
In[525]:= ε = Transpose[Eigensystem[Normal[Ξ]]];
In[526]:= Πs = Select[ε, #[[1]] == 1 &][[All, 2]] //Orthogonalize //SparseArray;
```

Similarly we construct a projector $\hat{\Pi}_a$ onto the space of antisymmetric states,

```
In[527]:= Πa = Select[ε, #[[1]] == -1 &][[All, 2]] //Orthogonalize //SparseArray;
```

With the help of these projectors, we define the Hamiltonians of the system restricted to the symmetric or antisymmetric subspace, respectively:

```
In[528]:= HPs[Ω_, κ_] = Πs.HP[Ω,κ].Transpose[Πs];
In[529]:= HPa[Ω_, κ_] = Πa.HP[Ω,κ].Transpose[Πa];
```

If the two particles in the present problem are indistinguishable bosons, then they can only populate the symmetric states (red dots in the above eigenvalue plot). We calculate the m lowest energy eigenstates of this symmetric subspace with the restricted Hamiltonian `HPs`:

```
In[530]:= Clear[sgs];
In[531]:= sgs[Ω_?NumericQ, κ_?NumericQ, m_Integer /; m >= 1] := sgs[Ω, κ, m] =
            {-#[[1]], #[[2]].Πs} &[Eigensystem[-HPs[N[Ω], N[κ]], m,
            Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}]]
```

Notice that we convert the calculated eigenstates back into the full Hilbert space by multiplying the results with `Πs` from the right.

In the same way, if the two particles in the present problem are indistinguishable fermions, then they can only populate the antisymmetric states (blue dots in the above eigenvalue plot). We calculate the m lowest energy eigenstates of this antisymmetric subspace with the restricted Hamiltonian `HPa`:

```
In[532]:= Clear[ags];
In[533]:= ags[Ω_?NumericQ, κ_?NumericQ, m_Integer /; m >= 1] := ags[Ω, κ, m] =
            {-#[[1]], #[[2]].Πa} &[Eigensystem[-HPa[N[Ω], N[κ]], m,
            Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}]]
```

As an example, here we calculate the six lowest energy eigenvalues of the full Hamiltonian for $\Omega = 0$ and $\kappa = 5$:

```
In[534]:= gs[0, 5, 6][[1]] //Sort
Out[534]= {15.2691, 24.674, 32.3863, 45.4849, 49.348, 58.1333}
```

The six lowest *symmetric* energy eigenvalues are

```
In[535]:= sgs[0, 5, 6][[1]] //Sort
Out[535]= {15.2691, 32.3863, 45.4849, 58.1333, 72.1818, 93.1942}
```

The six lowest *antisymmetric* energy eigenvalues are

```
In[536]:= ags[0, 5, 6][[1]] //Sort
Out[536]= {24.674, 49.348, 64.1524, 83.8916, 98.696, 123.37}
```

From Out[535] and Out[536] we can see which levels of Out[534] are symmetric or antisymmetric.

exercises

Q4.13 Show that Equation (4.65) is plausible by setting nmax=3, evaluating In[504], and then comparing its values to explicit integrals from Equation (4.65) for several tuples (j_1, j_2, j_3, j_4). *Hint:* use a=1 for simplicity.

Q4.14 In the problem of section 4.3.1, calculate the expectation value of the inter-particle distance $\langle x_1 - x_2 \rangle$, and its variance $\langle (x_1 - x_2)^2 \rangle - \langle x_1 - x_2 \rangle^2$, in the ground state as a function of κ (still keeping $\Omega = 0$). *Hint:* Using Equation (4.24), the position operators x1 and x2 are approximately

```
In[537]:= x = SparseArray[Band[{1,1}]->xgrid];
In[538]:= x1 = KroneckerProduct[x, id];
In[539]:= x2 = KroneckerProduct[id, x];
```

Q4.15 Show in Mathematica (by explicit calculation) that the Hamiltonian In[519] commutes with the particle interchange operator In[522]. *Hint:* use the Norm function to calculate the matrix norm of the commutator.

Q4.16 Show in Mathematica (by explicit calculation) that the antisymmetric Hamiltonian In[529] does not depend on κ.

Q4.17 The contact-interaction problem of this section can be solved analytically if $W(x) = 0$, which allows us to check the accuracy of the presented numerical calculations. We will study the dimensionless (a=m=$\hbar$=1) Hamiltonian $\hat{\mathcal{H}} = -\frac{1}{2}\left[\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}\right] + \kappa\delta(x_1 - x_2)$.

1. The ground-state wavefunction will be of the form

$$\psi(x_1, x_2) = A \times \begin{cases} \cos[\alpha(x_1 + x_2 - 1)]\cos[\beta(x_1 - x_2 + 1)] \\ \quad - \cos[\alpha(x_1 - x_2 + 1)]\cos[\beta(x_1 + x_2 - 1)] & \text{if } 0 \le x_1 \le x_2 \le 1, \\ \cos[\alpha(x_2 + x_1 - 1)]\cos[\beta(x_2 - x_1 + 1)] \\ \quad - \cos[\alpha(x_2 - x_1 + 1)]\cos[\beta(x_2 + x_1 - 1)] & \text{if } 0 \le x_2 \le x_1 \le 1. \end{cases} \tag{4.67}$$

   Check that this wavefunction satisfies the boundary conditions $\psi(x_1, 0) = \psi(x_1, 1) = \psi(0, x_2) = \psi(1, x_2) = 0$, that it is continuous across the boundary $x_1 = x_2$ (*i.e.*, that the two pieces of the wavefunction match up), and that it satisfies the symmetries of the calculation box: $\psi(x_1, x_2) = \psi(x_2, x_1) = \psi(1 - x_1, 1 - x_2)$.

2. Insert this wavefunction into the time-independent Schrödinger equation. Find the energy eigenvalue by assuming $x_1 \neq x_2$. You should find $E = \alpha^2 + \beta^2$.

3. Express the Hamiltonian and the wavefunction in terms of the new coordinates $R = (x_1 + x_2)/\sqrt{2}$ and $r = (x_1 - x_2)/\sqrt{2}$. *Hints:* $\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} = \frac{\partial^2}{\partial R^2} + \frac{\partial^2}{\partial r^2}$ and $\delta(ux) = u^{-1}\delta(x)$.

4. Integrate the Schrödinger equation, expressed in the (R, r) coordinates, over $r \in [-\epsilon, \epsilon]$ and take the limit $\epsilon \to 0^+$. Verify that the resulting expression is satisfied if $\alpha\tan(\alpha) = \beta\tan(\beta) = \kappa/2$. *Hint:* Do the integration analytically and use $\int_a^b dr \frac{\partial^2}{\partial r^2} f(r) = f'(b) - f'(a)$.

5. The ground state is found by numerically solving $\alpha\tan(\alpha) = \beta\tan(\beta) = \kappa/2$. Out of the many solutions of these equations, we choose the correct ones for the ground state by specifying the starting point of the numerical root solver:

```
In[540]:= Clear[a,b];
In[541]:= a[-∞] = π/2;
In[542]:= a[0] = π;
```

```
In[543]:= a[∞] = 3π/2;
In[544]:= a[κ_?NumericQ] := a[κ] = u /.
            FindRoot[u*Tan[u]==κ/2, {u,π+ArcTan[κ/(2π)]}]
In[545]:= b[-∞] = I*∞;
In[546]:= b[0] = 0;
In[547]:= b[∞] = π/2;
In[548]:= b[κ_ /; κ >= 0] := b[κ] = u /. FindRoot[u*Tan[u] == κ/2,
            {u, If[κ<π, 1, π/2 - π/κ + 2π/κ^2]}]
In[549]:= b[κ_ /; κ < 0] := b[κ] = I*u /. FindRoot[u*Tanh[u] == -κ/2, {u,-κ/2}]
```

Compare the resulting κ-dependent ground state energy to the numerically calculated ground-state energies from In[519].

4.3.2 two particles in one dimension with arbitrary interaction

Two particles in one dimension interacting via an arbitrary potential have a Hamiltonian very similar to Equation (4.61), except that the interaction is now

$$\hat{\mathcal{H}}_{\text{int}} = V_{\text{int}}(x_1, x_2), \tag{4.68}$$

or, more explicitly as an operator in the Dirac position basis,

$$\hat{\mathcal{H}}_{\text{int}} = \int_0^a \mathrm{d}x_1 \mathrm{d}x_2 |x_1\rangle \otimes |x_2\rangle V_{\text{int}}(x_1, x_2) \langle x_1| \otimes \langle x_2|. \tag{4.69}$$

As an example, for the Coulomb interaction we have $V_{\text{int}}(x_1, x_2) = \frac{Q_1 Q_2}{4\pi\epsilon_0 |x_1 - x_2|}$ with Q_1 and Q_2 the electric charges of the two particles. For many realistic potentials V_{int} only depends on $|x_1 - x_2|$.

In the finite-resolution position basis, the matrix elements of this interaction Hamiltonian can be approximated with a method similar to what we have already seen, for example in section 4.1.4:

$$\begin{aligned}\langle j_1, j_2|\hat{\mathcal{H}}_{\text{int}}|j_1', j_2'\rangle &= \int_0^a \vartheta_{j_1}(x_1)\vartheta_{j_2}(x_2) V_{\text{int}}(x_1, x_2)\vartheta_{j_1'}(x_1)\vartheta_{j_2'}(x_2)\mathrm{d}x_1\mathrm{d}x_2 \\ &\approx V_{\text{int}}(x_{j_1}, x_{j_2}) \int_0^a \vartheta_{j_1}(x_1)\vartheta_{j_2}(x_2)\vartheta_{j_1'}(x_1)\vartheta_{j_2'}(x_2)\mathrm{d}x_1\mathrm{d}x_2 = \delta_{j_1,j_1'}\delta_{j_2,j_2'} V_{\text{int}}(x_{j_1}, x_{j_2}).\end{aligned} \tag{4.70}$$

This approximation is easy to evaluate without the need for integration over basis functions. But realistic interaction potentials are usually singular for $x_1 = x_2$ (consider, for example, the Coulomb potential), and therefore the approximate Equation (4.70) fails for the evaluation of the matrix elements $\langle j, j|\hat{\mathcal{H}}_{\text{int}}|j, j\rangle$. This problem cannot be solved in all generality, and we can either resort to more accurate integration (as in section 4.3.1) or we can replace the true interaction potential with a less singular version: for the Coulomb potential, we could for example use a truncated singularity for $|x| < \delta$ for some small distance δ:

$$V_{\text{int}}(x) = \frac{Q_1 Q_2}{4\pi\epsilon_0} \times \begin{cases} \frac{1}{|x|} & \text{if } |x| \geq \delta \\ \frac{1}{\delta} & \text{if } |x| < \delta \end{cases} \tag{4.71}$$

As long as the particles move at energies much smaller than $V_{\text{int}}(\pm\delta) = \frac{Q_1 Q_2}{4\pi\epsilon_0\delta}$ they cannot distinguish the true Coulomb potential from this truncated form.

exercises

Q4.18 Consider two indistinguishable bosons in an infinite square well, interacting via the truncated Coulomb potential of Equation (4.71). Calculate the expectation value of the inter-particle distance, $\langle x_1 - x_2\rangle$, and its variance, $\langle (x_1 - x_2)^2\rangle - \langle x_1 - x_2\rangle^2$, in the ground state as a function of the Coulomb interaction strength (attractive and repulsive). *Hint:* set $\delta = \Delta = a/(n_{\text{max}} + 1)$ in Equation (4.71).

Q4.19 Answer **Q4.18** for two indistinguishable fermions.

4.4 one particle in several dimensions [code]

An important application of the imaginary-time propagation method of section 4.2.1 is the calculation of the shape of a three-dimensional Bose–Einstein condensate. In this section we use such a calculation as an example of how to extend single-particle lattice quantum mechanics to more spatial dimensions.

The non-linear Hamiltonian describing a three-dimensional Bose–Einstein condensate in a harmonic trap (to use a very common case) is

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)+\frac{m}{2}\left(\omega_x^2x^2+\omega_y^2y^2+\omega_z^2z^2\right)+(N-1)\frac{4\pi\hbar^2a_s}{m}|\psi(x,y,z)|^2, \quad (4.72)$$

where we have assumed that the single-particle wavefunction $\psi(x,y,z)$ is normalized: $\int|\psi(x,y,z)|^2\mathrm{d}x\mathrm{d}y\mathrm{d}z = 1$. As before, the contact interaction is described by the s-wave scattering length a_s. We will call $\kappa = \frac{4\pi\hbar^2a_s}{m}$ the interaction constant, as in previous sections.

We perform this calculation in a square box, where $|x| \le \frac{a}{2}$, $|y| \le \frac{a}{2}$, and $|z| \le \frac{a}{2}$; we will need to choose a large enough so that the BEC fits into this box, but small enough so that we do not need an unreasonably large $n_{\max}$ for the description of its wavefunction. Notice that this box is shifted by $\frac{a}{2}$ compared to the $[0\ldots a]$ boxes used so far; this does not influence the calculations in any way.

The ground state of the non-linear Hamiltonian of Equation (4.72) can be found by three-dimensional imaginary-time propagation, starting from (almost) any arbitrary state. Here we assemble a Mathematica function `groundstate` that, given an imaginary time step $\delta\beta$, propagates a random initial state until the state is converged to the ground state.

The units of the problem are dealt with as in section 4.1.1, differing from `In[409]`ff in that here we choose the length and time units freely:

```
In[550]:= LengthUnit = Quantity["Micrometers"];    (* choose freely *)
In[551]:= TimeUnit = Quantity["Seconds"];          (* choose freely *)
In[552]:= MassUnit = Quantity["ReducedPlanckConstant"]*TimeUnit/LengthUnit^2;
In[553]:= EnergyUnit = Quantity["ReducedPlanckConstant"]/TimeUnit;
In[554]:= ħ = Quantity["ReducedPlanckConstant"]/(EnergyUnit*TimeUnit);
```

We will be considering $N = 1000$ ^{87}Rb atoms in a magnetic trap with trap frequencies $\omega_x = 2\pi\times 115\,\mathrm{Hz}$ and $\omega_y = \omega_z = 2\pi\times 540\,\mathrm{Hz}$. The ^{87}Rb atoms are assumed to be in the $|F=1, M_F=-1\rangle$ hyperfine ground state, where their s-wave scattering length is $a_s = 100.4a_0$ (with $a_0 = 52.9177\,\mathrm{pm}$ the Bohr radius).

```
In[555]:= m = Quantity[86.909187, "AtomicMassUnit"]/MassUnit;
In[556]:= a = Quantity[10, "Micrometers"]/LengthUnit;
In[557]:= ωx = 2*π*Quantity[115, "Hertz"]*TimeUnit;
In[558]:= ωy = 2*π*Quantity[540, "Hertz"]*TimeUnit;
In[559]:= ωz = 2*π*Quantity[540, "Hertz"]*TimeUnit;
In[560]:= as = Quantity[100.4, "BohrRadius"]/LengthUnit;
In[561]:= κ = 4*π*ħ^2*as/m;
```

Next we define the grid on which the calculations will be done. In each Cartesian direction there are $n_{\max}$ grid points $x_j =$ `xgrid[[j]]` on the interval $[-a/2, +a/2]$:

```
In[562]:= nmax = 50;
In[563]:= Δ = a/(nmax+1);
In[564]:= xgrid = a*(Range[nmax]/(nmax+1) - 1/2);
```

We define the dimensionless harmonic-trap potential: the potential has its minimum at the center of the calculation box, *i.e.*, at $x = y = z = 0$.

```
In[565]:= W[x_,y_,z_] = m/2 * (ωx^2*x^2 + ωy^2*y^2 + ωz^2*z^2);
```

We only need the values of this potential on the grid points. To evaluate this, we build a three-dimensional array whose element `Wgrid[[jx,jy,jz]]` is given by the grid-point value `W[xgrid[[jx]],xgrid[[jy]],xgrid[[jz]]]`:

```
In[566]:= Wgrid=Table[W[xgrid[[jx]],xgrid[[jy]],xgrid[[jz]]],{jx,nmax},{jy,nmax},{jz,nmax}];
```

We could also define this more efficiently through functional programming:

```
In[567]:= Wgrid = Outer[W, xgrid, xgrid, xgrid];
```

The structure of the three-dimensional `Wgrid` array of potential values mirrors the structure of the wavefunction that we will be using: any wavefunction `v` will be a $n_{\max} \times n_{\max} \times n_{\max}$ array of coefficients in our finite-resolution position basis:

$$\psi(x,y,z) = \sum_{j_x,j_y,j_z=1}^{n_{\max}} \texttt{v[[jx,jy,jz]]}\vartheta_{j_x}(x)\vartheta_{j_y}(y)\vartheta_{j_z}(z). \tag{4.73}$$

From Equation (4.13) we find that on the three-dimensional grid points the wavefunction takes the values

$$\psi(x_{j_x}, x_{j_y}, x_{j_z}) = \texttt{v[[jx,jy,jz]]}/\Delta^{3/2}. \tag{4.74}$$

The norm of a wavefunction is

$$\int_{-\infty}^{\infty} |\psi(x,y,z)|^2 \mathrm{d}x\mathrm{d}y\mathrm{d}z = \sum_{j_x,j_y,j_z=1}^{n_{\max}} |\texttt{v[[jx,jy,jz]]}|^2 = \texttt{Norm[Flatten[v]]\^{}2}, \tag{4.75}$$

from which we define a wavefunction normalization function

```
In[568]:= nn[v_] := v/Norm[Flatten[v]]
```

The ground state calculation then proceeds by imaginary-time propagation, with step size $\delta\beta$ corresponding to an evolution $e^{-\delta\beta\hat{\mathcal{H}}}$ per step. The calculation is done for $N =$ `n` particles. Remember that the `FourierDST` function can do multi-dimensional discrete sine transforms, and therefore the kinetic-energy propagator can still be evaluated very efficiently. The last argument, `tolerance`, is optional and is given the value 10^{-6} if not specified (see section 1.6.5).

```
In[569]:= groundstate[n_?NumericQ, δβ_?NumericQ, tolerance_:10^(-6)] :=
            Module[{Kn, Ke, propKin, propPot2, prop, v0, γ, Ekin, Epot, Eint, Etot, μ},
              (* compute the diagonal elements of exp[-δβ*T] *)
              Kn = π^2*ħ^2/(2*m*a^2)*Table[nx^2+ny^2+nz^2,
                {nx,nmax}, {ny,nmax}, {nz,nmax}];
              Ke = Exp[-δβ*Kn] //N;
              (* propagate by a full imaginary-time-step with T *)
              propKin[v_] := nn[FourierDST[Ke*FourierDST[v, 1], 1]];
              (* propagate by a half imaginary-time-step with V *)
              propPot2[v_] := nn[Exp[-(δβ/2)*(Wgrid+κ*(n-1)*Abs[v]^2/Δ^3)]*v];
              (* propagate by a full imaginary-time-step by *)
              (* H=T+V using the Trotter approximation *)
              prop[v_] := propPot2[propKin[propPot2[v]]]
              (* random starting point *)
              v0 = nn @ RandomVariate[NormalDistribution[], {nmax, nmax, nmax}];
              (* propagation to the ground state *)
              γ = FixedPoint[prop, v0,
                SameTest -> Function[{v1,v2}, Norm[Flatten[v1-v2]]<tolerance]];
              (* energy components *)
              Ekin = Flatten[Kn].Flatten[Abs[FourierDST[γ, 1]]^2];
              Epot = Flatten[Wgrid].Flatten[Abs[γ]^2];
              Eint = (κ/2)*(n-1)*Total[Flatten[Abs[γ]^4]]/Δ^3;
              (* total energy *)
              Etot = Ekin + Epot + Eint;
```

```
25          (* chemical potential *)
26          μ = Ekin + Epot + 2*Eint;
27          (* return energy, chemical potential, coefficients *)
28          {Etot, μ, γ}]
```

As an example, we calculate the ground state for $N = 1000$ atoms and a time step of $\delta\beta = 10^{-5}$ time units, using the default convergence tolerance:

```
1  In[570]:= {Etot, μ, γ} = groundstate[1000, 10^(-4)];
2  In[571]:= {Etot, μ} * UnitConvert[EnergyUnit, "Joules"]
3  Out[571]= {6.88125*10^-31 J, 8.68181*10^-31 J}
```

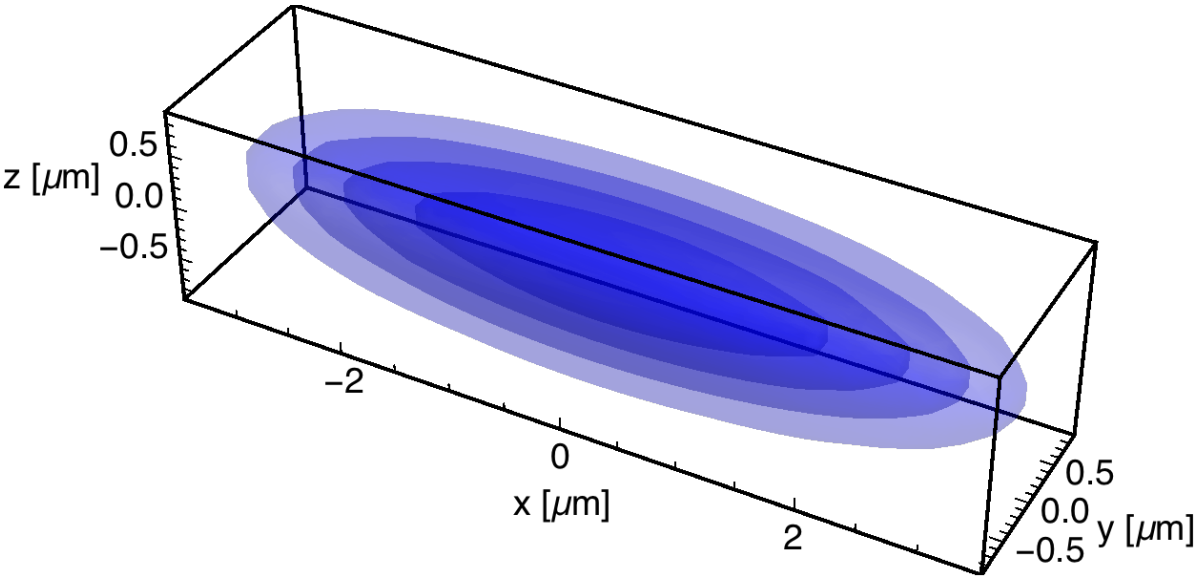
A more common energy unit is the Hertz, arrived at via Planck's constant:

```
1  In[572]:= {Etot, μ} * UnitConvert[EnergyUnit/Quantity["PlanckConstant"], "Hertz"]
2  Out[572]= {1038.51 Hz, 1310.25 Hz}
```

One way of plotting the ground-state density in 3D is as an iso-density surface. We plot the surface at half the peak density with

```
1  In[573]:= ρ = Abs[γ]^2/Δ^3;
2  In[574]:= ListContourPlot3D[ρ,
3            DataRange -> a*(1/(nmax+1)-1/2)*{{-1,1},{-1,1},{-1,1}},
4            Contours -> {Max[ρ]/2}, BoxRatios -> Automatic]
```

Here we show several such iso-density surfaces:



For more quantitative results we can, for example, calculate the expectation values `X` $= \langle x \rangle$, `Y` $= \langle y \rangle$, `Z` $= \langle z \rangle$, `XX` $= \langle x^2 \rangle$, `YY` $= \langle y^2 \rangle$, `ZZ` $= \langle z^2 \rangle$. We could define coordinate arrays as

```
1  In[575]:= xc = Table[xgrid[[jx]], {jx,nmax}, {jy,nmax}, {jz,nmax}];
2  In[576]:= yc = Table[xgrid[[jy]], {jx,nmax}, {jy,nmax}, {jz,nmax}];
3  In[577]:= zc = Table[xgrid[[jz]], {jx,nmax}, {jy,nmax}, {jz,nmax}];
```

but we define them more efficiently as follows:

```
In[578]:= ones = ConstantArray[1, nmax];
In[579]:= xc = Outer[Times, xgrid, ones, ones];
In[580]:= yc = Outer[Times, ones, xgrid, ones];
In[581]:= zc = Outer[Times, ones, ones, xgrid];
```

The desired expectation values are then computed with

```
In[582]:= X = Total[Flatten[xc * ρ]];
In[583]:= Y = Total[Flatten[yc * ρ]];
In[584]:= Z = Total[Flatten[zc * ρ]];
In[585]:= XX = Total[Flatten[xc^2 * ρ]];
In[586]:= YY = Total[Flatten[yc^2 * ρ]];
In[587]:= ZZ = Total[Flatten[zc^2 * ρ]];
```

The root-mean-square size of the BEC is calculated from these as the standard deviations of the position operators in the three Cartesian directions:

```
In[588]:= {Sqrt[XX-X^2], Sqrt[YY-Y^2], Sqrt[ZZ-Z^2]} * LengthUnit
Out[588]= {1.58829 μm, 0.417615 μm, 0.417615 μm}
```

4.4.1 exercises

Q4.20 Take the BEC Hamiltonian of Equation (4.72) in the absence of interactions ($a_s = 0$) and calculate analytically the expectation values $\langle x^2\rangle$, $\langle y^2\rangle$, $\langle z^2\rangle$ in the ground state.

Q4.21 Take the BEC Hamiltonian of Equation (4.72) in the limit of strong interactions (Thomas–Fermi limit), where the kinetic energy can be neglected. The Gross–Pitaevskii equation is then

$$\left[\frac{m}{2}\left(\omega_x^2x^2+\omega_y^2y^2+\omega_z^2z^2\right)+(N-1)\frac{4\pi\hbar^2a_s}{m}|\psi(x,y,z)|^2\right]\psi(x,y,z)=\mu\psi(x,y,z), \tag{4.76}$$

which has two solutions:

$$|\psi(x,y,z)|^2=\begin{cases}0 & \text{or}\\ \dfrac{\mu-\frac{m}{2}\left(\omega_x^2x^2+\omega_y^2y^2+\omega_z^2z^2\right)}{(N-1)\frac{4\pi\hbar^2a_s}{m}}.\end{cases} \tag{4.77}$$

Together with the conditions that $|\psi(x,y,z)|^2\geq 0$, that $\psi(x,y,z)$ should be continuous, and that $\int|\psi(x,y,z)|^2\mathrm{d}x\mathrm{d}y\mathrm{d}z=1$, this gives us the Thomas–Fermi "inverted parabola" density

$$|\psi(x,y,z)|^2=\begin{cases}\rho_0\left[1-\left(\frac{x}{R_x}\right)^2-\left(\frac{y}{R_y}\right)^2-\left(\frac{z}{R_z}\right)^2\right] & \text{if } \left(\frac{x}{R_x}\right)^2+\left(\frac{y}{R_y}\right)^2+\left(\frac{z}{R_z}\right)^2\leq 1,\\ 0 & \text{if not,}\end{cases} \tag{4.78}$$

which is nonzero only inside an ellipsoid with Thomas–Fermi radii

$$R_x=\left[\frac{15\hbar^2a_s(N-1)\omega_y\omega_z}{m^2\omega_x^4}\right]^{\frac{1}{5}}=\left[\frac{15\kappa(N-1)\omega_y\omega_z}{4\pi m\omega_x^4}\right]^{\frac{1}{5}}, \tag{4.79a}$$

$$R_y=\left[\frac{15\hbar^2a_s(N-1)\omega_z\omega_x}{m^2\omega_y^4}\right]^{\frac{1}{5}}=\left[\frac{15\kappa(N-1)\omega_x\omega_z}{4\pi m\omega_y^4}\right]^{\frac{1}{5}}, \tag{4.79b}$$

$$R_z=\left[\frac{15\hbar^2a_s(N-1)\omega_x\omega_y}{m^2\omega_z^4}\right]^{\frac{1}{5}}=\left[\frac{15\kappa(N-1)\omega_x\omega_y}{4\pi m\omega_z^4}\right]^{\frac{1}{5}}. \tag{4.79c}$$

The density at the origin of the ellipsoid is

$$\rho_0=\frac{1}{8\pi}\left[\frac{225m^6\omega_x^2\omega_y^2\omega_z^2}{\hbar^6a_s^3(N-1)^3}\right]^{\frac{1}{5}}=\left[\frac{225m^3\omega_x^2\omega_y^2\omega_z^2}{512\pi^2\kappa^3(N-1)^3}\right]^{\frac{1}{5}} \tag{4.80}$$

and the chemical potential is

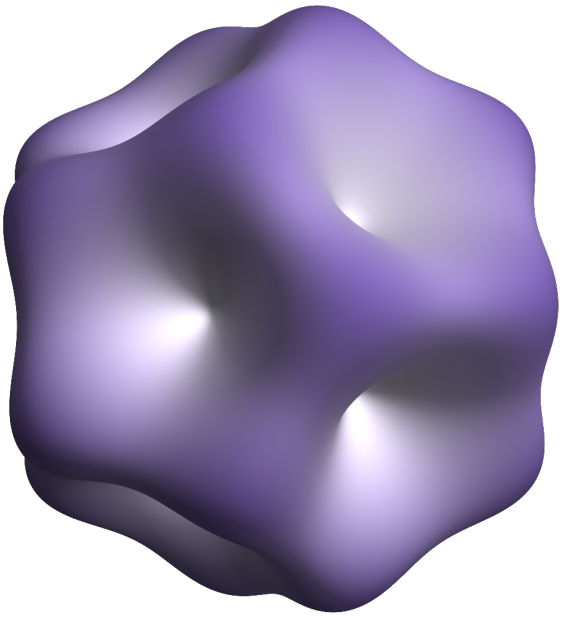
$$\mu = \frac{1}{2}\left[225m\hbar^4 a_s^2 (N-1)^2 \omega_x^2 \omega_y^2 \omega_z^2\right]^{\frac{1}{5}} = \left[\frac{225}{512\pi^2} m^3 \kappa^2 (N-1)^2 \omega_x^2 \omega_y^2 \omega_z^2\right]^{\frac{1}{5}}. \tag{4.81}$$

Using this Thomas–Fermi density profile, calculate the expectation values $\langle x^2\rangle$, $\langle y^2\rangle$, $\langle z^2\rangle$ in the ground state of the Thomas–Fermi approximation. *Hints:* Calculate $\langle x^2\rangle$ using Equation (4.78) without substituting Equations (4.79) and Equation (4.80); do these substitutions only after having found the result. You can find $\langle y^2\rangle$ and $\langle z^2\rangle$ by analogy, without repeating the calculation.

Q4.22 Compare the numerical expectation values $\langle x^2\rangle$, $\langle y^2\rangle$, $\langle z^2\rangle$ of our Mathematica code to the analytic results of **Q4.20** and **Q4.21**. What is the maximum ^{87}Rb atom number N which allows a reasonably good description (in this specific trap) with the non-interacting solution? What is the minimum atom number which allows a reasonably good description with the Thomas–Fermi solution?

5

combining spatial motion and spin



In this chapter we put together all the techniques studied so far: internal-spin degrees of freedom (chapter 3) and spatial (motional) degrees of freedom (chapter 4) are combined with the tensor-product formalism (chapter 2). We arrive at a complete numerical description of interacting spin-ful particles moving through space. To showcase these powerful tools, we study Rashba coupling as well as the Jaynes–Cummings model.

5.1 one particle in 1D with spin

5.1.1 separable Hamiltonian

The simplest problem combining a spatial and a spin degree of freedom in a meaningful way consists of a single spin-1/2 particle moving in one dimension in a state-selective potential:

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V_0(x) + V_z(x)\hat{S}_z, \tag{5.1}$$

where $\hat{S}_z = \frac{1}{2}\hat{\sigma}_z$ is given by the Pauli matrix. As was said before, Equation (5.1) is a short-hand notation of the full Hamiltonian

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x|x\rangle\frac{\mathrm{d}^2}{\mathrm{d}x^2}\langle x|\otimes\mathbb{1} + \int_{-\infty}^{\infty}\mathrm{d}x|x\rangle V_0(x)\langle x|\otimes\mathbb{1} + \int_{-\infty}^{\infty}\mathrm{d}x|x\rangle V_z(x)\langle x|\otimes\hat{S}_z, \tag{5.2}$$

where it is more evident that the first two terms act only on the spatial part of the wavefunction, while the third term couples the two degrees of freedom.

The Hilbert space of this particle consists of a one-dimensional degree of freedom x, which we had described in chapter 4 with a basis built from square-well eigenstates, and a spin-1/2 degree of freedom $\hat{\vec{S}} = \frac{1}{2}\hat{\vec{\sigma}}$ described in the Dicke basis (chapter 3). This tensor-product structure of the Hilbert space allows us to simplify the matrix elements of the Hamiltonian by factoring out the spin degree of freedom,

$$\begin{aligned}
\langle\phi,\uparrow|\hat{\mathcal{H}}|\psi,\uparrow\rangle &= -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\phi^*(x)\psi''(x)\mathrm{d}x\langle\uparrow|\uparrow\rangle + \int_{-\infty}^{\infty}\phi^*(x)V_0(x)\psi(x)\mathrm{d}x\langle\downarrow|\downarrow\rangle + \frac{1}{2}\int_{-\infty}^{\infty}\phi^*(x)V_z(x)\psi(x)\mathrm{d}x\langle\uparrow|\hat{\sigma}_z|\uparrow\rangle \\
&= -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\phi^*(x)\psi''(x)\mathrm{d}x + \int_{-\infty}^{\infty}\phi^*(x)V_0(x)\psi(x)\mathrm{d}x + \frac{1}{2}\int_{-\infty}^{\infty}\phi^*(x)V_z(x)\psi(x)\mathrm{d}x \\
\langle\phi,\uparrow|\hat{\mathcal{H}}|\psi,\downarrow\rangle &= -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\phi^*(x)\psi''(x)\mathrm{d}x\langle\uparrow|\downarrow\rangle + \int_{-\infty}^{\infty}\phi^*(x)V_0(x)\psi(x)\mathrm{d}x\langle\uparrow|\downarrow\rangle + \frac{1}{2}\int_{-\infty}^{\infty}\phi^*(x)V_z(x)\psi(x)\mathrm{d}x\langle\uparrow|\hat{\sigma}_z|\downarrow\rangle \\
&= 0 \\
\langle\phi,\downarrow|\hat{\mathcal{H}}|\psi,\uparrow\rangle &= -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\phi^*(x)\psi''(x)\mathrm{d}x\langle\downarrow|\uparrow\rangle + \int_{-\infty}^{\infty}\phi^*(x)V_0(x)\psi(x)\mathrm{d}x\langle\downarrow|\uparrow\rangle + \frac{1}{2}\int_{-\infty}^{\infty}\phi^*(x)V_z(x)\psi(x)\mathrm{d}x\langle\downarrow|\hat{\sigma}_z|\uparrow\rangle \\
&= 0 \\
\langle\phi,\downarrow|\hat{\mathcal{H}}|\psi,\downarrow\rangle &= -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\phi^*(x)\psi''(x)\mathrm{d}x\langle\downarrow|\downarrow\rangle + \int_{-\infty}^{\infty}\phi^*(x)V_0(x)\psi(x)\mathrm{d}x\langle\downarrow|\downarrow\rangle + \frac{1}{2}\int_{-\infty}^{\infty}\phi^*(x)V_z(x)\psi(x)\mathrm{d}x\langle\downarrow|\hat{\sigma}_z|\downarrow\rangle \\
&= -\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\phi^*(x)\psi''(x)\mathrm{d}x + \int_{-\infty}^{\infty}\phi^*(x)V_0(x)\psi(x)\mathrm{d}x - \frac{1}{2}\int_{-\infty}^{\infty}\phi^*(x)V_z(x)\psi(x)\mathrm{d}x.
\end{aligned} \tag{5.3}$$

We see that this Hamiltonian does not mix states with different spin states (since all matrix elements where the spin state differs between the left and right side are equal to zero). We can therefore solve the two disconnected problems of finding the particle's behavior with spin up or with spin down, with effective Hamiltonians

$$\hat{\mathcal{H}}_\uparrow = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V_0(x) + \frac{1}{2}V_z(x), \qquad \hat{\mathcal{H}}_\downarrow = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V_0(x) - \frac{1}{2}V_z(x). \tag{5.4}$$

These Hamiltonians now only describe the spatial degree of freedom, and the methods of chapter 4 can be used without further modifications.

5.1.2 non-separable Hamiltonian

A more interesting situation arises when the Hamiltonian is not separable as in section 5.1.1. Take, for example, the Hamiltonian of Equation (5.1) in the presence of a uniform transverse magnetic field B_x,

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V_0(x) + V_z(x)\hat{S}_z + B_x\hat{S}_x. \tag{5.5}$$

The interaction Hamiltonian with the magnetic field is not separable:

$$\begin{aligned}
\langle\phi,\uparrow|B_x\hat{S}_x|\psi,\uparrow\rangle &= \frac{1}{2}B_x\int_{-\infty}^{\infty}\phi^*(x)\psi(x)\mathrm{d}x\langle\uparrow|\hat{\sigma}_x|\uparrow\rangle = 0\\
\langle\phi,\uparrow|B_x\hat{S}_x|\psi,\downarrow\rangle &= \frac{1}{2}B_x\int_{-\infty}^{\infty}\phi^*(x)\psi(x)\mathrm{d}x\langle\uparrow|\hat{\sigma}_x|\downarrow\rangle = \frac{1}{2}B_x\int_{-\infty}^{\infty}\phi^*(x)\psi(x)\mathrm{d}x\\
\langle\phi,\downarrow|B_x\hat{S}_x|\psi,\uparrow\rangle &= \frac{1}{2}B_x\int_{-\infty}^{\infty}\phi^*(x)\psi(x)\mathrm{d}x\langle\downarrow|\hat{\sigma}_x|\uparrow\rangle = \frac{1}{2}B_x\int_{-\infty}^{\infty}\phi^*(x)\psi(x)\mathrm{d}x\\
\langle\phi,\downarrow|B_x\hat{S}_x|\psi,\downarrow\rangle &= \frac{1}{2}B_x\int_{-\infty}^{\infty}\phi^*(x)\psi(x)\mathrm{d}x\langle\downarrow|\hat{\sigma}_x|\downarrow\rangle = 0.
\end{aligned} \tag{5.6}$$

Therefore we can no longer study separate Hamiltonians as in Equation (5.4), and we must instead study the joint system of spatial motion and spin. In what follows we study a simple example of such a Hamiltonian, both analytically and numerically. We take the trapping potential to be harmonic,

$$V_0(x) = \frac{1}{2}m\omega^2x^2 \tag{5.7}$$

and the state-selective potential as a homogeneous force,

$$V_z(x) = -Fx. \tag{5.8}$$

ground state for $B_x = 0$

For $B_x = 0$ we know that the ground states of the two spin sectors are the ground states of the effective Hamiltonians of Equation (5.4), which are Gaussians:

$$\langle x|\gamma_\uparrow\rangle = \frac{e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}}{\sqrt{\sigma\sqrt{2\pi}}}\otimes|\uparrow\rangle \qquad\qquad \langle x|\gamma_\downarrow\rangle = \frac{e^{-\left(\frac{x+\mu}{2\sigma}\right)^2}}{\sqrt{\sigma\sqrt{2\pi}}}\otimes|\downarrow\rangle \tag{5.9}$$

with $\mu = \frac{F}{2m\omega^2}$ and $\sigma = \sqrt{\frac{\hbar}{2m\omega}}$. These two ground states are degenerate, with energy $E = \frac{1}{2}\hbar\omega - \frac{F^2}{8m\omega^2}$. In both of these ground states the spatial and spin degrees of freedom are entangled: the particle is more likely to be detected in the $|\uparrow\rangle$ state on the right side ($x > 0$), and more likely to be detected in the $|\downarrow\rangle$ state on the left side ($x < 0$) of the trap. This results in a positive expectation value of the operator $\hat{x}\otimes\hat{S}_z$:

$$\langle\gamma_\uparrow|\hat{x}\otimes\hat{S}_z|\gamma_\uparrow\rangle = \langle\gamma_\downarrow|\hat{x}\otimes\hat{S}_z|\gamma_\downarrow\rangle = \frac{\mu}{2} = \frac{F}{4m\omega^2}. \tag{5.10}$$

perturbative ground state for $B_x > 0$

For small $|B_x|$ the ground state can be described by a linear combination of the states in Equation (5.9). If we set

$$|\gamma_\mathrm{p}\rangle = \alpha\times|\gamma_\uparrow\rangle + \beta\times|\gamma_\downarrow\rangle \tag{5.11}$$

with $|\alpha|^2 + |\beta|^2 = 1$, we find that the expectation value of the energy is

$$\begin{aligned}
\langle\gamma_\mathrm{p}|\hat{\mathcal{H}}|\gamma_\mathrm{p}\rangle &= |\alpha|^2\langle\gamma_\uparrow|\hat{\mathcal{H}}|\gamma_\uparrow\rangle + \alpha^*\beta\langle\gamma_\uparrow|\hat{\mathcal{H}}|\gamma_\downarrow\rangle + \beta^*\alpha\langle\gamma_\downarrow|\hat{\mathcal{H}}|\gamma_\uparrow\rangle + |\beta|^2\langle\gamma_\downarrow|\hat{\mathcal{H}}|\gamma_\downarrow\rangle\\
&= \frac{1}{2}\hbar\omega - \frac{F^2}{8m\omega^2} + \frac{1}{2}B_x(\alpha^*\beta + \beta^*\alpha)e^{-\frac{F^2}{4m\hbar\omega^3}}
\end{aligned} \tag{5.12}$$

For $B_x > 0$ this energy is minimized for $\alpha = 1/\sqrt{2}$ and $\beta = -1/\sqrt{2}$, and the perturbative ground state is therefore the anti-symmetric combination of the states in Equation (5.9)

$$\langle x|\gamma_\mathrm{p}\rangle = \frac{e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma\sqrt{2\pi}}}\otimes|\uparrow\rangle - \frac{e^{-\left(\frac{x+\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma\sqrt{2\pi}}}\otimes|\downarrow\rangle. \tag{5.13}$$

with energy

$$\langle\gamma_\mathrm{p}|\hat{\mathcal{H}}|\gamma_\mathrm{p}\rangle = \frac{1}{2}\hbar\omega - \frac{F^2}{8m\omega^2} - \frac{1}{2}B_xe^{-\frac{F^2}{4m\hbar\omega^3}}. \tag{5.14}$$

The energy splitting between this ground state and the first excited state,

$$\langle x|\epsilon_p\rangle = \frac{e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma\sqrt{2\pi}}} \otimes |\uparrow\rangle + \frac{e^{-\left(\frac{x+\mu}{2\sigma}\right)^2}}{\sqrt{2\sigma\sqrt{2\pi}}} \otimes |\downarrow\rangle. \tag{5.15}$$

is $\Delta E = \langle\epsilon_p|\hat{\mathcal{H}}|\epsilon_p\rangle - \langle\gamma_p|\hat{\mathcal{H}}|\gamma_p\rangle = B_x e^{-\frac{F^2}{4m\hbar\omega^3}}$, which can be very small for large exponents $\frac{F^2}{4m\hbar\omega^3}$.

numerical calculation of the ground state **[code]**

For a numerical description of this particle we use dimensionless units such that `a=m=ħ=1`; other units can be used in the same was as presented in section 4.1.1. We describe the spatial degree of freedom with the finite-resolution position basis of section 4.1.2, centered at $x = 0$ as in section 4.4:

```
In[589]:= a = m = ħ = 1;
In[590]:= nmax = 100;
In[591]:= Δ = a/(nmax+1);
In[592]:= xgrid = a*(Range[nmax]/(nmax+1)-1/2);
```

The operator $\hat{x}$ is approximately diagonal in this representation (see Equation (4.24)):

```
In[593]:= xop = SparseArray[Band[{1,1}] -> xgrid];
```

The identity operator on the spatial degree of freedom is

```
In[594]:= idx = IdentityMatrix[nmax, SparseArray];
```

The identity and Pauli operators for the spin degree of freedom are

```
In[595]:= ids = IdentityMatrix[2, SparseArray];
In[596]:= {sx,sy,sz}=Table[SparseArray[PauliMatrix[i]/2], {i,3}];
```

The kinetic energy operator is constructed via a discrete sine transform, as before:

```
In[597]:= TM = SparseArray[Band[{1,1}]->Range[nmax]^2*π^2*ħ^2/(2*m*a^2)];
In[598]:= TP = FourierDST[TM, 1];
```

From these we assemble the Hamiltonian, assuming that F and B_x are expressed in matching units:

```
In[599]:= H[ω_, F_, Bx_] =
            KroneckerProduct[TP, ids]
            + m*ω^2/2 * KroneckerProduct[xop.xop, ids]
            - F * KroneckerProduct[xop, sz]
            + Bx * KroneckerProduct[idx, sx];
```

We compute the ground state of this Hamiltonian with

```
In[600]:= Clear[gs];
In[601]:= gs[ω_?NumericQ, F_?NumericQ, Bx_?NumericQ] :=
            gs[ω, F, Bx] = -Eigensystem[-H[N[ω],N[F],N[Bx]], 1,
              Method -> {"Arnoldi", "Criteria" -> "RealPart", MaxIterations -> 10^6}]
```

Once a ground state $|\gamma\rangle$ has been calculated, for example with

```
In[602]:= γ = gs[100, 5000, 500][[2, 1]];
```

the usual problem arises of how to display and interpret the wavefunction. Instead of studying the coefficients of γ directly, we calculate several specific properties of the ground state in what follows.

Operator expectation values: The mean spin direction (magnetization) $\langle\hat{\vec{S}}\rangle = \{\langle\hat{S}_x\rangle, \langle\hat{S}_y\rangle, \langle\hat{S}_z\rangle\}$ is calculated directly from the ground-state coefficients list with

```
In[603]:= mx = Re[Conjugate[γ].(KroneckerProduct[idx,sx].γ)];
In[604]:= my = Re[Conjugate[γ].(KroneckerProduct[idx,sy].γ)];
In[605]:= mz = Re[Conjugate[γ].(KroneckerProduct[idx,sz].γ)];
In[606]:= {mx,my,mz}
Out[606]= {-0.233037, 0., -2.08318*10^-12}
```

The mean position $\langle\hat{x}\rangle$ and its standard deviation are calculated with

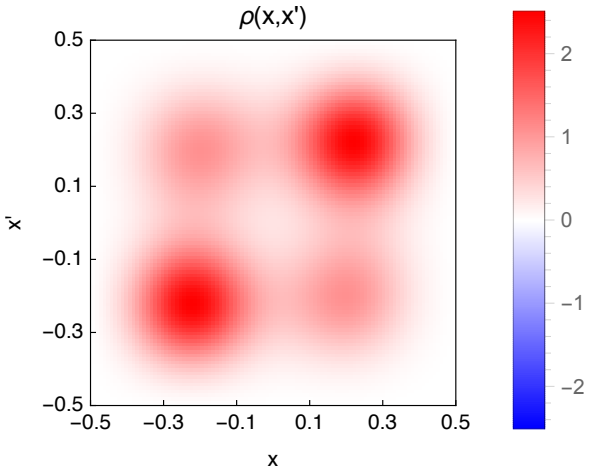
```
In[607]:= X = Re[Conjugate[γ].(KroneckerProduct[xop,ids].γ)];
In[608]:= XX = Re[Conjugate[γ].(KroneckerProduct[xop.xop,ids].γ)];
In[609]:= {X, Sqrt[XX-X^2]}
Out[609]= {1.2178*10^-11, 0.226209}
```

Even though we found $\langle\hat{x}\rangle = 0$ and $\langle\hat{S}_z\rangle = 0$ above, these coordinates are correlated: calculating $\langle\hat{x}\otimes\hat{S}_z\rangle$,

```
In[610]:= Xz = Re[Conjugate[γ].(KroneckerProduct[xop,sz].γ)]
Out[610]= 0.0954168
```

Reduced density matrix of the spatial degree of freedom: Using `In[257]` we trace out the spin degree of freedom (the last two dimensions) to find the density matrix in the spatial coordinate:

```
In[611]:= ρx = traceout[γ, -2];
In[612]:= ArrayPlot[Reverse[Transpose[ArrayPad[ρx/Δ, 1]]]]
```



Reduced density matrix of the spin degree of freedom: We can do the same for the reduced matrix of the spin degree of freedom, using `In[256]`, and find a 2×2 spin density matrix:

```
In[613]:= ρs = traceout[γ, nmax]
Out[613]= {{0.5, -0.233037}, {-0.233037, 0.5}}
```

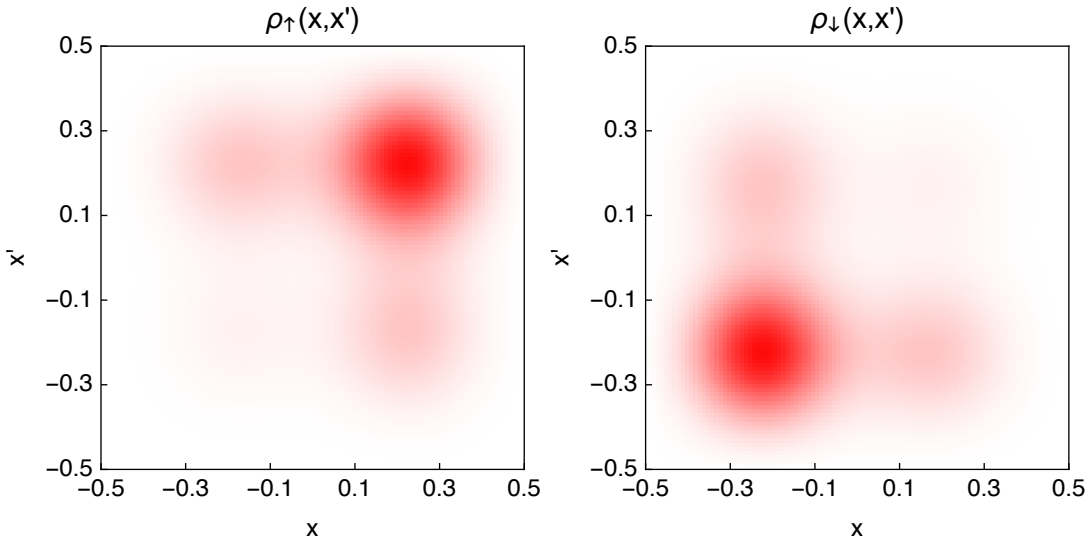
Spin-specific spatial densities: The reduced density matrix of particles in the spin-up state is found by projecting the ground state $|\gamma\rangle$ onto the spin-up sector with the projector $\hat{\Pi}_\uparrow = |\uparrow\rangle\langle\uparrow| = \frac{1}{2}\mathbb{1} + \hat{S}_z$.[1] Thus, $|\gamma_\uparrow\rangle = \hat{\Pi}_\uparrow|\gamma\rangle$ only describes the particles that are in the spin-up state:

[1] Remember that $\mathbb{1} = |\uparrow\rangle\langle\uparrow| + |\downarrow\rangle\langle\downarrow|$ and $\hat{S}_z = \frac{1}{2}|\uparrow\rangle\langle\uparrow| - \frac{1}{2}|\downarrow\rangle\langle\downarrow|$.

```
In[614]:= γup = KroneckerProduct[idx, ids/2+sz].γ;
In[615]:= ρxup = traceout[γup, -2];
```

In the same way the reduced density matrix of particles in the spin-down state $|\gamma_\downarrow\rangle = \hat{\Pi}_\downarrow|\gamma\rangle$ is calculated with the down-projector $\hat{\Pi}_\downarrow = |\downarrow\rangle\langle\downarrow| = \frac{1}{2}\mathbb{1} - \hat{S}_z$:

```
In[616]:= γdn = KroneckerProduct[idx, ids/2-sz].γ;
In[617]:= ρxdn = traceout[γdn, -2];
```



The positive correlation between the spin and the mean position, $\langle \hat{x} \otimes \hat{S}_z \rangle > 0$, is clearly visible in these plots.

Since $\hat{\Pi}_\uparrow + \hat{\Pi}_\downarrow = \mathbb{1}$, these two spin-specific spatial density matrices add up to the total density shown previously. This also means that the spin-specific density matrices do not have unit trace:

```
In[618]:= {Tr[ρxup], Tr[ρxdn]}
Out[618]= {0.5, 0.5}
```

Hence we have 50% chance of finding the particle in the up or down spin states.

Space-dependent spin expectation value: Similarly, we can calculate the reduced density matrix of the spin degree of freedom at a specific point in space by using projection operators $\hat{\Pi}_j = |j\rangle\langle j|$ onto single position-basis states $|j\rangle$:

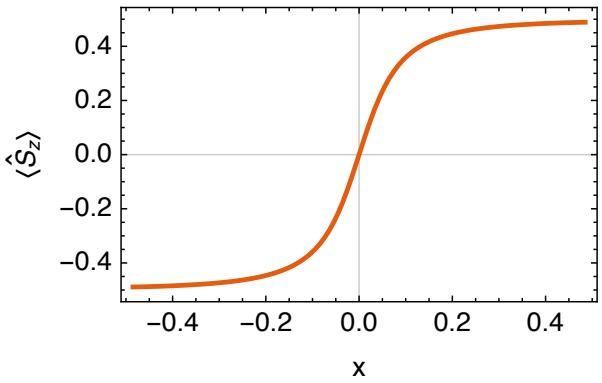
```
In[619]:= γx[j_Integer /; 1 <= j <= nmax] :=
            KroneckerProduct[SparseArray[{j, j} -> 1, {nmax, nmax}], ids].γ
In[620]:= ρsx[j_Integer /; 1 <= j <= nmax] := traceout[γx[j], nmax]
```

We notice that, as before, these spatially-local reduced density matrices do not have unit trace, but their traces sum up to 1:

```
In[621]:= Sum[Tr[ρsx[j]], {j, nmax}]
Out[621]= 1.
```

In fact, the traces of these local reduced density matrices give the probability of finding the particle at the given position. We can use this interpretation to calculate the mean spin expectation value of a particle measured at a given grid point:

```
In[622]:= meansx[j_Integer /; 1 <= j <= nmax] := Tr[ρsx[j].sz]/Tr[ρsx[j]]
In[623]:= ListLinePlot[Transpose[{xgrid, Table[meansx[j], {j, nmax}]}]]
```



This graph confirms the observation that particles detected on the left side are more likely to be in the $|\downarrow\rangle$ state, while particles detected on the right side are more likely to be in the $|\uparrow\rangle$ state.

5.1.3 exercises

Q5.1 In the problem described by the Hamiltonian of Equation (5.5), calculate the following expectation values (numerically) for several parameter sets $\{\omega, F, B_x\}$:

1. $\langle x\rangle$ for particles detected in the $|\uparrow\rangle$ state
2. $\langle x\rangle$ for particles detected in the $|\downarrow\rangle$ state
3. $\langle x\rangle$ for particles detected in any spin state
4. the mean and variance of $\hat{x}\otimes\hat{S}_z$

5.2 one particle in 2D with spin: Rashba coupling [code]

A particularly interesting kind of interaction is the Rashba coupling between a particle's momentum and its spin.[2] In general, this interaction is proportional to a component of the vector product $\hat{\boldsymbol{\kappa}} = \hat{\boldsymbol{p}}\times\hat{\boldsymbol{S}}$. For a particle moving in two dimensions (x, y), the coupling involves the z-component $\hat{\kappa}_z = \hat{p}_x\otimes\hat{S}_y - \hat{p}_y\otimes\hat{S}_x$.

In this section we study the 2D Rashba Hamiltonian

$$\hat{\mathcal{H}} = \frac{\hat{p}_x^2+\hat{p}_y^2}{2m} + V(x,y) + \delta\hat{S}_z + \alpha(\hat{p}_x\otimes\hat{S}_y - \hat{p}_y\otimes\hat{S}_x) \tag{5.16}$$

in a square box where $-\frac{a}{2}\le x, y\le\frac{a}{2}$ as before. With a Hilbert space composed as the tensor product of the x, y, and spin coordinates, in this order, the full Hamiltonian thus becomes

$$\begin{aligned}\hat{\mathcal{H}} = &\left[-\frac{\hbar^2}{2m}\int_{-a/2}^{a/2}\mathrm{d}x|x\rangle\frac{\partial^2}{\partial x^2}\langle x|\right]\otimes\mathbb{1}\otimes\mathbb{1} + \mathbb{1}\otimes\left[-\frac{\hbar^2}{2m}\int_{-a/2}^{a/2}\mathrm{d}x|y\rangle\frac{\partial^2}{\partial y^2}\langle y|\right]\otimes\mathbb{1}\\ &+\left[\int_{-a/2}^{a/2}\mathrm{d}x\mathrm{d}y|x\rangle|y\rangle V(x,y)\langle x|\langle y|\right]\otimes\mathbb{1} + \delta(\mathbb{1}\otimes\mathbb{1}\otimes\hat{S}_z) + \alpha(\hat{p}_x\otimes\mathbb{1}\otimes\hat{S}_y - \mathbb{1}\otimes\hat{p}_y\otimes\hat{S}_x).\end{aligned} \tag{5.17}$$

For simplicity we will set $V(x, y) = 0$; but any nonzero potential can be used with the techniques introduced previously. Further, we use `a=m=ħ=1` to simplify the units; but as usual, any system of units may be used (see section 4.1.1).

Since both the kinetic and the interaction operator are most easily expressed in the momentum representation, we use the momentum representation (see section 4.1.2) to express the spatial degrees of freedom of the Hamiltonian. The identity operator is

[2] See https://en.wikipedia.org/wiki/Rashba_effect.

```
In[624]:= nmax = 50;
In[625]:= Δ = a/(nmax+1);
In[626]:= idM = IdentityMatrix[nmax, SparseArray];
```

We use the exact form of the kinetic operator from In[423] and the exact form of the momentum operator from In[444]. As discussed previously, these two forms do not exactly satisfy $\hat{T} = \hat{p}^2/(2m)$. They are, however, the best available low-energy forms.

For the spin degree of freedom, we assume $S = 1/2$, giving us the usual spin operators and the identity operator,

```
In[627]:= {sx,sy,sz} = Table[SparseArray[PauliMatrix[i]/2], {i, 3}];
In[628]:= idS = IdentityMatrix[2, SparseArray];
```

With these definitions, we assemble the Rashba Hamiltonian of Equation (5.17) in the momentum representation with

```
In[629]:= HM[δ_, α_] = KroneckerProduct[TM, idM, idS]
          + KroneckerProduct[idM, TM, idS]
          + δ*KroneckerProduct[idM, idM, sz]
          + α*(KroneckerProduct[pM, idM, sy] - KroneckerProduct[idM, pM, sx]);
```

Given a state γ, for example the ground state of In[629] for specific values of $\delta = 1$ and $\alpha = 20$, we calculate the mean value $\langle \hat{x}^2 \rangle = \langle \hat{x}^2 \otimes \mathbb{1} \otimes \mathbb{1} \rangle$ with the position operator xM expressed in the momentum basis:

```
In[630]:= xgrid = a*(Range[nmax]/(nmax + 1) - 1/2);
In[631]:= xP = SparseArray[Band[{1, 1}] -> xgrid];
In[632]:= xM = FourierDST[xP, 1];
In[633]:= Conjugate[γ].(KroneckerProduct[xM.xM, idM, idS].γ) //Re
Out[633]= 0.0358875
```

In the same way, we calculate the mean value $\langle \hat{y}^2 \rangle = \langle \mathbb{1} \otimes \hat{y}^2 \otimes \mathbb{1} \rangle$:

```
In[634]:= Conjugate[γ].(KroneckerProduct[idM, xM.xM, idS].γ) //Re
Out[634]= 0.0358875
```

In order to study the spatial variation of the spin (the expectation value of the spin degree of freedom if the particle is detected at a specific spatial location), we calculate the reduced density matrix of the spin degree of freedom at a specific grid point (x_i, y_j) of the position grid.[3] For this, we first project the ground-state wavefunction γ onto the spatial grid point at $x = x_i$ and $y = y_j$ using the projector $|i\rangle\langle i| \otimes |j\rangle\langle j|$ in the momentum representation:

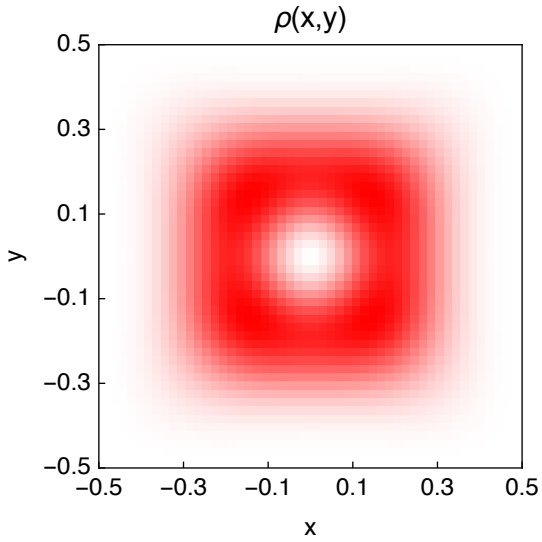
```
In[635]:= ΠP[j_] := SparseArray[{j, j} -> 1, {nmax, nmax}]
In[636]:= ΠM[j_] := FourierDST[ΠP[j], 1]
In[637]:= gP[i_,j_] := KroneckerProduct[ΠM[i], ΠM[j], idS].γ
```

Tracing out the spatial degrees of freedom with the procedure of section 2.4.3 gives the 2×2 spin density matrix at the desired grid point,

```
In[638]:= RsP[i_, j_] := traceout[gP[i,j], nmax^2]
```

The trace Tr[RsP[i,j]] of such a reduced density matrix gives the probability of finding the particle at grid point (x_i, y_j):

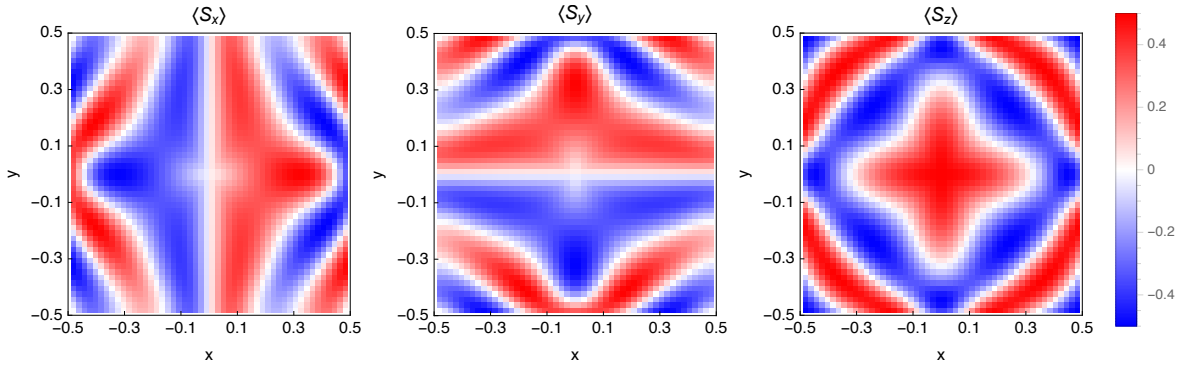
[3] Naturally, the following calculations would be simpler if we had represented the ground state in the position basis; however, we use this opportunity to show how to calculate in the momentum basis.



We can extract more information from these reduced spin density matrices: the magnetization (mean spin direction) at a grid point has the Cartesian components

```
In[639]:= mxP[i_, j_] := Re[Tr[RsP[i,j].sx]/Tr[RsP[i,j]]]
In[640]:= myP[i_, j_] := Re[Tr[RsP[i,j].sy]/Tr[RsP[i,j]]]
In[641]:= mzP[i_, j_] := Re[Tr[RsP[i,j].sz]/Tr[RsP[i,j]]]
```

Plotting these components over the entire grid shows interesting patterns of the mean spin orientation (magnetization) in the ground state:



5.2.1 exercises

Q5.2 While the Hamiltonian of Equation (5.16) and Equation (5.17) contains the distinct operators $\hat{p}_x$ and $\hat{p}_y$, the Mathematica form of the this Hamiltonian assembled in `In[629]` contains the same matrix `pM` representing both $\hat{p}_x$ and $\hat{p}_y$. Why is this so? What distinguishes the Mathematica representations of these two operators?

5.3 phase-space dynamics in the Jaynes–Cummings model [code]

As a final example, we study the interaction of an atom with the light field in an optical cavity. The atom is assumed to have only two internal states: the ground state $|g\rangle$ and some excited state $|e\rangle$. The atomic state is described as a (pseudo-)spin-1/2 system with the operators (see **Q5.3**)

$$\hat{S}_x = \frac{|e\rangle\langle g| + |g\rangle\langle e|}{2}, \qquad \hat{S}_y = \frac{|e\rangle\langle g| - |g\rangle\langle e|}{2\mathrm{i}}, \qquad \hat{S}_z = \frac{|e\rangle\langle e| - |g\rangle\langle g|}{2}, \tag{5.18}$$

as well as $\hat{S}^{\pm} = \hat{S}_x \pm \mathrm{i}\hat{S}_y$ (see section 3.2). The cavity field is assumed to consist of only one mode, described with creation and annihilation operators $\hat{a}^\dagger$ and $\hat{a}$, respectively; all other cavity modes are assumed to be so far off-resonant that they are not coupled to the atom.

The Jaynes–Cummings Hamiltonian[4] describing the combined system, as well as the coupling between the atom and the cavity field, is

$$\hat{\mathcal{H}}_{\mathrm{JC}} = \underbrace{\hbar\omega_{\mathrm{a}}\hat{S}_z}_{\text{atom}} + \underbrace{\hbar\omega_{\mathrm{c}}(\hat{a}^\dagger\hat{a} + \frac{1}{2})}_{\text{cavity field}} + \underbrace{\hbar g(\hat{S}^+\hat{a} + \hat{a}^\dagger\hat{S}^-)}_{\text{coupling}}. \tag{5.19}$$

- The atomic Hamiltonian describes the energy difference $\hbar\omega_{\mathrm{a}}$ between the two internal states of the atom.
- The cavity field Hamiltonian decribes the energy of $\hat{n} = \hat{a}^\dagger\hat{a}$ photons in the cavity mode, each photon carrying an energy $\hbar\omega_{\mathrm{c}}$.
- The coupling term describes the deexcitation of the field $\hat{a}$ together with the excitation of the atom $\hat{S}^+$, as well as the reverse process of the excitation of the field $\hat{a}^\dagger$ together with the deexcitation of the atom $\hat{S}^-$ (see **Q5.4**).

The cavity mode of the Jaynes–Cummings model is usually studied in the Fock basis of definite photon number, using harmonic-oscillator eigenfunctions as basis states. Here we take an alternative approach and look at the $X - P$ phase space spanned by the dimensionless quadrature operators $\hat{X}$ and $\hat{P}$,[5] which are related to the creation and annihilation operators $\hat{a}^\dagger$ and $\hat{a}$ via

$$\begin{aligned} \hat{X} &= \frac{\hat{a} + \hat{a}^\dagger}{\sqrt{2}} & \hat{a} &= \frac{\hat{X} + \mathrm{i}\hat{P}}{\sqrt{2}} = \frac{\hat{X} + \frac{\partial}{\partial\hat{X}}}{\sqrt{2}} \\ \hat{P} &= -\mathrm{i}\frac{\partial}{\partial\hat{X}} = \frac{\hat{a} - \hat{a}^\dagger}{\mathrm{i}\sqrt{2}} & \hat{a}^\dagger &= \frac{\hat{X} - \mathrm{i}\hat{P}}{\sqrt{2}} = \frac{\hat{X} - \frac{\partial}{\partial\hat{X}}}{\sqrt{2}} \end{aligned} \tag{5.20}$$

with the commutators $[\hat{a}, \hat{a}^\dagger] = 1$ and $[\hat{X}, \hat{P}] = \mathrm{i}$ (see **Q5.5**). We note that the quadrature $\hat{X}$ is the amplitude of the electromagnetic field of the cavity mode, and $\hat{P}$ its conjugate momentum; there is no motion in real space in this problem, only in amplitude space. Using these quadrature operators, we write the Jaynes–Cummings Hamiltonian as (see **Q5.6**)

$$\begin{aligned} \hat{\mathcal{H}}_{\mathrm{JC}} &= \hbar\omega_{\mathrm{a}}\hat{S}_z + \hbar\omega_{\mathrm{c}}(\frac{1}{2}\hat{P}^2 + \frac{1}{2}\hat{X}^2) + \sqrt{2}\hbar g(\hat{X}\hat{S}_x - \hat{P}\hat{S}_y) \\ &= \hbar\omega_{\mathrm{a}}\mathbb{1}\otimes\hat{S}_z + \hbar\omega_{\mathrm{c}}(\frac{1}{2}\hat{P}^2 + \frac{1}{2}\hat{X}^2)\otimes\mathbb{1} + \sqrt{2}\hbar g(\hat{X}\otimes\hat{S}_x - \hat{P}\otimes\hat{S}_y), \end{aligned} \tag{5.21}$$

where we have made its tensor-product structure explicit in the second line. To assemble this Hamiltonian in Mathematica, we define the Hilbert space to be the tensor product of the $X - P$ phase space and the spin-1/2 space, in this order.

The phase space is defined as before (see chapter 4) in a calculation box $X \in [-\frac{a}{2}, \frac{a}{2}]$ divided into a grid of $n_{\max} + 1$ intervals. We choose a such that the state fits well into the box (considering that the ground state of the cavity field has a size $\langle\hat{X}^2\rangle^{1/2} = \langle\hat{P}^2\rangle^{1/2} = 1/\sqrt{2}$), and we choose $n_{\max}$ such that the Wigner quasi-probability distribution plots have equal ranges in X and P. Naturally, any other values of a and $n_{\max}$ can be chosen.

```
In[642]:= ħ = 1;   (* natural units *)
In[643]:= a = 10;
In[644]:= nmax = Round[a^2/π]
Out[644]= 32
In[645]:= Δ = a/(nmax+1);
```

[4] See https://en.wikipedia.org/wiki/Jaynes-Cummings_model.

[5] In a harmonic oscillator of mass m and angular frequency ω, we usually introduce the position operator $\hat{x} = \sqrt{\frac{\hbar}{m\omega}}\hat{X}$ and the momentum operator $\hat{p} = \sqrt{\hbar m\omega}\hat{P}$. Here we restrict our attention to the dimensionless quadratures $\hat{X}$ and $\hat{P}$.

We represent the phase space in the position basis. The $\hat{X}$ quadrature operator is defined as in Equation (4.24),

```
In[646]:= xgrid = a*(Range[nmax]/(nmax+1) - 1/2);
In[647]:= X = SparseArray[Band[{1, 1}] -> xgrid];
```

The definition of the $\hat{P}$ quadrature operator follows In[444], with $\hat{P}^2$ defined directly through In[423] for better accuracy at finite n_{max}:

```
In[648]:= P = FourierDST[SparseArray[{n1_, n2_} /; OddQ[n1-n2] ->
            4*I*n1*n2)/(a*(n2^2-n1^2)), {nmax, nmax}], 1];
In[649]:= P2 = FourierDST[SparseArray[Band[{1, 1}] -> Range[nmax]^2*π^2/a^2], 1];
```

Finally, the phase-space identity operator is

```
In[650]:= idX = IdentityMatrix[nmax, SparseArray];
```

The operators on the pseudo-spin degree of freedom are defined directly from the Pauli matrices instead of using the general definitions of Equation (3.1) and Equation (3.2):

```
In[651]:= {Sx, Sy, Sz} = Table[SparseArray[PauliMatrix[i]/2], {i, 3}];
In[652]:= idS = IdentityMatrix[2, SparseArray];
```

The Hamiltonian of Equation (5.21) is assembled from three parts:

```
In[653]:= Ha = KroneckerProduct[idX, Sz];
In[654]:= Hc = KroneckerProduct[X.X/2 + P2/2, idS];
In[655]:= Hint = Sqrt[2]*(KroneckerProduct[X,Sx]-Re[KroneckerProduct[P,Sy]]);
In[656]:= HP[ωa_, ωc_, g_] = ħ*ωa*Ha + ħ*ωc*Hc + ħ*g*Hint;
```

Remember that we use `P2` instead of `P.P` for the operator $\hat{P}^2$ for better accuracy. We use the `Re` operator in In[655] to eliminate the imaginary parts, which are zero by construction but render the expression `Complex`-valued nonetheless.

In the Mathematica notebook attached to this section, the dynamics induced by this time-independent Hamiltonian is studied in the weak and strong coupling regimes, using the technique of section 2.3.4 to propagate the initial wavefunction.

Given a calculated space⊗spin wavefunction `ψ` (a vector of $2n_{\text{max}}$ complex numbers), we calculate the $n_{\text{max}} \times n_{\text{max}}$ reduced density matrix of the phase-space degree of freedom (cavity field) with In[257], tracing out the spin degree of freedom (the last 2 dimensions):

```
In[657]:= ρX = traceout[ψ, -2];
```

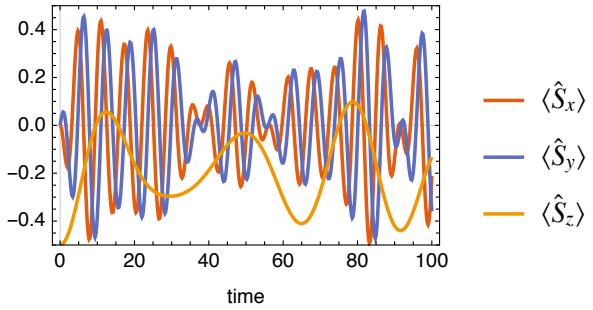
Similarly, we calculate the 2×2 reduced density matrix of the spin degree of freedom (atomic state) with In[256], tracing out the phase-space degree of freedom (the first n_{max} dimensions):

```
In[658]:= ρS = traceout[ψ, nmax];
```

Expectation values in the field or spin degrees of freedom are then easily calculated from these reduced density matrices.

To illustrate these techniques, we calculate the time-dependent wavefunction in the resonant weak-coupling regime ($\omega_a = \omega_c = 1$, $g = 0.1$; initial state: coherent field state at $\langle \hat{X} \rangle = \sqrt{2}$ and $\langle \hat{P} \rangle = 0$, spin down). First we show the time-dependence of the atomic spin expectation values, calculated from a reduced spin density matrix with

```
In[659]:= {Tr[ρS.Sx], Tr[ρS.Sy], Tr[ρS.Sz]}
```

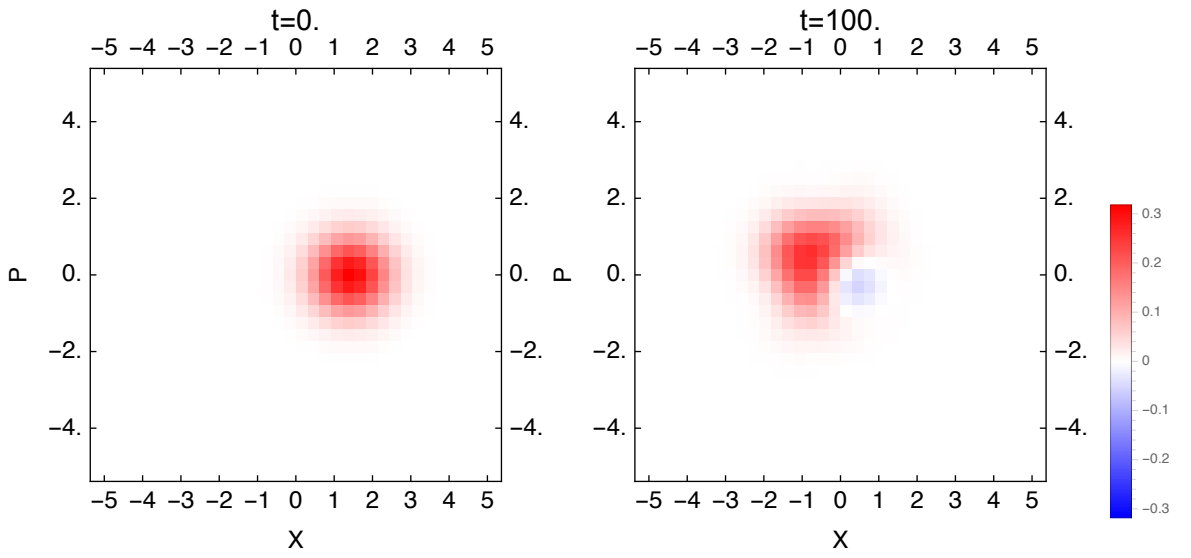


Observations:

- At $t = 0$ we recognize the initial spin-down state: $\langle \hat{S}_x \rangle = \langle \hat{S}_y \rangle = 0$ and $\langle \hat{S}_z \rangle = -\frac{1}{2}$.
- The S_x and S_y spin components rotate rapidly due to the pseudo-spin excitation energy $\hbar\omega_a$ (phase factor $e^{-i\omega_a t}$). They are 90° out of phase.
- The S_z spin component has a complicated time dependence. Since the atomic energy is $\hbar\omega_a \langle \hat{S}_z \rangle$, this curve shows the energy flowing between the atom and the cavity light field.

The phase-space Wigner quasi-probability distribution of the cavity field, calculated using In[481] from the reduced phase space density matrix of In[657], using the same weak-coupling conditions as above, is plotted here at two evolution times:

```
In[660]:= WignerDistributionPlot[ρX, {-a/2, a/2}]
```



Observations:

- At $t = 0$ we recognize the initial state: a coherent state (circular Gaussian of minimal area $\langle \hat{X}^2 \rangle = \langle \hat{P}^2 \rangle = \frac{1}{2}$) displaced by $\delta = \sqrt{2}$ in the X-direction, implying $\delta^2/2 = 1$ photon present initially.
- At $t = 100$ the structure of the Wigner distribution has taken on a qualitatively different shape, including a significant negative-valued region. Such negative regions are forbidden in classical phase-space distributions and hence indicate an essentially quantum-mechanical state.

5.3.1 exercises

Q5.3 Show that the operators of Equation (5.18) represent a pseudo-spin-1/2, like in **Q3.2**.

Q5.4 Express $\hat{S}^{\pm}$ in terms of $|g\rangle$, $|e\rangle$, $\langle g|$, and $\langle e|$.

Q5.5 Show that $[\hat{X}, \hat{P}] = \mathrm{i}$ using Equation (5.20) and assuming that $[\hat{a}, \hat{a}^{\dagger}] = 1$.

Q5.6 Show that Equation (5.21) follows from Equation (5.19) using Equation (5.20).

list of attached notebooks

If you cannot see the hyperlinks to the embedded Mathematica notebooks (white space instead of clickable links between the orange square brackets on the target pages), please use the Adobe® Acrobat® Reader® to view this document.

index

solutions to exercises

Chapter 1 Wolfram language overview

Q1.1 (page 2)

```
In[661]:= N[Zeta[3]]
Out[661]= 1.20206
```

Q1.2 (page 2)

```
In[662]:= %^2
Out[662]= 1.44494
```

Q1.3 (page 3)

```
In[663]:= Integrate[Sin[x]*Exp[-x], {x, 0, Infinity}]
Out[663]= 1/2
```

Q1.4 (page 3)

```
In[664]:= N[π, 1000]
Out[664]= 3.14159265358979323846264338327950288419716939937510582097494459230781640628620
          899862803482534211706798214808651328230664709384460955058223172535940812848
          111745028410270193852110555964462294895493038196442881097566593344612847564823
          378678316527120190914564856692346034861045432664821339360726024914127372458700
          660631558817488152092096282925409171536436789259036001133053054882046652138414
          695194151160943305727036575959195309218611738193261179310511854807446237996274
          956735188575272489122793818301194912983367336244065664308602139494639522473719
          070217986094370277053921717629317675238467481846766940513200056812714526356082
          778577134275778960917363717872146844090122495343014654958537105079227968925892
          352019956112129021960864034418159813629774771309960518707211349999983729780499
          510597317328160963185950244594553469083026425223082533446850352619311881710100
          031378387528865875332083814206171776691473035982534904287554687311595628638823
          537875937519577818577805321712268066130019278766111959092164201995
```

Q1.5 (page 3)

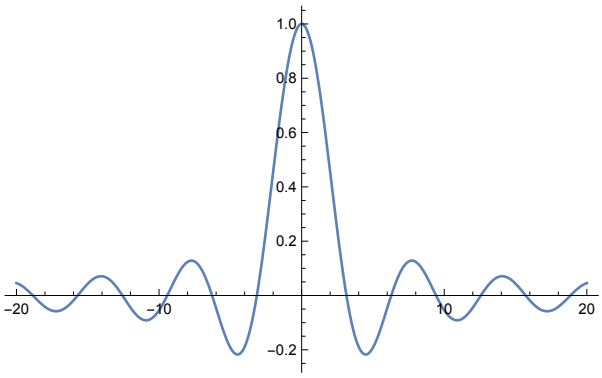
```
In[665]:= ClebschGordan[{100, 10}, {200, -12}, {110, -2}]
Out[665]= 826129779849910936101374227909252176768 1*
          Sqrt[769248995636473/29722486922289527474028523218044627174628912734745629147957669733897130076853320942746928207329]/14
In[666]:= % //N
Out[666]= 0.0949317
```

Q1.6 (page 3)

```
In[667]:= Limit[Sin[x]/x, x -> 0]
Out[667]= 1
```

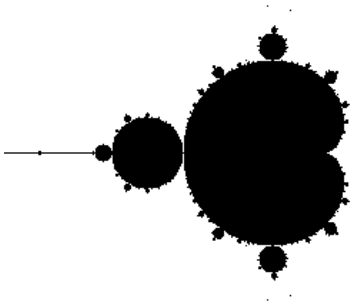
Q1.7 (page 3)

```
In[668]:= Plot[Sin[x]/x, {x, -20, 20}, PlotRange -> All]
```



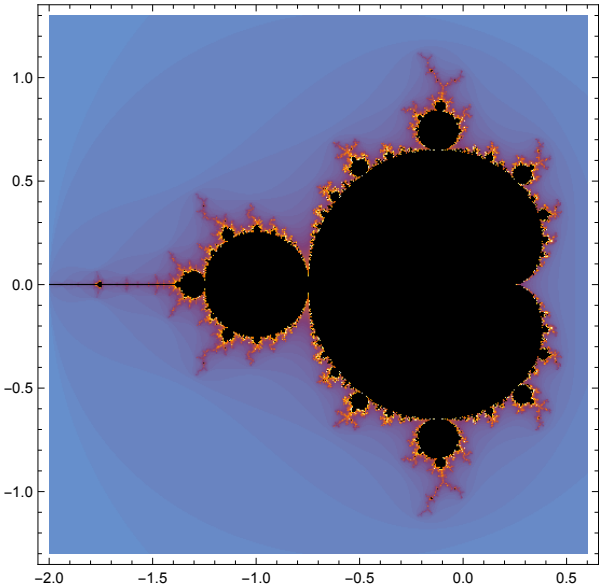
Q1.8 (page 3)

```
In[669]:= F[c_, imax_] := Abs[NestWhile[#^2+c&, 0., Abs[#] <= 2 &, 1, imax]] <= 2
In[670]:= With[{n = 100, imax = 1000},
          Graphics[Raster[Table[Boole[!F[x+I*y,imax]],{y,-2,2,1/n},{x,-2,2,1/n}]]]]
```



Q1.9 (page 3)

```
In[671]:= MandelbrotSetPlot[]
```



Q1.10 (page 4) In general, the definition of `x` depends on the values of `u` and `v` at the time of the definition of `x`, whereas `y` depends on the values at the time of using the symbol `y`. The second case below, however, needs special attention since the values of `u` and `v` are not defined at the time when `x` is defined.

- When `u` and `v` are already defined before `x` and `y` are defined, then `x` and `y` return the same value:

```
In[672]:=Clear[x, y, u, v];
In[673]:=u = 3; v = 7;
In[674]:=x = u+v; y := u+v;
In[675]:={x, y}
Out[675]={10, 10}
In[676]:=?x
        x=10
In[677]:=?y
        y:=u+v
```

- When `u` and `v` are defined after `x` and `y` are defined, then `x` and `y` also return the same value. Notice, however, that the definition of `x` is not static and thus depends on the values of `u` and `v` at the time of usage:

```
In[678]:=Clear[x, y, u, v];
In[679]:=x = u+v; y := u+v;
In[680]:=u = 3; v = 7;
In[681]:={x, y}
Out[681]={10, 10}
In[682]:=?x
        x=u+v
In[683]:=?y
        y:=u+v
```

- When `u` and `v` change values after `x` and `y` are defined, then `x` and `y` differ since only `y` reflects the new values of `u` and `v`:

```
In[684]:=Clear[x, y, u, v];
In[685]:=u = 3; v = 7;
In[686]:=x = u+v; y := u+v;
In[687]:=u = 8; v = 9;
In[688]:={x, y}
```

```
Out[688]= {10, 17}
In[689]:= ?x
          x=10
In[690]:= ?y
          y:=u+v
```

Q1.11 (page 6)

```
In[691]:= N[E]
Out[691]= 2.71828
In[692]:= N@E
Out[692]= 2.71828
In[693]:= E //N
Out[693]= 2.71828
```

Q1.12 (page 7)

```
In[694]:= Total[Range[123, 9968]]
Out[694]= 49677993
```

Q1.13 (page 7)

```
In[695]:= Module[{i},
            i = 123;
            s = 0;
            While[s <= 10000, s += i; i++];
            i - 1]
Out[695]= 187
```

Q1.14 (page 9)

```
In[696]:= f = #1*#2*#3 &;
```

Q1.15 (page 9)

```
In[697]:= a = {0.1, 0.9, 2.25, -1.9};
In[698]:= sa = Map[Sin[#]^2 &, a]
Out[698]= {0.00996671, 0.613601, 0.605398, 0.895484}
```

Q1.16 (page 9) The `Total` function is the same as applying `Plus` to a list:

```
In[699]:= Apply[Plus, sa]
In[700]:= 2.12445
In[701]:= Plus@@sa
In[702]:= 2.12445
In[703]:= Total[sa]
In[704]:= 2.12445
```

Q1.17 (page 15) All built-in symbols, like `Echo`, are protected in order to prevent accidental modification. Trying to modify `Echo` without unprotecting it first gives an error:

```
In[705]:= Echo = #1 &
          Set:  Symbol Echo is Protected.
Out[705]= #1 &
```

Q1.18 (page 15) See `In[128]` and `In[129]`: the full forms of `a/b` and `x_/y_` are similar and match,

```
In[706]:=FullForm[a/b]
Out[706]=Times[a, Power[b, -1]]
In[707]:=FullForm[x_/y_]
Out[707]=Times[Pattern[x, Blank[]], Power[Pattern[y, Blank[]], -1]]
```

while the full form of 2/3 is different and does not match the pattern for replacements,

```
In[708]:=FullForm[2/3]
Out[708]=Rational[2, 3]
```

Q1.19 (page 18) Not all delayed assignments can be replaced by immediate ones. Whenever an immediate assignment can be used, it tends to be faster.

1. = and := work equally well.
2. = and := work equally well.
3. = and := work equally well.
4. = and := work equally well. There is a significant difference though: while the delayed assignment executes as a product, the immediate assignment is simplified at the moment of definition to a factorial, which then executes much faster:

```
In[709]:=f[n_] = Product[i, {i, n}]
Out[709]=n!
```

5. Immediate assignment breaks the recursion, which cannot be executed at definition time.
6. = and := work equally well.
7. = and := work equally well.
8. Immediate assignment breaks the Do loop, which cannot be executed at definition time.
9. Immediate assignment breaks the For loop: since n is not defined at definition time, the comparison i<=n fails at the first iteration and the result is always f[n_]=1.
10. Immediate assignment breaks the Range command since n is not defined at definition time.
11. Immediate assignment breaks the Range command since n is not defined at definition time.
12. Immediate assignment breaks the Array command since n is not defined at definition time.
13. Immediate assignment breaks the Range command since n is not defined at definition time.
14. Immediate assignment always gives f[n_]=1 since the repeated replacement fails.
15. Immediate assignment breaks the Range command since n is not defined at definition time.
16. Immediate assignment always gives f[n_]=1 since the repeated replacement fails.
17. = and := work equally well.

Q1.20 (page 18) Not all delayed rules can be replaced by immediate ones. Whenever an immediate rule can be used, it tends to be faster.

14. -> and :> work equally well.
16. Immediate rule (->) breaks the Table command since m is not defined at definition time.

Q1.21 (page 18) In the recursive definitions 5 and 6, memoization gives a dramatic speedup, as it remembers intermediate results in the recursion. In the other examples, memoization only helps when the function is called repeatedly with the same argument.

Q1.22 (page 18) Using a built-in function:

```
In[710]:=Table[Fibonacci[n], {n, 100}]
```

Even more directly, by using the Listable attribute of the Fibonacci function:

```
In[711]:=Fibonacci[Range[100]]
```

Recursive with memoization:

```
In[712]:= g[1] = g[2] = 1;
In[713]:= g[n_] := g[n] = g[n-1] + g[n-2]
In[714]:= Table[g[n], {n, 100}]
```

Iterative construction of the list:

```
In[715]:= L = {1, 1};
In[716]:= Do[AppendTo[L, L[[-1]] + L[[-2]]], {98}];
In[717]:= L
```

Q1.23 (page 23) The eigenvectors are orthogonal, but not necessarily normalized.
Eigensystem of $\hat{\sigma}_x$:

```
In[718]:= {eval, evec} = Eigensystem[PauliMatrix[1]]
Out[718]= {{-1, 1}, {{-1, 1}, {1, 1}}}
In[719]:= Normalize /@ evec
Out[719]= {{-1/Sqrt[2], 1/Sqrt[2]}, {1/Sqrt[2], 1/Sqrt[2]}}
```

Eigensystem of $\hat{\sigma}_y$:

```
In[720]:= {eval, evec} = Eigensystem[PauliMatrix[2]]
Out[720]= {{-1, 1}, {{I, 1}, {-I, 1}}}
In[721]:= Normalize /@ evec
Out[721]= {{I/Sqrt[2], 1/Sqrt[2]}, {-I/Sqrt[2], 1/Sqrt[2]}}
```

Eigensystem of $\hat{\sigma}_z$:

```
In[722]:= {eval, evec} = Eigensystem[PauliMatrix[3]]
Out[722]= {{-1, 1}, {{0, 1}, {1, 0}}}
```

Q1.24 (page 23) The tensor index dimensions do not match:

```
In[723]:= TensorContract[u, {3, 4}]
        TensorContract:  Contraction levels {3,4} have different dimensions {3,2}.
```

Chapter 2 quantum mechanics: states and operators

Q2.1 (page 29) We use the computational basis $\{|\uparrow\rangle, |\downarrow\rangle\}$, in which the two given basis functions are

```
In[724]:= up[θ_,φ] = {Cos[θ/2], E^(I*φ)*Sin[θ/2]};
In[725]:= dn[θ_,φ] = {-E^(-I*φ)*Sin[θ/2], Cos[θ/2]};
```

The corresponding $\langle\Uparrow_{\vartheta,\varphi}|$ and $\langle\Downarrow_{\vartheta,\varphi}|$ are calculated with `Conjugate` (see section 1.11).

1. Calculate $\langle\Uparrow_{\vartheta,\varphi}|\Uparrow_{\vartheta,\varphi}\rangle = 1$, $\langle\Uparrow_{\vartheta,\varphi}|\Downarrow_{\vartheta,\varphi}\rangle = 0$, $\langle\Downarrow_{\vartheta,\varphi}|\Uparrow_{\vartheta,\varphi}\rangle = 0$, $\langle\Downarrow_{\vartheta,\varphi}|\Downarrow_{\vartheta,\varphi}\rangle = 1$:

```
In[726]:= Conjugate[up[θ,φ]].up[θ,φ] //ComplexExpand //FullSimplify
Out[726]= 1
In[727]:= Conjugate[up[θ,φ]].dn[θ,φ] //ComplexExpand //FullSimplify
Out[727]= 0
In[728]:= Conjugate[dn[θ,φ]].up[θ,φ] //ComplexExpand //FullSimplify
Out[728]= 0
In[729]:= Conjugate[dn[θ,φ]].dn[θ,φ] //ComplexExpand //FullSimplify
Out[729]= 1
```

2. Construct the ket-bra products with `KroneckerProduct`:

```
In[730]:= KroneckerProduct[up[θ,φ], Conjugate[up[θ,φ]]] +
          KroneckerProduct[dn[θ,φ], Conjugate[dn[θ,φ]]] //
          ComplexExpand //FullSimplify
Out[730]= {{1, 0}, {0, 1}}
```

3. $|\Uparrow\rangle = |\Uparrow_{\vartheta,\varphi}\rangle\langle\Uparrow_{\vartheta,\varphi}|\Uparrow\rangle + |\Downarrow_{\vartheta,\varphi}\rangle\langle\Downarrow_{\vartheta,\varphi}|\Uparrow\rangle = \cos(\vartheta/2)|\Uparrow_{\vartheta,\varphi}\rangle - e^{i\varphi}\sin(\vartheta/2)|\Downarrow_{\vartheta,\varphi}\rangle$:

```
In[731]:= Cos[θ/2]*up[θ,φ] - E^(I*φ)*Sin[θ/2]*dn[θ,φ] //FullSimplify
Out[731]= {1, 0}
```

$|\Downarrow\rangle = |\Uparrow_{\vartheta,\varphi}\rangle\langle\Uparrow_{\vartheta,\varphi}|\Downarrow\rangle + |\Downarrow(\vartheta,\varphi)\rangle\langle\Downarrow_{\vartheta,\varphi}|\Downarrow\rangle = e^{-i\varphi}\sin(\vartheta/2)|\Uparrow_{\vartheta,\varphi}\rangle + \cos(\vartheta/2)|\Downarrow_{\vartheta,\varphi}\rangle$:

```
In[732]:= E^(-I*φ)*Sin[θ/2]*up[θ,φ] + Cos[θ/2]*dn[θ,φ] //FullSimplify
Out[732]= {0, 1}
```

4. The Pauli operators are defined in Mathematica in our computational basis with the `PauliMatrix` command.

   The matrix elements of the Pauli operator $\hat{\sigma}_x$ are

```
In[733]:= sx = PauliMatrix[1];
In[734]:= Conjugate[up[θ,φ]].sx.up[θ,φ] //ComplexExpand //FullSimplify
Out[734]= Sin[θ]*Cos[φ]
In[735]:= Conjugate[up[θ,φ]].sx.dn[θ,φ] //ComplexExpand //FullSimplify
Out[735]= Exp[-I*φ]*(Cos[θ]*Cos[φ]+I*Sin[φ])
In[736]:= Conjugate[dn[θ,φ]].sx.up[θ,φ] //ComplexExpand //FullSimplify
Out[736]= Exp[I*φ]*(Cos[θ]*Cos[φ]-I*Sin[φ])
In[737]:= Conjugate[dn[θ,φ]].sx.dn[θ,φ] //ComplexExpand //FullSimplify
Out[737]= -Sin[θ]*Cos[φ]
In[738]:= sx == Sin[θ]*Cos[φ] * KroneckerProduct[up[θ,φ], Conjugate[up[θ,φ]]] +
          E^(-I*φ)*(Cos[θ]*Cos[φ]+I*Sin[φ]) *
          KroneckerProduct[up[θ,φ], Conjugate[dn[θ,φ]]] +
          E^(I*φ)*(Cos[θ]*Cos[φ]-I*Sin[φ]) *
          KroneckerProduct[dn[θ,φ], Conjugate[up[θ,φ]]] -
          Sin[θ]*Cos[φ] * KroneckerProduct[dn[θ,φ], Conjugate[dn[θ,φ]]] //
          ComplexExpand //FullSimplify
Out[738]= True
```

   The matrix elements of the Pauli operator $\hat{\sigma}_y$ are

```
In[739]:= sy = PauliMatrix[2];
In[740]:= Conjugate[up[θ,φ]].sy.up[θ,φ] //ComplexExpand //FullSimplify
Out[740]= Sin[θ]*Sin[φ]
In[741]:= Conjugate[up[θ,φ]].sy.dn[θ,φ] //ComplexExpand //FullSimplify
Out[741]= Exp[-I*φ]*(Cos[θ]*Sin[φ]-I*Cos[φ])
In[742]:= Conjugate[dn[θ,φ]].sy.up[θ,φ] //ComplexExpand //FullSimplify
Out[742]= Exp[I*φ]*(Cos[θ]*Sin[φ]+I*Cos[φ])
In[743]:= Conjugate[dn[θ,φ]].sy.dn[θ,φ] //ComplexExpand //FullSimplify
Out[743]= -Sin[θ]*Sin[φ]
In[744]:= sy == Sin[θ]*Sin[φ] * KroneckerProduct[up[θ,φ], Conjugate[up[θ,φ]]] +
          E^(-I*φ)*(Cos[θ]*Sin[φ]-I*Cos[φ]) *
          KroneckerProduct[up[θ,φ], Conjugate[dn[θ,φ]]] +
          E^(I*φ)*(Cos[θ]*Sin[φ]+I*Cos[φ]) *
          KroneckerProduct[dn[θ,φ], Conjugate[up[θ,φ]]] -
          Sin[θ]*Sin[φ] * KroneckerProduct[dn[θ,φ], Conjugate[dn[θ,φ]]] //
          ComplexExpand //FullSimplify
Out[744]= True
```

The matrix elements of the Pauli operator $\hat{\sigma}_z$ are

```
In[745]:= sz = PauliMatrix[3];
In[746]:= Conjugate[up[θ,φ]].sz.up[θ,φ] //ComplexExpand //FullSimplify
Out[746]= Cos[θ]
In[747]:= Conjugate[up[θ,φ]].sz.dn[θ,φ] //ComplexExpand //FullSimplify
Out[747]= -Exp[-I*φ]*Sin[θ]
In[748]:= Conjugate[dn[θ,φ]].sz.up[θ,φ] //ComplexExpand //FullSimplify
Out[748]= -Exp[I*φ]*Sin[θ]
In[749]:= Conjugate[dn[θ,φ]].sz.dn[θ,φ] //ComplexExpand //FullSimplify
Out[749]= -Cos[θ]
In[750]:= sz == Cos[θ] * KroneckerProduct[up[θ,φ], Conjugate[up[θ,φ]]] -
          E^(-I*φ)*Sin[θ] * KroneckerProduct[up[θ,φ], Conjugate[dn[θ,φ]]] -
          E^(I*φ)*Sin[θ] * KroneckerProduct[dn[θ,φ], Conjugate[up[θ,φ]]] -
          Cos[θ] * KroneckerProduct[dn[θ,φ], Conjugate[dn[θ,φ]]] //
          ComplexExpand //FullSimplify
Out[750]= True
```

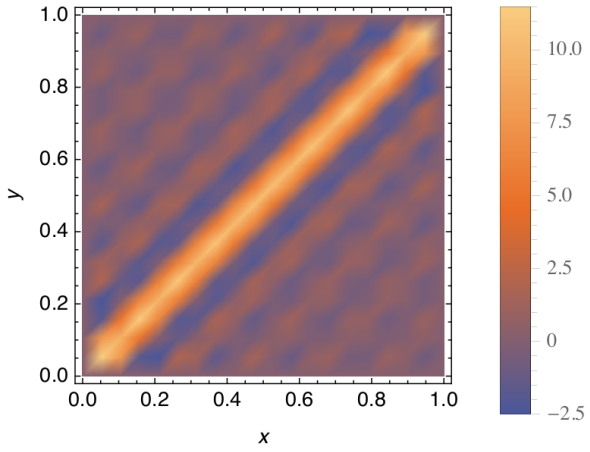
5. We check the eigenvalue equations with eigenvalues ± 1:

```
In[751]:= s = sx*Sin[θ]*Cos[φ] + sy*Sin[θ]*Sin[φ] + sz*Cos[θ];
In[752]:= Eigenvalues[s]
Out[752]= {-1, 1}
In[753]:= s.up[θ,φ] == up[θ,φ] //FullSimplify
Out[753]= True
In[754]:= s.dn[θ,φ] == -dn[θ,φ] //FullSimplify
Out[754]= True
```

Q2.2 (page 29)

1. Since $\sum_{n=1}^{\infty} |n\rangle\langle n| = \mathbb{1}$, we have $P_\infty(x,y) = \langle x|\mathbb{1}|y\rangle = \langle x|y\rangle = \delta(x-y)$.
2. $P_{n_{\max}}(x,y) = \langle x| \left[\sum_{n=1}^{n_{\max}} |n\rangle\langle n|\right] |y\rangle = \sum_{n=1}^{n_{\max}} \langle x|n\rangle\langle n|y\rangle = 2\sum_{n=1}^{n_{\max}} \sin(n\pi x)\sin(n\pi y)$:

```
In[755]:= With[{nmax = 10},
          P[x_, y_] = 2*Sum[Sin[n*π*x]*Sin[n*π*y], {n, nmax}];
          DensityPlot[P[x, y], {x, 0, 1}, {y, 0, 1},
          PlotRange -> All, PlotPoints -> 2*nmax]]
```



3. The operator $\hat{\Pi}_{n_{\max}} = \sum_{n=1}^{n_{\max}} |n\rangle\langle n|$ is the projector onto the computational subspace (see section 2.1.1). The function $P_{n_{\max}}(x,y) = \langle x|\hat{\Pi}_{n_{\max}}|y\rangle$ is its real-space representation. Since the plot of $P_{n_{\max}}(x,y)$ has a finite spatial resolution (*i.e.*, no structure at length scales smaller than $1/n_{\max}$), we see that this projection operator $\hat{\Pi}_{n_{\max}}$ is associated with a spatial smoothing operation.

Q2.3 (page 30) See **Q2.1**.

Q2.4 (page 33) Inserting Equation (2.32) into Equation (2.17) gives the quantum state

$$|\psi(t)\rangle = \exp\left[-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t}\hat{\mathcal{H}}(s)\mathrm{d}s\right]|\psi(t_0)\rangle \tag{1}$$

We calculate its time-derivative with the chain rule and $\frac{\mathrm{d}}{\mathrm{d}x}\left[\int_{f(x)}^{g(x)}h(x,y)\mathrm{d}y\right] = h(x,g(x))g'(x) - h(x,f(x))f'(x) + \int_{f(x)}^{g(x)}\frac{\partial h(x,y)}{\partial x}\mathrm{d}y$:

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = -\frac{\mathrm{i}}{\hbar}\hat{\mathcal{H}}(t)\exp\left[-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t}\hat{\mathcal{H}}(s)\mathrm{d}s\right]|\psi(t_0)\rangle = -\frac{\mathrm{i}}{\hbar}\hat{\mathcal{H}}(t)|\psi(t)\rangle, \tag{2}$$

which is the Schrödinger equation (2.16).

Q2.5 (page 33) The Hamiltonian is

```
In[756]:= {sx,sy,sz} = Table[PauliMatrix[i], {i, 3}];
In[757]:= H = Sin[θ]*Cos[φ]*sx + Sin[θ]*Sin[φ]*sy + Cos[θ]*sz //FullSimplify
Out[757]= {{Cos[θ], E^(-I*φ)*Sin[θ]}, {E^(I*φ)*Sin[θ], -Cos[θ]}}
```

and the propagator is calculated from Equation (2.34)

```
In[758]:= U = MatrixExp[-I*(t-t0)/ħ*H] //FullSimplify
Out[758]= {{Cos[(t-t0)/ħ]-I*Cos[θ]*Sin[(t-t0)/ħ], -I*E^(-I*φ)*Sin[θ]*Sin[(t-t0)/ħ]},
          {-I*E^(I*φ)*Sin[θ]*Sin[(t-t0)/ħ], Cos[(t-t0)/ħ]+I*Cos[θ]*Sin[(t-t0)/ħ]}}
```

Q2.6 (page 33) The definitions are ordered with decreasing specificity:

```
In[759]:= ?U
          Global‘U
          U[τ_?NumericQ] := MatrixExp[-I H N[τ]]
          U[τ_] = MatrixExp[-I H τ]
```

In this way, the more general definition `In[234]` does not override the more specific definition `In[235]`.

Q2.7 (page 38)

1. The Hamiltonian is

$$\hat{\mathcal{H}} = -\frac{\hbar^2}{2m}\left(\frac{\partial}{\partial x_1^2}+\frac{\partial}{\partial y_1^2}+\frac{\partial}{\partial z_1^2}+\frac{\partial}{\partial x_2^2}+\frac{\partial}{\partial y_2^2}+\frac{\partial}{\partial z_2^2}\right) + \frac{1}{2}m\omega^2(x_1^2+y_1^2+z_1^2+x_2^2+y_2^2+z_2^2) + g\delta(x_1-x_2)\delta(y_1-y_2)\delta(z_1-z_2) \tag{3}$$

2. For example, we could use the harmonic-oscillator basis functions that diagonalize the six degrees of freedom in the absence of coupling ($g = 0$): the states $|n\rangle$ for which

$$\left[-\frac{\hbar^2}{2m}\frac{\partial}{\partial x^2}+\frac{1}{2}m\omega^2x^2\right]|n\rangle = \hbar\omega(n+\frac{1}{2})|n\rangle, \tag{4}$$

with $n \in \mathbb{N}$. Explicitly, the position representations of these states are

$$\langle x|n\rangle = \phi_n(x) = x_0^{-1/2}\frac{H_n(x/x_0)}{\sqrt{2^n n!\sqrt{\pi}}}e^{-\frac{x^2}{2x_0^2}}. \tag{5}$$

For the six degrees of freedom we therefore propose the basis functions $|n_{x_1}, n_{y_1}, n_{z_1}, n_{x_2}, n_{y_2}, n_{z_2}\rangle$.

3. The matrix elements are

$$\begin{aligned}
&\langle n_{x_1}, n_{y_1}, n_{z_1}, n_{x_2}, n_{y_2}, n_{z_2}|\hat{\mathcal{H}}|n'_{x_1}, n'_{y_1}, n'_{z_1}, n'_{x_2}, n'_{y_2}, n'_{z_2}\rangle \\
&\qquad = \hbar\omega\delta_{n_{x_1},n'_{x_1}}\delta_{n_{y_1},n'_{y_1}}\delta_{n_{z_1},n'_{z_1}}\delta_{n_{x_2},n'_{x_2}}\delta_{n_{y_2},n'_{y_2}}\delta_{n_{z_2},n'_{z_2}}(n_{x_1}+n_{y_1}+n_{z_1}+n_{x_2}+n_{y_2}+n_{z_2}+3) \\
&+g\langle n_{x_1}, n_{y_1}, n_{z_1}, n_{x_2}, n_{y_2}, n_{z_2}|\int_{-\infty}^{\infty}\mathrm{d}x_1\mathrm{d}y_1\mathrm{d}z_1\mathrm{d}x_2\mathrm{d}y_2\mathrm{d}z_2|x_1\rangle\langle x_1|\otimes|y_1\rangle\langle y_1|\otimes|z_1\rangle\langle z_1|\otimes|x_2\rangle\langle x_2|\otimes|y_2\rangle\langle y_2|\otimes|z_2\rangle\langle z_2| \\
&\qquad \delta(x_1-x_2)\delta(y_1-y_2)\delta(z_1-z_2)|n'_{x_1}, n'_{y_1}, n'_{z_1}, n'_{x_2}, n'_{y_2}, n'_{z_2}\rangle \\
&\qquad = \hbar\omega\delta_{n_{x_1},n'_{x_1}}\delta_{n_{y_1},n'_{y_1}}\delta_{n_{z_1},n'_{z_1}}\delta_{n_{x_2},n'_{x_2}}\delta_{n_{y_2},n'_{y_2}}\delta_{n_{z_2},n'_{z_2}}(n_{x_1}+n_{y_1}+n_{z_1}+n_{x_2}+n_{y_2}+n_{z_2}+3) \\
&\qquad\quad + g\int_{-\infty}^{\infty}\mathrm{d}x_1\mathrm{d}y_1\mathrm{d}z_1\mathrm{d}x_2\mathrm{d}y_2\mathrm{d}z_2\delta(x_1-x_2)\delta(y_1-y_2)\delta(z_1-z_2) \\
&\times\phi_{n_{x_1}}(x_1)\phi_{n'_{x_1}}(x_1)\phi_{n_{y_1}}(y_1)\phi_{n'_{y_1}}(y_1)\phi_{n_{z_1}}(z_1)\phi_{n'_{z_1}}(z_1)\phi_{n_{x_2}}(x_2)\phi_{n'_{x_2}}(x_2)\phi_{n_{y_2}}(y_2)\phi_{n'_{y_2}}(y_2)\phi_{n_{z_2}}(z_2)\phi_{n'_{z_2}}(z_2) \\
&\qquad = \hbar\omega\delta_{n_{x_1},n'_{x_1}}\delta_{n_{y_1},n'_{y_1}}\delta_{n_{z_1},n'_{z_1}}\delta_{n_{x_2},n'_{x_2}}\delta_{n_{y_2},n'_{y_2}}\delta_{n_{z_2},n'_{z_2}}(n_{x_1}+n_{y_1}+n_{z_1}+n_{x_2}+n_{y_2}+n_{z_2}+3) \\
&+g\left[\int_{-\infty}^{\infty}\mathrm{d}x\phi_{n_{x_1}}(x)\phi_{n'_{x_1}}(x)\phi_{n_{x_2}}(x)\phi_{n'_{x_2}}(x)\right]\left[\int_{-\infty}^{\infty}\mathrm{d}y\phi_{n_{y_1}}(y)\phi_{n'_{y_1}}(y)\phi_{n_{y_2}}(y)\phi_{n'_{y_2}}(y)\right]\left[\int_{-\infty}^{\infty}\mathrm{d}z\phi_{n_{z_1}}(z)\phi_{n'_{z_1}}(z)\phi_{n_{z_2}}(z)\phi\right. \\
&\qquad = \hbar\omega\delta_{n_{x_1},n'_{x_1}}\delta_{n_{y_1},n'_{y_1}}\delta_{n_{z_1},n'_{z_1}}\delta_{n_{x_2},n'_{x_2}}\delta_{n_{y_2},n'_{y_2}}\delta_{n_{z_2},n'_{z_2}}(n_{x_1}+n_{y_1}+n_{z_1}+n_{x_2}+n_{y_2}+n_{z_2}+3) \\
&\qquad\quad + \frac{g}{x_0^3}R_{n_{x_1},n'_{x_1},n_{x_2},n'_{x_2}}R_{n_{y_1},n'_{y_1},n_{y_2},n'_{y_2}}R_{n_{z_1},n'_{z_1},n_{z_2},n'_{z_2}}. \qquad (6)
\end{aligned}$$

The required dimensionless integrals over products of four harmonic-oscillator eigenstates,

$$R_{a,b,c,d} = x_0\int_{-\infty}^{\infty}\mathrm{d}x\phi_a(x)\phi_b(x)\phi_c(x)\phi_d(x) = \int_{-\infty}^{\infty}\mathrm{d}\xi\frac{H_a(\xi)H_b(\xi)H_c(\xi)H_d(\xi)}{\pi\sqrt{2^{a+b+c+d}a!b!c!d!}}e^{-2\xi^2}, \qquad (7)$$

can either be calculated by analytic integration,

```
In[760]:= φ[n_, x_] = HermiteH[n, x]/Sqrt[2^n*n!*Sqrt[π]]*E^(-x^2/2);
In[761]:= R[a_Integer/;a>=0, b_Integer/;b>=0, c_Integer/;c>=0, d_Integer/;d>=0] :=
            Integrate[φ[a,x]*φ[b,x]*φ[c,x]*φ[d,x], {x, -∞, ∞}]
```

or by an explicit but hypergeometric formula[1] (much faster),

```
In[762]:= R[a_Integer/;a>=0, b_Integer/;b>=0, c_Integer/;c>=0, d_Integer/;d>=0] :=
            If[OddQ[a+b+c+d], 0,
              1/π*(-1)^((a+b-c+d)/2)*Sqrt[c!/(2a!b!d!)]*
                Gamma[(1+a-b+c-d)/2]*Gamma[(1-a+b+c-d)/2]*
                HypergeometricPFQRegularized[{(1+a-b+c-d)/2,(1-a+b+c-d)/2,-d},
                                             {1+c-d,(1-a-b+c-d)/2},1]]
```

Q2.8 (page 38)

```
In[763]:= ψ = Flatten[Table[ψ1[[i1]]*ψ2[[i2]]*ψ3[[i3]],
            {i1, Length[ψ1]}, {i2, Length[ψ2]}, {i3, Length[ψ3]}]]
```

Q2.9 (page 38)

```
In[764]:= A = Flatten[Table[a1[[i1,j1]]*a2[[i2,j2]]*a3[[i3,j3]],
            {i1, Length[a1]}, {i2, Length[a2]}, {i3, Length[a3]},
            {j1, Length[Transpose[a1]]}, {j2, Length[Transpose[a2]]},
            {j3, Length[Transpose[a3]]}], {{1,2,3}, {4,5,6}}]
```

Q2.10 (page 38) Manual calculation:

$$|\psi\rangle = [0.8|\uparrow\rangle - 0.6|\downarrow\rangle]\otimes[0.6\mathrm{i}|\uparrow\rangle + 0.8|\downarrow\rangle] = 0.48\mathrm{i}|\uparrow\uparrow\rangle + 0.64|\uparrow\downarrow\rangle - 0.36\mathrm{i}|\downarrow\uparrow\rangle - 0.48|\downarrow\downarrow\rangle, \qquad (8)$$

where $|\uparrow\downarrow\rangle = |\uparrow\rangle\otimes|\downarrow\rangle$ etc. In Mathematica, using the computational basis $\{|\uparrow\rangle, |\downarrow\rangle\}$, in this order:

[1] See http://www.ph.unimelb.edu.au/~jnnewn/cm-seminar-results/report/AnalyticIntegralOfFourHermites.pdf.

```
In[765]:= ψ1 = {0.8, -0.6};
In[766]:= ψ2 = {0.6*I, 0.8};
In[767]:= ψ = Flatten[KroneckerProduct[ψ1, ψ2]]
Out[767]= {0.+0.48*I, 0.64+0.*I, 0.-0.36*I, -0.48+0.*I}
```

The ordering of the joint basis in the `Kroneckerproduct` result is therefore $\{|\uparrow\uparrow\rangle, |\uparrow\downarrow\rangle, |\downarrow\uparrow\rangle, |\downarrow\downarrow\rangle\}$.

Q2.11 (page 38) We calculate the reduced density matrices with the `traceout` command of `In[256]` and `In[257]`:

```
In[768]:= ρ1 = traceout[ψ, -2]
Out[768]= {{0.64+0.*I, -0.48+0.*I}, {-0.48+0.*I, 0.36+0.*I}}
In[769]:= ρ2 = traceout[ψ, 2]
Out[769]= {{0.36+0.*I, 0.+0.48*I}, {0.-0.48*I, 0.64+0.*I}}
```

Since $|\psi\rangle$ is a product state, these reduced density matrices are equal to the pure states of the subsystems:

```
In[770]:= ρ1 == KroneckerProduct[ψ1, Conjugate[ψ1]]
Out[770]= True
In[771]:= ρ2 == KroneckerProduct[ψ2, Conjugate[ψ2]]
Out[771]= True
```

Chapter 3 spin and angular momentum

Q3.1 (page 41)

```
In[772]:= sx[1/2] == 1/2*PauliMatrix[1]
Out[772]= True
In[773]:= sy[1/2] == 1/2*PauliMatrix[2]
Out[773]= True
In[774]:= sz[1/2] == 1/2*PauliMatrix[3]
Out[774]= True
```

Q3.2 (page 41) We only check up to $S = 10$:

1. commutators:

```
In[775]:= Table[sx[S].sy[S]-sy[S].sx[S] == I*sz[S], {S, 0, 10, 1/2}]
Out[775]= {True,True,True,True,True,True,True,True,True,True,True,True,True,
          True,True,True,True,True,True,True,True}
In[776]:= Table[sy[S].sz[S]-sz[S].sy[S] == I*sx[S], {S, 0, 10, 1/2}]
Out[776]= {True,True,True,True,True,True,True,True,True,True,True,True,True,
          True,True,True,True,True,True,True,True}
In[777]:= Table[sz[S].sx[S]-sx[S].sz[S] == I*sy[S], {S, 0, 10, 1/2}]
Out[777]= {True,True,True,True,True,True,True,True,True,True,True,True,True,
          True,True,True,True,True,True,True,True}
```

2. spin length:

```
In[778]:= Table[sx[S].sx[S]+sy[S].sy[S]+sz[S].sz[S] == S*(S+1)*id[S], {S,0,10,1/2}]
Out[778]= {True,True,True,True,True,True,True,True,True,True,True,True,True,
          True,True,True,True,True,True,True,True}
```

3. Make sure to quit the Mathematica kernel before loading the spin-operator definitions and executing the following commands. On a MacBook Pro (Retina, 13-inch, Early 2015) with a 3.1 GHz Intel Core i7 CPU and 16 GB 1867 MHz DDR3 RAM, the limit is around $S = 10^5$ for all verifications:

```
In[779]:= s=100000;
In[780]:= sx[s].sy[s]-sy[s].sx[s] == I*sz[s] //Timing
Out[780]= {54.3985, True}
In[781]:= sy[s].sz[s]-sz[s].sy[s] == I*sx[s] //Timing
Out[781]= {58.4917, True}
In[782]:= sz[s].sx[s]-sx[s].sz[s] == I*sy[s] //Timing
Out[782]= {57.8856, True}
In[783]:= sx[s].sx[s]+sy[s].sy[s]+sz[s].sz[s] == s*(s+1)*id[s] //Timing
Out[783]= {33.5487, True}
```

Q3.3 (page 41) The expressions rapidly increase in complexity with increasing S:

```
In[784]:= n = {Sin[θ]*Cos[φ], Sin[θ]*Sin[φ], Cos[θ]};
In[785]:= With[{S=0}, MatrixExp[-I*α*n.{sx[S],sy[S],sz[S]}] //FullSimplify]
Out[785]= {{1}}
In[786]:= With[{S=1/2}, MatrixExp[-I*α*n.{sx[S],sy[S],sz[S]}] //FullSimplify]
Out[786]= {{Cos[α/2]-I*Cos[θ]*Sin[α/2], Sin[α/2]*Sin[θ]*(-I*Cos[φ]-Sin[φ])},
          {Sin[α/2]*Sin[θ]*(-I*Cos[φ]+Sin[φ]), Cos[α/2]+I*Cos[θ]*Sin[α/2]}}
In[787]:= % /. α -> 0
Out[787]= {{1, 0}, {0, 1}}
In[788]:= With[{S=1}, MatrixExp[-I*α*n.{sx[S],sy[S],sz[S]}] //FullSimplify]
Out[788]= {{(Cos[α/2]-I*Cos[θ]*Sin[α/2])^2,
            E^(-I*φ)*((-1+Cos[α])*Cos[θ]-I*Sin[α])*Sin[θ]/Sqrt[2],
            -E^(-2I*φ)*Sin[α/2]^2*Sin[θ]^2},
           {Sqrt[2]*E^(-I*α)*Sin[α/2]*(Cos[α/2]-I*Cos[θ]*Sin[α/2])
              *Sin[θ]*(-I*Cos[α+φ]+Sin[α+φ]),
            Cos[α/2]^2+Cos[2θ]*Sin[α/2]^2,
            E^(-I*φ)*(Cos[θ]-Cos[α]*Cos[θ]-I*Sin[α])*Sin[θ]/Sqrt[2]},
           {-E^(2I*φ)*Sin[α/2]^2*Sin[θ]^2,
            -E^(I*φ)*((-1+Cos[α])*Cos[θ]+I*Sin[α])*Sin[θ]/Sqrt[2],
            (Cos[α/2]+I*Cos[θ]*Sin[α/2])^2}}
In[789]:= % /. α -> 0
Out[789]= {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
```

Q3.4 (page 43) In the unit system of `In[275]` we have

```
In[790]:= {eval, evec} = Eigensystem[H[Quantity[1,"Teslas"]/MagneticFieldUnit, 0, 0]]
Out[790]= {{-14012.476, 14012.476}, {{-0.7071068, 0.7071068}, {0.7071068, 0.7071068}}}
```

To convert the energy eigenvalues to Joules (or Yoctojoules), we use

```
In[791]:= UnitConvert[eval*EnergyUnit, "Yoctojoules"]
Out[791]= {-9.284765 Yoctojoules, 9.284765 Yoctojoules}
```

The corresponding eigenvectors are in the $\pm x$ direction:

- ground state: $E_- = -9.28 \times 10^{-24}\,\mathrm{J} = -9.28\,\mathrm{yJ}$; $|\psi_-\rangle = |-x\rangle = \frac{|\uparrow\rangle - |\downarrow\rangle}{\sqrt{2}}$
- excited state: $E_+ = +9.28 \times 10^{-24}\,\mathrm{J} = +9.28\,\mathrm{yJ}$; $|\psi_+\rangle = |+x\rangle = \frac{|\uparrow\rangle + |\downarrow\rangle}{\sqrt{2}}$

Q3.5 (page 43) See also **Q2.1** and **Q2.3**.

```
In[792]:= Bvec = B*{Sin[θ]*Cos[φ], Sin[θ]*Sin[φ], Cos[θ]};
In[793]:= Svec = {sx[1/2], sy[1/2], sz[1/2]};
In[794]:= H = -μB*ge*Bvec.Svec //FullSimplify;
In[795]:= {eval, evec} = Eigensystem[H];
In[796]:= eval
Out[796]= {-B*ge*μB/2, B*ge*μB/2}
```

```
In[797]:= Assuming[0<θ<π, ComplexExpand[Normalize /@ evec] //FullSimplify]
Out[797]= {{E^(-I*φ)*Cos[θ/2], Sin[θ/2]}, {-E^(-I*φ)*Sin[θ/2], Cos[θ/2]}}
```

Q3.6 (page 52) We define all operators in the combined Hilbert space of both spins, so that the operators $\hat{\boldsymbol{F}}$ can be defined by addition:

```
In[798]:= With[{i=3, j=5},
          Ix = KroneckerProduct[sx[i], id[j]];
          Iy = KroneckerProduct[sy[i], id[j]];
          Iz = KroneckerProduct[sz[i], id[j]];
          Jx = KroneckerProduct[id[i], sx[j]];
          Jy = KroneckerProduct[id[i], sy[j]];
          Jz = KroneckerProduct[id[i], sz[j]];
          Fx=Ix+Jx; Fy=Iy+Jy; Fz=Iz+Jz;]
```

We calculate the eigenvalues in ascending order with `Sort`. Remember that $|I-J| \le F \le I+J$, and therefore $F \in \{2,3,4,5,6,7,8\}$ and $\langle \hat{F}^2 \rangle = F(F+1) \in \{6,12,20,30,42,56,72\}$.

```
In[799]:= Ix.Ix + Iy.Iy + Iz.Iz //Eigenvalues //Sort
Out[799]= {12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,
       12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,
       12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12}
In[800]:= Iz //Eigenvalues //Sort
Out[800]= {-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-1,-1,-1,
       -1,-1,-1,-1,-1,-1,-1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,
       2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3}
In[801]:= Jx.Jx + Jy.Jy + Jz.Jz //Eigenvalues //Sort
Out[801]= {30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,
       30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,
       30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30}
In[802]:= Jz //Eigenvalues //Sort
Out[802]= {-5,-5,-5,-5,-5,-5,-5,-4,-4,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,
       -2,-2,-2,-1,-1,-1,-1,-1,-1,-1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,
       3,3,3,3,4,4,4,4,4,4,4,5,5,5,5,5,5,5}
In[803]:= Fx.Fx + Fy.Fy + Fz.Fz //Eigenvalues //Sort
Out[803]= {6,6,6,6,6,12,12,12,12,12,12,12,20,20,20,20,20,20,20,20,20,30,30,30,30,30,30,
       30,30,30,30,30,42,42,42,42,42,42,42,42,42,42,42,42,42,56,56,56,56,56,56,56,56,
       56,56,56,56,56,56,56,72,72,72,72,72,72,72,72,72,72,72,72,72,72,72,72,72,72,72}
In[804]:= Fz //Eigenvalues //Sort
Out[804]= {-8,-7,-7,-6,-6,-6,-5,-5,-5,-5,-4,-4,-4,-4,-4,-3,-3,-3,-3,-3,-3,-2,-2,-2,-2,
       -2,-2,-2,-1,-1,-1,-1,-1,-1,-1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,2,2,2,2,2,2,2,3,3,3,
       3,3,3,4,4,4,4,4,5,5,5,5,6,6,6,7,7,8}
```

Q3.7 (page 52) The Clebsch–Gordan coefficients $\langle I, M_I, J, M_J | I, J, F, M_F \rangle$ serve to construct the states that simultaneously diagonalize $\hat{I}^2$, $\hat{J}^2$, $\hat{F}^2$, and $\hat{F}_z$ from those that simultaneously diagonalize $\hat{I}^2$, $\hat{I}_z$, $\hat{J}^2$, and $\hat{J}_z$: for $|I-J| \le F \le I+J$ and $|M_F| \le F$,

$$|I, J, F, M_F\rangle = \sum_{M_I=-I}^{I} \sum_{M_J=-J}^{J} \langle I, M_I, J, M_J | I, J, F, M_F \rangle |I, M_I\rangle \otimes |J, M_J\rangle \tag{9}$$

In Mathematica, `S[i,j,F,MF]` $= |I, J, F, M_F\rangle$:

```
In[805]:= S[i_,j_,F_,MF_] := Sum[ClebschGordan[{i,Mi},{j,Mj},{F,MF}]*
          Flatten[KroneckerProduct[SparseArray[i-Mi+1->1, 2i+1],
                                   SparseArray[j-Mj+1->1, 2j+1]]],
          {Mi,-i,i}, {Mj,-j,j}]
```

Check that these diagonalize $\hat{I}^2$, $\hat{J}^2$, $\hat{F}^2$, and $\hat{F}_z$ simultaneously:

```
In[806]:= With[{i=3, j=5},
          Table[(Ix.Ix+Iy.Iy+Iz.Iz).S[i,j,F,MF] == i(i+1)*S[i,j,F,MF] &&
                (Jx.Jx+Jy.Jy+Jz.Jz).S[i,j,F,MF] == j(j+1)*S[i,j,F,MF] &&
                (Fx.Fx+Fy.Fy+Fz.Fz).S[i,j,F,MF] == F(F+1)*S[i,j,F,MF] &&
                                    Fz.S[i,j,F,MF] == MF*S[i,j,F,MF],
            {F,Abs[i-j],i+j}, {MF,-F,F}]
```

(disregard the warnings about `ClebschGordan::phy`).

When we use the basis of product states $|I, M_I\rangle \otimes |J, M_J\rangle$ to calculate the eigenvectors of the matrices `Fx.Fx+Fy.Fy+Fz.Fz` and `Fz`, using either `Eigenvectors` or `Eigensystem`, these Clebsch–Gordan coefficients appear naturally as coefficients of the resulting eigenvectors.

Q3.8 (page 52)

```
In[807]:= With[{S = 100},
          ψ = SparseArray[1->1, 2S+1];
          {x,y,z, xx,yy,zz} = Conjugate[ψ].(#.ψ)& /@
            {sx[S],sy[S],sz[S], sx[S].sx[S],sy[S].sy[S],sz[S].sz[S]};
          {{x,y,z}, {xx-x^2,yy-y^2,zz-z^2}}]
Out[807]= {{0, 0, 100}, {50, 50, 0}}
```

In general,

$$\begin{array}{lll} \langle \hat{S}_x \rangle = 0 & \langle \hat{S}_y \rangle = 0 & \langle \hat{S}_z \rangle = S \\ \langle \hat{S}_x^2 \rangle = S/2 & \langle \hat{S}_y^2 \rangle = S/2 & \langle \hat{S}_z^2 \rangle = S^2 \\ \langle \hat{S}_x^2 \rangle - \langle \hat{S}_x \rangle^2 = S/2 & \langle \hat{S}_y^2 \rangle - \langle \hat{S}_y \rangle^2 = S/2 & \langle \hat{S}_z^2 \rangle - \langle \hat{S}_z \rangle^2 = 0 \end{array} \tag{10}$$

Q3.9 (page 52)

```
In[808]:= {Δ, Ω} /.
          {Ei -> eval[[2]], Ej -> eval[[7]], Tij -> T[[2,7]], Tji -> T[[7,2]]} /.
          hfc /. {Bz->3.22895,Bacx->0.1,Bacy->0,Bacz->0,ω->2π*6827.9}
Out[808]= {0.00476766, 0.762616}
```

The oscillation period is $2\pi/\Omega = 8.238\,99\,\mu s$, which matches the full oscillation periods of the plots of `In[321]` and `In[322]`.

Q3.10 (page 52) ^{23}Na has the same nuclear spin $I = 3/2$ as ^{87}Rb; they differ only in the constants:

- $A_{hfs} = 885.813\,064\,40\,\text{MHz}$
- $g_I = 0.000\,804\,610\,80$
- $g_L = -0.999\,976\,13$

As a result, there is a magic field between the same states as for ^{87}Rb, but at a field strength $B_z = 0.676\,851\,\text{G} \approx \frac{16 A g_I}{3\mu_B g_S^2}$.

Q3.11 (page 52) ^{85}Rb has a nuclear spin $I = 5/2$, which means that we must re-define the spin operators; all operators are now 12×12 matrices. Further, the constants to be used are

- $A_{hfs} = 1.011\,910\,813\,0\,\text{GHz}$
- $g_I = 0.000\,293\,640\,00$
- $g_L = -0.999\,993\,54$

There are two magic fields:

- $B_z = 0.357\,312\,\text{G} \approx \frac{27 A g_I}{4\mu_B g_S^2}$: the energy difference between $|F = 2, M_F = -1\rangle$ and $|F = 3, M_F = 1\rangle$ is stationary.
- $B_z = 1.143\,42\,\text{G} \approx \frac{108 A g_I}{5\mu_B g_S^2}$: the energy difference between $|F = 2, M_F = -2\rangle$ and $|F = 3, M_F = 2\rangle$ is stationary.

Q3.12 (page 52) ^{133}Cs has a nuclear spin $I = 7/2$, which means that we must re-define the spin operators; all operators are now 16×16 matrices. Further, the constants to be used are

- $A_{\text{hfs}} = 2.298\,157\,942\,5\,\text{GHz}$
- $g_I = 0.000\,398\,853\,95$
- $g_L = -0.999\,995\,87$

There are three magic fields:

- $B_z = 1.393\,34\,\text{G} \approx \frac{128 A g_I}{15 \mu_B g_S^2}$: the energy difference between $|F = 3, M_F = -1\rangle$ and $|F = 4, M_F = 1\rangle$ is stationary.
- $B_z = 3.483\,38\,\text{G} \approx \frac{64 A g_I}{3 \mu_B g_S^2}$: the energy difference between $|F = 3, M_F = -2\rangle$ and $|F = 4, M_F = 2\rangle$ is stationary.
- $B_z = 8.9572\,\text{G} \approx \frac{384 A g_I}{7 \mu_B g_S^2}$: the energy difference between $|F = 3, M_F = -3\rangle$ and $|F = 4, M_F = 3\rangle$ is stationary.

Q3.13 (page 52) In the result from

```
In[809]:= Assuming[A>0, FullSimplify[T/.{Bx->0,By->0,Bz->0,Bacx->B,Bacy->I*B,Bacz->0}]]
```

we can see that the transitions $1 \leftrightarrow 5$, $1 \leftrightarrow 6$, $3 \leftrightarrow 7$, $3 \leftrightarrow 8$, $4 \leftrightarrow 7$, $4 \leftrightarrow 8$ are allowed. Using Equation (3.9) we identify these transitions as $|2,2\rangle \leftrightarrow |1,1\rangle$, $|2,2\rangle \leftrightarrow |2,1\rangle$, $|1,0\rangle \leftrightarrow |1,-1\rangle$, $|1,0\rangle \leftrightarrow |2,-1\rangle$, $|2,0\rangle \leftrightarrow |1,-1\rangle$, $|2,0\rangle \leftrightarrow |2,-1\rangle$. These transitions are all $\Delta M_F = \pm 1$.

Q3.14 (page 52) In the result from

```
In[810]:= Assuming[A>0, FullSimplify[T/.{Bx->0,By->0,Bz->0,Bacx->0,Bacy->0,Bacz->B}]]
```

we can see that the transitions $3 \leftrightarrow 4$, $5 \leftrightarrow 6$, $7 \leftrightarrow 8$ are allowed. Using Equation (3.9) we identify these transitions as $|1,0\rangle \leftrightarrow |2,0\rangle$, $|1,1\rangle \leftrightarrow |2,1\rangle$, $|1,-1\rangle \leftrightarrow |2,-1\rangle$. These transitions are all $\Delta M_F = 0$. Further, the energy of levels 1, 2, 5, 6, 7, 8 (*i.e.*, all levels with $M_F \neq 0$) will be shifted by the non-zero diagonal elements of `T`.

Q3.15 (page 61) On a MacBook Pro (Retina, 13-inch, Early 2015) with a 3.1 GHz Intel Core i7 CPU and 16 GB 1867 MHz DDR3 RAM, it takes around 20 minutes (`AbsoluteTiming[γ=gs[1,1];]`) to calculate the ground state `gs[1,1]` with the definition of `In[350]` and $N = 22$. This calculation uses over 24 GB of compressed RAM (`MaxMemoryUsed[]`) and is the upper limit on N for this computer. For $N = 23$ the calculation runs out of memory.

Q3.16 (page 61) With the Mathematica code of section 3.4, setting $S = 1$ and $N = 12$, we find a phase transition around $b = \pm 2$. Similar to the $S = 1/2$ case, the Ising model is gapless for $|b| < 2$ and gapped for $|b| > 2$. The correlations look qualitatively similar to the ones found for $S = 1/2$.

Q3.17 (page 61)

1. For $b \to \pm\infty$ the ground states are analogous to those of the transverse Ising model, Equation (3.28), along the $\pm z$ axis:

$$|\psi_{+\infty}\rangle = |+z\rangle^{\otimes N}, \qquad |\psi_{-\infty}\rangle = |-z\rangle^{\otimes N}. \tag{11}$$

   Notice that, unlike the transverse Ising model, these asymptotic ground states are the exact ground states for $|b| > 2$, not just in the limits $b \to \pm\infty$.
2. There are phase transitions at $b = \pm 2$, recognizable in the ground-state gap.
3. Since the states of Equation (11) are product states, there are absolutely no correlations between the states of the spins for $|b| > 2$. For $|b| < 2$ the magnetization, spin–spin correlations, and entanglement entropy are qualitatively similar to those of the transverse Ising model. For $b = 0$ the spin–spin correlations do not reach the full uniform 0.25 as for the Ising model, but rather they still decay with distance.

Q3.18 (page 61)

1. For $b \to \pm\infty$ the ground states are the same as Equation (11).
2. At $b = 0$ the ground-state degeneracy is $N + 1$.

3. For any $b > 0$, $|\psi_{+\infty}\rangle$ is the exact ground state; for any $b < 0$, $|\psi_{-\infty}\rangle$ is the exact ground state. There is a phase transition at $b = 0$.
4. Since the states of Equation (11) are product states, there are absolutely no correlations between the states of the spins for any $b \neq 0$.

Q3.19 (page 62)

1. $\hat{\rho}_{AB} = |\psi\rangle\langle\psi| = |\uparrow\uparrow\rangle\langle\uparrow\uparrow|$:

```
In[811]:= ψ = Flatten[KroneckerProduct[{1,0}, {1,0}]]
Out[811]= {1, 0, 0, 0}
In[812]:= ρAB = KroneckerProduct[ψ, Conjugate[ψ]]
Out[812]= {{1,0,0,0}, {0,0,0,0}, {0,0,0,0}, {0,0,0,0}}
```

2. $\hat{\rho}_A = \mathrm{Tr}_B\, \hat{\rho}_{AB} = |\uparrow\rangle\langle\uparrow|$ is a pure state:

```
In[813]:= ρA = traceout[ρAB, -2]
Out[813]= {{1,0}, {0,0}}
In[814]:= Tr[ρA.ρA]
Out[814]= 1
```

3. $\hat{\rho}_B = \mathrm{Tr}_A\, \hat{\rho}_{AB} = |\uparrow\rangle\langle\uparrow|$ is a pure state:

```
In[815]:= ρB = traceout[ρAB, 2]
Out[815]= {{1,0}, {0,0}}
In[816]:= Tr[ρB.ρB]
Out[816]= 1
```

4. Using `In[353]` and `In[354]`, we see that the entropy of entanglement is $S_A - S_{AB} = S_B - S_{AB} = 0$ (no entanglement):

```
In[817]:= SAB = Total[s /@ Eigenvalues[ρAB]]
Out[817]= 0
In[818]:= SA = Total[s /@ Eigenvalues[ρA]]
Out[818]= 0
In[819]:= SB = Total[s /@ Eigenvalues[ρB]]
Out[819]= 0
```

Q3.20 (page 62)

1. $\hat{\rho}_{AB} = |\psi\rangle\langle\psi| = \frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{\sqrt{2}} \frac{\langle\uparrow\downarrow| - \langle\downarrow\uparrow|}{\sqrt{2}} = \frac{1}{2}(|\uparrow\downarrow\rangle\langle\uparrow\downarrow| - |\uparrow\downarrow\rangle\langle\downarrow\uparrow| - |\downarrow\uparrow\rangle\langle\uparrow\downarrow| + |\downarrow\uparrow\rangle\langle\downarrow\uparrow|)$:

```
In[820]:= ψ = Flatten[KroneckerProduct[{1,0}, {0,1}]
          - KroneckerProduct[{0,1}, {1,0}]]/Sqrt[2]
Out[820]= {0, 1/Sqrt[2], -1/Sqrt[2], 0}
In[821]:= ρAB = KroneckerProduct[ψ, Conjugate[ψ]]
Out[821]= {{0,0,0,0}, {0,1/2,-1/2,0}, {0,-1/2,1/2,0}, {0,0,0,0}}
```

2. $\hat{\rho}_A = \mathrm{Tr}_B\, \hat{\rho}_{AB} = \frac{1}{2}(|\uparrow\rangle\langle\uparrow| + |\downarrow\rangle\langle\downarrow|)$ is a mixed state:

```
In[822]:= ρA = traceout[ρAB, -2]
Out[822]= {{1/2,0}, {0,1/2}}
In[823]:= Tr[ρA.ρA]
Out[823]= 1/2
```

3. $\hat{\rho}_B = \mathrm{Tr}_A\, \hat{\rho}_{AB} = \frac{1}{2}(|\uparrow\rangle\langle\uparrow| + |\downarrow\rangle\langle\downarrow|)$ is a mixed state:

```
In[824]:= ρB = traceout[ρAB, 2]
Out[824]= {{1/2,0}, {0,1/2}}
In[825]:= Tr[ρB.ρB]
Out[825]= 1/2
```

4. Using In[353] and In[354], we see that the entropy of entanglement is $S_A - S_{AB} = S_B - S_{AB} = 1$ (maximal entanglement):

```
In[826]:= SAB = Total[s /@ Eigenvalues[ρAB]]
Out[826]= 0
In[827]:= SA = Total[s /@ Eigenvalues[ρA]]
Out[827]= 1
In[828]:= SB = Total[s /@ Eigenvalues[ρB]]
Out[828]= 1
```

Q3.21 (page 70) Don't forget to complex-conjugate ψout on the left, for generality:

```
In[829]:= ψout = {1/Sqrt[2], 0, 0, 1/Sqrt[2]};
In[830]:= Table[Conjugate[ψout].(KroneckerProduct[PauliMatrix[i],PauliMatrix[j]].ψout),
            {i,0,3}, {j,0,3}]
Out[830]= {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, 1}}
```

We find $\langle\psi_{\text{out}}|\mathbb{1}\otimes\mathbb{1}|\psi_{\text{out}}\rangle = 1$ (normalization), $\langle\psi_{\text{out}}|\hat\sigma_x\otimes\hat\sigma_x|\psi_{\text{out}}\rangle = 1$, $\langle\psi_{\text{out}}|\hat\sigma_y\otimes\hat\sigma_y|\psi_{\text{out}}\rangle = -1$, $\langle\psi_{\text{out}}|\hat\sigma_z\otimes\hat\sigma_z|\psi_{\text{out}}\rangle = 1$, and all others equal to zero. The density matrix is therefore

$$\begin{aligned}\hat\rho &= \frac{1}{4}\left(\mathbb{1}\otimes\mathbb{1} + \hat\sigma_x\otimes\hat\sigma_x - \hat\sigma_y\otimes\hat\sigma_y + \hat\sigma_z\otimes\hat\sigma_z\right)\\ &= \frac{1}{2}\left(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|\right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}}\,\frac{\langle 00| + \langle 11|}{\sqrt{2}}\end{aligned} \tag{12}$$

as expected.

Q3.22 (page 70) Replace In[390] and In[391] with

```
In[831]:= u = {1,1}/Sqrt[2];
In[832]:= U[φ_] = Exp[2π*I*φ] * {{1,0},{0,1}};
```

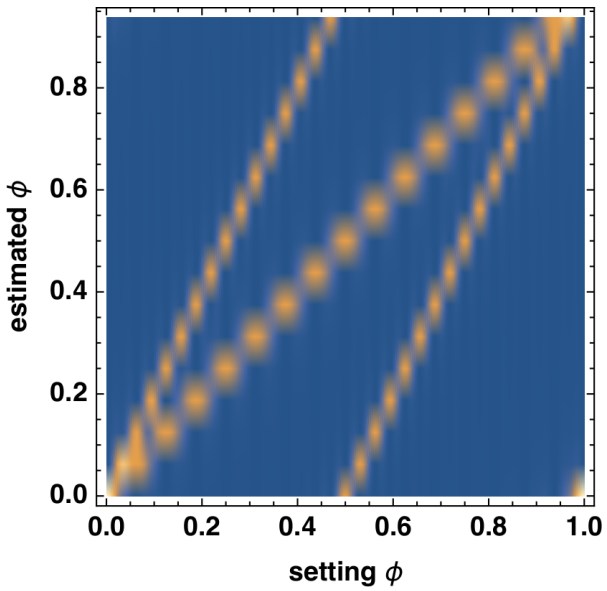
and re-evaluate the attached Mathematica notebook. All results remain unchanged.

Q3.23 (page 70) Replace In[390] and In[391] with

```
In[833]:= u = {1,1}/Sqrt[2];
In[834]:= U[φ_] = {{Exp[2π*I*φ], 0}, {0, Exp[4π*I*φ]}};
```

and re-evaluate the attached Mathematica notebook. The probabilities for the different estimates of φ show both frequencies simultaneously, and there is no cross-talk between them:

```
In[835]:= ListDensityPlot[Transpose[Table[prob[φ], {φ,0,1,1/256}]],
            PlotRange->All, DataRange->{{0,1},{0,1-2^-t}},
            FrameLabel->{"setting φ","estimated φ"}]
```



Chapter 4 quantum motion in real space

Q4.1 (page 72) Starting with the Schrödinger equation

$$\left[-\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x|x\rangle\frac{\mathrm{d}^2}{\mathrm{d}x^2}\langle x|+\int_{-\infty}^{\infty}\mathrm{d}x|x\rangle V(x)\langle x|\right]|\psi\rangle = E|\psi\rangle, \tag{13}$$

we (i) leave away the bracket and (ii) multiply by $\langle y|$ from the left ($y\in\mathbb{R}$):

$$-\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x\langle y|x\rangle\frac{\mathrm{d}^2}{\mathrm{d}x^2}\langle x|\psi\rangle+\int_{-\infty}^{\infty}\mathrm{d}x\langle y|x\rangle V(x)\langle x|\psi\rangle = E\langle y|\psi\rangle \tag{14}$$

Remembering that $\langle y|x\rangle=\delta(x-y)$ and $\langle x|\psi\rangle=\psi(x)$:

$$-\frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x\delta(x-y)\psi''(x)+\int_{-\infty}^{\infty}\mathrm{d}x\delta(x-y)V(x)\psi(x) = E\psi(y) \tag{15}$$

Simplify the integrals with the Dirac δ-functions:

$$-\frac{\hbar^2}{2m}\psi''(y)+V(y)\psi(y) = E\psi(y) \tag{16}$$

Since this is valid for any $y\in\mathbb{R}$, it concludes the proof.

Q4.2 (page 72)

$$\begin{aligned}\langle\psi|\chi\rangle &= \left[\int_{-\infty}^{\infty}\mathrm{d}x\psi^*(x)\langle x|\right]\left[\int_{-\infty}^{\infty}\mathrm{d}y\chi(y)|y\rangle\right]\\ &= \int_{-\infty}^{\infty}\mathrm{d}x\mathrm{d}y\psi^*(x)\chi(y)\langle x|y\rangle\\ &= \int_{-\infty}^{\infty}\mathrm{d}x\mathrm{d}y\psi^*(x)\chi(y)\delta(x-y)\\ &= \int_{-\infty}^{\infty}\mathrm{d}x\psi^*(x)\chi(x)\end{aligned} \tag{17}$$

Q4.3 (page 79)

```
In[836]:= a = m = ħ = 1;    (* natural units *)
In[837]:= nmax = 100;
In[838]:= TM = SparseArray[Band[{1,1}] -> Range[nmax]^2*π^2*ħ^2/(2*m*a^2)];
```

```
In[839]:= pM = SparseArray[{n1_,n2_}/;OddQ[n1-n2]->(4*I*ℏ*n1*n2)/(a*(n2^2-n1^2)),
                           {nmax,nmax}];
In[840]:= TM //N //Eigenvalues //Sort
Out[840]= {4.9348, 19.7392, 44.4132, 78.9568, 123.37, 177.653, ..., 48366., 49348.}
In[841]:= pM.pM/(2m) //N //Eigenvalues //Sort
Out[841]= {4.8183, 4.8183, 43.3646, 43.3646, 120.457, 120.457, ..., 47257.4, 47257.4}
```

The eigenvalues of $\hat{T}$ are quadratically spaced, whereas those of $\hat{p}^2/(2m)$ come in degenerate pairs (one involving only states of even n and one only states of odd n) and thus never converge to the eigenvalues of $\hat{T}$, even in the limit $n_{\max} \to \infty$.

Q4.4 (page 79) We use the more accurate form of the position operator from In[434]:

```
In[842]:= a = m = ℏ = 1;   (* natural units *)
In[843]:= nmax = 20;
In[844]:= xM = SparseArray[{
              Band[{1,1}] -> a/2,
              {n1_,n2_} /; OddQ[n1-n2] -> -8*a*n1*n2/(π^2*(n1^2-n2^2)^2)},
            {nmax,nmax}];
In[845]:= pM = SparseArray[{n1_,n2_}/;OddQ[n1-n2]->4*I*ℏ*n1*n2/(a*(n2^2-n1^2)),
            {nmax,nmax}];
In[846]:= coM = xM.pM - pM.xM;   (* commutator [x,p] in the momentum basis *)
In[847]:= coM/ℏ //N //MatrixForm
```

In the upper-left corner (low values of n) the result looks like the unit matrix multiplied by the imaginary unit i; but towards the lower-right corner (large values of n) it deviates dramatically from the correct expression. This is to be expected from the problematic nature of the momentum operator; see section 4.1.6.

Q4.5 (page 83) The exact probability is about 37.1%:

```
In[848]:= Integrate[ψ[1,x]^2, {x, 0, 1}] //N
Out[848]= 0.37087
```

Using In[460], the first numerical method gives a good approximation of 37.0%:

```
In[849]:= Integrate[ψ0[x]^2, {x, 0, 1}]
Out[849]= 0.369801
```

Using In[477], the second numerical method gives an approximation of 36.2%:

```
In[850]:= Integrate[ψ0[x]^2, {x, 0, 1}]
Out[850]= 0.362126
```

Alternatively, we set up an interpolating function from the data of In[474], and integrate it numerically. The result depends on the interpolation order: higher-order interpolations tend to yield more accurate results.

```
In[851]:= ψ0i1 = Interpolation[γ, InterpolationOrder -> 1];
In[852]:= NIntegrate[ψ0i1[x]^2, {x, 0, 1}]
Out[852]= 0.302899
In[853]:= ψ0i2 = Interpolation[γ, InterpolationOrder -> 2];
In[854]:= NIntegrate[ψ0i2[x]^2, {x, 0, 1}]
Out[854]= 0.358003
In[855]:= ψ0i3 = Interpolation[γ, InterpolationOrder -> 3];
In[856]:= NIntegrate[ψ0i3[x]^2, {x, 0, 1}]
Out[856]= 0.3812
```

Q4.6 (page 83) From Equation (4.31) the average height in state k is

$$\langle k|\hat{x}|k\rangle = \int_0^\infty \mathrm{d}x |\psi_k(x)|^2 x = -\alpha_k \cdot \left(\frac{4\hbar^2}{27m^2g}\right)^{1/3}, \tag{18}$$

which you can verify with In[448] and

```
In[857]:= Assuming[m>0&&g>0&&ℏ>0, Table[Integrate[ψ[k,x]^2*x, {x, 0, ∞}], {k, 1, 5}]]
```

For a neutron in earth's gravitational field this gives an average height of about 9 µm:

```
In[858]:= With[{k = 1,
                m = Quantity["NeutronMass"],
                g = Quantity["StandardAccelerationOfGravity"],
                ℏ = Quantity["ReducedPlanckConstant"]},
          UnitConvert[-AiryAiZero[k]*(4*ℏ^2/(27*m^2*g))^(1/3), "Micrometers"]]
Out[858]= 9.147654 µm
```

Q4.7 (page 83) The exact energy levels are $E_n = \hbar\omega(n + 1/2)$ with $n \in \mathbb{N}_0$.

In the given unit system, the mass is `m=1`, Planck's constant is $\hbar$=1, and the angular frequency is ω=1/$\hbar$=1.

We set up a calculation in the position basis with the mixed-basis numerical method:

```
In[859]:= a = 10;                    (* calculation box size *)
In[860]:= m = ℏ = ω = 1;             (* natural units *)
In[861]:= nmax = 100;
In[862]:= Δ = a/(nmax+1);            (* grid spacing *)
In[863]:= xgrid = Range[nmax]*Δ; (* the computational grid *)
In[864]:= TP = FourierDST[SparseArray[Band[{1,1}]->Range[nmax]^2*π^2*ℏ^2/(2*m*a^2)], 1];
In[865]:= W[x_] = m*ω^2*(x-a/2)^2/2;    (* the potential function, centered *)
In[866]:= Wgrid = Map[W, xgrid]; (* the potential on the computational grid *)
In[867]:= VP = SparseArray[Band[{1,1}] -> Wgrid];
In[868]:= HP = TP + VP;
```

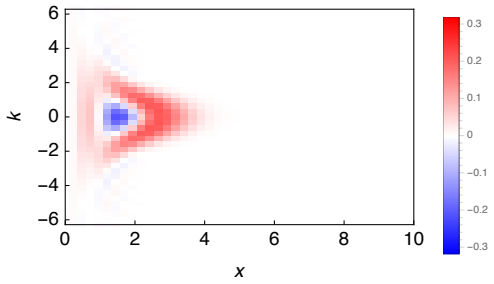
We find the energy eigenvalues (in units of $\mathcal{E} = \hbar\omega$) with

```
In[869]:= Eigenvalues[HP] //Sort
Out[869]= {0.5, 1.5, 2.5, 3.5, 4.50001, 5.5001, 6.5006, 7.50293, 8.51147, 9.53657, ...}
```

and see that at least the lowest eigenvalues match the analytic expression. Using a larger value of n_{max} will give more accurate eigenstates and eigenvalues.

Q4.8 (page 85) The excited-state Wigner distribution has a significant negative region around its center:

```
In[870]:= gsP = Transpose[Sort[Transpose[-Eigensystem[-N[HP], 2,
          Method->{"Arnoldi", "Criteria"->"RealPart", MaxIterations->10^6}]]]];
In[871]:= WignerDistributionPlot[gsP[[2, 2]], {0, a}]
```



Q4.9 (page 91)

```
In[872]:= X = RandomReal[{0,1}, {2,2}]
Out[872]= {{0.580888, 0.80848}, {0.218175, 0.979598}}
In[873]:= Y = RandomReal[{0,1}, {2,2}]
Out[873]= {{0.448364, 0.774595}, {0.490198, 0.310169}}
In[874]:= X.Y - Y.X
Out[874]= {{0.227318, -0.420567}, {0.225597, -0.227318}}
```

```
In[875]:= MatrixExp[X + Y]
Out[875]= {{4.68326, 6.06108}, {2.71213, 5.68068}}
In[876]:= MatrixExp[X].MatrixExp[Y]
Out[876]= {{5.0593, 5.38209}, {3.10936, 5.31705}}
```

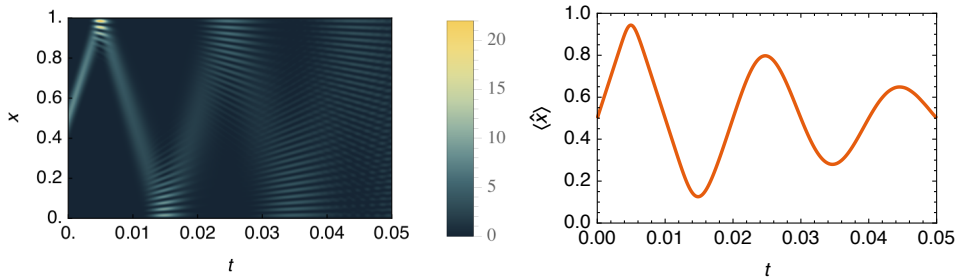
Q4.10 (page 91) We use the split-step propagation code of section 4.1.9 with the potential

```
In[877]:= a = m = ħ = 1;
In[878]:= With[{W0 = 0 * ħ^2/(m*a^2)},
            W[x_] = W0*Sin[10*π*x/a];]
```

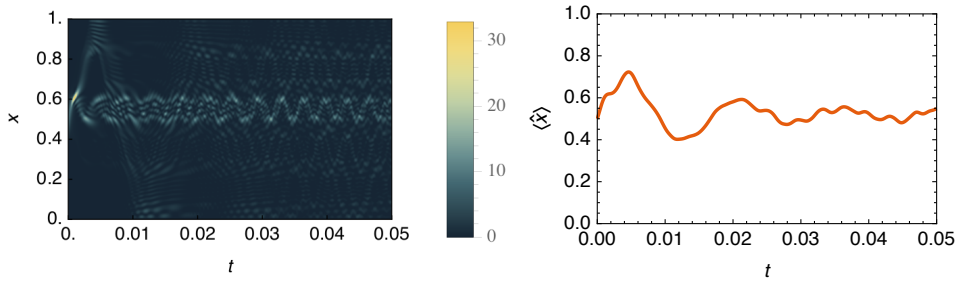
and the initial wavefunction

```
In[879]:= With[{x0=a/2, σ=0.05*a, k=100/a},
            v0=Normalize[Function[x, E^(-((x-x0)^2/(4*σ^2)))*E^(I*k*x)] /@ xgrid];]
```

For $W_0 = 0$ the Gaussian wavepacket bounces back and forth between the simulation boundaries and disperses slowly; the self-interference at the reflection points is clearly visible:



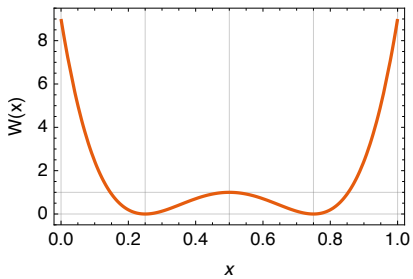
For $W_0 = 5000\frac{\hbar^2}{ma^2}$ the Gaussian wavepacket remains mostly trapped:



Q4.11 (page 92)

1. $W(x)$ is a double-well potential with minima at $x = \frac{1}{2} \pm \delta$ and a barrier height of Ω:

```
In[880]:= W[{Ω_, δ_}, x_] = Ω*(((x-1/2)/δ)^2-1)^2;
In[881]:= Table[W[{Ω,δ},x], {x,1/2-δ,1/2+δ,δ}]
Out[881]= {0, Ω, 0}
In[882]:= Table[D[W[{Ω,δ},y],y] /. y->x, {x,1/2-δ,1/2+δ,δ}]
Out[882]= {0, 0, 0}
In[883]:= Plot[W[{1, 1/4}, x], {x, 0, 1}]
```

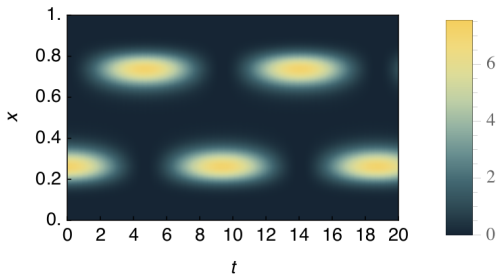


2. As in **Q4.10**, we use the split-step propagation code of section 4.1.9 with the potential

```
In[884]:= With[{Ω = 250, δ = 1/4},
           W[x_] = W[{Ω, δ}, x];]
```

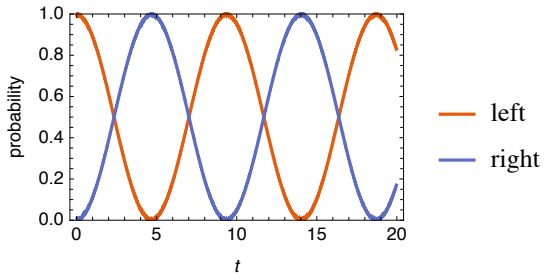
With the initial state from In[879] (**Q4.10**) with x0=0.2694, σ=0.0554, k=0, the time-dependent density is seen to oscillate between the wells:

```
In[885]:= With[{Δt = 20, M = 10^4},
           V = propApprox[Δt, M, v0];]
In[886]:= ρ = ArrayPad[(nmax+1)*Abs[#[[2]]]^2& /@ V, {{0,0},{1,1}}];
In[887]:= ArrayPlot[Reverse[Transpose[ρ]]]
```



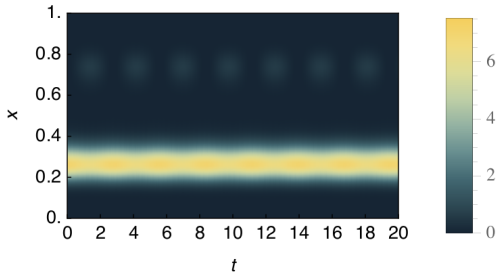
This oscillation is apparent in the left/right probabilities:

```
In[888]:= ListLinePlot[{{#[[1]],Norm[#[[2,;;(nmax/2)]]]^2}& /@ V,
                        {#[[1]],Norm[#[[2,nmax/2+1;;]]]^2}& /@ V}]
```

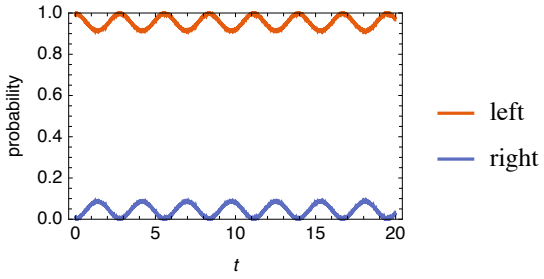


3. Now we use In[500] and observe that the attractive interactions prevent the particle from tunneling between the wells:

```
In[889]:= With[{κ = 0.5, Δt = 20, M = 10^4},
           V = propApprox[W[#1]&, κ, Δt, M, v0];]
In[890]:= ρ = ArrayPad[(nmax+1)*Abs[#[[2]]]^2& /@ V, {{0,0},{1,1}}];
In[891]:= ArrayPlot[Reverse[Transpose[ρ]]]
```




```
In[892]:= ListLinePlot[{{#[[1]],Norm[#[[2,;;(nmax/2)]]]^2}& /@ V,
                        {#[[1]],Norm[#[[2,nmax/2+1;;]]]^2}& /@ V}]
```



Q4.12 (page 95)

1. We do this calculation in Mathematica, for the more general potential $W(x) = \frac{1}{2}k(x - \frac{1}{2})^2$:

```
In[893]:= ζ[x_] = E^(-((x-1/2)/(2*σ))^2)/Sqrt[σ*Sqrt[2*π]];
In[894]:= Assuming[σ>0, Integrate[ζ[x]^2, {x, -∞, ∞}]]
Out[894]= 1
In[895]:= Assuming[σ>0,
            e = Integrate[ζ[x]*(-1/2*ζ''[x] + 1/2*k*(x-1/2)^2*ζ[x]), {x, -∞, ∞}]]
Out[895]= (1+4*k*σ^4)/(8*σ^2)
In[896]:= Solve[D[e, σ] == 0, σ]
Out[896]= {{σ -> -1/(Sqrt[2]*k^(1/4))}, {σ -> -I/(Sqrt[2]*k^(1/4))},
          {σ -> I/(Sqrt[2]*k^(1/4))}, {σ -> 1/(Sqrt[2]*k^(1/4))}}
```

   Of these four solutions, we choose $\sigma = (4k)^{-1/4}$ because it is real and positive:

```
In[897]:= e /. σ -> (4*k)^(-1/4)
Out[897]= Sqrt[k]/2
```

   For $k = 5000$, the ground state is therefore $\zeta(x)$ with $\sigma = 20\,000^{-1/4} \approx 0.084\,089\,6$ and energy $E = \sqrt{5000}/2 \approx 35.3553$.

2. We use the same code as in section 4.2.1 but with the potential

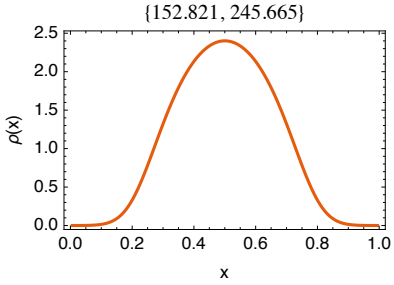
```
In[898]:= With[{k = 5000},
            W[x_] = 1/2*k*(x-1/2)^2;]
```

   Further, we use `nmax=1000` to describe the wavefunction with strongly attractive interactions better. The result matches the Gaussian approximation: both the energy and the chemical potential are approximately $\sqrt{k}/2$,

```
In[899]:= groundstate[10^-4, 0][[;;2]]
Out[899]= {35.3553, 35.3553}
```

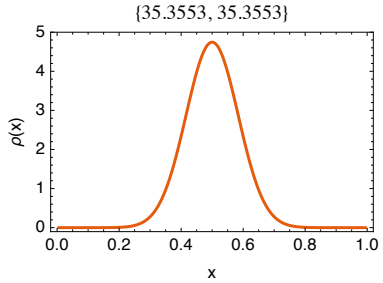
3. Ground-state density for repulsive interactions:

```
In[900]:= With[{κ = 100, δβ = 10^-4},
            {Etot, μ, γ} = groundstate[δβ, κ];
            ListLinePlot[Join[{{0, 0}}, Transpose[{xgrid,Abs[γ]^2/Δ}], {{a, 0}}],
              PlotRange -> All, PlotLabel -> {Etot, μ}]]
```



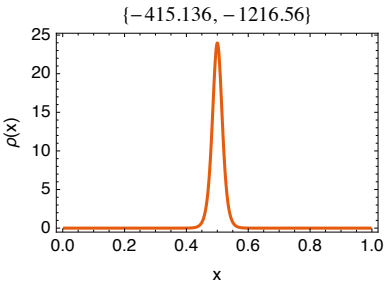
   Ground-state density for no interactions:

```
In[901]:= With[{κ = 0, δβ = 10^-4},
            {Etot, μ, γ} = groundstate[δβ, κ];
            ListLinePlot[Join[{{0, 0}}, Transpose[{xgrid,Abs[γ]^2/Δ}], {{a, 0}}],
              PlotRange -> All, PlotLabel -> {Etot, μ}]]
```



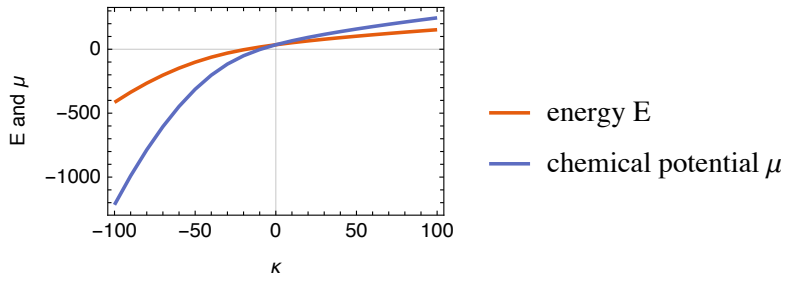
Ground-state density for attractive interactions:

```
In[902]:= With[{κ = -100, δβ = 10^-4},
            {Etot, μ, γ} = groundstate[δβ, κ];
            ListLinePlot[Join[{{0, 0}}, Transpose[{xgrid,Abs[γ]^2/Δ}], {{a, 0}}],
              PlotRange -> All, PlotLabel -> {Etot, μ}]]
```



4. The energy and chemical potential differ for $\kappa \neq 0$:

```
In[903]:= With[{δβ = 10^-4},
            ListLinePlot[Transpose[Table[{{κ,groundstate[δβ,κ][[1]]},
              {κ,groundstate[δβ,κ][[2]]}}, {κ, -100, 100, 10}]]]]
```



Q4.13 (page 100) We do this calculation for $a = 1$; the prefactor a^{-1} of the right-hand side of Equation (4.65) can be found by a variable substitution $x \mapsto ax'$. From the momentum basis functions

```
In[904]:= φ[n_, x_] = Sqrt[2]*Sin[n*π*x];
```

we define the position basis functions

```
In[905]:= θ[nmax_, j_, x_] := 1/Sqrt[nmax+1]*Sum[φ[n,j/(nmax+1)]*φ[n,x], {n,nmax}]
```

The exact overlap integrals are

```
In[906]:= J[nmax_, {j1_,j2_,j3_,j4_}] :=
            Integrate[θ[nmax,j1,x]*θ[nmax,j2,x]*θ[nmax,j3,x]*θ[nmax,j4,x], {x,0,1}]
```

We make a table of overlap integrals, calculated both exactly and approximately through In[504], and show that the difference is zero (up to numerical inaccuracies):

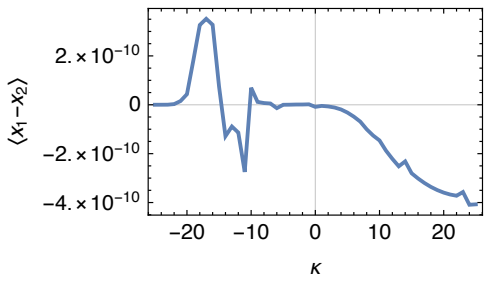
```
In[907]:= With[{nmax = 3},
            A = Table[J[nmax, {j1,j2,j3,j4}], {j1,nmax},{j2,nmax},{j3,nmax},{j4,nmax}];
            B = FourierDST[Table[KroneckerDelta[n1+n2,n3+n4]
              +KroneckerDelta[n1+n3,n2+n4]+KroneckerDelta[n1+n4,n2+n3]
              -KroneckerDelta[n1,n2+n3+n4]-KroneckerDelta[n2,n1+n3+n4]
              -KroneckerDelta[n3,n1+n2+n4]-KroneckerDelta[n4,n1+n2+n3],
              {n1,nmax}, {n2,nmax}, {n3,nmax}, {n4,nmax}],1]/2;]
          A - B //Abs //Max
Out[907]= 8.88178*10^-16
```

Q4.14 (page 100) We define memoizing functions that calculate $\langle \hat{x}_1 - \hat{x}_2 \rangle$ and $\langle (\hat{x}_1 - \hat{x}_2)^2 \rangle$ with

```
In[908]:= Clear[Δ1,Δ2];
In[909]:= Δ1[κ_?NumericQ] := Δ1[κ] =
            With[{γ=gs[0,κ,1][[2,1]]}, Re[Conjugate[γ].((x1-x2).γ)]]
In[910]:= Δ2[κ_?NumericQ] := Δ2[κ] =
            With[{γ=gs[0,κ,1][[2,1]]}, Re[Conjugate[γ].((x1-x2).(x1-x2).γ)]]
```

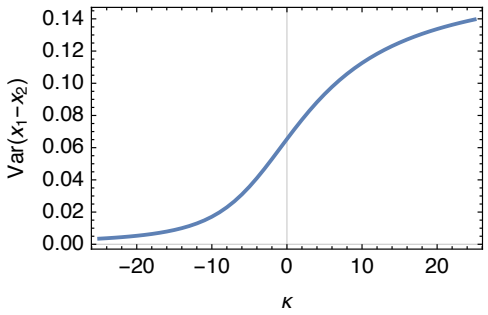
The mean distance in the ground state is zero for symmetry reasons: (notice the numerical inaccuracies)

```
In[911]:= ListLinePlot[Table[{κ, Δ1[κ]}, {κ, -25, 25, 1}]]
```



The variance of the distance in the ground state increases with κ:

```
In[912]:= ListLinePlot[Table[{κ, Δ2[κ]-Δ1[κ]^2}, {κ, -25, 25, 1}]]
```



Q4.15 (page 100) We show that all three terms of the Hamiltonian commute with the particle interchange operator:

```
In[913]:=Ξ.TP - TP.Ξ //Norm
Out[913]=0.
In[914]:=Ξ.VP - VP.Ξ //Norm
Out[914]=0
In[915]:=Ξ.HintP - HintP.Ξ //Norm
Out[915]=1.15903*10^-14
```

Q4.16 (page 100)

```
In[916]:=D[Normal[HPa[Ω, κ]], κ] //Abs //Max
Out[916]=1.03528*10^-15
```

Q4.17 (page 101) We define the wavefunctions $\psi_1(x_1, x_2)$ for $x_1 < x_2$ and $\psi_2(x_1, x_2)$ for $x_1 > x_2$:

```
In[917]:=ψ1[x1_,x2_] = A*(Cos[α*(x1+x2-1)]*Cos[β*(x1-x2+1)]
                        -Cos[α*(x1-x2+1)]*Cos[β*(x1+x2-1)]);
In[918]:=ψ2[x1_,x2_] = ψ1[x2,x1];
```

1. Check the boundary conditions $\psi(x_1, 0) = \psi(x_1, 1) = \psi(0, x_2) = \psi(1, x_2) = 0$:

```
In[919]:={ψ2[x1,0], ψ1[x1,1], ψ1[0,x2], ψ2[1,x2]} //FullSimplify
Out[919]={0, 0, 0, 0}
```

   Check that the two pieces match up for $x_1 = x_2$:

```
In[920]:=ψ1[x,x] == ψ2[x,x]
Out[920]=True
```

   Check the symmetries of the wavefunction:

```
In[921]:=ψ1[x1,x2] == ψ2[x2,x1] == ψ2[1-x1,1-x2] //FullSimplify
Out[921]=True
```

2. Check that the two pieces of the wavefunction satisfy the Schrödinger equation whenever $x_1 \neq x_2$, with energy value $E = \alpha^2 + \beta^2$:

```
In[922]:=-1/2*D[ψ1[x1,x2],{x1,2}]+D[ψ1[x1,x2],{x2,2}] ==
          (α^2+β^2)*ψ1[x1,x2] //FullSimplify
Out[922]=True
In[923]:=-1/2*D[ψ2[x1,x2],{x1,2}]+D[ψ2[x1,x2],{x2,2}] ==
          (α^2+β^2)*ψ2[x1,x2] //FullSimplify
Out[923]=True
```

3. The transformed Hamiltonian is

$$\hat{\mathcal{H}} = \left[-\frac{1}{2}\frac{\partial^2}{\partial R^2}\right] + \left[-\frac{1}{2}\frac{\partial^2}{\partial r^2} + \frac{\kappa}{\sqrt{2}}\delta(r)\right] \tag{19}$$

   and the transformed wavefunctions are

```
In[924]:=ψ1[(R+r)/Sqrt[2],(R-r)/Sqrt[2]] //FullSimplify
Out[924]=A*(Cos[α*(R*Sqrt[2]-1)]*Cos[β*(r*Sqrt[2]+1)]
          -Cos[β*(R*Sqrt[2]-1)]*Cos[α*(r*Sqrt[2]+1)])
In[925]:=ψ2[(R+r)/Sqrt[2],(R-r)/Sqrt[2]] //FullSimplify
Out[925]=A*(Cos[α*(R*Sqrt[2]-1)]*Cos[β*(r*Sqrt[2]-1)]
          -Cos[β*(R*Sqrt[2]-1)]*Cos[α*(r*Sqrt[2]-1)])
```

   with

$$\psi(R, r) = \begin{cases} \psi_1(R, r) & \text{if } r < 0 \\ \psi_2(R, r) & \text{if } r > 0 \end{cases} \tag{20}$$

4. The Schrödinger equation in (R, r) coordinates is

$$\left[-\frac{1}{2}\frac{\partial^2}{\partial R^2}\right]\psi(R,r)+\left[-\frac{1}{2}\frac{\partial^2}{\partial r^2}+\frac{\kappa}{\sqrt{2}}\delta(r)\right]\psi(R,r)=(\alpha^2+\beta^2)\psi(R,r) \tag{21}$$

We integrate Equation (21) over $r \in [-\epsilon, \epsilon]$:

$$-\frac{1}{2}\int_{-\epsilon}^{\epsilon}\mathrm{d}r\frac{\partial^2\psi(R,r)}{\partial R^2}-\frac{1}{2}\int_{-\epsilon}^{\epsilon}\mathrm{d}r\frac{\partial^2\psi(R,r)}{\partial r^2}+\frac{\kappa}{\sqrt{2}}\int_{-\epsilon}^{\epsilon}\mathrm{d}r\delta(r)\psi(R,r)=(\alpha^2+\beta^2)\int_{-\epsilon}^{\epsilon}\mathrm{d}r\psi(R,r) \tag{22}$$

Using partial integration on the second term of the left-hand side:

$$-\frac{1}{2}\int_{-\epsilon}^{\epsilon}\mathrm{d}r\frac{\partial^2\psi(R,r)}{\partial R^2}-\frac{1}{2}\left[\left.\frac{\partial\psi(R,r)}{\partial r}\right|_{r=\epsilon}-\left.\frac{\partial\psi(R,r)}{\partial r}\right|_{r=-\epsilon}\right]+\frac{\kappa}{\sqrt{2}}\psi(R,0)=(\alpha^2+\beta^2)\int_{-\epsilon}^{\epsilon}\mathrm{d}r\psi(R,r) \tag{23}$$

In the limit $\epsilon \to 0^+$ this equation becomes

$$-\frac{1}{2}\left[\left.\frac{\partial\psi_2(R,r)}{\partial r}\right|_{r=0}-\left.\frac{\partial\psi_1(R,r)}{\partial r}\right|_{r=0}\right]+\frac{\kappa}{\sqrt{2}}\psi(R,0)=0 \tag{24}$$

Inserting the definitions of ψ_1 and ψ_2:

```
In[926]:=-1/2*((D[ψ2[(R+r)/Sqrt[2],(R-r)/Sqrt[2]],r]/.r->0)
          - (D[ψ1[(R+r)/Sqrt[2],(R-r)/Sqrt[2]],r]/.r->0))
          + κ/Sqrt[2]*ψ1[R/Sqrt[2],R/Sqrt[2]] //FullSimplify
Out[926]=A*(Cos[β*(R*Sqrt[2]-1)]*(2α*Sin[α]-κ*Cos[α])
          -Cos[α*(R*Sqrt[2]-1)]*(2β*Sin[β]-κ*Cos[β]))/Sqrt[2]
```

The only way that this expression can be zero for all values of $R \in [0, \sqrt{2}]$ is if $2\alpha\sin(\alpha) - \kappa\cos(\alpha) = 2\beta\sin(\beta) - \kappa\cos(\beta) = 0$, and hence if $\alpha\tan(\alpha) = \beta\tan(\beta) = \kappa/2$.

5. See the attached Mathematica notebook `ContactInteraction.nb`.

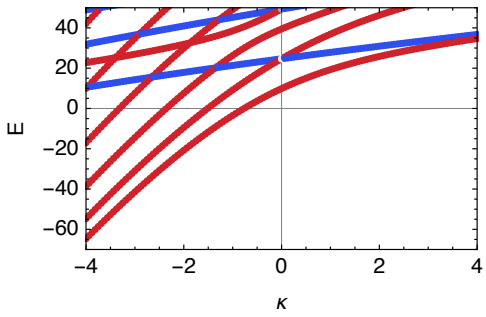
Q4.18 (page 101) We solve this problem with the code of section 4.3.1, in the same way as **Q4.14**. The interaction potential is, according to Equation (4.71),

```
In[927]:=With[{δ=Δ},
          Q[x_] = Piecewise[{{1/Abs[x],Abs[x]>δ}, {1/δ,Abs[x]<=δ}}];]
```

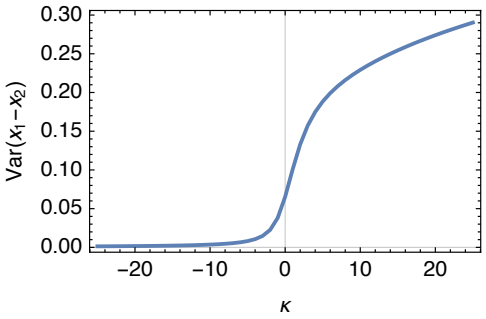
and the interaction Hamiltonian $\hat{\mathcal{H}}_{\text{int}} \approx \kappa/|x|$ is approximately

```
In[928]:=HintP = SparseArray[{j1_,j1_,j2_,j2_} :> Q[xgrid[[j1]]-xgrid[[j2]]],
          {nmax,nmax,nmax,nmax}] //ArrayFlatten;
```

With these definitions, the energy levels are (with `a=m=ħ=1`)



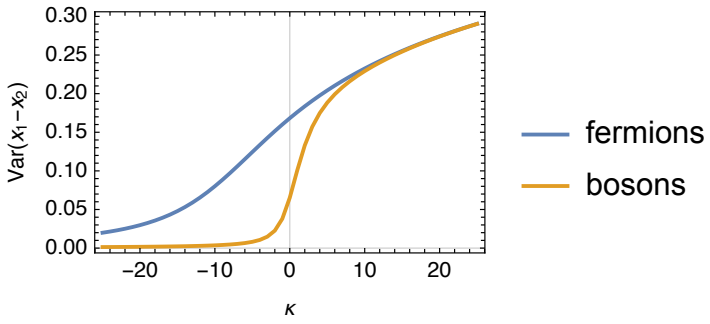
We see that the lowest energy level is always symmetric under particle exchange (colored in red); the bosonic ground state is therefore just the lowest energy level. The expectation value $\langle\hat{x}_1 - \hat{x}_2\rangle$ is zero by symmetry; its variance is



Q4.19 (page 101) We see in the answer of **Q4.18** that the lowest fermionic state (blue) depends on the coupling strength κ. In the spirit of section 4.3.1 we define the fermionic Hamiltonian with `In[529]` and calculate the fermionic ground state with `In[533]`. The expectation values of $\hat{x}_1 - \hat{x}_2$ and $(\hat{x}_1 - \hat{x}_2)^2$ are calculated from the antisymmetric ground state with

```
In[929]:= Clear[FΔx, FΔx2];
In[930]:= FΔx[κ_?NumericQ] := FΔx[κ] =
            With[{γ=ags[0,κ,1][[2,1]]}, Re[Conjugate[γ].((x1-x2).γ)]]
In[931]:= FΔx2[κ_?NumericQ] := FΔx2[κ] =
            With[{γ=ags[0,κ,1][[2,1]]}, Re[Conjugate[γ].((x1-x2).(x1-x2).γ)]]
```

The expectation value $\langle \hat{x}_1 - \hat{x}_2 \rangle$ is zero by symmetry; its variance is larger than that for bosons:



Q4.20 (page 105) The expectation values are the usual ones of the harmonic oscillator, given by

$$\langle x^2 \rangle = \frac{\hbar}{2m\omega_x}, \qquad \langle y^2 \rangle = \frac{\hbar}{2m\omega_y}, \qquad \langle z^2 \rangle = \frac{\hbar}{2m\omega_z}. \tag{25}$$

They are independent in the three Cartesian directions.

Q4.21 (page 106) We calculate the integral over the density in Cartesian coordinates by integrating only over the ellipsoid in which the density is nonzero:

```
In[932]:= A = Assuming[Rx>0 && Ry>0 && Rz>0,
            Integrate[ρ0*(1-(x/Rx)^2-(y/Ry)^2-(z/Rz)^2),
              {x, -Rx, Rx},
              {y, -Ry*Sqrt[1-(x/Rx)^2], Ry*Sqrt[1-(x/Rx)^2]},
              {z, -Rz*Sqrt[1-(x/Rx)^2-(y/Ry)^2], Rz*Sqrt[1-(x/Rx)^2-(y/Ry)^2]}]]
Out[932]= 8/15*π*Rx*Ry*Rz*ρ0
```

Similarly, we calculate the integral of the density times x^2 with

```
In[933]:= B = Assuming[Rx>0 && Ry>0 && Rz>0,
            Integrate[x^2 * ρ0*(1-(x/Rx)^2-(y/Ry)^2-(z/Rz)^2),
              {x, -Rx, Rx},
              {y, -Ry*Sqrt[1-(x/Rx)^2], Ry*Sqrt[1-(x/Rx)^2]},
              {z, -Rz*Sqrt[1-(x/Rx)^2-(y/Ry)^2], Rz*Sqrt[1-(x/Rx)^2-(y/Ry)^2]}]]
Out[933]= 8/105*π*Rx^3*Ry*Rz*ρ0
```

The expectation value $\langle x^2\rangle$ is the ratio of these two integrals,

```
In[934]:=B/A
Out[934]=Rx^2/7
```

With the value of R_x given in Equation (4.79)a, this becomes

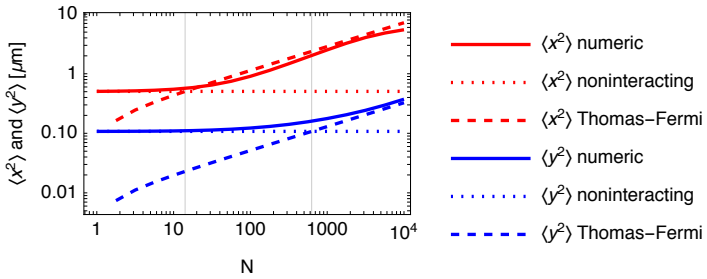
$$\langle x^2\rangle = \frac{1}{7}\left[\frac{15\hbar^2 a_s(N-1)\omega_y\omega_z}{m^2\omega_x^4}\right]^{\frac{2}{5}} = \frac{1}{7}\left[\frac{15\kappa(N-1)\omega_y\omega_z}{4\pi m\omega_x^4}\right]^{\frac{2}{5}}, \tag{26a}$$

$$\langle y^2\rangle = \frac{1}{7}\left[\frac{15\hbar^2 a_s(N-1)\omega_x\omega_z}{m^2\omega_y^4}\right]^{\frac{2}{5}} = \frac{1}{7}\left[\frac{15\kappa(N-1)\omega_x\omega_z}{4\pi m\omega_y^4}\right]^{\frac{2}{5}}, \tag{26b}$$

$$\langle z^2\rangle = \frac{1}{7}\left[\frac{15\hbar^2 a_s(N-1)\omega_x\omega_y}{m^2\omega_z^4}\right]^{\frac{2}{5}} = \frac{1}{7}\left[\frac{15\kappa(N-1)\omega_x\omega_y}{4\pi m\omega_z^4}\right]^{\frac{2}{5}}. \tag{26c}$$

We see that, in contrast to **Q4.20**, the expectation values of the three Cartesian directions are not independent of each other's trapping frequencies.

Q4.22 (page 106) We plot the second moments of Equation (25) and Equation (26)a as functions of the particle number N:



The values of $\langle z^2\rangle$ are equal to those of $\langle y^2\rangle$ because of the cylindrical symmetry of the problem. The crossover point where the Thomas–Fermi second moment is equal to the noninteracting second moment is at

$$\bar{N}_x = \frac{49}{60 a_s\omega_y\omega_z}\sqrt{\frac{7\hbar\omega_x^3}{2m}} + 1, \quad \bar{N}_y = \frac{49}{60 a_s\omega_x\omega_z}\sqrt{\frac{7\hbar\omega_y^3}{2m}} + 1, \quad \bar{N}_z = \frac{49}{60 a_s\omega_x\omega_y}\sqrt{\frac{7\hbar\omega_z^3}{2m}} + 1, \tag{27}$$

indicated with vertical lines in the above plot. The noninteracting limit, Equation (25), is good for $N \lesssim 10$. The Thomas–Fermi limit, Equation (26)a, is good for $N \gtrsim 5000$. Notice that for $N \gtrsim 3000$ the numeric value of $\langle x^2\rangle$ deviates from the Thomas–Fermi limit because of the finite size of the calculation box.

Chapter 5 combining spatial motion and spin

Q5.1 (page 113) The operators for these expectation values are

1. `A1 = KroneckerProduct[xop,Π↑] = KroneckerProduct[xop,ids/2+sz]`
2. `A2 = KroneckerProduct[xop,Π↓] = KroneckerProduct[xop,ids/2-sz]`
3. `A3 = KroneckerProduct[xop,ids] = A1+A2`
4. `A4 = KroneckerProduct[xop,sz] = (A1-A2)/2`

With these we evaluate the quantities

1. `Re[Conjugate[γ].(A1.γ)]`
2. `Re[Conjugate[γ].(A2.γ)]`
3. `Re[Conjugate[γ].(A3.γ)]`
4. `Re[Conjugate[γ].(A4.γ)]` for the mean
   `Re[Conjugate[γ].(A4.A4.γ)-(Conjugate[γ].(A4.γ))^2]` for the variance

Q5.2 (page 115) The ordering of the subspaces of the Hilbert space is what matters here. We have defined the Hilbert space to be a tensor product of the x, y, and spin degrees of freedom, in this order. In `In[629]` the operators $\hat{p}_x$ and $\hat{p}_y$ are distinguished by the position in the Kronecker product in which `pM` appears.

Q5.3 (page 119) We do the first two checks of **Q3.2**:

1.
$$\begin{aligned}
[\hat{S}_x, \hat{S}_y] &= [\frac{|e\rangle\langle g| + |g\rangle\langle e|}{2}, \frac{|e\rangle\langle g| - |g\rangle\langle e|}{2\mathrm{i}}] \\
&= \frac{(|e\rangle\langle g| + |g\rangle\langle e|)(|e\rangle\langle g| - |g\rangle\langle e|) - (|e\rangle\langle g| - |g\rangle\langle e|)(|e\rangle\langle g| + |g\rangle\langle e|)}{4\mathrm{i}} \\
&= \frac{(-|e\rangle\langle e| + |g\rangle\langle g|) - (|e\rangle\langle e| - |g\rangle\langle g|)}{4\mathrm{i}} \\
&= \frac{|g\rangle\langle g| - |e\rangle\langle e|}{2\mathrm{i}} = \mathrm{i}\frac{|e\rangle\langle e| - |g\rangle\langle g|}{2} = \mathrm{i}\hat{S}_z
\end{aligned} \tag{28}$$

$$\begin{aligned}
[\hat{S}_y, \hat{S}_z] &= [\frac{|e\rangle\langle g| - |g\rangle\langle e|}{2\mathrm{i}}, \frac{|e\rangle\langle e| - |g\rangle\langle g|}{2}] \\
&= \frac{(|e\rangle\langle g| - |g\rangle\langle e|)(|e\rangle\langle e| - |g\rangle\langle g|) - (|e\rangle\langle e| - |g\rangle\langle g|)(|e\rangle\langle g| - |g\rangle\langle e|)}{4\mathrm{i}} \\
&= \frac{(-|e\rangle\langle g| - |g\rangle\langle e|) - (|e\rangle\langle g| + |g\rangle\langle e|)}{4\mathrm{i}} \\
&= \frac{-|e\rangle\langle g| - |g\rangle\langle e|}{2\mathrm{i}} = \mathrm{i}\frac{|e\rangle\langle g| + |g\rangle\langle e|}{2} = \mathrm{i}\hat{S}_x
\end{aligned} \tag{29}$$

$$\begin{aligned}
[\hat{S}_z, \hat{S}_x] &= [\frac{|e\rangle\langle e| - |g\rangle\langle g|}{2}, \frac{|e\rangle\langle g| + |g\rangle\langle e|}{2}] \\
&= \frac{(|e\rangle\langle e| - |g\rangle\langle g|)(|e\rangle\langle g| + |g\rangle\langle e|) - (|e\rangle\langle g| + |g\rangle\langle e|)(|e\rangle\langle e| - |g\rangle\langle g|)}{4} \\
&= \frac{(|e\rangle\langle g| - |g\rangle\langle e|) - (-|e\rangle\langle g| + |g\rangle\langle e|)}{4} \\
&= \frac{|e\rangle\langle g| - |g\rangle\langle e|}{2} = \mathrm{i}\frac{|e\rangle\langle g| - |g\rangle\langle e|}{2\mathrm{i}} = \mathrm{i}\hat{S}_y
\end{aligned} \tag{30}$$

2.
$$\begin{aligned}
\hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2 &= \left(\frac{|e\rangle\langle g| + |g\rangle\langle e|}{2}\right)^2 + \left(\frac{|e\rangle\langle g| - |g\rangle\langle e|}{2\mathrm{i}}\right)^2 + \left(\frac{|e\rangle\langle e| - |g\rangle\langle g|}{2}\right)^2 \\
&= \frac{|e\rangle\langle e| + |g\rangle\langle g|}{4} + \frac{|e\rangle\langle e| + |g\rangle\langle g|}{4} + \frac{|e\rangle\langle e| + |g\rangle\langle g|}{4} \\
&= \frac{3}{4}(|g\rangle\langle g| + |e\rangle\langle e|) = \frac{3}{4}\mathbb{1} \text{ and hence } S = 1/2.
\end{aligned} \tag{31}$$

Q5.4 (page 119) $\hat{S}^+ = \hat{S}_x + \mathrm{i}\hat{S}_y = \frac{|e\rangle\langle g|+|g\rangle\langle e|}{2} + \mathrm{i}\frac{|e\rangle\langle g|-|g\rangle\langle e|}{2\mathrm{i}} = \frac{|e\rangle\langle g|+|g\rangle\langle e|}{2} + \frac{|e\rangle\langle g|-|g\rangle\langle e|}{2} = |e\rangle\langle g|$. $\hat{S}^- = \hat{S}_x - \mathrm{i}\hat{S}_y = \frac{|e\rangle\langle g|+|g\rangle\langle e|}{2} - \mathrm{i}\frac{|e\rangle\langle g|-|g\rangle\langle e|}{2\mathrm{i}} = \frac{|e\rangle\langle g|+|g\rangle\langle e|}{2} - \frac{|e\rangle\langle g|-|g\rangle\langle e|}{2} = |g\rangle\langle e|$. We can see that $\hat{S}^+$ is the operator that excites the atom ($\hat{S}^+|g\rangle = |e\rangle$) and $\hat{S}^-$ is the operator that deexcites the atom ($\hat{S}^+|e\rangle = |g\rangle$).

Q5.5 (page 119) $[\hat{X}, \hat{P}] = \hat{X}\hat{P} - \hat{P}\hat{X} = \frac{\hat{a}+\hat{a}^\dagger}{\sqrt{2}}\frac{\hat{a}-\hat{a}^\dagger}{\mathrm{i}\sqrt{2}} - \frac{\hat{a}-\hat{a}^\dagger}{\mathrm{i}\sqrt{2}}\frac{\hat{a}+\hat{a}^\dagger}{\sqrt{2}} = \frac{(\hat{a}\hat{a}-\hat{a}\hat{a}^\dagger+\hat{a}^\dagger\hat{a}-\hat{a}^\dagger\hat{a}^\dagger)-(\hat{a}\hat{a}+\hat{a}\hat{a}^\dagger-\hat{a}^\dagger\hat{a}-\hat{a}^\dagger\hat{a}^\dagger)}{2\mathrm{i}} = \mathrm{i}[\hat{a}, \hat{a}^\dagger] = \mathrm{i}$.

Q5.6 (page 119) Cavity field: $\hat{a}^\dagger\hat{a} = \frac{\hat{X}-\mathrm{i}\hat{P}}{\sqrt{2}}\frac{\hat{X}+\mathrm{i}\hat{P}}{\sqrt{2}} = \frac{\hat{X}^2+\hat{P}^2+\mathrm{i}[\hat{X},\hat{P}]}{2} = \frac{\hat{X}^2+\hat{P}^2-1}{2}$ and hence $\hat{a}^\dagger\hat{a}+\frac{1}{2} = \frac{1}{2}\hat{P}^2+\frac{1}{2}\hat{X}^2$. Coupling: $\hat{S}^+\hat{a}+\hat{a}^\dagger\hat{S}^- = (\hat{S}_x+\mathrm{i}\hat{S}_y)\frac{\hat{X}+\mathrm{i}\hat{P}}{\sqrt{2}} + \frac{\hat{X}-\mathrm{i}\hat{P}}{\sqrt{2}}(\hat{S}_x-\mathrm{i}\hat{S}_y) = \frac{\hat{S}_x\hat{X}+\mathrm{i}\hat{S}_x\hat{P}+\mathrm{i}\hat{S}_y\hat{X}-\hat{S}_y\hat{P}+\hat{X}\hat{S}_x-\mathrm{i}\hat{X}\hat{S}_y-\mathrm{i}\hat{P}\hat{S}_x-\hat{P}\hat{S}_y}{\sqrt{2}}$. Since the operators on the field and atom degrees of freedom commute (for example, $[\hat{X}, \hat{S}_x] = [\hat{X}\otimes\mathbb{1}, \mathbb{1}\otimes\hat{S}_x] = 0$), this becomes $\hat{S}^+\hat{a}+\hat{a}^\dagger\hat{S}^- = \frac{\hat{X}\hat{S}_x+\mathrm{i}\hat{P}\hat{S}_x+\mathrm{i}\hat{X}\hat{S}_y-\hat{P}\hat{S}_y+\hat{X}\hat{S}_x-\mathrm{i}\hat{X}\hat{S}_y-\mathrm{i}\hat{P}\hat{S}_x-\hat{P}\hat{S}_y}{\sqrt{2}} = \sqrt{2}(\hat{X}\hat{S}_x - \hat{P}\hat{S}_y)$.