State-dependent forces in cold quantum gases

Christopher Billington

Submitted in total fulfilment of the requirements
of the degree of Doctor of Philosophy

Supervisory committee:
Prof Kristian Helmerson
Dr Lincoln Turner
Dr Russell Anderson

School of Physics and Astronomy
Monash University

January, 2017

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

This page intentionally left blank

Contents


```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

This page intentionally left blank

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Chapter 3

Quantum mechanics on a computer

This chapter comprises a summary of some of the methods used by cold atom physicists to compute numerical results pertaining to cold atom systems. Many a problem in quantum mechanics is not analytically solvable, especially when the real world of experimental physics rears its ugly head, violating theorists' assumptions of simplicity left and right. In particular, atomic physics experiments are time-dependent, with each run of an experiment generally proceeding in stages. Lasers may turn on and off, magnetic fields may vary in magnitude and direction, RF pulses may be chirped to reliably induce particular transitions [1]. Much of the numerical computation performed by researchers in cold atom physics groups such as ours are accordingly of the time-dependent variety, and are fairly literal simulations of specific experiments that may be carried out in the lab.

Here I explain some fundamentals of representing quantum mechanical problems on a computer and then present my favourite algorithms for propagating state vectors and wavefunctions in time. To spoil the surprise: my favourite timestepping algorithms are the 4th-order split-step method and (unoriginally) 4th-order Runge–Kutta, and my favourite method of evaluating spatial derivatives is moderate (6th or so) order finite differences. I think Fourier transforms for spatial derivatives are overrated, and I show that the finite element discrete variable representation (FEDVR) is, despite appearances, actually less computationally efficient than simple finite differences for producing equally accurate solutions to the spatial Scrödinger equation. I also mention some methods of finding groundstates and other stationary states, and in section **??** I present a modification to 4th-order Runge–Kutta that enables it to take larger timesteps for certain problems.

3.1 From the abstract to the concrete: neglect, discretisation and representation

To numerically simulate a quantum mechanical system, one must evolve a state vector in time according to the Schrödinger equation:

$$i\hbar \frac{\mathrm{d}}{\mathrm{d}t} |\psi(t)\rangle = \hat{H}(t) |\psi(t)\rangle \,. \tag{3.1}$$

To do this on a computer, one must first decide which degrees of freedom are to be simulated. We necessarily neglect many degrees of freedom as a matter of course; which ones can be neglected is warranted by the specific situation and we do it so often we barely notice. For example, simulating a single component Bose–Einstein condensate entails neglecting the internal degrees of freedom of the atoms—as well as reducing the atom-light interaction to a simple potential such as an optical dipole trap or magnetic dipole interaction (neglecting the quantum degrees of freedom in the electromagnetic field).

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

We may ignore one or more spatial degrees of freedom as well, say, if we are simulating an experiment in which the condensate is confined to one or two dimensions [2–4] by way of a tight trapping potential in one or more directions. Or, when simulating laser cooling [SEE SECTION ON LASER COOLING SIMS IN ATOMIC PHYSICS CHAPTER], we may care very much about the electronic state of the atom, but treat its motional state classically. In these cases we are essentially imposing the assumption that the system will only occupy one state with respect to those degrees of freedom ignored (the condensate will remain in lowest excitation level in the direction of the tight trap; the atoms will remain in one specific Zeeman sublevel), or we are assuming those degrees of freedom can be treated classically (the electromagnetic field is well described by classical electromagnetism; the atoms' motional state is described well by Newtonian mechanics). Which degrees of freedom can be neglected and which cannot requires knowledge of the situation at hand, often informed by best-practices of the research community in question and ultimately justified by experiment.[1]

Once the degrees of freedom are known, one must decide on a basis in which to represent them concretely. The basis often cannot be complete, since for many degrees of freedom this would require an infinite number of basis states—for example the electronic state of an atom contains a countably infinite number of states, and a spatial wavefunction in free space has an uncountable number of states (one for each position in $\mathbb{R}^3$). For the internal state of an atom, therefore, we restrict ourselves to only the states we expect can become non-negligibly occupied, given the initial conditions and transitions involved. For example, at low temperature we can expect atoms to be almost completely in their electronic ground states, since energy gaps between ground and excited states are large compared to the thermal energy scale $k_B T$.[2]. We need only include the small number of excited states that might become occupied as a result of optical transitions present in the situation being simulated. This can still be a large number of states if one is studying Rydberg atoms [7, 8] or using ultrafast (and therefore broad-band) laser pulses [9–11], but is otherwise fairly small. For example, including both the D1 and D2 lines of Rubidium 87, with all hyperfine levels and Zeeman sublevels gives 32 states (see section [LASER COOLING SIMS IN ATOMIC PHYSICS CHAPTER]).

For spatial degrees of freedom, we usually limit ourselves firstly to a finite region of space (we don't expect the Bose–Einstein condensate to have much probability amplitude on the Moon or anywhere else outside the vacuum system), and then we need to discretise the region of space remaining. To do this one can either discretise space on a grid, or use a set of orthogonal basis functions, and strictly speaking these can be equivalent, as we will soon see.

Once the degrees of freedom and basis vectors have been chosen, the state vector is then represented on a computer as an array of complex numbers, giving the coefficients of each basis vector required to represent a particular state vector. Matrix elements of the Hamiltonian in the same basis must be calculated, and the Schrödinger equation can then be written:

$$i\hbar \frac{\mathrm{d}}{\mathrm{d}t} \langle n|\psi(t)\rangle = \sum_m \langle n|\hat{H}(t)|m\rangle \langle m|\psi(t)\rangle\,, \tag{3.2}$$

or in standard matrix/vector notation (without Dirac notation):

$$i\hbar \frac{\mathrm{d}}{\mathrm{d}t} \psi_n(t) = \sum_m H_{nm}(t)\psi_m(t) \tag{3.3}$$

$$\iff i\hbar \frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{\psi}(t) = H(t)\boldsymbol{\psi}(t), \tag{3.4}$$

where $\psi_n(t) = \langle n|\psi(t)\rangle$, $H_{nm}(t) = \langle n|\hat{H}(t)|m\rangle$ and $\boldsymbol{\psi}(t)$ and $H(t)$ are the vector and matrix with components and elements $\{\psi_n(t)\}$ and $\{H_{nm}\}$ respectively. This is now

[1] A classic example in the cold atom community of neglected degrees of freedom leading to *disagreement* with experiment is the discovery of polarisation gradient cooling (PGC), the explanation for which requires consideration of Zeeman sublevels of the atoms. The experiment that discovered PGC [5] was designed to measure the effect of Doppler cooling, which does not involve Zeeman sublevels, and it was not until afterwards that theorists determined [6] that transitions between Zeeman sublevels cannot be neglected and indeed are crucial in explaining the lower than predicted temperatures observed.

[2] The D-line of rubidium 87 has an energy gap of 0.6 eV, requiring a temperature of $\approx$ 650 K or higher in order for the Boltzmann factor $e^{-\frac{\Delta E}{k_B T}}$ describing the thermal occupation of the excited state to exceed 1×10^{-12}.

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

something very concrete that can be typed into a computer. Programming languages generally don't know about Dirac kets and operators, and so everything that is to be computed must be translated into matrices and vectors in specific bases. This may seem so obvious as to not be worth mentioning, but was nonetheless a stumbling block in my own experience of getting to grips with quantum mechanics. Once realising that every operator has a matrix representation in some basis, at least in principle (including differential operators), and that every ket is just a list of vector components in some basis (including ones representing spatial wavefunctions), similarly at least in principle, expressions dense in bras and kets become much more concrete as the reader has a feel for exactly how they would type it into a computer. Without a good feel for the mapping between operator algebra and the actual lists of numbers that these objects imply, doing quantum mechanics on paper can seem like an exercise in abstract mumbo-jumbo.

3.2 Solution to the Schrödinger equation by direct exponentiation

As an example of something seemingly abstract being more concrete than first appearances, it is sometimes said that the 'formal' solution to the Schrödinger equation (3.1) is:

$$|\psi(t)\rangle = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t}\hat{H}(t')\,\mathrm{d}t'}\,|\psi(t_0)\rangle\,. \tag{3.5}$$

Saying that this is the 'formal' solution rather than just 'the solution' is presumably intended to emphasise that the arithmetic operations involved in (3.5) might not make immediate sense for the types of mathematical objects they are operating on, and that we have to be careful in defining the operations such that they produce a result that is not only sensible, but also the solution to the Schrödinger equation. If both $\hat{H}(t)$ and $|\psi(t)\rangle$ were single-valued functions of time rather than an operator valued function of time (the values at different times of which don't necessarily commute) and a vector valued function of time, then we would have no problem. However, (3.5) as written with operators and vectors is ambiguous, and we need to elaborate on it in order to ensure it is correct. I will come back to this after considering a simpler case.

If the Hamiltonian is time-independent, then (3.5) reduces to

$$|\psi(t)\rangle = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}\hat{H}\Delta t}\,|\psi(t_0)\rangle\,, \tag{3.6}$$

where $\Delta t = t - t_0$. Given the matrix representation H of $\hat{H}$ and vector representation $\boldsymbol{\psi}(t_0)$ of $|\psi(t_0)\rangle$ in a particular basis, this can now be directly typed into a computer as the matrix multiplication:

$$\boldsymbol{\psi}(t) = U(t, t_0)\boldsymbol{\psi}(t_0), \tag{3.7}$$

where

$$U(t, t_0) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H\Delta t} \tag{3.8}$$

is the (matrix representation of the) unitary evolution operator for time evolution from the initial time t_0 to time t, and is computed using a matrix exponential of $-\frac{\mathrm{i}}{\hbar}H\Delta t$. Exponentiation of matrices is defined via the Taylor series of the exponential function:

$$\mathrm{e}^{A} = \sum_{n=0}^{\infty}\frac{A^n}{n!}, \tag{3.9}$$

which reduces matrix exponentiation to the known operations of matrix multiplication and addition. However, any linear algebra programming library worth the bytes it occupies will have a matrix exponentiation function that should be used instead, as there are

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

other methods of computing matrix exponentials that are more computationally efficient and numerically stable, such as the Padé approximant [CITE]. It should be noted that there is no known 'best' matrix exponential algorithm, all make compromises and perform poorly for certain types of matrices [12].

3.2.1 Matrix exponentiation by diagonalisation

Regardless of which method is used, matrix exponentiation is computationally expensive. It can be sped up however if a diagonalisation of H is known, since if

$$H = UDU^\dagger, \tag{3.10}$$

where D is a diagonal matrix and U is a unitary matrix[3], then

$$e^{-\frac{i}{\hbar}H\Delta t} = Ue^{-\frac{i}{\hbar}D\Delta t}U^\dagger. \tag{3.11}$$

[3] Note that the diagonals of D are the eigenvalues of H, and the columns of U are its eigenvectors.

This is simple to evaluate because the exponentiation of a diagonal matrix can be performed by exponentiating each diagonal matrix element individually[4]

[4] The reason for this is clear from the Taylor series definition of matrix exponentiation, since matrix multiplication and addition can both be performed elementwise for diagonal matrices.

Even if a diagonalisation of H is not analytically known, numerically diagonalising H (using a linear algebra library function or otherwise) can form the basis for writing your own matrix exponentiation function, if needed. I found this necessary for efficiently exponentiating an array of matrices in Python, since the `scipy` and `numpy` scientific and numeric libraries at the present time lack matrix exponentiation functions that can act on arrays of matrices. Writing a `for` loop in an interpreted language such as Python to exponentiate the matrices individually in many cases is unacceptably slow, so for these cases[5] I use a function such as the below:

[5] Such as simulating the internal state of a large number of atoms, or evolving a spinor Bose–Einstein condensate by exponentiating the Zeeman Hamiltonian with a spatially varying magnetic field.

```
import numpy as np
from numpy.linalg import eigh

def expmh(M):
    """compute exp(M), where M, shape (..., N, N) is an array of N by N
    Hermitian matrices, using the diagonalisation method. Made this function
    because scipy's expm can't take an array of matrices as input, it can only
    do one at a time."""

    # Diagonalise the matrices:
    evals, evecs = eigh(M)

    # Now we compute exp(M) = U exp(D) U^\dagger where U is the matrix of
    # eigenvectors (as columns) and D is the diagonal matrix of eigenvalues:

    U = evecs
    U_dagger = U.conj().swapaxes(-1, -2) # Only transpose the matrix dimensions
    exp_D_diags = np.exp(evals)

    # Compute the 3-term matrix product U*exp_D_diags*U_dagger using the
    # einsum function in order to specify which array axes of each array to
    # sum over:
    return np.einsum('...ik,...k,...kj->...ij', U, exp_D_diags, U_dagger)
```

Matrix diagonalisation (using singular value decomposition or QR decomposition) has computational time complexity $\mathcal{O}(n^3)$, where n is the number of rows/columns in the (square) matrix. Matrix multiplication is (in practice) $\mathcal{O}(n^3)$ and exponentiating a diagonal matrix is only $\mathcal{O}(n)$, so matrix exponentiation of a Hermitian matrix via numerical diagonalisation has total cost $\mathcal{O}(n^3)$. This compares to the Padé approximant, which is also $\mathcal{O}(n^3)$ [12]. So the numerical diagonalisation method is not any worse in terms of computational resources required.

On the other hand, if an analytic diagonalisation is already known, it would seem that exponentiation is just as slow, since the computational cost of matrix multiplication

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

alone is the same order in n as that of numerical diagonalisation. This is true - so there are only constant factors to be saved in computer time by using an analytic diagonalisation in order to exponentiate a matrix using (3.11). However if one's aim—as is often the case—is to ultimately compute

$$\psi(t) = e^{-\frac{i}{\hbar}H\Delta t}\psi(t_0), \tag{3.12}$$

for a specific $\psi(t_0)$, then one needs only matrix-vector multiplications and not matrix-matrix multiplications in order to evaluate

$$\psi(t) = Ue^{-\frac{i}{\hbar}D\Delta t}U^\dagger\psi(t_0), \tag{3.13}$$

from right-to-left, reducing the computational cost to $\mathcal{O}(n^2)$ compared to evaluating it left-to-right.

Whilst matrix exponentiation is a way to efficiently evolve systems with time-independent Hamiltonians, if you only exponentiate a matrix once, you don't much care about the time complexity of doing so. It is mostly of interest because the real real power of these exponentiation methods is as a building block for methods of approximate solutions to the Schrödinger equation in the case of time *dependent* Hamiltonians, as we will see in the next section.

3.2.2 Time-ordered exponentials and time-ordered products

As hinted to earlier, the solution (3.5) is not the whole picture. It can only be taken at face value if the Hamiltonian at each moment in time commutes with itself at all other times [CITE SOMETHING]. In this case, once represented in a specific basis, the solution to the Schrödinger equation is again the matrix multiplication

$$\psi(t) = U(t, t_0)\psi(t_0), \tag{3.14}$$

with

$$U(t, t_0) = e^{-\frac{i}{\hbar}\int_{t_0}^{t} H(t')\,dt'}, \tag{3.15}$$

i.e. with an integral in the exponent rather than a simple multiplication by a time interval. Since matrix addition can be performed elementwise, so can the integral in the exponent, yielding a matrix which once exponentiated will give the evolution operator $U(t, t_0)$ for the solution to the Schrödinger equation. If the Hamiltonian at each moment in time does not commute with itself at all other times, however, then the unitary evolution operator for the solution to the Schrödinger equation is instead given by the following *time-ordered exponential* [CITE]:

$$U(t, t_0) = \mathcal{T}\left\{e^{-\frac{i}{\hbar}\int_{t_0}^{t} H(t')\,dt'}\right\}. \tag{3.16}$$

In this expression, $\mathcal{T}$ denotes the *time-ordering operator*. The time ordering operator reorders terms within products that contain a time parameter (for us, the time parameter is the argument of the matrix-valued function H), such that the value of the time parameter is smallest in the rightmost term, largest in the leftmost term, and monotonically increasing right-to-left in between. For example:

$$\mathcal{T}\{H(4)H(1)H(2)H(5)H(3)\} = H(5)H(4)H(3)H(2)H(1). \tag{3.17}$$

Despite appearances, this time-ordered exponential is perfectly concretely defined via the definitions of all the operations involved that we have described so far, and can—with

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

some effort—be typed into a computer and evaluated directly. Even though this is not how I have evaluated time-ordered exponentials in my simulations of atomic systems, I'll quickly elaborate on this just to emphasise the concreteness of all these operations.

"What products is $\mathcal{T}$ reordering?" you might ask, as (3.16) doesn't appear to contain any products of $H(t)$. On the contrary, it does, since exponentiation is defined by its Taylor series, and so

$$U(t,t_0) = 1 + \mathcal{T}\left\{\sum_{n=1}^{\infty}\frac{1}{n!}\left[-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t}H(t')\,\mathrm{d}t'\right]^n\right\} \tag{3.18}$$

$$= 1 + \sum_{n=1}^{\infty}\frac{1}{n!}\left(-\frac{\mathrm{i}}{\hbar}\right)^n\mathcal{T}\left\{\left[\int_{t_0}^{t}H(t')\,\mathrm{d}t'\right]^n\right\}. \tag{3.19}$$

Each term in this series contains the n^{th} power (and hence a product) of an integral of $H(t)$. The time ordering operator doesn't allow us to evaluate each term by computing the matrix integral once and then raising it to a power—to do so would violate time-ordering since each integral involves evaluating $H(t)$ at all times. Instead we have to write each product of integrals as the integral of a product:

$$U(t,t_0) = 1 + \sum_{n=1}^{\infty}\frac{1}{n!}\left(-\frac{\mathrm{i}}{\hbar}\right)^n\int_{t_0}^{t}\mathrm{d}t_1'\int_{t_0}^{t}\mathrm{d}t_2'\cdots\int_{t_0}^{t}\mathrm{d}t_n'\cdots\mathcal{T}\left\{H(t_1')H(t_2')\cdots H(t_n')\right\}, \tag{3.20}$$

from which we can see exactly which product of matrices the time ordering operator is acting on.

Now we are close to seeing one might evaluate $U(t,t_0)$ numerically by summing each term in the Taylor series up to some order set by the required accuracy. For the n^{th} term, one needs to evaluate an n-dimensional integral over n time coordinates, with each coordinate having the same limits of integration. This can be computed in the usual way an integral is numerically computed,[6] with the minor change that each time the integrand is evaluated, the terms within it must be re-ordered to respect the required time-ordering. Alternatively, the integration region can be restricted to the region in which the terms are already time-ordered, and then the total integral inferred by symmetry, which gives:

$$U(t,t_0) = 1 + \sum_{n=1}^{\infty}\left(-\frac{\mathrm{i}}{\hbar}\right)^n\int_{t_0}^{t}\mathrm{d}t_n'\cdots\int_{t_0}^{t_3'}\mathrm{d}t_2'\int_{t_0}^{t_2'}\mathrm{d}t_1'\;H(t_n')\cdots H(t_2')H(t_1'). \tag{3.21}$$

This is now a perfectly concrete expression, with each term comprising an integral over an n-simplex[7] of a product of n matrices.

This expression for the unitary evolution operator is called the Dyson series [CITE]. It is not generally used for time-dependent simulations, though it is the basis for time-dependent perturbation theory, and sees use in high energy physics [CITE] for computing transition amplitudes between incoming and outgoing waves in scattering problems (in which U is called the S-matrix). In these problems, H is an interaction Hamiltonian containing terms for all particle interactions being considered. Accordingly, the integrand for the n^{th} term, being a product of n copies of H evaluated at different times, contains one term for each possible sequence of n particle interactions. The integral itself can be considered a sum of transition amplitudes over all possible times that each interaction could have occurred. Indeed, each term corresponds to a Feynman diagram with n nodes [CITE].

The Dyson series isn't really suited to time-dependent simulations, though perturbation theory is useful for approximate analytics. For one, the series must be truncated at

[6] By sampling the integrand on a uniform grid, using a quadrature method, or Monte-Carlo integration [13] which is widely used for high dimensional integrals such as these.

[7] A simplex is the generalisation of a triangle to higher dimensions, i.e. a 3-simplex is a tetrahedron.

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

some point, and the result won't be a U that is actually unitary.[8]. Also, we are typically interested in the intermediate states, not just the final state of a system or an average transition rate, as one might want to compute in a scattering problem.

In any case, usually when solving the Schrödinger equation by exponentiation we use the following, alternate expression for a time-ordered exponential:

$$U(t,t_0) = \mathcal{T}\left\{\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t} H(t')\,\mathrm{d}t'}\right\} \tag{3.22}$$

$$= \lim_{N\to\infty} \prod_{n=N-1}^{0} \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H(t_n)\Delta t}, \tag{3.23}$$

where here $\Delta t = (t-t_0)/N$ and $t_n = t_0 + n\Delta t$. Note that the product limits are written in the reverse of the usual order—this is important in order to produce terms with smaller n on the right and larger n on the left of the resulting product. You can convince yourself that (3.21) is equivalent to (3.23) this by replacing the integral in the exponent with a sum—as per the Riemann definition of an integral—and expanding the exponential according to its Taylor series. Expanding each exponential in (3.23) as a Taylor Series and collecting terms of equal powers of H then reveals that the two Taylor series are identical.

In any case, (3.23) paints an intuitive picture of solving the Schrödinger equation: one evolves the initial state vector in time by evolving it according to constant Hamiltonians repeatedly over small time intervals. This has the desirable property that all intermediate state vectors are computed at the intermediate steps, meaning one can study the dynamics of the system and not just obtain the final state. This is of course useful for comparison with experiments, plenty of which involve time-dependent data acquisition and not just post-mortem analysis of some evolution.

Numerically, we can't actually take $N \to \infty$, or equivalently $\Delta t \to 0$, and so we instead choose a Δt smaller than the timescale of any time-dependence of H, and step through time using

$$\psi(t_{n+1}) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H(t_n)\Delta t}\psi(t_n) + \mathcal{O}(\Delta t^2) \tag{3.24}$$

$$\Rightarrow \psi(t) = \left(\prod_{n=N-1}^{0} \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H(t_n)\Delta t}\right)\psi(t_0) + \mathcal{O}(\Delta t). \tag{3.25}$$

Thus the case of a time-dependent Hamiltonian reduces to repeated application of the solution (3.7) for a time-independent Hamiltonian, and is (globally) accurate to order Δt. Note that the entire expression can be evaluated right-to-left to propagate the intial state vector in time without explicitly computing the overall unitary, which ensures the computational complexity is $\mathcal{O}(n^2)$ in the size of the system (when the exponentiation is performed via an analytic or pre-computed diagonalisation) rather than $\mathcal{O}(n^3)$.

3.2.3 The operator product/split-step method

Here I'll present decompositions similar to (3.24), but which use variable timesteps to achieve an accuracy to higher order in Δt. I'll present them at the same time as addressing another problem, which is that Hamiltonians are often not in a simple enough form to be exponentiated efficiently at all, making (3.24) difficult to evaluate. Often Hamiltonians are a sum of non-commuting operators (such as kinetic and potential terms), with time dependence such that any diagonalisation of the overall Hamiltonian at one point in time will not diagonalise it at another point in time, with the system size large enough for numerical diagonalisation to be prohibitively expensive. In these cases, we

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

can use methods called *split-step* or *operator product* methods, which allow one to approximately exponentiate the entire Hamiltonian based on having exact diagonalisations of its component terms. In the case of time-dependent Hamiltonians, this allows us to avoid rediagonalising at every timestep whenever the time-dependence can be expressed as scalar coefficients multiplying time-independent operators:

$$\hat{H}(t) = \alpha(t)\hat{H}_1 + \beta(t)\hat{H}_2 + \cdots, \tag{3.26}$$

since multiplication of a matrix by a scalar merely scales its eigenvalues, leaving its eigenbasis unchanged.

There is little downside to having an only approximate exponentiation of the Hamiltonian when the timestepping is already only approximate, so long as we ensure that neither source of error is much greater than the other. To this end I'll show split-step methods that have (global) error $\mathcal{O}(\Delta t)$, $\mathcal{O}(\Delta t^2)$ and $\mathcal{O}(\Delta t^4)$. In section [secref:continuous degrees of freedom], we'll see how this method applies to the case of a spatial wavefunction obeying the Gross–Pitaevskii equation.

First order split-step

Say we have (the matrix representation of) a Hamiltonian that is the sum of two non-commuting terms:[9]

$$H(t) = H_1(t) + H_2(t). \tag{3.27}$$

[9] From here on, component terms of the Hamiltonian will be written with general time dependence, even though it is understood that these methods are mostly useful for the case where that time dependence can be written as scalar coefficients multiplying otherwise time-independent matrices.

The unitary for the solution to the Schrödinger equation is as before:

$$U(t, t_0) = \mathcal{T}\left\{\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}\int_{t_0}^{t} H(t')\,\mathrm{d}t'}\right\}, \tag{3.28}$$

which, without loss of exactness, we can split into N equal time intervals of size Δt and write:

$$U(t, t_0) = \prod_{n=N-1}^{0} U(t_{n+1}, t_n), \tag{3.29}$$

where again $\Delta t = (t - t_0)/N$ and $t_n = t_0 + n\Delta t$, and where

$$U(t_{n+1}, t_n) = \mathcal{T}\left\{\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}\int_{t_n}^{t_{n+1}} H(t')\,\mathrm{d}t'}\right\}. \tag{3.30}$$

Evaluating the integral using a one-point rectangle rule gives:

$$U(t_{n+1}, t_n) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H(t_n)\Delta t + \mathcal{O}(\Delta t^2)}, \tag{3.31}$$

in which we were able to drop the time ordering operator because $H(t)$ is only evaluated at a single time. Using the Taylor series definition of the exponential, we can take the error term out of the exponent and write:

$$U(t_{n+1}, t_n) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H(t_n)\Delta t} + \mathcal{O}(\Delta t^2). \tag{3.32}$$

So far all we've done is justify (3.24). Now we'll acknowledge that H is a sum of two terms and write:

$$U(t_{n+1}, t_n) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}(H_1(t_n)+H_2(t_n))\Delta t} + \mathcal{O}(\Delta t^2). \tag{3.33}$$

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Using the Baker–Campbell–Hausdorff formula [CITE], we can expand the exponential as:

$$e^{-\frac{i}{\hbar}(H_1(t_n)+H_2(t_n))\Delta t} = e^{-\frac{i}{\hbar}H_1(t_n)\Delta t}e^{-\frac{i}{\hbar}H_2(t_n)\Delta t}e^{\frac{1}{2\hbar^2}\left[H_1(t_n),H_2(t_n)\right]\Delta t^2+\mathcal{O}(\Delta t^3)} \tag{3.34}$$

$$= e^{-\frac{i}{\hbar}H_1(t_n)\Delta t}e^{-\frac{i}{\hbar}H_2(t_n)\Delta t} + \mathcal{O}(\Delta t^2) \tag{3.35}$$

$$\Rightarrow U(t_{n+1}, t_n) = e^{-\frac{i}{\hbar}H_1(t_n)\Delta t}e^{-\frac{i}{\hbar}H_2(t_n)\Delta t} + \mathcal{O}(\Delta t^2). \tag{3.36}$$

So we see that the 'commutation error' in (3.35) caused by treating the two terms as if they commute is no greater than the 'integration error' in (3.32) caused by treating the overall Hamiltonian as time-independent over one timestep, resulting in a method with local error $\mathcal{O}(\Delta t^2)$ and global error $\mathcal{O}(\Delta t)$; the first order split-step method:

$$U(t, t_0) = \prod_{n=N-1}^{0} U_1(t_{n+1}, t_n) + \mathcal{O}(\Delta t), \tag{3.37}$$

where

$$U_1(t_{n+1}, t_n) = e^{-\frac{i}{\hbar}H_1(t_n)\Delta t}e^{-\frac{i}{\hbar}H_2(t_n)\Delta t}. \tag{3.38}$$

Using the diagonalisation method of matrix exponentiation discussed in [SECREF], this unitary U_1 can be used to step a state vector through time with only matrix-vector multiplications and scalar exponentiation, where separate diagonalisations of H_1 and H_2 are either analytically known or pre-computed, even though a diagonalisation of the total Hamiltonian H may not be feasible.

Crucially, if H_1 and H_2 act on different subspaces of the overall Hilbert space, with respective dimensionalities n_1 and n_2, that is, $H_1(t) = \tilde{H}_1(t) \otimes \mathbb{I}_{n_2}$ and $H_2(t) = \mathbb{I}_{n_1} \otimes \tilde{H}_2(t)$, where $\mathbb{I}_m$ is the $m \times m$ identity matrix, then the matrix-vector products involved in applying U_1 to a state vector can be much faster ($\mathcal{O}(n_1^2 n_2 + n_1 n_2^2)$ rather than $\mathcal{O}(n_1^2 n_2^2)$) than if the Hamiltonian weren't split into two terms, even if an analytic diagonalisation of the total Hamiltonian were available:

$$U_1(t_{n+1}, t_n) = \left(e^{-\frac{i}{\hbar}\tilde{H}_1(t_n)\Delta t} \otimes \mathbb{I}_{n_2}\right)\left(\mathbb{I}_{n_1} \otimes e^{-\frac{i}{\hbar}\tilde{H}_2(t_n)\Delta t}\right). \tag{3.39}$$

By applying (3.35) recursively for the case where either H_1 or H_2 is itself the sum of two further terms, (3.38) immediately generalises to the case where H is a sum of an arbitrary number of non-commuting terms:

$$U_1(t_{n+1}, t_n) = \prod_{m=1}^{M} e^{-\frac{i}{\hbar}H_m(t_n)\Delta t}, \tag{3.40}$$

where $H(t) = \sum_{m=1}^{M} H_m(t)$.

Second and fourth order split-step

By using a two-point trapezoid rule for the integral in (3.30), evaluating $H(t)$ at the beginning and end of the timestep, one can instead obtain an integration error of $\mathcal{O}(\Delta t^3)$ per step:

$$U(t_{n+1}, t_n) = e^{-\frac{i}{\hbar}(H(t_n)+H(t_{n+1}))\frac{\Delta t}{2}} + \mathcal{O}(\Delta t^3) \tag{3.41}$$

$$= e^{-\frac{i}{\hbar}(H_1(t_n)+H_2(t_n)+H_1(t_{n+1})+H_2(t_{n+1}))\frac{\Delta t}{2}} + \mathcal{O}(\Delta t^3). \tag{3.42}$$

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Then, applying the Baker–Campbell–Hausdorff formula once more, and replacing $H_1(t_{n+1})$ (and similarly for $H_2(t_{n+1})$) wherever it appears in a commutator with the Taylor series $H_1(t_n)+H_1'(t_n)\Delta t+\mathcal{O}(\Delta t^2)$, one can show that if the individual exponentials are ordered in the following way, then the remaining commutation error is also $\mathcal{O}(\Delta t^3)$:

$$U(t_{n+1},t_n)=\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_2(t_{n+1})\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_1(t_{n+1})\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_1(t_n)\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_2(t_n)\frac{\Delta t}{2}}+\mathcal{O}(\Delta t^3). \tag{3.43}$$

This gives the second order split-step method:

$$U(t,t_0)=\prod_{n=N-1}^{0}U_2(t_{n+1},t_n)+\mathcal{O}(\Delta t^2), \tag{3.44}$$

where

$$U_2(t_{n+1},t_n)=\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_2(t_{n+1})\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_1(t_{n+1})\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_1(t_n)\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_2(t_n)\frac{\Delta t}{2}}, \tag{3.45}$$

or, for an arbitrary number of terms in the Hamiltonian,

$$U_2(t_{n+1},t_n)=\left(\prod_{m=M}^{1}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_m(t_{n+1})\frac{\Delta t}{2}}\right)\left(\prod_{m=1}^{M}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_m(t_n)\frac{\Delta t}{2}}\right). \tag{3.46}$$

U_2 can be concisely written in terms of U_1 as:

$$U_2(t_{n+1},t_n)=U_1^\dagger(t_n+\tfrac{\Delta t}{2},t_{n+1})U_1(t_n+\tfrac{\Delta t}{2},t_n), \tag{3.47}$$

which is to say it is simply two applications of U_1, each of duration half a total timestep, and with the multiplication order of the exponentials reversed in one compared to the other. Two shortcuts are immediately apparent when computing U_2 or its action on a vector: firstly, if there is a time-independent term in the Hamiltonian, it should be assigned to H_1, so that the innermost two exponentials in (3.45) can be collapsed into one; secondly the final exponential of each timestep is identical to the first exponential of the next timestep, and so these can also be collapsed together (being split apart only at points in time when one wishes to sample the state vector).

The fourth order split-step method is much more difficult to derive, with a large number of commutators needing to cancel exactly to ensure the local error cancels out up to fourth order in Δt. It can be stated in terms of the second order split-step method as [CITE]:

$$U(t,t_0)=\prod_{n=N-1}^{0}U_4(t_{n+1},t_n)+\mathcal{O}(\Delta t^4), \tag{3.48}$$

where

$$\begin{aligned}U_4(t_{n+1},t_n)=\;&U_2(t_{n+1},t_{n+1}-p\Delta t)\\&\times U_2(t_{n+1}-p\Delta t,t_{n+1}-2p\Delta t)\\&\times U_2(t_{n+1}-2p\Delta t,t_n+2p\Delta t)\\&\times U_2(t_n+2p\Delta t,t_n+p\Delta t)\\&\times U_2(t_n+p\Delta t,t_n),\end{aligned} \tag{3.49}$$

where $p=1/(4-4^{1/3})$. U_4 comprises five applications of U_2 with timesteps $p\Delta t$, $p\Delta t$, $(1-4p)\Delta t$, $p\Delta t$, and $p\Delta t$ respectively. The innermost timestep is backwards in time, since $(1-4p)<0$. But this is no problem, one simply evaluates the expression for U_2 with a negative timestep exactly as written, so long as one reads Δt when written within the above expressions for $U_1(t_f,t_i)$ and $U_2(t_f,t_i)$ as referring to t_f-t_i, i.e. the difference between the two arguments to the expression, not its absolute value, and not to the timestep of the method in which it is embedded.

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Parallelisability and other speedups for banded matrices

Expressions such as (3.45) don't at first glance appear particularly easy to parallelise, that is, to be evaluated in such a way as to leverage the computing power of multiple CPU cores, GPU cores, or multiple computers. A series of Hamiltonian terms must be exponentiated one by one and multiplied together. Whilst each exponential factor could be evaluated independently, and then all of them multiplied together before being applied to a state vector, this explicit construction of U is very costly compared to merely computing its action on a particular state vector, since the latter allows each term to act only in its specific subspace of the overall Hilbert space, and avoids having to pay the $\mathcal{O}(n^3)$ cost of matrix-matrix multiplication.

When one or more terms in the Hamiltonian has a matrix representation which is banded however, that is, all its entries further than a finite number of elements away from the main diagonal are zero, then those terms may be written as a sum of block diagonal matrices, for example:

$$A = \begin{bmatrix}
c & d & e & & & & & & & & & & & & & \\
b & c & d & e & & & & & & & & & & & & \\
a & b & c & d & e & & & & & & & & & & & \\
 & a & b & c & d & e & & & & & & & & & & \\
 & & a & b & c & d & e & & & & & & & & & \\
 & & & a & b & c & d & e & & & & & & & & \\
 & & & & a & b & c & d & e & & & & & & & \\
 & & & & & a & b & c & d & e & & & & & & \\
 & & & & & & a & b & c & d & e & & & & & \\
 & & & & & & & a & b & c & d & e & & & & \\
 & & & & & & & & a & b & c & d & e & & & \\
 & & & & & & & & & a & b & c & d & & & \\
 & & & & & & & & & & a & b & c & & & \\
\end{bmatrix}$$

$$= \begin{bmatrix}
c & d & e & & & & & & & & & & & & & \\
b & c & d & e & & & & & & & & & & & & \\
a & b & c & d & e & & & & & & & & & & & \\
 & a & b & c & d & e & & & & & & & & & & \\
 & & a & b & c/2 & d/2 & \cdot & & & & & & & & & \\
 & & & a & b/2 & c/2 & \cdot & \cdot & & & & & & & & \\
 & & & & \cdot & \cdot & \cdot & \cdot & \cdot & & & & & & & \\
 & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & & & & & & \\
 & & & & & & \cdot & \cdot & c/2 & d/2 & e & & & & & \\
 & & & & & & & \cdot & b/2 & c/2 & d & e & & & & \\
 & & & & & & & & a & b & c & d & e & & & \\
 & & & & & & & & & a & b & c & d & e & & \\
 & & & & & & & & & & a & b & c/2 & d/2 & \cdot & \\
 & & & & & & & & & & & a & b/2 & c/2 & \cdot & \cdot \\
 & & & & & & & & & & & & \cdot & \cdot & \cdot & \cdot \\
 & & & & & & & & & & & & & \cdot & \cdot & \cdot \\
\end{bmatrix} + \begin{bmatrix}
\cdot & \cdot & \cdot & & & & & & & & & & & & & \\
\cdot & \cdot & \cdot & \cdot & & & & & & & & & & & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & & & & & & & & & & & \\
 & \cdot & \cdot & \cdot & \cdot & \cdot & & & & & & & & & & \\
 & & \cdot & \cdot & c/2 & d/2 & e & & & & & & & & & \\
 & & & \cdot & b/2 & c/2 & d & e & & & & & & & & \\
 & & & & a & b & c & d & e & & & & & & & \\
 & & & & & a & b & c & d & e & & & & & & \\
 & & & & & & a & b & c/2 & d/2 & \cdot & & & & & \\
 & & & & & & & a & b/2 & c/2 & \cdot & \cdot & & & & \\
 & & & & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & & & \\
 & & & & & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\
 & & & & & & & & & & \cdot & \cdot & c/2 & d/2 & e & \\
 & & & & & & & & & & & \cdot & b/2 & c/2 & d & e \\
 & & & & & & & & & & & & a & b & c & d \\
 & & & & & & & & & & & & & a & b & c \\
\end{bmatrix} \tag{3.50}$$

$$= B + C, \tag{3.51}$$

where zero-valued matrix elements outside the band are omitted, and those within the band replaced with dots. In this example, we first take the original 16×16 matrix A, and create four 4×4 nonoverlapping submatrices whose diagonals lie along A's main diagonal. These submatrices don't quite cover all of A's nonzero elements, however, so we expand each submatrix (except the final one) from its bottom-right by two elements, making it a 6×6 matrix. The submatrices are no longer non-overlapping, so we take every second one and put it in a matrix B, every other one and put it in a matrix C such that B and C are both block diagonal, divide the elements they have in common by two so we don't double count them, and then declare that $A = B + C$. In the general case, the amount of overlap between the submatrices required to encompass all of A is equal to the bandwidth b of A (in this case $b = 2$ since A has two non-main diagonals on each side of its main diagonal).

Having split a term in the Hamiltonian into two terms, we can simply apply the split operator method as normal, with the same order accuracy in Δt as before. But now the matrices that we wish to exponentiate and have act on a vector are *block diagonal*. This

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

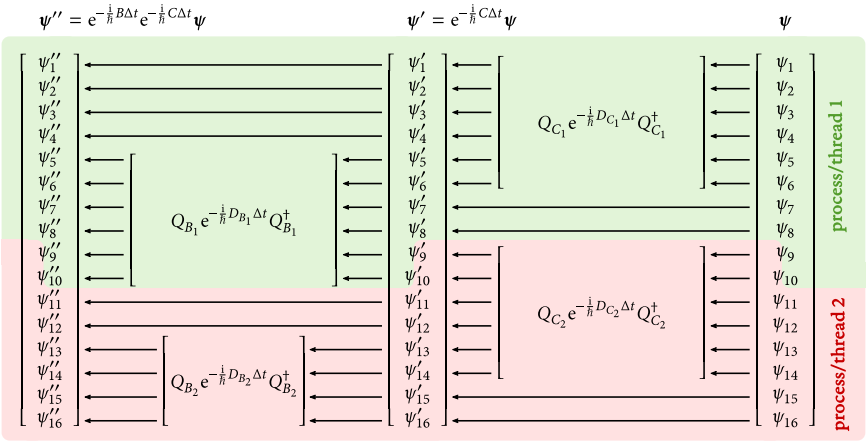

Figure 3.1: Schematic of data flow for parallel split-step method. Computation proceeds right to left. To compute the action of $e^{-\frac{i}{\hbar}B\Delta t}e^{-\frac{i}{\hbar}C\Delta t}$ on a vector ψ, one can treat the submatrices B_1, B_2, C_1 and C_2, of the block-diagonal matrices B and C separately. First, the exponentiation of C can be applied to ψ by applying the exponentiations of C_1 and C_2 to ψ. If diagonalisations $C_1 = Q_{C_1}D_{C_1}Q^\dagger_{C_1}$ and $C_2 = Q_{C_2}D_{C_2}Q^\dagger_{C_2}$ are known, then the two operations can be applied as a series of matrix-vector multiplications and the exponentiation of diagonal matrices. Because the two submatrices are non-overlapping, they can be applied to the vector completely independently in separate computer processes, CPU or GPU threads, or cluster nodes. The exponentiation of B can then be applied to the resulting intermediate vector ψ', similarly via two independent operarations acting on nonoverlapping elements of ψ'. However, because each submatrix of C overlaps each submatrix of B by $b = 2$ elements on either side (b being the bandwidth of the original matrix $A = B + C$), threads/processes must share these elements (here the ninth and tenth elements), sending them to each other whenever they have been updated.

means that the exponentiation of each block can be applied (using the diagonalisation method) separately to the vector, independently of each other block. This enables the application of the exponentials to be performed in parallel—each block submatrix modifies different elements of the vector. So one might store different parts of the state vector on different nodes on a cluster computer, and compute $e^{-\frac{i}{\hbar}C\Delta t}\psi$ in parallel. Then some data exchange between nodes would be neccesary before applying $e^{-\frac{i}{\hbar}B\Delta t}$ to the result (See Figure 3.1 for a schematic of how this works).

Whilst this specific banded matrix A is small for the sake of example, in general of course one might have a matrix of any size, allowing for B and C to contain more than two submatrices each. The submatrices have a minimum size of twice the bandwidth b of A, in order to cover all elements of A whilst only sharing elements with their nearest neighbor submatrices (any smaller and they would share elements with their next nearest neighbor submatrices as well, complicating things somewhat). But the only maximum is the size of A itself. So what is the optimal submatrix size? Although the split-step method is still acurate to the same order in Δt no matter how many pieces we split a banded matrix into, we clearly introduce additional 'commutation error' every time we split A into additional submatrices. So one might think that the number of pieces ought be be minimised, and hence the submatrix size maximimised. With regard to this, one might decide to split A into $2n_{\text{threads}}$ submatrices (where n_{threads} is the number of independent computational threads available), half of which will reside in B and half in C. This minimises the extra commutation error subject to the constraint that all threads are put to use—splitting A into yet smaller pieces within one computational thread will only yield unnecessary additional error.

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Is this the best option? No. Despite the extra commutation error, there is additional benefit to splitting A into more submatrices than required for parallelisation. Let s be the 'nominal' size of each submatrix, that is, the size of the corresponding *non-overlapping* submatrices prior to expanding each one along the diagonal (creating overlap) by a number of elements equal to the bandwidth b. The computational cost of the matrix-vector multiplications for computing the action of $e^{-\frac{i}{\hbar}B\Delta t}e^{-\frac{i}{\hbar}C\Delta t}$ on a vector is then $\mathcal{O}((s+b)^2)$ per submatrix, since $s+b$ is the size of the submatrices in terms of their nominal size and the bandwidth. The total number of submatrices required to cover A is ns^{-1}, where n is the size of A, and so the total cost of applying the exponentiations of all submatrices to a vector is $\mathcal{O}(ns^{-1}(s+b)^2)$. The cost *per unit time* of simulation is then $\mathcal{O}(n\Delta t^{-1}s^{-1}(s+b)^2)$. Here we see that splitting into smaller submatrices is desirable from the point of view of speed: because matrix-vector multiplication runs in quadratic time, a larger number of smaller matrices can be multiplied by vectors faster than a smaller number of larger matrices.

But the more submatrices, the more cummutation error. Extra error can be made up for by making the timestep smaller, which increases the cost per unit time once more. So is it worth it? The extra commutation error from splitting up A into more and more pieces in general depends on the form of A. I performed a small numerical experiment to see how the commutator $[B, C]$ (which the commutation error is proportional to) scales with s for random banded matrices, as well as those corresponding to 2nd, 4th and 6th order finite differences for first and second derivatives. The result in all cases was that the commutator scaled as $s^{-\frac{1}{2}}$ (see figure [FIG]).

[TODO FIG. point out why sum of squared elements is what makes sense for scaling.]

Back to our question—does decreasing the timestep size Δt to compensate for the additional cummutation error result in more, or less computational cost per unit time than if we hadn't split A into more pieces than required for parallelisation? Taking into account the nominal submatrix size s and the method's error in terms of Δt, the total error of integrating using the split-step method (assuming the $s^{-\frac{1}{2}}$ commutator scaling holds) is $\mathcal{O}(\Delta t^a s^{-\frac{1}{2}})$, where a is the order in Δt of the accuracy of the specific split-step method used (1, 2, or 4 for those I've discussed). Using these two pieces of information: the total error, and the total cost per unit time; we can now answer the question "What value of s minimises the computational cost per unit time, assuming constant error?".

For constant error we set $\Delta t^a s^{-\frac{1}{2}} \propto 1$ and get that the computational cost per unit time at constant error is $\mathcal{O}(ns^{-\frac{1}{2a}-1}(s+b)^2)$. For $a > \frac{1}{2}$, this expression has a minimum at $s = b$, which is the smallest possible submatrix size in any case. For our example banded

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

matrix A, decomposition into the smallest possible submatrices looks like this:

$$A = \begin{bmatrix}
c & d & e & & & & & & & & & & & & & & & & & \\
b & c & d & e & & & & & & & & & & & & & & & & \\
a & b & c & d & e & & & & & & & & & & & & & & & \\
 & a & b & c & d & e & & & & & & & & & & & & & & \\
 & & a & b & c & d & e & & & & & & & & & & & & & \\
 & & & a & b & c & d & e & & & & & & & & & & & & \\
 & & & & a & b & c & d & e & & & & & & & & & & & \\
 & & & & & a & b & c & d & e & & & & & & & & & & \\
 & & & & & & a & b & c & d & e & & & & & & & & & \\
 & & & & & & & a & b & c & d & e & & & & & & & & \\
 & & & & & & & & a & b & c & d & e & & & & & & & \\
 & & & & & & & & & a & b & c & d & e & & & & & & \\
 & & & & & & & & & & a & b & c & d & e & & & & & \\
 & & & & & & & & & & & a & b & c & d & e & & & & \\
 & & & & & & & & & & & & a & b & c & d & e & & & \\
 & & & & & & & & & & & & & a & b & c & d & e & & \\
 & & & & & & & & & & & & & & a & b & c & d & e & \\
 & & & & & & & & & & & & & & & a & b & c & d & \\
 & & & & & & & & & & & & & & & & a & b & c &
\end{bmatrix}$$

$$= \begin{bmatrix}
c & d & e & & & & & & & & & & & & & & & & & \\
b & c & d & e & & & & & & & & & & & & & & & & \\
a & b & c/2 & d/2 & \cdot & & & & & & & & & & & & & & & \\
 & a & b/2 & c/2 & \cdot & \cdot & & & & & & & & & & & & & & \\
 & & \cdot & \cdot & c/2 & d/2 & e & & & & & & & & & & & & & \\
 & & & \cdot & b/2 & c/2 & d & e & & & & & & & & & & & & \\
 & & & & a & b & c/2 & d/2 & \cdot & & & & & & & & & & & \\
 & & & & & a & b/2 & c/2 & \cdot & \cdot & & & & & & & & & & \\
 & & & & & & \cdot & \cdot & c/2 & d/2 & e & & & & & & & & & \\
 & & & & & & & \cdot & b/2 & c/2 & d & e & & & & & & & & \\
 & & & & & & & & a & b & c/2 & d/2 & \cdot & & & & & & & \\
 & & & & & & & & & a & b/2 & c/2 & \cdot & \cdot & & & & & & \\
 & & & & & & & & & & \cdot & \cdot & c/2 & d/2 & e & & & & & \\
 & & & & & & & & & & & \cdot & b/2 & c/2 & d & e & & & & \\
 & & & & & & & & & & & & a & b & c/2 & d/2 & \cdot & & & \\
 & & & & & & & & & & & & & a & b/2 & c/2 & \cdot & \cdot & & \\
 & & & & & & & & & & & & & & \cdot & \cdot & c/2 & d/2 & e & \\
 & & & & & & & & & & & & & & & \cdot & b/2 & c/2 & d & e \\
 & & & & & & & & & & & & & & & & a & b & c & d \\
 & & & & & & & & & & & & & & & & & a & b & c
\end{bmatrix} + \begin{bmatrix}
\cdot & \cdot & \cdot & & & & & & & & & & & & & & & & & \\
\cdot & \cdot & \cdot & \cdot & & & & & & & & & & & & & & & & \\
\cdot & \cdot & c/2 & d/2 & e & & & & & & & & & & & & & & & \\
 & \cdot & b/2 & c/2 & d & e & & & & & & & & & & & & & & \\
 & & a & b & c/2 & d/2 & \cdot & & & & & & & & & & & & & \\
 & & & a & b/2 & c/2 & \cdot & \cdot & & & & & & & & & & & & \\
 & & & & \cdot & \cdot & c/2 & d/2 & e & & & & & & & & & & & \\
 & & & & & \cdot & b/2 & c/2 & d & e & & & & & & & & & & \\
 & & & & & & a & b & c/2 & d/2 & \cdot & & & & & & & & & \\
 & & & & & & & a & b/2 & c/2 & \cdot & \cdot & & & & & & & & \\
 & & & & & & & & \cdot & \cdot & c/2 & d/2 & e & & & & & & & \\
 & & & & & & & & & \cdot & b/2 & c/2 & d & e & & & & & & \\
 & & & & & & & & & & a & b & c/2 & d/2 & \cdot & & & & & \\
 & & & & & & & & & & & a & b/2 & c/2 & \cdot & \cdot & & & & \\
 & & & & & & & & & & & & \cdot & \cdot & \cdot & \cdot & & & & \\
 & & & & & & & & & & & & & \cdot & \cdot & \cdot & & & &
\end{bmatrix}. \tag{3.52}$$

So the conlusion is: use the smallest submatrices possible when decomposing a banded matrix into a sum of two block-diagonal matrices. The decrease in computational costs, even when the increased error is compensated for by a smaller timestep, is worth it.

Limitations and nonlinearity

The split-step method is quite powerful and general. It allows you to approximately decompose the exponentiation of a Hamiltonian into exponentiations of its component terms, in the subspaces that they act on, and avoids exponentiating large banded matrices; saving on computing power immensely compared to exponentiating the full Hamiltonian. It is unitary (when the Hamiltonian is actually Hermitian), and stable—that is, unlike Runge–Kutta methods, the method's truncation error does not grow exponentially in time but is bounded (floating point rounding error still grows with time, but linearly) [TODO CONFIRM BASED ON FEDVR PAPER'S COMMENTS]. So there is no need to take much smaller timesteps than one otherwise would at earlier times in order to ensure the error remains acceptable at later times—an appropriate timestep at earlier times will be appropriate at later times, *ceteris paribus*. This is extremely appealing. And, whilst higher order split-step methods quickly become unwieldy [CITE], fourth order accuracy is quite acceptable for many problems.

Applying all the tricks described in the above sections results $\mathcal{O}(\Delta t)$, $\mathcal{O}(\Delta t^2)$ and $\mathcal{O}(\Delta t^4)$ accurate timestepping methods for solving the Schrodinger equation with total computational cost scaling as $\mathcal{O}(N \sum_i b_i)$, where $N = \prod_i n_i$ (for a product space of subspaces with dimensionalities $\{n_i\}$) is the dimensionality of the total Hilbert space, and $\{b_i\}$ are the bandwidths of the matrix representations of each term in the Hamiltonian in the chosen basis.[10] Furthermore, the method is efficiently parallelisable, provided the the maximum size-to-bandwidth ratio $\max_i(n_i/b_i)$ of the terms in the Hamiltonian is much larger than the number of parallel computing threads available. Although the constant factors that big-O notation neglects may not be optimal, this scaling would seem to be the best one could hope for—for each of the N elements in the state vector one must consider the $\sum_i b_i$ elements (including itself) that the Hamiltonian couples it with, and no more.[11]

[10] Now that we are here, we can finally say something about the best basis for simulating in: to minimise computational costs, the best basis is the one that minimises precisely this sum of bandwidths. The exception to this is when fast Fourier transforms are involved, which I discuss later.

[11] Assuming the banded matrices are otherwise dense within their band—further improvements would be possible if some elements within the band were always zero.

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

The main downside of this otherwise excellent method of exponentiating Hamiltonians is that the evolution modelled must actually be described by a linear system of equations. One cannot add arbitrary terms and nonlinear operators to the Hamiltonian, as the split-step method requires that one can evaluate each time-dependent term in the Hamiltonian at specific times, including times at which the solution for the state vector is not yet available. This would seem to limit the split-step method strictly to modelling *linear* dynamics, that is, terms in the Hamiltonian must not depend explicitly on the state vector they are operating on. Whilst nature might fundamentally be described by linear dynamics, once approximations of various kinds are made in order to make problems tractable, it's extremely common to end up with a nonlinear pseudopotential or nonlinear effective Hamiltonian. Note that non-*Hermitian*, pseudo-Hamiltonians—leading to non-unitary evolution—are fine. The split-step method has made no assumptions that rules them out, and can solve anything of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi(t) = \sum_n H_n(t)\psi(t), \tag{3.53}$$

where $\{H_n\}$ are a set of linear operators, Hermitian or not.

Fortunately, one specific form of nonlinearity that cold atom physicists are particularly interested in—the nonlinear term in the Gross–Pitaevskii equation—can be incorporated without much difficulty. As mentioned, the problem with nonlinearity is that all but the first-order split step methods require you to evaluate terms in the Hamiltonian at some future time at which the state vector is not yet known, that is the algorithm contains steps akin to $\psi(t_{n+1}) = U(t_{n+1}, t_n; \psi(t_{n+1}))\psi(t_n)$ for some nonlinear unitary matrix $U(t_{n+1}, t_n; \psi(t_{n+1}))$—$U$ cannot be explicitly constructed because it both requires and is required by $\psi(t_{n+1})$.

For the Gross–Pitaevskii effective Hamiltonian, the second-order split-step method (from which the fourth order method is constructed) for a single step might be naïvely written:

$$\psi(t_{n+1}) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}K\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}(g\rho(t_{n+1})+V(t_{n+1}))\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}(g\rho(t_n)+V(t_n))\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}K\frac{\Delta t}{2}}\psi(t_n), \tag{3.54}$$

where $\psi(t)$ is the state vector[12] represented in a discrete position basis, g is the nonlinear interaction constant, $V(t)$ and $\rho(t) = \psi(t)\psi^\dagger(t)$ are diagonal matrices for the external potential and the density matrix for $\psi(t)$ in the same position basis, and K is a discrete approximation to the kinetic energy operator in the position basis.

[12] Usually when modelling the GPE the single-particle state vector is normalised to the number of particles, rather than unity, and so strictly speaking it cannot be called a state vector, though it otherwise can be treated as one in most respects.

As written, this can't be evaluated because $\psi(t_{n+1})$—required to evaluate $\rho(t_{n+1})$—is not yet known. The order we choose to exponentiate the terms in our Hamiltonian is arbitrary, however (so long as we alternately reverse that order each half-step as required by the second order split-step method), and so swapping the order gives:

$$\psi(t_{n+1}) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}(g\rho(t_{n+1})+V(t_{n+1}))\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}K\Delta t}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}(g\rho(t_n)+V(t_n))\frac{\Delta t}{2}}\psi(t_n), \tag{3.55}$$

which incidentally has the benefit that since K is (ordinarily) time independent, the two adjacent exponentials containing it can be combined into one. Now $\rho(t_{n+1})$ is contained within the leftmost exponential, and so it is the last operator to be applied in the timestep. Note that since $g\rho(t)$ is real and diagonal in the position basis (as is $V(t)$), ths leftmost unitary merely changes the phase of the state vector at each point in space, having no effect on its density. This means that $\rho(t)$ is, in fact, unaffected by this last unitary evolution operator. Hence, $\rho(t_{n+1})$ can be computed simply as the density matrix of the intermediate state vector that this unitary was to act on:

$$\rho(t_{n+1}) = \psi(t_{n+1})\psi^\dagger(t_{n+1}) = \tilde{\psi}\tilde{\psi}^\dagger, \tag{3.56}$$

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

where

$$\psi(t_{n+1}) = e^{-\frac{i}{\hbar}(g\rho(t_{n+1})+V(t_{n+1}))\frac{\Delta t}{2}}\tilde{\psi}\tilde{\psi}^{\dagger}. \tag{3.57}$$

The inclusion of $V(t)$ with $g\rho(t)$ here is optional, and if $V(t)$ was not real valued, would not be valid (since in that case the density *would* be affected by the evolution induced by $V(t)$). In such a case the Hamiltonian would have to be split into three terms with $V(t)$ and $g\rho(t)$ treated separately.

So now we can put some conditions on what types of nonlinear operators can be used within the second-order split step method. The first condition is that at most one nonlinear operator can be included, since it must be placed last in the sandwich of exponentials (otherwise its value at the end of the timsestep cannot be inferred immediately prior to acting on the state vector). The second condition is that the nonlinear operator must be symmetric with respect to the evolution that it itself induces in the state vector. Here we have an operator that depends only on the state vector's absolute value, but for which the corresponding unitary only evolves the state vector's phase. Another example might be an operator that depends only on the state vector's phase gradients, but evolves the state vector's absolute value, and so forth.

Although I find this argument compelling for second-order split-step, it's less obvious that it should hold for fourth-order split-step as well, which, even though it is based on multiple applications of second-order split-step, involves a substep that is backwards in time. 'Evaluate the nonlinear operator based on the state vector at this specific time' becomes ambiguous when that moment in time is traversed in both directions by two different sub-steps. However, [Blah Blah et al] have verified using computer symbolic algebra that indeed, up to even higher order split-step methods, putting the nonlinear density term last in the splitting and always evaluating it using the value of the intermediate state vector immediately prior results in the method having the same order accuracy in Δt as for linear operators only. Given this and my argument above, as well as the reasoning that the second-order split-step method has no way of 'knowing' whether it is acting backward in time or not when embedded in a higher-order split step method, I would expect the same to hold for all nonlinear operators meeting the above two conditions, though I haven't shown this explicitly.

3.3 For everything else, there's fourth-order Runge–Kutta

- absorbing boundary conditions, reflective boundary conditions, periodic boundary conditions

3.4 Continuous degrees of freedom

The single-particle, non-relativistic, scalar Schrödinger wave equation, as distinct from the general Schrödinger equation [EQREF], is:

[EQUATION].

As mentioned in [SECREF], the equation for the single-particle wavefunction of an atom in a single-component Bose–Einstein condensate is the Gross–Pitaevskii equation

[EQUATION],

where $\psi(\boldsymbol{r}) = \sqrt{N}\langle \boldsymbol{r}|\psi\rangle$ is the single-particle wavefunction scaled by the square root number of atoms N.

Both these equations are partial differential equations involving both spatial and temporal derivatives. But in quantum mechanics all state vectors can be mapped to column vectors and all operators to matrices. Spatial wavefunctions are no exception to the former and differential operators such as ∇^2 are no exception to the latter. So

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

what do these vectors and operators look like? That depends on whether we choose to discretise space on a grid, or use a functional basis (and on which functional basis we choose). As we'll see below, however, spatial discretisation is actually just a particular choice of functional basis, namely the Fourier basis.

3.4.1 Spatial discretisation

Imagine a two dimensional spatial region within which we are solving the single-component Gross–Pitaevskii equation, evolving an initial condensate wavefunction in time. Having specified which degrees of freedom we want to simulate (two continuous degrees of freedom, one for each spatial dimension), the next step according to the method outlined in [SECREF ABOVE] is to choose a basis in which to represent this state vector.

Lets say we discretise space in an equally-spaced $N_x \times N_y = 5 \times 5$ rectangular grid,[13] with spacing Δx, and only represent the wavefunction at those 25 points in space. The state vector can then be represented by a list of 25 complex numbers, each taken to be the wavefunction's value at the spatial position corresponding to one gridpoint. This 25-vector is now a concrete representation of our state vector.

[13] For the sake of example—256 × 256 is a more realistic minimum.

[FIGURE SHOWING UNWRAPPING OF 2D REGION INTO COLUMN VECTOR. PERHAPS 2D REGION IS CONTINUOUS WITH A GRID SUPERIMPOSED ON IT, AND LINES JOIN EACH GRIDPOINT TO THE VECTOR AT RIGHT]

But at what point did we choose a basis just now—what are the basis vectors? This just looks like discretising space at a certain resolution, rather than the formal process of choosing a basis and projecting the state vector and operators onto each basis vector, as outlined in [SECREF ABOVE]. Assuming what we've done is equivalent to choosing a basis, that basis has a finite number (25) of basis vectors, which means it cannot be complete, since state vectors we're approximately representing with it require an infinite number of complex numbers to be described exactly.[14] So what do the basis functions look like, and what state vectors have we implicitly excluded from simulation by choosing a basis that is incomplete?

[14] One for each position within the two dimensional space we're representing.

As a sidenote, spatial wavefunctions are often described as the representation of wavefuctions in the "spatial basis"—a basis in which the basis vectors $\{|\boldsymbol{r}\rangle\}$ are Dirac-deltas centred on each point in space [CITE DIRAC'S BOOK MAYBE]. The wavefunction $\psi(\boldsymbol{r})$ is then simply a coefficient saying how much of the basis vector $|\boldsymbol{r}\rangle$ (the spatial representation of which is a Dirac delta centred on the position $\boldsymbol{r}$) to include in the overall state vector. What we have *not* done is chosen a subset of these Dirac delta basis functions[15] as our basis. This would be very strange—our representation of the wavefunction would allow it to have a value at one gridpoint, and at the next gridpoint, but not in between. Spatially separated Dirac deltas do not spatially overlap at all; the matrix elements of the kinetic energy operator:

[15] Strictly, distribtuions, not functions, but "basis distributions" just doesn't have the right ring to it.

$$\langle \boldsymbol{r}_i|\hat{K}|\boldsymbol{r}_j\rangle = \int \delta(\boldsymbol{r}-\boldsymbol{r}_i)\left(-\frac{\hbar^2}{2m}\nabla^2\right)\delta(\boldsymbol{r}-\boldsymbol{r}_j)\,\mathrm{d}\boldsymbol{r} \tag{3.58}$$

would all be zero for $i \neq j$, disallowing any flow of amplitude from one point in space to another by virtue of it not being able to pass through the intervening points. Neither have we chosen a set of two-dimenstional boxcar functions centred on each gridpoint with width Δx in each direction. These cover all space in between gridpoints, but are no good because they are not twice differentiable everywhere, and hence the kinetic energy operator's matrix elements cannot be evaluated. No, neither of these bases will do. To interpret our spatial grid as a basis, we need a set of functions $[FUNC_{ij}(r)]$ (where i and j are the indices of the gridpoints in the x and y directions respectively) that have unit norm, are nonzero only at one gridpoint and are zero at all others, and are twice

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

differentiable everywhere in our spatial region. Infinite choices present themselves to us, differing only in their incompleteness—the choice of which state vectors they will and won't be able to represent. A sensible choice is that we want to be able to represent the state vectors whose wavefunctions do not change much between adjacent gridpoints, and we are happy for the necessary incompleteness of our basis to exclude wavefunctions with any sort of structure in between gridpoints.

The discrete Fourier transform to the rescue

It turns out that discretising space in this way can indeed be equivalent to choosing an entirely sensible basis. This is made clearer by thinking in terms of what we have done in Fourier space, which I'll quickly go through now.

One possible basis for representing all possible state vectors is the Fourier basis $\{|\boldsymbol{k}_{i,j}\rangle\}$. With it, any state vector (whose wavefunction is nonzero only within the $2D$ region)can be represented as the sum of basis vectors whose wavefunctions are 2D plane waves, also localised to the 2D region:

$$\langle \boldsymbol{r}|\boldsymbol{k}_{i,j}\rangle = \begin{cases} \frac{1}{\sqrt{A}} e^{i\boldsymbol{k}_{i,j}\cdot\boldsymbol{r}} & (\boldsymbol{r} \text{ within 2D region}) \\ 0 & (\boldsymbol{r} \text{ not within 2D region}), \end{cases} \tag{3.59}$$

where A is the area of the $2D$ region and the wavevector of each plane wave is

$$\boldsymbol{k}_{i,j} = \left[\frac{2\pi i}{L_x}, \frac{2\pi j}{L_y}\right]^{\mathrm{T}}, \tag{3.60}$$

where i and j are (possibly negative) integers. Any state vector localised to the 2D region can then be written as the infinite sum:

$$|\psi\rangle = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \langle \boldsymbol{k}_{i,j}|\psi\rangle \; |\boldsymbol{k}_{i,j}\rangle \tag{3.61}$$

$$\Rightarrow \psi(\boldsymbol{r}) = \begin{cases} \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \langle \boldsymbol{k}_{i,j}|\psi\rangle \frac{1}{\sqrt{A}} e^{i\boldsymbol{k}_{i,j}\cdot\boldsymbol{r}} & (\boldsymbol{r} \text{ within 2D region}) \\ 0 & (\boldsymbol{r} \text{ not within 2D region}). \end{cases} \tag{3.62}$$

So $\{\langle \boldsymbol{k}_{i,j}|\psi\rangle\}$ are simply the coefficients of the 2D Fourier series of $\psi(\boldsymbol{r})$.

What does this have to do with our discretised space? These basis functions $\{\langle \boldsymbol{r}|\boldsymbol{k}_{i,j}\rangle\}$ don't have the required properties for a discrete basis. For one, there are an infinite number of them, and we require 25. Secondly, all of them are nonzero everywhere within the 2D region, whereas we require each basis function to be nonzero at exactly one of out 25 gridpoints.

We can solve the first problem by truncating the Fourier series. By only including basis vectors $|\boldsymbol{k}_{i,j}\rangle$ for which:

$$\begin{cases} i \in [-\frac{N_x}{2}, \frac{N_x}{2} - 1] & (N_x \text{ even}) \\ i \in [-\frac{N_x-1}{2}, \frac{N_x-1}{2}] & (N_x \text{ odd}) \end{cases} \tag{3.63}$$

and

$$\begin{cases} j \in [-\frac{N_y}{2}, \frac{N_y}{2} - 1] & (N_y \text{ even}) \\ j \in [-\frac{N_y-1}{2}, \frac{N_y-1}{2}] & (N_y \text{ odd}) \end{cases} \tag{3.64}$$

we include only the Nx and N_y (both equal to 5 in our example) longest wavelengths in each respective spatial dimension. This is a sensible truncation with a physically meaningful interpretation. By making it, we are no longer able to represent state vectors with

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

short wavelength components. Because the kinetic energy operator, when represented in the Fourier basis, is:

$$\langle \boldsymbol{k}_{i,j}|\hat{K}|\boldsymbol{k}_{i'j'}\rangle = \frac{\hbar^2 k^2}{2m}\delta_{ii'}\delta_{jj'}, \tag{3.65}$$

where $k = |\boldsymbol{k}_i|$, by excluding basis vectors with larger wavevectors, we are excluding state vectors with large kinetic energy. Thus the truncation is a kinetic energy cutoff, and is an accurate approximation whenever a simulation is such that the system is unlikely to obtain kinetic energies above the cutoff.[16]

[16] Because a *square* region in Fourier space is being carved out, by limiting each of k_x and k_y to finite ranges rather than the total wavenumber $k = \sqrt{k_x^2 + k_y^2}$, there is no single kinetic energy cutoff so to speak. Nonetheless there is a maximum wavenumber $k_{\text{max}} = \min(\{|k_x|\} \cup \{|k_y|\})$ defining a kinetic energy cutoff $K_{\text{max}} = \hbar^2 k_{\text{max}}^2/(2m)$ below which kinetic energies definitely are representable and above which they may not be.

Now we have 25 basis vectors—a discrete Fourier basis—but their spatial wavefunctions still don't have the property of being nonzero only at a single gridpoint each. On the contrary, each plane wave has has a constant amplitude everywhere in space. But consider the following superposition of Fourier basis vectors:

$$|\boldsymbol{r}_{i,j}\rangle = \sum_{i',j'} e^{-i\boldsymbol{k}_{i',j'}\cdot\boldsymbol{r}_{i,j}}\,|\boldsymbol{k}_{i',j'}\rangle \tag{3.66}$$

with $\boldsymbol{r}_{i,j} = (i\Delta x, j\Delta y)^{\mathrm{T}}$ and $i,j \in [0,4]$. This is simply a unitary transformation of the truncated Fourier basis, with the unitary transformation matrix elements:

$$\hat{U}_{\text{DFT2};i',j';i,j} = \langle \boldsymbol{k}_{i',j'}|\boldsymbol{r}_{i,j}\rangle = e^{-i\boldsymbol{k}_{i',j'}\cdot\boldsymbol{r}_{i,j}}. \tag{3.67}$$

This transformation is in fact a discrete Fourier transform (hence the subscript), and the basis vectors $\{|\boldsymbol{r}_{i,j}\rangle\}$ have spatially localised wavefunctions that are nonzero only at one of the spatial gridpoints, so we refer to them as a discrete real-space basis. State vectors and operators can be transformed from their discrete Fourier space representation to their discrete real-space representation and back using the unitary $\hat{U}_{\text{DFT2}}$:

$$\langle \boldsymbol{r}_{i,j}|\psi\rangle = \sum_{i',j'} \hat{U}^{\dagger}_{\text{DFT2};i,j;i',j'}\,\langle \boldsymbol{k}_{i',j'}|\psi\rangle \tag{3.68}$$

$$\langle \boldsymbol{k}_{i,j}|\psi\rangle = \sum_{i',j'} \hat{U}_{\text{DFT2};i',j';i,j}\,\langle \boldsymbol{r}_{i',j'}|\psi\rangle \tag{3.69}$$

[TODO OPERATOR TRANSFORMATIONS ONCE INDEXING IMPROVED AND VECTOR REPRESENTATIONS INTRODUCED]

The spatial representation of the basis vectors $\{|\boldsymbol{r}_{i,j}\rangle\}$ can be computed using (3.59) as:

$$\phi_{i,j}(\boldsymbol{r}) = \langle \boldsymbol{r}|\boldsymbol{r}_{i,j}\rangle = \sum_{i',j'} e^{-i\boldsymbol{k}_{i',j'}\cdot\boldsymbol{r}_{i,j}}\,\langle \boldsymbol{r}|\boldsymbol{k}_{i',j'}\rangle \tag{3.70}$$

$$\Rightarrow \phi_{i,j}(\boldsymbol{r}) = \begin{cases} \sum_{i',j'} \frac{1}{\sqrt{A}} e^{i\boldsymbol{k}_{i',j'}\cdot(\boldsymbol{r}-\boldsymbol{r}_{i,j})} & (\boldsymbol{r} \text{ within 2D region}) \\ 0 & (\boldsymbol{r} \text{ not within 2D region}), \end{cases} \tag{3.71}$$

and are plotted in [TODO MAKE FIGURE].

These functions are sometimes called *periodic sinc functions*, or *band-limited delta functions* [CITE]. Each of them is zero at all of our gridpoints except one, and and they form an orthonormal basis set. They satisfy all of our requirements to be a basis corresponding to our gridded discretisation of space. Thus, the approach of discretising space on a grid is indeed equivalent to choosing a (necessarily incomplete) orthonormal basis in which to work.

One thing to note is that these functions are periodic. By using the Fourier basis in the way we have to restrict our basis to cover only a finite region of both Fourier space and real space, we have necessarily imposed periodicity on the problem. This periodicity shows

rev: 37 (d1fcd3820aa1)
author: Chris Billington <chrisjbillington@gmail.com>
date: Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.

itself when we compute matrix elements of operators in this basis - if we compute the kinetic energy operator's matrix elements for example, it will couple basis states across the boundary of the region, resulting in spatial periodicity—a wavepacket moving rightward through the right boundary will emerge moving rightward from the left boundary.

Perhaps less obviously, the basis is also periodic in Fourier space, and so a wavepacket moving out of the region of Fourier space simulated will also wrap around to the opposite side of Fourier space. In real space, this may appear as a wavepacket undergoing acceleration only to suddenly reverse its velocity as if reflecting off a barrier [SEE FIG TODO FOR EXAMPLE]. This effect is entirely unphysical[17] and should be taken as a sign that the spatial grid is not fine enough for the dynamics being simulated.

[17] With the possible exception of the region of Fourier space being simulated corresponding to the first Brillouin zone of a lattice potential, in which case these velocity reversals are Bloch oscillations.

[18] Or rather, a state vector's projection onto the subspace being simulated.

Matrix elements using DFT basis

We now have a finite basis $\{|\boldsymbol{r}_{i,j}\rangle\}$ that matches our intuitions somewhat for representing a wavefunction at a set of gridpoints. A state vector[18] can be represented as a linear sum of these basis vectors, with the coefficient for each one simply being equal to the value of that state vector's wavefunction at the gridpoint where that basis vector is nonzero:

$$|\psi\rangle = \sum_{i,j} c_{i,j} |\boldsymbol{r}_{i,j}\rangle \,, \tag{3.72}$$

where

$$c_{i,j} = \langle \boldsymbol{r}_{i,j} | \psi \rangle = \psi(\boldsymbol{r}_{i,j}). \tag{3.73}$$

Armed with a basis, we can now proceed to calculate matrix elements of the Hamiltonian, and then proceed to solve the differential equation [TODO EQREF 3.3] to determine how the coefficients $\{c_{i,j}\}$ evolve in time.

The specific properties of our basis make it quite useful for a range of common Hamiltonians. For example, let's take the single particle Schrödinger Hamiltonian:

$$\hat{H}_{\text{Schrö}} = -\frac{\hbar^2 \hat{k}^2}{2m} + V(\hat{\boldsymbol{r}}), \tag{3.74}$$

where $\hat{k} = |\hat{\boldsymbol{k}}|$.

The two terms, kinetic and potential, are each diagonal in different bases. The kinetic term is diagonal in the Fourier basis:

$$\langle \boldsymbol{k}' | -\frac{\hbar^2 \hat{k}^2}{2m} | \boldsymbol{k} \rangle = -\frac{\hbar^2 k^2}{2m} \delta(\boldsymbol{k} - \boldsymbol{k}'), \tag{3.75}$$

where $k = |\boldsymbol{k}|$, and the potential term is diagonal in the spatial basis:

$$\langle \boldsymbol{r}' | V(\hat{\boldsymbol{r}}) | \boldsymbol{r} \rangle = V(\boldsymbol{r}) \delta(\boldsymbol{r} - \boldsymbol{r}'). \tag{3.76}$$

Equivalent relations hold for our discrete Fourier and spatial basis:

$$\langle \boldsymbol{k}_{i'j'} | -\frac{\hbar^2 \hat{k}^2}{2m} | \boldsymbol{k}_{i,j} \rangle = -\frac{\hbar^2 k_{i,j}^2}{2m} \delta_{i'i} \delta_{j'j}, \tag{3.77}$$

$$\langle \boldsymbol{r}_{i'j'} | V(\hat{\boldsymbol{r}}) | \boldsymbol{r}_{i,j} \rangle = V(\boldsymbol{r}_{i,j}) \delta_{i'i} \delta_{j'j}. \tag{3.78}$$

But to obtain our differential equation, we need all the matrix elements of $\hat{H}_{\text{Schrö}}$ in a single basis.

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

3.5 Discrete degrees of freedom

3.5.1 The interaction picture

A common situation in atomic physics is to be simulating the internal state of an atom, armed with the knowledge that only a small number of atomic states are able to become occupied.

- Sometimes called a "rotating frame"
- Is equivalent to basis change where new basis functions differ by a time-dependent phase factor
- Is defined by a time-independent Hamiltonian
- This has the effect of moving some time dependence into the operators (demonstrate, by writing some operators with the unitary in front of them. As you can see it is simply a change of basis - but a time-dependent one.)
- No need to remain in the same interaction picture - can be redefined arbitrarily often throughout a simulation and state vectors transformed into new basis.

3.5.2 Unitary integration

Direct exponentiation via diagonalisation of Hamiltonian

- Unitary - doesn't mean it's accurate but means it won't explode. Great for the bits of your simulation that are explosion-prone but don't matter (like regions of space where the wavefunction is near zero but the potential is large or steep)
- error is order [whatever it is] per timestep, not great compared to RK4
- can be combined with RK4 to improve accuracy (see later subsection)

Approximate exponentiation by operator product

[Comment in this section how the approximate total unitary can be used at each timestep to define an interaction picture, and the remaining dynamics simulated with RK4 like RKILIP does in the spatial basis. Interaction pictures are really useful!]

3.6 Continuous degrees of freedom

Every symbol on the paper has a representation in a computer. State vectors are arrays of complex numbers, operators are matrices - differential operators are no exception. Operators must have a concrete representation, their matrix elements can be computed and then things solved with linear algebra. For discrete degrees of freedom, the matrix representation of the operators may be known, for continuous ones you can find the matrix elements once you define what basis functions you will use [show how]. Or, for the DVR it is a little more subtle (because it's not a basis) but still basically the same process.

- Have to be discretised in some way to simulate on a computer - need basis functions. Often a spatial basis is used. Any spatial basis must be combined with assumptions about what the wavefunction is doing at points in between the basis points, in order to define differential operators. Finite differences approximates wavefunction as low-order polynomial in between points (is this equivalent to a polynomial *basis*?

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Probably not.). Fourier method assume the Fourier series of the wavefunction at the given points can be used to interpolate between points (or the wavefuntion can be Fourier transformed and calculations can be done directly in the Fourier basis). DVR is not actually a spatial basis despite appearances. It assumes the wavefunction is a sum of polynomial 'shape functions', but these shape functions are not basis functions as they are not orthonormal. This is why it is called a representation rather than a basis. Regardless, the shape functions can be used to define an interpolation of the wavefunction between points and thus define differential operators.

3.6.1 Finite differences

Show a matrix representation of a few different finite differences, to show that differential operators really are just matrices. They approximate the function as low-order polynomials about each point. You can take them to arbitrarily high order.

3.6.2 The Fourier basis

Because of properties of Fourier transforms, derivitives can be taken in Fourier space as simple multiplication. This is essentially because differential operators are diagonal in the Fourier basis. So you can use this fact to define a differential operator in the spatial basis [show matrix] ...or, you could just implement it with Fourier transforms, since FFTs are faster than matrix-vector multiplication ($O(n \log(n))$ rather than $O(n^2)$)

Split operator method

- Equivalent to approximate exponentiation via operator product with the discrete case

3.6.3 Harmonic oscillator basis functions

3.7 Finding ground states

3.7.1 Imaginary time evolution

3.7.2 Successive over-relaxation

3.7.3 Generalisation to excited states via Gram–Schmidt orthonormalisation

Directly diagonalising a Hamiltonian can be costly in a spatial basis. Another approach is to find the ground state using one of the above techniques, and then repeat the process, subtracting off the wavefunction's projection onto the already found ground state at every step. This yields the lowest energy state that is orthogonal to the first - i.e. the first excited state. Repeating the process, but subtracting off *both* eigenstates found so far, then yields the second excited state and so forth. This is simply the Gram-Schmidt process for finding orthonormal vectors, with the additional step of relaxing each vector to the lowest possible energy for each one - this ensures the eigenstates of the Hamiltonian are produced, rather than a different orthogonal basis. Extra conditions can be imposed on the wavefunction at each relaxation step in order to obtain particular solutions in the case of degenerate eigenstates. For example, a phase winding can be imposed in order to obtain a particular harmonic oscillator state - otherwise this process produces an arbitrary superposition of basis states that have equal energy.

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

3.8 The finite-element discrete variable representation

- Plots of representation of sine wave as function of number of points between FEDVR and FD. RMS error of a derivative operator perhaps.

[explanation of how it works, comparison of implementation with RSP4 vs something like RK4. RK4 is more general purpose, method does not need to be modified for different Hamiltonians. Main limitation is inability to factor out fast dynamical phases, see RK4ILIP for solution to this. MPI implementation and scaling properties with increasing cores. Superscaling at low number of cores. Mention how my implementation can tolerate high network latency due to early sending od data before all local computations are complete. Mention that it is ripe for GPU processing. Limitations: vulnerable to Runge's phenomenon for sharp potentials. Can't increase the order of the polynomials much because small spacing at the edges requires tiny timesteps. Possible solution: preconditioning the potential to be an approximation better representable in the DVR basis.]


```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

This page intentionally left blank

References

[1] L. M. Bennie, P. B. Wigley, S. S. Szigeti, M. Jasperse, J. J. Hope, L. D. Turner, and R. P. Anderson. *Precise wavefunction engineering with magnetic resonance*. arXiv:1412.6854 [cond-mat, physics:quant-ph] (2014). [p 1]

[2] A. Görlitz, J. M. Vogels, A. E. Leanhardt, C. Raman, T. L. Gustavson, J. R. Abo-Shaeer, A. P. Chikkatur, S. Gupta, S. Inouye, T. Rosenband, and W. Ketterle. *Realization of Bose-Einstein Condensates in Lower Dimensions*. Physical Review Letters **87**, 130402 (2001). DOI: 10.1103/PhysRevLett.87.130402. [p 2]

[3] S. Hofferberth, I. Lesanovsky, B. Fischer, T. Schumm, and J. Schmiedmayer. *Non-equilibrium coherence dynamics in one-dimensional Bose gases*. Nature **449**, 324 (2007). DOI: 10.1038/nature06149. [p 2]

[4] B. Rauer, P. Grišins, I.E. Mazets, T. Schweigler, W. Rohringer, R. Geiger, T. Langen, and J. Schmiedmayer. *Cooling of a One-Dimensional Bose Gas*. Physical Review Letters **116**, 030402 (2016). DOI: 10.1103/PhysRevLett.116.030402. [p 2]

[5] Paul D. Lett, Richard N. Watts, Christoph I. Westbrook, William D. Phillips, Phillip L. Gould, and Harold J. Metcalf. *Observation of Atoms Laser Cooled below the Doppler Limit*. Physical Review Letters **61**, 169 (1988). DOI: 10.1103/PhysRevLett.61.169. [p 2]

[6] J. Dalibard and C. Cohen-Tannoudji. *Laser cooling below the Doppler limit by polarization gradients: simple theoretical models*. JOSA B **6**, 2023 (1989). DOI: 10.1364/JOSAB.6.002023. [p 2]

[7] M. Saffman, T. G. Walker, and K. Mølmer. *Quantum information with Rydberg atoms*. Reviews of Modern Physics **82**, 2313 (2010). DOI: 10.1103/RevModPhys.82.2313. [p 2]

[8] E. Urban, T. A. Johnson, T. Henage, L. Isenhower, D. D. Yavuz, T. G. Walker, and M. Saffman. *Observation of Rydberg blockade between two atoms*. Nature Physics **5**, 110 (2009). DOI: 10.1038/nphys1178. [p 2]

[9] B. B. Blinov, Jr R. N. Kohn, M. J. Madsen, P. Maunz, D. L. Moehring, and C. Monroe. *Broadband laser cooling of trapped atoms with ultrafast pulses*. JOSA B **23**, 1170 (2006). DOI: 10.1364/JOSAB.23.001170. [p 2]

[10] A. J. McCulloch, D. V. Sheludko, M. Junker, and R. E. Scholten. *High-coherence picosecond electron bunches from cold atoms*. Nature Communications **4**, 1692 (2013). DOI: 10.1038/ncomms2699. [p 2]


```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

[11] Thomas Brabec and Ferenc Krausz. *Intense few-cycle laser fields: Frontiers of nonlinear optics*. Reviews of Modern Physics **72**, 545 (2000). DOI: 10.1103/RevModPhys.72.545. [p 2]

[12] C. Moler and C. Van Loan. *Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later*. SIAM Review **45**, 3 (2003). DOI: 10.1137/S00361445024180. [p 4]

[13] Stefan Weinzierl. *Introduction to Monte Carlo methods*. arXiv:hep-ph/0006269 (2000). [p 6]

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```

Word count

```
Total
Words in text: 25381
Words in headers: 300
Words outside text (captions, etc.): 3587
Number of headers: 76
Number of floats/tables/figures: 27
Number of math inlines: 608
Number of math displayed: 109
Files: 9
Subcounts:
  text+headers+captions (#headers/#floats/#inlines/#displayed)
  1753+28+325 (9/2/26/8) File(s) total: atomic_physics.tex
  725+17+385 (4/4/7/0) File(s) total: experiment.tex
  50+0+0 (0/0/0/0) File(s) total: front_matter.tex
  1370+19+65 (3/2/39/13) File(s) total: hidden_variables.tex
  2043+7+145 (3/1/4/0) File(s) total: introduction.tex
  12252+161+1217 (36/3/454/84) File(s) total: numerics.tex
  4435+33+760 (11/9/1/0) File(s) total: software.tex
  2717+22+690 (7/6/77/4) File(s) total: velocimetry.tex
  36+13+0 (3/0/0/0) File(s) total: wave_mixing.tex
```

```
rev:     37 (d1fcd3820aa1)
author:  Chris Billington <chrisjbillington@gmail.com>
date:    Mon Jan 09 12:17:44 2017 -0500
summary: Draft of split-step methods completed.
```