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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

This page intentionally left blank

Contents

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

This page intentionally left blank

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

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 3.6 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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:

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

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

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

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

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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

other methods of computing matrix exponentials that are more computationally efficient and numerically stable, such as the Padé approximant [12]. It should be noted that there is no known 'best' matrix exponential algorithm, all make compromises and perform poorly for certain types of matrices [13].

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

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

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

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

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

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:

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

[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.

[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.

```python
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) = Q exp(D) Q^\dagger where Q is the matrix of
    # eigenvectors (as columns) and D is the diagonal matrix of eigenvalues:

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

    # Compute the 3-term matrix product Q*exp_D_diags*Q_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', Q, exp_D_diags, Q_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 $\mathcal{O}(n^3)$ [13]. 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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

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

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

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

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 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 (we will see shortly why this is). If it does—that is, if $[\hat{H}(t_1'), \hat{H}(t_2')] = 0$ for all $t_1', t_2' \in [t_0, t]$, then (3.5) is the solution to the Schrdinger equation, and once represented in a specific basis can be written

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

with

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

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* [14, p. 193]:

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

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:

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

We see now why the time-ordering can be neglected when $H(t)$ commutes with itself at all times. When it does, all possible reorderings of a product of copies $H(t)$ are already equal, and so a time-ordering operator leaves the actual value of the product unchanged.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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 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:

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

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 [16]. 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 [16, 17] 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 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 in the Dyson series corresponds to a number of Feynman diagrams with n nodes [17].

[6] By sampling the integrand on a uniform grid, using a quadrature method, or Monte-Carlo integration [15] 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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 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.

[8] Although unitarity is not often a strict requirement - we also frequently solve (3.3) directly with fourth order Runge–Kutta, which is also not unitary.

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

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

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

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[9]. 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.

[9] This is the usual definition of the solution to a differential equation, which is why I prefer to think of the product formula (3.23) as the *definition* of the solution to the Schrödinger equation, and the time-ordered exponential as merely a shorthand notation for it.

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

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

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

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 review 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 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 3.4, we'll see how this method applies to the case of a spatial wavefunction obeying the Gross–Pitaevskii equation.

First order split-step

[10] 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.

Say we have (the matrix representation of) a Hamiltonian that is the sum of two non-commuting terms:[10]

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

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 $t_n = t_0 + n\Delta t$ and $\Delta t = (t - t_0)/N$, 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

Using the Baker–Campbell–Hausdorff formula [14, p. 158], we can expand the exponential as:

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

$$\Rightarrow U(t_{n+1},t_n) = e^{-\frac{i}{\hbar}H_1(t_n)\Delta t}e^{-\frac{i}{\hbar}H_2(t_n)\Delta t} + \mathcal{O}(\Delta t^2). \tag{3.35}$$

So we see that the 'commutation error' in (3.35) caused by treating the two terms as if they commute is of the same order in Δt as 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:

$$U(t,t_0) = \prod_{n=N-1}^{0} U_1(t_{n+1},t_n) + \mathcal{O}(\Delta t), \tag{3.36}$$

where

$$U_1(t_{n+1},t_n) = e^{-\frac{i}{\hbar}H_1(t_n)\Delta t}e^{-\frac{i}{\hbar}H_2(t_n)\Delta t}. \tag{3.37}$$

Using the diagonalisation method of matrix exponentiation discussed in section 3.2.1, 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:

$$U_1(t_{n+1},t_n) = \left(e^{-\frac{i}{\hbar}\tilde{H}_1(t_n)\Delta t} \otimes \mathbb{I}_{n_2}\right)\left(\mathbb{I}_{n_1} \otimes e^{-\frac{i}{\hbar}\tilde{H}_2(t_n)\Delta t}\right). \tag{3.38}$$

By applying (3.35) recursively for the case where either H_1 or H_2 is itself the sum of two further terms, (3.37) immediately generalises to the case where H is a sum of an arbitrary number of non-commuting terms:

$$U_1(t_{n+1},t_n) = \prod_{m=1}^{M} e^{-\frac{i}{\hbar}H_m(t_n)\Delta t}, \tag{3.39}$$

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:

$$U(t_{n+1},t_n) = e^{-\frac{i}{\hbar}(H(t_n)+H(t_{n+1}))\frac{\Delta t}{2}} + \mathcal{O}(\Delta t^3) \tag{3.40}$$

$$= e^{-\frac{i}{\hbar}(H_1(t_n)+H_2(t_n)+H_1(t_{n+1})+H_2(t_{n+1}))\frac{\Delta t}{2}} + \mathcal{O}(\Delta t^3). \tag{3.41}$$

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

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

$H_1(t_n)+\dot{H}_1(t_n)\Delta t+\mathcal{O}(\Delta t^2)$, one can show that if the individual exponentials are ordered in the following way, then the remaining commutation error is also $\mathcal{O}(\Delta t^3)$:

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

This gives the second order split-step method:

$$U(t,t_0) = \prod_{n=N-1}^{0} U_2(t_{n+1},t_n) + \mathcal{O}(\Delta t^2), \tag{3.43}$$

where

$$U_2(t_{n+1},t_n) = \mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_2(t_{n+1})\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_1(t_{n+1})\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_1(t_n)\frac{\Delta t}{2}}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_2(t_n)\frac{\Delta t}{2}}, \tag{3.44}$$

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

$$U_2(t_{n+1},t_n) = \left(\prod_{m=M}^{1}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_m(t_{n+1})\frac{\Delta t}{2}}\right)\left(\prod_{m=1}^{M}\mathrm{e}^{-\frac{\mathrm{i}}{\hbar}H_m(t_n)\frac{\Delta t}{2}}\right). \tag{3.45}$$

U_2 can be concisely written in terms of U_1 as:

$$U_2(t_{n+1},t_n) = U_1^\dagger(t_n+\tfrac{\Delta t}{2},t_{n+1})U_1(t_n+\tfrac{\Delta t}{2},t_n), \tag{3.46}$$

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 (3.44) 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 [18, p7; 19,20]:

$$U(t,t_0) = \prod_{n=N-1}^{0} U_4(t_{n+1},t_n) + \mathcal{O}(\Delta t^4), \tag{3.47}$$

where

$$\begin{aligned} U_4(t_{n+1},t_n) = \; & U_2(t_{n+1},t_{n+1}-p\Delta t) \\ & \times U_2(t_{n+1}-p\Delta t,t_{n+1}-2p\Delta t) \\ & \times U_2(t_{n+1}-2p\Delta t,t_n+2p\Delta t) \\ & \times U_2(t_n+2p\Delta t,t_n+p\Delta t) \\ & \times U_2(t_n+p\Delta t,t_n), \end{aligned} \tag{3.48}$$

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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

Parallelisability and other speedups for banded matrices

Expressions such as (3.44) 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:

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

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

$$
= B + C, \tag{3.49}
$$

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 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 ap-

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

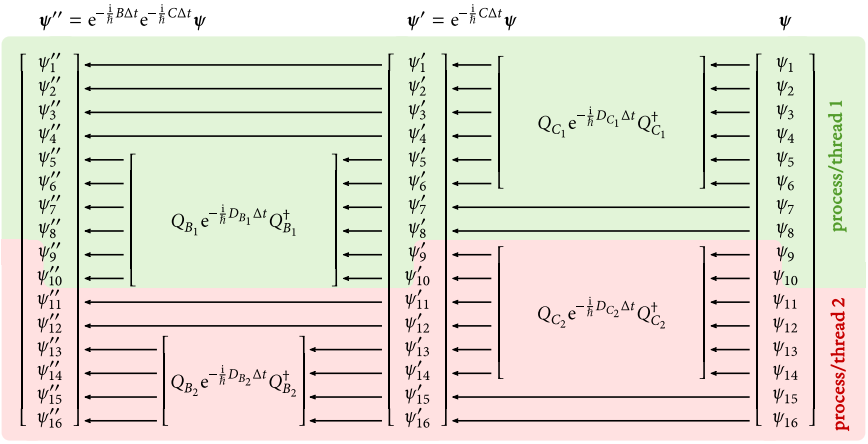

Figure 3.1: Schematic of data flow for parallel split-step method. Computation proceeds right to left. To compute the action of $e^{-\frac{i}{\hbar}B\Delta t}e^{-\frac{i}{\hbar}C\Delta t}$ on a vector ψ, one can treat the submatrices B_1, B_2, C_1 and C_2, of the block-diagonal matrices B and C separately. 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 operations 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. For simplicity this example has two compute threads and two submatrices in each term B and C, but in general there can be any number of either. When a single thread must apply the exponentiation of multiple submatrices to the state vector, it is advantageous for it to first compute the result of any submatrix whose output is required by another thread, then all submatrices that are independent of other threads, and finally any submatrix which requires input from another thread. In this way, data required by other threads can be sent as early as possible, and data needed from other threads called upon as late as possible, minimising the time that threads are waiting for each other whilst there is useful work to be done.

plication 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 neighbour submatrices (any smaller and they would share elements with their next nearest neighbour 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 accurate 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 maximised. With regard to this, one might decide to split A into $2n_{\text{threads}}$ submatrices (where n_{threads} is the number of independent computational

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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.

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 commutation 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 3.2).

Back to our question—does decreasing the timestep size Δt to compensate for the additional commutation 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

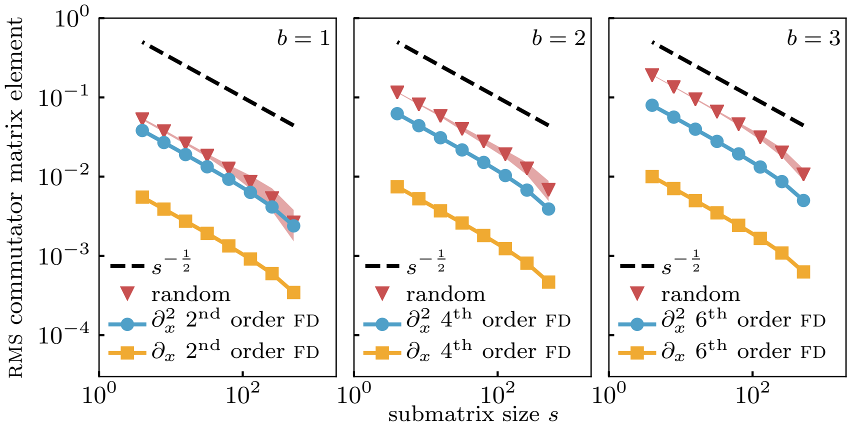

Figure 3.2: Numerical experiment to determine how the commutation error scales with the submatrix size when splitting certain banded matrices into the sum of block-diagonal matrices and using a split-operator method to exponentiate the sum. From left to right, matrices with bandwidth $b = 1$, $b = 1$ and $b = 2$ are considered. For each bandwidth, the matrices for the finite difference schemes of the order resulting in that bandwidth for first and second derivatives are considered, as well as random matrices of the given bandwidth. The random matrices have real and imaginary parts of each element within the band drawn from standard normal distributions. All matrices are 1024×1024. Calculations with random matrices were performed on 20 independent random matrices, and the mean and standard deviation (shaded) of the results plotted. The error metric is the RMS element of the commutator $[B, C]$, where the orginal matrix A is split into the sum of block-diagonal matrices $B + C$ using a submatrix size of s (desribed in-text). The RMS error in a vector propagated with the split-operator method is proportional this error metric. The result in all cases is that the commutator error, and therefore the error in the split-operator method scales with the submatrix size as $s^{-\frac{1}{2}}$.

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

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

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

So the conclusion is: use the smallest submatrices possible when decomposing a banded matrix into a sum of two block-diagonal matrices. The decrease in computational

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

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 without limit but is bounded (disregarding floating point rounding error, which is much smaller than the truncation error of either method) [21]. 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 [21], fourth order accuracy is quite acceptable for many problems.

Applying all the tricks described in the above sections results in $\mathcal{O}(\Delta t)$, $\mathcal{O}(\Delta t^2)$ and $\mathcal{O}(\Delta t^4)$ accurate timestepping methods for solving the Schrödinger 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.[11] 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.[12]

[11] 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.

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

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, it only assumes the differential equation can be expressed in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\psi(t) = \sum_n H_n(t)\psi(t), \tag{3.51}$$

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

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

$U(t_{n+1}, t_n; \psi(t_{n+1}))$—$U$ 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

$$\psi(t_{n+1}) = e^{-\frac{i}{\hbar}K\frac{\Delta t}{2}} e^{-\frac{i}{\hbar}(g\rho(t_{n+1})+V(t_{n+1}))\frac{\Delta t}{2}} e^{-\frac{i}{\hbar}(g\rho(t_n)+V(t_n))\frac{\Delta t}{2}} e^{-\frac{i}{\hbar}K\frac{\Delta t}{2}} \psi(t_n), \tag{3.52}$$

where $\psi(t)$ is the state vector[13] 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.

[13] 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

$$\psi(t_{n+1}) = e^{-\frac{i}{\hbar}(g\rho(t_{n+1})+V(t_{n+1}))\frac{\Delta t}{2}} e^{-\frac{i}{\hbar}K\Delta t} e^{-\frac{i}{\hbar}(g\rho(t_n)+V(t_n))\frac{\Delta t}{2}} \psi(t_n), \tag{3.53}$$

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:

$$\rho(t_{n+1}) = \psi(t_{n+1})\psi^\dagger(t_{n+1}) = \tilde{\psi}\tilde{\psi}^\dagger, \tag{3.54}$$

where

$$\tilde{\psi} = e^{-\frac{i}{\hbar}K\Delta t} e^{-\frac{i}{\hbar}(g\rho(t_n)+V(t_n))\frac{\Delta t}{2}} \psi(t_n). \tag{3.55}$$

The inclusion of $V(t)$ with $g\rho(t)$ 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 invariant 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, [22] have verified using computer symbolic algebra that

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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

Fourth-order Runge–Kutta (RK4) is the enduring workhorse of numerical integration methods. Of the Runge–Kutta methods, it offers a good balance of accuracy and computational cost. When a problem does not have the properties that allow manifestly unitary or error-bounded methods to be used, or when enough computing power can be deployed so as to make these concerns irrelevant, and the programmer's time more important, fourth order Runge–Kutta is a good choice.

It is not manifestly unitary, and its error is not bounded. For a given integration step size, the error will eventually diverge exponentially. This divergence can be delayed, but not prevented, by decreasing the step size. Nonetheless for many problems this is not a concern in practice.

The advantages of fourth-order Runge–Kutta are compelling: it has global error that is fourth order in the integration timestep, involves four function evaluations per timestep, requires only linear arithmetic operations outside of the function evaluations, and can be applied any problem that can be written in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x} = f(\boldsymbol{x}, t), \tag{3.56}$$

for some (possibly nonlinear) function f, where $\boldsymbol{x}$ is a vector of dynamical variables, often in our case the components of a state vector, or for classical dynamics the positions and velocities of an ensemble of particles[14]. Note that this formation allows for initial value problems with coupled ODEs, discretised PDEs, as well as second or higher order differential equations, since an equation of the form

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{x} = f(\boldsymbol{x}, t) \tag{3.57}$$

can be rewritten

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{x}, \dot{\boldsymbol{x}}\right) = \left(\dot{\boldsymbol{x}}, f(\boldsymbol{x}, \dot{\boldsymbol{x}}, t)\right), \tag{3.58}$$

treating the time derivative of each element of $\boldsymbol{x}$ as simply another coupled dynamical variable.

Not unrelated to its ease of implementation, the method itself can be stated concisely. Propagation of the dynamical variables for one timestep from time t to time $t + \Delta t$ is computed as follows:

$$\begin{aligned}
\boldsymbol{k}_1 &= f(\boldsymbol{x}(t), t), \\
\boldsymbol{k}_2 &= f(\boldsymbol{x}(t) + \tfrac{1}{2}\boldsymbol{k}_1 \Delta t, t + \tfrac{1}{2}\Delta t), \\
\boldsymbol{k}_3 &= f(\boldsymbol{x}(t) + \tfrac{1}{2}\boldsymbol{k}_2 \Delta t, t + \tfrac{1}{2}\Delta t), \\
\boldsymbol{k}_4 &= f(\boldsymbol{x}(t) + \boldsymbol{k}_3 \Delta t, t + \Delta t), \\
\boldsymbol{x}(t + \Delta t) &= \boldsymbol{x}(t) + \tfrac{\Delta t}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4).
\end{aligned} \tag{3.59}$$

[14] For classical particle dynamics, often lower order symplectic methods such as the leapfrog method [CITE] can be preferable, but nonetheless it is hard to overstate the usefulness of a general purpose algorithm such as RK4 that can be deployed as a first attempt, or as a plan B once the assumptions required by another algorithm are violated

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

Each step is self-contained, that is, the algorithm does not contain any state from previous steps. This is appealing as it means that f can change discontinuously between timesteps without giving rise to Runge's phenomenon [CITE] in the approximate solutions $\boldsymbol{x}(t)$, as can be the case with multistep methods which do retain some dependency on previous steps. $\boldsymbol{x}(t)$ can also be modified discontinuously between steps without causing problems, as is required by Monte-Carlo wavefunction or quantum jump methods [CITE], or even in the imaginary time evolution method [SEE SECTION SECREF] which requires normalisation of the state vector in between steps. Fourth order Runge–Kutta is therefore quite compatible with stochastic processes and many other models which may not be described by a differential equation alone.

One downside is that the timestep used must be much smaller than the timescale on which $\boldsymbol{x}$ changes - even if $\boldsymbol{x}$'s time variation is highly regular. If $\boldsymbol{x}$'s time variation is dominated by the simple accumulation of complex phase at some angular frequency—as is often the case in quantum mechanics—the timesteps used for RK4 must be small enough to resolve these circles about the complex plane. This is in contrast to the exponentiation methods, for which an energy offset (and hence overall change in the angular frequency at which the state vector's elements accumulate phase) is largely irrelevant. Energy offsets or use of an interaction picture can mitigate this problem but requires some foresight, and may not be possible if the required energy offsets change in time or are not known analytically in advance. The method I develop in section [SECREF] is a partial remedy for this specific problem, which in my opinion is the biggest weakness of fourth order Runge–Kutta as applied to quantum state evolution, when compared to the unitary methods.

3.3.1 Complexity and parallelisability for the Schrödinger equation

Excluding the evaluation of $\boldsymbol{f}$, RK4 requires a number of arithmetic operations proportional to the number of elements in $\boldsymbol{x}$, and so contributes time-complexity $\mathcal{O}(n)$ to the overall calculation, where n is the number of elements in $\boldsymbol{x}$. Since f itself usually doesn't run in linear time, the computational time complexity of RK4 is usually therefore that of evaluating f. Its parallelisability also comes down to that of f, since equations (3.59) above treat each element of $\boldsymbol{x}$ completely independently, with any couplings computed within f.

For the Schrödinger equation in a concrete basis, f is

$$f(\psi, t) = -\frac{\mathrm{i}}{\hbar} H(t)\psi, \tag{3.60}$$

where ψ is the state vector in the given basis and $H(t)$ is the matrix representation of the Hamiltonian in that basis[15]. Fourth order Runge–Kutta is therefore as computationally expensive and parallelisable as computing the matrix-vector product $H(t)\psi$. In the worst case this multiplication is $\mathcal{O}(n^2)$ in the size of the Hilbert space and barely parallelisable at all, if H is dense. But in the common case (as mentioned in section 3.2.3), of H being a sum of terms which act on different subspaces, i.e.

$$H(t) = \sum_{i=1}^{N} \mathbb{I}_{n_1} \otimes \cdots \otimes \mathbb{I}_{n_{i-1}} \otimes H_i(t) \otimes \mathbb{I}_{n_{i+1}} \otimes \cdots \otimes \mathbb{I}_{n_N} \tag{3.61}$$

where n_i is the dimensionality of the subspace acted on by the i^{th} term and $\mathbb{I}_m$ is the $m \times m$ identity matrix, then things are much better. If each term is dense, then the overall cost of evaluating the product $H(t)\psi$ is $\mathcal{O}(n \sum_i n_i)$, where $n = \prod_i n_i$ is the dimensionality of the total Hilbert space, which can be considerably less than the $\mathcal{O}(n^2)$ of evaluating the single matrix-vector product for the total Hamiltonian. And if each term is a banded matrix with bandwidth b_i, then the cost of applying a single term in

[15] For the Gross–Pitaevskii equation this would be a nonlinear $H(\psi, t)$, and everything else in this section still applies.

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

the Hamiltonian becomes $\mathcal{O}(nb_i)$ instead of $\mathcal{O}(nn_i)$, and so the cost of evaluating $H(t)\boldsymbol{\psi}$ becomes $\mathcal{O}(n\sum_i b_i)$. This is identical to the earlier result for the split-operator method once the trick of splitting up banded matrices into block-diagonal matrices was applied (section [SECREF]), but with much less work (in the sense of programmer effort rather computational complexity). Representing $H(t)$ as a sum of terms over different subspaces is no extra work—this is the form we are likely to be writing Hamiltonians in already, and constructing the total $H(t)$ is often not useful. Have you ever considered constructing the matrix form of say, $\frac{\hat{p}_x^2}{2m} + \frac{\hat{p}_y^2}{2m}$ for some finite difference or pseudospectral representations of $\hat{p}_x$ and $\hat{p}_y$? I can't think of a reason to do so [YES I CAN - DISPERSION RELATIONS]. No, we are already applying these operators to different subspaces of the Hilbert space and then summing the results without thinking twice about it, and so the above analysis mostly serves as a reminder that what we are already doing most of the time is in fact very efficient.

Parallelisation, provided one or more of the matrices are banded, is also straightforward, since if A is banded with bandwidth b, then

$$(A\boldsymbol{\psi})_i = \sum_{j=-b}^{b} A_{i,i+j}\psi_j, \tag{3.62}$$

that is, calculation of an element of $A\boldsymbol{\psi}$ requires only the corresponding element of $\boldsymbol{\psi}$ and its nearest b neighbours on each side. One can therefore divide up the state vector into contiguous regions (in the subspace in which A acts), and compute different elements of the product on different compute threads, requiring an exchange of only b elements at each boundary[16] between neighbouring threads.

An additional benefit of parallelised RK4 when compared to split-operator is that the same amount of data needs to be sent between threads regardless of the number of terms in the Hamiltonian. In split-step methods, each term in the Hamiltonian is exponentiated separately (multiple times for the higher order schemes), requiring an exchange of data each time. The amount of data needing to be exchanged per step therefore scales with the number of terms in the Hamiltonian being parallelised, whereas for RK4 it is constant.

The constant factors that big-O notation disregards also favour RK4 when compared to split-operator methods. For one, addition and multiplication are much cheaper than exponentiation, meaning that the exponentials in split-operator methods may add considerable computational cost if the Hamiltonian itself is simple. Furthermore, parallelised or not, each term in fourth order split-operator adds 20 matrix-vector products in the space the term acts, whereas RK4 requires only one additional matrix-vector product per term. In practice due to these properties, RK4 runs considerably faster than split-operator methods, even for simple systems, and the gap widens as complexity increases.

[16] A note about minimising the effect of latency: at the start of an RK4 substep, have each thread send the required elements at the edges of its region in each subspace to its neighbouring threads *first*. *Then* compute $H(t)\boldsymbol{\psi}$ on the interior elements. If each thread has a sufficiently large workload, then by the time this is complete, the data from neighbouring regions will have arrived and threads will not have spent any time waiting for each other.

3.4 Continuous degrees of freedom

The single-particle, non-relativistic, scalar Schrödinger wave equation, as distinct from the general Schrödinger equation (3.1), is:

$$\mathrm{i}\hbar\frac{\partial}{\partial t}\psi(\boldsymbol{r},t) = \left[-\frac{\hbar^2}{2m}\nabla^2 + V(\boldsymbol{r})\right]\psi(\boldsymbol{r},t). \tag{3.63}$$

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

$$\mathrm{i}\hbar\frac{\partial}{\partial t}\psi(\boldsymbol{r},t) = \left[-\frac{\hbar^2}{2m}\nabla^2 + V(\boldsymbol{r}) + g|\psi(\boldsymbol{r},t)|^2\right]\psi(\boldsymbol{r},t), \tag{3.64}$$

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

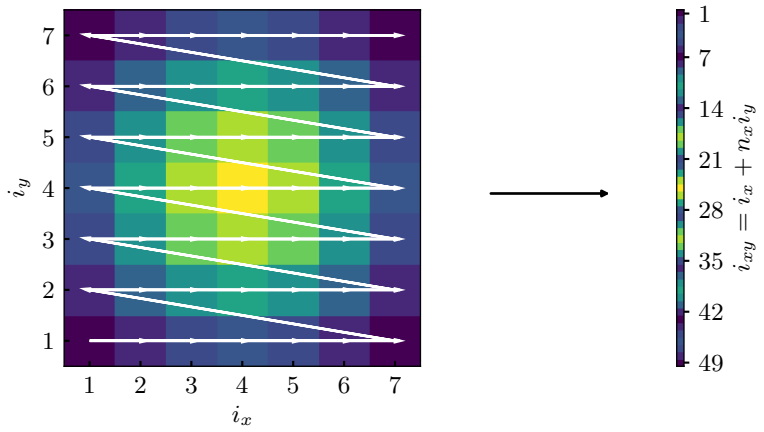

Figure 3.3: Discretising a function over two dimensional space on a grid yields a list of coefficients, one for each gridpoint. These can be arranged as a column vector, and in this way a two dimensional wavefunction approximated by a finite-dimensional state vector. Computationally we don't normally treat this state vector as a column vector—it is more convenient to leave it as a two-dimensional array. But conceptually it is a single vector living in the product space of the discretised x and y spaces.

where $\psi(\boldsymbol{r}, t) = \sqrt{N}\,\langle \boldsymbol{r}|\psi(t)\rangle$ 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 numerical quantum mechanics all state vectors are 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–these objects can be thought of as infinite-dimensional vectors and matrices. So the above two equations are the general Schrödinger equation (3.1), given specific Hamiltonians, an represented in a concrete—albeit infinite-dimensional–basis. But we can only perform a finite number of computations, so what do these vectors and operators look like once we reduce them to something finite? 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, however, spatial discretisation is actually just a special case of a functional basis, namely the Fourier basis, plus an additional approximation or two. The resulting matrices are *banded*, justifying the previous two sections' obsession with dealing with banded matrices efficiently.

3.4.1 Spatial discretisation on a uniform grid: the Fourier basis

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 section 3.1 is to choose a basis in which to represent this state vector.

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

Let's say we discretise space in an equally-spaced $n_x \times n_y = 7 \times 7$ rectangular grid,[17] with spacings Δx and Δy, and only represent the wavefunction at those 49 points in space. The state vector can then be represented by a list of 49 complex numbers, each taken to be the wavefunction's value at the spatial position corresponding to one gridpoint. This 49-vector is now a concrete representation of our state vector (Figure 3.3).

We can also evaluate the potential (and nonlinear term in the case of the Gross–Pitaevskii equation) at each gridpoint and declare this a diagonal operator (Figure 3.4).

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

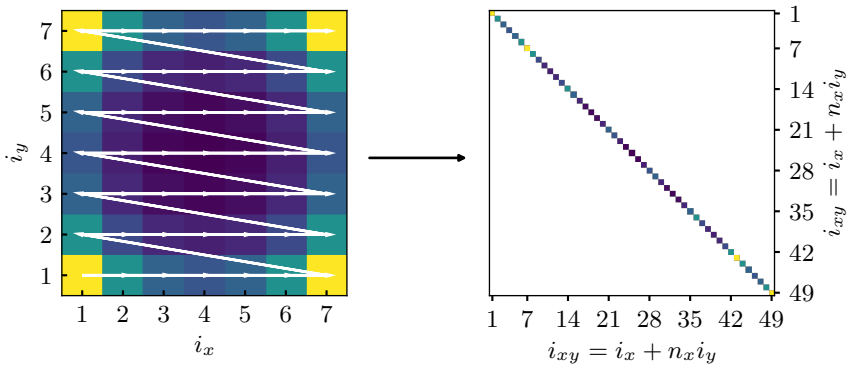

Figure 3.4: [Potential unravelling]

Finally, we could use finite differences to compute the Laplacian - equivalent to replacing the Laplacian with a matrix

$$L = L_x \otimes \mathbb{I}_{n_y} + \mathbb{I}_{n_x} \otimes L_y \tag{3.65}$$

where L_x and L_y are (banded) matrices for finite difference approximations to second derivatives in each direction. We might also use discrete Fourier transforms to evaluate the Laplacian, since

$$\nabla^2 \psi(\boldsymbol{r}) = \mathcal{F}^{-1}\left\{-k^2 \mathcal{F}\{\psi(\boldsymbol{r})\}(\boldsymbol{k})\right\}, \tag{3.66}$$

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

This is all well and good, and it works. 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 section 3.1. Assuming what we've done is equivalent to choosing a basis, that basis has a finite number (49) 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.[18] 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?

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

Prior to discretising, the spatial wavefunction $\psi(\boldsymbol{r}, t) = \langle \boldsymbol{r} | \psi(t) \rangle$ was already the representation of the abstract state vector $|\psi\rangle$ in the "spatial basis"—a basis in which the basis vectors $\{|\boldsymbol{r}\rangle\}$ are Dirac deltas positioned at each point in space. The value of the wavefunction $\psi(\boldsymbol{r})$ for a specific $\boldsymbol{r}$ is then simply a coefficient saying how much of the basis vector $|\boldsymbol{r}\rangle$ to include in the overall state vector. What we have *not* done is chosen a subset of these Dirac delta basis functions as our basis. This would be very strange—our representation of the wavefunction would allow it to be nonzero at the gridpoints, but not in between, like a comb. Spatially separated Dirac deltas do not spatially overlap at all; the matrix elements of the kinetic energy operator:

$$\langle \boldsymbol{r}_i | \hat{K} | \boldsymbol{r}_j \rangle = \int \delta(\boldsymbol{r} - \boldsymbol{r}_i) \left(-\frac{\hbar^2}{2m} \nabla^2 \right) \delta(\boldsymbol{r} - \boldsymbol{r}_j) \, \mathrm{d}\boldsymbol{r} \tag{3.67}$$

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 implicitly chosen a set of two-dimensional boxcar functions centred on each gridpoint with width Δx and Δy in each direction respectively. These cover

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

all space in between gridpoints, but are are not twice differentiable everywhere, and hence the kinetic energy operator's matrix elements cannot be evaluated[19]. No, neither of these bases makes sense. To interpret our spatial grid as a basis, we need a set of functions $\phi_{ij}(\boldsymbol{r})$ (where i and j are the indices of the gridpoints in the x and y directions respectively) that are orthonormal, are nonzero only at one gridpoint and are zero at all others, and are twice differentiable everywhere in our spatial region. Infinite choices are available, differing in which subspace of the original Hilbert space they cover. A sensible heuristic for choosing one 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 a sensible basis. This is made clearer by first discretising in Fourier space instead, and seeing how this can imply a discretisation in real space.

One possible basis for representing all possible state vectors is the Fourier basis $\{|\boldsymbol{k}_{ij}\rangle\}$. 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:

$$\langle \boldsymbol{r}|\boldsymbol{k}_{ij}\rangle = \begin{cases} \frac{1}{\sqrt{A}} e^{i\boldsymbol{k}_{ij}\cdot\boldsymbol{r}} & (\boldsymbol{r} \text{ within 2D region}) \\ 0 & (\boldsymbol{r} \text{ not within 2D region}), \end{cases} \tag{3.68}$$

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

$$\boldsymbol{k}_{ij} = \left[\frac{2\pi i}{L_x}, \frac{2\pi j}{L_y}\right]^{\mathrm{T}}, \tag{3.69}$$

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

$$|\psi\rangle = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \langle \boldsymbol{k}_{i,j}|\psi\rangle \, |\boldsymbol{k}_{i,j}\rangle \tag{3.70}$$

$$\Rightarrow \psi(\boldsymbol{r}) = \langle \boldsymbol{r}|\psi\rangle = \begin{cases} \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \langle \boldsymbol{k}_{ij}|\psi\rangle \frac{1}{\sqrt{A}} e^{i\boldsymbol{k}_{ij}\cdot\boldsymbol{r}} & (\boldsymbol{r} \text{ within 2D region}) \\ 0 & (\boldsymbol{r} \text{ not within 2D region}). \end{cases} \tag{3.71}$$

So $\{\langle \boldsymbol{k}_{ij}|\psi\rangle\}$ 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 $\{\langle \boldsymbol{r}|\boldsymbol{k}_{ij}\rangle\}$ don't have the required properties for a spatial discrete basis. For one, there are an infinite number of them, and we require 49 for our 7×7 example. 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 $|\boldsymbol{k}_{ij}\rangle$ for which:

$$\begin{cases} i \in [-\frac{n_x}{2}, \frac{n_x}{2} - 1] & (n_x \text{ even}) \\ i \in [-\frac{n_x-1}{2}, \frac{N_x-1}{2}] & (n_x \text{ odd}) \end{cases} \tag{3.72}$$

and

$$\begin{cases} j \in [-\frac{n_y}{2}, \frac{n_y}{2} - 1] & (n_y \text{ even}) \\ j \in [-\frac{n_y-1}{2}, \frac{n_y-1}{2}] & (n_y \text{ odd}) \end{cases} \tag{3.73}$$

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

we include only the n_x and n_y 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 short wavelength components. Because the kinetic energy operator, when represented in the Fourier basis, is:

$$\langle \boldsymbol{k}_{ij}|\hat{K}|\boldsymbol{k}_{i'j'}\rangle = \frac{\hbar^2 k^2}{2m}\delta_{ii'}\delta_{jj'}, \tag{3.74}$$

where $k = |\boldsymbol{k}_{ij}|$, 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.[20] It also matches our earlier intuition that our basis should represent wavefunctions that don't vary much between gridpoints—here we are discarding short wavelengths and hence limiting wavefunctions we can represent to ones that vary slowly in space compared to the cutoff wavelength.

[20] 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 a set of 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:

$$|\boldsymbol{r}_{ij}\rangle = \sum_{i'}^{n_x}\sum_{j'}^{n_y} e^{-i\boldsymbol{k}_{i'j'}\cdot\boldsymbol{r}_{ij}}\,|\boldsymbol{k}_{i'j'}\rangle \tag{3.75}$$

with $\boldsymbol{r}_{ij} = (i\Delta x, j\Delta y)^{\mathrm{T}}$ and. The set of vectors $\{|\boldsymbol{r}_{i,j}\rangle\}$ are also an orthonormal basis, related to the discrete Fourier basis by a unitary transformation with matrix elements:

$$U_{\text{DFT2},i'j'ij} = \langle \boldsymbol{k}_{i'j'}|\boldsymbol{r}_{ij}\rangle = e^{-i\boldsymbol{k}_{i'j'}\cdot\boldsymbol{r}_{ij}}. \tag{3.76}$$

This unitary transformation is in fact a two-dimensional discrete Fourier transform (hence the subscript), and the basis vectors $\{|\boldsymbol{r}_{ij}\rangle\}$ have spatially localised wavefunctions that are nonzero only at one of the spatial gridpoints. Vectors and matrices can be transformed from their discrete Fourier space representation to their discrete real-space representation and back using the unitary U_{DFT2}:

$$\boldsymbol{\psi}_{\text{real}} = U_{\text{DFT2}}^{\dagger}\boldsymbol{\psi}_{\text{Fourier}} \tag{3.77}$$
$$\boldsymbol{\psi}_{\text{Fourier}} = U_{\text{DFT2}}\boldsymbol{\psi}_{\text{real}} \tag{3.78}$$
$$A_{\text{real}} = U_{\text{DFT2}}^{\dagger}A_{\text{Fourier}}U_{\text{DFT2}} \tag{3.79}$$
$$A_{\text{Fourier}} = U_{\text{DFT2}}A_{\text{Real}}U_{\text{DFT2}}^{\dagger}. \tag{3.80}$$

where $\boldsymbol{\psi}_{\text{real}}$ is the vector of coefficients $\psi_{\text{real},ij} = \langle \boldsymbol{r}_{ij}|\psi\rangle$ for representing a state vector in the discrete real space basis, $\boldsymbol{\psi}_{\text{Fourier}}$ is the vector of coefficients $\psi_{\text{Fourier},ij} = \langle \boldsymbol{k}_{ij}|\psi\rangle$ for the state vector in the discrete Fourier basis, and A_{real} and A_{Fourier} are the representations of some operator $\hat{A}$ in the discrete real and Fourier bases respectively, with matrix elements $A_{\text{real},iji'j'} = \langle \boldsymbol{r}_{ij}|\hat{A}|\boldsymbol{r}_{i'j'}\rangle$ and $A_{\text{Fourier},iji'j'} = \langle \boldsymbol{k}_{ij}|\hat{A}|\boldsymbol{k}_{i'j'}\rangle$.

The spatial representation of the basis vectors $\{|\boldsymbol{r}_{ij}\rangle\}$ can be computed using (3.68) as:

$$\phi_{ij}(\boldsymbol{r}) = \langle \boldsymbol{r}|\boldsymbol{r}_{ij}\rangle = \sum_{i'j'} e^{-\mathrm{i}\boldsymbol{k}_{i'j'}\cdot\boldsymbol{r}_{ij}}\,\langle \boldsymbol{r}|\boldsymbol{k}_{i'j'}\rangle \tag{3.81}$$

$$\Rightarrow \phi_{ij}(\boldsymbol{r}) = \begin{cases} \sum_{i'j'} \frac{1}{\sqrt{A}} e^{\mathrm{i}\boldsymbol{k}_{i'j'}\cdot(\boldsymbol{r}-\boldsymbol{r}_{ij})} & (\boldsymbol{r} \text{ within 2D region}) \\ 0 & (\boldsymbol{r} \text{ not within 2D region}), \end{cases} \tag{3.82}$$

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

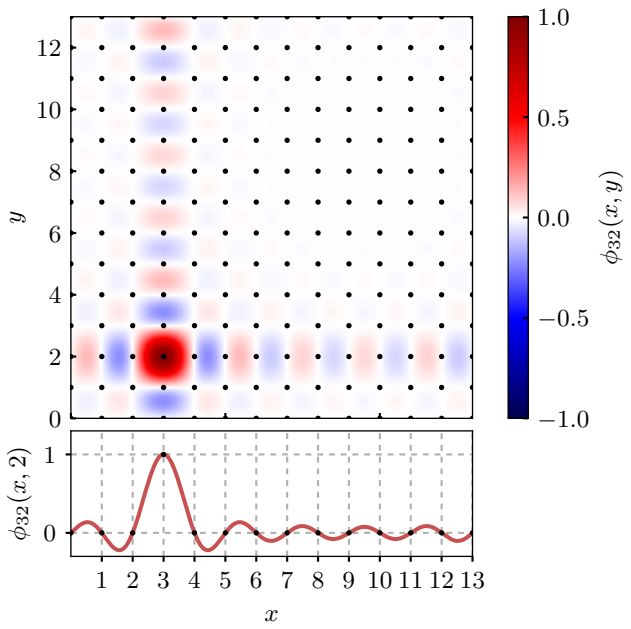

Figure 3.5: [basis vecs]

and an example is plotted in Figure 3.5.

These functions are sometimes called *periodic sinc functions*, or *band-limited delta functions* [CITE]. Each of them is zero at all of the 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 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 reflected off a barrier. This effect is entirely unphysical[21] and should be taken as a sign that the spatial grid is not fine enough for the dynamics being simulated.

[21] 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 would correspond to Bloch oscillations.

The discrete Fourier basis we've described is one example of a *spectral basis*, and a numerical method which represented the state vector solely in this basis would be called a *spectral method*. Other choices of basis functions, such as polynomials ([SECREF FEDVR]) or harmonic oscillator basis functions ([SECREF]) lead one to other spectral methods. Perhaps surprisingly, the corresponding discrete spatial basis discussed above, despite being related to the Fourier basis by a unitary transformation, is instead called a *pseudospectral* basis, and a method representing the state vector in this basis a *pseudospectral method*. Although it is "just another basis", the fact that the basis functions are zero at all spatial points bar one each leads to the possibility of a further approximation when

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

representing some operators, which makes bases with this property especially useful. I describe this in the following section.

Vector and matrix elements in the Fourier and pseudospectral bases

We now have a finite basis $\{|\boldsymbol{r}_{ij}\rangle\}$ that matches our intuitions somewhat for representing a wavefunction at a set of gridpoints. A state vector can be approximated as a linear sum of these basis vectors, with the coefficient for each one being equal to the projection of state vector's wavefunction onto the basis vector's wavefunction:

$$|\psi\rangle \approx \sum_{ij} \psi_{ij} |\boldsymbol{r}_{ij}\rangle\,, \tag{3.83}$$

where

$$\psi_{ij} = \int \phi^*_{ij}(\boldsymbol{r})\psi(\boldsymbol{r})\,\mathrm{d}\boldsymbol{r}\,. \tag{3.84}$$

In practice however this integral is rarely done. Instead, ψ_{ij} is simply taken to be the value of the exact wavefunction $\psi(\boldsymbol{r}) = \langle \boldsymbol{r}|\psi\rangle$ at the point $\boldsymbol{r} = \boldsymbol{r}_{ij}$:

$$\psi_{ij} \approx \psi(\boldsymbol{r}_{ij}) \tag{3.85}$$

To see why this is a good approximation, imagine that the approximation (3.83) were exact, that is, $\psi(\boldsymbol{r})$ were exactly equal to a linear sum of the functions $\{\phi_{ij}(\boldsymbol{r})\}$. Since all the basis functions are zero at the point $\boldsymbol{r}_{ij}$ except for ϕ_{ij}, the value of $\psi(\boldsymbol{r}_{ij})$ must come solely from the $\psi(\boldsymbol{r})$'s projection onto the basis function $\phi_{ij}(r)$. Since approximating (3.83) underlies the results of any simulation that discretises a state vector in this way, treating it as exact for the initial projection onto the discrete basis is making an assumption no worse than that already being relied upon.

With a basis and initial discrete state vector in hand, we can now proceed to calculate matrix elements of the Hamiltonian, after which we can proceed to solve the differential equation (3.3) to determine how the coefficients $\{\psi_{ij}\}$ evolve in time.

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

$$\hat{H}_{\text{Schrö}} = \frac{\hbar^2\hat{k}^2}{2m} + V(\hat{\boldsymbol{r}}), \tag{3.86}$$

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:

$$\langle \boldsymbol{k}'|\frac{\hbar^2\hat{k}^2}{2m}|\boldsymbol{k}\rangle = \frac{\hbar^2 k^2}{2m}\delta(\boldsymbol{k}-\boldsymbol{k}'), \tag{3.87}$$

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

$$\langle \boldsymbol{r}'|V(\hat{\boldsymbol{r}})|\boldsymbol{r}\rangle = V(\boldsymbol{r})\delta(\boldsymbol{r}-\boldsymbol{r}'). \tag{3.88}$$

The kinetic term is also diagonal our discrete Fourier basis:

$$K_{\text{Fourier},i'j'ij} = \langle \boldsymbol{k}_{i'j'}|\frac{\hbar^2\hat{k}^2}{2m}|\boldsymbol{k}_{ij}\rangle = \frac{\hbar^2 k_{ij}^2}{2m}\delta_{i'i}\delta_{j'j}, \tag{3.89}$$

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

however—perhaps surprisingly—the potential term is not diagonal in the pseudospectral basis, only approximately so:

$$V_{\text{real},i'j'ij} = \langle \boldsymbol{r}_{i'j'} | V(\hat{\boldsymbol{r}}) | \boldsymbol{r}_{ij} \rangle \approx V(\boldsymbol{r}_{ij}) \delta_{i'i} \delta_{j'j}. \tag{3.90}$$

Nonetheless, equation (3.90) is in practice treated as exact for the purpose of computing matrix elements in the discrete basis of operators that are diagonal in the full spatial basis. As above with projecting a state vector using simply the values of a wavefunction at the gridpoints (rather than doing integrals), treating this approximation as exact is equivalent to making the assumption that the potential $V(\boldsymbol{r})$ is already accurately representable in the discrete basis as a diagonal operator:

$$V(\boldsymbol{r}) \approx \sum_{ij} V(\boldsymbol{r}_{ij}) \phi^*_{ij}(\boldsymbol{r}) \phi_{ij}(\boldsymbol{r}), \tag{3.91}$$

and so similarly is going to be a good approximation whenever the discrete basis is well able to represent the potential. So long as your discrete basis can accurately represent the state vectors and spatially-diagonal operators you will be using, it makes little difference whether you project those vectors and operators onto the basis using integrals or using simply their values at the gridpoints.

This alternate method of projection has a name, it is called *collocation* [14, p. 227]. Using collocation instead of vector projection amounts to treating our basis vectors as a scheme for interpolating state vectors and operators in between gridpoints, given their values at the points, rather than as basis vectors to project upon. Another way to interpret collocation is to say that we are evaluating the vector projections using integrals after all, however, we're numerically computing those integrals using a quadrature scheme, only evaluating the integrand at the gridpoints and performing a discrete sum [14, p. 283]. Collocation is what puts the *pseudo* in pseudospectral—if we evaluated all these operators using integrals instead, we would be treating the discrete spatial basis exactly the same as any spectral basis.

Compared to a purely spectral method, pseudospectral methods are of comparable accuracy [23]. This makes intuitive sense—discrete sums instead of integrals is how we are going to do all inner products once we are in the discrete basis—so it can't be much worse to use the same method to approximate operators and initial state vectors. The sole downside of pseudospectral methods, according to [23], is that the error can lead to instability in the presence of certain nonlinearities. Specifically, if long wavelength waves interact in a way that would produce wavelengths shorter than two grid spacings (the Nyquist wavelength), pseudospectral methods will produce longer wavelength waves instead, whereas in a purely spectral method the interaction would not occur, being due to couplings to a Fourier mode outside the discrete Fourier basis. This *aliasing* can cause instability, but can be circumvented with smoothing techniques [24]. This is no great downside: if you want to simulate short length scales, you need to choose a grid spacing small enough to represent them, whereas if short wavelengths are produced despite your willingness to ignore them, you must smooth them away before they are aliased into long wavelengths that you do care about.

Finally, armed with a kinetic energy operator in Fourier space and a pseudospectral approximation to the potential operator in real space, we can write all the matrix elements of $\hat{H}_{\text{Schrö}}$ in a single basis, and thus our discretised, pseudospectral two-dimensional Schrödinger wave equation:

$$i\hbar \frac{\mathrm{d}}{\mathrm{d}t} \psi_{i'j'}(t) = \sum_{ij} H_{i'j',ij}(t) \psi_{ij}(t) \tag{3.92}$$

$$\Rightarrow i\hbar \frac{\mathrm{d}}{\mathrm{d}t} \psi_{i'j'}(t) = \sum_{ij} \left[U^\dagger_{\text{DFT2}} K_{\text{Fourier}} U_{\text{DFT2}} + V_{\text{real}}(t) \right]_{i'j',ij} \psi_{ij}(t), \tag{3.93}$$

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

where we have used the discrete Fourier transform to transform the kinetic energy operator into the discrete real space basis, and allowed the potential operator to be possibly time dependent.

The right hand side of this expression can now simply be evaluated, yielding the time derivative of each component $\psi_{ij}(t)$ of the discrete state vector $\boldsymbol{\psi}(t)$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}(t) = -\frac{\mathrm{i}}{\hbar}\left[U^{\dagger}_{\text{DFT2}} K_{\text{Fourier}} U_{\text{DFT2}} \boldsymbol{\psi}(t) + V_{\text{real}}(t)\boldsymbol{\psi}(t)\right] \tag{3.94}$$

$$\Rightarrow \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\psi}(t) = -\frac{\mathrm{i}}{\hbar}\left[\text{FFT}_2^{-1}\left\{\frac{\hbar^2 \tilde{\boldsymbol{k}}^{\odot 2}}{2m} \odot \text{FFT}_2\left\{\boldsymbol{\psi}(t)\right\}\right\} + V(\tilde{\boldsymbol{r}}, t) \odot \boldsymbol{\psi}(t)\right], \tag{3.95}$$

where FFT_2 is the two dimensional fast Fourier transform, an efficient implementation of the discrete Fourier transform (taking time $\mathcal{O}(n \log n)$ in the size of each dimension), $\tilde{\boldsymbol{r}}$ is a vector (of vectors) containing each discrete position vector (such that $V(\tilde{\boldsymbol{r}}, t)$ is a vector (of scalars) containing the potential evaluated at each discrete position), $\tilde{\boldsymbol{k}}$ is a vector (of vectors) containing each discrete k-vector, such that $\tilde{\boldsymbol{k}}^{\odot 2}$ is a vector (of scalars) containing the squared magnitude of each discrete k-vector, and $\odot$ represents elementwise multiplication (or exponentiation) of vectors. Other than $\boldsymbol{\psi}$, these vectors are more akin to arrays used in programming languages than to members of a vector space, hence the somewhat clunky notation in (3.95). Comparison with the continuous version of equation (3.95) (i.e. the Schrödinger wave equation (3.63)), since elementwise multiplication of functions is a more common operation in mathematics, might be clarifying:

$$\frac{\partial}{\partial t}\psi(\boldsymbol{r}, t) = -\frac{\mathrm{i}}{\hbar}\left[\mathcal{F}^{-1}\left\{\frac{\hbar^2 k^2}{2m}\mathcal{F}\left\{\psi(\boldsymbol{r}, t)\right\}(\boldsymbol{k})\right\}(\boldsymbol{r}) + V(\boldsymbol{r}, t)\psi(\boldsymbol{r}, t)\right], \tag{3.96}$$

where $\mathcal{F}$ is the continuous Fourier transform, and k as always is $|\boldsymbol{k}|$. So we see that the discretised Schrödinger equation for a single particle in a potential really is the same as evaluating the continuous equation at a set of gridpoints, evaluating spatial derivatives in Fourier space, and replacing the continuous Fourier transform with its discrete equivalent.

Here is an example of how one might compute the RHS of (3.95) in Python code:

```python
import numpy as np
from numpy.fft import fft2, ifft2, fftfreq

pi = np.pi
u  = 1.660539e-27 # unified atomic mass unit
m  = 86.909180*u # 87Rb atomic mass
omega = 15 # Harmonic trap frequency
hbar = 1.054571726e-34 # Reduced Planck's constant

# Space:
nx = ny = 256
x_max = y_max = 100e-6

# Arrays of components of position vectors. The reshaping is to ensure that
# when used in arithmetic with each other, these arrays will be treated as if
# they are two dimensional with repeated values along the dimensions of size
# 1, up to the size of the other array (this is called broadcasting in numpy).
x = np.linspace(-x_max, x_max, nx, endpoint=False).reshape(1, nx)
y = np.linspace(-y_max, y_max, ny, endpoint=False).reshape(ny, 1)

# Grid spacing:
dx = x[0, 1] - x[0, 0]

# Arrays of components of k vectors.
kx = 2*pi*fftfreq(nx, d=dx).reshape(1, nx)
ky = 2*pi*fftfreq(nx, d=dx).reshape(ny, 1)

# The kinetic energy operator in Fourier space (shape ny, nx).
K_fourier = hbar**2 * (kx**2 + ky**2)/(2*m)
```

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

```
30
31 # The potential operator in real space (shape ny, nx)
32 V_real = 0.5 * m * omega**2 * (x**2 + y**2)
33
34 def dpsi_dt(t, psi):
35     """Return a 2D array for  the time derivative of the 2D array psi
36     representing a discretised wavefucntion obeying the Schrodinger wave
37     equation"""
38     K_real_psi = ifft2(K_fourier * fft2(psi))
39     return -1j/hbar * (K_real_psi + V_real * psi)
```

Where the example is for a time-independent potential. If the potential were time dependent, `V_real` within the function `dpsi_dt(t, psi)` would need to be replaced with a call to a function that returned an array for `V_real` at time `t`.

Starting with some initial discrete wavefunction, this could then be solved with a forward differencing scheme like fourth order Runge–Kutta (section 3.3):

```
1  def rk4(t, t_final, dt, psi, dpsi_dt):
2      """Evolve the initial array psi_initial forward in time from time t to
3      t_final according to the differential equation dpsi_dt using fourth order
4      Runge-Kutta with timestep dt"""
5      while t < t_final:
6          k1 = dpsi_dt(t, psi)
7          k2 = dpsi_dt(t + 0.5 * dt, psi + 0.5 * k1 * dt)
8          k3 = dpsi_dt(t + 0.5 * dt, psi + 0.5 * k2 * dt)
9          k4 = dpsi_dt(t + dt, psi + k3 * dt)
10
11         psi[:] += dt/6.0 * (k1 + 2 * k2 + 2 * k3 + k4)
12
13         t += dt
14
15     return psi
```

The discretised differential equation (3.94) can also be solved using a split-step method (section 3.2.3), since the Hamiltonian matches the requirements of being written as a sum of terms for which individually an eigenbasis is known (the discrete real space basis for the potential term, and the Fourier basis for the kinetic term). For example, here is how one might implement second or fourth-order split-step (only a single timestep shown):

```
1  def split_step2(t, psi, dt):
2      """Evolve psi in time from t to t + dt using a single step of the second
3      order Fourier split-step method with timestep dt"""
4
5      # First evolve using the potential term for half a timestep:
6      psi *= np.exp(-1j/hbar * V_real * 0.5 * dt)
7
8      # Then evolve using the kinetic term for a whole timestep, tranforming to
9      # and from Fourier space where the kinetic term is diagonal:
10     psi = ifft2(np.exp(-1j/hbar * K_fourier * dt) * fft2(psi))
11
12     # Then evolve with the potential term again for half a timestep:
13     psi *= np.exp(-1j/hbar * V_real * 0.5 * dt)
14
15     return psi
16
17 def split_step4(t, psi, dt):
18     """Evolve psi in time from t to t + dt using a single step of the fourth
19     order Fourier split-step method with timestep dt"""
20     p = 1/(4 - 4**(1/3.0))
21
22     # Five applications of second-order split-step using timesteps
23     # of size p*dt, p*dt, (1 - 4*p)*dt, p*dt, p*dt
24     for subdt in [p*dt, p*dt, (1 - 4*p)*dt, p*dt, p*dt]:
25         psi = split_step2(t, psi, subdt)
26         t += subdt
27     return psi
```

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

In all the above code examples, a nonlinear term as in the case of the Gross-Pitaevskii equation can be included in the potential simply by adding a term `g * np.abs(psi)**2` to the potential `V_real` wherever it appears. As discussed in section 3.2.3, the nonlinearity poses no problem for the split-step methods so long as the potential term of the Hamiltonian is evaluated as the outermost sandwich of exponentials in the second-order split step method (which comprises the sub-steps of fourth-order split-step).

3.4.2 Finite differences

Fourier split-step, or using discrete Fourier transforms to evaluate the spatial derivatives at each gridpoint in order to time-evolve using Runge–Kutta are effective and versatile numerical methods.

The use of discrete Fourier transforms in the previous section can be seen as replacing the Laplacian operator in the Schrödinger wave equation (3.63) with the equivalent operation in Fourier space:

$$\nabla^2\psi(\tilde{\boldsymbol{r}}) \approx \text{FFT}_2^{-1}\left\{-\tilde{\boldsymbol{k}}^{\odot 2} \odot \text{FFT}_2\left\{\psi(\tilde{\boldsymbol{r}})\right\}\right\}, \tag{3.97}$$

where as before $\tilde{\boldsymbol{r}}$ is a vector (of vectors) containing the discrete positions, $\tilde{\boldsymbol{k}}$ is a vector (of vectors) containing discrete k-vectors such that $\tilde{\boldsymbol{k}}^{\odot 2}$ is a vector (of scalars) containing the squared magnitudes of each k-vector, and $\odot$ represents elementwise multiplication or exponentiation of vectors. More generally for any derivative,

$$\frac{\partial}{\partial x}\psi(\tilde{\boldsymbol{r}}) \approx \text{FFT}_2^{-1}\left\{\mathrm{i}\tilde{\boldsymbol{k}}_x \odot \text{FFT}_2\left\{\psi(\tilde{\boldsymbol{r}})\right\}\right\}, \tag{3.98}$$

where $\tilde{\boldsymbol{k}}_x$ is a vector (of scalars) containing the discrete angular wavenumbers for the x spatial dimension.

Equations (3.97) and (3.98) are exact for any wavefunction $\psi(\boldsymbol{r})$ which is periodic and band-limited to the discrete Fourier space (and thus exactly representable as a vector $\boldsymbol{\psi}$ of its values at each gridpoint), which is why the Fourier method of computing derivatives this way is sometimes said to be accurate to "infinite order" [25] in the grid spacings Δx and Δy, in contrast to fixed-order approximations to derivatives which are second, fourth, sixth order etc. In practice the Fourier method for derivatives is often used for wavefunctions which are *not* intended to be periodic (the periodicity imposed by using the method is unphysical), and so for these it has merely very high order accuracy, not literally infinite.

In any case, such high accuracy is not often necessary—if one is using only an $\mathcal{O}(\Delta t^4)$ accurate timestepping scheme, then the timestepping may be the limiting factor in overall accuracy and it might be wise to decrease the accuracy of computing spatial derivatives if there is otherwise a benefit to doing so.

To that end, the Fourier method of derivatives may be replaced with *finite differences* instead. Although finite differences are usually derived as approximations to derivatives directly from the definition of the derivative without reference to discrete Fourier transforms, they can be considered fixed-order approximations to the Fourier method [25]. Thus operators whose form in Fourier space corresponds to a derivative of some order can be approximated with finite differences:

$$U_{\text{DFT2}}^{\dagger} k_x U_{\text{DFT2}}\psi(\tilde{\boldsymbol{r}}) = -\mathrm{i}\delta_{\Delta x}^{(n)} \otimes \mathbb{I}_{n_y}\psi(\tilde{\boldsymbol{r}}) + \mathcal{O}(\Delta x^n) \tag{3.99}$$

where k_x is the diagonal matrix of the discrete angular wavenumbers for the x spatial dimension and $\delta_{\Delta x}^{(n)}$ is the matrix representing n^{th} order finite difference approximation to the first derivative using grid spacing Δx. The matrix elements of this and some other central finite differences are shown in Table 3.1.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

	$k=-3$	$k=-2$	$k=-1$	$k=0$	$k=1$	$k=2$	$k=3$
$\Delta x\left(\delta_{\Delta x}^{(2)}\right)_{i,i+k}$			$-\frac{1}{2}$	0	$\frac{1}{2}$		
$\Delta x\left(\delta_{\Delta x}^{(4)}\right)_{i,i+k}$		$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$	
$\Delta x\left(\delta_{\Delta x}^{(6)}\right)_{i,i+k}$	$-\frac{1}{60}$	$\frac{3}{20}$	$-\frac{3}{4}$	0	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$
$\Delta x^2\left(\delta_{\Delta x}^{2(2)}\right)_{i,i+k}$			1	-2	1		
$\Delta x^2\left(\delta_{\Delta x}^{2(4)}\right)_{i,i+k}$		$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$	
$\Delta x^2\left(\delta_{\Delta x}^{2(6)}\right)_{i,i+k}$	$\frac{1}{90}$	$-\frac{3}{20}$	$\frac{3}{2}$	$-\frac{49}{18}$	$\frac{3}{2}$	$-\frac{3}{20}$	$\frac{1}{90}$

Table 3.1: Matrix elements [26] for some finite-differencing schemes for first ($\delta_{\Delta x}^{(n)}$) and second ($\delta_{\Delta x}^{2(n)}$) derivatives using central finite differences of various orders n for uniform grid spacing Δx. All finite difference matrices are banded; each column here shows the matrix elements of k^{th} diagonal, which are all identical. Elements outside of each matrix's band are left blank. Each matrix element is shown multiplied by factors of Δx for clarity.

The fact that the finite difference matrices are banded allows them to be computing by applying a "stencil" to a discrete state vector, computing an approximation to some linear sum of derivative operators at each point of discrete space by consideration of only that point and a small number of surrounding point. For example, a $\mathcal{O}(\Delta x^2)$ approximation (assuming $\Delta x = \Delta y$) to the kinetic energy operator in two dimensions may be evaluated at each point as:

$$K_{\text{real}}\psi(\tilde{\boldsymbol{r}}) = U_{\text{DFT2}}^{\dagger}\frac{\hbar^2(k_x^2+k_y^2)}{2m}U_{\text{DFT2}}\psi(\tilde{\boldsymbol{r}}) \tag{3.100}$$

$$= -\frac{\hbar^2}{2m}\left[\delta_{\Delta x}^{2(n)}\otimes\mathbb{I}_{n_y} + \mathbb{I}_{n_x}\otimes\delta_{\Delta y}^{2(2)}\right]\psi(\tilde{\boldsymbol{r}}) + \mathcal{O}(\Delta x^2) \tag{3.101}$$

$$\Rightarrow \left(K_{\text{real}}\psi(\tilde{\boldsymbol{r}})\right)_{ij} = -\frac{\hbar^2}{2m}\Big[-4\psi(x_i,y_j) + \psi(x_{i-1},y_j) + \psi(x_{i+1},y_j) + \psi(x_i,y_{j-1}) + \psi(x_i,y_{j+1})\Big] + \mathcal{O}(\Delta x^2). \tag{3.102}$$

Thus the kinetic energy operator, when approximated using finite differences, is an example of an operator that can be written as a sum of banded operators acting on different subspaces of the total Hilbert space—the identity matrices in (3.101) each leave a part of the Hilbert space untouched. As mentioned in [SECREF], this in principle considerably reduces the computational cost of applying the approximate kinetic energy operator to a discretised state vector. Fourier transforms, when computed with the fast Fourier transform algorithm, are already less computationally expensive than a general matrix-vector multiplication, that is, the fast Fourier transform allows one to multiply the matrix U_{DFT2} in (3.100) by a vector considerably faster than $\mathcal{O}(n_x^2 n_y^2)$, which would be the computational time-complexity for a general $n_x n_y \times n_x n_y$ matrix. Firstly, a two-dimensional discrete Fourier transform can also be written as the sum of two one-dimensional transformations operating on different subspaces:

$$U_{\text{DFT2}} = U_{\text{DFT},x}\otimes\mathbb{I}_{n_y} + \mathbb{I}_{n_x}\otimes U_{\text{DFT},y}, \tag{3.103}$$

where $U_{\text{DFT},x}$ and $U_{\text{DFT},y}$ are the unitaries for one-dimensional discrete Fourier transforms in the x and y dimensions respectively. So even if $U_{\text{DFT},x}$ and $U_{\text{DFT},y}$ were arbitrary matrices, this already would reduce the cost of multiplying U_{DFT2} by a vector to $\mathcal{O}(n_y n_x^2 + n_x n_y^2)$ But they are not arbitrary matrices—each of these one-dimensional Fourier transforms has

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

computational cost $\mathcal{O}(n \log n)$ using the FFT algorithm, where n is the number of points in the relevant dimension, resulting in an overall cost of $\mathcal{O}(n_y n_x \log n_x + n_x n_y \log n_y)$ for applying the two-dimensional Fourier transform U_{DFT2} to a vector. Finite differences improves on this further. As discussed in [SECREF], since our finite-differences approximation to the kinetic energy operator can written as the sum of banded matrices operating on different subspaces, the computational cost of applying it to a vector is $\mathcal{O}(b n_x n_y)$, where b is the bandwidth of the banded matrices, which depends on which order accuracy is used (for example, for second order finite-differences $b = 1$). This is faster than the Fourier method of computing the kinetic energy operator by a factor of $\mathcal{O}(\log n_x + \log n_y)$. Whilst this seems considerable, the difference is hard to observe in pracice. On ordinary computers the number of points needs to be increased so much in order to measure any difference in speed between the algorithms that the data no longer fits in CPU cache and copying the data to and from main memory becomes the bottleneck. Although copying data from memory is a linear-time process, the coefficient of that linear time is large enough to make the asymptotic speed of finite differences vs. Fourier transforms not relevant for ordinary computers, in my experience.

No, the practical advantages of finite differences compared to Fourier transforms do not come down to single-core speed. Rather, they are:

1. Being banded matrices, the finite-difference approximations to the kinetic energy operator can be multiplied by a vector, or exponentiated and applied to a vector, in parallel on a cluster computer or GPU using the techniques discussed in section [SECREF], whereas the fast Fourier transform is less efficiently parallelisable [27]. The speed of finite differences can have very nice scaling with the number of CPU cores or cluster nodes used, since only b points need to be exchanged between cores at each step when applying or exponentiating the kinetic energy operator. For large problems, 'superscaling' can even be observed, whereby the speedup factor obtained by moving to multiple CPUs or compute nodes on a cluster is larger than the number of CPUs/nodes used. This is counterintuitive, but comes from more effectively using CPU cache—by spreading the data over multiple cores, one minimises the proportion of the state vector that needs to reside in main memory instead of in (much faster) CPU cache at any one time.

2. One can intervene at the boundaries to impose boundary conditions other than periodicity. Strictly speaking, as an approximation to the Fourier method of computing derivatives, the indices for the matrix elements as given in Table 3.1 should be read as wrapping around to the other side of the spatial region whenever they would go out of bounds—that is, $(\delta^{2(n)}_{\Delta x})_{i,i+k}$ should be read as $(\delta^{2(n)}_{\Delta x})_{i,(i+k) \mod n_x}$. However, as mentioned, periodicity is often an undesired consequence of the Fourier method of derivatives. Alternatively one can simply omit these matrix elements that would couple spatial points across the boundary, which has the result of imposing zero boundary conditions instead of periodic. Judicious deletion of matrix elements at other points in space can also be used to impose zero boundary conditions elsewhere, equivalent to an infinite potential barrier which would otherwise be numerically troublesome if done with the potential energy term of the Hamiltonian. Other interventions in the application of the kinetic energy operator can be used to impose other boundary conditions such as constant-value, constant-gradient, etc., whereas the Fourier method is less flexible in this respect.

3. Finite differences are compatible with non-uniform grids, whereas the Fourier method is limited to uniform grids. Non-uniform grids imply different matrix elements [26] for the finite difference operators, but are otherwise treated exactly the same. This allows more dense placement of gridpoints in regions where wavefunctions may have finer spatial structure, without having to waste computational

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

power on regions of space where the wavefunction is known to have only coarser structure. An example is an electron in a Coulomb potential, an accurate simulation of which would need to capture fine details at small radii but less detail at larger radii. A transformation into spherical coordinates with a non-uniform grid for the radial coordinate could be well treated with finite differences.

4. Finite differences are compatible with the use of split-step methods with some operators that are diagonal in neither Fourier nor real space. For example, the real-space representation of the operator for the z component of angular momentum is $L_z = -i\hbar\left(x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}\right)$. With diagonal matrices X and Y being used for the the $\hat{x}$ and $\hat{y}$ operators in accordance with the pseudospectral method, and finite differences being used to approximate the derivatives, the result is a banded matrix representation of L_z, compatible with the techniques from section [SECREF parallel] for reducing the problem to that of many small matrices instead of one big one. There is a little more complexity, L_z cannot be represented as a sum of operators acting on the x and y subspaces separately—instead, each term in L_z is a *product* of operators that act on different subspaces. Consider the first term of a discretised version of L_z using fourth-order finite differences. It can be diagonalised in the following way:

$$-i\hbar X \otimes \delta^{(4)}_{\Delta y} = -i\hbar\left(\mathbb{I}_{n_x} X \mathbb{I}_{n_x}\right) \otimes \left(Q^\dagger_{\delta_y} D_{\delta_y} Q_{\delta_y}\right), \tag{3.104}$$

where Q_{δ_y} and D_{δ_y} are the unitary and diagonal matrix that diagonalise $\delta^{(4)}_{\Delta y}$ (X is already diagonal, and so is 'diagonalised' by the identity matrix for the x subspace). $\mathbb{I}_{n_x}$ and X act on the x subspace, whereas Q_{δ_y} and D_{δ_y} act on the y subspace, and since matrices operating on different subspaces commute, this can be rearranged to:

$$-i\hbar X \otimes \delta^{(4)}_{\Delta y} = \left(\mathbb{I}_{n_x} \otimes Q^\dagger_{\delta_y}\right)\left(-i\hbar X \otimes D_{\delta_y}\right)\left(\mathbb{I}_{n_x} \otimes Q_{\delta_y}\right), \tag{3.105}$$

yielding a diagonalisation of the original matrix that can be used to apply an exponentiation of the original matrix to a vector with matrix-vector multiplications only in the x any y subspaces and not the total Hilbert space. Here, the matrix-vector multiplication in the x subspace is the identity, but more generally the above idea can be used to exponentiate any operator that can be written as the product of operators that act on different subspaces:

$$e^{A\otimes B} = \left(Q^\dagger_A \otimes Q^\dagger_B\right) e^{(D_A \otimes D_B)} \left(Q_A \otimes Q_B\right). \tag{3.106}$$

Since X is already diagonal, $\delta^{(4)}_{\Delta y}$ can be written as the sum of two block-diagonal matrices as described in [SECREF PARALLEL], allowing (3.105) to be evaluated as the sum of two terms $-i\hbar X \otimes \delta^{(4)}_{\Delta y} = -i\hbar(X \otimes \delta_{\text{even}} + X \otimes \delta_{\text{odd}})$, one for each of the block-diagonal matrices δ_{even} and δ_{odd}. The diagonalisation of each term then has matrix-vector products taking place in a space of the size of each block, that is, in the expression

$$-i\hbar X \otimes \delta_{\text{even}} = \left(\mathbb{I}_{n_x} \otimes Q^\dagger_{\text{even}}\right)\left(-i\hbar X \otimes D_{\text{even}}\right)\left(\mathbb{I}_{n_x} \otimes Q_{\text{even}}\right), \tag{3.107}$$

the unitary Q_{even} is also block-diagonal. This splitting allows for parallel application of the exponentiation of L_z to a vector as well as speeding up single-core computation on account of the smaller matrices, as described in section [SECREF].

However If a matrix that were not diagonal were present instead of X, such as another derivative operator, then if we wanted to split *both* matrices each into a sum

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

of two block-diagonal matrices, equation (3.106) would become *four* terms rather than two, and the flow of data for a parallel computation would be somewhat more complicated. For a term in the Hamiltonian that is a product of n operators acting on different subspaces, the number of terms obtained by splitting them all in this way grows exponentially with n. But, when all but one operator in the product is diagonal already, as is the case for angular momentum operators, then splitting can be done as normal.

To conclude this section, there are clear advantages to using finite differences as opposed to Fourier transforms when it comes to parallelisability, boundary conditions, and non-uniform grids, but if these are not a concern then both Fourier transforms and finite differences run at approximately equal speeds in practice, meaning one should use whichever is easiest to implement.

Many of these advantages of finite differences over Fourier methods are also enjoyed by the finite element discrete variable representation (FEDVR) method, discussed in the next section. Whilst it's also claimed that FEDVR has a number of advantages over finite-differences [19, 21], I'll argue that most of the comparisons don't stand up to scrutiny, and that finite differences are often still the right choice for the contexts in which FEDVR is argued to be superior.

3.4.3 The finite element discrete variable representation

Another method of spatial discretisation is the finite element discrete variable representation (FEDVR). As the name suggests, it is a finite element method, using a set of basis functions within each of a number of spatially distributed *elements*, with adjacent elements linked at their boundaries. Here I will not provide a self-contained description of the FEDVR method itself, more detail can be found in [14, p. 285] and specifically in the context of Bose–Einstein condensation, [19, 21]. I'll instead introduce the points that are relevant to the conclusion that I drew regarding the method for the purposes of time-evolving wavefunctions, which is that all practicalities considered, it is less useful than simple central finite differences for these problems.

As a finite element method, FEDVR divides space into a number of 'elements', joined to their neighbouring elements at their edges, within each of which the function being solved for is represented as a linear sum of basis functions. Finite element methods such as this are useful for problems with irregular boundary conditions, or requiring variable grid sizes, as the elements need not have the same size (area/volume, etc.) as each other, and parameters on which the accuracy of the numerical method depends can be varied from element to element, such that precision is high where it is needed and low where it is not. In addition, the basis functions used within the elements can be chosen so that conserved properties of the differential equation are also inherently conserved by the simulation resulting in a *geometric integrator*. The FEDVR method though does not make use of this possibility, although it can be used with split-step methods for time propagation, which preserve the wavefunction's norm.

Within each element in the FEDVR method, the wavefunction is represented as a linear sum of a set of polynomial basis functions, specially chosen to have some desirable properties. The polynomials are not orthogonal, but at a set of gridpoints, all but one of them is zero, meaning that as with the Fourier pseudospectral method, they can be used as a pseudospectral basis—computing how much of each polynomial to include in the representation of a wavefunction by using only the wavefunction's value at the gridpoint at which each polynomial is zero, and computing the matrix elements of operators using approximate integration based on a discrete sum taking into account only those same gridpoints.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

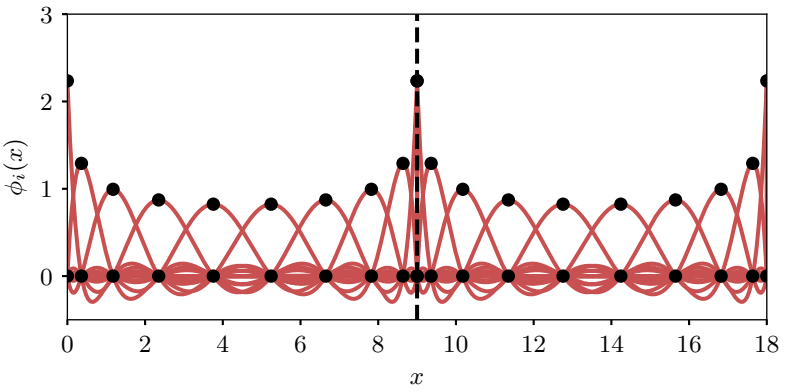

Figure 3.6: An example of the normalised, but non-orthogonal basis polynomials used in the finite-element discrete variable representation, shown here for ten-point Gauss–Lobatto quadrature and two elements. Note that each basis polynomial is nonzero at exactly one point and zero at all others, though it can be nonzero in between points. Outside of its element, a given basis function is defined to be zero, except for the so-called "bridge functions", which are nonzero in two adjacent elements but zero everywhere else, and surprisingly have a discontinuous derivative at the boundary—necessitating some care in evaluating matrix elements of differential operators.

So far (within each element at least) what I have described does not sound too different to the Fourier pseudospectral method. And it would be not different at all if the gridpoints were equally spaced, if one allowed the number of points to go to infinity, and one used the lowest order polynomials that were each zero at all points but one. But in the discrete variable representation—which is what the method used within each element of FEDVR is called when used by itself—the gridpoints are not equally spaced. Rather they are more densely packed toward the edges of the element, and least densely packed in the middle of each element. The locations of the gridpoints are chosen according to a *quadrature rule*, which is a method of approximating the definite integral of a function as a weighted sum of the function's values at a specific set of points. There are many quadrature rules, each with a different set of points and weights, but a common feature to them is that the points are more closely spaced at the edges of the integration region. In the context of the pseudospectral method, the chosen quadrature rule is what we are using when we are evaluating integrals based only on the function's values at discrete points. For equally spaced points, the weights are all equal, but for unequally spaced points they are not (and vary depending on the quadrature rule in use).

The use of unequally spaced points increases the accuracy of integrals evaluated this way, to the extent that the result will be exact if the integrand is a polynomial of degree less than a given degree that depends on the quadrature scheme. One would think that equally spaced points would produce the most accurate approximate integrals, but this is not true: approximating functions as polynomials equal to the value of the function at a discrete set of points is vulnerable to Runge's phenomenon—spurious oscillations in the approximation near the edges of the region[22]. The oscillations are minimised, however, if the density of points increases near the edge, specifically if the density of points approaches $1/\sqrt{1-x^2}$ for the normalised integraiton region $x \in [-1, 1]$ as the number of points goes to infinity [28].

By choosing a quadrature scheme with two points located exactly on each edge of the integration regions, such as *Gauss-Lobatto* quadrature [19], the elements of the FEDVR method can be joined together, with adjacent elements sharing these edge points (see

[22] Note that although finite differences are also based on polynomial interpolation, *central* finite differences are not vulnerable to Runge's phenomenon because the interpolated polynomials at the 'edge' of the region are never used for anything: a different polynomial is used for each point, with each polynomial and its derivatives only ever being evaluated at its central interpolation point.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

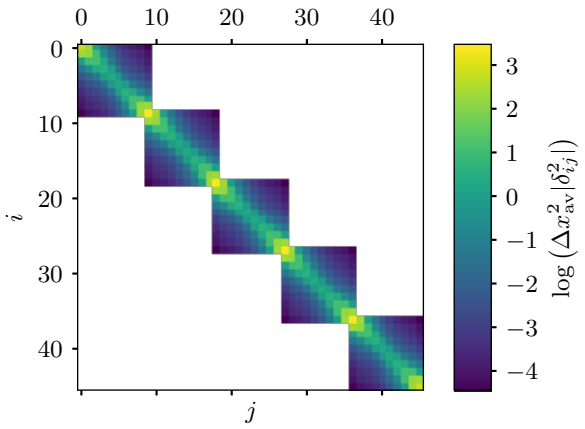

Figure 3.7: The matrix form of the second derivative operator in FEDVR, for five elements each with ten DVR points. The colour scale is logarithmic, showing the log of the absolute values of the matrix elements δ^2_{ij} scaled by Δx^2_{av}, where Δx_{av} is the average spacing between the unequally spaced grid points. Zero matrix elements are shown as white. One can see that the second derivative at each grid point is computed using only the points within the same element, except for the points shared by adjacent elements, at which the second derivative depends on points in both adjacent elements.

Figure 3.6). This allows one to represent a wavefunction as a list of coefficients for how much of each basis polynomial in each element contributes to it, with the extra condition that some of these coefficients—those corresponding to the edge points—must be equal in order to ensure the wavefunction is continuous across the boundaries between elements.

With some care taken in regard to the points shared between the elements, the FEDVR method provides one with a means of approximating wavefunctions as a finite sum of basis functions, and of approximately calculating the matrix elements of operators in this basis. After that, one can simply apply the operators to the state vectors in order to compute time derivatives, and propagate in time using fourth order Runge–Kutta (section 3.3) or similar, or one can exponentiate the operators, all at once or one at a time, as part of a split-step method (section 3.2.3). The FEDVR method does not require anything in particular about time propagation—like finite differences or the Fourier pseudospectral method, it is only a way of spatially discretising the differential equation you are trying to solve.

Now we arrive at what makes the FEDVR method exciting for those who want to massively parallelise their simulations. When one calculates the matrix representation of, say, a one-dimensional differential operator in the FEDVR basis, one gets matrices that look like Figure 3.7, With an almost-block-diagonal form.

This is because when the state vector is acted upon by some operator, the result for most points in the resulting vector depends only on the points in the input vector *within the same element*. The exceptions are the points shared between elements—for which the value in the output vector depends on the points in the input vector in *both* adjacent elements, which is why the matrix is not perfectly block diagonal.

Why is this exciting? Well, it means that the operator can be written like this:

[SUM OF MATRICES]

Which is very similar to the splitting of finite-difference matrices in section 3.2.3, and in the same way allows the matrix or its exponentiation to be efficiently applied to a state vector in parallel. The difference between this and the case of finite differences is the number of points of overlap between the two matrices, which corresponds to the amount of data that must be transferred between parallel threads or cluster nodes at each step

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

during a parallel computation. In finite differences, the number of points that must be exchanged at boundaries in each step or sub-step of whichever time propagation method is used is equal to the bandwidth of the matrix. In FEDVR, so long as the boundaries of the regions of space assigned to each thread or cluster node aligns with the border between two elements, *only one* point must be exchanged. Increasing the number of points within each element—but reducing the number of elements so as to keep the total number of points constant, decreases the discretisation error of the method, but still, only one point needs to be exchanged at boundaries. This contrasts with finite differences, for which increasing the order of the finite difference scheme increases the matrix bandwidth and hence increases the number points required to be exchanged at the boundaries. So it would seem that FEDVR ought to scale much better in parallel implementations, which is a large part of its appeal [19, 21].

However, I've noticed that when using both FEDVR and finite differences to simulate Bose–Einstein condensates using the Gross–Pitaevskii equation, smaller timesteps are required when using the FEDVR method [SHOULD I SHOW THIS EXPLICITLY?] in otherwise comparable setups. By this I mean that, without damping, there is a timestep size at which the GPE simulated using RK4 (which is a conditionally stable algorithm) is unstable and diverges[23]. Similarly in split-step methods, although the error is bounded, there is a timestep size at which the error rapidly grows to that bound and the wavefunction no longer approximately resembles the true solution. Below this threshold timestep, the error scales as $\mathcal{O}(\Delta t^4)$ for RK4, and $\mathcal{O}(\Delta t^4)$, $\mathcal{O}(\Delta t^2)$, $\mathcal{O}(\Delta t)$ for fourth, second, and first order split-step respectively, as expected. But in practice, all these methods are so accurate that one desires to use the largest timestep one can without this blowup (or soft-blowup in the case of the unitary split-step methods) occurring during the time interval one wants to simulate. And my observation was that the threshold timestep is always smaller for FEDVR than for finite differences or the Fourier method for determining spatial derivatives, necessitating smaller timesteps for stability or, in the case of the unitary methods, reasonable accuracy.

[23] Note that since the Gross–Pitaevskii equation is nonlinear, most definitions of stability do not apply to it, strictly speaking. As mentioned earlier in this chapter, I've noticed that the per-step error when simulating the GPE with RK4 does grow exponentially with time seemingly regardless of how small the stepsize is, but smaller step sizes slow the growth rate, and step sizes large enough to violate the linear stability criterion result in almost immediate blowup, making the linear stability criterion still of interest.

So why is this? For RK4, what is the stability criterion and why might FEDVR violate it more easily than finite differences? And how might we understand the sudden decrease in accuracy of the split-step methods at a similar threshold timestep size, despite them being unconditionally stable?

The stability critereon for RK4 when applied to a linear differential equation of the form:

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = A\psi, \tag{3.108}$$

where A is a matrix with all imaginary eigenvalues (as is the case for Hamiltonian evolution where $A = -\frac{\mathrm{i}}{\hbar}H$), is [29]:

$$\Delta t < \frac{2\sqrt{2}}{\rho(A)} \tag{3.109}$$

where $\rho(A)$ is the *spectral radius* of A, defined as the absolute value of the largest (by absolute value) eigenvalue of A.

The Gross–Pitaevskii equation is not linear, but we'll put that to the side for the moment—it will be relevant shortly. In the linear case of the Schrödinger wave equation, an upper bound for the spectral radius of A can be computed from the absolute eigenvalue from the kinetic term, plus the maximum absolute eigenvalue from the potential term. Using the Fourier method to compute spatial derivatives, the largest eigenvalue of the kinetic part of the Hamiltonian is that of the Nyquist mode with $k_{\text{Nyquist}} = \pi/\Delta x$ for

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

grid spacing Δx, leading to:

$$\rho(A) < \left| -\frac{\mathrm{i}}{\hbar} \frac{\hbar^2 k_{\text{Nyquist}}}{2m} \right| + \left| -\frac{\mathrm{i}}{\hbar} V(\boldsymbol{r}) \right|_{\max} \tag{3.110}$$

$$\Rightarrow \Delta t < \frac{2\sqrt{2}\hbar}{\frac{\hbar^2\pi^2}{2m\Delta x^2} + |V(\boldsymbol{r})|_{\max}}. \tag{3.111}$$

In the limit of small Δx, the kinetic term dominates and the stability criterion becomes:

$$\Delta t < \frac{4\sqrt{2}m\Delta x^2}{\pi^2\hbar}, \tag{3.112}$$

whereas in the limit of large potential:

$$\Delta t < \frac{2\sqrt{2}\hbar}{|V(\boldsymbol{r})|_{\max}}. \tag{3.113}$$

These results match our intuition somewhat in terms of dynamical phase evolution—eigenvalues of the Hamiltonian are energies and determine how quickly the elements of the state vector accumulate dynamical phase. For each circle around the complex plane that a dynamical phase of 2π entails, we expect to require at least a few timesteps to resolve said circle, and the above shows that 'a few' means $2\sqrt{2}$. When this dynamical phase evolution is in Fourier space, an accumulated phase of 2π will appear in real space as that component having propagated a distance one wavelength, so the intuition here is that several timepoints are required in the time interval during which the fastest wave (also the Nyquist mode) propagates a distance of one wavelength at its phase velocity.

As a sidenote, if it is known which term of the Hamiltonian dominates the stability criterion, that term can be removed by use of an *interaction picture* [CITE, SECREF?], essentially treating the dynamical phase evolution due to that term analytically. And if the term is not constant, but is *almost* constant, one can still treat *most* of the dynamical phase evolution analytically, transforming the differential equation onto one with much smaller eigenvalues for that term, in both cases allowing one to take potentially much larger timesteps due to the above reasoning. Fourth order Runge–Kutta in the interaction picture (RK4IP) [30] uses an interaction picture to treat the kinetic term of the Schrödinger or Gross–Pitaevskii equation analytically, whereas my 'fourth order Runge–Kutta in an instantaneous, local interaction picture' method presented in section 3.6 removes (most of) the potential term. Both methods allow larger timesteps to be taken, but in different circumstances depending on which term is dominating the Hamiltonian.

The problem with FEDVR then, is that it requires smaller timesteps than finite differences because its kinetic energy operator has larger eigenvalues than an equally accurate finite difference method. How much larger?

In order to make a fair comparison, we need to know how many DVR basis functions are required per element in order to compute equally accurate second derivatives as a given finite difference scheme. With N points per element, FEDVR represents the state vector within each element in a spectral basis of polynomials up to degree $N-1$. Therefore a polynomial of degree $N-1$ or less can be represented exactly, any other function is subject to truncation error[24]. If one varies the number of points per element, varying the number of elements in order to keep the total number of points constant, the truncation error in representing an arbitrary function in this basis is therefore $\mathcal{O}(\Delta x^N)$, where Δx is either the size of each element, or equivalently the average spacing between grid points (these two differing only by a constant factor if the total number of points is held constant).

In FEDVR the derivative operator, regardless of whether its matrix elements are computed with integrals or the quadrature rule, is exact [19]. The relevant error in a derivative

[24] Note that this truncation error is distinct from that of evaluating *integrals* using the Gauss–Lobatto quadrature rule, which is exact for integrands that are polynomials of degree $2N-2$ or less.

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

of a wavefunction is therefore determined by this truncation error of representing it in the spectral basis in the first place:

$$\psi_{\text{approx}}(x_i) = \psi_{\text{exact}}(x_i) + \mathcal{O}(\Delta x^N) \tag{3.114}$$

$$\Rightarrow \psi''_{\text{approx}}(x_i) = \psi''_{\text{exact}}(x_i) + \mathcal{O}(\Delta x^{N-2}). \tag{3.115}$$

Central finite difference approximations to second derivatives on the other hand have error $\mathcal{O}(\Delta x^{N-1})$ where N is the total number of points used to compute the derivative at each point (for example the three-point central finite difference rule for second derivatives has error $\mathcal{O}(\Delta x^2)$, the five-point rule is accurate to $\mathcal{O}(\Delta x^4)$, etc). With this knowledge we can translate our question

> Which is less computationally intensive to simulate the GPE or Schrödinger wave equation: m^{th}-order accurate FEDVR or m^{th}-order accurate finite differences?

to

> which is less computationally intensive to simulate the GPE or Schrödinger wave equation: $m + 2$ points-per-element FEDVR or $m + 1$ point central finite differences?

The argument in favour of FEDVR is that as m grows, finite differences require an increasing number of points to be exchanged at the boundaries between cluster nodes/threads etc in a parallel implementation, whereas so long as the border between spatial regions allocated to different cluster nodes aligns with the border between elements, only one point need be exchanged per timestep in FEDVR, no matter how many points per element there are. Therefore, in the limit of high accuracy, FEDVR wins, it seems.

The problem with this argument is that it compares only the amount of work that needs to be done *per timestep*. But, since the allowed timestep size required for stability (at least for fourth order Runge–Kutta, I'll argue shortly why I think this generalises to other timestepping schemes as well) depends on the spectral radius of the kinetic energy operator, and the kinetic energy operator is not the same as m grows, more work is needed to show which method is least computationally expensive per unit *simulation time*.

The spectral radius of the kinetic energy operator when approximated using central finite differences does not grow without limit as the number of points used to compute derivatives increases—indeed, its eigenspectrum converges to that of the Fourier method (since the Fourier method can be considered the limit of 'infinite order' finite differences [25]), with maximum eigenvalue equal to the kinetic energy of the Nyquist mode. The kinetic energy operator approximated with FEDVR on the other hand does not have a bounded spectral radius as one increases the number of points per element whilst holding the total number of gridpoints constant. Instead its spectral radius increases quadratically with the number of points (equivalently with the order of accuracy of the derivatives), as shown in Figure 3.8.

This result ought to be expected, at least qualitatively. The density of points in FEDVR is higher toward the edges of the elements than in their centres, and if one increases the number of points per element whilst decreasing the number of elements so as to hold the total number of points approximately constant, the smallest spacing between any two adjacent points will be inversely proportional to the number of points per element. We already know that in high order finite differences or the Fourier method, the spectral radius of the kinetic energy operator is simply the kinetic energy of the Nyquist mode, and being the mode described by a wave with half a wavelength spanning one grid spacing, its kinetic energy is proportional to the square of the grid spacing. It not therefore surprising

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

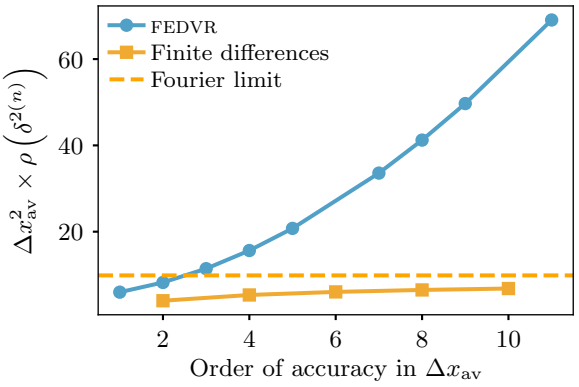

Figure 3.8: Scaling of the spectral range (maximum absolute eigenvalue) of the second derivative operator with respect to the order of accuracy for finite differences and FEDVR. Results were numerically computed by constructing a 721×721 matrix (chosen to allow various combinations of number of DVR points per element whilst holding total number of points constant) for each order of accuracy and numerically diagonalising. The spectral range for FEDVR can be seen to scale quadratically with the order of accuracy, whereas for finite differences the maximum eigenvalue approaches that of the Fourier method.

that in FEDVR when the grid spacing is linearly smaller—albeit locally—that the spectral radius of the kinetic energy operator is quadratically larger. The same result as above could be obtained by asking: "what's the smallest grid spacing, and what is the kinetic energy of the Nyquist mode corresponding to that grid spacing"?, and then declaring the stability criterion to be that one must have at least a few points per period of the frequency corresponding to that energy.

Since the spectral radius of FEDVR operators grows faster with increasing order of accuracy in derivative operators than finite differences, FEDVR requires smaller timesteps for stability when used with RK4. Even at low order accuracy, finite difference operators have smaller spectral radii, and so at all levels of accuracy RK4 is stable with finite differences for larger timesteps than with FEDVR. In the limit of high accuracy then, compared to finite differences FEDVR requires *quadratically* more timesteps to be taken for stability, whereas it requires only *linearly* fewer points to be transferred at the boundaries between threads or cluster nodes per timestep. The larger number of timesteps is therefore the dominant effect, more than cancelling out the benefit of having to transfer fewer points at boundaries per timestep than with finite differences. Indeed, even if transferring data between nodes or threads is the bottleneck of a parallel simulation, *more* points are being transferred all up with FEDVR—they are just spread out over more timesteps.

That's fourth order Runge–Kutta. What about split-step methods? The split-step methods discussed in section [SECREF] are unconditionally stable and have bounded error. However, their error can still be large enough to make the results not useful. As shown in (3.35), the leading error term in first-order split-step is $1/2\hbar^2[V,K]\Delta t^2$ where $[V,K]$ is the commutator between the (discretised) kinetic and potential energy operators. The leading term of V is proportional to the discretised $\hat{x}$ operator X, and the kinetic energy operator is proportional to an n^{th} order accurate discretised second derivative operator $\delta^{2(n)}$. Therefore the error per timestep scales with $[X,\delta^{2(n)}]\Delta t^2$. Although this expression is not bounded, the error in first-order split-step is nonetheless bounded because this error appears as the argument of a complex exponential. When expanding the complex exponential as its Taylor series, this error term is the leading term, but when it grows it leads merely to a complex exponential with unbounded phase error, not to

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

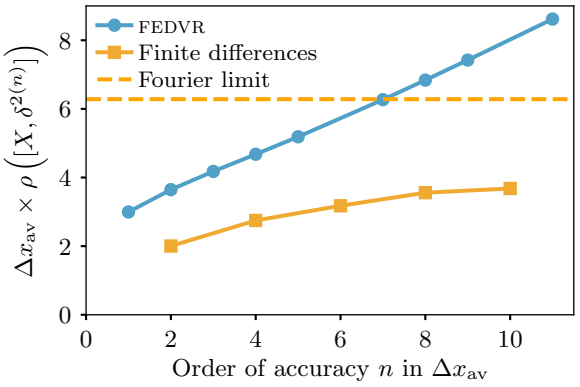

Figure 3.9: TODO CAPTION once mentioned in text about unitary methods

unbounded absolute error. Nonetheless unbounded phase error still makes the results of a simulation unlikely to be useful.

The largest (by absolute value) eigenvalue of this commutator sets the rate at which erroneous dynamical phase is accumulated by some eigenstate of the commutator. When this phase becomes comparable to 2π per timestep, the simulation has clearly lost any semblance of accuracy, despite still being technically "stable". Therefore as before, the spectral radius of the matrix of this commutator can be used to define a *pseudo*-stability criterion, such that when the spectral radius of $[V, K]\Delta t^2/2\hbar^2$ becomes comparable to unity, the error will dominate the simulation results. Although the value of the spectral radius will depend on the details of the potential energy operator V for the particular problem, we can still ask how it scales with increasing order of accuracy of K given that X is the leading term of V. This is done in Figure 3.9, comparing the spectral radius of the commutator when using derivative operators of various orders in both finite differences and FEDVR approximations.

The result is that as with RK4, the pseudo-stability criterion allows at all orders of accuracy for larger timesteps when using finite differences than when using FEDVR, and that the required timestep is bounded from below when increasing the order of accuracy of finite differences, but can become arbitrarily small for increasing accuracy of FEDVR. However the situation is not so dire for FEDVR when used with split-step as it is with RK4 timestepping. Because the error term in split-step is this commutator multiplied by Δt^2, the timestep required for pseudo-stability scales inversely with the *square root* of the spectral radius of the commutator—not with the spectral radius itself as with RK4. Furthermore the spectral radius of the FEDVR commutator increases only *linearly* with increasing order of accuracy, not quadratically as with RK4. Therefore if transferring data between nodes (with time cost proportional to the amount of data transferred) is the bottleneck of one's simulation, then with increasing accuracy, FEDVR *does* win out. For high enough order of accuracy n, a factor of $\sqrt{n}$ more timesteps are needed with FEDVR compared to finite differences, but a factor of n *fewer* points are sent per timestep. This results in a factor of $1/\sqrt{n}$ fewer points being sent per unit simulation time with FEDVR as compared to finite differences. The question of which method is faster then comes down to which is more costly: extra timesteps, or extra data transfer? Using FEDVR will save you a factor of $\sqrt{n}$ in data transfer at a cost of a factor of $\sqrt{n}$ in the number of timesteps. Given InfiniBand interconnects on cluster computers with tens of gigabits per second of bandwidth, and latency that does not scale with the amount of data, I suspect that a $\sqrt{n}$ increase in cost of processing within nodes/threads is almost always the larger cost to pay.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

Therefore I suspect the benefits of FEVDR for parallel simulations is limited.

[SECTION TITLE NONLINEARITY]

The above arguments about stability all disregarded the nonlinearity of the Gross–Pitaevskii equation. Although the stability of a numerical method is very difficult to analyse for nonlinear differential equations, there is a simple argument for heuristically putting an upper bound on the timestep required in order to accurately model the effect of the GPE's nonlinear term. Essentially, density waves in the condensate propagate not at the phase velocity of the condensate wavefunction, but at its group velocity. The group velocity of the shortest possible wavelength in the system therefore sets an upper bound for the propagation of information due to the nonlinear term. In order to correctly model this term, timesteps must be short enough that one's numerical method is evaluating the nonlinear term frequently enough that fast moving density waves can't 'skip' gridpoints in between evaluations. If the smallest wavelength is that of the Nyquist mode, or for nonuniform grids the mode whose wavelength is twice the smallest grid spacing, then the maximum group velocity is:

$$v_g(k_{max}) = \frac{\partial E_K(k_{max})}{\partial p(k_{max})} = \frac{\hbar k_{max}}{m} \frac{\pi\hbar}{m\Delta x_{min}} \tag{3.116}$$

Asking how long it takes a wave to move a distance of Δx_{min} at this velocity then results in what I'm calling the *dispersion timescale*:

$$\tau_d = \frac{m\Delta x_{min}^2}{\pi\hbar}. \tag{3.117}$$

The numerical method being used is now fairly irrelevant: one must evaluate the nonlinear term at least as often as every τ_d in order to be able to model it accurately, otherwise density waves may move past each other faster than they can be resolved by the timestepping, and their interference patterns will be aliased by the too-slow timestepping, resulting in incorrect nonlinear dynamics. Note that up to constant factors, this criterion for accurate modelling is the same as the stability criterion (3.112) for RK4 when the kinetic term dominates the Hamiltonian. Therefore although first-order split-step as argued above appears to be more forgiving to FEDVR than RK4 is, this is no longer the case when modelling the GPE as opposed to the linear Schroödinger wave equation, and although I didn't extend the argument in the previous section to second or fourth order split-step (figuring out which commutator is the leading term is much more involved), they too are subject to the same requirement to sample the nonlinear term this frequently, even if their linear pseudo-stability region might be larger.

[CONCLUSION SUBSEC]

This leaves us with little remaining benefit to FEDVR over finite differences in my opinion. The argument that FEDVR scales better for parallel computation is unconvincing as argued above. The fact that it allows nonuniform grids is also not unique—central finite differences allow for nonuniform grids as well [CITE]. Reference [CITE THE PRE] compares the accuracy of finite differences to FEDVR in the context of computing the groundstate energy of a harmonic oscillator using imaginary time evolution (see section [SECREF]), but only uses second-order finite differences in the comparison whereas higher-order FEDVR schemes are used. If this restriction was in order to hold the number of points transferred at boundaries between parallel computing units constant (since second-order finite differences require, like FEDVR, only one), then this was not a fair comparison as the number of points transferred does not limit computation time as I've shown–had the authors included comparisons with higher order finite differencing schemes, they would have resulted in comparable accuracy to the matching order FEDVR runs, but completed in less time owing to the larger timesteps allowed by the increased range of stability of finite differences.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

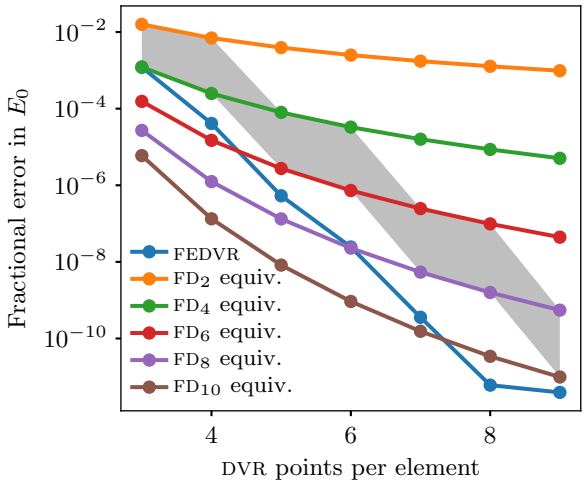

Figure 3.10: TODO CAPTION once mentioned in text

Perhaps for problems with certain unusual boundary conditions or strangely shaped regions, or where one can construct a geometric integrator that conserves some quantities of interest other than merely the wavefunction norm, FEDVR may still make sense. But unless there is a compelling reason, it seems that simple finite differences, as naïve as it might seem at first, remains a practical choice for highly accurate and potentially massively parallelised simulations of the GPE.

[TODO: CONSIDER SUBSECTIONS]

[TODO: RENAME THIS SECTION "STABILITY AND FEDVR" or similar?]

3.4.4 Harmonic oscillator basis functions

[Another way of discretising space. Not a grid at all. Useful in, well, harmonic and approximately harmonic potentials, or (well, equivalently) when Hamiltonian is sparse in the SHO eigenbasis. Otherwise not great - without FFTs, applying the Hamiltonian means big, dense matrices - slow.]

3.5 Finding ground states

3.5.1 Imaginary time evolution

[Summarise ITEM]

3.5.2 Successive over-relaxation

[Summarise SOR]

3.5.3 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

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

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.

3.6 Fourth order Runge–Kutta in an instantaneous local interaction picture

Consider the differential equation for the components of a state vector $|\psi(t)\rangle$ in a particular basis with basis vectors $|n\rangle$. This might simply be the Schrödinger equation, or perhaps some sort of nonlinear or other approximate, effective or phenomenological equation not corresponding to pure Hamiltonian evolution. Though they may have additional terms, such equations are generally of the form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle n|\psi(t)\rangle = -\frac{i}{\hbar}\sum_m \langle n|\hat{H}(t)|m\rangle\,\langle m|\psi(t)\rangle\,, \tag{3.118}$$

where $\langle n|\hat{H}(t)|m\rangle$ are the matrix elements in that basis of the Hamiltonian $\hat{H}(t)$, which in general can be time dependent, or even a function of $|\psi(t)\rangle$, depending on the exact type of equation in use. If $\hat{H}(t)$ is almost diagonal in the $|n\rangle$ basis, then the solution to (3.118) is dominated by simple dynamical phase evolution, that is:

$$|\psi(t)\rangle \approx \sum_m e^{-\frac{i}{\hbar}E_m t}|m\rangle\,, \tag{3.119}$$

where E_m is the energy eigenvalue corresponding to the eigenstate $|m\rangle$.

A transformation into an interaction picture (IP) [31, p. 318] is commonly used to treat this part of the evolution analytically, before solving the remaining dynamics with further analytics or numerics. For numerical methods, integration in the interaction picture allows one to use larger integration timesteps, as one does not need to resolve the fast oscillations around the complex plane due to this dynamical phase.

Choosing an interaction picture typically involves diagonalising the time-independent part of a Hamiltonian, and then proceeding in the basis in which that time-independent part is diagonal. However, often one has a good reason to perform computations in a different basis, in which the time independent part of the Hamiltonian is only approximately diagonal,[25] and transforming between bases may be computationally expensive (involving large matrix-vector multiplications). Furthermore, the Hamiltonian may change sufficiently during the time interval being simulated that the original time-independent Hamiltonian no longer dominates the dynamics at later times. In both these cases it would still be useful to factor out the time-local oscillatory dynamics in whichever basis is being used, in order to avoid taking unreasonably small timesteps.

[25] For example, a spatial basis which allows for partitioning the integration region over multiple nodes on a cluster or cores on a GPU.

To that end, suppose we decompose $\hat{H}(t)$ into diagonal and non-diagonal (in the $|n\rangle$ basis) parts at each moment in time:

$$\hat{H}(t) = \hat{H}_{\text{diag}}(t) + \hat{H}_{\text{nondiag}}(t), \tag{3.120}$$

and use the diagonal part at a specific time $t = t'$ to define a time-independent Hamiltonian:

$$\hat{H}_0^{t'} = \hat{H}_{\text{diag}}(t'), \tag{3.121}$$

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

which is diagonal in the $|n\rangle$ basis. We can then use then use $\hat{H}_0^{t'}$ to define an interaction picture state vector:

$$|\psi_{\mathrm{I}}^{t'}(t)\rangle = e^{\frac{i}{\hbar}(t-t')\hat{H}_0^{t'}} |\psi(t)\rangle\,, \tag{3.122}$$

which obeys the differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi_{\mathrm{I}}^{t'}(t)\rangle = e^{\frac{i}{\hbar}(t-t')\hat{H}_0^{t'}} \frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle + \frac{i}{\hbar}\hat{H}_0^{t'}|\psi_{\mathrm{I}}^{t'}(t)\rangle\,, \tag{3.123}$$

where:

$$|\psi(t)\rangle = e^{-\frac{i}{\hbar}(t-t')\hat{H}_0^{t'}} |\psi_{\mathrm{I}}^{t'}(t)\rangle \tag{3.124}$$

is the original Schrödinger picture (SP) state vector.

This transformation is exact, no approximations or assumptions have been made. If indeed the dynamics of $|\psi(t)\rangle$ in the given basis are dominated by fast oscillating dynamical phases, that is, the diagonals of $\hat{H}_{\mathrm{diag}}(t)$ are much greater than all matrix elements of $\hat{H}_{\mathrm{nondiag}}(t)$ in the $|n\rangle$ basis, then solving the differential equation (3.123) for $|\psi_{\mathrm{I}}^{t'}(t)\rangle$ should allow one to use larger integration timesteps than solving (3.118) directly. And if not, then it should do no harm other than the (small) computational costs of computing some extra scalar exponentials.

Equation (3.122) defines an *instantaneous* interaction picture, in that it depends on the dynamics at a specific time $t = t'$, and can be recomputed repeatedly throughout a computation in order to factor out the fast dynamical phase evolution even as the oscillation rates change over time. It is *local* in that $H_0^{t'}$ is diagonal in the $|n\rangle$ basis, which means that transformations between Schrödinger picture and interaction picture state vectors involves ordinary, elementwise exponentiation of vectors, rather than matrix products. Thus (3.122), (3.123) and (3.124) can be written componentwise as:

$$\langle n|\psi_{\mathrm{I}}^{t'}(t)\rangle = e^{i(t-t')\omega_n^{t'}} \langle n|\psi(t)\rangle\,, \tag{3.125}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle n|\psi_{\mathrm{I}}^{t'}(t)\rangle = e^{i(t-t')\omega_n^{t'}} \frac{\mathrm{d}}{\mathrm{d}t}\langle n|\psi(t)\rangle + i\omega_n^{t'}\langle n|\psi_{\mathrm{I}}^{t'}(t)\rangle\,, \tag{3.126}$$

and:

$$\langle n|\psi(t)\rangle = e^{-i(t-t')\omega_n^{t'}} \langle n|\psi_{\mathrm{I}}^{t'}(t)\rangle\,, \tag{3.127}$$

where we have defined:

$$\omega_n^{t'} = \frac{1}{\hbar}\langle n|\hat{H}_0^{t'}|n\rangle \tag{3.128}$$

This is in contrast to fourth order Runge–Kutta in the interaction picture (RK4IP) [30], in which the interaction picture uses the Fourier basis and thus transforming to and from it involves fast Fourier transforms (FFTs). RK4IP was developed to augment computations in which FFTs were already in use for evaluating spatial derivatives, and so its use of FFTs imposes no additional cost. Nonetheless, an interaction picture based on the kinetic term of the Schrödinger equation (which is the term of the Hamiltonian that RK4IP takes as its time-independent part) may not be useful if that term does not dominate the Hamiltonian, as in the case of a Bose–Einstein condensate in the Thomas–Fermi limit. We compare the two methods below.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

3.6.1 Algorithm

The *fourth order Runge–Kutta in an instantaneous local interaction picture* RK4ILIP algorithm is now obtained by using (3.122) to define a new interaction picture at the beginning of each fourth-order Runge–Kutta (RK4) integration timestep. The differential equation and initial conditions supplied to the algorithm are in the ordinary Schrödinger picture, and the interaction picture is used only within a timestep, with the Schrödinger picture state vector returned at the end of each timestep. Thus differential equations need not be modified compared to if ordinary RK4 were being used, and the only modification to calling code required is for a function to compute and return $\omega_n^{t'}$.

Being based on fourth order Runge–Kutta integration, this new method enjoys all the benefits of a workhorse method that is time-proven, and—as evidenced by its extremely widespread use—at a sweet-spot of ease of implementation, accuracy, and required computing power [32].

Below is the resulting algorithm for performing one integration timestep. It takes as input the time t_0 at the start of the timestep, the timestep size Δt, an array $\boldsymbol{\psi}_0$ containing the components $\{\langle n|\psi(t_0)\rangle\}$ of the state vector at time t_0, a function $F(t, \boldsymbol{\psi})$ which takes a time and (the components of) a state vector and returns an array containing the time derivative of each component, and a function $G(t, \boldsymbol{\psi})$ which takes the same inputs and returns an array containing the interaction picture oscillation frequency ω_n for each component at that time.

For example, for the case of the Gross–Pitaevskii equation [33] in the spatial basis $\psi(\boldsymbol{r}, t) = \langle \boldsymbol{r}|\psi(t)\rangle$, these would be:

$$F(t, \psi(\boldsymbol{r}, t)) = -\frac{i}{\hbar}\Big[\underbrace{-\frac{\hbar^2}{2m}\nabla^2}_{\hat{H}_{\text{nondiag}}} + \underbrace{V(\boldsymbol{r}, t) + g|\psi(\boldsymbol{r}, t)|^2}_{\hat{H}_{\text{diag}}}\Big]\psi(\boldsymbol{r}, t), \tag{3.129}$$

and

$$G(t, \psi(\boldsymbol{r}, t)) = \frac{1}{\hbar}\Big[\underbrace{V(\boldsymbol{r}, t) + g|\psi(\boldsymbol{r}, t)|^2}_{\hat{H}_{\text{diag}}}\Big]. \tag{3.130}$$

Note that each symbol in bold in the algorithm below denotes an array containing one element for each basis vector $|n\rangle$, subscripts denote the different stages of RK4, and all arithmetic operations between arrays are elementwise[26] The only opportunity for non-elementwise operations to occur is within F, which contains the details (via $\hat{H}_{\text{nondiag}}$) of any couplings between basis states for whatever system of equations is being solved, for example, using FFTs or finite differences to evaluate the Laplacian in (3.129).

[26] For example, the expression $\boldsymbol{a} \leftarrow e^{-i\boldsymbol{\omega}\Delta t}\boldsymbol{b}$ indicates that for all n, $a_n \leftarrow e^{-i\omega_n\Delta t}b_n$, where a_n denotes the n^{th} element of $\boldsymbol{a}$ etc.

Algorithm 1 RK4ILIP

function RK4ILIP(t_0, Δt, $\boldsymbol{\psi}_0$, F)

$\boldsymbol{f}_1 \leftarrow F(t_0, \boldsymbol{\psi}_0)$ ▷ First evaluation of Schrödinger picture DE

$\boldsymbol{\omega} \leftarrow G(t_0, \boldsymbol{\psi}_0)$ ▷ Oscillation frequencies: $\hbar\omega_n = \langle n|\hat{H}_{\text{diag}}(t_0)|n\rangle$

$\boldsymbol{k}_1 \leftarrow \boldsymbol{f}_1 + i\boldsymbol{\omega}\boldsymbol{\psi}_0$ ▷ Evaluate (3.126) with $t - t' = 0$

$\boldsymbol{\phi}_1 \leftarrow \boldsymbol{\psi}_0 + \boldsymbol{k}_1\frac{\Delta t}{2}$ ▷ First RK4 estimate of IP state vector, at $t = t_0 + \frac{\Delta t}{2}$

$\boldsymbol{\psi}_1 \leftarrow e^{-i\boldsymbol{\omega}\frac{\Delta t}{2}}\boldsymbol{\phi}_1$ ▷ Convert first estimate back to SP with (3.127)

$\boldsymbol{f}_2 \leftarrow F(t_0 + \frac{\Delta t}{2}, \boldsymbol{\psi}_1)$ ▷ Second evaluation of Schrödinger picture DE

$\boldsymbol{k}_2 \leftarrow e^{i\boldsymbol{\omega}\frac{\Delta t}{2}}\boldsymbol{f}_2 + i\boldsymbol{\omega}\boldsymbol{\phi}_1$ ▷ Evaluate (3.126) with $t - t' = \frac{\Delta t}{2}$

$\boldsymbol{\phi}_2 \leftarrow \boldsymbol{\psi}_0 + \boldsymbol{k}_2\frac{\Delta t}{2}$ ▷ Second RK4 estimate of IP state vector, at $t = t_0 + \frac{\Delta t}{2}$

$\boldsymbol{\psi}_2 \leftarrow e^{-i\boldsymbol{\omega}\frac{\Delta t}{2}}\boldsymbol{\phi}_2$ ▷ Convert second estimate back to SP with (3.127)

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

11: $\boldsymbol{f}_3 \leftarrow F(t_0 + \frac{\Delta t}{2}, \boldsymbol{\psi}_2)$ ▷ Third evaluation of Schrödinger picture DE
12: $\boldsymbol{k}_3 \leftarrow e^{i\boldsymbol{\omega}\frac{\Delta t}{2}}\boldsymbol{f}_3 + i\boldsymbol{\omega}\boldsymbol{\phi}_2$ ▷ Evaluate (3.126) with $t - t' = \frac{\Delta t}{2}$
13: $\boldsymbol{\phi}_3 \leftarrow \boldsymbol{\psi}_0 + \boldsymbol{k}_3\Delta t$ ▷ Third RK4 estimate of IP state vector, at $t = t_0 + \Delta t$
14: $\boldsymbol{\psi}_3 \leftarrow e^{-i\boldsymbol{\omega}\Delta t}\boldsymbol{\phi}_3$ ▷ Convert third estimate back to SP with (3.127)
15: $\boldsymbol{f}_4 \leftarrow F(t_0 + \Delta t, \boldsymbol{\psi}_3)$ ▷ Fourth evaluation of Schrödinger picture DE
16: $\boldsymbol{k}_4 \leftarrow e^{i\boldsymbol{\omega}\Delta t}\boldsymbol{f}_4 + i\boldsymbol{\omega}\boldsymbol{\phi}_3$ ▷ Evaluate (3.126) with $t - t' = \Delta t$
17: $\boldsymbol{\phi}_4 \leftarrow \boldsymbol{\psi}_0 + \frac{\Delta t}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$ ▷ Fourth RK4 estimate, at $t = t_0 + \Delta t$
18: $\boldsymbol{\psi}_4 \leftarrow e^{-i\boldsymbol{\omega}\Delta t}\boldsymbol{\phi}_4$ ▷ Convert fourth estimate back to SP with (3.127)
19: **return** $\boldsymbol{\psi}_4$ ▷ Return the computed SP state vector at $t = t_0 + \Delta t$
20: **end function**

Note on imaginary time evolution

When RK4ILIP is used for imaginary time evolution (ITE) [34], the oscillation frequencies $\boldsymbol{\omega}$ may have a large imaginary part. If the initial guess is different enough from the ground state, then the exponentials in (3.125), (3.126) and (3.127) may result in numerical overflow. To prevent this, one can define a clipped copy of $\boldsymbol{\omega}$,

$$\boldsymbol{\omega}_{\text{clipped}} = \text{Re}(\boldsymbol{\omega}) + i\begin{cases} -\frac{\log X}{\Delta t} & \text{Im}(\boldsymbol{\omega})\Delta t < -\log X \\ \text{Im}(\boldsymbol{\omega}) & -\log X \le \text{Im}(\boldsymbol{\omega})\Delta t \le \log X \\ \frac{\log X}{\Delta t} & \text{Im}(\boldsymbol{\omega})\Delta t > \log X \end{cases}, \tag{3.131}$$

where X is very large but less than the largest representable floating-point number, and use $\boldsymbol{\omega}_{\text{clipped}}$ in the exponents instead. In the below results I used RK4ILIP with ITE to smooth initial states of a Bose–Einstein condensate after a phase printing, and performed clipping with [27] $\log X = 400$.

[27] 400 being about half the largest (base e) exponent representable in double-precision floating point.

This clipped version of $\boldsymbol{\omega}$ should be used in all exponents in the above algorithm, but only in exponents—not in the second term of (3.126). If it is used everywhere then all we have done is chosen a different (less useful) interaction picture, and the algorithm will still overflow. By clipping only the exponents, we produce temporarily "incorrect" evolution[28], limiting the change in magnitude of each component of the state vector to a factor of X per step (remembering that X is very large). This continues for the few steps that it takes ITE to get all components of the state vector to within a factor of X of the ground state, after which no clipping is necessary and convergence to the ground state proceeds as normal, subject to the ordinary limitations on which timesteps may be used with ITE.

[28] Of no concern since we are using ITE as a relaxation method, and are not interested in intermediate states. Only the final state's correctness concerns us.

3.6.2 Domain of improvement over other methods

For simulations in the spatial basis, RK4ILIP treats the spatially local part of the Hamiltonian analytically to first order, and hence can handle larger potentials than ordinary RK4. However, since a global energy offset can be applied to any potential with no physically meaningful change in the results, ordinary RK4 can also handle large potentials — if they are large due a a large constant term which can simply be subtracted off.

So RK4ILIP is only of benefit in the case of large *spatial variations* in the potential. Only one constant can be subtracted off potentials without changing the physics — subtracting a spatially varying potential would require modification of the differential equation in the manner of a gauge transformation in order to leave the system physically unchanged[29].

[29] Though a numerical solution based on analytically gauging away potentials at each timestep might be equally as fruitful as RK4ILIP.

However that's not quite all: large spatial variation in potentials often comes with the prospect of the potential energy turning into kinetic energy, in which case RK4ILIP is

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

Method	RK4	RK4IP	RK4ILIP	FSS
Error	$\mathcal{O}(\Delta t^4)$	$\mathcal{O}(\Delta t^4)$	$\mathcal{O}(\Delta t^4)$	$\mathcal{O}(\Delta t^2)$
FFTs per step	4	4	4	2
Large ΔV	No	No	Yes	Yes
Large kinetic term	No	Yes	No	Yes
Arbitrary operators	Yes	Yes†	Yes	No
Locally parallelisable	Yes	No	Yes	No
Arbitrary boundary conditions	Yes	No	Yes	No

Table 3.2: Advantages and disadvantages of four timestepping methods for simulating Bose–Einstein condensates. *Large* ΔV refers to whether the method can simulate potentials that vary throughout space by an amount larger than the energy scale $2\pi\hbar/\Delta t$ associated with the simulation timestep Δt. *Arbitrary operators* refers to whether the method permits operators that are not diagonal in either the spatial or Fourier basis, such as angular momentum operators. *Locally parallelisable* means the method can be formulated so as to use only spatially nearby points in evaluating operators, and thus is amenable to parallelisation by splitting the simulation over multiple cores in the spatial basis. † Whilst one can include arbitrary operators within the RK4IP method, only operators diagonal in Fourier space can be analytically treated the way RK4IP treats the kinetic term, and so there is no advantage for these terms over ordinary RK4.

also of little benefit, since in order to resolve the dynamical phase due to the large kinetic term, it would require timesteps just as small as those which ordinary RK4 would need to resolve the dynamical phase evolution from the large potential term.

This leaves RK4ILIP with an advantage only in the case of large spatial variations in the potential that do not lead to equally large kinetic energies. Hence the examples I show in the next section are ones in which the condensate is trapped in a steep potential well—the trap walls are high and hence involve large potentials compared to the interior, but do not lead to large kinetic energies because the condensate is trapped close to its ground state.

The Fourier split-step (FSS) method [35] (see section [TODO]) also models dynamical phases due to the potential analytically to low order. As such it is also quite capable of modeling large potentials. However, it requires that all operators be diagonal in either the spatial basis or the Fourier basis [35]. Therefore BECs in rotating frames, due to the Hamiltonian containing an angular momentum operator, are not amenable to simulation with FSS[30].

This use of FFTs in both the FSS and RK4IP methods necessarily imposes periodic boundary conditions on a simulation, which may not be desirable. By contrast, if different boundary conditions are desired, finite differences instead of FFTs can be used to evaluate spatial derivatives in the RK4 and RK4ILIP methods, so long as a sufficiently high-order finite difference scheme is used so as not to unacceptably impact accuracy.

Along with the ability to impose arbitrary boundary conditions, finite differences require only local data, that is, only points spatially close to the point being considered need be known in order to evaluate derivatives there. This makes finite differences amenable to simulation on cluster computers [36, p. 100], with only a small number of points (depending on the order of the scheme) needing to be exchanged at node-boundaries each step. By contrast, FFT based derivatives require data from the entire spatial region. Whilst this can still be parallelised on a GPU, where all the data is available, it cannot be done on a cluster without large amounts of data transfer between nodes [27]. Thus, RK4 and RK4ILIP, being implementable with finite difference schemes, are considerably friendlier to cluster computing.

Table 3.2 summarises the capabilities of the four methods considered in the following results section. RK4ILIP is the only method capable of modelling a large spatial variation in the potential term whilst being locally parallelisable, and supporting arbitrary operators

[30] Split-step with more than these two bases is however possible in other schemes such as the finite element discrete variable representation [21]—each operator can be diagonalised and exponentiated locally in each element and applied as a (relatively small) matrix multiplication rather than using FFTs.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

and boundary conditions.

3.6.3 Results

Here I compare four numerical methods: Fourier split-step (FSS), fourth order Runge–Kutta in the interaction picture (RK4IP), ordinary fourth order Runge–Kutta (RK4), and my new method — fourth order Runge–Kutta in an instananeous local interaction picture (RK4ILIP).

The example chosen is a 2D simulation of a turbulent Bose–Einstein condensate, in both a rotating and nonrotating frame. For the nonrotating frame the differential equation simulated was equation (3.129), and for the rotating frame the same equation was with an additional two terms added to the Hamiltonian:

$$\hat{H}_{\text{rot}} + \hat{H}_{\text{comp}} = -\boldsymbol{\Omega} \cdot \hat{\boldsymbol{L}} + \frac{1}{2}\hbar m^2 \Omega^2 r^2 \tag{3.132}$$

$$= \mathrm{i}\hbar\Omega\left(x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x}\right) + \frac{1}{2}\hbar m^2 \Omega^2 r^2. \tag{3.133}$$

The addition of the first term transforms the original Hamiltonian into a frame rotating at angular frequency Ω in the (x, y) plane, and is equivalent to the the Coriolis and centrifugal forces that appear in rotating frames in classical mechanics [37]. The second term is a harmonic potential that exactly compensates for the centrifugal part of this force. In this way the only potential in the rotating frame is the applied trapping potential, and the only effect of the rotating frame is to add the Coriolis force.

Four trapping potentials were used, all radial power laws with different powers. These examples were chosen to demonstrate the specific situation in which RK4ILIP provides a benefit over the other methods for spatial Schrödinger-like equations, as discussed above.

The results of 120 simulation runs are shown in Figure 3.11. Each simulation was of a ^{87}Rb condensate in the $|F = 2, m_F = 2\rangle$ state, in which the two-body s-wave scattering length is $a = 98.98$ Bohr radii [38]. The simulation region was 20 μm in the x and y directions, and the Thomas–Fermi radius of the condensate was $R = 9$ μm. The chemical potential was $\mu = 2\pi\hbar \times 1.91$ kHz, which is equivalent to a maximum Thomas–Fermi density $\rho_{\text{max}} = 2.5 \times 10^{14}$ cm^{-3} and a healing length $\xi = 1.1$ μm. There were 256 simulation grid points in each spatial dimension, which is 14 points per healing length.

Four different potentials were used, all of the form $V(r) = \mu(r/R)^\alpha$ with $\alpha = 4, 8, 12, 16$. For the rotating frame simulations, the rotation frequency was $\Omega = 2\pi \times 148$ Hz. This is 89% of the effective harmonic trap frequency, defined as the frequency of a harmonic trap that would have the same Thomas–Fermi radius given the same chemical potential.

All ground states were determined using successive over-relaxation (See section [TODO]) with sixth-order finite differences for spatial derivatives. For the nonrotating simulations, convergence was reached with $\Delta\mu/\mu < 1 \times 10^{-13}$, with:

$$\Delta\mu = \sqrt{\frac{\langle\psi|(\hat{H} - \mu)^2|\psi\rangle}{\langle\psi|\psi\rangle}}, \tag{3.134}$$

where $\hat{H}$ is the nonlinear Hamiltonian and $\langle \boldsymbol{r}|\psi\rangle$ is the condensate wavefunction, which does not have unit norm. For the rotating frame simulations the ground states converged to $\Delta\mu/\mu \approx 9\times10^{-7}, 2\times10^{-6}, 3\times10^{-6}$ and 2×10^{-6} for $\alpha = 16, 12, 8$, and 4 respectively.

After each ground state was found, it was multiplied by a spatially varying phase factor corresponding to the phase pattern of a number of randomly positioned vortices:

$$\psi_{\text{vortices}}(x, y) = \psi_{\text{groundstate}}(x, y) \prod_{n=1}^{N} e^{\pm_n i \arctan 2(y-y_n, x-x_n)} \tag{3.135}$$

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

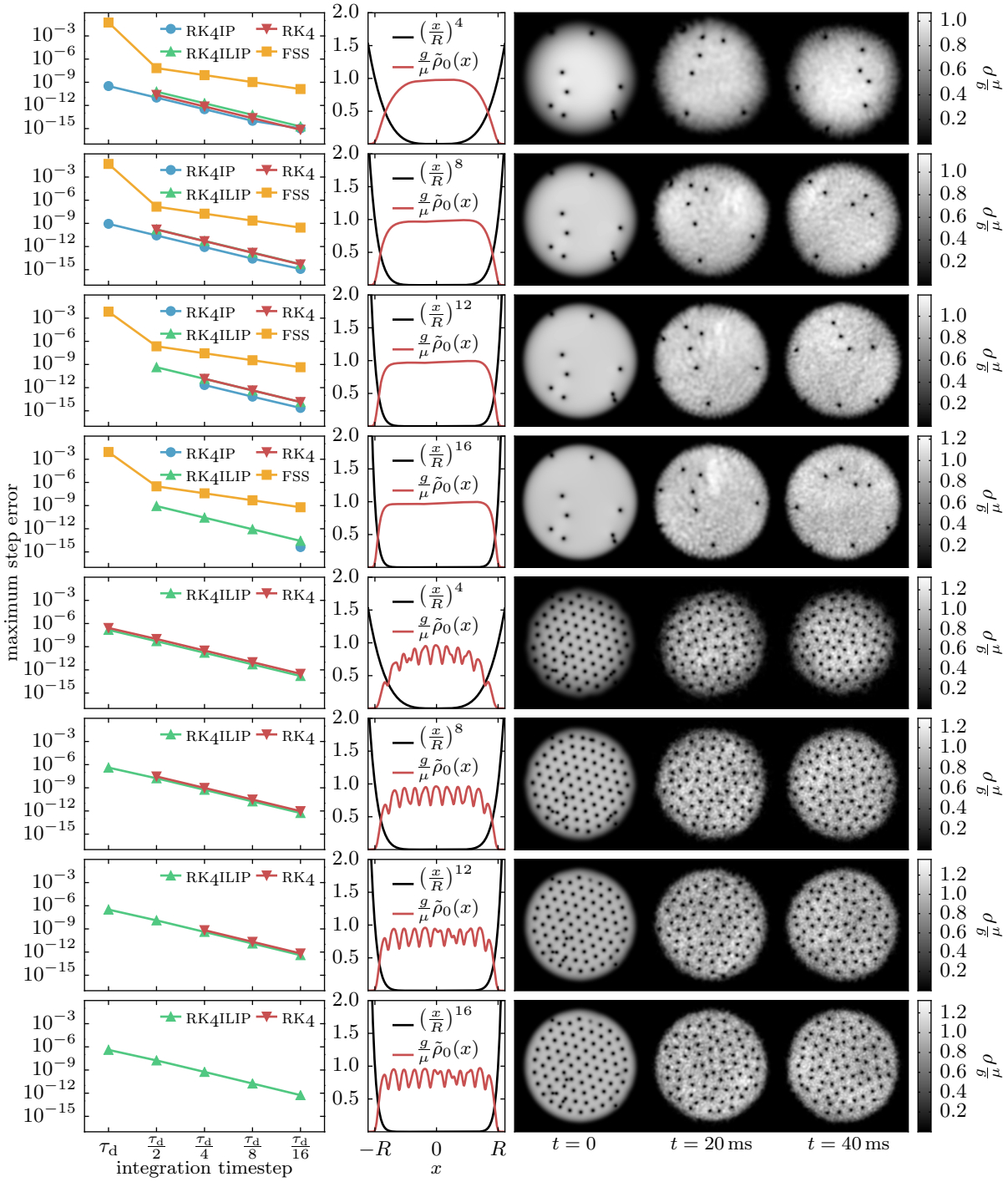

Figure 3.11: Results of simulations to compare RL4ILIP to other timestepping methods. Top four rows: Nonrotating frame simulations with four different radial power-law potentials. Bottom four rows: Rotating frame simulations with same four potentials. Left column: maximum per-step error $\int |\psi - \tilde{\psi}|^2 \,\mathrm{d}\boldsymbol{r} / \int |\tilde{\psi}|^2 \,\mathrm{d}\boldsymbol{r}$ of fourth order Runge–Kutta (RK4), its interaction picture variants (RK4IP and RK4ILIP) and Fourier split-step (FSS) as a function of timestep. Solutions were checked every 100 timesteps against a comparison solution $\tilde{\psi}$ computed using half sized steps for RK4 methods, and quarter sized steps for FSS. Simulations encountering numerical overflow not plotted. Centre column: potential (black) and average density $\tilde{\rho}_0$ of the initial state (red) over a slice of width $R/5$ in the y direction. Right column: Density of solution at initial, intermediate and final times for each configuration simulated (taken from RK4ILIP results). RK4ILIP is the only method usable in rotating frames and not encountering overflow in the steeper traps for the timesteps considered.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

where arctan2 is the two-argument arctan function,[31] $N = 30$, $\pm_n$ is a randomly chosen sign, and (x_n, y_n) are vortex positions randomly drawn from a Gaussian distribution centred on $(0, 0)$ with standard deviation equal to the Thomas–Fermi radius R. The same seed was used for the pseudorandom number generator in each simulation run, and so the vortex positions were identical in each simulation run.

[31] Defined as the principle value of the argument of the complex number $x + iy$: $\text{arctan2}(y, x) = \text{Arg}(x + iy)$.

After vortex phase imprinting, the wavefunctions were evolved in imaginary time [34]. For the nonrotating frame simulations, imaginary time evolution was performed for a time interval equal to the chemical potential timescale $\tau_\mu = 2\pi\hbar/\mu$, and for the rotating frame simulations, for $\tau_\mu/10$. This was done to smooth out the condensate density in the vicinity of vortices, producing the correct density profile for vortex cores. However, since imaginary time evolution decreases the energy of the state indiscriminately, it also had the side effect of causing vortices of opposite sign to move closer together and annihilate. This decreased the number of vortices, and is the reason the smoothing step in the rotating frame simulations was cut short to $\tau_\mu/10$, as otherwise all vortices had time to annihilate with one of the lattice vortices. A vortex pair in the process of annihilating is visible in Figure 3.11 as a partially filled hole in the initial density profile near the top of the condensate in the $\alpha = 4$, 12, and 16 rotating frame simulations.[32]

[32] The initial states for the four different potentials are not identical, so by chance the corresponding vortex in the $\alpha = 8$ case was not close enough to a lattice vortex to annihilate.

The smoothed, vortex imprinted states were then evolved in time for 40 ms. For each simulation, five different timesteps were used: $\Delta t = \tau_\text{d}, \tau_\text{d}/2, \tau_\text{d}/4, \tau_\text{d}/8, \tau_\text{d}/16$, where $\tau_\text{d} = m\Delta x^2/\pi\hbar \approx 2.68\,\mu\text{s}$ is the dispersion timescale associated with the grid spacing Δx, defined as the time taken to move one gridpoint at the group velocity of the Nyquist mode.

For the nonrotating frame simulations, spatial derivatives for the RK4 and RK4ILIP methods were determined using the Fourier method [see section TODO]. This was to ensure a fair comparison with the other two methods, which necessarily use Fourier transforms to perform computations pertaining due to the kinetic term.

For the rotating frame simulations, sixth-order finite differences with zero boundary conditions were used instead for the kinetic terms of the RK4 and RK4ILIP methods, which were the only two methods used for those simulations (due to the other methods being incompatible with the angular momentum operator required for a rotating frame). This choice was fairly arbitrary, but did allow the condensate to be closer to the boundary than is otherwise possible with the periodic boundary conditions imposed by use of the Fourier method for spatial derivatives. This is because the rotating frame Hamiltonian is not periodic in space, and so its discontinuity at the boundary can be a problem if the wavefunction is not sufficiently small there.

As shown in Figure 3.11, all methods tested generally worked well until they didn't work at all, with the per-step error of RK4-based methods being either small and broadly the same as the other RK4-based methods, or growing rapidly to the point of numerical overflow (shown as missing datapoints). The break down of FSS was less dramatic, though it too had a clear jump in its per-step error for larger timesteps. Comparing methods therefore came down to mostly whether or not a simulation experienced numerical overflow during the time interval being simulated.

The main result was that RK4ILIP and FSS remained accurate over the widest range of timesteps and trap steepnesses, with RK4 and RK4IP requiring ever smaller timesteps in order to not overflow as the trap steepness increased.

For the rotating frame simulations, which were only amenable to the RK4 and RK4ILIP methods, the same pattern was observed, with RK4 only working at smaller timesteps as the trap steepness was increased, and ultimately diverging for all timesteps tested at the maximum trap steepness. By contrast, RK4ILIP remained accurate over the entire range of timesteps at the maximum trap steepness.

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

3.6.4 Discussion

As mentioned, RK4ILIP is mostly useful for continuum quantum mechanics only when there are large spatial differences in the potential, which cannot give rise to equally large kinetic energies[33]. Furthermore, the advantage that RK4ILIP has over other methods with that same property is that it is does not require a particular form of Hamiltonian or a particular method of evaluating spatial derivatives. The former means is is applicable in rotating frames or to situations with unusual Hamiltonians, and the latter means is can be used with finite differences or FEDVR [21] and thus is amenable to parallelisation on a cluster computer.

[33] This is essentially due to such a situation violating the condition we laid out at the beginning of this section — that the simulation basis must be nearly an eigenbasis of the total Hamiltonian.

The ability to model large spatial variations in the potential provides only a narrow domain of increased usefulness over other methods. If a large kinetic energy results from the large potential, then the method requires just as small timesteps as any other. And if the large potential is supposed to approximate an an infinite well, then an actual infinite well may be modelled using zero boundary conditions, negating the need for something like RK4ILIP. However, when potential wells are steep, but not infinitely steep, here RK4ILIP provides a benefit. The only other model that can handle these large potentials—Fourier split-step—has the disadvantage that it cannot deal with arbitrary operators such as those arising from a rotating frame, and is not parallelisable with local data. The benefits of parallelisability are obvious, and the above results demonstrate RK4ILIP's advantage at simulating BECs in tight traps and rotating frames.

Note that whilst the *Fourier* split-step method can't handle Hamiltonian terms such as $\hat{\boldsymbol{r}} \cdot \hat{\boldsymbol{p}}$ that are not diagonal in either real space or Fourier space [14, p. 315], a split-step method based on an approximation to the momentum operator as a banded matrix, such as that obtained with finite differences, can. Using the techniques discussed in section 3.2.3, such a scheme is also parallelisable. The remaining limitations then, when compared to fourth-order Runge–Kutta are the restriction on the types of nonlinearity that can be included, and the complexity of implementation.

For systems with discrete degrees of freedom, RK4ILIP may be useful in the case where an approximate diagonalisation of the Hamiltonian is analytically known, and when the Hamiltonian's eigenvalues vary considerably in time (making a single interaction picture insufficient to factor out dynamical phases throughout the entire simulation). In this situation an analytic transformation into the diagonal basis can be performed at each timestep (or the differential equation analytically re-cast in that basis in the first place), and RK4ILIP can be used to factor out the time-varying dynamical phase evolution at each timestep. An example may be an atom with a magnetic moment in a time-varying magnetic field which varies over orders of magnitude. The transformation into the spin basis in the direction of the magnetic field can be analytically performed, and if the field varies by orders of magnitude, so do the eigenvalues of the Hamiltonian. Although the eigenvalues in this case and other similar cases can be computed analytically too, unless all time dependence of the Hamiltonian is known in advance of the simulation, it would be difficult to incorporate this into a re-casting of the differential equation in a time-dependent interaction picture. RK4ILIP may be useful in these cases to automate this process and evolve the system in the appropriate interaction picture at each timestep.

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

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 (2014). arXiv: 1412.6854. [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, E. Mazets, I. 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] P. D. Lett, R. N. Watts, C. I. Westbrook, W. D. Phillips, P. L. Gould, and H. 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, J. 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

[11] T. Brabec and F. 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. Brezinski. *Extrapolation algorithms and Padé approximations: A historical survey*. Applied Numerical Mathematics **20**, 299 (1996). DOI: 10.1016/0168-9274(95)00110-7. [p 4]

[13] 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]

[14] D. J. Tannor. *Introduction to Quantum Mechanics: A Time-Dependent Perspective*. University Science Books (2007). [pp 5, 9, 26, 33, and 51]

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

[16] F. J. Dyson. *The S Matrix in Quantum Electrodynamics*. Physical Review **75**, 1736 (1949). DOI: 10.1103/PhysRev.75.1736. [p 6]

[17] V. Kaplunovsky. On Perturbation Theory, Dyson Series, and Feynman Diagrams. http://bolvan.ph.utexas.edu/~vadim/Classes/2016f/dyson.pdf, (2016). Online lecture notes. [p 6]

[18] M. Suzuki. *Quantum Monte Carlo Methods in Condensed Matter Physics*. World Scientific (1993). [p 10]

[19] B. I. Schneider and L. A. Collins. *The discrete variable method for the solution of the time-dependent Schrödinger equation*. Journal of Non-Crystalline Solids **351**, 1551 (2005). DOI: 10.1016/j.jnoncrysol.2005.03.028. [pp 10, 33, 34, 36, and 37]

[20] M. Suzuki. *General Decomposition Theory of Ordered Exponentials*. Proceedings of the Japan Academy, Series B **69**, 161 (1993). DOI: 10.2183/pjab.69.161. [p 10]

[21] B. I. Schneider, L. A. Collins, and S. X. Hu. *Parallel solver for the time-dependent linear and nonlinear Schrödinger equation*. Physical Review E **73**, 036708 (2006). DOI: 10.1103/PhysRevE.73.036708. [pp 15, 33, 36, 47, and 51]

[22] J. Javanainen and J. Ruostekoski. *Symbolic calculation in development of algorithms: Split-step methods for the Gross–Pitaevskii equation*. Journal of Physics A: Mathematical and General **39**, L179 (2006). DOI: 10.1088/0305-4470/39/12/L02. [p 16]

[23] S. A. Orszag. *Comparison of pseudospectral and spectral approximation*. Studies in Applied Mathematics **51**, 253 (1972). DOI: 10.1002/sapm1972513253. [p 26]

[24] N. Phillips. *An example of non-linear computational instability*. In *The Atmosphere and the Sea in Motion*, pages 501–504. Rockefeller Univ. Press (1959). [p 26]

[25] B. Fornberg. *The pseudospectral method: Comparisons with finite differences for the elastic wave equation*. GEOPHYSICS **52**, 483 (1987). DOI: 10.1190/1.1442319. [pp 29 and 38]

[26] B. Fornberg. *Generation of finite difference formulas on arbitrarily spaced grids*. Mathematics of Computation **51**, 699 (1988). DOI: 10.1090/S0025-5718-1988-0935077-0. [pp 30 and 31]

[27] A. Gupta and V. Kumar. *The scalability of FFT on parallel computers*. IEEE Transactions on Parallel and Distributed Systems **4**, 922 (1993). [pp 31 and 47]

```
rev:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```

[28] J.-P. Berrut and L. N. Trefethen. *Barycentric lagrange interpolation*. SIAM Review **46**, 501 (2004). DOI: 10.1137/S0036144502417715. [p 34]

[29] R. M. Caplan and R. Carretero-González. *Numerical stability of explicit runge-kutta finite difference schemes for the nonlinear schrödinger equation*. CoRR **abs/1107.4810** (2011). [p 36]

[30] B. M. C. Davies. *Vortex dynamics in Bose–Einstein condensates*. PhD thesis, University of Otago, Dunedin, New Zealand (2000). [pp 37 and 44]

[31] J. J. Sakurai. *Modern quantum mechanics*. Addison-Wesley Pub. Co, Reading, Mass (1994). [p 43]

[32] W. Press. *The art of scientific computing*. Cambridge University Press (1992). [p 45]

[33] C. Pethick and H. Smith. *Bose–Einstein condensation in dilute gases*. Cambridge University Press (2002). [p 45]

[34] M. L. Chiofalo, S. Succi, and M. P. Tosi. *Ground state of trapped interacting Bose–Einstein condensates by an explicit imaginary-time algorithm*. Phys. Rev. E **62**, 7438 (2000). DOI: 10.1103/PhysRevE.62.7438. [pp 46 and 50]

[35] G. M. Muslu and H. A. Erbay. *Higher-order split-step Fourier schemes for the generalized nonlinear Schrödinger equation*. Mathematics and Computers in Simulation **7**, 581 (2005). DOI: 10.1016/j.matcom.2004.08.002. [p 47]

[36] M. Heroux, P. Raghavan, and H. Simon. *Parallel processing for scientific computing*. Society for Industrial and Applied Mathematics, Philadelphia, PA (2006). [p 47]

[37] P. Gulshani and D. J. Rowe. *Quantum mechanics in rotating frames. I. The impossibility of rigid flow*. Canadian Journal of Physics **56**, 468 (1978). DOI: 10.1139/p78-060. [p 48]

[38] E. G. M. van Kempen, S. J. J. M. F. Kokkelmans, D. J. Heinzen, and B. J. Verhaar. *Interisotope determination of ultracold rubidium interactions from three high-precision experiments*. Phys. Rev. Lett. **88**, 093201 (2002). DOI: 10.1103/PhysRevLett.88.093201. [p 48]

rev: 69 (6f411aaed797)
author: Chris Billington
date: Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons

This page intentionally left blank

Word count

```
Total
Words in text: 33820
Words in headers: 290
Words outside text (captions, etc.): 4735
Number of headers: 69
Number of floats/tables/figures: 37
Number of math inlines: 861
Number of math displayed: 143
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
  20691+151+2365 (29/13/707/118) 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:     69 (6f411aaed797)
author:  Chris Billington
date:    Wed Sep 06 17:24:42 2017 -0400
summary: Working on FEDVR operator accuracy comparisons
```