Notes on Solving the Gross-Pitaevskii Equation Numerically with MATLAB

Tom Barrett

November 2020

Contents

1 Introduction

This document is a collection of notes and things I have learned when calculating numerical solutions to differential equations (Matlab is used here, but the general techniques are the same in other programming languages). The original motivation was to learn how to numerically solve the Gross-Pitaevskii equation (GPE), in order to model the ground states and dynamics of a BEC from the experiment. The GPE can typically be solved using either finite differences (using the Crank-Nicolson method usually), or using spectral techniques. The latter uses Fourier transforms to calculate derivatives to a high accuracy, so a large part of the document in the beginning focuses on understanding how to carry out Fourier forward and inverse transforms numerically using the fast Fourier transform algorithm (FFT).

It can be tricky to get all the factors correct when doing forward and inverse Fourier transforms, and so I have summarised analytic expressions for the transforms as well (such as transforms of Gaussian functions) which have been used where possible to sanity check the numerical results. It can also be tricky to set up spatial grids and angular spatial frequency grids in Matlab, without giving rise to unwanted phase shifts in the results, so this is tackled directly in this document.

After going through how to calculate spectral derivatives, Ch. 4 reviews a variety of general introductory techniques for solving order ordinary differential equations and partial differential equations of various orders, in 1D, 2D, and 3D. This is in preparation for solving the non-linear GPE in later chapters.

2 Fourier Transforms

This section collects several definitions which are useful to lookup when working with Fourier transforms, in 1D, and 2D. There are different conventions for symbols to use (time vs space coordinates, or linear frequency vs angular frequency), as well as different ways of symmetrising the transforms themselves. The following are the definitions and symbols that I usually like to adhere to.

2.1 Definitions

Let's say we have a function of a single variable, time, given by $\psi(t)$. Recall the definition of the forward and inverse Fourier transforms, for converting back and forth between real space and reciprocal (or frequency) space, given respectively by

$$\mathcal{F}\{\,\psi(t)\,\} = \hat{\psi}(f) = \int_{-\infty}^{+\infty} \psi(t)\; e^{-2\pi i f t}\,\mathrm{d}t \tag{1}$$

$$\mathcal{F}^{-1}\{\,\hat{\psi}(f)\,\} = \psi(t) = \int_{-\infty}^{+\infty} \hat{\psi}(f)\; e^{+2\pi i f t}\,\mathrm{d}f \tag{2}$$

The above definition is in terms of linear frequency, which is the reciprocal of the time coordinate. However, we can also perform the Fourier transforms into angular frequency space, where $\omega = 2\pi f$,

with the transforms now being given by

$$\mathcal{F}\{\,\psi(t)\,\} = \hat{\psi}(\omega) = \int_{-\infty}^{+\infty} \psi(t)\; e^{-i\omega t}\,\mathrm{d}t \tag{3}$$

$$\mathcal{F}^{-1}\{\,\hat{\psi}(\omega)\,\} = \psi(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \hat{\psi}(\omega)\; e^{+i\omega t}\,\mathrm{d}\omega. \tag{4}$$

The additional factor in Eq (4) arises because $\mathrm{d}f = \mathrm{d}\omega/2\pi$. Note that in the above definitions, the Fourier transforms were performed with respect to the variable time. However, if instead our function was a function of space, i.e. $\psi(x)$, then the convention is to use the angular spatial frequency k as the reciprocal coordinate, and this takes on the role of ω as follows

$$\mathcal{F}\{\,\psi(x)\,\} = \hat{\psi}(k) = \int_{-\infty}^{+\infty} \psi(x)\; e^{-ikx}\,\mathrm{d}x \tag{5}$$

$$\mathcal{F}^{-1}\{\,\hat{\psi}(k)\,\} = \psi(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \hat{\psi}(k)\; e^{+ikx}\,\mathrm{d}k. \tag{6}$$

If we have a function of both space and time, $\psi(x,t)$, then we can take the Fourier transform with respect to one of the variables - for example in the case of the space variable we would have

$$\mathcal{F}_x\{\,\psi(x,t)\,\} = \hat{\psi}(k,t) = \int_{-\infty}^{+\infty} \psi(x,t)\; e^{-ikx}\,\mathrm{d}x \tag{7}$$

$$\mathcal{F}_x^{-1}\{\,\hat{\psi}(k,t)\,\} = \psi(x,t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \hat{\psi}(k,t)\; e^{+ikx}\,\mathrm{d}k, \tag{8}$$

and it can be seen that the time variable t is not converted to reciprocal space under the Fourier transform operation.

If we have a function of two variables, for example $\psi(x,y)$, then a two-dimensional version of the Fourier transform can be performed, which will bring the signal into the angular spatial frequency domain in both dimensions simultaneously. The forward and inverse transforms are defined respectively by

$$\mathcal{F}_{2\mathrm{D}}\{\,\psi(x,y)\,\} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \psi(x,y)\,\exp\Big(-i\left[k_x x + k_y y\right]\Big)\;\mathrm{d}x\;\mathrm{d}y \tag{9}$$

$$\mathcal{F}_{2\mathrm{D}}^{-1}\{\,\hat{\psi}(k_x,k_y)\,\} = \frac{1}{(2\pi)^2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \hat{\psi}(k_x,k_y)\,\exp\Big(i\left[k_x x + k_y y\right]\Big)\;\mathrm{d}k_x\;\mathrm{d}k_y. \tag{10}$$

The extension of the expression to 3D to include a k_z variable is straightforward, and results in a prefactor of $1/(2\pi)^3$.

2.2 Analytic Fourier Transform of a 1D Gaussian

Let's say we a have a pulse-like voltage signal $\psi(t)$ that is described by a Gaussian function centered at time zero, and which is defined over all time from $-\infty$ to $+\infty$, given by

$$\psi(t) = A \exp\left(-\frac{t^2}{2\sigma^2}\right), \tag{11}$$

where A is the amplitude at $t = 0$, and σ is the RMS width (also called standard deviation) of the pulse, and both are real constants. If you insert $t = \sigma$ into the function, you will see that the value of σ is the time at which the signal has dropped to $1/\sqrt{e} \approx 60.7\%$ of its peak value A. The value of A required for this function to be normalised to 1 is given by integrating the signal

$$\int_{-\infty}^{+\infty} \psi(t)\,\mathrm{d}t = 1 \tag{12}$$

$$\implies \int_{-\infty}^{+\infty} A \exp\left(-\frac{t^2}{2\sigma^2}\right) \mathrm{d}t = 1 \tag{13}$$

$$\implies A\sqrt{\pi}\sqrt{2}\sigma = 1 \tag{14}$$

$$\implies A = \frac{1}{\sqrt{2\pi}\sigma}. \tag{15}$$

In order to concretely fix the dimensions and units of the signal, let's say the signal is a voltage, and an example of an (un-normalised) signal with $A = 1.5\,[\mathrm{V}]$ and $\sigma = 2\,[\mathrm{s}]$ is plotted in Fig. 1 (a).

Forward Transform

The Fourier transform of this signal can be obtained using Eq. (1) as follows

$$\mathcal{F}\{\,\psi(t)\,\} = \int_{-\infty}^{+\infty} \psi(t) e^{-2\pi i f t}\,\mathrm{d}t = \hat{\psi}(f) \tag{16}$$

$$= \int_{-\infty}^{+\infty} A \exp\left(-\frac{t^2}{2\sigma^2}\right) e^{-2\pi i f t}\,\mathrm{d}t \tag{17}$$

$$= A \int_{-\infty}^{+\infty} \exp\left(-\frac{t^2}{2\sigma^2} - 2\pi i f t\right) \mathrm{d}t \tag{18}$$

$$= A \int_{-\infty}^{+\infty} \exp\left(-\left[\frac{t}{\sqrt{2}\sigma} + \sqrt{2}\pi i \sigma f\right]^2 - 2\pi^2 f^2 \sigma^2\right) \mathrm{d}t \tag{19}$$

$$= A \exp\left(-2\pi^2 f^2 \sigma^2\right) \int_{-\infty}^{+\infty} \exp\left(-\left[\frac{t}{\sqrt{2}\sigma} - \sqrt{2}\pi i \sigma f\right]^2\right) \mathrm{d}t. \tag{20}$$

Now, the integral can be evaluated using the standard result

$$\int_{-\infty}^{+\infty} \exp\left(-\left[\frac{y}{b} - c\right]^2\right) \mathrm{d}y = \sqrt{\pi} b \tag{21}$$

to finally obtain

$$\hat{\psi}(f) = A\sigma\sqrt{2\pi}\exp\left(-2\pi^2 f^2\sigma^2\right) \tag{22}$$

$$= B\exp\left(-\frac{f^2}{2\gamma^2}\right). \tag{23}$$

Therefore, the Fourier transform of a Gaussian is another Gaussian, with parameters $B = A\sigma\sqrt{2\pi}$ and $\gamma = 1/(2\pi\sigma)$. Note that if the original signal was normalised to 1 as in Eq. 15, then in frequency space we obtain $B = 1$. This is important when doing convolutions and you don't want to change the total energy in an input signal (achieved by having a DC ($f = 0$) term equal to 1).

We could also carry out the Fourier transform in angular frequency reciprocal space using Eq. 3, leading to

$$\mathcal{F}\{\,\psi(t)\,\} = \hat{\psi}(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-i\omega t}\,\mathrm{d}t \tag{24}$$

$$= A\sigma\sqrt{2\pi}\exp\left(-\frac{\omega^2\sigma^2}{2}\right), \tag{25}$$

where the integral was evaluated by completing the square in the same way as before. We can see that in angular frequency space, the RMS width of the Gaussian is given by $1/\sigma$, and so the smaller is the width in real space, the larger is the width in reciprocal space. If the signal was a real space wavefunction in quantum mechanics, this property is responsible for the uncertainty principle, since the position and momentum representation are a Fourier transform pair.

In general, the Fourier transform of a signal, $\psi(f)$ will be a complex function. A complex number can be written as $z = a + bi$, or in polar form using Euler's formula as $z = re^{i\phi}$, where the magnitude and phase of the complex number are given respectively by

$$r = |z| = \sqrt{a^2 + b^2}, \tag{26}$$

$$\phi = \arg(z) = \arctan2\left(\mathrm{Im}(z), \mathrm{Re}(z)\right) = \arctan2(b, a) \tag{27}$$

MATLAB provides the functions `abs()` and `angle()` for obtaining the magnitude and phase of a complex number, and these are used in Fig. 1 (b) and (c) to plot $|\hat{\psi}(f)|$, which has units $[\mathrm{V}\cdot\mathrm{s}]$, and $\arg(\hat{\psi}(f))$, which has units of $[\mathrm{rad}]$, respectively, for our Gaussian signal example. In this case, because the Fourier transform $\hat{\psi}(f)$ is completely real, with no imaginary component, the phase of this signal is simply zero everywhere. Note, the reason for this is that the signal is centered at the origin, $t = 0$, and if the signal was to be displaced in time then the phase would become non-zero.

Since the units of $\psi(t)$ were $[\mathrm{V}]$, the units of $\hat{\psi}(f)$ are $[\mathrm{V}\cdot\mathrm{s}]$, which can be seen from the definition of the forward Fourier transform in Eq. (1), because the exponential is always dimensionless and the $\mathrm{d}t$ introduces an additional dimensional factor of time to the input signal (keeping track of the dimensions of the Fourier transforms of signals becomes critical when working with power/energy spectra and power/energy spectral densities).

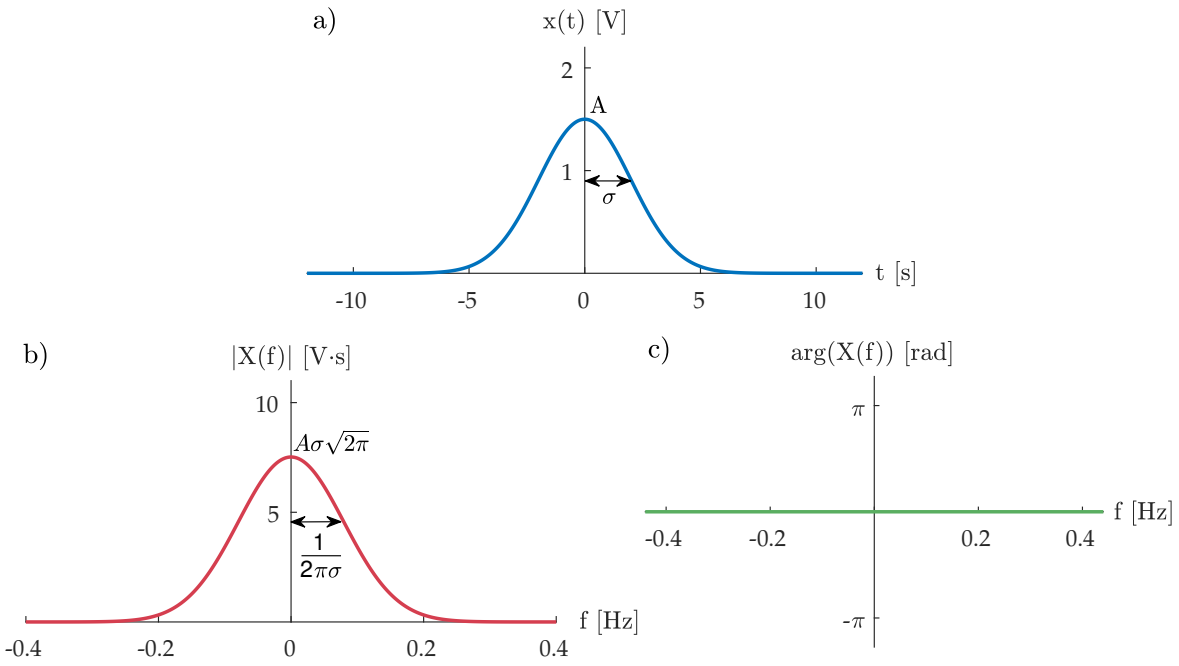

FIGURE 1: The continuous Fourier transform of an analytic Gaussian signal. a) The input signal $x(t)$, with $A = 1.5\,[\mathrm{V}]$ and $\sigma = 2\,[\mathrm{s}]$. (b) The magnitude $|\hat{\psi}(f)|$, and (c) the argument $\arg(\hat{\psi}(f))$ (also known as the *phase*), of the complex Fourier transform of a). Here, the phase is zero as the signal is centered on $t = 0$.

Inverse Transform

We can as well verify that the inverse Fourier transform, defined in Eq. (2), allows the input signal to be faithfully recovered. The following is done whilst keeping the redefined parameters B and γ in Eq. (23) for the sake of readability

$$\mathcal{F}^{-1}\{\hat{\psi}(f)\} = \psi(t) = \int_{-\infty}^{+\infty} \hat{\psi}(f) e^{+2\pi i f t}\,\mathrm{d}f \tag{28}$$

$$= \int_{-\infty}^{+\infty} B \exp\left(-\frac{f^2}{2\gamma^2}\right) e^{+2\pi i f t}\,\mathrm{d}f \tag{29}$$

$$= B\int_{-\infty}^{+\infty} \exp\left(-\frac{f^2}{2\gamma^2} + 2\pi i f t\right)\mathrm{d}f \tag{30}$$

$$= B\int_{-\infty}^{+\infty} \exp\left(-\left[\frac{f}{\sqrt{2}\gamma} - \sqrt{2}\pi i\gamma t\right]^2 - 2\pi^2\gamma^2 t^2\right)\mathrm{d}f \tag{31}$$

$$= B\exp\left(-2\pi^2\gamma^2 t^2\right)\int_{-\infty}^{+\infty} \exp\left(-\left[\frac{f}{\sqrt{2}\gamma} - \sqrt{2}\pi i\gamma t\right]^2\right)\mathrm{d}f \tag{32}$$

$$= B\gamma\sqrt{2\pi}\exp\left(-2\pi^2\gamma^2 t^2\right). \tag{33}$$

Now, substituting back in the redefined expressions for B and γ in terms of the original signal's parameters A and σ, we obtain

$$\mathcal{F}^{-1}\{\, X(f)\,\} = x(t) = A\sigma\sqrt{2\pi}\frac{1}{2\pi\sigma}\sqrt{2\pi}\exp\left(-\frac{2\pi^2 t^2}{(2\pi\sigma)^2}\right) \tag{34}$$

$$= A\exp\left(-\frac{t^2}{2\sigma^2}\right), \tag{35}$$

demonstrating that the original signal from Eq. (11) is recovered. The inverse transform of the angular frequency version can also be performed in a similar way, where it becomes clear that the factor of $1/2\pi$ in Eq. 4 then becomes necessary to faithfully recover the original signal.

2.3 Fourier Transform Shift Theorem

The Shift Property of the Fourier transform is useful when the function being considered is not centered on the zero of the coordinate system, but is displaced / delayed. Let's take the function $\psi(t)$ and delay it by some time t_0 so that we have $\psi(t) \to \psi(t - t_0)$. The Fourier transform of this is given by

$$\mathcal{F}\{\, \psi(t-t_0)\,\} = \int_{-\infty}^{+\infty} \psi(t-t_0)\; e^{-2\pi i f t}\,\mathrm{d}t. \tag{36}$$

Now we can make a substitution by defining a variable $\tau = t - t_0$, and so $\mathrm{d}\tau = \mathrm{d}t$, which allows us to write

$$\mathcal{F}\{\, \psi(t-t_0)\,\} = \int_{-\infty}^{+\infty} \psi(\tau)\; e^{-2\pi i f(\tau+t_0)}\,\mathrm{d}\tau \tag{37}$$

$$= e^{-2\pi i f t_0}\int_{-\infty}^{+\infty} \psi(\tau)\; e^{-2\pi i f \tau}\,\mathrm{d}\tau, \tag{38}$$

$$= e^{-2\pi i f t_0}\int_{-\infty}^{+\infty} \psi(t)\; e^{-2\pi i f t}\,\mathrm{d}t, \tag{39}$$

where the exponential in t_0 is a constant from the perspective of the integral over τ and has been brought outside. After doing this, the variable τ was simply the integration variable, and so could be simply relabelled it back to t again. This factor is then just the Fourier transform of $\psi(t)$, so we can write

$$\boxed{\mathcal{F}\{\, \psi(t-t_0)\,\} = e^{-2\pi i f t_0}\mathcal{F}\{\, \psi(t)\,\}.} \tag{40}$$

This result shows that the Fourier transform of a signal which is delayed in the time domain by t_0 (or shifted in the spatial domain if working with a space coordinate) is the Fourier transform of the non-delayed signal but multiplied by an additional phase factor of $\exp(-2\pi i f t_0) = \exp(-i\omega t_0)$. This phase factor is *proportional* to the frequency variable f.

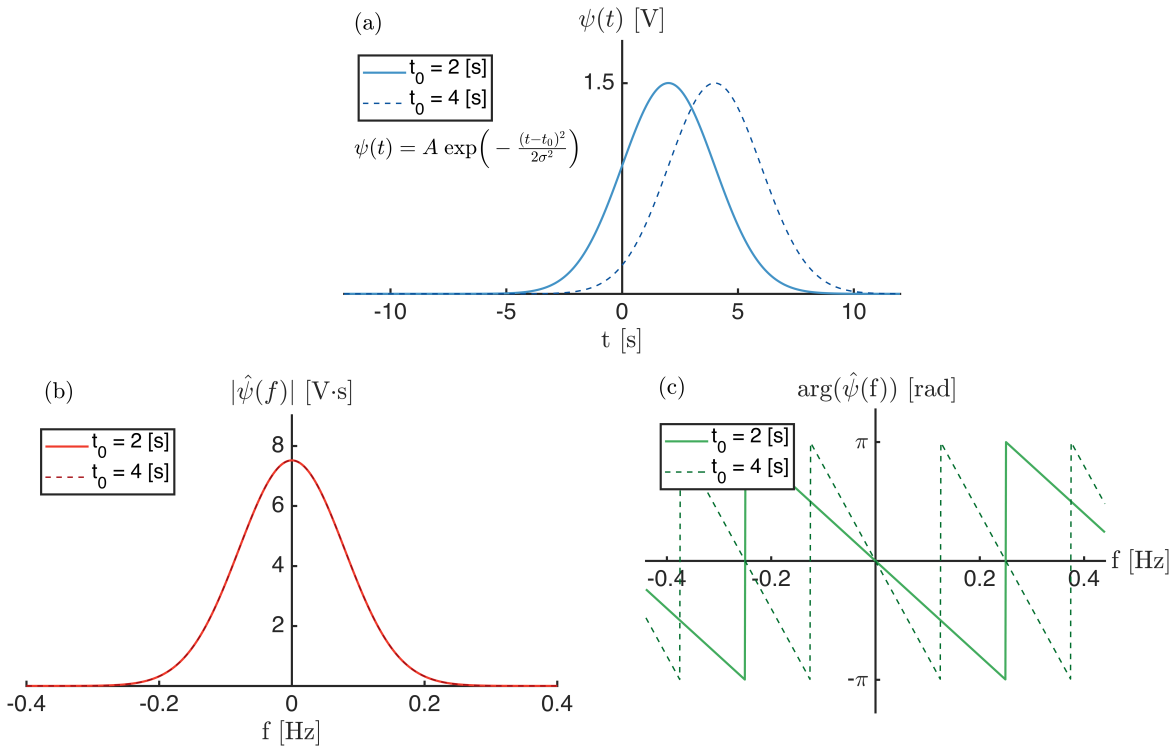

FIGURE 2: The analytic continuous Fourier transform of a shifted Gaussian signal with two different shifts of $t_0 = 2$ [s] and $t_0 = 4$ [s] (solid and dashed lines, respectively). a) The input signal $\psi(t)$, with $A = 1.5$ [V] and $\sigma = 2$ [s]. (b) The magnitude is unaffected by any shifts in the original signal. (c) The phase (plotted here wrapped on the interval $[-\pi, +\pi]$) becomes non-zero when there is some shift in the input signal. This phase is linear with frequency, and the slope is proportional to the shift t_0.

For example, if we have a 1D Gaussian signal which is delayed in time, given by

$$\psi(t, t_0) = A \exp\left(-\frac{(t-t_0)^2}{2\sigma^2}\right), \tag{41}$$

we can use the Shift Theorem to write the Fourier transform as

$$\mathcal{F}\{\,\psi(t, t_0)\,\} = e^{-2\pi i f t_0} \mathcal{F}\{\,\psi(t, t_0 = 0)\,\} \tag{42}$$

$$= e^{-2\pi i f t_0} A\sigma\sqrt{2\pi} \exp\left(-2\pi^2 f^2 \sigma^2\right). \tag{43}$$

This is plotted in Fig. 2 for two different t_0 delays, where it is apparent that the magnitude of the Fourier transform is not affected by the shift, but the phase is.

2.4 Analytic Fourier Transform of a 2D Gaussian

Consider a Gaussian function of two spatial variables

$$\psi(x,y) = A\exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right), \tag{44}$$

where the peak value A required to normalise the area under the function to 1 is given by

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \psi(x,y)\,\mathrm{d}x\,\mathrm{d}y = 1 \tag{45}$$

$$\implies \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} A\exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right)\mathrm{d}x\,\mathrm{d}y = 1 \tag{46}$$

$$\implies A\int_{-\infty}^{+\infty}\exp\left(-\frac{y^2}{2\sigma_y^2}\right)\int_{-\infty}^{+\infty}\exp\left(-\frac{x^2}{2\sigma_x^2}\right)\mathrm{d}x\,\mathrm{d}y = 1 \tag{47}$$

$$\implies A\sqrt{2\pi}\sigma_x\int_{-\infty}^{+\infty}\exp\left(-\frac{y^2}{2\sigma_y^2}\right)\mathrm{d}y = 1 \tag{48}$$

$$\implies A\sqrt{2\pi}\sigma_x\sqrt{2\pi}\sigma_y = 1 \tag{49}$$

$$\implies A = \frac{1}{2\pi\sigma_x\sigma_y}. \tag{50}$$

The 2D forward Fourier transform in the angular spatial frequency domain using Eq. 9 is given by

$$
\begin{aligned}
\mathcal{F}_{2\mathrm{D}}\left\{\,\psi(x,y)\,\right\} &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \psi(x,y)\,\exp\Big(-i\left[k_x x + k_y y\right]\Big)\,\mathrm{d}x\,\mathrm{d}y \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} A\exp\left(-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}\right)\,\exp\Big(-i\left[k_x x + k_y y\right]\Big)\,\mathrm{d}x\,\mathrm{d}y \\
&= A\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \exp\left(-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2}+2ik_x x\right]-\frac{1}{2}\left[\frac{y^2}{\sigma_y^2}+2ik_y y\right]\right)\,\mathrm{d}x\,\mathrm{d}y \\
&= A\exp\left(-\frac{k_x^2\sigma_x^2}{2}-\frac{k_y^2\sigma_y^2}{2}\right)\times \ldots \\
&\qquad \ldots \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \exp\left(-\frac{1}{2}\left[\frac{x}{\sigma_x}+ik_x x\sigma_x\right]^2-\frac{1}{2}\left[\frac{y}{\sigma_y}+ik_y y\sigma_y\right]^2\right)\,\mathrm{d}x\,\mathrm{d}y \\
&= A\exp\left(-\frac{k_x^2\sigma_x^2}{2}-\frac{k_y^2\sigma_y^2}{2}\right)\sqrt{2\pi}\sigma_x \int_{-\infty}^{+\infty} \exp\left(-\frac{1}{2}\left[\frac{y}{\sigma_y}+ik_y y\sigma_y\right]^2\right)\,\mathrm{d}y \\
&= 2\pi\sigma_x\sigma_y A\exp\left(-\frac{k_x^2\sigma_x^2}{2}-\frac{k_y^2\sigma_y^2}{2}\right)
\end{aligned}
$$

So the 2D Fourier transform of an un-normalised 2D Gaussian is given by another Gaussian in k-space

$$
\boxed{\hat{\psi}(k_x,k_y) = 2\pi\sigma_x\sigma_y A\exp\left(-\frac{k_x^2\sigma_x^2}{2}-\frac{k_y^2\sigma_y^2}{2}\right),} \tag{51}
$$

where a factor of $A = 1/(2\pi\sigma_x\sigma_y)$ would make the orginal function normalised to unity.

2.5 Calculating the Fourier Transform of a 1D Gaussian Numerically in Matlab

To calculate Fourier transforms numerically on the computer, we need to move from continuous space (all the Fourier transform definitions so far were continuous integrals over space) into discrete space. This is done by implementing the *discrete Fourier transform* (DFT) instead of the *continuous Fourier tranform* (CFT).

```
Fs = 1.8;  % Sampling frequency
N  = 42;   % Number of samples to collect
dt = 1/Fs; % Sampling period (separation between time samples)
T  = N*dt; % Extent of sampling window (aka "record length")

% Create time vector for a non-causal signal (i.e centred on t=0)
N_IS_EVEN = ~rem(N,2);
if N_IS_EVEN; t = -(T/2):dt:(T/2-dt);             % If number of points is even
else;         t = -(T/2-dt/2):dt:(T/2-dt/2); end  % If number of points is odd
```

```
s = 2;       % RMS width (Sigma)
A = 1.5;     % Amplitude
psi = A*exp(-t.^2/(2*s^2)); % Create Gaussian signal sampled at the specified times

df = Fs/N; % Frequency resoltuion (separation in frequency space)

% Create frequency vector for a Fourier Tranform centered on f=0
if N_IS_EVEN; f = -(Fs/2):df:(Fs/2-df);
else;         f = -(Fs/2-df/2):df:(Fs/2-df/2); end

% Approximate the continuous Fourier transform using DFT (via FFT algorithm)
psi_hat = fftshift( fft( ifftshift(psi) ) )*dt;

% Analytically calculated result for comparison
psi_hat_analytic = A*sqrt(2*pi)*s*exp(-2*pi^2*s^2*f.^2);

figure;plot(t, psi, 'o')
figure;hold all;plot(f, abs(psi_hat_analytic));plot(f, abs(psi_hat),'o')
figure;hold all;plot(f, angle(psi_hat_analytic));plot(f, angle(psi_hat),'o')
```

The following code takes the Fourier transformed vector (which is a complex quantity) and the sampling rate, and uses these to recover the original signal by approximating the continuous inverse Fourier transform. The result is plotted on top of the original sampled signal in Fig. 3 (a).

```
% The transformed vector and sampling frequency of acquisition are needed in
% order to recover the original input signal using inverse transform
clearvars -except psi_hat Fs

Nf = length(psi_hat); % Obtain number of frequency samples present
df = Fs/Nf;           % Frequency resolution
dt = 1/Fs;            % Separation of time-domain points (sampling period)
T = Nf*dt;            % Record length

Nf_IS_EVEN = ~rem(Nf,2);
if Nf_IS_EVEN; t_recovered = -(T/2):dt:(T/2-dt);
else;          t_recovered = -(T/2 - dt/2):dt:(T/2 - dt/2);
end

% Approximate the inverse continuous Fourier transform using IDFT (via IFFT)
psi_recovered = fftshift( ifft( ifftshift(psi_hat) ) )*Nf*df;
```

2.6 When and How to Use ifftshift

The definition of the Fast Fourier Transform used by Matlab is given by

$$X[k] = \sum_{n=1}^{N} \exp\left(-2\pi i \frac{(n-1)(k-1)}{N}\right), \tag{52}$$

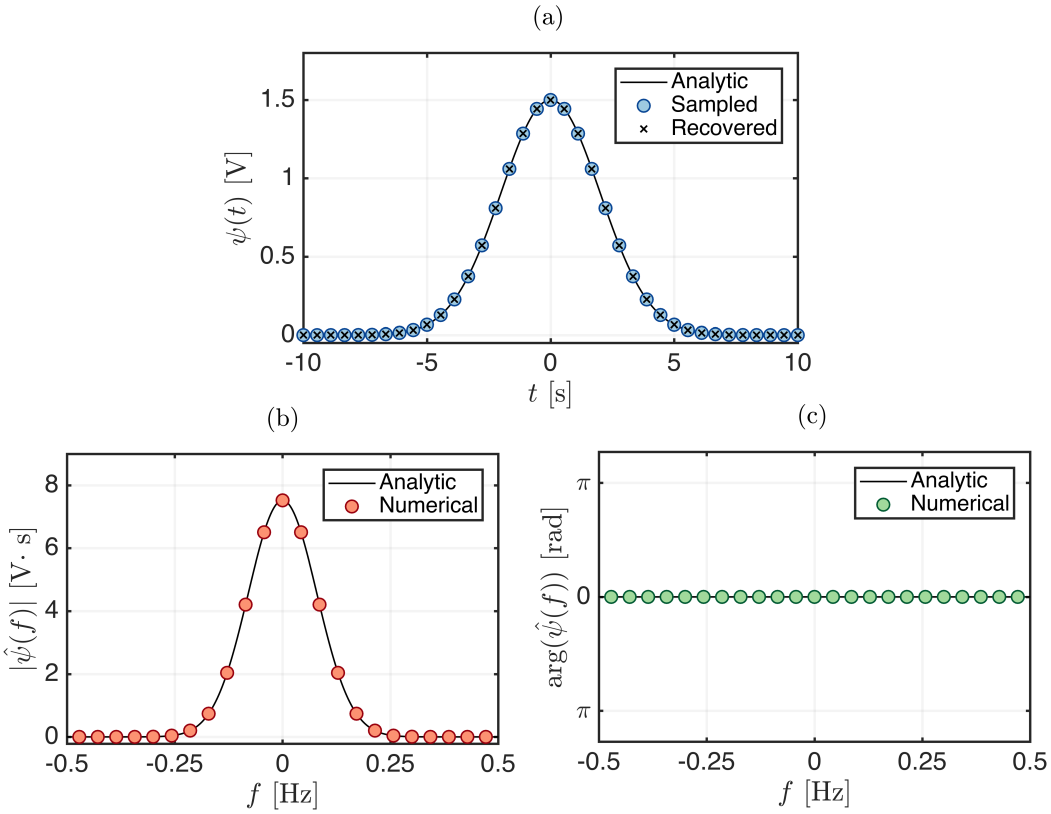

FIGURE 3: Fourier transform of a 1D Gaussian function calculated numerically. (a) An analytic 1D Gaussian function (solid line) is used to sample a series of $N = 42$ points (filled markers) at 1.8 Hz that are used to perform the numerical Fourier transform using an FFT algorithm. The absolute value and phase of the numerical FFT result are shown in (b) and (c), respectively, and agree well with the expected analytic results. Finally, the numerical result from (b) and (c) are used to inverse transform and correctly recover the original input signal, which is plotted as crosses in (a).

which transforms a sequence of N complex numbers into another sequence of complex numbers, where n denotes the indexing for the input vector, and k denotes the indexing for the returned vector. In Matlab, the first index is given by $n = 1$ and $k = 1$ (unlike some other languages which label their first index as 0).

The key thing to remember is that the `fft()` algorithm requires that the sample corresponding to the $t = 0$ time *must* be placed at the first index in the vector. If we are working with a causal signal, i.e. one which is not defined for negative times, then the first value in the vector should correspond to $t = 0$ anyway and there is no problem. However, often we work with non-causal signals, which are defined for negative times as well. The Gaussian in the previous section is an example of this, which is centered on $t = 0$ and exists for both positive and negative times. In these cases, we need to prepare the signal first, before passing it to `fft()`. The `ifftshift()` function is designed exactly for this purpose, and swaps the two halves of the input vector, in order to bring the $t = 0$ point to the first index. Depending on whether the number of points in the vector N is odd or even, the`ifftshift()` command behaves slightly differently:

- If N is even, the `ifftshift()` operation will result in the point at index $(N/2+1)$ being moved to the first index in the vector. For example if $N = 6$,

  `ifftshift([1 2 3 4 5 6]) = [4 5 6 1 2 3].`

- If N is odd, the `ifftshift()` operation will result in the point at index $(N+1)/2$ being moved to the first index in the vector. For example if $N = 5$,

  `ifftshift([1 2 3 4 5]) = [3 4 5 1 2].`

Since we are going to pass our sampled signal into `ifftshift()`, we must take care to define the corresponding time vector in the correct way, so that after shifting we have everything in the right place:

- If we are working with a *causal* signal, i.e. one which does not exist for negative times, then we can define the first index to be $t = 0$, and the rest of the vector should be defined using

  `t = 0:dt:(T-dt)`

  where `T=N*dt`. The zero is already at the first index using this method, and so the sampled vector can then be passed directly into `fft()`, and no shifting is required.

- If we are working with a *non-causal* signal, i.e. one which exists for negative times, and has its $t = 0$ point in the centre of the vector, then the time vector should be defined using

  `t = -(T/2):dt:(T/2-dt)` (If N is even)

  `t = -(T/2-dt/2):dt:(T/2-dt/2)` (If N is odd)

  where again `T=N*dt`. In this case, because the origin of time is placed in the middle of the vector, we will need to perform an `ifftshift()` operation before passing into `fft()` (again, because fft is defined such that it expects the $t = 0$ point to be at the first index). Defining the time vector for a causal signal as above ensures two things: 1) there is always a $t = 0$ point, no matter whether N is off or even; 2) If N is odd the zero will be placed at index $(N + 1)/2$, whilst if N is even the zero will be placed at index $(N/2 + 1)$. This then means that after `ifftshift()` the $t = 0$ point will be placed at the first index (ready to be passed properly to `fft()`), regardless of whether N is odd or even. For example, with `dt=2` and N odd,

```
N = 5;
T = N*dt;
t = -(T/2-dt/2):dt:(T/2-dt/2);
  = [-4 -2 0 2 4]
ifftshift(t) = [0 2 4 -4 -2]
```

Or with N even,

```
N = 6;
T = N*dt;
t = -(T/2):dt:(T/2-dt);
  = [-6 -4 -2 0 2 4]
ifftshift(t) = [0 2 4 -6 -4 -2]
```

In both cases, the $t = 0$ point is sucessfully moved to the first index after applying `ifftshift()`.

If we are working with non-causal (centered) signals, it is critical to apply `ifftshift()` correctly before performing your `fft()`. Without shifting, because the fft interprets the first index as zero time, if you pass in a centered signal then this is equivalent to a time-delayed input. A time delay equates to an additional phase shift in the frequency domain, and so an unwanted phase would appear in the spectrum. This is demonstrated in Fig. 4, for the case of a single impulse at $t = 0$ in both the causal and non-causal versions. The transform of an impulse at $t = 0$ should be a constant value, which is correctly recovered directly in the causal case. However, in the non-causal case, `ifftshift()` *must* be applied before `fft()` to get the correct result, otherwise unwanted phase components appear in the spectrum.

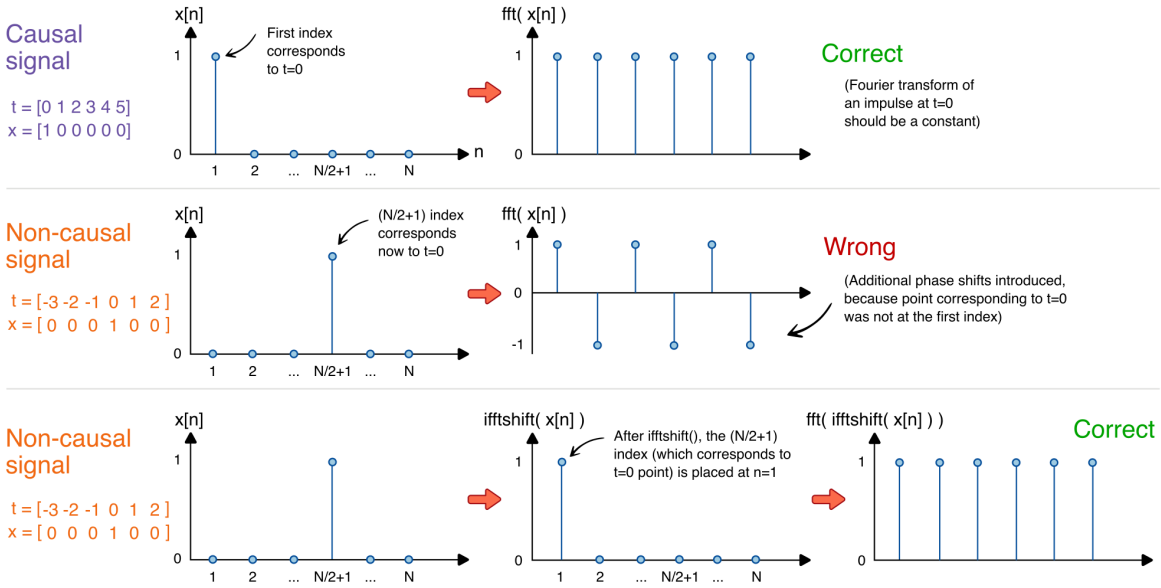

FIGURE 4: Demonstration of the need to perform `ifftshift()` before passing to `fft()` in the case of a non-causal (centered) input signal. When attempting to apply `fft()` directly for a centered signal, additional phase shifts are introduced to the spectrum (middle row).

3 Spectral Derivatives

3.1 First Derivatives

The spectral method is an extremely accurate and computationally efficient way to calculate the derivative of a function by making use of Fourier transforms. Returning to a function of a single variable, time, given by $\psi(t)$, we would like to calculate the derivative

$$\frac{\mathrm{d}\psi(t)}{\mathrm{d}t}. \tag{53}$$

Taking the Fourier transform of the derivative in angular frequency space using Eq. 3, we have

$$\mathcal{F}_t\left\{\frac{\mathrm{d}\psi(t)}{\mathrm{d}t}\right\} = \int_{-\infty}^{+\infty} \frac{\mathrm{d}\psi(t)}{\mathrm{d}t}\ e^{-i\omega t}\,\mathrm{d}t \tag{54}$$

Using the general formula for integration by parts

$$\int_a^b u(x)v'(x)\ \mathrm{d}x = \Big[u(x)v(x)\Big]_a^b - \int_a^b u'(x)v(x)\ \mathrm{d}x \tag{55}$$

we can write

$$\mathcal{F}_t\left\{\frac{\mathrm{d}\psi(t)}{\mathrm{d}t}\right\} = \Big[e^{-i\omega t}\ \psi(t)\Big]_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \psi(t)\ (-i\omega)\ e^{-i\omega t}\,\mathrm{d}t. \tag{56}$$

The first term must be zero, because for any function to be Fourier-transformable, it must be square integrable, which means that it must go to zero at infinity (note that you can still Fourier transform a sine wave for instance, but that requires some specialised tools - in that case you solve it in terms of the dirac delta function). We then have

$$\mathcal{F}_t\left\{\frac{\mathrm{d}\psi(t)}{\mathrm{d}t}\right\} = i\omega \int_{-\infty}^{+\infty} \psi(t)\ e^{-i\omega t}\,\mathrm{d}t = i\omega\mathcal{F}_t\big\{\psi(t)\big\}. \tag{57}$$

This gives a way of calculating the derivative by finally inverse transforming as follows

$$\boxed{\frac{\mathrm{d}\psi(t)}{\mathrm{d}t} = \mathcal{F}_t^{-1}\Big\{i\omega\mathcal{F}_t\big\{\psi(t)\big\}\Big\}} \tag{58}$$

which requires multiplying the Fourier transformed function by $i\omega t = i2\pi ft$. This operation is much more accurate than finite-difference methods, and can be used when solving differential equations which have derivative terms.

3.2 Calculating a First Derivative Spectrally in 1D Using Matlab

Take as an example the following function

$$\psi(t) = t^3\ e^{-t^2/9}, \tag{59}$$

which tends to zero at infinity, and therefore the Fourier transform exists. This function has the following analytic first derivative, which we can use for checking the result

$$\frac{\mathrm{d}\psi(t)}{\mathrm{d}t} = \left(3 - \frac{2t^2}{9}\right)\, t^2\, e^{-t^2/9}. \tag{60}$$

The following Matlab code calculates this derivative using the spectral method. The code uses the built-in fast-Fourier transform functions `fft()` and `ifft()`, which efficiently calculate the discrete Fourier transform used to approximate the continuous Fourier transform in Eq. (58).

```
1  N = 100;   % Number of samples
2  T = 30;    % Extent of sampling window (aka record length)
3  dt = T/N;  % Sampling period (distance between successive t points)
4  Fs = 1/dt; % Sampling frequency
5  t = (-T/2):dt:(T/2-dt);   % Ensure zero at index (N/2+1) for non-causal signal
6  psi = exp(-t.^2/9).*t.^3; % Generate signal
7  figure;plot(t, psi)       % Plot the signal
8
9  df = Fs/N;  % Frequency resolution (separation of points in frequency space)
10 f = (-Fs/2):df:(Fs/2-df); % Generate frequency vector (maximum is Nyquist)
11
12 psi_hat = fftshift( fft( ifftshift(psi) ) )*dt;   % Calculate psi in Fourier space
13 FT = 2*pi*1i*f.*psi_hat;                          % Apply derivative factor
14 dpsi_dt = fftshift( ifft( ifftshift(FT) ) )*N*df; % Inverse transform to real space
15
16 dxdt_exact = (3 - 2*t.^2/9).*t.^2.*exp(-t.^2/9);  % Analytic result for comparison
17
18 figure;hold all;
19 plot(t, dpsi_dt)     % Plot the numerical spectral derivative
20 plot(t, dxdt_exact)  % Plot the analytic derivative
```

Some notes about the code:

- Line 5 - It is possible to deal with two types of signals. First, there are *causal* signals, which are assumed to be zero for all times $t < 0$, i.e something caused the signal to come into existence at $t = 0$. Secondly, there are *non-causal* signals, which can exist for all times. The signal defined in this code, as can be seen in Fig. 5, is symmetric around $t = 0$, and must be defined for negative times. In Matlab, for non-causal signals, the point representing $t = 0$ must be placed at index $(N/2 + 1)$ if N is even , where N is the length of the vector. Defining the vector using `t = (-T/2):dt:(T/2-dt)` ensures this.

- Line 10 - The frequency vector definition follows a similar reasoning to the time vector, and for an even length vector the DC ($f = 0$) point must be again placed at index $(N/2+1)$. For an even length vector, the maximum frequency that can be present is the Nyquist, $F_N = F_s/2$, which is equal to half of the sampling frequency, and we know that the frequency vector must increment in steps of `df`.

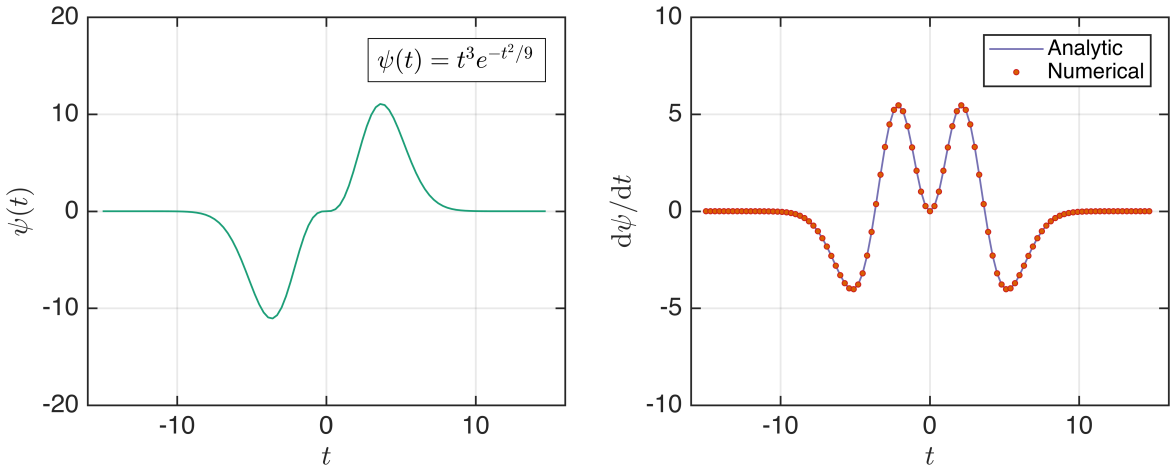

FIGURE 5: Example calculating a derivative using the spectral method. (left) Input signal to be differentiated - a polynomial modulated by a Gaussian envelope. (right) The derivative calculated through the spectral method using Matlab's `fft()` discrete Fourier transform algorithm, with the analytic result shown for comparison.

- Line 12 - The `fft()` algorithm expects the $t = 0$ point to be the first index in the vector (more specifically, it defines this to be the reference point of zero phase shift). Since our signal is non-causal, our time vector was defined shrewdly so that the zero time point is actually at index $(N/2+1)$. The Matlab function `ifftshift()` swaps the two halves of the vector, bringing the $(N/2+1)$ point now to the first index. For example `ifftshift([1 2 3 4 5 6])` results in `[4 5 6 1 2 3]`.

  After applying `ifftshift()` to the input vector, then `fft()` can be correctly performed (otherwise, an additional phase shift would result). After executing the `fft()`, the output contains the DC term first, followed by all the positive frequencies ($f \geq 0$), then followed by the negative frequencies ($f < 0$). The frequency spectrum is usually displayed opposite to this, with negative frequencies on the left, then DC in the centre, followed by positive frequencies on the right. In addition, the order must match how we have defined our frequency vector in Line 10. For this reason, the function `fftshift()` is finally applied to centre the transform after the `fft()` operation. Note that we used `ifftshift()` to "undo" the centering which was present in the original signal, whilst we used `fftshift()` to make the signal centered. For even length vectors, these two functions are identical, but if using an odd length vector you must make sure the correct one is used in the right situation.

  Finally, there is a multiplication by a factor of `dt`, which is necessary because we are using the Matlab discrete Fourier transform algorithm to approximate the continuous Fourier transform, using a Riemann sum (compare Eq. 1 with the FFT definition in the Matlab help documents.)

- Line 14 - The inverse transform follows exactly the same logic as the forward transform above, except that now we are approximating the continuous inverse Fourier transform of Eq. 2. Again comparing with the Matlab help definition of `ifft()`, we see that factors of `df` and N are necessary because of how Matlab defines its FFT (note that this convention is different in other

programming languages).

3.3 Second Derivatives

If we instead want to calculate the second derivative of the function $\psi(t)$, i.e

$$\frac{\mathrm{d}^2\psi(t)}{\mathrm{d}t^2}, \tag{61}$$

then we can write this as

$$\frac{\mathrm{d}^2\psi(t)}{\mathrm{d}t^2} = \frac{\mathrm{d}v(t)}{\mathrm{d}t}, \qquad \text{where} \quad v(t) = \frac{\mathrm{d}\psi(t)}{\mathrm{d}t}. \tag{62}$$

Then, using Eq. 58, this can be expressed as

$$\mathcal{F}_t\left\{\frac{\mathrm{d}^2\psi(t)}{\mathrm{d}t^2}\right\} = \mathcal{F}_t\left\{\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}\psi(t)}{\mathrm{d}t}\right\} \tag{63}$$

$$= \mathcal{F}_t\left\{\frac{\mathrm{d}v(t)}{\mathrm{d}t}\right\} \tag{64}$$

$$= i\omega\ \mathcal{F}_t\big\{v(t)\big\} \tag{65}$$

$$= i\omega\ \mathcal{F}_t\left\{\frac{\mathrm{d}\psi(t)}{\mathrm{d}t}\right\} \tag{66}$$

$$= i\omega \cdot i\omega\ \mathcal{F}_t\big\{\psi(t)\big\} \tag{67}$$

$$= -\omega^2\ \mathcal{F}_t\big\{\psi(t)\big\}. \tag{68}$$

Therefore the second derivative is simply an extension of the first derivative with an extra factor of $i\omega$

$$\boxed{\frac{\mathrm{d}^2\psi(t)}{\mathrm{d}t^2} = \mathcal{F}_t^{-1}\Big\{-\omega^2\ \mathcal{F}_t\big\{\psi(t)\big\}\Big\}} \tag{69}$$

The Matlab code in Sec. 3.2 is easily altered to calculate a second derivative, with an extra factor of $2\pi i f = i\omega$ on Line 12.

3.4 Calculating a Laplacian Spectrally in 2D Using Matlab

Consider a function of two variables $\psi(x, y)$, where

$$\psi(x, y) = \Big(x + y^2x^2 + 2xy\Big)\ e^{-x^2}\ e^{-y^2/2}, \tag{70}$$

which is plotted in Fig. 6. The Laplacian in 2D is given by

$$\nabla^2\psi(x, y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\psi(x, y), \tag{71}$$

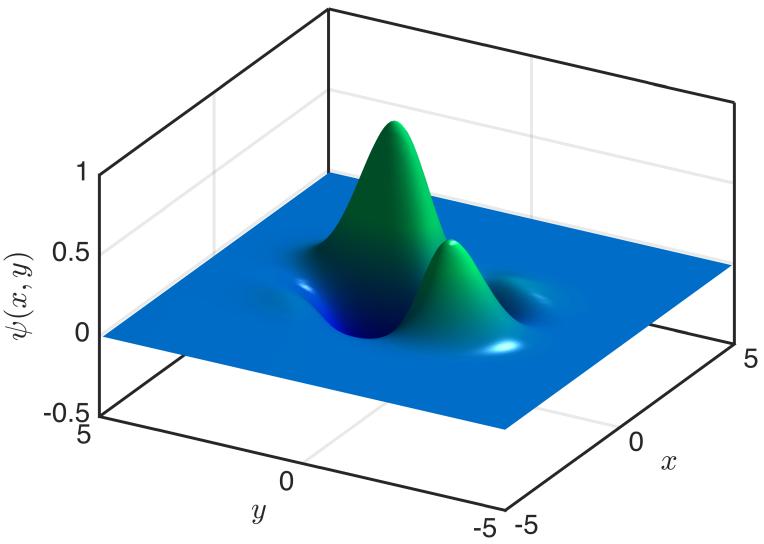

FIGURE 6: Example function of two variables $\psi(x, y)$, which is used for demonstrating calculating a 2D Laplacian $\nabla^2\psi(x, y)$ using the spectral method.

which can be calculated analytically, as a way of verifying the method, to be

$$\begin{aligned}\nabla^2\psi(x,y) &= e^{-x^2}\ e^{-y^2/2} \times ... \\ &... \left(4x^4y^2 + x^3\left(8y+4\right) + x^2\left(y^4 - 15y^2 + 2\right) + x\left(2y^3 + y^2 - 18y - 7\right) + 2y^2\right).\end{aligned} \tag{72}$$

To calculate this using the spectral method, the 2D Laplacian can be written using Eq. 69 as

$$\nabla^2\psi(x,y) = \frac{\partial^2\psi(x,y)}{\partial x^2} + \frac{\partial^2\psi(x,y)}{\partial y^2} \tag{73}$$

$$= \mathcal{F}_x^{-1}\Big\{ -k_x^2\ \mathcal{F}_x\left\{\psi(x,y)\right\}\Big\} + \mathcal{F}_y^{-1}\Big\{ -k_y^2\ \mathcal{F}_y\left\{\psi(x,y)\right\}\Big\} \tag{74}$$

$$= \mathcal{F}_x^{-1}\Big\{ -k_x^2\ \hat{\psi}(k_x,y)\Big\} + \mathcal{F}_y^{-1}\Big\{ -k_y^2\ \hat{\psi}(x,k_y)\Big\}, \tag{75}$$

where k_x and k_y are the angular spatial frequencies in the x and y directions, respectively, and the Fourier transform in the x and y directions are given by

$$\mathcal{F}_x\left\{\psi(x,y)\right\} = \hat{\psi}(k_x,y) = \int_{-\infty}^{+\infty} \psi(x,y)\ e^{-ik_x x}\,\mathrm{d}x \tag{76}$$

$$\mathcal{F}_y\left\{\psi(x,y)\right\} = \hat{\psi}(x,k_y) = \int_{-\infty}^{+\infty} \psi(x,y)\ e^{-ik_y y}\,\mathrm{d}y. \tag{77}$$

In order to calculate the Laplacian in 2D, this shows that we need to transform into reciprocal space for both coordinates, and multiply by the corresponding k-vectors.

The following Matlab code calculates the Laplacian of the function in Eq. 70, and the result, along with the analytic function for comparison, is shown in Fig. 7

```
1  Nx = 100;     % Number of samples in x
2  Ny = 120;     % Number of samples in y
3  Lx = 10;      % Extent of sampling window in x
4  Ly = 10;      % Extent of sampling window in y
5  dx = Lx/Nx;   % Grid spacing in x
6  dy = Ly/Ny;   % Grid spacing in y
7  x = (-Lx/2):dx:(Lx/2-dx);   % Ensure zero at index (N/2+1) for centred signal
8  y = (-Ly/2):dy:(Ly/2-dy);   % Ensure zero at index (N/2+1) for centred signal
9  [X,Y] = meshgrid(x,y);
10
11 psi = (X + X.^2.*Y.^2 + 2*Y.*X).*exp(-X.^2).*exp(-Y.^2/2);
12
13 % Analytic Laplacian result, for comparison
14 laplacian_psi_exact = (4*X.^4.*Y.^2 + x.^3.*(8*Y + 4) + X.^2.*(Y.^4-15*Y.^2+2) + ...
15                        X.*(2*Y.^3+Y.^2-18*Y-7) +2*Y.^2).*exp(-X.^2).*exp(-Y.^2/2);
16
17 Fsx = 1/dx;   % Sampling frequency in x
18 Fsy = 1/dy;   % Sampling frequency in y
19 dfx = Fsx/Nx; % Frequency resolution in x
20 dfy = Fsy/Ny; % Frequency resolution in y
21 fx = (-Fsx/2):dfx:(Fsx/2-dfx); % Frequency vector in x
22 fy = (-Fsy/2):dfy:(Fsy/2-dfy); % Frequency vector in y
23 kx = 2*pi*fx; % k vector in x
24 ky = 2*pi*fy; % k vector in y
25
26 % Calculate second derivative with respect to x, by Fourier transforming into
27 % kx space, and back again to real space:
28 psi_hatx =  fftshift(  fft( ifftshift(psi,2) ,Nx,2 ) ,2)*dx;
29 FTx = -kx.^2.*psi_hatx;
30 d2psi_dx2 = fftshift( ifft( ifftshift(FTx,2) ,Nx,2 ) ,2)*Nx*dfx;
31
32 % Calculate second derivative with respect to y, by Fourier transforming into
33 % ky space, and back again to real space:
34 psi_haty =  fftshift(  fft( ifftshift(psi,1) ,Ny,1 ),1 )*dy;
35 FTy = -(ky').^2.*psi_haty;
36 d2psi_dy2 = fftshift( ifft( ifftshift(FTy,1) ,Ny,1 ),1 )*Ny*dfy;
37
38 laplacian_psi = d2psi_dx2 + d2psi_dy2;
39 laplacian_psi_real = real(laplacian_psi);
```

Some notes about the code:

- Line 28 - The Fourier transform and shifting is done with respect to only the x-direction, indicated by the 2 in the argument which instructs to Fourier transform each row.
- Line 35 - We have to take the transpose of the `ky` vector, because the multiplication must be done column by column.

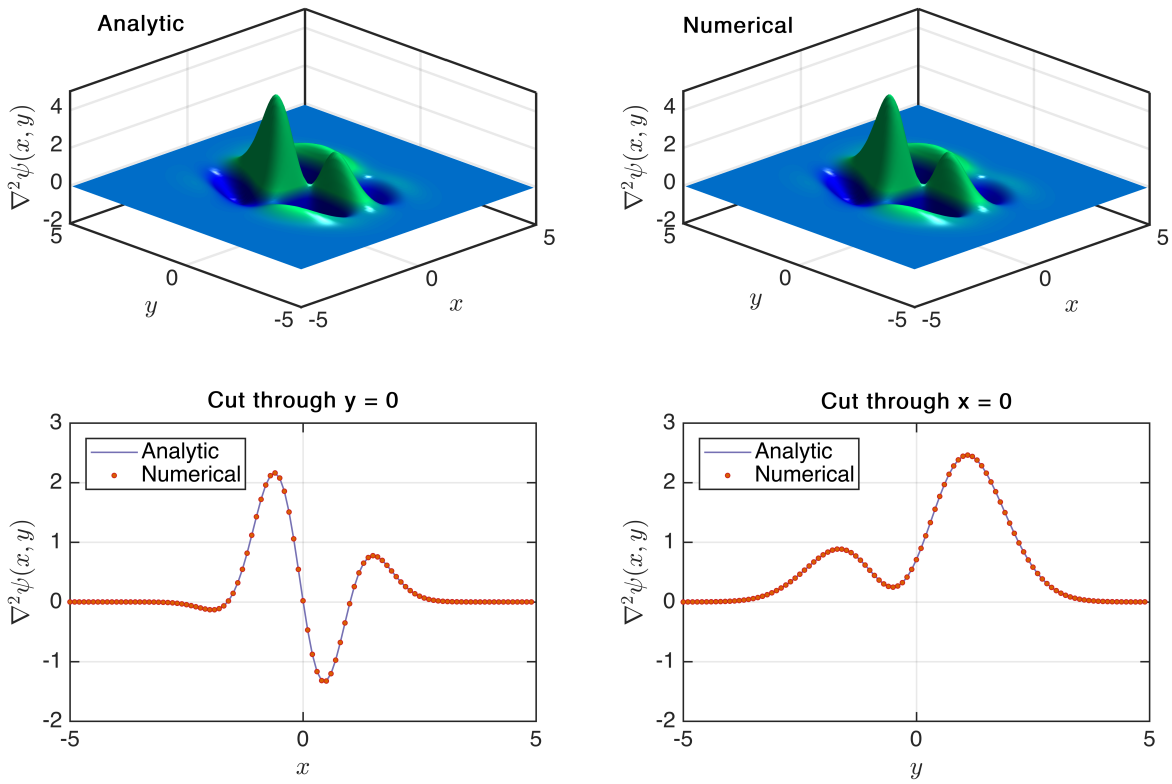

FIGURE 7: Comparison between the analytic and spectral numerical calculations of the Laplacian $\nabla^2\psi(x,y)$ of the function in Eq. 70.

Note that there are some improvements we can make to the previous code, which does not require individual Fourier transforms in each of x and y. The previous code was written out explicitly to demonstrate the method, but the 2D Laplacian can also be written in terms of a 2D Fourier transform

$$\nabla^2\psi(x,y) = \mathcal{F}_{2\mathrm{D}}^{-1}\Big\{ -\left(k_x^2 + k_y^2\right) \cdot \mathcal{F}_{2\mathrm{D}}\left\{\,\psi(x,y)\,\right\}\Big\} \tag{78}$$

$$= \mathcal{F}_{2\mathrm{D}}^{-1}\Big\{ -\left(k_x^2 + k_y^2\right) \cdot \hat{\psi}(k_x,k_y)\Big\}, \tag{79}$$

where the two-dimensional Fourier forward and inverse transforms are given in Eq. 9 and Eq. 10.

The following code instead carries out the Fourier transform in 2D, and gives the same result as the more verbose version:

```
1 [KX,KY] = meshgrid(kx,ky);                              % Create a 2D grid of k-vectors
2 psi_hat = fftshift( fft2( ifftshift(psi) ) )*dx*dy; % Perform 2D Fourier transform
3 FT = -(KX.^2 + KY.^2).*psi_hat;                         % Calculate Laplacian in k-space
4 laplacian_psi = fftshift( ifft2( ifftshift(FT) ))*Nx*Ny*dfx*dfy;
```

Finally, the code can be simplified even further. Firstly, the factor on the inverse transform on Line 4 of `Nx*Ny*dfx*dfy` is actually equal to `(1/dx)*(1/dy)`, because `dfx = Fsx/Nx` and `dfy = Fsy/Ny`. Therefore, this factor cancels with that applied to the forward transform on Line 2, and so can be omitted as long as we always perform a forward transform followed later by an inverse transform (this is the convenience of Matlab's `ifft()` definition including a a/N factor). Additionally, we can apply the `ifftshift()` function instead to the k-matrix at the beginning, instead of applying it every time (usually inside a loop) to the output of the fft itself. As long as we multiply the k-vector components with the correct corresponding points in the transformed matrix, then it doesn't matter.

The following code produces the same result as the other versions, but is much more slimmed down and executes approximately twice as fast as the original version:

```
1 [KX,KY] = meshgrid(kx,ky); % Create a 2D grid of k-vectors
2 K = KX.^2 + KY.^2;         % Calculate the magnitude of k, required by the Laplacian
3 K = ifftshift(K);          % Shift the k vectors instead of the fft output
4
5 psi_hat = fft2( ifftshift(psi) );       % Bring the function into kx and ky space
6 FT = -K.*psi_hat;                       % Calculate Laplacian in k-space
7 laplacian_psi = fftshift( ifft2( FT )); % Convert back to real space
```

4 Numerical Solutions of Differential Equations

4.1 A Simple, First-Order ODE

Consider a simple, first-order ordinary differential equation of the form

$$\frac{\mathrm{d}\psi(t)}{\mathrm{d}t} = -\frac{1}{\tau}\psi(t), \tag{80}$$

with the initial condition

$$\psi(t=0) = \psi_0. \tag{81}$$

Here, the function $\psi(t)$ could represent, for example, the number of atoms contained in a magnetic trap over time, and the value ψ_0 defines the number of atoms initially trapped at $t = 0$. The task of numerically solving this equation is to find $\psi(t)$, i.e to find how the function evolves over time, given that it is governed by the above differential equation. The analytic solution of this equation is well known to be an exponential decay (as long as $\tau > 0$), given by

$$\psi(t) = \psi_0\, e^{-t/\tau}. \tag{82}$$

This is because as seen in Eq. 80, the rate of change of ψ at any time t is proportional to the actual value of ψ itself (again, if this was number of atoms in a trap, the rate at which you lose atoms should be proportional to the number of atoms that are actually currently trapped).

The first step to numerically solving this equation on the computer is to discretize the time coordinate onto a grid. We can define a discrete version ψ_k of the function, which is evaluated at specific times

$$\psi_k = \psi[k] := \psi(t = t_k) = \psi\left(t = (k-1)\cdot\Delta t\right), \tag{83}$$

where Δt is the spacing between the discrete time points, and $k = 1, 2, 3, ..., N$ are integers (following Matlab's indexing convention of the first element in a vector being denoted by 1). This leads to

$$\psi_1 = \psi\Big(t = 0\Big) \tag{84}$$

$$\psi_2 = \psi\Big(t = \Delta t\Big) \tag{85}$$

$$\psi_3 = \psi\Big(t = 2\Delta t\Big) \tag{86}$$

$$\vdots \tag{87}$$

$$\psi_N = \psi\Big(t = (N-1)\cdot\Delta t\Big). \tag{88}$$

The indexing is illustrated in Fig. 8.

Now we need a way to approximate the derivative in Eq. 80. This can be done by recalling the definition of a Taylor expansion of some function $f(x)$ is given in terms of the function at some point

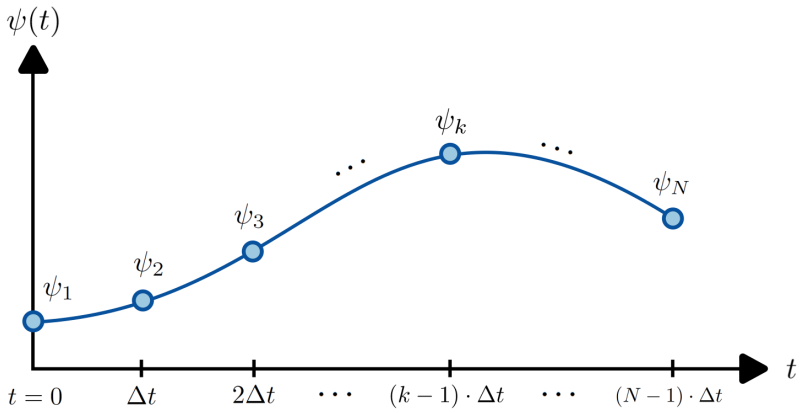

FIGURE 8: Solution of the simple first order ODE, which has an exponentially decaying solution, using forward Euler method in time. Analytic result is compared to numerical calculation for two different time steps. The smaller the timestep, the more accurate the result.

$x = a$ by

$$f(x) = f(a) + \frac{(x-a)}{1!}\frac{\mathrm{d}f}{\mathrm{d}x}\bigg|_{x=a} + \frac{(x-a)^2}{2!}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\bigg|_{x=a} + \ldots \tag{89}$$

This means that if you know the properties of the function at $x = a$, you can approximate the value of the function at some general point $f(x)$ in terms of the derivatives at a. So, in terms of the differential equation, if we know the value of our function ψ at some time t^*, we can write down an expression for ψ at some general time t as

$$\psi(t) \approx \psi(t^*) + (t-t^*)\frac{\mathrm{d}\psi}{\mathrm{d}t}\bigg|_{t=t^*} + \frac{(t-t^*)^2}{2}\frac{\mathrm{d}^2\psi}{\mathrm{d}t^2}\bigg|_{t=t^*} + \ldots \tag{90}$$

Now, if we don't consider any general t, but focus specifically on a time which is Δt away from t^*, i.e. $t = t^* + \Delta t$, as illustrated in Fig. 9, we can write

$$\psi(t^* + \Delta t) \approx \psi(t^*) + \Delta t\frac{\mathrm{d}\psi}{\mathrm{d}t}\bigg|_{t=t^*} + \frac{\Delta t^2}{2}\frac{\mathrm{d}^2\psi}{\mathrm{d}t^2}\bigg|_{t=t^*} + \ldots \tag{91}$$

If we let t^* be the k'th time coordinate in the discretised time vector, then we have $t^* = t_k$ and therefore $t^* + \Delta t = k_k + \Delta t = t_{k+1}$. This leads to

$$\psi(t_{k+1}) \approx \psi(t_k) + \Delta t\frac{\mathrm{d}\psi}{\mathrm{d}t}\bigg|_{t=t_k} + \frac{\Delta t^2}{2}\frac{\mathrm{d}^2\psi}{\mathrm{d}t^2}\bigg|_{t=t_k} + \ldots \tag{92}$$

But we have already defined $\psi(t_k) = \psi_k$, i.e the k'th entry in the ψ vector, so

$$\psi_{k+1} \approx \psi_k + \Delta t\frac{\mathrm{d}\psi_k}{\mathrm{d}t} + \frac{\Delta t^2}{2}\frac{\mathrm{d}^2\psi}{\mathrm{d}t^2} + \ldots \tag{93}$$

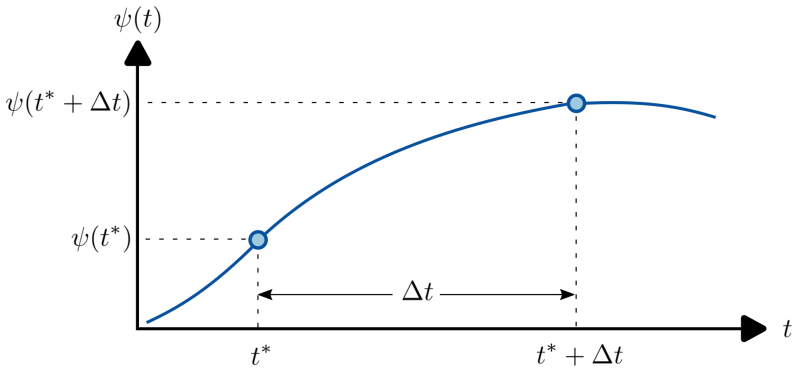

FIGURE 9: Illustration of time coordinates for approximating a function at some time $t = t^* + \Delta t$ in terms of the function at time $t = t^*$.

Rearranging this and dropping all terms Δt^2 and above, we obtain an approximation for the derivative at the k'th index in terms of the function and the neighbour at $k + 1$

$$\boxed{\frac{\mathrm{d}\psi_k}{\mathrm{d}t} \approx \frac{\psi_{k+1} - \psi_k}{\Delta t}.} \tag{94}$$

This equation is called the *forward difference in time* approximation, because it calculates the derivative at a given point based on the function at that point together with the next neighbour. In discrete space, the differential equation Eq. 80 becomes

$$\frac{\mathrm{d}\psi(t_k)}{\mathrm{d}t} = -\frac{1}{\tau}\psi(t_k) \tag{95}$$

$$\implies \frac{\mathrm{d}\psi_k}{\mathrm{d}t} = -\frac{1}{\tau}\psi_k = f(\psi_k), \tag{96}$$

i.e at each point in time t_k the equation must hold. The function $f(\psi_k)$ defines the right hand side of the equation, and can be more complicated in general than the example here. Rearranging the forward finite difference formula, we can write

$$\frac{\mathrm{d}\psi_k}{\mathrm{d}t} = f(\psi_k) \approx \frac{\psi_{k+1} - \psi_k}{\Delta t} \tag{97}$$

$$\boxed{\psi_{k+1} \approx \psi_k + \Delta t f(\psi_k).} \tag{98}$$

This updating formula provides a numerical way to obtain the value of ψ_{k+1} in an iterative fashion, based on the previous value, and is known as the *forward Euler method*. This is equivalent to taking the slope of the function at some time, and then extrapolating a straight line to estimate where the next point should reside. For the specific example in Eq. 80, the updating formula becomes

$$\psi_{k+1} \approx \psi_k - \frac{\Delta t}{\tau}\psi_k. \tag{99}$$

The following Matlab code solves Eq. 80 using the described forward finite difference method, and the result is shown in Fig. 10 for two different time steps. It can be seen that smaller time steps in general increase the accuracy of the forward Euler method, at the cost of computational overhead.

```
dt = 0.4;       % Time step
T = 20;         % Duration of total time window

t = 0:dt:T-dt; % Define time vector
N = length(t); % Number of time iterations

tau = 10; % Time constant in RHS of equation
f = @(psi,tau) -(1/tau)*psi; % Function for RHS of equation

psi = zeros([1 N]); % Preallocate array for speed
psi0 = 20;          % Initial condition at t=0
psi(1) = psi0;      % Put the first value into the vector

% Iterate the equation over steps of dt
for k = 1:(N-1)
    psi(k+1) = psi(k) + f(psi(k),tau)*dt; % Forward Euler method
end

t_analytic = linspace(0,T,1000);
psi_analytic = 20*exp(-t_analytic/tau);

figure;hold all
plot(t_analytic,psi_analytic )
plot(t,psi,'.')
```

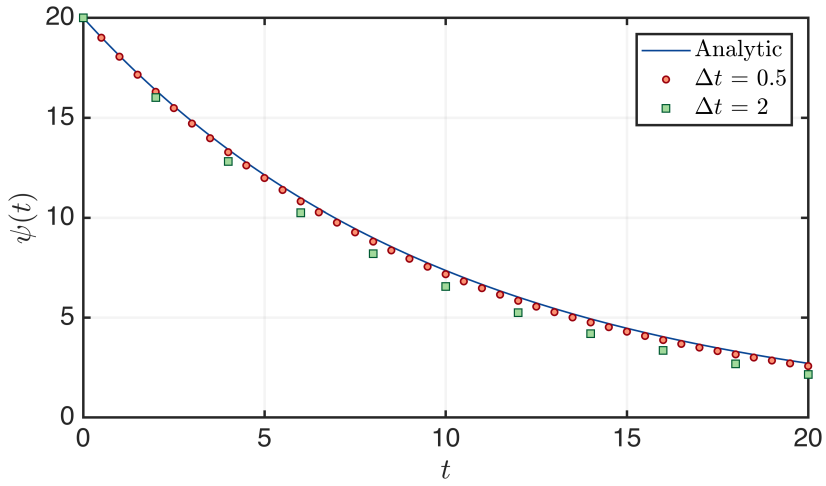

FIGURE 10: Solution of the simple first order ODE, which has an exponentially decaying solution, using forward Euler method in time. Analytic result is compared to numerical calculation for two different time steps. The smaller the timestep, the more accurate the result.

4.2 A PDE Example - The 1D Heat Equation

I will now look at an example of a partial differential equation. Consider a function of both time and one-dimension of space, denoted by $\psi(x,t)$. The 1D heat equation is given by

$$\frac{\partial \psi(x,t)}{\partial t} = \alpha \frac{\partial^2 \psi(x,t)}{\partial x^2}, \tag{100}$$

where α is a constant. This is a specific example of the diffusion equation, and the function $\psi(x,t)$ would usually represent the temperature distribution in a material. Loosely, this equation says that the rate at which material at a given point heats/cools is proportional to how much hotter/colder the material surrounding that point it. The constant α is called the thermal diffusivity of the material, and depends on the thermal conductivity, specific heat capacity, and mass density.

This is an example of a *partial differential equation*, because it contains derivatives with respect to two variables x and t. In order to solve it numerically, we can handle the time derivative in the same way as before - approximate the first derivative with a finite difference after discretizing the time variable

$$\psi(x, t_{k+1}) \approx \psi(x,t_k) + \Delta t \cdot f\left(\psi(t_k)\right) \tag{101}$$

$$\approx \psi(x,t_k) + \Delta t\ \alpha \frac{\partial^2 \psi(x,t_k)}{\partial x^2}. \tag{102}$$

So, in order to advance to the next time step, we need a way to calculate the second derivative. From Eq. 91, we can write a Taylor expansion in the spatial coordinate instead around a point x^*, at a fixed time t_k, as

$$\psi(x^* + \Delta x, t_k) \approx \psi(x^*, t_k) + \Delta x \frac{\mathrm{d}\psi}{\mathrm{d}x}\bigg|_{x=x^*} + \frac{\Delta x^2}{2} \frac{\mathrm{d}^2\,\psi}{\mathrm{d}x^2}\bigg|_{x=x^*} + ... \tag{103}$$

Similarly, if we look at a point $x^* - \Delta x$, we obtain

$$\psi(x^* - \Delta x, t_k) \approx \psi(x^*, t_k) - \Delta x \frac{\mathrm{d}\psi}{\mathrm{d}x}\bigg|_{x=x^*} + \frac{\Delta x^2}{2} \frac{\mathrm{d}^2\,\psi}{\mathrm{d}x^2}\bigg|_{x=x^*} + ... \tag{104}$$

Adding these two equations together, rearranging, and dropping the higher order terms, the first derivative terms cancel and we get

$$\frac{\mathrm{d}^2\,\psi}{\mathrm{d}x^2}\bigg|_{x=x^*} \approx \frac{\psi(x^* + \Delta x, t_k) + \psi(x^* - \Delta x, t_k) - 2\psi(x^*, t_k)}{(\Delta x)^2}. \tag{105}$$

We can now discretize the x-grid into N_x individual points x_m, in steps of Δx, in such a way that the $x = 0$ point is at the middle of the vector (instead of at the beginning)

$$x_m = x[m] := \left(m - 1 - \frac{N_x}{2}\right) \cdot \Delta x \qquad \text{(if } N_x \text{ is even)} \tag{106}$$

$$x_m = x[m] := \left(m - 1 - \frac{N_x - 1}{2}\right) \cdot \Delta x \quad \text{(if } N_x \text{ is odd)} \tag{107}$$

$$\tag{108}$$

where $m = 1, 2, 3, ..., N_x$ are integers. This provides the following discrete x coordinates

- if N_x is even

$$x_1 = -\frac{N_x}{2}\Delta x \tag{109}$$
$$x_2 = -\frac{N_x}{2}\Delta x + \Delta x \tag{110}$$
$$x_3 = -\frac{N_x}{2}\Delta x + 2\Delta x \tag{111}$$
$$\vdots \tag{112}$$
$$x_N = +\frac{N_x}{2}\Delta x - \Delta x \tag{113}$$

- if N_x is odd

$$x_1 = -\frac{(N_x - 1)}{2}\Delta x \tag{114}$$
$$x_2 = -\frac{(N_x - 1)}{2}\Delta x + \Delta x \tag{115}$$
$$x_3 = -\frac{(N_x - 1)}{2}\Delta x + 2\Delta x \tag{116}$$
$$\vdots \tag{117}$$
$$x_N = +\frac{(N_x - 1)}{2}\Delta x \tag{118}$$

Examples of what the x-grid would look like for wither $N_x = 5$ (odd) and $N_x = 6$ (even) are shown in Fig. 11. It is defined in such a way that *both* cases contain an $x = 0$ point - in the odd case it is located at index $m = (N_x + 1)/2$ and in the even case at $m = (N_x/2) + 1$ (this is important when applying FFT() and fftshift() to the vectors later).

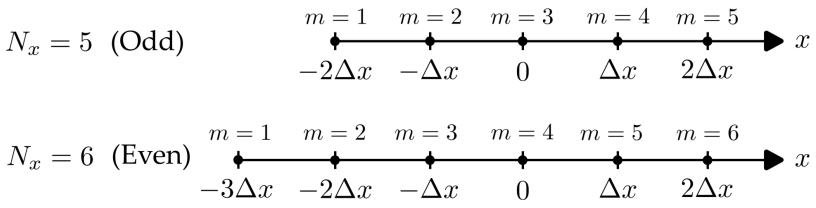

FIGURE 11: Example of the x-grid locations for $N_x = 5$ (odd) and $N_x = 6$ (even) number of grid points, showing which index m corresponds to which physical position.

Returning now to Eq. 105, we can let x^* represent the m'th x grid point, so that $x^* = x_m$ and therefore $x^* + \Delta x = x_{m+1}$ and $x^* - \Delta x = x_{m-1}$. Then the approximation for the second derivative at time t_k is given by

$$\boxed{\left.\frac{\mathrm{d}^2\,\psi(x,t_k)}{\mathrm{d}x^2}\right|_{x=x_m} \approx \frac{\psi(x_{m+1},t_k) + \psi(x_{m-1},t_k) - 2\psi(x_m,t_k)}{(\Delta x)^2}.} \tag{119}$$

This is called the *central difference approximation for the second derivative*, because it calculates the derivative at some central point by looking what happens at the two points either side of it. If we define the notation $\psi_m^k = \psi(x_m, t_k)$ to be the value of the function at time $t = t_k$ at position $x = x_m$, then this can be written more compactly as

$$\frac{\mathrm{d}^2 \psi_m^k}{\mathrm{d}x^2} \approx \frac{\psi_{m+1}^k + \psi_{m-1}^k - 2\psi_m^k}{(\Delta x)^2}. \tag{120}$$

Looking again at the original PDE in Eq. 100, we now have our x variable discretized, and since we know that the differential equation holds at *all* x locations we can examine the equation at only a single one of them x_m

$$\frac{\partial \psi(x_m, t)}{\partial t} = \alpha \frac{\partial^2 \psi(x, t)}{\partial x^2}\bigg|_{x=x_m}, \tag{121}$$

and using the central difference approximation for the second derivative at time t (noting there is no difference between partial and full derivatives here because the variables t and x do not depend on each other), this then becomes

$$\frac{\partial \psi(x_m, t)}{\partial t} \approx \alpha \frac{\psi(x_{m+1}, t) + \psi(x_{m-1}, t) - 2\psi(x_m, t)}{(\Delta x)^2}. \tag{122}$$

Since we have $m = 1, 2, 3, ...N_x$, this shows that by applying the finite difference approximation in space, x, we have reduced the partial differential equation to a system of N_x *ordinary differential equations*, which are simple to solve. Applying the forward Euler method to approximate finite difference in time, we have

$$\psi(x_m, t_{k+1}) \approx \psi(x_m, t_k) + \Delta t\ \alpha\ \frac{\psi(x_{m+1}, t_k) + \psi(x_{m-1}, t_k) - 2\psi(x_m, t_k)}{(\Delta x)^2}, \tag{123}$$

which can be written in the compact $\psi(x_m, t_k) = \psi_m^k$ notation as

$$\boxed{\psi_m^{k+1} \approx \psi_m^k + \Delta t\ \alpha\ \frac{\psi_{m+1}^k + \psi_{m-1}^k - 2\psi_m^k}{(\Delta x)^2}.} \tag{124}$$

This method is called the *forward time, centered space* (FTCS) scheme for solving a partial differential equation. A finite difference scheme becomes numerically unstable if the numerical errors become magnified and accumulate between successive time iterations. It can be shown using von Neumann stability analysis that the FTCS scheme for this 1D heat equation is numerically stable if

$$\frac{\alpha \Delta t}{(\Delta x)^2} \leq \frac{1}{2}, \tag{125}$$

indicating that the grid steps need to be chosen appropriately in order for this to work.

The following Matlab code solves the 1D heat equation using the FTCS method:

```
%%% Set up discrete time grid %%%
dt = 0.01;      % Time step
T = 40;         % Duration of total time window
t = 0:dt:T-dt;  % Define time vector
Nt = length(t); % Number of time iterations

%%% Set up discrete space grid %%%
Nx = 200;       % Number of x grid points
dx = 0.5;       % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);

if Nx_IS_EVEN; x = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

alpha = 1;  % Diffusion coefficient
disp(['Stability parameter = ',num2str(alpha*dt/(dx^2))])

% Preallocate arrays for speed
psi = zeros([Nt Nx]);
d2psi_dx2 = zeros([1,Nx]);

% Define initial condition
psi0 = zeros([1,Nx]);
psi0( x<=10 & x>=-10 ) = 1; % Set some values initially to 1
psi(1,:) = psi0;            % Put the first psi(x,t=0) into the matrix


% Iterate the equation over steps of dt
for k = 1:(Nt-1)

    % Calculate d2psi_dx2
    for m = 2:(Nx-1)
    d2psi_dx2(1,m) = ( psi(k,m+1) + psi(k,m-1) - 2*psi(k,m) )/(dx^2);
    end

    % Forward Euler method in time
    psi(k+1,:) = psi(k,:) + alpha*d2psi_dx2*dt;
end

%%% Plotting %%%
figure;subplot(1,2,1)
waterfall(x, t(1:100:end), psi(1:100:end,:));

subplot(1,2,2);hold all
plot(x,psi(1,:))
plot(x,psi(round(Nt/50),:))
plot(x,psi(round(Nt/5),:))
plot(x,psi(end,:))
```

Note that in this example, the space grid has been made large enough that the effects at the edges are not significant, and to solve the problem fully we should also include boundary conditions in the problem. This would then specify exactly what should happen to ψ (and its derivatives) at the edges.

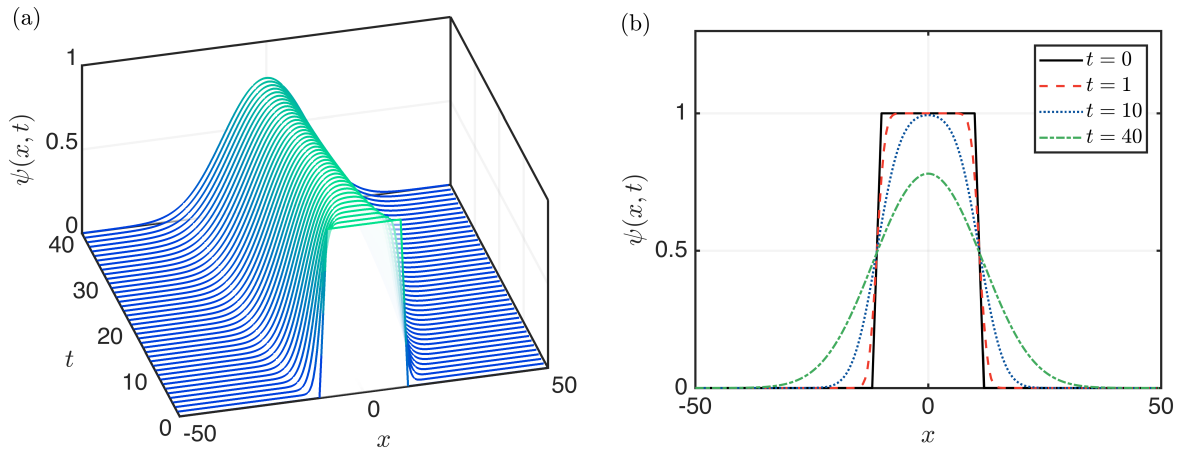

FIGURE 12: Numerical solution $\psi(x,t)$ of the 1D heat equation for $t \in$ [0, 40] and $x \in$ [-50,+50], obtained using the forward time centered space approximation. An initial top hat heat profile at $t = 0$ gradually diffuses into a Gaussian distribution for longer times.

Additionally, the loop over m when calculating the second derivative in space can also be vectorised, i.e. we don't need to go element-by-element to populate the second derivative vector. The following codes are equivalent, but may offer performance differences in different situations (it is not necessarily the case that vectorised code is always faster to execute):

```
%%% Element-by-element loop version %%%
for m = 2:(Nx-1)
    d2psi_dx2(1,m) = ( psi(k,m+1) + psi(k,m-1) - 2*psi(k,m) )/(dx^2);
end

%%% Equivalent vectorised version %%%
d2psi_dx2(1,2:(Nx-1)) = ( psi_k(3:Nx) + psi_k(1:(Nx-2)) - 2*psi_k(2:(Nx-1)) )/(dx^2);
```

Summary

The forward Euler method described so far is an example of an *explicit* method, in that it calculates the state of the system at some later time based on the values at the current time. This is numerically simple to implement, and executes quickly. However, in many problems there are often strict constraints that have to be met in order to ensure stability, requiring a prohibitively small time step to keep the errors small. These types of equations are called *stiff* equations.

Other finite difference methods:

- An alternative is the *backward Euler method*, which evaluates the finite difference approximation of the derivatives at the next time step, instead of the current time step. This procedure is known as an *implicit* method, and is unconditionally stable. The downside is that there is an extra computation step involved, but this can sometimes result in faster code than using an explicit method which has a small enough time step to be stable.
- The *Crank-Nicolson* scheme is essentially a combination of the forward Euler (explicit) and backward Euler (implicit) methods, and has second order accuracy in both space and time, whilst benefiting from the stability of the implicit methods.

4.2.1 Using a Spectral Derivative in a PDE

In the previous section, I used the central finite difference to approximate the second derivative in the right hand side of the 1D heat equation. However, using the result from Sec. 3.3, this second derivative can also be calculated in terms of Fourier transforms. Using Eq. 69, the 1D heat equation becomes

$$\frac{\partial \psi(x,t)}{\partial t} = \alpha \mathcal{F}_x^{-1}\Big\{ -k_x^2 \, \mathcal{F}_x \left\{ \psi(x,t) \right\} \Big\}. \tag{126}$$

The time derivative can then be handled in the same way as before, by applying a simple forward Euler method from Eq. 98. The following Matlab code solves the 1D heat equation similar to the previous section, but now the second derivative is calculated using the fast Fourier transform instead of a finite difference. The spectral derivative is much more accurate in space than a finite difference, even for small grid steps.

```
%%% Set up discrete time grid %%%
dt = 0.01;      % Time step
T = 40;         % Duration of total time window
t = 0:dt:T-dt;  % Define time vector
Nt = length(t); % Number of time iterations

%%% Set up discrete space grid %%%
Nx = 200;       % Number of x grid points
dx = 0.5;       % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);
if Nx_IS_EVEN; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x  = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

%%% Set up discrete angular spatial frequency grid %%%
Fsx = 1/dx;
dfx = Fsx/Nx;
if Nx_IS_EVEN; fx = -(Fsx/2):dfx:(Fsx/2-dfx);
else;          fx = -(Fsx/2-dfx/2):dfx:(Fsx/2-dfx/2);
end
kx = 2*pi*fx;       % k vector in x
kx = ifftshift(kx); % Shift frequencies instead of psi for speed

alpha = 1;  % Diffusion coefficient
```

```
25
26  % Preallocate arrays for speed
27  psi = zeros([Nt Nx]);
28  d2psi_dx2 = zeros([1,Nx]);
29
30  % Define initial condition
31  psi0 = zeros([1,Nx]);
32  inds = x<=10 & x>=-10;
33  psi0( inds ) = 1;      % Set some values initially to 1
34  psi(1,:) = psi0;       % Put the first psi(x,t=0) into the matrix
35
36  % Iterate the equation over steps of dt
37  for k = 1:(Nt-1)
38
39      psi_k = psi(k,:); % Retrieve the previous iteration
40
41      % Calculate d2psi_dx2
42      psi_hatx = fft( ifftshift(psi_k) );
43      FTx = -kx.^2.*psi_hatx;
44      d2psi_dx2 = fftshift( ifft(FTx) );
45
46      % Forward Euler method in time
47      psi(k+1,:) = psi_k + alpha*d2psi_dx2*dt;
48
49  end
50
51  psi = real(psi);
52
53  %%% Plotting %%%
54  figure;subplot(1,2,1)
55  waterfall(x, t(1:100:end), psi(1:100:end,:));
```

4.2.2 Using Runge-Kutta Time Stepping

So far, the Euler method was used in order to calculate the next time step form the current one. This was done by taking the value of the slope at a particular time step, and extrapolating as a straight line to the next point. A more accurate method is to break the time step down and use multiple estimates for the slope including at the midpoint of the step.

The most common way to do this is the *fourth order Runga-Kutta* algorithm, which uses four estimates of the slope and then combines them using a weighted average. If the differential equation is in the form

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = f(t, \psi) \tag{127}$$

then the algorithm to advance from time step with index k to the next time step with index $(k+1)$ is given by

$$t_{k+1} = t_k + \Delta t \qquad \psi_{k+1} = \psi_k + \frac{\Delta t}{6}\left(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4\right) \tag{128}$$

where the Runga-Kutta terms κ are given by

$$\kappa_1 = f(t_k, \psi_k) \tag{129}$$

$$\kappa_2 = f(t_k + \frac{\Delta t}{2}, \psi_k + \frac{\Delta t}{2}\kappa_1) \tag{130}$$

$$\kappa_3 = f(t_k + \frac{\Delta t}{2}, \psi_k + \frac{\Delta t}{2}\kappa_2) \tag{131}$$

$$\kappa_4 = f(t_k + \Delta t, \psi_k + \Delta t\,\kappa_3). \tag{132}$$

Essentially, κ_1 is an estimate of the slope at the beginning of the time step, which amounts to the same as the first order Euler method.The κ_4 is the slope at the end of the time step. The two central ones κ_2 and κ_3 are estimates of the slope at the middle of the time step at time $t_k + \Delta 2/2$, and these are given more weight when piecing all terms back together to calculate ψ_{k+1} using the equation above, because they are expected to be a better estimate of the slope than the end points.

Euler's method has a local truncation error (error made in a single step) proportional to $(\Delta t)^2$, and a global error (accumulated over many steps) that is proportional to (Δt). The RK4 method has local error on the order of $(\Delta t)^5$ and total accumulated global error on the order of $(\Delta t)^4$. This means that, for example in a single step if we were to take a step with half the time step instead, i.e $\Delta t \rightarrow \Delta t/2$ the error in the Euler method would reduce by a factor of 4, but for the RK4 step it would reduce by a factor of 32. So, the RK4 algorithm offers much smaller error as the time steps are reduced.

Matlab code to solve the 1D heat equation using a manually-written 4th order Runge-Kutta algorithm is given below. The right hand side of the equation and the second derivative (calculated spectrally) are placed inside separate functions at the end of the script. Since the RHS of the 1D heat equation does not depend explicitly on time, it is not necessary to pass the variable t into the function `f()`. The main loop calculates the x4 RK4 terms and pieces them together using equations 128 and 132.

```
%%% Set up discrete time grid %%%
dt = 0.01;      % Time step
T = 40;         % Duration of total time window
t = 0:dt:T;     % Define time vector
Nt = length(t); % Number of time iterations

%%% Set up discrete space grid %%%
Nx = 200;       % Number of x grid points
dx = 0.5;       % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);
if Nx_IS_EVEN; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x  = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

%%% Set up discrete angular spatial frequency grid %%%
Fsx = 1/dx;
dfx = Fsx/Nx;
if Nx_IS_EVEN; fx = -(Fsx/2):dfx:(Fsx/2-dfx);
else;          fx = -(Fsx/2-dfx/2):dfx:(Fsx/2-dfx/2);
```

```
end
kx = 2*pi*fx;       % k vector in x
kx = ifftshift(kx); % Shift frequencies instead of psi for speed

alpha = 1;  % Diffusion coefficient

% Preallocate matrix for speed
psi = zeros([Nt Nx]);

% Define initial condition
psi0 = zeros([1,Nx]);
inds = x<=10 & x>=-10;
psi0( inds ) = 1;     % Set some values initially to 1
psi(1,:) = psi0;      % Put the first psi(x,t=0) into the matrix

% Iterate the equation over steps of dt
for k = 1:(Nt-1)

    psi_k = psi(k,:); % Retrieve the previous iteration

    % Implement 4th order Runge-Kutta (calculate x4 terms)
    kappa1 = f(psi_k,              kx,alpha);
    kappa2 = f(psi_k + kappa1*dt/2,kx,alpha);
    kappa3 = f(psi_k + kappa2*dt/2,kx,alpha);
    kappa4 = f(psi_k + kappa3*dt,  kx,alpha);

    % Piece back together the Runge-Kutta terms
    psi(k+1,:) = psi_k + (dt/6)*( kappa1 + 2*kappa2 + 2*kappa3 + kappa4 );

end

psi = real(psi);

%%% Plotting %%%
figure;subplot(1,2,1)
waterfall(x, t(1:100:end), psi(1:100:end,:));

% Function representing RHS of 1D Heat Equation
function dpsi_dt = f(psi,kx,alpha)

    d2psi_dx2 = secondDerivative(kx, psi); % Calculate second derivative
    dpsi_dt = alpha*d2psi_dx2; % Construct RHS of 1D heat equation

end

% Function for calculating d2psi_dx2 using spectral method
function d2psi_dx2 = secondDerivative(kx, psi)

    psi_hatx = fft( ifftshift(psi) );  % Move into k-space
    FTx = -(kx).^2.*psi_hatx;          % Calculate transform of derivative
    d2psi_dx2 = fftshift( ifft(FTx) ); % Move back to real space

end
```

4.2.3 Using ODE45 to Solve the PDE

Matlab has several built-in ODE solvers, the most popular of which is `ode45()`. This function implements a Runge-Kutta method, and includes a variable time step, in contrast to the standard Euler method which has a fixed time step in its most basic implementation. It is designed to solve systems

$$\frac{\mathrm{d}\boldsymbol{\psi}(t)}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{\psi}), \tag{133}$$

where the bold font denotes a vector, and so this is a system of multiple ordinary differential equations. The vector of initial conditions is defined as $\boldsymbol{\psi}(t_0) = \boldsymbol{\psi}_0$.

The following Matlab code uses `ode45` to solve the 1D heat equation. The right hand side (RHS) of the differential equation is built up inside a separate function, which is then passed to `ode45` in order to be iterated. Much of the code is the same as before, except that the main forward Euler time stepping loop has been replaced by a call to the Matlab solver:

```
%%% Set up discrete time grid %%%
dt = 0.01;      % Time step
T = 40;         % Duration of total time window
t = 0:dt:T-dt;  % Define time vector
Nt = length(t); % Number of time iterations

%%% Set up discrete space grid %%%
Nx = 200;       % Number of x grid points
dx = 0.5;       % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);
if Nx_IS_EVEN; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x  = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

%%% Set up discrete angular spatial frequency grid %%%
Fsx = 1/dx;
dfx = Fsx/Nx;
if Nx_IS_EVEN; fx = -(Fsx/2):dfx:(Fsx/2-dfx);
else;          fx = -(Fsx/2-dfx/2):dfx:(Fsx/2-dfx/2);
end
kx = 2*pi*fx;       % k vector in x
kx = ifftshift(kx); % Shift frequencies instead of psi for speed

alpha = 1;  % Diffusion coefficient

% Define initial condition
psi0 = zeros([1,Nx]);
inds = x<=10 & x>=-10;
psi0( inds ) = 1;     % Set some values initially to 1

% Iterate over chosen times
[t, psi] = ode45( @(t,psi) RHS_1DHeatEqn(t,psi,kx,alpha), t, psi0);

psi = real(psi);
```

```
35
36  %%% Plotting %%%
37  figure;subplot(1,2,1)
38  waterfall(x, t(1:100:end), psi(1:100:end,:));
39
40  function dpsi_dt = RHS_1DHeatEqn(t,psi,kx,alpha)
41
42      % Calculate d2psi_dx2
43      psi_hatx = fft( ifftshift(psi) );
44      FTx = -(kx').^2.*psi_hatx;
45      d2psi_dx2 = fftshift( ifft(FTx) );
46
47      % Construct RHS of 1D Heat Equation
48      dpsi_dt = alpha*d2psi_dx2;
49
50  end
```

4.2.4 Solving the PDE Entirely in Fourier Space

In the previous section, although the second derivative was calculated using a Fourier transform, the equation was still solved in the spatial domain (i.e we always converted right back into real space after the Fourier transform, so that we iterated the function $\psi(t,x)$). This is fine, but it requires Fourier transforming back and forth at *every* time step. Even though the transform is implemented with the extremely efficient FFT algorithm, it still induces some computational overhead.

In fact, in the case of the 1D heat equation, we can actually solve the entire equation in the Fourier domain, which is possible because the equation is linear. By Fourier transforming *the equation itself* and using Eq. 68 we obtain

$$\frac{\partial \psi(x,t)}{\partial t} = \alpha \frac{\partial^2 \psi(x,t)}{\partial x^2} \tag{134}$$

$$\xrightarrow{\mathcal{F}_x} \quad \mathcal{F}_x\left\{\frac{\partial \psi(x,t)}{\partial t}\right\} = \mathcal{F}_x\left\{\alpha \frac{\partial^2 \psi(x,t)}{\partial x^2}\right\} \tag{135}$$

$$\implies \frac{\partial \hat{\psi}(k_x,t)}{\partial t} = -\alpha k_x^2\, \hat{\psi}(k_x,t). \tag{136}$$

In the last step the order commutes, and so I have swapped the order of integration. For every individual value of k_x, this equation is just a function of time, and we obtain a *family* of ODEs - one for every k_x - in the angular spatial frequency domain. The system can be solved by first transforming the initial condition into Fourier space $\psi(x, t=0) \rightarrow \hat{\psi}(k_x, t=0)$. Then we can iterate in Fourier space, and finally transform the solution back to real space at the end. The benefit of this is that the Fourier transform need only be performed once at the beginning and once at the end, and *not* every time step as in the previous example. The following Matlab code solves the 1D heat equation entirely in Fourier space, converting back at the end:

```
%%% Set up discrete time grid %%%
dt = 0.01;      % Time step
T = 40;         % Duration of total time window
t = 0:dt:T-dt;  % Define time vector
Nt = length(t); % Number of time iterations

%%% Set up discrete space grid %%%
Nx = 201;       % Number of x grid points
dx = 0.5;       % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);
if Nx_IS_EVEN; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x  = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

%%% Set up discrete angular spatial frequency grid %%%
Fsx = 1/dx;
dfx = Fsx/Nx;
if Nx_IS_EVEN; fx = -(Fsx/2):dfx:(Fsx/2-dfx);
else;          fx = -(Fsx/2-dfx/2):dfx:(Fsx/2-dfx/2);
end
kx = 2*pi*fx;       % k vector in x
kx = ifftshift(kx); % Shift frequencies instead of psi for speed

alpha = 1;  % Diffusion coefficient

% Define initial condition
psi0 = zeros([1,Nx]);
inds = x<=10 & x>=-10;            % Define an initial top hat distribution
psi0( inds ) = 1;                 % Set central values initially to 1
psihat0 = fft( ifftshift(psi0) ); % Convert initial condition to Fourier space

% Iterate over chosen times in Fourier space
[t, psihat] = ode45( @(t,psihat) RHS_1DHeatEqn_Fourier(t,psihat,kx,alpha), t, psihat0);

% Transform psihat matrix back to real space domain, leaving time dimension intact
psi = fftshift( ifft( psihat,Nx,2 ) ,2 );

psi = real(psi);

%%% Plotting %%%
figure;subplot(1,2,1)
waterfall(x, t(1:100:end), psi(1:100:end,:));

function dpsihat_dt = RHS_1DHeatEqn_Fourier(t,psihat,kx,alpha)

    % Construct RHS of 1D Heat Equation in Fourier Space
    dpsihat_dt = -alpha*(kx').^2.*psihat;

end
```

4.2.5 Analytic Solution of the 1D Heat Equation

Returning to Eq. 136, for each k_x we have an ODE in which k_x can be treated as a constant

$$\frac{\partial \hat{\psi}(k_x, t)}{\partial t} = -\alpha k_x^2 \, \hat{\psi}(t). \tag{137}$$

This differential equation is

- *ordinary*, because it contains only derivatives with respect to one variable t, and $\hat{\psi}(t)$ only depends on t (for a particular fixed k_x).
- *first order*, because there are no higher order derivative terms of the form $d^2\hat{\psi}/dt^2$ or $d^3\hat{\psi}/dt^3$, for example.
- *linear*, because there are no terms of the form $\hat{\psi}^2$, $\hat{\psi}^3$, etc, or $(d\hat{\psi}/dt)^2$, etc.
- *homogeneous*, because there is no t dependence.

Since this equation is separable, we can solve it in the usual simple way

$$\int \frac{1}{\hat{\psi}(k_x, t)} \, \mathrm{d}\hat{\psi} = \int -\alpha k_x^2 \, \mathrm{d}t + C \tag{138}$$

$$\implies \ln|\hat{\psi}(k_x, t)| = -\alpha k_x^2 t + C \tag{139}$$

$$\implies \hat{\psi}(k_x, t) = e^{-\alpha k_x^2 t + C} \tag{140}$$

$$= A \, e^{-\alpha k_x^2 t} \tag{141}$$

$$= \hat{\psi}(k_x, t = 0) \times e^{-\alpha k_x^2 t}, \tag{142}$$

where I have used the fact that we can know the Fourier transform of the initial condition at $t = 0$. So, even though the solution is in Fourier space, the PDE has nevertheless been solved. This was much easier than trying to analytically solve the initial PDE in real space. Ofcourse, I want to know the solution in real space though, so all that remains is to inverse transform the solution

$$\psi(x, t) = \mathcal{F}_x^{-1}\Big\{\hat{\psi}(k_x, t)\Big\} \tag{143}$$

$$= \mathcal{F}_x^{-1}\Big\{\hat{\psi}(k_x, t = 0) \times e^{-\alpha k_x^2 t}\Big\}. \tag{144}$$

Now we can make use of the *convolution theorem*, which states that

$$\mathcal{F}^{-1}\Big\{\hat{f} \cdot \hat{g}\Big\} = f * g, \tag{145}$$

where the $*$ symbol indicates a convolution operation and the $\cdot$ indicates multiplication, to write

$$\psi(x, t) = \psi(x, t = 0) * \mathcal{F}_x^{-1}\Big\{e^{-\alpha k_x^2 t}\Big\}. \tag{146}$$

Now, the time dependent factor is simply a Gaussian function, and we know that the inverse Fourier transform will be another Gaussian given by Eq. 25

$$\mathcal{F}_x\left\{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right)\right\} = \exp\left(-\frac{k_x^2\sigma^2}{2}\right), \tag{147}$$

$$\implies \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right) = \mathcal{F}_x^{-1}\left\{\exp\left(-\frac{k_x^2\sigma^2}{2}\right)\right\} \tag{148}$$

$$= \mathcal{F}_x^{-1}\left\{e^{-\alpha k_x^2 t}\right\}, \tag{149}$$

with $\sigma = \sqrt{2\alpha t}$. Substituting into Eq. 146, this becomes

$$\boxed{\psi(x,t) = \psi(x,t=0) * \frac{1}{\sqrt{4\pi\alpha t}}\exp\left(-\frac{x^2}{4\alpha t}\right).} \tag{150}$$

This very nice result shows that the distribution $\psi(x,t)$ in the heat equation (which usually describes a temperature distribution) as a function of time is actually given by a convolution between the initial distribution $\psi(x,t=0)$ and a Gaussian function. The width of the convolving Gaussian is given by $\sqrt{2\alpha t}$, and therefore increases with time. This behaviour is captured at the various times in Fig. 12, where we can see that the heat spreads quickly at the sharp edges early times (where the second derivative with respect to space is large). However, for late times the exact shape of the initial distribution becomes unrecognisable, and the function is dominated by the Gaussian in the convolution.

4.2.6 Summary

I have looked at numerically solving partial differential equations through the specific example of the simple 1D heat equation. Several approaches have been explored

- Method 1 - Using a simple forward time centered space (FTCS) scheme, in which the time step was performed with a forward Euler approximation, and the second derivative was approximated with a central finite difference.
- Method 2 - Again using forward Euler for the time step, but this time implementing a spectral method with FFT for calculating the second derivative with respect to space at each time step.
- Method 3 - Using a 4th order Runge-Kutta scheme for the time part, as a way of improving on the simple Euler method. This has a more favourable scaling for the error accumulated from each time step.
- Method 4 - Making use of Matlab's built-in ODE solver `ode45` instead of a hand-coded Runge-Kutta. In this case, I also made use of the spectral method for calculating the second spatial derivative.
- Method 5 - Since the heat equation is linear, it is possible to represent the equation itself in the Fourier domain, and solve the entire equation in Fourier space using ODE45. The solution then

simply needs to be converted back to real space at the end. This method alleviates any need to calculate second derivatives in space.

All of these methods are able to reproduce a good result under certain conditions. The analytic solution was shown to be a convolution of the initial condition with a Gaussian kernel, and can be used to check the results. Fig. 13 shows that all the Matlab scripts presented are able to provide reasonable solutions.

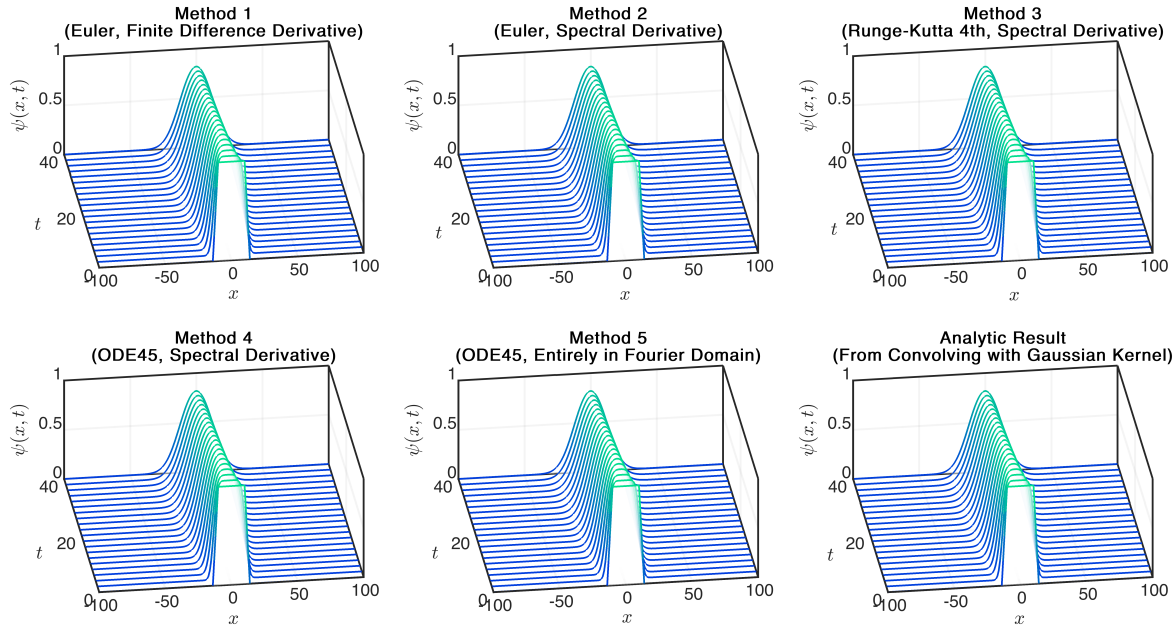

FIGURE 13: Summary of the various methods for solving the 1D heat equation covered in this section. The parameters used were $\alpha = 1$, $\Delta x = 0.5$, $\Delta t = 0.01$, $N_x = 200$, $t \in [0, 40]$. All give similar results.

At this point it would be good to analyse the accuracy of each of these methods, but I am not going to look deeply into that. The error depends largely on the time step Δt and the grid spacing Δx. Additionally, some of these will exhibit instability depending on the conditions, and the stability of differential equations is a whole other subject in itself. Just as an example, in Fig. 14 I have plotted the absolute value of the difference between the analytic solution and the various numerical solutions for the final time $t = 40$. It can be seen that the worst performing is the simple Euler method with a finite difference approximation to the second derivative, as may be expected. The best seems to be the hand-coded Runge-Kutta method with a spectral derivative. I must again stress that this will depend significantly on the chosen parameters, and is only an illustration for the case of these particular numbers. In addition, there will also be considerations in the time it takes the code to execute. The Runge-Kutta method will clearly take longer to run, because of the additional lines of computation. The best method to use will depend on the specific situation and problem being investigated.

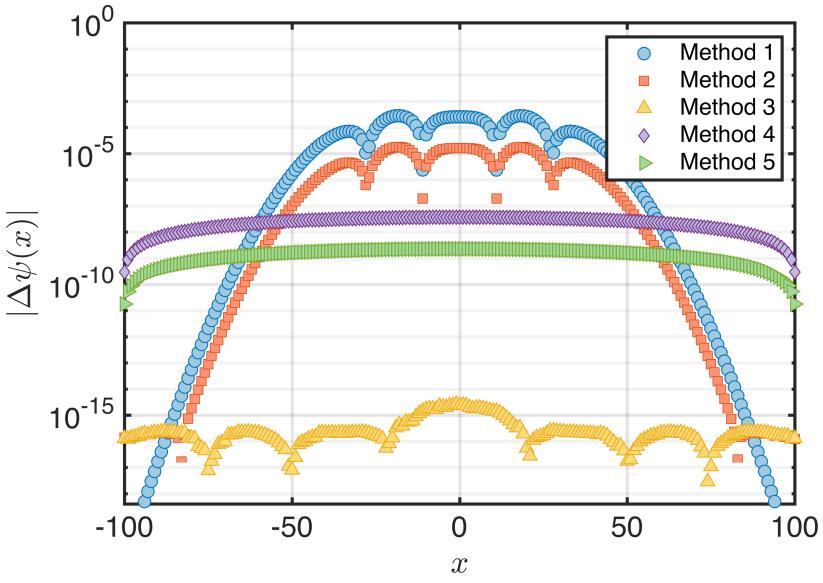

FIGURE 14: Absolute value of the difference between each numerical result and the analytic result for the final time $t = 40$, for the solutions presented in Fig. 13.

4.3 A 3D PDE Example - The 3D Heat Equation

Now I will look at trying to solve a PDE in higher dimensions. One of the simplest ways to learn about this additional level of complexity is probably to consider the heat equation as before but in three dimensions

$$\frac{\partial\psi(x,y,z,t)}{\partial t} = \alpha\nabla^2\psi(x,y,z,t) \tag{151}$$

$$= \alpha\left(\frac{\partial^2\psi(x,y,z,t)}{\partial x^2} + \frac{\partial^2\psi(x,y,z,t)}{\partial y^2} + \frac{\partial^2\psi(x,y,z,t)}{\partial z^2}\right). \tag{152}$$

where ∇^2 is the 3D Laplacian in cartesian coordinates. As before, we can write the second derivatives in terms of Fourier tranforms with repsect to each of the individual coordinates

$$\begin{aligned}\frac{\partial\psi(x,y,z,t)}{\partial t} &= \alpha\Bigg(\mathcal{F}_x^{-1}\Big\{-k_x^2\,\mathcal{F}_x\{\psi(x,y,z,t)\}\Big\} + \mathcal{F}_y^{-1}\Big\{-k_y^2\,\mathcal{F}_y\{\psi(x,y,z,t)\}\Big\} + \ldots \\ &\qquad \ldots\,\mathcal{F}_z^{-1}\Big\{-k_z^2\,\mathcal{F}_z\{\psi(x,y,z,t)\}\Big\}\Bigg) \\ &= -\alpha\Bigg(\mathcal{F}_x^{-1}\Big\{k_x^2\,\hat{\psi}(k_x,y,z,t)\Big\} + \mathcal{F}_y^{-1}\Big\{k_y^2\,\hat{\psi}(x,k_y,z,t)\Big\} + \ldots \\ &\qquad \ldots\,\mathcal{F}_z^{-1}\Big\{k_z^2\,\hat{\psi}(x,y,k_z,t)\Big\}\Bigg).\end{aligned} \tag{153}$$

This equation shows that the partial derivatives can be calculated by performing three *individual* Fourier transforms, but with each one only transforming along one of the spatial dimensions. So for

example, to calculate the second derivative with respect to x, we need to Fourier transform only along x, which converts $\psi(x, y, z)$ into $\hat{\psi}(k_x, y, z)$, and similarly for y and z.

The following Matlab code solves the 3D heat equation by calculating the second derivatives using FFT, with the result for four selected times shown in Fig. 15.

```
%%% ========= Set up discrete time grid ========= %%%
dt = 0.01;       % Time step
T = 5;           % Duration of total time window
t = 0:dt:T;      % Define time vector
Nt = length(t); % Number of time iterations
%%% ============================================= %%%

%%% ======= Set up discrete X space grid ======== %%%
Nx = 64;         % Number of x grid points
dx = 0.5;        % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);
if Nx_IS_EVEN; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x  = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

%%% ======= Set up discrete Y space grid ======== %%%
Ny = 64;          % Number of y grid points
dy = 0.5;         % Spacing of y grid
Ny_IS_EVEN =  ~rem(Ny,2);
if Ny_IS_EVEN; y  = -(Ny*dy/2):dy:(Ny*dy/2-dy);
else;          y  = -((Ny-1)*dy/2):dy:((Ny-1)*dy/2);
end

%%% ======= Set up discrete Z space grid ======== %%%
Nz = 64;          % Number of y grid points
dz = 0.5;         % Spacing of y grid
Nz_IS_EVEN =  ~rem(Nz,2);
if Nz_IS_EVEN; z  = -(Nz*dz/2):dz:(Nz*dz/2-dz);
else;          z  = -((Nz-1)*dz/2):dz:((Nz-1)*dz/2);
end
%%% ============================================= %%%

[X,Y,Z] = meshgrid(x,y,z); % Generate 3D meshgrid from x,y,z

%%% == Set up angular spatial frequency X grid == %%%
Fsx = 1/dx;          % Sampling frequency in x
dfx = Fsx/Nx;        % Frequency resolution in x
if Nx_IS_EVEN; fx = -(Fsx/2):dfx:(Fsx/2-dfx);
else;          fx = -(Fsx/2-dfx/2):dfx:(Fsx/2-dfx/2);
end
kx = 2*pi*fx;        % Angular frequency k vector in x

%%% == Set up angular spatial frequency Y grid == %%%
Fsy = 1/dy;          % Sampling frequency in y
dfy = Fsy/Ny;        % Frequency resolution in y
if Ny_IS_EVEN; fy = -(Fsy/2):dfy:(Fsy/2-dfy);
else;          fy = -(Fsy/2-dfy/2):dfy:(Fsy/2-dfy/2);
```

```
end
ky = 2*pi*fy;       % Angular frequency k vector in y

%%% == Set up angular spatial frequency Z grid == %%%
Fsz = 1/dz;         % Sampling frequency in z
dfz = Fsz/Nz;       % Frequency resolution in z
if Nz_IS_EVEN; fz = -(Fsz/2):dfz:(Fsz/2-dfz);
else;          fz = -(Fsz/2-dfz/2):dfz:(Fsz/2-dfz/2);
end
kz = 2*pi*fz;       % Angular frequency k vector in z
%%% ============================================== %%%

% ifftshift() k vectors instead of psi later, for speed.
% Also permute k vectors to be compatible with dot
% multiplication with 3D psi matrix later.
kx = permute(ifftshift(kx), [1 2 3]);
ky = permute(ifftshift(ky), [2 1 3]);
kz = permute(ifftshift(kz), [3 1 2]);

alpha = 1;  % Diffusion coefficient

psi = zeros(Ny,Nx,Nz,Nt); % Preallocate for speed

%%% ========== Define initial condition ========= %%%
x_box_size = 1;  % Initial condition is a cuboid
y_box_size = 10;
z_box_size = 5;
psi0 = zeros(Ny,Nx,Nz);
inds = X<=(x_box_size/2) & X>=-(x_box_size/2) & ...
       Y<=(y_box_size/2) & Y>=-(y_box_size/2) & ...
       Z<=(z_box_size/2) & Z>=-(z_box_size/2);
psi0( inds ) = 1;     % Set values in cuboid to 1
psi(:,:,:,1) = psi0;  % Put the first psi(x,t=0) into the matrix
%%% ============================================== %%%

%%% === Iterate the equation over steps of dt === %%%
for k = 1:(Nt-1)

    psi_k = psi(:,:,:,k); % Retrieve the previous iteration

    % Calculate d2psi_dx2
    psi_hatx = fft( ifftshift(psi_k,2) ,Nx,2 )*dx;
    FTx = -kx.^2.*psi_hatx;
    d2psi_dx2 = fftshift( ifft(FTx,Nx,2) ,2 )*Nx*dfx;

    % Calculate d2psi_dy2
    psi_haty = fft( ifftshift(psi_k,1) ,Ny,1 )*dy;
    FTy = -ky.^2.*psi_haty;
    d2psi_dy2 = fftshift( ifft(FTy,Ny,1) ,1 )*Nx*dfx;

    % Calculate d2psi_dz2
    psi_hatz = fft( ifftshift(psi_k,3) ,Nz,3 )*dz;
    FTz = -kz.^2.*psi_hatz;
```

```
    d2psi_dz2 = fftshift( ifft(FTz,Nz,3) ,3 )*Nz*dfz;

    % Forward Euler method in time
    psi(:,:,:,k+1) = psi_k + dt*alpha*(d2psi_dx2 + d2psi_dy2 + d2psi_dz2);

end
%%% ============================================== %%%

%%% ================== Plotting ================== %%%
k_plot = 20; % Time index to plot

figure; % Plot 3D isosurface
isosurface(X,Y,Z,psi(:,:,:,k_plot),0.01);

figure; % Plot x,y,z line cuts through origin
subplot(1,3,1);plot(x, squeeze(psi(Ny/2,:,Nz/2,k_plot)))
subplot(1,3,2);plot(y, squeeze(psi(:,Nx/2,Nz/2,k_plot)))
subplot(1,3,3);plot(z, squeeze(psi(Ny/2,Nx/2,:,k_plot)))
```

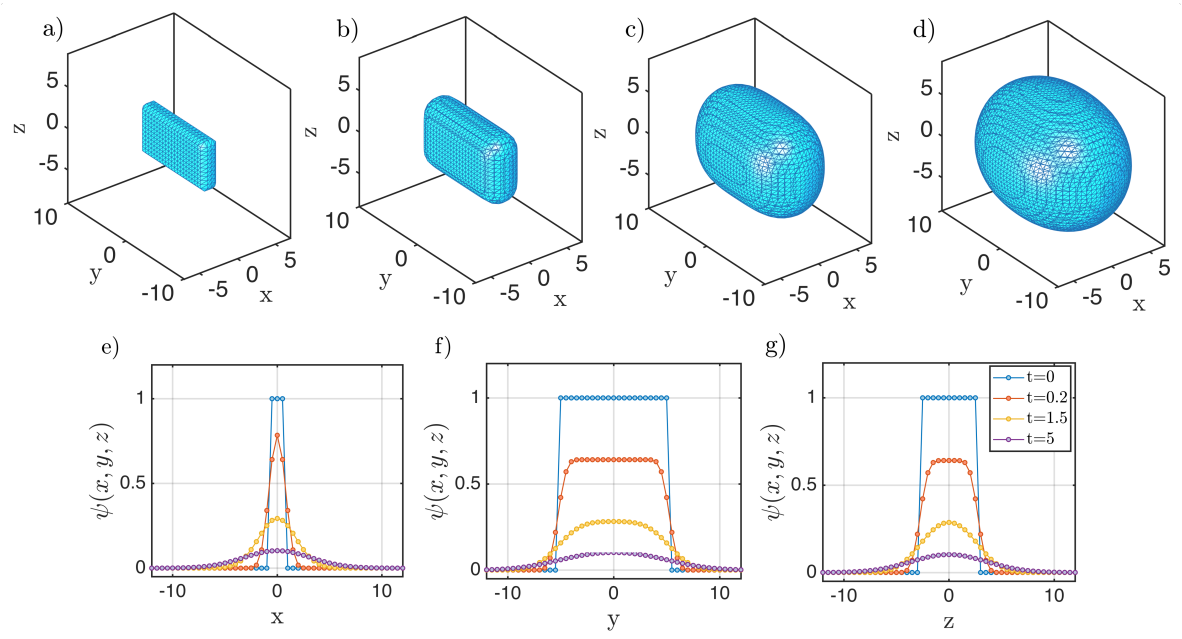

FIGURE 15: Numerical solution of the 3D heat equation for spatial grids of $\Delta x = \Delta y = \Delta z = 0.5$ and $N_x = N_y = N_z = 64$, over the time interval $t \in [0, 5]$ with $\Delta t = 0.01$. Top row: Isosurfaces of constant $\psi(x, y, z) = 0.01$ at times $t = [0, 0.2, 1.5, 5]$ (a-d respectively). Bottom row: x, y, and z 1D line cuts of the solution through the origin for the same times. The initial condition was defined to be $\psi = 1$ for $|x| \leq 0.5$, $|y| \leq 5$, $|z| \leq 2.5$, and $\psi = 0$ otherwise, i.e initially a cuboid distribution. The distribution spreads quickly in all dimensions, eventually tending to a Gaussian, as expected from the heat equation.

In the main loop of the code when calculating the Fourier transforms, I have had to use the additional arguments in the Matlab FFT() functions. These specify which dimension to take the Fourier transform in for a multidimensional array (in this case 3D). So the line

```
psihatx = fft( ifftshift(psik,2) ,Nx,2 )*dx
```

first performs an `ifftshift()` along dimension 2, which corresponds to column-wise and therefore the x-direction. Then an `fft()` is performed again along dimension 2 (column-wise) with `Nx` points. If the argument passed to FFT is greater than `Nx` then zero-padding will be applied, but this is not done here. This line is calculating $\hat{\psi}(k_x, y, z)$ from $\psi(x, y, z)$.

Note that we can also make some speed improvement to the code by performing a full 3-dimensional FFT, instead of x3 individual ones along each dimension. In terms of a 3D Fourier transform, the differential equation becomes

$$\frac{\partial \psi(x,y,z,t)}{\partial t} = -\alpha\, \mathcal{F}_{3\mathrm{D}}^{-1}\Big\{ \left(k_x^2 + k_y^2 + k_z^2\right)\ \hat{\psi}(k_x, k_y, k_z, t)\Big\} \tag{154}$$

where the 3D Fourier forward and inverse transforms are defined by

$$\mathcal{F}_{3\mathrm{D}}\Big\{\psi(x,y,z,t)\Big\} = \hat{\psi}(k_x,k_y,k_z,t) = \iiint_{-\infty}^{+\infty} \psi(x,y,z,t)\, \exp\Big(-i\left[k_x x + k_y y + k_z z\right]\Big)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z \tag{155}$$

$$\mathcal{F}_{3\mathrm{D}}^{-1}\Big\{\hat{\psi}(k_x,k_y,k_z,t)\Big\} = \frac{1}{(2\pi)^3} \iiint_{-\infty}^{+\infty} \hat{\psi}(k_x,k_y,k_z,t)\, \exp\Big(i\left[k_x x + k_y y + k_z z\right]\Big)\, \mathrm{d}k_x\, \mathrm{d}k_y\, \mathrm{d}k_z. \tag{156}$$

An n-dimensional Fourier transform in Matlab is performed by the built-in function `fftn()`, which is equivalent to doing a 1D Fourier transform along each dimension of the input array. The following modifications can be made to the code to solve the differential equation using Eq. 154:

```
%%% ====== Create meshgrid of k vectors ========= %%%
[KX,KY,KZ] = meshgrid(kx,ky,kz);
K = KX.^2 + KY.^2 + KZ.^2;
K = ifftshift(K);

%%% === Iterate the equation over steps of dt === %%%
for k = 1:(Nt-1)
    psi_k = psi(:,:,:,k); % Retrieve the previous iteration

    % Calculate 3D Laplacian
    psi_hat = fftn( ifftshift(psi_k) );    % Obtain psi in reciprocal space
    FT = -K.*psi_hat;                      % Apply spectral derivative operation
    laplacian_psi = fftshift( ifftn(FT) ); % Convert psihat back to real space

    % Forward Euler method in time
    psi(:,:,:,k+1) = psi_k + dt*alpha*laplacian_psi;
end
```

For the parameters selected here, the equation is now solved over the time range in around 6 seconds on a standard, compared to 10.5 seconds when doing x3 individual Fourier transforms. Ofcourse, this may scale differently when changing time/space grid steps, but is seems at least like `fftn()` offers an improvement in execution time.

5 Schrödinger Equation for Free Propagation

Let's say that we have a single particle in the ground state of a two-dimensional (2D) harmonic oscillator potential in the x and y directions described as the scalar function

$$V_{\text{ext}}(x,y) = \frac{1}{2}m\left(\omega_x^2 x^2 + \omega_y^2 y^2\right), \tag{157}$$

where ω_i are the two trap oscillation frequencies, characterising the curvature of the confining potential, and m is the mass of the particle. The particle has an associated wavefunction $\psi(x,y)$ which describes the probability amplitude to find a particle at a given position. This wavefunction evolves in time according to the 2D time-dependent Schrödinger equation (TDSE) in terms of the Hamiltonian operator

$$\begin{aligned}
i\hbar\frac{\partial\psi(x,y,t)}{\partial t} &= \hat{H}\ \psi(x,y,t) && (158)\\
&= \left[\hat{T}+\hat{V}\right]\psi(x,y,t) && (159)\\
&= \left[\frac{\hat{\mathbf{p}}\cdot\hat{\mathbf{p}}}{2m} + V_{\text{ext}}(x,y)\right]\psi(x,y,t) && (160)\\
&= \left[\frac{-\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(x,y)\right]\psi(x,y,t) && (161)\\
&= \left[\frac{-\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right) + V_{\text{ext}}(x,y)\right]\psi(x,y,t). && (162)
\end{aligned}$$

This equation is very similar to the diffusion equations looked at in previous sections, except with an additional factor of the imaginary constant i.

If we assume that at $t = 0$ the particle is in the ground state of the trap, we know that the ground state wavefunction will be a Gaussian distribution (the lowest eigen-energy value of the time-independent Schrödinger equation), given by

$$\psi(x,y,t=0) = \frac{1}{\sqrt{\pi a_x a_y}}\exp\left(-\frac{x^2}{2a_x^2}-\frac{y^2}{2a_y^2}\right), \tag{163}$$

where $a_i = \sqrt{\hbar/m\omega_i}$ are the harmonic oscillator lengths in each of the two directions, which are equivalent to what we would call the σ ($1/\sqrt{e}$ of the peak) of the Gaussian. The probability density to measure the particle at a given position is given by the square modulus of the wavefunction

$$n(x,y,t) = \psi^*(x,y,t)\ \psi(x,y,t) = |\psi(x,y,t)|^2 \tag{164}$$

$$= \frac{1}{\pi a_x a_y} \exp\left(-\frac{x^2}{a_x^2} - \frac{y^2}{a_y^2}\right) \tag{165}$$

Note that the square operation has removed the factors of 2 inside the exponential. The wavefunction is normalised such that the integral of the probability density is unity at every time

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |\psi(x,y,t)|^2 \,\mathrm{d}x\,\mathrm{d}y = 1. \tag{166}$$

Now imagine that the wavepacket was initially confined in a harmonic trap with the Gaussian distribution above, but then the trapping potential was immediately switched off at $t = 0$. The wavepacket will evolve according to the TDSE in Eq. 162 but with $V_{\text{ext}} = 0$

$$i\hbar\frac{\partial\psi(x,y,t)}{\partial t} = \frac{-\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\psi(x,y,t) \tag{167}$$

$$\implies \frac{\partial\psi(x,y,t)}{\partial t} = \frac{i\hbar}{2m}\left(\frac{\partial^2\psi(x,y,t)}{\partial x^2} + \frac{\partial^2\psi(x,y,t)}{\partial y^2}\right) \tag{168}$$

5.1 Analytic Solution

To solve this equation, we can take a 2D Fourier transform (i.e convert both of x and y into reciprocal coordinates k_x and k_y) of both sides to get

$$\mathcal{F}_{\text{2D}}\left\{\frac{\partial\psi(x,y,t)}{\partial t}\right\} = \mathcal{F}_{\text{2D}}\left\{\frac{i\hbar}{2m}\left(\frac{\partial^2\psi(x,y,t)}{\partial x^2} + \frac{\partial^2\psi(x,y,t)}{\partial y^2}\right)\right\} \tag{169}$$

$$\implies \frac{\partial\hat{\psi}(k_x,k_y,t)}{\partial t} = \frac{i\hbar}{2m}\left(\mathcal{F}_{\text{2D}}\left\{\frac{\partial^2\psi(x,y,t)}{\partial x^2}\right\} + \mathcal{F}_{\text{2D}}\left\{\frac{\partial^2\psi(x,y,t)}{\partial y^2}\right\}\right) \tag{170}$$

$$= \frac{i\hbar}{2m}\left(-k_x^2\,\mathcal{F}_{\text{2D}}\Big\{\psi(x,y,t)\Big\} - k_y^2\,\mathcal{F}_{\text{2D}}\Big\{\psi(x,y,t)\Big\}\right) \tag{171}$$

$$= -\frac{i\hbar}{2m}\left(k_x^2 + k_y^2\right)\hat{\psi}(k_x,k_y,t) \tag{172}$$

This is a family of ordinary differential equations (one for each k_x and k_y pair) in Fourier space, which has the standard solution

$$\hat{\psi}(k_x,k_y,t) = \exp\left(-\frac{i\hbar}{2m}\left[k_x^2 + k_y^2\right]t\right)\cdot\hat{\psi}(k_x,k_y,t=0). \tag{173}$$

We can calculate $\hat{\psi}(k_x,k_y,t=0)$ by taking the Fourier transform of the initial condition in Eq. 163

$$\hat{\psi}(k_x,k_y,t=0) = \mathcal{F}_{\text{2D}}\Big\{\psi(x,y,t=0)\Big\} \tag{174}$$

$$= \mathcal{F}_{\text{2D}}\left\{\frac{1}{\sqrt{\pi a_x a_y}}\exp\left(-\frac{x^2}{2a_x^2} - \frac{y^2}{2a_y^2}\right)\right\} \tag{175}$$

$$= 2\sqrt{\pi a_x a_y}\,\exp\left(-\frac{a_x^2 k_x^2}{2} - \frac{a_y^2 k_y^2}{2}\right), \tag{176}$$

where in the last step I have used Eq. 51. Inserting this into Eq. 173, we obtain

$$\hat{\psi}(k_x, k_y, t) = 2\sqrt{\pi a_x a_y}\,\exp\left(-\frac{i\hbar}{2m}\left[k_x^2 + k_y^2\right] t\right)\exp\left(-\frac{a_x^2 k_x^2}{2} - \frac{a_y^2 k_y^2}{2}\right) \tag{177}$$

$$= 2\sqrt{\pi a_x a_y}\,\exp\left(-\frac{k_x^2}{2}\left[a_x^2 + \frac{i\hbar}{m}t\right] - \frac{k_y^2}{2}\left[a_y^2 + \frac{i\hbar}{m}t\right]\right). \tag{178}$$

To find the final solution $\psi(x, y, t)$ we now need to take the inverse transform to get back into real space

$$\psi(x, y, t) = \mathcal{F}_{\text{2D}}^{-1}\left\{\hat{\psi}(k_x, k_y, t)\right\} \tag{179}$$

$$= \frac{2\sqrt{\pi a_x a_y}}{2\pi\sqrt{a_x^2 + \frac{i\hbar}{m}t}\sqrt{a_y^2 + \frac{i\hbar}{m}t}}\,\exp\left(-\frac{x^2}{2\left[a_x^2 + \frac{i\hbar}{m}t\right]} - \frac{y^2}{2\left[a_y^2 + \frac{i\hbar}{m}t\right]}\right) \tag{180}$$

$$= \frac{\sqrt{a_x a_y}}{\sqrt{\pi}\sqrt{a_x^2 + \frac{i\hbar}{m}t}\sqrt{a_y^2 + \frac{i\hbar}{m}t}}\,\exp\left(-\frac{x^2}{2\left[a_x^2 + \frac{i\hbar}{m}t\right]} - \frac{y^2}{2\left[a_y^2 + \frac{i\hbar}{m}t\right]}\right), \tag{181}$$

where I have again used Eq. 51. Note that in the case of a circularly symmetric potential, we have $a_x = a_y = a_\perp$, and the wavefunction simplifies to

$$\psi(x, y, t) = \frac{a_\perp}{\sqrt{\pi}\left(a_\perp^2 + \frac{i\hbar}{m}t\right)}\,\exp\left(-\frac{(x^2 + y^2)}{2\left[a_\perp^2 + \frac{i\hbar}{m}t\right]}\right). \tag{182}$$

The probability density at time t is then given by the modulus squared of Eq. 181 to be

$$\begin{aligned} n(x, y, t) &= |\psi(x, y, t)|^2 = \psi^*(x, y, t)\,\psi(x, y, t) \\ &= \frac{\sqrt{a_x a_y}}{\sqrt{\pi}\sqrt{a_x^2 - \frac{i\hbar}{m}t}\sqrt{a_y^2 - \frac{i\hbar}{m}t}}\,\exp\left(-\frac{x^2}{2\left[a_x^2 - \frac{i\hbar}{m}t\right]} - \frac{y^2}{2\left[a_y^2 - \frac{i\hbar}{m}t\right]}\right) \times \ldots \\ &\qquad \ldots \frac{\sqrt{a_x a_y}}{\sqrt{\pi}\sqrt{a_x^2 + \frac{i\hbar}{m}t}\sqrt{a_y^2 + \frac{i\hbar}{m}t}}\,\exp\left(-\frac{x^2}{2\left[a_x^2 + \frac{i\hbar}{m}t\right]} - \frac{y^2}{2\left[a_y^2 + \frac{i\hbar}{m}t\right]}\right) \\ &= \frac{a_x a_y}{\pi\sqrt{\left(a_x^4 + \hbar^2 t^2/m^2\right)\left(a_y^4 + \hbar^2 t^2/m^2\right)}}\,\exp\left(-\frac{a_x^2 x^2}{a_x^4 + \hbar^2 t^2/m^2} - \frac{a_y^2 y^2}{a_y^4 + \hbar^2 t^2/m^2}\right). \end{aligned} \tag{183}$$

Therefore, this equation shows that a Gaussian (ground state) wavepacket undergoing free expansion (i.e after switching off the external potential, and in the absence of any interactions) will *remain*

Gaussian with an RMS width ($1/\sqrt{e}$ of the peak) of

$$\sigma_i(t) = \frac{\sqrt{a_i^4 + \hbar^2 t^2/m^2}}{\sqrt{2}a_i}, \tag{184}$$

where $i = x, y$ are the two directions. Since $a_i = \sqrt{\hbar/m\omega_i}$, this can also be expressed in terms of the initial trap frequency as

$$\sigma_i(t) = \frac{a_i}{\sqrt{2}}\sqrt{1 + (\omega_i t)^2}. \tag{185}$$

Remember that here the factor of $\sqrt{2}$ appears because a is the RMS value of the wavefunction $\psi(t = 0)$, but σ_i is the RMS value of the density, $n = |\psi|^2$, which introduces a factor of 2 in the exponential of the Gaussian function. The in-trap RMS value is then $\sigma_i(t = 0) = a_i/\sqrt{2}$, so

$$\sigma_i(t) = \sigma_i(t = 0)\sqrt{1 + (\omega_i t)^2}. \tag{186}$$

Additionally, the three-dimensional version of the time-dependent density would be

$$\begin{aligned} n(x, y, z, t) = |\psi(x, y, z, t)|^2 = & \frac{a_x a_y a_z}{\pi^{3/2}\sqrt{\left(a_x^4 + \hbar^2 t^2/m^2\right)\left(a_y^4 + \hbar^2 t^2/m^2\right)\left(a_z^4 + \hbar^2 t^2/m^2\right)}} \times \ldots \\ & \exp\left(-\frac{a_x^2 x^2}{a_x^4 + \hbar^2 t^2/m^2} - \frac{a_y^2 y^2}{a_y^4 + \hbar^2 t^2/m^2} - \frac{a_z^2 z^2}{a_z^4 + \hbar^2 t^2/m^2}\right). \end{aligned} \tag{187}$$

5.2 Numerical Solution

The Schrödinger equation can be written as

$$\frac{\partial\psi(x, y, t)}{\partial t} = -\frac{i}{\hbar}\hat{H}\psi(x, y, t), \tag{188}$$

and with the external potential switched off at $t = 0$ this becomes for $t \geq 0$

$$\frac{\partial\psi(x, y, t)}{\partial t} = \frac{i\hbar}{2m}\left(\frac{\partial^2\psi(x, y, t)}{\partial x^2} + \frac{\partial^2\psi(x, y, t)}{\partial y^2}\right). \tag{189}$$

The second derivatives in the RHS can be calculated spectrally using a 2D Fourier transform as shown in Sec. 3.3

$$\frac{\partial\psi(x, y, t)}{\partial t} = \frac{i\hbar}{2m}\mathcal{F}_{2\mathrm{D}}^{-1}\left\{-\left(k_x^2 + k_y^2\right)\mathcal{F}_{2\mathrm{D}}\left\{\psi(x, y, t)\right\}\right\} = f(\psi(x, y, t)). \tag{190}$$

The right hand side of this equation $f(\psi)$ can be stepped forward in time using either forward Euler scheme or 4th order Runge-Kutta, for example.

The following Matlab code solves the Schrödinger equation without a potential term, in order to observe the free expansion of a Gaussian wavepacket. The expansion can be seen in Fig. 16, and x line cuts are compared to the analytic results of Eq. 181 in Fig. 17.

```
hbar = 1;
mass = 1;

%%% ========= Set up discrete time grid ========= %%%
dt = 0.002;     % Time step
T = 3;          % Duration of total time window
t = 0:dt:T;     % Define time vector
Nt = length(t); % Number of time iterations
%%% ============================================= %%%

%%% ======= Set up discrete X space grid ======== %%%
Nx = 150;        % Number of x grid points
dx = 0.15;       % Spacing of x grid
Nx_IS_EVEN =  ~rem(Nx,2);
if Nx_IS_EVEN; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx);
else;          x  = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);
end

%%% ======= Set up discrete Y space grid ======== %%%
Ny = 150;         % Number of y grid points
dy = 0.15;        % Spacing of y grid
Ny_IS_EVEN =  ~rem(Ny,2);
if Ny_IS_EVEN; y  = -(Ny*dy/2):dy:(Ny*dy/2-dy);
else;          y  = -((Ny-1)*dy/2):dy:((Ny-1)*dy/2);
end

[X,Y] = meshgrid(x,y); % Generate 2D meshgrid from x,y

%%% == Set up angular spatial frequency X grid == %%%
Fsx = 1/dx;         % Sampling frequency in x
dfx = Fsx/Nx;       % Frequency resolution in x
if Nx_IS_EVEN; fx = -(Fsx/2):dfx:(Fsx/2-dfx);
else;          fx = -(Fsx/2-dfx/2):dfx:(Fsx/2-dfx/2);
end
kx = 2*pi*fx;       % Angular frequency k vector in x

%%% == Set up angular spatial frequency Y grid == %%%
Fsy = 1/dy;         % Sampling frequency in y
dfy = Fsy/Ny;       % Frequency resolution in y
if Ny_IS_EVEN; fy = -(Fsy/2):dfy:(Fsy/2-dfy);
else;          fy = -(Fsy/2-dfy/2):dfy:(Fsy/2-dfy/2);
end
ky = 2*pi*fy;       % Angular frequency k vector in y

%%% ====== Create meshgrid of k vectors ========= %%%
[KX,KY] = meshgrid(kx,ky);
K = KX.^2 + KY.^2;
K = ifftshift(K);

psi = zeros(Ny,Nx,Nt); % Preallocate for speed

% Define initial condition to be a Gaussian wavepacket
ax = 1;
```

```
ay = 2;
psi0 = 1/sqrt(pi*ax*ay)*exp(-X.^2/(2*ax^2)-Y.^2/(2*ay^2));
psi(:,:,1) = psi0;    % Put the first psi(x,t=0) into the matrix

% Iterate the equation over steps of dt
for k = 1:(Nt-1)

    psi_k = psi(:,:,k); % Retrieve the previous iteration

    % Implement 4th order Runge-Kutta (calculate x4 terms)
    kappa1 = f(psi_k,              K,hbar,mass);
    kappa2 = f(psi_k + kappa1*dt/2,K,hbar,mass);
    kappa3 = f(psi_k + kappa2*dt/2,K,hbar,mass);
    kappa4 = f(psi_k + kappa3*dt,  K,hbar,mass);

    % Piece back together the Runge-Kutta terms
    psi(:,:,k+1) = psi_k + (dt/6)*( kappa1 + 2*kappa2 + 2*kappa3 + kappa4 );

end

n = conj(psi).*psi;

ind_plot = 100; % Choose the time index for plotting
figure;
surf(X,Y,psi(:,:,ind_plot))

% Function representing RHS of 1D Heat Equation
function dpsi_dt = f(psi,K,hbar,mass)

    laplacian_2D_psi = laplacian_2D(K, psi);     % Calculate second derivative
    dpsi_dt = 1i*hbar/(2*mass)*laplacian_2D_psi; % Construct RHS of 2D Schrodinger Eqn.

end

% Function for calculating Laplacian in 2D using spectral method
function laplacian_2D_psi = laplacian_2D(K, psi)

    % Calculate 2D Laplacian
    psi_hat = fftn( ifftshift(psi) );         % Obtain psi in reciprocal space
    FT = -K.*psi_hat;                         % Apply spectral derivative operation
    laplacian_2D_psi = fftshift( ifftn(FT) ); % Convert psihat back to real space

end
```

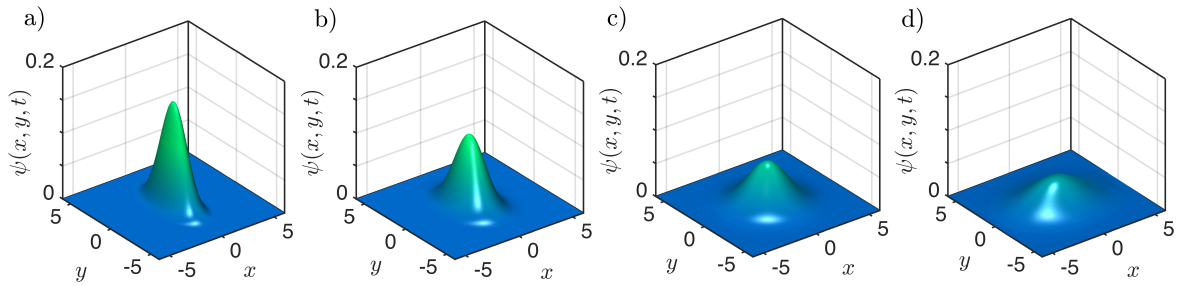

FIGURE 16: Numerical solution for free expansion of a Gaussian wavepacket after removing the external potential according to the Schrodinger equation. a) to d) are surface plots of $\psi(x, y)$ at times 0, 1, 2, and 3, respectively. The parameters for the calculation are $N_x = N_y = N_z = 150$, $\Delta x = \Delta y 0.15$, $\Delta t = 0.002$, and $t \in [0, 3]$.

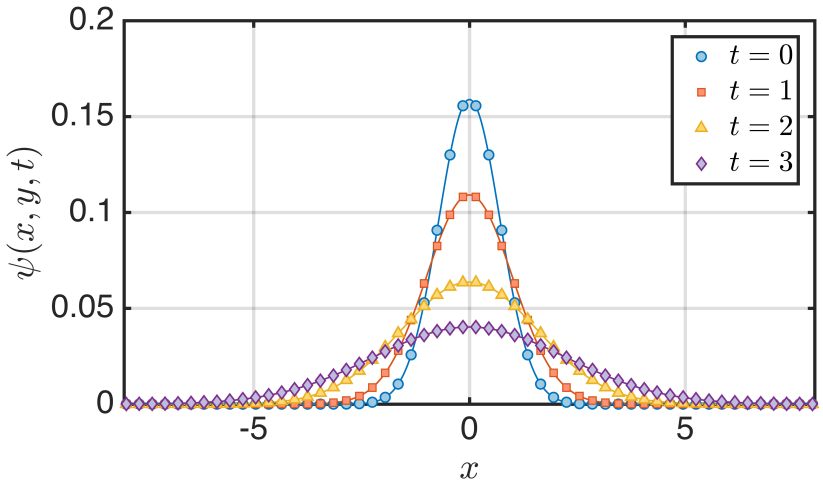

FIGURE 17: Comparison of x line cuts (at $y = 0$) between the numerical (points) and analytic (lines) solutions.

6 Crank-Nicolson Method

Using an explicit forward method (such as Euler's method) for solving the heat equation or Schrodinger equation can easily lead to unstable results, with the solution blowing up if tight constraints on the time and space grid steps aren't met. The spectral method using Fourier transforms is stable, but only works if the potential is smooth, the wavefunction goes to zero at the edges (to prevent aliasing, and to ensure a Fourier transform exists), and if we are working in cartesian coordinates (which does not allow one to make use of a cylindrically-symmetric geometry, for example).

Another very common approach is to use the Crank-Nicolson method, which like the forward Euler method relies again on finite differences. Finite difference methods work by calculating the derivatives at each point by considering the neighbouring (or multiple neighbouring) points around it. This is in contrast to the Fourier method, which is a global approach, since the transform of the entire solution is used when calculating the derivative spectrally. For a differential equation of the form

$$\frac{\partial \psi(x,t)}{\partial t} = f\left(x, t, \psi, \frac{\partial \psi}{\partial x}, \frac{\partial^2 \psi}{\partial x^2}\right), \tag{191}$$

the Crank-Nicolson discretization scheme *in time* looks like the following

$$\psi(x, t_{k+1}) \approx \psi(x, t_k) + \frac{\Delta t}{2}\left[f\left(x, t_k, \psi(x,t_k), \frac{\partial \psi(x,t_k)}{\partial x}, \frac{\partial^2 \psi(x,t_k)}{\partial x^2}\right) + ... \right. \tag{192}$$

$$\left. f\left(x, t_{k+1}, \psi(x,t_{k+1}), \frac{\partial \psi(x,t_{k+1})}{\partial x}, \frac{\partial^2 \psi(x,t_{k+1})}{\partial x^2}\right)\right]. \tag{193}$$

We can see that in order to calculate ψ at time t_{k+1}, then we need to know ψ at time t_k, i.e the previous value, as well as the function f (the RHS) at both times t_k and t_{k+1}. However, f at t_{k+1} depends on ψ at t_{k+1}, which is what we are trying to calculate. Therefore, this is called an *implicit* finite difference scheme, and generally requires solving a system of equations at each time step. If only the first term is used above, i.e. the function f at time t_k only, then this is equivalent to the regular forward Euler method, and would then be an explicit method because everything needed to calculate the next time step is already known. The evolution of the function $\psi(x, t_k)$ is shown schematically, in Fig. 18, where the space variable x is still continuous at this stage. The Crank-Nicolson scheme iterates over subsequent time steps to evaluate the time evolution of $\psi(x)$.

In order to apply the scheme numerically on the computer to solve a differential equation, we would also usually need to discretize the spatial (x) coordinate onto a grid. This is done by defining N_x grid positions, indexed by the integer m where $m = 1, 2, ..., N_x$, so that a given point is denoted by x_m, as shown in Fig. 19. Note that at the moment these grid positions are general, and the grid need not be equally space, and can even change dynamically in time if necessary. Now, the Crank-Nicolson scheme must be valid at all spatial grid points x_m, and so we can incorporate the spatial discretization to write

$$\psi(x_m, t_{k+1}) \approx \psi(x_m, t_k) + \frac{\Delta t}{2}\left[f\left(x_m, t_k, \psi(x_m,t_k), \left.\frac{\partial \psi(x,t_k)}{\partial x}\right|_{x=x_m}, \left.\frac{\partial^2 \psi(x,t_k)}{\partial x^2}\right|_{x=x_m}\right) + ... \right. \tag{194}$$

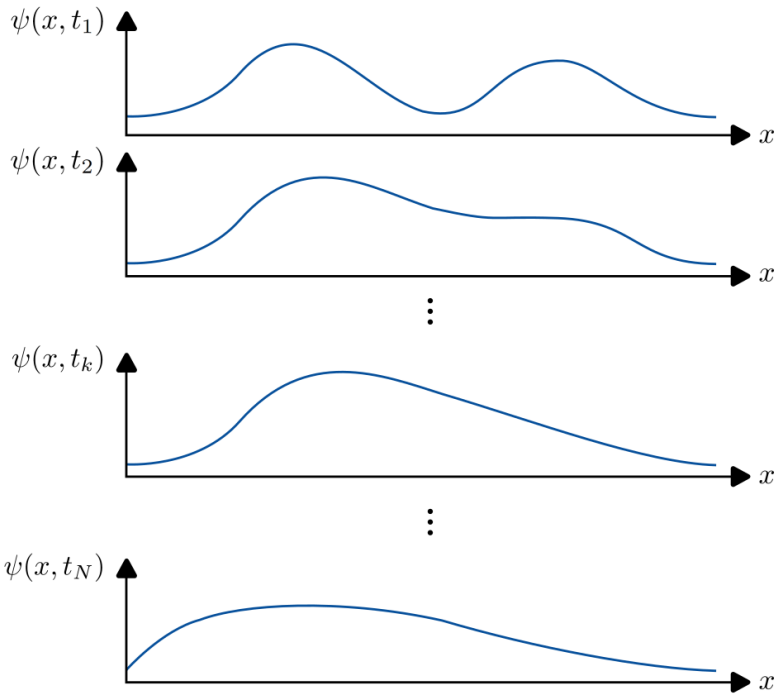

FIGURE 18: Time discretisation of the function $\psi(x, t_k)$.

$$f\left(x_m, t_{k+1}, \psi(x_m, t_{k+1}), \left.\frac{\partial\psi(x, t_{k+1})}{\partial x}\right|_{x=x_m}, \left.\frac{\partial^2\psi(x, t_{k+1})}{\partial x^2}\right|_{x=x_m}\right]. \quad (195)$$

If we use the notation $\psi(x_m, t_k) := \psi_m^k$, and the RHS function evaluated at the points t_k and x_m to be f_m^k, this can be written more compactly as

$$\psi_m^{k+1} \approx \psi_m^k + \frac{\Delta t}{2}\left[f_m^k + f_m^{k+1}\right]. \quad (196)$$

This gives a way to calculate the wavefunction at the subsequent time t_{k+1}, provided the functions f_m^k and f_m^{k+1} are known. Often, these will contain derivatives in the partial differential equation, and these derivatives can be estimated numerically using finite difference methods.

6.1 Example - Schrodinger Equation with Crank-Nicolson

Consider the time-dependent one-dimensional Schrodinger equation

$$i\hbar\frac{\partial\psi(x,t)}{\partial t} = \hat{H}\,\psi(x,t) = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right]\psi(x,t) \quad (197)$$

$$\implies \frac{\partial\psi(x,t)}{\partial t} = -\frac{i}{\hbar}\hat{H}\,\psi(x,t) = -\frac{i}{\hbar}\left[-\frac{\hbar^2}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x)\,\psi(x,t)\right]. \quad (198)$$

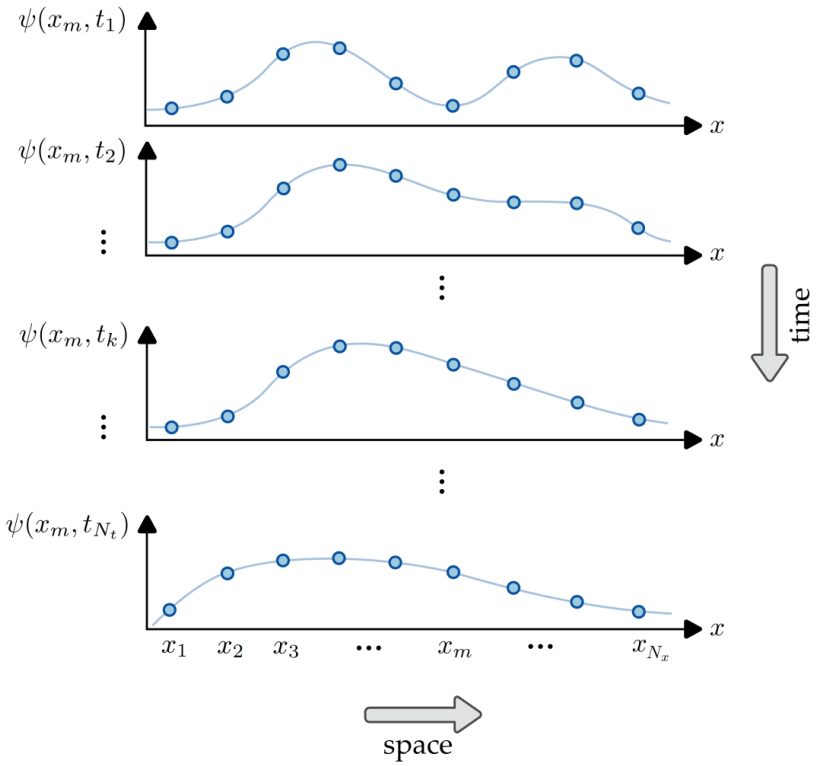

FIGURE 19: Space and time discretisation of the function $\psi(x,t) \to \psi(x_m, t_k) = \psi_m^k$.

The Crank-Nicolson scheme for discretization of the time variable specifies the following updating formula

$$\psi_m^{k+1} \approx \psi_m^k + \frac{\Delta t}{2}\left[f_m^k + f_m^{k+1}\right], \tag{199}$$

and using a second-order central finite difference to calculate the spatial second derivative we can write

$$\psi_m^{k+1} \approx \psi_m^k - \frac{i\Delta t}{2\hbar}\left[-\frac{\hbar^2}{2m}\frac{\psi_{m+1}^k + \psi_{m-1}^k - 2\psi_m^k}{(\Delta x)^2} + V_m^k\psi_m^k - \frac{\hbar^2}{2m}\frac{\psi_{m+1}^{k+1} + \psi_{m-1}^{k+1} - 2\psi_m^{k+1}}{(\Delta x)^2} + V_m^{k+1}\psi_m^{k+1}\right]. \tag{200}$$

Now, by defining

$$\alpha = \frac{i\hbar\Delta t}{4m\,(\Delta x)^2} \quad \text{and} \quad \beta = -\frac{i\Delta t}{2\hbar}, \tag{201}$$

this simplifies to

$$\psi_m^{k+1} = \psi_m^k + \alpha\left(\psi_{m+1}^k + \psi_{m-1}^k - 2\psi_m^k + \psi_{m+1}^{k+1} + \psi_{m-1}^{k+1} - 2\psi_m^{k+1}\right) + \beta\left(V_m^k\psi_m^k + V_m^{k+1}\psi_m^{k+1}\right). \tag{202}$$

This can be rearranged to collect all the terms at time $k+1$ on the left hand side, and all the terms at time k on the right hand side

$$\left(1+2\alpha-\beta V_m^{k+1}\right)\psi_m^{k+1}-\alpha\psi_{m+1}^{k+1}-\alpha\psi_{m-1}^{k+1}=\left(1-2\alpha+\beta V_m^{k}\right)\psi_m^{k}+\alpha\psi_{m+1}^{k}+\alpha\psi_{m-1}^{k}. \tag{203}$$

This represents a system of equations - one for each grid point at each spatial position x. For example, assuming we know the entire spatial distribution of the wavefunction at time t_k, and we want to calculate the grid point values at time t_{k+1}, the equations read

$$\begin{aligned}
&m=1: \quad \psi_1^{k+1}=0 \quad \text{(set by boundary condition)}\\
&m=2: \quad \left(1+2\alpha-\beta V_2^{k+1}\right)\psi_2^{k+1}-\alpha\psi_3^{k+1}-\alpha\psi_1^{k+1}=\left(1-2\alpha+\beta V_2^{k}\right)\psi_2^{k}+\alpha\psi_3^{k}+\alpha\psi_1^{k}\\
&m=3: \quad \left(1+2\alpha-\beta V_3^{k+1}\right)\psi_3^{k+1}-\alpha\psi_4^{k+1}-\alpha\psi_2^{k+1}=\left(1-2\alpha+\beta V_3^{k}\right)\psi_3^{k}+\alpha\psi_4^{k}+\alpha\psi_2^{k}\\
&\vdots\\
&m=N_x-2: \left(1+2\alpha-\beta V_{N_x-2}^{k+1}\right)\psi_{N_x-2}^{k+1}-\alpha\psi_{N_x-1}^{k+1}-\alpha\psi_{N_x-1}^{k+1}=\left(1-2\alpha+\beta V_{N_x-2}^{k}\right)\psi_{N_x-2}^{k}+\alpha\psi_{N_x-1}^{k}+\alpha\psi_{N_x-1}^{k}\\
&m=N_x-1: \left(1+2\alpha-\beta V_{N_x-1}^{k+1}\right)\psi_{N_x-1}^{k+1}-\alpha\psi_{N_x}^{k+1}-\alpha\psi_{N_x-2}^{k+1}=\left(1-2\alpha+\beta V_{N_x-1}^{k}\right)\psi_{N_x-1}^{k}+\alpha\psi_{N_x}^{k}+\alpha\psi_{N_x-2}^{k}\\
&m=N_x: \quad \psi_{N_x}^{k+1}=0 \quad \text{(set by boundary condition)}
\end{aligned}$$

where I have chosen to set the boundary conditions manually as zero at the edges. We can see that the equation for any given grid point depends on the neighbouring grid points - as expected since we calculated the derivative using finite difference between neighbours. The grid arrangement is shown schematically in Fig. 20.

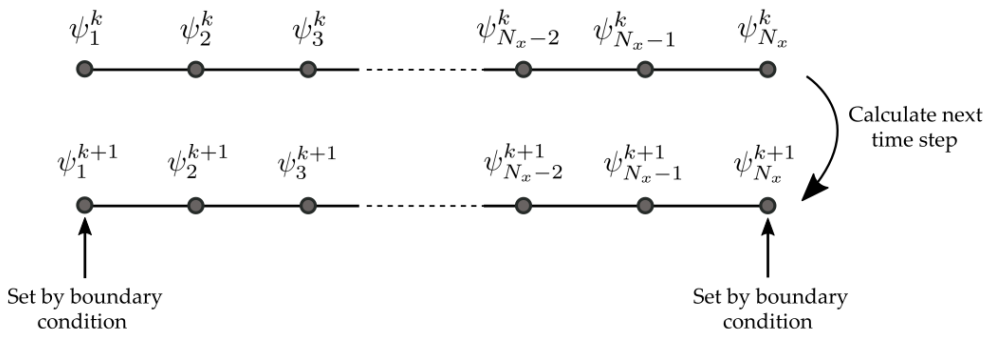

FIGURE 20: Grid arrangement when going from time t_k to time t_{k+1}.

The system of equations can be written in terms of the wavefunction vectors, and two *tridiagonal* matrices. This means that the matrix is zero everywhere except for the main diagonal and each of the

diagonals above and below the main one.

$$\begin{pmatrix} 1+2\alpha-\beta V_2^{k+1} & -\alpha & 0 & \dots & 0 \\ -\alpha & 1+2\alpha-\beta V_3^{k+1} & -\alpha & \ddots & \vdots \\ 0 & -\alpha & \ddots & \alpha & 0 \\ \vdots & \ddots & -\alpha & 1+2\alpha-\beta V_{N_x-2}^{k+1} & -\alpha \\ 0 & \dots & 0 & -\alpha & 1+2\alpha-\beta V_{N_x-1}^{k+1} \end{pmatrix} \begin{pmatrix} \psi_2^{k+1} \\ \psi_3^{k+1} \\ \vdots \\ \psi_{N_x-2}^{k+1} \\ \psi_{N_x-1}^{k+1} \end{pmatrix} =$$

$$\begin{pmatrix} 1-2\alpha+\beta V_2^{k} & \alpha & 0 & \dots & 0 \\ \alpha & 1-2\alpha+\beta V_3^{k} & \alpha & \ddots & \vdots \\ 0 & \alpha & \ddots & \alpha & 0 \\ \vdots & \ddots & \alpha & 1-2\alpha+\beta V_{N_x-2}^{k} & \alpha \\ 0 & \dots & 0 & \alpha & 1-2\alpha+\beta V_{N_x-1}^{k} \end{pmatrix} \begin{pmatrix} \psi_2^{k} \\ \psi_3^{k} \\ \vdots \\ \psi_{N_x-2}^{k} \\ \psi_{N_x-1}^{k} \end{pmatrix}$$

or simply as

$$\mathbf{A}\psi^{k+1} = \mathbf{B}\psi^k, \tag{204}$$

and so the wavefunction at a subsequent time can be found by matrix inversion to be given in terms of the tridiagonal matrices

$$\psi^{k+1} = \mathbf{A}^{-1}\mathbf{B}\psi^k = \mathbf{M_{CN}}\psi^k. \tag{205}$$

In fact, if the potential does not change with time, the Crank-Nicolson matrix $\mathbf{M_{CN}}$ need only be calculated once at the start of the loop, since it does not change during the time evolution. The following code calculates the time evolution by creating the above matrices and then iterating over time steps. The matrices $\mathbf{A}$ and $\mathbf{B}$ are created as sparse matrices to save memory, since they mostly contain zeroes.

```
hbar = 1.055e-34; % Reduced Planck constant
mass = 1.443e-25; % Mass of Rb87

%%% Set up discrete time grid %%%
dt = 30e-6;      % Time step
T = 100e-3;      % Duration of total time window
t = 0:dt:T-dt;   % Define time vector
Nt = length(t); % Number of time iterations

%%% Set up discrete space grid %%%
Nx = 201;        % Number of x grid points
dx = 0.2e-6;     % Spacing of x grid
x = -((Nx-1)*dx/2):dx:((Nx-1)*dx/2);

%%% Set up trapping potential %%%
omega = 2*pi*15;
V = (1/2)*mass*omega^2*x.^2;
```

```
%%% Construct the forward and backward CN matrices (sparse)
% https://www.mathworks.com/matlabcentral/answers/3006-tridiagonal-matrix-thomas-algorithm
alpha = 1i*hbar*dt/(4*mass*dx^2);
beta = -1i*dt/(2*hbar);
A = spdiags([-alpha*ones(Nx-2,1)...
             (1 + 2*alpha)-beta*V(2:Nx-1)'...
             -alpha*ones(Nx-2,1)],...
             -1:1,Nx-2,Nx-2);
B = spdiags([+alpha*ones(Nx-2,1)...
             (1 - 2*alpha)+beta*V(2:Nx-1)'...
             +alpha*ones(Nx-2,1)],...
             -1:1,Nx-2,Nx-2);
M_CN =A\B; % Create the Crank-Nicholson evolution matrix (see Matlab's backslash)

psi = zeros([Nx Nt]); % Preallocate wavefunction storage array

%%% Define initial condition
omega_initial = 2*pi*10; % Different to trap freq. to see dynamics
ax = sqrt(hbar/(mass*omega_initial));
psi0 = 1/pi^(1/4)/sqrt(ax)*exp( -x.^2/(2*ax^2) );
psi(:,1) = psi0;    % Put the first psi(x,t=0) into the matrix

%%% Iterate the equation over steps of dt using Crank-Nicholson method
for k = 1:(Nt-1)
    psi(2:Nx-1,k+1) = M_CN*psi(2:Nx-1,k);
end
n = abs(psi).^2; % Calculate density

waterfall(x, t(1:100:end), n(:,1:100:end)'); %%% Plotting %%%
```

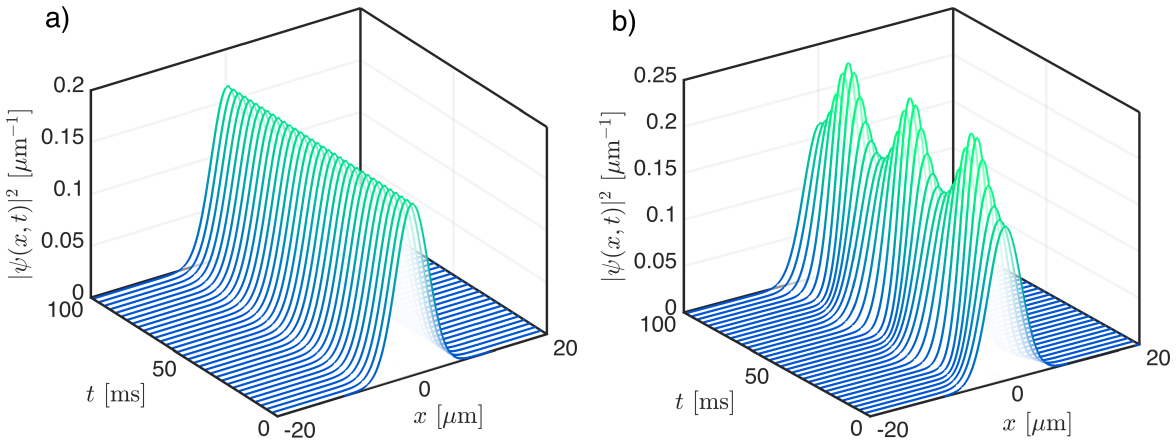

FIGURE 21: a) Evolution in a $\omega = 2\pi \times 10$ Hz trap, with the initial condition set to be the ground state of this trap. b) The same initial condition as (a) except now the evolution is in a $\omega = 2\pi \times 15$ Hz trap, leading to oscillations.

7 Gross-Pitaevskii Equation in 3D Cartesian Coordinates

The macroscopic wavefunction of a Bose-Einstein condensate in three dimensions (3D) in an external trapping potential including interactions can be described by a Schrödinger equation with a non-linear term, known as the Gross-Pitaevskii equation

$$i\hbar\frac{\partial\psi(\mathbf{r},t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g_{\text{3D}}|\psi(\mathbf{r},t)|^2 \right)\psi(\mathbf{r},t), \tag{206}$$

where ∇^2 is the three-dimensional Laplacian in cartesian coordinates

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}. \tag{207}$$

The complex scalar classical field (wavefunction) $\psi(\mathbf{r},t)$ has been used to replace the expectation value of the field operator $\langle\hat{\psi}(\mathbf{r},t)\rangle$, which is valid at very low or zero temperature, i.e. when the depletion of the condensate is small. Here the interactions are treated as s-wave, which is valid at low temperatures and densities, and are considered as a mean-field experienced by the condensate depending on its local density. The strength of the interactions is characterised by the 3D interaction coupling constant

$$g_{\text{3D}} = \frac{4\pi\hbar^2 a_s}{m}, \tag{208}$$

where a_s is the scattering length of the atom. For the S=1 triplet state of rubidium-87 we have $a_s = 98.98\,a_0 = 5.24\text{nm}$. The density of the condensate is defined as the square modulus of the complex wavefunction

$$n(\mathbf{r},t) = \psi^*(\mathbf{r},t)\;\psi(\mathbf{r},t) = |\psi(\mathbf{r},t)|^2 \tag{209}$$

and the normalisation requires that the integral at each time must equal the total number of particles present

$$\iiint_{-\infty}^{+\infty} |\psi(\mathbf{r},t)|^2\,\mathrm{d}\mathbf{r} = N. \tag{210}$$

Note that an alternative form of the GPE is sometimes used, given by

$$i\hbar\frac{\partial\psi(\mathbf{r},t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g_{\text{3D}}N|\psi(\mathbf{r},t)|^2 \right)\psi(\mathbf{r},t), \tag{211}$$

where the only difference is that now the atom number N appears as part of the non-linear term. The consequence of this is only that the normalisation of the wavefunction changes so that rather than integrating to give the total atom number it will integrate to unity

$$\iiint_{-\infty}^{+\infty} |\psi(\mathbf{r},t)|^2\,\mathrm{d}\mathbf{r} = 1. \tag{212}$$

See Refs. [1–4] for examples of this unity-normalised version being used, or Refs. [5–7] for examples of the first version normalised to total atom number. The only difference is a factor of $\sqrt{N}$ between the two wavefunctions (and therefore a factor of N in the density distributions).

7.1 Analytic Limits

Analytic limits are useful for verifying the numerical results, as well as for providing approximate initial conditions for finding a true ground state.

7.1.1 Non-Interacting Regime

In the case where the interactions are neglected, the equation reduces to a Schrödinger equation with an external potential. If the potential is harmonic of the form

$$V_{\text{ext}}(\mathbf{r}) = \frac{1}{2}m\left(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2\right), \tag{213}$$

then the solution for the wavefunction will be a Gaussian

$$\psi(\mathbf{r}) = \sqrt{\frac{N}{(\pi)^{3/2} a_x a_y a_z}} \exp\left(-\frac{x^2}{2a_x^2} - \frac{y^2}{2a_y^2} - \frac{z^2}{2a_z^2}\right) \tag{214}$$

where $a_i = \sqrt{\hbar/m\omega_i}$ are the harmonic oscillator lengths in each of the three directions, which are equivalent to what we would call the σ ($1/\sqrt{e}$ of the peak) of the Gaussian. The density distribution is then given by

$$n(\mathbf{r}) = |\psi(\mathbf{r})|^2 = \frac{N}{(\pi)^{3/2} a_x a_y a_z} \exp\left(-\frac{x^2}{a_x^2} - \frac{y^2}{a_y^2} - \frac{z^2}{a_z^2}\right). \tag{215}$$

This wavefunction is normalised such that its square modulus integrates over all space to give the total number of particles

$$\iiint_{-\infty}^{+\infty} |\psi(\mathbf{r})|^2 \, \mathrm{d}\mathbf{r} = \iiint_{-\infty}^{+\infty} n(\mathbf{r}) \, \mathrm{d}\mathbf{r} = N. \tag{216}$$

7.1.2 Interaction-Dominated Regime

When the interactions are significant, the wavefunction (and therefore density) distribution can deviate significantly from the Gaussian case above. Another simple limit is found when the kinetic energy term (the laplacian) is neglected, and the interaction energy is assumed to dominate - this is known as the 3D Thomas-Fermi limit.

To find this limit, we first obtain a time-independent version of the equation, by assuming that the time dependence of the wavefunction can be captured by an evolving phase factor, by writing $\psi(\mathbf{r}, t) \to \psi(\mathbf{r}) \exp\left(-i\mu_{3\text{D}}\, t/\hbar\right)$, where $\mu_{3\text{D}}$ is the chemical potential of the condensate in the three dimensional limit (note that the chemical potential has a different value in 1D). Making this replacement in the GPE and carrying out the derivative of the phase factor leads to the following time-independent

equation

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g_{3\text{D}}|\psi(\mathbf{r})|^2 \right)\psi(\mathbf{r}) = \mu_{3\text{D}}\,\psi(\mathbf{r}). \tag{217}$$

Now, if the kinetic energy term is neglected, this can be simply rearranged to give

$$|\psi(\mathbf{r})|^2 = n(\mathbf{r}) = \frac{1}{g_{3\text{D}}}\Big(\mu_{3\text{D}} - V_{\text{ext}}(\mathbf{r})\Big), \tag{218}$$

for $\mu_{3\text{D}} > V_{\text{ext}}(\mathbf{r})$ and zero otherwise. This shows that the density distribution directly reflects the external trapping potential in the interaction-dominated regime, and can be visualised as the the interactions "pressing" the cloud into the potential. For the case of a harmonic potential, we can specify the density even further

$$|\psi(\mathbf{r})|^2 = n(\mathbf{r}) = \frac{1}{g_{3\text{D}}}\Big(\mu_{3\text{D}} - \frac{1}{2}m\left[\omega_x^2x^2 + \omega_z^2z^2 + \omega_y^2y^2\right]\Big) \tag{219}$$

$$= \frac{\mu_{3\text{D}}}{g_{3\text{D}}}\left(1 - \frac{x^2}{R_x^2} - \frac{y^2}{R_y^2} - \frac{z^2}{R_z^2}\right), \tag{220}$$

where $R_i = \sqrt{2\mu_{3\text{D}}/\left(m\omega_i^2\right)}$ is the distance at which the density goes to zero in each direction, and is known as the Thomas-Fermi radius of the condensate. In this case, the density distribution takes the form of an inverted parabola, due to the shape of the harmonic potential. We can see from this equation that the peak density is given by $n(\mathbf{r}=0) = \mu_{3\text{D}}/g_{3\text{D}}$. Assuming a constant phase across the entire condensate, the ground state wavefunction can be obtained by taking the square root of the density

$$\psi(\mathbf{r}) = \sqrt{\frac{\mu_{3\text{D}}}{g_{3\text{D}}}}\left(1 - \frac{x^2}{R_x^2} - \frac{y^2}{R_y^2} - \frac{z^2}{R_z^2}\right)^{1/2}. \tag{221}$$

The normalisation condition constrains the value of $\mu_{3\text{D}}$ through the integral over all space

$$\iiint_{-\infty}^{+\infty} n(\mathbf{r})\,\mathrm{d}\mathbf{r} = N \tag{222}$$

$$\implies \iiint_{-\infty}^{+\infty} \frac{\mu_{3\text{D}}}{g_{3\text{D}}}\left(1 - \frac{x^2}{R_x^2} - \frac{y^2}{R_y^2} - \frac{z^2}{R_z^2}\right)\mathrm{d}\mathbf{r} = N \tag{223}$$

$$\implies \frac{\mu_{3\text{D}}}{g_{3\text{D}}}\frac{8\pi}{15}R_xR_yR_z = N \tag{224}$$

$$\implies n(\mathbf{r}=0) = \frac{\mu_{3\text{D}}}{g_{3\text{D}}} = \frac{15}{8\pi}\frac{N}{R_xR_yR_z}. \tag{225}$$

In terms of the trap frequencies, the chemical potential can then be written as

$$\mu_{3\text{D}} = \left(\frac{15\sqrt{2}}{32\pi}Ng_{3\text{D}}\,m^{3/2}\omega_x\omega_y\omega_z\right)^{2/5} = \frac{1}{2}\hbar\omega_{\text{ho}}\left(15N\frac{a}{a_{\text{ho}}}\right)^{2/5}, \tag{226}$$

where the average harmonic oscillator length $a_{\text{ho}} = \sqrt{\hbar/(m\omega_{\text{ho}})}$ and the geometric average of the trap frequencies is $\omega_{\text{ho}} = \left(\omega_x\,\omega_y\,\omega_z\right)^{1/3}$. The 3D Thomas-Fermi description is expected to be accurate when the interaction energy is much greater than the kinetic energy, which leads to the validity condition in each direction

$$\frac{N|a_s|}{a_i} \gg 1, \tag{227}$$

where the magnitude is taken because the scattering length can be positive (repulsive interactions) or negative (attractive interactions). So, for a spherical trap with frequencies $\omega_x = \omega_y = \omega_z = (2\pi) \times 100\text{Hz}$ we require $N \gg 200$ atoms. This equivalently amounts to requiring that $\mu_{\text{3D}} \gg \hbar\omega_x, \hbar\omega_y, \hbar\omega_z$, i.e. the chemical potential energy is much greater than the harmonic oscillator level spacing. As long as these conditions are fulfilled, the density will take the form of a three-dimensional parabola.

7.2 Energy Expressions

The total mean energy per particle of the system is obtained in general by taking the expectation value of the Hamiltonian operator, which is written as

$$E_{\text{tot}} = \langle\psi|\hat{H}|\psi\rangle \tag{228}$$

and since the GPE in Eq. 211 can be formally written in the form

$$i\hbar\frac{\partial\psi(\mathbf{r},t)}{\partial t} = \hat{H}\psi(\mathbf{r},t), \tag{229}$$

we can recognise the single-particle Hamiltonian operator which can be broken down into the sum of the kinetic, potential, and interaction operator contributions

$$\hat{H} = \hat{H}_{\text{kin}} + \hat{H}_{\text{pot}} + \hat{H}_{\text{int}} \tag{230}$$

$$= \underbrace{-\frac{\hbar^2}{2m}\nabla^2}_{\text{kinetic}} + \underbrace{V_{\text{ext}}(\mathbf{r})}_{\text{potential}} + \underbrace{\frac{1}{2}g_{\text{3D}}N|\psi(\mathbf{r},t)|^2}_{\text{interaction}}. \tag{231}$$

Note that when evaluating the energy of the system, an additional factor of $1/2$ must be included as above in the interaction energy term, to ensure interaction energy between two particles is not counted twice [8–10] (this factor is not included when using the Hamiltonian to solve the equation). The total mean energy per particle is given by the sum of the individual contributions

$$E_{\text{tot}} = \langle\psi|\hat{H}|\psi\rangle \tag{232}$$

$$= \langle\psi|\left(\hat{H}_{\text{kin}} + \hat{H}_{\text{pot}} + \hat{H}_{\text{int}}\right)|\psi\rangle \tag{233}$$

$$= \langle\psi|\hat{H}_{\text{kin}}|\psi\rangle + \langle\psi|\hat{H}_{\text{pot}}|\psi\rangle + \langle\psi|\hat{H}_{\text{int}}|\psi\rangle \tag{234}$$

$$= E_{\text{kin}} + E_{\text{pot}} + E_{\text{int}}. \tag{235}$$

Kinetic Energy

Now I will calculate the kinetic energy contribution to the total energy. The expectation bra-ket is evaluated with an integral over all space, using the wavefunction and its complex conjugate, given by

$$E_{\text{kin}} = \langle\psi|\hat{H}_{\text{kin}}|\psi\rangle \tag{236}$$

$$= \iiint_{-\infty}^{+\infty} \psi^* \left\{ -\frac{\hbar^2}{2m}\nabla^2 \right\} \psi \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{237}$$

$$= \iiint_{-\infty}^{+\infty} \psi^* \left\{ -\frac{\hbar^2}{2m} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) \right\} \psi \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{238}$$

$$\boxed{E_{\text{kin}} = -\frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \psi^* \left(\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2} \right) \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z.} \tag{239}$$

We can see that calculating the mean kinetic energy per particle requires calculating the sum of the partial second derivatives (Laplacian) and then integrating over the volume of the wavefunction.

Aside: an alternative form

Sometimes, the expression for the total kinetic energy Eq. 239 can be seen expressed in another form, which I will write here for completeness. Separating the individual terms in the integral, we have

$$E_{\text{kin}} = -\frac{\hbar^2}{2m} \left[\iiint_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial x^2} \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z + \iiint_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial y^2} \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z + \iiint_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial z^2} \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \right] \tag{240}$$

$$= -\frac{\hbar^2}{2m} \left[\iint_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial x^2} \, \mathrm{d}x \right) \mathrm{d}y \, \mathrm{d}z + \iint_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial y^2} \, \mathrm{d}y \right) \mathrm{d}x \, \mathrm{d}z + \iint_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial z^2} \, \mathrm{d}z \right) \mathrm{d}x \, \mathrm{d}y \right], \tag{241}$$

where the order of integration has just been interchanged. Now each of the individual terms in parentheses can be carried out using integration by parts. For example, the x-coordinate integral is

$$\int_{-\infty}^{+\infty} \psi^* \frac{\partial^2\psi}{\partial x^2} \, \mathrm{d}x = \left[\psi^* \frac{\partial\psi}{\partial x} \right]_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \frac{\partial\psi^*}{\partial x} \frac{\partial\psi}{\partial x} \, \mathrm{d}x. \tag{242}$$

The boundary condition can be set to require the wavefunction goes to zero at infinity in each of the coordinates, i.e. $\psi(x = \pm\infty) = 0$, and therefore the first term goes to zero. The remaining term can be

simplified further by considering the derivative of a general complex function

$$f(x) = a(x) + i\,b(x) \tag{243}$$

$$\implies \frac{\partial f}{\partial x} = \frac{\partial}{\partial x}\left(a + i\,b\right) \tag{244}$$

$$= \frac{\partial a}{\partial x} + i\frac{\partial b}{\partial x}. \tag{245}$$

Next if we consider the complex conjugate of f, we have

$$f^*(x) = a(x) - i\,b(x) \tag{246}$$

$$\implies \frac{\partial f^*}{\partial x} = \frac{\partial}{\partial x}\left(a - i\,b\right) \tag{247}$$

$$= \frac{\partial a}{\partial x} - i\frac{\partial b}{\partial x}, \tag{248}$$

where we can compare these expressions now to see that in general the derivative of a complex conjugate is the complex conjugate of the derivative

$$\frac{\partial f^*}{\partial x} = \left(\frac{\partial f}{\partial x}\right)^*. \tag{249}$$

Therefore, Eq. 242 can be simplified to

$$\int_{-\infty}^{+\infty} \psi^* \frac{\partial^2 \psi}{\partial x^2}\,\mathrm{d}x = -\int_{-\infty}^{+\infty} \frac{\partial \psi^*}{\partial x}\frac{\partial \psi}{\partial x}\,\mathrm{d}x \tag{250}$$

$$= -\int_{-\infty}^{+\infty} \left(\frac{\partial \psi}{\partial x}\right)^* \frac{\partial \psi}{\partial x}\,\mathrm{d}x \tag{251}$$

$$= -\int_{-\infty}^{+\infty} \left|\frac{\partial \psi}{\partial x}\right|^2 \mathrm{d}x. \tag{252}$$

Inserting this expression now into Eq. 241, and following the same procedure for the y and z derivatives, we obtain

$$E_{\text{kin}} = \frac{\hbar^2}{2m}\iiint_{-\infty}^{+\infty}\left(\left|\frac{\partial \psi}{\partial x}\right|^2 + \left|\frac{\partial \psi}{\partial y}\right|^2 + \left|\frac{\partial \psi}{\partial z}\right|^2\right)\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = \frac{\hbar^2}{2m}\iiint_{-\infty}^{+\infty}|\nabla\psi|^2\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \tag{253}$$

This expression for the kinetic energy can be found for example in Ref. [9], and is equivalent to Eq. 239. Whereas the latter requires calculating second derivatives (since it contains the laplacian), the former requires only calculating first derivatives and then taking their absolute square. This can be useful if solving equations with finite differences, where the computation time required increases for higher order derivatives (it is not necessary when calculating the derivatives spectrally, because higher order derivatives are easily calculated by muliplying by higher powers of the wavevectors in

Fourier space in that case). Both versions are included here only for reference.

It is also possible to separate the contributions to the mean kinetic energy per particle associated with each of the three directions [4], evaluating them individually as

$$E_{\text{kin}} = E_{\text{kin}}^{(x)} + E_{\text{kin}}^{(y)} + E_{\text{kin}}^{(z)}, \tag{254}$$

where

$$E_{\text{kin}}^{(x)} = -\frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \psi^* \frac{\partial^2 \psi}{\partial x^2} \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{255a}$$

$$E_{\text{kin}}^{(y)} = -\frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \psi^* \frac{\partial^2 \psi}{\partial y^2} \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{255b}$$

$$E_{\text{kin}}^{(z)} = -\frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \psi^* \frac{\partial^2 \psi}{\partial z^2} \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z, \tag{255c}$$

or using the alternative form in terms of first derivatives (instead of second derivatives), obtained using integration by parts as described above

$$E_{\text{kin}}^{(x)} = \frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \left| \frac{\partial \psi}{\partial x} \right|^2 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{256a}$$

$$E_{\text{kin}}^{(y)} = \frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \left| \frac{\partial \psi}{\partial y} \right|^2 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{256b}$$

$$E_{\text{kin}}^{(z)} = \frac{\hbar^2}{2m} \iiint_{-\infty}^{+\infty} \left| \frac{\partial \psi}{\partial z} \right|^2 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z. \tag{256c}$$

Potential Energy

The contribution from the external potential to the total energy is much simpler to evaluate than the kinetic one because the potential operator is diagonal in position basis, which leads to a simple point-wise multiplication (in contrast to the kinetic operator which is diagonal only in momentum basis). The mean potential energy per particle is given by

$$E_{\text{pot}} = \langle \psi | \hat{H}_{\text{pot}} | \psi \rangle \tag{257}$$

$$= \iiint_{-\infty}^{+\infty} \psi^* \Big\{ V_{\text{ext}}(x, y, z) \Big\} \psi \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{258}$$

$$= \iiint_{-\infty}^{+\infty} V_{\text{ext}}(x, y, z) \, |\psi|^2 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z. \tag{259}$$

Interaction Energy

Similarly, the mean interaction energy per particle is also simple to evaluate and is given by

$$E_{\text{int}} = \langle\psi|\hat{H}_{\text{int}}|\psi\rangle \tag{260}$$

$$= \iiint_{-\infty}^{+\infty} \psi^* \left\{ \frac{1}{2} g_{3\text{D}} N |\psi|^2 \right\} \psi \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{261}$$

$$= \frac{g_{3\text{D}} N}{2} \iiint_{-\infty}^{+\infty} |\psi|^4 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z. \tag{262}$$

Chemical Potential

The chemical potential is often described as "the energy required to add a particle to the system", and its thermodynamic definition is given by

$$\mu = \frac{\mathrm{d}E[\psi^*, \psi, N]}{\mathrm{d}N}, \tag{263}$$

where the total energy functional (meaning, a function of a function) is given by [10]

$$E[\psi^*, \psi, N] = N \iiint_{-\infty}^{+\infty} \frac{\hbar^2}{2m} |\nabla\psi|^2 + V_{\text{ext}}(\mathbf{r})|\psi|^2 + \frac{g_{3\text{D}} N}{2} |\psi|^4 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z. \tag{264}$$

Note here that compared with the energy expressions above, the total energy includes an extra factor of N out front, since before we were dealing with energy per particle, but here we are dealing with the total energy of the gas. Now, the variation of ψ and ψ^* with N vanish to first order [10], and so the chemical potential is given by

$$\mu = \frac{\mathrm{d}E[\psi^*, \psi, N]}{\mathrm{d}N} \tag{265}$$

$$= \frac{\partial E}{\partial \psi^*} \frac{\partial \psi^*}{\partial N} + \frac{\partial E}{\partial \psi} \frac{\partial \psi}{\partial N} + \frac{\partial E}{\partial N} \frac{\partial N}{\partial N} \tag{266}$$

$$= \frac{\partial E}{\partial N} \tag{267}$$

$$= \frac{\partial}{\partial N} \left(N \iiint_{-\infty}^{+\infty} \frac{\hbar^2}{2m} |\nabla\psi|^2 + V_{\text{ext}}(\mathbf{r})|\psi|^2 + \frac{g_{3\text{D}} N}{2} |\psi|^4 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \right) \tag{268}$$

$$= \iiint_{-\infty}^{+\infty} \frac{\hbar^2}{2m} |\nabla\psi|^2 + V_{\text{ext}}(\mathbf{r})|\psi|^2 + g_{3\text{D}} N |\psi|^4 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z. \tag{269}$$

Note that this is almost the same as the total energy per particle expression, except for here the factor of $1/2$ in the interaction term has cancelled when carrying out the derivative with respect to N. Since this is the only difference, we can see that in the non-interacting case the chemical potential is simply

equal to the mean total energy per particle

$$\mu \text{ (non-interacting)} = E_{\text{kin}} + E_{\text{pot}}, \tag{270}$$

where the kinetic and potential energies per particle are given by Eq. 253 and Eq. 259, respectively. In the interacting case $g_{3\text{D}} \neq 0$, the chemical potential is no longer equal to the mean total energy per particle, and instead can be expressed as [9]

$$\mu \text{ (interacting)} = E_{\text{kin}} + E_{\text{pot}} + 2E_{\text{int}}. \tag{271}$$

7.3 Dimensionless Form

In the GPE, there are factors of mass m and reduced Plank constant $\hbar$, which are very small numbers. In addition, the length scales associated with a BEC are often in micrometers, and the timescales are in milliseconds. When working with such small numbers, there is a stricter requirement on the machine precision of the PC on which you are doing the calculations, the possibility of rounding errors, as well as potentially the need to store data as more precise data types (e.g. double float instead of single). This was perhaps more of a problem in the past, and is less of an issue with modern machines, but nevertheless it is still a good idea to use scales which are suitably adapted to the problem being studied. For this reason, the GPE is often written in terms of scaled lengths, scaled times, and scaled energies for the purposes of numerically solving the equation - with the results being converted back to SI units at the end if desired [11, 12].

For converting to dimensionless units, we will use the GPE in the form from Eq. 211, reporoduced here

$$i\hbar\frac{\partial\psi(\mathbf{r},t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g_{3\text{D}}N|\psi(\mathbf{r},t)|^2 \right)\psi(\mathbf{r},t), \tag{272}$$

which leads to a wavefunction normalised to unity

$$\iiint_{-\infty}^{+\infty} |\psi(\mathbf{r},t)|^2 \,\mathrm{d}\mathbf{r} = 1. \tag{273}$$

It is a common approach to then scale all the spatial dimensions by a length such that

$$\tilde{x} = x/l \qquad \tilde{y} = y/l \qquad \tilde{z} = z/l \tag{274}$$

where the tilde notation denotes the scaled version of each variable. In principle, the length l can be chosen as anything, for example $l = 1\mu\text{m}$ would be a valid choice, but it is usually chosen to be the harmonic oscillator length in one of the directions. If the trap was cylindrically symmetric, it is sensible to choose the tightly confining radial direction which has $\omega_\perp = \omega_x = \omega_y$, so the length scale would be

$$l = \sqrt{\frac{\hbar}{m\omega_x}}. \tag{275}$$

For a trap frequency of $\omega_x = 2\pi \times 1\text{kHz}$ with rubidium-87, this length takes the value $l = 0.34\mu\text{m}$.

Similarly, the time coordinate is usually scaled by the radial trap frequency, which provides a time scale for any dynamics

$$\tilde{t} = \omega_x \, t. \tag{276}$$

Finally the wavefunction is scaled as

$$\tilde{\psi}(\tilde{\mathbf{r}}, \tilde{t}) = \tilde{\psi}(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{t}\,) = \sqrt{l^3}\, \psi(x, y, z, t), \tag{277}$$

where the factor of $\sqrt{l^3}$ is a reflection of the fact that $|\psi|^2$ is a density with units $[\mathrm{m}^{-3}]$. To convert the GPE to dimensionless variables, it is simply a case of plugging in the scaled quantities and evaluating them term by term.

Scaled Time Derivative

The time derivative of the left hand side of the GPE can be evaluated by the chain rule

$$\frac{\partial \psi}{\partial t} = \frac{\partial \psi}{\partial \tilde{t}} \frac{\partial \tilde{t}}{\partial t}, \tag{278}$$

and since

$$\tilde{t} = \omega_x \, t \implies \frac{\partial \tilde{t}}{\partial t} = \omega_x \tag{279}$$

this becomes

$$\frac{\partial \psi}{\partial t} = \omega_x \frac{\partial \psi}{\partial \tilde{t}} \tag{280}$$

$$= \omega_x \frac{\partial}{\partial \tilde{t}} \left(\frac{1}{\sqrt{l^3}} \tilde{\psi} \right) \tag{281}$$

$$= \frac{\omega_x}{\sqrt{l^3}} \frac{\partial \tilde{\psi}}{\partial \tilde{t}}. \tag{282}$$

Scaled Spatial Derivatives

We can follow a similar procedure to evaluate the spatial second derivatives in the Laplacian of the GPE. For example in the x-direction we have

$$\frac{\partial^2 \psi}{\partial x^2} = \frac{\partial}{\partial x} \left(\frac{\partial \psi}{\partial x} \right) \tag{283}$$

$$= \frac{\partial}{\partial x} \left(\frac{\partial \psi}{\partial \tilde{x}} \frac{\partial \tilde{x}}{\partial x} \right) \tag{284}$$

$$= \frac{\partial}{\partial x} \left(\frac{\partial \psi}{\partial \tilde{x}} \frac{1}{l} \right) \quad \text{(since } \tilde{x} = x/l\text{)} \tag{285}$$

$$= \frac{1}{l} \frac{\partial}{\partial x} \left(\frac{\partial \psi}{\partial \tilde{x}} \right) \tag{286}$$

$$= \frac{1}{l} \frac{\partial}{\partial \tilde{x}} \left(\frac{\partial \psi}{\partial \tilde{x}} \right) \frac{\partial \tilde{x}}{\partial x} \quad \text{(using the chain rule)} \tag{287}$$

$$= \frac{1}{l} \frac{\partial}{\partial \tilde{x}} \left(\frac{\partial \psi}{\partial \tilde{x}} \right) \frac{1}{l} \tag{288}$$

$$= \frac{1}{l^2}\frac{\partial^2\psi}{\partial\tilde{x}^2} \tag{289}$$

$$= \frac{1}{l^2}\frac{\partial^2}{\partial\tilde{x}^2}\left(\frac{1}{\sqrt{l^3}}\tilde{\psi}\right) \tag{290}$$

$$= l^{-7/2}\,\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}. \tag{291}$$

Following the same procedure, we get similar expressions for the other directions

$$\frac{\partial^2\psi}{\partial y^2} = l^{-7/2}\,\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2} \qquad \text{and} \qquad \frac{\partial^2\psi}{\partial z^2} = l^{-7/2}\,\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}. \tag{292}$$

Scaled Potential Term

The external potential term becomes simply

$$V_{\text{ext}}(\mathbf{r})\,\psi(\mathbf{r},t) = \frac{1}{\sqrt{l^3}}\,V_{\text{ext}}(\tilde{\mathbf{r}})\,\tilde{\psi}(\tilde{\mathbf{r}},t). \tag{293}$$

Scaled Interaction Term

Finally the non-linear interaction term is

$$g_{3\text{D}}N|\psi|^2\psi = g_{3\text{D}}N\left|\frac{1}{\sqrt{l^3}}\tilde{\psi}\right|^2\frac{1}{\sqrt{l^3}}\tilde{\psi} = l^{-9/2}\,g_{3\text{D}}N|\tilde{\psi}|^2\tilde{\psi}. \tag{294}$$

Final Scaled Equation

Now that we have all the individual scaled terms, they can all be put back together into the original GPE

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right) + V_{\text{ext}}\psi + g_{3\text{D}}N|\psi|^2\psi, \tag{295}$$

which after making the substitution with the scaled coordinates using the terms previously calculated becomes

$$\overset{x_i\to\tilde{x}_i}{\Longrightarrow}\quad i\hbar\frac{\omega_x}{\sqrt{l^3}}\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{\hbar^2}{2m}\frac{1}{\sqrt{l^7}}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right) + \frac{1}{\sqrt{l^3}}\,V_{\text{ext}}\,\tilde{\psi} + \frac{1}{\sqrt{l^9}}\,g_{3\text{D}}N|\tilde{\psi}|^2\tilde{\psi}. \tag{296}$$

Now inserting the expression for the length scale $l = \sqrt{\hbar/m\omega_x}$, then dividing both sides of the equation by $\left(\hbar m^3\omega_x^7\right)^{1/4}$, and using the expression for the 3D interaction constant $g_{3\text{D}} = 4\pi\hbar^2 a_s/m$, most of the factors will cancel and we finally obtain the scaled form of the GPE in cartesian coordinates

$$\boxed{i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right) + \frac{V_{\text{ext}}}{\hbar\omega_x}\tilde{\psi} + \mathcal{G}|\tilde{\psi}|^2\tilde{\psi},} \tag{297}$$

where the strength of the non-linear term has been defined in terms of the scattering length a_s by introducing a non-linearity parameter

$$\mathcal{G} = \frac{4\pi a_s N}{l}. \tag{298}$$

In the scaled GPE, we can see that the external potential term is effectively included in units of $\hbar\omega_x$, the harmonic oscillator energy level spacing corresponding to the x-direction. Note that changing to scaled cooridnates doesn't change the normalisation (because all the factors of l cancel), and the scaled wavefunction respects the same normalisation as before, i.e.

$$\iiint_{-\infty}^{+\infty} |\tilde{\psi}(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{t}\,)|^2 \,\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} = 1. \tag{299}$$

Scaled GPE for a Harmonic Trap

For the special case when the external trap is harmonic, the scalar potential can be written in the form

$$V_{\text{ext}} = \frac{1}{2} m \left(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2\right) \tag{300}$$

$$= \frac{1}{2} m\omega_x^2 \left(x^2 + \kappa^2 y^2 + \lambda^2 z^2\right), \tag{301}$$

where the parameters $\kappa = \omega_y/\omega_x$ and $\lambda = \omega_z/\omega_x$ define the anisotropy of the external trap along the remaining two dimensions. In this case, the potential term appearing in the scaled GPE can be written as

$$\frac{V_{\text{ext}}}{\hbar\omega_x}\,\tilde{\psi} = \frac{1}{\hbar\omega_x}\left[\frac{1}{2} m\omega_x^2 \left(x^2 + \kappa^2 y^2 + \lambda^2 z^2\right)\right]\tilde{\psi} \tag{302}$$

$$= \frac{m\omega_x}{2\hbar}\left[(l\tilde{x})^2 + \kappa^2 (l\tilde{y})^2 + \lambda^2 (l\tilde{z})^2\right]\tilde{\psi} \tag{303}$$

$$= \frac{m\omega_x}{2\hbar} l^2 \left(\tilde{x}^2 + \kappa^2\tilde{y}^2 + \lambda^2\tilde{z}^2\right)\tilde{\psi} \tag{304}$$

$$= \frac{1}{2}\left(\tilde{x}^2 + \kappa^2\tilde{y}^2 + \lambda^2\tilde{z}^2\right)\tilde{\psi}. \tag{305}$$

So, we can define a scaled potential function given by

$$\tilde{V}_{\text{ext}} = \frac{V_{\text{ext}}}{\hbar\omega_x} = \frac{1}{2}\left(\tilde{x}^2 + \kappa^2\tilde{y}^2 + \lambda^2\tilde{z}^2\right), \tag{306}$$

and the 3D GPE in dimensionless cartesian coordinates for a harmonic external trapping potential can be written as

$$\boxed{i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right) + \frac{1}{2}\left(\tilde{x}^2 + \kappa^2\tilde{y}^2 + \lambda^2\tilde{z}^2\right)\tilde{\psi} + \mathcal{G}|\tilde{\psi}|^2\tilde{\psi}.} \tag{307}$$

7.3.1 Dimensionless Energy Expressions

The energy can also be written in terms of dimensionless variables, by making the substitutions defined earlier for the i^{th} direction ($i = \{x, y, z\}$)

$$x_i = l\,\tilde{x}_i \qquad \mathrm{d}x_i = l\;\mathrm{d}\tilde{x}_i \qquad \frac{\partial^2\psi}{\partial x_i^2} = l^{-7/2}\,\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}_i^2} \qquad \psi = \frac{1}{\sqrt{l^3}}\,\tilde{\psi}. \tag{308}$$

Dimensionless Kinetic Energy

For the kinetic energy, making these substitutions into Eq. 239 leads to

$$E_{\text{kin}} = -\frac{\hbar^2}{2m}\iiint_{-\infty}^{+\infty}\frac{1}{\sqrt{l^3}}\,\tilde{\psi}^*\,l^{-7/2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)l^3\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} \tag{309}$$

$$= -\frac{\hbar^2}{2m}\frac{1}{l^2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} \tag{310}$$

$$= -\frac{\hbar\omega_x}{2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} \tag{311}$$

$$= -\frac{\hbar\omega_x}{2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\,\nabla^2\tilde{\psi}\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z}, \tag{312}$$

where on the penultimate line the expression for the length scale $l = \sqrt{\hbar/m\omega_x}$ has been inserted. Now we can see that a dimensionless kinetic energy $\tilde{E}_{\text{kin}}$ can be defined when working with all the scaled quantities, which is simply the energy but given in units of $\hbar\omega_x$

$$\tilde{E}_{\text{kin}} = \frac{E_{\text{kin}}}{\hbar\omega_x} = -\frac{1}{2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} = -\frac{1}{2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\,\nabla^2\tilde{\psi}\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z}. \tag{313}$$

In a similar way, the individual contributions to the total kinetic energy associated with each of the three dimensions from Eqs. (255a) to (255c) and Eqs. (256a) to (256c) can be also be written in a scaled way

$$\tilde{E}_{\text{kin}}^{(x)} = \frac{E_{\text{kin}}^{(x)}}{\hbar\omega_x} = -\frac{1}{2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} = \frac{1}{2}\iiint_{-\infty}^{+\infty}\left|\frac{\partial\tilde{\psi}}{\partial\tilde{x}}\right|^2\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} \tag{314}$$

$$\tilde{E}_{\text{kin}}^{(y)} = \frac{E_{\text{kin}}^{(y)}}{\hbar\omega_x} = -\frac{1}{2}\iiint_{-\infty}^{+\infty}\tilde{\psi}^*\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} = \frac{1}{2}\iiint_{-\infty}^{+\infty}\left|\frac{\partial\tilde{\psi}}{\partial\tilde{y}}\right|^2\;\mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z} \tag{315}$$

$$\tilde{E}_{\text{kin}}^{(z)} = \frac{E_{\text{kin}}^{(z)}}{\hbar\omega_x} = -\frac{1}{2}\iiint_{-\infty}^{+\infty} \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2}\, \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z} = \frac{1}{2}\iiint_{-\infty}^{+\infty} \left|\frac{\partial \tilde{\psi}}{\partial \tilde{z}}\right|^2 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z}. \tag{316}$$

Dimensionless Potential Energy

For the potential energy, using the scaled energy $\tilde{V}_{\text{ext}} = V_{\text{ext}}/\hbar\omega_x$ and substituting all the scaled variables into Eq. 259 we have

$$E_{\text{pot}} = \iiint_{-\infty}^{+\infty} \hbar\omega_x \tilde{V}_{\text{ext}}(\tilde{x},\tilde{y},\tilde{z}) \left|\frac{1}{\sqrt{l^3}}\tilde{\psi}\right|^2 l^3\, \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z} \tag{317}$$

$$= \hbar\omega_x \iiint_{-\infty}^{+\infty} \tilde{V}_{\text{ext}}(\tilde{x},\tilde{y},\tilde{z}) \left|\tilde{\psi}\right|^2 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z}, \tag{318}$$

and when working with scaled quantites the dimensionless potential energy $\tilde{E}_{\text{pot}}$ is the energy but given in units of $\hbar\omega_x$

$$\boxed{\tilde{E}_{\text{pot}} = \frac{E_{\text{pot}}}{\hbar\omega_x} = \iiint_{-\infty}^{+\infty} \tilde{V}_{\text{ext}}(\tilde{x},\tilde{y},\tilde{z}) \left|\tilde{\psi}\right|^2 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z}.} \tag{319}$$

Dimensionless Interaction Energy

For the interaction energy, substituting scaled variables into Eq. 262 we obtain

$$E_{\text{int}} = \frac{g_{3\text{D}}N}{2} \iiint_{-\infty}^{+\infty} \left|\frac{1}{\sqrt{l^3}}\tilde{\psi}\right|^4 l^3\, \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z} \tag{320}$$

$$= \frac{g_{3\text{D}}N}{2\,l^3} \iiint_{-\infty}^{+\infty} \left|\tilde{\psi}\right|^4 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z} \tag{321}$$

$$= \frac{4\pi\hbar^2 a_s}{m}\frac{N}{2\,l^3} \iiint_{-\infty}^{+\infty} \left|\tilde{\psi}\right|^4 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z} \tag{322}$$

$$= \frac{\mathcal{G}\hbar^2}{2m\,l^2} \iiint_{-\infty}^{+\infty} \left|\tilde{\psi}\right|^4 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z} \tag{323}$$

$$= \hbar\omega_x \frac{\mathcal{G}}{2} \iiint_{-\infty}^{+\infty} \left|\tilde{\psi}\right|^4 \mathrm{d}\tilde{x}\, \mathrm{d}\tilde{y}\, \mathrm{d}\tilde{z}, \tag{324}$$

where the expressions for the 3D interaction coupling constant $g_{3\text{D}}$ from Eq. 208 and the scaled nonlinearity constant $\mathcal{G}$ from Eq. 298 have been used. Again, we can finally define a dimensionless interaction energy when working with scaled quantities, which is just the interaction energy in units

of $\hbar\omega_x$, given by

$$\boxed{\tilde{E}_{\text{int}} = \frac{E_{\text{int}}}{\hbar\omega_x} = \frac{\mathcal{G}}{2}\iiint_{-\infty}^{+\infty} |\tilde{\psi}|^4 \, \mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z}.} \tag{325}$$

Dimensionless Chemical Potential

The chemical potential in the general interacting case is given by Eq. 271

$$\mu = E_{\text{kin}} + E_{\text{pot}} + 2E_{\text{int}}, \tag{326}$$

where the energy terms are defined as being the energies per particle. Using the results above, a scaled chemical potential can be defined

$$\tilde{\mu} = \frac{\mu}{\hbar\omega_x} = \tilde{E}_{\text{kin}} + \tilde{E}_{\text{pot}} + 2\tilde{E}_{\text{int}} \tag{327}$$

$$= \iiint_{-\infty}^{+\infty} -\frac{1}{2}\tilde{\psi}^* \nabla^2\tilde{\psi} + \tilde{V}_{\text{ext}}\left|\tilde{\psi}\right|^2 + \mathcal{G}\left|\tilde{\psi}\right|^4 \, \mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z}, \tag{328}$$

or alternatively

$$\boxed{\tilde{\mu} = \iiint_{-\infty}^{+\infty} \frac{1}{2}|\nabla\tilde{\psi}|^2 + \tilde{V}_{\text{ext}}\left|\tilde{\psi}\right|^2 + \mathcal{G}\left|\tilde{\psi}\right|^4 \, \mathrm{d}\tilde{x}\,\mathrm{d}\tilde{y}\,\mathrm{d}\tilde{z}.} \tag{329}$$

7.4 Split-Step Fourier Method for Numerical Solution

The Gross-Pitaevskii equation is a partial differential equation similar to the simple examples looked at earlier in this document, except that now it contains a non-linear term due to the effect of interactions, which makes it more difficult to solve. The GPE from Eq. 211 is formally written as

$$\frac{\partial\psi(\mathbf{r},t)}{\partial t} = -\frac{i}{\hbar}\hat{H}\,\psi(\mathbf{r},t), \tag{330}$$

where the Hamiltonian of the system is given by

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g_{3\text{D}}N|\psi(\mathbf{r},t)|^2. \tag{331}$$

Provided the Hamiltonian commutes with itself at different times, i.e. $\left[\hat{H}(t), \hat{H}(t')\right] = 0$, the general solution over time is given by rearranging the equation and integrating from t to $t + \Delta t$, leading to (see for example p.110 of Ref. [13])

$$\int \frac{1}{\psi}\,\mathrm{d}\psi = \int_{t}^{t+\Delta t} -\frac{i}{\hbar}\hat{H}\,\mathrm{d}t \tag{332}$$

$$\implies \ln(\psi) + C = -\frac{i}{\hbar}\int_{t}^{t+\Delta t} \hat{H}\ \mathrm{d}t \tag{333}$$

$$\implies \psi(\mathbf{r}, t+\Delta t) = \exp\left(-\frac{i}{\hbar}\int_{t}^{t+\Delta t} \hat{H}\ \mathrm{d}t\right)\psi(\mathbf{r}, t). \tag{334}$$

If in addition we assume the Hamiltonian is time-independent (for now - which is typically okay as long as Δt is sufficiently small), this becomes simply

$$\psi(\mathbf{r}, t+\Delta t) = \exp\left(-\frac{i}{\hbar}\hat{H}\Delta t\right)\psi(\mathbf{r}, t). \tag{335}$$

With the caveats above (i.e. the Hamiltonian commuting with itself, as well as being time-independent between time steps), this type of exponentiated form is the general exact solution of an equation of the form in Eq. 330.

The evolution equation looks deceptively simple, but it is actually not trivial to apply. The Hamiltonian can be written as a sum of a kinetic part and a (total) potential part $\hat{H} = \hat{T} + \hat{V}$, where the total potential includes both the external potential, as well as the mean-field interaction potential

$$\hat{T} = -\frac{\hbar^2}{2m}\nabla^2 \qquad \text{and} \qquad \hat{V} = V_{\text{ext}}(\mathbf{r}) + g_{\text{3D}}N|\psi(\mathbf{r}, t)|^2. \tag{336}$$

The problem is that the exponential contains the second derivatives as well as the scalar potential fields. While the potential operator is diagonal in position basis (and therefore can be calculated by a simple multiplication with the wavefunction), the kinetic operator is only diagonal in momentum basis, so the Hamiltonian itself is not diagonal in either basis. Another way to think about this is to notice that the kinetic and potential operators do not commute, and so the two exponentials cannot be separated, i.e.

$$e^{\hat{H}} = e^{\hat{T}+\hat{V}} \neq e^{\hat{T}}e^{\hat{V}}. \tag{337}$$

Exponentials cannot be separated in this way for non-commuting operators, and the error can be obtained by the Baker-Campbell-Hausdorff formula. However the following approximation (known as *Strang Splitting* [14]) holds up to error $\mathcal{O}(\Delta t^3)$

$$e^{-i\hat{H}\Delta t/\hbar} \approx e^{-i\hat{V}\Delta t/2\hbar}\, e^{-i\hat{T}\Delta t/\hbar}\, e^{-i\hat{V}\Delta t/2\hbar} + \mathcal{O}(\Delta t^3), \tag{338}$$

and higher order approximations can also be obtained if desired [15–19]. Now, applying this symmetrised Strang formula to Eq. 335, we have

$$\psi(\mathbf{r}, t+\Delta t) = e^{-i\hat{V}\Delta t/2\hbar}\, e^{-i\hat{T}\Delta t/\hbar}\, e^{-i\hat{V}\Delta t/2\hbar}\, \psi(\mathbf{r}, t), \tag{339}$$

as a way to obtain the wavefunction at some time step Δt later, given that we know the wavefunction at time t. The exponential operators need to be applied to the wavefunction from right to left, so the first one to act is the potential operator. This will produce an intermediate wavefunction $\psi^{(1/3)}$

between the one at time t and the one that we are trying to calculate at $t + \Delta t$

$$\psi^{(1/3)}(\mathbf{r}) = e^{-i\hat{V}\Delta t/2\hbar}\,\psi(\mathbf{r}, t), \tag{340}$$

and since the total potential operator $\hat{V}$ is diagonal in the position basis, this turns into a simple multiplication with the exponential of the potential

$$\psi^{(1/3)}(\mathbf{r}) = e^{-iV(\mathbf{r})\Delta t/2\hbar}\,\psi(\mathbf{r}, t), \tag{341}$$

where

$$V(\mathbf{r}) = V_{\text{ext}}(\mathbf{r}) + g_{3\text{D}}N|\psi(\mathbf{r}, t)|^2. \tag{342}$$

Next, we need to apply the exponential of the kinetic operator to this result for $\psi^{(1/3)}$ to get a further intermediate wavefunction

$$\psi^{(2/3)}(\mathbf{r}) = e^{-i\hat{T}\Delta t/\hbar}\,\psi^{(1/3)}(\mathbf{r}). \tag{343}$$

This cannot be applied directly, because the kinetic operator is diagonal only in momentum basis, but the wavefunction $\psi^{(1/3)}(\mathbf{r})$ is currently in position basis. We can instead write this step in momentum basis

$$\psi^{(2/3)}(\mathbf{k}) = e^{-i\hat{T}\Delta t/\hbar}\,\psi^{(1/3)}(\mathbf{k}). \tag{344}$$

Now, since the eigenfunctions of the momentum operator are written as

$$\hat{p}\,|k\rangle = \hbar k\,|k\rangle \tag{345}$$

and the kinetic energy operator can be written in terms of the momentum operator

$$\hat{T} = \frac{\hat{\mathbf{p}}\cdot\hat{\mathbf{p}}}{2m}, \tag{346}$$

the eigenvalues of the kinetic energy operator are [20, 21]

$$\hat{T}\,|k\rangle = \frac{\hbar^2|\mathbf{k}|^2}{2m}\,|k\rangle\,. \tag{347}$$

This means that applying the exponential of the kinetic energy operator simply picks out the eigenvalue, such that

$$\psi^{(2/3)}(\mathbf{k}) = \exp\left(-i\frac{\hat{T}\Delta t}{\hbar}\right)\,\psi^{(1/3)}(\mathbf{k}) \tag{348}$$

$$= \exp\left(-i\frac{\hbar\Delta t}{2m}|\mathbf{k}|^2\right)\,\psi^{(1/3)}(\mathbf{k}) \tag{349}$$

$$= \exp\left(-i\frac{\hbar\Delta t}{2m}|\mathbf{k}|^2\right)\,\mathcal{F}\Big\{\psi^{(1/3)}(\mathbf{r})\Big\} \tag{350}$$

where $|\mathbf{k}|^2 = k_x^2 + k_y^2 + k_z^2$, and the wavefunction $\psi^{(1/3)}(\mathbf{k})$ in momentum basis has been obtained

from its position basis version through a Fourier transform, denoted by $\mathcal{F}\{...\}$.

Finally, there is one remaining potential operator to be applied in Eq. 339. We know that this becomes a simple multiplicaion but *only* as long as the function that it is operating on is in the position representation (because that's where the potential operator is diagonal). Therefore, we cannot apply it directly to $\psi^{(2/3)}(\mathbf{k})$, but need to inverse Fourier transform to get back to position basis first

$$\psi^{(2/3)}(\mathbf{r}) = \mathcal{F}^{-1}\Big\{\psi^{(2/3)}(\mathbf{k})\Big\}. \tag{351}$$

Applying the final operator to this then gives

$$\psi(\mathbf{r}, t+\Delta t) = \exp\left(-i\frac{V(\mathbf{r})\Delta t}{2\hbar}\right)\,\psi^{(2/3)}(\mathbf{r}). \tag{352}$$

Collecting together the expressions Eqs. (341) and (350) to (352) results in

$$\boxed{\psi(\mathbf{r}, t+\Delta t) = \exp\left(-i\frac{V(\mathbf{r})\Delta t}{2\hbar}\right)\mathcal{F}^{-1}\left\{\exp\left(-i\frac{\hbar\Delta t}{2m}|\mathbf{k}|^2\right)\,\mathcal{F}\Big\{\exp\left(-i\frac{V(\mathbf{r})\Delta t}{2\hbar}\right)\,\psi(\mathbf{r},t)\Big\}\right\}.} \tag{353}$$

We have now dealt with all the operators, and everything is in terms of multiplications. Therefore if we know the wavefunction at some time t we can calculate the value at time Δt later by moving back and forth between real space and Fourier space to apply each of the kinetic and potential operators. This method is known as the *split-step Fourier method* and was previously used in optics for beams travelling through atmosphere in which there are both free propagation (through the air), and absorption (by the atmosphere) [22], and other non-linear optical effects such as self-focusing [23]. It was also more recently employed to simulate a probe beam passing through a BEC during the absorption imaging stage of an experiment [24]. It is useful in such situations when there are different processes which need to be treated in different ways, as in the GPE where we have a kinetic energy term and a potential term, and therefore found use in this field as well [11, 25, 26], and is still often used in recent research papers [27, 28].

An alternative way of understanding the split-step Fourier method:

The overall result of the split-step method described above is actually equivalent to solving the equation in several sub-steps. Here, I will show that solving in sub-steps is equivalent to the Strang splitting approximation employed in the previous section.

To recap, we are looking to solve the equation

$$\frac{\partial\psi(\mathbf{r},t)}{\partial t} = -\frac{i}{\hbar}\hat{H}\,\psi(\mathbf{r},t) \tag{354}$$

$$= -\frac{i}{\hbar}\left(\hat{T}+\hat{V}\right)\,\psi(\mathbf{r},t), \tag{355}$$

where the kinetic and (total) potential operators are, respectively

$$\hat{T} = -\frac{\hbar^2}{2m}\nabla^2 \qquad \hat{V} = V_{\text{ext}}(\mathbf{r}) + g_{\text{3D}}N|\psi(\mathbf{r},t)|^2. \tag{356}$$

Now, we would like to obtain the wavefunction $\psi(\mathbf{r}, t+\Delta t)$ from some known wavefunction $\psi(\mathbf{r}, t)$ by evolving over a time Δt. Instead of trying to write down the solution in terms of $\hat{H}$, another approach is to assume the kinetic and potential operators can be treated as acting independently over several sub-steps if the time step is sufficiently small. For Strang splitting, the three substeps are illustrated schematically in Fig. 22.

The first sub-step consists of neglecting completely the kinetic operator, and solving under the action of the total potential operator only for a time step of $\Delta t/2$. The equation to be solved in this sub-step is then

$$\frac{\partial\psi(\mathbf{r},t)}{\partial t} = -\frac{i}{\hbar}\hat{V}\,\psi(\mathbf{r},t) \tag{357}$$

$$= -\frac{i}{\hbar}V(\mathbf{r})\,\psi(\mathbf{r},t), \tag{358}$$

where the operator has been replaced with a multiplication because the potential operator is diagonal in position basis. The solution of this equation is exact, and given by

$$\psi^{(1/3)}(\mathbf{r}) = \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\psi(\mathbf{r},t), \tag{359}$$

where, as in the previous section, $\psi^{(1/3)}$ is an intermediate wavefunction after the first sub-step which evolved the initial state at time t over a half time step of $\Delta t/2$ under the action of the potential only.

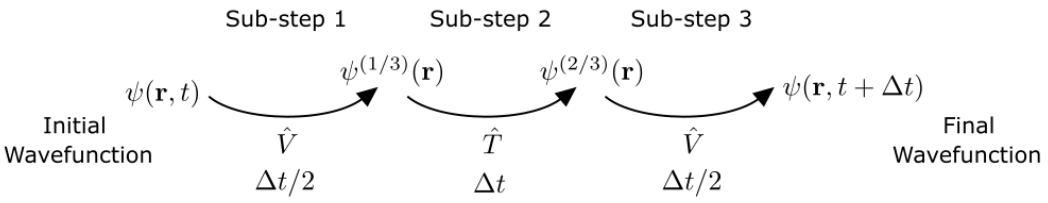

FIGURE 22: Illustration of the sub-steps involved in the split-step Fourier method with Strang splitting. To iterate the wavefunction $\psi(\mathbf{r}, t)$ by a time step Δt to the new wavefunction $\psi(\mathbf{r}, t + \Delta t)$, the process is split into three sub-steps. First, a half step of $\Delta t/2$ under the action of only the potential operator $\hat{V}$ provides an intermediate wavefunction $\psi^{(1/3)}$. Then a full step over Δt using only the kinetic operator $\hat{T}$ gives a new intermediate wavefunction $\psi^{(2/3)}$. Finally another half step with the potential operator gives the final wavefunction $\psi(\mathbf{r}, t + \Delta t)$.

The next sub-step consists of evolving this intermediate state under the action of the kinetic energy operator only (neglecting the potential operator), but this time over a full step of Δt. The equation therefore to evolve $\psi^{(1/3)}$ is

$$\frac{\partial\psi(\mathbf{r},t)}{\partial t} = -\frac{i}{\hbar}\hat{T}\,\psi(\mathbf{r},t) \tag{360}$$

$$= \frac{i\hbar}{2m}\nabla^2\psi(\mathbf{r},t) \tag{361}$$

$$= \frac{i\hbar}{2m}\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right). \tag{362}$$

This equation is solved most easily by taking both sides of the equation into Fourier space

$$\mathcal{F}\left\{\frac{\partial\psi(\mathbf{r},t)}{\partial t}\right\} = \mathcal{F}\left\{\frac{i\hbar}{2m}\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)\right\} \tag{363}$$

$$\frac{\partial\tilde{\psi}(\mathbf{k},t)}{\partial t} = \frac{i\hbar}{2m}\left(\mathcal{F}\left\{\frac{\partial^2\psi}{\partial x^2}\right\}+\mathcal{F}\left\{\frac{\partial^2\psi}{\partial x^2}\right\}+\mathcal{F}\left\{\frac{\partial^2\psi}{\partial x^2}\right\}\right) \tag{364}$$

$$= -\frac{i\hbar}{2m}\left(k_x^2+k_y^2+k_z^2\right)\tilde{\psi}(\mathbf{k},t) \tag{365}$$

$$= -\frac{i\hbar|\mathbf{k}|^2}{2m}\tilde{\psi}(\mathbf{k},t), \tag{366}$$

where I have used the expression for the spectral second derivative in each of the three directions - for example, in the x-coordinate

$$\frac{\partial^2\psi(x)}{\partial x^2} = \mathcal{F}^{-1}\Big\{-k_x^2\,\mathcal{F}\big\{\psi(x)\big\}\Big\} \tag{367}$$

$$= \mathcal{F}^{-1}\Big\{-k_x^2\,\tilde{\psi}(k_x)\Big\}. \tag{368}$$

Note here that the tilde notation indicates the function in k-space, and should not be confused with the tilde used in previous sections which denotes dimensionless quantities. Now we have a much simpler ordinary differential equation in k-space, which again can be solved exactly using $\psi^{(1/3)}(\mathbf{k}) = \mathcal{F}\{\psi^{(1/3)}(\mathbf{r})\}$ as an initial value

$$\psi^{(2/3)}(\mathbf{k}) = \exp\left(-\frac{i\hbar|\mathbf{k}|^2}{2m}\Delta t\right)\psi^{(1/3)}(\mathbf{k}). \tag{369}$$

The final sub-step is again under the action of the potential operator only, again with a half time step of $\Delta t/2$. The solution is the same as the first one, except that now we need to use as an initial value $\psi^{(2/3)}(\mathbf{r}) = \mathcal{F}^{-1}\{\psi^{(2/3)}(\mathbf{k})\}$, where this time we have to convert back from k-space to position space before using it as an initial condition, since the potential operator is only diagonal in momentum basis. The result of this final sub-step can be written as

$$\psi(\mathbf{r},t+\Delta t) = \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\psi^{(2/3)}(\mathbf{r}). \tag{370}$$

Finally, putting everything together we have

$$\psi(\mathbf{r},t+\Delta t) = \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\psi^{(2/3)}(\mathbf{r}) \tag{371}$$

$$= \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\mathcal{F}^{-1}\Big\{\psi^{(2/3)}(\mathbf{k})\Big\} \tag{372}$$

$$= \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\mathcal{F}^{-1}\left\{\exp\left(-\frac{i\hbar|\mathbf{k}|^2}{2m}\Delta t\right)\psi^{(1/3)}(\mathbf{k})\right\} \tag{373}$$

$$= \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\mathcal{F}^{-1}\left\{\exp\left(-\frac{i\hbar|\mathbf{k}|^2}{2m}\Delta t\right)\mathcal{F}\Big\{\psi^{(1/3)}(\mathbf{r})\Big\}\right\} \tag{374}$$

$$= \exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\mathcal{F}^{-1}\left\{\exp\left(-\frac{i\hbar|\mathbf{k}|^2}{2m}\Delta t\right)\mathcal{F}\left\{\exp\left(-\frac{i}{\hbar}V(\mathbf{r})\frac{\Delta t}{2}\right)\psi(\mathbf{r},t)\right\}\right\}. \tag{375}$$

This is identical to Eq. 353, and demonstrates that the Strang splitting split-step Fourier method can be viewed as a series of sub-steps alternating which operator action is neglected in each one.

Split-step Fourier operation in dimensionless variables:

When solving the equation numerically in practice, it is typical to work with the GPE in dimensionless units, given by Eq. 297, and reproduced here

$$i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)+\tilde{V}_{\text{ext}}\tilde{\psi}+\mathcal{G}|\tilde{\psi}|^2\tilde{\psi}, \tag{376}$$

where the tilde notation denotes that a physical variable has been scaled, with the scaling given in Eqs. (274) to (277). This can be wrtten in terms of scaled operators as above

$$\frac{\partial\tilde{\psi}(\tilde{\mathbf{r}},\tilde{t}\,)}{\partial\tilde{t}} = -i\,\tilde{\hat{H}}\,\tilde{\psi}(\tilde{\mathbf{r}},\tilde{t}\,) \tag{377}$$

$$= -i\left(\tilde{\hat{T}}+\tilde{\hat{V}}\right)\tilde{\psi}(\tilde{\mathbf{r}},\tilde{t}\,), \tag{378}$$

where the kinetic and (total) potential operators in dimensionless variables as given respectively by

$$\tilde{\hat{T}} = -\frac{1}{2}\nabla^2 \qquad \text{and} \qquad \tilde{\hat{V}} = \tilde{V}_{\text{ext}}(\tilde{\mathbf{r}})+\mathcal{G}|\tilde{\psi}(\tilde{\mathbf{r}},\tilde{t}\,)|^2. \tag{379}$$

Following the same procedure outlined in the previous section, we obtain the split-step algorithm in analogy with Eq. 353 but for the dimensionless version of the GPE

$$\boxed{\tilde{\psi}(\tilde{\mathbf{r}},\tilde{t}+\Delta\tilde{t}\,) = \exp\left(-\frac{i\Delta\tilde{t}}{2}\tilde{V}(\tilde{\mathbf{r}})\right)\mathcal{F}^{-1}\left\{\exp\left(-\frac{i\Delta\tilde{t}}{2}|\tilde{\mathbf{k}}|^2\right)\mathcal{F}\left\{\exp\left(-\frac{i\Delta\tilde{t}}{2}\tilde{V}(\tilde{\mathbf{r}})\right)\tilde{\psi}(\tilde{\mathbf{r}},\tilde{t}\,)\right\}\right\}.} \tag{380}$$

where the scaled square modulus wavevector $|\tilde{\mathbf{k}}|^2 = \tilde{k}_x^2+\tilde{k}_y^2+\tilde{k}_z^2$, and the scaled wavevectors along each dimension are

$$\tilde{k}_x = \frac{2\pi}{\tilde{x}} = \frac{2\pi}{x/l} = l\,k_x \tag{381}$$

$$\tilde{k}_y = \frac{2\pi}{\tilde{y}} = \frac{2\pi}{y/l} = l\, k_y \tag{382}$$

$$\tilde{k}_z = \frac{2\pi}{\tilde{z}} = \frac{2\pi}{z/l} = l\, k_z. \tag{383}$$

Matlab Code to Perform Split-Step Fourier Iteration:

The split-step operation in Eq. 380 is written in continuous space variables, and the forward and inverse Fourier transforms $\mathcal{F}$ and $\mathcal{F}^{-1}$ are continuous, i.e. they are defined as integrals over all space. In practice, when solving the equation numerically the problem would be discretised, and the space / momentum variables would be evaluated on a grid. When this is done, the integrals in the Fourier transforms can be approximated by a Riemann summation, which allows to make use of the forward and inverse fast Fourier transform (FFT) algorithms. Details will not be given here, because it is quite involved to demonstrate rigorously the transition between continuous and discrete variables, and more details can be found at the link in Ref. [29]. The main result is that the 3D continuous Fourier transform can be approximated using FFT in Matlab with

$$\psi_2(\mathbf{k}) = \mathcal{F}\Big\{\psi_1(\mathbf{r})\Big\} = \iiint_{-\infty}^{+\infty} \psi_1 \exp\Big(-i\left[k_x x + k_y y + k_z z\right]\Big)\, \mathrm{d}k_x\, \mathrm{d}k_y\, \mathrm{d}k_z \tag{384}$$

$$= \iiint_{-\infty}^{+\infty} \psi_1 \exp\Big(-2\pi i\left[\nu_x x + \nu_y y + \nu_z z\right]\Big)\, (2\pi)^3\, \mathrm{d}\nu_x\, \mathrm{d}\nu_y\, \mathrm{d}\nu_z \tag{385}$$

$$\approx (2\pi)^3\, \Delta\nu_x \Delta\nu_y \Delta\nu_z \sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty} \psi_1 \exp\Big(-2\pi i\left[\nu_x x + \nu_y y + \nu_z z\right]\Big) \tag{386}$$

$$= (2\pi)^3\, \Delta\nu_x \Delta\nu_y \Delta\nu_z \times \texttt{fftn}\left[\psi_1\right]. \tag{387}$$

This is not a rigorous derivation, as continuous and discrete variables have been used loosely, but it demonstrates the main steps that are involved. Specifically, the continuous Fourier transform is approximated by a Riemann sum, which can be conveniently and efficiently evaluated with a n-dimensional FFT (see the Mathworks documentation page for fftn to see exactly the summation that is carried out by fftn [30]). To allow direct use of fftn, a change of variables was made from angular spatial frequency k_i to linear spatial frequency ν_i along each direction, which results in a factor of $(2\pi)^3$ since $\mathrm{d}k_i = 2\pi\, \mathrm{d}\nu_i$. Finally, as a result of the Riemann sum approximation, the differentials $\mathrm{d}\nu_i$ are replaced by the frequency steps $\Delta\nu_i$, which are not included in the fftn operation, so must be manually left in the expression.

For carrying out an inverse transform numerically, a similar procedure holds

$$\psi_3(\mathbf{r}) = \mathcal{F}^{-1}\Big\{\psi_2(\mathbf{k})\Big\} = \frac{1}{(2\pi)^3} \iiint_{-\infty}^{+\infty} \psi_2 \exp\Big(i\left[k_x x + k_y y + k_z z\right]\Big)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z \tag{388}$$

$$= \frac{1}{(2\pi)^3} \iiint_{-\infty}^{+\infty} \psi_2 \exp\Big(2\pi i\left[\nu_x x + \nu_y y + \nu_z z\right]\Big)\, \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}z \tag{389}$$

$$\approx \frac{1}{(2\pi)^3}\,\Delta x\Delta y\Delta z \sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty} \psi_2 \exp\Big(2\pi i\left[\nu_x x + \nu_y y + \nu_z z\right]\Big) \tag{390}$$

$$= \frac{1}{(2\pi)^3}\,\Delta x\Delta y\Delta z\, N_x N_y N_z \times \texttt{ifftn}\left[\psi_2\right]. \tag{391}$$

The extra factors of N_x, N_y, N_z (the number of points in each direction of the discretised wavefunction) appear since in Matlab's definition of `ifftn` there are already included these factors in the function definition (see the help page), and so the triple summation that we require is actually carried out by $N_x N_y N_z \times \texttt{ifftn}\left[...\right]$. Note that this is true for Matlab, but other languages have their own definitions of the FFT - some include the factors of N, some include $\sqrt{N}$, and some have no extra factors - always check the documentation for the algorithm you are using. When working with discrete space, the frequency spacing (distance between points in frequency space) is given by $\Delta\nu_i = 1/(\Delta x_i N_i)$, and so the factors of N_i cancel and the expression becomes

$$\psi_3(\mathbf{r}) = \mathcal{F}^{-1}\Big\{\psi_2(\mathbf{k})\Big\} \approx = \frac{1}{(2\pi)^3}\,\frac{1}{\Delta\nu_x\Delta\nu_y\Delta\nu_z} \times \texttt{ifftn}\left[\psi_2\right]. \tag{392}$$

Now we come to an important point when approximating continuous forward and backward Fourier transforms in Matlab. We have seen that to approximate the transforms alone we need additional factors of 2π and $\Delta\nu$ in order to get the correct scaling of the results with

$$\text{Forward transform:} \quad \psi_2(\mathbf{k}) = \mathcal{F}\Big\{\psi_1(\mathbf{r})\Big\} \approx (2\pi)^3\,\Delta\nu_x\Delta\nu_y\Delta\nu_z \times \texttt{fftn}\left[\psi_1\right] \tag{393}$$

$$\text{Inverse transform:} \quad \psi_3(\mathbf{r}) = \mathcal{F}^{-1}\Big\{\psi_2(\mathbf{k})\Big\} \approx \frac{1}{(2\pi)^3}\,\frac{1}{\Delta\nu_x\Delta\nu_y\Delta\nu_z} \times \texttt{ifftn}\left[\psi_2\right]. \tag{394}$$

However, very often we will need to perform a forward transform, then do some multiplication with a factor or other manipulation, and finally perform in inverse transform. This is the case for the split-step operation in Eq. 380. In this case, the factors conveniently cancel, and need not be included in the code, such that

$$\psi_3(\mathbf{r}) = \mathcal{F}^{-1}\Big\{\psi_2(\mathbf{k})\Big\} \tag{395}$$

$$= \mathcal{F}^{-1}\Big\{\mathcal{F}\big\{\psi_1(\mathbf{r})\big\}\Big\} \tag{396}$$

$$\approx \texttt{ifftn}\left[\texttt{fftn}\left[\psi_1\right]\right]. \tag{397}$$

This is the reason you will often see FFT used in code without any additional scaling factors, and it is safe to do so as long as the transforms are always performed in forward/backward pairs. If you only want to do a forward transform, for example to study a wavefunction in frequency space, then the scaling factors are required.

The following snippet of Matlab code is what would be required to numerically evolve a three-dimensional matrix `psi` representing a wavefunction using the split-step Fourier method with Strang splitting given by Eq. 380 in dimensionless variables

```
psi = psi_initial;  % Initialise wavefunction

%%% Iterate wavefunction over time steps dt
for k = 2:Nt

    %%% Split-Step Spectral Evolution (using Strang Splitting)

    % Half step with potential operator
    psi = exp( -1i*dt/2*(  V_ext + G*abs(psi).^2  ) ).*psi;

     % Bring wavefunction into k-space
    psi_k = fftshift( fftn( ifftshift(psi) ) );

    % Full step with kinetic operator
    psi_k = exp( -1i*dt/2 * (KX.^2 + KY.^2 + KZ.^2) ).*psi_k;

    % Bring wavefunction back to real space
    psi = fftshift( ifftn( ifftshift(psi_k) ) );

    % Half step with potential operator
    psi = exp( -1i*dt/2*(  V_ext + G*abs(psi).^2  ) ).*psi;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

Note that the functions `fftshift` and `ifftshift` must be used in Matlab, to swap the halves of the matrix along each dimension and ensure everything is correctly ordered before multiplying the wavefunction in k-space by the kinetic operator. This is essentially because the wavefunction is defined with its zero of coordinates at the centre of the matrix, whereas `fft` expects it to be at the edges. If this is not done, additional phase shifts are introduced into the result of the transform. See Sec. 2.6 and Ref. [29] for more information on why this is needed.

7.5 Imaginary Time Method for Finding Ground States

The split-step method is suitable for calculating the dynamics of a wavefunction according to the Gross-Pitaevskii equation (GPE), given some initial condition. However, we need to now choose the initial condition to start the iterative process of solving the equation. Typically, this would consist of starting the system in the ground state in some external trapping potential. Although we can find analytic solutions to the 3D GPE in certain limits (e.g. a parabola when kinetic energy is neglected, or Gaussian when interactions are neglected), there are no general analytic solutions for the full equation. Any wavefunction which is not truly the ground state will experience some excitations or oscillations - even if it remains in the same potential - an the algorithms can be quite sensitive to initial conditions. Therefore, before running the time-domain solver to extract the dynamics, we must first numerically calculate the ground state - the imaginary time method provides an excellent way to do this, gives a good starting point before running the real time dynamics.

To see how the imaginary time method works, we go back to the general exact solution from

Eq. 335, which describes how the wavefunction will evolve under the action of a given Hamiltonian

$$\psi(\mathbf{r}, t + \Delta t) = \exp\left(-\frac{i}{\hbar}\hat{H}\Delta t\right)\psi(\mathbf{r}, t). \tag{398}$$

The wavefunction can generally be expressed as a superposition of many eigenstates $\phi_m(\mathbf{r})$ which each have their own associated eigenenergies $E_m(t)$ and amplitudes $a_m(t)$ that can both be time-dependent. The wavefunction can then be written in the form

$$\psi(\mathbf{r}, t) = a_0(t)\phi_0(\mathbf{r}) + a_1(t)\phi_1(\mathbf{r}) + a_2(t)\phi_2(\mathbf{r}) + ... \tag{399}$$

$$= \sum_m a_m(t)\phi_m(\mathbf{r}). \tag{400}$$

Inserting this into Eq. 398 gives

$$\psi(\mathbf{r}, t + \Delta t) = \exp\left(-\frac{i}{\hbar}\hat{H}\Delta t\right)\sum_m a_m(t)\phi_m(\mathbf{r}). \tag{401}$$

Now, the imaginary time method consists of making a substitution for the time variable, by defining an "imaginary" version of the time coordinate τ, such that $t \to -i\tau$, and therefore $\Delta t \to -i\Delta\tau$. In imaginary time, Eq. 401 evolves as

$$\psi(\mathbf{r}, \tau + \Delta\tau) = \exp\left(-\frac{1}{\hbar}\hat{H}\Delta\tau\right)\sum_m a_m(\tau)\phi_m(\mathbf{r}). \tag{402}$$

Now we know that the functions ϕ_m are eigenstates of the Hamiltonian (because we defined them as such), and therefore when acted on by the Hamiltonian operator the result is to simply pick out the corresponding energy eigenvalue E_m associated with each eigenfunction, such that

$$\hat{H}\phi_m(\mathbf{r}) = E_m\phi_m(\mathbf{r}), \tag{403}$$

which leads to

$$\begin{aligned}\psi(\mathbf{r}, \tau + \Delta\tau) &= \sum_m a_m(\tau)\exp\left(-\frac{1}{\hbar}E_m\Delta\tau\right)\phi_m(\mathbf{r}) \\ &= a_0(\tau)\exp\left(-\frac{E_0\Delta\tau}{\hbar}\right)\phi_0(\mathbf{r}) + a_1(\tau)\exp\left(-\frac{E_1\Delta\tau}{\hbar}\right)\phi_1(\mathbf{r}) \\ &\quad + a_2(\tau)\exp\left(-\frac{E_2\Delta\tau}{\hbar}\right)\phi_2(\mathbf{r}) + ...\end{aligned}$$

Looking closely at this equation, we can see that, in imaginary time, the amplitude of each eigenstate decays exponentially over time. However, the rate of decay ("time constant" of the exponential) of the different amplitudes is different for each energy E_m, and that the state with the *smallest* value of E_m (i.e the ground state) will decay the slowest, as indicated in Fig. 23.

Therefore, if we numerically solve the system over imaginary time, all of the higher order states will be extinguished one by one, until the total energy settles to a minimum and the system reaches

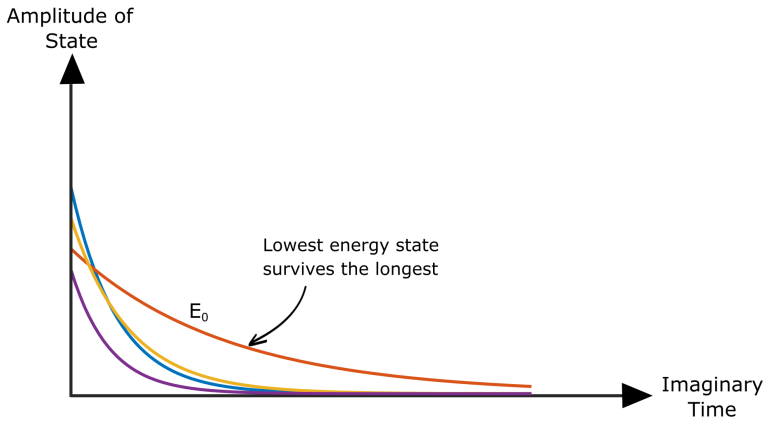

FIGURE 23: Illustration of the exponential decay of the amplitudes of each eigenstate when a system is evolved forwards in imaginary time. The eigenstate with the lowest energy (ground state) will have the slowest decay constant, and will persist for the longest, with higher excited states beign attenuated.

the ground state. Since the higher states amplitudes are being attenuated, the wavefunction will lose its normalisation gradually more and more at each time step, so to ensure that the number of atoms is conserved at all times we must make sure to *renormalise* the wavefunction at each iteration (note that this is not necessary when running the real time algorithm, since then the wavefunction should conserve atom number if a suitable stepping scheme is chosen). Conveniently, the same algorithm (whatever it may be that is chosen) that is used for the forward time dynamics can be used for the imaginary time ground state finding, except for

1. we must replace $t \to -i\tau$ in the equation
2. we must renormalise the wavefunction at each step

By making the change to imaginary time, the GPE is essentially transformed into a "reaction-diffusion" equation, which can be seen by considering that when switching to imaginary time the original equation becomes

$$i\hbar \frac{\partial \psi(\mathbf{r},t)}{\partial t} = \hat{H}\psi(\mathbf{r},t) \tag{404}$$

$$\overset{t\to -i\tau}{\Longrightarrow} \frac{\partial \psi(\mathbf{r},\tau)}{\partial \tau} = -\frac{1}{\hbar}\hat{H}\psi(\mathbf{r},\tau), \tag{405}$$

where the factor of i has cancelled because

$$\frac{\partial \psi}{\partial t} = \frac{\partial \psi}{\partial \tau}\frac{\partial \tau}{\partial t} \tag{406}$$

$$= i\frac{\partial \psi}{\partial \tau}. \tag{407}$$

Reaction-diffusion equations typically reach their steady-state solution when the system settles into a state with the lowest energy. This reflects the fact that over time the BEC "diffuses" into the lowest

energy configuration given by the combined external and mean-field interaction potential. The imaginary time technique for finding ground states was originally used on the Schrödinger equation [31], and then later applied to the non-linear Gross-Pitaevskii equation [5, 32]. It was also shown in Ref. [33] that the imaginary time method is actually equivalent to the more direct method of minimising an energy functional to find the lowest energy state [8].

7.6 Example: Numerical Ground State in the Non-Interacting Case

As a first test, I will consider the case of a non-interacting system. The dimensionless GPE from Eq. 307 is to be solved, reproduced here for convenience

$$i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right) + \frac{1}{2}\left(\tilde{x}^2 + \kappa^2\tilde{y}^2 + \lambda^2\tilde{z}^2\right)\tilde{\psi} + \mathcal{G}|\tilde{\psi}|^2\tilde{\psi}, \tag{408}$$

and with the non-linearity constant $\mathcal{G} = 0$ to remove the effect of interactions. The potential used is the anisotropic harmonic oscillator with trapping frequencies $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz, leading to dimensionless anisotropy parameters $\kappa = \omega_y/\omega_x = 2$ and $\lambda = \omega_z/\omega_x = 0.4$. An initial Gaussian guess is used to begin the calculation, with RMS widths along each direction around twice those of the expected ground state, to provide something reasonably close to the final solution. We know that when the non-linear term is dropped in the GPE, then the solution should just be that of the regular Schrodinger equation, given by

$$\psi(\mathbf{r}) = \sqrt{\frac{1}{(\pi)^{3/2}\,a_x a_y a_z}}\exp\left(-\frac{x^2}{2a_x^2} - \frac{y^2}{2a_y^2} - \frac{z^2}{2a_z^2}\right) \tag{409}$$

where $a_i = \sqrt{\hbar/m\omega_i}$ are the harmonic oscillator lengths in each of the three directions. The density distribution is then given by

$$n(\mathbf{r}) = N|\psi(\mathbf{r})|^2 = \frac{N}{(\pi)^{3/2}\,a_x a_y a_z}\exp\left(-\frac{x^2}{a_x^2} - \frac{y^2}{a_y^2} - \frac{z^2}{a_z^2}\right). \tag{410}$$

The split-step Fourier algorithm from Eq. 380 is then used to evolve the initial guess, but having made the change into dimensionless imaginary time $\tilde{\tau} = \omega_x\tau = i\omega_x t$ because we are trying to find the ground state (which essentially amounts to removing the factors of i from Eq. 380)

$$\tilde{\psi}(\tilde{\mathbf{r}}, \tilde{\tau} + \Delta\tilde{\tau}\,) = \exp\left(-\frac{\Delta\tilde{\tau}}{2}\tilde{V}(\tilde{\mathbf{r}})\right)\mathcal{F}^{-1}\left\{\exp\left(-\frac{\Delta\tilde{\tau}}{2}|\tilde{\mathbf{k}}|^2\right)\mathcal{F}\left\{\exp\left(-\frac{\Delta\tilde{\tau}}{2}\tilde{V}(\tilde{\mathbf{r}})\right)\tilde{\psi}(\tilde{\mathbf{r}}, \tilde{\tau}\,)\right\}\right\}. \tag{411}$$

The dimensionless wavefunction returned by the split-step method $\tilde{\psi}$ is related to the true physical wavefunction by $\psi = \tilde{\psi}/\sqrt{l^3}$ and therefore the absolute squared is related by $|\tilde{\psi}|^2 = l^3|\psi|^2$. This quantity is a 3D matrix and line cuts along each direction are plotted in Fig. 24 as the system evolves in imaginary time, where it can be seen that the initial guess eventually tends towards matching nicely with the expected analytic Gaussian result.

The kinetic, potential, and interaction energies per particle are also calculated numerically using

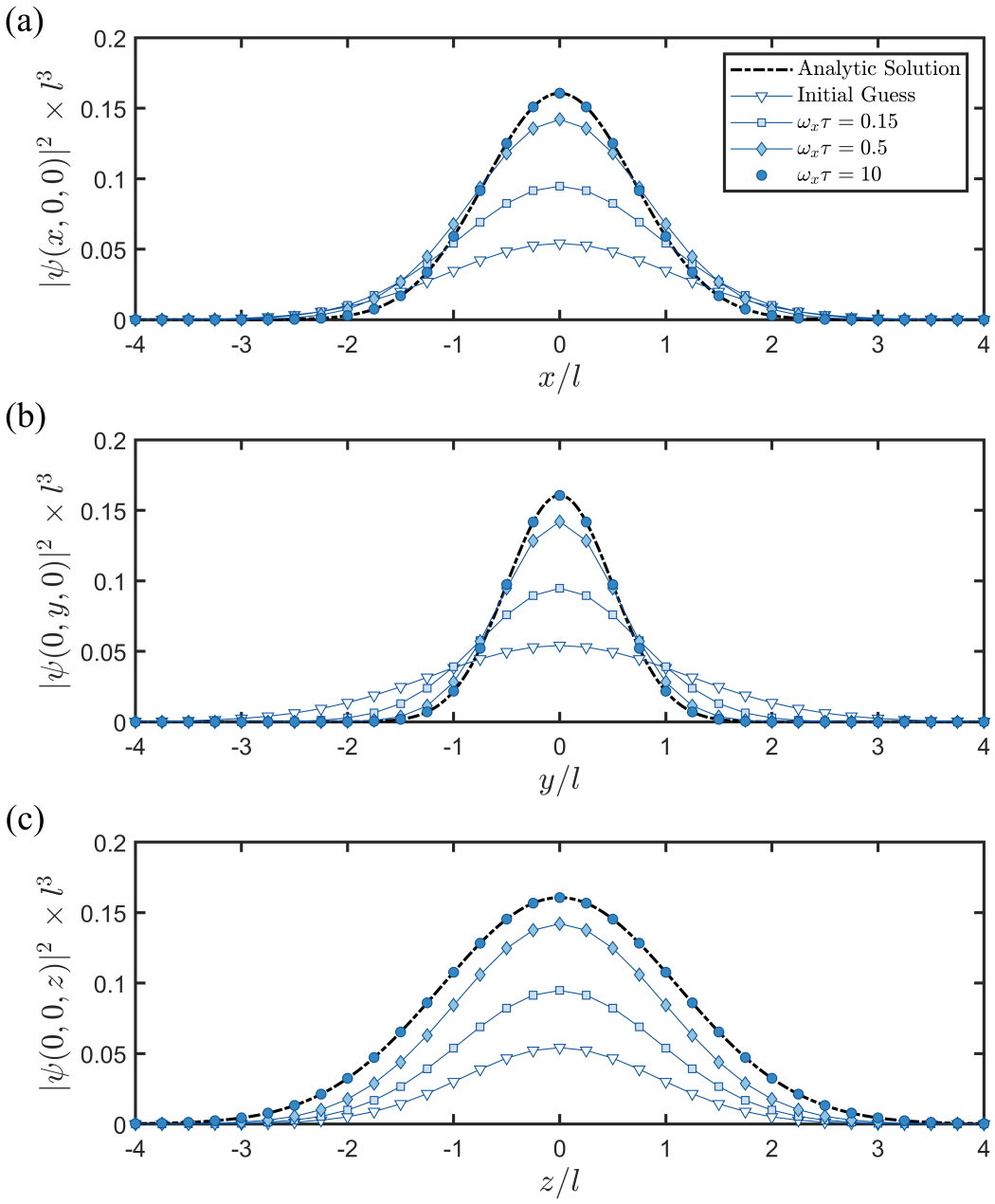

FIGURE 24: Finding the ground state using the imaginary time split-step Fourier method in the non-interacting case ($\mathcal{G} = 0$). Line cuts through the square modulus of the 3D wavefunction along the x, y, and z directions in (a), (b), and (c), respectively. The harmonic trapping frequencies were set to be $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz. As the system evolves forwards in imaginary time τ, the initial guess gradually settles to the ground state and eventually agrees well with the expected analytic Gaussian result for late times. The calculation used $N_\tau = 2000$ imaginary time iterations with a time step of $\Delta\tilde{\tau} = 0.005$. Spatial grid resolutions were $\Delta\tilde{x} = \Delta\tilde{y} = \Delta\tilde{z} = 0.25$ with $N_x \times N_y \times N_z = 64 \times 64 \times 64$ points.

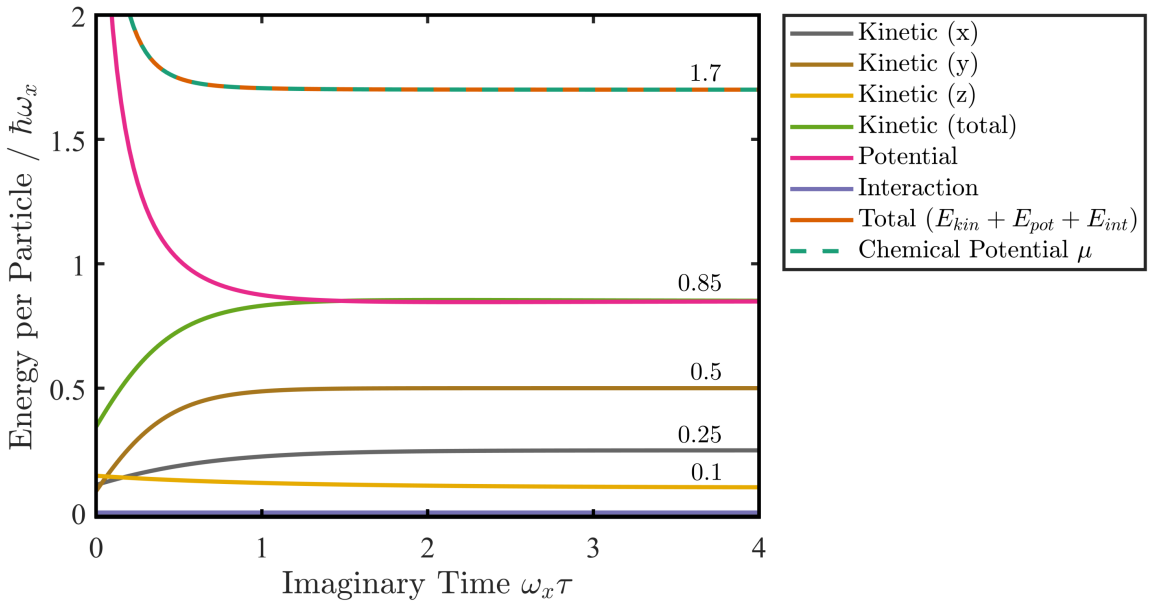

FIGURE 25: Evolution of the energy components per particle (in units of the harmonic oscillator spacing along the x-direction) as the system is propagated in dimensionless imaginary time, showing that the lowest energy state (ground state) is eventually reached. The harmonic trapping frequencies were set to be $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz, leading to the observed unequal distribution of kinetic energies. Due to the absence of interactions, the chemical potential is equal to the mean total energy per particle, which is split equally between the kinetic and potential contributions.

Eqs. (313), (319) and (325), respectively, as well as the chemical potential using Eq. 329. These energy components are displayed in Fig. 25 as the system evolved through imaginary time. It can be seen that the system settles into the lowest total energy state, therefore finding the ground state as the imaginary time method is expected to do. For the non-interacting case, the chemical potential should be equal to the total mean energy per particle, and this energy should be divided equally between the kinetic and potential contributions - both of which can be seen in the numerical result.

In the non-interacting case, it is expected that the energy per particle for a harmonically trapped system is given by

$$\varepsilon_{x,y,z} = \left(n_x + \frac{1}{2}\right)\hbar\omega_x + \left(n_y + \frac{1}{2}\right)\hbar\omega_y + \left(n_z + \frac{1}{2}\right)\hbar\omega_z, \tag{412}$$

where n_i are the quantum numbers of the energy level occupied in each direction. For the ground state, with the lowest level occupied, this becomes

$$\varepsilon_{000} = \frac{1}{2}\hbar\left(\omega_x + \omega_y + \omega_z\right). \tag{413}$$

Since the numerical calculation is carried out in dimensionless units of $\hbar\omega_x$, this becomes

$$\varepsilon_{000} = \frac{1}{2}\left(1 + \frac{\omega_y}{\omega_x} + \frac{\omega_z}{\omega_x}\right), \tag{414}$$

and with the specific trap frequencies of $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz we expect a result of

$$\varepsilon_{000} = \frac{1}{2}\left(1 + \frac{100}{50} + \frac{20}{50}\right) \tag{415}$$

$$= 0.5 + 1 + 0.2. \tag{416}$$

This should be divided equally between the kinetic and potential terms, so a result of $\left(\tilde{E}^{(x)}_{\text{kin}}, \tilde{E}^{(y)}_{\text{kin}}, \tilde{E}^{(z)}_{\text{kin}}\right) = (0.25, 0.5, 0.1)$ is expected - which can be seen correctly in Fig. 25. The Matlab code to calculate this ground state of a non-interacting system in an anisotropic harmonic trap using the imaginary time method is given in Appendix A.

7.7 Example: Numerical Ground State in the Interaction-Dominated Case

Another important case to test the code is the interaction-dominated limit of the dimensionless GPE from Eq. 307 given by

$$i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{x}^2} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{y}^2} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right) + \frac{1}{2}\left(\tilde{x}^2 + \kappa^2\tilde{y}^2 + \lambda^2\tilde{z}^2\right)\tilde{\psi} + \mathcal{G}|\tilde{\psi}|^2\tilde{\psi}, \tag{417}$$

with the non-linear parameter from Eq. 298 now included as

$$\mathcal{G} = \frac{4\pi a_s N}{l}. \tag{418}$$

To examine the interaction-dominated case, we can use the same external trapping frequencies from the non-interacting example in Sec. 7.6 given as $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz, i.e. $\kappa = \omega_y/\omega_x = 2$ and $\lambda = \omega_z/\omega_x = 0.4$, except now with a large atom number $N = 10^7$. This results in a dimensionless non-linearity of $\mathcal{G} = 430,000$. With such a large non-linearity, we expect that the system is in the Thomas-Fermi regime (i.e. interactions dominate over kinetic energy), and the density is simply the inverse of the external trapping potential. Since we are using a harmonic potential, the density is expected to be an inverted parabola given by Eq. 220 (with the absolute square of the wavefunction normalised to unity, since the atom number is included in the non-linearity)

$$n(\mathbf{r}) = N|\psi(\mathbf{r})|^2 = \frac{\mu_{3D}}{g_{3D}}\left(1 - \frac{x^2}{R_x^2} - \frac{y^2}{R_y^2} - \frac{z^2}{R_z^2}\right), \tag{419}$$

where $R_i = \sqrt{2\mu_{3D}/\left(m\omega_i^2\right)}$, and the analytic result for the chemical potential in the 3D Thomas-Fermi limit is taken from Eq. 226

$$\mu_{3D} = \left(\frac{15\sqrt{2}}{32\pi} N g_{3D}\, m^{3/2}\omega_x\,\omega_y\,\omega_z\right)^{2/5}. \tag{420}$$

The properties of the solution in both SI and dimensionless variables for the trap frequencies and atom number listed above are therefore expected to be

$$\mu_{3D} = 2.920 \times 10^{-30}\ \text{J} \implies \tilde{\mu}_{3D} = \frac{\mu_{3D}}{\hbar\omega_x} = 88.09 \tag{421}$$

$$R_x = 20.3\,\mu\text{m} \implies \tilde{R}_x = \frac{R_x}{l} = 13.3 \tag{422}$$

$$R_y = 10.1\,\mu\text{m} \implies \tilde{R}_y = \frac{R_y}{l} = 6.64 \tag{423}$$

$$R_z = 50.6\,\mu\text{m} \implies \tilde{R}_z = \frac{R_z}{l} = 33.2 \tag{424}$$

After evolving the GPE in imaginary time using the split-step Fourier algorithm from Eq. 380 (without the factors of i to ensure imaginary time), the system eventually converged to the ground state. Snapshots throughout this evolution are shown in Fig. 26, where the final solution is seen to correctly be an inverted parabola with the expected Thomas-Fermi radii along each direction.

The kinetic, potential, and interaction energies per particle are also calculated numerically using

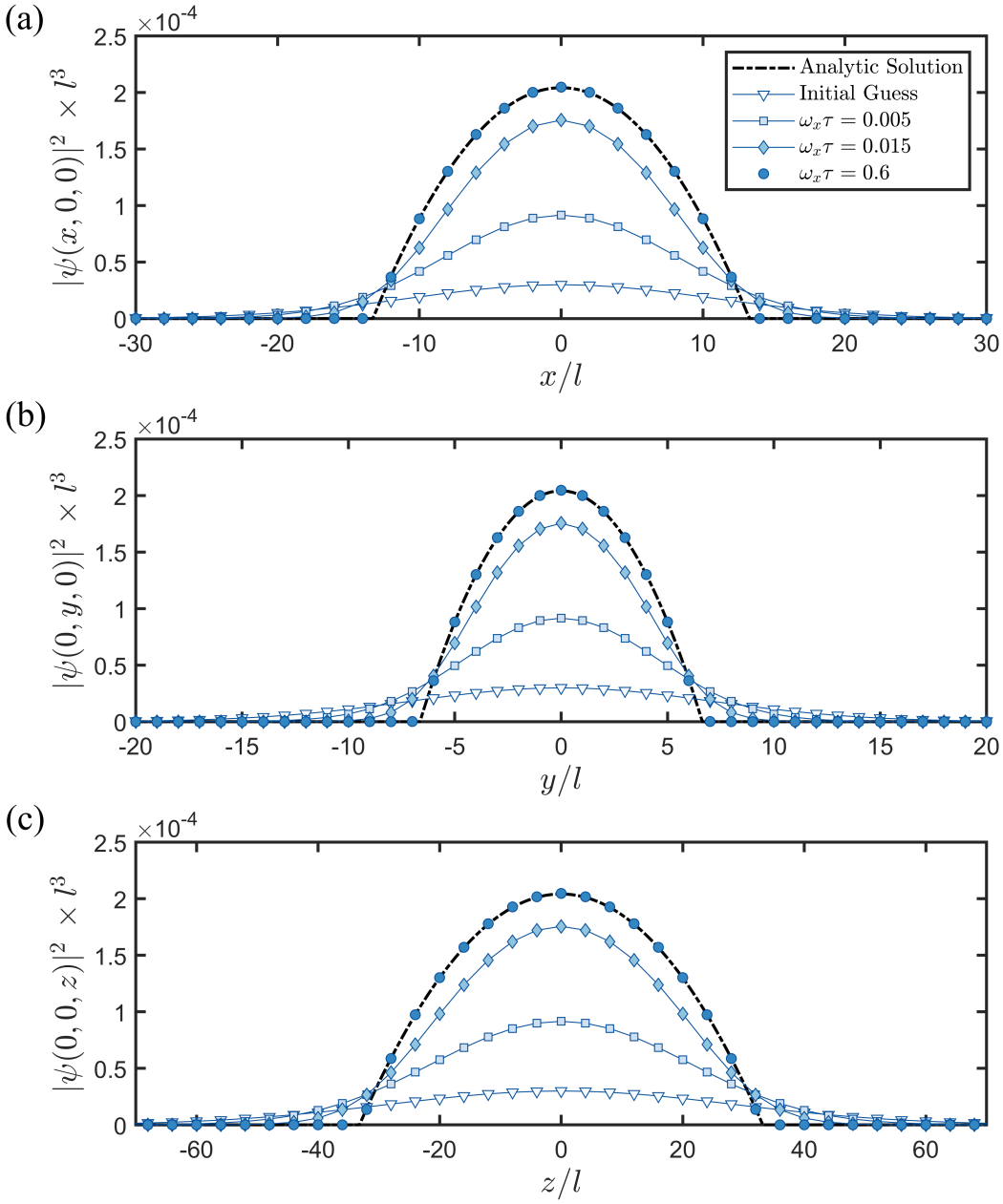

FIGURE 26: Finding the ground state using the imaginary time split-step Fourier method in the interaction-dominated case ($\mathcal{G} \neq 0$). Line cuts through the square modulus of the 3D wavefunction along the x, y, and z directions in (a), (b), and (c), respectively. The harmonic trapping frequencies were set to be $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz, with an atom number of $N = 10^7$. As the system evolves forwards in imaginary time τ, the initial guess (a Gaussian) gradually settles to the ground state and eventually agrees well with the expected analytic Thomas-Fermi inverted parabola for late times. The calculation used $N_\tau = 6000$ imaginary time iterations with a time step of $\Delta\tilde{\tau} = 0.0001$. Spatial grid resolutions were $\Delta\tilde{x} = 2$, $\Delta\tilde{y} = 1$, $\Delta\tilde{z} = 4$ with $N_x \times N_y \times N_z = 64 \times 64 \times 64$ points.

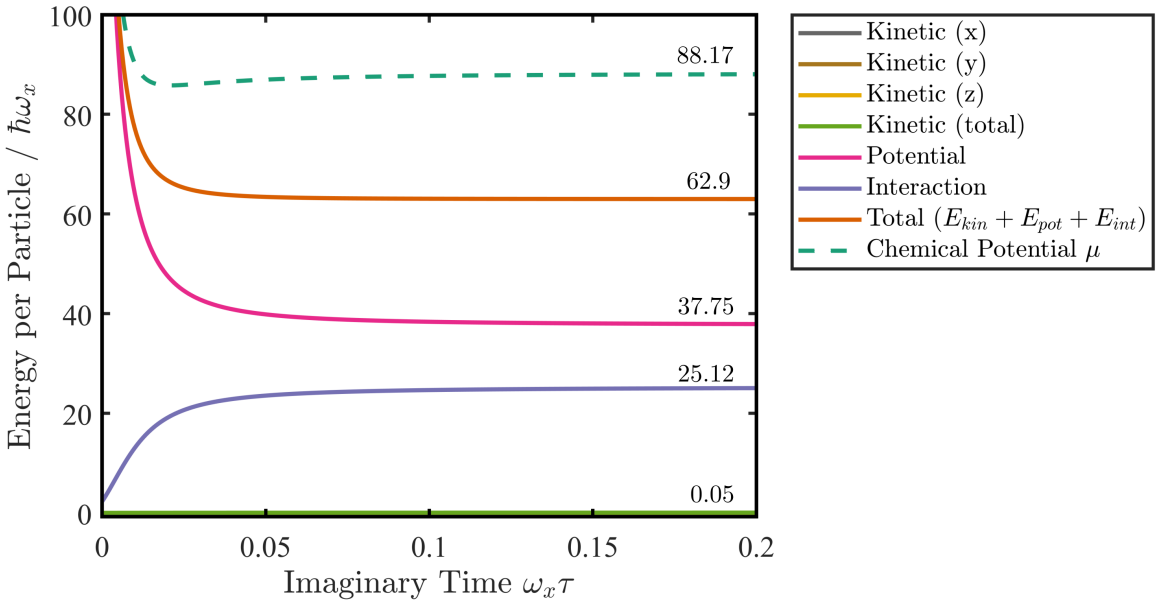

FIGURE 27: Evolution of the energy components per particle (in units of the harmonic oscillator spacing along the x-direction) for a cloud in the interaction-dominated (Thomas-Fermi) regime as the system is propagated in dimensionless imaginary time, showing that the lowest energy state (ground state) is eventually reached. The harmonic trapping frequencies were set to be $(\omega_x, \omega_y, \omega_z) = 2\pi \times (50, 100, 20)$ Hz with an atom number of $N = 10^7$, leading to the observed distribution of energies. All components of the kinetic energy are negligible (lines overlapping), and the ratio of interaction to potential contributions is $2/3$, as set by the virial theorem. The chemical potential is greater than the mean total energy per particle due to the presence of interactions, and agrees well with the expected 3D Thomas-Fermi analytic result.

Eqs. (313), (319) and (325), respectively, as well as the chemical potential using Eq. 329. The evolution of the energies through imaginary time are shown in Fig. 27, where it can be seen that the system indeed evolves to the lowest energy state. In addition, the numerical result for the chemical potential agrees well with the expected analytic result (the small discrepancy is because the analytic result neglects the kinetic energy completely, whereas the numerical result does not), and that the kinetic energy is almost negligible as expected for a cloud in the Thomas-Fermi regime. It can be shown [9] that the virial theorem for the GPE leads to the energies being distributed as

$$2E_{\text{kin}} - 2E_{\text{pot}} + 3E_{\text{int}} = 0, \tag{425}$$

and so in the Thomas-Fermi regime we expect the ratio of interaction to potential energies to be $E_{\text{int}}/E_{\text{pot}} = 2/3$, as indeed is observed in Fig. 27. Finally, since we know that the chemical potential is given by Eq. 271, it can be expressed in terms of the interaction energy $\mu = E_{\text{pot}} + 2E_{\text{int}} = (7/2)E_{\text{int}}$. In addition, in the Thomas-Fermi limit the relationship $E_{\text{tot}} = (5/7)\mu$ holds [9], and so in the interacting case the chemical potential is greater than the total mean energy per particle (in contrast to the non-interacting case, where the two coincide). These relationships between the energies are all correctly seen in Fig. 27.

7.8 Example: Numerical Ground State in the 1D-3D Crossover (Gerbier Profile)

TO DO

7.9 Example: Collective Oscillations

The previous sections have shown how to calculate the ground state of a system numerically, by evolving in imaginary time. This section will give an example of a real time simulation, to show how a system would dynamically evolve in practice. In all cases, the imaginary time method is first used to find the ground state, and then this wavefunction will be used as the initial condition for a real time evolution. The split-step Fourier algorithm for dimensionless variables in Eq. 380 is used (with the factors of i included now) in order to do the real time evolution.

Dipole Oscillation

One of the simplest examples to demonstrate real time evolution is that of a cloud of atoms oscillating back and forth in a harmonic trap, known as the *dipole* mode of oscillation. For this example, the ground state for first found for a system in an isotropic (spherical) harmonic potential with trap frequencies $\omega_x = \omega_y = \omega_z = \omega_{\text{ho}} = 2\pi \times 100\,\text{Hz}$ and an atom number of $N = 10^5$. This initial potential for finding the ground state was displaced along the $\tilde{z}$-direction by $\tilde{z} = 3$, after which the real time evolution was carried out immediately in a trap centered at the origin. Therefore, the system sees a sudden displacement of the trapping potential which excites a dipole centre of mass oscillation along the $\tilde{z}$-direction, shown schematically in Fig. 28.

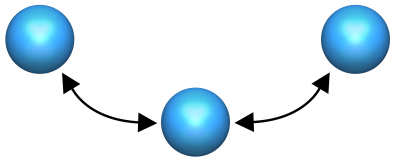

FIGURE 28: Schematic of a dipole mode excitation (centre of mass oscillation) in a trap.

Figure 29 shows the dipole oscillation in the absolute square of the wavefunction (proportional to the density), which appears only along the $\tilde{z}$-direction, with no detectable oscillation along the remaining two dimensions. The time period for an oscillation in dimensionless time units can be extracted from this figure to be $\tilde{T} = 6.28$, which in SI units is $T = \tilde{T}/\omega_x = 10\,\text{ms}$. The oscillation frequency therefore is the inverse of this at $100\,\text{Hz}$, which matches the harmonic trap frequency, as expected for a dipole oscillation.

The various energy components per particle are also calculated numerically using Eqs. (313), (319) and (325), respectively, as well as the chemical potential using Eq. 329, and their evolution over time is shown in Fig. 30. As expected, the energy oscillates back and forth between kinetic and potential, i.e. when the cloud reaches an extremum of the oscillation, the potential energy is maximum but the kinetic energy is minimum since the direction of travel reverses and therefore the velocity is zero. Note that the oscillation in energy proceeds at twice the rate of the dipole oscillation, since there are two extrema per cycle. Also interesting to note from Fig. 30 is that the total kinetic energy is contained

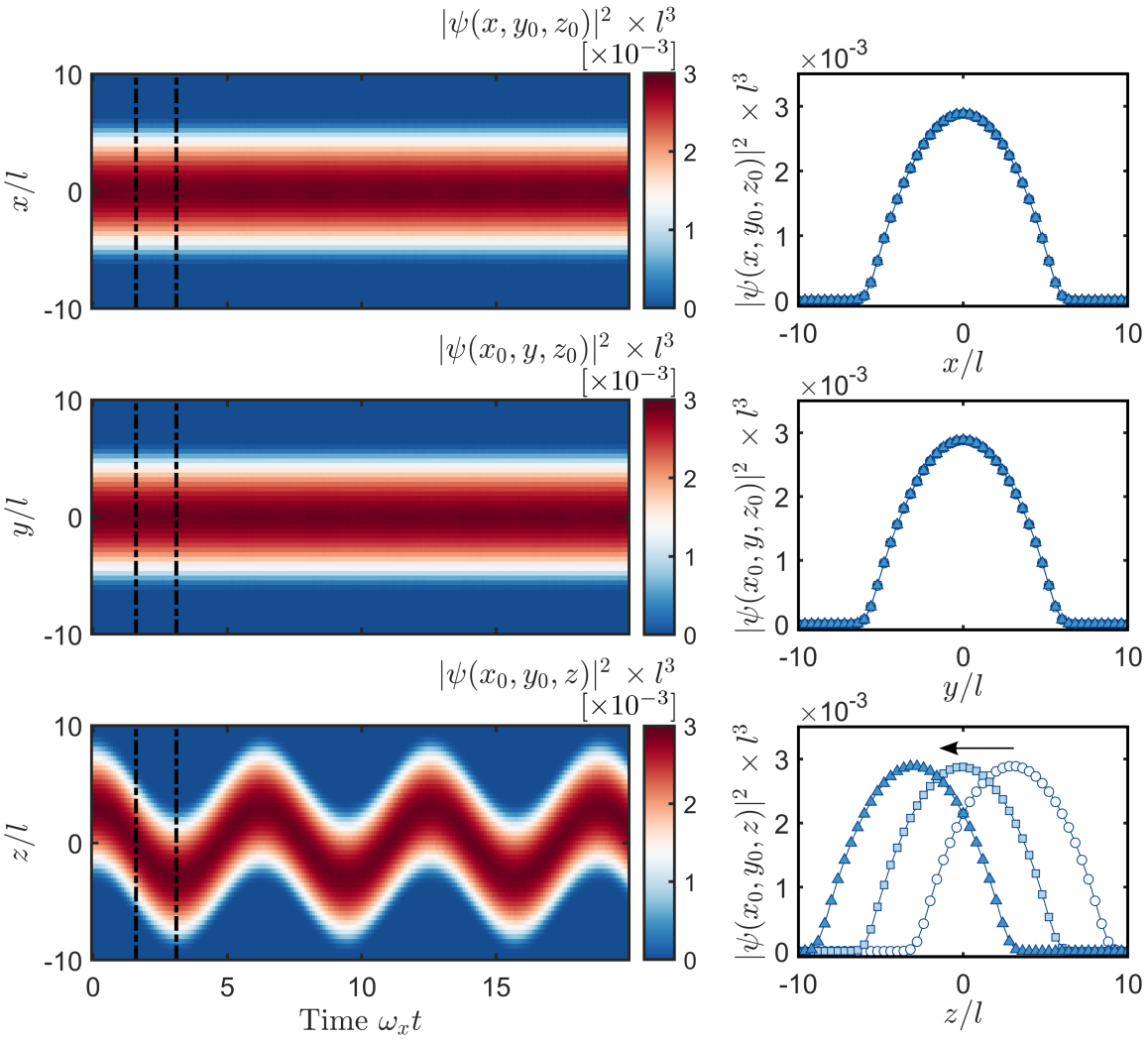

FIGURE 29: Dipole oscillation for a cloud in a harmonic trap by solving the 3D Gross-Pitaevskii equation in real time using the split-step Fourier method. The initial condition is given by the ground state of a cloud with $N = 10^5$ atoms in a spherical harmonic trap with trap frequency $\omega_x = \omega_y = \omega_z = \omega_{\rm ho} = 2\pi \times 100\,{\rm Hz}$, which is displaced along the $\tilde{z}$-direction by 3 units. The simulation then evolves this initial condition in a trap centered on the origin, exciting a dipole (centre of mass) oscillation along the $\tilde{z}$-direction only which proceeds at the trap frequency. The simulation parameters were $N_x = N_y = N_z = 64 \times 64 \times 64$ grid points, with grid spacings of $\Delta\tilde{x} = \Delta\tilde{y} = \Delta\tilde{z} = 0.4$, and a time step of $\Delta\tilde{t} = 0.002$ ($N_t = 10000$ iterations). The position of the cloud center is tracked to be x_0, y_0, z_0, and line cuts of the absolute square of the wavefunction (proportional to density) are taken through the centre and plotted over time in the left column along the $\tilde{x}$ (top), $\tilde{y}$ (middle), and $\tilde{z}$ (bottom) directions. The right column plots three snapshots taken at $\omega_x t = 0$ (initial condition, circles), $\omega_x t = 1.6$ (centre of oscillation, squares), and $\omega_x t = 3.1$ (opposite extreme of oscillation, triangles). These two latter times are indicated by vertical dashed lines in the time-evolution 2D colour plots of the left column.

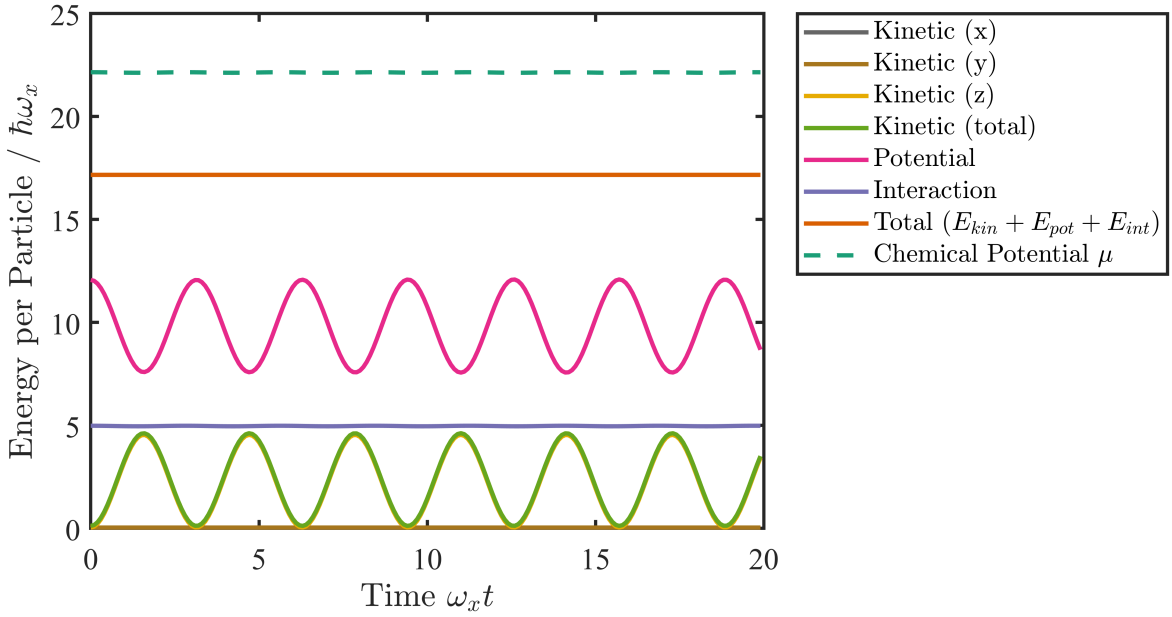

FIGURE 30: Evolution of the energy components per particle (in units of the harmonic oscillator spacing along the x-direction) for a cloud undergoing a dipole (centre of mass) oscillation in an isotropic harmonic trap. The harmonic trapping frequencies were set to be $\omega_x = \omega_y = \omega_z = \omega_{\rm ho} = 2\pi\times\,100\,\text{Hz}$ with an atom number of $N = 10^5$, and the simulation parameters were $N_x = N_y = N_z = 64 \times 64 \times 64$ grid points, with grid spacings of $\Delta\tilde{x} = \Delta\tilde{y} = \Delta\tilde{z} = 0.4$, and a time step of $\Delta\tilde{t} = 0.002$ ($N_t = 10000$ iterations). The oscillation was initiated by using as an initial condition the ground state in a displaced trap along the z-direction, calculated using the imaginary time method. Energy is transferred back and forth between potential energy and kinetic energy in the z-direction, with the total energy remaining conserved.

only along the z-direction, as well as the fact that the interaction energy does not change because the density is fixed with the cloud maintaining its shape. Finally, it is a reassuring check that the split-step Fourier algorithm is energy preserving, as seen by the constant total energy throughout the evolution.

Monopole Oscillation

The result for the dipole mode in the previous section is the same whether the cloud in the interaction-dominated regime or the non-interacting regime, i.e. the centre of mass oscillation always occurs at the harmonic trap frequency. This section will look at another simple collective excitation called the *monopole* oscillation, which is expected to differ from the non-interacting case, and provides another good check for the code. The monopole mode for an isotropic (spherical) trap is illustrated schematically in Fig. 31, and corresponds to a "breathing" mode in which all three directions contract and expand in phase with each other.

To numerically solve the GPE for this case, the ground state was found using the imaginary time split-step Fourier method for a cloud with $N = 10^5$ atoms in an isotropic trap with trap frequencies $\omega_x = \omega_y = \omega_z = \omega_{\rm ho} = 2\pi\times\,200\,\text{Hz}$. The resulting wavefunction was then used as the initial condition

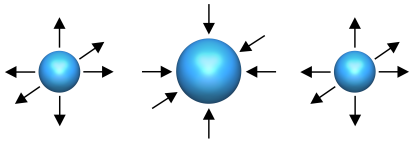

FIGURE 31: Schematic of a monopole mode excitation (breathing oscillation) in a trap.

for a real time simulation in a trap with half of the trap frequencies in every direction, i.e. $\omega_{\text{ho}} = 2\pi\times$ 100 Hz. This quench excites a breathing mode along all directions in phase with each other, which can be seen in Fig. 32 in the absolute square of the wavefunction (proportional to the density). The time period for an oscillation in dimensionless time units can be extracted from this figure to be $\tilde{T} = 2.8$, which in SI units is $T = \tilde{T}/\omega_x = 4.456\,\text{ms}$. The measured oscillation frequency therefore is the inverse of this at 224 Hz.

For an isotropic trap in the interaction-dominated limit (as is satisfied in this simulation, due to the reasonably large atom number), the oscillation frequencies of the normal modes can be calculated in a hydrodynamic approach to be [34]

$$\omega(n,l) = \omega_{\text{ho}}\sqrt{2n^2 + 2nl + 3n + l}. \tag{426}$$

For compressional modes, the quantum number $n \neq 0$, and further the monopole is the lowest solution of the compressional modes with quantum number $l = 0$. This leads to an expected collective excitation frequency of $\omega(n = 1, l = 0) = \sqrt{5}\,\omega_{\text{ho}}$. For the simulation here with $\omega_{\text{ho}} = 2\pi\times\,100\,\text{Hz}$, this predicts a breathing frequency of $2\pi\times$ 224 Hz, as is measured in Fig. 32. Note that this is greater than the result expected for a non-interacting cloud, which would have a monopole breathing frequency of $\omega(n = 1, l = 0) = 2\,\omega_{\text{ho}}$ [34].

The various energy components per particle are also calculated numerically using Eqs. (313), (319) and (325), respectively, as well as the chemical potential using Eq. 329, and their evolution over time is shown in Fig. 33. This time, the energy mainly oscillates back and forth between potential and interaction, as the cloud contracts/expands the density is increased/decreased resulting in larger mean field interaction energy. For the same reason, there is also an oscillation now in the chemical potential, which was absent in the dipole mode case. It is again a reassuring check that the split-step Fourier algorithm is energy preserving, as seen by the constant total energy throughout the evolution.

General Collective Oscillations

Reference [6] provides a general eigenvalue equation obtained by variational analysis to find the collective eigenfrequencies of oscillation for any interaction strength for a triaxially deformed trap (meaning that all three frequencies can be different). In the Thomas-Fermi limit (strongly repulsive) the eigenvalue problem reduces to solutions of the equation

$$0 = \omega^6 - 3\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2\right)\omega^4 + 8\left(\lambda_1^2\lambda_2^2 + \lambda_1^2\lambda_3^2 + \lambda_2^2\lambda_3^2\right)\omega^2 - 20\lambda_1^2\lambda_2^2\lambda_3^2, \tag{427}$$

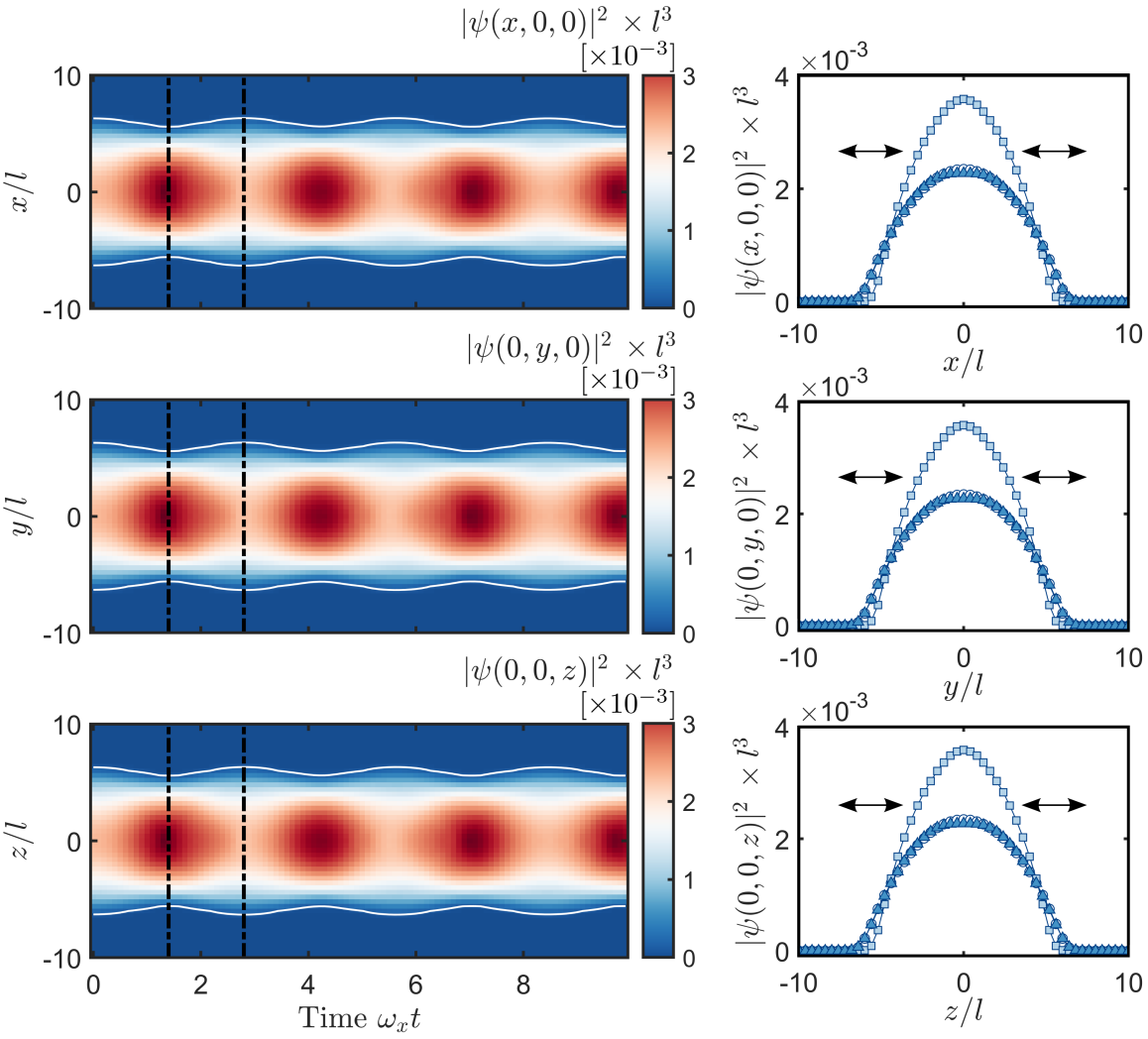

FIGURE 32: Monopole oscillation for a cloud in a harmonic trap by solving the 3D Gross-Pitaevskii equation in real time using the split-step Fourier method. The initial condition is given by the ground state of a cloud with $N = 10^5$ atoms in a spherical harmonic trap with trap frequency $\omega_x = \omega_y = \omega_z = \omega_{\rm ho} = 2\pi \times 200\,$Hz. The simulation then evolves this initial condition in a trap with half the trapping frequencies in each direction, i.e. $\omega_{\rm ho} = 2\pi \times 100\,$Hz, exciting a monopole (breathing) oscillation along all directions in phase. The simulation parameters were $N_x = N_y = N_z = 64 \times 64 \times 64$ grid points, with grid spacings of $\Delta\tilde{x} = \Delta\tilde{y} = \Delta\tilde{z} = 0.4$, and a time step of $\Delta\tilde{t} = 0.002$ ($N_t = 10000$ iterations). Line cuts of the absolute square of the wavefunction (proportional to density) are taken through the origin and plotted over time in the left column along the $\tilde{x}$ (top), $\tilde{y}$ (middle), and $\tilde{z}$ (bottom) directions. The contours display the points at which the value is equal to 10^{-4}, to emphasise the oscillation of the cloud width. The right column plots three snapshots taken at $\omega_x t = 0$ (initial condition, circles), $\omega_x t = 1.4$ (contracted state, squares), and $\omega_x t = 2.8$ (expanded phase, triangles). These two latter times are indicated by vertical dashed lines in the time-evolution 2D colour plots of the left column. The measured breathing frequency is $2\pi \times 224\,$Hz, which agrees with the expected value of $\sqrt{5}\,\omega_{\rm ho}$ for an isotropic trap in the Thomas-Fermi limit [34].

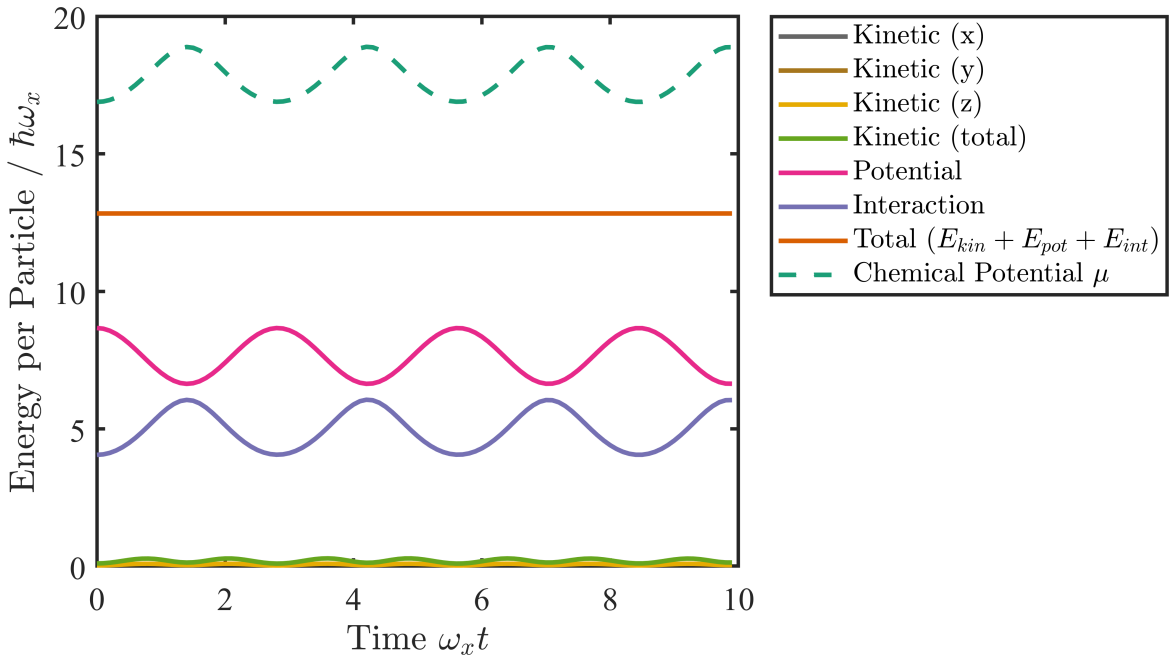

FIGURE 33: Evolution of the energy components per particle (in units of the harmonic oscillator spacing along the x-direction) for a cloud undergoing a monopole (breathing) oscillation in an isotropic harmonic trap. The harmonic trapping frequencies were set to be $\omega_x = \omega_y = \omega_z = \omega_{\text{ho}} = 2\pi\times\, 100\,\text{Hz}$ with an atom number of $N = 10^5$, and the simulation parameters were $N_x = N_y = N_z = 64 \times 64 \times 64$ grid points, with grid spacings of $\Delta\tilde{x} = \Delta\tilde{y} = \Delta\tilde{z} = 0.4$, and a time step of $\Delta\tilde{t} = 0.002$ ($N_t = 10000$ iterations). The oscillation was initiated by using as an initial condition the ground state in trap with twice the trapping frequencies in all directions $\omega_{\text{ho}} = 2\pi\times\, 200\,\text{Hz}$, calculated using the imaginary time method. Energy is transferred back and forth between potential energy and interaction energy (since the density and therefore mean field energy increase as the cloud contracts), with the total energy remaining conserved.

which was also obtained using a hydrodynamic approach [9].

7.10 Expectation Values of a Wavepacket

When numerically solving the GPE, useful quantities to extract are often the average position and variance (spread) of the wavepacket distribution.

In 1D

Consider a 1D wavepacket centered on x_0, with RMS ($1/\sqrt{e}$) width a_x, and peak value B, defined by

$$\psi(x) = B\,\exp\left(-\frac{(x-x_0)^2}{2a_x^2}\right), \tag{428}$$

and a normalisation constant of

$$B = \frac{1}{\pi^{1/4}\sqrt{a_x}} \tag{429}$$

would ensure that the probability distribution integrates to unity over all space

$$\int_{-\infty}^{+\infty} |\psi(x)|^2 \, \mathrm{d}x = 1. \tag{430}$$

In general, the expectation value of any operator $\hat{A}$ is given by

$$\langle \hat{A} \rangle = \frac{\langle \psi | \, \hat{A} \, | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\int_{-\infty}^{+\infty} \psi^* \hat{A} \, \psi \, \mathrm{d}x}{\int_{-\infty}^{+\infty} \psi^* \psi \, \mathrm{d}x}, \tag{431}$$

where the denominator differs from 1 if $|\psi|^2$ is not normalised. This can also be interpreted statistically in terms of weighting by a normalised probability distribution function $p(x)$, where the expectation value of a general function $f(x)$ is given by

$$\langle f(x) \rangle = \int_{-\infty}^{+\infty} f(x) \, p(x) \, \mathrm{d}x. \tag{432}$$

First Moment (Mean)

The expectation value of x is the first moment of the distribution, also called the mean μ, and is given by

$$\mu = \mathrm{E}[x] = \langle x \rangle = \frac{\langle \psi | \, x \, | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\displaystyle\int_{-\infty}^{+\infty} \psi^* x \, \psi \, \mathrm{d}x}{\displaystyle\int_{-\infty}^{+\infty} \psi^* \psi \, \mathrm{d}x} \tag{433}$$

$$= \frac{\displaystyle\int_{-\infty}^{+\infty} B^2 \, x \, \exp\left(-\frac{(x - x_0)^2}{a_x^2} \right) \mathrm{d}x}{\displaystyle\int_{-\infty}^{\infty} B^2 \, \exp\left(-\frac{(x - x_0)^2}{a_x^2} \right) \mathrm{d}x} \tag{434}$$

$$= \frac{1}{\sqrt{\pi} a_x} \int_{-\infty}^{+\infty} x \, \exp\left(-\frac{(x - x_0)^2}{a_x^2} \right) \mathrm{d}x. \tag{435}$$

Making a substitution

$$z = \frac{x - x_0}{a_x} \qquad \Longrightarrow \qquad x = a_x z + x_0 \qquad \Longrightarrow \qquad \mathrm{d}x = a_x \, \mathrm{d}z \tag{436}$$

we have

$$\langle x \rangle = \frac{1}{\sqrt{\pi} a_x} \int_{-\infty}^{+\infty} (a_x z + x_0) \, e^{-z^2} \, a_x \, \mathrm{d}z \tag{437}$$

$$= \frac{1}{\sqrt{\pi}} \left(a_x \int_{-\infty}^{+\infty} z e^{-z^2} \, \mathrm{d}z \, + \, x_0 \int_{-\infty}^{+\infty} e^{-z^2} \, \mathrm{d}z \right). \tag{438}$$

The first integral is zero by symmetry (positive and negative areas cancel), and the second one is a standard integral evaluating to $\sqrt{\pi}$, giving

$$\langle x \rangle = x_0, \tag{439}$$

and so the expected value of x is indeed the one which maximises the probability, i.e. the centre of the distribution.

Second Moment (Variance)

The second moment (variance) is given by the expectation of x^2, whilst removing the offset due to the mean

$$\mathrm{Var}\,(x) = \mathrm{E}\left[(x - \mu)^2 \right] = \langle \, (x - \mu)^2 \, \rangle. \tag{440}$$

Assuming that we have already subtracted the mean from each value of x, which simply removes the offset x_0 from the Gaussian function (i.e the distribution in centered on zero), the variance is given by

$$\mathrm{Var}\,(x) = \langle x^2 \rangle = \frac{\langle \psi | \, x^2 \, | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\displaystyle\int_{-\infty}^{+\infty} \psi^* x^2 \, \psi \, \mathrm{d}x}{\displaystyle\int_{-\infty}^{+\infty} \psi^* \psi \, \mathrm{d}x} \tag{441}$$

$$= \frac{\displaystyle\int_{-\infty}^{+\infty} B^2 \, x^2 \, \exp\left(-\frac{x^2}{a_x^2} \right) \mathrm{d}x}{\displaystyle\int_{-\infty}^{\infty} B^2 \, \exp\left(-\frac{x^2}{a_x^2} \right) \mathrm{d}x} \tag{442}$$

$$= \frac{1}{\sqrt{\pi} a_x} \int_{-\infty}^{+\infty} x^2 \, \exp\left(-\frac{x^2}{a_x^2} \right) \mathrm{d}x. \tag{443}$$

Making the substitution $z^2 = x^2 / a_x^2$,

$$\mathrm{Var}\,(x) = \langle x^2 \rangle = \frac{1}{\sqrt{\pi} a_x} \int_{-\infty}^{+\infty} a_x^2 \, z^2 \, e^{-z^2} a_x \, \mathrm{d}z \tag{444}$$

$$= \frac{a_x^2}{\sqrt{\pi}} \int_{-\infty}^{+\infty} z^2 \, e^{-z^2} \, \mathrm{d}z \tag{445}$$

$$= \frac{a_x^2}{\sqrt{\pi}} \frac{\sqrt{\pi}}{2} \tag{446}$$

$$= \frac{a_x^2}{2}. \tag{447}$$

The variance can also be written as the square of the standard deviation $\text{Var}(x) = \sigma_x^2$, and so we have $\sigma_x = a_x/\sqrt{2}$. Recall that the wavefunction was originally defined to be

$$\psi(x) = B \exp\left(-\frac{(x-x_0)^2}{2a_x^2}\right), \tag{448}$$

and therefore the probability distribution (which is proportional to the density) is

$$|\psi(x)|^2 = B^2 \exp\left(-\frac{(x-x_0)^2}{a_x^2}\right), \tag{449}$$

which we can write in the form

$$|\psi(x)|^2 = B^2 \exp\left(-\frac{(x-x_0)^2}{2\sigma_x^2}\right) \tag{450}$$

with $\sigma_x = a_x/\sqrt{2}$. This means indeed that if you calculate the expectation of x^2 (i.e the variance $\text{Var}(x) = \langle\psi| x^2 |\psi\rangle / \langle\psi|\psi\rangle$) from a wavefunction state ψ, the result is the $1/\sqrt{e}$ width ("sigma") of the *probability* (or *density*) distribution.

Numerical Example

The following code creates an unnormalised gaussian wavefunction with the parameters $a_x = 7.5$ and $x_0 = 9$, and then calculates the first and second moments (mean and variance). It can be seen that the square root of the variance returns 5.303, i.e. the expected "sigma" of the probability distribution.

```
Nx = 150;                          % Number of grid points
dx = 0.5;                          % Grid spacing
x  = -(Nx*dx/2):dx:(Nx*dx/2-dx); % Spatial grid vector

a_x = 7.5; % 1/sqrt(e) width of wavefunction
x0 = 9;

B = 20; % Peak value of psi
psi = B * exp( -(x-x0).^2 / ( 2*a_x^2 ) ); % Define wavefunction
n = conj(psi).*psi; % Probability distribtuion (proportional to density)

% First moment: E[x] = <x>
mu_x = sum( conj(psi).* x .*psi )*dx / ( sum( conj(psi).*psi )*dx );

% Second moment: E[(x-mu)^2] = <(x-mu)^2>
variance_x = sum( conj(psi).* (x - mu_x).^2 .*psi )*dx / ( sum( conj(psi).*psi )*dx );
sigma_x = sqrt(variance_x); % Standard deviation

figure; plot(x, n )

fprintf('Var(x) = sx^2 = %.4f \n',variance_x)
```

```
fprintf('sx = sqrt( Var(x) ) = %.4f \n',sigma_x)
fprintf('Expected sigma of density dist. = a_x/sqrt(2) = %.4f \n', a_x/sqrt(2))
```

The output printed is

```
Var(x) = sigma_x^2 = 28.1250
sigma_x = sqrt( Var(x) ) = 5.3033
Expected sigma of density dist. = a_x/sqrt(2) = 5.3033
```

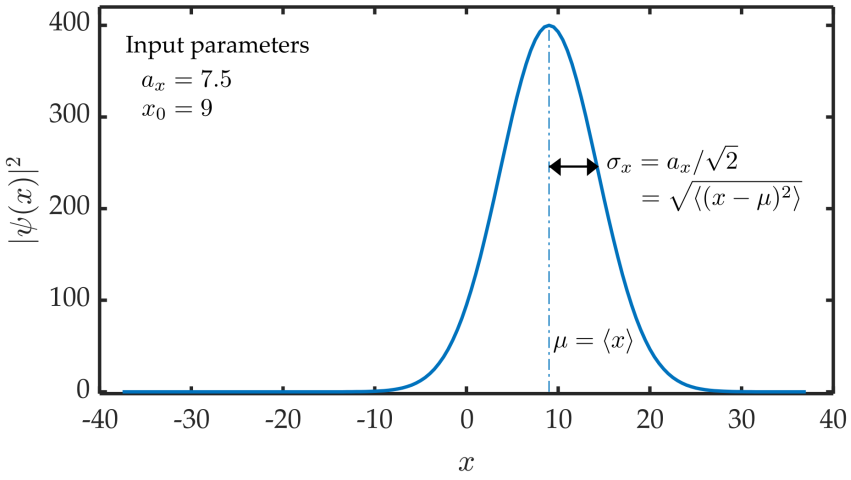

FIGURE 34: Probability distribution $|\psi(x)|^2$ for a wavefunction with $1/\sqrt{e}$ width $a_x = 7.5$ and centred on $x_0 = 9$. The probability distribution has a $1/\sqrt{e}$ width of $\sigma_x = 5.303$ (smaller than that of the wavefunction by $\sqrt{2}$), and is calculated by the square root of the variance.

In 3D

A similar example in a full 3D case is given in the code below. This time, the integrals for the expectation are calculated over all volume, and there are three variances - one for each dimension. The mean value has not been subtracted, since the function is already centered on zero here.

```
Nx = 80;  dx = 0.8; x  = -(Nx*dx/2):dx:(Nx*dx/2-dx); % X grid
Ny = 102; dy = 1;   y  = -(Ny*dy/2):dy:(Ny*dy/2-dy); % Y grid
Nz = 84;  dz = 0.4; z  = -(Nz*dz/2):dz:(Nz*dz/2-dz); % Z grid
dV = dx*dy*dz; % Volume element
[X,Y,Z] = meshgrid(x,y,z);

a_x = 7.5; % 1/sqrt(e) widths of the wavefunction psi
a_y = 12;
```

```
a_z = 4;

B = 5;                      % Peak value of wavefunction
psi = B*exp( -X.^2/(2*a_x^2) -Y.^2/(2*a_y^2) -Z.^2/(2*a_z^2) ); % Gaussian wavefunction
n = abs(psi).^2;            % Density distribution
N = sum(sum(sum( n )))*dV; % Integral over |psi|^2 for normalisation

% Second moments: E[(x-mu)^2] = <(x-mu)^2>
variance_x = sum(sum(sum( conj(psi).* X.^2 .*psi )))*dV / N;
sigma_x = sqrt(variance_x);
variance_y = sum(sum(sum( conj(psi).* Y.^2 .*psi )))*dV / N;
sigma_y = sqrt(variance_y);
variance_z = sum(sum(sum( conj(psi).* Z.^2 .*psi )))*dV / N;
sigma_z = sqrt(variance_z);
```

```
a_x             = 7.5000
a_x/sqrt(2)     = 5.3033  (Expected RMS width along x)
sqrt( Var(x) ) = 5.3033
a_y             = 12.0000
a_y/sqrt(2)     = 8.4853  (Expected RMS width along y)
sqrt( Var(y) ) = 8.4853
a_z             = 4.0000
a_z/sqrt(2)     = 2.8284  (Expected RMS width along z)
sqrt( Var(z) ) = 2.8284
```

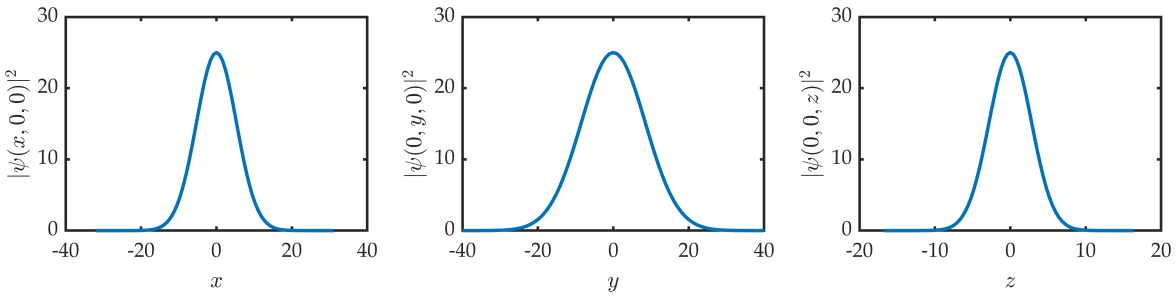

FIGURE 35: Cuts through the origin of the probability distribution $|\psi(x,y,z)|^2$ along each direction for a wavefunction with widths $a_x = 7.5$, $a_y = 12$, and $a_z = 4$. The probability distribution has $1/\sqrt{e}$ widths of $\sigma_x = 5.303$, $\sigma_y = 8.485$, and $\sigma_z = 2.828$ (smaller than those of the wavefunction by $\sqrt{2}$), calculated by the square root of the variance.

In the example above, I have used the `meshgrid()` function, which creates x3 3D grids of spatial coordinates. Whilst this is simple and allows easy point-wise multiplication, it is very memory intensive because there are many repeated coordinates stored. In practice, the only information we need are x3 1D vectors to represent the spatial grids. We can then make use of Matlab's *implicit ex-*

pansion feature, in order to multiply the 3D wavefunction by a 1D vector. An example looks like the following, and does not make use of meshgrid, saving memory

```
%%% Permute spatial vectors to enable implicit expansion
x_pm = permute(x, [1 2 3]); % Results in [1 x Nx x 1] vector
y_pm = permute(y, [2 1 3]); % Results in [Ny x 1 x 1] vector
z_pm = permute(z, [3 1 2]); % Results in [1 x 1 x Nz] vector

variance_x = sum(sum(sum( n.* x_pm.^2 )))*dV / N ;
sigma_x = sqrt(variance_x);
variance_y = sum(sum(sum( n.* y_pm.^2 )))*dV / N ;
sigma_y = sqrt(variance_y);
variance_z = sum(sum(sum( n.* z_pm.^2 )))*dV / N ;
sigma_z = sqrt(variance_z);
```

7.11 A Note on Memory Requirements

When calculating solutions of the Gross-Pitaevskii equation numerically using the split-step method, the minimum thing that *must* be stored in memory (RAM) at any one time (assuming no chunking is employed) is a 3D array of values representing the wavefunction on a 3D spatial grid. To obtain a reasonable resolution in a simulation, to see fine details and maintain accuracy, the number of grid points required can become relatively large, which means that we need to consider the amount of memory that will be required to store the object.

Usually, the wavefunction will be stored in floating point format with *double precision*, to take advantage of the increased precision offered by the format as opposed to *single precision*. A number of type double requires 64 bits (= 8 bytes) of memory in order to represent it (of these, 52 bits are dedicated to the significand, 11 bits to the exponent, and 1 bit for the sign of the number). In addition, the wavefunction will be a complex quantity, and requires both the real and imaginary part of its value at each grid point to be stored, which leads to a factor of two increase in memory requirements. Therefore, per grid point the memory required to store a complex double value is 16 bytes. Examples of the RAM requirements depending on the 3D grid size are given in Table 1. A typical PC may have something like 8 GB of RAM, and so very quickly the RAM can be used up as the grid size increases.

TABLE 1: RAM requirements for various sized 3D complex double arrays.

$N_x \times N_y \times N_z$	Number of Elements	RAM Required
$64 \times 64 \times 64$	$262,144$	4.19 MB
$512 \times 512 \times 512$	$134,217,728$	2.14 GB
$1024 \times 1024 \times 1024$	$1,073,741,824$	17.2 GB

To see how much RAM a particular variable is occupying, we can use the command `whos` in Matlab. For example, the code below creates a $64 \times 64 \times 64$ complex double array, and then verifies that the space used is 4.19 MB, as expected.

```
>> psi = complex( zeros(64,64,64,'double') );
>>
>> whos psi
  Name        Size                 Bytes  Class     Attributes

  psi        64x64x64           4194304  double    complex
```

7.12 Optimum Grid Sizes for FFT Speed

Conventional wisdom says that array lengths which are powers of 2 are best for FFTs (64, 128, 256, etc). However, this can be restrictive if, for example, a grid size of 600 points would be suitable for the computation but the next power of 2 available is actually the much larger value of 1024. The gain in speed by using a power of 2 in that case can be wasted because the grid size is unnecessarily large. However, Matlab's FFT function is based on the FFTW algorithm (*Fastest Fourier Transform in the West*), which works most efficiently for array lengths that can be factorized into products of small prime numbers (2, 3, 5, and 7). Specifically, this means that the array length N should be able to be written in the form

$$N = 2^{\alpha} \times 3^{\beta} \times 5^{\gamma} \times 7^{\delta}. \tag{451}$$

In addition, for using fftshift/ifftshift and arranging the grid vectors in Matlab, it is convenient to take even length arrays only. Matlab has a function `nextpow2()` for finding the next power of 2 of a number, so for example the command `nextpow2(600)` returns `10`, since the next power of 2 is $2^{10} = 1024$. However, it does not have a function for calculating the even products of small prime factors which we need for the FFTW algorithm. For reference, I have precalculated in Table 2 a list of the array lengths which can be factorized into the form $N = 2^{\alpha} \times 3^{\beta}$, which in practice provides enough flexibility in the choice of grids.

TABLE 2: List of array lengths which can be factorized into the form $N = 2^{\alpha} \times 3^{\beta}$, as required for optimum speed of the FFTW algorithm.

N	Factorized	N	Factorized	N	Factorized
2	2	64	2^6	384	$2^7 \times 3$
4	2^2	72	$2^3 \times 3^2$	432	$2^4 \times 3^3$
6	2×3	96	$2^5 \times 3$	486	2×3^5
8	2^3	108	$2^2 \times 3^3$	512	2^9
12	$2^2 \times 3$	128	2^7	576	$2^6 \times 3^2$
16	2^4	144	$2^4 \times 3^2$	648	$2^3 \times 3^4$
18	2×3^2	162	2×3^4	768	$2^8 \times 3$
24	$2^3 \times 3$	192	$2^6 \times 3$	864	$2^5 \times 3^3$
32	2^5	216	$2^3 \times 3^3$	972	$2^2 \times 3^5$
36	$2^2 \times 3^2$	256	2^8	1024	2^{10}
48	$2^4 \times 3$	288	$2^5 \times 3^2$	1152	$2^7 \times 3^2$
54	2×3^3	324	$2^2 \times 3^4$	1296	$2^4 \times 3^4$

7.13 Acceleration of Code Using a GPU

When performing calculations on a CPU, it is the CPU's speed which influences how fast the operations can be carried out. For example, an *Intel Core i7-6700* CPU runs at $3.4\,\text{GHz}$, with 4 cores (and 8 threads) available, and using a faster processor will reduce the calculation time. To carry out calculations even faster, we can make use of a *graphics processing unit* (GPU). These are optimised for doing many calculations in parallel, since they were designed to render the values of many thousands of pixels simultaneously on a display monitor. An example of a GPU which can be used for this purpose is the *Nvidia Quadro P4000*, which has 1792 CUDA cores and 8 GB of memory on the device.

The typical workflow is to first move a data array that we want to manipulate (this will be the 3D array representing the values of the wavefunction in the case of solving the Gross-Pitaevskii equation) from the PC's RAM over to the GPU. This can be done in Matlab simply with the command `gpuArray()`, which creates a new version of the variable on the GPU. This new variable will be of type `gpuArray`. Next, the manipulation of the data is done directly on the GPU. Many Matlab functions are already able to work directly on the GPU, and do so automatically when passing them a gpuArray type variable. For example, passing a gpuArray object to the FFT algorithm using `fftn()` on the object will automatically carry out the FFT on the GPU. FFTs in particular can benefit from large speedup when done on the GPU, typically by a factor of $\times 20$. Point-wise multiplication of large matrices can also benefit greatly from GPU operation. Once all the calculations have finished, the new variable is brought back from the GPU memory into the PC's RAM again using the `gather()` command. A typical workflow might look like the following

```
Nx = 64;
Ny = 64;
Nz = 64;
psi_in = rand(Nx,Ny,Nz) + i*rand(Nx,Ny,Nz); % Create a random complex wavefunction
psi_gpu = gpuArray(psi_in);     % Move the 3D array to the GPU
psi_hat_gpu = fftn(psi_gpu);    % Perform an FFT transform directly on the GPU
psi_out = gather(psi_hat_gpu);  % Bring the result back to the PC's RAM
```

The properties of the GPU can be viewed using `gpuDevice`. If one has access to a GPU, this can be a very simple way to obtain a large speed-up factor in the code - especially when using the split-step method which requires many thousands of consecutive forward and inverse transfom operations.

8 Time-of-Flight Expansion

In experiments, the density of a BEC is usually measured with absorption imaging [24, 35–37] after allowing the cloud to evolve for some time-of-flight after extinguishing the trapping potential. This is done to either study the momentum distribution of the in-trap gas, or simply to allow the density to become more dilute so as not to be optically thick to the resonant probe light. For this reason, it is important to be able to simulate how a cloud evolves under the GPE, in order to compare with experimental data.

Calculation of time-of-flight distributions is not particularly common, because it can be a computationally intensive task - the wavefunction can grow by several orders of magnitude during expansion, requiring significant amounts of RAM and calculation time. Often, ballistic expansion is assumed, which reduces the evolution to that of the Schrodinger equation but neglects the effect of interactions. Time-of-flight evolution with the Gross-Pitaevskii equation can be done with modern PC workstations and GPU, as described in the following sections.

8.1 Reproducing the Results of Holland et al.

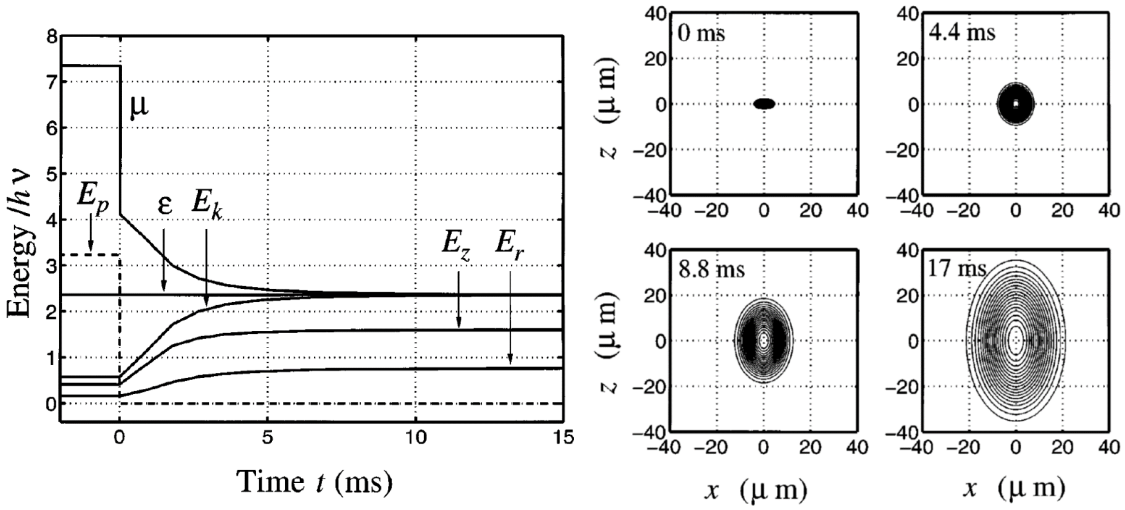

FIGURE 36: Results from Holland et al. 1997 [4]. (left) Energy components during the expansion of a 4000 atom condensate. The trap frequencies were $\omega_x = \omega_y = 2\pi\nu = 2\pi\times 56.25\,\text{Hz}$ and $\omega_z = 2\pi\times 160\,\text{Hz}$, and were removed at $t = 0$. Shown are the radial E_r, axial E_z, and total E_k kinetic energies, the potential energy E_p, the release energy ε, and the chemical potential μ. (right) Contour images of the wave function density at four times during the expansion.

The first example in the literature of a numerical calculation of a BEC being released from a trapping potential was Holland et al. in 1997 [4]. In order to test my code, I have calculated the dynamics of an expanding BEC using the same parameters, to check the results are the same. Their study used the imaginary time method to find the ground state, and then evolved the system in real time using the finite difference method in cylindrical coordinates (reducing the problem to two dimensions due

to axial symmetry), and the two dimensions were dealt with using an alternating direction implicit (ADI) approach. The trap frequencies were $\omega_x = \omega_y = 2\pi \times 56.25\,\mathrm{Hz}$ in the radial plane, and $\omega_z = 2\pi \times 160\,\mathrm{Hz}$ in the tightly confined axial direction, with a total atom number of $N = 4000$. The dynamics were studied over $17\,\mathrm{ms}$ of free expansion (potential removed), and their main figures are reproduced here in Fig. 36. We can see that the chemical potential decreases during the first few milliseconds of the expansion, as the mean field interaction energy drops (due to reducing density) and is converted into kinetic energy in a period of accelerated expansion. More of the converted energy goes into the z direction, since this was more strongly confined in trap. Once this conversion phase is over, the kinetic energy is constant and the expansion proceeds at constant velocity along each direction. In the contour density plots we can see that the expansion is anisotropic, and an inversion of the aspect ratio takes place after around 4 ms.

I have simulated the same situation (trap frequencies of $\omega_x = \omega_y = 2\pi \times 56.25\,\mathrm{Hz}$ in the radial plane, and $\omega_z = 2\pi \times 160\,\mathrm{Hz}$ in the tightly confined axial direction, with a total atom number of $N = 1000$) using my GPE code which employs the split-step Fourier spectral method with Strang splitting as described in Sec. 7.4. First, the true numerical ground state was found by evolving the system in imaginary time from an initial guess (the non-interacting Gaussian ground state).The parameters for this stage were grid spacings of $\Delta x = \Delta y = 0.72\,\mu\mathrm{m}$ and $\Delta z = 0.47\,\mu\mathrm{m}$, and the initial number of grid points was $64 \times 64 \times 64$. Then, this ground state was used as the initial condition for evolving the system in real time with the trapping potential set to zero, to simulate the time-of-flight free expansion over $17\,\mathrm{ms}$, with a time step of $\Delta t = 23.8\,\mu\mathrm{s}$ and $N_t = 800$ iterations. Figure 37 shows 3D isosurfaces and 2D slices of the density $n(x, y, z) = N|\psi(x, y, z)|^2$ at several times through the expansion, and line cuts of the density through the cloud center along each direction are given in Fig. 38.

Dynamically Expanding Grids

One of the biggest challenges when calculating time-of-flight dynamics with the GPE is the massive expansion of the wavefunction over time. In the calculation for the parameters above, the weakly confined direction expands by a factor of ~ 4, whilst the tightly confined direction expands by ~ 20 over the full course of the expansion, as can be seen in Figs. 37 and 38. To accommodate the wavefunction as it grows, I have implemented a dynamically-expanding spatial grid. This works by checking every few time steps (typically every 5 or 10 steps) what is the spatial extent of the cloud. The spatial extent is approximated by calculating the RMS (from the square root of the expectation value of x_i^2 as described in Sec. 7.10) and multiplying it by 3, which typically captures well what one would take to be the cloud's size (if the cloud was Gaussian, this would encompass 99.7% of the atoms, for example). The cloud size is then compared with the extent of the grid (known as the support), and if the size fills more than half of the grid along any direction then this direction is expanded by padding the wavefunction with more zeros around the edges (the expansion factor is chosen to be 1.3, which means the grid is expanded by 30% each time the trigger is met). In the simulation presented here, several grid expansions take place over the full course, and the initial grid of $64 \times 64 \times 64$ grows to $128 \times 128 \times 384$ by the end. Each new grid size chosen is rounded up to a product of small prime factors, which is necessary for obtaining the best performance of the FFT algorithm as described in

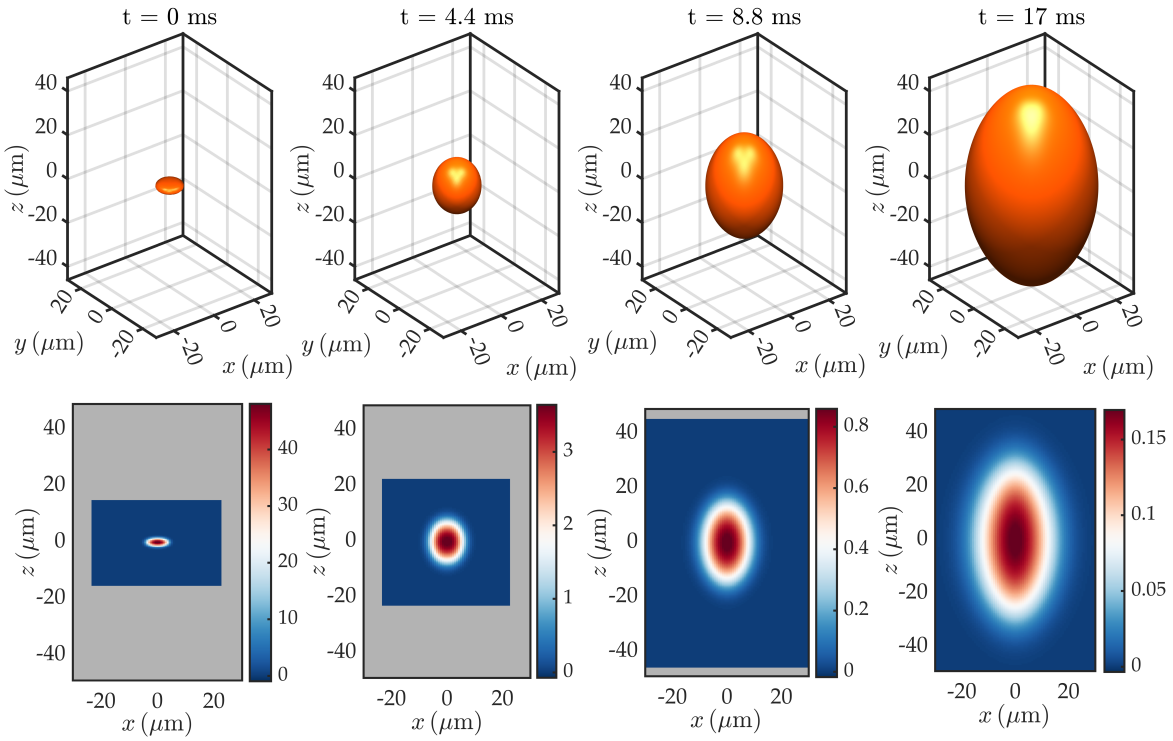

FIGURE 37: Snapshots at several times during the evolution for a cloud undergoing time-of-flight free expansion from an anisotropic cylindrically-symmetric harmonic trap with trapping frequencies $\omega_x = \omega_y = 2\pi\times$ 56.25 Hz and $\omega_z = 2\pi\times$ 160 Hz (all removed at $t = 0$), and an atom number of $N = 4000$. (Top row) Surfaces of constant density, determined when the density reaches 1% of it's peak value, in order to illustrate the spatial extent of the cloud. (Bottom row) Snapshots of the density $n(x, 0, z) = N|\psi(x, 0, z)|^2$ in the $x - z$ plane through $y = 0$, in units of atoms per μm^3. As the cloud expands in size, the number of grid points was increased by 30% whenever the cloud extent (taken to be 3σ from the centre) reached half of the grid support along each direction, with the grey shading denoting the regions not yet covered by the grid. Grid spacings were $\Delta x = \Delta y =$0.72 μm and $\Delta z =$0.47 μm, and the initial grid of $64 \times 64 \times 64$ was expanded to $128 \times 128 \times 384$ by the end of the simulation. The cloud expands anisotropically, faster along the direction which was initially more strongly confined.

Sec. 7.12. The grid expansion can be seen in Figs. 37 and 38.

Note that it is *not possible* to downsample the data when doing the grid expansion (i.e. trying to keep the same number of grid points), because although that might seem fine in terms of the density (which appears smooth enough to downsample) we need to consider the wavefunction's distribution in k-space. Throughout the evolution an oscillation builds up in k-space whose frequency increases with time, as shown in Fig. 40. The overall width of the distribution in k-space does not really change through the expansion (in the z-direction for example it extends from around -5 to +5 for all TOFs). If we were to downsample the grid (i.e. make the real space grid spacings larger to maintain the same number of points as the grid grows) then the maximum k-vector value would necessarily be reduced (since fine features could no longer be resolved) and the k-space distribution would be clipped -

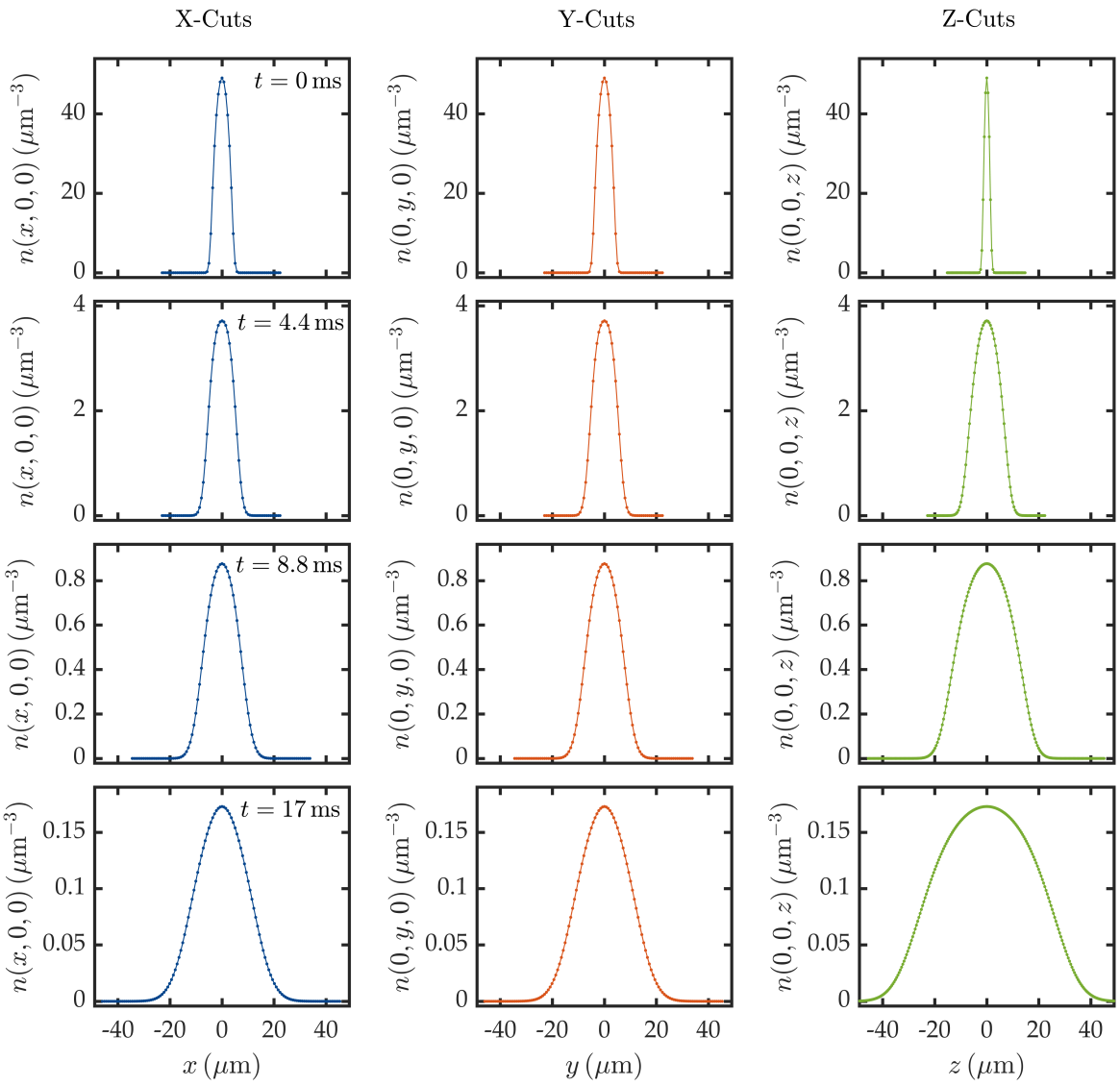

FIGURE 38: Snapshots of line cuts of the density $n(x, y, z) = N|\psi(x, y, z)|^2$ through the cloud center along each direction (columns, left to right) at several times (rows, top to bottom) during the evolution for a cloud undergoing time-of-flight free expansion. The initial harmonic trap was an anisotropic and cylindrically-symmetric, with trapping frequencies $\omega_x = \omega_y = 2\pi\times\, 56.25\,\mathrm{Hz}$ and $\omega_z = 2\pi\times\, 160\,\mathrm{Hz}$ (all removed at $t = 0$), and an atom number of $N = 4000$. As the cloud expands in size, the number of grid points was increased by 30% whenever the cloud extent (taken to be 3σ from the centre) reached half of the grid support along each direction. Grid spacings were $\Delta x = \Delta y = 0.72\,\mu\mathrm{m}$ and $\Delta z = 0.47\,\mu\mathrm{m}$, and the initial grid of $64 \times 64 \times 64$ was expanded to $128 \times 128 \times 384$ by the end of the simulation. The cloud expands anisotropically, faster along the z direction which was initially more strongly confined.

resulting in aliasing - and the wavefunction would be scrambled. This is the reason that number of points needs to increase during this phase. For much larger times-of-flight the number of points can grow so much that the PC will run out of RAM (see Sec. 7.11 for estimates of the amount of RAM

needed). In this case the method described in Sec. 8.4 can be used for ballistic expansion (i.e. once the interactions can be neglected) whilst avoiding the memory problems.

The various numerically calculated energy components throughout the initial time-of-flight expansion stage are shown in Fig. 39, and are in good agreement with the result from Holland et al. in Fig. 36. Note that their value of the chemical potential before switching off the trap is slightly higher due to the fact that they used a value for the scattering length of 110 times the Bohr radius, whereas my simulation uses 98.98 Bohr radius, in accordance with more recent measurements [38]. These results give confidence that my time-of-flight expansion code is working correctly. When running on a Lenovo Thinkpad with Intel Core i7-7600U 2.8 GHz CPU, the time to calculate the ground state was 9.3 s and the real time evolution for time-of-flight expansion was 216 s. When accelerating the code by carrying out the calculations on a Nvidia Quadro P4000 GPU (1792 CUDA cores), the ground state was found in 5 s and the real time evolution in 9.1 s (giving a speedup of $\times 24$).

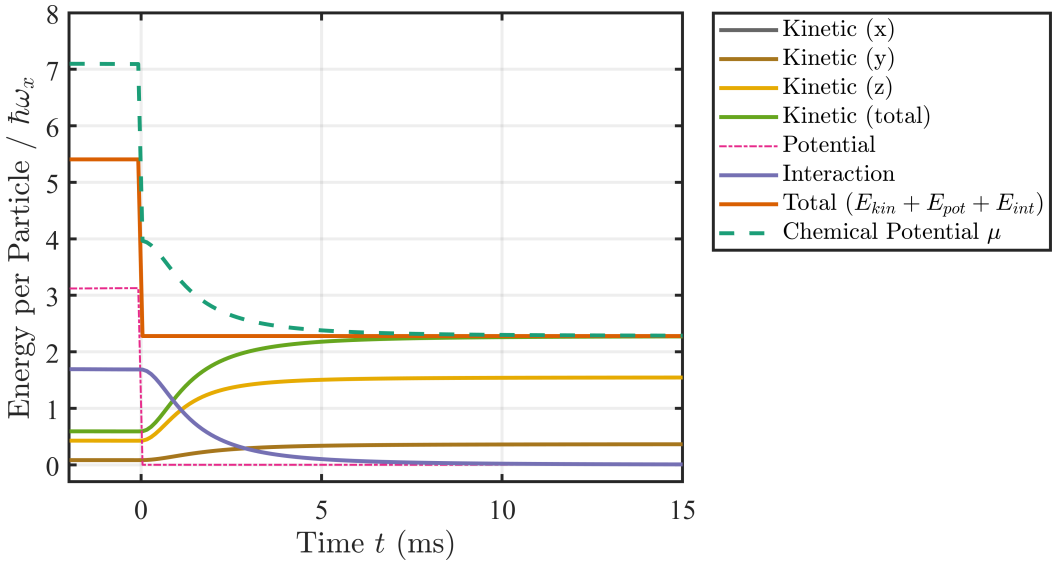

FIGURE 39: Evolution of the energy components per particle (in units of the harmonic oscillator spacing along the x-direction) for a cloud undergoing time-of-flight free expansion from an isotropic harmonic trap. The harmonic trapping frequencies were $\omega_x = \omega_y = 2\pi \times 56.25$ Hz and $\omega_z = 2\pi \times 160$ Hz with an atom number of $N = 4000$. The in-trap ground state was calculated using the imaginary time method, before evolving in real time with a time step of $\Delta t = 23.8\,\mu$s ($N_t = 800$ iterations), switching off the trap at $t = 0$, at which point energy is transferred from stored interaction energy (as the density drops) into kinetic energy in a period of accelerated expansion, with the total energy remaining conserved.

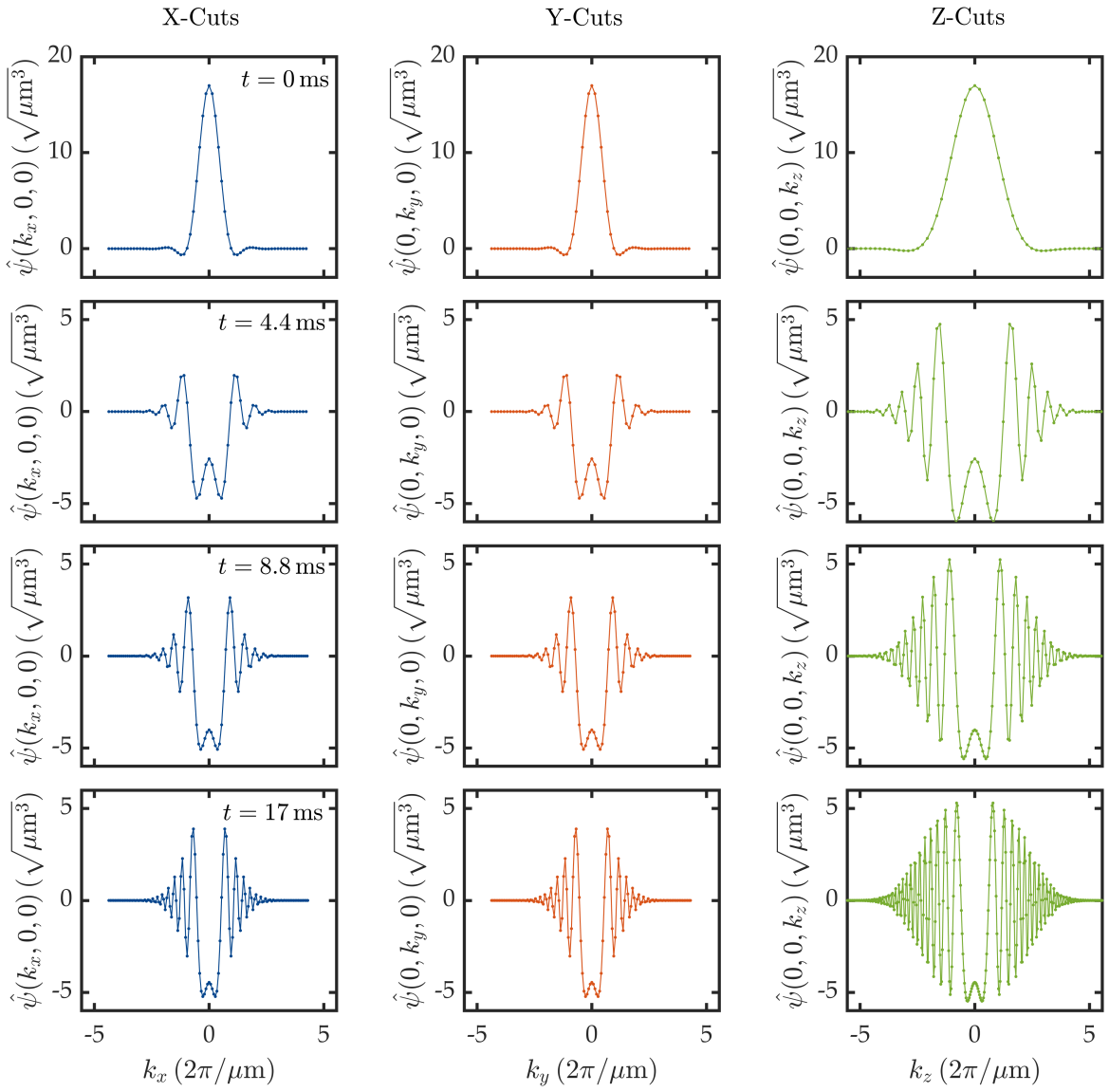

FIGURE 40: Snapshots of line cuts of the real part of the k-space wavefunction $\mathrm{Re}[\hat{\psi}(k_x, k_y, k_z)]$ through the cloud center along each direction (columns, left to right) at several times (rows, top to bottom) during the evolution for a cloud undergoing time-of-flight free expansion. The initial harmonic trap was an anisotropic and cylindrically-symmetric, with trapping frequencies $\omega_x = \omega_y = 2\pi\times\, 56.25\,\mathrm{Hz}$ and $\omega_z = 2\pi\times\, 160\,\mathrm{Hz}$ (all removed at $t = 0$), and an atom number of $N = 4000$. As the cloud expands in size, the number of grid points was increased by 30% whenever the cloud extent (taken to be 3σ from the centre) reached half of the grid support along each direction. This expansion of the grid in real space leads to an interpolation in k-space, which is necessary to capture the increasingly fast oscillation in k-space, and prevent aliasing. Grid spacings were $\Delta x = \Delta y = 0.72\,\mu\mathrm{m}$ and $\Delta z = 0.47\,\mu\mathrm{m}$, and the initial grid of $64 \times 64 \times 64$ was expanded to $128 \times 128 \times 384$ by the end of the simulation. Since these real-space grid spacings were fixed, the maximum k-vector does not change throughout the simulation.

8.2 Castin-Dum Scaling Approximation

The Thomas-Fermi approximation for the Gross-Pitaevskii equation is valid when the interactions are dominant and the kinetic energy can be neglected. In this case, the repulsive interactions "press" the cloud into the shape of the external trap, and the density distribution for a cloud in a harmonic trapping potential is given by Eq. 220, and takes the form of an inverted parabola

$$n(x,y,z) = \max\left\{\frac{15}{8\pi}\frac{N}{R_x R_y R_z}\left(1 - \frac{x^2}{R_x^2} - \frac{y^2}{R_y^2} - \frac{z^2}{R_z^2}\right), 0\right\}, \tag{452}$$

where R_i are the Thomas-Fermi radii of the condensate along each direction (representing the distance at which the density drops to zero). When the frequencies of the external trap are varied in time $\omega_i(t)$, it has been shown by Castin and Dum [39–41] that in the Thomas-Fermi limit the time-dependent GPE can be described by hydrodynamic equations, and the solution is a *self-similar scaling* of the density. Specifically, the density remains parabolic but now with time-dependent Thomas-Fermi radii $R_i(t)$ which evolve according to three scaling factors $\lambda_i(t)$ along each direction

$$R_i(t) = \lambda_i(t) R_i(0) \qquad (i = x, y, z). \tag{453}$$

To find the scaling factors, one needs to solve the three differential equations

$$\frac{\mathrm{d}^2\lambda_i}{\mathrm{d}t^2} = \frac{\omega_i^2(0)}{\lambda_i\lambda_x\lambda_y\lambda_z} - \omega_i^2(t)\lambda_i \qquad (i = x, y, z), \tag{454}$$

with the initial conditions $\lambda_i(0) = 1$ (the Thomas-Fermi radius is equal to its initial value before changing the frequencies) and $\dot{\lambda}_i(0) = 0$ (because the gas is initially at rest). In this equation, the first term on the right hand side accounts for the effect of interactions, while the second term describes the change in the trap potential. This can be used, for example, to investigate collective oscillations upon abruptly changing the trap frequencies. However, setting $\omega_i(t) = 0$ corresponds to a complete removal of the trap, and so time-of-flight expansion can be calculated by solving the coupled equations

$$\frac{\mathrm{d}^2\lambda_i}{\mathrm{d}t^2} = \frac{\omega_i^2(0)}{\lambda_i\lambda_x\lambda_y\lambda_z} \qquad (i = x, y, z). \tag{455}$$

The following Matlab code uses ODE45 to numerically solve the general form in Eq. 454 to give the time-dependence of the scaling factors:

```
dt = 0.1e-3;     % Time step
tmax = 17e-3;    % Maximum time
t = 0:dt:tmax;   % Create time vector

% Initial trap frequencies
omega_x_0 = 2*pi*56.25;
omega_y_0 = 2*pi*56.25;
omega_z_0 = 2*pi*160;
```

```
9
10  % New trap frequencies
11  omega_x_t = 2*pi*0;
12  omega_y_t = 2*pi*0;
13  omega_z_t = 2*pi*0;
14
15  % Initial conditions
16  lambda_x_0 = 1;
17  lambda_y_0 = 1;
18  lambda_z_0 = 1;
19  dlambda_x_dt_0 = 0;
20  dlambda_y_dt_0 = 0;
21  dlambda_z_dt_0 = 0;
22  initial_conds = [lambda_x_0 dlambda_x_dt_0 ...
23                   lambda_y_0 dlambda_y_dt_0 ...
24                   lambda_z_0 dlambda_z_dt_0];
25
26  % Solve the second order Castin-Dum scaling equations
27  [t, x] = ode45( @(t,x) rhs(t, x, omega_x_0, omega_y_0, omega_z_0, ...
28                                   omega_x_t, omega_y_t, omega_z_t), t, initial_conds );
29
30  % Plot the scaling factors for each direction
31  plot(t, x(:,1) )
32  plot(t, x(:,3) )
33  plot(t, x(:,5) )
34
35  function Xd = rhs(t, x, omega_x_0, omega_y_0, omega_z_0, omega_x_t, omega_y_t, omega_z_t)
36
37  lambda_x_t     = x(1);
38  dlambda_x_dt_t = x(2);
39  lambda_y_t     = x(3);
40  dlambda_y_dt_t = x(4);
41  lambda_z_t     = x(5);
42  dlambda_z_dt_t = x(6);
43
44  Xd = [ dlambda_x_dt_t;...
45         omega_x_0^2/(lambda_x_t^2*lambda_y_t*lambda_z_t) - omega_x_t^2*lambda_x_t;...
46         dlambda_y_dt_t;...
47         omega_y_0^2/(lambda_x_t*lambda_y_t^2*lambda_z_t) - omega_y_t^2*lambda_y_t;...
48         dlambda_z_dt_t;...
49         omega_z_0^2/(lambda_x_t*lambda_y_t*lambda_z_t^2) - omega_z_t^2*lambda_z_t;];
50  end
```

For comparison with the earlier results in Sec. 8.1 , I have used the above code to show in Fig. 41 the evolution of the scaling factors from Castin-Dum theory for the parameters used in the numerical study of Holland, et. al. [4]. The initial trap frequencies were $\omega_x = \omega_y = 2\pi \times 56.25\,\mathrm{Hz}$ and $\omega_z = 2\pi \times 160\,\mathrm{Hz}$, which were then set to zero for all subsequent times (to study the time-of-flight expansion). The in-trap radii in the Thomas-Fermi approximation were calculated using the atom number of $N = 4000$. Once the scaling parameters are calculated, the time-dependence of the Thomas-Fermi radii during the expansion are easily obtained using the relation in Eq. 453. We can see that indeed like the study of Holland, et. al., an inversion of the aspect ratio takes place around $3.7\,\mathrm{ms}$, due to the

tighter initial confinement along the z-direction, and therefore faster increasing scaling factor $\lambda_z(t)$.

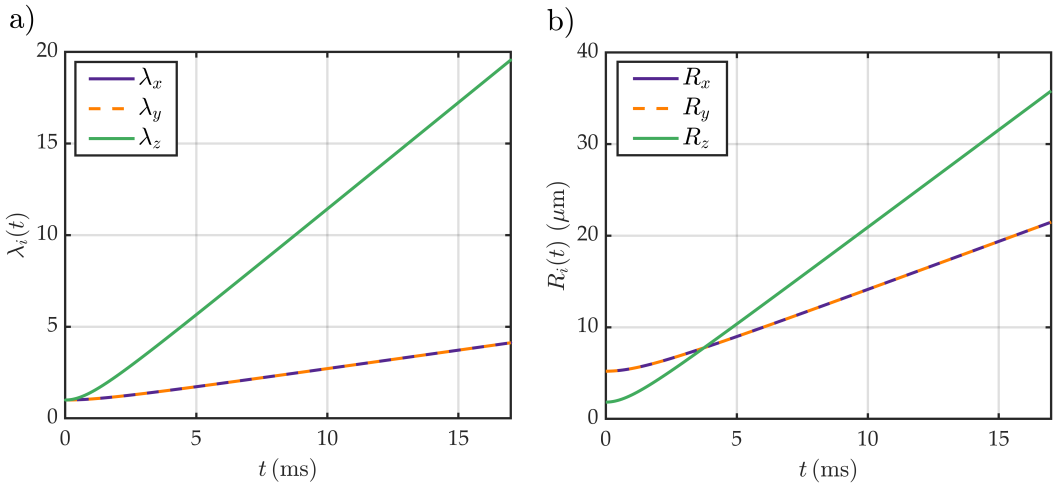

FIGURE 41: Numerical solution of the Castin-Dum scaling equations for the parameters of the study of Holland, et. al. [4]. These parameters were $\omega_x = \omega_y = 2\pi\times 56.25$ Hz and $\omega_z = 2\pi\times 160$ Hz, $N = 4000$ atoms, for times-of-flight up to 17 ms. (a) The evolution of the scaling parameters along each direction $\lambda_x(t), \lambda_y(t), \lambda_z(t)$. (b) The evolution of the Thomas-Fermi radii throughout the expansion using the relation $R_i(t) = \lambda_i(t)\, R_i(0)$, where $R_i(0)$ are the in-trap radii in the Thomas-Fermi approximation. The radius along the z-direction increases faster (due to initial tighter confinement), and an inversion of aspect ratio takes place around 3.7 ms.

Once the Thomas-Fermi radii have been obtained, as in Fig. 41 (b), the full parabolic density profile can be calculated. As an example, Fig. 42 shows cuts through the Castin-Dum density profile for the parameters in the study of Holland et al., which had $\omega_x = \omega_y = 2\pi\times 56.25$ Hz and $\omega_z = 2\pi\times 160$ Hz, with $N = 4000$ atoms, after 17 ms time-of-flight. For comparison, the full 3D numerical solution of the GPE from Sec. 8.1 is also shown. It can be seen that the Castin-Dum theory captures the approximate sizes of the cloud correctly, but predicts a slightly lower peak density, as well as some deviation in the shape. This is likely due to the fact that the cloud was not initially truly in the Thomas-Fermi regime (having a relatively small atom number of $N = 4000$), and so the density profile after TOF is not parabolic but exhibits more of a Gaussian behaviour - as expected when the influence of interactions is smaller. In order to verify this, I have repeated the same calculation but with 10^5 atoms instead of 4000 (25 times more), to put the cloud deeper into the Thomas-Fermi regime. The result is given in Fig. 43, and we can see that the in-trap density matches better the Thomas-Fermi prediction now, and the expanded time-of-flight density also agrees more with the Castin-Dum scaling prediction.

Special Case of a Cigar-Shaped Trap

A common situation is that of a "cigar-shaped" trap, which means the trap has axial symmetry ($\omega_x = \omega_y = \omega_\perp$) and is elongated with large aspect ratio such that $\omega_\perp \gg \omega_z$ (this is typically the situation in atom chip experiments). In this case, Eq. 455 simplifies to two equations, one for each of the transverse

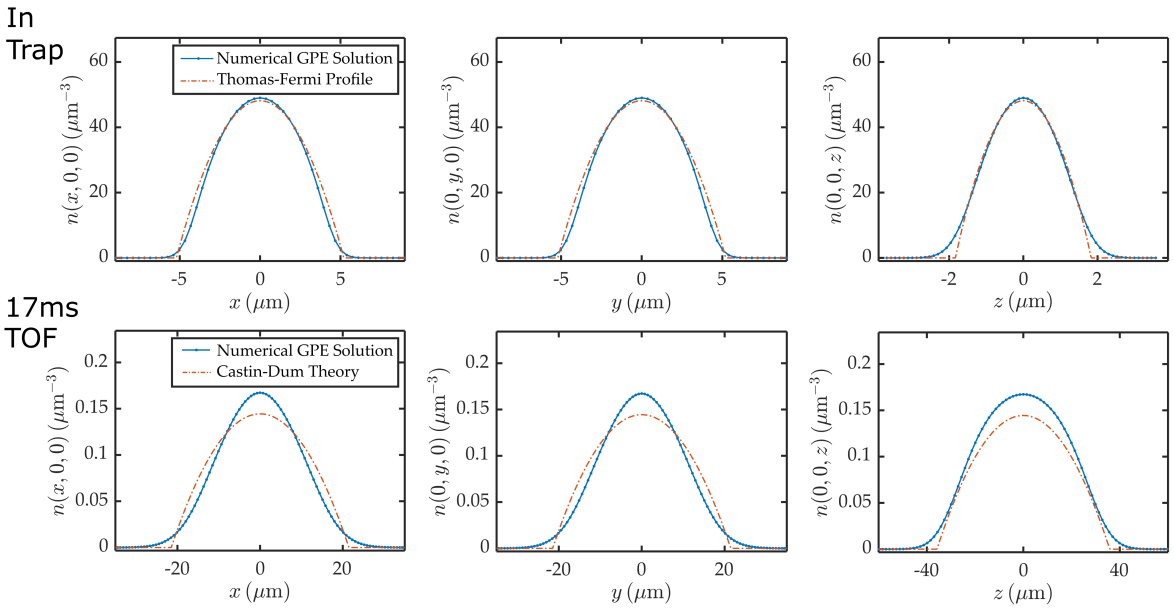

FIGURE 42: Comparison of density profiles between the full numerical 3D GPE solution (solid lines) and the Castin-Dum scaling theory (dashed lines), for the parameters of Holland et al. [4]. Parameters were $\omega_x = \omega_y = 2\pi\times\ 56.25\,\mathrm{Hz}$ and $\omega_z = 2\pi\times\ 160\,\mathrm{Hz}$, with $N = 4000$ atoms. (Top Row) Line cuts through the centre of the density distribution along each direction before switching off the trapping potential, calculated using the imaginary time method for the ground state. The parabolic profile expected for in the Thomas-Fermi regime is shown as the dashed line. (Bottom Row) Line cuts after 17 ms time-of-flight expansion. Although the approximate size of the cloud is captured reasonably well, the profiles deviate from each other - likely due to the cloud not being fully in the Thomas-Fermi regime.

and axial directions

$$\frac{\mathrm{d}^2\lambda_\perp}{\mathrm{d}t^2} = \frac{\omega_\perp^2(0)}{\lambda_\perp^3\lambda_z} \qquad \frac{\mathrm{d}^2\lambda_z}{\mathrm{d}t^2} = \frac{\omega_z^2(0)}{\lambda_\perp^2\lambda_z^2}. \tag{456}$$

By transforming to a time variable which is in units of the initial transverse trap frequency $\tau = \omega_\perp(0)\,t$, and defining the anisotropy parameter $\epsilon = \omega_z(0)/\omega_\perp(0)$, these become

$$\frac{\mathrm{d}^2\lambda_\perp}{\mathrm{d}\tau^2} = \frac{1}{\lambda_\perp^3\lambda_z} \qquad \frac{\mathrm{d}^2\lambda_z}{\mathrm{d}\tau^2} = \frac{\epsilon^2}{\lambda_\perp^2\lambda_z^2}. \tag{457}$$

A cigar-shaped trap has anisotropy parameter $\epsilon \ll 1$, and Ref. [39] presents an analytic solution for this situation by expanding Eq. 457 in powers of ϵ, with the results

$$\lambda_\perp(\tau) = \sqrt{1+\tau^2} \tag{458}$$

$$\lambda_z(\tau) = 1 + \epsilon^2\left[\tau\arctan\tau - \ln\sqrt{1+\tau^2}\right]. \tag{459}$$

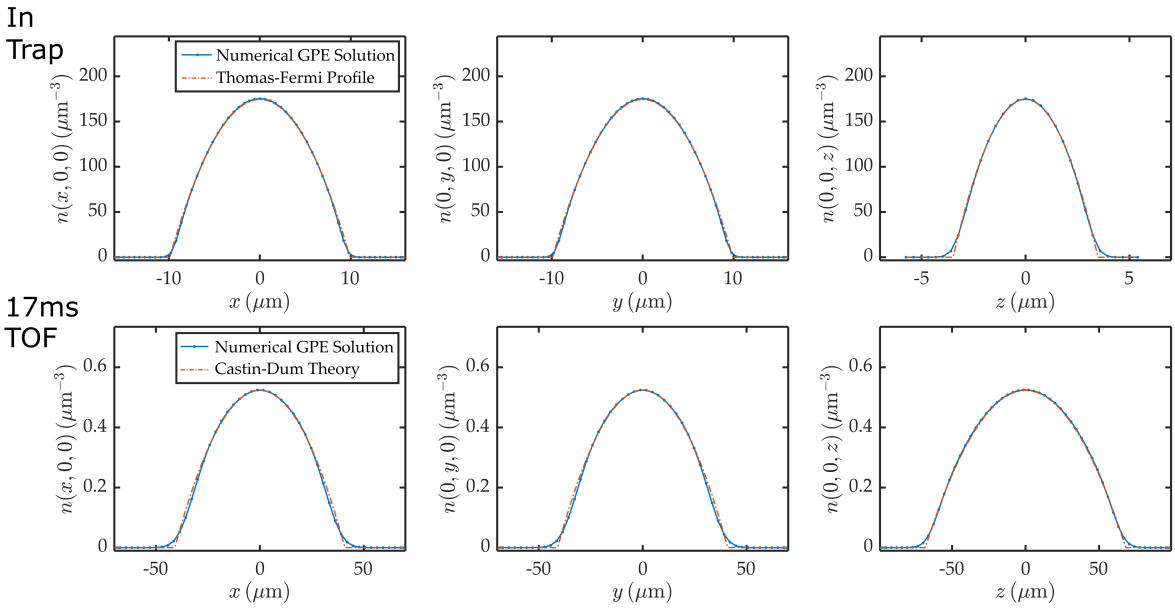

FIGURE 43: Same plots as Fig. 42, except with 10^5 atoms instead of 4000 (a factor of 25 greater. This puts the cloud more deeply in the Thomas-Fermi regime, and so the Castin-Dum scaling theory now better agrees with the true result from the 3D numerical GPE simulation.

8.3 Absorbing Boundary Conditions

In practice, the range of the numerical grid (known as the *support*) is finite, meaning that even though the wavefunction decays towards the edges it will always have some finite value at the grid boundary. The Fourier transform which is used to calculate the derivatives in the split-step method assumes that the function decays to zero at $\pm\infty$ (in fact for the Fourier transform of a function to exist it must be square integrable, otherwise the Fourier integral is infinite), and if this is not satisfied then taking the numerical FFT will lead to reflections at the boundaries (this is a result of the 2π periodicity of the discrete Fourier transform). Any reflections can lead to ripples in the wavefunction density, and so typically the grid is chosen to be large enough that such numerical artifacts do not occur (a good rule of thumb is to take around twice the wavefunction's size for the grid support [12]). Nevertheless, the simulation can be made more robust, and less sensitive to choosing too small a grid support, if absorbing boundary conditions are enforced on the domain for calculation. There are a variety of options for implementing boundary conditions, such as making the trapping potential a complex function with negative imaginary part outside some boundary, which essentially dampens the wavefunction to zero within this region, whilst not affecting the inner computational domain of interest. See Refs. [42, 43] for examples and Refs. [44, 45] for reviews. Such boundary conditions have often been used when numerically solving the Schrodinger equation to study ionisation rates in atoms, since once an electron has been ejected by a laser's electric field you need to make sure that it is removed at infinity, and does not cause reflections at the grid boundary [46–50] (actually, the loss of energy from the system when the electron is absorbed at the boundary is a way to measure the ionisation rate itself).

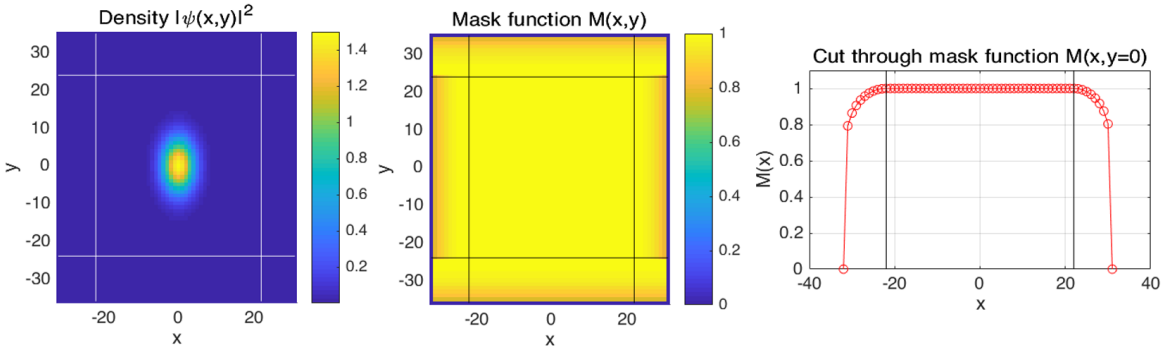

FIGURE 44: Illustration of the mask function in 2D.

In Ref. [51], a simple masking function was compared to a more involved exterior complex scaling method, with the latter shown to be superior. However, the simple mask function still produces reasonable results, and so I will use it here. The same mask function was also used in Ref. [50]. The function is given by

$$M(X) = \cos^{1/8}\left(\frac{|X - x_i|}{d}\frac{\pi}{2}\right), \tag{460}$$

and is shown in Fig. 44 for the 2D case, with the extension to 3D being straightforward. Here, d is the

width of the absorbing region, and x_i is the location of the boundary. The function M reduces from 1 at the inside of the boundary (and therefore does not modify the wavefunction in this region at all) to 0 at the outer edge, and importantly a very smooth onset of the function is guaranteed by the $\cos^{1/8}$ form. Multiplying the wavefunction by this function at each time step attenuates it smoothly to zero at the edge of the numerical grid, and thereby suppresses reflections, as demonstrated in Fig. 45.

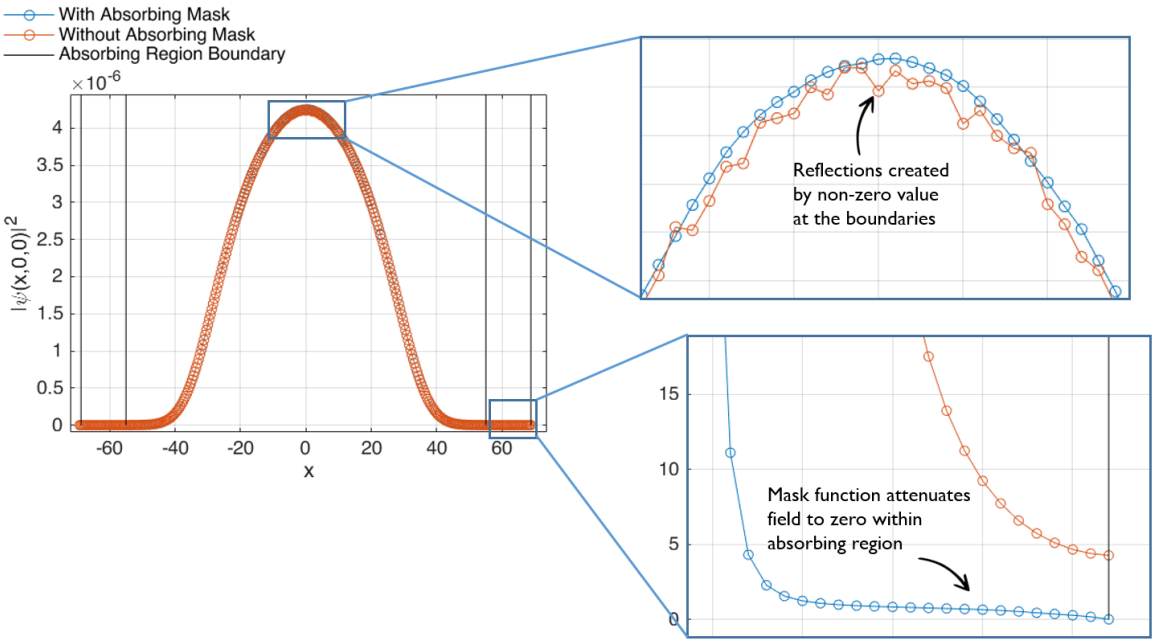

FIGURE 45: Effectiveness of the absorbing boundary mask at preventing unwanted reflections due to non-zero values of the wavefunction at the edges.

8.4 Deuar's Free Expansion Trick

At some point, when calculating larger times-of-flight, the method of dynamically expanding grids described in Sec. 8.1 will reach a limitation that the amount of RAM required becomes too great to store the entire wavefunction in memory anymore. To avoid the large amount of RAM needed to represent the wavefunction after expansion (including the high frequency oscillation in k-space which builds up during free propagation as shown in Fig. 40), the ballistic expansion (i.e. neglecting the effect of interactions) can be carried out with a nice trick from Piotr Deuar - using a sum over FFTs on a scaled lattice [52]. This method allows times-of-flight of many tens of milliseconds (typical of experimental absorption imaging) to be accurately calculated, which would prohibitively require RAM on the order of $\sim$ 100 GB to do brute force with the split-step method.

Deuar's method takes as an input an initial wavefunction on some "small lattice", with (N_x, N_y, N_z) lattice points and grid spacings $(\Delta x, \Delta y, \Delta z)$. It then performs a one-shot expansion out to a final wavefunction on an "expanded lattice", which has larger grid spacings $(\lambda_x \Delta x, \lambda_y \Delta y, \lambda_z \Delta z)$, where λ_i are the lattice expansion factors along each direction, and need to be chosen such that the final wavefunction fits comfortably inside the final expanded lattice. A schematic illustrating the initial

and final lattices is given in Fig. 46. The full equations for the Deuar recipe are given in Eqs.42 of the paper [52], and crucially they rely on the assumption that the region outside the initial grid volume is vacuum (zeros). This essentially allows interpolation of the high frequency oscillation that would otherwise be built up in k-space, without needing to sample all the points directly. A derivation is given in the paper.

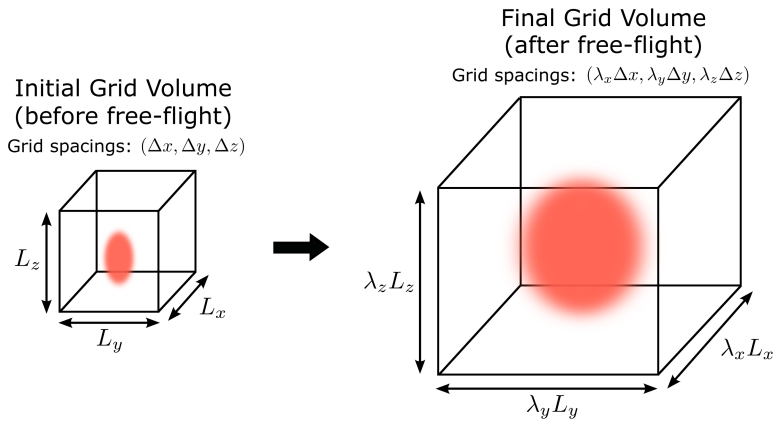

FIGURE 46: Illustration of the lattices for the Deuar free-flight recipe. The final lattice is expanded by a factor of λ_i in each direction, ensuring the final wavefunction fits comfortably into the volume. Crucially, the grid spacings are expanded by the factor λ_i, which means that the number of grid points in both the initial and final lattices are the same - so no additional RAM is rwquired to store the wavefunction.

I have written a function which codes Eqs. 42 of the Deuar paper [52], and it is given in Appendix B. A simple analytic check that we can do on this code is to look at the free expansion of the non-interacting ground state wavefunction in a trap, since there are analytic expressions available in for this situation. The ground state will have a Gaussian density distribution given by

$$n(\mathbf{r}) = N|\psi(\mathbf{r})|^2 = \frac{N}{(\pi)^{3/2} a_x a_y a_z} \exp\left(-\frac{x^2}{a_x^2} - \frac{y^2}{a_y^2} - \frac{z^2}{a_z^2}\right), \tag{461}$$

where $a_i = \sqrt{\hbar/m\omega_i}$ are the harmonic oscillator lengths in each of the three directions, and the wavefunction is normalised to unity. After free expansion under the linear Schrodinger equation, the density distribution is given by Eq. 187 and reproduced here

$$\begin{aligned} n(\mathbf{r},t) = N|\psi(\mathbf{r},t)|^2 = & \frac{N a_x a_y a_z}{\pi^{3/2}\sqrt{\left(a_x^4 + \hbar^2 t^2/m^2\right)\left(a_y^4 + \hbar^2 t^2/m^2\right)\left(a_z^4 + \hbar^2 t^2/m^2\right)}} \times \ldots \\ & \exp\left(-\frac{a_x^2 x^2}{a_x^4 + \hbar^2 t^2/m^2} - \frac{a_y^2 y^2}{a_y^4 + \hbar^2 t^2/m^2} - \frac{a_z^2 z^2}{a_z^4 + \hbar^2 t^2/m^2}\right). \end{aligned} \tag{462}$$

Therefore, the evolution of a wavepacket under free expansion (no interactions) remains Gaussian, but with a RMS width which grows with time along each direction. Figure 47 shows that Deuar's free flight recipe does indeed provide the correct density distribution after some free expansion time.

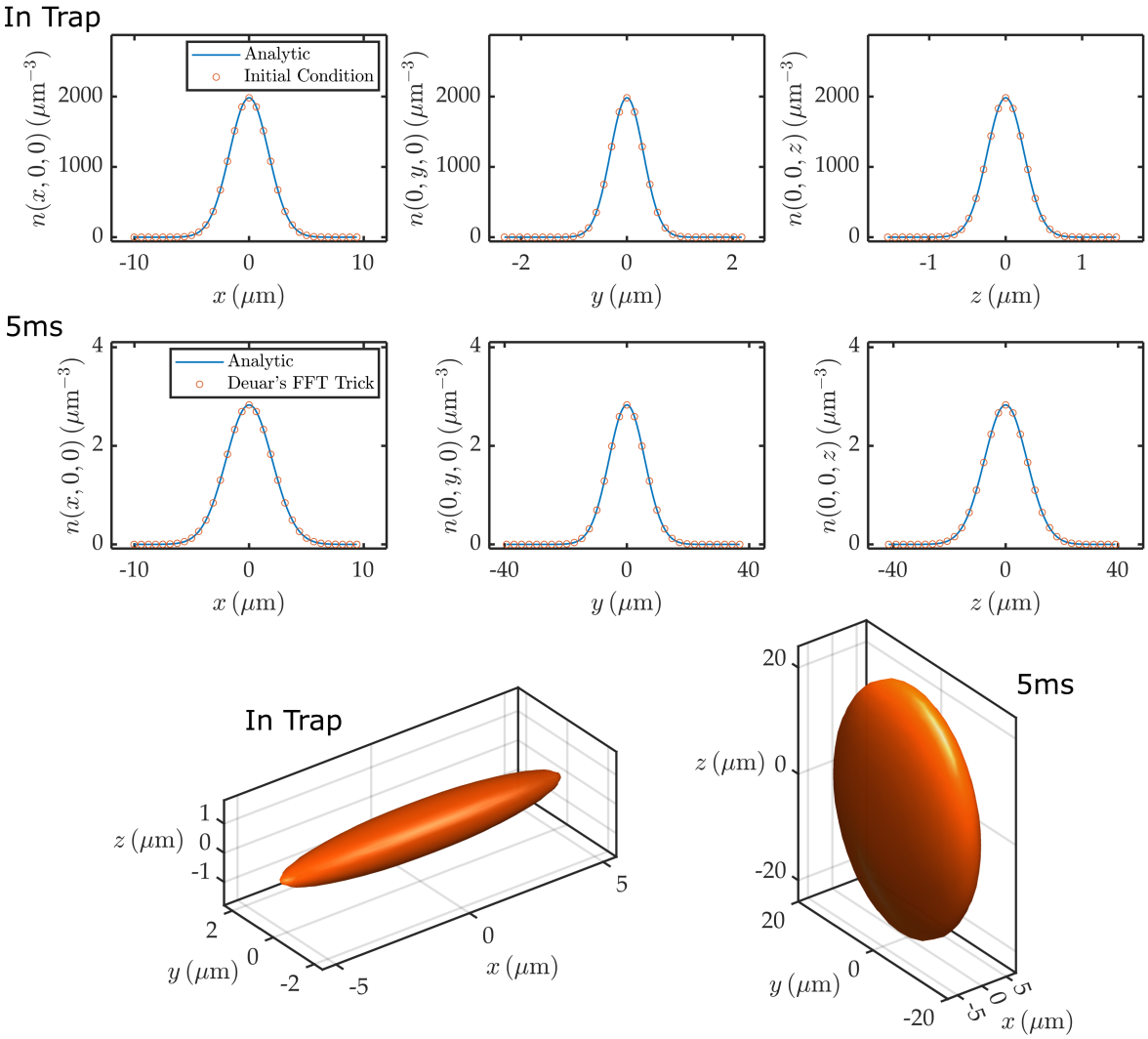

FIGURE 47: Testing Deuar's free flight recipe [52] for a non-interacting wavepacket. (Upper row) The analytic ground state density (solid lines) for a cloud of $N = 4000$ non-interacting atoms in a harmonic trap with frequencies $(\omega_x, \omega_y, \omega_z) = 2\pi \times (20, 600, 1000)$ Hz. The analytic wavefunction is sampled (circles) and used as the input for the free-flight recipe. The sampled wavefunction had $N_x = N_y = N_z = 32$ lattice points, and grid spacings of $(\Delta x, \Delta y, \Delta z) = (0.627, 0.145, 0.097)$ μm. (Middle row) The resulting density after applying the free-flight for 5 ms (circles) using lattice expansion factors $(\lambda_x, \lambda_y, \lambda_z) = (1, 17, 27)$, and comparison with the analytic result (solid lines). (Bottom row) Isosurfaces of where the density reaches 1% of its peak value, to illustrate the approximate size of the cloud before and after expansion. Importantly, despite the large change in scale of the cloud after expansion, the number of grid points (and therefore the RAM required to store the wavefunction) is unchanged from the starting state.

The free-flight recipe is very useful because it can be combined with the Gross-Pitaevskii evolution, as shown in Fig. 48. During time-of-flight, there is a conversion of energy which takes place during the initial phase, when the stored interaction energy in trap is transformed into kinetic energy. Once this process has finished, there is only kinetic energy remaining in the cloud and the system evolves according to the Schrodinger equation - which is exactly what the free-flight recipe performs. In a typical example of an atom chip trap with radial trapping frequencies at least several hundred Hertz, the conversion phase lasts one or two milliseconds, until the interaction energy has reached less than 1% of its initial in-trap value. This stage of the evolution can be calculated with the more computationally-intensive split-step Fourier method for solving the full GPE including interactions. However, continuing next to evolve the GPE up to the desired time-of-flight (which may be several tens of milliseconds) would be very slow and likely run out of RAM very quickly as the grid necessarily expands, since the cloud typically expands by a factor of ~ 15 in each of the two radial directions. The free-flight recipe is instead perfectly suited to calculate this final stage.

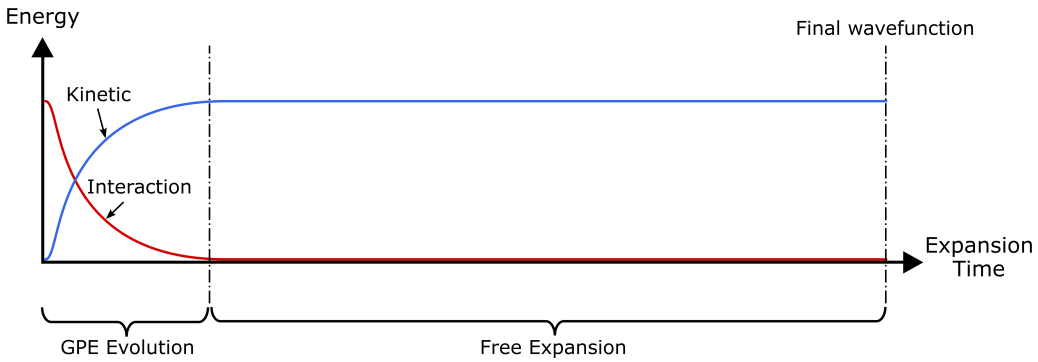

FIGURE 48: Illustration of using the free-flight recipe together with the GPE evolution. For the initial phase of the time-of-flight expansion when the interactions are still significant, the full GPE including the non-linear interaction term must be solved (for example using the split-step Fourier method). During this phase, stored interaction energy is converted to kinetic energy (potential energy is zero as soo as the trap is extinguished), and the number of grid points must increase significantly as the cloud expands in size and a faster k-space oscillation builds up. However, once the interaction energy becomes negligible (e.g. around 0.1% of its initial in-trap value), the expansion becomes ballistic (non-interacting) and the cloud evolves according to the Schrodinger equation. This phase is conveniently calculated using Deuar's free flight recipe [52].

9 Expansion of Phase Fluctuating BECs

In 1D quasicondensates and elongated 3D condensates, the phase distribution ϕ of the cloud in the trap is not constant as in simulations in previous sections, but is known to fluctuate due to the fact that the cloud is at some finite temperature [53–55]. The phase coherence is described by the first order correlation function, which for a quasicondensate with homogeneous density n_{1D} is calculated using Bogoliubov theory and is given by [56, 57]

$$g^{(1)}(x-x') = \frac{\langle \hat{\psi}^\dagger(x)\hat{\psi}(x')\rangle}{n_{1D}} = \exp\left(-\frac{1}{2}\langle\left[\phi(x)-\phi(x')\right]^2\rangle\right) = e^{-|x-x'|/\lambda_T}, \tag{463}$$

where $\phi(x)-\phi(x')$ is the phase difference between two points x and x' and λ_T is the thermal phase coherence length

$$\lambda_T = \frac{2\hbar^2 n_{1D}}{m k_B T}. \tag{464}$$

Therefore, since the density n_{1D} is constant, then λ_T is also constant, and the average square of the fluctuations is linear with distance

$$\langle\left[\phi(x)-\phi(x')\right]^2\rangle = \frac{2|x-x'|}{\lambda_T}, \tag{465}$$

meaning that the first order coherence function decays *exponentially* with distance.

If instead the cloud is in an elongated external harmonic trap (non-homogeneous density), where x is the long direction, and is in the 3D Thomas-Fermi regime (i.e. $\mu \gg \hbar\omega_\perp, \hbar\omega_x$), the phase can still vary along the long direction due to low-energy excitations with wavelengths which are longer that the radial size but shorter than the axial size - therefore leading to 1D behaviour [53]. In this case, a single realisation of a phase profile for a BEC in trap with Thomas-Fermi radii $R_\perp, R_x$ can be can calculated using [53, 54]

$$\phi(x) = \sum_{j=1}^{\infty} \sqrt{\frac{(j+2)(2j+3)\, g_{3D}}{4\pi R_\perp^2 R_x \varepsilon_j (j+1)}}\, P_j^{(1,1)}\left(\frac{x}{R_x}\right) \frac{1}{2}\left(\alpha_j + \alpha_j^*\right), \tag{466}$$

where $\varepsilon_j = \hbar\omega_x\sqrt{j(j+3)/4}$ is the spectrum of low-energy axial excitations, and $P_j^{(1,1)}$ are Jacobi polynomials of order j. The complex number α_j and its complex conjugate α_j^* are chosen randomly such that the statistical properties over all choices reproduce the required condition $\langle|\alpha_j|^2\rangle = N_j$, where $N_j = \left[\exp\left(\varepsilon_j/k_BT\right)-1\right]^{-1}$ is the occupation number of mode j. This whole procedure is relying on the fact that the phase will vary randomly in a given realisation, with the amount of fluctuation determined by the cloud's temperature T.

9.1 Generating Petrov/Dettmer Phase Profiles

The first thing is to draw the random variables α_j and α_j^* correctly. In Matlab, the function `randn` generates a real random number drawn from a normal distribution with mean = zero and standard

deviation = 1. We can generate `M` random complex numbers as follows

```
M = 100;  % Number of samples
for j = 1:M
   a_j(j) = randn(1,1) + 1i*randn(1,1); % a complex random number
end
```

but then the quantity $\langle|\alpha_j|^2\rangle$ is equal to 2:

```
sum( abs(a_j).^2 )/M = 2.01
```

This is because when adding two random numbers together as above in order to generate a complex one, their variances add together. In order to obtain a complex random number with $\langle|\alpha_j|^2\rangle$ equal to some number N_j, we need to first remove a factor of $\sqrt{2}$ from each part, and then scale accordingly:

```
M = 100;  % Number of samples
N_j = 28; % Choose some required property
for j = 1:M
   a_j(j) = sqrt(N_j) * ( randn(1,1) + 1i*randn(1,1) ) / sqrt(2); % a complex random number
end
```

Now the property of the random number is correct:

```
sum( abs(a_j).^2 )/M = 28.005
```

In fact, what we need to be able to calculate Eq. 466 is the quantity $\left(\alpha_j + \alpha_j^*\right)/2$. We never actually need to use α_j or α_j^* on their own, but only ever use their sum. Since for any complex number z we have $z + z^* = 2\text{Re}(z)$, which is always real. Therefore, we should be able to generate a *real* random number instead - as long as the statistical properties remain the same. The following code and Fig. 49 (a) illustrates that it is equivalent to choose a real random variable β_j

```
M = 10000;
N = 28;
for j = 1:M
   a_j(j) = sqrt(N) * ( randn(1,1) + 1i*randn(1,1) )  /sqrt(2);
   beta_j(j) = sqrt(N) * randn(1,1) / sqrt(2); % A real random variable
end
histogram( ( a_j + conj(a_j) )/2 )
histogram( beta_j )
```

Next we require the Jacobi polynomials $P_j^{(1,1)}$ of order j. The code for this is given in Appendix C, and the first four orders are shown in Fig. 49(b).

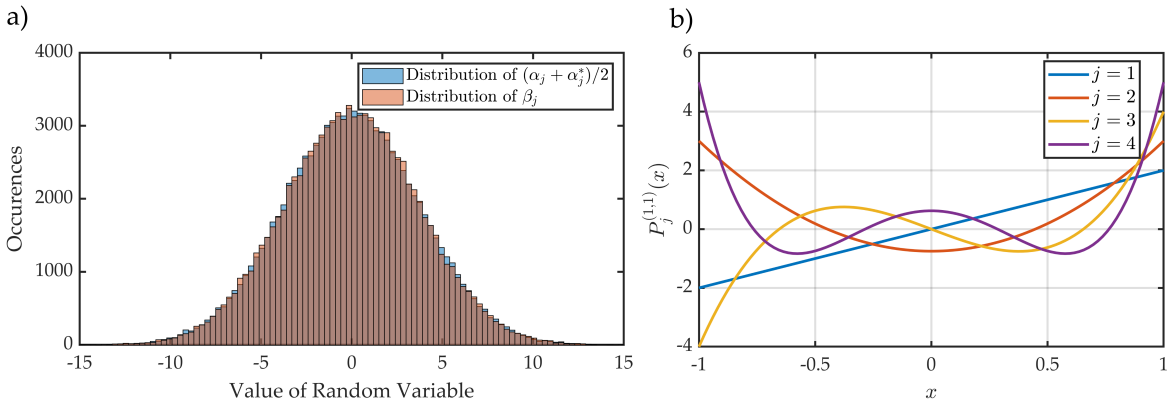

FIGURE 49: Some of the components needed for calculating the phase profiles according to Petrov's formula. (a) Choosing a real random variable β_j is equivalent to using $(\alpha_j + \alpha_j^*)/2$ where α_j is a complex number. (b) First four orders of the Jacobi polynomials $P_j^{(1,1)}(x)$.

The following code calculates many realisations of the random phases using Eq. 466 in a fully vectorized way:

```
omega_ax = 2*pi*20;    % Axial harmonic trap frequency
omega_rad = 2*pi*500; % Radial harmonic trap frequency
N = 1e5;               % BEC atom number
T = 300e-9;            % BEC temperature [K]
nProfiles = 1000;      % Number of profiles to simulate
hbar = 1.055e-34;      % Reduced Plancks constant
g3D = 5.0763e-51;      % 3D interaction coupling constant
kB = 1.38e-23;         % Boltzmanns constant
m = 1.44e-25;          % Rb87 mass

% Obtain Jacobi polynomials (can be precalculated and loaded if grid doesn't change)
Nx = 1024;                          % Number of axial (x) grid points
dx = 0.0018;                        % dx = floor(0.99*2/(Nx-1) *100000)/100000
xsc  = -(Nx*dx/2):dx:(Nx*dx/2-dx); % Dimensionless axial coordinate,in units of TF radius
jmax = 2000;                        % Number of Bogoliubov terms to include in summation
P_j = zeros(jmax,Nx);
for jj = 1:jmax
    P_j(jj,1:Nx) = jacobpol(1,1,jj,xsc); % Jacobi polynomial P_j^(1,1)(x/Rx)
end

% Calculate phases according to Petrov/Dettmer
mu = ( 15*sqrt(2)/(32*pi) * N * g3D *m^(3/2) * omega_ax*omega_rad^2 )^(2/5);
R0_rad = sqrt( 2*mu/(m*omega_rad^2) );  % Radial Thomas-Fermi radius in-situ
R0_ax  = sqrt( 2*mu/(m*omega_ax^2) );   % Axial Thomas-Fermi radius in-situ
phi = calculate_Dettmer_phases(omega_ax,R0_ax,R0_rad,T,P_j,nProfiles,hbar,g3D,kB);

% Analyse central region of phases
[dphi_sq_av, dphi_sq_av_mean, dx_vec] = calculate_dphi_sq_av( phi(:,abs(xsc) < 0.2), ...
                                                              xsc(abs(xsc) < 0.2));
```

```
g1_uniform = exp( -1/2 * 32*mu*kB*T/(15*N*(hbar*omega_ax)^2) * dx_vec  );

figure;hold all;
plot(dx_vec, g1_uniform,'LineStyle','-.')
plot(dx_vec, exp(-1/2*dphi_sq_av_mean) )

function phi = calculate_Dettmer_phases(omega_ax,R0_ax,R0_rad,T,P_j,nProfiles,hbar,g3D,kB)

j_max = size(P_j,1);
j_vec = (1:j_max)';
e_j = hbar*omega_ax*sqrt( j_vec.*(j_vec+3)/4 );  % Energy spectrum value
N_j = 1./( exp(e_j.'/(kB*T)) - 1 );              % Occupation number for given temperature
R = sqrt(N_j/2).*randn(nProfiles,j_max);         % Real random variable

prefac1 = sqrt( (j_vec+2).*(2*j_vec+3)*g3D ./ ...
                ( 4*pi*R0_rad^2*R0_ax*e_j.*(j_vec+1) ) );

phi = R * (prefac1 .* P_j);

end

function [dphi_sq_av, dphi_sq_av_mean, dx_vec] = calculate_dphi_sq_av(phi, x)

if size(phi,1) ~= length(x)
   phi = phi.';
end

Nx = length(x);
dx = x(2) - x(1);
num_profiles = size(phi,2);

dphi_sq_av = zeros(Nx-1,num_profiles);
for p = 1:num_profiles
    for n = 1:(Nx-1)

        temp = 0;
        for k = 1:(Nx-n)
            temp = temp + ( phi(k,p) - phi(k+n,p) )^2;
        end

        dphi_sq_av(n,p) = temp/(Nx-n);

    end
end

dphi_sq_av_mean = mean(dphi_sq_av,2);
dx_vec = (1:(Nx-1))*dx;

end
```

The resulting phases for a cloud with $(\omega_\perp, \omega_x) = 2\pi \times (500, 20)$ Hz, $T = 300\,\mathrm{nK}$, and $N = 10^5$ atoms are shown in Fig. 50 (b). The 1D density profile (radially-integrated density) for a cloud in the Thomas-Fermi regime is found by integrating over the 3D parabola in Eq. 220, resulting in

$$n_{1\mathrm{D}}(x) = \iint_{-\infty}^{+\infty} n(x,y,z)\,\mathrm{d}y\,\mathrm{d}z \tag{467}$$

$$= \iint_{-\infty}^{+\infty} \frac{15N}{8\pi R_x R_y R_z}\left(1 - \frac{x^2}{R_x^2} - \frac{y^2}{R_y^2} - \frac{z^2}{R_z^2}\right)\mathrm{d}y\,\mathrm{d}z \tag{468}$$

$$= \frac{15N}{16R_x}\left(1 - \frac{x^2}{R_x^2}\right)^2 \tag{469}$$

This function for the trap parameters above is shown in Fig. 50 (a). The first order correlation function $g^{(1)}(x)$ is calculated from the average square of the deviation of the phase for various separations, and is shown in Fig. 50 (c). If the cloud was homogeneous with a constant density equal to the peak density of the Thomas-Fermi profile, then the correlation function would be expected to decay exponentially with a decay constant

$$\lambda_T = \frac{2\hbar^2 n_{1\mathrm{D}}(x=0)}{mk_BT} = \frac{2\hbar^2}{mk_BT} \times \frac{15N}{16R_x} = \frac{15\hbar^2 N}{8mk_BTR_x}, \tag{470}$$

and such a curve is shown by the dashed line in Fig. 50 (c). Note that in the Thomas-Fermi regime, we have $R_x = \sqrt{2\mu_{3\mathrm{D}}/m\omega_x^2}$, and so we can rearrange this to write $mR_x = 2\mu_{3\mathrm{D}}/R_x\omega_x^2$, which we can use to rewrite the thermal coherence length as

$$\lambda_T = \frac{15\hbar^2\omega_x^2 N R_x}{16k_BT\mu_{3\mathrm{D}}} \tag{471}$$

and the average square of the phase fluctuations is expected to be

$$\left\langle\left[\phi(x) - \phi(x')\right]^2\right\rangle = \frac{2|x-x'|}{\lambda_T} = \frac{32k_BT\mu_{3\mathrm{D}}}{15\hbar^2\omega_x^2 N}\frac{|x-x'|}{R_x}, \tag{472}$$

which is an expression that can be found in Refs. [53, 55]. We can see in Fig. 50 (c) that if only the central region of the phases is considered, which corresponds to the region over which the density is approximately equal to the peak value [shown by the shaded bands in (a) and (b)], then the correlation function does indeed agree well with that expected for a cloud with homogeneous density. If instead the full phases profiles are considered for calculating the correlation function, the figure shows that the coherence decays much faster than the exponential curve. This is because if we include the parts of the cloud which are further from the center then the density decreases there, leading to a smaller value for λ_T in those regions of lower density. This in turn gives rise to greater variation in the phase - as seen at the edges of the phase profiles in Fig. 50 (b) - which overall decreases the average coherence further than that expected for a homogeneous gas with the same peak density.

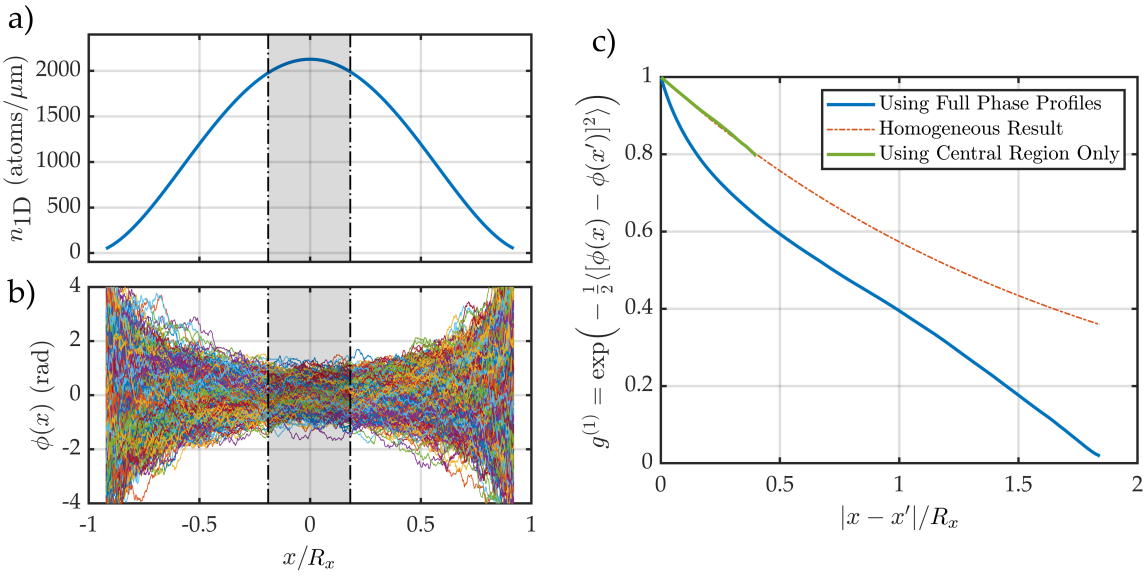

FIGURE 50: Phase fluctuations according to Petrov's formula, which comes from solving the Bogoliubov equations for a cloud in the Thomas-Fermi regime in an elongated harmonic trap. (a) The 1D density profile, calculated by integrating a 3D parabolic distribution, for the parameters $(\omega_\perp, \omega_x) = 2\pi \times (500, 20)$ and $N = 10^5$ atoms. (b) 1000 individual realisations of the phase calculated using Eq. 466 for $T = 300\,\mathrm{nK}$. (c) First order correlation function (a measure of in-trap phase coherence) for the full phase profiles in (b), as well as when only using the central region of the cloud ($|x/R_x| < 0.2$) indicated by the shaded bounds in (a) and (b). The expected exponentially-decaying result for a homogeneous cloud with the same peak density is given by the dashed line.

10 Gross-Pitaevskii Equation in 3D Cylindrical Coordinates

The Gross-Pitaevskii Equation (GPE) can be solved in full 3D cartesian coordinates, as previously described, using either spectral methods (performing 3D Fourier transforms with FFT) or finite difference methods. However, the problem becomes extremely computationally expensive when moving from 1D to 3D. In our experiment, the magnetic potentials generated by the atom chip are most often cylindrically symmetric. If we make use of this symmetry, then the full 3D problem is reduced from 3D to 2D. In cylindrical coordinates, with $\rho = \sqrt{x^2 + y^2}$ denoting the radial direction and z denoting the axial direction, the wavefunction can be described by a complex two-dimensional matrix at each time $\psi(\rho, z, t)$. In cylindrical coordinates, the GPE is given by

$$i\hbar \frac{\partial \psi(\rho, z, t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\rho, z) + g_{3\text{D}}|\psi(\rho, z, t)|^2 \right)\psi(\rho, z, t), \tag{473}$$

where the external potential is cylindrically symmetric since $\omega_x = \omega_y$, and can be written as

$$V_{\text{ext}} = \frac{1}{2}m\left(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2\right) \tag{474}$$

$$= \frac{1}{2}m\left(\omega_\rho^2 \rho^2 + \omega_z^2 z^2\right) \tag{475}$$

$$= \frac{1}{2}m\omega_\rho^2\left(\rho^2 + \lambda^2 z^2\right), \tag{476}$$

where $\omega_z = \lambda\omega_\rho$, i.e λ expresses the aspect ratio of the trap as a ratio of the two trapping frequencies. The Laplacian is now expressed in cylindrical coorindinates

$$\nabla^2 = \frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}\right) + \frac{1}{\rho^2}\frac{\partial^2}{\partial\theta^2} + \frac{\partial^2}{\partial z^2}. \tag{477}$$

Since we are requiring that the system has cylindrical symmetry, the problem should not depend on the angle θ around the z axis, and so the derivative with respect to θ is zero. We can use the chain rule to expand the radial derivative

$$\nabla^2 = \frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}\right) + \frac{\partial^2}{\partial z^2} \tag{478}$$

$$= \frac{1}{\rho}\left[\rho\frac{\partial^2}{\partial\rho^2} + \frac{\partial}{\partial\rho}\right] + \frac{\partial^2}{\partial z^2} \tag{479}$$

$$= \frac{\partial^2}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial}{\partial\rho} + \frac{\partial^2}{\partial z^2}. \tag{480}$$

The GPE in cylindrical coordinates (in SI units) can finally be written fully as

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2\psi}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial\psi}{\partial\rho} + \frac{\partial^2\psi}{\partial z^2}\right) + V_{\text{ext}}\psi + g_{3\text{D}}|\psi|^2\psi, \tag{481}$$

where the writing of the dependent coordinates has been dropped for brevity. In this form, the modulus squared of the wavefunction must be normalised to the total atom number, which in cylindrical coordinates means

$$2\pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} |\psi|^2 \rho \, \mathrm{d}\rho \, \mathrm{d}z = N_{\text{tot}} \tag{482}$$

where N_{tot} is the total number of atoms.

10.1 Dimensionless Form

In the GPE, there are factors of mass m and reduced Plank constant $\hbar$, which are very small numbers. In addition, the length scales associated with a BEC are often in micrometers, and the timescales are in milliseconds. When working with such small numbers, there is a stricter requirement on the machine precision of the PC on which you are doing the calculations, the possibility of rounding errors, as well as potentially the need to store data as more precise data types (e.g. double float instead of single). This was perhaps more of a problem in the past, and is less of an issue with modern machines, but nevertheless it is still a good idea to use scales which are suitably adapted to the problem being studied. For this reason, the GPE is often written in terms of scaled lengths, scaled times, and scaled energies for the purposes of numerically solving the equation - with the results being converted back to SI units at the end if desired.

One of the most common approaches is to scale all the length dimensions as

$$\tilde{\rho} = \rho/l \qquad \text{and} \qquad \tilde{z} = z/l, \tag{483}$$

where the tilde notation denotes the scaled version of the variable. In principle, the length l can be chosen as anything, for example $l = 1\mu\mathrm{m}$ would be a valid choice, but it is usually chosen to be the harmonic oscillator length in the radial direction

$$l = \sqrt{\frac{\hbar}{m\omega_\rho}}. \tag{484}$$

For a trap frequency of $\omega_\rho = 2\pi \times 1\mathrm{kHz}$ with rubidium-87, this length takes the value $l = 0.34\mu\mathrm{m}$. Similarly, the time coordinate is usually scaled by the radial trap frequency, which provides a time scale for any dynamics

$$\tilde{t} = \omega_\rho \, t. \tag{485}$$

Finally the wavefunction is scaled as

$$\tilde{\psi}(\tilde{\rho}, \tilde{z}, \tilde{t}) = \sqrt{l^3} \, \psi(\rho, z, t), \tag{486}$$

where the factor of $\sqrt{l^3}$ is a reflection of the fact that $|\psi|^2$ is a density with units $[\mathrm{m}^{-3}]$. To convert the GPE to dimensionless variables, it is simply a case of plugging in the scaled quantities and evaluating them term by term. For example, the time derivative of the left hand side can be evaluated by the

chain rule

$$\frac{\partial \psi}{\partial t} = \frac{\partial \psi}{\partial \tilde{t}} \frac{\partial \tilde{t}}{\partial t}, \tag{487}$$

and since

$$\tilde{t} = \omega_\rho \, t \implies \frac{\partial \tilde{t}}{\partial t} = \omega_\rho \tag{488}$$

this becomes

$$\begin{aligned}
\frac{\partial \psi}{\partial t} &= \omega_\rho \frac{\partial \psi}{\partial \tilde{t}} && (489)\\
&= \omega_\rho \frac{\partial}{\partial \tilde{t}} \left(\frac{1}{\sqrt{l^3}} \tilde{\psi} \right) && (490)\\
&= \frac{\omega_\rho}{\sqrt{l^3}} \frac{\partial \tilde{\psi}}{\partial \tilde{t}}. && (491)
\end{aligned}$$

We can follow a similar procedure to evaluate the spatial derivatives, for example

$$\begin{aligned}
\frac{\partial^2 \psi}{\partial \rho^2} &= \frac{\partial}{\partial \rho} \left(\frac{\partial \psi}{\partial \rho} \right) && (492)\\
&= \frac{\partial}{\partial \rho} \left(\frac{\partial \psi}{\partial \tilde{\rho}} \frac{\partial \tilde{\rho}}{\partial \rho} \right) && (493)\\
&= \frac{\partial}{\partial \rho} \left(\frac{\partial \psi}{\partial \tilde{\rho}} \frac{1}{l} \right) && (494)\\
&= \frac{1}{l} \frac{\partial}{\partial \rho} \left(\frac{\partial \psi}{\partial \tilde{\rho}} \right) && (495)\\
&= \frac{1}{l} \frac{\partial}{\partial \tilde{\rho}} \left(\frac{\partial \psi}{\partial \tilde{\rho}} \right) \frac{\partial \tilde{\rho}}{\partial \rho} && (496)\\
&= \frac{1}{l} \frac{\partial}{\partial \tilde{\rho}} \left(\frac{\partial \psi}{\partial \tilde{\rho}} \right) \frac{1}{l} && (497)\\
&= \frac{1}{l^2} \frac{\partial^2 \psi}{\partial \tilde{\rho}^2} && (498)\\
&= \frac{1}{l^2} \frac{\partial^2}{\partial \tilde{\rho}^2} \left(\frac{1}{\sqrt{l^3}} \tilde{\psi} \right) && (499)\\
&= l^{-7/2} \frac{\partial^2 \tilde{\psi}}{\partial \tilde{\rho}^2}. && (500)
\end{aligned}$$

A similar thing can be done with the remaining spatial derivative terms, resulting in

$$\frac{1}{\rho} \frac{\partial \psi}{\partial \rho} = l^{-7/2} \frac{1}{\tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \qquad \text{and} \qquad \frac{\partial^2 \psi}{\partial z^2} = l^{-7/2} \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2}. \tag{501}$$

The external potential term becomes simply

$$V_{\text{ext}} \, \psi = \frac{1}{\sqrt{l^3}} \, V_{\text{ext}} \, \tilde{\psi}. \tag{502}$$

Finally the non-linear interaction term is

$$g_{3D}|\psi|^2\psi = g_{3D}\left|\frac{1}{\sqrt{l^3}}\tilde{\psi}\right|^2\frac{1}{\sqrt{l^3}}\tilde{\psi} = l^{-9/2}\,g_{3D}|\tilde{\psi}|^2\tilde{\psi}. \tag{503}$$

All of these converted terms can now be inserted into the cylindrical version of the GPE

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\left(\frac{\partial^2\psi}{\partial\rho^2}+\frac{1}{\rho}\frac{\partial\psi}{\partial\rho}+\frac{\partial^2\psi}{\partial z^2}\right)+V_{\text{ext}}\psi+g_{3D}|\psi|^2\psi \tag{504}$$

$$\overset{x\to\tilde{x}}{\Longrightarrow}\; i\hbar\frac{\omega_\rho}{\sqrt{l^3}}\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{\hbar^2}{2m}\frac{1}{\sqrt{l^7}}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2}+\frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)+\frac{1}{\sqrt{l^3}}V_{\text{ext}}\,\tilde{\psi}+\frac{1}{\sqrt{l^9}}\,g_{3D}|\tilde{\psi}|^2\tilde{\psi}. \tag{505}$$

Now inserting the expression $l=\sqrt{\hbar/m\omega_\rho}$, dividing both sides of the equation by $\left(\hbar m^3\omega_\rho^7\right)^{1/4}$, and using the expression for the 3D interaction constant $g_{3D}=4\pi\hbar^2a_s/m$, we finally obtain the scaled form of the GPE in cylindrical coordinates

$$i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2}+\frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}+\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)+\frac{V_{\text{ext}}}{\hbar\omega_\rho}\tilde{\psi}+\mathcal{G}|\tilde{\psi}|^2\tilde{\psi}, \tag{506}$$

where we have defined the strength of the non-linear term in terms of the scattering length a_S with the parameter

$$\mathcal{G}=\frac{4\pi a_s N_{\text{tot}}}{l}. \tag{507}$$

In the scaled GPE, we can see that the external potential term has been scaled into units of $\hbar\omega_\rho$, the harmonic oscillator energy level spacing. The atom number N_{tot} has also been included in the non-linear term, which changes the normalisation such that the modulus squared of the wavefunction must integrate to unity instead (note that changing to scaled cooridnates doesn't change the normalisation because all the factor of l cancel)

$$2\pi\int_{-\infty}^{+\infty}\int_{0}^{+\infty}|\tilde{\psi}|^2\tilde{\rho}\,\mathrm{d}\tilde{\rho}\,\mathrm{d}\tilde{z}=1. \tag{508}$$

If the external potential is harmonic (and cylindrically symmetric), this term can be written as

$$\frac{V_{\text{ext}}}{\hbar\omega_\rho}\,\tilde{\psi} = \frac{1}{\hbar\omega_\rho}\left[\frac{1}{2}m\omega_\rho^2\left(\rho^2+\lambda^2z^2\right)\right]\tilde{\psi} \tag{509}$$

$$= \frac{m\omega_\rho}{2\hbar}\left[(l\tilde{\rho})^2+\lambda^2\,(l\tilde{z})^2\right]\tilde{\psi} \tag{510}$$

$$= \frac{m\omega_\rho}{2\hbar}l^2\left(\tilde{\rho}^2+\lambda^2\tilde{z}^2\right)\tilde{\psi} \tag{511}$$

$$= \frac{1}{2}\left(\tilde{\rho}^2+\lambda^2\tilde{z}^2\right)\tilde{\psi} \tag{512}$$

$$= \tilde{V}_{\text{ext}}\tilde{\psi}. \tag{513}$$

and the GPE for a harmonic external trapping potential can be written as

$$\boxed{i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -\frac{1}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right) + \frac{1}{2}\left(\tilde{\rho}^2 + \lambda^2\tilde{z}^2\right)\tilde{\psi} + \mathcal{G}|\tilde{\psi}|^2\tilde{\psi}} \tag{514}$$

10.1.1 Dimensionless Energy Expressions

The energy of the system is obtained by taking the expectation value of the Hamiltonian operator, which in the scaled coordinates we can write as

$$\tilde{E} = \langle\tilde{\psi}|\tilde{\hat{H}}|\tilde{\psi}\rangle \tag{515}$$

where the Hamiltonian operator is the sum of the kinetic, potential, and interaction contributions

$$\tilde{\hat{H}} = \tilde{\hat{H}}_{\text{kin}} + \tilde{\hat{H}}_{\text{pot}} + \tilde{\hat{H}}_{\text{int}} \tag{516}$$

$$= \underbrace{-\frac{1}{2}\left(\frac{\partial^2}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}} + \frac{\partial^2}{\partial\tilde{z}^2}\right)}_{\text{kinetic}} + \underbrace{\tilde{V}_{\text{ext}}}_{\text{potential}} + \underbrace{\frac{1}{2}\mathcal{G}|\tilde{\psi}|^2}_{\text{interaction}}. \tag{517}$$

Note that when evaluating the energy of the system, an additional factor of $1/2$ must be included as above in the interaction energy term, to ensure interaction energy between two particles is not counted twice (see, e.g. Dalfovo - *Theory of Bose-Einstein condensation in trapped gases*, or Castin - *Bose-Einstein condensates in atomic gases: simple theoretical results*). This factor is not included when solving the time-dependent equation. The total energy is given by the sum of the individual contributions

$$\tilde{E} = \tilde{E}_{\text{kin}} + \tilde{E}_{\text{pot}} + \tilde{E}_{\text{int}}. \tag{518}$$

Kinetic Energy Term

Now we will evaluate the kinetic energy contribution to the total energy. The expectation bra-ket is evaluated with an integral over all space, using the wavefunction and its complex conjugate, given by

$$\tilde{E}_{\text{kin}} = \langle\tilde{\psi}|\tilde{\hat{H}}_{\text{kin}}|\tilde{\psi}\rangle \tag{519}$$

$$= 2\pi\int_{-\infty}^{+\infty}\int_{0}^{+\infty}\tilde{\psi}^*\left\{-\frac{1}{2}\left(\frac{\partial^2}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}} + \frac{\partial^2}{\partial\tilde{z}^2}\right)\right\}\tilde{\psi}\,\tilde{\rho}\,\mathrm{d}\tilde{\rho}\,\mathrm{d}\tilde{z} \tag{520}$$

$$= -\pi\int_{-\infty}^{+\infty}\int_{0}^{+\infty}\tilde{\psi}^*\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}} + \frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\right)\tilde{\rho}\,\mathrm{d}\tilde{\rho}\,\mathrm{d}\tilde{z} \tag{521}$$

$$= -\pi\int_{-\infty}^{+\infty}\int_{0}^{+\infty}\left(\tilde{\psi}^*\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2}\tilde{\rho} + \tilde{\psi}^*\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}} + \tilde{\psi}^*\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2}\tilde{\rho}\right)\mathrm{d}\tilde{\rho}\,\mathrm{d}\tilde{z} \tag{522}$$

$$= -\pi \int_{-\infty}^{+\infty} \left[\int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{\rho}^2} \tilde{\rho} \, \mathrm{d}\tilde{\rho} + \int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \, \mathrm{d}\tilde{\rho} \right] \mathrm{d}\tilde{z} - \pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2} \tilde{\rho} \, \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z}. \tag{523}$$

The first term can be carried out using integration by parts to give

$$\int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{\rho}^2} \tilde{\rho} \, \mathrm{d}\tilde{\rho} = \left[\tilde{\psi}^* \tilde{\rho} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right]_{0}^{+\infty} - \int_{0}^{+\infty} \frac{\partial}{\partial \tilde{\rho}} \left(\tilde{\psi}^* \tilde{\rho} \right) \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \, \mathrm{d}\tilde{\rho}. \tag{524}$$

We can now here set a boundary condition to require that the wavefunction and its derivative vanish at $\rho = +\infty$ (even though ofcourse the wavefunction itself does *not* vanish at the origin. In addition, because the system has cylindrical symmetry, we require

$$\left. \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right|_{\tilde{\rho}=0} = 0, \tag{525}$$

which ensures the symmetry, ensuring that there is a turning point at $\tilde{\rho} = 0$, and that $\tilde{\psi}(-\tilde{\rho}) = \tilde{\psi}(\tilde{\rho})$. So, the first term in Eq. () is zero, and we can also evaluate the second term using the chain rule

$$\int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{\rho}^2} \tilde{\rho} \, \mathrm{d}\tilde{\rho} = - \int_{0}^{+\infty} \frac{\partial}{\partial \tilde{\rho}} \left(\tilde{\psi}^* \tilde{\rho} \right) \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \, \mathrm{d}\tilde{\rho} \tag{526}$$

$$= - \int_{0}^{+\infty} \left(\tilde{\psi}^* + \tilde{\rho} \frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \right) \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \, \mathrm{d}\tilde{\rho} \tag{527}$$

$$= - \int_{0}^{+\infty} \left(\tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \tilde{\rho} \frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right) \mathrm{d}\tilde{\rho}. \tag{528}$$

Inserting this back into Eq. () now, we have

$$\tilde{E}_{\text{kin}} = -\pi \int_{-\infty}^{+\infty} \left[- \int_{0}^{+\infty} \left(\tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \tilde{\rho} \frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right) \mathrm{d}\tilde{\rho} + \int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \, \mathrm{d}\tilde{\rho} \right] \mathrm{d}\tilde{z} - \pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2} \tilde{\rho} \, \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z} \tag{529}$$

$$= -\pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \left(-\tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} - \tilde{\rho} \frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2} \tilde{\rho} \right) \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z} \tag{530}$$

$$= -\pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \left(-\tilde{\rho} \frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2} \tilde{\rho} \right) \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z} \tag{531}$$

$$= -\pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \left(-\tilde{\rho} \frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right) \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z} - \pi \int_{0}^{+\infty} \left[\int_{-\infty}^{+\infty} \tilde{\rho} \, \tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2} \, \mathrm{d}\tilde{z} \right] \mathrm{d}\tilde{\rho} \tag{532}$$

The second term inside the square brackets can be integrated by parts in the same way as the radial direction, except this time with respect to the $\tilde{z}$ variable

$$\int_{-\infty}^{+\infty} \tilde{\rho}\,\tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2}\,\mathrm{d}\tilde{z} = \left[\tilde{\rho}\,\tilde{\psi}^* \frac{\partial \tilde{\psi}}{\partial \tilde{z}}\right]_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} \frac{\partial}{\partial \tilde{z}}\left(\tilde{\rho}\tilde{\psi}^*\right) \frac{\partial \tilde{\psi}}{\partial \tilde{z}}\,\mathrm{d}\tilde{z}. \tag{533}$$

The boundary condition can be set to require the wavefunction goes to zero at infinity in the axial coordinate, i.e. $\tilde{\psi}(\tilde{z} = \pm\infty) = 0$, and therefore the first term goes to zero. So we can again use the chain rule

$$\int_{-\infty}^{+\infty} \tilde{\rho}\,\tilde{\psi}^* \frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2}\,\mathrm{d}\tilde{z} = -\int_{-\infty}^{+\infty} \frac{\partial}{\partial \tilde{z}}\left(\tilde{\rho}\tilde{\psi}^*\right) \frac{\partial \tilde{\psi}}{\partial \tilde{z}}\,\mathrm{d}\tilde{z} \tag{534}$$

$$= -\int_{-\infty}^{+\infty} \tilde{\rho}\,\frac{\partial \tilde{\psi}^*}{\partial \tilde{z}} \frac{\partial \tilde{\psi}}{\partial \tilde{z}}\,\mathrm{d}\tilde{z}. \tag{535}$$

Inserting this now back into Eq. (), we have

$$\tilde{E}_{\text{kin}} = -\pi \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \left(-\tilde{\rho}\frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}}\right) \mathrm{d}\tilde{\rho}\,\mathrm{d}\tilde{z} - \pi \int_{0}^{+\infty} \left[-\int_{-\infty}^{+\infty} \tilde{\rho}\,\frac{\partial \tilde{\psi}^*}{\partial \tilde{z}} \frac{\partial \tilde{\psi}}{\partial \tilde{z}}\,\mathrm{d}\tilde{z}\right] \mathrm{d}\tilde{\rho} \tag{536}$$

$$= \pi \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \left(\frac{\partial \tilde{\psi}^*}{\partial \tilde{\rho}} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \frac{\partial \tilde{\psi}^*}{\partial \tilde{z}} \frac{\partial \tilde{\psi}}{\partial \tilde{z}}\right) \tilde{\rho}\,\mathrm{d}\tilde{\rho}\,\mathrm{d}\tilde{z}. \tag{537}$$

This can be simplified further by considering the derivative of a general complex function

$$f(x) = a(x) + i\,b(x) \tag{538}$$

$$\implies \frac{\partial f}{\partial x} = \frac{\partial}{\partial x}\left(a + i\,b\right) \tag{539}$$

$$= \frac{\partial a}{\partial x} + i\frac{\partial b}{\partial x}. \tag{540}$$

Next if we consider the complex conjugate of f, we have

$$f^*(x) = a(x) - i\,b(x) \tag{541}$$

$$\implies \frac{\partial f^*}{\partial x} = \frac{\partial}{\partial x}\left(a - i\,b\right) \tag{542}$$

$$= \frac{\partial a}{\partial x} - i\frac{\partial b}{\partial x}, \tag{543}$$

where we can compare these expressions now to see that in general the derivative of a complex conjugate is the complex conjugate of the derivative

$$\frac{\partial f^*}{\partial x} = \left(\frac{\partial f}{\partial x}\right)^* \tag{544}$$

Using this in the expression for the kinetic energy, it becomes

$$\tilde{E}_{\text{kin}} = \pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \left(\left(\frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right)^{*} \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} + \left(\frac{\partial \tilde{\psi}}{\partial \tilde{z}} \right)^{*} \frac{\partial \tilde{\psi}}{\partial \tilde{z}} \right) \tilde{\rho} \, \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z} \tag{545}$$

$$= \pi \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \left(\left| \frac{\partial \tilde{\psi}}{\partial \tilde{\rho}} \right|^{2} + \left| \frac{\partial \tilde{\psi}}{\partial \tilde{z}} \right|^{2} \right) \tilde{\rho} \, \mathrm{d}\tilde{\rho} \, \mathrm{d}\tilde{z}. \tag{546}$$

Numerically, this is much more efficient to calculate, because it does not contain first and second derivatives anymore but only the first derivative with respect to each coordinate.

Potential Energy Term

The potential energy term is simple to calculate

$$\tilde{E}_{\text{pot}} = \langle\tilde{\psi}|\tilde{\hat{H}}_{\text{pot}}|\tilde{\psi}\rangle \tag{547}$$

$$= 2\pi \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \tilde{\psi}^*\, \tilde{V}_{\text{ext}}\, \tilde{\psi}\, \tilde{\rho}\, \mathrm{d}\tilde{\rho}\, \mathrm{d}\tilde{z} \tag{548}$$

$$= 2\pi \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \tilde{V}_{\text{ext}}\, |\tilde{\psi}|^2\, \tilde{\rho}\, \mathrm{d}\tilde{\rho}\, \mathrm{d}\tilde{z}. \tag{549}$$

Interaction Energy Term

The interaction energy is again simple to calculate

$$\tilde{E}_{\text{int}} = \langle\tilde{\psi}|\tilde{\hat{H}}_{\text{int}}|\tilde{\psi}\rangle \tag{550}$$

$$= 2\pi \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \tilde{\psi}^* \left\{\frac{1}{2}\mathcal{G}|\tilde{\psi}|^2\right\} \tilde{\psi}\, \tilde{\rho}\, \mathrm{d}\tilde{\rho}\, \mathrm{d}\tilde{z} \tag{551}$$

$$= 2\pi \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \frac{1}{2}\mathcal{G}|\tilde{\psi}|^4\, \tilde{\rho}\, \mathrm{d}\tilde{\rho}\, \mathrm{d}\tilde{z}. \tag{552}$$

10.1.2 Energy in Certain Analytic Limits

It is useful to have some analytic results for the energy for checking the results of the numerical calculations.

Non-Interacting Limit

In the non-interacting limit, the system is simply the harmonic oscillator. In 3D cartesian coordinates, the energy eigenvalues of the n^{th} state are given by

$$\varepsilon_{xyz} = \left(n_x + \frac{1}{2}\right)\hbar\omega_x + \left(n_y + \frac{1}{2}\right)\hbar\omega_y + \left(n_z + \frac{1}{2}\right)\hbar\omega_z. \tag{553}$$

If we have N_{tot} particles in the ground state ($n_x = n_y = n_z = 0$), the total energy will be

$$E = N_{\text{tot}} \times \varepsilon_{000} \tag{554}$$

$$= N_{\text{tot}}\, \frac{\hbar}{2}\left(\omega_x + \omega_y + \omega_z\right). \tag{555}$$

In cylindrical coordinates, $\omega_x = \omega_y = \omega_\rho$, and so in the absence of interactions the total energy should be

$$E = N_{\text{tot}}\, \frac{\hbar}{2}\left(2\omega_\rho + \omega_z\right). \tag{556}$$

In terms of the dimensionless variables, the non-interacting energy should be

$$\tilde{E} = \frac{E}{\hbar\omega_\rho} \tag{557}$$

$$= N_{\text{tot}} \frac{\hbar}{2} \left(2\omega_\rho + \omega_z\right) \times \frac{1}{\hbar\omega_\rho} \tag{558}$$

$$= N_{\text{tot}} \frac{\hbar}{2} \left(2\omega_\rho + \lambda\omega_\rho\right) \times \frac{1}{\hbar\omega_\rho} \tag{559}$$

$$= N_{\text{tot}} \left(1 + \frac{\lambda}{2}\right). \tag{560}$$

Interacting-Dominated Limit

By direct integration of the Gross-Pitaevskii equation, Dalfovo (1999) gives an expression for the chemical potential μ of the BEC in terms of the energy contributions for a general interaction strength

$$\mu = \frac{E_{\text{kin}} + E_{\text{pot}} + 2E_{\text{int}}}{N_{\text{tot}}}. \tag{561}$$

Now if the gas is specifically in the interaction-dominated limit (Thomas-Fermi regime), the kinetic energy term can be dropped in the GPE and the chemical potential can be found to be

$$\mu^{\text{TF}} = \left(\frac{15\sqrt{2}}{32\pi} N_{\text{tot}}\, g_{3\text{D}} m^{3/2} \omega_x \omega_y \omega_z\right)^{2/5} \tag{562}$$

$$= \left(\frac{15\sqrt{2}}{32\pi} N_{\text{tot}}\, g_{3\text{D}} m^{3/2} \omega_\rho^2 \omega_z\right)^{2/5}. \tag{563}$$

Combining these equations, the energy contributions found when the system is in the Thomas-Fermi regime should sum as

$$E_{\text{kin}} + E_{\text{pot}} + 2E_{\text{int}} = N_{\text{tot}} \left(\frac{15\sqrt{2}}{32\pi} N_{\text{tot}}\, g_{3\text{D}} m^{3/2} \omega_\rho^2 \omega_z\right)^{2/5}. \tag{564}$$

10.2 Method for Numerical Solution

The task is now to numerically solve Eq. (514). In 1D, the numerical solution of the 1D GPE can be done efficiently in several ways. For example, the spectral method can be used, whereby a Fourier transform (FFT) is used to calculate the derivative term in the kinetic energy. Secondly, the Crank-Nicolson finite difference method can be used, which reduces the problem to a matrix multiplication with a sparse, tridiagonal matrix - which is therefore efficient and easy to solve (for example using a Thomas algorithm).

However, in 3D cylindrical coordinates we cannot use Fourier transforms because these work by decomposing the cartesian coordinates into a sum of sine and cosines as a basis. In cylindrical coordinates, the radial direction cannot be resolved using the Fourier spectral method. Note however, that there are methods which use Bessel functions instead of sines and cosines as a basis (this is called

a Hankel transform, and is effectively a circularly-symmetric version of a Fourier transform), and these can be used to solve the radial direction (see, for example, Ronen 2006). The algorithms usually require that the radial grid points be chosen where the Bessel function has its zeros in order to work nicely, and so the grid isn't usually chosen arbitrarily.

If we try to apply the Crank-Nicolson method with more than one dimension, it still works but the resulting equations are coupled and the resulting matrix is no longer tridiagonal - it is banded to quite a high order (although still very sparse), which makes it computationally costly to solve. In this case, the equation is typically solved using an Alternating Direction Implicit (ADI) method (see Numerical Recipes in C), and is commonly used for 2D and 3D diffusion equations. ADI essentially breaks up the time step into multiple smaller time steps (one for each dimension), and in each one a different one of the dimensions is treated implicitly. For example, in 2D, you would break up into time steps of $\Delta t/2$, and solve the first dimension in the first step implicitly, and the second dimension implicitly in the second step. This means that only tridiagonal matrices appear now, and also ensures stability because of the implicit solving. In all steps, all of the terms in the equation appear, it's just that you change which terms you treat implicitly and which explcitly.

10.2.1 Operator Splitting for the Cylindrical GPE

The ADI method is a type of more general approach known as *operator splitting*. This consists of splitting operators in differential equations into a sum of individual operators, in such a way that they become convenient to solve each successively in turn. In the context of the cylindrically-symmetric GPE, which is written in general in dimensionless coorndinates as

$$i\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = \tilde{\hat{H}}\tilde{\psi}, \tag{565}$$

it is most convenient to split the full Hamiltonian operator into 3 parts (Adhikari, 2002) - those which depend on the axial coordinates, the radial coordinates, and the non-derivative terms, in such a way that

$$\tilde{\hat{H}} = \tilde{\hat{H}}_1 + \tilde{\hat{H}}_2 + \tilde{\hat{H}}_3 \tag{566}$$

$$= \underbrace{-\frac{1}{2}\left(\frac{\partial^2}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}\right)}_{\tilde{\hat{H}}_2} \underbrace{-\frac{1}{2}\frac{\partial^2}{\partial\tilde{z}^2}}_{\tilde{\hat{H}}_3} + \underbrace{\tilde{V}_{\text{ext}} + \mathcal{G}|\tilde{\psi}|^2}_{\tilde{\hat{H}}_1} \tag{567}$$

where

$$\tilde{V}_{\text{ext}} = \frac{1}{2}\left(\tilde{\rho}^2 + \lambda^2\tilde{z}^2\right). \tag{568}$$

Now, the Hamiltonian is split into a non-linear but non-derivative part ($\tilde{\hat{H}}_1$), and two terms which are linear, derivative operators ($\tilde{\hat{H}}_2$ and $\tilde{\hat{H}}_3$).

- The strategy is to solve $\tilde{\hat{H}}_1$ first (neglecting $\tilde{\hat{H}}_2$ and $\tilde{\hat{H}}_3$) over a time step $\Delta\tilde{t}$, to obtain some intermediate solution $\tilde{\psi}^{k+1/3}$ from $\tilde{\psi}^k$. Since this operator does not contain derivatives, it is

essentially solved exactly for small $\Delta\tilde{t}$, and the time step is solved simply using

$$\tilde{\psi}^{k+1/3} = \exp\left(-i\,\Delta\tilde{t}\,\hat{\tilde{H}}_1\right)\tilde{\psi}^{k}. \tag{569}$$

This is nice, because many of the early attempts at solving the GPE included the non-linearity together with the derivatives, which can lead to instability for large values of the non-linear term if random numerical noise gets amplified (see, for example, Ruprecht 1995). This operator splitting approach treats the non-linear term essentially exactly, and works well for large non-linearities (such as the Thomas-Fermi regime).

- Once $\tilde{\psi}^{k+1/3}$ is calculated, we need to use it to calculate the next fraction of a time step $\tilde{\psi}^{k+2/3}$ using the radial direction operator $\hat{\tilde{H}}_2$. This contains the $1/\tilde{\rho}$ term, so needs to be done using either finite differences, or using the Hankel transform.
- Finally, we need to calculate $\tilde{\psi}^{k+1}$ from $\tilde{\psi}^{k+2/3}$, using the axial direction operator $\hat{\tilde{H}}_3$. This can be done again using finite differences, but could as well be done using the regular Fourier spectral method (provided the wavefunction is smooth enough for the spectral method to work appropriately).

10.2.2 Discretization

The wavefunction $\tilde{\psi}(\tilde{\rho},\tilde{z})$ must now be discretized onto a radial $\tilde{\rho}$ and axial $\tilde{z}$ coordinate grid, so that it can be numerically solved. We can discretize the $\tilde{\rho}$-grid into N_ρ individual lattice points $\tilde{\rho}_n$, indexed by n, in steps of $\Delta\tilde{\rho}$, beginning with $\tilde{\rho}=0$

$$\tilde{\rho}_n = \tilde{\rho}[n] := (n-1)\cdot\Delta\tilde{\rho} \tag{570}$$

where $n = 1, 2, 3, ..., N_\rho$ are integers. This provides the following discrete $\tilde{\rho}$ coordinates

$$\tilde{\rho}_1 = 0 \tag{571}$$
$$\tilde{\rho}_2 = \Delta\tilde{\rho} \tag{572}$$
$$\tilde{\rho}_3 = 2\Delta\tilde{\rho} \tag{573}$$
$$\vdots \tag{574}$$
$$\tilde{\rho}_{N_\rho} = \left(N_\rho - 1\right)\Delta\tilde{\rho}. \tag{575}$$

Similarly, we also need to discretize the axial coordinate. This direction can take positive or negative values, and will have the zero point somewhere in the centre of the vector. We can now discretize the $\tilde{z}$-grid into N_z individual points $\tilde{z}_m$, indexed by m, in steps of $\Delta\tilde{z}$, in such a way that the $\tilde{z}=0$ point is at the middle of the vector (instead of at the beginning)

$$\tilde{z}_m = \tilde{z}[m] := \left(m - 1 - \frac{N_z}{2}\right)\cdot\Delta\tilde{z} \qquad \text{(if } N_z \text{ is even)} \tag{576}$$

$$\tilde{z}_m = \tilde{z}[m] := \left(m - 1 - \frac{N_z - 1}{2}\right) \cdot \Delta\tilde{z} \quad \text{(if } N_z \text{ is odd)} \tag{577}$$

$$\tag{578}$$

where $m = 1, 2, 3, ..., N_z$ are integers. This provides the following discrete $\tilde{z}$ coordinates

- if N_z is even

$$\tilde{z}_1 = -\frac{N_z}{2}\Delta\tilde{z} \tag{579}$$

$$\tilde{z}_2 = -\frac{N_z}{2}\Delta\tilde{z} + \Delta\tilde{z} \tag{580}$$

$$\tilde{z}_3 = -\frac{N_z}{2}\Delta\tilde{z} + 2\Delta\tilde{z} \tag{581}$$

$$\vdots \tag{582}$$

$$\tilde{z}_{N_z} = +\frac{N_z}{2}\Delta\tilde{z} - \Delta\tilde{z} \tag{583}$$

- if N_z is odd

$$\tilde{z}_1 = -\frac{(N_z - 1)}{2}\Delta\tilde{z} \tag{584}$$

$$\tilde{z}_2 = -\frac{(N_z - 1)}{2}\Delta\tilde{z} + \Delta\tilde{z} \tag{585}$$

$$\tilde{z}_3 = -\frac{(N_z - 1)}{2}\Delta\tilde{z} + 2\Delta\tilde{z} \tag{586}$$

$$\vdots \tag{587}$$

$$\tilde{z}_{N_z} = +\frac{(N_z - 1)}{2}\Delta\tilde{z} \tag{588}$$

This ensures that there is always a $\tilde{z} = 0$ point in the vector, no matter if N_z is odd or even. If N_z is even, the zero is placed at index $N_z/2 + 1$, and if N_z is odd the zero is placed at index $(N_z + 1)/2$.

With the coordinate system discretized, the wavefunction is similarly discretized as

$$\tilde{\psi}_{m,n} = \tilde{\psi}[m, n] := \tilde{\psi}(\tilde{\rho} = \tilde{\rho}_n, \tilde{z} = \tilde{z}_m), \tag{589}$$

where the first index m denotes the row number (axial coordinate), and the second index n denotes the column number (radial coordinate) in the matrix, shown in Fig. 51.

10.2.3 Solving the Non-Derivative Part

Considering only $\tilde{\hat{H}}_1$ (with $\tilde{\hat{H}}_2 = \tilde{\hat{H}}_3 = 0$) the equation reads

$$\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = -i\tilde{\hat{H}}_1\tilde{\psi} \tag{590}$$

$$= -i\left[\tilde{V}_{\text{ext}} + \mathcal{G}|\tilde{\psi}|^2\right]\tilde{\psi}. \tag{591}$$

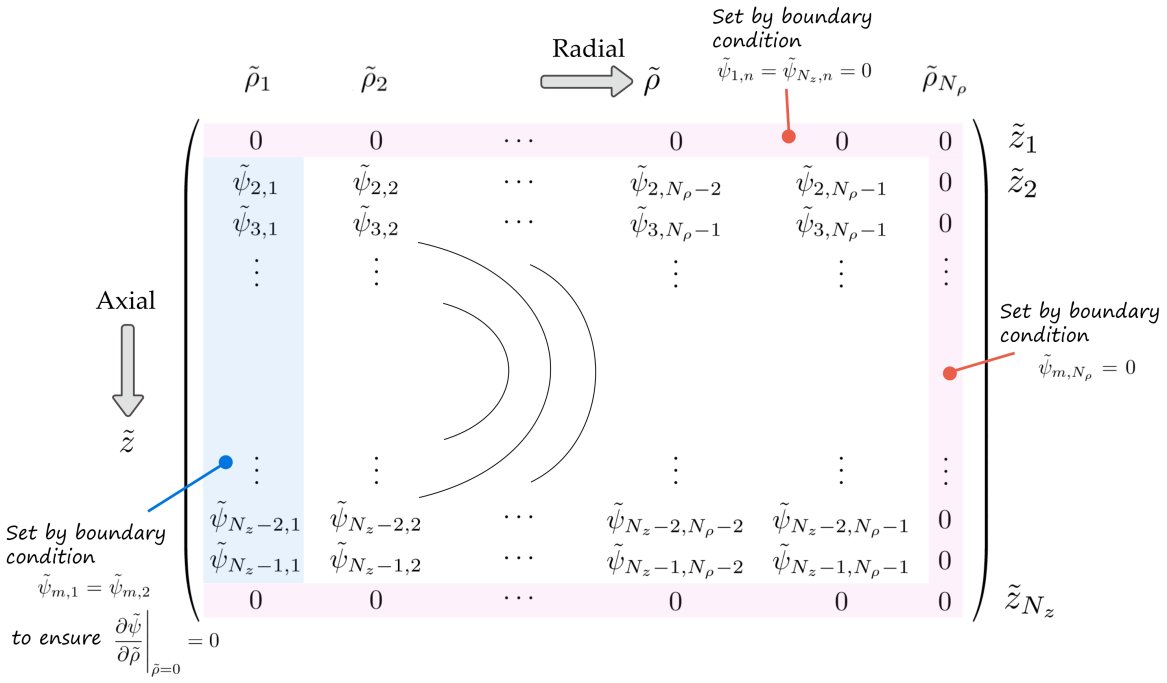

FIGURE 51: Schematic of the matrix used to store the discretized wavefunction $\tilde{\psi}_{m,n}$ in memory, with the associated enforced boundary conditions.

This is an ODE, and has the solution

$$\tilde{\psi}(\tilde{t} + \Delta\tilde{t}) = \exp\left(-i\tilde{\hat{H}}_1 \Delta\tilde{t}\right) \tilde{\psi}(\tilde{t}) \tag{592}$$

$$= \exp\left(-i\left[\tilde{V}_{\text{ext}} + \mathcal{G}|\tilde{\psi}|^2\right] \Delta\tilde{t}\right) \tilde{\psi}(\tilde{t}), \tag{593}$$

i.e. we can obtain the wavefunction at the next time step by applying a simple point-wise multiplication with the exponential of a matrix.

10.2.4 Solving the Radial Derivative Part

Considering now only $\tilde{\hat{H}}_2$ (with $\tilde{\hat{H}}_1 = \tilde{\hat{H}}_3 = 0$), i.e. the derivatives with respect to the radial direction, the equation reads

$$\frac{\partial \tilde{\psi}}{\partial \tilde{t}} = -i\tilde{\hat{H}}_2 \tilde{\psi} \tag{594}$$

$$= \frac{i}{2}\left(\frac{\partial^2 \tilde{\psi}}{\partial \tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial \tilde{\psi}}{\partial \tilde{\rho}}\right). \tag{595}$$

This equation can be solved either with a Hankel transform (expanding into a basis of Bessel functions), or by the Crank-Nicolson finite difference method. Here, I will use the latter. The equation can be discretized in time using the Crank-Nicolson method (which considers a combination of im-

plicit and explicit terms at the current and previous time step), as well as using second order finite differences to discretize the derivatives in space. The discretized version is then

$$\tilde{\psi}_{m,n}^{k+1} \approx \tilde{\psi}_{m,n}^{k} + \frac{i\Delta\tilde{t}}{4}\left[\frac{\tilde{\psi}_{m,n+1}^{k+1} + \tilde{\psi}_{m,n-1}^{k+1} - 2\tilde{\psi}_{m,n}^{k+1}}{\Delta\tilde{\rho}^2} + \frac{1}{\tilde{\rho}_n}\frac{\tilde{\psi}_{m,n+1}^{k+1} - \tilde{\psi}_{m,n-1}^{k+1}}{2\Delta\tilde{\rho}} + ... \right. \tag{596}$$

$$\left. \frac{\tilde{\psi}_{m,n+1}^{k} + \tilde{\psi}_{m,n-1}^{k} - 2\tilde{\psi}_{m,n}^{k}}{\Delta\tilde{\rho}^2} + \frac{1}{\tilde{\rho}_n}\frac{\tilde{\psi}_{m,n+1}^{k} - \tilde{\psi}_{m,n-1}^{k}}{2\Delta\tilde{\rho}} \right]. \tag{597}$$

We can see that this is an implicit equation, because the wavefunction at time $k+1$ (the one we are trying to calculate) appears on both the left and the right side of the equation. Now, defining the constants

$$\alpha = \frac{i\Delta\tilde{t}}{4\Delta\tilde{\rho}^2} \qquad \text{and} \qquad \beta = \frac{i\Delta\tilde{t}}{8\Delta\tilde{\rho}}, \tag{598}$$

the equation becomes

$$\tilde{\psi}_{m,n}^{k+1} \approx \tilde{\psi}_{m,n}^{k} + \alpha\Big[\tilde{\psi}_{m,n+1}^{k+1} + \tilde{\psi}_{m,n-1}^{k+1} - 2\tilde{\psi}_{m,n}^{k+1} + \tilde{\psi}_{m,n+1}^{k} + \tilde{\psi}_{m,n-1}^{k} - 2\tilde{\psi}_{m,n}^{k}\Big] + ... \tag{599}$$

$$+ \frac{\beta}{\tilde{\rho}_n}\Big[\tilde{\psi}_{m,n+1}^{k+1} - \tilde{\psi}_{m,n-1}^{k+1} + \tilde{\psi}_{m,n+1}^{k} - \tilde{\psi}_{m,n-1}^{k}\Big]. \tag{600}$$

Rearranging to bring all the $k+1$ terms onto the left, and all the k terms to the right, we have

$$(1+2\alpha)\,\tilde{\psi}_{m,n}^{k+1} - \big(\alpha - \beta/\tilde{\rho}_n\big)\,\tilde{\psi}_{m,n-1}^{k+1} - \big(\alpha + \beta/\tilde{\rho}_n\big)\,\tilde{\psi}_{m,n+1}^{k+1} = \tag{601}$$

$$(1-2\alpha)\,\tilde{\psi}_{m,n}^{k} + \big(\alpha - \beta/\tilde{\rho}_n\big)\,\tilde{\psi}_{m,n-1}^{k} + \big(\alpha + \beta/\tilde{\rho}_n\big)\,\tilde{\psi}_{m,n+1}^{k}. \tag{602}$$

This is a system of equations, one for each m (i.e. each axial coordinate). Now, the radial coordinate vector increases from zero in increments of $\Delta\tilde{\rho}$, such that $\tilde{\rho}_1 = 0$, $\tilde{\rho}_2 = \Delta\tilde{\rho}$, $\tilde{\rho}_3 = 2\Delta\tilde{\rho}$, ... , $\tilde{\rho}_{N_\rho} = \big(N_\rho - 1\big)\,\Delta\tilde{\rho}$. This means that when $n = 1$ the equation above includes a term of the form $\beta/\tilde{\rho}_1$, which diverges to infinity - a reflection of the fact that the Laplacian diverges at $\tilde{\rho} = 0$. There are several ways that the coordinate singularity can be dealt with, but here I will simply not calculate the value for $n = 1$, but will instead start at $n = 2$. Then once $n = 2$ is calculated I will force the wavefunction at the origin to equal this value, i.e. $\tilde{\psi}_{m,1} = \tilde{\psi}_{m,2}$. This has the added benefit of essentially enforcing the boundary condition

$$\left.\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}\right|_{\tilde{\rho}=0} = 0, \tag{603}$$

which ensures that the wavefunction is even around $\tilde{\rho} = 0$, i.e. $\tilde{\psi}(\tilde{\rho}) = \tilde{\psi}(-\tilde{\rho})$, and the system is cylindrically symmetric.

So, for $n = 2$ (for any m) we have

$$(1+2\alpha)\,\tilde{\psi}_{m,2}^{k+1} - \big(\alpha - \beta/\tilde{\rho}_2\big)\,\tilde{\psi}_{m,1}^{k+1} - \big(\alpha + \beta/\tilde{\rho}_2\big)\,\tilde{\psi}_{m,3}^{k+1} = \tag{604}$$

$$(1-2\alpha)\,\tilde{\psi}_{m,2}^{k} + \big(\alpha - \beta/\tilde{\rho}_2\big)\,\tilde{\psi}_{m,1}^{k} + \big(\alpha + \beta/\tilde{\rho}_2\big)\,\tilde{\psi}_{m,3}^{k}, \tag{605}$$

but since $\tilde{\psi}_{m,1} = \tilde{\psi}_{m,2}$, this becomes

$$\left(1 + \alpha + \beta/\tilde{\rho}_2\right) \tilde{\psi}_{m,2}^{k+1} - \left(\alpha + \beta/\tilde{\rho}_2\right) \tilde{\psi}_{m,3}^{k+1} = \left(1 - \alpha - \beta/\tilde{\rho}_2\right) \tilde{\psi}_{m,2}^{k} + \left(\alpha + \beta/\tilde{\rho}_2\right) \tilde{\psi}_{m,3}^{k}. \tag{606}$$

For $n = 3$ similarly we have

$$\begin{aligned} (1 + 2\alpha) \tilde{\psi}_{m,3}^{k+1} - \left(\alpha - \beta/\tilde{\rho}_3\right) \tilde{\psi}_{m,2}^{k+1} - \left(\alpha + \beta/\tilde{\rho}_3\right) \tilde{\psi}_{m,4}^{k+1} = & \quad (607) \\ (1 - 2\alpha) \tilde{\psi}_{m,3}^{k} + \left(\alpha - \beta/\tilde{\rho}_3\right) \tilde{\psi}_{m,2}^{k} + \left(\alpha + \beta/\tilde{\rho}_3\right) \tilde{\psi}_{m,4}^{k}, & \quad (608) \end{aligned}$$

This continues until finally $n = N_\rho - 1$, where we have

$$\begin{aligned} (1 + 2\alpha) \tilde{\psi}_{m,N_\rho-1}^{k+1} - \left(\alpha - \beta/\tilde{\rho}_{N_\rho-1}\right) \tilde{\psi}_{m,N_\rho-2}^{k+1} - \left(\alpha + \beta/\tilde{\rho}_{N_\rho-1}\right) \tilde{\psi}_{m,N_\rho}^{k+1} = & \quad (609) \\ (1 - 2\alpha) \tilde{\psi}_{m,N_\rho-1}^{k} + \left(\alpha - \beta/\tilde{\rho}_{N_\rho-1}\right) \tilde{\psi}_{m,N_\rho-2}^{k} + \left(\alpha + \beta/\tilde{\rho}_{N_\rho-1}\right) \tilde{\psi}_{m,N_\rho}^{k}, & \quad (610) \end{aligned}$$

but we can also include the boundary condition that the wavefunction must go to zero at infinity, so as long as a large enough grid support is used then we can specify $\tilde{\psi}_{m,N_\rho} = 0$, and so the equation simplifies to

$$(1 + 2\alpha) \tilde{\psi}_{m,N_\rho-1}^{k+1} - \left(\alpha - \beta/\tilde{\rho}_{N_\rho-1}\right) \tilde{\psi}_{m,N_\rho-2}^{k+1} = (1 - 2\alpha) \tilde{\psi}_{m,N_\rho-1}^{k} + \left(\alpha - \beta/\tilde{\rho}_{N_\rho-1}\right) \tilde{\psi}_{m,N_\rho-2}^{k}. \tag{611}$$

This system of equations can be collected together and written as a single matrix equation

$$\begin{pmatrix} 1 + \alpha + \beta/\tilde{\rho}_2 & -\left(\alpha + \beta/\tilde{\rho}_2\right) & 0 & \dots & 0 \\ -\left(\alpha - \beta/\tilde{\rho}_3\right) & 1 + 2\alpha & -\left(\alpha + \beta/\tilde{\rho}_3\right) & \ddots & \vdots \\ 0 & -\left(\alpha - \beta/\tilde{\rho}_4\right) & \ddots & -\left(\alpha + \beta/\tilde{\rho}_{N_\rho-3}\right) & 0 \\ \vdots & \ddots & -\left(\alpha - \beta/\tilde{\rho}_{N_\rho-2}\right) & 1 + 2\alpha & -\left(\alpha + \beta/\tilde{\rho}_{N_\rho-2}\right) \\ 0 & \dots & 0 & -\left(\alpha - \beta/\tilde{\rho}_{N_\rho-1}\right) & 1 + 2\alpha \end{pmatrix} \begin{pmatrix} \tilde{\psi}_{m,2}^{k+1} \\ \tilde{\psi}_{m,3}^{k+1} \\ \vdots \\ \tilde{\psi}_{m,N_\rho-2}^{k+1} \\ \tilde{\psi}_{m,N_\rho-1}^{k+1} \end{pmatrix} =$$

$$\begin{pmatrix} 1 - \alpha - \beta/\tilde{\rho}_2 & \left(\alpha + \beta/\tilde{\rho}_2\right) & 0 & \dots & 0 \\ \left(\alpha - \beta/\tilde{\rho}_3\right) & 1 - 2\alpha & \left(\alpha + \beta/\tilde{\rho}_3\right) & \ddots & \vdots \\ 0 & \left(\alpha - \beta/\tilde{\rho}_4\right) & \ddots & \left(\alpha + \beta/\tilde{\rho}_{N_\rho-3}\right) & 0 \\ \vdots & \ddots & \left(\alpha - \beta/\tilde{\rho}_{N_\rho-2}\right) & 1 - 2\alpha & \left(\alpha + \beta/\tilde{\rho}_{N_\rho-2}\right) \\ 0 & \dots & 0 & \left(\alpha - \beta/\tilde{\rho}_{N_\rho-1}\right) & 1 - 2\alpha \end{pmatrix} \begin{pmatrix} \tilde{\psi}_{m,2}^{k} \\ \tilde{\psi}_{m,3}^{k} \\ \vdots \\ \tilde{\psi}_{m,N_\rho-2}^{k} \\ \tilde{\psi}_{m,N_\rho-1}^{k} \end{pmatrix}$$

or simply as

$$\mathbf{A}\psi_m^{k+1} = \mathbf{B}\psi_m^k, \tag{612}$$

and so the wavefunction at a subsequent time can be found by matrix inversion to be given in terms

of the tridiagonal matrices

$$\psi_m^{k+1} = \mathbf{A}^{-1}\mathbf{B}\psi_m^k = \mathbf{M}_{\mathbf{CN}}^{\rho}\psi_m^k, \tag{613}$$

and a time step amounts to multiplying each row by the Crank-Nicolson matrix for the radial direction $\mathbf{M}_{\mathbf{CN}}^{\rho}$. The matrices $\mathbf{A}$ and $\mathbf{B}$ are tridiagonal, and so it makes sense to create them as *sparse* matrices in Matlab. This is a data type which allows only the non-zero elements of the matrix to be retained in memory (as a list of numbers), instead of storing all the zero entries as well. This can alleviate memory problems for large matrices, and makes manipulating them much more efficient. An example of creating a simple tridiagonal matrix in Matlab is the following using the `spdiags()` function:

```
A_rho_dd = [1 2 3 4 5].';        % Main diagonal elements
A_rho_mm = [6 7 8 9 NaN].';      % Subdiagonal elements
A_rho_pp = [NaN 10 11 12 13].'; % Superdiagonal elements
A_rho = spdiags( [A_rho_mm A_rho_dd A_rho_pp] , -1:1, 5,5);
```

Looking at the sparse version, we see the list of non-zero entries and their respective positions in the matrix:

```
>> disp(A_rho)

   (1,1)         1
   (2,1)         6
   (1,2)        10
   (2,2)         2
   (3,2)         7
   (2,3)        11
   (3,3)         3
   (4,3)         8
   (3,4)        12
   (4,4)         4
   (5,4)         9
   (4,5)        13
   (5,5)         5
```

The full version can be inspected using the function `full()`, which shows the zero entries as well, but would be much more inefficient to store in memory:

```
>> disp( full(A_rho) )

     1    10     0     0     0
     6     2    11     0     0
     0     7     3    12     0
     0     0     8     4    13
     0     0     0     9     5
```

Once we have generated the two matrices $\mathbf{A}$ and $\mathbf{B}$, we need to obtain the Crank-Nicolson matrix $\mathbf{M}^{\rho}_{\mathbf{CN}}$. It is in general not a good idea to find the inverse $\mathbf{A}^{-1}$, and is not necessary. Instead, Matlab has the built-in *backslash operator* for solving systems of linear equations. The Crank-Nicolson matrix can be found from the left and right matrices using simply:

```
M_CN_rho = A_rho\B_rho;
```

10.2.5 Solving the Axial Derivative Part

Considering now only $\tilde{\hat{H}}_3$ (with $\tilde{\hat{H}}_1 = \tilde{\hat{H}}_2 = 0$), i.e. the derivatives with respect to the axial direction, the equation reads

$$\frac{\partial \tilde{\psi}}{\partial \tilde{t}} = -i\tilde{\hat{H}}_3 \tilde{\psi} \tag{614}$$

$$= \frac{i}{2}\frac{\partial^2 \tilde{\psi}}{\partial \tilde{z}^2}. \tag{615}$$

This can be discretized in the same way as for the radial direction, using second-order finite differences for the space derivative (this time using the m index, denoting the axial z coordinate), and Crank-Nicolson for the time derivative

$$\tilde{\psi}^{k+1}_{m,n} \approx \tilde{\psi}^{k}_{m,n} + \frac{i\Delta\tilde{t}}{4}\left[\frac{\tilde{\psi}^{k+1}_{m+1,n} + \tilde{\psi}^{k+1}_{m-1,n} - 2\tilde{\psi}^{k+1}_{m,n}}{\Delta\tilde{z}^2} + \frac{\tilde{\psi}^{k}_{m+1,n} + \tilde{\psi}^{k}_{m-1,n} - 2\tilde{\psi}^{k}_{m,n}}{\Delta\tilde{z}^2}\right]. \tag{616}$$

Defining the constant

$$\gamma = \frac{i\Delta\tilde{t}}{4\Delta\tilde{z}^2} \tag{617}$$

this reduces to

$$\tilde{\psi}^{k+1}_{m,n} = \tilde{\psi}^{k}_{m,n} + \gamma\left[\tilde{\psi}^{k+1}_{m+1,n} + \tilde{\psi}^{k+1}_{m-1,n} - 2\tilde{\psi}^{k+1}_{m,n} + \tilde{\psi}^{k}_{m+1,n} + \tilde{\psi}^{k}_{m-1,n} - 2\tilde{\psi}^{k}_{m,n}\right]. \tag{618}$$

Collecting all the $k+1$ terms on the left and the k terms on the right, this becomes

$$(1+2\gamma)\,\tilde{\psi}^{k+1}_{m,n} \;-\; \gamma\,\tilde{\psi}^{k+1}_{m-1,n} \;-\; \gamma\,\tilde{\psi}^{k+1}_{m+1,n} = (1-2\gamma)\,\tilde{\psi}^{k}_{m,n} \;+\; \gamma\,\tilde{\psi}^{k}_{m-1,n} \;+\; \gamma\,\tilde{\psi}^{k}_{m+1,n}. \tag{619}$$

If we set the boundary conditions at the two end axial coordinates at $m = 1$ and $m = N_z$ to ensure the wavefunction is zero at those points (i.e. $\tilde{\psi}_{1,n} = \tilde{\psi}_{N_z,n} = 0$ for all n) then this system of equations

can be written as a matrix equation

$$\begin{pmatrix} 1+2\gamma & -\gamma & 0 & \dots & 0 \\ -\gamma & 1+2\gamma & -\gamma & \ddots & \vdots \\ 0 & -\gamma & \ddots & -\gamma & 0 \\ \vdots & \ddots & -\gamma & 1+2\gamma & -\gamma \\ 0 & \dots & 0 & -\gamma & 1+2\gamma \end{pmatrix} \begin{pmatrix} \tilde{\psi}^{k+1}_{2,n} \\ \tilde{\psi}^{k+1}_{3,n} \\ \vdots \\ \tilde{\psi}^{k+1}_{N_z-2,n} \\ \tilde{\psi}^{k+1}_{N_z-1,n} \end{pmatrix} =$$

$$\begin{pmatrix} 1-2\gamma & \gamma & 0 & \dots & 0 \\ \gamma & 1-2\gamma & \gamma & \ddots & \vdots \\ 0 & \gamma & \ddots & \gamma & 0 \\ \vdots & \ddots & \gamma & 1-2\gamma & \gamma \\ 0 & \dots & 0 & \gamma & 1-2\gamma \end{pmatrix} \begin{pmatrix} \tilde{\psi}^{k}_{2,n} \\ \tilde{\psi}^{k}_{3,n} \\ \vdots \\ \tilde{\psi}^{k}_{N_z-2,n} \\ \tilde{\psi}^{k}_{N_z-1,n} \end{pmatrix}$$

or simply as

$$\mathbf{A}\psi^{k+1}_n = \mathbf{B}\psi^k_n, \tag{620}$$

and so the wavefunction at a subsequent time can be found by matrix inversion to be given in terms of the tridiagonal matrices

$$\psi^{k+1}_n = \mathbf{A}^{-1}\mathbf{B}\psi^k_n = \mathbf{M}^{\mathbf{z}}_{\mathbf{CN}}\psi^k_n. \tag{621}$$

The Crank-Nicolson matrix should be stored in sparse format in the same way as for the radial direction.

10.2.6 Alternative: Solving the Axial Derivative Part Spectrally

For the axial direction, the equation can be solved as in the previous section by using the Crank-Nicolson finite difference method in space and time. However, this part can also be solved spectrally instead (because the z direction is equivalent in either cylindrical or cartesian coordinates). To do this, we need to Fourier transform the equation itself with respect to the axial spatial coordinate

$$\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = \frac{i}{2}\frac{\partial^2\tilde{\psi}}{\partial\tilde{z}^2} \tag{622}$$

$$\xrightarrow{\mathcal{F}_z} \quad \mathcal{F}_z\left\{\frac{\partial\tilde{\psi}(\tilde{\rho},\tilde{z})}{\partial\tilde{t}}\right\} = \mathcal{F}_z\left\{\frac{i}{2}\frac{\partial^2\tilde{\psi}(\tilde{\rho},\tilde{z})}{\partial\tilde{z}^2}\right\} \tag{623}$$

$$\implies \frac{\partial\tilde{\tilde{\psi}}(\tilde{\rho},\tilde{k}_z)}{\partial\tilde{t}} = \frac{i}{2}\mathcal{F}_z\left\{\frac{\partial^2\tilde{\psi}(\tilde{\rho},\tilde{z})}{\partial\tilde{z}^2}\right\} \tag{624}$$

$$= -\frac{i}{2}\tilde{k}_z^2\,\tilde{\tilde{\psi}}(\tilde{\rho},\tilde{k}_z). \tag{625}$$

Now that the derivative has been converted to a multiplication by the wavevector squared in Fourier space, the equation is reduced to an ODE and can be solved exactly for small time step

$$\tilde{\tilde{\psi}}(\tilde{\rho}, \tilde{k}_z, \tilde{t} + \Delta\tilde{t}) = \exp\left(-\frac{i}{2}\tilde{k}_z^2 \Delta\tilde{t}\right) \tilde{\tilde{\psi}}(\tilde{\rho}, \tilde{k}_z, \tilde{t}). \tag{626}$$

To get the final solution, we then simply need to inverse transform both sides of the equation

$$\tilde{\psi}(\tilde{\rho}, \tilde{z}, \tilde{t} + \Delta\tilde{t}) = \mathcal{F}_z^{-1}\left\{\exp\left(-\frac{i}{2}\tilde{k}_z^2 \Delta\tilde{t}\right) \mathcal{F}_z\left\{\tilde{\psi}(\tilde{\rho}, \tilde{z}, \tilde{t})\right\}\right\}. \tag{627}$$

In general, this approach is a very efficient way to solve the equation - it is spectrally accurate (excellent convergence), and can be much more relaxed with grid sizes than the finite difference approach. However, it is only good if the wavefunction is "smooth", and can be represented faithfully as a sum of sines and cosines (so that the Fourier decomposition works), which means it will not work well for wavefunctions which are discontinuous or random, for example. In such cases, the finite difference approach is more appropriate.

10.2.7 Alternative: Solving the Radial Derivative Part Spectrally

The radial equation needed to be solved is

$$\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = \frac{i}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}\right), \tag{628}$$

however if we try to follow the same idea as with the axial direction - taking a Fourier transform - we find that the result is not a simple multiplication anymore, but that it is a less simple convolution. This is because of the second term above, and the fact that the Fourier transform only works for a cartesian system.

However, an alternative to the Fourier transform is the *Hankel transform*. Whereas the Fourier transform decomposes a function into an infinite summation of sine and cosine functions, the Hankel transform does the same thing but for Bessel functions of the first kind instead, and is appropriate for circularly symmetric problems. The forward and inverse Hankel transforms of order ν of a function $\psi(\rho)$ are given respectively by

$$\mathcal{H}_\nu\left\{\,\psi(\rho)\,\right\} = \hat{\psi}\left(k_\rho\right) = \int_0^\infty \psi\left(\rho\right) J_\nu\left(k_\rho\rho\right) \rho \,\mathrm{d}\rho \tag{629}$$

$$\mathcal{H}_\nu^{-1}\left\{\,\hat{\psi}(k_\rho)\,\right\} = \psi\left(\rho\right) = \int_0^\infty \hat{\psi}(k_\rho) J_\nu\left(k_\rho\rho\right) k_\rho \,\mathrm{d}k_\rho, \tag{630}$$

where k_ρ is the wavevector associated with the radial coordinate ρ, i.e. the Hankel transform is its own inverse (in contrast to the Fourier transform). Here, $J_\nu()$ is the Bessel function of the first kind of order ν, given by

$$J_\nu\left(x\right) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\,\Gamma\left(m+\nu+1\right)}\left(\frac{x}{2}\right)^{2m+\nu}, \tag{631}$$

and $\Gamma(z)$ is the gamma function. For positive integers n, the gamma function is simply $\Gamma(n) = (n-1)!$, a shifted factorial. Bessel functions of the first kind are known to be standard solutions of Bessel's differential equation

$$x^2 \frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + x \frac{\mathrm{d}y}{\mathrm{d}x} + \left(x^2 - \nu^2\right) y = 0, \tag{632}$$

i.e. the function $y(x) = J_\nu(x)$ satisfies this equation. The Bessel function of the first kind in Matlab can be calculated with the command `J = besselj(nu,x)`, and the first three orders are shown in Fig. 52.

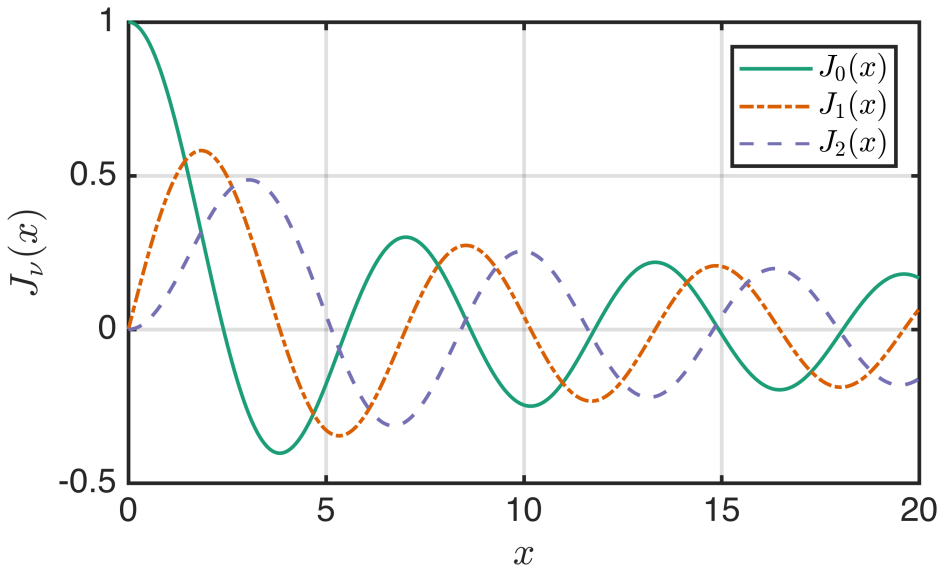

FIGURE 52: The first three orders of the Bessel function of the first kind.

Just as the Laplacian in cartesian coordinates turns into a multiplication in Fourier space, similarly the Laplacian in cylindrical coordinates turns into a multiplication in Hankel space. For a function with cylindrical symmetry, i.e. independent of the azimuthal angle ϕ, this is achieved by using a zero-order Hankel transform $\mathcal{H}_0$ (although the result can be extended to higher orders if needed). The key result is

$$\boxed{\nabla^2 \psi(\rho) = \frac{\partial^2 \psi}{\partial \rho^2} + \frac{1}{\rho}\frac{\partial \psi}{\partial \rho} = \mathcal{H}_0^{-1}\Big\{ -k_\rho^2 \cdot \mathcal{H}_0\Big\{\psi(\rho)\Big\}\Big\}} \tag{633}$$

and so to calculate the Laplacian in cylindrical coordinates we can do it by first transforming into Hankel space, then multiplying by $-k_\rho^2$, and finally inverse transforming back into coordinate space.

Proof:

We start by taking the zero-order Hankel transform of the Laplacian (assuming independent of azimuthal angle ϕ), which decomposes the Laplacian into a weighted sum of zero-order Bessel functions

of the first kind

$$\mathcal{H}_0\Big\{\nabla^2\psi\left(\rho\right)\Big\} = \mathcal{H}_0\left\{\frac{\partial^2\psi}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right\} \tag{634}$$

$$= \int_0^\infty \left(\frac{\partial^2\psi}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right) J_0\left(k_\rho\rho\right)\,\rho\,\mathrm{d}\rho \tag{635}$$

$$= \int_0^\infty \frac{\partial^2\psi}{\partial\rho^2}\,\rho\,J_0\left(k_\rho\rho\right)\,\mathrm{d}\rho \;+\; \int_0^\infty \frac{\partial\psi}{\partial\rho}\,J_0\left(k_\rho\rho\right)\,\mathrm{d}\rho. \tag{636}$$

Integrating the first term by parts, we have

$$\mathcal{H}_0\left\{\frac{\partial^2\psi}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right\} = \left[\frac{\partial\psi}{\partial\rho}\,\rho\,J_0\left(k_\rho\rho\right)\right]_0^\infty - \int_0^\infty \frac{\partial\psi}{\partial\rho}\frac{\partial}{\partial\rho}\Big(\rho\,J_0\left(k_\rho\rho\right)\Big)\,\mathrm{d}\rho \tag{637}$$

$$+ \int_0^\infty \frac{\partial\psi}{\partial\rho}\,J_0\left(k_\rho\rho\right)\,\mathrm{d}\rho \tag{638}$$

$$= \left[\frac{\partial\psi}{\partial\rho}\,\rho\,J_0\left(k_\rho\rho\right)\right]_0^\infty - \int_0^\infty \frac{\partial\psi}{\partial\rho}\left(\rho\,\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho} + J_0\left(k_\rho\rho\right)\right)\,\mathrm{d}\rho \tag{639}$$

$$+ \int_0^\infty \frac{\partial\psi}{\partial\rho}\,J_0\left(k_\rho\rho\right)\,\mathrm{d}\rho \tag{640}$$

$$= \left[\frac{\partial\psi}{\partial\rho}\,\rho\,J_0\left(k_\rho\rho\right)\right]_0^\infty - \int_0^\infty \frac{\partial\psi}{\partial\rho}\rho\,\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\,\mathrm{d}\rho, \tag{641}$$

where I have used the chain rule for differentiation in the second step, and the last two terms have cancelled out in the last step. Now we make the assumption that

$$\left(\frac{\partial\psi}{\partial\rho}\,\rho\right) \to 0 \;\text{ as }\; \rho\to 0 \;\text{ and }\; \rho\to\infty, \tag{642}$$

which is okay because the J_0 is finite, and then wavefunction will typically have gone to zero (and-therefore its derivative) even while ρ is still finite. Under this assumption, the first term in square brackets goes to zero. So, next applying integration by parts again, this can be written as

$$\mathcal{H}_0\left\{\frac{\partial^2\psi}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right\} = -\int_0^\infty \frac{\partial\psi}{\partial\rho}\rho\,\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\,\mathrm{d}\rho \tag{643}$$

$$= -\left[\psi\,\rho\,\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\right]_0^\infty + \int_0^\infty \psi\,\frac{\partial}{\partial\rho}\left(\rho\,\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\right)\,\mathrm{d}\rho \tag{644}$$

$$= -\left[\psi\,\rho\,\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\right]_0^\infty + \int_0^\infty \psi\,\left(\rho\,\frac{\partial^2 J_0\left(k_\rho\rho\right)}{\partial\rho^2} + \frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\right)\,\mathrm{d}\rho. \tag{645}$$

Here I will make another assumption, similar to before, that

$$\left(\psi\,\rho\right) \to 0 \;\text{ as }\; \rho\to 0 \;\text{ and }\; \rho\to\infty, \tag{646}$$

which again means the first term goes to zero so that we simply have

$$\mathcal{H}_0\left\{\frac{\partial^2\psi}{\partial\rho^2}+\frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right\}=\int_0^\infty\psi\left(\rho\frac{\partial^2 J_0\left(k_\rho\rho\right)}{\partial\rho^2}+\frac{\partial J_0\left(k_\rho\rho\right)}{\partial\rho}\right)\,\mathrm{d}\rho. \tag{647}$$

Now we recall that the zero-order Bessel function of the first kind $J_0(x)$ is a solution of Bessel's zero-order differential equation (i.e. Eq. (632) with $\nu=0$ after dividing through by a factor of x)

$$x\frac{\partial^2 J_0(x)}{\partial x^2}+\frac{\partial J_0(x)}{\partial x}=-xJ_0(x). \tag{648}$$

Making a change of variables $x=k_\rho\rho$, to get it into a form similar to Eq. (647), we have

$$\frac{\partial J_0(x)}{\partial x}=\frac{\partial J_0(k_\rho\rho)}{\partial\rho}\frac{\partial\rho}{\partial x}=\frac{1}{k_\rho}\frac{\partial J_0(k_\rho\rho)}{\partial\rho} \tag{649}$$

and similarly

$$\frac{\partial^2 J_0(x)}{\partial x^2}=\frac{\partial}{\partial x}\left(\frac{1}{k_\rho}\frac{\partial J_0(k_\rho\rho)}{\partial\rho}\right)=\frac{1}{k_\rho^2}\frac{\partial^2 J_0(k_\rho\rho)}{\partial\rho^2}. \tag{650}$$

Inserting these into Eq. (648), the Bessel differential equation in terms of the scaled parameter $k_\rho\rho$ becomes

$$\frac{\rho}{k_\rho}\frac{\partial^2 J_0(k_\rho\rho)}{\partial\rho^2}+\frac{1}{k_\rho}\frac{\partial J_0(k_\rho\rho)}{\partial\rho}=-k_\rho\,\rho\,J_0(k_\rho\rho) \tag{651}$$

$$\implies\rho\frac{\partial^2 J_0(k_\rho\rho)}{\partial\rho^2}+\frac{\partial J_0(k_\rho\rho)}{\partial\rho}=-k_\rho^2\,\rho\,J_0(k_\rho\rho). \tag{652}$$

The left hand side is exactly that appearing in the brackets in Eq. (647), and so the right hand side can be substituted in to give

$$\mathcal{H}_0\left\{\frac{\partial^2\psi}{\partial\rho^2}+\frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right\}=\int_0^\infty\psi\left(-k_\rho^2\,\rho\,J_0(k_\rho\rho)\right)\,\mathrm{d}\rho \tag{653}$$

$$=-k_\rho^2\int_0^\infty\psi\,J_0(k_\rho\rho)\,\rho\,\mathrm{d}\rho, \tag{654}$$

where we recognise the right hand side now to be simply the Hankel transform of ψ but multiplied by $-k_\rho^2$

$$\mathcal{H}_0\left\{\frac{\partial^2\psi}{\partial\rho^2}+\frac{1}{\rho}\frac{\partial\psi}{\partial\rho}\right\}=-k_\rho^2\,\mathcal{H}_0\{\psi\}. \tag{655}$$

Finally, taking the inverse transform of both sides we arrive at

$$\frac{\partial^2\psi}{\partial\rho^2}+\frac{1}{\rho}\frac{\partial\psi}{\partial\rho}=\mathcal{H}_0^{-1}\Big\{-k_\rho^2\,\mathcal{H}_0\{\psi\}\Big\}, \tag{656}$$

as advertised.

Now that I have a simple way to calculate the radial Laplacian using Hankel transforms, I can take the radial term in the Gross-Pitaevskii equation and Hankel transform of order zero the equation itself along the ρ coordinate

$$\frac{\partial\tilde{\psi}}{\partial\tilde{t}} = \frac{i}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}\right) \tag{657}$$

$$\xrightarrow{\mathcal{H}_0} \quad \mathcal{H}_0\left\{\frac{\partial\tilde{\psi}(\tilde{\rho},\tilde{z})}{\partial\tilde{t}}\right\} = \mathcal{H}_0\left\{\frac{i}{2}\left(\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}\right)\right\} \tag{658}$$

$$\implies \frac{\partial\tilde{\tilde{\psi}}(\tilde{k_\rho},\tilde{z})}{\partial\tilde{t}} = \frac{i}{2}\mathcal{H}_0\left\{\frac{\partial^2\tilde{\psi}}{\partial\tilde{\rho}^2} + \frac{1}{\tilde{\rho}}\frac{\partial\tilde{\psi}}{\partial\tilde{\rho}}\right\} \tag{659}$$

$$= -\frac{i}{2}k_\rho^2\,\mathcal{H}_0\{\tilde{\psi}(\tilde{\rho},\tilde{z})\} \tag{660}$$

$$= -\frac{i}{2}k_\rho^2\,\tilde{\tilde{\psi}}(\tilde{k_\rho},\tilde{z}) \tag{661}$$

where $\tilde{\tilde{\psi}}(\tilde{k_\rho},\tilde{z})$ is the Hankel transform of $\tilde{\psi}(\tilde{\rho},\tilde{z})$ along the ρ direction. The partial differential equation with the difficult Laplacian terms has now been reduced to an ordinary differential equation in Hankel space, and has the simple solution in Hankel space

$$\tilde{\tilde{\psi}}(\tilde{k}_\rho,\tilde{z},\tilde{t}+\Delta\tilde{t}) = \exp\left(-\frac{i}{2}\tilde{k}_\rho^2\Delta\tilde{t}\right)\,\tilde{\tilde{\psi}}(\tilde{k}_\rho,\tilde{z},\tilde{t}), \tag{662}$$

and the final solution back in coordinate space is obtained by simply inverse Hankel transforming

$$\tilde{\psi}(\tilde{\rho},\tilde{z},\tilde{t}+\Delta\tilde{t}) = \mathcal{H}_0^{-1}\left\{\exp\left(-\frac{i}{2}\tilde{k}_\rho^2\Delta\tilde{t}\right)\,\mathcal{H}_0\left\{\tilde{\psi}(\tilde{\rho},\tilde{z},\tilde{t})\right\}\right\}. \tag{663}$$

10.2.8 Calculating a Cylindrical Laplacian with the Hankel Transform

Once we have access to the Bessel functions (which are built into Matlab with the `besselj()` command), there are several algorithms for calculating the Hankel transform.

- The simplest option would be to brute force directly integrate the Hankel transform formula, but this is not particularly efficient.
- A "quasi-fast Hankel transform" algorithm was presented by Siegman [58]. This approach approach uses a logarithmic change of variables to turn the Hankel transform into a convolution, which can then be done efficiently using Fourier transforms (with FFT). However, the algorithm requires a logarithmically spaced grid, which results in over-sampling at small radii. This apporach can also lead to accumulation of errors in the energy over many time steps. Siegman's algorithm was applied to the Schrodinger equation by Bisseling and Kosloff [59].
- A method for calculating the zero-order Hankel transform numerically which better preserves the energy was presented by Yu *et al* [60], which requires that the grid be sampled at the zeroes

of the Bessel function. This was then extended to higher order Hankel transforms by Guizar-Sicairos [61], and the algorithm was then applied to the Gross-Pitaevskii equation by Ronen [62], and is now used in the XMDS2 differential equation solver.

11 Gross-Pitaevskii Equation in One Dimension

The full 3D GPE is given by

$$i\hbar\frac{\partial\psi(\mathbf{r},t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g_{3\text{D}}|\psi(\mathbf{r},t)|^2\right)\psi(\mathbf{r},t). \tag{664}$$

It has been shown in the theoretically in the literature [63] and measured experimentally [64] that if $\mu \ll \hbar\omega_\perp$, then the transverse dynamics are frozen out and the wavefunction takes on the form of a Gaussian harmonic oscillator ground state along this dimension. We can then write the full wavefunction as a separable product

$$\psi(\mathbf{r},t) = \Phi(x,y)f(z,t) \tag{665}$$

where

$$\Phi(x,y) = \frac{1}{\sqrt{\pi}a_\perp}\exp\left(-\frac{(x^2+y^2)}{2a_\perp^2}\right), \tag{666}$$

and the wavefunctions Φ and f are both normalised to unity. In this case, the 3D equation itself can be integrated over the transverse directions, leading to a 1D version of the GPE for the axial wavefunction only

$$i\hbar\frac{\partial f(z,t)}{\partial t} = \left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial z^2} + V_{\text{ext}}(z) + g_{1\text{D}}|f(z,t)|^2\right)f(z,t). \tag{667}$$

where the effective 1D interaction coupling constant is given by

$$g_{1\text{D}} = \frac{g_{3\text{D}}}{2\pi a_\perp^2}\left(1 - \mathcal{C}\frac{a_s}{\sqrt{2}a_\perp}\right)^{-1} \approx \frac{g_{3\text{D}}}{2\pi a_\perp^2} = \frac{2\hbar^2 a_s}{m a_\perp^2} = 2\hbar\omega_\perp a_s, \tag{668}$$

with the constant $\mathcal{C} = 1.4603$, and we can see that the 1D coupling constant is essentially the 3D one averaged over the transverse dimensions. This approach assumes that the scattering between atoms is essentially still of a 3D nature, due to the fact that the typical transverse ground state size is around $a_\perp = 340\ \text{nm}$ (for a 1 kHz trap frequency) and the scattering length is much smaller at $a_s = 5.24\ \text{nm}$.

11.1 Dimensionless Form

Similar to the approach taken with the 3D GPE, we can scale the quantities so that

$$\tilde{z} = z/l \qquad \tilde{t} = \omega_\perp t \qquad \tilde{f}(\tilde{z},\tilde{t}) = \sqrt{l}\,f(z,t), \tag{669}$$

where the tilde notation denotes the scaled version of the variable, and the length scale is given by the harmonic oscillator length in the radial direction

$$l = \sqrt{\frac{\hbar}{m\omega_\perp}}. \tag{670}$$

Now we have

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial \tilde{t}} \frac{\partial \tilde{t}}{\partial t}, \tag{671}$$

and since

$$\tilde{t} = \omega_\perp t \implies \frac{\partial \tilde{t}}{\partial t} = \omega_\perp \tag{672}$$

this becomes

$$\begin{aligned}
\frac{\partial f}{\partial t} &= \omega_\perp \frac{\partial f}{\partial \tilde{t}} && (673)\\
&= \omega_\perp \frac{\partial}{\partial \tilde{t}} \left(\frac{1}{\sqrt{l}} \tilde{f} \right) && (674)\\
&= \frac{\omega_\perp}{\sqrt{l}} \frac{\partial \tilde{f}}{\partial \tilde{t}}. && (675)
\end{aligned}$$

We can follow a similar procedure to evaluate the spatial derivative in z

$$\begin{aligned}
\frac{\partial^2 f}{\partial z^2} &= \frac{\partial}{\partial z} \left(\frac{\partial f}{\partial z} \right) && (676)\\
&= \frac{\partial}{\partial z} \left(\frac{\partial f}{\partial \tilde{z}} \frac{\partial \tilde{z}}{\partial z} \right) && (677)\\
&= \frac{\partial}{\partial z} \left(\frac{\partial f}{\partial \tilde{z}} \frac{1}{l} \right) && (678)\\
&= \frac{1}{l} \frac{\partial}{\partial z} \left(\frac{\partial f}{\partial \tilde{z}} \right) && (679)\\
&= \frac{1}{l} \frac{\partial}{\partial \tilde{z}} \left(\frac{\partial f}{\partial \tilde{z}} \right) \frac{\partial \tilde{z}}{\partial z} && (680)\\
&= \frac{1}{l} \frac{\partial}{\partial \tilde{z}} \left(\frac{\partial f}{\partial \tilde{z}} \right) \frac{1}{l} && (681)\\
&= \frac{1}{l^2} \frac{\partial^2 f}{\partial \tilde{z}^2} && (682)\\
&= \frac{1}{l^2} \frac{\partial^2}{\partial \tilde{z}^2} \left(\frac{1}{\sqrt{l}} \tilde{f} \right) && (683)\\
&= l^{-5/2} \frac{\partial^2 \tilde{f}}{\partial \tilde{z}^2}. && (684)
\end{aligned}$$

The external potential term becomes simply

$$V_{\text{ext}}\, f = \frac{1}{\sqrt{l}} V_{\text{ext}}\, \tilde{f}. \tag{685}$$

Finally the non-linear interaction term is

$$g_{1D}|f|^2 f = g_{1D}\left|\frac{1}{\sqrt{l}}\tilde{f}\right|^2 \frac{1}{\sqrt{l}}\tilde{f} = l^{-3/2}\, g_{1D}|\tilde{f}|^2\tilde{f}. \tag{686}$$

All of these converted terms can now be inserted into the 1D version of the GPE

$$\overset{z\to\tilde{z}}{\Longrightarrow}\; i\hbar\frac{\omega_\perp}{\sqrt{l}}\frac{\partial\tilde{f}}{\partial\tilde{t}} = -\frac{\hbar^2}{2m}\frac{1}{\sqrt{l^5}}\frac{\partial^2\tilde{f}}{\partial\tilde{z}^2} + \frac{1}{\sqrt{l}}\,V_{\text{ext}}\,\tilde{f} + \frac{1}{\sqrt{l^3}}\,g_{1D}|\tilde{f}|^2\tilde{f}. \tag{687}$$

Now inserting the expression $l = \sqrt{\hbar/m\omega_\perp}$, dividing both sides of the equation by $\left(\hbar^3\omega_\perp^5 m\right)^{1/4}$, and using the expression for the 1D interaction constant $g_{1D} = 2\hbar\omega_\perp a_s$, we finally obtain the scaled form of the 1D GPE

$$i\frac{\partial\tilde{f}}{\partial\tilde{t}} = -\frac{1}{2}\frac{\partial^2\tilde{f}}{\partial\tilde{z}^2} + \frac{V_{\text{ext}}}{\hbar\omega_\perp}\tilde{f} + \mathcal{G}_{1D}|\tilde{f}|^2\tilde{f}, \tag{688}$$

where we have defined the strength of the non-linear term in terms of the scattering length a_s with the parameter

$$\mathcal{G}_{1D} = \frac{2a_s N_{\text{tot}}}{l}. \tag{689}$$

In the scaled 1D GPE, we can see that the external potential term has been scaled into units of $\hbar\omega_\perp$, the harmonic oscillator energy level spacing in the transverse direction. The atom number N_{tot} has also been included in the non-linear term, which changes the normalisation such that the modulus squared of the wavefunction must integrate to unity (note that changing to scaled cooridnates doesn't change the normalisation because all the factor of l cancel)

$$\int_{-\infty}^{+\infty} |\tilde{f}|^2\,\mathrm{d}\tilde{z} = 1. \tag{690}$$

If in addition the external potential is harmonic, this term can be written as

$$\begin{aligned}
\frac{V_{\text{ext}}}{\hbar\omega_\perp}\,\tilde{f} &= \frac{1}{\hbar\omega_\perp}\left[\frac{1}{2}m\omega_z^2 z^2\right]\tilde{f} && (691)\\
&= \frac{1}{\hbar\omega_\perp}\left[\frac{1}{2}m\omega_z^2 (l\tilde{z})^2\right]\tilde{f} && (692)\\
&= \frac{1}{\hbar\omega_\perp}\left[\frac{1}{2}m\omega_z^2\frac{\hbar}{m\omega_\perp}\tilde{z}^2\right]\tilde{f} && (693)\\
&= \frac{1}{2}\left(\frac{\omega_z}{\omega_\perp}\right)^2\tilde{z}^2\tilde{f} && (694)\\
&= \frac{1}{2}\lambda^2\tilde{z}^2\tilde{f} && (695)\\
&= \tilde{V}_{\text{ext}}\tilde{f}. && (696)
\end{aligned}$$

where the aspect ratio of the the 1D trap is given by $\lambda = \omega_z/\omega_\perp$, and I have defined a scaled version of the harmonic potential $\tilde{V}_{\text{ext}}$. The 1D GPE for a harmonic external trapping potential can finally be

written as

$$\boxed{i\frac{\partial \tilde{f}}{\partial \tilde{t}} = -\frac{1}{2}\frac{\partial^2 \tilde{f}}{\partial \tilde{z}^2} + \frac{1}{2}\lambda^2 \tilde{z}^2 \tilde{f} + \mathcal{G}_{1\mathrm{D}}|\tilde{f}|^2 \tilde{f}} \tag{697}$$

11.2 Energy Expressions

The energy of the system is obtained by taking the expectation value of the Hamiltonian operator, which we can write as

$$E = \langle f|\hat{H}|f\rangle \tag{698}$$

where the Hamiltonian operator is the sum of the kinetic, potential, and interaction contributions

$$\hat{H} = \hat{H}_{\mathrm{kin}} + \hat{H}_{\mathrm{pot}} + \hat{H}_{\mathrm{int}} \tag{699}$$

$$= \underbrace{-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial z^2}}_{\text{kinetic}} + \underbrace{V_{\mathrm{ext}}}_{\text{potential}} + \underbrace{\frac{1}{2}g_{1\mathrm{D}}|f(z,t)|^2}_{\text{interaction}}. \tag{700}$$

Note that when evaluating the energy of the system, an additional factor of $1/2$ must be included as above in the interaction energy term, to ensure interaction energy between two particles is not counted twice (see, e.g. Dalfovo - *Theory of Bose-Einstein condensation in trapped gases*, or Castin - *Bose-Einstein condensates in atomic gases: simple theoretical results*). The total energy is given by the sum of the individual contributions

$$E = E_{\mathrm{kin}} + E_{\mathrm{pot}} + E_{\mathrm{int}}. \tag{701}$$

The contribution of the kinetic energy term to the total energy is given by

$$E_{\mathrm{kin}} = \langle f|\hat{H}_{\mathrm{kin}}|f\rangle \tag{702}$$

$$= \int_{-\infty}^{+\infty} f^* \Big\{ -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial z^2} \Big\} f \, \mathrm{d}z \tag{703}$$

$$= -\frac{\hbar^2}{2m}\int_{-\infty}^{+\infty} f^* \frac{\partial^2 f}{\partial z^2} \, \mathrm{d}z. \tag{704}$$

This second derivative can be calculated spectrally using the Fourier transform along the z-direction

$$\frac{\partial^2 f(z)}{\partial z^2} = \mathcal{F}_z^{-1}\Big\{ -k_z^2 \, \mathcal{F}_z\big\{f(z)\big\}\Big\} \tag{705}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{+\infty} \Big\{ -k_z^2 \int_{-\infty}^{+\infty} f(z)\, e^{-ik_z z} \, \mathrm{d}z \Big\} e^{ik_z z} \, \mathrm{d}k_z. \tag{706}$$

Changing variables on the integrals from angular spatial frequency to spatial frequency $k_z = 2\pi\nu_z$ gives

$$\frac{\partial^2 f(z)}{\partial z^2} = \int_{-\infty}^{+\infty} \left\{ -k_z^2 \int_{-\infty}^{+\infty} f(z)\, e^{-2\pi i \nu_z z}\, \mathrm{d}z \right\} e^{2\pi i \nu_z z}\, \mathrm{d}\nu_z, \tag{707}$$

where the factor of 2π has cancelled because $\mathrm{d}k_z = 2\pi\, \mathrm{d}\nu_z$. The reason for changing variables to ν is because this allows direct use of standard fast Fourier transform (FFT) algorithms to be used when the problem is discretised onto a spatial grid. Specifically, discretising the above expression, and approximating the integrals by Riemann summations, leads to

$$\frac{\partial^2 f(z)}{\partial z^2} \approx \sum_{-\infty}^{+\infty} \left\{ -k_z^2 \sum_{-\infty}^{+\infty} f(z)\, e^{-2\pi i \nu_z z} \Delta z \right\} e^{2\pi i \nu_z z} \Delta\nu_z \tag{708}$$

$$= \Delta z \Delta\nu_z \sum_{-\infty}^{+\infty} \left\{ -k_z^2 \sum_{-\infty}^{+\infty} f(z)\, e^{-2\pi i \nu_z z} \right\} e^{2\pi i \nu_z z} \tag{709}$$

$$= \frac{1}{N_z} \sum_{-\infty}^{+\infty} \left\{ -k_z^2 \sum_{-\infty}^{+\infty} f(z)\, e^{-2\pi i \nu_z z} \right\} e^{2\pi i \nu_z z}, \tag{710}$$

where I have used the fact that the separation between points in frequency space (the spatial frequency resolution) is given by $\Delta\nu_z = 1/(\Delta z N_z)$, and N_z is the number of points in the vector. If you look at the Matlab documentation page for FFT, you will see that the sums over exponentials above are exactly what is carried out by the commands FFT() and IFFT(). In addition, the factor of $1/N_z$ is also built into Matlab's IFFT() command, so this is taken care of. To evaluate the above expression therefore, the code would be of the form

$$\frac{\partial^2 f(z)}{\partial z^2} = \mathrm{IFFT}\Big[-k_z^2 \cdot \mathrm{FFT}\big[f\big] \Big], \tag{711}$$

and so the kinetic energy contribution is calculated using

$$E_{\mathrm{kin}} = -\frac{\hbar^2}{2m} \int_{-\infty}^{+\infty} f^* \cdot \mathrm{IFFT}\Big[-k_z^2 \cdot \mathrm{FFT}\big[f\big] \Big]\, \mathrm{d}z. \tag{712}$$

The integral could be carried out using any accurate approach (such as trapezium rule, Simpson's rule, or quadrature methods), but if a simple Riemann sum was used then the energy would be calculated as

$$E_{\mathrm{kin}} \approx -\frac{\hbar^2 \Delta z}{2m} \sum_{-\infty}^{+\infty} f^* \cdot \mathrm{IFFT}\Big[-k_z^2 \cdot \mathrm{FFT}\big[f\big] \Big]. \tag{713}$$

A Code for Finding GPE Ground States in 3D Cartesian Coordinates Using the Imaginary Time Method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------- Imaginary Time GPE Evolution -----------%
%---(for non-interacting system in harmonic trap)--%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;clc;close all;

c.hbar = 1.055e-34;        % Reduced Planck constant
c.a0 = 5.291772e-11;       % Bohr radius
c.aRb87 = 98.98*c.a0;      % Scattering length, Rb87 S=1 triplet state
                           % (van Kempen, Phys. Rev. Lett. 88 (2002),p. 093201)
c.amu = 1.660539e-27;      % Atomic mass unit
c.mRb87 = 86.909181*c.amu; % Mass of Rb87 (see D. Steck, Rb87 D Line Data)
c.g_int3D = 4*pi*c.hbar^2*c.aRb87/c.mRb87; % 3D interaction coupling constant

%%% Input parameters in SI units %%%
N = 1000;             % Number of particles in BEC
omega_x = 2*pi*50;    % x trap frequency in radians per second = 2*pi*Hz
omega_y = 2*pi*100;   % y trap frequency in radians per second = 2*pi*Hz
omega_z = 2*pi*20;    % z trap frequency in radians per second = 2*pi*Hz

%%% Dimensionless parameters %%%
kappa = omega_y/omega_x;                % Anisotropy parameter 1
lambda = omega_z/omega_x;               % Anisotropy parameter 2
l = sqrt( c.hbar/(c.mRb87*omega_x) );   % Unit of length ( harmonic oscillator size in x )
G = 0;                                  % Scaled 3D non-linearity for non-interacting case
% G = 4*pi*c.aRb87*N/l;                 % Use this to include interactions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------- Imaginary Time Grid ----------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt = 0.005;      % Time step
Nt = 1000;       % Number of time iterations
T = (Nt-1)*dt;   % Duration of total time window
t = 0:dt:T;      % Define time vector


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------------- Spatial Grids -----------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nx = 64;    % Number of grid points ( make these power of 2 for FFT speed )
dx = 0.25;  % Grid spacing
Ny = 64;
dy = 0.25;
Nz = 64;
dz = 0.25;


```

```
% Create grid vectors and associated k-vectors
[x, kx] = grid_vectors(Nx, dx);
[y, ky] = grid_vectors(Ny, dy);
[z, kz] = grid_vectors(Nz, dz);
[~,x0_ind] = min(abs(x));  % Get indices of zero points for later slicing
[~,y0_ind] = min(abs(y));  % ( should be (Nz/2 + 1) if Nx is even )
[~,z0_ind] = min(abs(z));  %

[X, Y, Z]  = meshgrid(x, y, z);
[KX,KY,KZ] = meshgrid(kx,ky,kz);
Ksq = KX.^2 + KY.^2 + KZ.^2;
Ksq = ifftshift(Ksq); % Ifftshift the k-vector instead of shifting psi in the main loop

kx = permute(ifftshift(kx), [1 2 3]); % Reshape vectors so that Matlab's
ky = permute(ifftshift(ky), [2 1 3]); % implicit expansion can be used.
kz = permute(ifftshift(kz), [3 1 2]);

%%% External trapping potential %%%
V_ext = (1/2)*(X.^2 + kappa^2*Y.^2 + lambda^2*Z.^2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------- Initial Condition ----------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% A gaussian, but some size which is not the ground state %%%
a_x_initial = 1.5;
a_y_initial = 1.7;
a_z_initial = 1.3;

psi_initial = 1/sqrt( pi^(3/2) * a_x_initial * a_y_initial * a_z_initial )*...
    exp( -X.^2/(2*a_x_initial^2) -Y.^2/(2*a_y_initial^2) -Z.^2/(2*a_z_initial^2) );
n_initial = abs(psi_initial).^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------------ Set Up Plots ------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n_analytic = calculate_n_analytic(X*l, Y*l, Z*l,omega_x,omega_y,omega_z,N,c);

figure;
subplot(1,3,1);hold all;
p1 = plot(x, squeeze( n_initial(y0_ind,:,z0_ind) ) ,'o-');
plot(x, squeeze( n_analytic(y0_ind,:,z0_ind) )/N*l^3); xlabel('x')
legend('Numerical','Analytic')
subplot(1,3,2);hold all;
p2 = plot(y, squeeze( n_initial(:,x0_ind,z0_ind) ) ,'o-');
plot(y, squeeze( n_analytic(:,x0_ind,z0_ind) )/N*l^3); xlabel('y')
subplot(1,3,3);hold all;
p3 = plot(z, squeeze( n_initial(y0_ind,x0_ind,:) ) ,'o-');
plot(z, squeeze( n_analytic(y0_ind,x0_ind,:)/N*l^3 )); xlabel('z')
drawnow()
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------- Set up storage vectors/matrices ---------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dStore = 5;                                % Only store every 'dStore' iterations
num_to_store = ceil(Nt/dStore);            % Total number of iterations that will be stored
t_store = zeros( 1, num_to_store );        % Preallocate storage arrays
Etot_store = zeros( 7, num_to_store );     % Preallocate storage arrays
mu_store = zeros( 1, num_to_store );       % Preallocate storage arrays

%%% Store the first value in each storage matrix %%%
store_ind = 1;
t_store(store_ind) = t(1);
[Etot_store(:,store_ind), mu_store(store_ind)] = calculate_energy_and_mu(...
    psi_initial, dx, dy, dz, kx, ky, kz, V_ext, G);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------ Main Time Stepping Loop -------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_k = psi_initial; % Initiallize the wavefunction to be iterated
tic
for k = 2:Nt

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% Split-Step Spectral Evolution (using Strang Splitting) %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Half step with potential operator
    psi_k = exp( -dt/2*(  V_ext + G*abs(psi_k).^2  ) ).*psi_k;

    % Full step with kinetic operator (spectral derivative)
    psihat_k = fftn( ifftshift(psi_k) );
    psihat_k = exp( -dt/2 * Ksq ).*psihat_k;
    psi_k = fftshift( ifftn(psihat_k) );

    % Half step with potential operator
    psi_k = exp( -dt/2*(  V_ext + G*abs(psi_k).^2  ) ).*psi_k;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Renormalise the wavefunction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    INT_psi = sum(sum(sum( abs(psi_k).^2 )))*dx*dy*dz;
    psi_k = psi_k / sqrt(INT_psi);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Store Values Every "dStore" Iterations %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if mod(k,dStore) == 1 || dStore == 1
        store_ind = store_ind + 1;
        t_store(store_ind) = t(k);
        [E, mu] = calculate_energy_and_mu(psi_k, dx, dy, dz,...
            kx, ky, kz, V_ext, G);
        fprintf('Storing iteration %i of %i: E = %0.10f, mu = %0.10f \n',k, Nt, E(7),mu);
        Etot_store(:,store_ind) = E;
        mu_store(:,store_ind) = mu;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
155
156         %%% Update Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
157         if mod(k,floor(Nt/10)) == 0
158             n_k = abs(psi_k).^2;
159             set(p1,'YData', squeeze( n_k(y0_ind,:,z0_ind) ) );
160             set(p2,'YData', squeeze( n_k(:,x0_ind,z0_ind) ) );
161             set(p3,'YData', squeeze( n_k(y0_ind,x0_ind,:) ) );
162             drawnow
163         end
164         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
165
166     end
167     toc
168
169     [E_final, mu_final] = calculate_energy_and_mu(psi_k, dx, dy, dz, ...
170         kx, ky, kz, V_ext, G);
171     fprintf('Final values: E = %0.10f, mu = %0.10f \n', E_final(7,1), mu_final);
172
173     figure;hold all;
174     plot(t_store, Etot_store')
175     plot(t_store, mu_store)
176     xlabel('Imaginary Time');ylabel('Energy')
177
178
179     function [x, kx] = grid_vectors(Nx, dx)
180     %%% Function for generating real space and frequency vectors in each dirn.
181
182     Fsx = 1/dx;          % Sampling frequency
183     dfx = Fsx/Nx;        % Frequency resolution
184
185     x  = -(Nx*dx/2):dx:(Nx*dx/2-dx); % Real space vector
186     fx = -(Fsx/2):dfx:(Fsx/2-dfx);   % Spatial frequency vector
187
188     kx = 2*pi*fx;        % Angular spatial frequency k vector
189
190     end
191
192
193     function [E, mu] = calculate_energy_and_mu(psi, dx, dy, dz, kx, ky, kz, V_ext, G)
194     %%% Calculate total energy of wavefunction (sum of kinetic, potential and
195     %%% interaction energies). Obtained using the expectation value of the
196     %%% Hamiltonian operator. Note, interaction energy includes a factor of
197     %%% 1/2 for double counting (see Dalfovo Rev. Mod. Phys 71 1999, Eq. 37),
198     %%% but the calculation of chemical potential does not.
199
200     dV = dx*dy*dz;
201
202     %%% Kinetic Energy %%%%%%%%%%%%%%%%%%%
203     psi_hat = fftn( ifftshift(psi) );
204     E_kin_x = 1/2 * sum(sum(sum(  abs( ifftn( 1i*kx.*psi_hat ) ).^2 )))*dV;
205     E_kin_y = 1/2 * sum(sum(sum(  abs( ifftn( 1i*ky.*psi_hat ) ).^2 )))*dV;
206     E_kin_z = 1/2 * sum(sum(sum(  abs( ifftn( 1i*kz.*psi_hat ) ).^2 )))*dV;
207     E_kin = E_kin_x + E_kin_y + E_kin_z;
208
```

```
%%% External Potential Energy %%%%%%%
psi_sq = abs(psi).^2;
E_pot = sum(sum(sum( V_ext.*psi_sq )))*dV;


%%% Mean Field Interaction Energy %%%
E_int = G/2 * sum(sum(sum( psi_sq.^2 )))*dV;

%%% Total Energy %%%%%%%%%%%%%%%%%%%%
E_tot = E_kin + E_pot + E_int;
E = [E_kin_x ; E_kin_y ; E_kin_z ; E_kin ; E_pot ; E_int ; E_tot];

%%% Chemical Potential %%%%%%%%%%%%%%
mu = E_kin + E_pot + 2*E_int;

end

function n_analytic = calculate_n_analytic(X,Y,Z,omega_x,omega_y,omega_z,N,c)
%%% Calculate 3D Non-Interacting limit (3D Gaussian)

ax = sqrt( c.hbar/(c.mRb87*omega_x) );
ay = sqrt( c.hbar/(c.mRb87*omega_y) );
az = sqrt( c.hbar/(c.mRb87*omega_z) );
psi_analytic = sqrt( 1/( pi^(3/2)*ax*ay*az ) ) * ...
    exp( -X.^2/(2*ax^2) - Y.^2/(2*ay^2) - Z.^2/(2*az^2) );
n_analytic = N * abs(psi_analytic).^2;

end
```

B Code for Calculating Wavefunction After Free-Flight Expansion Using Deuar's Method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Free Flight Prescriptor %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (according to E. 42 of Deuar,P., Comp. Phys. Comm. 208, 92-102 (2016) %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xf, yf, zf, psi_out] = free_flight(psi_in, ...
    dxo, dyo, dzo, mult_x, mult_y, mult_z, t_flight, UseGPU)

% suffix = o : denotes "original" lattice variables
% suffix = f : denotes "final" lattice variables

tic
fprintf('Running free flight evolution... \n');

Nx = size(psi_in,2);
Ny = size(psi_in,1);
Nz = size(psi_in,3);
M = Nx * Ny * Nz;  % total number of grid points

Nx_32 = uint32( Nx );
Ny_32 = uint32( Ny );
Nz_32 = uint32( Nz );
M_32 = uint32( M );
IND = uint32( 0:(M_32-1) );

num_mu = mult_x * mult_y * mult_z;   % number of terms in sum

dxf = dxo * mult_x;   % Final lattice real space grid spacings
dyf = dyo * mult_y;
dzf = dzo * mult_z;

Lxo = dxo * Nx;       % Original lattice box lengths
Lyo = dyo * Ny;
Lzo = dzo * Nz;

Lxf = Lxo * mult_x;   % Final lattice box lengths
Lyf = Lyo * mult_y;
Lzf = Lzo * mult_z;

dkxo = 2*pi / Lxo;    % Original lattice k space grid spacings
dkyo = 2*pi / Lyo;
dkzo = 2*pi / Lzo;

dkxf = dkxo / mult_x; % Final lattice k space grid spacings
dkyf = dkyo / mult_y;
dkzf = dkzo / mult_z;

axo = -Lxo / 2;       % Original lattice offsets
```

```
ayo = -Lyo / 2;
azo = -Lzo / 2;

axf = - Lxf / 2;      % Final lattice offsets
ayf = - Lyf / 2;
azf = - Lzf / 2;

xo = ( 0:dxo:(Nx-1)*dxo ) + axo; % Original lattice real space grid vectors
yo = ( 0:dyo:(Ny-1)*dyo ) + ayo;
zo = ( 0:dzo:(Nz-1)*dzo ) + azo;

% Change dimension orientation to allow implicit expansion in Matlab
xo = permute(xo, [1 2 3]); % Creates a [1 x Nx x 1] vector
yo = permute(yo, [2 1 3]); % Creates a [Ny x 1 x 1] vector
zo = permute(zo, [3 1 2]); % Creates a [1 x 1 x Nz] vector

xf = (xo * mult_x); % Final lattice real space grid vectors
yf = (yo * mult_y);
zf = (zo * mult_z);

% Final lattice k space grid vectors
kxf = ifftshift( (-pi/dxf):dkxf:(pi/dxf - dkxf) );
kyf = ifftshift( (-pi/dyf):dkyf:(pi/dyf - dkyf) );
kzf = ifftshift( (-pi/dzf):dkzf:(pi/dzf - dkzf) );

kxo = kxf * mult_x; % Original lattice k space grid vectors
kyo = kyf * mult_y;
kzo = kzf * mult_z;

% Change dimension orientation to allow implicit expansion in Matlab
kxo = permute(kxo, [1 2 3]); % Creates a [1 x Nx x 1] vector
kyo = permute(kyo, [2 1 3]); % Creates a [Ny x 1 x 1] vector
kzo = permute(kzo, [3 1 2]); % Creates a [1 x 1 x Nz] vector

%%% PREFACTORS WHICH DO NOT DEPEND ON LOOP VARIABLES %%%
Ekfacx = exp( -1i/2 * t_flight * kxo.^2 );  % Prefactors used in Eq. (42c)
Ekfacy = exp( -1i/2 * t_flight * kyo.^2 );
Ekfacz = exp( -1i/2 * t_flight * kzo.^2 );
kdafacx = exp( 1i * kxo * (axf-axo) );       % Prefactors used in Eq. (42c)
kdafacy = exp( 1i * kyo * (ayf-ayo) );
kdafacz = exp( 1i * kzo * (azf-azo) );
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

psi_out = zeros(Ny,Nx,Nz); % Preallocate final wavefunction array

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%------------ Move arrays to GPU memory -----------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if UseGPU
    try
        gpuArray(1);
        UseGPU = 1;
        fprintf('GPU device available to use.\n')
```

```
        psi_in = gpuArray(psi_in);
        psi_out = gpuArray(psi_out);
        xo = gpuArray(xo);
        yo = gpuArray(yo);
        zo = gpuArray(zo);
        kxo = gpuArray(kxo);
        kyo = gpuArray(kyo);
        kzo = gpuArray(kzo);
        xf = gpuArray(xf);
        yf = gpuArray(yf);
        zf = gpuArray(zf);
    catch ME
        UseGPU = 0;
        fprintf('GPU device not available to use.\n')
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for mu_x = 0:(mult_x-1)
    for mu_y = 0:(mult_y-1)
        for mu_z = 0:(mult_z-1)

            %%% PREFACTORS WHICH DEPEND ON LOOP VARIABLES %%%
            % Prefactors used in exponential in Eq. (42d)
            xofac = exp( -1i*xo*mu_x*dkxo/mult_x );
            yofac = exp( -1i*yo*mu_y*dkyo/mult_y );
            zofac = exp( -1i*zo*mu_z*dkzo/mult_z );
            % Prefactors used in exponential in Eq. (42c)
            kxofac = exp( -1i*kxo*t_flight*mu_x*dkxo/mult_x );
            kyofac = exp( -1i*kyo*t_flight*mu_y*dkyo/mult_y );
            kzofac = exp( -1i*kzo*t_flight*mu_z*dkzo/mult_z );
            % Prefactors used in exponential in Eq. (42b)
            xffac  = exp( 1i*xf*mu_x*dkxo/mult_x );
            yffac  = exp( 1i*yf*mu_y*dkyo/mult_y );
            zffac  = exp( 1i*zf*mu_z*dkzo/mult_z );
            % Prefactor used in exponential in Eq. (42b)
            dtfac  = exp( -1i/2 * t_flight *( (mu_x*dkxo/mult_x)^2 + ...
                        (mu_y*dkyo/mult_y)^2 + (mu_z*dkzo/mult_z)^2 ) );
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            fieldmu = psi_in .* xofac .* yofac .* zofac; % produces kernel for A in Eq.42d
            fieldmu = fftn(fieldmu);                          % produces A in Eq.42d
            fieldmu = fieldmu .* (Ekfacx.*Ekfacy.*Ekfacz) .* ...
                                 (kdafacx.*kdafacy.*kdafacz) .* ...
                                 (kxofac.*kyofac.*kzofac); % kernel for B in Eq.42c
            fieldmu = ifftn(fieldmu);                              % produces B in Eq.42c

            % Final part calculates f in Eq. 42b and accumulates to sum in Eq. 42a
            % idivide to avoid "double" casting, does "fix" rounding
            J1 = idivide( idivide(IND,Nz_32), Ny_32 );
            J2 = mod( idivide( IND, Nz_32 ), Ny_32 );
            J3 = mod( IND, Nz_32 );
            J1 = mod( mult_x*J1, Nx_32 );
```

```
            J2 = mod( mult_y*J2, Ny_32 );
            J3 = mod( mult_z*J3, Nz_32 );
            ind_prime = mod( ( J3 + Nz_32*(J2 + Ny_32*J1) ) , M_32 ) + 1;
            % Convert 3D form back to 1D form, ready to use ind_prime
            fieldmu = reshape( permute(fieldmu,[3,1,2]) , [M 1] );
            psi_out = psi_out +  permute( reshape( fieldmu(ind_prime), ...
                [Nz,Ny,Nx] ) , [2,3,1] ) .* (xffac .* yffac .* zffac) * dtfac;

            fprintf("Accumulated [%i %i %i] multiplier (%i of %li) ... (%.0f%% complete) \n",...
                mu_x,mu_y,mu_z, mu_z + mu_y*mult_z + mu_x*(mult_y*mult_z) +1 , num_mu, ...
                (mu_z + mu_y*mult_z + mu_x*(mult_y*mult_z) +1)/num_mu*100);
        end
    end
end

psi_out = psi_out / num_mu;  % Correct normalisation after lattice sums

% Bring results back from GPU, if using
if UseGPU
    psi_out = gather(psi_out);
    xf = gather(xf);
    yf = gather(yf);
    zf = gather(zf);
end

toc
end
```

C Code for Calculating Jacobi Polynomials

```
function [P,c] = jacobpol( a, b, n, x )

%JACOBPOL: Jacobi polynomial for order n and argument X.
%
% Usage:   [P,c] = jacobpol( a, b, n, X ) where P is the polynomial and c is a vector with
%           the polynomial coefficients.
%
%          EXAMPLE:
%           x = rand(5,5,5,5); P = jacobpol(1.5,-0.5,5,x); max( P(:) ), min( P(:) )
%           clear x P
%           x = linspace(-1,1,201);
%          for i = 1:5, P(i,:) = jacobpol(1.5,-0.5,i,x); end
%          plot(x,P), box on, grid on, axis([-1 1 -1.5 3])

%*************************************************************************************
% First version: Sat Dec 14 18:22:06 WET 2013
%
% Improvements? contact: orodrig@uaHg.pt
%
% Questions? Look at the references
%
% References:
% http://dHmf.nist.gov/18.5
% http://www.mymathlib.com/functions/orthogonal_polynomials/jacobi.html
% http://en.wikipedia.org/wiki/Jacobi_polynomials
% Abramowitz and Stegun, Chapter 22
%*************************************************************************************

% x in (-1,1)

P = []; c = [];

if n < 0
disp('Polynomial order < 0!')
disp('aborting calculations...')
return
end

if max( abs( x(:) ) ) > 1
disp('Found |x| > 1!')
disp('aborting calculations...')
return
end

if a <= -1
disp('alpha <= -1!')
disp('aborting calculations...')
return
end

```

```
if b <= -1
disp('beta <= -1!')
disp('aborting calculations...')
return
end

g = a + b;

P0 = ones( size( x ) );
c0 = 1;
P1 = 0.5*( (g+2)*x + a - b );
c1 = [0.5*(g+2),0.5*(a-b)];

switch n
case 0
P = P0;
c = c0;
case 1
P = P1;
c = c1;
otherwise

    k = 1;
    P_nmo = P0;
    P_n   = P1;

    while k < n
    A = 2*(k+1)*(k+g+1)*(2*k+g);
    B = (2*k+g+1)*( (2*k+g+2)*(2*k+g)*x + a*a - b*b );
    C = 2*(k+a)*(k+b)*(2*k+g+2);
    P = B.*P_n - C*P_nmo; P = P/A;
    P_nmo = P_n;
    P_n = P;
    k = k + 1;
    end

end

if n > 1

cnmo = c0;
cn   = c1;

     for k = 1:n-1
     A = 2*(k+1)*(k+g+1)*(2*k+g);
     C = 2*(k+a)*(k+b)*(2*k+g+2);
     c = conv([(2*k+g+1)*(2*k+g+2)*(2*k+g),(2*k+g+1)*( a*a-b*b )],cn)/A - [0 0 cnmo]*C/A;
     cnmo = cn;
     cn = c;
     end

end
```

References

[1] M. Edwards and K. Burnett, "Numerical solution of the nonlinear Schrödinger equation for small samples of trapped neutral atoms", *Phys. Rev. A* **51**, 1382–1386 (1995).

[2] P. A. Ruprecht et al., "Time-dependent solution of the nonlinear Schrödinger equation for Bose-condensed trapped neutral atoms", *Phys. Rev. A* **51**, 4704–4711 (1995).

[3] M. Holland and J. Cooper, "Expansion of a Bose-Einstein condensate in a harmonic potential", *Phys. Rev. A* **53**, R1954–R1957 (1996).

[4] M. J. Holland et al., "Emergence of Interaction Effects in Bose-Einstein Condensation", *Phys. Rev. Lett.* **78**, 3801–3805 (1997).

[5] F. Dalfovo and S. Stringari, "Bosons in anisotropic traps: Ground state and vortices", *Phys. Rev. A* **53**, 2477–2485 (1996).

[6] E. Cerboneschi et al., "Oscillation frequencies for a Bose condensate in a triaxial magnetic trap", *Phys. Lett. A* **249**, 495–500 (1998).

[7] A. Gammal, T. Frederico, and L. Tomio, "Improved numerical approach for the time-independent Gross-Pitaevskii nonlinear Schrödinger equation", *Phys. Rev. E* **60**, 2421–2424 (1999).

[8] W. Bao and W. Tang, "Ground-state solution of Bose–Einstein condensate by directly minimizing the energy functional", *J. Comput. Phys.* **187**, 230–254 (2003).

[9] F. Dalfovo et al., "Theory of Bose-Einstein condensation in trapped gases", *Rev. Mod. Phys.* **71**, 463–512 (1999).

[10] Y. Castin, "Bose-Einstein Condensates in Atomic Gases: Simple Theoretical Results", Coherent atomic matter waves, Vol. 72, edited by R. Kaiser, C. Westbrook, and F. David, Les Houches - Ecole d'Ete de Physique Theorique (2001), pp. 1–136.

[11] W. Bao, D. Jaksch, and P. A. Markowich, "Numerical solution of the Gross–Pitaevskii equation for Bose–Einstein condensation", *J. Comput. Phys.* **187**, 318–342 (2003).

[12] P. Muruganandam and S. Adhikari, "Fortran programs for the time-dependent Gross–Pitaevskii equation in a fully anisotropic trap", *Comput. Phys. Commun.* **180**, 1888–1912 (2009).

[13] D. A. Steck, *Quantum and Atom Optics*, University of Oregon lecture notes (2007), http://steck.us/teaching (visited on 09/22/2021).

[14] G. Strang, "On the Construction and Comparison of Difference Schemes", *SIAM J. Numer. Anal.* **5**, 506–517 (1968).

[15] H. Yoshida, "Construction of higher order symplectic integrators", *Phys. Lett. A* **150**, 262–268 (1990).

[16] A. D. Bandrauk and H. Shen, "High-order split-step exponential methods for solving coupled nonlinear Schrodinger equations", *J. Phys. A: Math. Gen.* **27**, 7147–7155 (1994).

[17] R. I. McLachlan and G. R. W. Quispel, "Splitting methods", *Acta Numer.* **11**, 341–434 (2002).

[18] G. Muslu and H. Erbay, "Higher-order split-step Fourier schemes for the generalized nonlinear Schrödinger equation", *Math. Comput. Simul.* **67**, 581–595 (2005).

[19] J. Javanainen and J. Ruostekoski, "Symbolic calculation in development of algorithms: split-step methods for the Gross–Pitaevskii equation", *J. Phys. A: Math. Gen.* **39**, L179–L184 (2006).

[20] C. C. Marston and G. G. Balint-Kurti, "The Fourier grid Hamiltonian method for bound state eigenvalues and eigenfunctions", *J. Chem. Phys.* **91**, 3571–3576 (1989).

[21] G. G. Balint-Kurti, R. N. Dixon, and C. C. Marston, "Grid methods for solving the Schrödinger equation and time dependent quantum dynamics of molecular photofragmentation and reactive scattering processes", *Int. Rev. Phys. Chem.* **11**, 317–344 (1992).

[22] J. A. Fleck, J. R. Morris, and M. D. Feit, "Time-dependent propagation of high energy laser beams through the atmosphere", *Appl. Phys.* **10**, 129–160 (1976).

[23] A. Korpel et al., "Split-step-type angular plane-wave spectrum method for the study of self-refractive effects in nonlinear wave propagation", *J. Opt. Soc. Am. B* **3**, 885–890 (1986).

[24] A. Putra et al., "Optimally focused cold atom systems obtained using density-density correlations", *Review of Scientific Instruments* **85**, 013110 (2014).

[25] T. R. Taha and M. I. Ablowitz, "Analytical and numerical aspects of certain nonlinear evolution equations. II. Numerical, nonlinear Schrödinger equation", *J. Comput. Phys.* **55**, 203–230 (1984).

[26] B. Jackson, J. F. McCann, and C. S. Adams, "Output coupling and flow of a dilute Bose-Einstein condensate", *J. Phys. B: At. Mol. Opt. Phys.* **31**, 4489–4499 (1998).

[27] J.-F. Mennemann et al., "Optimal control of Bose–Einstein condensates in three dimensions", *New J. Phys.* **17**, 113027 (2015).

[28] J.-F. Mennemann et al., "Relaxation in an extended bosonic Josephson junction", *Phys. Rev. Research* **3**, 023197 (2021).

[29] User:teeeeee, *Why do we have to rearrange a vector and shift the zero point to the first index, in preparation for an FFT?*, StackExchange Signal Processing question (April 2020), https://dsp.stackexchange.com/q/66716 (visited on 09/24/2021).

[30] Mathworks, *Matlab fftn documentation*, https://uk.mathworks.com/help/matlab/ref/fftn.html#bvhcsbe-7 (visited on 09/24/2021).

[31] R. Kosloff and H. Tal-Ezer, "A direct relaxation method for calculating eigenfunctions and eigenvalues of the Schrödinger equation on a grid", *Chem. Phys. Lett.* **127**, 223–230 (1986).

[32] 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**, 7438–7444 (2000).

[33] W. Bao and Q. Du, "Computing the ground state solution of Bose–Einstein condensates by a normalized gradient flow", *SIAM J. Sci. Comput.* **25**, 1674–1697 (2004).

[34] S. Stringari, "Collective Excitations of a Trapped Bose-Condensed Gas", *Phys. Rev. Lett.* **77**, 2360–2363 (1996).

[35] D. A. Smith et al., "Absorption imaging of ultracold atoms on atom chips", *Opt. Express* **19**, 8471–8485 (2011).

[36] G. Reinaudi et al., "Strong saturation absorption imaging of dense clouds of ultracold atoms", *Opt. Lett.* **32**, 3143–3145 (2007).

[37] J. Szczepkowski et al., "Analysis and calibration of absorptive images of Bose–Einstein condensate at nonzero temperatures", *Review of Scientific Instruments* **80**, 053103 (2009).

[38] E. G. M. van Kempen et al., "Interisotope Determination of Ultracold Rubidium Interactions from Three High-Precision Experiments", *Phys. Rev. Lett.* **88**, 093201 (2002).

[39] Y. Castin and R. Dum, "Bose-Einstein Condensates in Time Dependent Traps", *Phys. Rev. Lett.* **77**, 5315–5319 (1996).

[40] Y. Kagan, E. L. Surkov, and G. V. Shlyapnikov, "Evolution of a Bose-condensed gas under variations of the confining potential", *Phys. Rev. A* **54**, R1753–R1756 (1996).

[41] F. Dalfovo et al., "Nonlinear dynamics of a Bose condensed gas", *Phys. Lett. A* **227**, 259–264 (1997).

[42] A. Arnold, "Numerically Absorbing Boundary Conditions for Quantum Evolution Equations", *VLSI Design* **6**, 038298 (1900).

[43] Z. Xu and H. Han, "Absorbing boundary conditions for nonlinear Schrödinger equations", *Phys. Rev. E* **74**, 037704 (2006).

[44] X. Antoine, W. Bao, and C. Besse, "Computational methods for the dynamics of the nonlinear Schrödinger/Gross–Pitaevskii equations", *Comput. Phys. Commun.* **184**, 2621–2633 (2013).

[45] X. Antoine, E. Lorin, and Q. Tang, "A friendly review of absorbing boundary conditions and perfectly matched layers for classical and relativistic quantum waves equations", *Molecular Physics* **115**, 1861–1879 (2017).

[46] K. C. Kulander, "Multiphoton ionization of hydrogen: A time-dependent theory", *Phys. Rev. A* **35**, 445–447 (1987).

[47] J. L. Krause, K. J. Schafer, and K. C. Kulander, "Calculation of photoemission from atoms subject to intense laser fields", *Phys. Rev. A* **45**, 4998–5010 (1992).

[48] S. Chelkowski, C. Foisy, and A. D. Bandrauk, "Electron-nuclear dynamics of multiphoton H_2^+ dissociative ionization in intense laser fields", *Phys. Rev. A* **57**, 1176–1185 (1998).

[49] R. Santra and C. H. Greene, "Multiphoton ionization of xenon in the vuv regime", *Phys. Rev. A* **70**, 053401 (2004).

[50] D. Wells and H. Quiney, "A fast and adaptable method for high accuracy integration of the time-dependent Schrödinger equation", *Scientific Reports* **9**, 782 (2019).

[51] F. He, C. Ruiz, and A. Becker, "Absorbing boundaries in numerical solutions of the time-dependent Schrödinger equation on a grid using exterior complex scaling", *Phys. Rev. A* **75**, 053407 (2007).

[52] P. Deuar, "A tractable prescription for large-scale free flight expansion of wavefunctions", *Comput. Phys. Commun.* **208**, 92–102 (2016).

[53] D. S. Petrov, G. V. Shlyapnikov, and J. T. M. Walraven, "Phase-Fluctuating 3D Bose-Einstein Condensates in Elongated Traps", *Phys. Rev. Lett.* **87**, 050404 (2001).

[54] S. Dettmer et al., "Observation of Phase Fluctuations in Elongated Bose-Einstein Condensates", *Phys. Rev. Lett.* **87**, 160406 (2001).

[55] D. Hellweg et al., "Phase fluctuations in Bose–Einstein condensates", *Applied Physics B* **73**, 781–789 (2001).

[56] C. Mora and Y. Castin, "Extension of Bogoliubov theory to quasicondensates", *Phys. Rev. A* **67**, 053615 (2003).

[57] A. Imambekov et al., "Density ripples in expanding low-dimensional gases as a probe of correlations", *Phys. Rev. A* **80**, 033604 (2009).

[58] A. E. Siegman, "Quasi fast Hankel transform", *Opt. Lett.* **1**, 13–15 (1977).

[59] R. Bisseling and R. Kosloff, "The fast Hankel transform as a tool in the solution of the time dependent Schrödinger equation", *J. Comput. Phys.* **59**, 136–151 (1985).

[60] L. Yu et al., "Quasi-discrete Hankel transform", *Opt. Lett.* **23**, 409–411 (1998).

[61] M. Guizar-Sicairos and J. C. Gutiérrez-Vega, "Computation of quasi-discrete Hankel transforms of integer order for propagating optical wave fields", *J. Opt. Soc. Am. A* **21**, 53–58 (2004).

[62] S. Ronen, D. C. E. Bortolotti, and J. L. Bohn, "Bogoliubov modes of a dipolar condensate in a cylindrical trap", *Phys. Rev. A* **74**, 013623 (2006).

[63] M. Olshanii, "Atomic Scattering in the Presence of an External Confinement and a Gas of Impenetrable Bosons", *Phys. Rev. Lett.* **81**, 938–941 (1998).

[64] P. Krüger et al., "Weakly Interacting Bose Gas in the One-Dimensional Limit", *Phys. Rev. Lett.* **105**, 265302 (2010).