Wavepacket propagation in D-dimensional non-adiabatic crossings

Master Thesis

written by
Raoul Bourquin

supervised by
Dr. Vasile Grădinaru
and
Prof. Dr. Ralf Hiptmair

Seminar for Applied Mathematics
ETH Zurich

Spring semester 2012

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

The topic of this thesis is the time propagation of semi-classical wavepackets in special potentials which feature avoided crossings. We extend the formalism and the algorithms to the multi-dimensional case and finally show some applications. The work is a continuation of some earlier work [5, 1].
In this first chapter we introduce the basic situation and the various physical and mathematical ingredients. The main object of our studies is the time-dependent Schrödinger equation. The Hamiltonian of the physical systems in consideration consists of the usual kinetic operator part and a potential part. In the current studies this potential part has a special form and models energy hypersurfaces. We threat the fully non-adiabatic case where we allow multiple energy levels.

1.1 The non-adiabatic potential

First we introduce the potentials that are part of the Hamiltonian of our system. In the non-adiabatic case the potential $\mathbf{V}$ has multiple energy levels and the particles may switch between these surfaces. The *transition probabilities* are one of the things we are most interested in.
A potential with N energy levels is given by a symmetric real-valued $N \times N$ matrix:

$$\mathbf{V}(\underline{x}) := \begin{pmatrix} v_{0,0} & \cdots & v_{0,N-1} \\ \vdots & & \vdots \\ v_{N-1,0} & \cdots & v_{N-1,N-1} \end{pmatrix} \tag{1.1}$$

where each entry $v_{r,c}$ is a multi-variate function depending on $\underline{x}$. The energy levels are then given through the eigenvalues $\lambda_i(\underline{x})$ which we collect in the matrix:

$$\mathbf{\Lambda}(\underline{x}) := \begin{pmatrix} \lambda_0(\underline{x}) & & \\ & \ddots & \\ & & \lambda_{N-1}(\underline{x}) \end{pmatrix}. \tag{1.2}$$

These levels can of course intersect on a complicated subset of space. But we are most interested in the case where they do not touch but always keep a minimal distance between each other. This also excludes fully degenerate cases with

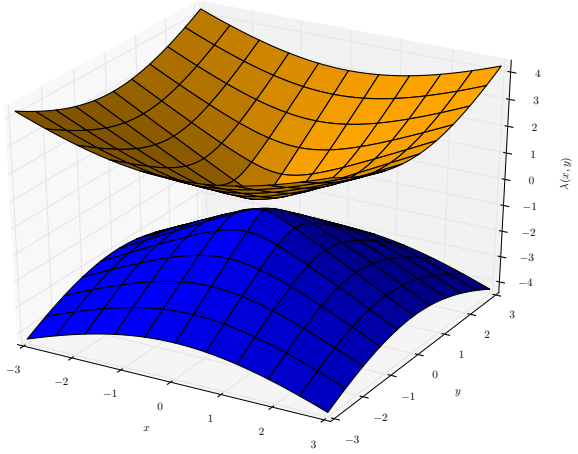


Figure 1.1: A two-dimensional potential having two energy levels with a conical avoided crossing. (Type 3 avoided crossing according to [9].)

eigenvalues of higher multiplicity. Therefore it is possible to sort the energy levels like:

$$\lambda_0(\underline{x}) > \lambda_1(\underline{x}) > \cdots > \lambda_i(\underline{x}) > \cdots > \lambda_{N-1}(\underline{x}) \tag{1.3}$$

independently of $\underline{x}$. The situation where two energy levels come closer to each other in a (small) region of space but do not intersect is called an *avoided crossing*. An example of a two-dimensional non-adiabatic potential with two energy levels is shown in figure 1.1.
Given a non-adiabatic potential by its matrix $\mathbf{V}(\underline{x})$ the diagonalisation process can be written as follows:

$$\mathbf{\Lambda}(\underline{x}) = \mathbf{M}^{-1}(\underline{x})\mathbf{V}(\underline{x})\mathbf{M}(\underline{x}) \tag{1.4}$$

and $\mathbf{\Lambda}(\underline{x})$ is the diagonalised version containing the different energy levels $\lambda_i(\underline{x})$. This is often called the adiabatic basis, exhibiting no coupling between the energy levels $\lambda_i(\underline{x})$. The linear transformation is then given by the matrix:

$$\mathbf{M}(\underline{x}) := \begin{pmatrix} \vdots & & \vdots \\ \nu_0(\underline{x}) & \cdots & \nu_{N-1}(\underline{x}) \\ \vdots & & \vdots \end{pmatrix} \tag{1.5}$$

where we stacked the eigenvectors ν_i as columns. We will need this eigentransformation later during the computation of observables.

1.2 The wavefunction

The next objects we introduce are *wavefunctions* which are omnipresent in quantum physics and chemistry. We work in D space dimensions of flat Euclidean

space. Let $x_0, x_1, \ldots, x_{D-1}$ denote the basis directions and $\underline{x} \in \mathbb{R}^D$ be an arbitrary point in this space.
The wavefunction ψ depends on position $\underline{x}$ and time t. Formally we define the wavefunction to be:

$$\begin{aligned} \psi : \mathbb{R}^D \times \mathbb{R} &\to \mathbb{C} \\ (\underline{x}, t) &\mapsto \psi(\underline{x}, t) = z \end{aligned} \tag{1.6}$$

and use the following Dirac braket notation:

$$|\psi\rangle := \psi(\underline{x}, t) = \psi(x_0, \ldots, x_{D-1}, t) \; . \tag{1.7}$$

Often we will omit the explicit time dependence and consider only a static picture at a fixed time t_0 where the wavefunction is given as:

$$\begin{aligned} \psi : \mathbb{R}^D &\to \mathbb{C} \\ \underline{x} &\mapsto \psi(\underline{x}) = z \, . \end{aligned} \tag{1.8}$$

For performing simulations within non-adiabatic potentials we need compatible wavefunctions. In our case this implies that the wavefunction must have N components:

$$\psi := \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{N-1} \end{pmatrix} \tag{1.9}$$

with each component ψ_i complying with (1.6). Such a wavefunction is appropriate for a N-level potential. After an eigentransformation we can interpret each ψ_i as the probability distribution on the energy level λ_i.

1.3 The time-dependent Schrödinger equation

The next subject to look at is the time-dependent Schrödinger equation. It is the governing equation for all time-dependent quantum mechanical phenomena. This equation describes the time evolution of a wavefunction. The partial differential equation can be written as:

$$i\hbar \frac{\partial}{\partial t} |\psi\rangle = H |\psi\rangle \tag{1.10}$$

where H denotes the Hamiltonian operator and $\hbar$ is the Planck constant. However we do not use this version directly but study a rescaled version. What we use is the so-called *semi-classical scaling* of the Schrödinger equation. Instead of $\hbar$ we introduce a real-valued parameter consistently denoted by ε which is strictly positive [1]. This parameter usually takes values in the range of about 0.001 up to 0.1. It is worthwhile to note that in the limit $\varepsilon \to 0$ we are back in the classical world and for bigger ε we get more and more quantum effects.

[1] Other authors use ε or even $\hbar$ (without its physical meaning and value) for the quantity we denote by ε^2.

The rescaled Schrödinger equation still keeps its mathematical form and looks like:

$$i\varepsilon^2 \frac{\partial}{\partial t} |\psi\rangle = H |\psi\rangle \ . \tag{1.11}$$

The Hamiltonian operator is given by:

$$H := T + V(\underline{x}) \tag{1.12}$$

where we obtain the kinetic operator T by the correspondence principle from classical mechanics. Start with the definition of the kinetic energy:

$$T := \frac{\underline{p} \cdot \underline{p}}{2m}$$

and set the mass $m = 1$ (absorb it into the scaling parameter). In one dimension the momentum operator is:

$$p := -i\varepsilon^2 \frac{\partial}{\partial x}$$

and we find that:

$$T := -\frac{\varepsilon^4}{2} \frac{\partial^2}{\partial x^2} \, .$$

In higher dimensions we have:

$$p_i := -i\varepsilon^2 \frac{\partial^2}{\partial x_i^2}$$

and in turn:

$$T := -\frac{\varepsilon^4}{2} \Delta \tag{1.13}$$

for the quantum mechanical equivalent.
In case of non-adiabatic potentials the Schrödinger equation becomes vector-valued and instead of (1.11) we should write:

$$i\varepsilon^2 \frac{\partial}{\partial t} \left| \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{N-1} \end{pmatrix} \right\rangle = \begin{pmatrix} \quad \\ \mathbf{H} \\ \quad \end{pmatrix} \left| \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{N-1} \end{pmatrix} \right\rangle \ . \tag{1.14}$$

The Hamiltonian operator is matrix-valued now:

$$\mathbf{H} := \mathbf{T} + \mathbf{V}(\underline{x}) \, . \tag{1.15}$$

The kinetic operator $\mathbf{T}$ is block-diagonal with each entry being of the form shown in (1.13).
We can solve the Schrödinger equation by a separation of variables ansatz and find the following result for the time propagation of any quantum state $|\psi(t)\rangle$:

$$|\psi(t)\rangle = \exp\left(-\frac{i}{\varepsilon^2}\mathbf{H}t\right) |\psi(0)\rangle \ . \tag{1.16}$$

However we can almost never compute the exponential of $\mathbf{H}$ in closed form. In the remaining chapters of this thesis we try to solve this equation by superior numerical methods.

Chapter 2

Wavefunction Propagation

In this chapter we take the first steps towards a numerical solution of the time-dependent Schrödinger equation. The techniques presented in this chapter are not new but can serve as reference point for other algorithms.

2.1 Evolution operator splitting

The time-evolution of a wavefunction $|\psi\rangle$ is given by the following formula which can be obtained from the full time-dependent equation by separation of variables:

$$|\psi(t)\rangle = \exp\left(-\frac{i}{\varepsilon^2}Ht\right)|\psi\rangle \ . \tag{2.1}$$

The exponential of the Hamiltonian operator is known as the time evolution operator. Computing its action is the main difficulty now. It is not feasible to compute the exponential straight away. We rewrite the operator as follows by inserting the definition of H:

$$\exp\left(-\frac{i}{\varepsilon^2}Ht\right) = \exp\left(-\frac{i}{\varepsilon^2}\left(T+V\right)t\right) = \exp\left(\left(-\frac{i}{\varepsilon^2}T - \frac{i}{\varepsilon^2}V\right)t\right) \ .$$

The last term can be put into the form $\exp\left(\left(A+B\right)t\right)$ for which we now can apply the Baker-Campbell-Hausdorff formula to compute a splitting into a product of commuting simpler exponentials:

$$\exp\left(\left(A+B\right)t\right) = \exp\left(At\right)\exp\left(Bt\right)\exp\left(-\frac{t^2}{2}[A,B]\right)$$

where we truncated the series after the $\mathcal{O}\left(t^2\right)$ term. Applying this idea to the time evolution of an arbitrary function $u(t)$ by a time step τ we get:

$$\begin{aligned} u(t+\tau) &= \exp\left(\left(A+B\right)\tau\right)u(t) \\ &\approx \exp\left(\frac{1}{2}B\tau\right)\exp\left(A\tau\right)\exp\left(\frac{1}{2}B\tau\right)u(t) \end{aligned}$$

which is the so called symmetric Lie-Trotter splitting. Going back to the time-dependent Schrödinger equation and its solution we obtain:

$$\begin{aligned}\psi(t+\tau) &= \exp\left(\left(-\frac{i}{\varepsilon^2}T - \frac{i}{\varepsilon^2}V\right)\tau\right)\psi(t) \\ &\approx \exp\left(\frac{1}{2}\left(-\frac{i}{\varepsilon^2}V\right)\tau\right)\exp\left(\left(-\frac{i}{\varepsilon^2}T\right)\tau\right)\exp\left(\frac{1}{2}\left(-\frac{i}{\varepsilon^2}V\right)\tau\right)\psi(t)\end{aligned}$$

and are left with the much easier problem of computing the three simpler applications above. We work in position space and the wavefunction is given as $\psi(\underline{x}, t)$. The potential $V(\underline{x})$ is given in position space too. The computation of the following exponential is therefore trivial:

$$\exp\left(-\frac{i}{2\varepsilon^2}V\tau\right) = \exp\left(-\frac{i}{2\varepsilon^2}V(\underline{x})\tau\right). \tag{2.2}$$

In the non-adiabatic case where the potential is matrix-valued this becomes a matrix exponential. But as $V \in \mathbb{R}^{N\times N}$ and N really small this is not a big issue. Compare to [1] for the details on how to compute this efficiently.
The other exponential involving the kinetic operator T is more of a problem since it includes a differential operator:

$$\begin{aligned}\exp\left(-\frac{i}{\varepsilon^2}T\tau\right) &= \exp\left(-\frac{i}{\varepsilon^2}\left(-\frac{\varepsilon^4}{2m}\Delta\right)\tau\right) \\ &= \exp\left(\frac{i\varepsilon^2}{2m}\Delta\tau\right) \\ &= \exp\left(\frac{i\varepsilon^2}{2}\Delta\tau\right)\end{aligned} \tag{2.3}$$

where we set the mass $m = 1$. The nasty point here is the Laplace operator inside the exponential. Luckily going to momentum space by a simple Fourier transform solves all our problems.

2.2 Fourier transformations

Since there are several closely related Fourier transforms we first write down what we use as notation.

Definition 1 (Fourier transformation in one dimension)**.** *The Fourier transformation for ordinary and angular frequency are given by:*

$$\begin{aligned}\mathcal{F}_x : f(x) \to \hat{f}(k) \qquad & \mathcal{F}_x := \int_{-\infty}^{\infty} f(x)\exp(-2\pi ikx)\mathrm{d}x \\ \mathcal{F}_x : f(x) \to \hat{f}(\omega) \qquad & \mathcal{F}_x := \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)\exp(-i\omega x)\mathrm{d}x\end{aligned}$$

and the respective inverse transformations are:

$$\begin{aligned}{\mathcal{F}_k}^{-1} : \hat{f}(k) \to f(x) \qquad & {\mathcal{F}_k}^{-1} := \int_{-\infty}^{\infty} \hat{f}(k)\exp(2\pi ikx)\mathrm{d}k \\ {\mathcal{F}_\omega}^{-1} : \hat{f}(\omega) \to f(x) \qquad & {\mathcal{F}_\omega}^{-1} := \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \hat{f}(\omega)\exp(i\omega x)\mathrm{d}\omega\end{aligned}$$

with $\omega = 2\pi k$.

In higher dimensions the transformations are of course similar. It is advantageous to use vector notation for their definition.

Definition 2 (Fourier transformation in D dimensions)**.** *Define the vectors* $\underline{x} = (x_0, \ldots, x_{D-1})$*,* $\underline{k} = (k_0, \ldots, k_{D-1})$ *and* $\underline{\omega} = (\omega_0, \ldots, \omega_{D-1})$*. Then the Fourier transformation for angular frequency is given by:*

$$\mathcal{F}_{\underline{x}} : f(\underline{x}) \to \hat{f}(\underline{\omega}) \qquad \mathcal{F}_{\underline{x}} := \frac{1}{(2\pi)^{\frac{D}{2}}} \int_{\mathbb{R}^D} f(\underline{x}) \exp(-i\underline{\omega} \cdot \underline{x}) \mathrm{d}\underline{x}$$

and the respective inverse transformation is:

$$\mathcal{F}_{\underline{\omega}}{}^{-1} : \hat{f}(\underline{\omega}) \to f(\underline{x}) \qquad \mathcal{F}_{\underline{\omega}} := \frac{1}{(2\pi)^{\frac{D}{2}}} \int_{\mathbb{R}^D} f(\underline{\omega}) \exp(i\underline{\omega} \cdot \underline{x}) \mathrm{d}\underline{\omega}\,.$$

Now we can use the following well-known theorem to simplify the exponential of T. The differential operator transforms under the Fourier transformation as follows:

$$\frac{\mathrm{d}^n}{\mathrm{d}x^n} f(x) \to (i\omega)^n \hat{f}(\omega) = (2\pi i k)^n \hat{f}(k)\,.$$

First we do the computation in one space dimension because it is easier to get the idea there. As known from the introduction the operator T is of the form:

$$T = -\frac{\varepsilon^4}{2} \frac{\partial^2}{\partial x^2} \tag{2.4}$$

and we compute now its Fourier transform as:

$$\mathcal{F}_x(T) = \mathcal{F}_x\left(-\frac{\varepsilon^4}{2}\frac{\partial^2}{\partial x^2}\right) = -\frac{\varepsilon^4}{2}\mathcal{F}_x\left(\frac{\partial^2}{\partial x^2}\right) = -\frac{\varepsilon^4}{2}(i\omega)^2 = \frac{\varepsilon^4}{2}\omega^2$$

by using the fundamental properties of the Fourier transform. Plugging this into the exponential where we compute $\exp\left(-\frac{i}{\varepsilon^2}T\tau\right)$ we get:

$$\exp\left(-\frac{i\varepsilon^2}{2}\omega^2\tau\right) \tag{2.5}$$

which is easy to evaluate. Doing the same computations in D dimensions is much more grinding. First we get for the operator T:

$$T = -\frac{\varepsilon^4}{2}\Delta = -\frac{\varepsilon^4}{2}\left(\frac{\partial^2}{\partial x_0^2} + \frac{\partial^2}{\partial x_1^2} + \cdots + \frac{\partial^2}{\partial x_{D-1}^2}\right) = -\frac{\varepsilon^4}{2}\sum_{d=0}^{D-1}\frac{\partial^2}{\partial x_d^2} \tag{2.6}$$

by replacing the Laplace through its definition in the current coordinate system. The principle now is to recursively compute Fourier transforms for a single variable x_d only. For the first iteration by $\mathcal{F}_{x_0}$ we get:

$$
\begin{aligned}
\mathcal{F}_{x_0}(T) &= \mathcal{F}_{x_0}\left(-\frac{\varepsilon^4}{2}\left(\frac{\partial^2}{\partial x_0^2}+\sum_{d=1}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)\right)\\
&= -\frac{\varepsilon^4}{2}\mathcal{F}_{x_0}\left(\frac{\partial^2}{\partial x_0^2}+\sum_{d=1}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)\\
&= -\frac{\varepsilon^4}{2}\left(\mathcal{F}_{x_0}\left(\frac{\partial^2}{\partial x_0^2}\right)+\sum_{d=1}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)\\
&= -\frac{\varepsilon^4}{2}\left((i\omega_0)^2+\sum_{d=1}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)
\end{aligned}
$$

where the terms in the sum are constant with respect to x_0. Performing the next iteration and computing $\mathcal{F}_{x_1}$ we get:

$$
\begin{aligned}
&\mathcal{F}_{x_1}\left(-\frac{\varepsilon^4}{2}\left((i\omega_0)^2+\sum_{d=1}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)\right)\\
=&-\frac{\varepsilon^4}{2}\mathcal{F}_{x_1}\left(\left((i\omega_0)^2+\frac{\partial^2}{\partial x_1^2}+\sum_{d=2}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)\right)\\
=&-\frac{\varepsilon^4}{2}\left((i\omega_0)^2+\mathcal{F}_{x_1}\left(\frac{\partial^2}{\partial x_1^2}\right)+\sum_{d=2}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)\\
=&-\frac{\varepsilon^4}{2}\left((i\omega_0)^2+(i\omega_1)^2+\sum_{d=2}^{D-1}\frac{\partial^2}{\partial x_d^2}\right)
\end{aligned}
$$

and we can see the pattern. Finally following this path we will get the result:

$$
\begin{aligned}
\mathcal{F}_{\underline{x}}(T) &= \mathcal{F}_{x_{D-1}}\cdots\mathcal{F}_{x_0}\left(-\frac{\varepsilon^4}{2}\Delta\right)\\
&= -\frac{\varepsilon^4}{2}\sum_{d=0}^{D-1}(i\omega_d)^2\\
&= \frac{\varepsilon^4}{2}\underline{\omega}\cdot\underline{\omega}\,.
\end{aligned}
$$

For the exponential $\exp\left(-\frac{i}{\varepsilon^2}T\tau\right)$ we get the expression:

$$
\exp\left(-\frac{i\varepsilon^2}{2}\underline{\omega}\cdot\underline{\omega}\tau\right) \tag{2.7}
$$

which simplifies to (2.5) above if we set $D=1$. At this point we have all parts together to compute the time evaluation of a wavefunction $\psi(\underline{x},t)$.

2.3 Time propagation of wavefunctions

Using the operator splitting shown earlier we can write the approximative time propagation of a wavefunction ψ by small time steps τ as:

$$\psi(\underline{x}, t+\tau) = \exp\left(-\frac{i}{2\varepsilon^2}\tau V\right)\exp\left(-\frac{i}{\varepsilon^2}\tau T\right)\exp\left(-\frac{i}{2\varepsilon^2}\tau V\right)\psi(\underline{x}, t)$$

or better with three separate steps as:

$$\begin{aligned}
\psi(\underline{x}, t+\tau)' &= \exp\left(-\frac{i}{2\varepsilon^2}\tau\, V(\underline{x})\right)\psi(\underline{x}, t) \\
\psi(\underline{x}, t+\tau)'' &= \exp\left(-\frac{i}{\varepsilon^2}\tau\, T(\underline{x})\right)\psi(\underline{x}, t+\tau)' \\
\psi(\underline{x}, t+\tau) &= \exp\left(-\frac{i}{2\varepsilon^2}\tau\, V(\underline{x})\right)\psi(\underline{x}, t+\tau)''\,.
\end{aligned} \tag{2.8}$$

The first and third step are easy, they only involve the potential operator. In contrast the second step is problematic because it includes a differential operator. This is now the point where we use the Fourier transformation to get rid of the differential operator. We compute the angular-frequency Fourier transform in the spatial coordinates only $\hat{\psi}(\underline{\omega}, t) = \mathcal{F}_{\underline{x}}\left(\psi(\underline{x}, t)\right)$ and then the second step becomes:

$$\psi(\underline{x}, t+\tau)'' = {\mathcal{F}_{\underline{\omega}}}^{-1}\left[\exp\left(-\frac{i}{\varepsilon^2}\tau\,\hat{T}(\underline{\omega})\right)\mathcal{F}_{\underline{x}}\left[\psi(\underline{x}, t+\tau)'\right]\right]\,. \tag{2.9}$$

For the sake of completeness we can write out the whole procedure as:

$$\begin{aligned}
\psi(\underline{x}, t+\tau)' &= \exp\left(-\frac{i}{2\varepsilon^2}\tau\, V(\underline{x})\right)\psi(\underline{x}, t) \\
\psi(\underline{x}, t+\tau)'' &= {\mathcal{F}_{\underline{\omega}}}^{-1}\left[\exp\left(-\frac{i}{\varepsilon^2}\tau\,\hat{T}(\underline{\omega})\right)\mathcal{F}_{\underline{x}}\left[\psi(\underline{x}, t+\tau)'\right]\right] \\
\psi(\underline{x}, t+\tau) &= \exp\left(-\frac{i}{2\varepsilon^2}\tau\, V(\underline{x})\right)\psi(\underline{x}, t+\tau)''\,.
\end{aligned} \tag{2.10}$$

This is the full formula for the time propagation of wavefunctions with discrete timesteps τ but still continuous space $\underline{x}$.
To get shorter notation we define the following propagation operators:

$$V_e(\underline{x}) := \exp\left(-\frac{i}{2\varepsilon^2}\tau V(\underline{x})\right) \tag{2.11}$$

for the potential part in steps 1 and 3 and:

$$T_e(\underline{\omega}) := \exp\left(-\frac{i\varepsilon^2}{2}\tau\underline{\omega}\cdot\underline{\omega}\right) \tag{2.12}$$

for the kinetic part in step 2. In case of vector-valued wavefunctions we get matrix-valued operators here. If we write them out we obtain:

$$V_e(\underline{x}) := \exp\left(-\frac{i}{2\varepsilon^2}\tau\begin{pmatrix} v_{0,0}\left(\underline{x}\right) & \cdots & v_{0,N-1}\left(\underline{x}\right) \\ \vdots & & \vdots \\ v_{N-1,0}\left(\underline{x}\right) & \cdots & v_{N-1,N-1}\left(\underline{x}\right)\end{pmatrix}\right) \tag{2.13}$$

which is another $N \times N$ matrix that is in general non-diagonal. For the other operator we get a similar matrix:

$$T_e(\underline{\omega}) := \exp\left(-\frac{i\varepsilon^2}{2}\tau\underline{\omega}\cdot\underline{\omega}\right)\mathbb{1}_{N\times N} = \exp\left(-\frac{i\varepsilon^2}{2}\tau\begin{pmatrix}\underline{\omega}\cdot\underline{\omega} & & \\ & \ddots & \\ & & \underline{\omega}\cdot\underline{\omega}\end{pmatrix}\right) \tag{2.14}$$

but this time the matrix is diagonal and therefore the exponential of it is diagonal too. If we write the operator in the following form:

$$T_e(\underline{\omega}) := \begin{pmatrix}\exp\left(-\frac{i\varepsilon^2}{2}\tau\underline{\omega}\cdot\underline{\omega}\right) & & \\ & \ddots & \\ & & \exp\left(-\frac{i\varepsilon^2}{2}\tau\underline{\omega}\cdot\underline{\omega}\right)\end{pmatrix} \tag{2.15}$$

this even allows for different vectors $\underline{\omega}^j$ for each component $\hat{\psi}_j\left(\underline{\omega}^j, t\right)$ of ψ. The advantage is that we can take different position space grids Γ_j for each component.

2.4 Discretised position and momentum space

The final goal is to perform numerical simulations on a computer. Therefore we need not only to discretise time by taking small steps forward, but to discretise space as well. This is done by introducing a fine grid of nodes on the whole computational domain. We show the process in one space dimension first. The domain is just the interval $[a, b[\in \mathbb{R}$ with a size of $\Delta := |b - a|$. We assume x to be periodic with respect to this interval. Now we place N nodes on $[a, b[$ where we take care not to put a node at b. The grid mesh size δ is then defined by $\delta := \frac{\Delta}{N} = \frac{|b-a|}{N}$. Hence single grid nodes $\gamma_j \in \mathbb{R}$ can be constructed as:

$$\gamma_j := a + j\frac{\Delta}{N} = a + j\delta \quad j \in 0, \ldots, N-1 \tag{2.16}$$

and the whole position space grid Γ is:

$$\Gamma := \{\gamma_j\}_{j=0}^{N-1}. \tag{2.17}$$

In D dimensions the computational domain is the hypercubic tensor product $[a_0, b_0[\times \cdots \times [a_{D-1}, b_{D-1}[$. Along each axis x_d we place N_d grid nodes in the interval $[a_d, b_d[$. We can construct the full grid by taking the following tensor product over one-dimensional grids:

$$\Gamma := \bigotimes_{d=0}^{D-1} \Gamma_d\,. \tag{2.18}$$

Notice that $\prod_{d=0}^{D-1} \Delta_d$ is the volume of the computational domain. Further $\prod_{d=0}^{D-1} N_d$ is the total number of grid nodes $|\Gamma|$ of Γ since we are dealing with tensor product grids. Finally the value $\prod_{d=0}^{D-1} \frac{\Delta_d}{N_d}$ is kind of an inverse grid node density.

If we shift the arbitrary interval $[a, b[$ to $[0, \Delta[$ then the position space grid nodes are given by $\Gamma = \{\frac{\Delta j}{N}\}_{j=0}^{N-1}$. Hence the momentum space grid Ω consists of the following nodes:

$$\Omega := \{\omega_j\}_{j=0}^{N-1} = \left\{ \frac{1}{\Delta} \left(j - \frac{N}{2} \right) \right\}_{j=0}^{N-1} \tag{2.19}$$

with fundamental frequency $\frac{1}{\Delta}$. The Fourier space grid for a D dimensional space is obtained by a tensor product of D one-dimensional grids:

$$\Omega := \bigotimes_{d=0}^{D-1} \Omega_d \,. \tag{2.20}$$

A grid node $\underline{\omega} \in \Omega$ is then of the form $\underline{\omega} = (\omega_0, \ldots, \omega_{D-1})$ with ω_d the frequency index along the axis d.

2.5 Discrete Fourier transformations

Of course we have to replace the Fourier transformations in the propagator (2.10) by a discrete analogue. We use the following definitions for the one-dimensional discrete Fourier transformations.

Definition 3 (Discrete Fourier transformation in one dimension)**.** *Let $\underline{x}$ be a vector having N entries $\underline{x}_n$. Then the discrete Fourier transformation in unitary scaling is given by:*

$$\underline{\hat{x}}_k := \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \underline{x}_n \exp\left(-2\pi i \, \frac{kn}{N} \right)$$

while the inverse is:

$$\underline{x}_n := \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \underline{\hat{x}}_k \exp\left(2\pi i \, \frac{kn}{N} \right) \,.$$

For our implementation we use the fast Fourier transform algorithm to compute these vectors efficiently.
Next we work out the discrete Fourier transform in the multi-dimensional case. Start with a D dimensional data tensor $x_{n_0,\ldots,n_{D-1}}$. Then we do the discrete Fourier transform along the first axis only:

$$x_{k_0,n_1,\ldots,n_{D-1}} = \frac{1}{\sqrt{N_0}} \sum_{n_0=0}^{N_0-1} x_{n_0,n_1,\ldots,n_{D-1}} \exp\left(-2\pi i \, \frac{k_0 n_0}{N_0} \right) \,.$$

In the next step we do the transform along the second axis only:

$$
\begin{aligned}
x_{k_0,k_1,n_2\ldots,n_{D-1}} &= \frac{1}{\sqrt{N_0}}\frac{1}{\sqrt{N_1}}\sum_{n_1=0}^{N_1-1} x_{k_0,n_1,\ldots,n_{D-1}} \exp\left(-2\pi i\,\frac{k_1 n_1}{N_1}\right) \\
&= \frac{1}{\sqrt{N_0 N_1}}\sum_{n_1=0}^{N_1-1}\sum_{n_0=0}^{N_0-1} x_{n_0,n_1,\ldots,n_{D-1}} \exp\left(-2\pi i\,\frac{k_0 n_0}{N_0}\right)\exp\left(-2\pi i\,\frac{k_1 n_1}{N_1}\right) \\
&= \frac{1}{\sqrt{N_0 N_1}}\sum_{n_1=0}^{N_1-1}\sum_{n_0=0}^{N_0-1} x_{n_0,n_1,\ldots,n_{D-1}} \exp\left(-2\pi i\left(\frac{k_0 n_0}{N_0}+\frac{k_1 n_1}{N_1}\right)\right).
\end{aligned}
$$

Going on in the same way evaluating the transformations for all remaining axes we get:

$$
x_{k_0,\ldots,k_{D-1}} = \frac{1}{\prod_{d=0}^{D-1}\sqrt{N_d}}\sum_{n_{D-1}=0}^{N_{D-1}-1}\cdots\sum_{n_0=0}^{N_0-1} x_{n_0,n_1,\ldots,n_{D-1}} \exp\left(-2\pi i\sum_{d=0}^{D-1}\frac{k_d n_d}{N_d}\right).
$$

To get a more compact notation we may introduce the following vectors $\underline{n} := (n_0,\ldots,n_{D-1})$, $\underline{k} := (k_0,\ldots,k_{D-1})$ and $\underline{N} := (N_0,\ldots,N_{D-1})$. Then we have:

$$
\hat{x}_{\underline{k}} := \frac{1}{\prod_{d=0}^{D-1}\sqrt{\underline{N}_d}}\sum_{\underline{n}=\underline{0}}^{\underline{N}-\underline{1}} x_{\underline{n}} \exp\left(-2\pi i\,\frac{\underline{k}\cdot\underline{n}}{\underline{N}}\right)
$$

and for the inverse:

$$
x_{\underline{n}} := \frac{1}{\prod_{d=0}^{D-1}\sqrt{\underline{N}_d}}\sum_{\underline{k}=\underline{0}}^{\underline{N}-\underline{1}} \hat{x}_{\underline{k}} \exp\left(2\pi i\,\frac{\underline{k}\cdot\underline{n}}{\underline{N}}\right)
$$

where all operations on vectors are understood as element-wise. If we only use the angular frequency $\underline{\omega} = 2\pi\underline{k}$ we arrive at our final transformation formulae.

Definition 4 (Discrete Fourier transformation in D dimensions)**.** *Let x be a D dimensional data tensor $x_{n_0,\ldots,n_{D-1}}$ with N_d entries along the direction d. Then the discrete Fourier transformation and its inverse both in unitary scaling are given by:*

$$
\begin{aligned}
\hat{x}_{\underline{\omega}} &:= \frac{1}{\prod_{d=0}^{D-1}\sqrt{\underline{N}_d}}\sum_{\underline{n}=\underline{0}}^{\underline{N}-\underline{1}} x_{\underline{n}} \exp\left(-i\frac{\underline{\omega}\cdot\underline{n}}{\underline{N}}\right) \\
x_{\underline{n}} &:= \frac{1}{\prod_{d=0}^{D-1}\sqrt{\underline{N}_d}}\sum_{\underline{\omega}=\underline{0}}^{\underline{N}-\underline{1}} \hat{x}_{\underline{\omega}} \exp\left(i\frac{\underline{\omega}\cdot\underline{n}}{\underline{N}}\right).
\end{aligned}
$$

2.6 Basis transformations

Using the formula (1.4) from the introduction we can write the basis transformation of a wavefunction $|\psi\rangle$ as:

$$
\begin{aligned}
|\psi_{\text{canonical}}\rangle &= \mathbf{M}(\underline{x})\,|\psi_{\text{eigen}}\rangle \\
|\psi_{\text{eigen}}\rangle &= \mathbf{M}^{-1}(\underline{x})\,|\psi_{\text{canonical}}\rangle = \mathbf{M}^{\mathrm{H}}(\underline{x})\,|\psi_{\text{canonical}}\rangle
\end{aligned}
\tag{2.21}
$$

where we used that the transformation matrix $\mathbf{M}$ is unitary.
The time propagation of wavefunctions $|\psi\rangle$ has to happen in the canonical basis because we know the operators only there. On the other hand we are interested in the behaviour of $|\psi\rangle$ on the different energy levels. Therefore the computation of observables must be done in the eigenbasis. Only the overall norm of wavefunctions as well as the total energy are basis independent.
Assume the wavefunction $|\psi\rangle$ is vector-valued with N components ψ_i. The explicit transformation to the eigenbasis is then:

$$\psi^e = \begin{pmatrix} \psi_0^e \\ \vdots \\ \psi_{N-1}^e \end{pmatrix} = \mathbf{M}^{\mathrm{H}} \begin{pmatrix} \psi_0^c \\ \vdots \\ \psi_{N-1}^c \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{N-1} \mathbf{M}_{0,i}^{\mathrm{H}} \, \psi_0^e \\ \vdots \\ \sum_{i=0}^{N-1} \mathbf{M}_{N-1,i}^{\mathrm{H}} \, \psi_{N-1}^e \end{pmatrix} .$$

2.7 Observables

2.7.1 Norm of a scalar wavefunction

Given the scalar wavefunction $|\psi\rangle$ we want to compute the norm $\langle\psi \,|\, \psi\rangle$. The norm is interesting because it gives us the probability density function according to the Kopenhagen interpretation. In one space dimension the norm is defined as:

$$\langle\psi \,|\, \psi\rangle = \int_{\mathbb{R}} \overline{\psi(x)}\psi(x)\mathrm{d}x \, .$$

In discretised space we evaluate ψ on the nodes of a grid Γ and get:

$$\begin{aligned} \langle\psi(\Gamma) \,|\, \psi(\Gamma)\rangle &= \|\psi(\Gamma)\|_2^2 \\ &= \sum_{\gamma\in\Gamma} \overline{\psi(\gamma)}\psi(\gamma) \\ &= \sum_{i=1}^{N} \overline{\psi(\gamma_i)}\psi(\gamma_i) \end{aligned}$$

where we employ Parseval's identity and go to momentum space:

$$\begin{aligned} &= \frac{T}{N^2} \sum_{i=1}^{N} \overline{\hat{\psi}(\omega_i)}\hat{\psi}(\omega_i) \\ &= \frac{T}{N^2} \|\hat{\psi}(\Omega)\|_2^2 \end{aligned}$$

where $N = |\Gamma|$ the number of grid nodes and $T = |b-a|$ the extension of our grid (the length of the interval) [1]. The final result for the norm of a wavefunction in one space dimension is therefore:

$$\|\psi(\Gamma)\|_2 = \frac{\sqrt{T}}{N} \|\hat{\psi}(\Omega)\|_2 \, .$$

[1] A possible derivation of the prefactors goes as follows:

In an arbitrary number D of space dimensions the grid Γ has an extension T_d in each direction d and is subdivided into N_d nodes along this axis. Define the vector $\underline{N} := (N_0, \ldots, N_{D-1})$ used for D-dimensional summation. Following the same route we get:

$$\begin{aligned}\int_{T_0} \cdots \int_{T_{D-1}} \overline{\psi\left(\underline{x}\right)}\psi\left(\underline{x}\right) \mathrm{d}\underline{x} &\approx \prod_{d=0}^{D-1} \frac{T_d}{N_d} \sum_{\underline{j}=0}^{\underline{N}-1} \overline{\psi\left(\underline{\gamma}_{\underline{j}}\right)}\psi\left(\underline{\gamma}_{\underline{j}}\right) \\ &= \prod_{d=0}^{D-1} \frac{T_d}{N_d^2} \sum_{\underline{j}=0}^{\underline{N}-1} \overline{\hat{\psi}\left(\underline{\omega}_{\underline{j}}\right)}\hat{\psi}\left(\underline{\omega}_{\underline{j}}\right) .\end{aligned}$$

At the end of the day we find that:

$$\|\psi(\Gamma)\|_2 = \prod_{d=0}^{D-1} \frac{\sqrt{T_d}}{N_d} \|\hat{\psi}(\Omega)\|_2 \tag{2.22}$$

holds for the norm of a D-dimensional scalar wavefunction $|\psi\rangle$ evaluated on an appropriate grid $\Gamma \subset \mathbb{R}^D$.

2.7.2 Norm of a vectorial wavefunction

We do not stop at the scalar case but want to find norms of vectorial wavefunctions $|\psi\rangle$ with N components. This computation is done in the eigenbasis and therefore we need to transform $|\psi\rangle$. Computing the total norm of $|\psi^e\rangle$ we find that:

$$\begin{aligned}\langle \psi^e \,|\, \psi^e \rangle &= \langle \psi_0^e \,|\, \psi_0^e \rangle + \cdots + \left\langle \psi_{N-1}^e \,\middle|\, \psi_{N-1}^e \right\rangle \\ &= \sum_{i=0}^{N-1} \langle \psi_i^e \,|\, \psi_i^e \rangle\end{aligned}$$

where $\langle \psi_i^e \,|\, \psi_i^e \rangle$ is the norm of the part ψ_i of the wavefunction ψ that resides on the energy level $\lambda_i(\underline{x})$. For each of these brakets we can now apply the formula (2.22) from section 2.7.1. This was easy because there is no operator in the middle of the braket which could mix up the components ψ_i.

2.7.3 Energy of a scalar wavefunction

The energy of a scalar wavefunction $|\psi\rangle$ is given by the expectation value:

$$E = \langle \psi \,|\, H \,|\, \psi \rangle \tag{2.23}$$

$$\begin{aligned}\|\psi\|_{L^2(T)}^2 &= \int_T \overline{\psi(x)}\psi(x)\mathrm{d}x \approx \frac{T}{N} \sum_{i=1}^{N} \overline{\psi(\gamma_i)}\psi(\gamma_i) \\ &= \frac{T}{N} \sum_{i=1}^{N} \frac{1}{\sqrt{N}} \overline{\hat{\psi}(\omega_i)} \frac{1}{\sqrt{N}} \psi(\omega_i) = \frac{T}{N^2} \sum_{i=1}^{N} \overline{\hat{\psi}(\omega_i)}\psi(\omega_i) \\ &= \frac{T}{N^2} \|\hat{\psi}\|_2^2\end{aligned}$$

where we have used Riemann sum approximation and the unitary discrete Fourier transform.

of the Hamiltonian operator $H = T + V$. We can split this into an expression for the kinetic and one for the potential energy:

$$E = E_{kin} + E_{pot} = \langle \psi \,|\, T + V \,|\, \psi \rangle = \langle \psi \,|\, T \,|\, \psi \rangle + \langle \psi \,|\, V \,|\, \psi \rangle \ . \tag{2.24}$$

In the following short sections we will compute both parts starting with the potential energy.

Potential energy

Write down the following expression for a fixed time t:

$$\begin{aligned} E_{pot} &= \langle \psi \,|\, V \,|\, \psi \rangle = \langle \psi(\underline{x}) \,|\, V(\underline{x}) \,|\, \psi(\underline{x}) \rangle \\ &= \int \cdots \int_{\mathbb{R}^D} \overline{\psi(\underline{x})} V(\underline{x}) \psi(\underline{x}) \mathrm{d}\underline{x} \end{aligned}$$

then transforming to Fourier space:

$$= \int \cdots \int_{\mathbb{R}^D} \mathcal{F}\left(\overline{\psi(\underline{x})}\right) \mathcal{F}\left(V(\underline{x})\psi(\underline{x})\right) \mathrm{d}\underline{\omega} \, .$$

Computing separately the subexpression $\varphi(\underline{x}) := V(\underline{x})\psi(\underline{x})$ we obtain:

$$E_{pot} = \int \cdots \int_{\mathbb{R}^D} \overline{\hat{\psi}(\underline{\omega})} \hat{\varphi}(\underline{\omega}) \mathrm{d}\underline{\omega} \, .$$

But this is still assuming a continuous space representation. Switching to the discretised space by introducing the grid Γ we get:

$$\int \cdots \int_{\mathbb{R}^D} \overline{\psi(\underline{x})} \varphi(\underline{x}) \mathrm{d}\underline{x} \approx \prod_{d=0}^{D-1} \frac{T_d}{N_d} \sum_{\underline{j}=0}^{\underline{N}-1} \overline{\psi\left(\underline{\gamma}_{\underline{j}}\right)} \varphi\left(\underline{\gamma}_{\underline{j}}\right) \ .$$

Applying the unitary discrete Fourier transform next gives the final result:

$$E_{pot} = \prod_{d=0}^{D-1} \frac{T_d}{N_d^2} \sum_{\underline{j}=0}^{\underline{N}-1} \overline{\hat{\psi}\left(\underline{\omega}_{\underline{j}}\right)} \hat{\varphi}\left(\underline{\omega}_{\underline{j}}\right) \ . \tag{2.25}$$

Kinetic energy

The kinetic energy is given from theory by:

$$E_{kin} = \langle \psi \,|\, T \,|\, \psi \rangle = \left\langle \psi(\underline{x}) \,\middle|\, -\frac{1}{2}\varepsilon^4 \Delta \,\middle|\, \psi(\underline{x}) \right\rangle \ .$$

The Laplace operator in this expression is bad because we have difficulties to compute its action. However we can circumvent these issues by going to Fourier space. We already know that $\mathcal{F}\left(-\frac{\varepsilon^4}{2}\Delta\right) = \frac{\varepsilon^4}{2}\underline{\omega} \cdot \underline{\omega}$ and for this reason we can get:

$$\left\langle \psi(\underline{x}) \,\middle|\, -\frac{\varepsilon^4}{2}\Delta \,\middle|\, \psi(\underline{x}) \right\rangle = \left\langle \hat{\psi}(\underline{\omega}) \,\middle|\, \frac{\varepsilon^4}{2}\underline{\omega}\cdot\underline{\omega} \,\middle|\, \hat{\psi}(\underline{\omega}) \right\rangle$$
$$= \frac{\varepsilon^4}{2}\int\cdots\int_{\mathbb{R}^D} \overline{\hat{\psi}(\underline{\omega})}\,\underline{\omega}\cdot\underline{\omega}\,\hat{\psi}(\underline{\omega})\mathrm{d}\underline{\omega}\,.$$

In discretised space we avoid computing the Fourier transform at first. Instead we take this formula from the continuous space representation:

$$\left\langle \mathcal{F}\left(\psi(\underline{\gamma})\right) \,\middle|\, \frac{\varepsilon^4}{2}\underline{\omega}\cdot\underline{\omega} \,\middle|\, \mathcal{F}\left(\psi(\underline{\gamma})\right) \right\rangle$$

and discretise the integral first giving approximately:

$$\frac{\varepsilon^4}{2}\prod_{d=0}^{D-1}\frac{T_d}{N_d}\sum_{\underline{j}=0}^{\underline{N}-1}\mathcal{F}\left(\overline{\psi\left(\underline{\gamma}_{\underline{j}}\right)}\right)\underline{\omega}_{\underline{j}}\cdot\underline{\omega}_{\underline{j}}\mathcal{F}\left(\psi\left(\underline{\gamma}_{\underline{j}}\right)\right)\,.$$

Now we can easily apply the discrete Fourier transformation and get our final result for the kinetic energy:

$$E_{kin} = \frac{\varepsilon^4}{2}\prod_{d=0}^{D-1}\frac{T_d}{N_d^2}\sum_{\underline{j}=0}^{\underline{N}-1}\overline{\hat{\psi}\left(\underline{\omega}_{\underline{j}}\right)}\left(\underline{\omega}_{\underline{j}}\cdot\underline{\omega}_{\underline{j}}\right)\hat{\psi}\left(\underline{\omega}_{\underline{j}}\right)\,. \tag{2.26}$$

This is all we need to compute energies of scalar wavefunctions $|\psi\rangle$. In an efficient implementation all the multi-sums over the grid nodes are fully vectorised and drop out. Also we can precompute the inner product of the $\underline{\omega}$ vectors.

2.7.4 Energy of a vectorial wavefunction

Now we consider vectorial wavefunctions having N components and try to compute their energies. Again we want to find the energies given in the eigenbasis. For the kinetic energy of $|\psi^e\rangle$ we have to compute:

$$E_{kin} = \langle \psi^e \,|\, \mathbf{T}^e \,|\, \psi^e \rangle$$

where $\mathbf{T}^e$ is the kinetic operator in the eigenbasis. The problem is that we do not know what $\mathbf{T}^e$ looks like. Therefore we take each single component ψ_i^e, put it into a wavefunction vector like:

$$\psi'^e := \begin{pmatrix} 0 \\ \vdots \\ \psi_i^e \\ \vdots \\ 0 \end{pmatrix}$$

and transform this object back to the canonical basis to avoid picture change errors. We obtain in general:

$$\mathbf{M}\psi'^e = \psi'^c = \begin{pmatrix} \psi_0'^c \\ \vdots \\ \psi_{N-1}'^c \end{pmatrix}\,.$$

From this we can easily compute:

$$E^i_{kin} = \langle \psi'^c \,|\, \mathbf{T} \,|\, \psi'^c \rangle$$

where $\mathbf{T}$ is block-diagonal of size $N \times N$:

$$\mathbf{T} = \begin{pmatrix} T_i & & \\ & \ddots & \\ & & T_i \end{pmatrix}$$

with all T_i being identical. Therefore we get for the kinetic energy of ψ^e_i:

$$E^i_{kin} = \sum_{i=0}^{N-1} \langle \psi'^c_i \,|\, T_i \,|\, \psi'^c_i \rangle \ . \tag{2.27}$$

For each braket we apply (2.26) and the techniques of section 2.7.3. The total kinetic energy of $|\psi^e\rangle$ is then of course:

$$E_{kin} = \sum_{i=0}^{N-1} E^i_{kin} \, . \tag{2.28}$$

Computing the potential energy of $|\psi^e\rangle$ is easier. We can do this in the eigenbasis where we need to compute:

$$E_{pot} = \langle \psi^e \,|\, \mathbf{\Lambda} \,|\, \psi^e \rangle \ .$$

Because $\mathbf{\Lambda}$ is a diagonal matrix we can decompose this into:

$$E_{pot} = \sum_{i=0}^{N-1} \langle \psi^e_i \,|\, \lambda_i \,|\, \psi^e_i \rangle$$

and the potential energy of ψ^e_i is:

$$E^i_{pot} = \langle \psi^e_i \,|\, \lambda_i \,|\, \psi^e_i \rangle \ . \tag{2.29}$$

The brakets here can in turn be computed by the formula (2.25) from 2.7.3. To conclude this chapter we show that the total energy is basis independent:

$$\begin{aligned} \langle \psi^e \,|\, \mathbf{\Lambda} \,|\, \psi^e \rangle &= \left\langle \psi^e \,\middle|\, \mathbf{M}^{-1}\mathbf{M}\mathbf{\Lambda}\mathbf{M}^{-1}\mathbf{M} \,\middle|\, \psi^e \right\rangle = \left\langle \psi^e\mathbf{M} \,\middle|\, \mathbf{M}\mathbf{\Lambda}\mathbf{M}^{-1} \,\middle|\, \mathbf{M}\psi^e \right\rangle \\ &= \left\langle \psi^c \,\middle|\, \mathbf{M}\mathbf{\Lambda}\mathbf{M}^{-1} \,\middle|\, \psi^c \right\rangle = \langle \psi^c \,|\, \mathbf{V} \,|\, \psi^c \rangle \ . \end{aligned}$$

Chapter 3

Semi-classical Wavepackets

3.1 The basis functions

We first show once more the one-dimensional semi-classical wavepackets. This short section will constitute a reference to compare with when we will head for the D-dimensional case. The full mathematical details can be found in [8].

3.1.1 The one-dimensional case

In the 1-dimensional case the semi-classical Hagedorn wavepacket is defined as:

$$\phi_0[\Pi](x) = \pi^{-\frac{1}{4}}(\varepsilon^2)^{-\frac{1}{4}}Q^{-\frac{1}{2}}\exp\left(iPQ^{-1}(x-q)^2/(2\varepsilon^2) + ip(x-q)/\varepsilon^2\right) \tag{3.1}$$

$$:= (\pi\varepsilon^2)^{-\frac{1}{4}}Q^{-\frac{1}{2}}\exp\left(\frac{i}{2\varepsilon^2}PQ^{-1}(x-q)^2 + \frac{i}{\varepsilon^2}p(x-q)\right) \tag{3.2}$$

where $x, q, p \in \mathbb{R}$ and $Q, P \in \mathbb{C}$ some parameters. We gather them into the set $\Pi := q, p, Q, P$ which we often call the *Hagedorn parameter*. For these parameters, there is a condition that must hold:

$$\overline{Q}P - \overline{P}Q = 2i \tag{3.3}$$

$$PQ - QP = 0\,. \tag{3.4}$$

The second equation is trivial in the one-dimensional case because complex numbers commute.
The equation (3.1) represents some kind of ground state ϕ_0. We can construct higher order states ϕ_k by the help of raising operators. The raising operator $\mathcal{R}$ and lowering operator $\mathcal{L}$ are defined as:

$$\mathcal{R} = \frac{i}{\sqrt{2\varepsilon^2}}\left(\overline{P}(x-q) - \overline{Q}(y-p)\right) \tag{3.5}$$

$$\mathcal{L} = -\frac{i}{\sqrt{2\varepsilon^2}}\left(P(x-q) - Q(y-p)\right) \tag{3.6}$$

where y denotes the momentum operator $y := -i\varepsilon^2 \frac{\partial}{\partial x}$. See [1] for the details of the one-dimensional case. Applied to the function ϕ_0 we get:

$$\phi_k = \frac{1}{\sqrt{k}} \mathcal{R}^k \phi_0 \tag{3.7}$$

by repeated application of $\mathcal{R}$. For a single application to the function ϕ_k we know that:

$$\phi_{k-1} = \frac{1}{\sqrt{k}} \mathcal{L} \phi_k \tag{3.8}$$

$$\phi_{k+1} = \frac{1}{\sqrt{k+1}} \mathcal{R} \phi_k \,. \tag{3.9}$$

In principle we can now write the expression for ϕ_k in closed-form:

$$\begin{aligned} \phi_k[\Pi](x) = 2^{-\frac{k}{2}} (k!)^{-\frac{1}{2}} (\pi\varepsilon^2)^{-\frac{1}{4}} Q^{-\frac{k+1}{2}} \overline{Q}^{\frac{k}{2}} H_k\left(\varepsilon^{-1}|Q|^{-1}(x-q)\right) \cdot \\ \exp\left(\frac{i}{2\varepsilon^2} P Q^{-1} (x-q)^2 + \frac{i}{\varepsilon^2} p(x-q)\right) \end{aligned}$$

where the Hermite polynomials H_k are defined as $H_k(z) := e^{z^2} \left(-\frac{\partial}{\partial z}\right)^k e^{-z^2}$. Although we have this closed-form expression for ϕ_k we will never use it due to numerical issues for example with the factorial there.
With the use of the ladder operators we can find a three-term recursion formula involving ϕ_{k+1} , ϕ_k and ϕ_{k-1} only:

$$\phi_{k+1}(x) = \sqrt{\frac{2}{\varepsilon^2}} \frac{1}{\sqrt{k+1}} Q^{-1}(x-q)\phi_k(x) - \sqrt{\frac{k}{k+1}} Q^{-1} \overline{Q} \phi_{k-1}(x) \,. \tag{3.10}$$

For this recursive computation we need two starting points. Obviously we use ϕ_0 which we can compute easily. For the second point we could use $\phi_1 = \mathcal{R}\phi_0 = \sqrt{\frac{2}{\varepsilon^2}} Q^{-1}(x-q)\phi_0(x)$ which we can find by hand. Alternatively we can use the fundamental property of the lowering operator $\mathcal{L}\phi_0 \equiv 0$ and find that $\phi_{-1} \equiv 0$. Although this appears to be a very technical argument it works fine.

3.1.2 The multi-dimensional case

In the D-dimensional case everything works analogously but also becomes more complicated. However we can check correctness of the results by taking $D = 1$. All formulae from the last section have to be contained as special cases.
First we note that $\underline{x} \in \mathbb{R}^D$ is a vector with D components denoted by $\underline{x} = (x_0, \ldots, x_{D-1})$. It follows that the position and momentum means are also vectors, in short $\underline{q}, \underline{p} \in \mathbb{R}^D$. The parameters $\mathbf{Q}, \mathbf{P} \in \mathbb{C}^{D \times D}$ are hence complex matrices of shape $D \times D$. For these matrices two relations analogue to (3.3) hold:

$$\mathbf{Q}^{\mathrm{H}} \mathbf{P} - \mathbf{P}^{\mathrm{H}} \mathbf{Q} = 2i\mathbb{1} \tag{3.11}$$

$$\mathbf{P}^{\mathrm{T}} \mathbf{Q} - \mathbf{Q}^{\mathrm{T}} \mathbf{P} = \mathbf{0} \,. \tag{3.12}$$

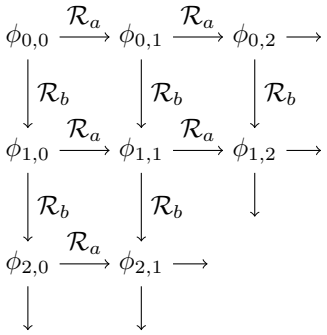


Figure 3.1: The grid of basis functions $\phi_{k,l}$ for the case $D = 2$.

The ground state $\phi_{\underline{0}}$ can now be defined as:

$$\phi_{\underline{0}}[\Pi](\underline{x}) := (\pi\varepsilon^2)^{-\frac{D}{4}} (\det \mathbf{Q})^{-\frac{1}{2}} \exp\left(\frac{i}{2\varepsilon^2} \left\langle (\underline{x} - \underline{q}), \mathbf{P}\mathbf{Q}^{-1}(\underline{x} - \underline{q}) \right\rangle + \frac{i}{\varepsilon^2} \left\langle \underline{p}, (\underline{x} - \underline{q}) \right\rangle \right) . \tag{3.13}$$

We will see soon that we have to index it by a multi-index $\underline{0} := (0, \ldots, 0)$ instead of a simple 0.

3.2 Raising and lowering operators

Defining the raising and lowering operators is not as trivial as in the one-dimensional case. We have no linear arrangement of all functions $\phi_{\underline{k}}$. In fact the index $\underline{k}$ is a vector- or multi-index:

$$\underline{k} := (k_0, \ldots, k_{D-1}) \in \mathbb{N}_0^D . \tag{3.14}$$

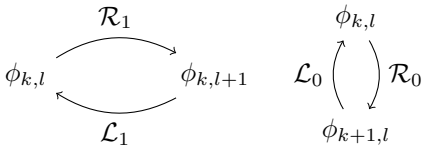
Its *length* is defined as:

$$|\underline{k}| := \sum_{i=0}^{D-1} k_i \tag{3.15}$$

and its *factorial* as:

$$\underline{k}! := (k_0!)(k_1!) \cdots (k_{D-1}!) = \prod_{i=0}^{D-1} k_i! . \tag{3.16}$$

In the following we will set $D = 2$ for most examples. In the two-dimensional case we take $\underline{k} = (k, l)$ and get the grid of functions $\phi_{k,l}$ indexed by two non-negative integers k and l as shown in figure 3.1.
Every arrow stands for a raising operator $\mathcal{R}$ and we see that there are two kinds of arrows, vertical and horizontal ones. Following an arrow only one of the two indices (k, l) changes. This is shown in more details in figure 3.2.
This fact suggests that we assign a raising operator to each arrow type hence we get two operators $\mathcal{R}_0$ and $\mathcal{R}_1$ which are of different nature. We can assign a distinct set $\{\mathcal{R}_i, \mathcal{L}_i\}$ to each axis $i \in [0, \ldots, D-1]$ of the lattice.



(a) (b)

Figure 3.2: The two raising and lowering operator pairs in action.

Leaving this introductionary example and following the formal derivation from [8] now, we take $\underline{v} \in \mathbb{C}^D$, define $\underline{y} := -i\varepsilon^2\nabla_x$ and begin with:

$$\mathcal{R}_v := \frac{1}{\sqrt{2\varepsilon^2}}\left(\left\langle -i\mathbf{P}\overline{\underline{v}}, (\underline{x}-\underline{q})\right\rangle - i\left\langle \mathbf{Q}\overline{\underline{v}}, (\underline{y}-\underline{p})\right\rangle\right) \tag{3.17}$$

$$= \frac{1}{\sqrt{2\varepsilon^2}}\left(i\left\langle \mathbf{P}\overline{\underline{v}}, (\underline{x}-\underline{q})\right\rangle - i\left\langle \mathbf{Q}\overline{\underline{v}}, (\underline{y}-\underline{p})\right\rangle\right) \tag{3.18}$$

$$= \frac{i}{\sqrt{2\varepsilon^2}}\left(\left\langle \mathbf{P}\overline{\underline{v}}, (\underline{x}-\underline{q})\right\rangle - \left\langle \mathbf{Q}\overline{\underline{v}}, (\underline{y}-\underline{p})\right\rangle\right) \tag{3.19}$$

and similarly we get:

$$\mathcal{L}_v := \frac{1}{\sqrt{2\varepsilon^2}}\left(\left\langle \overline{-i\mathbf{P}}\underline{v}, (\underline{x}-\underline{q})\right\rangle + i\left\langle \overline{\mathbf{Q}}\underline{v}, (\underline{y}-\underline{p})\right\rangle\right) \tag{3.20}$$

$$= \frac{1}{\sqrt{2\varepsilon^2}}\left(-i\left\langle \overline{\mathbf{P}}\underline{v}, (\underline{x}-\underline{q})\right\rangle + i\left\langle \overline{\mathbf{Q}}\underline{v}, (\underline{y}-\underline{p})\right\rangle\right) \tag{3.21}$$

$$= -\frac{i}{\sqrt{2\varepsilon^2}}\left(\left\langle \overline{\mathbf{P}}\underline{v}, (\underline{x}-\underline{q})\right\rangle - \left\langle \overline{\mathbf{Q}}\underline{v}, (\underline{y}-\underline{p})\right\rangle\right) . \tag{3.22}$$

For the one-dimensional case we can get back at the definitions from (3.5). Following along the lines of [8] we can compute several commutators:

$$[\mathcal{L}_v, \mathcal{L}_w] = \mathcal{L}_v\mathcal{L}_w - \mathcal{L}_w\mathcal{L}_v = 0 \tag{3.23}$$

$$[\mathcal{R}_v, \mathcal{R}_w] = \mathcal{R}_v\mathcal{R}_w - \mathcal{R}_w\mathcal{R}_v = 0 \tag{3.24}$$

$$[\mathcal{L}_v, \mathcal{R}_w] = \mathcal{L}_v\mathcal{R}_w - \mathcal{R}_w\mathcal{L}_v = \langle \underline{v}, \underline{w}\rangle \tag{3.25}$$

for general $\underline{v}, \underline{w} \in \mathbb{C}^D$. The fact that we are allowed to interchange two raising operators is important and could have been guessed from figure 3.1. There we have:

$$\phi_{1,1} = \mathcal{R}_0\phi_{0,1} = \mathcal{R}_0\mathcal{R}_1\phi_{0,0}$$
$$\phi_{1,1} = \mathcal{R}_1\phi_{1,0} = \mathcal{R}_1\mathcal{R}_0\phi_{0,0}$$

thus the order in which we compute $\phi_{k,l}$ from $\phi_{0,0}$ does not matter. If we flip all arrows we get to the same conclusion but this time for lowering operators.

From the scalar case we know that we can not simply exchange two different operators, remember for example the definition of the number operator $\mathcal{N}$. However, we can go from $\phi_{1,0}$ to $\phi_{0,1}$ by either way:

$$\begin{aligned}\phi_{0,1} &= \mathcal{L}_0\phi_{1,1} = \mathcal{L}_0\mathcal{R}_1\phi_{1,0} \\ \phi_{0,1} &= \mathcal{R}_1\phi_{0,0} = \mathcal{R}_1\mathcal{L}_0\phi_{1,0}\,.\end{aligned}$$

This is the consequence of the third commutation relation above which tells us that the two operators commute iff $\underline{v}$ and $\underline{w}$ are orthogonal.
Continuing with the formal derivation we choose a basis of $\mathbb{R}^D$. For simplicity we take the canonical basis $\{\underline{e}^j\}_{j=0}^{D-1}$. In this basis we can more precisely say what $\mathcal{R}_v$ is. We define the following two sets of ladder operators:

$$\mathcal{R}_j := \mathcal{R}_{e^j} \tag{3.26}$$
$$\mathcal{L}_j := \mathcal{L}_{e^j} \tag{3.27}$$

each containing exactly D operators. Finally we can build the vector-valued operators:

$$\mathcal{R} := \begin{pmatrix} \mathcal{R}_0 \\ \vdots \\ \mathcal{R}_{D-1} \end{pmatrix} \qquad \text{and} \qquad \mathcal{L} := \begin{pmatrix} \mathcal{L}_0 \\ \vdots \\ \mathcal{L}_{D-1} \end{pmatrix}. \tag{3.28}$$

Recalling the defining equations for $\mathcal{R}_v$ and $\mathcal{L}_v$ we can then find explicit expressions for $\mathcal{R}$ and $\mathcal{L}$:

$$\mathcal{R} = \frac{i}{\sqrt{2\varepsilon^2}}\left(\mathbf{P}^{\mathrm{H}}(\underline{x}-\underline{q}) - \mathbf{Q}^{\mathrm{H}}(\underline{y}-\underline{p})\right) \tag{3.29}$$
$$\mathcal{L} = -\frac{i}{\sqrt{2\varepsilon^2}}\left(\mathbf{P}^{\mathrm{T}}(\underline{x}-\underline{q}) - \mathbf{Q}^{\mathrm{T}}(\underline{y}-\underline{p})\right). \tag{3.30}$$

At this point we should stop for a moment and see what happens if we set $D = 1$ now. If we carried out all calculations we would return step by step to the expressions given in (3.5).
With the help of $\mathcal{R}$ we can now go on and define all higher states $\phi_{\underline{k}}$ properly:

$$\phi_{\underline{k}} := \mathcal{R}^{\underline{k}}\phi_{\underline{0}} \tag{3.31}$$
$$= \frac{1}{\sqrt{\underline{k}!}}\mathcal{R}_0^{k_0}\mathcal{R}_1^{k_1}\cdots\mathcal{R}_{D-1}^{k_{D-1}}\phi_{\underline{0}} \tag{3.32}$$
$$= \frac{1}{\sqrt{\prod_{i=0}^{D-1}k_i!}}\prod_{i=0}^{D-1}\mathcal{R}_i^{k_i}\phi_{\underline{0}}\,. \tag{3.33}$$

We can show theorem 3.3 of [8]:

Theorem 1. *The functions $\phi_{\underline{k}}[\Pi](\underline{x})$ form an orthonormal basis of $L^2\left(\mathbb{R}^D\right)$.*

For a proof see again this reference.
To close this section we take a closer look at the application of $\mathcal{R}_j$ on $\phi_{\underline{k}}$:

$$\mathcal{R}_j \phi_{\underline{k}} = \sqrt{k_j + 1}\phi_{\underline{k}'} \tag{3.34}$$

where:

$$\underline{k}' := \underline{k} + \underline{e}^j = (k_0, \ldots, k_{j-1}, k_j + 1, k_{j+1}, \ldots, k_{D-1}) \tag{3.35}$$

and similarly:

$$\mathcal{L}_j \phi_{\underline{k}} = \sqrt{k_j}\phi_{\underline{k}'} \tag{3.36}$$

with:

$$\underline{k}' := \underline{k} - \underline{e}^j = (k_0, \ldots, k_{j-1}, k_j - 1, k_{j+1}, \ldots, k_{D-1})\,. \tag{3.37}$$

This justifies the formula (3.31) allowing us to construct $\phi_{\underline{k}}$ out of $\phi_{\underline{0}}$ by raising each index multiple times as necessary. We build a path through the lattice starting at the origin and ending at the point $\underline{k}$.
From these formula we see that computing $\mathcal{R}\phi_{\underline{0}}$ is sufficient to access all higher order functions. The remaining question is how to do this efficiently. Computing the action of $\mathcal{R}$ is not straight forward because it contains the differential operator $y := -i\varepsilon^2\nabla_x$. For this reason we seek a way to compute $\mathcal{R}\phi_{\underline{0}}$ without ever applying y explicitly. The solution to this task is given by the adjoint pair $\mathcal{R}, \mathcal{L}$ of operators. We can set up a system of two operator equations. First we solve the equation defining $\mathcal{L}$ for y. For simplicity of notation we define $\theta := \frac{i}{\sqrt{2\varepsilon^2}}$ and transform as follows:

$$\begin{aligned}
\mathcal{L} &= -\theta\left(\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) - \mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p})\right) \\
\mathcal{L} &= -\theta\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) + \theta\mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p}) \\
\mathcal{L} + \theta\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) &= \theta\mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p}) \\
\mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p}) &= \frac{1}{\theta}\mathcal{L} + \mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) \\
\underline{y} - \underline{p} &= \frac{1}{\theta}\mathbf{Q}^{\text{-T}}\mathcal{L} + \mathbf{Q}^{\text{-T}}\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) \\
\underline{y} &= \frac{1}{\theta}\mathbf{Q}^{\text{-T}}\mathcal{L} + \mathbf{Q}^{\text{-T}}\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) + \underline{p}\,.
\end{aligned} \tag{3.38}$$

Note that solving $\mathcal{R}$ for y gives us the complex conjugate of this last line:

$$\underline{y} = -\frac{1}{\theta}\mathbf{Q}^{\text{-H}}\mathcal{R} + \mathbf{Q}^{\text{-H}}\mathbf{P}^{\mathrm{H}}(\underline{x} - \underline{q}) + \underline{p}\,. \tag{3.39}$$

In the next step we plug the result (3.38) into the definition of $\mathcal{R}$:

$$
\begin{aligned}
\mathcal{R} &= \theta\left(\mathbf{P}^{\mathrm{H}}(\underline{x}-\underline{q}) - \mathbf{Q}^{\mathrm{H}}(\underline{y}-\underline{p})\right)\\
\mathcal{R} &= \theta\left(\mathbf{P}^{\mathrm{H}}(\underline{x}-\underline{q}) - \mathbf{Q}^{\mathrm{H}}\left(\left(\frac{1}{\theta}\mathbf{Q}^{-\mathrm{T}}\mathcal{L} + \mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}}(\underline{x}-\underline{q}) + \underline{p}\right) - \underline{p}\right)\right)\\
\mathcal{R} &= \theta\left(\mathbf{P}^{\mathrm{H}}(\underline{x}-\underline{q}) - \mathbf{Q}^{\mathrm{H}}\left(\frac{1}{\theta}\mathbf{Q}^{-\mathrm{T}}\mathcal{L} + \mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}}(\underline{x}-\underline{q})\right)\right)\\
\mathcal{R} &= \theta\left(\mathbf{P}^{\mathrm{H}}(\underline{x}-\underline{q}) - \frac{1}{\theta}\mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathcal{L} - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}}(\underline{x}-\underline{q})\right)\\
\mathcal{R} &= \theta\left(-\frac{1}{\theta}\mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathcal{L} + \underbrace{(\mathbf{P}^{\mathrm{H}} - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}})}_{*}(\underline{x}-\underline{q})\right). \qquad (3.40)
\end{aligned}
$$

This is already quite useful a result. But let's see if we can simplify the underbraced part further. Simplifying the $*$ part needs a bit of algebra. We start with the basic relations in (3.11) and multiply the first one by $\mathbf{Q}^{-1}$ from the right:

$$
\begin{aligned}
\mathbf{Q}^{\mathrm{H}}\mathbf{P} - \mathbf{P}^{\mathrm{H}}\mathbf{Q} &= 2i\mathbb{1}\\
\mathbf{Q}^{\mathrm{H}}\mathbf{P}\mathbf{Q}^{-1} - \mathbf{P}^{\mathrm{H}}\mathbf{Q}\mathbf{Q}^{-1} &= 2i\mathbf{Q}^{-1}\\
\mathbf{Q}^{\mathrm{H}}\mathbf{P}\mathbf{Q}^{-1} - \mathbf{P}^{\mathrm{H}} &= 2i\mathbf{Q}^{-1}\\
\mathbf{P}^{\mathrm{H}} - \mathbf{Q}^{\mathrm{H}}\underbrace{\mathbf{P}\mathbf{Q}^{-1}}_{**} &= -2i\mathbf{Q}^{-1}.
\end{aligned}
$$

Now we are left with $**$ where we can apply the other fundamental relation which we have to transform a little bit first:

$$
\begin{aligned}
\mathbf{P}^{\mathrm{T}}\mathbf{Q} - \mathbf{Q}^{\mathrm{T}}\mathbf{P} &= \mathbf{0}\\
\mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{Q} - \mathbf{Q}^{-\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{P} &= \mathbf{0}\\
\mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{Q} &= \mathbf{P}.
\end{aligned}
$$

The last line can now be used to replace the $\mathbf{P}$ in $**$ which yields:

$$
\begin{aligned}
\mathbf{P}^{\mathrm{H}} - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{Q}^{-1} &= -2i\mathbf{Q}^{-1}\\
\mathbf{P}^{\mathrm{H}} - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathbf{P}^{\mathrm{T}} &= -2i\mathbf{Q}^{-1}.
\end{aligned}
$$

This is the part $*$ we wanted to simplify. Going back to (3.40) we can write:

$$
\begin{aligned}
\mathcal{R} &= \theta\left(-\frac{1}{\theta}\mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathcal{L} - 2i\mathbf{Q}^{-1}(\underline{x}-\underline{q})\right)\\
\mathcal{R} &= -\mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathcal{L} - 2i\theta\mathbf{Q}^{-1}(\underline{x}-\underline{q}).
\end{aligned}
$$

At the end of the day we get:

$$
\boxed{\mathcal{R} = \sqrt{\frac{2}{\varepsilon^2}}\mathbf{Q}^{-1}(\underline{x}-\underline{q}) - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}}\mathcal{L}} \qquad (3.41)
$$

where we reinserted the term for θ and took into account the i therein. Of course we would get the very same result if we used (3.39) and plugged it into the definition of $\mathcal{L}$. Just for the sake of completeness we state the complex conjugate result for the $\mathcal{L}$ operator too:

$$\boxed{\mathcal{L} = \sqrt{\frac{2}{\varepsilon^2}}\overline{\mathbf{Q}}^{-1}(\underline{x} - \underline{q}) - \mathbf{Q}^{\mathrm{T}}\mathbf{Q}^{-\mathrm{H}}\mathcal{R}} \tag{3.42}$$

with a similar derivation as the one above.

3.3 Higher order basis functions

With this equation at hand we can continue in (3.34) where we left off computing $\mathcal{R}_d\phi_{\underline{k}}$. We start applying the operator for the d-th direction. But we do not have an explicit expression for $\mathcal{R}_d$ and on the other hand we need the result for all $d \in [0, \ldots, D-1]$. Therefore is seems wise to do these computations simultaneously for all D components:

$$\begin{pmatrix} \sqrt{k_0 + 1}\phi_{\underline{k}+\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1} + 1}\phi_{\underline{k}+\underline{e}^{D-1}} \end{pmatrix} = \begin{pmatrix} \mathcal{R}_0\phi_{\underline{k}} \\ \vdots \\ \mathcal{R}_{D-1}\phi_{\underline{k}} \end{pmatrix} = \mathcal{R}\phi_{\underline{k}} \tag{3.43}$$

Using the formula (3.41) for $\mathcal{R}$ gives us:

$$\begin{pmatrix} \sqrt{k_0 + 1}\,\phi_{\underline{k}+\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1} + 1}\,\phi_{\underline{k}+\underline{e}^{D-1}} \end{pmatrix} = \sqrt{\frac{2}{\varepsilon^2}}\mathbf{Q}^{-1}(\underline{x} - \underline{q})\phi_{\underline{k}} - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}} \begin{pmatrix} \sqrt{k_0}\phi_{\underline{k}-\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1}}\phi_{\underline{k}-\underline{e}^{D-1}} \end{pmatrix}. \tag{3.44}$$

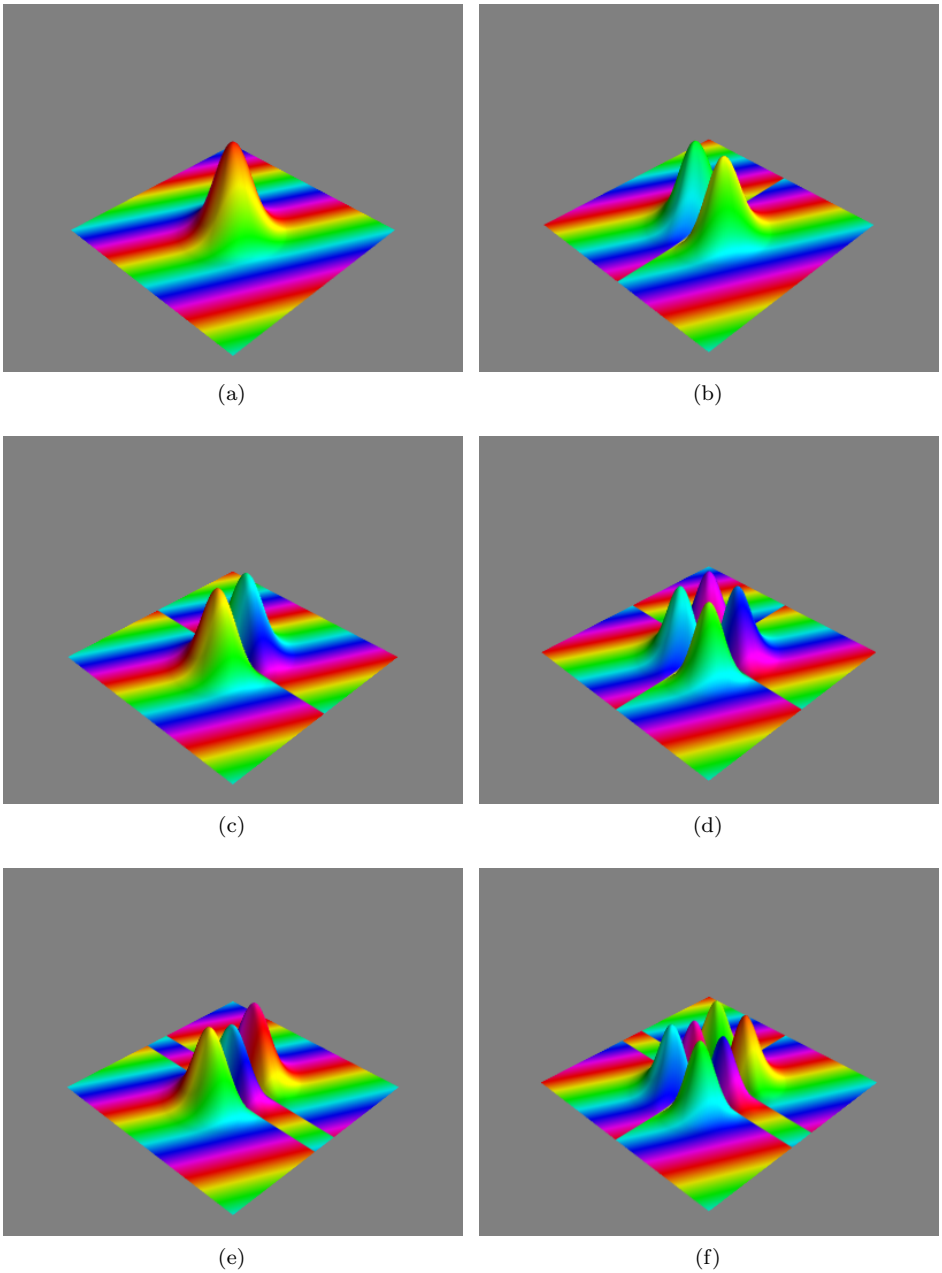
From this we get the new functions $\phi_{\underline{k}+\underline{e}^d}$ as:

$$\begin{pmatrix} \phi_{\underline{k}+\underline{e}^0} \\ \vdots \\ \phi_{\underline{k}+\underline{e}^{D-1}} \end{pmatrix} = \left(\sqrt{\frac{2}{\varepsilon^2}}\mathbf{Q}^{-1}(\underline{x} - \underline{q})\phi_{\underline{k}} - \mathbf{Q}^{\mathrm{H}}\mathbf{Q}^{-\mathrm{T}} \begin{pmatrix} \sqrt{k_0}\phi_{\underline{k}-\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1}}\phi_{\underline{k}-\underline{e}^{D-1}} \end{pmatrix} \right) \oslash \begin{pmatrix} \sqrt{k_0 + 1} \\ \vdots \\ \sqrt{k_{D-1} + 1} \end{pmatrix}$$

where the operator $\oslash$ denotes component-wise division. In a next step we use this formula for evaluation of all $\phi_{\underline{k}}$ basis functions of D dimensional semi-classical wavepackets. This last formula is so important that it should carry a box too but it seems there is no space left.

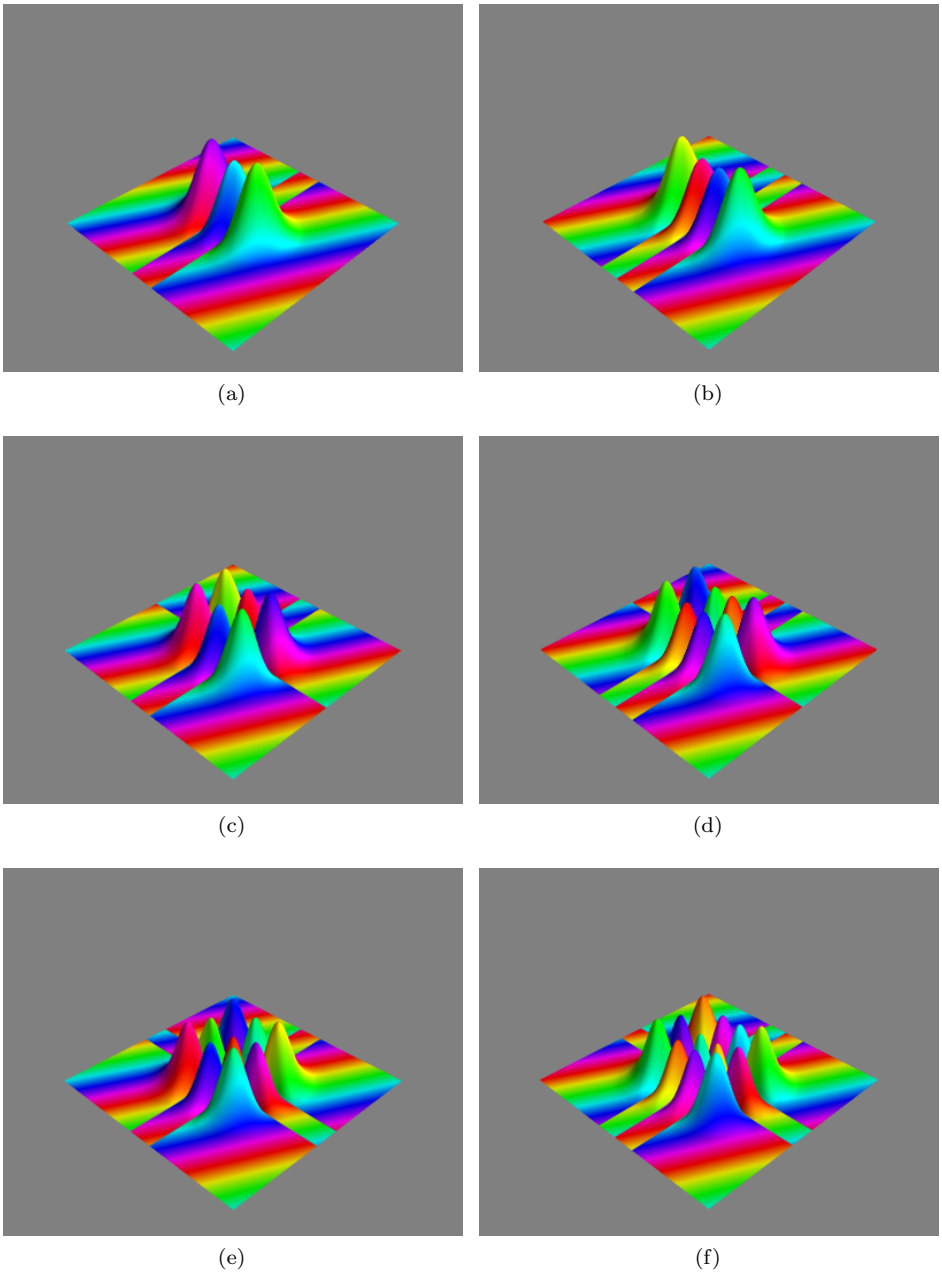
3.4 Construction of wavepackets

In the previous section we saw how to compute the functions $\phi_{\underline{k}}[\Pi](\underline{x})$. Now we can take a very general set $\mathfrak{K}$ of indices $\underline{k}$ and use the corresponding functions



(a) (b) (c) (d) (e) (f)

Figure 3.3: Plots of the first few functions $\phi_{k,l}$ with parameters set to $\underline{q} = \underline{0}$, $\underline{p} = (1, \frac{1}{2})$, $\mathbf{Q} = \mathbb{1}$, $\mathbf{P} = i\mathbb{1}$ and $\varepsilon = 1$. The surface represents ten times the value $\sqrt{\langle \phi_{k,l} \,|\, \phi_{k,l} \rangle}$ where the factor of 10 is just for visual purpose. For an explanation of the colours, see appendix B. (a) $\phi_{0,0}$ (b) $\phi_{1,0}$ (c) $\phi_{0,1}$ (d) $\phi_{1,1}$ (e) $\phi_{0,2}$ (f) $\phi_{1,2}$



(a) (b) (c) (d) (e) (f)

Figure 3.4: Plots of the first few functions $\phi_{k,l}$ with parameters set to $\underline{q} = \underline{0}$, $\underline{p} = (1, \frac{1}{2})$, $\mathbf{Q} = \mathbb{1}$, $\mathbf{P} = i\mathbb{1}$ and $\varepsilon = 1$. The surface represents ten times the value $\sqrt{\langle \phi_{k,l} \,|\, \phi_{k,l} \rangle}$ where the factor of 10 is just for visual purpose. For an explanation of the colours, see appendix B. (a) $\phi_{2,0}$ (b) $\phi_{3,0}$ (c) $\phi_{2,1}$ (d) $\phi_{3,1}$ (e) $\phi_{2,2}$ (f) $\phi_{3,2}$

$\phi_{\underline{k}}$ to build a (more or less truncated) basis for $L^2(\mathbb{R}^D)$. In a first step we can construct so called *scalar wavepackets* Φ by linear combinations:

$$\Phi(\underline{x}) := \exp\left(\frac{iS}{\varepsilon^2}\right) \sum_{\underline{k}\in\mathfrak{K}} c_{\underline{k}} \phi_{\underline{k}} \tag{3.45}$$

where the coefficients $c_{\underline{k}} \in \mathbb{C}$ depend on time only and the basis functions $\phi_{\underline{k}}$ depend on space but also on time through the parameter set $\Pi(t)$ which is time-dependent[1]. We added a global phase S which is also time-dependent. An overly precise notation reads:

Definition 5 (Scalar semi-classical wavepacket)**.**

$$|\Phi\rangle := \Phi\left[\Pi(t)\right](\underline{x}, t) = \exp\left(\frac{iS(t)}{\varepsilon^2}\right) \sum_{\underline{k}\in\mathfrak{K}} c_{\underline{k}}(t) \phi_{\underline{k}}\left[\Pi(t)\right](\underline{x})\,. \tag{3.46}$$

Every single basis function $\phi_{\underline{k}}$ is a perfectly valid wavepacket too. Sometimes we will append the parameter S to the set Π and use the notation $\Pi = \{q, p, Q, P, S\}$. This should be clear from the context. Also we will drop the time variable since we look at wavepackets at fixed times.

3.5 Basis set expansion and basis shapes

The above formulation is a basis expansion for the true wavefunction $\varphi(\underline{x}, t)$. Remember that the set $\{\phi_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}}$ is a (complete) basis of the function space $L^2(\mathbb{R}^D)$. Hence the basis expansion is exact if we take the full lattice $\mathfrak{K} = \mathbb{N}_0^D$ of indices. In theoretical considerations we can use the full lattice but for all practical purposes we need to truncate the basis and make the set $\mathfrak{K}$ finite. This can be done in various ways and we refer to the *shape* of a basis set if we speak about these details of $\mathfrak{K}$. Basis shapes usually depend on some parameters θ, we occasionally write $\mathfrak{K}(\theta)$ for this.
For a first ansatz we can use a hypercubic basis set $\mathfrak{K}$ which means that we take the subset of all lattice points for which $\underline{k} < \underline{K}$ holds. The components of $\underline{K}$ specify the number of points along each of the D directions. A more formal definition is:

Definition 6 (Hypercubic basis shape)**.**

$$\mathfrak{K}(\underline{K}) := \left\{\underline{k} \in \mathbb{N}_0^D : k_d < K_d \,\forall\, d \in [0, \ldots, D-1]\right\}\,. \tag{3.47}$$

If we use this basis shape we call the resulting wavepacket *dense*. By $|\mathfrak{K}|$ we denote the *basis size*, the overall number of basis functions $\phi_{\underline{k}}$ we use. In the hypercubic case this is obviously:

$$|\mathfrak{K}| = \prod_{d=0}^{D-1} K_d\,. \tag{3.48}$$

As a shorthand notation to specify the hypercubic shape we simply write $\mathfrak{K} = \underline{K} = [K_0, \ldots, K_{D-1}]$. We should think of $\underline{K}$ as being both the vector in the

[1] In general we neglect explicit time-dependence of the parameter set Π in our notation.

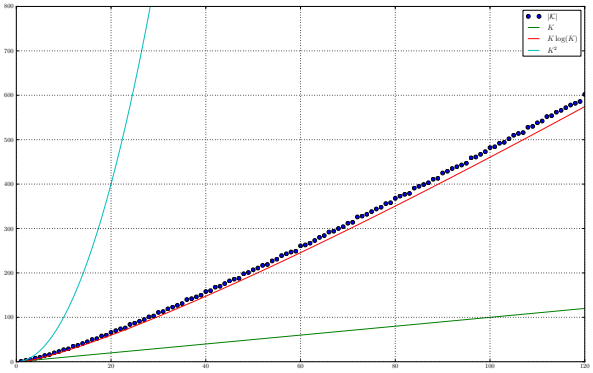


Figure 3.5: The size $|\mathfrak{K}|$ of two-dimensional hyperbolic basis shapes with various cut-off values K.

lattice $\mathbb{N}_0^D$ and the set of all index points in the hypercube spanned by the origin $\underline{0} = (0, \ldots, 0)$ and $\underline{K} - \underline{1} = (K_0 - 1, \ldots, K_{D-1} - 1)$, depending on the context. The size of this basis shape grows exponentially with the number D of dimensions. Therefore we introduce more sparse basis sets which grow slower as the number of dimensions increases. The first example is the *hyperbolic cut* basis shape defined as follows:

Definition 7 (Hyperbolic cut basis shape)**.**

$$\mathfrak{K}(K) := \left\{ \underline{k} \in \mathbb{N}_0^D : \prod_{d=0}^{D-1} (1 + k_d) \leq K \right\} \tag{3.49}$$

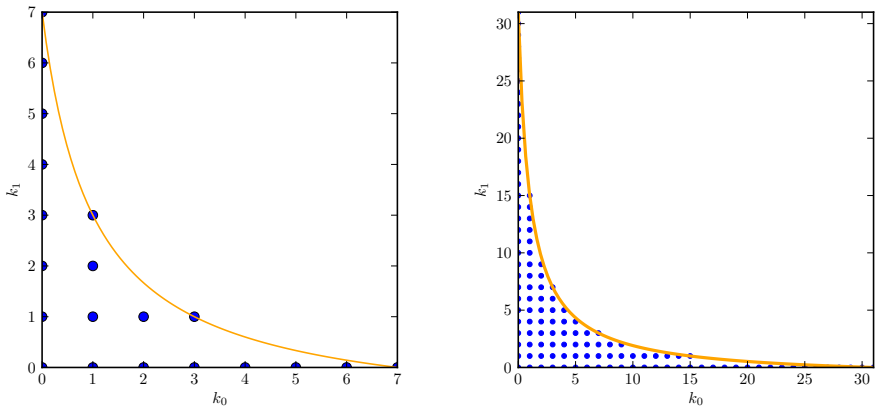
where we limit the number of basis functions by hyperbolic cuts. For this we introduce a scalar parameter $K \in \mathbb{N}$ which we call *sparsity* in this context. The number of basis functions is then bounded by:

$$|\mathfrak{K}| \leq \mathcal{C} K \left(\log K\right)^{D-1} . \tag{3.50}$$

where $\mathcal{C}$ is some constant.
Figure 3.5 shows the application of this bound to the two-dimensional case.
As we see in figures 3.6 and 3.7 this basis shape has long tails consisting of functions with high frequencies in one direction. Sometimes we do not need these tails. We can combine the two basis shapes introduced and define a new shape:

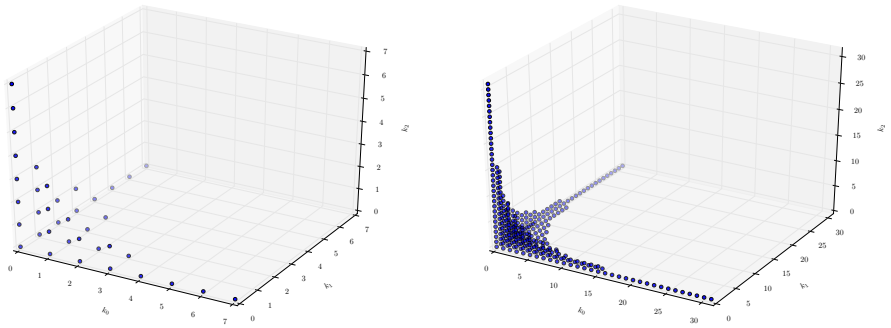
Definition 8 (Hyperbolic cut basis shape with limits)**.**

$$\mathfrak{K}(K, \underline{L}) := \left\{ \underline{k} \in \mathbb{N}_0^D : \prod_{d=0}^{D-1} (1 + k_d) \leq K \wedge k_d < L_d \, \forall \, d \in [0, \ldots, D-1] \right\} . \tag{3.51}$$



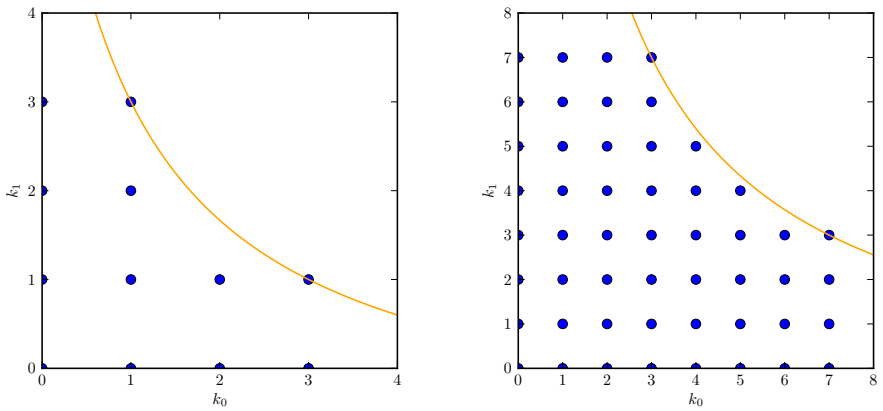
(a) (b)

Figure 3.6: The lattice nodes that are part of a two-dimensional hyperbolic cut basis shape. Additionally the cut-off function is shown. Compared to the full hypercubic basis shape the sparsity of this type of basis shape becomes clearly visible. (a) $K = 8$ (b) $K = 32$



(a) (b)

Figure 3.7: The lattice nodes that are part of a three-dimensional hyperbolic cut basis shape. Compared to the full hypercubic basis shape the sparsity of this type of basis shape becomes clearly visible. (a) $K = 8$ (b) $K = 32$



(a) (b)

Figure 3.8: The lattice nodes that are part of a two-dimensional limited hyperbolic cut basis shape. Compared to the unlimited hyperbolic cut basis shape the long tails are missing here. (a) $K = 8$ and $\underline{L} = (4, 4)$ (b) $K = 32$ and $\underline{L} = (8, 8)$

The parameter $K \in \mathbb{N}$ is again the sparsity defining the hyperbolic cuts. The parameter $\underline{L}$ is a list of D elements with each $L_d \in \mathbb{N}$. These limits act as sharp upper bounds on the entries of $\underline{k}$. In that way we can combine the two previous basis shapes. If we choose $L_d \geq K \, \forall \, d \in [0, \ldots, D-1]$ then we obtain a simple hyperbolic cut basis shape. On the other hand, if we set $K \geq \prod_{d=0}^{D-1} L_d$ we are back in the full hypercubic case.
There exist many more possibilities for specific basis shapes. For a generic basis shape $\mathfrak{K}$ a fundamental property has to hold. We can state it by the following implication:

$$\forall \, \underline{k} \in \mathbb{N}_0^D : \underline{k} \in \mathfrak{K} \Rightarrow \underline{k} - \underline{e}^d \in \mathfrak{K} \forall \, d \in [0, \ldots, D-1] \, . \tag{3.52}$$

In other words, for each $\underline{k} \in \mathfrak{K}$ it must hold that for all $d = 0, \ldots, D-1$ the multi-index defined by $k' := \underline{k} - \underline{e}^d$ has no negative components and $\underline{k}' \in \mathfrak{K}$. In less formal terms this means that an index $\underline{k}$ is part of $\mathfrak{K}$ iff all its backward neighbours are also part of $\mathfrak{K}$. This condition is necessary for the recursive evaluation of wavepackets which we will show later.
Probably the most general way to write a scalar wavepacket (3.46) now is:

$$|\Phi\rangle := \Phi \left[\Pi(t), \mathfrak{K}(t) \right] (\underline{x}, t) = \exp\left(\frac{iS(t)}{\varepsilon^2} \right) \sum_{\underline{k} \in \mathfrak{K}(t)} c_{\underline{k}}(t) \phi_{\underline{k}} \left[\Pi(t) \right] (\underline{x}) \tag{3.53}$$

which uses an arbitrary, possibly time-adaptive basis shape $\mathfrak{K}(t)$. In principle we can exchange the basis shapes at each timestep during the simulation. This provides us with adaptivity for all parameters θ a basis shape depends on.
Sometimes we will need to bring the elements $\underline{k}$ of a basis shape $\mathfrak{K}$ into a fixed total order. This is done by the *linearisation mapping*.

Definition 9 (Linearisation mapping)**.** *A mapping:*

$$\begin{aligned} \mu : \mathfrak{K} &\to \mathbb{N}_0 \\ \underline{k} &\mapsto n \end{aligned}$$

that fixes a total order of the set $\mathfrak{K}$*.*

If it is not clear from the context we denote this mapping by $\mu_{\mathfrak{K}}$ to mark explicitly which basis shape it belongs to. In practical cases we usually have $\mu(\underline{0}) = 0$ but this is not a requirement.
Finally the following two algorithms 1 and 2 can be used to find the neighbourhood of a given multi-index $\underline{k}$ in a general basis shape $\mathfrak{K}$.

Algorithm 1 Find forward neighbours

Require: The number D of space dimensions
Require: The basis shape $\mathfrak{K}$
Require: The multi-index $\underline{k}$ whose neighbours we search
  // List for the result
  $N := \{\}$
  // Find neighbourhood
  for $d = 0$ **to** $d = D - 1$ **do**
    $\underline{k}' := \underline{k} + \underline{e}^d$
    if $\underline{k}' \in \mathfrak{K}$ **then**
      $N = N \cup \{(\underline{k}', d)\}$
    end if
  end for
  return N

Algorithm 2 Find backward neighbours

Require: The number D of space dimensions
Require: The basis shape $\mathfrak{K}$
Require: The multi-index $\underline{k}$ whose neighbours we search
  // List for the result
  $N := \{\}$
  // Find neighbourhood
  for $d = 0$ **to** $d = D - 1$ **do**
    $\underline{k}' := \underline{k} - \underline{e}^d$
    if $\underline{k}' \in \mathfrak{K}$ **then**
      $N = N \cup \{(\underline{k}', d)\}$
    end if
  end for
  return N

3.5.1 Basis shape transformation mappings

Given two basis shapes $\mathfrak{K}$ and $\mathfrak{K}'$ and their linearisation mappings:

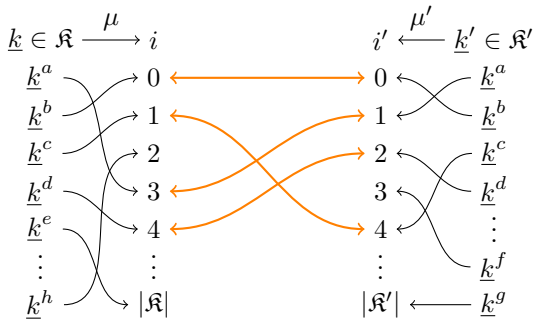


Figure 3.9: The two mappings μ and μ' together with the transformation rule T (orange arrows).

$$\mu : \mathfrak{K} \to \mathbb{N}_0$$
$$\mu' : \mathfrak{K}' \to \mathbb{N}_0 \,.$$

We look for a way how to change a data array $\underline{c} \in \mathbb{C}^{|\mathfrak{K}|}$ (storing complex scalar data for each multi-index $\underline{k} \in \mathfrak{K}$ at the position $\underline{c}_{\mu(\underline{k})}$) if we replace the basis shape $\mathfrak{K}$ by $\mathfrak{K}'$. The problem is that the two mappings are in general incompatible with each other. Hence we have to permute the elements of the array $\underline{c}$. Referring to figure 3.9, the task is now to connect the keys i and i' such that for all multi-indices $\underline{k} \in \mathfrak{K} \cap \mathfrak{K}'$ it holds that:

$$(\mu')^{-1} \left(T \left(\mu \left(\underline{k} \right) \right) \right) = \underline{k} \,.$$

The transformation T is bijective if and only if $\mathfrak{K} \equiv \mathfrak{K} \cap \mathfrak{K}' \equiv \mathfrak{K}'$.
When remapping linearly indexed data vectors $\underline{c}_{\mu(\underline{k})}$, we drop the entries for all $\underline{k} \in \mathfrak{K} \setminus \mathfrak{K}'$ and we fill in zero values for all $\underline{k} \in \mathfrak{K}' \setminus \mathfrak{K}$. The procedure is shown in algorithm 3 below.

Algorithm 3 Transformation of basis shapes and data remapping

Require: The old basis shape $\mathfrak{K}$
Require: The new basis shape $\mathfrak{K}'$
Require: An array $\underline{c}$ of length $|\mathfrak{K}|$ containing arbitrary data
  // Set up the new array of length $|\mathfrak{K}'|$
  $\underline{c}' := \underline{0} \in \mathbb{C}^{|\mathfrak{K}'|}$
  // Copy over the data we can keep
  for $\underline{k} \in \mathfrak{K} \cap \mathfrak{K}'$ **do**
    // Compute linear mapping of $\underline{k}$ in both basis shapes
    $i := \mu \left(\underline{k} \right)$
    $j := \mu' \left(\underline{k} \right)$
    // Update the array
    $\underline{c}' \left[j \right] = \underline{c} \left[i \right]$
  end for

An example of such a mapping is given in figure 3.10.

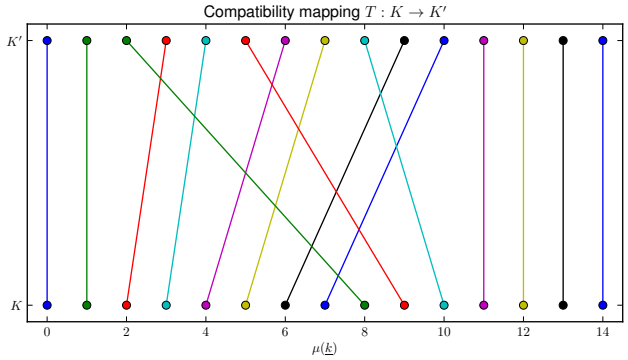


Figure 3.10: An example of a basis shape transformation mapping T. The original shape was $\mathfrak{K} = \underline{K} = (4, 2)$ and $|\mathfrak{K}| = 8$ while the new shape is $\mathfrak{K}' = \underline{K}' = (5, 3)$ and $|\mathfrak{K}'| = 15$.

3.5.2 Basis shape extensions

For computing the gradients of wavepackets (see section 3.8) we need to extend the basis shape. Informally spoken, extending a basis shape means that we add a node in each direction. For each lattice point of a shape we add all its neighbours. A formal definition is:

Definition 10 (Basis shape extension)**.** *Given a basis shape $\mathfrak{K}$ we define its extension $\overline{\mathfrak{K}}$ by:*

$$\overline{\mathfrak{K}} := \mathfrak{K} \cup \left\{ \underline{k}' : \underline{k}' = \underline{k} + \underline{e}^d \, \forall \, d \in [0, \ldots, D-1] \, \forall \, \underline{k} \in \mathfrak{K} \right\} .$$

This defines the most tight extension. But any even larger basis shape is a valid extension too. In any case it holds that $\mathfrak{K} \subset \overline{\mathfrak{K}}$.

This definition is not handy enough to work with. For some of the less complex basis shapes $\mathfrak{K}(\theta)$ it is often possible to express the extended shape $\overline{\mathfrak{K}(\theta')}$ simply by modifying the parameters θ the shape depends on.
If we look at a hypercubic basis shape $\mathfrak{K}(\underline{K})$ then we can express its extension $\overline{\mathfrak{K}(\underline{K}')}$ by the new parameters $\underline{K}'$ which obviously are:

$$\underline{K}' = [K_0 + 1, \ldots, K_{D-1} + 1] . \tag{3.54}$$

The attentive reader may notice that in case $D > 1$ this new basis includes one node too much, namely the lattice point $(K_0, \ldots, K_{D-1})$. This however poses no problems beside wasting a very small amount of memory. A basis shape extension has not to be tight with respect to the above definition.
For the hyperbolic cut shape, extension is less trivial. Assume that the sparsity is K. We first look at the extension of a two-dimensional shape. Here the two nodes $(K-1, 0)$ and $(0, K-1)$ are the outermost ones. Since the whole basis shape is symmetric under permutation of the axes, we focus only on $(K-1, 0)$ lying on the first axis. We have to make sure that its neighbours are part of $\overline{\mathfrak{K}}$. This is guaranteed (by definition 7) if we can construct a new hyperbolic cut shape $\mathfrak{K}'(K')$ that contains the node $(K-1, 1)$. By using the equation of the above definition we get:

$$(1 + K - 1)(1 + 1) \leq K'$$

which we can solve for the minimal K' and find that:

$$K' = 2K\,. \tag{3.55}$$

In the general case of $D > 1$ dimensions we have a similar equation:

$$(1 + K - 1)(1 + 1) \cdots (1 + 1) = (1 + K - 1) \left(\prod_{d=1}^{D-1} (1 + 1) \right) \leq K'$$

from which we obtain:

$$K' = 2^{D-1} K\,. \tag{3.56}$$

This is only valid for $D > 1$ and gives for $D = 1$ the wrong result $K' = K$. If we want a single formula valid for all dimensions then we have to start from the point $\underline{k} = (K, 0, \ldots, 0)$. For this point we get:

$$(1 + K) \left(\prod_{d=1}^{D-1} (1 + 1) \right) \leq K'$$

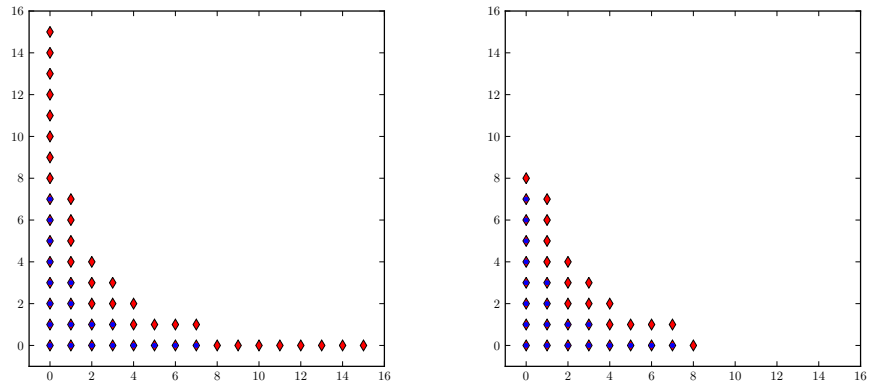
giving:

$$K' = 2^{D-1}(K + 1)\,. \tag{3.57}$$

However we should note that this leads to overly big extensions for $D > 1$ compared to the other formula. For example if we start with a two-dimensional basis shape with $K = 4$ then we get $K' = 8$ and $K' = 10$ respectively. The basis sizes of the extended shapes are then 20 and 27 and the tight extension given by the direct definition would have size 14 but is not of hyperbolic cut type.
Extending a hyperbolic cut basis shape with limits is easy again. First we change the limits $\underline{L}$ according to (3.54). Then we increase K by using one of the formulae (3.55) or (3.56) depending on the dimensionality. That is all we need to do for this type of basis shape.
We can use the limited hyperbolic cut basis shape to produce more tight extensions of the standard hyperbolic cut basis shapes. For an example, refer to figure 3.11.

3.6 Evaluation of wavepackets

For the numerical simulation we will need to evaluate a wavepacket at sets of grid nodes $\underline{\gamma} \in \mathbb{R}^D$. The formula for the ground state (3.13) and the recursion (3.44) for higher order basis functions is in principle all we need to do this. Even if this seems to be very simple at first glance there are a few tricky details involved. In this section we start from the mathematical formulae and work towards an algorithmic description for computing $\Phi(\underline{\gamma})$.
First we can evaluate the ground state ϕ_0 directly by algorithm 4. If this is programmed by using vectorisation for the linear algebra functions we can even



(a) (b)

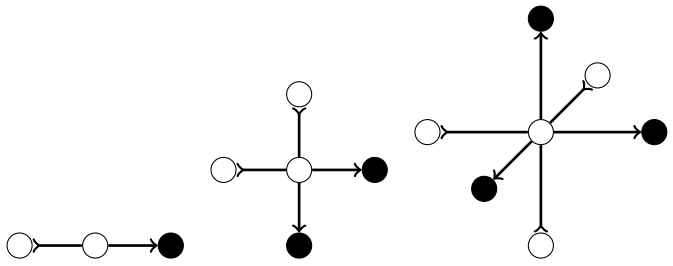
Figure 3.11: Comparison of two methods to extend a hyperbolic cut basis shape with $K = 8$. The original nodes (blue circles) and the nodes of the extension (red diamonds) are shown for both methods. The difference in basis size is 14 nodes. (a) Extension is again an unlimited hyperbolic cut shape with $K = 16$. (b) Extension is a limited hyperbolic cut shape with $K = 16$ and $\underline{L} = (9, 9)$. This extension introduces only 3 nodes more than required.

perform the evaluation on a set $\Gamma = \{\gamma_i\}_i$ of nodes simultaneously. The return value is then not a single scalar value but an array of scalars. Usually we imagine this to be a row vector of $|\Gamma|$ elements.
For the higher order basis functions $\phi_{\underline{k}}$ we employ the recursion formula. With the help of this relation we can compute any $\phi_{\underline{k}}$ given some of the predecessors. For $D = 1$ this is easy since we just compute ϕ_{k+1} from ϕ_k and ϕ_{k-1}. In the multi-dimensional case it is less obvious what happens. To compute the set of all successors $\{\phi_{\underline{k}+\underline{e}^d}\}_{d=0}^{D-1}$ we need the function $\phi_{\underline{k}}$ as well as all antecessors $\{\phi_{\underline{k}-\underline{e}^d}\}_{d=0}^{D-1}$. We call this rule that specifies how we get new functions from old ones a *stencil* and denote it by $\mathcal{S}^D$. The figure 3.12 shows the stencils we get in one, two and three dimensions.
For all these recurrences we need starting points. The only point we know is $\phi_{\underline{0}}$. But as we found earlier we can insert a 0 for all $\underline{k}$ where $\exists k_d$ with $k_d < 0$. Hence we have indeed enough known values to get the process started. An example of how this works in two dimensions is shown in figure 3.13 and algorithm 5.
The most severe issue with this naive stencil application is that we compute almost all functions twice, here this becomes obvious the first time for $\phi_{1,1}$. Even worse, in D dimensions we would compute almost all functions D times!
Therefore we seek a better way to organise the computations. To achieve this goal we modify the stencil in a way that seems to be counter intuitive at first. Define the modified *cheap stencil* as follows.

Definition 11 (Cheap recursion stencil)**.** *The cheap recursion stencil does not compute all D outputs $\phi_{\underline{k}+\underline{e}^d}$ for all $d \in [0, \ldots, D-1]$ but only one output for a single and fixed direction d. We denote the cheap stencil for direction d and in*

Algorithm 4 Evaluate the ground state ϕ_0 directly

Require: The number D of space dimensions
Require: The Hagedorn parameter set $\Pi = \{q, p, Q, P\}$
Require: The semi-classical scaling parameter ε
Require: The grid node $\underline{\gamma} \in \mathbb{R}^D$
  // Whether to include the problematic prefactor or not
  if prefactor is True **then**
    $\alpha := (\pi\varepsilon^2)^{-\frac{D}{4}} (\det \mathbf{Q})^{-\frac{1}{2}}$
  else
    $\alpha := (\pi\varepsilon^2)^{-\frac{D}{4}}$
  end if
  // The exponent
  $\underline{u} := \underline{\gamma} - \underline{q}$
  $\beta_1 := \left\langle \underline{u}, \mathbf{P}\mathbf{Q}^{-1}\underline{u} \right\rangle$
  $\beta_2 := \left\langle \underline{p}, \underline{u} \right\rangle$
  // The full ground state
  $\phi_{\underline{0}} := \alpha \exp\left(\frac{i}{\varepsilon^2}\left(\frac{1}{2}\beta_1 + \beta_2\right)\right)$
  return $\phi_{\underline{0}}$



(a) $\mathcal{S}^1$ (b) $\mathcal{S}^2$ (c) $\mathcal{S}^3$

Figure 3.12: The basis function recursion stencils in one, two and three dimensions. The black nodes can be computed given the white ones. This works in one single step, i.e. we need all white ones and get all black ones.

D dimensions by $\mathcal{S}_d^D$*. In D dimensions we have D different stencils:*

$$\mathcal{S}_d^D \quad d \in [0, \ldots, D-1] \,.$$

Figures 3.14 and 3.15 give an impression how the stencils look like and how they work in two and three dimensions. In one dimension we obviously find $\mathcal{S}_0^1 \equiv \mathcal{S}^1$. In all dimensions it holds that we recover the full stencil $\mathcal{S}^D$ as the sum:

$$\mathcal{S}^D = \bigoplus_{d=0}^{D-1} \mathcal{S}_d^D$$

where all stencils are applied centred at the very same node $\phi_{\underline{k}}$.
Formally we have just used the d-th row of equation (3.44) only to compute the application of $\mathcal{S}_d^D$ to $\phi_{\underline{k}}$. By using colon or slicing notation we can write the

Algorithm 5 Evaluate higher order states ϕ_k recursively (naive version)

Require: The number D of space dimensions
Require: The Hagedorn parameter set $\Pi = \{q, p, Q, P\}$
Require: The semi-classical scaling parameter ε
Require: The basis shape $\mathfrak{K}$ and its linearisation mapping μ
Require: The grid node $\underline{\gamma} \in \mathbb{R}^D$
  // Storage space for the result
  $\underline{\psi} := \underline{0} \in \mathbb{C}^{|\mathfrak{K}|}$
  // Evaluate the ground state by algorithm 4
  $\underline{\psi}[\mu(\underline{0})] =$ **evaluate_ground_state**$(D, \Pi, \varepsilon, \underline{\gamma})$
  // Loop over the multi-indices
  for $\underline{k} \in \mathfrak{K}$ **do**
    // Backward neighbours
    $\underline{\xi} := \underline{0} \in \mathbb{C}^D$
    for $d = 0$ **to** $d = D - 1$ **do**
      $\underline{k}' := \underline{k} - \underline{e}^d$
      if $\underline{k}' \in \mathfrak{K}$ **then**
        $\underline{\xi}[d] = \sqrt{\underline{k}[d]}\, \underline{\psi}[\mu(\underline{k}')]$
      end if
    end for
    // Compute 3-term recursion
    $\underline{\alpha} := (\underline{\gamma} - \underline{q})\, \underline{\psi}[\mu(\underline{k})]$
    $\underline{\beta_1} := \sqrt{\frac{2}{\varepsilon^2}}\, \mathbf{Q}^{-1} \cdot \underline{\alpha}$
    $\underline{\beta_2} := \mathbf{Q}^{-1} \overline{\mathbf{Q}} \cdot \underline{\xi}$
    $\underline{\beta} := \underline{\beta_1} - \underline{\beta_2}$
    // Store the results at the correct positions (forward neighbours)
    for $d = 0$ **to** $d = D - 1$ **do**
      $\underline{k}' := \underline{k} + \underline{e}^d$
      if $\underline{k}' \in \mathfrak{K}$ **then**
        $\underline{\psi}[\mu(\underline{k}')] = \frac{\underline{\beta}[d]}{\sqrt{\underline{k}[d]+1}}$
      end if
    end for
  end for
  // Whether to include the problematic prefactor or not
  // Make sure not to divide twice for $\phi_{\underline{0}}$!
  if prefactor is True **then**
    $\underline{\psi} = \frac{1}{\sqrt{\det \mathbf{Q}}}\, \underline{\psi}$
  end if
  return $\underline{\psi}$

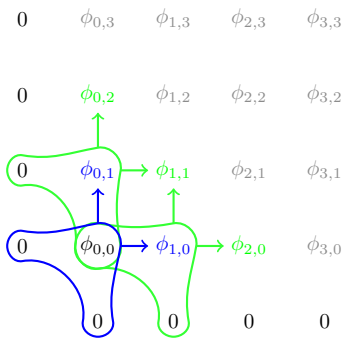
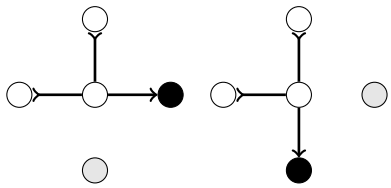
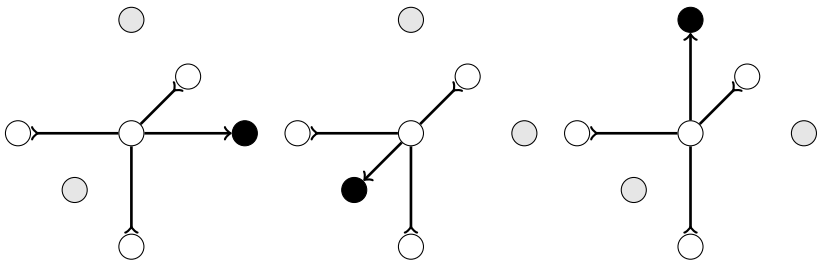


Figure 3.13: Naive stencil application in two dimensions. All values we initially know are printed in black. In a first step we centre the stencil $\mathcal{S}^2$ at the node $\underline{k} = (0,0)$ (blue) and compute $\phi_{0,1}$ and $\phi_{1,0}$. Next we can apply the stencil there (green) and find $\phi_{0,2}$, $\phi_{1,1}$ and $\phi_{2,0}$.



(a) $\mathcal{S}_0^2$ (b) $\mathcal{S}_1^2$

Figure 3.14: The modified basis function recursion stencils $\mathcal{S}_d^2$ in two dimensions. The black node can be computed given the white nodes. The grey node would be computed by the full stencil $\mathcal{S}^2$ but is intentionally left out by the modified ones.



(a) $\mathcal{S}_0^3$ (b) $\mathcal{S}_1^3$ (c) $\mathcal{S}_2^3$

Figure 3.15: The modified basis function recursion stencils $\mathcal{S}_d^3$ in three dimensions. The black node can be computed given the white nodes. The grey nodes would be computed by the full stencil $\mathcal{S}^3$ but is intentionally left out by the modified ones.

formula for the modified stencils $\mathcal{S}_d^D$ as follows:

$$\sqrt{k_d+1}\phi_{k+e_d} = \sqrt{\frac{2}{\varepsilon^2}}\left(Q^{-1}\right)_{d,:}(x-q)\,\phi_k - \left(Q^{-1}\overline{Q}\right)_{d,:}\begin{pmatrix}\sqrt{k_0}\phi_{k-e_0}\\ \vdots \\ \sqrt{k_{D-1}}\phi_{k-e_{D-1}}\end{pmatrix}. \tag{3.58}$$

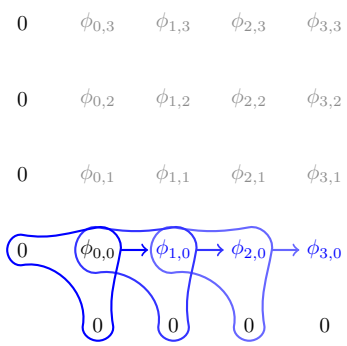


Figure 3.16: First step in the efficient application of the recursion stencils in two dimensions. All values we initially know are printed in black. In a first step we centre the cheap stencil $\mathcal{S}_0^2$ at the node $\underline{k} = (0, 0)$ and compute $\phi_{1,0}$ only. Next we can use the same stencil centred at $\phi_{1,0}$ and find $\phi_{2,0}$. We continue like this until an overflow occurs. (In this little example we are done after the next step.)

What can we do with these modified stencils? And why should they be more efficient? The most important gain is that we do never compute any function twice. But for this we have to use the new stencils in a clever way. We start with an empty grid like the one in figure 3.13. It contains only the ground state evaluated $\phi_{\underline{0}}$. Now we apply the stencil $\mathcal{S}_0^D$ along the first direction $d = 0$ until we overflow. This works fine as we use the ground state as anchor point for this *recursion chain* and produce new anchor points successively. This process is shown in figure 3.16.
After this first step we can not apply the stencil $\mathcal{S}_0^2$ anymore. There is no anchor point left that would give us new nodes in the lattice. But we have build a whole chain $\phi_{k,0}$ of starting points. Hence we take the stencil $\mathcal{S}_1^2$ at hand and go along the second direction $d = 1$ starting a chain for each $\phi_{k,0}$. Again we follow each chain until an overflow occurs. Note that we must do this in increasing order of the index k because we need the values to the left of where we centre $\mathcal{S}_1^2$. Figure 3.17 shows the process for the first two chains starting at $\phi_{0,0}$ and $\phi_{1,0}$.
For a three-index recursion evaluating all the functions ϕ_{k_0,k_1,k_2} we would start with a chain along the first dimension computing all $\phi_{k_0,0,0}$. Then we build new chains along the second direction starting one at each of the nodes $\phi_{k_0,0,0}$ and computing all $\phi_{k_0,k_1,0}$. And finally we build chains starting at the functions $\phi_{k_0,k_1,0}$ which gives us all the remaining functions ϕ_{k_0,k_1,k_2}. If we do this correctly we never compute any function more than once and nonetheless get all functions evaluated. Since we are building chains starting at some node $\underline{k}$ going along a given direction d we call this procedure *chain building*. And we need a very specialised way of iterating over all $\underline{k} \in \mathfrak{K}$ which we call *chain mode iteration* (compared to, for example, *lexicographical iteration*).

Definition 12 (Chain mode basis shape iterator)**.** *An iterator $\mathcal{I}$ is just a list of multi-indices with a well specified order. By iteration over this list we retrieve the contained values in this fixed sequence. Given a basis shape $\mathfrak{K}$ in D dimensions we can obtain D different chain mode iterators $\mathcal{I}_d^D$ for $d \in [0, \ldots, D-1]$. Any iterator is tightly related to the basis shape it belongs to. For that reason we sometimes write $\mathcal{I}_d^D[\mathfrak{K}]$ to make this important connection absolutely manifest.*

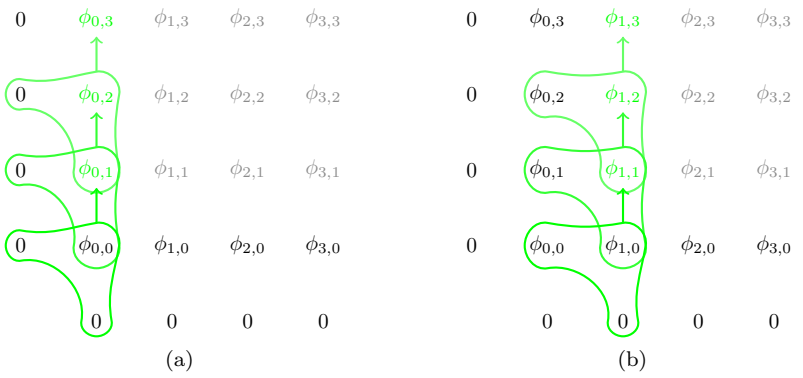


Figure 3.17: Second step in the efficient application of the recursion stencils in two dimensions. All values we initially know are printed in black. In a first step we centre the cheap stencil $\mathcal{S}_1^2$ at the node $\underline{k} = (0,0)$ and compute $\phi_{0,1}$ only. Next we can use the same stencil centred at $\phi_{0,1}$ and find $\phi_{0,2}$. We continue like this until an overflow occurs. (In this little example we are done after the next step.) Now we can start the second chain at the anchor $(1,0)$ and recursively compute $\phi_{1,1}$, $\phi_{1,2}$, $\phi_{1,3}$ until the overflow occurs. In a next step (not shown here) we start a chain at $(2,0)$, then at $(3,0)$ and so forth until the last node $(k,0)$.

Clearly the intersection $\mathcal{I}_d \cap \mathcal{I}_{d'}$ is never empty since all iterators $\mathcal{I}_d$ contain the multi-index $\underline{0}$ as starting point.

Referring back to the figures 3.16 and 3.17 we computed all the $\phi_{\underline{k}}$ for $\underline{k} \in \mathcal{I}_0^2$ and $\underline{k} \in \mathcal{I}_1^2$ respectively. The iterator $\mathcal{I}_0^2$ there yields the values $(0,0)$, $(1,0)$ and $(2,0)$ in this order and then gets exhausted. The next higher iterator $\mathcal{I}_1^2$ gives the values $(0,0)$, $(0,1)$, $(0,2)$, $(1,0)$, $(1,1)$, $(1,2)$, $\ldots$, $(3,0)$, $(3,1)$, $(3,2)$ before it gets exhausted too.
Going back to the general D dimensional case and an arbitrary basis shape $\mathfrak{K}$ we recognise that each iterator $\mathcal{I}_d^D[\mathfrak{K}]$ exactly yields the nodes $\underline{k} \in \mathfrak{K}$ where we have to apply the modified stencil $\mathcal{S}_d^D$. This is the quintessence of this whole idea! Now we are ready to formulate the algorithm 6 for efficient basis evaluation.
Note that all algorithms in this section are quite general and do not depend on the details of the basis shape $\mathfrak{K}$. We only assume that $\mathfrak{K}$ provides us with enough information about itself. We use the size $|\mathfrak{K}|$ and we have to get an answer for the question if $\underline{k} \in \mathfrak{K}$. We need the linearisation mapping $\mu_{\mathfrak{K}}$ and we have to be able to obtain chain mode iterators $\mathcal{I}_d^D$ for $\mathfrak{K}$. Any reasonable basis shape implements these functions without much trouble.

3.6.1 Number of stencil applications

We want to count how many times we apply the stencils. Assume we work in a hypercubic basis shape $\mathfrak{K}$ where the limits are given as $K = [K_0, \ldots, K_{D-1}]$.
For the full stencil $\mathcal{S}^D$ this is easy. We have to apply it maximally $\prod_{d=0}^{D-1} K_d$

Algorithm 6 Evaluate higher order states ϕ_k recursively (efficient version)

Require: The number D of space dimensions
Require: The Hagedorn parameter set $\Pi = \{q, p, Q, P\}$
Require: The semi-classical scaling parameter ε
Require: The basis shape $\mathfrak{K}$ and its linearisation mapping μ
Require: The grid node $\underline{\gamma} \in \mathbb{R}^D$
  // Storage space for the result
  $\underline{\psi} := \underline{0} \in \mathbb{C}^{|\mathfrak{K}|}$
  // Evaluate the ground state by algorithm 4
  $\underline{\psi}[\mu(\underline{0})] =$ **evaluate_ground_state**$(D, \Pi, \varepsilon, \underline{\gamma})$
  // Loop over the directions
  for $d = 0$ **to** $d = D - 1$ **do**
    // Get the iterator
    $\mathcal{I}_d :=$ **chain_mode_iterator**$(\mathfrak{K}, d)$
    // Start the chain building process
    for $\underline{k} \in \mathcal{I}_d$ **do**
      // Backward neighbours
      $\underline{\xi} := \underline{0} \in \mathbb{C}^D$
      for $d' = 0$ **to** $d' = D - 1$ **do**
        $\underline{k}' := \underline{k} - \underline{e}^{d'}$
        if $\underline{k}' \in \mathfrak{K}$ **then**
          $\underline{\xi}[d'] = \sqrt{\underline{k}[d']}\, \underline{\psi}[\mu(\underline{k}')]$
        end if
      end for
      // Compute 3-term recursion
      $\underline{\alpha} := (\underline{\gamma} - \underline{q})\, \underline{\psi}[\mu(\underline{k})]$
      $\beta_1 := \sqrt{\frac{2}{\varepsilon^2}}\, (\mathbf{Q}^{-1})[d, :] \cdot \underline{\alpha}$
      $\beta_2 := (\mathbf{Q}^{-1}\overline{\mathbf{Q}})[d, :] \cdot \underline{\xi}$
      // Store the result in correct position (forward neighbour in direction d)
      $\underline{k}' := \underline{k} + \underline{e}^d$
      if $\underline{k}' \in \mathfrak{K}$ **then**
        $\underline{\psi}[\mu(\underline{k}')] = \frac{\beta_1 - \beta_2}{\sqrt{\underline{k}[d]+1}}$
      end if
    end for
  end for
  // Whether to include the problematic prefactor or not
  // Make sure not to divide twice for $\phi_{\underline{0}}$!
  if prefactor is True **then**
    $\underline{\psi} = \frac{1}{\sqrt{\det \mathbf{Q}}}\, \underline{\psi}$
  end if
  return $\underline{\psi}$

Algorithm 7 Evaluate the whole basis set $\{\phi_k\}_{k\in\mathfrak{K}}$ on a grid Γ

Require: The basis shape $\mathfrak{K}$
Require: The grid Γ containing the nodes $\underline{\gamma} \in \mathbb{R}^D$
  // Storage space for the result
  $\mathbf{B} := \mathbf{0} \in \mathbb{C}^{|\mathfrak{K}|\times|\Gamma|}$
  // Evaluate the basis functions $\{\phi_k\}_{k\in\mathfrak{K}}$ for all grid nodes by algorithm 6
  // A real implementation of course exploits vectorisation.
  for $i = 0$ **to** $i = |\Gamma| - 1$ **do**
    $\mathbf{B}[:, i] =$ **evaluate_higher_order_states**(γ_i)
  end for
  return $\mathbf{B}$

times. And a more careful analysis shows that we have to apply it exactly:

$$\prod_{d=0}^{D-1} (K_d - 1) + 1 \tag{3.59}$$

times where the last 1 is to get the node $\phi_{\underline{K}-\underline{1}}$. This is only necessary for $D > 1$. For the modified stencils this gets more complicated. The number of applications of each individual stencil is given in the table below.

Stencil	Number of applications
$\mathcal{S}_0^D$	$K_0 - 1$
$\mathcal{S}_1^D$	$K_0(K_1 - 1)$
$\mathcal{S}_2^D$	$K_0K_1(K_2 - 1)$
$\vdots$	
$\mathcal{S}_d^D$	$\prod_{i=0}^{d-1} K_i(K_d - 1)$
$\vdots$	
$\mathcal{S}_{D-1}^D$	$\prod_{i=0}^{D-2} K_i(K_{D-1} - 1)$

Now the overall number of stencil applications is then:

$$\begin{aligned}
\sum_{d=0}^{D-1}\prod_{i=0}^{d-1} K_i(K_d - 1) &= \sum_{d=0}^{D-1}\left((K_d - 1)\prod_{i=0}^{d-1} K_i\right) = \sum_{d=0}^{D-1}\left(\prod_{i=0}^{d} K_i - \prod_{i=0}^{d-1} K_i\right) \\
&= \sum_{d=0}^{D-1}\prod_{i=0}^{d} K_i - \sum_{d=0}^{D-1}\prod_{i=0}^{d-1} K_i \\
&= \prod_{i=0}^{D-1} K_i - \prod_{i=0}^{-1} K_i = \prod_{i=0}^{D-1} K_i - 1
\end{aligned}$$

where the sum is resolved via telescoping and we define the empty product as the multiplicative identity. Of course it holds that:

$$\prod_{d=0}^{D-1} (K_d - 1) + 1 \leq \prod_{d=0}^{D-1} K_d - 1$$

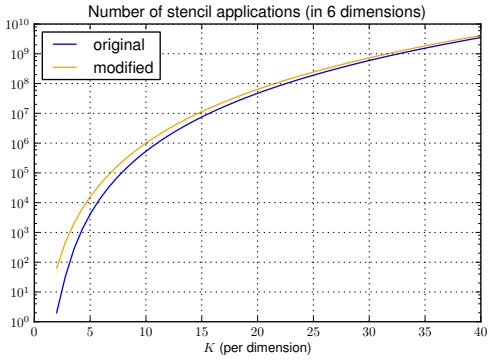


Figure 3.18: Total number of stencil applications for a hypercubic basis shape in $D = 6$ dimensions and with K nodes along each direction.

for large enough K and D. The consequence is that we perform *more* applications when using the modified stencil. But it will turn out that this is not a bad fact. An example in 6 dimensions is shown in figure 3.18.

3.6.2 Cost of a stencil application

We define the cost of a stencil application $\mathcal{S}\phi_{\underline{k}}$ as the number of multiplications which involve a ϕ. The reason is that we will evaluate $\phi_{\underline{k}}$ on large grids with many nodes hence $\phi_{\underline{k}}$ is a very long vector.
During the application of the full stencil $\mathcal{S}^D$ we multiply a D vector by ϕ in the first term. In the second one we multiply a $D \times D$ matrix by a vector of size D containing a (different) ϕ in each element. Hence the application $\mathcal{S}^D\phi_k$ has a total cost of $D^2 + D$.
For the modified stencil we only evaluate the first element of the vector equation and therefore we compute the product of a scalar with ϕ and form an inner product between two vectors of length D, one containing a (different) ϕ in each element. The application of any modified stencil $\mathcal{S}_d^D$ has therefore a total cost of $D + 1$ which is linear in the number of dimensions. Figure 3.19 shows the cost of both stencils as a function of dimension D.
If we now compare both evaluation schemes by the number of stencil applications and the costs of a single application it turns out that the scheme using modified stencils is much more efficient! In formal notation we find:

$$\left(\prod_{d=0}^{D-1}(K_d - 1) + 1\right)(D^2 + D) < \left(\prod_{d=0}^{D-1} K_d - 1\right)(D + 1)$$

for large enough K where the cross over point depends on the dimension D. For $D = 1$ both schemes are equivalent and have the same costs. Figure 3.20 shows the overall costs for an hypercubic basis shape in $D = 6$ dimensions.

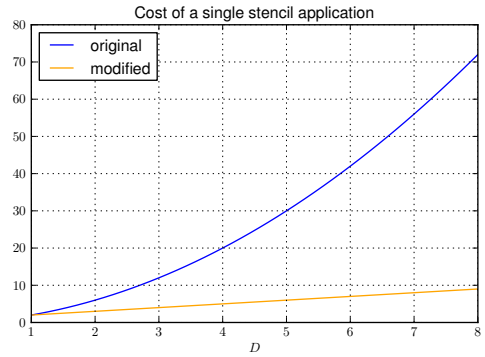


Figure 3.19: The costs of both stencil types in several dimensions.

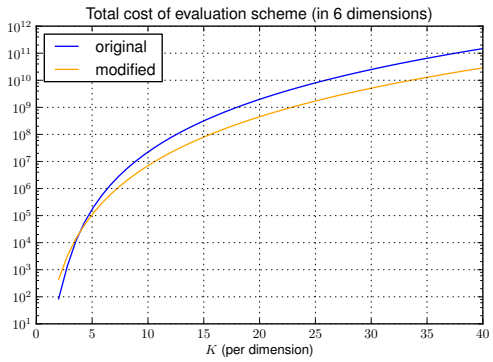


Figure 3.20: Total cost of a basis evaluation for a hypercubic basis shape in $D = 6$ dimensions and with K nodes along each direction.

3.7 Vector-valued wavepackets

For the purpose of solving the time-dependent Schrödinger equation with non-adiabatic potentials the scalar wavepackets are not enough. Recall that the potential has N different energy levels. The wavefunction $|\varphi\rangle$ needs therefore N components stacked into a vector. In the following we construct *vectorial wavepackets* where each component is given by a scalar wavepacket. Formally we are in this situation:

$$\Psi(\underline{x}, t) := \begin{pmatrix} \Phi_0(\underline{x}, t) \\ \vdots \\ \Phi_{N-1}(\underline{x}, t) \end{pmatrix} . \tag{3.60}$$

How many free input parameters does this object have? Each component Φ_i needs a set Π_i of Hagedorn parameters. (We should write $\Phi_i[\Pi_i]$ but this becomes lengthy so we drop this soon.) If we use the very same parameter set Π for each component we arrive at what we call a *homogeneous wavepacket*.

Definition 13 (Homogeneous vectorial wavepacket)**.**

$$|\Psi\rangle := \Psi[\Pi](\underline{x}, t) = \begin{pmatrix} \Phi_0[\Pi](\underline{x}, t) \\ \vdots \\ \Phi_{N-1}[\Pi](\underline{x}, t) \end{pmatrix} \tag{3.61}$$

where $\Pi_i \equiv \Pi_j \equiv \Pi \, \forall \, i, j$

However, at any fixed time each component is independent from all other ones. For this reason we can choose possibly different sets Π_i of parameters for all N components. By doing this we get an *inhomogeneous wavepacket*, formally defined as:

Definition 14 (Inhomogeneous vectorial wavepacket)**.**

$$|\Psi\rangle := \Psi[\Pi_0, \ldots, \Pi_{N-1}](\underline{x}, t) = \begin{pmatrix} \Phi_0[\Pi_0](\underline{x}, t) \\ \vdots \\ \Phi_{N-1}[\Pi_{N-1}](\underline{x}, t) \end{pmatrix} \tag{3.62}$$

where $\Pi_i \neq \Pi_j$ *is possible.*

In some applications, for example the spawning approach applied to tunneling [6] and the non-adiabatic case [2], it perfectly makes sense to expand every component into a different basis of $L^2(\mathbb{R}^D)$. And since each basis $\phi_{\underline{k}}$ is essentially fully determined by the parameter set Π of its basis functions $\phi_{\underline{k}}$, this results in different parameter sets Π_i for each component Φ_i.
For some computations we find a more explicit representation of Ψ to be of greater use. The above definitions can be trivially rewritten as follows by introducing the unit vectors $\underline{e}^j$ of the canonical basis of $\mathbb{R}^N$. In the homogeneous case we get:

$$\Psi[\Pi](\underline{x}, t) = \sum_{n=0}^{N-1} \underline{e}^n \Phi_n[\Pi](\underline{x}, t) \tag{3.63}$$

and in the inhomogeneous one:

$$\Psi\left[\Pi_0, \ldots, \Pi_{N-1}\right](\underline{x}, t) = \sum_{n=0}^{N-1} \underline{e}^n \Phi_n\left[\Pi_n\right](\underline{x}, t)\,. \tag{3.64}$$

We should also note that each component can have an individual basis shape $\mathfrak{K}_n$. But from now on we will not mention this too often but implicitly assume that each scalar wavepacket uses the currently best basis shape whatever that means.

3.8 Gradient computation

In this section we want to compute the gradient of a scalar wavepacket Φ. More precisely we find the result of the operator application $-i\varepsilon^2\nabla\Phi$. From now on we define the short hand notation $y := -i\varepsilon^2\nabla$. We would like to have an explicit representation of y. Of course it includes just the ordinary gradient differential operator. But we need a more involved representation allowing us to easily compute the application of y to a wavepacket of the form given in (3.46). Luckily there is a term of precisely the form of y included in the definition of the ladder operators in equation (3.29). We proceed by solving the linear system consisting of the two operator definitions of $\mathcal{L}$ and $\mathcal{R}$ for y. We begin by transforming the definition of $\mathcal{R}$ as follows (with the usual definition of θ):

$$\begin{aligned}
\mathcal{R} &= \frac{i}{\sqrt{2\varepsilon^2}}\left(\mathbf{P}^{\mathrm{H}}(\underline{x} - \underline{q}) - \mathbf{Q}^{\mathrm{H}}(\underline{y} - \underline{p})\right) \\
\mathcal{R} &= \theta\mathbf{P}^{\mathrm{H}}(\underline{x} - \underline{q}) - \theta\mathbf{Q}^{\mathrm{H}}(\underline{y} - \underline{p}) \\
\theta\mathbf{Q}^{\mathrm{H}}(\underline{y} - \underline{p}) &= -\mathcal{R} + \theta\mathbf{P}^{\mathrm{H}}(\underline{x} - \underline{q}) \\
\mathbf{Q}^{\mathrm{H}}(\underline{y} - \underline{p}) &= -\frac{1}{\theta}\mathcal{R} + \mathbf{P}^{\mathrm{H}}(\underline{x} - \underline{q})\,.
\end{aligned}$$

Before we proceed in solving this for y we transform the definition of $\mathcal{L}$ such that we can replace the term $(\underline{x} - \underline{q})$:

$$\begin{aligned}
\mathcal{L} &= -\theta\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) + \theta\mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p}) \\
\theta\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) &= -\mathcal{L} + \theta\mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p}) \\
\mathbf{P}^{\mathrm{T}}(\underline{x} - \underline{q}) &= -\frac{1}{\theta}\mathcal{L} + \mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p}) \\
\underline{x} - \underline{q} &= -\frac{1}{\theta}\mathbf{P}^{\text{-T}}\mathcal{L} + \mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}(\underline{y} - \underline{p})\,.
\end{aligned}$$

Then we can plug this into the above equation:

$$
\begin{aligned}
\mathbf{Q}^{\mathrm{H}}(\underline{y}-\underline{p}) &= -\frac{1}{\theta}\mathcal{R}+\mathbf{P}^{\mathrm{H}}\left(-\frac{1}{\theta}\mathbf{P}^{\text{-T}}\mathcal{L}+\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}(\underline{y}-\underline{p})\right)\\
\mathbf{Q}^{\mathrm{H}}(\underline{y}-\underline{p}) &= -\frac{1}{\theta}\mathcal{R}+-\frac{1}{\theta}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}(\underline{y}-\underline{p})\\
\mathbf{Q}^{\mathrm{H}}\underline{y}-\mathbf{Q}^{\mathrm{H}}\underline{p} &= -\frac{1}{\theta}\mathcal{R}-\frac{1}{\theta}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\underline{y}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\underline{p}\\
\mathbf{Q}^{\mathrm{H}}\underline{y}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\underline{y} &= -\frac{1}{\theta}\mathcal{R}-\frac{1}{\theta}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+\mathbf{Q}^{\mathrm{H}}\underline{p}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\underline{p}\\
\underbrace{\left(\mathbf{Q}^{\mathrm{H}}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\right)}_{**}\underline{y} &= -\frac{1}{\theta}\mathcal{R}-\frac{1}{\theta}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+\left(\mathbf{Q}^{\mathrm{H}}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\right)\underline{p}\,.
\end{aligned}
$$

Our next task is the computation of the subexpression $**$. For this we start again with the basis relations from (3.11). For the first one we get:

$$
\begin{aligned}
\mathbf{Q}^{\mathrm{H}}\mathbf{P}-\mathbf{P}^{\mathrm{H}}\mathbf{Q} &= 2i\mathbb{1}\\
\mathbf{Q}^{\mathrm{H}}-\mathbf{P}^{\mathrm{H}}\mathbf{Q}\mathbf{P}^{-1} &= 2i\mathbf{P}^{-1}
\end{aligned}
$$

and from the second one:

$$
\begin{aligned}
\mathbf{P}^{\mathrm{T}}\mathbf{Q}-\mathbf{Q}^{\mathrm{T}}\mathbf{P} &= 0\\
\mathbf{Q}-\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\mathbf{P} &= 0\\
\mathbf{Q} &= \mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\mathbf{P}\,.
\end{aligned}
$$

Combining these two results (replacing the second $\mathbf{Q}$) we obtain:

$$
\begin{aligned}
\mathbf{Q}^{\mathrm{H}}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}}\mathbf{P}\mathbf{P}^{-1} &= 2i\mathbf{P}^{-1}\\
\mathbf{Q}^{\mathrm{H}}-\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathbf{Q}^{\mathrm{T}} &= 2i\mathbf{P}^{-1}\,.
\end{aligned}
$$

This last line can then be used as a substitution for $**$ above:

$$
\begin{aligned}
2i\mathbf{P}^{-1}\underline{y} &= -\frac{1}{\theta}\mathcal{R}-\frac{1}{\theta}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+2i\mathbf{P}^{-1}\underline{p}\\
\mathbf{P}^{-1}\underline{y} &= -\frac{1}{2i\theta}\mathcal{R}-\frac{1}{2i\theta}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+\mathbf{P}^{-1}\underline{p}\\
\underline{y} &= -\frac{1}{2i\theta}\mathbf{P}\mathcal{R}-\frac{1}{2i\theta}\mathbf{P}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{\text{-T}}\mathcal{L}+\underline{p}\,.
\end{aligned}
$$

Finally cleaning up and undoing the introduction of θ we arrive at:

$$
\underline{y} = \sqrt{\frac{\varepsilon^2}{2}}\left(\mathbf{P}\mathcal{R}+\overline{\mathbf{P}}\mathcal{L}\right)+\underline{p}\,. \tag{3.65}
$$

If we had begun by using the operators $\mathcal{L}$ and $\mathcal{R}$ in the opposite order we would have got the complex conjugate equation:

$$
\underline{y} = \sqrt{\frac{\varepsilon^2}{2}}\left(\overline{\mathbf{P}}\mathcal{L}+\mathbf{P}\mathcal{R}\right)+\underline{p}\,. \tag{3.66}
$$

We can simplify the matrix product $\mathbf{P}\mathbf{P}^{\mathrm{H}}\mathbf{P}^{-\mathrm{T}}$ to obtain $\overline{\mathbf{P}}$. The reason is the conditions (3.11) which must be fulfilled by $\mathbf{P}$ and $\mathbf{Q}$, see [8].

3.8.1 Applying the gradient operator

With this explicit representation of the y operator in terms of raising and lowering operators we can now study its application to an arbitrary basis function $\phi_{\underline{k}}$. In the end we then need to apply y to the whole scalar wavepacket Φ in order to find its kinetic energy.
What do we have to expect when applying y to a basis function $\phi_{\underline{k}}$? First, we know that the gradient is defined as usual as:

$$\nabla_{\underline{x}} := \begin{pmatrix} \frac{\partial}{\partial x_0} \\ \vdots \\ \frac{\partial}{\partial x_{D-1}} \end{pmatrix}.$$

Hence we have to expect that the gradient applied to $\phi_{\underline{k}} : \mathbb{R}^D \to \mathbb{C}$ is a vector with D components. Next we conclude from (3.28) that the ladder operators applied to a function give another vector of same shape. We wish to compute:

$$y\phi_{\underline{k}}(\underline{x}) = \sqrt{\frac{\varepsilon^2}{2}} \left(\mathbf{P}\mathcal{R} + \overline{\mathbf{P}}\mathcal{L}\right) \phi_{\underline{k}}(\underline{x}) + \underline{p}\phi_{\underline{k}}(\underline{x})$$

and if we carry out the application of the ladder operators we get step by step the following relation for the gradient of a single basis function:

$$y\phi_{\underline{k}}(\underline{x}) = \sqrt{\frac{\varepsilon^2}{2}} \left(\mathbf{P} \begin{pmatrix} \mathcal{R}_0 \\ \vdots \\ \mathcal{R}_{D-1} \end{pmatrix} + \overline{\mathbf{P}} \begin{pmatrix} \mathcal{L}_0 \\ \vdots \\ \mathcal{L}_{D-1} \end{pmatrix} \right) \phi_{\underline{k}}(\underline{x}) + \underline{p}\phi_{\underline{k}}(\underline{x})$$

where we apply the ladder operators now:

$$y\phi_{\underline{k}}(\underline{x}) = \sqrt{\frac{\varepsilon^2}{2}} \left(\mathbf{P} \begin{pmatrix} \sqrt{k_0+1}\phi_{\underline{k}+\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1}+1}\phi_{\underline{k}+\underline{e}^{D-1}} \end{pmatrix} + \overline{\mathbf{P}} \begin{pmatrix} \sqrt{k_0}\phi_{\underline{k}-\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1}}\phi_{\underline{k}-\underline{e}^{D-1}} \end{pmatrix} \right) + \underline{p}\phi_{\underline{k}}\,. \tag{3.67}$$

This is just for one single basis function. But we need more. Thus we compute the gradient of a whole scalar wavepacket $\Phi = \sum_{\underline{k}\in\mathfrak{K}} c_{\underline{k}}\phi_{\underline{k}}$:

$$y\Phi = \sum_{\underline{k}\in\mathfrak{K}} y\, c_{\underline{k}}\phi_{\underline{k}} = \sum_{\underline{k}\in\mathfrak{K}} c_{\underline{k}}\, y\phi_{\underline{k}}$$

For a shorter notation we define the following variables:

$$\theta := \sqrt{\frac{\varepsilon^2}{2}}$$
$$\underline{\alpha}(\phi_{\underline{k}}) := \begin{pmatrix} \sqrt{k_0+1}\phi_{\underline{k}+\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1}+1}\phi_{\underline{k}+\underline{e}^{D-1}} \end{pmatrix}$$
$$\underline{\beta}(\phi_{\underline{k}}) := \begin{pmatrix} \sqrt{k_0}\phi_{\underline{k}-\underline{e}^0} \\ \vdots \\ \sqrt{k_{D-1}}\phi_{\underline{k}-\underline{e}^{D-1}} \end{pmatrix}.$$

Continuing we derive:

$$\begin{aligned} &= \sum_{\underline{k}\in\mathfrak{K}} \left(c_{\underline{k}} \underline{p}\phi_{\underline{k}} + \theta c_{\underline{k}} \mathbf{P}\underline{\alpha}(\phi_{\underline{k}}) + \theta c_{\underline{k}} \overline{\mathbf{P}}\underline{\beta}(\phi_{\underline{k}}) \right) \\ &= \sum_{\underline{k}\in\mathfrak{K}} c_{\underline{k}} \underline{p}\phi_{\underline{k}} + \sum_{\underline{k}\in\mathfrak{K}} \theta c_{\underline{k}} \mathbf{P}\underline{\alpha}(\phi_{\underline{k}}) + \sum_{\underline{k}\in\mathfrak{K}} \theta c_{\underline{k}} \overline{\mathbf{P}}\underline{\beta}(\phi_{\underline{k}}) \,. \end{aligned}$$

It becomes obvious that $y\phi_{\underline{k}}$ has contributions from all neighbours $\phi_{\underline{k}+\underline{e}^d}$ and $\phi_{\underline{k}-\underline{e}^d}$ for all $d \in [0, \ldots, D-1]$ and from $\phi_{\underline{k}}$. This immediately raises the question of an efficient computation. In the end we will need $y\phi_{\underline{k}}$ for all $\underline{k} \in \mathfrak{K}$. Since there are ladder operators involved we have to be careful for all $\underline{k}$ on the border of $\mathfrak{K}$ who don't have a full set of neighbours in all directions d. For each neighbour $\underline{k} \pm \underline{e}^d$ that is not part of the basis shape $\mathfrak{K}$ we have to decide what to do. And the rules are as follows.
If the vector $\underline{k} - \underline{e}^d$ has negative components, then we can take the corresponding basis function to be equivalent zero. We call this situation an *underflow* [2].
The other case is more involved. The functions $\phi_{\underline{k}}$ clearly never vanish identically zero. However we only have finite linear combinations Φ of basis functions $\phi_{\underline{k}}$. Hence all coefficients for sufficiently *large* $\underline{k}$ vanish. And finally we are only interested in the new coefficients $c'^i_{\underline{k}}$ of each component i of the gradient $y\phi_{\underline{k}}$. For this reason we can also insert zeros in the case such an *overflow* occurs. For an example of the situation and the issues that may arise refer to figure 3.21.
Before we continue let's make a not so small but very simple example. Assume the hypercubic basis shape in two dimensions is $\mathfrak{K} = \underline{K} = (3, 3)$. The wavepacket consists of the following linear combination:

$$\begin{aligned} \Phi := &\, c_{0,0}\phi_{0,0} + c_{1,0}\phi_{1,0} + c_{2,0}\phi_{2,0} \\ &+ c_{0,1}\phi_{0,1} + c_{1,1}\phi_{1,1} + c_{2,1}\phi_{2,1} \\ &+ c_{0,2}\phi_{0,2} + c_{1,2}\phi_{1,2} + c_{2,2}\phi_{2,2} \,. \end{aligned}$$

Computing the gradients individually for some of the terms we get:

[2] This has nothing to do with the usual arithmetic over/underflow. It just serves as a notation of *where* we access elements outside of our basis shape $\mathfrak{K}$.

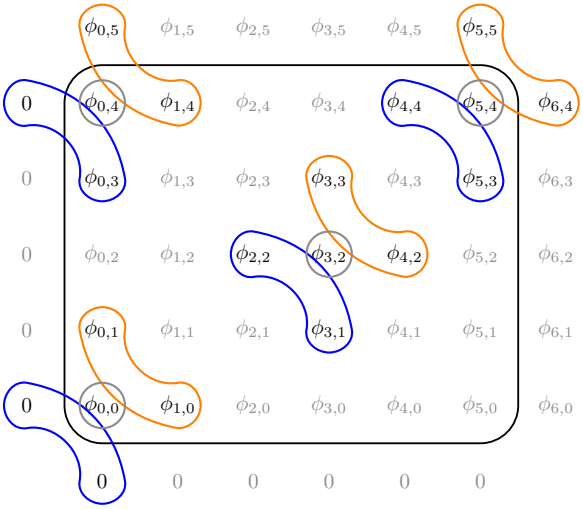


Figure 3.21: Stencil application for computing the gradient $y\phi_{\underline{k}}$ for different $\underline{k}$ in two dimensions. For each computation we need the function itself (grey circles), the backward neighbours $\phi_{\underline{k}-\underline{e}^d}$ (blue contours) and the forward neighbours $\phi_{\underline{k}+\underline{e}^d}$ (orange contours). Clearly we need additional functions outside of the basis shape $\mathfrak{K}$ (black rectangle). In all corners and along all sides we under- and overflow respectively.

Multi-index node	Gradient
$\underline{k} = (0,0)$	$c_{0,0}\,\underline{p}\phi_{0,0} + \theta c_{0,0}\,\mathbf{P}\begin{pmatrix}\sqrt{1}\,\phi_{1,0}\\ \sqrt{1}\,\phi_{0,1}\end{pmatrix} + \theta c_{0,0}\,\overline{\mathbf{P}}\begin{pmatrix}\sqrt{0}\,\phi_{-1,0}\\ \sqrt{0}\,\phi_{0,-1}\end{pmatrix}$
$\underline{k} = (1,0)$	$c_{1,0}\,\underline{p}\phi_{1,0} + \theta c_{1,0}\,\mathbf{P}\begin{pmatrix}\sqrt{2}\,\phi_{2,0}\\ \sqrt{1}\,\phi_{1,1}\end{pmatrix} + \theta c_{1,0}\,\overline{\mathbf{P}}\begin{pmatrix}\sqrt{1}\,\phi_{0,0}\\ \sqrt{0}\,\phi_{1,-1}\end{pmatrix}$
$\underline{k} = (0,1)$	$c_{0,1}\,\underline{p}\phi_{0,1} + \theta c_{0,1}\,\mathbf{P}\begin{pmatrix}\sqrt{1}\,\phi_{1,1}\\ \sqrt{2}\,\phi_{0,2}\end{pmatrix} + \theta c_{0,1}\,\overline{\mathbf{P}}\begin{pmatrix}\sqrt{0}\,\phi_{-1,1}\\ \sqrt{1}\,\phi_{0,0}\end{pmatrix}$
$\underline{k} = (1,1)$	$c_{1,1}\,\underline{p}\phi_{1,1} + \theta c_{1,1}\,\mathbf{P}\begin{pmatrix}\sqrt{2}\,\phi_{2,1}\\ \sqrt{2}\,\phi_{1,2}\end{pmatrix} + \theta c_{1,1}\,\overline{\mathbf{P}}\begin{pmatrix}\sqrt{1}\,\phi_{0,1}\\ \sqrt{1}\,\phi_{1,0}\end{pmatrix}$
$\underline{k} = (2,1)$	$c_{2,1}\,\underline{p}\phi_{2,1} + \theta c_{2,1}\,\mathbf{P}\begin{pmatrix}\sqrt{3}\,\phi_{3,1}\\ \sqrt{2}\,\phi_{2,2}\end{pmatrix} + \theta c_{2,1}\,\overline{\mathbf{P}}\begin{pmatrix}\sqrt{2}\,\phi_{1,1}\\ \sqrt{1}\,\phi_{2,0}\end{pmatrix}$
$\underline{k} = (1,2)$	$c_{1,2}\,\underline{p}\phi_{1,2} + \theta c_{1,2}\,\mathbf{P}\begin{pmatrix}\sqrt{2}\,\phi_{2,2}\\ \sqrt{3}\,\phi_{1,3}\end{pmatrix} + \theta c_{1,2}\,\overline{\mathbf{P}}\begin{pmatrix}\sqrt{1}\,\phi_{0,2}\\ \sqrt{2}\,\phi_{1,1}\end{pmatrix}$
$\vdots$	$\ldots$

and we would sum up all these terms to get the overall gradient of Φ. In this form however it is not of much use to us. We want to represent the gradient again as linear combinations over the same set of basis functions:

$$y\Phi = \sum_{\underline{k}\in\overline{\mathfrak{K}}} \underline{c}'_{\underline{k}}\phi_{\underline{k}} \tag{3.68}$$

where $\underline{c}'_{\underline{k}} \in \mathbb{C}^D$ and $\overline{\mathfrak{K}}$ is the extended basis shape (in case of our example $\overline{\mathfrak{K}} = \underline{K}' = (4,4)$). This is indeed possible, expanding the fourth row from the

table above in instances of $\phi_{\underline{k}}$ and summing up we get:

$$c_{1,1}\,\underline{p}\phi_{1,1} + \theta c_{1,1}\,\sqrt{2}\,\mathbf{P}_{:,0}\phi_{2,1} + \theta c_{1,1}\,\sqrt{2}\,\mathbf{P}_{:,1}\phi_{1,2} + \theta c_{1,1}\,\overline{\mathbf{P}}_{:,0}\phi_{0,1} + \theta c_{1,1}\,\overline{\mathbf{P}}_{:,1}\phi_{1,0}\,.$$

In finding the coefficient $\underline{c}_{1,1}$ of $\phi_{1,1}$ we have to be very careful, because there are further terms on up to 4 other rows that contribute to $\underline{c}_{1,1}$. (Note that this coefficient is a good example as its lattice node has a full set of 4 neighbours.) Summing up all relevant terms we find that:

$$\underline{c}'_{1,1} = c_{1,1}\,\underline{p} + \theta c_{0,1}\,\mathbf{P}_{:,0}\,. + \theta c_{1,0}\,\mathbf{P}_{:,1} + \theta c_{2,1}\,\sqrt{2}\,\overline{\mathbf{P}}_{:,0} + \theta c_{1,2}\,\sqrt{2}\,\overline{\mathbf{P}}_{:,1}\,.$$

The last four terms can be recombined into one per pair and we end up with:

$$\underline{c}'_{1,1} = c_{1,1}\,\underline{p} + \theta\mathbf{P}\begin{pmatrix} c_{0,1} \\ c_{1,0} \end{pmatrix} + \theta\overline{\mathbf{P}}\begin{pmatrix} \sqrt{2}\,c_{2,1} \\ \sqrt{2}\,c_{1,2} \end{pmatrix}$$

Doing these transformations in a systematic way is the major issue in computing the gradients. With a bit of guessing we can find the following general rule for the coefficient vectors $\underline{c}'_{\underline{k}}$ for all $\underline{k} \in \overline{\mathfrak{K}}$:

$$\underline{c}'_{\underline{k}} = c_{\underline{k}}\,\underline{p} + \sqrt{\frac{\varepsilon^2}{2}}\sum_{d=0}^{D-1} c_{\underline{k}+\underline{e}^d}\,\sqrt{\underline{k}_d + 1}\,\overline{\mathbf{P}}_{:,d} + \sqrt{\frac{\varepsilon^2}{2}}\sum_{d=0}^{D-1} c_{\underline{k}-\underline{e}^d}\,\sqrt{\underline{k}_d}\,\mathbf{P}_{:,d}$$

which again can be cast in compact form:

$$\underline{c}'_{\underline{k}} = c_{\underline{k}}\,\underline{p} + \sqrt{\frac{\varepsilon^2}{2}}\left(\overline{\mathbf{P}}\begin{pmatrix} c_{\underline{k}+\underline{e}^0}\,\sqrt{\underline{k}_0 + 1} \\ \vdots \\ c_{\underline{k}+\underline{e}^{D-1}}\,\sqrt{\underline{k}_{D-1} + 1} \end{pmatrix} + \mathbf{P}\begin{pmatrix} c_{\underline{k}-\underline{e}^0}\,\sqrt{\underline{k}_0} \\ \vdots \\ c_{\underline{k}-\underline{e}^{D-1}}\,\sqrt{\underline{k}_{D-1}} \end{pmatrix}\right). \tag{3.69}$$

3.8.2 Gather-type algorithm

The most simple algorithm to compute $y\phi_{\underline{k}}$ for all $\underline{k} \in \mathfrak{K}$ has to iterate over all $\underline{k} \in \overline{\mathfrak{K}}$ and apply the two stencils to get the data from backward and forward neighbours of $\underline{k}$ and to compute $y\phi_{\underline{k}}$ by the formula (3.69). This is simple to understand and works reasonably efficient. We call this algorithm the *gather-type* stencil application.

The only not so easy point here is that we can not iterate just over $\mathfrak{K}$ but have to extend the basis shape due to the nature of the neighbourhood stencil. In more detail, this is necessary as there are $\underline{k}' \notin \mathfrak{K}$ for which the above formula (3.69) still yields a non-zero result since the stencil centred at $\underline{k}'$ still overlaps with the basis shape $\mathfrak{K}$. And we do not want to loose these contributions. Therefore we extend the basis shape $\mathfrak{K}$ by one node in all directions and get what we denote by $\overline{\mathfrak{K}}$. This should be clear by looking at figure 3.21.

In one single space dimension x the gradient equals the derivative and the result is not a vector of functions but just a function too. Therefore it is easier to understand the gradient computation for one-dimensional wavepackets $\Phi(x)$ first. The whole process of the gather-type algorithm is shown in figure 3.22 in considerable detail.

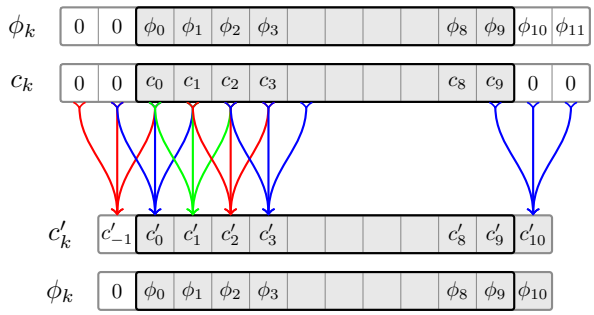


Figure 3.22: The gather-type stencil application for computing the gradient $y\Phi$. The upper two arrays show the initial linear combination Φ consisting of basis functions ϕ_k and coefficients c_k. The lower two arrays show the linear combination of $\frac{\mathrm{d}\Phi}{\mathrm{d}x}$ consisting of the same basis functions ϕ_k and new coefficients c'_k. Each of the triple arrows is one single stencil application (formula (3.69)), not all applications are shown. We see that the basis shape $\overline{\mathfrak{K}}$ for the gradient is larger. And also that we access several elements not part of the original basis shape $\mathfrak{K}$. During the computation we insert the zeros on the fly and we drop the coefficient c'_{-1} (because $\phi_{-1} \equiv 0$). The original basis shape $\mathfrak{K}$ is represented by the black rectangle and the actual basis shapes are shown shaded grey.

The next figure 3.23 shows the same algorithm but this time for a two-dimensional wavepacket. It should now be clear what happens. The principle is the same for an arbitrary number D of space dimensions and arbitrary basis shapes $\mathfrak{K}$.
The general procedure for D dimensions is shown in listing (8).

3.8.3 Scatter-type algorithm

An improved version of the algorithm for computing gradient coefficients $\underline{c_k}$ can be obtained if we use formula (3.67). And instead of iteration over the extended basis shape $\overline{\mathfrak{K}}$ we iterate over the original shape $\mathfrak{K}$. We use the formula mentioned and split it into its three parts. Each part is then used independently inside the algorithm. The main point is that we do not compute $\underline{c_k}$ at once but assemble it from these three pieces. For each $\underline{k} \in \mathfrak{K}$ we compute the contributions to the coefficients $\underline{c_{k \pm e^d}}$ of (3.68) for all $d \in [0, \ldots, D-1]$ independently.
The scatter-type algorithm is shown in figure 3.24 for $D = 1$ and in figure 3.25 for $D = 2$. The generic procedure is shown in algorithm 9.
Maybe we should make a little test run of this algorithm by hand to show how it works. The best way to understand it is to set up a little table. We work in $D = 1$ dimension to make things easier but the principle exactly applies also to any higher dimensional case.
Each row results from the application of formula (3.67) to a single function ϕ_k. We then arrange the terms depending on which $c'_{k'}$ they belong to and write the result into the correct column k'. After we finished all the rows we can sum along the column k' to get the full coefficient $c'_{k'}$.
In the following three tables we strip the common factor of $\sqrt{\frac{\varepsilon^2}{2}}$ from all off-diagonal entries. Additionally each row k should be multiplied by c_k.
Note that the two extra columns with captions $k = -1$ and $k = |\mathcal{K}|$ are not part

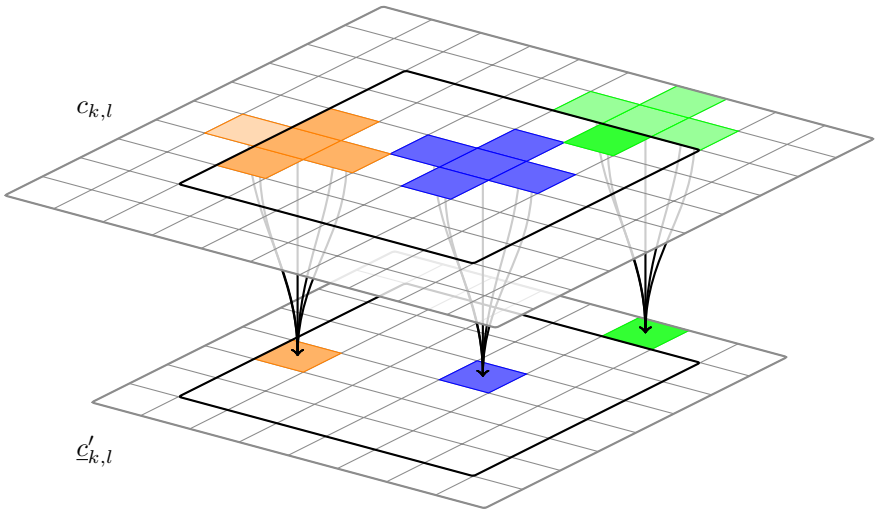


Figure 3.23: The gather-type stencil application for computing the gradient $y\Phi$. The upper array (plane) shows the coefficients $c_{\underline{k}}$ of the initial linear combination $\Phi(\underline{x})$. The lower array (plane) shows the coefficients $c'_{\underline{k}}$ of the linear combination of $\nabla\Phi(\underline{x})$ where each square is not a single number but stands for a whole vector. Each arrow bundle is one single stencil application (formula (3.69)), not all applications are shown. The original basis shape $\mathfrak{K}$ is given by the black rectangle. We then see that the basis shape $\overline{\mathfrak{K}}$ for the gradient is larger by one square on each side. (But we again drop all coefficients with negative indices.) The orange stencil shows that we sometimes have to access elements (coefficients) that are not part of the original linear combination. We can safely insert zeros there. The green stencil application shows that formula (3.69) produces (in general) non-zero values for indices $\underline{k} \notin \mathfrak{K}$. Therefore we need to extend the basis shape where the whole lower plane stands for $\overline{\mathfrak{K}}$.

Algorithm 8 Compute the gradient $y\Phi$ by gather-type stencil application

Require: Scalar wavepacket Φ in D space dimensions
Require: Basis shape $\mathfrak{K}$ (including linearisation mapping $\mu_{\mathfrak{K}}$) of Φ
Require: Parameters Π and coefficients $\{c_{\underline{k}}\}$ of Φ
  // Extend the basis shape $\mathfrak{K}$
  $\overline{\mathfrak{K}} :=$ **extend_basis_shape**$(\mathfrak{K})$
  // Storage space for the result
  $\mathbf{c}' = \mathbf{0} \in \mathbb{C}^{D \times |\overline{\mathfrak{K}}|}$
  // Iterate over extended basis
  for $\underline{k} \in \overline{\mathfrak{K}}$ **do**
    // Central node
    if $\underline{k} \in \mathfrak{K}$ **then**
      $c_{\mathrm{c}} = c_{\underline{k}}$
    else
      $c_{\mathrm{c}} = 0$
    end if
    // Backward neighbours
    $\underline{c}_{\mathrm{b}} = \underline{0} \in \mathbb{C}^D$
    for $d = 0$ **to** $d = D - 1$ **do**
      $\underline{k}' = \underline{k} - \underline{e}^d$
      if $\underline{k}' \in \mathfrak{K}$ **then**
        $\underline{c}_{\mathrm{b}}[d] = \sqrt{\underline{k}[d]}\, c_{\underline{k}'}$
      end if
    end for
    // Forward neighbours
    $\underline{c}_{\mathrm{f}} = \underline{0} \in \mathbb{C}^D$
    for $d = 0$ **to** $d = D - 1$ **do**
      $\underline{k}' = \underline{k} + \underline{e}^d$
      if $\underline{k}' \in \mathfrak{K}$ **then**
        $\underline{c}_{\mathrm{f}}[d] = \sqrt{\underline{k}[d] + 1}\, c_{\underline{k}'}$
      end if
    end for
    // Compute (3.69)
    $\mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k})] = \sqrt{\frac{\varepsilon^2}{2}} \left(\overline{\mathbf{P}} \underline{c}_{\mathrm{f}} + \mathbf{P} \underline{c}_{\mathrm{b}} \right) + c_{\mathrm{c}} \underline{p}$
  end for
  return $\overline{\mathfrak{K}}$ and $\mathbf{c}'$

	-1	0	1	2	3	4	...
$y\phi_0$	$\overline{P}\sqrt{0}$	p	$P\sqrt{1}$				
$y\phi_1$		$\overline{P}\sqrt{1}$	p	$P\sqrt{2}$			
$y\phi_2$			$\overline{P}\sqrt{2}$	p	$P\sqrt{3}$		
$y\phi_3$				$\overline{P}\sqrt{3}$	p	$P\sqrt{4}$	
$\vdots$					$\ddots$	$\ddots$	$\ddots$

Table 3.1: First few functions $y\phi_0$, $y\phi_1$ and so on.

	...	$k-1$	k	$k+1$	...
$\vdots$	$\ddots$	$\ddots$	$\ddots$		
$y\phi_k$		$\overline{P}\sqrt{k}$	p	$P\sqrt{k+1}$	
$\vdots$			$\ddots$	$\ddots$	$\ddots$

Table 3.2: General case $y\phi_k$.

of the original basis set anymore.

3.8.4 An example

As an example we take a wavepacket $|\Psi\rangle$ in two space dimensions with the following parameter set $\Pi = \{\underline{0}, \underline{0}, \mathbb{1}, i\mathbb{1}\}$ and $\varepsilon = 0.6$. The coefficients are set to the values printed in the next table.

$\underline{k}$	$c_{\underline{k}}$
$(0,0)$	0.5
$(0,1)$	0.5
$(1,1)$	0.5
$(2,1)$	0.5

A plot of the wavepacket evaluated on a small region of position space is shown in figure 3.26.
Next we compute the gradient $-i\varepsilon\nabla\Psi$ by one of the above methods. For the new coefficients $\underline{c}_{\underline{k}}$ we get the values (only non-zero ones) shown in the next table. The two components of the gradient are plotted in figure 3.27.

	...	$\lvert\mathfrak{K}\rvert-3$	$\lvert\mathfrak{K}\rvert-3$	$\lvert\mathfrak{K}\rvert-2$	$\lvert\mathfrak{K}\rvert-1$	$\lvert\mathfrak{K}\rvert$
$\vdots$	$\ddots$	$\ddots$	$\ddots$			
$y\phi_{\lvert\mathfrak{K}\rvert-3}$		$\overline{P}\sqrt{\lvert\mathfrak{K}\rvert-3}$	p	$P\sqrt{\lvert\mathfrak{K}\rvert-2}$		
$y\phi_{\lvert\mathfrak{K}\rvert-2}$			$\overline{P}\sqrt{\lvert\mathfrak{K}\rvert-2}$	p	$P\sqrt{\lvert\mathfrak{K}\rvert-1}$	
$y\phi_{\lvert\mathfrak{K}\rvert-1}$				$\overline{P}\sqrt{\lvert\mathfrak{K}\rvert-1}$	p	$P\sqrt{\lvert\mathfrak{K}\rvert}$

Table 3.3: Highest order functions.

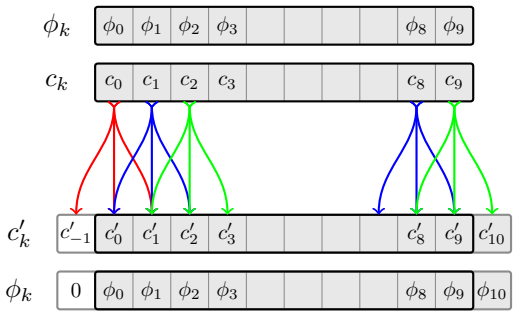


Figure 3.24: The scatter-type stencil application for computing the gradient $y\Phi$. The upper two arrays show the initial linear combination Φ consisting of basis functions ϕ_k and coefficients c_k. The lower two arrays show the linear combination of $\frac{\mathrm{d}\Phi}{\mathrm{d}x}$ consisting of the same basis functions ϕ_k and new coefficients c'_k. Each of the triple arrows represents the computation for a fixed $\underline{k} \in \mathfrak{K}$ (formula (3.67)), not all computations are shown. We see that the basis shape $\overline{\mathfrak{K}}$ for the gradient has to be larger. And also that we write to several elements not part of the original basis shape $\mathfrak{K}$. We drop the coefficient c'_{-1} (because $\phi_{-1} \equiv 0$). The original basis shape $\mathfrak{K}$ is represented by the black rectangle and the actual basis shapes are shown shaded grey.

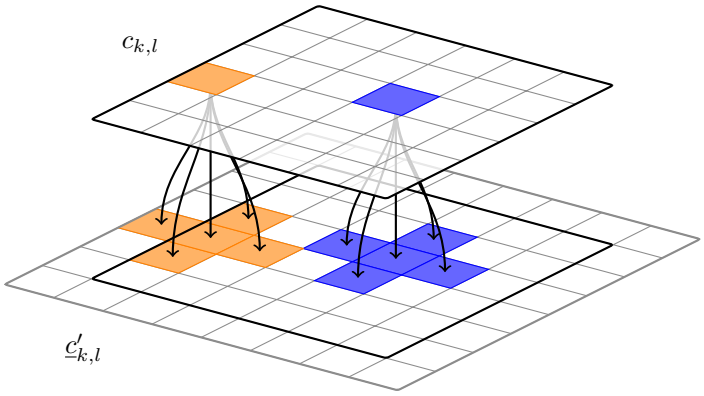


Figure 3.25: The scatter-type stencil application for computing the gradient $y\Phi$. The upper array (plane) shows the coefficients $c_{\underline{k}}$ of the initial linear combination $\Phi(\underline{x})$. The lower array (plane) shows the coefficients $\underline{c}'_{\underline{k}}$ of the linear combination of $\nabla\Phi(\underline{x})$ where each square is not a single number but stands for a whole vector. Each arrow bundle represents the computation for a fixed $\underline{k} \in \mathfrak{K}$ (formula (3.67)), not all computations are shown. The original basis shape $\mathfrak{K}$ is given by the black rectangle. We then see that the basis shape $\overline{\mathfrak{K}}$ for the gradient is larger by one square on each side. (But we again drop all coefficients with negative index.) The orange stencil shows that we sometimes have to write to elements (coefficients) that are not part of the original basis shape. Therefore we need to extend the basis shape where the whole lower plane stands for $\overline{\mathfrak{K}}$.

Algorithm 9 Compute the gradient $y\Phi$ by scatter-type stencil application

Require: Scalar wavepacket Φ in D space dimensions
Require: Basis shape $\mathfrak{K}$ (including linearisation mapping $\mu_{\mathfrak{K}}$) of Φ
Require: Parameters Π and coefficients $\{c_{\underline{k}}\}$ of Φ
  // Extend the basis shape $\mathfrak{K}$
  $\overline{\mathfrak{K}} :=$ **extend_basis_shape**$(\mathfrak{K})$
  // Storage space for the result
  $\mathbf{c}' = \mathbf{0} \in \mathbb{C}^{D \times |\overline{\mathfrak{K}}|}$
  // Iterate over original basis shape
  for $\underline{k} \in \mathfrak{K}$ **do**
    // Central node
    $\mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k})] = \mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k})] + c_{\underline{k}}\, \underline{p}$
    // Backward neighbours
    for $d = 0$ **to** $d = D - 1$ **do**
      $\underline{k}' = \underline{k} - \underline{e}^d$
      if $\underline{k}' \in \overline{\mathfrak{K}}$ **then**
        $\mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k}')] = \mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k}')] + \sqrt{\frac{\varepsilon^2}{2}} \sqrt{\underline{k}[d]}\, c_{\underline{k}}\, \overline{\mathbf{P}}[:, d]$
      end if
    end for
    // Forward neighbours
    for $d = 0$ **to** $d = D - 1$ **do**
      $\underline{k}' = \underline{k} + \underline{e}^d$
      if $\underline{k}' \in \overline{\mathfrak{K}}$ **then**
        $\mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k}')] = \mathbf{c}'[:, \mu_{\overline{\mathfrak{K}}}(\underline{k}')] + \sqrt{\frac{\varepsilon^2}{2}} \sqrt{\underline{k}[d] + 1}\, c_{\underline{k}}\, \mathbf{P}[:, d]$
      end if
    end for
  end for
  return $\overline{\mathfrak{K}}$ and $\mathbf{c}'$

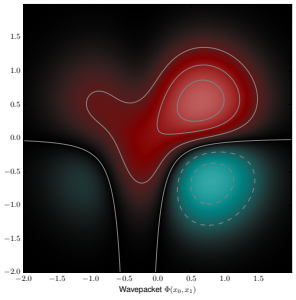


Figure 3.26: The wavepacket $|\Psi\rangle$.

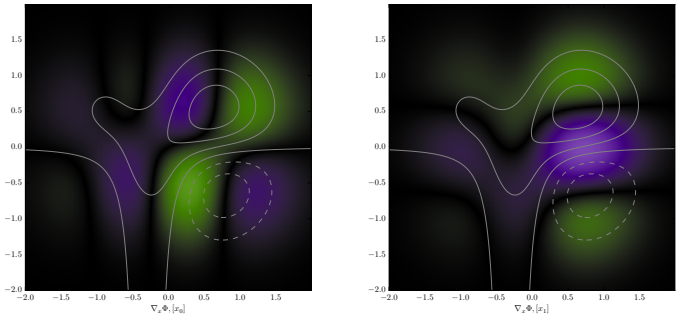


Figure 3.27: The x_0 (left) and x_1 (right) components of the gradient $-i\varepsilon\nabla\Psi$. The wavepacket Ψ is indicated by some contour levels just for reference.

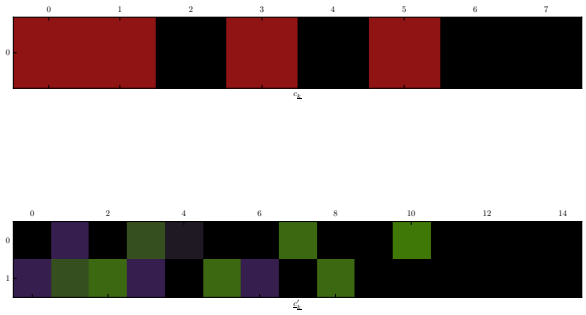


Figure 3.28: Coefficients $c_{\underline{k}}$ of Ψ (top) and $\underline{c}_{\underline{k}}$ of $-i\varepsilon\nabla\Psi$ (bottom). Notice also the different basis sizes $|\mathfrak{K}|$ of 8 and 15.

$\underline{k}$	x_0 component of $\underline{c}_{\underline{k}}$	x_1 component of $\underline{c}_{\underline{k}}$
$(0,0)$	0	$-0.21213203\imath$
$(0,1)$	$-0.21213203\imath$	$0.21213203\imath$
$(1,0)$	$0.21213203\imath$	$-0.21213203\imath$
$(1,1)$	$-0.08786797\imath$	0
$(2,0)$	0	$-0.21213203\imath$
$(2,1)$	$0.3\imath$	0
$(3,1)$	$0.36742346\imath$	0
$(0,2)$	0	$0.3\imath$
$(1,2)$	0	$0.3\imath$
$(2,2)$	0	$0.3\imath$

Finally, figure 3.28 shows again the coefficients of both, the original wavepacket and the two components of its gradient.

Chapter 4

Observables and Inner Products

In this chapter we develop the machinery necessary to get information out of the wavepackets whose time-evolution we will simulate. An essential part is the computation of several types of brakets. This will be done numerically by a special, high-order quadrature rule.

4.1 Observables in general

To compute any observable O we have to form the full braket:

$$O = \left\langle \Psi \,\middle|\, \hat{O} \,\middle|\, \Psi \right\rangle \tag{4.1}$$

which boils down to a multi-dimensional integral. We look at the most general case and seek to compute:

$$\langle \Psi' \,|\, \mathcal{F} \,|\, \Psi \rangle \tag{4.2}$$

where the operator $\mathcal{F}$ is a $N \times N$ matrix of scalar functions $\mathcal{F}_{r,c}(\underline{x})$. The wavepackets Ψ and Ψ' are assumed to be of inhomogeneous type and can have different parameter sets Π and Π'. The ansatz is:

$$\begin{aligned}
\langle \Psi' \,|\, \mathcal{F} \,|\, \Psi \rangle &= \left\langle \begin{pmatrix} \Phi'_0 \\ \vdots \\ \Phi'_{N-1} \end{pmatrix} \middle| \begin{pmatrix} & \vdots & \\ \cdots & \mathcal{F}_{r,c} & \cdots \\ & \vdots & \end{pmatrix} \middle| \begin{pmatrix} \Phi_0 \\ \vdots \\ \Phi_{N-1} \end{pmatrix} \right\rangle \\
&= \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} \langle \Phi'_r \,|\, \mathcal{F}_{r,c} \,|\, \Phi_c \rangle \ .
\end{aligned}$$

We continue by using the definition (3.46) for the individual components and resolve:

$$\begin{aligned}\langle \Phi'_r \,|\, \mathcal{F}_{r,c} \,|\, \Phi_c \rangle &= \left\langle \sum_{\underline{k} \in \mathfrak{K}'_r} \phi'_{\underline{k}} \,\middle|\, \mathcal{F}_{r,c} \,\middle|\, \sum_{\underline{l} \in \mathfrak{K}_c} \phi_{\underline{l}} \right\rangle \\ &= \sum_{\underline{k} \in \mathfrak{K}'_r} \sum_{\underline{l} \in \mathfrak{K}_c} \left\langle \phi'_{\underline{k}} \,\middle|\, \mathcal{F}_{r,c} \,\middle|\, \phi_{\underline{l}} \right\rangle\end{aligned}$$

where we left out the global phase as well as the coefficients. The last braket consists of basis functions only and we know that:

$$\left\langle \phi'_{\underline{k}} \,\middle|\, \mathcal{F}_{r,c} \,\middle|\, \phi_{\underline{l}} \right\rangle = \int \cdots \int \overline{\phi'_{\underline{k}}(\underline{x})} \mathcal{F}_{r,c}(\underline{x}) \phi_{\underline{l}}(\underline{x}) \mathrm{d}\underline{x} \,.$$

Hence we make a longer detour and examine the computation of any inner-product like this one before we return to observables.

4.2 Inner products

4.2.1 Integrals over basis functions

In the following we want to compute inner products of the form $\langle \phi_k[\Pi_k] \,|\, \phi_l[\Pi_l] \rangle$ with just an identity in place of $\mathcal{F}_{r,c}$ from above. We abuse the notation here. The indices k and l are not multi-indices and do not index the ϕ in the corresponding basis set. They are solely used to discriminate between the bra and the ket, to make clear which function ϕ and parameter set Π we speak of. There is no useful closed form solution to this integral and we have to compute it numerically by using quadrature. For the derivation of the quadrature formulae we work with the ground states only and hence $\phi_k = \phi'_{\underline{0}}$ and $\phi_l = \phi_{\underline{0}}$.

4.2.2 Quadrature rules

To compute the integral shown in the last section we use a Gauss-Hermite quadrature rule of very high order. The properties of this quadrature rule make it well-suited for our purpose. The quadrature rule ρ consists of nodes γ and weights ω. In the case of Gauss-Hermite quadrature these values are built to integrate $f(x)$ in:

$$\int_{\mathbb{R}} e^{-x^2} f(x) \mathrm{d}x \approx \sum_{i=0}^{R-1} \omega_i f(\gamma_i) \,. \tag{4.3}$$

For a quadrature of order R the nodes $\{\gamma_i\}_{i=0}^{R-1}$ are then given as the roots of the Hermite polynomial $H_R(x)$:

$$H_R(x) = (-1)^R e^{x^2} \frac{d^R}{dx^R} e^{-x^2} \,.$$

Of course we do not compute the nodes by finding the roots of these polynomials as this is inherently unstable. The quadrature weights are then given by:

$$\omega_i = \frac{2^{R-1} R! \sqrt{\pi}}{R^2 H_{R-1}^2(\gamma_i)} \,.$$

Since our integrals are not of the form (4.3) but instead we have:

$$\int_{\mathbb{R}} g(x)\mathrm{d}x \tag{4.4}$$

where the $\exp(-x^2)$ is built into the function $g(x)$ such that $g(x) = \exp(-x^2)f(x)$, we have to alter the ansatz. We can not divide by $\exp(-x^2)$ without getting major numerical instabilities. But we can modify our quadrature weights to take that factor into account. We define new quadrature weights ω_i' as:

$$\omega_i' := \frac{1}{R\, h_R^2(\gamma_i)}$$

where h_R are the Hermite functions defined as:

$$h_R(x) := \frac{1}{\sqrt{2^R R! \sqrt{\pi}}} e^{-x^2/2} H_R(x)\,.$$

We can evaluate the Hermite function for any point x by a stable, recursive scheme. Finally the quadrature rule ρ in use is given by:

$$\rho := \{(\gamma_i, \omega_i')\}_{i=0}^{R-1} \tag{4.5}$$

for one space dimension. In higher dimensions we build a quadrature rule ρ by computing the full tensor product of D one-dimensional quadrature rules ρ_d:

$$\rho := \bigotimes_{d=0}^{D-1} \rho_d\,. \tag{4.6}$$

Using the rules ρ_d, each of order R_d, we get the D-dimensional rule then denoted by:

$$\rho := \left\{ \left(\underline{\gamma}_{\underline{i}}, \omega_i \right) \right\}_{i=0}^{R-1} \tag{4.7}$$

with a total number $R = \prod_{d=0}^{D-1} R_d$ of quadrature nodes. The quadrature nodes $\underline{\gamma}_{\underline{j}} \in \mathbb{R}^D$ are constructed as:

$$\underline{\gamma}_{\underline{j}} := \begin{pmatrix} \gamma_{\underline{j}_0}^0 \\ \vdots \\ \gamma_{\underline{j}_{D-1}}^{D-1} \end{pmatrix}$$

with $\underline{j} \in [0, R_0 - 1] \times \cdots \times [0, R_{D-1} - 1]$ a multi-index. Each of the γ^d belongs to the one-dimensional rule ρ_d. For the weights ω_i of ρ we have:

$$\omega_{\underline{j}} := \prod_{d=0}^{D-1} \omega_{\underline{j}_d}^d$$

and again ω^d is part of ρ_d. Figure 4.1 shows a typical two-dimensional quadrature rule.

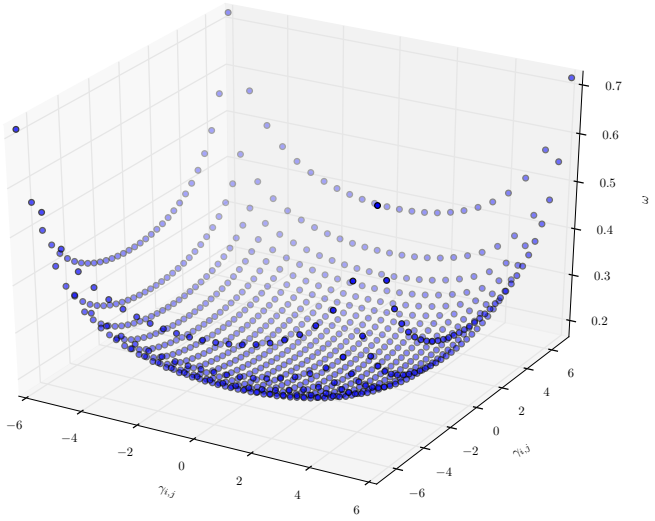


Figure 4.1: Plot of the nodes $\underline{\gamma}_i$ and weights ω_i of a two-dimensional quadrature rule ρ. The rule is build from two one-dimensional rules of order 24 and 32.

4.2.3 Adapting the quadrature

Since the basis functions are parametrised we have to adapt the quadrature rule $\{\underline{\gamma}_i, \omega_i\}_i$ to fit best the given situation depending on the sets Π_k and Π_l. If both functions in the bra and the ket are members of the same family with $\Pi_k \equiv \Pi_l$ the process is much easier and we will do this case first. But we will need also the more general case where $\Pi_k \neq \Pi_l$ later. The purpose of this subsection is to find a transformation rule for the quadrature nodes $\underline{\gamma}_i$ to suit the wavepacket's basis best. The final rule will be an affine transformation like:

$$\underline{\gamma_i}' = \underline{v} + \mathbf{A}\underline{\gamma_i} \tag{4.8}$$

where the quadrature node $\underline{\gamma_i} \in \mathbb{R}^D$, the offset vector $\underline{v} \in \mathbb{R}^D$ and the transformation matrix $\mathbf{A} \in \mathbb{R}^{D \times D}$.

4.2.4 The homogeneous case

We first threat the homogeneous case of the overlap integral. This case is much easier as we assume that we have the same wavepacket in the bra as well as in the ket. Hence the inner-product simplifies to $\langle \phi \,|\, \phi \rangle$ and we have the same set Π of Hagedorn Parameters. This makes combining the two exponential terms from the definition (3.13) into a single one of the same form much easier.
We concentrate on the exponential parts which dominate the overall shape of the basis functions and we are interested in the quadratic term only. Essentially we only need to know where the peak of the Gaussian is and how big the spread is.

$$
\begin{aligned}
&\langle \phi \,|\, \phi \rangle = \\
&\quad \exp\Bigg(\overline{\frac{i}{2\varepsilon^2} \left\langle \left(\underline{x}-\underline{q}\right), \mathbf{P}\mathbf{Q}^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle + \frac{i}{\varepsilon^2}\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle} \\
&\qquad\quad + \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x}-\underline{q}\right), \mathbf{P}\mathbf{Q}^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle + \frac{i}{\varepsilon^2}\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle \Bigg) \\
&= \exp\Bigg(\frac{-i}{2\varepsilon^2} \overline{\left\langle \left(\underline{x}-\underline{q}\right), \mathbf{P}\mathbf{Q}^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle} + \frac{-i}{\varepsilon^2}\overline{\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle} \\
&\qquad\quad + \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x}-\underline{q}\right), \mathbf{P}\mathbf{Q}^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle + \frac{i}{\varepsilon^2}\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle \Bigg) \\
&= \exp\Bigg(\frac{-i}{2\varepsilon^2} \left\langle \mathbf{P}\mathbf{Q}^{-1}\left(\underline{x}-\underline{q}\right), \left(\underline{x}-\underline{q}\right)\right\rangle + \frac{-i}{\varepsilon^2}\left\langle \left(\underline{x}-\underline{q}\right), \underline{p}\right\rangle \\
&\qquad\quad + \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x}-\underline{q}\right), \mathbf{P}\mathbf{Q}^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle + \frac{i}{\varepsilon^2}\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle \Bigg)
\end{aligned}
$$

For the sake of readability we define the matrix:

$$\mathbf{\Gamma} := \mathbf{P}\mathbf{Q}^{-1}$$

and continue. From now on we drop the exponential and work on the exponent only. (The equal signs have to be understood within this laziness.)

$$
\begin{aligned}
&= \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x}-\underline{q}\right), \mathbf{\Gamma}\left(\underline{x}-\underline{q}\right)\right\rangle - \frac{i}{2\varepsilon^2} \left\langle \mathbf{\Gamma}\left(\underline{x}-\underline{q}\right), \left(\underline{x}-\underline{q}\right)\right\rangle + \frac{i}{\varepsilon^2}\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle - \frac{i}{\varepsilon^2}\left\langle \left(\underline{x}-\underline{q}\right), \underline{p}\right\rangle \\
&= \frac{i}{2\varepsilon^2}\Big(\left\langle \left(\underline{x}-\underline{q}\right), \mathbf{\Gamma}\left(\underline{x}-\underline{q}\right)\right\rangle - \left\langle \mathbf{\Gamma}\left(\underline{x}-\underline{q}\right), \left(\underline{x}-\underline{q}\right)\right\rangle\Big) + \frac{i}{\varepsilon^2}\Big(\left\langle \underline{p}, \left(\underline{x}-\underline{q}\right)\right\rangle - \left\langle \left(\underline{x}-\underline{q}\right), \underline{p}\right\rangle\Big)
\end{aligned}
$$

The linear terms vanish as the inner-product is symmetric for entirely real arguments. We then rearrange and combine the brakets:

$$
\begin{aligned}
&= \frac{i}{2\varepsilon^2}\Big(\left\langle \left(\underline{x}-\underline{q}\right), \mathbf{\Gamma}\left(\underline{x}-\underline{q}\right)\right\rangle - \left\langle \left(\underline{x}-\underline{q}\right), \mathbf{\Gamma}^{\mathrm{H}}\left(\underline{x}-\underline{q}\right)\right\rangle\Big) \\
&= \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x}-\underline{q}\right), \left(\mathbf{\Gamma}-\mathbf{\Gamma}^{\mathrm{H}}\right)\left(\underline{x}-\underline{q}\right)\right\rangle .
\end{aligned}
$$

Now we are almost done. The term on the last line is of the same form as the quadratic one in the definition of $|\phi\rangle$. Hence we succeeded in combining the Gaussian exponential parts of two wavepackets into a single one.
The only thing left is to simplify the operator $\mathbf{\Gamma} - \mathbf{\Gamma}^{\mathrm{H}}$ but this is not difficult. Recalling the definition of $\mathbf{\Gamma}$ we get:

$$
\begin{aligned}
\mathbf{\Gamma} - \mathbf{\Gamma}^{\mathrm{H}} &= \mathbf{P}\mathbf{Q}^{-1} - \left(\mathbf{P}\mathbf{Q}^{-1}\right)^{\mathrm{H}} \\
&= \mathbf{P}\mathbf{Q}^{-1} - \mathbf{Q}^{-\mathrm{H}}\mathbf{P}^{\mathrm{H}} \\
&= \mathbf{P}\mathbf{Q}^{-1} - \left(\mathbf{P}\mathbf{Q}^{-1}\right)^{\mathrm{H}} .
\end{aligned}
$$

Remembering the second of the two compatibility conditions in (3.11) we get:

$$\begin{aligned}
\mathbf{Q}^{\mathrm{H}}\mathbf{P} - \mathbf{P}^{\mathrm{H}}\mathbf{Q} &= 2i\mathbb{1} \\
\mathbf{Q}^{-\mathrm{H}}\mathbf{Q}^{\mathrm{H}}\mathbf{P} - \mathbf{Q}^{-\mathrm{H}}\mathbf{P}^{\mathrm{H}}\mathbf{Q} &= 2i\mathbf{Q}^{-\mathrm{H}} \\
\mathbf{P}\mathbf{Q}^{-1} - \mathbf{Q}^{-\mathrm{H}}\mathbf{P}^{\mathrm{H}}\mathbf{Q}\mathbf{Q}^{-1} &= 2i\mathbf{Q}^{-\mathrm{H}}\mathbf{Q}^{-1} \\
\mathbf{P}\mathbf{Q}^{-1} - \mathbf{Q}^{-\mathrm{H}}\mathbf{P}^{\mathrm{H}} &= 2i\mathbf{Q}^{-\mathrm{H}}\mathbf{Q}^{-1} \\
\mathbf{P}\mathbf{Q}^{-1} - \left(\mathbf{P}\mathbf{Q}^{-1}\right)^{\mathrm{H}} &= 2i\left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}
\end{aligned}$$

where we used identities from [16]. Hence:

$$\mathbf{\Gamma} - \mathbf{\Gamma}^{\mathrm{H}} = 2i\left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}.$$

We want the combined wavepacket to look like the following (which is motivated by the reason that this exactly yields a Gaussian with shift $\underline{q_0}$ and spread $\mathbf{Q_0}$):

$$\exp\left(-\frac{1}{\varepsilon^2}\left\langle\left(\underline{x}-\underline{q_0}\right), \mathbf{Q_0}\left(\underline{x}-\underline{q_0}\right)\right\rangle + \text{optional junk}\right) \tag{4.9}$$

where $\underline{q_0} \in \mathbb{R}^D$ and $\mathbf{Q_0} \in \mathbb{C}^{D\times D}$ are the final parameters to determine.
From above we have:

$$\begin{aligned}
&\exp\left(\frac{i}{2\varepsilon^2}\left\langle\left(\underline{x}-\underline{q}\right), \left(\mathbf{\Gamma}-\mathbf{\Gamma}^{\mathrm{H}}\right)\left(\underline{x}-\underline{q}\right)\right\rangle\right) \\
&= \exp\left(\frac{i}{2\varepsilon^2}\left\langle\left(\underline{x}-\underline{q}\right), 2i\left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle\right) \\
&= \exp\left(-\frac{1}{\varepsilon^2}\left\langle\left(\underline{x}-\underline{q}\right), \left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}\left(\underline{x}-\underline{q}\right)\right\rangle\right).
\end{aligned}$$

Therefore we immediately see that we have to set:

$$\boxed{\underline{q_0} = \underline{q}} \tag{4.10}$$

and:

$$\boxed{\mathbf{Q_0} = \left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}} \tag{4.11}$$

Notice that in the limit of $D \to 1$ we retrieve the scalar case presented in [1] where $q_0 \in \mathbb{R}$ and $Q_0 \in \mathbb{C}$ and:

$$\begin{aligned}
q_0 &= q \\
Q_0 &= \left(Q\overline{Q}\right)^{-1} = \frac{1}{|Q|^2}.
\end{aligned}$$

4.2.5 The parameter $\mathbf{Q_S}$

For the transformation of the quadrature rule $\{\gamma_i, \omega_i\}_i$ the values $\mathbf{Q_0}$ and $\mathbf{q_0}$ are not enough. We need a parameter denoted by $\mathbf{Q_S}$ which is in the one-dimensional case given by:

$$Q_S := \frac{1}{\sqrt{Q_0}}\,.$$

Therefore our next goal is to find a relation which holds in the D-dimensional case too. We propose in analogy the following formula:

$$\boxed{\mathbf{Q_S} := \left(\sqrt{\mathbf{Q_0}}\right)^{-1}} \tag{4.12}$$

But we have to find a way to express the root of a matrix in a suitable way. There is a way to compute almost any scalar function $f(x)$ for a matrix argument $\mathbf{X}$. The trick goes by the similarity transform of $\mathbf{X}$ and then computing f on the eigenvalues λ_i of $\mathbf{X}$. But this method relies on $\mathbf{X}$ being diagonalisable and f being defined on the whole spectrum of $\mathbf{X}$. We will now take a longer but probably cleaner way which in the end turns out to be not that different.
We are in the homogeneous case and thus $\mathbf{Q_0} = \left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}$. Refining the above expression step by step we get:

$$\begin{aligned}
\mathbf{Q_S} &:= \left(\sqrt{\mathbf{Q_0}}\right)^{-1} \\
&= \left(\sqrt{\left(\mathbf{Q}\mathbf{Q}^{\mathrm{H}}\right)^{-1}}\right)^{-1} \\
&= \left(\sqrt{\left(\mathbf{Q}^{\mathrm{H}}\right)^{-1}\mathbf{Q}^{-1}}\right)^{-1} \\
&= \left(\sqrt{\left(\mathbf{Q}^{-1}\right)^{\mathrm{H}}\mathbf{Q}^{-1}}\right)^{-1}.
\end{aligned}$$

If we now define $\mathbf{X} := \mathbf{Q}^{-1}$ we get:

$$\mathbf{Q_S} := \left(\sqrt{\mathbf{X}^{\mathrm{H}}\mathbf{X}}\right)^{-1}. \tag{4.13}$$

For a complex square matrix $\mathbf{A}$ one can define the so called *polar decomposition* as follows:

$$\mathbf{A} =: \mathbf{U}\mathbf{P}$$

where $\mathbf{U}$ is unitary and $\mathbf{P}$ is a positive-semidefinite Hermitian matrix. The two factors are given as:

$$\begin{aligned}
\mathbf{P} &:= \sqrt{\mathbf{A}^{\mathrm{H}}\mathbf{A}} \\
\mathbf{U} &:= \mathbf{A}\mathbf{P}^{-1}
\end{aligned}$$

with $\mathbf{P}$ being unique. When looking at $\mathbf{P}$ we recognise the root expression from above if we take $\mathbf{A} \equiv \mathbf{X} = \mathbf{Q}^{-1}$. We have shown that we can find the matrix $\mathbf{Q_S}$ by polar decomposition of $\mathbf{Q}^{-1} = \mathbf{U}\mathbf{P}$. With this factorisation we can write:

$$\mathbf{Q_S} = \mathbf{P}^{-1} = \mathbf{QU}\,.$$

The remaining question is now how to compute this decomposition of $\mathbf{Q}^{-1}$ into $\mathbf{U}$ and $\mathbf{P}$. And the answer is quite trivial. One can compute the polar decomposition of $\mathbf{A}$ from its *singular value decomposition*. We write the singular value decomposition as:

$$\mathbf{A} =: \mathbf{W\Sigma V}^{\mathrm{H}}$$

where $\mathbf{W}$ and $\mathbf{V}$ are two unitary matrices (which in our case are both of the same size $D \times D$ because we started with a square matrix $\mathbf{A}$) and $\mathbf{\Sigma}$ is a diagonal matrix. Now we can write the two factors $\mathbf{U}$, $\mathbf{P}$ of the polar decomposition of $\mathbf{A}$ as:

$$\begin{aligned}\mathbf{P} &= \mathbf{V}\Sigma\mathbf{V}^{\mathrm{H}}\\ \mathbf{U} &= \mathbf{WV}^{\mathrm{H}}\,.\end{aligned}$$

We obtain the final result for $\mathbf{Q_S}$ as follows, first we compute the singular value decomposition of $\mathbf{Q}^{-1}$:

$$\mathbf{W\Sigma V}^{\mathrm{H}} := \mathbf{Q}^{-1}$$

then we form $\mathbf{P}^{-1}$ and simplify the result using the fact that $\mathbf{V}$ is unitary:

$$\begin{aligned}\mathbf{Q_S} &= \mathbf{P}^{-1}\\ &= \left(\mathbf{V\Sigma V}^{\mathrm{H}}\right)^{-1}\\ &= \mathbf{V}^{-\mathrm{H}}\mathbf{\Sigma}^{-1}\mathbf{V}^{-1}\\ &= \mathbf{V}^{\mathrm{H}\mathrm{H}}\mathbf{\Sigma}^{-1}\mathbf{V}^{\mathrm{H}}\,.\end{aligned}$$

At the end of the day we reached our goal and write:

$$\boxed{\mathbf{Q_S} := \mathbf{V\Sigma}^{-1}\mathbf{V}^{\mathrm{H}}}$$

Now we may remember what was mentioned above about computing a function of a matrix. But instead of the *eigenvalue decomposition* $\mathbf{T\Lambda T}^{-1}$ of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$ we used the singular value decomposition. For $\mathbf{W\Sigma V}^{\mathrm{H}} = \mathbf{A}$ recall that $\mathbf{\Sigma} = \sqrt{\mathbf{\Lambda}}$ where $\mathbf{\Lambda}$ is the diagonal matrix which contains the eigenvalues of $\mathbf{A}^{\mathrm{H}}\mathbf{A}$. So what we did is essentially the same but with less magic. Also notice that the transformation matrices $\mathbf{V}$ are unitary hence minimising numerical errors.
To justify this formula further we can show that the reduction $D \to 1$ to the scalar case yields the correct result. In this case $\mathbf{Q_0}$ is a 1×1 matrix as well as $\mathbf{Q}$. The singular value decomposition $\mathbf{W\Sigma V}^{\mathrm{H}} = \mathbf{A}$ can be computed as:

$$\begin{aligned}\mathbf{W} &:= \text{eigenvectors}\left(\mathbf{AA}^{\mathrm{H}}\right)\\ \mathbf{V} &:= \text{eigenvectors}\left(\mathbf{A}^{\mathrm{H}}\mathbf{A}\right)\\ \mathbf{\Sigma} &:= \sqrt{\text{diagonal}\left(\text{eigenvalues}\left(\mathbf{AA}^{\mathrm{H}}\right)\right)}\,.\end{aligned}$$

The eigenvector of a 1×1 matrix is of course just 1 and the eigenvalue equals the single entry. Therefore we have:

$$\Sigma = \sqrt{(QQ^{\mathrm{H}})^{-1}} = \sqrt{\frac{1}{\overline{Q}Q}} = \frac{1}{\sqrt{|Q|^2}} = \frac{1}{|Q|}$$

and finally:

$$Q_S = \Sigma^{-1} = |Q| \,.$$

We derived this equation assuming the matrix $\mathbf{Q_0}$ from the homogeneous mixing case and also $\mathbf{Q}$ enters in the singular value decomposition. In the inhomogeneous case however we cannot apply it because we do not know what to feed into the singular value decomposition. In that case we have to rely on other techniques of computing the square root of a matrix. Details about the computation of matrix square roots can be found in [10].

4.2.6 The inhomogeneous case

In this section we do the same derivation as in the last one but now in the fully generalised case where both wavepackets have different parameter sets Π_k and Π_l. We derive the mixing relations for the D dimensional case.

$$\begin{aligned}
\langle \phi_k \,|\, \phi_l \rangle = & \\
& \exp\left(\overline{\frac{i}{2\varepsilon^2} \left\langle \left(\underline{x} - \underline{q_k}\right), \mathbf{P_k Q_k}^{-1} \left(\underline{x} - \underline{q_k}\right) \right\rangle + \frac{i}{\varepsilon^2} \left\langle \underline{p_k}, \left(\underline{x} - \underline{q_k}\right) \right\rangle} \right. \\
& \qquad \left. + \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x} - \underline{q_l}\right), \mathbf{P_l Q_l}^{-1} \left(\underline{x} - \underline{q_l}\right) \right\rangle + \frac{i}{\varepsilon^2} \left\langle \underline{p_l}, \left(\underline{x} - \underline{q_l}\right) \right\rangle \right) \\
= & \exp\left(\frac{-i}{2\varepsilon^2} \overline{\left\langle \left(\underline{x} - \underline{q_k}\right), \mathbf{P_k Q_k}^{-1} \left(\underline{x} - \underline{q_k}\right) \right\rangle} + \frac{-i}{\varepsilon^2} \overline{\left\langle \underline{p_k}, \left(\underline{x} - \underline{q_k}\right) \right\rangle} \right. \\
& \qquad \left. + \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x} - \underline{q_l}\right), \mathbf{P_l Q_l}^{-1} \left(\underline{x} - \underline{q_l}\right) \right\rangle + \frac{i}{\varepsilon^2} \left\langle \underline{p_l}, \left(\underline{x} - \underline{q_l}\right) \right\rangle \right) \\
= & \exp\left(\frac{-i}{2\varepsilon^2} \left\langle \mathbf{P_k Q_k}^{-1} \left(\underline{x} - \underline{q_k}\right), \left(\underline{x} - \underline{q_k}\right) \right\rangle + \frac{-i}{\varepsilon^2} \left\langle \left(\underline{x} - \underline{q_k}\right), \underline{p_k} \right\rangle \right. \\
& \qquad \left. + \frac{i}{2\varepsilon^2} \left\langle \left(\underline{x} - \underline{q_l}\right), \mathbf{P_l Q_l}^{-1} \left(\underline{x} - \underline{q_l}\right) \right\rangle + \frac{i}{\varepsilon^2} \left\langle \underline{p_l}, \left(\underline{x} - \underline{q_l}\right) \right\rangle \right)
\end{aligned}$$

Define the two matrices:

$$\begin{aligned}
\mathbf{\Gamma_k} &:= \mathbf{P_k Q_k}^{-1} \\
\mathbf{\Gamma_l} &:= \mathbf{P_l Q_l}^{-1}
\end{aligned}$$

and place them in the above expressions. From this line on we only care about the argument of the exponential:

$$= \frac{i}{2\varepsilon^2}\left\langle \left(\underline{x}-\underline{q_l}\right), \mathbf{\Gamma_l}\left(\underline{x}-\underline{q_l}\right)\right\rangle - \frac{i}{2\varepsilon^2}\left\langle \mathbf{\Gamma_k}\left(\underline{x}-\underline{q_k}\right), \left(\underline{x}-\underline{q_k}\right)\right\rangle + \frac{i}{\varepsilon^2}\left\langle \underline{p_l}, \left(\underline{x}-\underline{q_l}\right)\right\rangle - \frac{i}{\varepsilon^2}\left\langle \left(\underline{x}-\underline{q_k}\right), \underline{p_k}\right\rangle$$
$$= \frac{i}{2\varepsilon^2}\Big(\left\langle \left(\underline{x}-\underline{q_l}\right), \mathbf{\Gamma_l}\left(\underline{x}-\underline{q_l}\right)\right\rangle - \left\langle \mathbf{\Gamma_k}\left(\underline{x}-\underline{q_k}\right), \left(\underline{x}-\underline{q_k}\right)\right\rangle\Big) + \frac{i}{\varepsilon^2}\Big(\left\langle \underline{p_l}, \left(\underline{x}-\underline{q_l}\right)\right\rangle - \left\langle \left(\underline{x}-\underline{q_k}\right), \underline{p_k}\right\rangle\Big).$$

This time the linear part won't vanish and we can not drop it. But we can ignore it. (Be very attentive while reading the formulae as we will ignore more and more junk terms that are not helpful for reaching our goal.)

$$= \frac{i}{2\varepsilon^2}\Big(\left\langle \left(\underline{x}-\underline{q_l}\right), \mathbf{\Gamma_l}\left(\underline{x}-\underline{q_l}\right)\right\rangle - \left\langle \left(\underline{x}-\underline{q_k}\right), \mathbf{\Gamma_k}^{\mathrm{H}}\left(\underline{x}-\underline{q_k}\right)\right\rangle\Big)$$

At this stage we can not combine the two inner products as we did in the homogeneous case. So the only option left is to expand the brakets:

$$\begin{aligned} = \frac{i}{2\varepsilon^2}\Big(&\left\langle \underline{x}, \mathbf{\Gamma_l}\underline{x}\right\rangle - \left\langle \underline{x}, \mathbf{\Gamma_l}\underline{q_l}\right\rangle - \left\langle \underline{q_l}, \mathbf{\Gamma_l}\underline{x}\right\rangle + \left\langle \underline{q_l}, \mathbf{\Gamma_l}\underline{q_l}\right\rangle \\ &- \left\langle \underline{x}, \mathbf{\Gamma_k}^{\mathrm{H}}\underline{x}\right\rangle + \left\langle \underline{x}, \mathbf{\Gamma_k}^{\mathrm{H}}\underline{q_k}\right\rangle + \left\langle \underline{q_k}, \mathbf{\Gamma_k}^{\mathrm{H}}\underline{x}\right\rangle - \left\langle \underline{q_k}, \mathbf{\Gamma_k}^{\mathrm{H}}\underline{q_k}\right\rangle\Big). \end{aligned}$$

As a first step towards the goal (4.9) we can combine the two terms that are quadratic in $\underline{x}$.

$$= \frac{i}{2\varepsilon^2}\Big(\left\langle \underline{x}, \left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\underline{x}\right\rangle + \mathrm{junk}\Big)$$

Notice that when we expand the braket in (4.9) we get one that is quadratic in $\underline{x}$ and looks like $\langle \underline{x}, \mathbf{Q_0}\underline{x}\rangle$. This is essentially what we wrote on the line above. We just have to find a way to transform the prefactors. And the way to achieve this goes by taking the imaginary part and pulling a factor of $\frac{1}{2}$ inside the braket:

$$\begin{aligned} &= \frac{i}{2\varepsilon^2}\left\langle \underline{x}, \left(\Re\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right) + i\Im\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\right)\underline{x}\right\rangle \\ &= \underbrace{\frac{i}{2\varepsilon^2}\left\langle \underline{x}, \Re\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\underline{x}\right\rangle}_{\mathrm{junk}} + \frac{i}{2\varepsilon^2}\left\langle \underline{x}, i\Im\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\underline{x}\right\rangle \\ &= -\frac{1}{2\varepsilon^2}\left\langle \underline{x}, \Im\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\underline{x}\right\rangle \\ &= -\frac{1}{\varepsilon^2}\left\langle \underline{x}, \frac{1}{2}\Im\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\underline{x}\right\rangle \end{aligned}$$

where we made use of the sesquilinearity in many steps. From the last line and when we compare the quadratic term to (4.9) it is obvious that the parameter $\mathbf{Q_0}$ has to be:

$$\boxed{\mathbf{Q_0} := \frac{1}{2}\Im\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)} \tag{4.14}$$

Now we compute the other parameter $\underline{q_0}$. It turns out that this is more difficult. We could take a look at the scalar one-dimensional case where the real variable q_0 is of the following form:

$$q_0 := \frac{\Im\left(\Gamma_l q_l - \overline{\Gamma_k} q_k\right)}{\Im\left(\Gamma_l - \overline{\Gamma_k}\right)} .$$

In the multi-dimensional case however we deal with vectors in $\mathbb{R}^D$ and matrices in $\mathbb{C}^{D\times D}$ thus we have to get rid of the fractions and write proper inverses. Hence we suggest that in our case the vector $\underline{q_0}$ should look like:

$$\boxed{\underline{q_0} := \left(\Im\left(\mathbf{\Gamma_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\right)\right)^{-1} \Im\left(\mathbf{\Gamma_l}\underline{q_l} - \mathbf{\Gamma_k}^{\mathrm{H}}\underline{q_k}\right)} \tag{4.15}$$

This expression is also motivated from the fact that $\underline{q_k}$, $\underline{q_l}$ are (column) vectors and $\mathbf{\Gamma_k}$, $\mathbf{\Gamma_l}$ and $\mathbf{Q_0}$ are square matrices. Hence $\underline{q_0}$ is a vector as it should be. It is easy to show that these two formulae reduce to (4.10) and (4.11) when we choose $\mathbf{\Gamma_k} \equiv \mathbf{\Gamma_l}$. Algorithm 10 implements this general D-dimensional mixing of two parameter sets Π_k and Π_l.

Algorithm 10 Mixing two sets Π_r and Π_c of Hagedorn parameters

Require: Two sets Π_r and Π_c of Hagedorn parameters
// Apply the mixing formula (4.15) and (4.14) to the parameters
$\mathbf{\Gamma_r} := \mathbf{P_r}\mathbf{Q_r}^{-1}$
$\mathbf{\Gamma_c} := \mathbf{P_c}\mathbf{Q_c}^{-1}$
$\mathbf{\Gamma} := \Im\left(\mathbf{\Gamma_c} - \mathbf{\Gamma_r}^{\mathrm{H}}\right)$
$\underline{g} := \Im\left(\mathbf{\Gamma_c}\underline{q_c} - \mathbf{\Gamma_r}^{\mathrm{H}}\underline{q_r}\right)$
$\underline{q_0} := \mathbf{\Gamma}^{-1}\underline{g}$
$\mathbf{Q_0} := \frac{1}{2}\mathbf{\Gamma}$
// Apply the formula (4.12)
$\mathbf{Q_S} := \left(\sqrt{\mathbf{Q_0}}\right)^{-1}$
return $\mathbf{q_0}$ and $\mathbf{Q_S}$

4.2.7 Quadrature applied

After we discussed in details the transformation of the quadrature nodes in the last section we now look at the final quadrature rule. It's not really difficult, but it's good to write down all the details at least once. First we transform the quadrature nodes by (4.8) giving:

$$\underline{\gamma_i}' = \underline{q_0} + \varepsilon \mathbf{Q_S}\underline{\gamma_i} . \tag{4.16}$$

Then we carry out the quadrature for computing the braket:

$$\boxed{\langle \phi_k \,|\, f \,|\, \phi_l \rangle \approx \varepsilon^D \cdot |\det(\mathbf{Q_S})| \cdot \sum_{r=0}^{R-1} \overline{\phi_k\left(\underline{\gamma_r}'\right)} \cdot f\left(\underline{\gamma_r}'\right) \cdot \phi_l\left(\underline{\gamma_r}'\right) \cdot \omega_r} \tag{4.17}$$

where the two ϕ in general belong to the different families. But if they really have the same parameter set Π then we can simplify this formula slightly. The

trick is that we omit a prefactor of $\frac{1}{\sqrt{\det(\mathbf{Q})}}$ when evaluating $\phi\left(\underline{\gamma_r}'\right)$. Because we have two times this evaluation both prefactors accumulate to $\frac{1}{\det(\mathbf{Q})}$. In the case of a single family we have[1] $\det(\mathbf{Q_S}) = |\det(\mathbf{Q})|$ hence the $|\det(\mathbf{Q_S})|$ outside the sum cancels nicely with these prefactors. And the above formula becomes:

$$\boxed{\langle\phi\,|\,f\,|\,\phi\rangle \approx \varepsilon^D \cdot \sum_{r=0}^{R-1} \overline{\phi\left(\underline{\gamma_r}'\right)} \cdot f\left(\underline{\gamma_r}'\right) \cdot \phi\left(\underline{\gamma_r}'\right) \cdot \omega_r} \tag{4.18}$$

4.3 Matrix elements

Given a scalar function $f(\underline{x})$ and two sets $\left\{\phi_{\underline{k}}\right\}_{\underline{k}\in\mathfrak{K}}$ and $\left\{\phi'_{\underline{l}}\right\}_{\underline{l}\in\mathfrak{K}'}$ of basis functions. These functions depend as usual on their parameter sets like $\phi\,[\Pi]$ and $\phi\,[\Pi']$. In the general case both parameter sets will be different. We now try to compute *matrix elements* like this one:

$$\mathbf{F}_{\mu_{\mathfrak{K}}(\underline{k}),\mu_{\mathfrak{K}'}(\underline{l})} := \left\langle \phi_{\underline{k}}\,[\Pi] \,\middle|\, f \,\middle|\, \phi_{\underline{l}}\,[\Pi'] \right\rangle . \tag{4.19}$$

We assumed that $\underline{k} \in \mathfrak{K}$ and $\underline{l} \in \mathfrak{K}'$. The matrix $\mathbf{F}$ we can build out of these elements is of the following form:

$$\mathbf{F} := \begin{pmatrix} & \vdots & \\ \cdots & \left\langle \phi_{\underline{k}}\,[\Pi] \,\middle|\, f \,\middle|\, \phi_{\underline{l}}\,[\Pi'] \right\rangle & \cdots \\ & \vdots & \end{pmatrix} \tag{4.20}$$

and has size $|\mathfrak{K}| \times |\mathfrak{K}'|$. The order of the entries is given by the linearisation mappings $\mu_{\mathfrak{K}}$ and $\mu_{\mathfrak{K}'}$.
A generalisation of this algorithm is given in chapter 5 by algorithms 14 and 16. The basic principle is the same. Instead of a scalar function f we have a matrix $\mathcal{F}$ full of scalar functions $f_{r,c}$. The packets are then vector-valued wavepackets of homogeneous or inhomogeneous kind. Their number of components obviously has to match the shape of $\mathcal{F}$.
The matrix $\mathbf{F}$ is also used to compute the braket $\langle\Phi\,|\,f\,|\,\Phi'\rangle$ efficiently as shown in algorithm 12.

[1] This is in principle a simple but not necessarily obvious computation. We use several fundamental properties of the determinant.

$$\begin{aligned}\det(\mathbf{Q_s}) = \det\left(\left(\sqrt{\mathbf{Q_0}}\right)^{-1}\right) &= \frac{1}{\det\left(\sqrt{\mathbf{Q_0}}\right)} = \frac{1}{\det\left(\sqrt{(\mathbf{Q}\mathbf{Q}^{\mathrm{H}})^{-1}}\right)} = \frac{1}{\sqrt{\det\left((\mathbf{Q}\mathbf{Q}^{\mathrm{H}})^{-1}\right)}} \\ &= \frac{1}{\sqrt{\frac{1}{\det(\mathbf{Q}\mathbf{Q}^{\mathrm{H}})}}} = \frac{1}{\sqrt{\frac{1}{\det\mathbf{Q}\det\mathbf{Q}^{\mathrm{H}}}}} = \sqrt{\det\mathbf{Q}\det\mathbf{Q}^{\mathrm{H}}} = \sqrt{\det\mathbf{Q}\overline{\det\mathbf{Q}}}\end{aligned}$$

Let $\det\mathbf{Q}$ be a complex number z. Then we get:

$$\sqrt{\det\mathbf{Q}\overline{\det\mathbf{Q}}} = \sqrt{z\bar{z}} = \sqrt{|z|^2} = |z| = |\det\mathbf{Q}|$$

When taking square roots one has to take into account branch cuts of the complex plane. This is exactly the reason why we try hard to circumvent this computation at all by omitting the prefactor $\frac{1}{\det\mathbf{Q}}$ when evaluating the functions ϕ.

Algorithm 11 Build the matrix $\mathbf{F}$ of matrix elements of f

Require: Two sets $\Phi := \left\{\phi_{\underline{k}}\right\}_{\underline{k}\in\mathfrak{K}}$ and $\Phi' := \left\{\phi'_{\underline{l}}\right\}_{\underline{l}\in\mathfrak{K}'}$ of basis functions
Require: The parameter sets Π and Π'
Require: The basis shapes $\mathfrak{K}$ and $\mathfrak{K}'$
Require: A scalar-valued function $f(\underline{x})$
Require: A quadrature rule $(\underline{\gamma}_j, \omega_j)$ with R node-weight pairs
// Initialise $\mathbf{F}$ as the zero-matrix
$\mathbf{F} \in \mathbb{C}^{|\mathfrak{K}|\times|\mathfrak{K}'|}, \quad \mathbf{F} := \mathbf{0}$
// Apply the mixing formula from procedure 10 to the parameters
$\underline{q}_0, \mathbf{Q_S} := \mathbf{mix\_parameters}(\Pi, \Pi')$
// Transform the quadrature nodes according to (4.16)
$\underline{\gamma}'_j = \underline{q}_0 + \varepsilon\, \mathbf{Q_S}\, \underline{\gamma}_j \quad j = 0, \ldots R-1$
// Evaluate the basis functions of both Φ with algorithm 7
$\mathbf{B} := \mathbf{evaluate\_basis\_at}[\Phi]\left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1}\right)\right)$
$\mathbf{B}' := \mathbf{evaluate\_basis\_at}[\Phi']\left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1}\right)\right)$
// Evaluate the function f for all quadrature nodes $\underline{\gamma}_j$
$(v_0, \ldots, v_{R-1}) := f\left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1}\right)\right)$
// Iterate over all R quadrature pairs $\left(\underline{\gamma}'_j, \omega_j\right)$
for $j = 0$ **to** $j = R-1$ **do**
  $\mathbf{F} := \mathbf{F} + \varepsilon^D\, v_j\, \omega_j\, \overline{\mathbf{B}[:, j]}\mathbf{B}'[:, j]^{\mathrm{T}}$
end for
return $\mathbf{F}$

Algorithm 12 Efficient computation of $\langle \Phi \,|\, f \,|\, \Phi' \rangle$

Require: Two scalar wavepackets Φ and Φ'
Require: The basis shapes $\mathfrak{K}$ and $\mathfrak{K}'$ with corresponding mappings μ
Require: The scalar function $f(x)$
// Build the matrix $\mathbf{F}$ by algorithm 11
$\mathbf{F} := \mathbf{build\_block\_matrix}(\Phi, \Phi', f)$
// Stack the coefficients $\{c_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}}$ and $\{c'_{\underline{l}}\}_{\underline{l}\in\mathfrak{K}'}$ into vectors
$\underline{c} := \left(c_{\mu_{\mathfrak{K}}(\underline{0})} \cdots c_{\mu_{\mathfrak{K}}(\underline{k})} \cdots\right)^{\mathrm{T}}$
$\underline{c}' := \left(c'_{\mu'_{\mathfrak{K}}(\underline{0})} \cdots c'_{\mu'_{\mathfrak{K}}(\underline{l})} \cdots\right)^{\mathrm{T}}$
// Multiply by the coefficient vectors
$I := \underline{c}^{\mathrm{H}} \mathbf{F} \underline{c}'$
// In case $\Pi \neq \Pi'$ add the global phase
$\pi := \exp\left(\frac{i}{\varepsilon^2}\left(S' - \overline{S}\right)\right)$
return $\pi\, I$

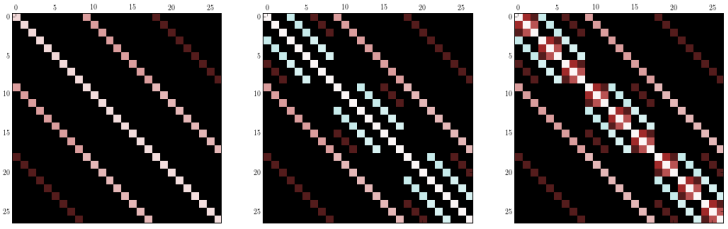


Figure 4.2: The plots show the matrices $\mathbf{F}$ consisting of the matrix elements $\left\langle \phi_{\underline{k}} \middle| V \middle| \phi_{\underline{l}} \right\rangle$ for $V = \frac{1}{2}x^2$ (left), $V = \frac{1}{2}\left(x^2 + y^2\right)$ (middle) and $V = \frac{1}{2}\left(x^2 + y^2 + z^2\right)$ (right). For an explanation of the colours, see appendix B.

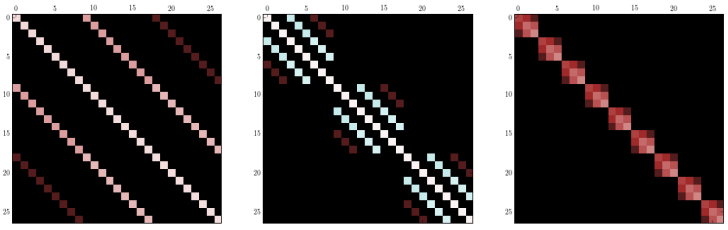


Figure 4.3: The plots show the matrices $\mathbf{F}$ consisting of the matrix elements $\left\langle \phi_{\underline{k}} \middle| V \middle| \phi_{\underline{l}} \right\rangle$ for $V = \frac{1}{2}x^2$ (left), $V = \frac{1}{2}y^2$ (middle) and $V = \frac{1}{2}z^2$ (right). For an explanation of the colours, see appendix B.

Concluding this section we make an artificial example. Assume we have a wavepacket $|\Psi\rangle$ with one single component Φ. Set the parameters to:

$$\underline{q} = \begin{pmatrix} 3.5 \\ -6.5 \\ 1.2 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} -0.5 \\ -2 \\ 3.4 \end{pmatrix} \quad \mathbf{Q} = \mathbb{1} \quad \mathbf{P} = i\mathbb{1} \quad S = 0$$

and take $\varepsilon = 0.9$. We use a $3 \times 3 \times 3$ hypercubic basis shape $\mathfrak{K}$ and set $c_{1,2,0} = 1$ and all other coefficients to zero. Next we compute matrix elements $\mathbf{F}_{\mu(\underline{k}),\mu(\underline{l})} = \left\langle \phi_{\underline{k}} \middle| f \middle| \phi_{\underline{k}} \right\rangle$ for $\underline{k}, \underline{l} \in \mathfrak{K}$ by algorithm 11 (or one of the algorithms 14 or 16). The results for different functions f are shown in figure 4.2 and 4.3. The block structures in the plots are due to the linearisation mapping $\mu_{\mathfrak{K}}$ and the dependence of V on the variables x, y and z.

4.4 Computing norms of wavepackets

In the following we want to compute the norm of a wavepacket $|\Psi\rangle$. This is the most simple observable or braket where the operator in between the bra and the ket is just an identity. We start with:

$$\begin{aligned}\|\Psi\|_{L^2}^2 &= \langle \Psi \,|\, \Psi \rangle \\ &= \sum_{i=0}^{N-1} \langle \Phi_i \,|\, \Phi_i \rangle \\ &= \sum_{i=0}^{N-1} \|\Phi_i\|_{L^2}^2 \,.\end{aligned}$$

Obviously the squared norm of the vector valued wavepacket is the sum of the squared norms of its components. This is a pattern we will find several times more when computing observables. In the general case however, the components may get mixed up by the operator in the middle. Let's continue with calculating the norms of an individual component which can be considered as a scalar wavepacket. Therefore we also drop the index i from above.

$$\begin{aligned}\langle \Phi[\Pi] \,|\, \Phi[\Pi] \rangle &= \left\langle \exp\left(\frac{iS}{\varepsilon^2}\right) \sum_{\underline{k} \in \mathfrak{K}} c_{\underline{k}} \phi_{\underline{k}}[\Pi](\underline{x}) \,\middle|\, \exp\left(\frac{iS}{\varepsilon^2}\right) \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \phi_{\underline{l}}[\Pi](\underline{x}) \right\rangle \\ &= \exp\left(\frac{-iS}{\varepsilon^2}\right) \exp\left(\frac{iS}{\varepsilon^2}\right) \left\langle \sum_{\underline{k} \in \mathfrak{K}} c_{\underline{k}} \phi_{\underline{k}}[\Pi](\underline{x}) \,\middle|\, \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \phi_{\underline{l}}[\Pi](\underline{x}) \right\rangle\end{aligned}$$

The global phase cancels here and we get:

$$= \sum_{\underline{k} \in \mathfrak{K}} \overline{c_{\underline{k}}} \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \left\langle \phi_{\underline{k}}[\Pi](\underline{x}) \,\middle|\, \phi_{\underline{l}}[\Pi](\underline{x}) \right\rangle \,.$$

Now we can exploit the orthonormality condition $\left\langle \phi_{\underline{k}} \,\middle|\, \phi_{\underline{l}} \right\rangle = \delta_{\underline{k},\underline{l}}$ of the basis functions and arrive at:

$$\|\Phi\|_{L^2}^2 = \langle \Phi[\Pi] \,|\, \Phi[\Pi] \rangle = \sum_{\underline{k} \in \mathfrak{K}} \overline{c_{\underline{k}}} c_{\underline{k}} = \sum_{\underline{k} \in \mathfrak{K}} |c_{\underline{k}}|^2 \,. \tag{4.21}$$

4.5 Computing overlap integrals of wavepackets

If we wanted to compute *overlap integrals* between scalar wavepackets, then the procedure is very similar but allows for less simplifications. Notice that now the parameter set Π of $\Phi[\Pi]$ can be and in general will be different. To distinguish them we use Π and Π'.

$$\begin{aligned}\langle \Phi[\Pi] \,|\, \Phi'[\Pi'] \rangle &= \left\langle \exp\left(\frac{iS}{\varepsilon^2}\right) \sum_{\underline{k} \in \mathfrak{K}} c_{\underline{k}} \phi_{\underline{k}}[\Pi](\underline{x}) \,\middle|\, \exp\left(\frac{iS'}{\varepsilon^2}\right) \sum_{\underline{l} \in \mathfrak{K}'} c'_{\underline{l}} \phi_{\underline{l}}[\Pi'](\underline{x}) \right\rangle \\ &= \exp\left(\frac{-iS}{\varepsilon^2}\right) \exp\left(\frac{iS'}{\varepsilon^2}\right) \left\langle \sum_{\underline{k} \in \mathfrak{K}} c_{\underline{k}} \phi_{\underline{k}}[\Pi](\underline{x}) \,\middle|\, \sum_{\underline{l} \in \mathfrak{K}'} c'_{\underline{l}} \phi_{\underline{l}}[\Pi'](\underline{x}) \right\rangle\end{aligned}$$

The global phase does not cancel here and we get:

$$= \exp\left(\frac{-iS}{\varepsilon^2}\right) \exp\left(\frac{iS'}{\varepsilon^2}\right) \sum_{\underline{k} \in \mathfrak{K}} \overline{c_{\underline{k}}} \sum_{\underline{l} \in \mathfrak{K}'} c'_{\underline{l}} \left\langle \phi_{\underline{k}}[\Pi](\underline{x}) \,\middle|\, \phi_{\underline{l}}[\Pi'](\underline{x}) \right\rangle .$$

At the end of the day we get the following formula for the overlap of two scalar wavepackets:

$$\langle \Phi[\Pi] \,|\, \Phi'[\Pi'] \rangle = \exp\left(\frac{i}{\varepsilon^2}(S' - S)\right) \sum_{\underline{k} \in \mathfrak{K}} \sum_{\underline{l} \in \mathfrak{K}'} \overline{c_{\underline{k}}} c'_{\underline{l}} \left\langle \phi_{\underline{k}}[\Pi] \,\middle|\, \phi_{\underline{l}}[\Pi'] \right\rangle . \tag{4.22}$$

This is the most general formula for overlap integrals (which includes computation of norms). We allowed for different Hagedorn parameter sets Π and Π' as well as different basis shapes $\mathfrak{K}$ and $\mathfrak{K}'$. Hereby we conclude this section and continue with the computation of energies in the next one.

4.6 Computing energies of wavepackets

To compute the energies of a wavepacket we start with the full Hamiltonian $\mathbf{H} = \mathbf{T} + \mathbf{V}(\underline{x})$. Then the total energy of $|\Psi\rangle$ is given by:

$$E_{\text{total}} = \langle \Psi \,|\, \mathbf{H} \,|\, \Psi \rangle . \tag{4.23}$$

Next we split this braket into the kinetic energy and the potential energy contribution:

$$E_{\text{total}} = \langle \Psi \,|\, \mathbf{H} \,|\, \Psi \rangle \quad = \langle \Psi \,|\, \mathbf{T} \,|\, \Psi \rangle + \langle \Psi \,|\, \mathbf{V}(\underline{x}) \,|\, \Psi \rangle = E_{\text{kinetic}} + E_{\text{potential}} .$$

4.6.1 Potential energy

We start by computing the potential energy explicitly. Recall the definition of the potential given in equation (1.1). Plugging this matrix $\mathbf{V}(\underline{x})$ into the braket above we get:

$$\begin{aligned}
E_{\text{potential}} &= \langle \Psi \,|\, \mathbf{V}(\underline{x}) \,|\, \Psi \rangle \\
&= \left\langle \begin{pmatrix} \Phi_0(\underline{x}) \\ \vdots \\ \Phi_{N-1}(\underline{x}) \end{pmatrix} \middle| \begin{pmatrix} v_{0,0}(\underline{x}) & \cdots & v_{0,N-1}(\underline{x}) \\ \vdots & & \vdots \\ v_{N-1,0}(\underline{x}) & \cdots & v_{N-1,N-1}(\underline{x}) \end{pmatrix} \begin{pmatrix} \Phi_0(\underline{x}) \\ \vdots \\ \Phi_{N-1}(\underline{x}) \end{pmatrix} \right\rangle \\
&= \left\langle \begin{pmatrix} \Phi_0(\underline{x}) \\ \vdots \\ \Phi_{N-1}(\underline{x}) \end{pmatrix} \middle| \begin{pmatrix} \sum_{i=0}^{N-1} v_{0,i}(\underline{x}) \Phi_i(\underline{x}) \\ \vdots \\ \sum_{i=0}^{N-1} v_{N-1,i}(\underline{x}) \Phi_i(\underline{x}) \end{pmatrix} \right\rangle \\
&= \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} \langle \Phi_j(\underline{x}) \,|\, v_{j,i}(\underline{x}) \Phi_i(\underline{x}) \rangle
\end{aligned}$$

where we expressed the potential energy of the vector-valued wavepacket $|\Psi\rangle$ by a sum of potential energies of its components Φ_i. This is a first step towards our goal.

If we want to compute the potential energy of the wavepacket on each energy surface $\lambda(\underline{x})$ then we have to do the calculation in the eigenbasis shown in equation (1.2). Because the matrix is diagonal now we can obtain a simplified version of the double sum above:

$$\begin{aligned} E_{\text{potential}} &= \langle \Psi \,|\, \mathbf{\Lambda}(\underline{x}) \,|\, \Psi \rangle \\ &= \sum_{i=0}^{N-1} \langle \Phi_i(\underline{x}) \,|\, \lambda_i(\underline{x}) \Phi_i(\underline{x}) \rangle \end{aligned} \tag{4.24}$$

where the braket $\langle \Phi_i(\underline{x}) \,|\, \lambda_i(\underline{x}) \Phi_i(\underline{x}) \rangle$ on the last line is the potential energy of the part Φ_i of $|\Psi\rangle$ residing on the energy surface $\lambda_i(\underline{x})$. Of course we need to transform the wavepacket $|\Psi\rangle$ to the eigenbasis before we can apply the above formula. How to do this is shown in section 4.7. The integral is then approximated by a high-order quadrature as shown at the beginning of this chapter.

4.6.2 Kinetic energy

From the chapter 2 we know that the kinetic operator $\mathbf{T}$ is block-diagonal and does not couple the different components of Ψ. Hence we find that:

$$\begin{aligned} E_{\text{kinetic}} = \langle \Psi \,|\, \mathbf{T} \,|\, \Psi \rangle &= \left\langle \begin{pmatrix} \Phi_0(\underline{x}) \\ \vdots \\ \Phi_{N-1}(\underline{x}) \end{pmatrix} \middle| \begin{pmatrix} T & & 0 \\ & \ddots & \\ 0 & & T \end{pmatrix} \begin{pmatrix} \Phi_0(\underline{x}) \\ \vdots \\ \Phi_{N-1}(\underline{x}) \end{pmatrix} \right\rangle \\ &= \sum_{i=0}^{N-1} \langle \Phi_i(\underline{x}) \,|\, T \,|\, \Phi_i(\underline{x}) \rangle \ . \end{aligned}$$

We can concentrate at the last braket including a scalar wavepacket Φ only. To take the next step we bring to mind that T actually is given by $T = -\frac{1}{2}\varepsilon^4 \Delta$ and therefore we have:

$$\langle \Phi(\underline{x}) \,|\, T \,|\, \Phi(\underline{x}) \rangle = \left\langle \Phi(\underline{x}) \middle| -\frac{1}{2}\varepsilon^4 \Delta \middle| \Phi(\underline{x}) \right\rangle \ .$$

We split the operator into two parts by formally taking the square root of T:

$$= \frac{1}{2} \left\langle \Phi(\underline{x}) \,\middle|\, (-i\varepsilon^2 \nabla)(-i\varepsilon^2 \nabla) \,\middle|\, \Phi(\underline{x}) \right\rangle \ .$$

Then we put the two parts into the bra and the ket respectively

$$\begin{aligned} &= \frac{1}{2} \left\langle +i\varepsilon^2 \nabla \Phi(\underline{x}) \,\middle|\, -i\varepsilon^2 \nabla \Phi(\underline{x}) \right\rangle \\ &= \frac{1}{2} \| -i\varepsilon^2 \nabla \Phi(\underline{x}) \|^2 \, . \end{aligned}$$

The braket simply expresses the squared norm of $-i\varepsilon^2 \nabla \Phi$. If we want to compute this norm we need to know how the operator $y := -i\varepsilon^2 \nabla$ acts on the wavepacket. If we remember correctly, we have done all necessary computation already in section 3.8.

4.7 Basis transformations

This is a last small section before we can go to the time propagation algorithm for wavepackets in the next chapter. In this part we describe the basis transformation of vectorial wavepackets $|\Psi\rangle$ in detail. Recall the definition of the eigenbasis from (1.2) and the basis transformation from (1.4). Then the basis transformation of our wavepacket $|\Psi\rangle$ from and to the canonical basis hence looks like:

$$\begin{aligned} |\Psi_{\text{canonical}}\rangle &= \mathbf{M}(\underline{x})\,|\Psi_{\text{eigen}}\rangle \\ |\Psi_{\text{eigen}}\rangle &= \mathbf{M}^{-1}(\underline{x})\,|\Psi_{\text{canonical}}\rangle = \mathbf{M}^{\mathrm{H}}(\underline{x})\,|\Psi_{\text{canonical}}\rangle \end{aligned} \tag{4.25}$$

and we exploited the fact that all eigenvectors are orthonormal. We can write a more explicit version of $\mathbf{M}$ (we do not reuse 1.5 for notational reasons):

$$\mathbf{M}(\underline{x}) := \begin{pmatrix} m_{0,0}(\underline{x}) & \cdots & m_{0,N-1}(\underline{x}) \\ \vdots & & \vdots \\ m_{N-1,0}(\underline{x}) & \cdots & m_{N-1,N-1}(\underline{x}) \end{pmatrix} .$$

From this the application of $\mathbf{M}$ to $|\Psi\rangle$ becomes obvious:

$$\begin{aligned} \mathbf{M}(\underline{x})\,|\Psi(\underline{x}\rangle &= \begin{pmatrix} m_{0,0}(\underline{x}) & \cdots & m_{0,N-1}(\underline{x}) \\ \vdots & & \vdots \\ m_{N-1,0}(\underline{x}) & \cdots & m_{N-1,N-1}(\underline{x}) \end{pmatrix} \begin{pmatrix} \Phi_0(\underline{x}) \\ \vdots \\ \Phi_{N-1}(\underline{x}) \end{pmatrix} \\ &= \begin{pmatrix} m_{0,0}(\underline{x})\Phi_0(\underline{x}) + \cdots + m_{0,N-1}(\underline{x})\Phi_{N-1}(\underline{x}) \\ \vdots \\ m_{N-1,0}(\underline{x})\Phi_0(\underline{x}) + \cdots + m_{N-1,N-1}(\underline{x})\Phi_{N-1}(\underline{x}) \end{pmatrix} \\ &= \begin{pmatrix} \Phi'_0(\underline{x}) \\ \vdots \\ \Phi'_{N-1}(\underline{x}) \end{pmatrix} = |\Psi'\rangle\ . \end{aligned}$$

4.7.1 Transformation of homogeneous wavepackets

Next we dig deeper and take a detailed look at a single row j of the above matrix vector product:

$$\begin{aligned} m_{j,0}\Phi_0 + \cdots + m_{j,N-1}\Phi_{N-1} &= m_{j,0}\sum_{\underline{k}\in\mathfrak{K}} c^0_{\underline{k}}\phi_{\underline{k}} + \cdots + m_{j,N-1}\sum_{\underline{k}\in\mathfrak{K}} c^{N-1}_{\underline{k}}\phi_{\underline{k}} \\ &= \sum_{\underline{k}\in\mathfrak{K}} m_{j,0}c^0_{\underline{k}}\phi_{\underline{k}} + \cdots + \sum_{\underline{k}\in\mathfrak{K}} m_{j,N-1}c^{N-1}_{\underline{k}}\phi_{\underline{k}} \\ &= \sum_{\underline{k}\in\mathfrak{K}} \left(m_{j,0}c^0_{\underline{k}} + \cdots + m_{j,N-1}c^{N-1}_{\underline{k}} \right) \phi_{\underline{k}} \\ &= \sum_{\underline{k}\in\mathfrak{K}} \underbrace{\sum_{l=0}^{N-1} m_{j,l}c^l_{\underline{k}}}_{c'_{\underline{k}}}\,\phi_{\underline{k}} = \sum_{\underline{k}\in\mathfrak{K}} c'_{\underline{k}}\phi_{\underline{k}} = \Phi'_j\ . \end{aligned}$$

This is a simple calculation but with a big potential for errors. But we are almost done with the most error-prone parts. Since the new coefficients $c'_{\underline{k}}$ depend on $\underline{x}$ through the entries $m_{j,l}(\underline{x})$ of $\mathbf{M}$ we need to project on the subspace spanned by our basis functions $\phi_{\underline{k}}$. This works as follows. First we compute new coefficients:

$$d^j_{\underline{p}} := \left\langle \phi_{\underline{p}} \,\middle|\, \Phi'_j \right\rangle \quad \forall \underline{p} \in \mathfrak{K}$$

and then we build the transformed scalar and vectorial wavepackets step by step as:

$$\Phi''_j(\underline{x}) = \sum_{\underline{k}\in\mathfrak{K}} d^j_{\underline{k}} \phi_{\underline{k}}(\underline{x}) \quad \text{and finally} \quad |\Psi''\rangle = \begin{pmatrix} \Phi''_0 \\ \vdots \\ \Phi''_{N-1} \end{pmatrix}.$$

In principle we are done now. One very last thing to show is the efficient computation of the new coefficients $d^j_{\underline{p}}$. It seems to be complicated but in fact is very simple once we look at a coarser picture. The first step is to rewrite:

$$\begin{aligned} d^j_{\underline{p}} := \left\langle \phi_{\underline{p}} \,\middle|\, \Phi'_j \right\rangle &= \left\langle \phi_{\underline{p}} \,\middle|\, \sum_{\underline{k}\in\mathfrak{K}} \sum_{l=0}^{N-1} m_{j,l} c^l_{\underline{k}} \phi_{\underline{k}} \right\rangle \\ &= \sum_{\underline{k}\in\mathfrak{K}} \sum_{l=0}^{N-1} \left\langle \phi_{\underline{p}} \,\middle|\, m_{j,l}\phi_{\underline{k}} \right\rangle c^l_{\underline{k}} \end{aligned}$$

where we obtained $|\mathfrak{K}|^2$ matrix elements of the form:

$$\left\langle \phi_{\underline{p}} \,\middle|\, m_{j,l} \,\middle|\, \phi_{\underline{k}} \right\rangle = \int \cdots \int_{\mathbb{R}^D} \overline{\phi_{\underline{p}}(\underline{x})}\, m_{j,l}(\underline{x})\, \phi_{\underline{k}}(\underline{x}) \mathrm{dx}\,.$$

Computing every single $d^j_{\underline{p}}$ individually would be very cumbersome and inefficient too. We can do better by noticing that it is possible to rearrange the above expression into a matrix vector product form. Next we stack all $c_{\underline{k}}$ and $d_{\underline{p}}$ into long column vectors of size $|\mathfrak{K}|$. The order is given according to $\mu_{\mathfrak{K}}$. We gather all matrix elements and build the following matrix $\mathbf{F_{j,l}} \in \mathbb{C}^{|\mathfrak{K}|\times|\mathfrak{K}|}$:

$$\mathbf{F_{j,l}} := \begin{pmatrix} & \vdots & \\ \cdots & \left\langle \phi_{\underline{p}} \,\middle|\, m_{j,l} \,\middle|\, \phi_{\underline{k}} \right\rangle & \cdots \\ & \vdots & \end{pmatrix}$$

where we put the integral $\left\langle \phi_{\underline{p}} \,\middle|\, m_{j,l} \,\middle|\, \phi_{\underline{k}} \right\rangle$ at position $\big(\mu(\underline{p}), \mu(\underline{k})\big)$.
Therefore we can now compute $d^l_{\underline{p}}$ for all $\underline{p} \in \mathfrak{K}$ at once by:

$$\begin{pmatrix} d^j_0 \\ \vdots \\ d^j_{|\mathfrak{K}|} \end{pmatrix} = \sum_{l=0}^{N-1} \begin{pmatrix} & \vdots & \\ \cdots & \left\langle \phi_{\underline{p}} \,\middle|\, m_{j,l} \,\middle|\, \phi_{\underline{k}} \right\rangle & \cdots \\ & \vdots & \end{pmatrix} \begin{pmatrix} c^l_0 \\ \vdots \\ c^l_{|\mathfrak{K}|} \end{pmatrix}$$

and thereby express the new coefficients $\underline{d^j}$ of our transformed component Φ_j by a sum of matrix vector products. This can be done very efficiently. Aiming for an even coarser view we can stack the computations for all $j \in (0, \ldots, N-1)$. First we build the block matrix $\mathbf{F} \in \mathbb{C}^{N|\mathfrak{K}| \times N|\mathfrak{K}|}$:

$$\mathbf{F} := \begin{pmatrix} \mathbf{F_{0,0}} & \cdots & \mathbf{F_{0,N-1}} \\ \vdots & \mathbf{F_{j,l}} & \vdots \\ \mathbf{F_{N-1,0}} & \cdots & \mathbf{F_{N-1,N-1}} \end{pmatrix}$$

The matrix can be computed efficiently by algorithm 14. Using the same partition scheme we stack the coefficient vectors $\underline{d^j}$ and $\underline{c^j}$ for all j into even longer column vectors of size $N|\mathfrak{K}|$ (we assume each component having the same basis shape):

$$\underline{c} := \left(\underline{c^0}, \ldots, \underline{c^{N-1}}\right)^{\mathrm{T}} \quad \text{and} \quad \underline{d} := \left(\underline{d^0}, \ldots, \underline{d^{N-1}}\right)^{\mathrm{T}}$$

At the end of the day we can express the whole basis transformation as given in equation (4.25) by a simple matrix vector product. For the transformation to the eigenbasis we get:

$$\underline{c}_{\text{eigen}} = \mathbf{F}\underline{c}_{\text{canonical}} \tag{4.26}$$

and we only need to split up $\underline{c}$ according to the partition used. For the opposite transformation to the canonical basis we can write:

$$\underline{c}_{\text{canonical}} = \mathbf{F}^{\mathrm{H}}\underline{c}_{\text{eigen}} \tag{4.27}$$

since $\mathbf{F}$ is unitary. The only point not shown here is the detailed computation of the matrix elements of $\mathbf{F_{j,l}}$. This is not complicated but a little bit tricky to do efficiently. The integrals are of course approximated by the quadrature rules shown earlier in this chapter. The whole process is presented in [1].

4.7.2 Transformation of inhomogeneous wavepackets

In the case of inhomogeneous wavepackets the whole basis transformation process works similar. There are just a few points where we have to be more general. Again we first look at a single row j of $|\Psi'\rangle = \mathbf{M}\,|\Psi\rangle$. Main point here is to take into account the possibly different parameter sets Π_j and basis shapes $\mathfrak{K}_j$ of each component:

$$\begin{aligned} & m_{j,0}\Phi_0[\Pi_0] + \cdots + m_{j,N-1}\Phi_{N-1}[\Pi_{N-1}] \\ = & \sum_{\underline{k} \in \mathfrak{K}_0} m_{j,0} c^0_{\underline{k}} \phi_{\underline{k}}[\Pi_0] + \cdots + \sum_{\underline{k} \in \mathfrak{K}_{N-1}} m_{j,N-1} c^{N-1}_{\underline{k}} \phi_{\underline{k}}[\Pi_{N-1}] \\ = & \sum_{l=0}^{N-1} \sum_{\underline{k} \in \mathfrak{K}_l} m_{j,l}\, c^l_{\underline{k}}\, \phi_{\underline{k}}[\Pi_l]\,. \end{aligned}$$

Now we enter the projection step with this last expression giving:

$$
\begin{aligned}
d^j_{\underline{p}} &:= \left\langle \phi_{\underline{p}}[\Pi_j] \,\middle|\, \sum_{l=0}^{N-1} \sum_{\underline{k} \in \mathfrak{K}_l} m_{j,l}\, c^l_{\underline{k}}\, \phi_{\underline{k}}[\Pi_l] \right\rangle \\
&= \sum_{l=0}^{N-1} \sum_{\underline{k} \in \mathfrak{K}_l} \left\langle \phi_{\underline{p}}[\Pi_j] \,\middle|\, m_{j,l} \,\middle|\, \phi_{\underline{k}}[\Pi_l] \right\rangle c^l_{\underline{k}} \,.
\end{aligned}
$$

For these brakets on the last line we have to use the inhomogeneous quadrature rule as the basis functions in the bra and the ket can have different parameter sets Π. The matrix $\mathbf{F_{j,l}}$ is of size $|\mathfrak{K}_j| \times |\mathfrak{K}_l|$ and not necessarily square. (Theoretically we could have used different basis shapes $\mathfrak{K}_j$ per component Φ_j already in the homogeneous case.)

$$
\mathbf{F_{j,l}} := \begin{pmatrix} & \vdots & \\ \cdots & \left\langle \phi_{\underline{p}}[\Pi_j] \,\middle|\, m_{j,l} \,\middle|\, \phi_{\underline{k}}[\Pi_l] \right\rangle & \cdots \\ & \vdots & \end{pmatrix}
$$

The block matrix $\mathbf{F}$ is of shape $\sum_{i=0}^{N-1} |\mathfrak{K}_i| \times \sum_{i=0}^{N-1} |\mathfrak{K}_i|$ and obviously square. (This is a requirement because the transformation mapping should be bijective.) We can compute this matrix by algorithm 16. Everything else works exactly the same as in the homogeneous case.

Chapter 5

Wavepacket Propagation

This last theoretical chapter is about time propagation of semi-classical wavepackets. The last remaining missing piece of this puzzle is a good scheme for time propagation of semi-classical Hagedorn wavepackets. Such an algorithm was first described in [5] for the adiabatic one-level case. The generalisation to multiple energy levels was done in [1] but for one space dimension only. These results were published in [4].
In this chapter we review this algorithm once more and show a generalisation to several energy levels in an arbitrary dimensional space.

5.1 Time propagation of Hagedorn wavepackets

The first step is to consider the free particle Schrödinger equation which contains only the kinetic operator T and $V \equiv 0$:

$$i\varepsilon^2 \frac{\partial \Psi}{\partial t} = -\frac{1}{2}\varepsilon^4 \Delta \Psi \,.$$

The proposition 2.1 of [5] tells us that a Hagedorn wavepacket $\Phi[\Pi]$ of the form (3.46) solves the free Schrödinger equation with the following time evolution of its parameter set $\Pi(t)$:

$$\begin{aligned} \underline{q}(t) &= \underline{q}(0) + t\,\mathbf{M}^{-1}\underline{p}(0) \\ \mathbf{Q}(t) &= \mathbf{Q}(0) + t\,\mathbf{M}^{-1}\mathbf{P}(0) \\ S(t) &= S(0) + \frac{1}{2} t\,\underline{p}(0)^{\mathrm{T}}\mathbf{M}^{-1}\underline{p}(0) \end{aligned} \tag{5.1}$$

where $\underline{p}(t) = \underline{p}(0)$ and $\mathbf{P}(t) = \mathbf{P}(0)$ remain unchanged. Also unchanged are the coefficients $\{c_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}}$ of Φ. The matrix $\mathbf{M}$ is the mass scaling matrix. It is set to the identity if nothing else is written. For details see [5, equation 2.8]. A free particle is not that interesting. The important point is that we can solve its quantum equation of motion by only evolving the parameters $\Pi(t)$ using the simple update scheme above.
Next we consider the potential equation lacking the kinetic part:

$$i\varepsilon^2 \frac{\partial \Psi}{\partial t} = U(\underline{x})\Psi \,.$$

For a first approach we assume $U(\underline{x})$ to be quadratic at most. The proposition 2.2 of [5] gives us another set of update rules for $\Pi(t)$:

$$\begin{aligned} \underline{p}(t) &= \underline{p}(0) - t\,\nabla U(\underline{q}(0)) \\ \mathbf{P}(t) &= \mathbf{P}(0) - t\,\nabla^2 U(\underline{q}(0))\mathbf{Q}(0) \\ S(t) &= S(0) - t\,U(\underline{q}(0)) \end{aligned} \tag{5.2}$$

and $\underline{q}(t) = \underline{q}(0)$ and $\mathbf{Q}(t) = \mathbf{Q}(0)$. In this case the coefficients $\{c_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}}$ are left alone. Using the formulae from both sets (5.1) and (5.2) together we can solve the time dependent harmonic oscillator. All we have to do is to propagate in time the parameter set $\Pi(t)$ of $\Phi[\Pi]$. In the whole process we never touch or change the coefficients $\{c_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}}$. This is the reason that makes this time propagation scheme extraordinarily efficient. But this is not really astonishing given that the basis functions $\phi_{\underline{k}}$ are just generalised eigenstates of the harmonic oscillator. (For more details see the paper [8]).
Our next step aims for handling almost arbitrary potentials. But dropping the restriction of U being quadratic provides us with several new hurdles. Assume the potential $V(\underline{x})$ is not quadratic anymore. We can always split V into a quadratic part U and the non-quadratic remainder W:

$$V(\underline{x}) = U(\underline{x}) + W(\underline{x})\,.$$

This is done by a simple Taylor expansion of $V(\underline{x})$ around a point [1] $\underline{q}$ giving:

$$\begin{aligned} U(\underline{x}) &= V(\underline{q}) + \nabla V(\underline{q})(\underline{x}-\underline{q}) + \frac{1}{2}(\underline{x}-\underline{q})^{\mathrm{T}}\nabla^2 V(\underline{q})(\underline{x}-\underline{q}) \\ W(\underline{x}) &= V(\underline{x}) - U(\underline{x})\,. \end{aligned}$$

This expansion allows us to focus on $W(\underline{x})$ solely while assuming that $U(\underline{x})$ is perfectly handled by (5.2). We look at the potential equation:

$$i\varepsilon^2\frac{\partial\Psi}{\partial t} = W(\underline{x})\Psi$$

once more but this time using the non-quadratic remainder part $W(\underline{x})$. Since W can be arbitrarily complicated there is no hope to solve this equation analytically by a simple scheme as we did before. The trick is to perform a Galerkin approximation to solve this equation. To do this we need a Hilbert space of test functions $v(\underline{x})$ defined by:

$$\mathcal{M} := \left\{ v \in \mathrm{L}^2(\mathbb{R}^D)\colon v(\underline{x}) := \sum_{\underline{k}\in\mathfrak{K}} c_{\underline{k}}\phi_{\underline{k}}[\Pi](\underline{x})\,, c_{\underline{k}} \in \mathbb{C} \right\}. \tag{5.3}$$

At this point we may repeat that $\phi_{\underline{k}}$ constitutes a (complete) basis for $\mathrm{L}^2(\mathbb{R}^D)$. The Galerkin approximation (see section 2.4 of [5]) can then be written as:

> At every time t determine $\partial_t u \in \mathcal{M}$ such that
>
> $$\forall \underline{k} \in \mathfrak{K} \quad \langle \phi_{\underline{k}}, (i\varepsilon^2\partial_t - W)u\rangle = 0$$
>
> with $u \in \mathcal{M}$.

[1] It's not a coincidence that we call the expansion point $\underline{q}$ here.

This integral can now be split like:

$$\begin{aligned} 0 &= \langle \phi_{\underline{k}}, (i\varepsilon^2 \partial_t - W)u \rangle \\ &= \langle \phi_{\underline{k}}, i\varepsilon^2 \partial_t u \rangle - \langle \phi_{\underline{k}}, Wu \rangle \end{aligned}$$

and the solution u can in turn be replaced by its basis expansion:

$$\langle \phi_{\underline{k}}, i\varepsilon^2 \partial_t \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \phi_{\underline{l}} \rangle - \langle \phi_{\underline{k}}, W \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \phi_{\underline{l}} \rangle = 0 \,.$$

Next we carry out the differentiation $\partial_t c_{\underline{l}} \phi_{\underline{l}}$. We know that $c_{\underline{l}}(t)$ depends only on time and $\phi_{\underline{l}}[\Pi](\underline{x})$ depends only on space (ignoring the time dependence of Π) and whence we get:

$$\partial_t c_{\underline{l}} \phi_{\underline{l}} = \dot{c}_{\underline{l}} \phi_{\underline{l}} \,.$$

Plugging this into the integrals above we obtain:

$$\langle \phi_{\underline{k}}, i\varepsilon^2 \sum_{\underline{l} \in \mathfrak{K}} \dot{c}_{\underline{l}} \phi_{\underline{l}} \rangle - \langle \phi_{\underline{k}}, W \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \phi_{\underline{l}} \rangle = 0$$

and pulling out the summations and constants yields:

$$i\varepsilon^2 \sum_{\underline{l} \in \mathfrak{K}} \dot{c}_{\underline{l}} \langle \phi_{\underline{k}}, \phi_{\underline{l}} \rangle - \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \langle \phi_{\underline{k}}, W \phi_{\underline{l}} \rangle = 0 \,.$$

The first integral vanishes by orthonormality of the basis functions and what remains is:

$$i\varepsilon^2 \dot{c}_{\underline{k}} = \sum_{\underline{l} \in \mathfrak{K}} c_{\underline{l}} \langle \phi_{\underline{k}}, W \phi_{\underline{l}} \rangle \quad \forall \underline{k} \in \mathfrak{K} \,.$$

We can stack all the $|\mathfrak{K}|$ equations and get the following system of coupled ordinary differential equations:

$$i\varepsilon^2 \begin{pmatrix} \vdots \\ \dot{c}_{\underline{k}} \\ \vdots \end{pmatrix} = \begin{pmatrix} \cdots & \vdots & \cdots \\ & \langle \phi_{\underline{k}}, W \phi_{\underline{l}} \rangle & \\ & \vdots & \end{pmatrix} \begin{pmatrix} \vdots \\ c_{\underline{l}} \\ \vdots \end{pmatrix}$$

or in more compact matrix notation:

$$\dot{\underline{c}} = -\frac{i}{\varepsilon^2} \mathbf{F} \underline{c} \,. \tag{5.4}$$

The solution of this system is trivially given by:

$$\underline{c}(t) = \exp\left(-\frac{i}{\varepsilon^2} t \mathbf{F} \right) \underline{c}(0) \,. \tag{5.5}$$

This was the last missing bit for the time propagation of scalar wavepackets $|\Phi\rangle$ inside arbitrarily shaped potentials $V(\underline{x})$. In algorithm 13 we assembled all the pieces and provide a pseudo code of the implementation.

Algorithm 13 Time propagation of scalar wavepackets $|\Phi\rangle$

Require: A semi-classical wavepacket $|\Phi\left(\underline{x}, t\right)\rangle$
Require: The Hagedorn parameter set Π and the coefficients $\{c_{\underline{k}}\}_{\underline{k} \in \mathfrak{K}}$
Require: The basis shape $\mathfrak{K}$ and the mapping $\mu_{\mathfrak{K}}$
Require: The time step τ
  // Propagate with the kinetic operator
  $\underline{q} := \underline{q} + \frac{\tau}{2} \mathbf{M}^{-1} \underline{p}$
  $\mathbf{Q} := \mathbf{Q} + \frac{\tau}{2} \mathbf{M}^{-1} \mathbf{P}$
  $S := S + \frac{\tau}{4} \underline{p}^{\mathrm{T}} \mathbf{M}^{-1} \underline{p}$
  // Propagate with the local quadratic potential
  $\underline{p} := \underline{p} - \tau \nabla V\left(\underline{q}\right)$
  $\mathbf{P} := \mathbf{P} - \tau \nabla^2 V\left(\underline{q}\right) \mathbf{Q}$
  $S := S - \tau V\left(\underline{q}\right)$
  // Propagate with the non-quadratic remainder W
  // Assemble the matrix $\mathbf{F}$
  $\mathbf{F}_{\mu(\underline{k}), \mu(\underline{l})} := \left\langle \phi_{\underline{k}} \middle| W\left(\underline{x}\right) \middle| \phi_{\underline{l}} \right\rangle \quad \forall \underline{k}, \underline{l} \in \mathfrak{K}$
  // And propagate the coefficients
  $\underline{c} := \exp\left(-\tau \frac{i}{\varepsilon^2} \mathbf{F}\right) \underline{c}$
  // Propagate with the kinetic operator again
  $\underline{q} := \underline{q} + \frac{\tau}{2} \mathbf{M}^{-1} \underline{p}$
  $\mathbf{Q} := \mathbf{Q} + \frac{\tau}{2} \mathbf{M}^{-1} \mathbf{P}$
  $S := S + \frac{\tau}{4} \underline{p}^{\mathrm{T}} \mathbf{M}^{-1} \underline{p}$
  return $|\Phi\left(\underline{x}, t + \tau\right)\rangle$

5.2 Vector valued wavepackets

In the last section we reviewed the time propagation algorithm for scalar semi-classical wavepackets $|\Phi\rangle$ or equivalently for potentials with a single energy level only. For simulations in the non-adiabatic case we need an extended version that handles vectorial wavepackets $|\Psi\rangle$ as defined in (3.61) and (3.62). We examine the case of homogeneous wavepackets first. This has mainly two reasons, namely homogeneous wavepackets play a more important role in practical simulations and we can show the important concepts easier while the inhomogeneous case can be treated by simple generalisation later.

5.2.1 Homogeneous wavepacket propagation

As we defined in (3.61) a homogeneous wavepacket Ψ consists of N components Φ_i all sharing the same parameter set Π. This set of parameters is propagated by the same rules as in the scalar case. The propagation of Π has to take place in the eigenbasis of $\mathbf{V}$ where the energy levels λ_i decouple. We need to choose one of these levels $\lambda_i(\underline{x})$ which then governs the propagation of Π and plays the role of V in the scalar case. We denote this particular level by λ_χ and call $\chi \in [0, \ldots, N-1]$ the *characteristic component index*. Then we apply the usual splitting into quadratic part and non-quadratic remainder. Formally we write:

$$u_\chi(\underline{x}) = \lambda_\chi(\underline{x}) + \nabla\lambda_\chi(\underline{q})(\underline{x}-\underline{q}) + \frac{1}{2}(\underline{x}-\underline{q})^{\mathrm{T}}\nabla^2\lambda_\chi(\underline{q})(\underline{x}-\underline{q})$$
$$w_\chi(\underline{x}) = \lambda_\chi(\underline{x}) - u_\chi(\underline{x})\,.$$

This is sufficient for time propagation of Π but not for the Galerkin step where we propagate the coefficients $\{c_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}}$. There we need a splitting of the full matrix $\mathbf{V}(\underline{x}) = \mathbf{U}(\underline{x}) + \mathbf{W}(\underline{x})$ into quadratic part $\mathbf{U}(\underline{x})$ and remainder $\mathbf{W}(\underline{x})$ such that:

$$\mathbf{V} = \begin{pmatrix} u_\chi & & \\ & \ddots & \\ & & u_\chi \end{pmatrix} + \begin{pmatrix} v_{0,0}-u_\chi & \cdots & v_{0,N-1} \\ \vdots & \ddots & \vdots \\ v_{N-1,0} & \cdots & v_{N-1,N-1}-u_\chi \end{pmatrix}.$$

We will need $\mathbf{W}$ for building the matrix $\mathbf{F}$ later. Notice that up to now we did not mix the different components Φ_i. This only happens during propagation of the coefficients. For this we stack the coefficients $\{c^i_{\underline{k}}\}_{\underline{k}\in\mathfrak{K}_i}$ of all components Φ_i into a long column vector $\underline{c}$:

$$\underline{c} := \begin{pmatrix} \cdots & c^0_{\underline{k}} & \cdots & | & \cdots & | & \cdots & c^{N-1}_{\underline{k}} & \cdots \end{pmatrix}^{\mathrm{T}}$$

of length $\sum_{i=0}^{N-1}|\mathfrak{K}_i|$ or $N|\mathfrak{K}|$ if all components have a basis shape of same size. Then we build the $\mathbf{F}$ matrix for use in equation (5.5). Obviously it must have the following block structure:

$$\mathbf{F} := \begin{pmatrix} \mathbf{F_{0,0}} & \cdots & \mathbf{F_{0,N-1}} \\ \vdots & \mathbf{F_{i,j}} & \vdots \\ \mathbf{F_{N-1,0}} & \cdots & \mathbf{F_{N-1,N-1}} \end{pmatrix}$$

and each block is of the form:

$$\mathbf{F_{i,j}} := \begin{pmatrix} & \vdots & \\ \cdots & \langle \phi_{\underline{k}} \,|\, \mathbf{W}_{i,j} \,|\, \phi_{\underline{l}} \rangle & \cdots \\ & \vdots & \end{pmatrix}$$

for $\underline{k}\in\mathfrak{K}_i$ and $\underline{l}\in\mathfrak{K}_j$. These blocks are not necessarily square but the matrix $\mathbf{F}$ always is. (Compare this to the matrices used for the basis transformation of wavepackets.)
Algorithm 14 gives pseudo code for the computation of $\mathbf{F}$ given a quadrature rule, the remainder $\mathbf{W}$ and the homogeneous wavepacket $\Psi[\Pi]$. The time propagation then is summarised in algorithm 15.

5.2.2 Inhomogeneous wavepacket propagation

The case of inhomogeneous wavepacket propagation is very similar to the homogeneous one presented in the last section. Here we only describe the differences that are necessary. An inhomogeneous wavepacket Ψ consists of N components Φ_i each having its own parameter set Π_i. It is self-evident that now each energy level $\lambda_i(\underline{x})$ is used to propagate the corresponding parameter set Π_i of the

Algorithm 14 Build the homogeneous block matrix $\mathbf{F} := (\mathbf{F}_{\mathbf{r},\mathbf{c}})_{r,c}$

Require: A homogeneous wavepacket Ψ with parameter set Π
Require: The basis shape $\mathfrak{K}_i$ of each component Φ_i of Ψ, $i = 0, \ldots, N-1$
Require: A $N \times N$ matrix $\mathbf{W}(\underline{x})$ of scalar functions $w_{r,c}(\underline{x})$
Require: A quadrature rule $(\underline{\gamma}_j, \omega_j)$ with R node-weight pairs

// Initialise $\mathbf{F}$ as the zero-matrix
$\eta := \sum_{i=0}^{N-1} |\mathfrak{K}_i|$
$\mathbf{F} \in \mathbb{C}^{\eta \times \eta}, \quad \mathbf{F} := \mathbf{0}$
// Transform the quadrature nodes according to (4.8)
$\underline{\gamma}'_j = \underline{q} + \varepsilon \mathbf{Q} \underline{\gamma}_j \quad j = 0, \ldots R-1$
// Evaluate the basis functions of each component Φ_i with algorithm 7
for $i = 0$ **to** $i = N-1$ **do**
  $\mathbf{B_i} := \mathbf{evaluate\_basis\_at}[\Phi_i] \left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1} \right) \right)$
end for
// Iterate over all row and column blocks of this matrix
for $r = 0$ **to** $r = N-1$ **do**
  for $c = 0$ **to** $c = N-1$ **do**
    // Evaluate the function $w_{r,c}$ for all quadrature nodes $\underline{\gamma}_j$
    $(v_0, \ldots, v_{R-1}) := w_{r,c} \left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1} \right) \right)$
    // Set up a zero matrix
    $\mathbf{F}_{\mathbf{r},\mathbf{c}} \in \mathbb{C}^{|\mathfrak{K}_r| \times |\mathfrak{K}_c|}, \quad \mathbf{F}_{\mathbf{r},\mathbf{c}} := \mathbf{0}$
    // Iterate over all R quadrature pairs $\left(\underline{\gamma}'_j, \omega_j \right)$
    for $j = 0$ **to** $j = R-1$ **do**
      $\mathbf{F}_{\mathbf{r},\mathbf{c}} := \mathbf{F}_{\mathbf{r},\mathbf{c}} + \varepsilon^D \, v_j \, \omega_j \, \overline{\mathbf{B}_{\mathbf{r}}[:, j]} \mathbf{B}_{\mathbf{c}}[:, j]^{\mathrm{T}}$
    end for
    // Insert the block $\mathbf{F}_{\mathbf{r},\mathbf{c}}$ into the block matrix $\mathbf{F}$
    $\mathbf{F}_{r,c} := \mathbf{F}_{\mathbf{r},\mathbf{c}}$
  end for
end for
return $\mathbf{F}$

Algorithm 15 Time propagation of a homogeneous wavepacket $|\Psi\rangle$

Require: A semi-classical wavepacket $|\Psi\left(\underline{x}, t\right)\rangle$
Require: The corresponding Hagedorn parameter set Π
Require: For all components $n \in [0, \ldots, N-1]$: the coefficients $\{c^n_{\underline{k}}\}_{\underline{k} \in \mathfrak{K}_n}$,
Require: the basis shapes $\mathfrak{K}_n$ and the mappings $\mu_{\mathfrak{K}_n}$
Require: The leading component index χ
Require: The time step τ
  // Propagate with the kinetic operator
  $\underline{q} := \underline{q} + \frac{\tau}{2}\mathbf{M}^{-1}\underline{p}$
  $\mathbf{Q} := \mathbf{Q} + \frac{\tau}{2}\mathbf{M}^{-1}\mathbf{P}$
  $S := S + \frac{\tau}{4}\underline{p}^{\mathrm{T}}\mathbf{M}^{-1}\underline{p}$
  // Propagate with the local quadratic potential
  $\underline{p} := \underline{p} - \tau\,\nabla\lambda_\chi\left(\underline{q}\right)$
  $\mathbf{P} := \mathbf{P} - \tau\,\nabla^2\lambda_\chi\left(\underline{q}\right)\mathbf{Q}$
  $S := S - \tau\,\lambda_\chi\left(\underline{q}\right)$
  // Propagate with the non-quadratic remainder $\mathbf{W}$
  // Stack the coefficient vectors $\underline{c}^n$ of all components
  $\underline{C} := \left(\underline{c}^0 | \ldots | \underline{c}^{N-1}\right)^{\mathrm{T}}$
  // Assemble the block matrix $\mathbf{F}$ using algorithm 14
  $\Pi' := \{\underline{q}, \underline{p}, \mathbf{Q}, \mathbf{P}, S\}$
  $\mathbf{F} :=$ **build_homogeneous_block_matrix**$(\mathbf{W}, \Psi[\Pi'])$
  // Propagate the coefficients
  $\underline{C} := \exp\left(-\tau\,\frac{i}{\varepsilon^2}\mathbf{F}\right)\underline{C}$
  // Split the coefficients
  $\left(\underline{c}^0 | \ldots | \underline{c}^{N-1}\right) := \underline{C}$
  // Propagate with the kinetic operator again
  $\underline{q} := \underline{q} + \frac{\tau}{2}\mathbf{M}^{-1}\underline{p}$
  $\mathbf{Q} := \mathbf{Q} + \frac{\tau}{2}\mathbf{M}^{-1}\mathbf{P}$
  $S := S + \frac{\tau}{4}\underline{p}^{\mathrm{T}}\mathbf{M}^{-1}\underline{p}$
  return $|\Psi\left(\underline{x}, t + \tau\right)\rangle$

component Φ_i that resides on this level. Whence we have to apply the splitting to each eigenvalue:

$$u_i(\underline{x}) = \lambda_i(\underline{x}) + \nabla\lambda_i(\underline{q}^i)(\underline{x} - \underline{q}^i) + \frac{1}{2}(\underline{x} - \underline{q}^i)^{\mathrm{T}}\nabla^2\lambda_i(\underline{q}^i)(\underline{x} - \underline{q}^i)$$
$$w_i(\underline{x}) = \lambda_i(\underline{x}) - u_i(\underline{x})\,.$$

for all $i \in [0, \ldots, N-1]$ and $\underline{q}^i \in \Pi_i$. This covers all we need to propagate the parameter sets $\{\Pi_i\}_{i=0}^{N-1}$. The splitting of the whole potential matrix $\mathbf{V}$ into $\mathbf{U}$ and $\mathbf{W}$ is straight forward:

$$\mathbf{V} = \begin{pmatrix} u_0 & & \\ & \ddots & \\ & & u_{N-1} \end{pmatrix} + \begin{pmatrix} v_{0,0} - u_0 & \cdots & v_{0,N-1} \\ \vdots & \ddots & \vdots \\ v_{N-1,0} & \cdots & v_{N-1,N-1} - u_{N-1} \end{pmatrix}.$$

Again we use the non-quadratic remainder $\mathbf{W}(\underline{x})$ to compute the block matrix $\mathbf{F}$. The main difference here lies inside the off-diagonal blocks. While building $\mathbf{F_{i,j}}$ with $i \neq j$ the functions $\phi_{\underline{k}}$ and $\phi_{\underline{l}}$ appearing there belong to different families with in general different parameter sets Π_i and Π_j. Therefore the correct formula for these blocks reads:

$$\mathbf{F_{i,j}} := \begin{pmatrix} & \vdots & \\ \cdots & \left\langle \phi_{\underline{k}}[\Pi_i] \,\middle|\, \mathbf{W}_{i,j} \,\middle|\, \phi_{\underline{l}}[\Pi_j] \right\rangle & \cdots \\ & \vdots & \end{pmatrix}$$

where $\underline{k} \in \mathfrak{K}_i$ and $\underline{l} \in \mathfrak{K}_j$. The block is of size $|\mathfrak{K}_i| \times |\mathfrak{K}_j|$. To compute the entries we are required to use an inhomogeneous quadrature rule.
Everything else works exactly the same as in the homogeneous case. Pseudo code for building the block matrix 16 and for the time propagation 17 is presented below for explicit reference.

Algorithm 16 Build the inhomogeneous block matrix $\mathbf{F} := \left(\mathbf{F}_{\mathbf{r},\mathbf{c}}\right)_{r,c}$

Require: An inhomogeneous wavepacket Ψ with N parameter sets Π_i
Require: The basis shape $\mathfrak{K}_i$ of each component Φ_i of Ψ, $i = 0, \ldots, N-1$
Require: A $N \times N$ matrix $\mathbf{W}(\underline{x})$ of scalar functions $w_{r,c}(\underline{x})$
Require: A quadrature rule $(\underline{\gamma}_j, \omega_j)$ with R node-weight pairs
  // Initialise $\mathbf{F}$ as the zero-matrix
  $\eta := \sum_{i=0}^{N-1} |\mathfrak{K}_i|$
  $\mathbf{F} \in \mathbb{C}^{\eta \times \eta}, \quad \mathbf{F} := \mathbf{0}$
  // Iterate over all row and column blocks of this matrix
  for $r = 0$ **to** $r = N-1$ **do**
    for $c = 0$ **to** $c = N-1$ **do**
      // Apply the mixing formula from procedure 10 to the parameters
      $\underline{q_0}, \mathbf{Q_S} := \textbf{mix\_parameters}(\Pi_r, \Pi_c)$
      // Transform the quadrature nodes according to (4.16)
      $\underline{\gamma}'_j = \underline{q_0} + \varepsilon\, \mathbf{Q_S}\, \underline{\gamma}_j \quad j = 0, \ldots R-1$
      // Evaluate the function $w_{r,c}$ for all quadrature nodes $\underline{\gamma}'$
      $(v_0, \ldots, v_{R-1}) := w_{r,c}\left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1}\right)\right)$
      // Evaluate the basis functions for all quadrature nodes $\underline{\gamma}'$
      // Apply algorithm 7 to evaluate the basis of Φ_r and Φ_c
      $\mathbf{B_r} := \textbf{evaluate\_basis\_at}[\Phi_r]\left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1}\right)\right)$
      $\mathbf{B_c} := \textbf{evaluate\_basis\_at}[\Phi_c]\left(\left(\underline{\gamma}'_0, \ldots, \underline{\gamma}'_{R-1}\right)\right)$
      // Do not forget the non-vanishing phase
      $\pi_{r,c} := \exp\left(\frac{i}{\varepsilon^2}\left(S_c - \overline{S_r}\right)\right)$
      // Set up a zero matrix
      $\mathbf{F}_{\mathbf{r},\mathbf{c}} \in \mathbb{C}^{|\mathfrak{K}_r| \times |\mathfrak{K}_c|}, \quad \mathbf{F}_{\mathbf{r},\mathbf{c}} := \mathbf{0}$
      // Iterate over all R quadrature pairs $\left(\underline{\gamma}'_j, \omega_j\right)$
      for $j = 0$ **to** $j = R-1$ **do**
        $\mathbf{F}_{\mathbf{r},\mathbf{c}} := \mathbf{F}_{\mathbf{r},\mathbf{c}} + \varepsilon^D\, v_j\, \omega_j\, \det\left(\mathbf{Q_S}\right)\, \overline{\mathbf{B_r}[:,j]} \mathbf{B_c}[:,j]^{\mathrm{T}}$
      end for
      // Insert the block $\mathbf{F}_{\mathbf{r},\mathbf{c}}$ into the block matrix $\mathbf{F}$
      $\mathbf{F}_{r,c} := \pi_{r,c}\, \mathbf{F}_{\mathbf{r},\mathbf{c}}$
    end for
  end for
  return $\mathbf{F}$

Algorithm 17 Time propagation of an inhomogeneous wavepacket $|\Psi\rangle$

Require: A semi-classical wavepacket $|\Psi\left(\underline{x}, t\right)\rangle$
Require: For all components $n \in [0, \ldots, N-1]$:
Require: the Hagedorn parameter sets Π_n, the coefficients $\{c^n_{\underline{k}}\}_{\underline{k} \in \mathfrak{K}_n}$
Require: the basis shapes $\mathfrak{K}_n$ and the mappings $\mu_{\mathfrak{K}_n}$
Require: The time step τ
  // Propagate with the kinetic operator
  for $n = 0$ **to** $n = N-1$ **do**
    $\underline{q}_n := \underline{q}_n + \frac{\tau}{2}\mathbf{M}^{-1}\underline{p}_n$
    $\mathbf{Q}_n := \mathbf{Q}_n + \frac{\tau}{2}\mathbf{M}^{-1}\mathbf{P}_n$
    $S_n := S_n + \frac{\tau}{4}\underline{p}_n{}^{\mathrm{T}}\mathbf{M}^{-1}\underline{p}_n$
  end for
  // Propagate with the local quadratic potential
  for $n = 0$ **to** $n = N-1$ **do**
    $\underline{p}_n := \underline{p}_n - \tau\,\nabla\lambda_n\left(\underline{q}_n\right)$
    $\mathbf{P}_n := \mathbf{P}_n - \tau\,\nabla^2\lambda_n\left(\underline{q}_n\right)\mathbf{Q}_n$
    $S_n := S_n - \tau\,\lambda_n\left(\underline{q}_n\right)$
  end for
  // Propagate with the non-quadratic remainder
  // Stack the coefficient vectors $\underline{c}^n$ of all components
  $\underline{C} := \left(\underline{c}^0 | \ldots | \underline{c}^{N-1}\right)^{\mathrm{T}}$
  // Assemble the matrix $\mathbf{F}$ using algorithm 16
  $\Pi'_i := \{\underline{q}_n, \underline{p}_n, \mathbf{Q}_n, \mathbf{P}_n, S_n\} \quad \forall n = 0, \ldots, N-1$
  $\mathbf{F} :=$ **build_inhomogeneous_block_matrix**$(\mathbf{W}, \Psi[\Pi'_0, \ldots, \Pi'_{N-1}])$
  // Propagate the coefficients
  $\underline{C} := \exp\left(-\tau\,\frac{i}{\varepsilon^2}\mathbf{F}\right)\underline{C}$
  // Split the coefficients
  $\left(\underline{c}^0 | \ldots | \underline{c}^{N-1}\right) := \underline{C}$
  // Propagate with the kinetic operator again
  for $n = 0$ **to** $n = N-1$ **do**
    $\underline{q}_n := \underline{q}_n + \frac{\tau}{2}\mathbf{M}^{-1}\underline{p}_n$
    $\mathbf{Q}_n := \mathbf{Q}_n + \frac{\tau}{2}\mathbf{M}^{-1}\mathbf{P}_n$
    $S_n := S_n + \frac{\tau}{4}\underline{p}_n{}^{\mathrm{T}}\mathbf{M}^{-1}\underline{p}_n$
  end for
  return $|\Psi\left(\underline{x}, t+\tau\right)\rangle$

Chapter 6

Simulation Results

This chapter contains the results of some selected examples. First we show that the code works correctly by simulating the well known toy examples like harmonic oscillators in different settings. Next we reproduce some less trivial results from [5] in two space dimensions. Finally we show several results of non-adiabatic transition in two and more space dimensions.
All simulations were performed with our WaveBlocksND simulation code[3]. The code can perform simulations with an arbitrary number N of energy levels defined in an arbitrary number D of space dimensions. The only limitations are available computer memory and time.

6.1 Harmonic oscillators

The harmonic oscillator is the first toy example one usually studies to check if a new simulation code works as expected. This is also what we are going to do next. We start with a Gaussian wavepacket having the parameters:

$$\underline{q} = \begin{pmatrix} 1.8 \\ 1.2 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0.6 \\ 0.8 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0\,.$$

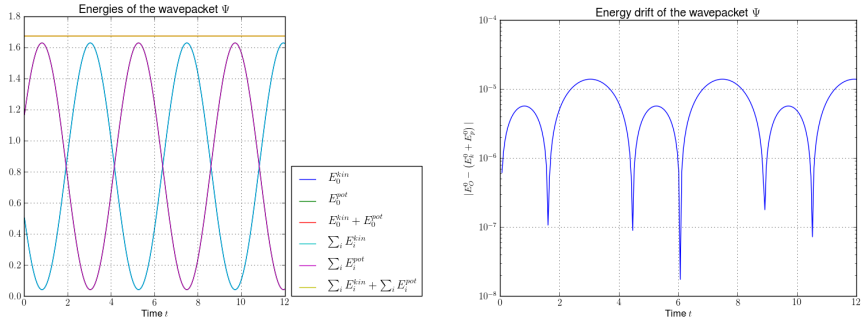
The scaling parameter ε is set to 0.1. These values for Π result in a cyclic oscillation with elliptical orbits, see figure 6.3a. The potential is given by:

$$V(x,y) := \frac{1}{2}\left(\frac{1}{2}x^2 + \frac{1}{2}y^2\right)\,. \tag{6.1}$$

Another trivial but nice example is the combination of a free particle potential with a harmonic oscillator. The potential is constant along one direction while quadratic along the other, we get:

$$V(x,y) := \frac{1}{4}y^2\,. \tag{6.2}$$

The following simulation results were obtained by starting with a Gaussian ϕ_0 with parameters:



(a) (b)

Figure 6.1: The kinetic, potential and total energy and the drift of the total energy of a wavepacket in a two-dimensional harmonic oscillator. (a) The energies. (b) The energy drift.

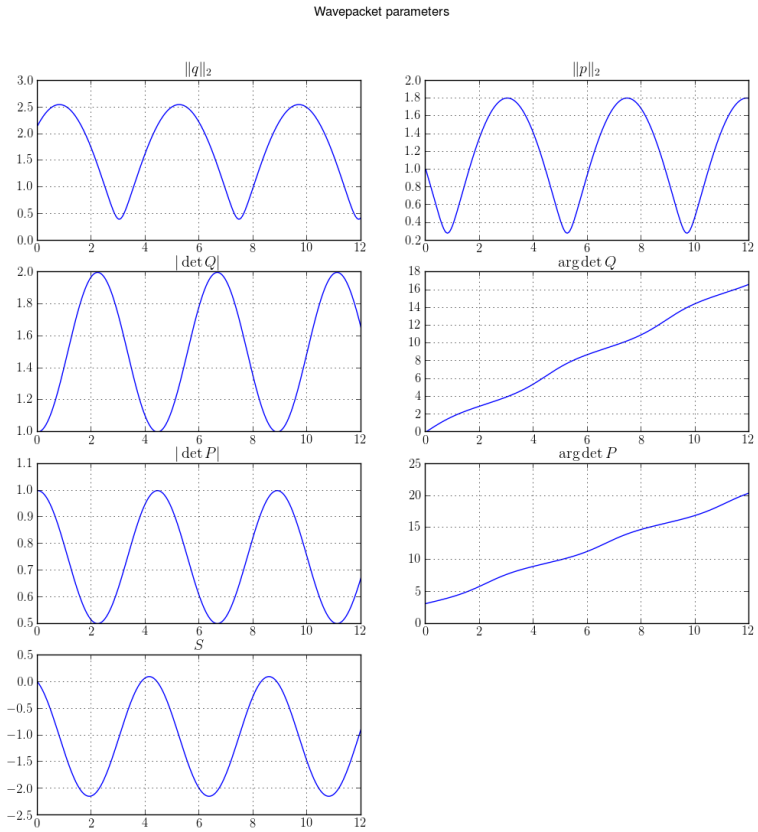
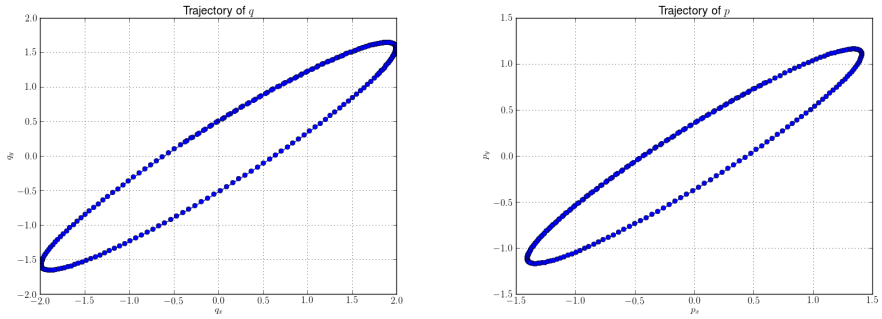
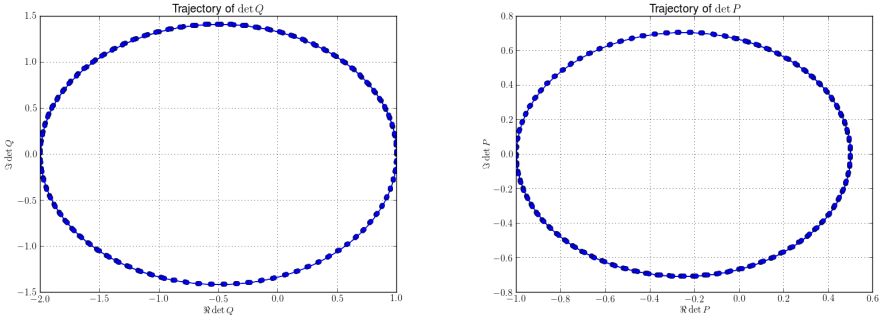


Figure 6.2: Time-evolution of the parameter set Π of a wavepacket in a two-dimensional harmonic oscillator.



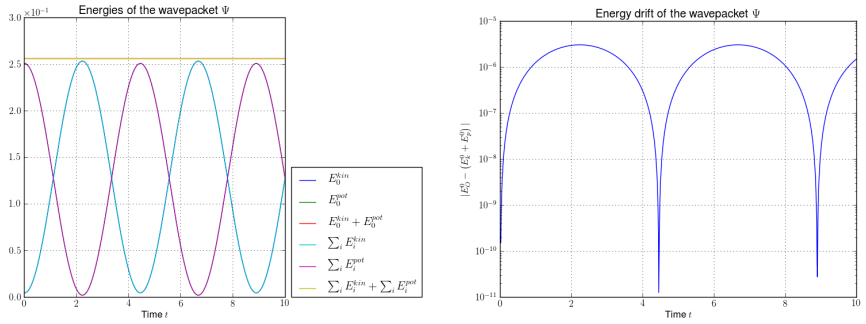
(a) (b)

Figure 6.3: Trajectories of the parameters $\underline{q}$ and $\underline{p}$. (a) Trajectory of $\underline{q}$. (b) Trajectory of $\underline{p}$.



(a) (b)

Figure 6.4: Trajectories of $\det \mathbf{Q}$ and $\det \mathbf{P}$ in the complex plane. (a) Trajectory of $\det \mathbf{Q}$. (b) Trajectory of $\det \mathbf{P}$.



(a) (b)

Figure 6.5: The kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket in the potential in (6.2). (a) The energies. (b) The energy drift.

$$\underline{q} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0$$

and $\varepsilon = 0.1$.
An interesting fact appears if we look at the two eigenvalues λ_0 and λ_1 of $\mathbf{Q}$. We see that one oscillates and the other grows ad infinitum. Figure 6.8 shows plots of $\lambda_0(t)$ and $\lambda_1(t)$.

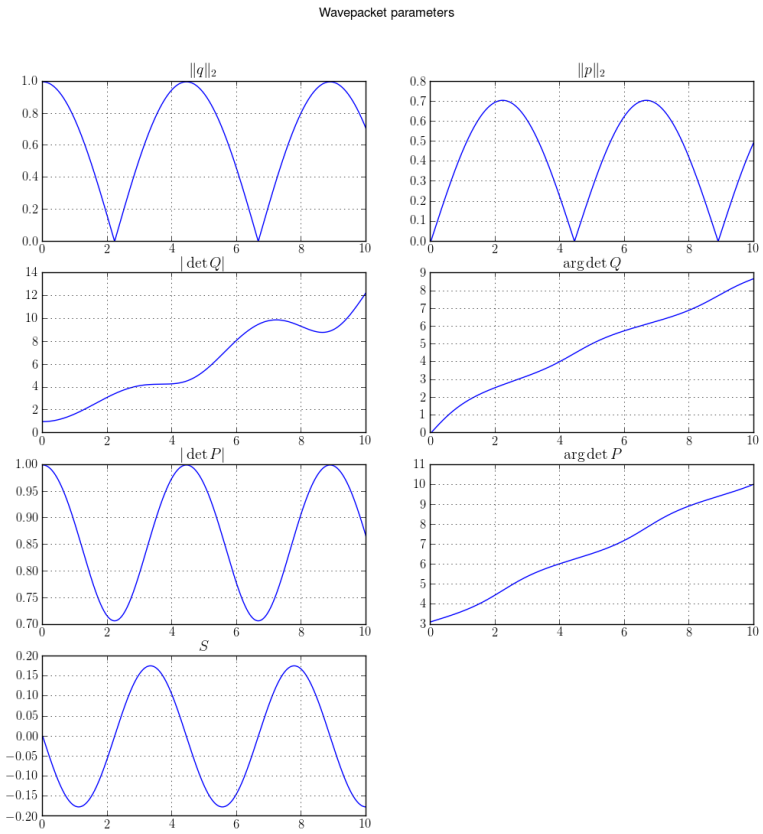
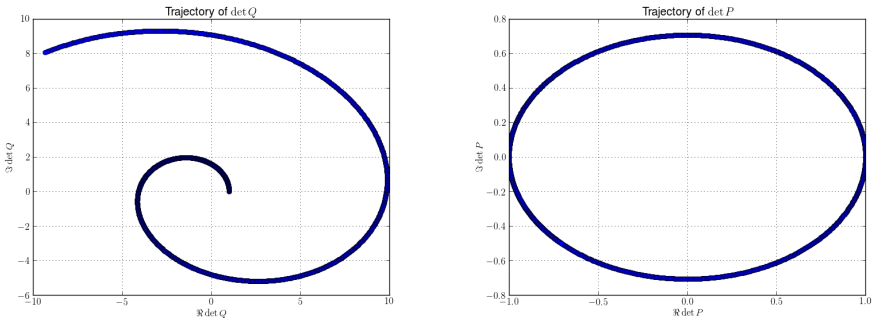
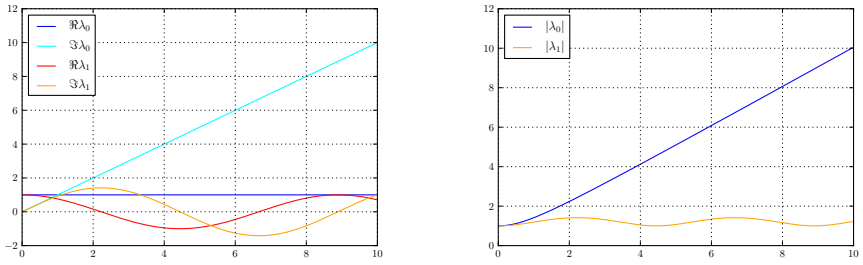


Figure 6.6: Time-evolution of the parameter set Π of a Gaussian wavepacket in the potential in (6.2).



(a) (b)

Figure 6.7: Trajectories of $\det \mathbf{Q}$ and $\det \mathbf{P}$ in the complex plane. (a) Trajectory of $\det \mathbf{Q}$. (b) Trajectory of $\det \mathbf{P}$.



(a) (b)

Figure 6.8: Time-evolution of the eigenvalues λ_0 and λ_1 of $\mathbf{Q}$. (a) Real and imaginary parts. (b) Absolute values.

6.2 Reproducing some other results

We try to reproduce the results in section 5.1 of [5]. The torsional potential in two dimensions is given by the expression:

$$V(x, y) := (1 - \cos(x)) + (1 - \cos(y)) \tag{6.3}$$

and plotted in figure 6.9.
The initial parameter set $\Pi = \{\underline{q}, \underline{p}, \mathbf{Q}, \mathbf{P}, S\}$ for the wavepacket $\Psi = \phi_{0,0}$ is given as:

$$\underline{q} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0\,.$$

We use a hyperbolic cut basis shape $\mathfrak{K}$ with a cutoff value of $K = 8$. This yields 20 basis functions in total. For each simulation we use a time step $\tau = 0.01$ and simulate until an end time of $T = 20$. We perform three simulations for different

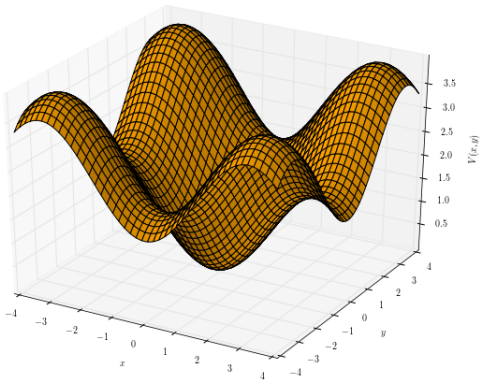


Figure 6.9: The torsional potential in two dimensions.

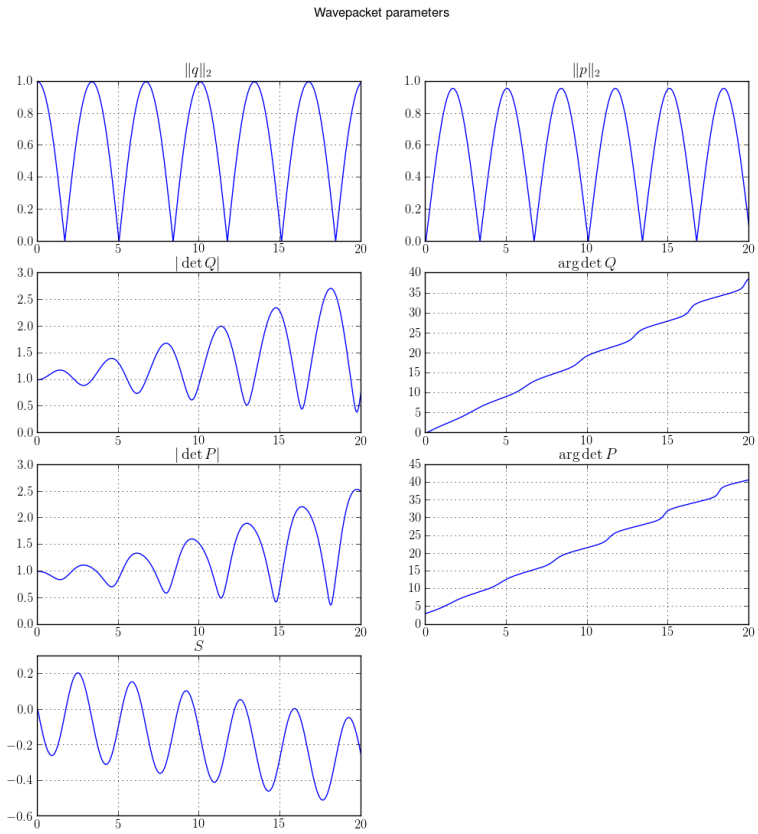


Figure 6.10: Propagation of the parameter set Π. This is the same for all ε.

values of the semi-classical scaling parameter ε [1].

[1] Note that we write ε^2 for what the authors of the paper call ϵ.

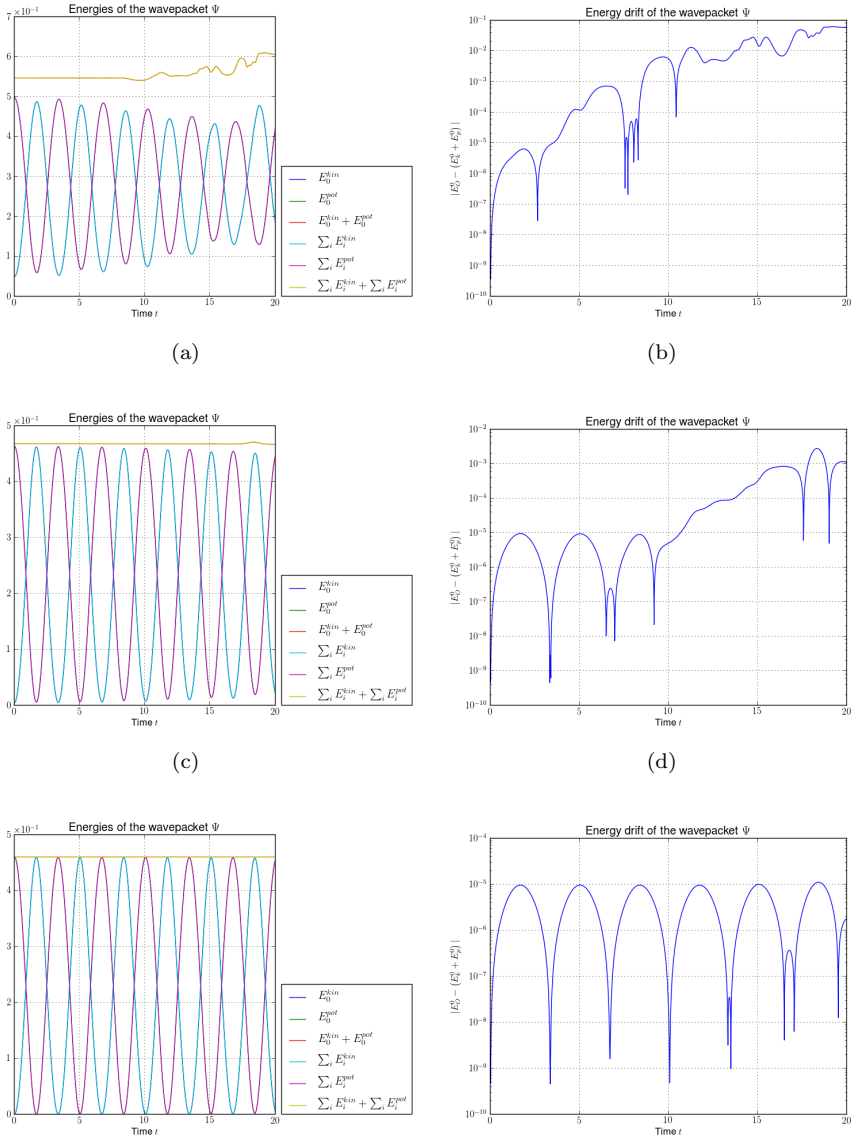


Figure 6.11: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket in a 2D torsional potential. Note that despite of some violation of the total energy conservation we obtained perfect norm conservation (not shown here). A larger basis set would reduce the error. (a), (b) A wavepacket $|\Psi\rangle = \phi_{0,0}$ with $\varepsilon = \sqrt{0.1}$. (c), (d) A wavepacket $|\Psi\rangle = \phi_{0,0}$ with $\varepsilon = \sqrt{0.01}$. (e), (f) A wavepacket $|\Psi\rangle = \phi_{0,0}$ with $\varepsilon = \sqrt{0.001}$.

6.3 A simple avoided crossing

We generalise a model for a single avoided crossing of two energy levels. The one-dimensional potential was studied in [5, 1]. We extend this potential to two dimensions by making it rotationally symmetric. The equation for $\mathbf{V}$ becomes:

$$\mathbf{V}(x,y) := \begin{pmatrix} \frac{1}{2}\tanh\left(\sqrt{x^2+y^2}\right) & \delta \\ \delta & -\frac{1}{2}\tanh\left(\sqrt{x^2+y^2}\right) \end{pmatrix} \tag{6.4}$$

with δ being half of the energy level gap. The potential is shown in figure 6.12.

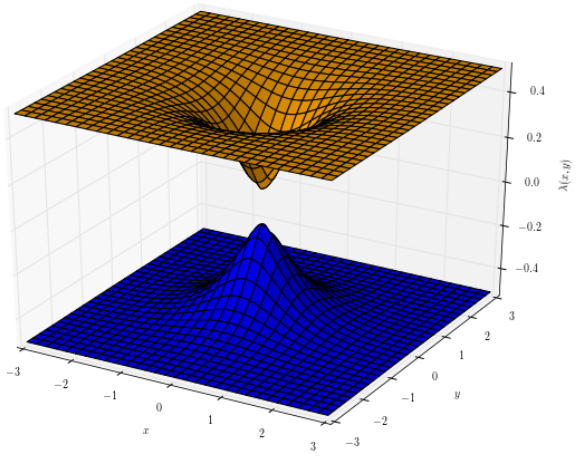


Figure 6.12: Energy levels of the avoided crossing given by equation (6.4) for $\delta = 0.08$.

In the following simulation results we varied the value of ε and the gap size δ. The initial parameter values are:

$$\underline{q} = \begin{pmatrix} -3 \\ 0 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0\,.$$

The timestep was set to $\tau = 0.01$.
We see how for larger ε the energy conservation becomes worse. This is a sign of a too small basis shape $\mathfrak{K}$ which can not capture the emergence of more and more quantum effects. The plots of the norms show that we get higher transition probabilities for smaller gaps δ. Additionally we get faster transitions for smaller ε. However we have to be careful interpreting these results because in most simulations there is no energy conservation. Maybe a smaller timestep τ would be appropriate in some cases.

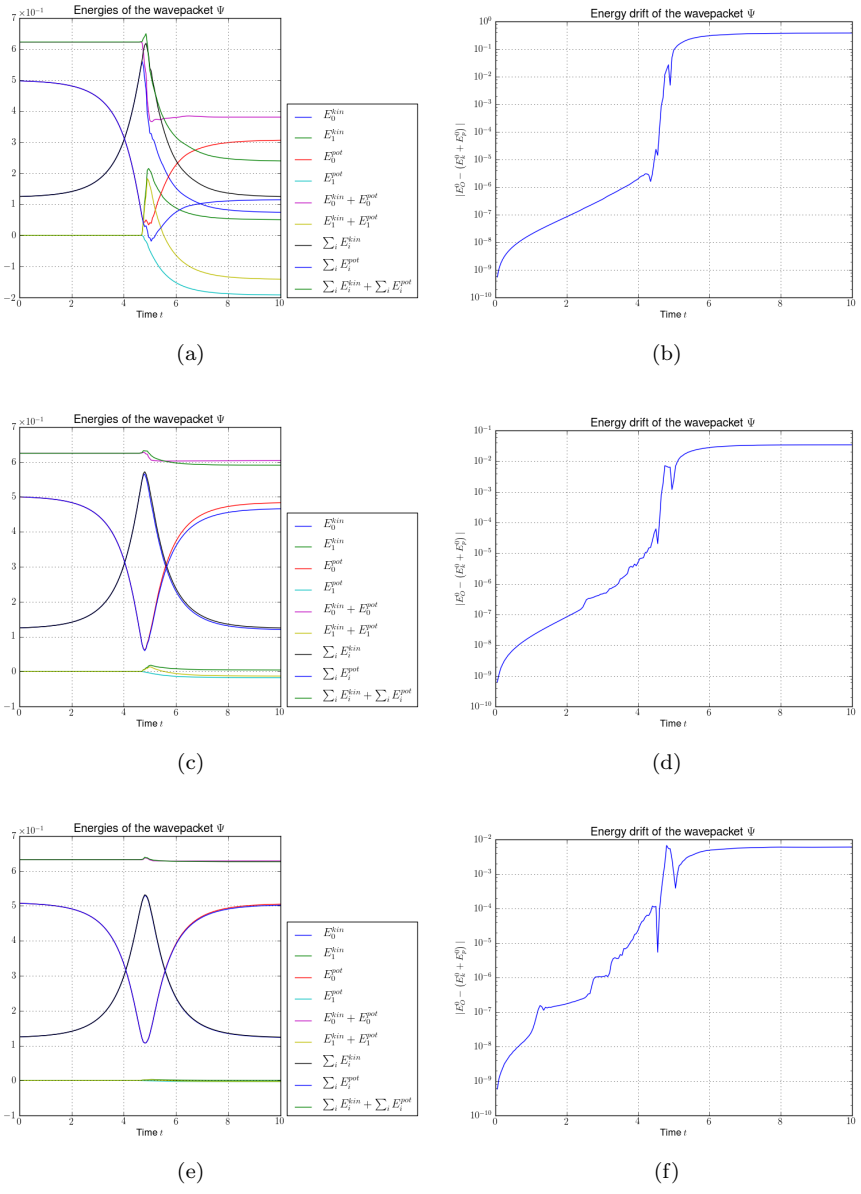


Figure 6.13: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. The simulations for $\delta = 0.01$ would need a larger basis to work properly. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$

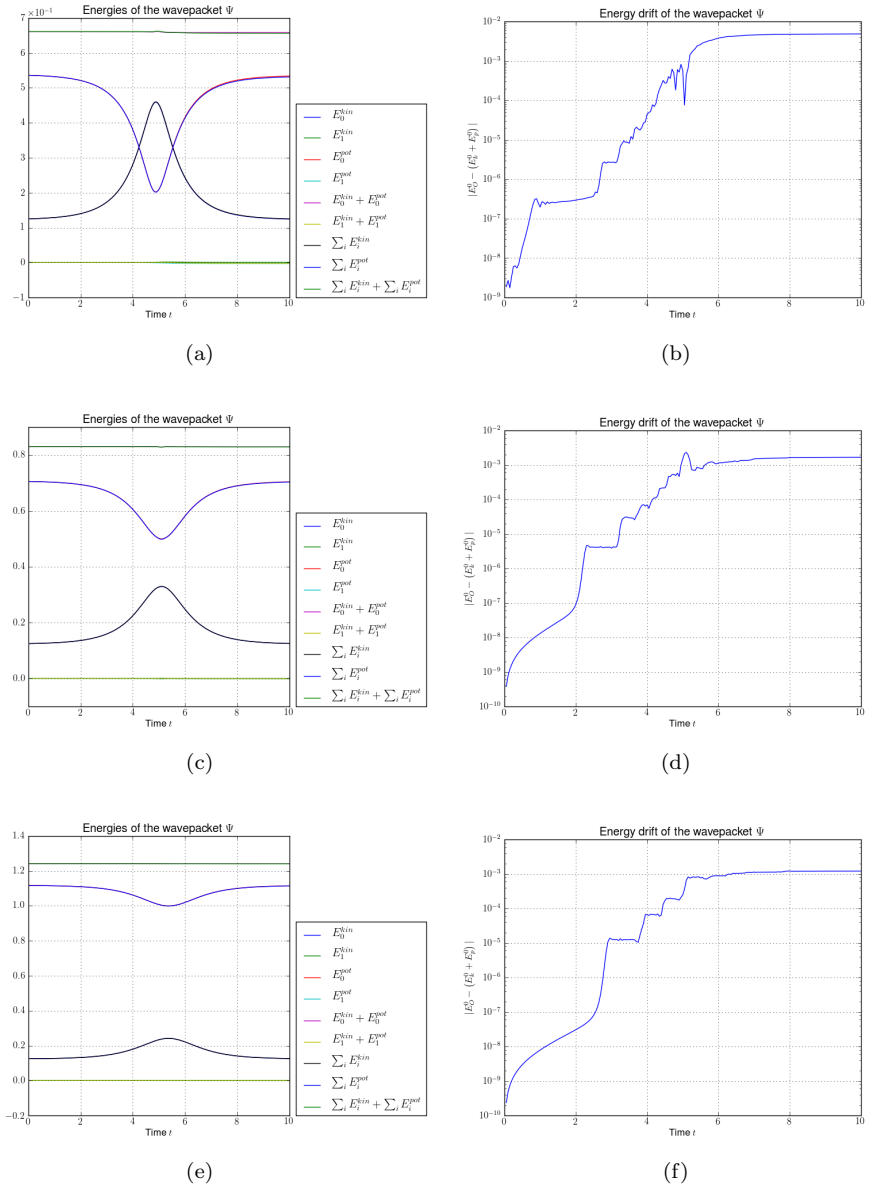
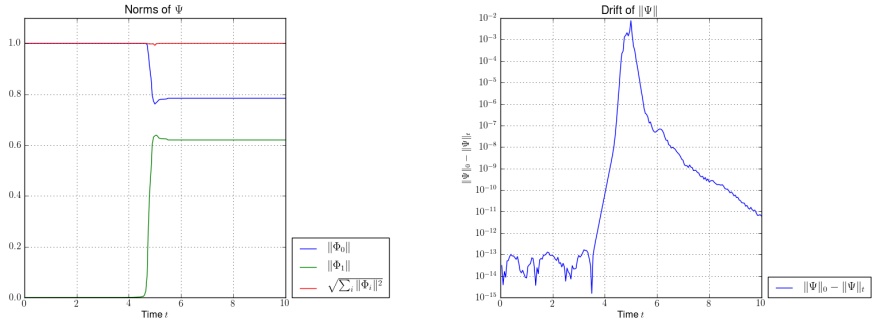
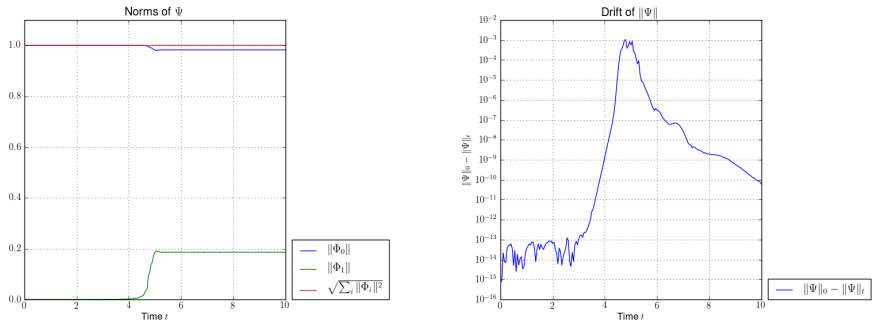


Figure 6.14: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$



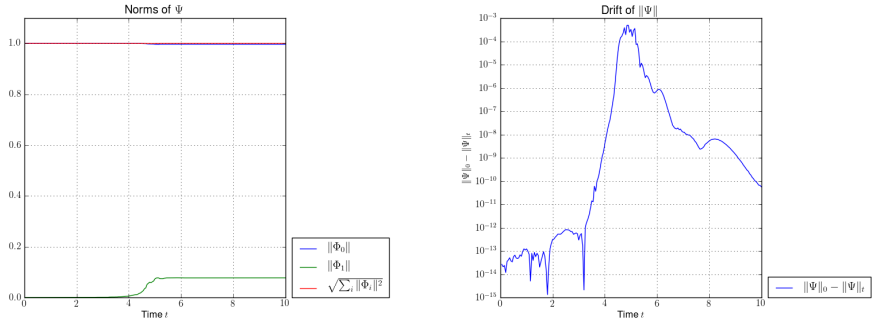
(a)

(b)



(c)

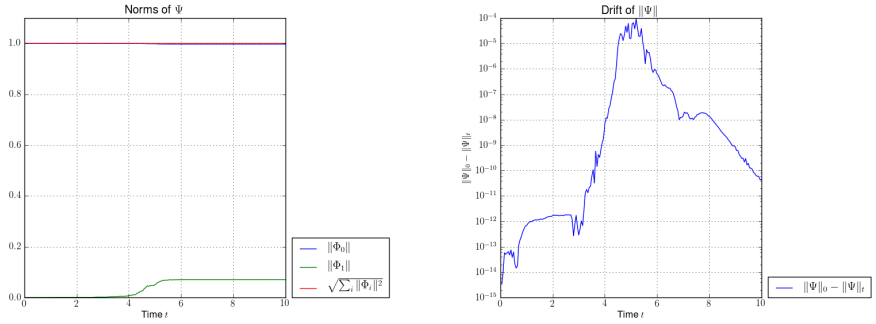
(d)



(e)

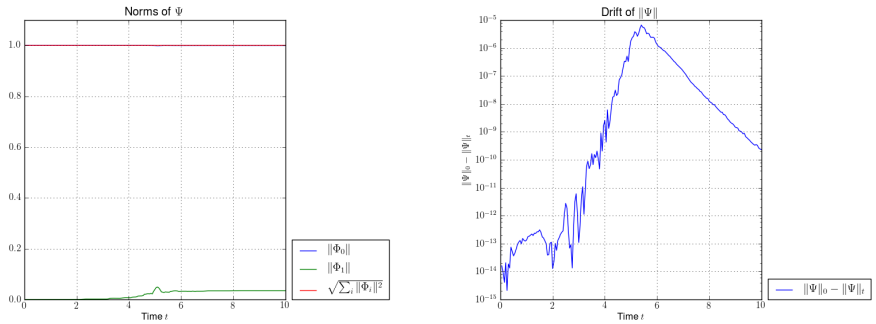
(f)

Figure 6.15: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



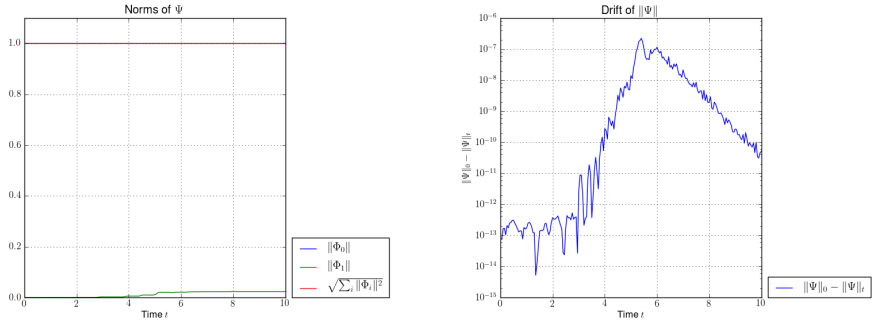
(a)

(b)



(c)

(d)



(e)

(f)

Figure 6.16: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

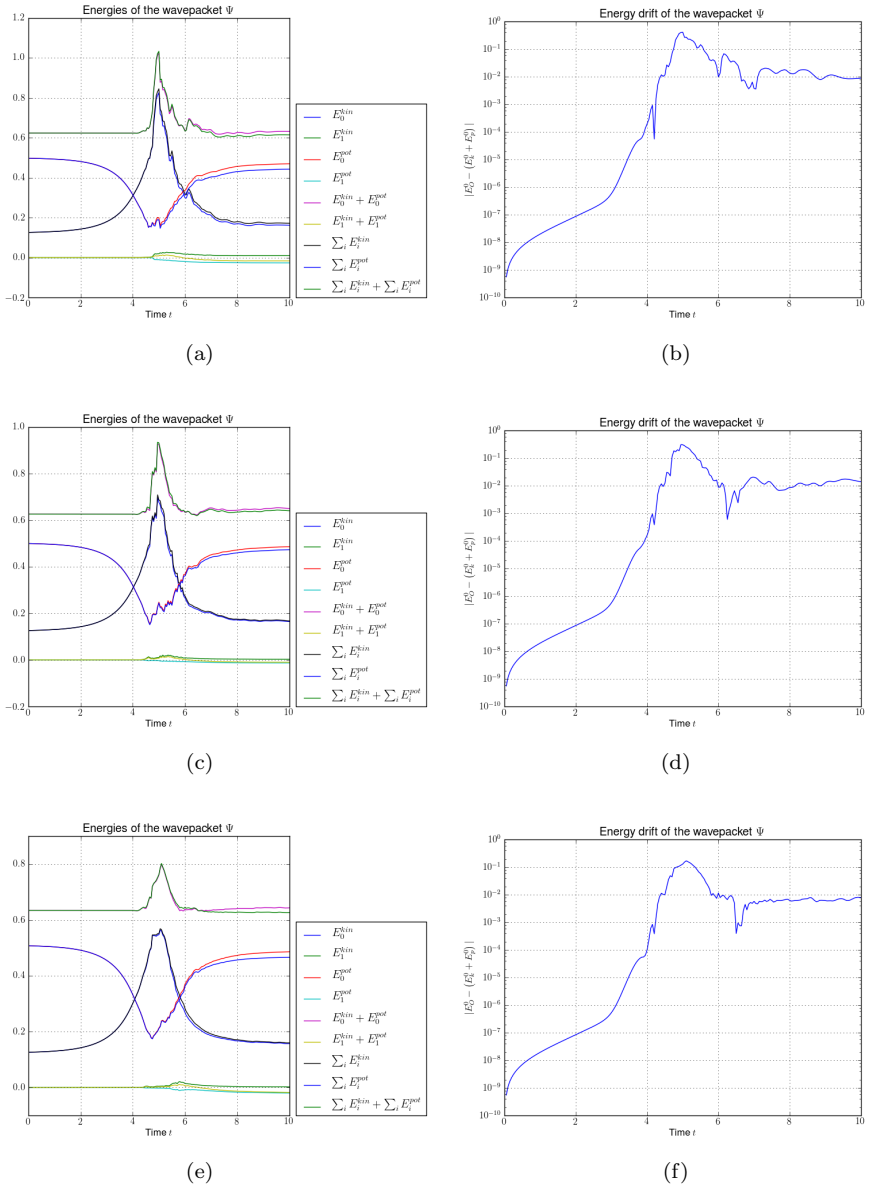


Figure 6.17: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.05$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$

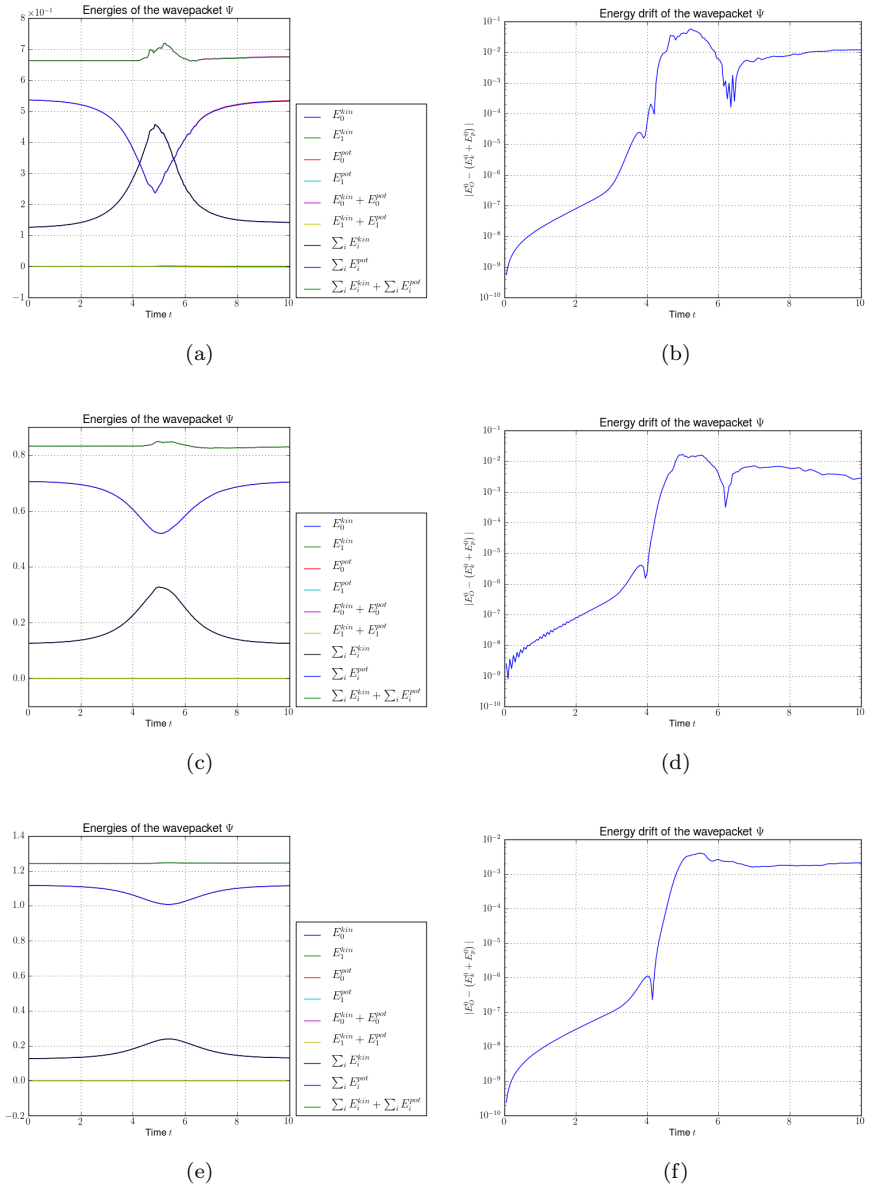
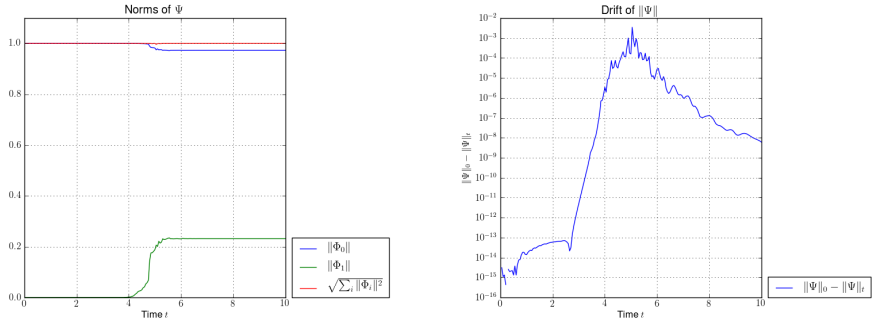
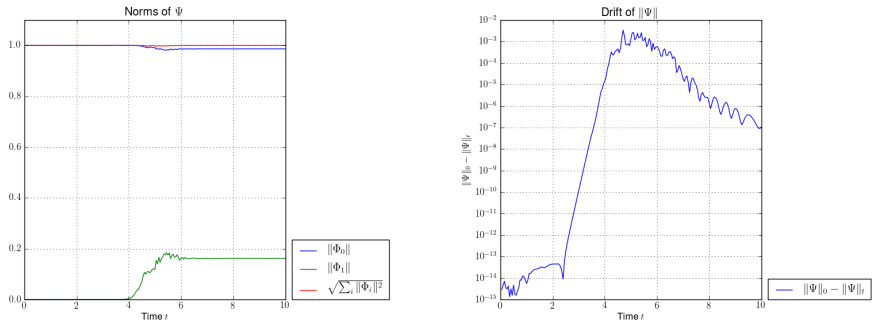


Figure 6.18: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.05$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$



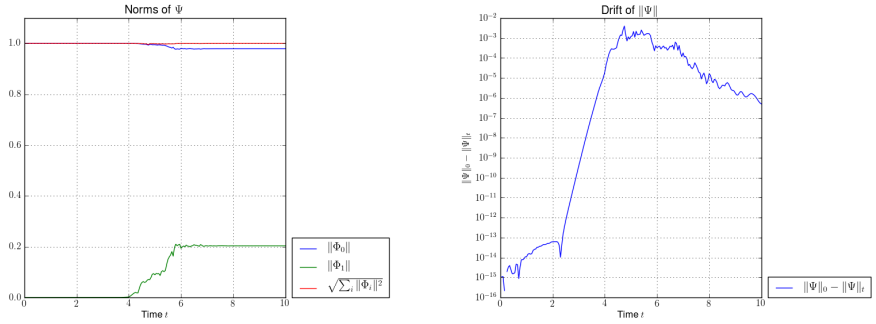
(a)

(b)



(c)

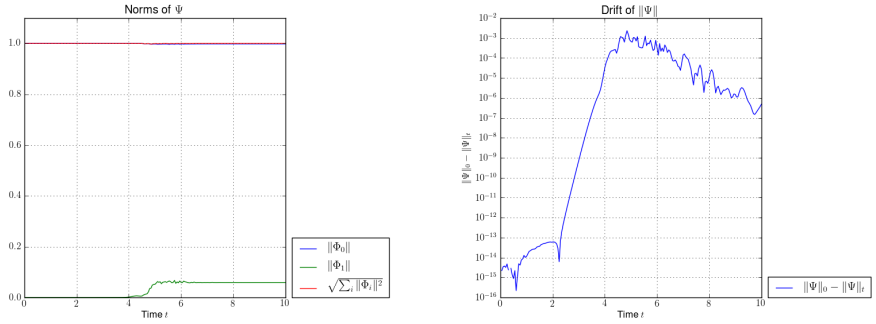
(d)



(e)

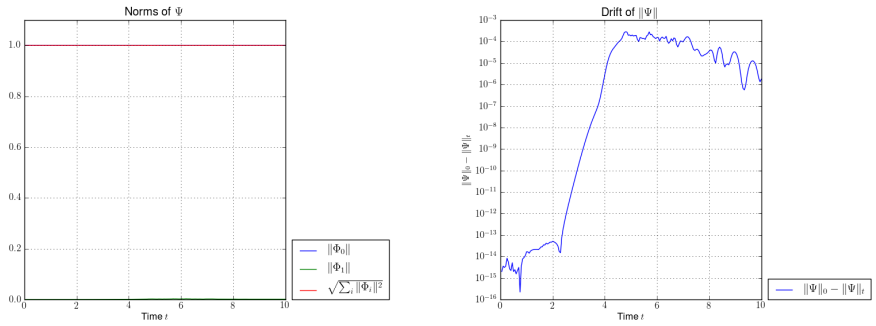
(f)

Figure 6.19: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.05$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



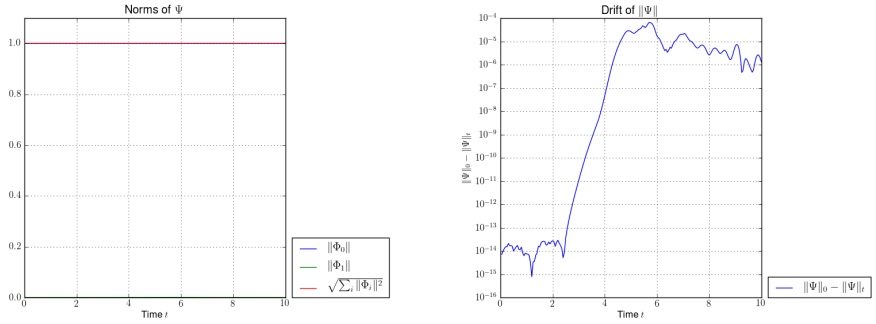
(a)

(b)



(c)

(d)



(e)

(f)

Figure 6.20: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.05$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

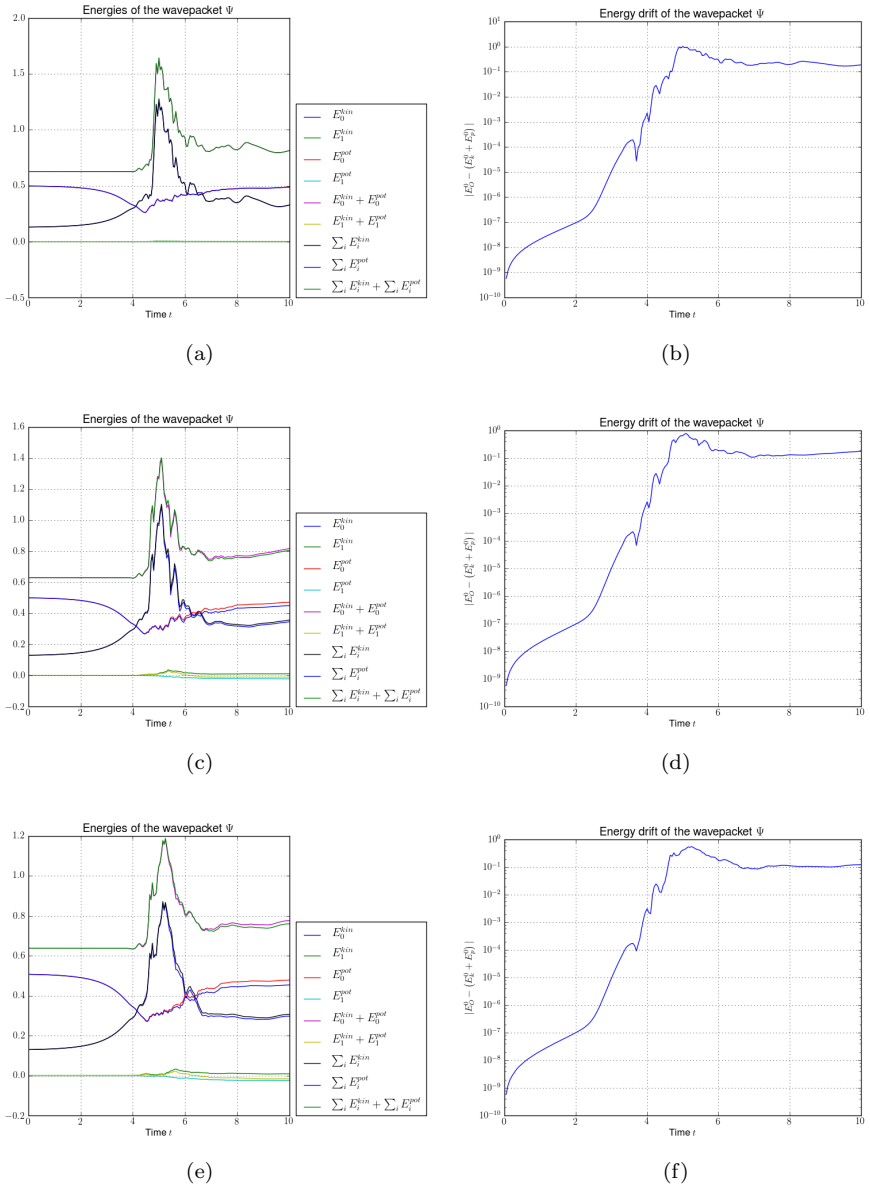
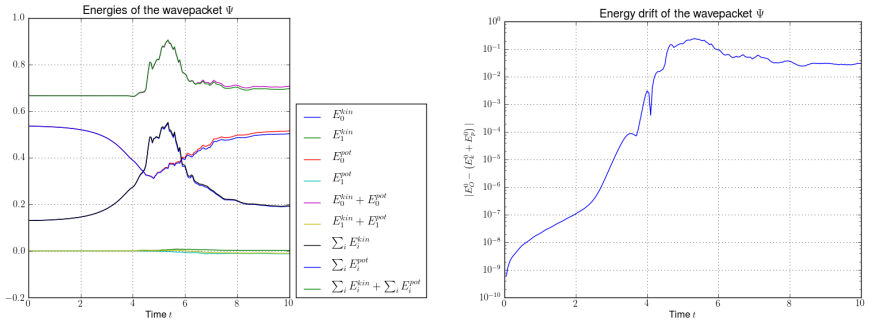
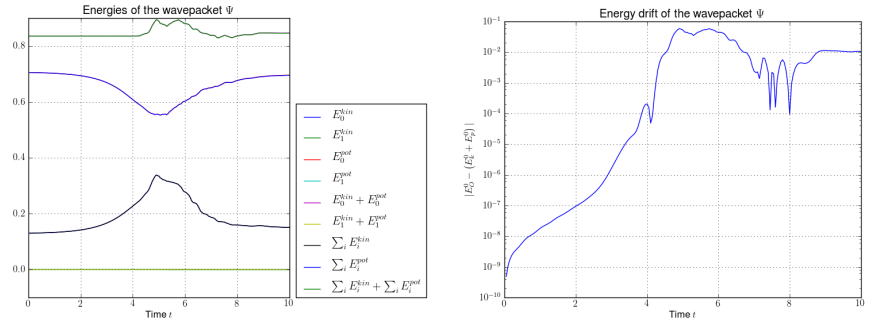


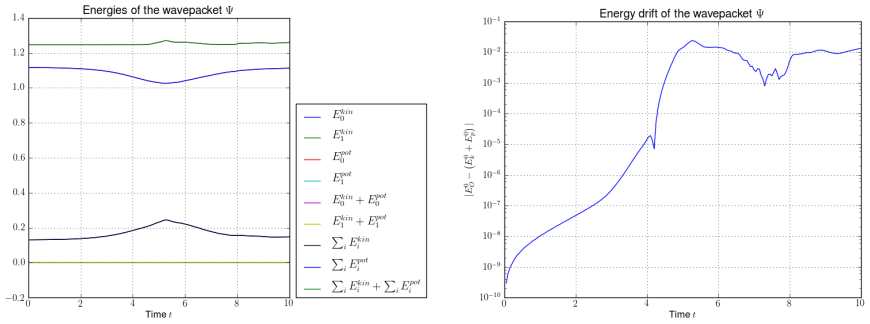
Figure 6.21: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.1$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



(a) (b)

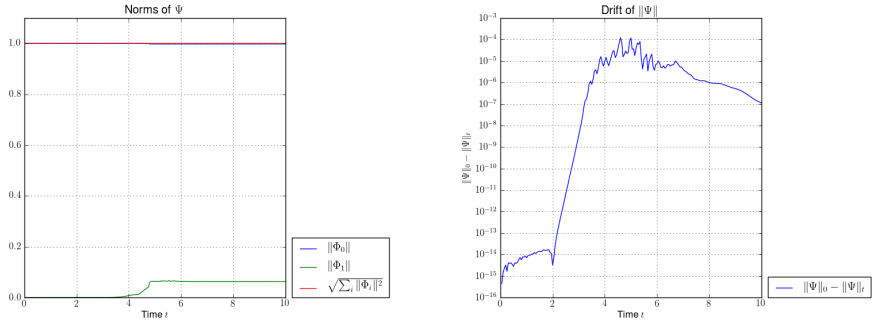


(c) (d)



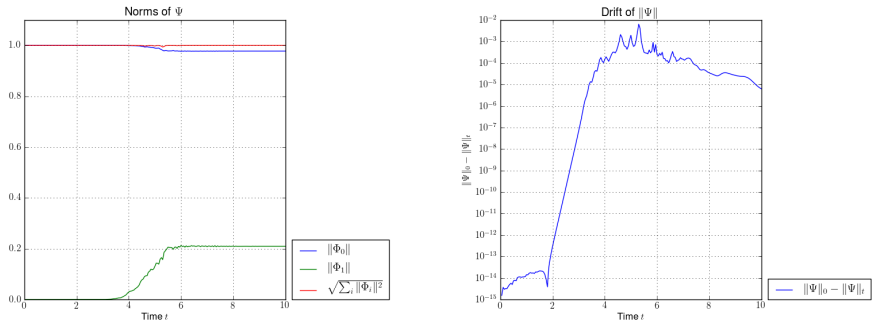
(e) (f)

Figure 6.22: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.1$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$



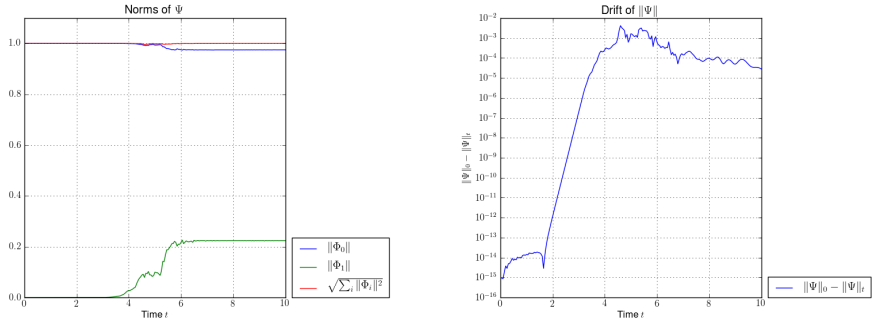
(a)

(b)



(c)

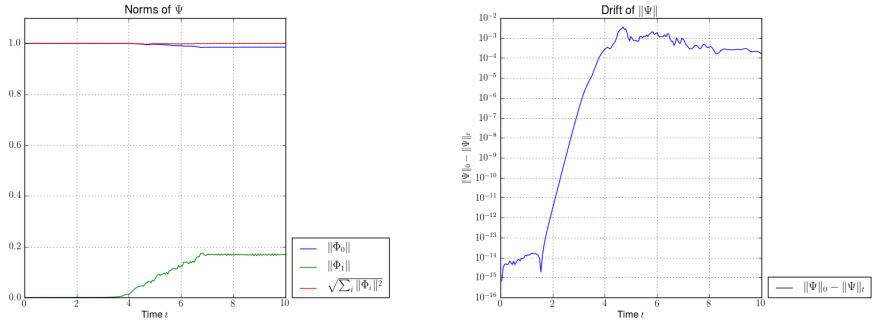
(d)



(e)

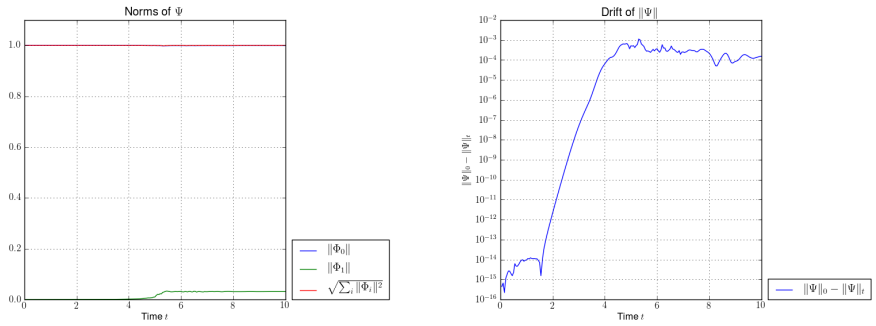
(f)

Figure 6.23: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.1$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



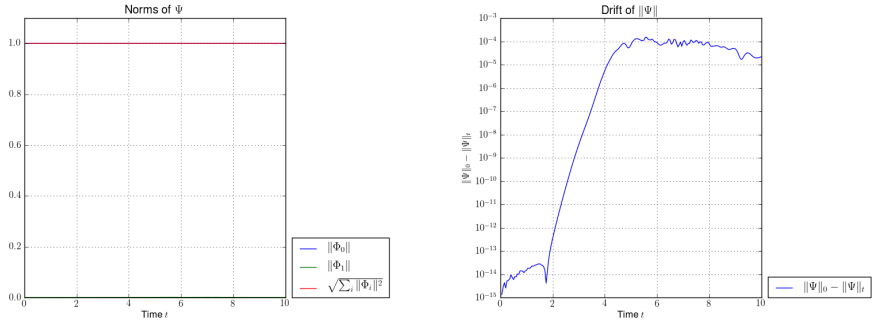
(a)

(b)



(c)

(d)



(e)

(f)

Figure 6.24: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.1$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

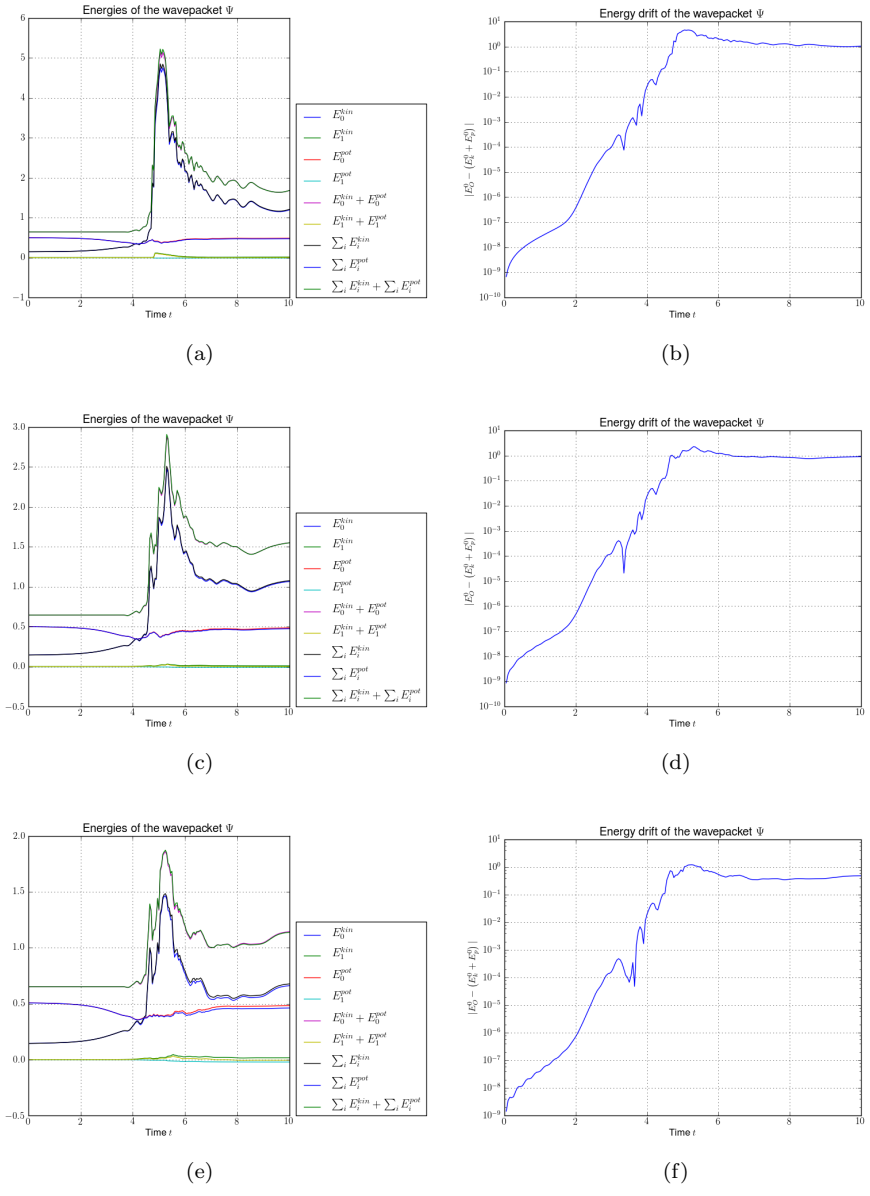


Figure 6.25: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.2$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$

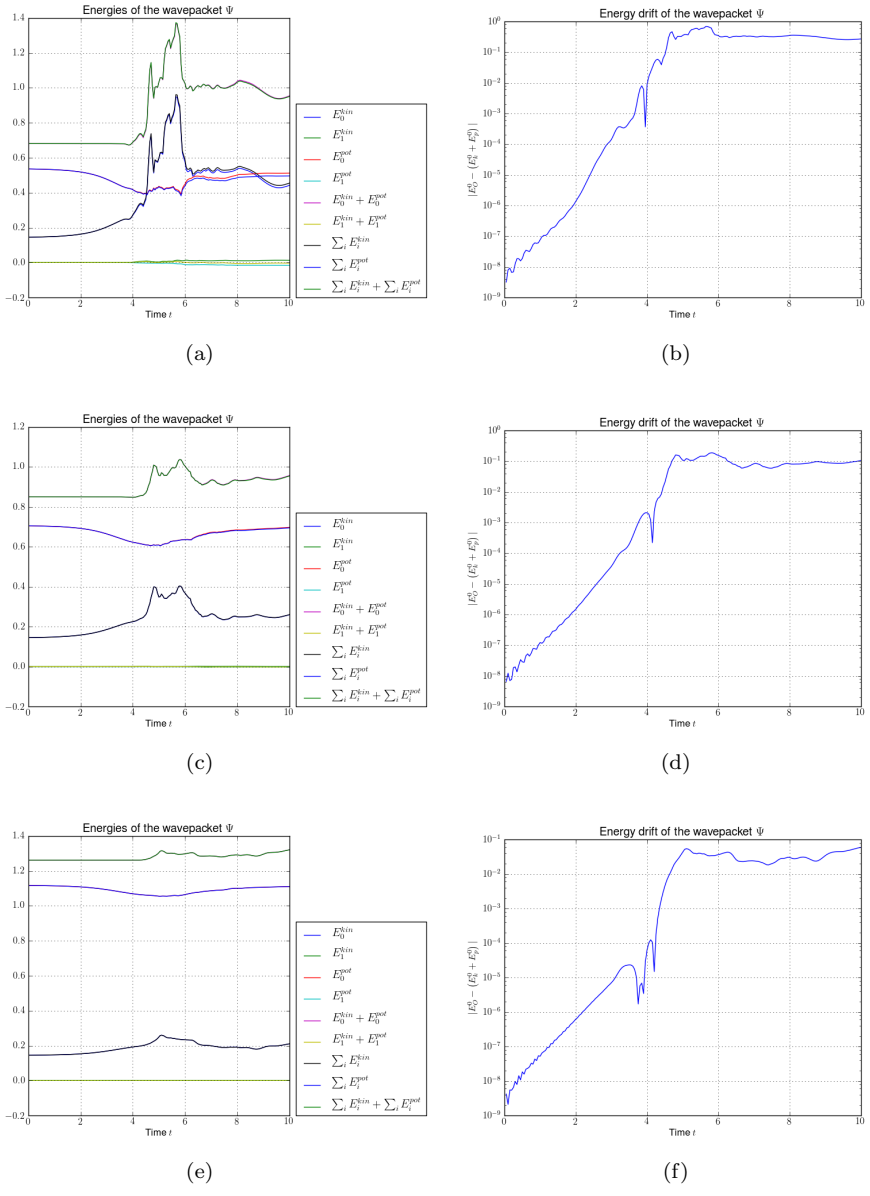
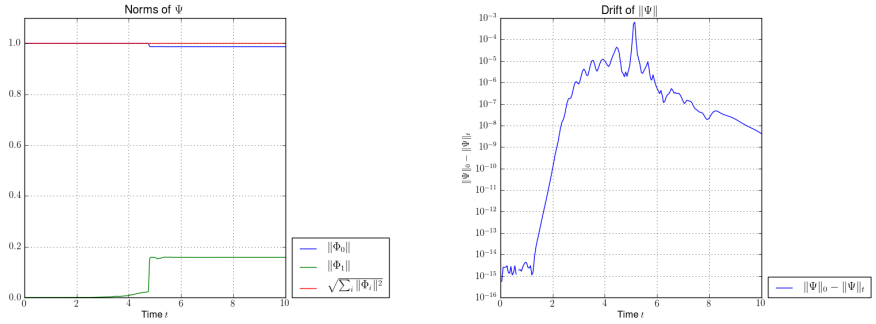
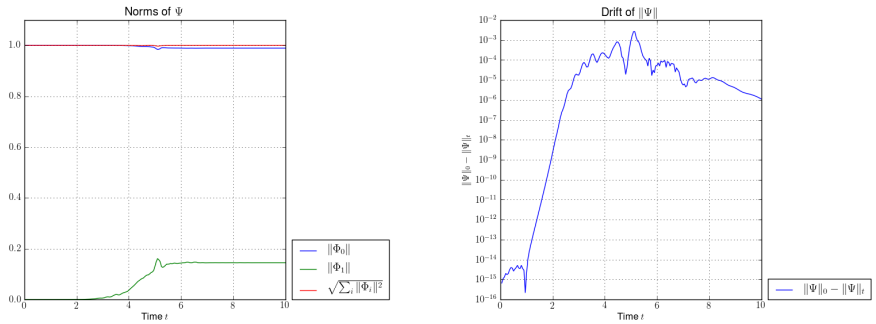


Figure 6.26: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.2$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$



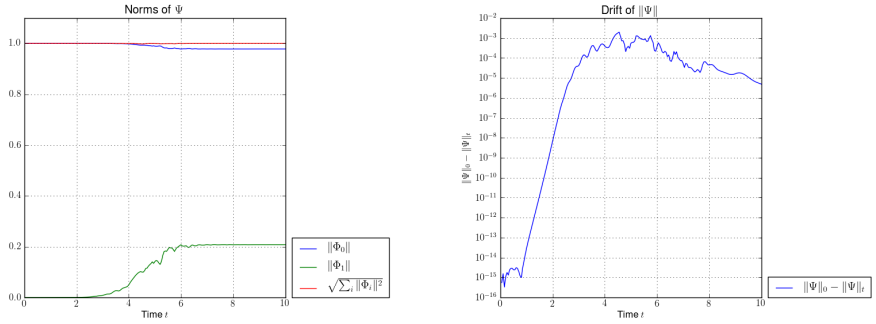
(a)

(b)



(c)

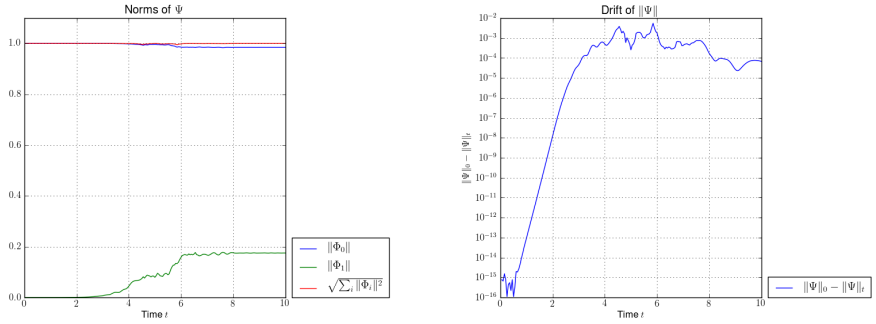
(d)



(e)

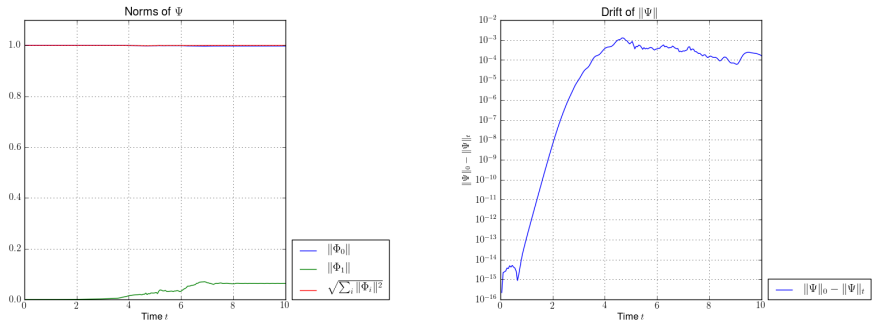
(f)

Figure 6.27: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.2$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



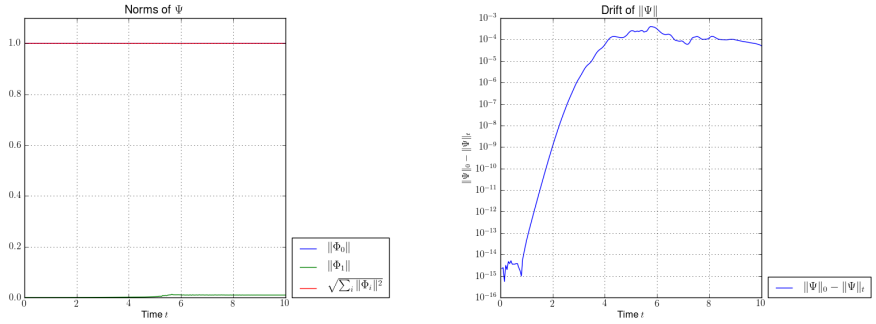
(a)

(b)



(c)

(d)



(e)

(f)

Figure 6.28: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.2$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

6.4 A conical avoided crossing

In this section we simulate a conical avoided crossing taken from [9] where it is classified as type 3, see also [7]. The two-dimensional potential $V(x, y)$ is given by the following real symmetric matrix:

$$\mathbf{V}(x, y) := \begin{pmatrix} x & \sqrt{y^2 + \delta^2} \\ \sqrt{y^2 + \delta^2} & -x \end{pmatrix} \tag{6.5}$$

where $\delta > 0$ is a small real number related to the gap width which is actually 2δ. The effect of shrinking gap width is shown in figure 6.29.

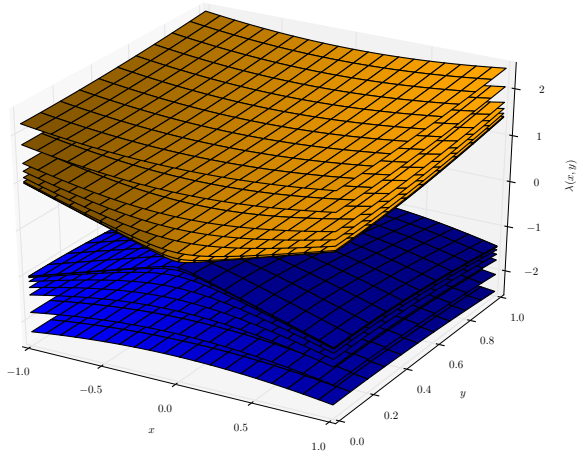
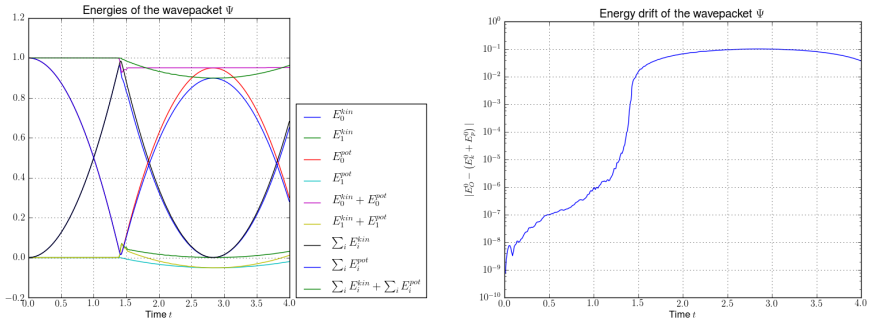


Figure 6.29: Energy levels of the conic avoided crossing given by equation (6.5) for different values of δ.

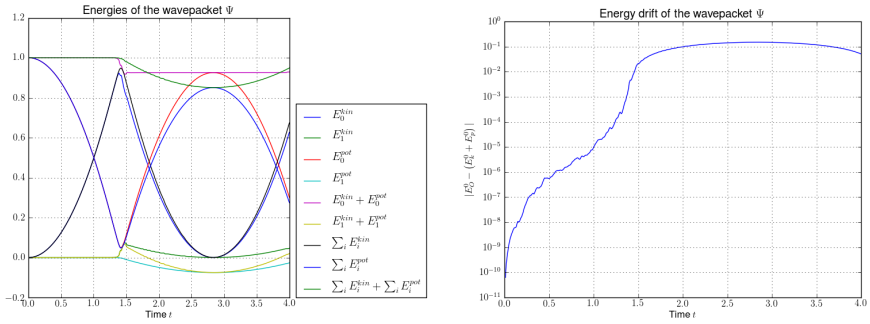
The initial parameter values are:

$$\underline{q} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0$$

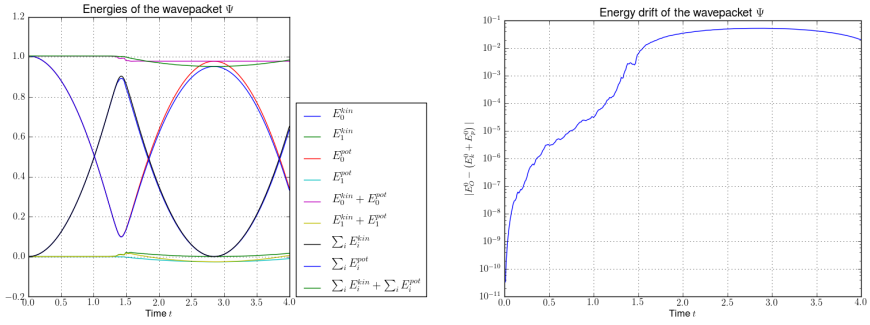
and we used a timestep $\tau = 0.01$.



(a) (b)

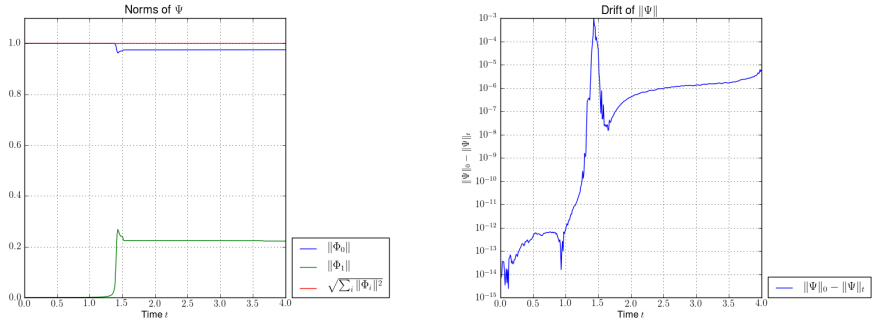


(c) (d)



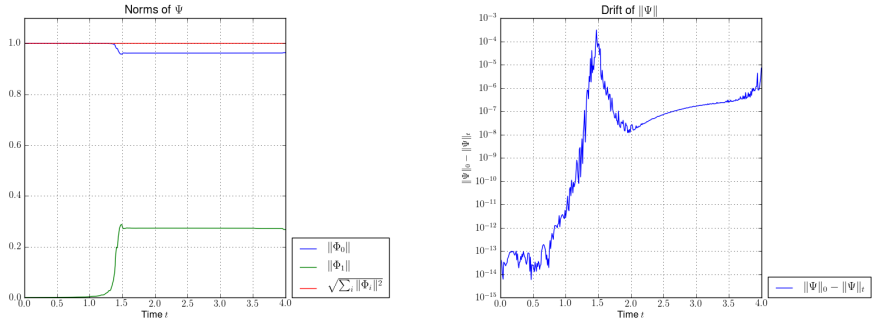
(e) (f)

Figure 6.30: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 32$ and $\varepsilon = 0.01$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



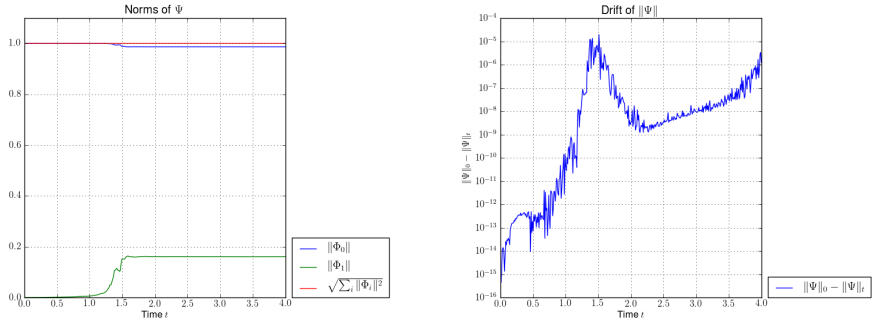
(a)

(b)



(c)

(d)



(e)

(f)

Figure 6.31: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 32$ and $\varepsilon = 0.01$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

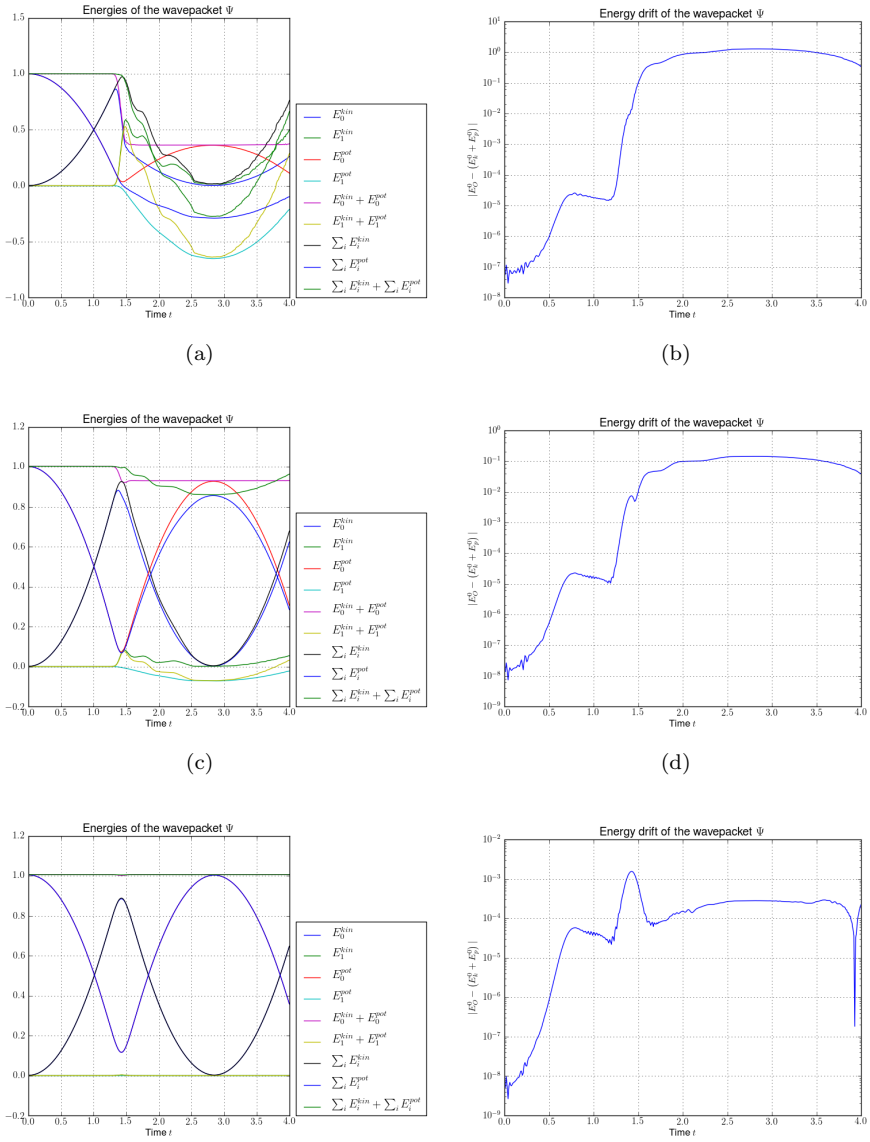
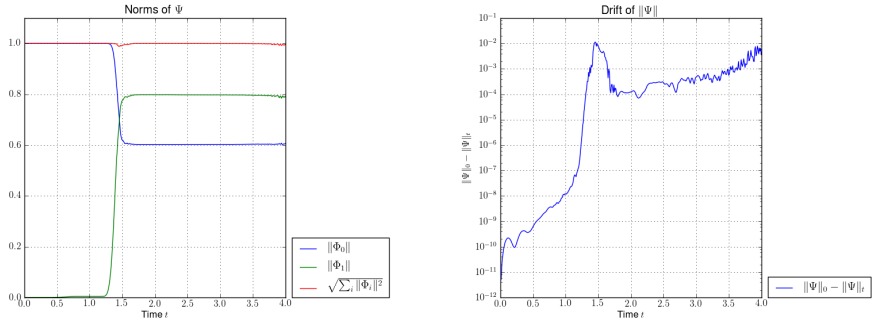
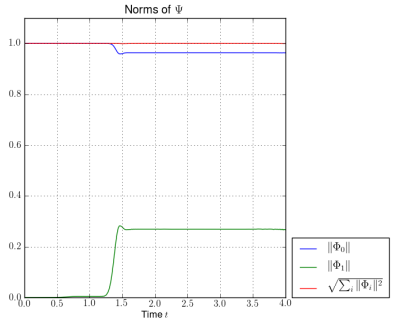


Figure 6.32: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 32$ and $\varepsilon = 0.05$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$

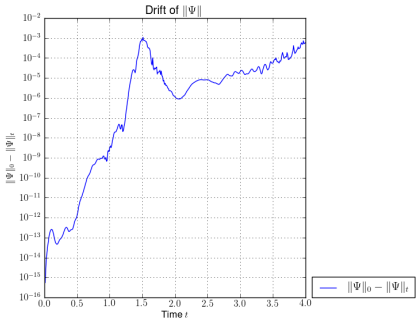


(a)

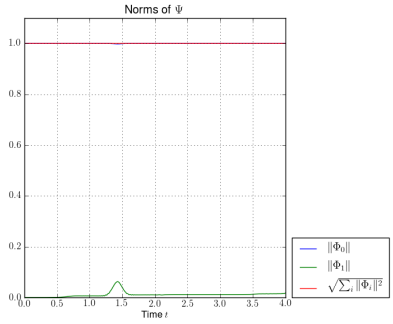
(b)



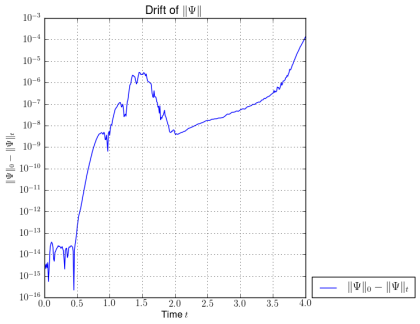
(c)



(d)



(e)



(f)

Figure 6.33: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 32$ and $\varepsilon = 0.05$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

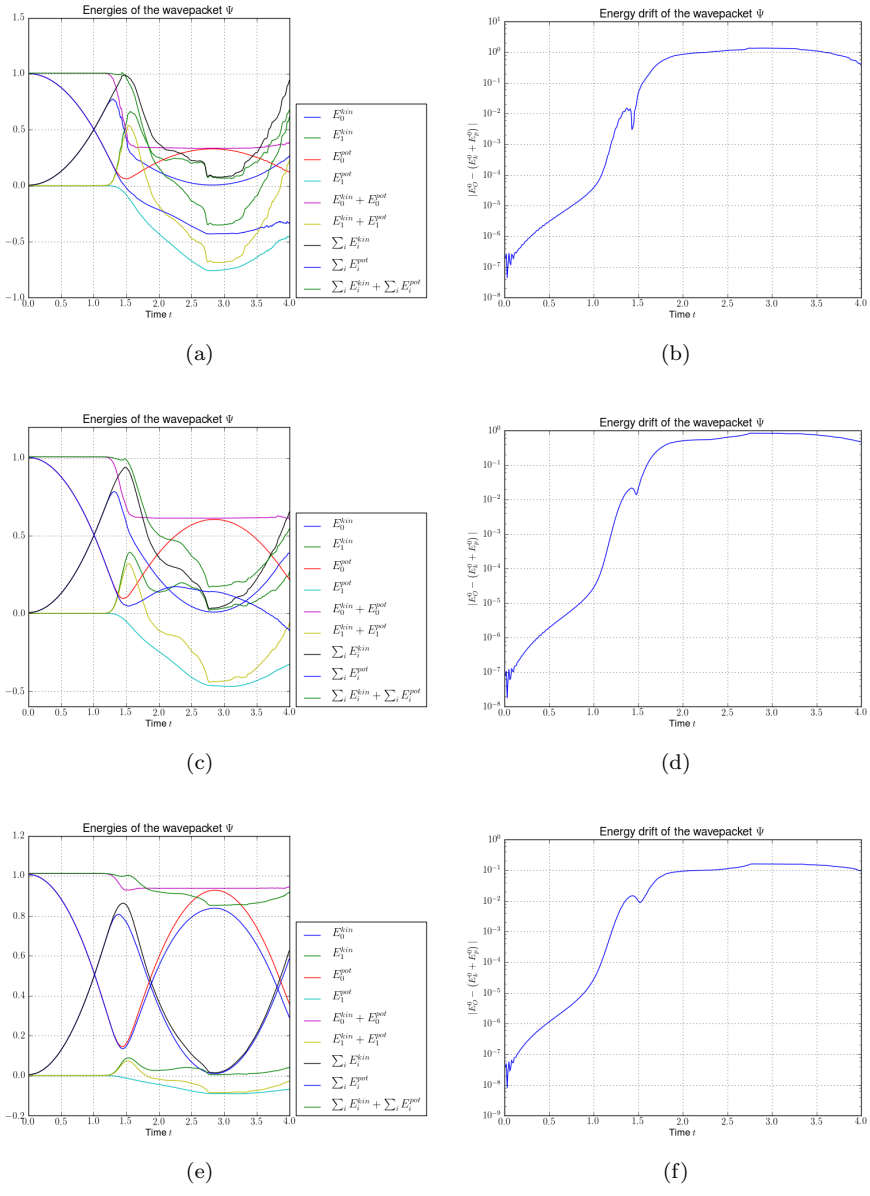
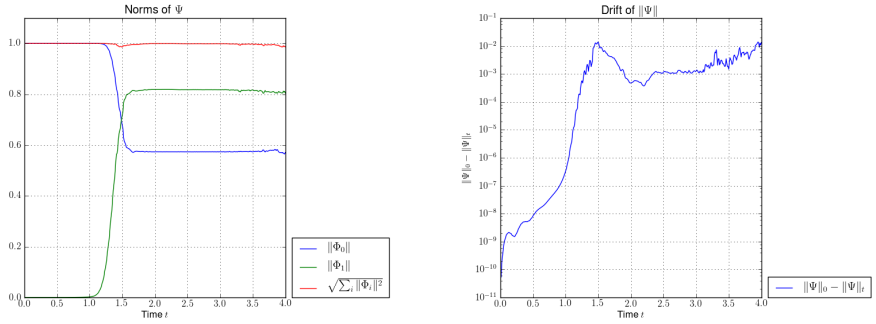
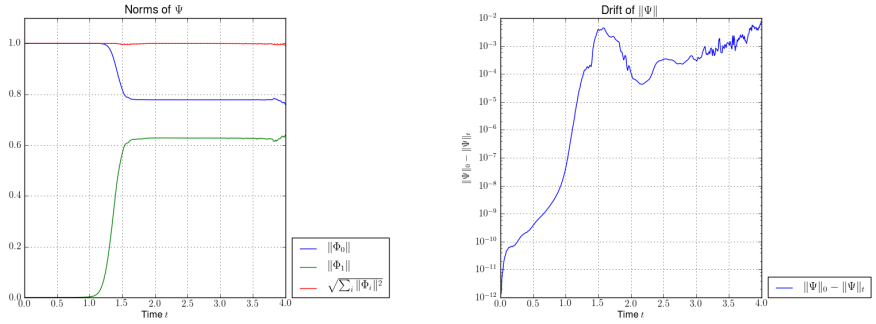


Figure 6.34: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 32$ and $\varepsilon = 0.1$. (a), (b) $\delta = 0.01$ (c), (d) $\delta = 0.05$ (e), (f) $\delta = 0.1$



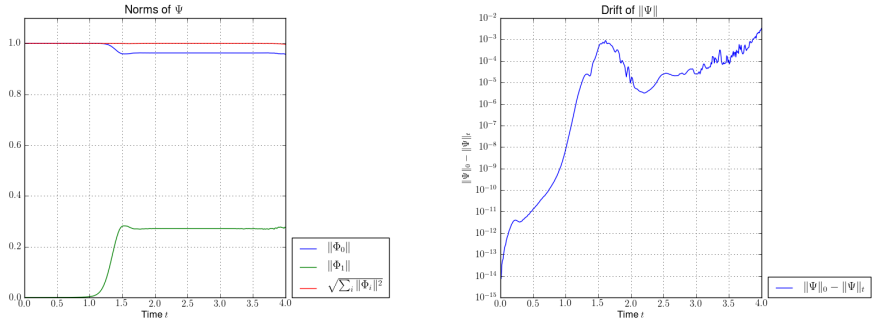
(a)

(b)



(c)

(d)



(e)

(f)

Figure 6.35: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 32$ and $\varepsilon = 0.1$. (a), (b) $\delta = 0.2$ (c), (d) $\delta = 0.5$ (e), (f) $\delta = 1.0$

6.5 A conical crossing

In this section we show some results for a special case of the potential (6.5) given in the last section. The potential is shown in figure 6.36.

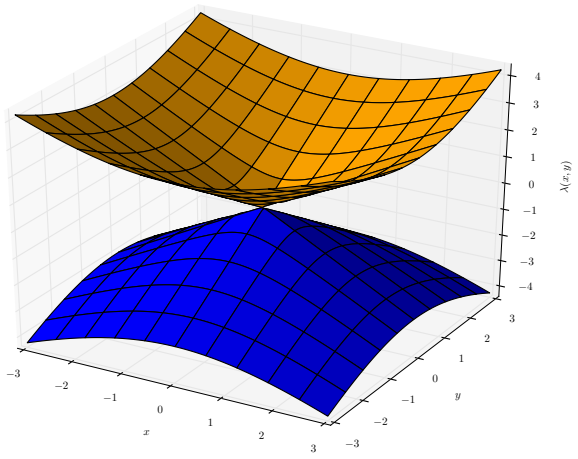
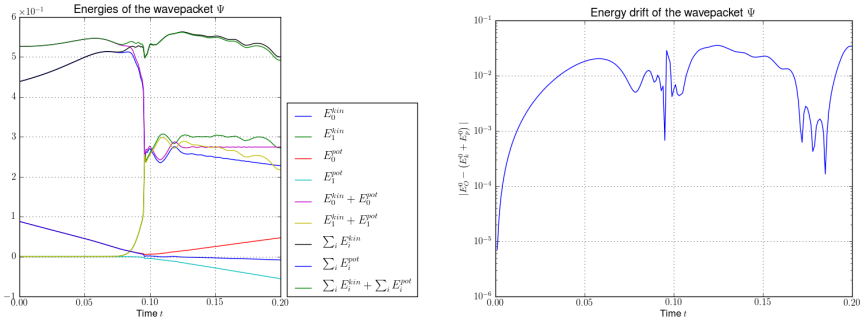


Figure 6.36: Energy levels of the conic crossing given by equation (6.5) for $\delta = 0$. The energy levels touch for $x = y = 0$.

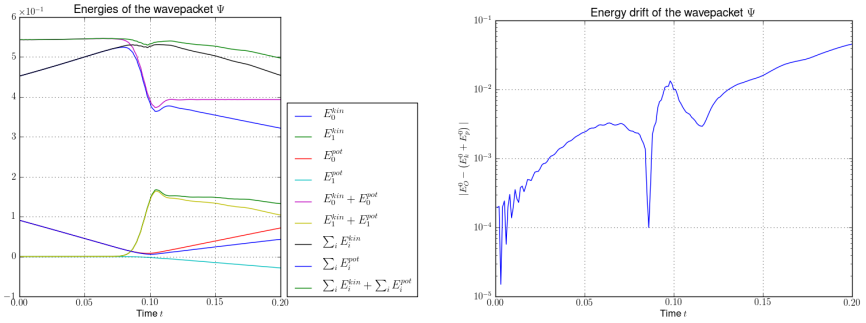
We start with a Gaussian having the following initial parameter set Π:

$$\underline{q} = \begin{pmatrix} -0.1 \\ \alpha\varepsilon \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0\,.$$

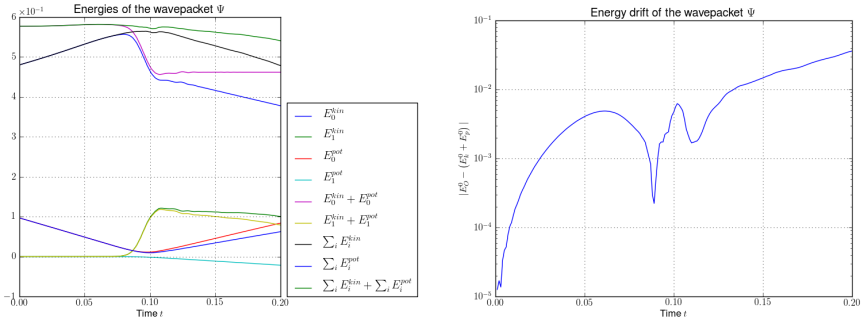
The packet is initialised such that it misses the crossing point. This is controlled by the parameter α which we will vary. Depending on this value, the potential appears to be an avoided crossing. The figures 6.37 and 6.38 show the energies of the wavepacket, the figures 6.39 and 6.40 show the norms and transition probabilities. The time-evolution of Π is shown in figure 6.41 while the trajectories of $\underline{q}$ are shown in figure 6.42.



(a) (b)



(c) (d)



(e) (f)

Figure 6.37: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\alpha = 0.0$ (c), (d) $\alpha = 0.5$ (e), (f) $\alpha = 1.0$

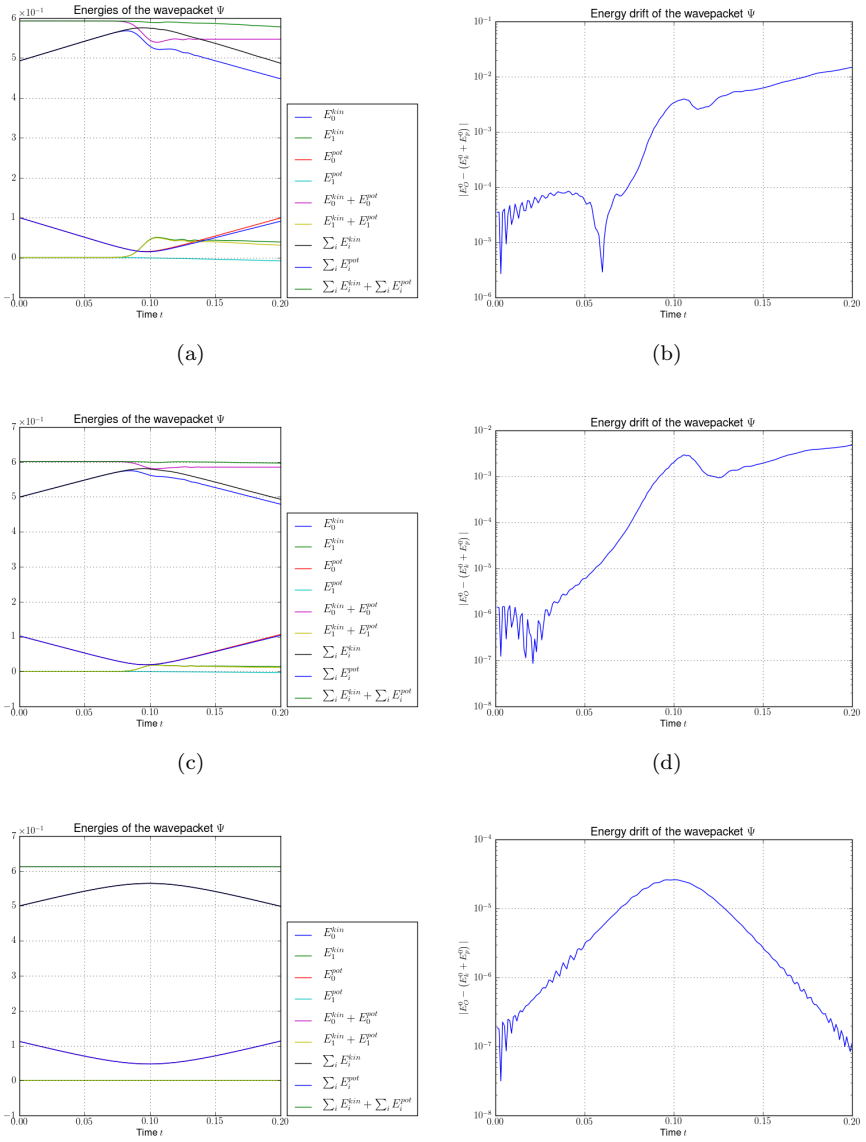
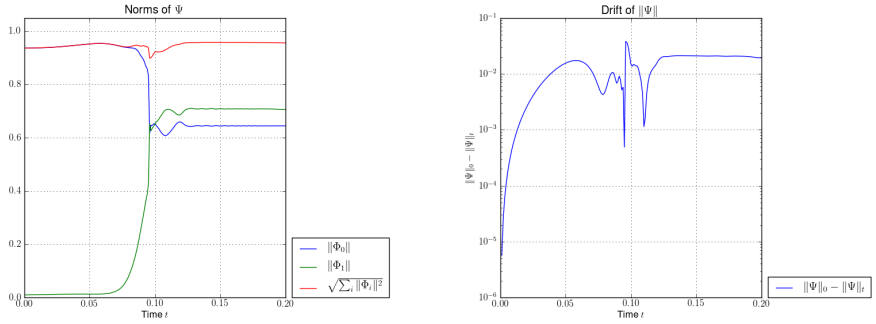
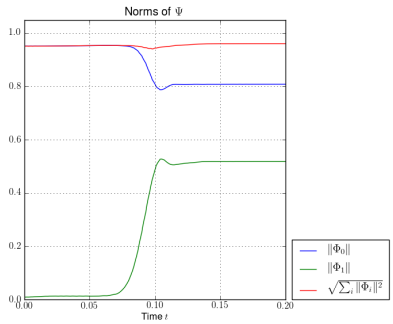


Figure 6.38: Plots of the kinetic, potential and total energy and the drift of the total energy of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\alpha = 1.5$ (c), (d) $\alpha = 2.0$ (e), (f) $\alpha = 5.0$

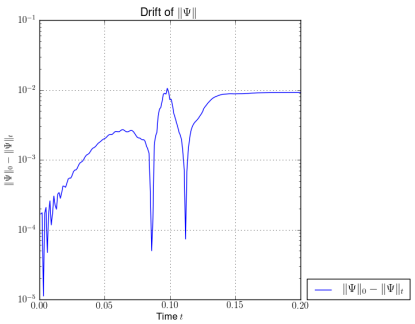


(a)

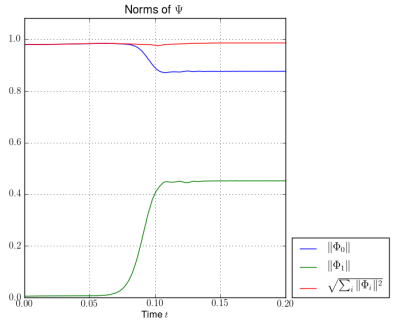
(b)



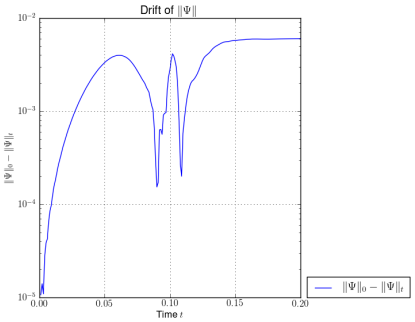
(c)



(d)

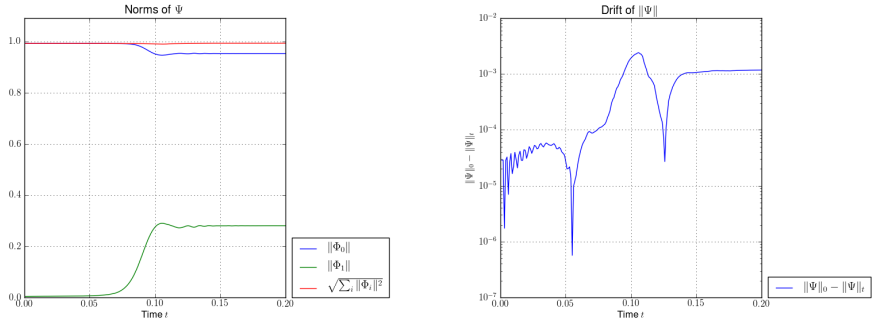


(e)



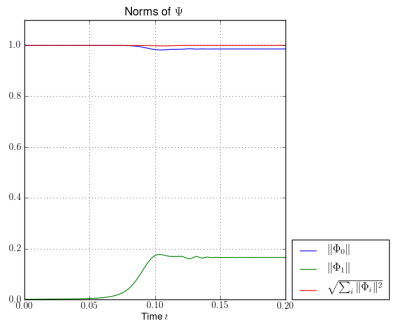
(f)

Figure 6.39: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\alpha = 0.0$ (c), (d) $\alpha = 0.5$ (e), (f) $\alpha = 1.0$

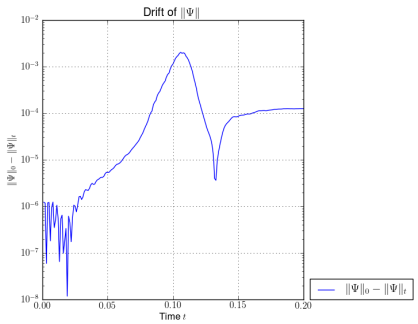


(a)

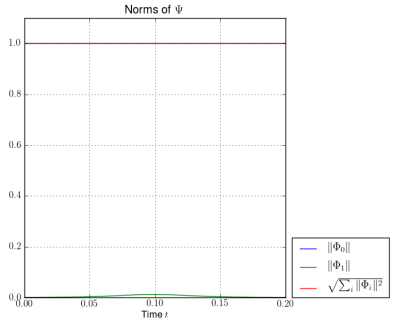
(b)



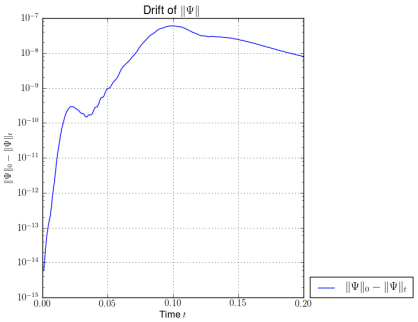
(c)



(d)



(e)



(f)

Figure 6.40: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a), (b) $\alpha = 1.5$ (c), (d) $\alpha = 2.0$ (e), (f) $\alpha = 5.0$

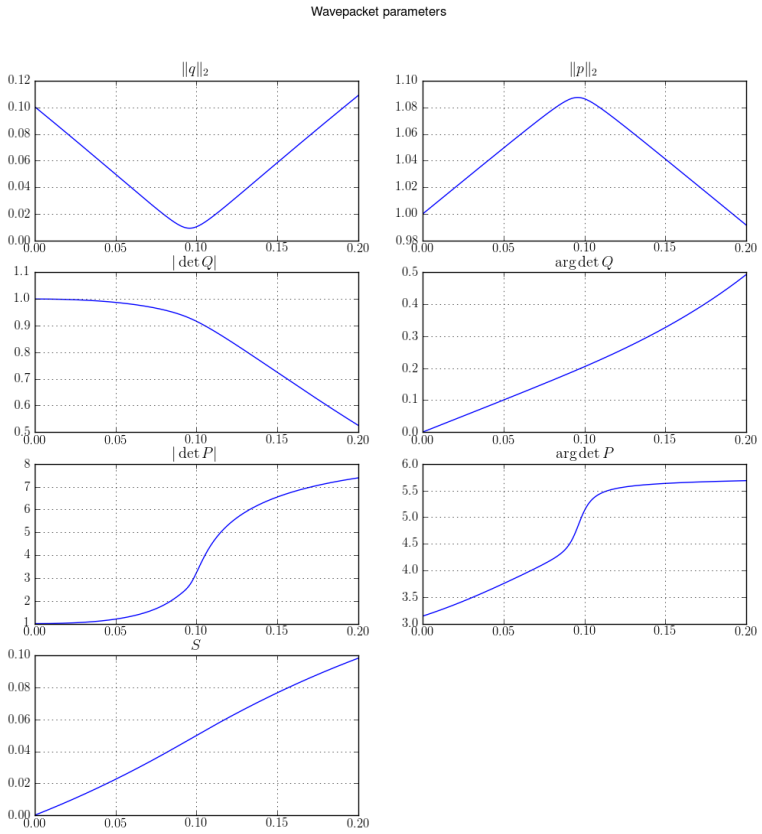
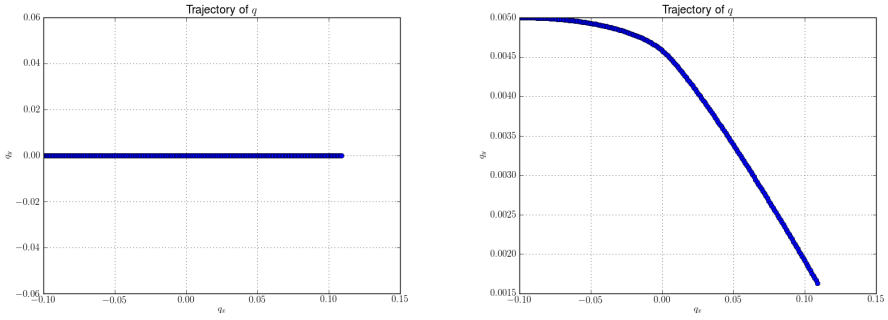
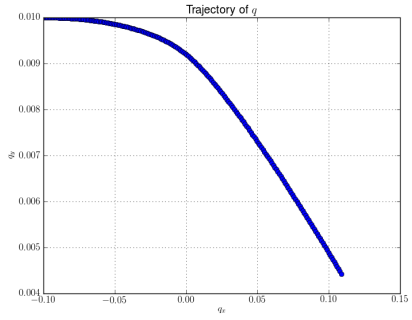


Figure 6.41: Time-evolution of the parameter set Π. The curves look similar for all values of α.

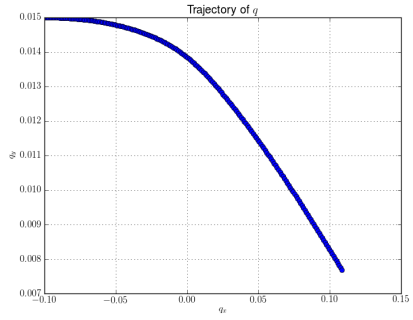


(a)

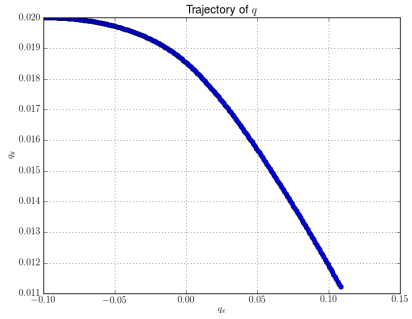
(b)



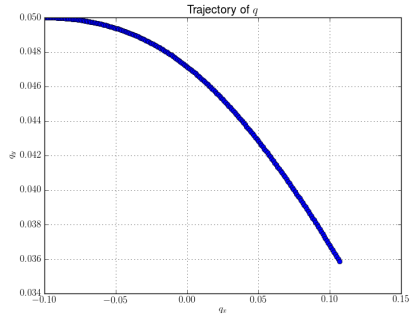
(c)



(d)



(e)



(f)

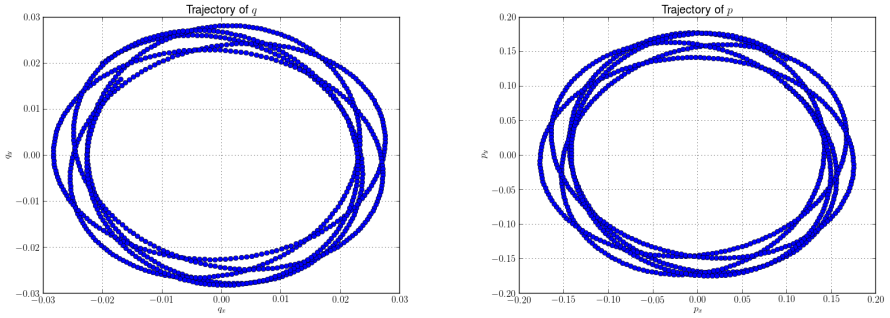
Figure 6.42: The trajectories of $\underline{q}$ of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$ in the x-y-plane. The basis shape is a hyperbolic cut with cut-off $K = 16$ and $\varepsilon = 0.01$. (a) $\alpha = 0.0$ (b) $\alpha = 0.5$ (c) $\alpha = 1.0$ (d) $\alpha = 1.5$ (e) $\alpha = 2.0$ (f) $\alpha = 5.0$

6.5.1 A special initial condition

In this small section we show a simulation result for a very specialised, unnatural initial configuration. The parameter set Π is:

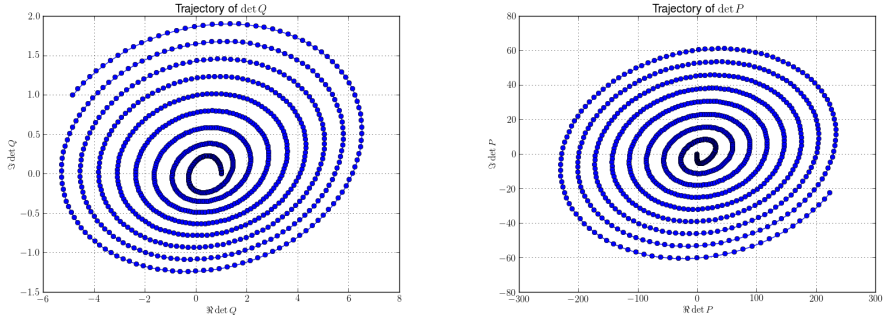
$$\underline{q} = \begin{pmatrix} -\alpha\varepsilon \\ \alpha\varepsilon \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix} \quad S = 0$$

with $\alpha = 2.0$ and $\varepsilon = 0.01$. We take a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$ for this experiment.



(a) (b)

Figure 6.43: The trajectories of $\underline{q}$ and $\underline{p}$ in the x-y-plane.



(a) (b)

Figure 6.44: The trajectories of $\det \mathbf{Q}$ and $\det \mathbf{P}$ in the complex plane.

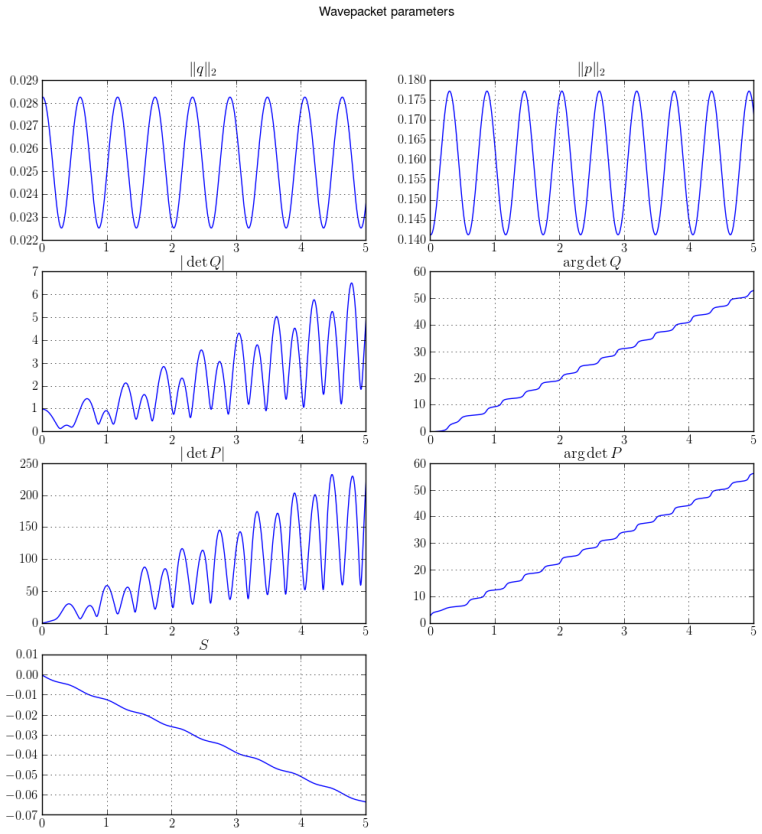
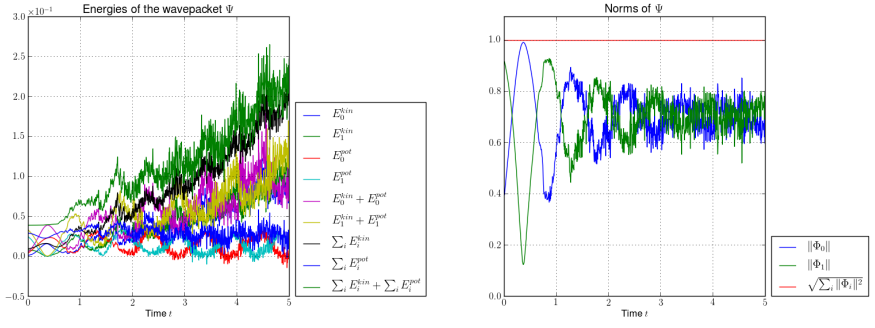


Figure 6.45: Time-evolution of the parameter set Π for a special setting



(a) (b)

Figure 6.46: The plots show the energies and norms of a very special initial setting where the packet orbits around the crossing point.

6.6 Higher dimensional conical avoided crossings

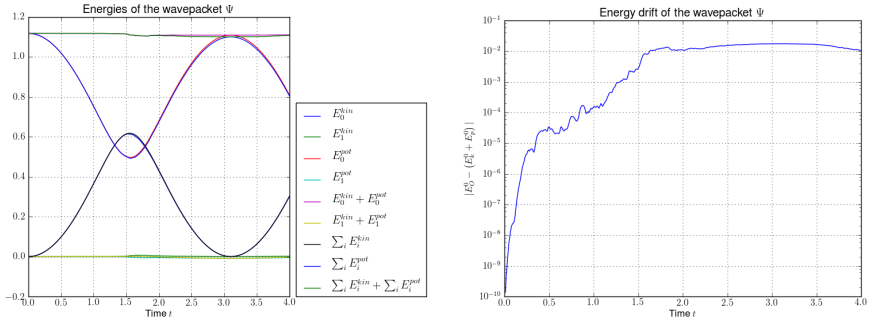
We can generalise the potential given in (6.5) to an arbitrary number D of dimensions. The matrix looks like:

$$\mathbf{V}(\underline{x}) := \begin{pmatrix} x_0 & \sqrt{\delta^2 + \sum_{d=1}^{D-1} x_d^2} \\ \sqrt{\delta^2 + \sum_{d=1}^{D-1} x_d^2} & -x_0 \end{pmatrix} . \tag{6.6}$$

In the following we show a three-dimensional example. The parameters are initialised to:

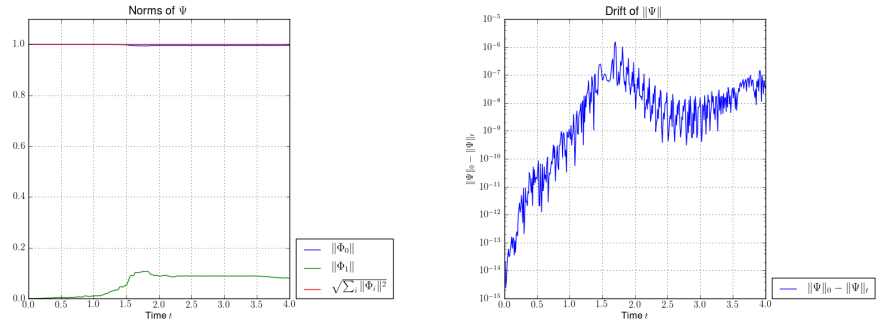
$$\underline{q} = \begin{pmatrix} 1.0 \\ 0 \\ 0 \end{pmatrix} \quad \underline{p} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.0 \end{pmatrix} \quad \mathbf{P} = \begin{pmatrix} i & 0 & 0 \\ 0 & i & 0 \\ 0 & 0 & i \end{pmatrix} \quad S = 0$$

and $\varepsilon = 0.01$. The timestep is set to $\tau = 0.01$ and we use a hyperbolic cut basis shape with cut-off $K = 16$. The energy gap is taken to be $\delta = 0.5$.



(a) (b)

Figure 6.47: The kinetic, potential and total energy and the drift of the total energy of a wavepacket in a three-dimensional conical avoided crossing. (a) The energies. (b) The energy drift.



(a) (b)

Figure 6.48: Plots of the norm of the components on the upper and lower level and the drift of the total norm of a Gaussian wavepacket $|\Psi\rangle = \phi_{0,0}$ in a three-dimensional conical avoided crossing. (a) The norms. (b) The norm drift.

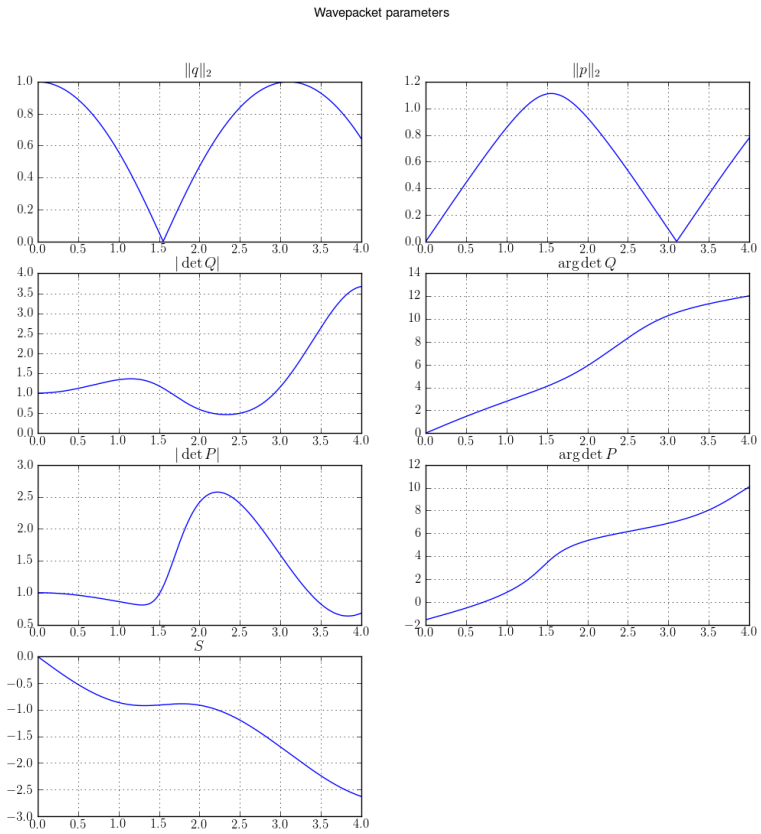
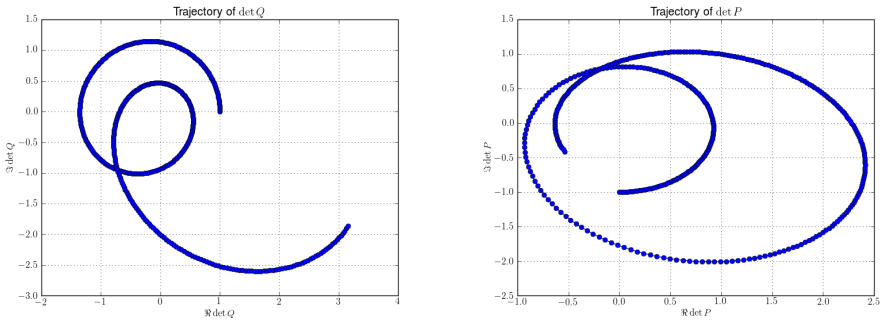


Figure 6.49: Time-evolution of the parameter set Π of a wavepacket in a three-dimensional conical avoided crossing.



(a) (b)

Figure 6.50: Trajectories of $\det \mathbf{Q}$ and $\det \mathbf{P}$ in the complex plane. (a) Trajectory of $\det \mathbf{Q}$. (b) Trajectory of $\det \mathbf{P}$.

6.7 Examples from chemistry

We are currently working on some real world examples directly taken from chemistry. Most recent work is done on the example of Pyrazine, studied extensively in [12, 13, 20, 21]. Some other interesting problems are presented in [17, 18]. The results of these efforts will be published elsewhere.

Appendix A

Derivatives of Eigenvalues

In this small appendix we show the computation of derivatives of the eigenvalues. Although these formulae are well known for example in structural mechanics and similar areas of applied mathematics, we think that it is a good idea to show them here. For more detailed comments and proofs see the references given below.
We start with a real and symmetric $N \times N$ matrix $\mathbf{A}$. Each of its entries is a function $a_{r,c}$ dependent on some parameters. In our case these are the position space variables $x_0, \dots x_{D-1}$. Hence we write $\mathbf{A}(\underline{x})$. Next we assume that the matrix is diagonalisable:

$$\mathbf{A} = \mathbf{M}\mathbf{\Lambda}\mathbf{M}^{\mathrm{T}} \tag{A.1}$$

also compare to (1.4). The eigenvectors are orthogonal by theory and we take them normalised, which we can write as:

$$\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbb{1}\,. \tag{A.2}$$

And finally we require the eigenvalues $\lambda(\underline{x})$ to be all distinct for all values of the parameters $\underline{x}$. Then we can write the first derivatives as:

$$\frac{\partial \lambda_i(\underline{x})}{\partial x_j} = \underline{\nu_i}^{\mathrm{T}} \frac{\partial \mathbf{A}(\underline{x})}{\partial x_j} \underline{\nu_i} \tag{A.3}$$

for all $i, j \in 0, \dots N-1$. With this formula we can compute gradients and Jacobians for all eigenvalues. The important point is that we know the analytic closed-form expressions for $a_{r,c}(\underline{x})$ and computing their derivatives thus is trivial. For the second derivatives we have a similar formula:

$$\frac{\partial^2 \lambda_i(\underline{x})}{\partial x_j \partial x_k} = \underline{\nu_i}^{\mathrm{T}} \frac{\partial^2 \mathbf{A}(\underline{x})}{\partial x_j \partial x_k} \underline{\nu_i} + 2 \sum_{l \neq i} \frac{\left(\underline{\nu_i}^{\mathrm{T}} \frac{\partial \mathbf{A}(\underline{x})}{\partial x_j} \underline{\nu_l}\right)\left(\underline{\nu_i}^{\mathrm{T}} \frac{\partial \mathbf{A}(\underline{x})}{\partial x_k} \underline{\nu_l}\right)}{\lambda_i - \lambda_l}\,. \tag{A.4}$$

With the help of this formula we can compute Hessian matrices for all eigenvalues easily. These formulae provide us with the tools to compute the quadratic approximation in algorithms 15 and 17.
For more comprehensive information on this topic see for example [11, 14, 15].

Appendix B

Colour coding convention

For plotting complex-valued functions we use a special colour coding. This allows us to plot phase and absolute values within the same figures. The colour code shown here was introduced by Thaller in [19] many years ago.
Sometimes one also darkens the colours depending on the magnitude of the absolute value. For details about this see the given reference.

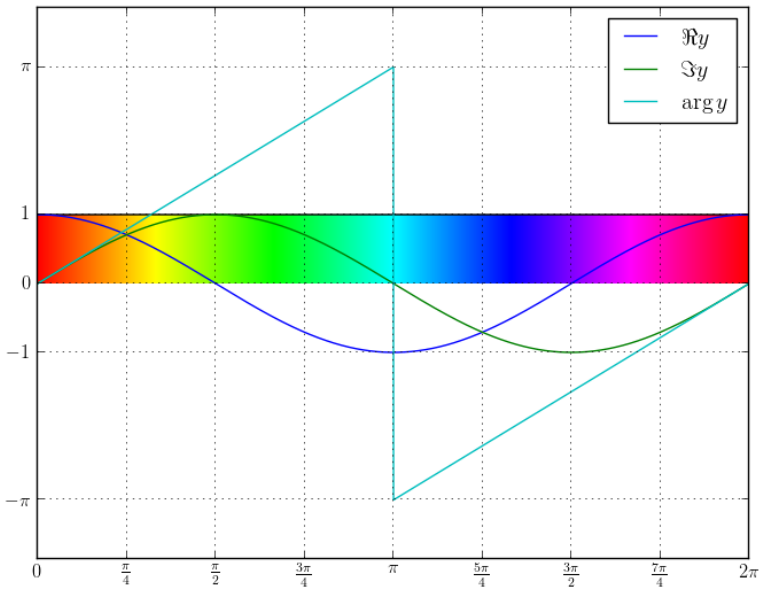


Figure B.1: This plot shows a complex-valued function $f : \mathbb{R} \to \mathbb{C}$ in colour-coded fashion as well as the real and imaginary parts and the phase.

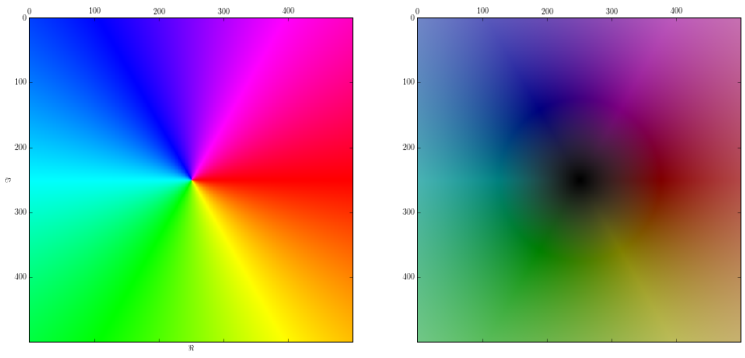


Figure B.2: This plot shows the colour distribution for complex numbers z.

Appendix C

Acknowledgements

I wish to thank my supervisor Vasile Grădinaru for many helpful comments and crucial ideas that stand behind this work. Further I have to thank George A. Hagedorn for interesting discussions during his visit in Zurich.

Bibliography

[1] R. Bourquin. Algorithms for non-adiabatic transitions with one-dimensional wavepackets. 2010. http://www.sam.math.ethz.ch/~raoulb/research/bachelor_thesis/tex/main.pdf. (Cited on pages 7, 12, 25, 71, 85, 87 and 105.)

[2] R. Bourquin. Spawning of wave-packets in 1D non-adiabatic transitions. 2011. http://www.sam.math.ethz.ch/~raoulb/research/term_thesis/main.pdf. (Cited on page 52.)

[3] R. Bourquin and V. Gradinaru. WaveBlocks: Reusable building blocks for simulations with semiclassical wavepackets. https://github.com/WaveBlocks/WaveBlocksND, 2010 - 2016. (Cited on page 97.)

[4] R. Bourquin, V. Gradinaru, and G.A. Hagedorn. Non-adiabatic transitions near avoided crossings: theory and numerics. *Journal of Mathematical Chemistry*, 50:602–619, 2012. (Cited on page 87.)

[5] Erwan Faou, Vasile Gradinaru, and Christian Lubich. Computing semiclassical quantum dynamics with Hagedorn wavepackets. *SIAM Journal on Scientific Computing*, 31(4):3027–3041, 2009. (Cited on pages 7, 87, 88, 97, 102 and 105.)

[6] Vasile Gradinaru, George A. Hagedorn, and Alain Joye. Exponentially accurate semiclassical tunneling wavefunctions in one dimension. *Journal of Physics A: Mathematical and Theoretical*, 43(47):474026, 2010. (Cited on page 52.)

[7] George A. Hagedorn. Classification and normal forms for avoided crossings of quantum-mechanical energy levels. *Journal of Physics A: Mathematical and General*, 31:369, 1998. (Cited on page 122.)

[8] George A. Hagedorn. Raising and lowering operators for semiclassical wave packets. *Annals of Physics*, 269(1):77–104, 1998. (Cited on pages 24, 27, 28, 55 and 88.)

[9] George A. Hagedorn and Alain Joye. Molecular propagation through small avoided crossings of electron energy levels. *Rev. Math. Phys*, 11:41–101, 1997. (Cited on pages 8 and 122.)

[10] Nicholas J. Higham. *Functions of matrices - theory and computation*. SIAM, 2008. (Cited on page 74.)

[11] P. Lancaster. On eigenvalues of matrices dependent on a parameter. *Numerische Mathematik*, 6:377–387, 1964. (Cited on page 142.)

[12] Caroline Lasser and Torben Swart. Single switch surface hopping for a model of pyrazine. *The Journal of Chemical Physics*, 129(3):034302, 2008. (Cited on page 141.)

[13] Soo-Y. Lee and Eric J. Heller. Exact time-dependent wave packet propagation: Application to the photodissociation of methyl iodide. *The Journal of Chemical Physics*, 76(6):3035–3044, 1982. (Cited on page 141.)

[14] Durbha V Murthy and Raphael T Haftka. Derivatives of eigenvalues and eigenvectors of a general complex matrix. *International Journal for Numerical Methods in Engineering*, 26(2):293–311, 1988. (Cited on page 142.)

[15] Michael L. Overton and Robert S. Womersley. Second derivatives for optimizing eigenvalues of symmetric matrices. *SIAM J. Matrix Anal. Appl.*, 16(3):697–718, 1995. (Cited on page 142.)

[16] Kaare Brandt Petersen, Michael Syskind Pedersen, Jan Larsen, Korbinian Strimmer, Lars Christiansen, Kai Hansen, Liguo He, Loic Thibaut, Miguel Barão, Stephan Hattinger, Vasile Sima, and We The. The matrix cookbook. techreport, 2006. (Cited on page 71.)

[17] Subhankar Sardar, Amit Paul, and Satrajit Adhikari. A quantum-classical simulation of the nuclear dynamics in no 2 and c6h 6 + with realistic model hamiltonian. *Journal of Chemical Sciences*, 122(4):491–510, 2010. (Cited on page 141.)

[18] Subhankar Sardar, Amit Kumar Paul, Rahul Sharma, and Satrajit Adhikari. A "classical" trajectory driven nuclear dynamics by a parallelized quantum-classical approach to a realistic model hamiltonian of benzene radical cation. *International Journal of Quantum Chemistry*, 111(12):2741–2759, 2011. (Cited on page 141.)

[19] Bernd Thaller. *Visual Quantum Mechanics: Selected Topics with Computer Generated Animations of Quantum Mechanical Phenomena with Cdrom*. Springer-Verlag New York, Inc., 1st edition, 2000. (Cited on page 143.)

[20] Till Westermann, Ralf Brodbeck, Alexander B. Rozhenko, Wolfgang Schoeller, and Uwe Manthe. Photodissociation of methyl iodide embedded in a host-guest complex: A full dimensional (189d) quantum dynamics study of ch[sub 3]i@resorc[4]arene. *The Journal of Chemical Physics*, 135(18):184102, 2011. (Cited on page 141.)

[21] Daiqian Xie, Hua Guo, Yoshiaki Amatatsu, and Ronnie Kosloff. Three-dimensional photodissociation dynamics of rotational state selected methyl iodide. *The Journal of Physical Chemistry A*, 104A(5):1009–1019, 2000. (Cited on page 141.)