Universität Tübingen

Mathematisches Institut

Abschlussarbeit

zur Erlangung des akademischen Grades eines Master of Science (M.Sc.)

An Algorithm for Variational Multi-Configurational Gaussians

in der Arbeitsgruppe numerische Analysis

von

Dominik Sulz

geboren am 20.06.1995 in Stuttgart

Matrikel-Nr. 3879254

Betreuer:	Prof. Dr. Christian Lubich
Zweitgutachter:	Prof. Dr. Andreas Prohl
Einreichung:	14.05.2020

Contents

Acknowledgments

First of all I would like to thank Professor Lubich for the opportunity to write my master thesis in the numerical analysis group and for the interesting topic. I appreciated the opportunity to ask for advice at any time.
Further I want to thank Gianluca Ceruti for the constant help during my work. I am very grateful for all the stimulating discussions and the helpful hints.
Also I want to thank my sister Aylin and again Gianluca Ceruti for proofreading the thesis.

Finally I want to thank my family for the support during my study and everything they did for me.

Introduction

We introduce the time dependent Schrödinger equation in a semiclassical scaling

$$i\epsilon\frac{d\psi}{dt} = H\psi, \tag{1}$$

where $\psi = \psi(x,t)$ is the wave function, $t \in \mathbb{R}$ is the time and $x \in \mathbb{R}^d$ the spatial variable. Further is $\epsilon > 0$ a semiclassical parameter, representing the scaled Planck constant $\hbar$, i.e. it is a ratio between $\hbar$ and a characteristic action. H is the Hamiltonian of the system. We will consider a Hamiltonian which is of the form

$$H\psi(x,t) = -\frac{\epsilon^2}{2}\Delta\psi(x,t) + V(x)\psi(x,t),$$

where $-\Delta$ is the Laplacian with respect to the spatial variable x and V is a real-valued potential. ϵ^2 represents the mass ratio of electrons and nuclei in a molecule [LL, p.2]. The aim of this thesis is to present an algorithm which approximates a solution to (1) for a small parameter $\epsilon > 0$.
Finding a good numerical approximation for this partial differential equation is in general very difficult, as the problem is high-dimensional and the solution is highly oscillatory in space and time [LL, p.2].

A promising ansatz to find a good approximation uses Gaussian wave packets. This method is called variational multi-configurational Gaussians (vMCG) and finds applications in quantum chemistry. It was first proposed by Heller in 1975. He considered multi-dimensional Gaussians and derived equations of motions for their parameters [Hel75]. This idea was also extended to linear combinations of Gaussians, for example in [RPS+15].

The purpose of this thesis is to introduce a new way of implementing the vMCG method. In chapter 1 we formulate the problem, introduce basic notation, explain the idea for an algorithm and derive equations of motion for the parameters of the Gaussians. Most of the presented techniques rely on [Lub08].
Chapter 2 introduces Hagedorn functions and Hagedorn wave packets, which were developed by George Hagedorn in [Hag81], [Hag85] and [Hag98]. These functions allow us to calculate high-dimensional integrals of a Gaussian times a

polynomial. This is crucial for the algorithm, because we will apply a modified Gram-Schmidt method to functions in $L^2(\mathbb{R}^d)$ and need their inner products.

In chapter 3 the algorithm and how it is implemented will be explained in detail. The final chapter 4 presents various numerical experiments. We consider one and multi-dimensional quantum systems with different approaches. Finally the results will be evaluated.

Chapter 1

Problem formulation

1.1 Abstract formulation

For an abstract formulation of the problem consider a complex Hilbert space $\mathcal{H}$ with the inner product $< \cdot, \cdot >$. As in 1 we consider a self-adjoint Hamiltonian H with $H = -\Delta + V$, where V is a real-valued potential. Let now $\mathcal{M}$ be a submanifold of $\mathcal{H}$. For $u \in \mathcal{M}$ let $\mathcal{T}_u\mathcal{M}$ be the tangent space at the point u, which consists of the derivatives of all differentiable paths on $\mathcal{M}$ passing through u. The tangent space $\mathcal{T}_u\mathcal{M}$ can be interpreted as a linear approximation of the submanifold $\mathcal{M}$ at the point u. In this thesis, our goal is to find an approximation $u(t) \in \mathcal{M}$ of the wave function $\psi(t)$. The approximation $u(t)$ is determined by the condition that for every time t, its time derivative $\partial u/\partial t$, which lies in $\mathcal{T}_u\mathcal{M}$, be such that the following condition holds:

$$\partial_t u(t) \in \mathcal{T}_u\mathcal{M} \; s.t. \;\; \langle v | \partial_t u(t) - \frac{1}{i\hbar} H u(t) \rangle = 0 \quad \forall v \in \mathcal{T}_u\mathcal{M}. \tag{1.1}$$

This variational formulation of the problem is called the Dirac–Frenkel Time-Dependent Variational Principle. This condition can also be seen as a Galerkin condition on the approximation space $\mathcal{T}_u\mathcal{M}$. More details can be found in [Lub08, p. 19ff].
Taking the real part of (1.1), we get a minimum condition for the linear approximation problem

$$\frac{\partial u}{\partial t} \text{ is choosen as } w \in \mathcal{T}_u\mathcal{M} \text{ s.t. } \left\| w - \frac{1}{i\hbar} H u \right\| \text{ is minimal.} \tag{1.2}$$

This formulation of the problem leads to the interpretation of (1.2) as a orthogonal projection onto the tangent space $\mathcal{T}_u\mathcal{M}$ [Lub08, p.20].

1.1.1 Application to Gaussian wave packets

For the approximation with Gaussian wave packets we choose an approximation $u(\cdot,t) = u(t)$ to the solution $\psi(t)$ in the following manifold:

$$\mathcal{M} := \left\{ u \in L^2(\mathbb{R}^d) \;\middle|\; u(x) = \sum_{j=1}^{m} a_j \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q_j)^T C_j (x-q_j) + p_j^T (x-q_j) + \zeta_j\right)\right), \right.$$
$$\left. a_j \in \mathbb{C}, q_j, p_j \in \mathbb{R}^d, C_j = C_j^T \in \mathbb{C}^{d\times d}, \mathrm{Im}(C_j) \text{ is positiv definit}, \zeta_j \in \mathbb{C} \right\}. \tag{1.3}$$

All the appearing parameters $y := (a_j, q_j, p_j, C_j, \zeta_j)_j$ and $u(x)$ depend on the time t, therefore one should write $y(t) = (a_j(t), q_j(t), p_j(t), C_j(t), \zeta_j(t))_j$ and $u(x,t)$. For sake of readiness, we omit it. The ϵ is a semiclassical parameter. This application to Gaussian wave packets can be found in [LL, p. 12ff], with the difference that they only use one Gaussian instead of linear combinations. For simplicity we will assume $a_j = 1$ for all times and for all j, because its use is a normation of the sum of the Gaussians. u can also be normalized by the parameters ζ_j and we hope that the oscillations of the ζ_j will not be as high as the oscillations of the a_j.

Tangent space
Assuming an approximation u as in definition (1.3) we still need to know how the tangent space $\mathcal{T}_u\mathcal{M}$ looks like. The following theorem holds:

Theorem 1.1

For $u = \sum_{j=1}^{m} g_j(x) \in \mathcal{M}$, where g_j is a Gaussian of the form

$$g_j(x) = \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q_j)^T C_j (x-q_j) + p_j^T (x-q_j) + \zeta_j\right)\right)$$

as in definition (1.3), the tangent space equals

$$\mathcal{T}_u\mathcal{M} = \left\{ v \;\middle|\; v = \sum_{j=1}^{m} \varphi_j g_j \text{ , } \varphi_j \text{ is a complex } d\text{-variate polynomial of deg}\leq 2 \right\}. \tag{1.4}$$

Proof.
In section 1.1 we defined $\mathcal{T}_u\mathcal{M}$ in the way that it consists the derivatives of all differentiable paths on $\mathcal{M}$ passing through u. A direct calculation shows us that

an arbitrary element of the tangent space $v \in \mathcal{T}_u\mathcal{M}$ has the form

$$v(x) = \sum_{j=1}^{m} \underbrace{\frac{i}{\epsilon}\left(-\dot{q_j}^T C_j(x-q_j) + \frac{1}{2}(x-q_j)^T \dot{C_j}(x-q_j) + \dot{p_j}(x-q_j) - p_j^T \dot{q_j} + \dot{\zeta_j}\right)}_{=:\varphi_j} g_j(x)$$

$$= \sum_{j=1}^{m} \varphi_j g_j,$$

with arbitrary $\dot{q_j}, \dot{p_j} \in \mathbb{R}^d, \dot{C_j} = \dot{C_j}^T \in \mathbb{C}^{dxd}, \dot{\zeta_j} \in \mathbb{C}$. Every d-variate complex polynomial of degree ≤ 2 can be written in the form of a φ_j. Thus we have that each element v in the tangent space has the stated form. □

The statement and the proof can be found in [LL, p.13f] lemma 3.1, with the difference that there u is not a linear combination but one single Gaussian.

Position and momentum averages
It is worth to have a closer look on what the parameters p and q stand for. Assume $m = 1$, take a $u \in \mathcal{M}$ of unit norm and separate C and ζ in real- and imaginary part, i.e. $C = A + iB$ and $\zeta = \gamma + i\delta$. An easy calculation shows that

$$\begin{aligned}\overline{u(x)}u(x) &= \exp\left(-\frac{1}{\epsilon}\left((x-q)^T B(x-q) + 2\delta\right)\right) \\ &= \exp\left(\frac{-2\delta}{\epsilon}\right)\exp\left(-\frac{1}{2}\left((x-q)^T \frac{2B}{\epsilon}(x-q)\right)\right). \end{aligned} \tag{1.5}$$

As $u(x)$ is of unit norm, (1.5) is a density function of a d-dimensional normal distribution with expectation value q and covariance matrix $C = (2B/\epsilon)^{-1}$. As by assumption B is a real positive definite symmetric matrix, it is well known that it holds

$$\langle u|x_i u\rangle = q_i \;\Rightarrow\; \langle u|xu\rangle = q.$$

For the statement of the expectation value of a d-dimensional normal distribution see for example theorem 15.53 in [Kle06, p.312]. Therefore we saw that the position average equals the parameter q.
In the same we we calcluate

$$\begin{aligned}\langle u|(-i\epsilon\nabla_x u)_i\rangle = \langle u|(C(x-q)+p)_i u\rangle &= \int_{\mathbb{R}^d} \sum_{l=1}^{d} C_{il}(x_l - q_l) + p_i)u(x)\; dx \\ &= \sum_{l=1}^{d} C_{il} \underbrace{\int_{\mathbb{R}^d} x_l |u(x)|^2 dx}_{=q_l} - \sum_{l=1}^{d} C_{il} q_l \underbrace{\int_{\mathbb{R}^d} |u(x)|^2 dx}_{=1} + p_i \underbrace{\int_{\mathbb{R}^d} |u(x)|^2 dx}_{=1} \\ &= p_i.\end{aligned}$$

Therefore the momentum average $\langle u| - i\epsilon\nabla_x u\rangle$ equals the parameter p.

Energy conservation
The presented variational Gaussian wave packets are energy and norm preserving as proved in the following theorem:

Theorem 1.2

The time-dependent variational approximation (1.1) is norm and energy preserving, i.e. for a fixed time t_0 it holds

$$\langle u(t_0) \mid Hu(t_0)\rangle = \langle u(t) \mid Hu(t)\rangle \quad \text{and} \quad \|u(t_0)\| = \|u(t)\| \tag{1.6}$$

for all $t \in \mathbb{R}$.

This theorem and a proof can be found in [LL, p.14].

Proof.
Recall that the approximation $u(t)$ is determined by the condition that $\partial_t u(t) \in \mathcal{T}_u\mathcal{M}$. Then we calculate

$$\begin{aligned}
\frac{d}{dt}\langle u(t) \mid Hu(t)\rangle &= \langle \partial_t u(t) \mid Hu(t)\rangle + \langle u(t) \mid H\partial_t u(t)\rangle \\
&= \langle \partial_t u(t) \mid Hu(t)\rangle + \langle Hu(t) \mid \partial_t u(t)\rangle \\
&= \langle \partial_t u(t) \mid Hu(t)\rangle + \overline{\langle \partial_t u(t) \mid Hu(t)\rangle} \\
&= 2\,\mathrm{Re}\langle \partial_t u(t) \mid Hu(t)\rangle \\
&= 2\,\mathrm{Re}\langle \partial_t u(t) \mid i\hbar\partial_t u(t)\rangle = 0,
\end{aligned}$$

where the second equality holds as H is a self-adjoint operator. Thus we have the energy conservation over time. For the norm preservation we note that $u(t) \in \mathcal{T}_u\mathcal{M}$, therefore we can insert it in (1.1) and get

$$\begin{aligned}
\frac{d}{dt}\|u(t)\|^2 &= \langle u(t) \mid \partial_t u(t)\rangle + \langle \partial_t u(t) \mid u(t)\rangle \\
&= 2\,\mathrm{Re}\langle u(t) \mid \partial_t u(t)\rangle \\
&= 2\,\mathrm{Re}\langle u(t) \mid \frac{1}{i\hbar} Hu(t)\rangle = 0.
\end{aligned}$$

□

1.1.2 Different types of Gaussians

Due to Heller we define three different types of Gaussian functions. The following definitions can be found in [RPS+15, p.5f]. The type of a Gaussian is determined by the matrix C. Later we will see in numerical experiments that the stability of the propagation depends on which of the following types is choosen.

Frozen Gaussians
We call a Gaussian frozen, if the diagonal elements of C are kept fixed during the whole propagation. These type of Gaussian is usually used for quantum and semiclassical dynamics simulations. Recent studies have found that the choice of C is crucial for stable dynamics [RPS+15, p.5].

Separable Gaussians
We call a Gaussian separable, if the C is a diagonal matrix, but its entries may change during the propagation.

Thawed Gaussians
We call a Gaussian thawed, if the matrix C contains diagonal and off-diagonal elements. This allows coupling between different modes and is the most general choice we can have in these kind of approximations.

The original idea of Heller was that a single thawed Gaussian is not flexible enough to describe a full quantum system. Therefore he introduced linear combinations of frozen Gaussians (with constant widths) in [Hel81]. There has been done much work with this kind of approximations, see for example in [RPS+15, p.2f].

1.2 Idea of the algorithm

In this section we will present an existing method of solving (1.8) and the problems with this, as it relies on the ansatz of the normal equation. Further we explain the idea for an algorithm which is based on an orthogonalization that avoids the normal equation.

1.2.1 Current methods

Let $\lambda_{j\alpha}$ be an arbitrary parameter of the j-th Gaussian of u. In [RPS+15] they apply the Dirac-Frenkel Variational Principle (1.1) with $v = \lambda_{j\alpha} a_j \frac{\partial g_j}{\partial \lambda_{j\alpha}}$. We get

$$\begin{aligned} 0 =& \langle \lambda_{j\alpha} a_j \frac{\partial g_j}{\partial \lambda_{j\alpha}} \left| H - i\frac{\partial}{\partial t} \right| u(t) \rangle \\ =& \langle \lambda_{j\alpha} a_j \frac{\partial g_j}{\partial \lambda_{j\alpha}} | Hu(t) \rangle - \langle \lambda_{j\alpha} a_j \frac{\partial g_j}{\partial \lambda_{j\alpha}} | i\dot{u}(t) \rangle, \end{aligned}$$

what is equivalent to

$$\begin{aligned}
&\Leftrightarrow \langle \lambda_{j\alpha} a_j \frac{\partial g_j}{\partial \lambda_{j\alpha}} | i\dot{u}(t) \rangle = \langle \lambda_{j\alpha} a_j \frac{\partial g_j}{\partial \lambda_{j\alpha}} | Hu(t) \rangle \\
&\Leftrightarrow -i \langle \frac{\partial g_j}{\partial \lambda_{j\alpha}} | i\dot{u}(t) \rangle = -i \langle \frac{\partial g_j}{\partial \lambda_{j\alpha}} | Hu(t) \rangle \\
&\Leftrightarrow \langle \underbrace{i \frac{\partial g_j}{\partial \lambda_{j\alpha}}}_{=A_j} | \underbrace{i\dot{u}(t)}_{=A\dot{y}} \rangle = \langle \underbrace{i \frac{\partial g_j}{\partial \lambda_{j\alpha}}}_{=A_j} | \underbrace{Hu(t)}_{=b} \rangle,
\end{aligned}$$

if A_j is the j-th column of A. Writing this in matrix notation and defining the multiplication of L^2-functions as the inner product of these functions we get $A^T A\dot{y} = A^T b$. Therefore [RPS+15] gets $\dot{y}$ via solving the normal equations. If the matrix A is ill-conditioned, then we expect the problem to be even worse conditioned, as it holds $\kappa(A^T A) = \kappa(A)^2$, see for example in [DH08, p.75]. The aim of this thesis is to present and analyse an algorithm which avoids the normal equation.

1.2.2 Orthogonalization

Let us now define a function Λ which maps the parameters y to the corresponding L^2-function defined as in (1.3). With this definition we can write $u = \Lambda(y)$ and $\dot{u} = \Lambda'(y)\dot{y}$. Further we define $A := \Lambda'(y)$ and $b := \frac{1}{i\epsilon} H\Lambda(y)$. With this, the Dirac-Frenkel Variational Principle (1.1) changes to

$$\langle v | \Lambda'(y)\dot{y} - \frac{1}{i\epsilon} H\Lambda(y) \rangle = \langle v | A\dot{y} - b \rangle = 0 \quad \forall v \in \mathcal{T}_u \mathcal{M}. \tag{1.7}$$

The minimization condition (1.2) reads as

$$\left\| \Lambda'(y)\dot{y} - \frac{1}{i\epsilon} H\Lambda(y) \right\| = \|A\dot{y} - b\| = \min!. \tag{1.8}$$

Instead of using the normal equation, we made an orthonormalization of the matrix A. First assume that this A is the matrix defined as above. Later we will use a slightly different matrix, which we will call X, to make the whole problem a real valued one. This will be specified later, for now assume we have a matrix A whose columns are L^2-functions.

Modified Gram-Schmidt method

The orthonormalization is obtained by a modified Gram-Schmidt method. One can find the definition of the algorithm and a detailed explanation in [Kan05, p. 104ff]. We only have to use an appropriate inner product instead of the euclidean one, as we have L^2-functions instead of vectors.

As above $A_j = \frac{\partial u}{\partial y_j}$ is the partial derivative of u with respect to the j-th parameter of u. With this, the modified Gram-Schmidt method has the following

form:

$$q_1 := \frac{A_1}{\|A_1\|}.$$

Let now $q_1, ..., q_{j-1}$ be a orthonormal system. Further define

$$r_{ij} :=< q_i, \tilde{q_j} > \text{ for } j < i, \text{ with } \tilde{q_j} := A_j - \sum_{l=1}^{i-1} r_{lj} q_l.$$

Then finally

$$r_{jj} := \left\| A_j - \sum_{l=1}^{j-1} r_{lj} q_l \right\| \text{ and } q_j := \frac{1}{r_{jj}} \left(A_j - \sum_{l=1}^{j-1} r_{lj} q_l \right).$$

The Gram-Schmidt method in this version is numerically stable [Kan05, p.106]. In total one gets a decomposition of the form $A = QR$, where $Q = (q_1|...|q_n)$ and $R_{ij} = r_{ij}$.

Equation of motion
As the constructed family $q_1, ...q_n \in L^2(\mathbb{R}^d)$ is an orthonormal family with respect to the used inner product, it holds $< q_i, q_j >= \delta_{ij}$. Inserting a $q_i, i \in 1, .., n$ in the Galerkin condition (1.1), we get

$$\begin{aligned} 0 &= \langle q_i \mid QR\dot{y} - b \rangle \\ &= \langle q_i \mid (q_1|...|q_n) R\dot{y} - b \rangle \\ &= \sum_{l=1}^{n} \underbrace{\langle q_i | q_l \rangle}_{=\delta_{il}} (R\dot{y})_l - \langle q_i | b \rangle \\ &= (R\dot{y})_i - \langle q_i | b \rangle. \end{aligned}$$

Defining $Q^* b := (\langle q_i | b \rangle)_{i=1}^n$ we can write this vector valued as

$$0 = R\dot{y} - Q^* b \Leftrightarrow R\dot{y} = Q^* b. \tag{1.9}$$

Therefore we get the new parameters $y(t_1)$ at time t_1, by evaluating the solution of the ordinary differential equation (ODE)

$$\dot{y}(t) = R^{-1} Q^* b, \text{ with } y(t_0) = y, \tag{1.10}$$

at time t_1, where y are the parameters at time t_0. With the new parameters we get a new A and b. With them we can redo the orthonormalization for the next time step and so on.

1.2.3 Real formulation of the problem

As u is a complex valued function in $L^2(\mathbb{R}^d)$ one could think of using the inner

product

$$< f, g > = \int_{\mathbb{R}^d} \overline{f(x)} g(x)\ dx. \tag{1.11}$$

If we use this inner product in the Gram Schmidt method from above, the solution of the ODE in (1.10) would in general be complex. This would mean that especially the parameters p and q could become complex. As we have seen in section 1.1, these parameters are related to the momentum and position average, therefore they should stay real. Still this inner product will be useful to re-formulate it as a real-value problem.
First we write the complex parameters C_j and ζ_j with its real- and imaginary parts, i.e. $C_j = A_j + iB_j$ and $\zeta_j = \gamma_j + i\delta_j$. With this u writes as

$$u(x) = \sum_{j=1}^{m} \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q_j)^T(A_j+iB_j)(x-q_j) + p_j^T(x-q_j) + \gamma_j + i\delta_j\right)\right). \tag{1.12}$$

Recall the definition $\dot{u} = \Lambda'(y)\dot{y}$ with the only difference that y contains now only real parameters, i.e. $y = (q_j, p_j, A_j, B_j, \gamma_j, \delta_j)_{j=1}^m$. Define $X(y) := \Lambda'(y)$ and separate it into real and imaginary part. Then we have

$$X(y) = \begin{pmatrix} \mathrm{Re}\,\frac{\partial u}{\partial q_1},\ \mathrm{Re}\,\frac{\partial u}{\partial p_1},\ \mathrm{Re}\,\frac{\partial u}{\partial A_1},\ \mathrm{Re}\,\frac{\partial u}{\partial B_1},\ \mathrm{Re}\,\frac{\partial u}{\partial \gamma_1},\ \mathrm{Re}\,\frac{\partial u}{\partial \delta_1},\ \mathrm{Re}\,\frac{\partial u}{\partial q_2}, \dots \\ \mathrm{Im}\,\frac{\partial u}{\partial q_1},\ \mathrm{Im}\,\frac{\partial u}{\partial p_1},\ \mathrm{Im}\,\frac{\partial u}{\partial A_1},\ \mathrm{Im}\,\frac{\partial u}{\partial B_1},\ \mathrm{Im}\,\frac{\partial u}{\partial \gamma_1},\ \mathrm{Im}\,\frac{\partial u}{\partial \delta_1},\ \mathrm{Im}\,\frac{\partial u}{\partial q_2}, \dots \end{pmatrix}. \tag{1.13}$$

Note that each entry of this matrix is now a real-valued L^2-function. Now we use a real inner product, which we denote with $(\cdot,\cdot)$ and which is defined as

$$\left(\begin{pmatrix} \mathrm{Re}\,\frac{\partial u}{\partial y_i} \\ \mathrm{Im}\,\frac{\partial u}{\partial y_i} \end{pmatrix}, \begin{pmatrix} \mathrm{Re}\,\frac{\partial u}{\partial y_j} \\ \mathrm{Im}\,\frac{\partial u}{\partial y_j} \end{pmatrix}\right) := \int_{\mathbb{R}^d} \mathrm{Re}\,\frac{\partial u}{\partial y_i}\,\mathrm{Re}\,\frac{\partial u}{\partial y_j} + \mathrm{Im}\,\frac{\partial u}{\partial y_i}\,\mathrm{Im}\,\frac{\partial u}{\partial y_j}\ dx \tag{1.14}$$

$$= \mathrm{Re}\langle \frac{\partial u}{\partial y_i}, \frac{\partial u}{\partial y_j} \rangle. \tag{1.15}$$

We will use the inner product $(\cdot,\cdot)$ in the Gram Schmidt method on the matrix $X(y)$ to get the wanted decomposition $X(y) = QR$. With Q and R we solve the ODE from (1.10) to get new parameters at time $t+\tau$. To calculate the real inner product $(\cdot,\cdot)$, we will calculate the complex inner product of these functions and then take the real part, because of relation (1.15). The calculation of the complex integrals will be discussed in chapter 2.

Now we want to formulate the Dirac-Frenkel variational condition in (1.1) for the real problem and we will do this in the same way as in [Lub08, p.48ff]. First write u in $u = v + iw$, where v, w are now real valued L^2 functions. Further note that our Hamiltonian H is a real operator. Now insert $u = v + iw$ in the Schrödinger equation and separate it into real and imaginary part, we get

$$\begin{aligned} \epsilon\dot{v} &= Hw \\ \epsilon\dot{w} &= -Hv. \end{aligned} \tag{1.16}$$

Now we define the matrix

$$J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Note that the multiplication with J corresponds to the multiplication with the imaginary unit i. With that we can rewrite (1.16) to

$$\begin{pmatrix} \dot{v} \\ \dot{w} \end{pmatrix} = \frac{1}{\epsilon} J^{-1} \begin{pmatrix} Hv \\ Hw \end{pmatrix}.$$

Taking now the real part of the Dirac-Frenkel variational principle (1.1) yields the following real formulation

$$\left(\mu \;\middle|\; \dot{u} - \frac{1}{\epsilon} J^{-1} \begin{pmatrix} Hv \\ Hw \end{pmatrix} \right) = 0 \quad \text{for all } \mu \in \mathcal{T}_u \mathcal{M}. \tag{1.17}$$

In the notation with $X(y)$ we have

$$\left(\mu \;\middle|\; X(y)\dot{y} - \frac{1}{\epsilon} J^{-1} \begin{pmatrix} Hv \\ Hw \end{pmatrix} \right) = 0 \quad \text{for all } \mu \in \mathcal{T}_u \mathcal{M}. \tag{1.18}$$

Applying the modified Gram-Schmidt method with the real inner product, we get from (1.18) via an analogous calculation as for equation (1.9) an equation of motion

$$R\dot{y} = \frac{1}{\epsilon} \left(\mathrm{Im}\langle q_i | Hu \rangle \right)_{j=1}^{n}, \tag{1.19}$$

where $(q_j)_{j=1}^n$ is the orthonormal family we get from the Gram-Schmidt method.

Calculating the inner product
The hardest task in the implementation of this algorithm is the calculation of all appearing inner products, as these inner products are high-dimensional integrals over the whole $\mathbb{R}^d$. The curse of dimensionality could for example be managed by using Monte-Carlo methods.
We will introduce an other way to calculate these kind of integrals using so called Hagedorn wave packets, which allow us calculations without an approximation error. This will be explained in detail in chapter 2.

1.3 Appearing derivatives

In section (1.2) we saw that we need the time derivatives of all the parameters to get the matrix A, or in the real formulation, of the matrix $X(y)$, i.e. we want to calculate $\partial u / \partial y_j$, where y_j is a parameter of the approximation u. Recall

$$u(x) = \sum_{j=1}^{m} a_j g_j(x), \text{ where}$$
$$g_j(x) = \exp\left(\frac{i}{\epsilon} \left(\frac{1}{2} (x - q_j)^T C_j (x - q_j) + p_j^T (x - q_j) + \zeta_j \right) \right).$$

A straight forward calculation gives us the derivatives with respect to the (in general) complex parameters

$$\frac{\partial u}{\partial a_j} = g_j(x) \tag{1.20}$$

$$\frac{\partial u}{\partial (q_j)_\mu} = \frac{i}{\epsilon} a_j g_j(x)(-C_j x + C_j q_j - p_j)_\mu \tag{1.21}$$

$$\frac{\partial u}{\partial (p_j)_\mu} = \frac{i}{\epsilon} a_j g_j(x)(x - q_j)_\mu \tag{1.22}$$

$$\frac{\partial u}{\partial C_j} = \frac{i}{2\epsilon} a_j g_j(x)(x - q_j)^T (x - q_j) \tag{1.23}$$

$$\frac{\partial u}{\partial \zeta_j} = \frac{i}{\epsilon} a_j g_j(x). \tag{1.24}$$

When deriving to the real parameters, the derivatives with respect to q and p do not change. For the other parameters we get

$$\frac{\partial u}{\partial A_j} = \frac{\partial u}{\partial C_j} \tag{1.25}$$

$$\frac{\partial u}{\partial B_j} = i \frac{\partial u}{\partial C_j} \tag{1.26}$$

$$\frac{\partial u}{\partial \gamma_j} = \frac{\partial u}{\partial \zeta_j} \tag{1.27}$$

$$\frac{\partial u}{\partial \delta_j} = i \frac{\partial u}{\partial \zeta_j}. \tag{1.28}$$

We can see that it is enough to know the complex derivatives. To get inner products of the form (1.14) we need to calculate the complex inner product of these functions and take the real part of the result. Calculating the complex inner product will be presented in chapter 2.

As in the Hamiltonian H appears the Laplace-operator, we also need to calculate $-\frac{\epsilon^2}{2}\Delta u$. An explicit calculation gives us

$$-i\epsilon \nabla_x g_j(x) = (C_j(x - q_j) + p_j) g_j(x)$$

and

$$-\frac{\epsilon^2}{2}\Delta g_j(x) = \Gamma_j(x) g_j(x), \text{ with}$$

$$\Gamma_j(x) = \left(\frac{1}{2}(x - q_j)^T C_j^2 (x - q_j) + p_j^T C_j (x - q_j) + \frac{1}{2}|p_j|^2 - \frac{i\epsilon}{2} tr(C_j) \right).$$

The computation for a single Gaussian can also be found in [LL, p.24]. All together we get

$$-\frac{\epsilon^2}{2}\Delta u(x) = \sum_{j=1}^{m} a_j g_j(x) \Gamma_j(x), \tag{1.29}$$

with the $\Gamma_j(x)$ from above.

Multiplication of Gaussians
When using the inner product (1.11), we will need to know what the multiplication of two Gaussians with the form as in (1.3) is, i.e. we want to calculate $\overline{g_j(x)}g_l(x)$, where g_j has the parameters (q_j, p_j, C_j, ζ_j) and g_l the parameters (q_l, p_l, C_l, ζ_l). If we define

$$\begin{aligned}
C &:= -\overline{C_j} + C_l, \\
q &:= (\mathrm{Im}(C))^{-1}(\mathrm{Im}(-\overline{C_j})q_j + \mathrm{Im}(C_l)q_l), \\
p &:= -p_j + p_l - 2\,\mathrm{Re}(-\overline{C_j})q_j - 2\,\mathrm{Re}(C_l)q_l - \mathrm{Re}(C)q, \\
\zeta &:= -\overline{\zeta_j} + \zeta_l - q_j^T\overline{C_j}q_j + p_j^T q_j + q_l^T C_l q_l - p_l^T q_l - q^T C q + p^T q,
\end{aligned} \tag{1.30}$$

then we have

$$< g_j, g_l >= \int_{\mathbb{R}^d} \overline{g_j(x)}g_l(x)dx = \int_{\mathbb{R}^d} g(x)dx,$$

where $g(x)$ is the Gaussian with parameters (q, p, C, ζ), defined as above.

Chapter 2

Hagedorn wave packets

In 1981 and 1985 George Hagedorn presented a parameter based orthonormal basis of $L^2(\mathbb{R}^d)$ in [Hag81] and [Hag85]. These functions are multi-dimensional Gaussians multiplied with a polynomial and are now called Hagedorn functions. This theory was developed further in the important publication of Hagedorn [Hag98]. Here, the Hagedorn functions are formally defined by rising and lowering operators and main properties are proofed. In this chapter, we will recap and summarize main properties of the Hagedorn functions that will be needed later for the formulation of the novel algorithm. The definitions and theorems presented in this chapter are mainly taken from [Tro17] and [Lub08].

2.1 Theoretical background

First of all, recall that the matrices C_j in (1.3) are complex symmetric with positive definite imaginary part. This property allows us to find a certain decomposition of C in two matrices Q and P.

Lemma 2.1

Let C be a complex symmetric matrix in $\mathbb{C}^{d\times d}$ with positive definite imaginary part. Then there exist two invertible matrices Q and P with $C = PQ^{-1}$.
Conversely, arbitrary matrices $Q, P \in \mathbb{C}^{d\times d}$ which satisfy

$$(1)\ Q^TP - P^TQ = 0 \qquad (2)\ Q^*P - P^*Q = 2iI,$$

are invertible and the matrix defined as $C = PQ^{-1}$ is complex symmetric with positive definite imaginary part.

This result and the proof can be found in [Lub08, p.123f].

Proof.
Suppose we have matrices Q and P which satisfy condition (1) and (2). Let v

be an arbitrary vector in $\mathbb{C}^d$. Now multiply equation (2) form the left with v^* and from the right with v. Then we have

$$\begin{aligned} & v^*Q^*Pv - v^*P^*Qv = 2iv^*v \\ \Leftrightarrow\ & (Qv)^*(Pv) - (Pv)^*Qv = 2i\|v\|^2. \end{aligned}$$

This shows that $Qv = 0$ implies directly $v = 0$, thus $\ker Q = \{0\}$ which implies Q is invertible. The same argument holds for P.
Now multiply equation (1) from the left with $(Q^{-1})^T$ and from the right with Q^{-1}. Then we have

$$PQ^{-1} - (Q^{-1})^T P^T = 0,$$

therefore the matrix $C = PQ^{-1}$ is complex symmetric. Further it holds

$$\begin{aligned} (\operatorname{Im} C)QQ^* &= \frac{1}{2i}(C - C^*)QQ^* = \frac{1}{2i}(PQ^{-1} - (Q^{-1})^*P^*)QQ^* \\ &\overset{(2)}{=} \frac{1}{2i}2i(QQ^*)^{-1}QQ^* = I, \end{aligned}$$

where the last line can be obtained from equation (2). Therefore we have $\operatorname{Im} C = (QQ^*)^{-1}$ and as QQ^* has only positive eigenvalues, $\operatorname{Im} C$ is positive definite. Conversely if we define $Q = (\operatorname{Im} C)^{-1/2}$ and $P = CQ$, it is easy to see that Q and P satisfy equation (1) and (2).

$$\begin{aligned} &(1)\ Q^TP - P^TQ = Q^TCQ - Q^TCQ = 0 \\ &(2)\ Q^*P - P^*Q = Q^*CQ - (CQ)^*Q = Q^*(C - C^*)Q \\ &\quad = Q^*(2i\operatorname{Im} C)Q = 2i((\operatorname{Im} C)^{-1/2})^* \operatorname{Im} C(\operatorname{Im} C)^{-1/2} = 2iI. \end{aligned}$$

□

Remark
In Hagedorn's original papers he used matrices A and iB instead of Q and P. This gives prefactors in some of the following formulas. As the current literature mainly uses the QP notation, we will do it as well. Also the next definition of the Hagedorn functions is not the original one of Hagedorn and can be found in [Tro17, p.72f].

Definition 2.1: Hagedorn functions

Let C be a complex symmetric matrix with positive definite imaginary part, $Q = \mathrm{Im}(C)^{-1/2}$, $P = CQ$ and $p, q \in \mathbb{R}^d$. Then the first Hagedorn function is defined as

$$\varphi_0(q,p,Q,P,x) := (\pi\epsilon)^{-d/4} \det(Q)^{-1/2} e^{\frac{i}{2\epsilon}(x-q)^T PQ^{-1}(x-q) + \frac{i}{\epsilon}p^T(x-q)}. \tag{2.1}$$

The higher order Hagedorn functions are defined recursively. Let $k \in \mathbb{N}^d$ now be a multi-index and $M = Q^{-1}\bar{Q}$. Then the k-th Hagedorn function $\varphi_k(q,p,Q,P,x)$ is defined as

$$\varphi_k(q,p,Q,P,x) := \frac{1}{\sqrt{2^{|k|}k!}} p_k\left(\frac{1}{\sqrt{\hbar}}Q^{-1}(x-q)\right)\varphi_0(q,p,Q,P,x), \quad x \in \mathbb{R}^d, \tag{2.2}$$

where p_k are multivariate polynomials which are defined via the following recursion

$$p_0 = 1, \quad \left(p_{k+e_j}\right)_{j=1}^d = \hat{B}^\dagger p_k \quad \text{where} \quad \hat{B}^\dagger = 2x - M\nabla_x.$$

For the computation of the multivariate integrals we need to have some certain Hagedorn functions. Therefore we need all polynomials p_k up to a certain order. We will see that it suffices to calculate all p_k up to order 4. A list with all needed polynomials p_k can be found in the appendix (A.1). With this computation one can see that the φ_k are indeed a Gaussian times a multi-variate polynomial.

2.1.1 Hagedorn functions via ladder operators

As mentioned before, we can define the Hagedorn functions via ladder operators, as it was originally done by Hagedorn. The way of defining the Hagedorn functions with the ladder operators can be found in [Hag98], [Lub08] or [Tro17]. Taking the notation from [Lub08] we define for a $\psi \in \mathcal{S}(\mathbb{R}^d)$,

$$(\hat{q}\psi)(x) = x\psi(x), \quad (\hat{p}\psi)(x) = -i\epsilon\nabla\psi(x), \ x \in \mathbb{R}^d.$$

With this operators we can define the ladder operators:

$$A = A(q,p,Q,P) = -\frac{i}{\sqrt{2\epsilon}}\left(P^T(\hat{q}-q) - Q^T(\hat{p}-p)\right) \tag{2.3}$$

$$A^\dagger = A^\dagger(q,p,Q,P) = \frac{i}{\sqrt{2\epsilon}}\left(P^*(\hat{q}-q) - Q^*(\hat{p}-p)\right). \tag{2.4}$$

Now let e_i be the d-dimensional i-th unit vector, i.e. $e_i = (0, .., 0, 1, 0, .., 0)$. Then for a $k \in \mathbb{N}^d$ and the φ_0 from (2.1), the φ_k can be defined as

$$\varphi_{k+e_i}(q,p,Q,P,x) = \frac{1}{\sqrt{k_i+1)}} A_i^\dagger \varphi_k(q,p,Q,P,x). \tag{2.5}$$

Further it can be found in [Lub08, p.128] that

$$\varphi_{k-e_i}(q,p,Q,P,x) = \frac{1}{\sqrt{k_i}} A_i \varphi_k(q,p,Q,P,x). \tag{2.6}$$

This shows that the operators $A^\dagger$ and A can be seen as raising and lowering operators. The definition via ladder operators is equivalent to the one in 2.1. An other interesting fact is that the functions φ_k form a complete orthonormal family of functions of $L^2(\mathbb{R}^d)$. One can find this in theorem 2.3 in [Lub08, p.128].

2.1.2 Properties of Hagedorn functions

A very useful property of the Hagedorn functions is that one can easily calculate its Fourier transform. The Fourier transform with the semiclassical parameter ϵ reads

$$\mathcal{F}^\epsilon \varphi(\xi) = (2\pi\epsilon)^{-d/2} \int_{\mathbb{R}^d} \varphi(x) e^{-\frac{i}{\epsilon} x^T \xi} \, dx, \quad \xi \in \mathbb{R}^d,$$

for all $\varphi \in L^2(\mathbb{R}^d)$. This can be found in [Tro17, p.74]. The following theorem is crucial for the presented algorithm, as it will be clarified later.

Theorem 2.1

Let $k \in \mathbb{N}^d$ be a multi-index and φ_k the corresponding Hagedorn function defined as in definition (2.1). Then it holds

$$\mathcal{F}^\epsilon \varphi_k(q,p,Q,P,\cdot)(\xi) = e^{-\frac{i}{\epsilon} p^T q} \varphi_k(p,-q,P,-Q,\xi). \tag{2.7}$$

This result and a full proof of it can be found in [Tro17, p.75f]. The original proof was done by Hagedorn and can be found in [Hag85, p.371]. He proofed this via an induction over $|k|$, which is a lengthy calculation.
The importance of this result can not be emphasized enough. The left side is, in general, a high-dimensional integral of a $L^2(\mathbb{R}^d)$-function, while the right side is only an evaluation of the same function at a certain point. In addition, we don't have any approximation errors, we get the exact values for the integrals.

Another important result is a recursion relation of the Hagedorn functions. The following theorem will allow us to calculate integrals of a Gaussian times a low-dimensional polynomial.

Theorem 2.2

Let $k \in \mathbb{N}^d$ be a multi-index, $Z = (q, p, Q, P)$ and $\varphi_k(Z, x)$ the k-th Hagedorn function as in Definition 2.1. Further let $e_j = (0, ..., 0, 1, 0, ..., 0)$ be the j-th d-dimensional unit vector and define $\varphi_{k-e_j} = 0$, if $k - e_j \notin \mathbb{N}^d$. Then the three-term recurrence

$$Q\left(\sqrt{k_j+1}\varphi_{k+e_j}(Z,x)\right)_{j=1}^{d} = \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_k(Z,x) - \overline{Q}\left(\sqrt{k_j}\varphi_{k-e_j}(Z,x)\right)_{j=1}^{d} \tag{2.8}$$

holds for $\forall k \in \mathbb{N}^d$.

The tree-term recursion formula as above can be found in [Lub08, p.128] and a proof of it for the centred case with $q = 0$ in [Tro17, p.84]. Here we will proof this also for the case $q \neq 0$.

Proof.
We will proof the statement with help of the ladder operators from (2.3) and (2.4). Multiply (2.5) with Q and bring the constant on the other side. We get

$$\begin{aligned} Q\left(\sqrt{k_j+1}\varphi_{k+e_j}\right)_{j=1}^{d} &= QA^\dagger\varphi_k \\ &= \frac{i}{\sqrt{2\epsilon}}\left(QP^*(x-q) - QQ^*(\hat{p}-p)\right)\varphi_k. \end{aligned} \tag{2.9}$$

Using equation (2) from lemma 2.1 we get

$$\begin{aligned} Q^*P - P^*Q = 2iI &\Leftrightarrow QQ^*PQ^{-1} - QP^* = 2iI \\ &\Leftrightarrow QP^* = QQ^*PQ^{-1} - 2iI. \end{aligned}$$

We insert this is (2.9), use that $P^T = Q^TPQ^{-1}$ by equation (1) from lemma 2.1 and $QQ^* = \overline{QQ^*} = \overline{Q}Q^T$. All together we get

$$\begin{aligned} 2.9 &= \frac{i}{\sqrt{2\epsilon}}\left(-2iI(x-q) + QQ^*PQ^{-1}(x-q) - QQ^*(\hat{p}-p)\right)\varphi_k \\ &= \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_k + \frac{i}{\sqrt{2\epsilon}}\left(QQ^*PQ^{-1}(x-q) - QQ^*(\hat{p}-p)\right)\varphi_k \\ &= \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_k + \frac{i}{\sqrt{2\epsilon}}\left(\overline{Q}\underbrace{Q^TPQ^{-1}}_{=P^T}(x-q) - \overline{Q}Q^T(\hat{p}-p)\right)\varphi_k \\ &= \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_k - \overline{Q}\frac{-i}{\sqrt{2\epsilon}}\left(P^T(x-q) - Q^T(\hat{p}-p)\right)\varphi_k \\ &= \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_k - \overline{Q}A\varphi_k \overset{2.6}{=} \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_k - \overline{Q}\left(\sqrt{k_j}\varphi_{k-e_j}\right)_{j=1}^{d}. \end{aligned}$$

□

2.2 Application of Hagedorn's wave packets

We now present how to efficiently compute high-dimensional integrals of a Gaussian times a polynomial. We will get this by applying theorem 2.1 and 2.2. A complete list for all intergrals of the form $\int_{\mathbb{R}^d} f(x)g(x)dx$, with f a polynomial of degree ≤ 4, can be found in appendix (A.2). We will present first the case where $f(x) = x_i$.

Consider a Gaussian g as in (1.3). Apply the tree-term recursion (theorem 2.2) to the case $k = (0,..,0)$. As $k - e_j \notin \mathbb{N}^d \ \forall j$, the last term of the recursion vanishes. Then we have

$$\begin{aligned} & Q\left(\varphi_{e_j}(Z,x)\right)_{j=1}^d = \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_0(Z,x) \\ \Leftrightarrow & \left(\sum_{l=1}^d Q_{jl}\varphi_{e_l}(Z,x)\right)_{j=1}^d = \sqrt{\frac{2}{\epsilon}}(x-q)\varphi_0(Z,x). \end{aligned} \tag{2.10}$$

Now take only the i-th component of the equation and integrate over the whole $\mathbb{R}^d$. Then we can apply theorem 2.1 and get for the left side

$$\sum_{l=1}^d Q_{il}\int_{\mathbb{R}_d}\varphi_{e_l}(Z,x)dx = \sum_{l=1}^d Q_{il}(2\pi\epsilon)^{d/2}\mathcal{F}^\epsilon\varphi_{e_i}(0). \tag{2.11}$$

For the right side of the equation (2.10) we get

$$\begin{aligned} & \sqrt{\frac{2}{\epsilon}}\int_{\mathbb{R}^d}(x-q)_i\varphi_0(Z,x)dx \\ & = \sqrt{\frac{2}{\epsilon}}\int_{\mathbb{R}^d} x_i\varphi_0(Z,x)dx - \sqrt{\frac{2}{\epsilon}}\int_{\mathbb{R}^d} q_i\varphi_0(Z,x)dx \\ & = \sqrt{\frac{2}{\epsilon}}\int_{\mathbb{R}^d} x_i\varphi_0(Z,x)dx - \sqrt{\frac{2}{\epsilon}}q_i(2\pi\epsilon)^{d/2}\mathcal{F}^\epsilon\varphi_0(0). \end{aligned} \tag{2.12}$$

Inserting this and (2.11) in (2.10) and using definition 2.1 we get

$$\begin{aligned} & \sqrt{\frac{2}{\epsilon}}\int_{\mathbb{R}^d} x_i(\pi\epsilon)^{-d/4}\det(Q)^{-1/2}e^{-\frac{i}{\epsilon}\zeta}g(x)dx \\ & = (2\pi\epsilon)^{d/2}\left(\sqrt{\frac{2}{\epsilon}}q_i\mathcal{F}^\epsilon\varphi_0(0) + \sum_{l=1}^d Q_{il}\mathcal{F}^\epsilon\varphi_{e_l}(0)\right). \end{aligned}$$

Finally define $\tilde{Z} := (p,-q,P,-Q)$. We apply now theorem 2.1 to calculate the Fourier transform of the appearing Hagedorn functions. Then we get

$$\begin{aligned} & \int_{\mathbb{R}^d} x_i g(x)dx \\ & = (\pi\epsilon)^{d/4}\det(Q)^{1/2}(2\pi\epsilon)^{d/2}e^{\frac{i}{\epsilon}\zeta}\left(q_i e^{\frac{i}{\epsilon}p^Tq}\varphi_0(\tilde{Z},0) + \sqrt{\frac{\epsilon}{2}}\sum_{l=1}^d Q_{il}e^{\frac{i}{\epsilon}p^Tq}\varphi_{e_l}(\tilde{Z},0)\right). \end{aligned} \tag{2.13}$$

This shows us how we can calculate integrals of the type Gaussian times x_i. The same strategy can be applied with polynomials of higher order, the only difference is that one needs to solve a linear system of equations. For further details, we refer to Appendix A.2. We now provide an example, that explains how to use this information to calculate appearing complex inner products in the algorithm.

Example
Lets say we have an approximation with two Gaussians, i.e. $u(x) = g_1(x)+g_2(x)$. During the modified Gram Schmidt method, we will need to calculate the inner product of the derivatives $A_j = \partial u/\partial\zeta_1$ and $A_l = \partial u/\partial(p_2)_1$, where ζ_1 stems form the first and $(p_2)_1$ is the first component of p_2 from the second Gaussain. The derivatives were explicitly calculated in (1.24) and (1.22). The function $g(x)$ is the multiplication of $g_1(x)$ and $g_2(x)$ defined as in chapter 1.

$$\begin{aligned}
< A_j, A_l > =< \frac{\partial u}{\partial \zeta_1}, \frac{\partial u}{\partial p_2} >= \int_{\mathbb{R}^d} \overline{\frac{\partial u}{\partial \zeta_1}(x)} \frac{\partial u}{\partial p_2}(x) dx \\
= \int_{\mathbb{R}^d} \overline{\frac{i}{\epsilon} g_1(x)} \frac{i}{\epsilon} g_2(x)(x-q_2)_1 dx \\
= \frac{1}{\epsilon^2} \int_{\mathbb{R}^d} \overline{g_1(x)} g_2(x)(x-q_2)_1 dx \\
= \frac{1}{\epsilon^2} \int_{\mathbb{R}^d} g(x)(x-q_2)_1 dx \\
= \frac{1}{\epsilon^2} \left(\int_{\mathbb{R}^d} x_i g(x) dx - (q_2)_1 \int_{\mathbb{R}^d} g(x) dx \right).
\end{aligned}$$

The first integral can be calculated as shown in (2.13). The second one can be calculated in a similar way, it is just an evaluation of the first Hagedorn function $\varphi_0(x)$ at a certain point. An explicit formula can be found in appendix (A.2).

Chapter 3

The algorithm

In this chapter, it will be explained the algorithm, its structure and how ti can be implemented in Matlab R2018b. The following chart presents the work-flow of the algorithm.

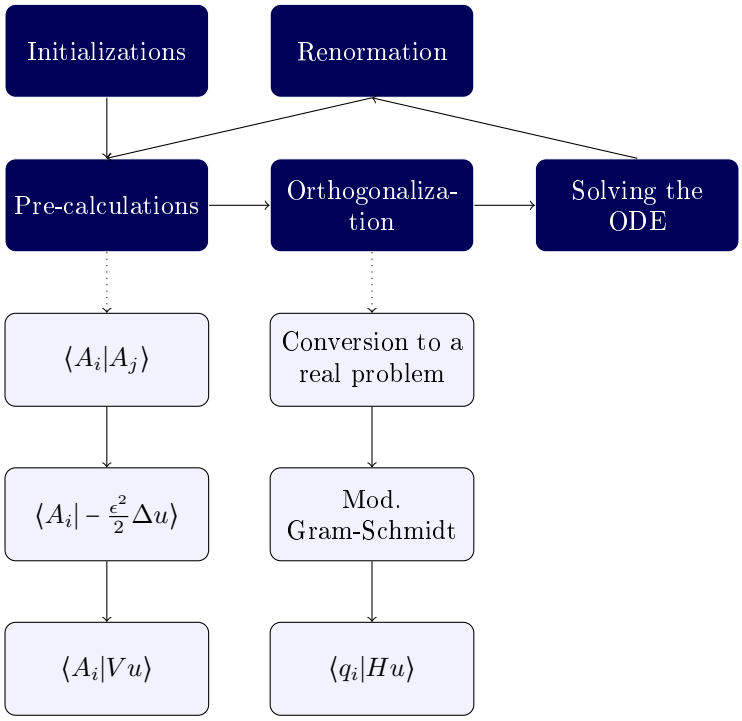

Figure 3.1: Structure of the Code

3.1 Renormation

As we have seen in theorem 1.2, the norm of u should stay constant. In the numerical experiments, due to round off-errors, it has been observed that the norm does not remain constant. Therefore, we renormalise u after each time step. This can efficiently done as follows. Assume that g_{jl} is the multiplication of two Gaussians g_j and g_l, defined as in chapter 1. It suffices to multiply a phase $e^{\frac{b}{\epsilon}}$ to each Gaussian. Further define $C = e^{\frac{2b}{\epsilon}}$. Then we have

$$1 \overset{!}{=} \|u\|_2^2 = \sum_{j,l=1}^{m} \int_{\mathbb{R}^d} \overline{e^{\frac{b}{\epsilon}} g_j(x)} e^{\frac{b}{\epsilon}} g_l(x) dx = e^{\frac{2b}{\epsilon}} \sum_{j,l=1}^{m} \int_{\mathbb{R}^d} g_{jl}(x) dx.$$

With that we get

$$\frac{1}{\|u\|_2^2} = C \;\Leftrightarrow b = \frac{\epsilon}{2} \log\left(\frac{1}{\|u\|_2^2}\right).$$

This calculation shows us that if we add $-ib$ to each ζ_j, u will be of unit norm. We must only calculate $\|u\|_2^2$ after each time step. We recall the definition of the Hagedorn function φ_0. Suppose g is a Gaussian with the parameters (q, p, C, ζ) and φ_0 the Hagedorn function of definition (2.1) with parameters (q, p, C). Then the following identity holds

$$\exp^{\frac{i\zeta}{\epsilon}} (\pi\epsilon)^{d/4} \det(Q)^{1/2} \varphi_0(x) = g(x).$$

Integrating the last identity over $\mathbb{R}^d$ and applying Theorem 2.1, we obtain that

$$\begin{aligned}
\int_{\mathbb{R}^d} g(x) dx &= \exp^{\frac{i\zeta}{\epsilon}} (\pi\epsilon)^{d/4} \det(Q)^{1/2} \int_{\mathbb{R}^d} \varphi_0(x) dx \\
&= \exp^{\frac{i\zeta}{\epsilon}} (\pi\epsilon)^{d/4} \det(Q)^{1/2} (2\pi\epsilon)^{d/2} \mathcal{F}^{\epsilon} \varphi_0(0) \\
&\overset{2.1}{=} \exp^{\frac{i\zeta}{\epsilon}} (\pi\epsilon)^{d/4} \det(Q)^{1/2} (2\pi\epsilon)^{d/2} e^{-\frac{i}{\epsilon} p^T q} \varphi_0(p, -q, P, -Q, 0).
\end{aligned}$$

To get now $\|u\|_2^2$ one must calculate all integrals of g_{jl} and sum them up.

3.2 Pre-calculations

The pre-calculation of all appearing inner products is the most intensive task in the implementation, as these are high dimensional integrals. As it will be shown in the numerical experiments presented in chapter 4, the pre-calculations make up most of the computational time of the algorithm.

Complex inner products
Let A_i, A_j be the derivatives of the approximation u with respect to a given parameter. In order to make the orthogonalization process efficient, we need

to pre-compute all inners products of the form (A_i, A_j), where $(\cdot,\cdot)$ is the real inner product (1.14). As shown in section 1.2.3 it holds

$$\left(\begin{pmatrix}\operatorname{Re} A_i\\ \operatorname{Im} A_i\end{pmatrix},\begin{pmatrix}\operatorname{Re} A_j\\ \operatorname{Im} A_j\end{pmatrix}\right) = \operatorname{Re}\langle A_i|A_j\rangle. \tag{3.1}$$

For a fast performance of the code, we calculate all possible combinations of inner products of the form $\langle A_i|A_j\rangle$. If d is the dimension of the system, one Gaussian has $2d+2$ derivatives. $2d$ derivatives for the parameters q_i, p_i $i \in \{1,...,d\}$, one derivative for C and one derivative for ζ. If we use m Gaussians we have a total number of $(m(2d+2))^2$ possible combinations. As the complex inner product is anti-linear it suffices to calculate only half of them.
One single calculation has the following structure:

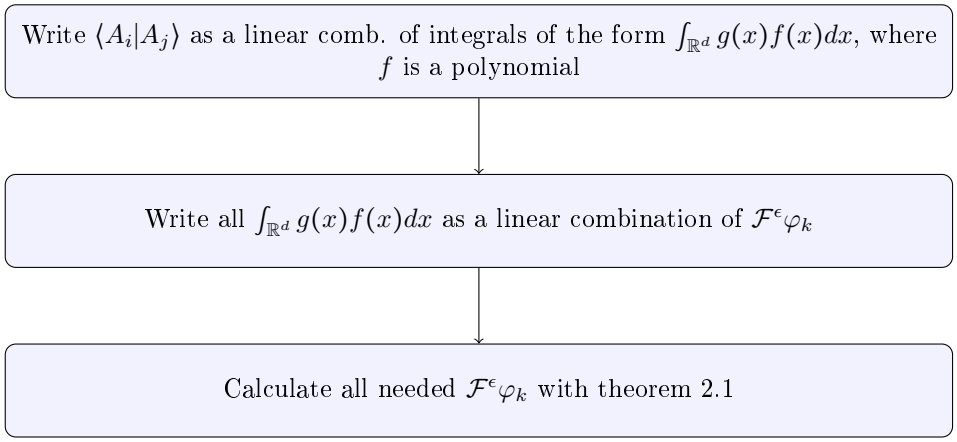

Figure 3.2: Calculation of $\langle A_i|A_j\rangle$

An explicit example for the computation was given in the example in section 2.2. Formulas for all cases can be found in (A.3.1).

Laplacian
In the same way as above, we want to pre-calculate all inner products of the form $\langle A_i| - \frac{\epsilon^2}{2}\Delta u\rangle$, where A_i is a derivative of u with respect to a parameter. We follow a similar strategy as presented in figure 3.2, where the only difference is that we replace the derivative A_j with $-\frac{\epsilon^2}{2}\Delta u$. Explicit formulas for the computation can be found in (A.3.2).
For the computation of the kinetic energy, we consider the general case where the parameters a_j for $j = 1...d$ are not necessarily constant. Recall that $\frac{\partial u}{\partial a_j} = g_j$. Then we can see that for the kinetic energy holds

$$\langle u| - \frac{\epsilon^2}{2}\Delta u\rangle = \sum_{j=1}^{m}\langle g_j| - \frac{\epsilon^2}{2}\Delta u\rangle = \sum_{j=1}^{m}\langle a_j| - \frac{\epsilon^2}{2}\Delta u\rangle.$$

Therefore, the above quantities with respect to the parameter a_j need to be pre-computed.

Potential
As a potential we choose the so called torsional potential, which is defined as

$$V(x) = \sum_{j=1}^{d} (1 - \cos(x_j)), \quad \forall x \in \mathbb{R}^d, \tag{3.2}$$

as considered in [FGL09, p.11]. We need to calculate scalar products of the form $\langle A_j | Vu \rangle$, where A_j is the derivative with respect to one of the parameters.

We remind the identity $\cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$, which has to be used to calculate efficiently all the inner products of V with respect to the derivatives of u. Let e_l be the l-th unit vector. Define for $j \in 1, ..., d$

$$\zeta_{jl}^{+} := \zeta_j + \epsilon q_l \qquad p_{jl}^{+} := p_j + \epsilon e_l$$
$$\zeta_{jl}^{-} := \zeta_j - \epsilon q_l \qquad p_{jl}^{-} := p_j - \epsilon e_l.$$

Further let g_{jl}^{+} be the Gaussian with parameters $(q_j, p_{jl}^{+}, C_j, \zeta_{jl}^{+})$ and analogously g_{jl}^{-} be the Gaussian with parameters $(q_j, p_{jl}^{-}, C_j, \zeta_{jl}^{-})$. If y_μ is the μ-th parameter of u we have

$$\langle A_\mu | Vu \rangle = \sum_{j,l=1}^{d} \langle A_\mu | g_j - \frac{1}{2} g_{jl}^{+}(x) - \frac{1}{2} g_{jl}^{-}(x) \rangle.$$

It suffices to calculate the inner products of A_j with an arbitrary Gaussian g. We refer to the appendix (A.3.2) for an explicit calculation of all the appearing complex inner products. Finally the real inner product is obtained considering the real part of the complex inner product. This results by the same calculation as in (1.15).

3.3 Orthogonalization

Conversion to a real problem
As already explained in section 1.2.3 we have to formulate the whole problem as a real one. All the pre-calculations are complex valued. We have already seen in (1.15) that it holds

$$\left(\begin{pmatrix} \operatorname{Re} A_i \\ \operatorname{Im} A_i \end{pmatrix}, \begin{pmatrix} \operatorname{Re} A_j \\ \operatorname{Im} A_j \end{pmatrix} \right) = \operatorname{Re} \langle A_i, A_j \rangle$$

and in (1.19) that

$$\begin{aligned} \left(\begin{pmatrix} \operatorname{Re} A_i \\ \operatorname{Im} A_i \end{pmatrix}, \frac{1}{\epsilon} J^{-1} \begin{pmatrix} \operatorname{Re} Hu \\ \operatorname{Im} Hu \end{pmatrix} \right) &= \frac{1}{\epsilon} \left(\begin{pmatrix} \operatorname{Re} A_i \\ \operatorname{Im} A_i \end{pmatrix}, \begin{pmatrix} \operatorname{Im} Hu \\ -\operatorname{Re} Hu \end{pmatrix} \right) \\ &= \frac{1}{\epsilon} \operatorname{Im} \langle A_i, Hu \rangle. \end{aligned}$$

Recall that in the real formulation we have more parameters as in the complex one, because one writes $C = A + iB$ and $\zeta = \gamma + i\delta$. Getting the inner products of the derivatives with respect to the real parameters is straightforward, as $du/dC = du/dA$ and $idu/dC = du/dB$. Analogously this holds for the derivatives with respect to γ and δ.

Modified Gram-Schmidt method

Now we have all ingredients to apply the modified Gram-Schmidt method introduced in chapter 1, with respect to the real inner product (1.14). It is important to underline the fact that we do not calculate the orthonormal family $\{q_1, ..., q_n\}$ explicitly. Along the implementation, it suffices to compute the inner products of the quantities $(q_i|A_j)$ or $(q_i|Hu)$.

Recall the definition of the entries of the R matrix. We can see that for $j < i$ it holds

$$\begin{aligned} r_{ij} &= \frac{1}{r_{ii}} \left(A_i - \sum_{l=1}^{i-1} r_{li} q_l \;\middle|\; A_j - \sum_{l=1}^{i-1} r_{lj} q_l \right) \\ &= \frac{1}{r_{ii}} \left[(A_i|A_j) - \sum_{l=1}^{i-1} r_{li}(A_j|q_l) + r_{lj}(A_i|q_l) + \sum_{l,m=1}^{i-1} r_{li} r_{lj} (q_i|q_j) \right]. \end{aligned}$$

As $(q_i|q_j) = \delta_{ij}$, it just suffices to know the inner products of the form $(A_j|q_l)$. These can be calculated recursively by the formula

$$\begin{aligned} (A_j|q_l) &= \left(A_j \middle| \frac{1}{r_{ll}} \left[A_l - \sum_{m=1}^{l-1} r_{ml} q_m \right] \right) \\ &= \frac{1}{r_{ll}} \left[(A_j|A_l) - \sum_{m=1}^{j-1} r_{ml} (A_j|q_m) \right]. \end{aligned}$$

Saving the already calculated values of $(A_j|q_l)$ avoids long recursions in the code and allows a fast computation.

Calculation of $\langle q_i|Hu\rangle$

In a similar way as in the Gram-Schmidt method, we can get the inner products of the form $(q_i|\frac{1}{\epsilon i}Hu)$. This will be the short notation for the inner product $\big((\mathrm{Re}\, q_l, \mathrm{Im}\, q_l)^T | \epsilon^{-1} (\mathrm{Im}\, Hu, -\mathrm{Re}\, Hu)^T\big)$. Using this abbreviation we have

$$(q_i|(\epsilon i)^{-1} Hu) = \frac{1}{r_{ii}} \left[(A_i|(\epsilon i)^{-1} Hu) - \sum_{l=1}^{i-1} r_{li} (q_l|(\epsilon i)^{-1} Hu) \right],$$

therefore it suffices to know the values for all $(A_i|(\epsilon i)^{-1} Hu)$. The rest can be obtained recursively again. We get these values by taking the imaginary part of the complex inner product, i.e. Im $\langle A_i|(\epsilon i)^{-1} Hu\rangle$.

3.4 Solving the equation of motion

To conclude we need to solve the equations of motion (1.19)

$$R\dot{y} = \frac{1}{\epsilon}\left(\mathrm{Im}\langle q_i|Hu\rangle\right)_{j=1}^{n}.$$

Defining $Q^*b := \left(\mathrm{Im}\langle q_i|Hu\rangle\right)_{j=1}^{n}$ this is equivalent to

$$R\dot{y} = Q^*b \quad \Leftrightarrow \quad \dot{y} = R^{-1}Q^*b.$$

This is a system of linear ordinary differential equations with initial data $y(0) = y$. As some of the entries are only real numbers and some $d \times d$ matrices we use the explicit Euler method.

Chapter 4

Numerical experiments

In this section, numerical experiments will be presented. We will consider different approximations u, where we change the dimension of the system and the number of Gaussians in the approximation. First recall the definition of the torsional potential

$$V(x) = \sum_{j=1}^{d} (1 - \cos(x_j)), \quad \forall x \in \mathbb{R}^d. \tag{4.1}$$

The torsional potential describes the rotation of an atom or molecule around a symmetry axis. Therefore one expects that after a certain time, the position operator q comes back to its initial position, as the cosine is a periodic function. We will compare our results with the ones in [FGL09], where they compare sparse Hagedorn wave packets with the Fourier method.

Kinetic energy
Consider a normalized Gaussian u as in definition (1.3). Further suppose that the matrices C_j have the structure $C_j = (A_j + iB_j)I$, where I is the identity matrix and A_j, B_j are real numbers with $B_j > 0$. Then holds for the kinetic energy of our approximation u

$$\langle u \mid -\frac{\epsilon^2}{2}\Delta u \rangle = \sum_{j=1}^{m} \left(\frac{|p_j|^2}{2} + \frac{\epsilon d}{2} \frac{(A_j/2)^2 + (B_j/2)^2}{B_j/2} \right). \tag{4.2}$$

This result can be found in [FL06, p.46]. As they do use wave packets without the pre-factor $1/2$ before the matrix C_j, we have to write $A_j/2$ and $B_j/2$ instead of A_j and B_j in (4.2). The formula gives us an exact value for the kinetic energy with given parameters.

4.1 One dimensional experiments

In this section we are considering one dimensional systems. For the approximation u we will analyse single Gaussians and linear combinations of Gaussians.

In both cases we will use separable Gaussians, i.e. the complex number C in each Gaussian might change during the propagation.

4.1.1 Single Gaussian

First, we assume the approximation u to be a single Gaussian, i.e. u can be expressed as

$$u(x) = g(x) = \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q)C(x-q) + p(x-q) + \zeta\right)\right), \quad x \in \mathbb{R}$$

with $q, p \in \mathbb{R}$, $C, \zeta \in \mathbb{C}$ and $\epsilon > 0$. We will set the parameters for the initial condition to be $q(0) = 1$, $p(0) = 0$, $C(0) = i$ and $\zeta(0) = 0$. As C might change during the propagation, we consider a separable Gaussian. We observe that such a wave packet is not normalized yet and this procedure will be executed in the algorithm as a first step by adjusting the phase $\zeta(0)$. We will fix the step-size to $\Delta t = 10^{-2}$ and we propagate the system for 1000 steps in the case of one Gaussian and 700 in the case with linear combinations.

Energy conservation
As stated in theorem 1.2, the total energy of the system remains constant. Using the initial data from above, we show in figure 4.1 the value of the total energy for different $\epsilon > 0$.

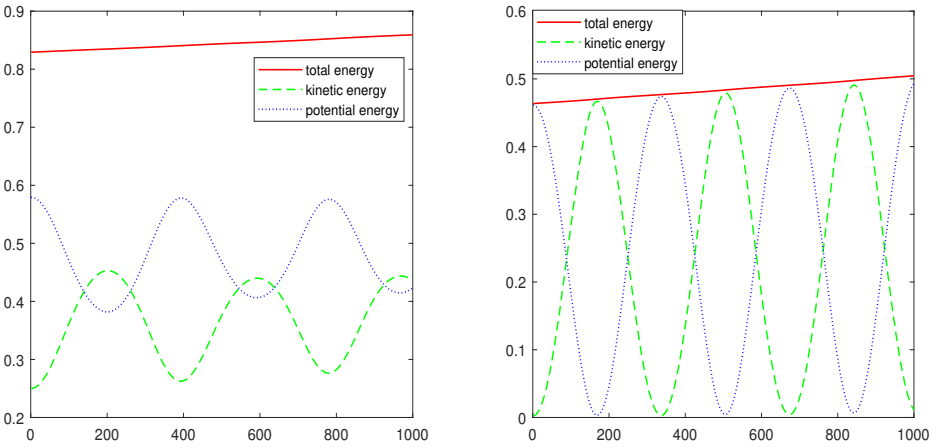

Figure 4.1: Separable Gaussian, left with $\epsilon = 1$ and right with $\epsilon = 0.01$

The total energy is conserved over the time, i.e. stays almost constant. The small up-slope of the total energy is caused by the used time integrator. If one uses a frozen Gaussian instead, the upward slope vanishes almost completely. The periodic behaviour of the kinetic energy and potential energy presented in figure 4.1 are due to the choice of the torsional potential. By formula (4.2) we

would expect the kinetic energy at time $t = 0$ be $1/4$ for the case $\epsilon = 1$ and be $1/400$ for the case $\epsilon = 0.01$. These are exactly the results we have in figure 4.1.

Position and momentum
In the next figures, we present the behaviour of the position q and the momentum p over time. We assume the same initial condition as before.

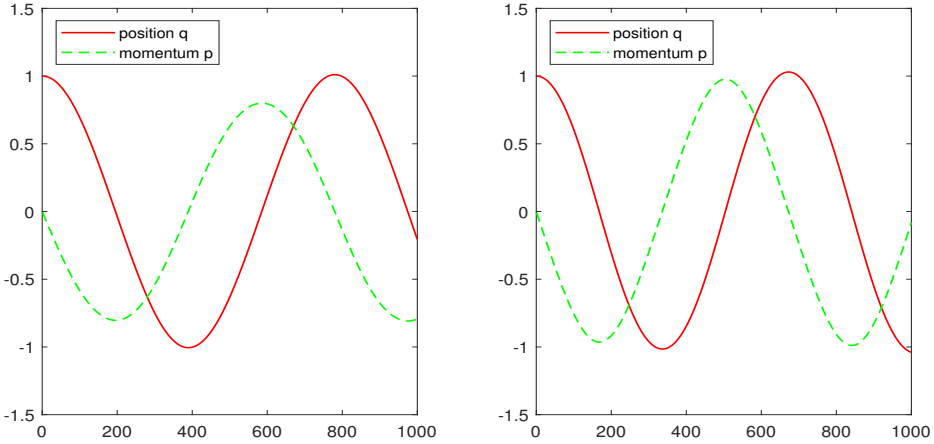

Figure 4.2: Left with $\epsilon = 1$ and right with $\epsilon = 0.01$

We observe the expected behaviour of the position q and the momentum p. When the atom or molecule has rotated fully around an axis, it comes back to the same position where it stated to rotate. This results is due to the periodicity of the torsional potential.
From theorem 3.11 in [LL, p.20] we get that for the exact position and momentum holds the differential equation $\dot{q} = p$. This relation can be observed in figure (4.2) as well.

4.1.2 Linear combination of Gaussians

Now we are assuming linear combinations of one dimensional Gaussians, i.e.

$$u(x) = \sum_{j=1}^{m} \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q_j)C_j(x-q_j) + p_j(x-q_j) + \zeta_j\right)\right), \quad x \in \mathbb{R},$$

with $q_j, p_j \in \mathbb{R}$, $C_j, \zeta_j \in \mathbb{C}$ and $\epsilon > 0$. As initial data we use for g_1 the parameters $q_1 = 0.25$, $p_1 = 0.5$, $C_1 = i$ and $\zeta_1 = 0$. For the second Gaussian g_2 we use $q_2 = 0$, $p_2 = -0.5$, $C_2 = i$ and $\zeta_2 = 0$. Therefore we have $m = 2$. Again this is not a normalized function yet. The normalization in the code will change ζ_1 and ζ_2 such that u has unit norm. As we allow the width matrix C to change in each time step, we are considering a separable Gaussian.

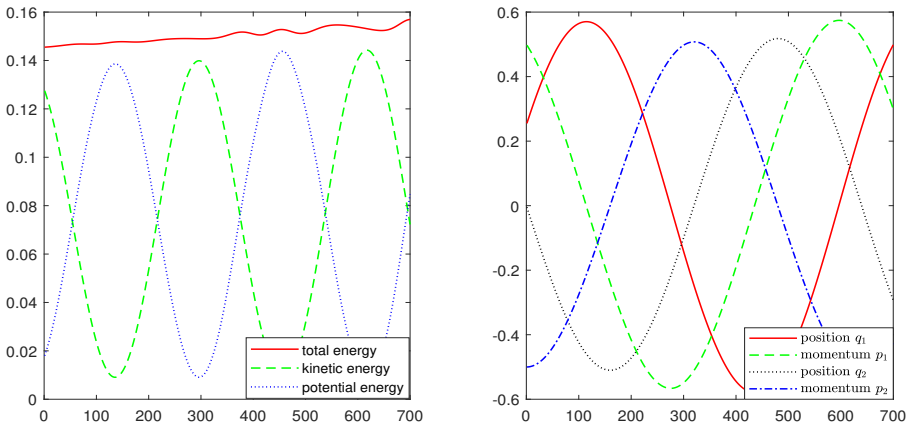

Figure 4.3: $\epsilon = 0.01$

Here we can see that adding an other Gaussian keeps the oscillating structure of the positions and momentums and of the kinetic and potential energy.

Numerical tests with different initial data has shown that the stability of the time integration hardly depend on the initial data, even if we are only in a one dimensional setting. Slight changes can have a huge effect on the result. If one changes C_1 from i to $0.5 + i$, the total energy starts to oscillate with an increasing amplitude over time (after 500 steps an amplitude of 0.1). Using frozen, instead of separable Gaussians, also improves the energy conservation over time.

Further it is interesting to know how many Gaussians one can add to the linear combination. Experiments with the algorithm have shown that up to approximately $m = 8$ one-dimensional, frozen Gaussians, the energy conservation is as stable as for example in figure 4.3. Still this stability depends hardly on the used initial data, hence it must be chosen cautiously.

4.1.3 Convergence in time

For numerical analysis it is crucial to know the convergence of the algorithm in time. We suppose some randomly generated initial data for a frozen Gaussian with width matrix $C = i$, i.e. we have a single Gaussian only. As a reference solution u_{ref} we take the result computed with a step size $\tau = 0.1 \cdot 2^{-11}$. Then we compare in L^2-norm the approximation u with the reference solution u_{ref} at final time $t = 0.5$ and $t = 1$. The exact value for $\|u(t) - u_{\text{ref}}(t)\|_2$ is obtained using the Hagedorn wave packets from chapter 2.

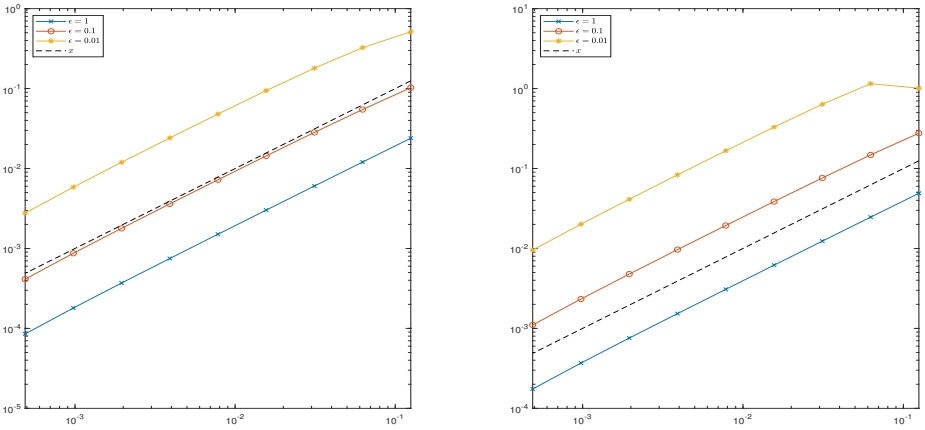

Figure 4.4: Temporal convergence in the L^2-norm at $t = 0.5$ (left) and $t = 1$ (right) for a single, frozen Gaussian

We can see in figure (4.4) that we have a temporal convergence in the L^2-norm of order 1. The order of the convergence does not depend on the choice ϵ.

Time convergence for linear combinations
Suppose now a linear combination of $m = 3$ frozen Gaussians, all with the same width matrix $C_j = i$ and $\zeta_j = 0$. The rest of the initial data is randomly generated. We do the same calculations as in the one dimensional case to see how convergence changes. As a reference solution u_{ref} we take the solution to this initial data with time step size $\tau = 0.1 \cdot 2^{-13}$.

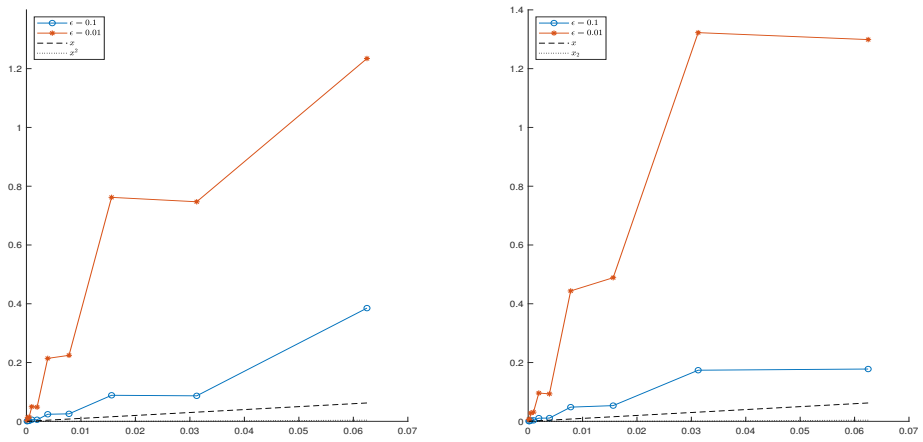

Figure 4.5: Temporal convergence in the L^2-norm at $t = 0.1$ (left) and $t = 0.2$ (right) for a linear combination of $m = 3$ frozen Gaussians

We observe similar results for linear combinations. In this case, the convergence depends on the parameter ϵ. we obtain good results only for relatively small time step sizes τ.

4.2 Higher dimensional experiments

For applications in quantum chemistry one must consider high dimensional quantum systems. Therefore we assume in this section a system of dimension $d > 1$. We will consider different types of Gaussians, as presented in chapter 1. Further we will consider linear combinations of these Gaussians.

4.2.1 Single Gaussian

Suppose for the approximation u the following form

$$u(x) = g(x) = \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q)^T C(x-q) + p^T(x-q) + \zeta\right)\right), \quad x \in \mathbb{R}^d,$$

with $q, p \in \mathbb{R}^d$, $C \in \mathbb{C}^{d\times d}$, $\zeta \in \mathbb{C}$ and $\epsilon > 0$. First we will have a look on how the different types of energies behave in a higher dimensional setting.

3 dimensional system
Consider a 3-dimensional system, i.e. $d = 3$. For the frozen and the separable Gaussian we use the initial data $q = (1,0,0)^T$, $p = (0.3,0.3,0.3)^T$, $C = iI$, where I is the identity matrix, and $\zeta = 0$. For the thawed Gaussian we use the same

q, p and ζ, but choose

$$C = \begin{pmatrix} i & 0.2 & 0.2 \\ 0.2 & i & 0.2 \\ 0.2 & 0.2 & i \end{pmatrix}.$$

Energy conservation
For the different types of Gaussians we get different propagations of the energy.

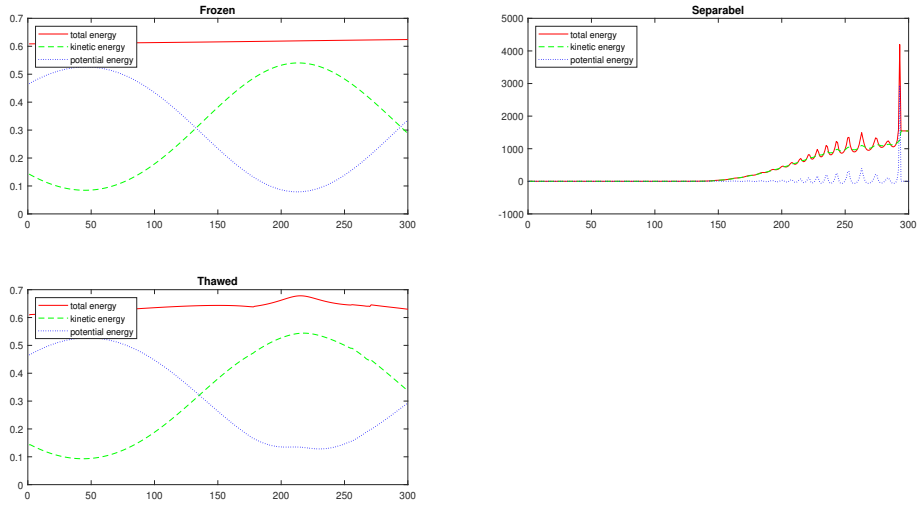

Figure 4.6: Energy propagation

In this figure we can see an interesting behaviour of the propagation. The propagation for a frozen Gaussian behaves as expected in theory. More interesting is that the propagation of the thawed Gaussian behaves much regular than the one of the separable Gaussian, though a thawed Gaussian is more general than a separable one.
The propagation for the separable one is symptomatic for many other cases. The presented algorithm is quite unstable in the separable case and for many thawed cases. The higher the dimension of the system, the less time steps can be done before the total energy increases rapidly. As presented in chapter 1, it is recommendable to use linear combination of frozen Gaussians instead of single thawed or separable ones. Increasing the dimension still gives similar results as for this example, but requires much more CPU time.

Position average
Using the same 3-dimensional, frozen Gaussian as above, we analyze the time propagation of the position average q. In accordance with the choice of the torsional potential, it describes the rotation around an axis. The following propagation shows 1300 steps with step-size $\tau = 10^{-2}$.

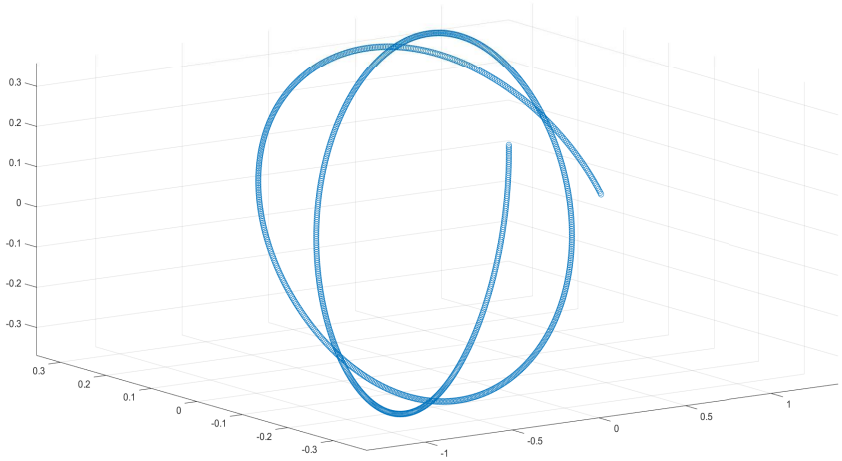

Figure 4.7: Propagation over time of the position average q.

Note that all these points lie in the same two dimensional plane. We can observe the expected rotation around an axis. However we see that the position average does not exactly come back to the same position where the propagation started, what is most likely caused by round-off errors.

4.2.2 Linear combination of Gaussians

Suppose now for the approximation u the following ansatz

$$u(x) = \sum_{j=1}^{m} \exp\left(\frac{i}{\epsilon}\left(\frac{1}{2}(x-q_j)^T C_j (x-q_j) + p_j^T (x-q_j) + \zeta_j\right)\right), \quad x \in \mathbb{R}^d,$$

with $q_j, p_j \in \mathbb{R}^d$, $C_j \in \mathbb{C}^{d\times d}$, $\zeta_j \in \mathbb{C}$ and $\epsilon > 0$. We are now in the most general setting. As initial data we choose $C_j = iI$ and $\zeta_j = 0$ for all j and generate random q_j and p_j for the frozen and separable Gaussians for all j. For the thawed case we choose the same C_j as in the single Gaussian case, for all j. For the dimension we choose $d = 3$. Choosing higher dimension gives similar results for the energy propagation, but the computational time increases very fast. We consider a linear combination of 3 Gaussians, i.e. $m = 3$.

Energy conservation
The next plot shows the energy propagation for the different types of Gaussians defined above.

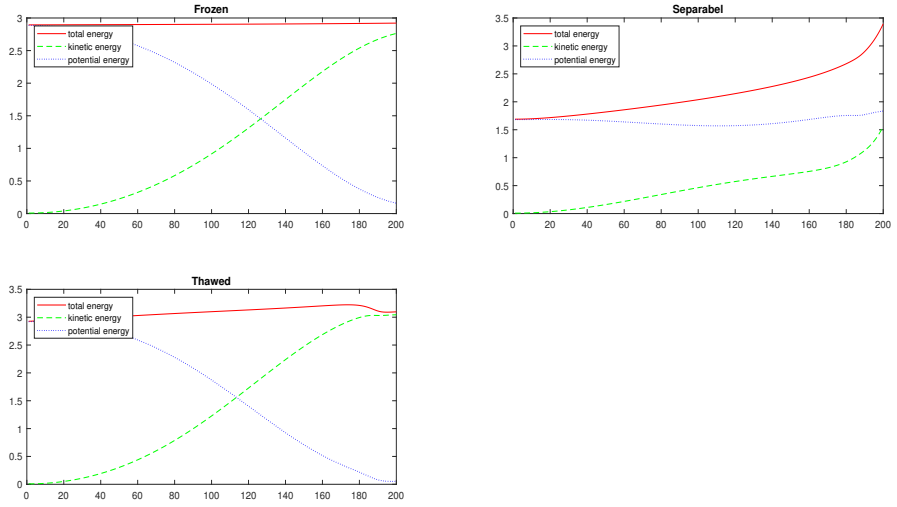

Figure 4.8: Energy propagation, all for $\epsilon = 0.01$.

We observe that frozen Gaussians have a more regular behaviour in time. Separable and thawed Gaussians are only stable for a short times. Other experiments have shown that the frozen Gaussians are stable even for long time intervals. Still the stability of the energy propagation depends hardly on the initial data, especially when we consider linear combinations of high-dimensional Gaussians.

4.2.3 Convergence in time

Again we suppose randomly generated initial data for a single, frozen Gaussian, with width matrix $C = i$ and $\zeta = 0$. For the dimension we choose $d = 3$. As for the one dimensional convergence we consider as reference solution u_{ref} with time step-size $\tau = 0.1 \cdot 2^{-11}$. We consider different time step-sizes and $\epsilon > 0$. For the error in the L^2-norm, i.e. $\|u - u_{\text{ref}}\|_2$ we get

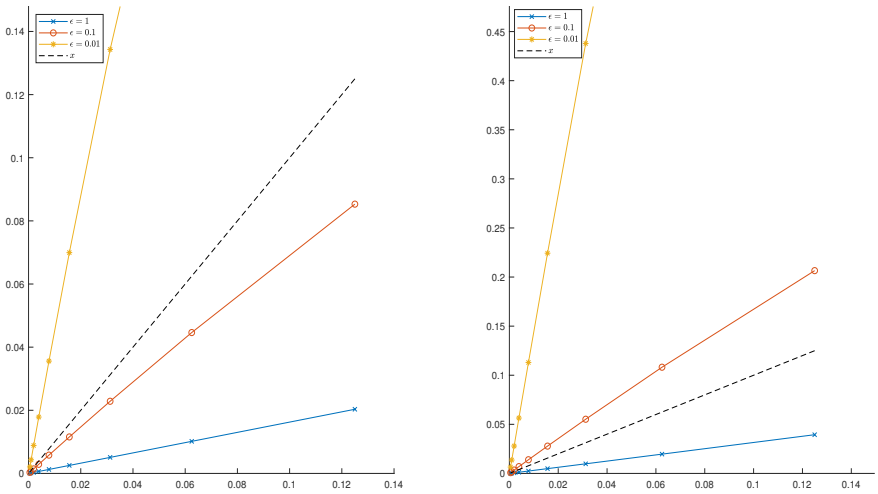

Figure 4.9: Temporal convergence in the L^2-norm at $t = 0.5$ (left) and $t = 1$ (right) for a single, frozen 3-dimensional Gaussian

We see that we have different order of convergence in time for different choices of the semiclassical parameter ϵ. We have to choose a relatively small time step-size τ if we have a small ϵ. This makes calculations costly, but the error decreases fast as well.

4.3 Evaluation of the algorithm

Performance
For practical applications we need to know how the algorithm performs. We want to know how it behaves if one increases the dimension of the system and what happens if one increases the number of Gaussians in the approximation u. For the first case we consider a single Gaussian with the initial data $q = (1, 0, ..., 0)^T$, $p = (0, ..., 0)^T$,$\zeta = 0$ and $C = iI$, where I is the d-dimensional identity matrix.
For the case with increasing number of Gaussians we consider a one dimensional approximation with parameters $C_j = i$, $\zeta_j = 0$ for all j and randomly generated q_j and p_j. The plotted values are the needed time to make 10 time-steps of the algorithm.

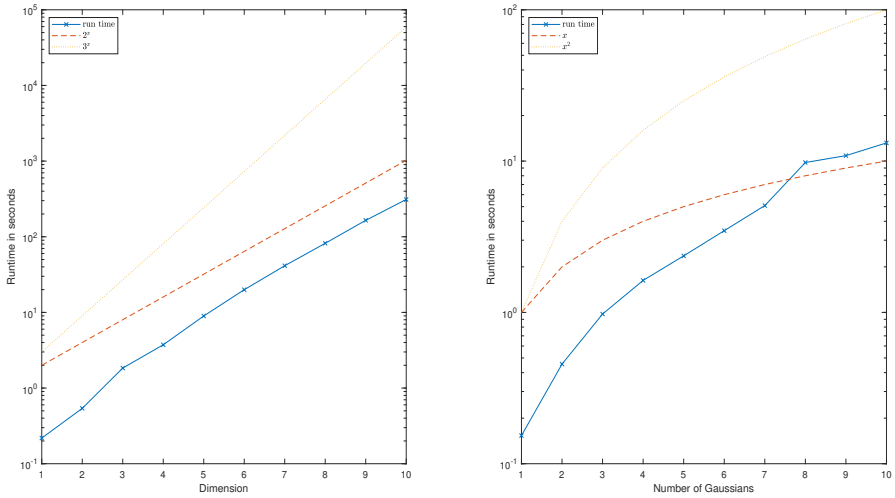

Figure 4.10: CPU time of the code plottet against the dimension (left) and the number of Gaussians (right)

We can see that run time increases exponentially with the dimension ($\approx \mathcal{O}(2^d)$), while the run time increases only quadratically ($\approx \mathcal{O}(m^2)$) with the number of Gaussians in the approximation. Hence increasing the dimension of the system is very costly. Talking in absolute values, a 10-dimensional propagation with 100 steps in time needs 12000 seconds in total, which is approximate 3.5 hours.

4.3.1 Conclusion

As every code, this choice of implementing the vMCG method has its advantages and disadvantages. Like other algorithms, this one is quite slow if one considers a high-dimensional quantum system, see figure (4.10). Further it has an unstable time propagation for long times, if separable or certain thawed Gaussians are used in the approximation, as outlined in section 4.2. Nevertheless it gives us good approximations if one considers frozen Gaussians or linear combination of those. For those, the algorithm preserves the total energy also for long times and the position and momentum average follow expected trajectories, see section 4.1 and 4.2.
Compared to a standard method like MCTDH, the vMCG method does not seem very competitive. However it is not meant to compete, but be a more general and flexible method for computing direct dynamics [RPS+15, p.14].

More complex potentials can be considered easily, if they are a low dimensional polynomial or can be expressed as an exponential function. Then it suffices to implement the inner products as a linear combination of Hagedorn wave packets as described in chapter 2. If not, one has to find an other way to calculate all

the appearing inner products, which in general requires problem-related approximation techniques.

Numerical experiments have shown that using a single thawed Gaussian is not flexible enough to simulate a real quantum system like a molecule. A lot of research has been done to make linear combinations of frozen Gaussians a useful method [RPS+15, p.2]. As the presented implementation of the algorithm is energy preserving and the position and momentum average follow expected trajectories in the frozen case even for long times, it makes it a useful and general method for semiclassical approximations.

In further research it would be interesting to improve the time propagation, to get a more stable algorithm even for separable and thawed Gaussians. Since we need to normalize the Gaussian wave packet after each time step, developing a norm preserving algorithm would be useful as well. This could possibly lead to a stabilization in the time propagation. Furthermore, Hagedorn wave packets might have further applications in different fields than quantum dynamics.

Deutsche Zusammenfassung

Grundlage dieser Arbeit ist die zeitabhängige Schrödingergleichung in einer semiklassischen Skalierung

$$i\epsilon\frac{d\psi}{dt} = H\psi,$$

wobei $\psi = \psi(x,t)$ eine Wellenfunktion, $t \in \mathbb{R}$ die zeitliche und $x \in \mathbb{R}^d$ die räumliche Variable ist. Weiter ist $\epsilon > 0$ ein semiklassischer Parameter, welcher einer skalierten Plank Konstante $\hbar$ entspricht. H ist der Hamiltonoperator des Systems, welcher im Folgenden stets die folgende Form hat

$$H\psi(x,t) = -\frac{\epsilon^2}{2}\Delta\psi(x,t) + V(x)\psi(x,t).$$

Hierbei ist Δ der Laplaceoperator bezüglich der räumlichen Variable x und V ein reellwertiges Potential. Der Faktor ϵ^2 repräsentiert den Massequotienten von Elektronen und Atomkernen in einem Molekül [LL, p.2]. In dieser Arbeit wird ein Algorithmus vorgestellt, welcher für kleine $\epsilon > 0$ eine Lösung der Schrödingergleichung approximiert.
Klassische numerische Verfahren scheitern daran, dass das Problem im Allgemeinen hochdimensional und hoch oszillatorisch ist [LL, p.2].

Ein vielversprechender Ansatz verwendet dazu Gaußsche Wellenpakete. Diese Klasse von Verfahren nennt sich variational multi-configurational Gaussians (vMCG) und findet Anwendung in der Quantenchemie. Die Idee geht zurück auf Heller [Hel75], welcher mehrdimensionale Gaußfunktionen betrachtete und Bewegungsgleichungen für deren Parameter herleitete. Diese Grundidee wurde im Laufe der Zeit auch auf Linearkombinationen von solchen Gaußfunktionen angewendet, siehe hierfür zum Beispiel [RPS+15].

Diese Arbeit beschreibt einen neuen Ansatz wie man das vMCG Verfahren implementieren kann. Dafür wird im Kapitel 1 das Problem abstrakt formuliert, grundlegende Notationen werden eingeführt und Bewegungsgleichungen für die Parameter der Gaußfunktionen werden hergeleitet. Die abstrakte Formulierung des Problems beruht in großen Teilen auf [Lub08].
Kapitel 2 führt die Theorie der Hagedornschen Wellenpaketen ein, welche von

George Hagedorn in [Hag81], [Hag85] und [Hag98] entwickelt wurde. Diese speziellen Funktionen erlauben es uns, hochdimensionale Integrale von Gaußfunktionen multipliziert mit einem Polynom niedrigem Grades exakt auszurechnen. Diese Eigenschaft ist essenziell für den Algorithmus, da wir ein modifiziertes Gram-Schmidt Verfahren auf L^2-Funktionen anwenden wollen und deren Skalarprodukte berechnen müssen.

In Kapitel 3 wird erklärt, wie der Algorithmus im Detail implementiert wurde. Im letzten Kapitel 4 werden einige numerische Experimente mit dem Algorithmus vorgestellt. Dieser wird auf ein- und mehrdimensionale Quantensysteme angewendet und das Verhalten verschiedener Typen von Gaußfunktionen wird analysiert.
Es zeigt sich, dass der vorgestellte Algorithmus gute Ergebnisse liefet, wenn man frozen Gaußfunktionen als Approximation betrachtet. Hierfür wird die Gesamtenergie des Systems auch für lange Zeiten konstant gehalten und auch die Postions- und Momentumsparameter q und p folgen erwarteten Trajektorien. Für separable und thawed Gaußfunktionen ist der verwendete Zeitintegrator oft instabil und liefert nur für kurze Zeiten gute Ergebnisse. Verwendet man stattdessen Linearkombinationen von frozen Gaußfunktionen, so bleiben die oben beschriebenen Eigenschaften erhalten und liefern eine brauchbare Approximation an die Wellenfunktion des Systems.

Bibliography

[DH08] Peter Deuflhard and Andreas Hohmann. *Eine algorithmisch orientierte Einführung: An algorithm-based introduction*, volume / Peter Deuflhard ... ; 1 of *De-Gruyter-Lehrbuch*. de Gruyter, Berlin, 4., überarb. und erw. aufl., [elektronische ressource] edition, 2008.

[FGL09] Erwan Faou, Vasile Gradinaru, and Christian Lubich. Computing semiclassical quantum dynamics with hagedorn wavepackets. *SIAM Journal on Scientific Computing*, 31(4):3027–3041, 2009.

[FL06] Erwan Faou and Christian Lubich. A poisson integrator for gaussian wavepacket dynamics. *Computing and Visualization in Science*, 9(2):45–55, 2006.

[Hag81] George A. Hagedorn. Semiclassical quantum mechanics. iii. the large order asymptotics and more general states. *Annals of Physics*, 135(1):58–70, 1981.

[Hag85] George A. Hagedorn. Semiclassical quantum mechanics, iv : large order asymptotics and more general states in more than one dimension. *Annales de l'I.H.P. Physique théorique*, pages 363–374, 1985.

[Hag98] George A. Hagedorn. Raising and lowering operators for semiclassical wave packets. *Annals of Physics*, 269(1):77–104, 1998.

[Hel75] Eric J. Heller. Time–dependent approach to semiclassical dynamics. *The Journal of Chemical Physics*, 62(4):1544–1555, 1975.

[Hel81] Eric J. Heller. Frozen gaussians: A very simple semiclassical approximation. *The Journal of Chemical Physics*, 75(6):2923–2931, 1981.

[Kan05] Christian Kanzow. *Numerik linearer Gleichungssysteme: Direkte und iterative Verfahren*. Springer-Lehrbuch. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2005.

[Kle06] Achim Klenke. *Wahrscheinlichkeitstheorie*. Springer-Lehrbuch Masterclass. Springer, Berlin, 2006.

[LL] Caroline Lasser and Christian Lubich. Computing quantum dynamics in the semiclassical regime.

[Lub08] Christian Lubich. *From quantum to classical molecular dynamics: Reduced models and numerical analysis.* Zurich lectures in advanced mathematics. European Mathematical Society, Zuerich, 2008.

[RPS+15] G. W. Richings, I. Polyak, K. E. Spinlove, G. A. Worth, I. Burghardt, and B. Lasorne. Quantum dynamics simulations using gaussian wavepackets: The vmcg method. *International Reviews in Physical Chemistry*, 34(2):269–308, 2015.

[Tro17] Stephanie Troppmann. *Non-Hermitian Schrödinger dynamics with Hagedorn's wave packets.* Dissertation, Universitätsbibliothek der TU München, München, 2017.

Eidesstaatliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Literatur angefertigt habe. Sämtliche Stellen, welche dem Wortlaut oder dem Sinn nach anderen Quellen entnommen sind, wurden als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt.

.....	.....
Ort, Datum	Unterschrift

Appendix A

Calculations and formulas

A.1 Hagedorn polynomials

Let $e_i = (0, ..., 0, 1, 0, .., 0)$, where e_i is 1 in the i-th component. Recall the definition of the Hagedorn polynomials

$$p_0 = 1, \quad \left(p_{k+e_j}\right)_{j=1}^{d} = \hat{B}^\dagger p_k \quad \text{where} \quad \hat{B}^\dagger = 2x - M\nabla_x.$$

An explicit calculation gives the following polynomials up to order 4.

k	$p_k(x)$
0	1
e_i	$2x_i$
$2e_i$	$4x_i^2 - 2M_{ii}$
$3e_i$	$8x_i^3 - 12M_{ii}x_i$
$4e_i$	$16x_i^4 - 48M_{ii}x_i^2 + 12M_{ii}^2$
$e_i + e_j$	$4x_ix_j - 2M_{ji}$
$2e_i + e_j$	$8x_i^2x_j - (4M_{ji} + 4M_{ij})x_i - 4M_{ii}x_j$
$2e_i + 2e_j$	$16x_i^2x_j^2 - 8M_{ii}x_j^2 - 8M_{jj}x_i^2 - (8M_{ij} + 24M_{ji})x_ix_j$ $+4M_{ji}^2 + 4M_{ij}M_{ii} + 4M_{ii}M_{jj}$
$e_i + e_j + e_l$	$8x_ix_jx_l - 4M_{ji}x_l - 4M_{li}x_j - 4M_{lj}x_i$
$2e_i + e_j + e_l$	$16x_i^2x_jx_l - (8M_{ji} + 8M_{ij})x_ix_l - 8M_{ii}x_jx_l - 16M_{li}x_ix_j$ $-8M_{lj}x_i^2 + 4M_{li}M_{ji} + 4M_{li}M_{ij} + 4M_{lj}M_{ii}$
$3e_i + e_j$	$16x_i^3x_j - (8M_{ji} + 16M_{ij})x_i^2 - 24M_{ii}x_ix_j + 4M_{ii}M_{ji} + 8M_{ii}M_{ij}$

Table A.1: Poylnominals $p_k(x)$

A.2 Gaussain times polynomial

Here one can find explicit formulas for integrals of the type Gaussian times a polynomial of degree ≤ 4. These formulas suffice to calculate all appearing complex inner products in the algorithm. In the following let g be a Gaussian as in definition (1.3). Recall the decomposition of the matrix C in $C = PQ^{-1}$. With that the Gaussian g reads as

$$g(x) = \exp\left(\frac{i}{2\epsilon}(x-q)^T PQ^{-1}(x-q) + \frac{i}{\epsilon}p^T(x-q) + \frac{i}{\epsilon}\zeta\right).$$

Further let $k \in \mathbb{N}^d$ be a multi index, φ_k the corresponding Hagedorn function as defined in (2.2) and $\mathcal{F}^\epsilon\varphi_k$ its Fourier transform. Further let e_i be the i-th unit vector and define the following constant

$$T := \exp^{\frac{i\zeta}{\epsilon}}(\pi\epsilon)^{d/4}\det(Q)^{1/2}(2\pi\epsilon)^{d/2}.$$

Order 0 and 1:
For polynomials of order 0 and 1 we have

$$\begin{aligned}\int_{\mathbb{R}^d} g(x)dx &= T\mathcal{F}^\epsilon\varphi_0(0)\\ \int_{\mathbb{R}^d} x_i g(x)dx &= T\sqrt{\frac{\epsilon}{2}}\left(\sum_{l=1}^d Q_{il}\mathcal{F}^\epsilon\varphi_{e_i}(0) + \sqrt{\frac{2}{\epsilon}}q_i\mathcal{F}^\epsilon\varphi_0(0)\right).\end{aligned}$$

For higher order of polynomials there are no explicit formulas, but one need solve an linear equation system. If for some multi index k holds $k - e_l \notin \mathbb{N}^d$, we set $\mathcal{F}^\epsilon\varphi_{k-e_l}(0) = 0$. Further define $k! := k_1!\cdots k_d!$.

Order 2:
For polynomials of order 2 we define

$$\begin{aligned}A &:= Q^{-1}\\ b_j &:= \frac{\epsilon T}{2}\left(\sum_{l=1}^d Q_{il}\sqrt{e_l+1}\mathcal{F}^\epsilon\varphi_{e_j+e_l}(0) + \sum_{l=1}^d \overline{Q_{il}}\sqrt{e_l}\mathcal{F}^\epsilon\varphi_{e_j-e_l}(0)\right.\\ &\quad \left. + \frac{2}{\epsilon}T^{-1}(2\pi\epsilon)^{d/2}\sum_{l=1}^d Q^{-1}_{jl}q_l\int_{\mathbb{R}^d} x_i g(x)dx + \sqrt{\frac{2}{\epsilon}}q_i\mathcal{F}^\epsilon\varphi_{e_j}(0)\right).\end{aligned}$$

Let now z be the solution of the linear equation system $Az = b$. Then it holds

$$z_l = \int_{\mathbb{R}^d} x_i x_l g(x)dx.$$

Order 3:
For polynomials of order 3 we define $k_{ij} = e_i + e_j$. Further set $M = Q^{-1}\overline{Q}$. Then

$$A_{j+(i-1)d,n+(m-1)d} := \sqrt{\frac{2}{\epsilon}}\frac{4}{\epsilon}\frac{1}{\sqrt{2^{|k_{ij}|}k_{ij}!}}Q^{-1}_{in}Q^{-1}_{jm}$$

$$b_{j+(i-1)d} := T\Bigg(\sum_{n=1}^{d} Q_{ln}\sqrt{k_{ij}+1}\mathcal{F}^{\epsilon}\varphi_{k_{ij}+e_n}(0) + \sum_{n=1}^{d}\overline{Q_{ln}}\sqrt{k_{ij}}\mathcal{F}^{\epsilon}\varphi_{k_{ij}-e_n}(0)$$
$$+ q_l\sqrt{\frac{2}{\epsilon}}\mathcal{F}^{\epsilon}\varphi_{k_{ij}}(0) + \sqrt{\frac{2}{\epsilon}}\frac{(2\pi\epsilon)^{d/2}}{\sqrt{2^{|k_{ij}|}k_{ij}!}}\frac{4T^{-1}}{\epsilon}\Bigg[\sum_{n,m=1}^{d} Q^{-1}_{in}Q^{-1}_{jm}q_n\int_{\mathbb{R}^d} x_l x_m g(x)dx$$
$$+ \sum_{n,m=1}^{d} Q^{-1}_{in}Q^{-1}_{jm}q_m\int_{\mathbb{R}^d} x_l x_n g(x)dx - \left(\sum_{n,m=1}^{d} Q^{-1}_{in}Q^{-1}_{jm}q_n q_m + \frac{\epsilon}{2}M_{ji}\right)\int_{\mathbb{R}^d} x_l g(x)dx\Bigg]\Bigg).$$

Let now z be the solution of the linear equation system $Az = b$. Then it holds

$$z_{j+(i-1)d} = \int_{\mathbb{R}^d} x_i x_j x_l g(x)dx.$$

Order 4:
For polynomials of order 4 we define $k_{ijl} = e_i + e_j + e_l$. Further set $M = Q^{-1}\overline{Q}$ and $C = Q^{-1}_{in}Q^{-1}_{jm}Q^{-1}_{lk}$ for arbitrary i, j, l, n, m, k. Then

$$A_{l+(j-1)d+(i-1)d^2,n+(m-1)d+(k-1)d^2} := \frac{1}{\sqrt{2^{|k_{ijl}|}k_{ijl}!}}\sqrt{\frac{2}{\epsilon}}\frac{8}{\epsilon^{3/2}}C$$

$$b_{l+(j-1)d+(i-1)d^2} := T\Bigg(\sum_{n=1}^{d} Q_{\mu n}\sqrt{k_{ijl}+1}\mathcal{F}^{\epsilon}\varphi_{k_{ijl}+e_n}(0) + q_\mu\sqrt{\frac{2}{\epsilon}}\mathcal{F}^{\epsilon}\varphi_{k_{ijl}}(0)$$
$$+ \sum_{n=1}^{d}\overline{Q_{\mu n}}\sqrt{k_{ijl}}\mathcal{F}^{\epsilon}\varphi_{k_{ijl}-e_n}(0) + \frac{(2\pi\epsilon)^{d/2}}{\sqrt{2^{|k_{ijl}|}k_{ijl}!}}\sqrt{\frac{2}{\epsilon}}\frac{8T^{-1}}{\epsilon^{3/2}}\Bigg[\sum_{n,m,k=1}^{d} Cq_k\int_{\mathbb{R}^d} x_n x_m x_\mu g(x)dx$$
$$+ \sum_{n,m,k=1}^{d} Cq_m\int_{\mathbb{R}^d} x_n x_k x_\mu g(x)dx + \sum_{n,m,k=1}^{d} Cq_n\int_{\mathbb{R}^d} x_k x_m x_\mu g(x)dx$$
$$- \sum_{n,m,k=1}^{d} Cq_m q_k\int_{\mathbb{R}^d} x_n x_\mu g(x)dx - \sum_{n,m,k=1}^{d} Cq_m q_n\int_{\mathbb{R}^d} x_k x_\mu g(x)dx$$
$$- \sum_{n,m,k=1}^{d} Cq_n q_k\int_{\mathbb{R}^d} x_m x_\mu g(x)dx + \sum_{n,m,k=1}^{d} Cq_n q_m q_k\int_{\mathbb{R}^d} x_\mu g(x)dx\Bigg]$$
$$+ \frac{1}{\sqrt{2^{|k_{ijl}|}k_{ijl}!}}\sqrt{\frac{2}{\epsilon}}\frac{4T^{-1}}{\epsilon^{3/2}}\Bigg[M_{ji}\sum_{n=1}^{d} Q^{-1}_{ln}\int_{\mathbb{R}^d} x_n x_\mu g(x)dx + M_{li}\sum_{n=1}^{d} Q^{-1}_{jn}\int_{\mathbb{R}^d} x_n x_\mu g(x)dx$$
$$+ M_{lj}\sum_{n=1}^{d} Q^{-1}_{in}\int_{\mathbb{R}^d} x_n x_\mu g(x)dx - \sum_{n=1}^{d}(Q^{-1}_{ln} + Q^{-1}_{jn} + Q^{-1}_{in})q_n\int_{\mathbb{R}^d} x_\mu g(x)dx\Bigg]\Bigg).$$

Let now z be the solution of the linear equation system $Az = b$. Then it holds

$$z_{l+(j-1)d+(i-1)d^2} = \int_{\mathbb{R}^d} x_i x_j x_l x_\mu g(x) dx.$$

A.3 Appearing inner products

A.3.1 Inner products of parameter-derivatives

Here one can find a complete list of all the appearing complex inner products of the orthogonaliation. In the following $(q_j)_l, (p_j)_l$ means the l-th component of q_j and p_j. The function $g(x)$ is the Gaussian which one get in the same way as described in (1.30) when multiplying $g_i(x)$ and $g_j(x)$. All can be obtained by a straight forward calculation.

$$\langle \frac{\partial u}{\partial \zeta_i}, \frac{\partial u}{\partial (q_j)_l} \rangle = \frac{1}{\epsilon^2} \left(\sum_{n=1}^{d} -C_{ln}^j \int_{\mathbb{R}^d} x_n g(x) dx + (q_j^T C_j - p_j)_l \int_{\mathbb{R}^d} g(x) dx \right)$$

$$\begin{aligned} \langle \frac{\partial u}{\partial (p_j)_l}, \frac{\partial u}{\partial (q_i)_m} \rangle = \frac{1}{\epsilon^2} \Bigg(& \sum_{n=1}^{d} -C_{mn}^i \int_{\mathbb{R}^d} x_n x_m g(x) dx + (q_j)_l \sum_{n=1}^{d} C_{mn}^i \int_{\mathbb{R}^d} x_n g(x) dx \\ & + (C_i q_i - p_i)_m \int_{\mathbb{R}^d} x_l g(x) dx - (C_i q_i - p_i)_m (q_j)_l \int_{\mathbb{R}^d} g(x) dx \Bigg) \end{aligned}$$

$$\begin{aligned} \langle \frac{\partial u}{\partial (q_i)_l}, \frac{\partial u}{\partial (q_j)_m} \rangle = \frac{1}{\epsilon^2} \Bigg(& \sum_{n=1}^{d} \sum_{k=1}^{d} \overline{C_{ln}^i} C_{mk}^j \int_{\mathbb{R}^d} x_n x_k g(x) dx \\ & - (C_j q_j - p_j)_m \sum_{n=1}^{d} \overline{C_{ln}^i} \int_{\mathbb{R}^d} x_n g(x) dx \\ & - \overline{(C - i q_i - p_i)_l} \sum_{n=1}^{d} C_{mn}^j \int_{\mathbb{R}^d} x_n g(x) dx \\ & + \overline{(C - i q_i - p_i)_l} (C_j q_j - p_j)_m \int_{\mathbb{R}^d} g(x) dx \Bigg) \end{aligned}$$

$$\begin{aligned} \langle \frac{\partial u}{\partial C_j}, \frac{\partial u}{\partial (q_i)_m} \rangle = \frac{1}{\epsilon^2} \Bigg(& \sum_{l=1}^{d} \sum_{r=1}^{d} \frac{-1}{2} C_{mr}^i \int_{\mathbb{R}^d} x_l^2 x_r g(x) dx - \frac{1}{2} q_j^T q_j \sum_{l=1}^{d} C_{ml}^i \int_{\mathbb{R}^d} x_l g(x) dx \\ & + \sum_{l=1}^{d} \sum_{r=1}^{d} (q_j)_l C_{mr}^i \int_{\mathbb{R}^d} x_r x_l g(x) dx + \frac{1}{2} \sum_{l=1}^{d} (C_i q_i - p_i)_m \int_{\mathbb{R}^d} x_l^2 g(x) dx \\ & + \frac{1}{2} q_j^T q_j (C_i q_i - p_i)_m \int_{\mathbb{R}^d} g(x) dx - \sum_{l=1}^{d} (q_j)_l (C_i q_i - p_i)_m \int_{\mathbb{R}^d} x_l g(x) dx \Bigg) \end{aligned}$$

$$\langle \frac{\partial u}{\partial \zeta_j}, \frac{\partial u}{\partial (p_i)_l} \rangle = \frac{1}{\epsilon^2} \left(\int_{\mathbb{R}^d} x_l g(x) dx - (q_i)_l \int_{\mathbb{R}^d} g(x) dx \right)$$

$$\begin{aligned} \langle \frac{\partial u}{\partial (p_j)_l}, \frac{\partial u}{\partial (p_i)_m} \rangle = \frac{1}{\epsilon^2} \Bigg(& \int_{\mathbb{R}^d} x_l x_m g(x) dx - (q_i)_m \int_{\mathbb{R}^d} x_l g(x) dx - (q_j)_l \int_{\mathbb{R}^d} x_m g(x) dx \\ & + (q_i)_m (q_j)_l \int_{\mathbb{R}^d} g(x) dx \Bigg) \end{aligned}$$

$$\langle \frac{\partial u}{\partial C_j}, \frac{\partial u}{\partial (p_i)_m} \rangle = \frac{1}{\epsilon^2}\Bigg(\frac{1}{2}\sum_{l=1}^{d} \int_{\mathbb{R}^d} x_l^2 x_m g(x)dx - \frac{1}{2}(q_i)_m \sum_{l=1}^{d} \int_{\mathbb{R}^d} x_l^2 g(x)dx$$
$$+ \frac{1}{2} q_j^T q_j \int_{\mathbb{R}^d} x_m g(x)dx - \sum_{l=1}^{d} (q_j)_l \int_{\mathbb{R}^d} x_l x_m g(x)dx$$
$$+ (q_i)_m \sum_{l=1}^{d} (q_j)_l \int_{\mathbb{R}^d} x_l g(x)dx - \frac{1}{2} q_j^T q_j (q_i)_m \int_{\mathbb{R}^d} g(x)dx \Bigg)$$

$$\langle \frac{\partial u}{\partial \zeta_j}, \frac{\partial u}{\partial \zeta_i} \rangle = \frac{1}{\epsilon^2} \int_{\mathbb{R}^d} g(x)dx$$

$$\langle \frac{\partial u}{\partial C_j}, \frac{\partial u}{\partial \zeta_i} \rangle = \frac{1}{\epsilon^2}\Bigg(\frac{1}{2}\sum_{l=1}^{d} \int_{\mathbb{R}^d} x_l^2 g(x)dx - \sum_{l=1}^{d} (q_j)_l \int_{\mathbb{R}^d} x_l g(x)dx$$
$$+ \frac{1}{2} q_j^T q_j \int_{\mathbb{R}^d} g(x)dx \Bigg)$$

$$\langle \frac{\partial u}{\partial C_j}, \frac{\partial u}{\partial C_i} \rangle = \frac{1}{\epsilon^2}\Bigg(\frac{1}{4}\sum_{l=1}^{d}\sum_{r=1}^{d} \int_{\mathbb{R}^d} x_l^2 x_r^2 g(x)dx + \frac{1}{4} q_i^T q_i \sum_{l=1}^{d} \int_{\mathbb{R}^d} x_l^2 g(x)dx$$
$$- \frac{1}{2}\sum_{l=1}^{d}\sum_{r=1}^{d} (q_i)_l \int_{\mathbb{R}^d} x_l x_r^2 g(x)dx + \frac{1}{4} q_j^T q_j \sum_{l=1}^{d} \int_{\mathbb{R}^d} x_l^2 g(x)dx$$
$$+ \frac{1}{4} q_j^T q_j q_i^T q_i \int_{\mathbb{R}^d} g(x)dx - \frac{1}{2} q_j^T q_j \sum_{l=1}^{d} (q_i)_l \int_{\mathbb{R}^d} x_l g(x)dx$$
$$- \frac{1}{2}\sum_{l=1}^{d}\sum_{r=1}^{d} (q_j)_l \int_{\mathbb{R}^d} x_l x_r^2 g(x)dx - \frac{1}{2} q_i^T q_i \sum_{l=1}^{d} (q_j)_l \int_{\mathbb{R}^d} x_l g(x)dx$$
$$+ \sum_{l=1}^{d}\sum_{r=1}^{d} (q_j)_l (q_i)_r \int_{\mathbb{R}^d} x_l x_r g(x)dx \Bigg).$$

A.3.2 Inner products of parameter-derivatives and Hamiltonian

Here one can find a complete list of all appearing complex inner products of the form $\langle \partial u/\partial y, Hu\rangle$, where y is a parameter of a Gaussian. $H = -\frac{\epsilon^2}{2}\Delta + V$, with $V(x) = \sum_{l=1}^{d}(1-\cos(x_l))$. Further m is the number of Gaussians in the approximation u, i.e. $u(x) = \sum_{k=1}^{m} \tilde{g_k}(x)$. With g_k we mean the multiplication of g_j, the Gaussian which corresponds to the parameter y, and $\tilde{g_k}$. Further we define the abbreviation

$$T_k := \left(\frac{1}{2}q_k^T C_k^2 q_k - p_k^T C_k q_k + \frac{1}{2}|p_k|^2 - \frac{i\epsilon}{2}tr(C_k)\right).$$

Laplacian:
First we calculate all inner product of the form $\langle \partial u/\partial y, -\frac{\epsilon^2}{2}\Delta u\rangle$. With the upper abbreviation we have

$$\begin{aligned}
\langle \frac{\partial u}{\partial (q_j)_\mu}, -\frac{\epsilon^2}{2}\Delta u\rangle = -\frac{i}{\epsilon}\sum_{k=1}^{m}\Bigg(&\overline{(C_j q_j - p_j)}_\mu T_k \int_{\mathbb{R}^d} g_k(x)dx - \sum_{r=1}^{d}\overline{C_j}_{\mu r} T_k \int_{\mathbb{R}^d} x_r g_k(x)dx \\
&+ \sum_{r=1}^{d}\overline{(C_j q_j - p_j)}_\mu (-q_k^T C_k^2 + p_k^T C_k)_r \int_{\mathbb{R}^d} x_r g_k(x)dx \\
&- \sum_{r,s=1}^{d}\overline{C_j}_{\mu r}(-q_k^T C_k^2 + p_k^T C_k)_s \int_{\mathbb{R}^d} x_r x_s g_k(x)dx \\
&+ \sum_{r,s=1}^{d}\frac{1}{2}\overline{(C_j q_j - p_j)}_\mu (C_k^2)_{rs}\int_{\mathbb{R}^d} x_r x_s g_k(x)dx \\
&- \sum_{l,r,s=1}^{d}\frac{1}{2}\overline{C_j}_{\mu l}(C_k^2)_{rs}\int_{\mathbb{R}^d} x_r x_s x_l g_k(x)dx\Bigg) \\
\langle \frac{\partial u}{\partial \zeta_j}, -\frac{\epsilon^2}{2}\Delta u\rangle = -\frac{i}{\epsilon}\sum_{k=1}^{m}\Bigg(&\sum_{r,s=1}^{d}\frac{1}{2}(C_k^2)_{rs}\int_{\mathbb{R}^d} x_r x_s g_k(x)dx \\
&+ \sum_{r=1}^{d}(-q_k^T C_k^2 + p_k^T C_k)_r \int_{\mathbb{R}^d} x_r g_k(x)dx + T_k\int_{\mathbb{R}^d} g_k(x)dx\Bigg) \\
\langle \frac{\partial u}{\partial (p_j)_\mu}, -\frac{\epsilon^2}{2}\Delta u\rangle = -\frac{i}{\epsilon}\sum_{k=1}^{m}\Bigg(&\sum_{r,s=1}^{d}\frac{1}{2}(C_k^2)_{rs}\int_{\mathbb{R}^d} x_r x_s x_\mu g_k(x)dx \\
&+ \sum_{r=1}^{d}(-q_k^T C_k^2 + p_k^T C_k)_r \int_{\mathbb{R}^d} x_r x_\mu g_k(x)dx + T_k\int_{\mathbb{R}^d} x_\mu g_k(x)dx \\
&- (q_j)_\mu \sum_{r=1}^{d}(-q_k^T C_k^2 + p_k^T C_k)_r \int_{\mathbb{R}^d} x_r g_k(x)dx \\
&- (q_j)_\mu \sum_{r,s=1}^{d}\frac{1}{2}(C_k^2)_{rs}\int_{\mathbb{R}^d} x_r x_s g_k(x)dx - (q_j)_\mu T_k\int_{\mathbb{R}^d} g_k(x)dx\Bigg)
\end{aligned}$$

$$\begin{aligned}\langle\frac{\partial u}{\partial C_j},-\frac{\epsilon^2}{2}\Delta u\rangle = -\frac{i}{\epsilon}\sum_{k=1}^{m}\Bigg(&\frac{1}{2}q_j^Tq_jT_k\int_{\mathbb{R}^d}g_k(x)dx - T_k\sum_{r=1}^{d}(q_j)_r\int_{\mathbb{R}^d}x_rg_k(x)dx\\ &+\frac{1}{2}q_j^Tq_j\sum_{r=1}^{d}(-q_k^TC_k^2+p_kC_k)_r\int_{\mathbb{R}^d}x_rg_k(x)dx\\ &+\frac{T_k}{2}\sum_{r=1}^{d}\int_{\mathbb{R}^d}x_r^2g_k(x)dx+\frac{1}{4}q_j^Tq_j\sum_{r,s=1}^{d}(C_k^2)_{rs}\int_{\mathbb{R}^d}x_rx_sg_k(x)dx\\ &-\sum_{r,s=1}^{d}(q_j)_r(-q_k^TC_k^2+p_kC_k)_s\int_{\mathbb{R}^d}x_rx_sg_k(x)dx\\ &+\frac{1}{2}\sum_{r,s=1}^{d}(-q_k^TC_k^2+p_kC_k)_s\int_{\mathbb{R}^d}x_r^2x_sg_k(x)dx\\ &-\frac{1}{2}\sum_{r,s,l=1}^{d}(q_j)_l(C_k^2)_{rs}\int_{\mathbb{R}^d}x_rx_sx_lg_k(x)dx\\ &+\frac{1}{4}\sum_{r,s,l=1}^{d}(C_k^2)_{rs}\int_{\mathbb{R}^d}x_rx_sx_l^2g_k(x)dx\Bigg).\end{aligned}$$

Potential:
Now we calculate all inner products of the form $\langle A_\mu, Vu\rangle = \langle \partial u/\partial y, Vu\rangle$. As we have seen in chapter 1, it holds

$$\langle A_\mu, Vu\rangle = \sum_{j,l=1}^{d}\langle A_\mu, g_j - \frac{1}{2}g_{jl}^+(x) - \frac{1}{2}g_{jl}^-(x)\rangle,$$

if we use the same notation and definitions as in chapter 1. Therefore it suffices to know how to calculate integrals of the form $\int_{\mathbb{R}^d}\overline{A_\mu(x)}\tilde{g}(x)dx$, where $\tilde{g}$ is an arbitrary Gaussian. Define g_{jl} as the multiplication of $\overline{g_j}$ with $\tilde{g}_l$. In the following we have explicit formulas for these kind of integrals.

$$\begin{aligned}\langle\frac{\partial u}{\partial \zeta_j},Vu\rangle &= -\frac{i}{\epsilon}\sum_{l=1}^{m}\int_{\mathbb{R}^d}g_{jl}(x)dx\\ \langle\frac{\partial u}{\partial (p_j)_\mu},Vu\rangle &= -\frac{i}{\epsilon}\sum_{l=1}^{m}\left(\int_{\mathbb{R}^d}x_\mu g_{jl}(x)dx - (q_j)_\mu\int_{\mathbb{R}^d}g_{jl}(x)dx\right)\\ \langle\frac{\partial u}{\partial (q_j)_\mu},Vu\rangle &= -\frac{i}{\epsilon}\sum_{l=1}^{m}\left(\overline{(C_jq_j-p_j)}_\mu\int_{\mathbb{R}^d}g_{jl}(x)dx - \sum_{r=1}^{d}\overline{C_{\mu r}^j}\int_{\mathbb{R}^d}x_rg_{jl}(x)dx\right)\\ \langle\frac{\partial u}{\partial C_j},Vu\rangle &= -\frac{i}{\epsilon}\sum_{l=1}^{m}\Bigg(\frac{1}{2}\sum_{r=1}^{d}\int_{\mathbb{R}^d}x_r^2g_{jl}(x)dx - \sum_{r=1}^{d}(q_j)_r\int_{\mathbb{R}^d}x_rg_{jl}(x)dx\\ &\quad+\frac{1}{2}q_j^Tq_j\int_{\mathbb{R}^d}g_{jl}(x)dx\Bigg).\end{aligned}$$

Appendix B

List of all Matlab-functions

Name	**Function**
direct_ Orthogonalization	This is the main script of the Code. The user can specify the used initial data, dimension etc. there.
Normation_u	This function normalizes the current wave packet by adjusting the phase ζ_j in each Gaussian.
pre_calc_SP	Here are all done all pre-calculations. All the complex inner products of the form $\langle A_j \vert A_l \rangle$ and $\langle A_j \vert Hu \rangle$ are calculated and saved.
HG_to_gauss	Calculates the value of the integral $\int g(x)f(x)$, where f is a polynomial of low degree, from the Hagedrorn functions with the same order.
FT_Hagedorn	This function calculates the FT of the Hagedorn function φ_k for a $k \in \mathbb{N}^d$ via theorem 2.1.
convert_to_real	Here the problem is converted form a complex to a real one.
real_Gram_Schmidt	Here the presented modified Gram-Schmidt method is applied to the real problem. Further all real inner products of the form $(q_i \vert Hu)$ are calculated.
Parameter	If $X(y)_j$ is the j-th derivative of $X(y)$, Parameter(j) tells us to which derivative due to a complex parameter j is related to.
Parameter_real	If $X(y)_j$ is the j-th derivative of $X(y)$, Parameter_real(j) tells us to which derivative due to a real parameter j is related to.
ai_ql_real	Calculates in a recursive way the real inner products of the form $(a_i \vert q_l)$, where a_i is a derivative with respect to a real parameter and q_l is a orthonormalized function from the real Gram-Schmidt method.
qi_Hu_real	Calculates in a recursive way the real inner products of the form $(q_i \vert Hu)$, where q_l is a orthonormalized function from the real Gram-Schmidt method and Hu the Hamiltonian applied to the approximation u.

x_i_phi	Calculates integrals of the form $\int_{\mathbb{R}^d} x_i \varphi_0(x)\, dx$, where φ_0 is the 0-th Hagedorn function, in the way as presented in chapter 2 and appendix A.2.
xi_xj_phi_LGS	Calculates integrals of the form $\int_{\mathbb{R}^d} x_i x_j \varphi_0(x)\, dx$, where φ_0 is the 0-th Hagedorn function, in the way as presented in chapter 2 and appendix A.2.
xi_xj_xl_phi_LGS	Calculates integrals of the form $\int_{\mathbb{R}^d} x_i x_j x_l \varphi_0(x)\, dx$, where φ_0 is the 0-th Hagedorn function, in the way as presented in chapter 2 and appendix A.2.
xi_xj_xk_xl _phi_LGS	Calculates integrals of the form $\int_{\mathbb{R}^d} x_i x_j x_k x_l \varphi_0(x)\, dx$, where φ_0 is the 0-th Hagedorn function, in the way as presented in chapter 2 and appendix A.2.
mult_gauss	This function calculates the parameters of a Gaussian g which is the multiplication of two other Gaussians. Explicit formulas can be found in chapter 1.
ODE_solver	This function solves the equations of motion via the matlab function 'ODE45'.
init_Lubich	Contains the initial data for a single Gaussian as used in [FGL09].
init_rand	init_rand(m) generates m arbitrary d-dimensional Gaussians with the same width matrix $C_j = iI$.
init_test_1d	Generates one or two Gaussians of dimension 1.
init_test_4d	Generates one or two Gaussians of dimension 4.
generate_C_matrix	This function generates a matrix in $\mathbb{C}^{d\times d}$, which is complex symmetric and has positive imaginary part.