Calculating the Zero Point Energy of Ethane Using the Diffusion Quantum Monte Carlo Method

Simon Neidhart

Supervisor: Prof. Dr. Stefan Goedecker

University of Basel
November 19, 2021

Contents

Chapter 1

Introduction

Monte Carlo methods are almost as old as computers. The first application was to simulate neutron transport in fissionable material. The outcome of collisions with the material as well as the different types of collisions or reactions are the random parts of the problem and thus require random numbers to simulate in a computer. This simulation is very intuitive as the computer simulates directly what happens in nature. The mathematical reason that this works, is that the underlying integral is in fact solved by numerical integration using these random numbers. Once people realized this, the application of Monte Carlo methods started to drift away from the obvious problems such as fission and more towards any problem that needs to solve a high dimensional integral. It is very important to keep this mathematical foundation in mind as the most intuitive approaches are often not the most efficient when doing a Monte Carlo calculation.

Inspired by this, we exploit the similarity between the Schrödinger equation and the diffusion equation to develop the Diffusion Quantum Monte Carlo (DQMC) algorithm. It is rather obvious where the the randomness lies in pure diffusion and a short derivation will show, that this can also be applied to solving the many-body Schrödinger equation. In the field of quantum mechanics, Monte Carlo simulations have a long tradition as there are a lot of high dimensional integrals that need to be solved. Of the many approaches that exist in quantum Monte Carlo, DQMC is the most versatile.

In this work, the aim is to calculate zero-point energies (ZPE) with the developed DQMC algorithm. The ZPE can be understood as an energy that arises from the uncertainty of quantum mechanics. Compared to classical mechanics, where the ground state energy is just the minimum of the potential energy, in quantum mechanics there is an uncertainty in the position and the momentum that will lead to a higher ground state energy than in the classical case. This difference is known as the ZPE. Nowadays, this zero-point correction to the energy is often only approximated in the harmonic approximation. We will try to overcome this approximation, by providing better zero-point corrections to ground state energy. These correction become important for accurate thermochemical calculations

especially for semi-rigid molecules with hydrogen atoms [1]. We will therefore study different systems such as ethane and the C_{60}-Buckministerfullerene. Lastly, we test out algorithm on the $H_2@C_{60}$ endofullerene, which is presumed to have large anharmonic effects on the ZPE [2].

Chapter 2

Theoretical Background

This chapter covers the derivation and the theory behind the developed DQMC algorithm from a mathematical and physical standpoint. It aims to act as a guide to anyone who wants to understand the theory behind the algorithm or implement and use the algorithm.

2.1 The Born-Oppenheimer Approximation

When considering a system of N_{at} nuclei with positions $\boldsymbol{R}_i, ..., \boldsymbol{R}_{N_{at}}$, masses $m_1, ..., m_{N_{at}}$ and charge numbers $Z_1, ..., N_{N_{at}}$ and N electrons with positions $\boldsymbol{r}_i, ..., \boldsymbol{r}_N$, the Born-Oppenheimer approximation justifies splitting the problem (and therefore also the Hamiltonian) into two separate problems. First, the electronic part of the Hamiltonian

$$\mathcal{H}^e = -\frac{1}{2} \sum_{i=1}^{N} \nabla^2_{\boldsymbol{r}_i} + \sum_{i=1}^{N} \sum_{j=1}^{i-1} \frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} - \sum_{i=1}^{N} \sum_{j=1}^{N_{at}} \frac{Z_j}{|\boldsymbol{r}_i - \boldsymbol{R}_j|} + \sum_{i=1}^{N_{at}} \sum_{j=1}^{i-1} \frac{Z_i Z_j}{|\boldsymbol{R}_i - \boldsymbol{R}_j|} \quad (2.1)$$

containing the kinetic term of the electrons, the electron-electron, electron-nuclei and nuclei-nuclei interactions, is solved. In this work, we will use existing electronic structure codes to solve the electronic Schrödinger equation. This gives a potential energy surface (PES) $E_0^e(\boldsymbol{R}_i, ..., \boldsymbol{R}_{N_{at}})$, depending only on the position coordinates of the nuclei, with a minimum E_{min}^e. Since we do not have to worry about the electrons anymore, we will from now on write all atom coordinates in one vector $\boldsymbol{x} = \boldsymbol{R}_i, ..., \boldsymbol{R}_{N_{at}}$ of length $3N_{at}$. The second part of the problem is the nucleonic Schrödinger equation

$$\sum_{i=1}^{N_{at}} \frac{-1}{2m_i} \nabla^2_{\boldsymbol{R}_i} \psi^n + E_0^e(\boldsymbol{R}_i, ..., \boldsymbol{R}_{N_{at}}) \psi^n = \mathcal{H}\psi^n = E_0 \psi^n, \quad (2.2)$$

which we will solve for the energy E_0 using the DQMC algorithm. The Born-Oppenheimer approximation is justified by the fact that the nuclei are much heavier than the electrons and can therefore be treated classically in the electronic structure problem but then

quantum-mechanically in the nucleonic Schrödinger equation. From the point of view of the electrons, the nuclei are basically stationary and therefore, their slow movement does not influence the solution of the electronic structure problem. The nuclei then feel the potential from the electrons and move accordingly on the PES. In this work, the goal is to calculate the zero-point energy (ZPE) of a system of N_{at} atoms. The ZPE is defined as the difference between E_{min}^e and E_0. Because the kinetic energy (first term in eq. (2.2)) is always positive, E_0 is always greater than E_{min}^e and therefore the ZPE is always positive.

2.2 Monte Carlo Methods

The core of the Monte Carlo approach is to do numerical calculations by using lots of random numbers distributed according to a desired probability distribution $p(x)$ to approximate expectation values or integrals with averages over many trials.

$$\langle \mathcal{A} \rangle = \sum_{x \in \Omega} \mathcal{A}(x) p(x) \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{A}(x_i), \tag{2.3}$$

or

$$\int_{x \in \Omega} \mathcal{A}(x) p(x) d^n x \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{A}(x_i), \tag{2.4}$$

where x_i are the random numbers generated according to the probability distribution $p(x)$. Because doing a sum or integral over the entire space Ω is often not possible because the space is too large, one generates the x_i from a Markov Chain. This means that from a given x_i, $x_{i+1} = f(x_i, i, \text{"noise"})$ can be generated using a stochastic process. To ensure that the x_i follow the probability distribution $p(x)$ a acceptance-rejection step needs to be introduced. Namely, if $\alpha = \frac{p(x_{i+1})}{p(x_i)} > 1$ the new value x_{i+1} always gets accepted and if $\alpha < 1$ the new value x_{i+1} only gets accepted with probability α. Put together, these steps leads to the Metropolis algorithm (Appendix B). These ingredients form the basis of most Monte Carlo methods including the here presented DQMC algorithm.

2.3 Diffusion Quantum Monte Carlo

To get the ZPE, we now solve the nucleonic Schrödinger equation (2.2) for the energy E_0 using diffusion quantum Monte Carlo. There are different ways to derive the DQMC algorithm but the most common one is by a transformation to of the Schrödinger equation to an imaginary time $\tau = it$ [3, 4, 5]. The Schrödinger equation then becomes

$$\frac{\partial}{\partial \tau} \psi(\boldsymbol{x}, \tau) = -(\mathcal{H} - E_T) \psi(\boldsymbol{x}, \tau), \tag{2.5}$$

where the energy is shifted by the trial energy E_T. The wave function can now be written as

$$\psi(\boldsymbol{x}, \tau) = \sum_{i=1}^{\infty} e^{-(E_i - E_T)\tau} \psi_i(\boldsymbol{x}), \qquad (2.6)$$

where E_i and $\psi_i(\boldsymbol{x})$ solve the time independent nucleonic Schrödinger equation $\mathcal{H}\psi_i(\boldsymbol{x}) = E_i\psi_i(\boldsymbol{x})$ and $E_0 \leq E_1 \leq E_2 \leq \dots$. As we now increase our simulation time τ, the sum converges to the ground state wave function $\psi_0(\boldsymbol{x})$ if we adjust E_T close to E_0 as all other terms decay exponentially faster. This happens independent of the initial choice of our wave function. As we now discretize our time step, we get the known short time propagator of (2.5) for moving a particle form position $\boldsymbol{x}$ to position $\boldsymbol{y}$ during a short time step $\Delta\tau$

$$G(\boldsymbol{x_i}, \boldsymbol{y_i}; \Delta\tau) = \sqrt{\frac{1}{2\pi\Delta\tau}} e^{-(\boldsymbol{x}_i - \boldsymbol{y}_i)^2/(2\Delta\tau)} e^{-\Delta t(E_0(\boldsymbol{y}) - E_T)} + \mathcal{O}(\Delta\tau^2), \qquad (2.7)$$

for $i = 1, \dots, 3N_{at}$. This propagator has two parts. The first part $\sqrt{\frac{1}{2\pi\Delta\tau}} e^{-(\boldsymbol{x}_i - \boldsymbol{y}_i)^2/(2\Delta\tau)}$ is pure diffusion for dimension i and corresponds to a Gaussian distribution centered around $\boldsymbol{x}_i$. The second part $w = e^{-\Delta\tau(E_0^e(\boldsymbol{y}) - E_T)}$ represents a branching process. If $E_0^e(\boldsymbol{y}) > E_T$ then $w < 1$ and if $E_0^e(\boldsymbol{y}) < E_T$ then $w > 1$.

Up to this point, we have derived the unguided DQMC algorithm. To improve the convergence it is very useful to introduce a trial or guiding wave function $\psi_T(\boldsymbol{x})$. $\psi_T(\boldsymbol{x})$ should approximate $\psi(\boldsymbol{x})$ as good as possible, without requiring to much effort to compute and evaluate. Our algorithm will later need a lot of evaluations of the first and second derivative of $\psi_T(\boldsymbol{x})$. We now multiply the Schrödinger equation (2.5) with $\psi_T(\boldsymbol{x})$, define $f(\boldsymbol{x}) = \psi(\boldsymbol{x})\psi_T(\boldsymbol{x})$ and after some calculations [6] we get

$$\sum_{i=1}^{N_{at}} \frac{-1}{2m_i} \nabla_{\boldsymbol{R}_i}^2 f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}(\boldsymbol{v}(\boldsymbol{x})f(\boldsymbol{x})) + (E_L(\boldsymbol{x}) - E_T)f(\boldsymbol{x}) = 0. \qquad (2.8)$$

We have now introduced the drift velocity $\boldsymbol{v}(\boldsymbol{x}) = \nabla\psi_T(\boldsymbol{x})/\psi_T(\boldsymbol{x})$ and the local energy

$$E_L(\boldsymbol{x}) = (H\psi_T(\boldsymbol{x}))/\psi_T(\boldsymbol{x}) = \sum_{i=1}^{N_{at}} \frac{-1}{2m_i} \nabla_{\boldsymbol{R}_i}^2 \psi_T(\boldsymbol{x}) + E_0^e(\boldsymbol{x}), \qquad (2.9)$$

and similarly to the unguided case, we can get the short-time propagator

$$\tilde{G}(\boldsymbol{x_i}, \boldsymbol{y_i}; \Delta\tau) = \sqrt{\frac{1}{2\pi\Delta\tau}} e^{-(\boldsymbol{x}_i - \boldsymbol{y}_i - \boldsymbol{v}(\boldsymbol{x})\Delta\tau)^2/(2\Delta\tau)} e^{-\Delta\tau(E_L(\boldsymbol{x}) + E_L(\boldsymbol{y}))/2 - E_T)} + \mathcal{O}(\Delta\tau^2).$$
$$(2.10)$$

There are two major differences compared to the unguided propagator. In addition to the pure diffusion described by the Gaussian distribution, we now have the drift velocity, which drifts $\boldsymbol{x}$ towards a region where $f(\boldsymbol{x})$ is large. This means that the position $\boldsymbol{x}$

is now biased by the guiding wave function. Second, the local energy adds the kinetic energy of the nuclei to the evaluation of the PES and the local energy is averaged over the position before and after the propagation. This kinetic energy counteracts the drifting, since it raises the energy that was decreased by moving $\boldsymbol{x}$ towards the center of the trial wave function. With these ingredients we can construct our DQMC algorithm using a population of walkers and simulating diffusion on them. First, we just update the walker position according to the first part of the propagator by displacing it according to

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\eta}\sqrt{\Delta\tau} + \boldsymbol{v}(\boldsymbol{x})\Delta\tau, \tag{2.11}$$

where $\boldsymbol{\eta}$ is a vector of N_{at} normal distributed random numbers with mean 0 and standard deviation 1 obtained by the Box-Muller algorithm A. The branching process can be implemented by letting walkers die, survive or reproduce according to their weight w which is updated from step k to step $k+1$ according to

$$w^{(k+1)} = w^{(k)}e^{(-\Delta\tau(E_L(\boldsymbol{x})+E_L(\boldsymbol{y}))/2 - E_T))}. \tag{2.12}$$

With these non-integer weights, one can implement the split-join algorithm [7] to do the branching process. There are many alternatives, but split-join has the advantage that it limits the duplication of walker that would occur with other methods. In each iteration, the weight for each walker is updated according to (2.12). Walkers with a weight $w > 2$ split into $\lfloor w \rfloor$ walkers with weights $w/\lfloor w \rfloor$. For two walkers α and β with weight $w_\alpha < 0.5$ and $w_\beta < 0.5$ one keeps walker α with probability $w_\alpha/(w_\alpha + w_\beta)$ and walker β otherwise. The surviving walker gets the new weight $(w_\alpha + w_\beta)$. All walkers with weight $0.5 \leq w \leq 2$ survive and keep their weight. This algorithm keeps the total weight of all walkers $W^{(k)} = \sum_{i=1}^{N_w^{(k)}} w_i^{(k)}$ constant during iteration k and limits the duplication of walkers that would occur with the standard branching process. This process also ensures that even walkers with a local energy higher than E_T have a chance of surviving and therefore the entire PES is sampled if we run the simulation long enough.

The last step is to adjust the trial energy E_T. E_T controls the population of walker N_w and it therefore needs to be adjusted to keep the walker population from dying out or growing exponentially. Generally speaking, we need to decrease E_T if the number of walkers is increasing and increase E_T if the number of walkers is decreasing. Like with the split-join algorithm, there are multiple ways to adjust the trial energy [3, 8]. We will here use an approach [4] that makes use of the total weight $W^{(k)}$ after iteration k and updates E_T according to

$$E_T^{(k+1)} = E_0^{(k)} + \xi \log(W^{(k)}/W^{(0)}), \tag{2.13}$$

where $\xi > 0$ is a damping parameter, $W^{(0)}$ is the sum of weight at the start of the simulation and

$$E_0^{(k)} = \frac{\sum_{i=1}^{N_w^{(k)}} w_i^{(k)} E_L(\boldsymbol{x}_i^{(k)})}{\sum_{i=1}^{N_w^{(k)}} w_i^{(k)}}, \tag{2.14}$$

7

is the current estimation of the ground state energy E_0. This artificial adjustment to the trial energy introduces the so called population-control error. This error is generally small and can be easily controlled as it decreases as $1/N_w$. At the end of the simulation, this estimate can be improved by averaging it over may iterations after the equilibration phase giving the final result of the calculation, the energy eigenvalue of the Schrödinger equation and, by subtracting E_{min}^e, also the ZPE. It makes sense to further average the ZPE over many identical runs to get even more accuracy. The statistical uncertainty of the ZPE is then defined as the standard deviation over M identical runs [5]

$$\Delta ZPE = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (ZPE_i - \overline{ZPE})^2},$$ (2.15)

where $\overline{ZPE}$ is the average of the ZPE over the M runs.

2.4 The DQMC Algorithm

Putting all this theory together, we get the guided DQMC algorithm.

Algorithm 1 Guided DQMC

1: **procedure** DQMC($steps, \Delta\tau, E_0^e(\boldsymbol{x}), \psi_T(\boldsymbol{x}), m$)
2:     **initialize** $walkers$
3:     **for** $i = 1, steps$ **do**
4:         **for all** $\boldsymbol{x}$ **in** $walkers$ **do**
5:             **draw** $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{x}, \sqrt{\Delta\tau})$
6:             $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{\eta} + \Delta\tau \boldsymbol{v}(\boldsymbol{x})$ $\triangleright$ Moving the walker
7:             $w = w \, \exp\left(-\Delta\tau(E_L(\boldsymbol{x}) + E_L(\boldsymbol{x}_{prev}))/2 - E_T)\right)$
8:             **if** ($w > 2$) **then**
9:                 **make** $\lfloor w \rfloor$ **copies of** $\boldsymbol{x}$ **with weight** $w/\lfloor w \rfloor$ **for the next cycle**
10:            **end if**
11:            **if** ($w < 0.5$) **then**
12:                 **find another walker** $\boldsymbol{y}$ **with** $w_y < 0.5$
13:                 **draw** $r \sim \mathcal{U}([0, 1])$
14:                 **if** ($w/(w + w_y) > r$)) **then**
15:                     **make a copy of** $\boldsymbol{x}$ **with weight** $w + w_y$ **for the next cycle**
16:                 **else**
17:                     **make a copy of** $\boldsymbol{y}$ **with weight** $w + w_y$ **for the next cycle**
18:                 **end if**
19:            **end if**
20:         **end for**
21:         **update** E_T $\triangleright$ According to (2.14)
22:     **end for**
23: **end procedure**

2.5 Gaussian Trial Wave Functions

This section discusses a simple way of choosing a trial wave function that can be used for various molecular systems. The idea is to transform from the Cartesian coordinate system to a coordinate system defined by the vibrational modes. In this coordinate system, we can use the harmonic approximation to calculate a Gaussian trial wave function that gives a good approximation of the ZPE and the exact trial wave function.

First, we need to calculate the Hessian matrix of the PES

$$H_{i,j} = \frac{\partial^2 E_0^e(\boldsymbol{x})}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}\bigg|_{\boldsymbol{x}=\boldsymbol{x}^0}, \tag{2.16}$$

where $\boldsymbol{x}^0$ is the minimum energy configuration of the system. The Hessian matrix should be symmetric ($H_{i,j} = H_{j,i}$). Because it is calculated numerically it often is not exactly symmetric. It is therefore a good practice to symmetrize it by replacing it by $\frac{1}{2}HH^T$. It is also much faster and often more accurate to calculate the Hessian as the first derivative of the force vectors instead of the second derivative of the postilions. Both calculations can be easily done according to C. Then, we calculate the eigenvectors $\boldsymbol{u}^i$ and eigenvalues λ_i of the Hessian matrix

$$H\boldsymbol{u}^i = \lambda_i \boldsymbol{u}^i = m_i \omega_i^2 \boldsymbol{u}^i, \tag{2.17}$$

where m_i is the mass and ω_i the frequency of the harmonic oscillation along the vibrational mode $\boldsymbol{u}^i$.

Because a translation or rotation of the entire molecule does not change the energy, six of the eigenvalues are zero for molecules with more than two atoms and only five eigenvalues are zero for diatomic molecules. Assumig systems with more than two atoms, we now have our new coordinate system with $3N_{at} - 6$ dimensions defined by the eigenvectors $\boldsymbol{u}^i$.

For the calculation of the local energy E_L in equation (2.24) we need the effective masses for all the new $3N_{at} - 6$ dimensions. To do this, we calculate the mass-scaled Hessian matrix

$$\tilde{H}_{i,j} = \frac{1}{\sqrt{m_i m_j}} \frac{\partial^2 E_0^e(\boldsymbol{x})}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}\bigg|_{\boldsymbol{x}=\boldsymbol{x}^0}. \tag{2.18}$$

and its eigenvalues and eigenvectors

$$\tilde{H}\tilde{\boldsymbol{u}}^i = \tilde{\lambda}_i \tilde{\boldsymbol{u}}^i = \omega_i^2 \tilde{\boldsymbol{u}}^i. \tag{2.19}$$

Comparing $\tilde{\lambda}_i \boldsymbol{u}^i = \omega_i^2 \boldsymbol{u}^i$ with $\lambda_i \boldsymbol{u}^i = m_i \omega_i^2 \boldsymbol{u}^i$ from (2.17), we get $m_i = \lambda_i/\tilde{\lambda}_i$ for the effective masses of the $3N_{at} - 6$ new dimensions. From the eigenvalues of the Hessian matrix, we can now calculate the harmonic potential to approximate the PES

$$V(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \lambda_i \boldsymbol{x}_i^2 = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} m_i \omega_i^2 \boldsymbol{x}_i^2. \tag{2.20}$$

Note, that the vector $\boldsymbol{x}$ is now only of length $3N_{at} - 6$ and each element $\boldsymbol{x}_i$ describes the displacement from the minimum energy configuration along the normal mode $\boldsymbol{u}^i$. We can also calculate the ground state energy in the harmonic approximation

$$E_0^{harmonic} = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \omega_i = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \sqrt{\tilde{\lambda}_i} = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \sqrt{\frac{\lambda_i}{m_i}}. \tag{2.21}$$

This harmonic approximation energy gives a good starting value for the trial energy E_T at the beginning of the calculation. Finally we get the Gaussian trial wave function

$$\psi_T(\boldsymbol{x}) = \mathcal{N} \prod_{i=1}^{3N_{at}-6} e^{-\frac{1}{2} m_i \omega_i (\boldsymbol{x}_i - \boldsymbol{x}_i^0)^2} = \mathcal{N} \prod_{i=1}^{3N_{at}-6} e^{-\frac{1}{2} \frac{(\boldsymbol{x}_i - \boldsymbol{x}_i^0)^2}{\sigma_i^2}}, \tag{2.22}$$

where $\sigma_i^2 = 1/(m_i \omega_i) = 1/(m_i \sqrt{\tilde{\lambda}_i})$. The big advantage of Gaussian trial wave functions is that the first and second derivatives can be calculated analytically. This gives

$$\boldsymbol{v}_i(\boldsymbol{x}) = \nabla_{\boldsymbol{x}_i} \psi_T(\boldsymbol{x})/\psi_T(\boldsymbol{x}) = \frac{-(\boldsymbol{x}_i - \boldsymbol{x}_i^0)}{\sigma_i^2} \tag{2.23}$$

for the i-th element of the drift velocity and

$$E_L(\boldsymbol{x}) = (H\psi_T(\boldsymbol{x}))/\psi_T(\boldsymbol{x}) = \sum_{i=1}^{N_{at}} \frac{-1}{2m_i \sigma_i^2} \left(\frac{(\boldsymbol{x}_i - \boldsymbol{x}_i^0)^2}{\sigma_i^2} - 1 \right) + E_0^e(\boldsymbol{x}) \tag{2.24}$$

for the local energy. Gaussian trial wave functions only work well if the harmonic approximation is good. To check that, one can compare the PES with the harmonic approximation $V(x)$ for each dimension. The harmonic approximation should match the true PES as good as possible to get fast and accurate results.

2.6 Implementation Details and Improvements

2.6.1 Energy Calculations

To evaluate the potential energy surface in the calculation of $E_L(\boldsymbol{x})$, we use an external electronic structure code. As we saw in the last section, it is very simple to make calculations in the coordinate system of the normal modes $\boldsymbol{u}^i$. It therefore makes sense to keep track of the displacement from the minimal energy postion of each walker in this coordinate system and only transform back to the Cartesian representation when doing the energy evaluation. The transformation back to Cartesian coordinates is done using the simple matrix-vector multiplication $\boldsymbol{x}_{cart} = U\boldsymbol{x}$, where U is the $3N_{at}$ by $3N_{at} - 6$ matrix with the eigenvectors $\boldsymbol{u}^i$ as its columns. After that, we need to add the inital configuration to the displacements to get the full Cartesian coordinates. Now,

we can perform the energy calculation by calling a wrapper subroutine that has the form `energyandforces`$(N_{at}, \boldsymbol{x}, \boldsymbol{f}, E_0^e)$. The subroutine takes the position as an input and gives the corresponding energy and forces. It can then be implemented to call a suitable electronic structure code to perform the calculation. Because we need so many energy calculation to get accurate DQMC results, is is important to find a good compromise between accuracy and speed. For the purpose of this thesis, we used DFTB+ [9, 10] with the 3ob-3-1 parameter set for organic molecules [11]. Of course, a full density functional theory (DFT) method would give more accurate results but this would require much more computation time.

In a software package like DFTB+, the hessian matrix can directly be calculated by DFTB+ at the beginning of the simulation after doing a geometry optimization of the atomic positions. Alternatively, one can calculate the hessian by taking the derivative of the forces with respect to the position as described in C.

2.6.2 Metropolis Step

The short time propagator (2.10) introduces a time-step error of the order $\mathcal{O}(\Delta\tau^2)$. To get rid of this error, we can introduce an Metropolis step for the propagation of the walker. Instead of just moving the walker to a new position $\boldsymbol{y}$, we propose a new position $\boldsymbol{x}'$ according to (2.12), calculate a acceptance probability

$$P_{acc}(\boldsymbol{x}', \boldsymbol{x}) = \frac{P_{prop}(\boldsymbol{x}, \boldsymbol{x}')\psi(\boldsymbol{x}')^2}{P_{prop}(\boldsymbol{x}', \boldsymbol{x})\psi(\boldsymbol{x})^2} \tag{2.25}$$

with

$$P_{prop}(\boldsymbol{x}', \boldsymbol{x}) = \frac{1}{(2\pi\tau)^{(3N_{at}-6)/2}} e^{-\frac{(\boldsymbol{x}'-\boldsymbol{x}-\boldsymbol{v}(\boldsymbol{x})\tau)^2}{2\tau}} \tag{2.26}$$

and perform a Metropololis accept-reject step B. The time-step error still is a problem in the calculation of the weight in (2.12). To solve that, we calculate a effective time step $\Delta\tau_{eff} < \Delta\tau$ to take into account that certain moves are rejected. This means that we can just use the average acceptance ratio to calculate

$$\Delta\tau_{eff} = \Delta\tau \frac{N_{acc}}{N_{tot}}, \tag{2.27}$$

where N_{acc} is the number of accepted metropolis steps and N_{tot} is the total number of metropolis steps. Generally, it is more efficient to move along each dimension separately and do an accept-reject step each time compared to moving all walkers together and doing just one accept-reject step. In ether case, we only to the expensive evaluation of the PES after the accept-reject step(s). However, if we move all walkers together and the move gets rejected, we do not need to recalculate the energy. Depending on the acceptance ratio, this can save a lot of energy calculations compared to the separate moving, where it

is extremely unlikely that all the moves along each dimension get rejected. We therefore chose to move all walkers together an do just one accept-reject step.

2.6.3   Initial Configuration and Walker Initialization

One of the input parameters of the DQMC algorithm is the initial configuration of the atoms. Because we do not want to assume that this configuration is the perfectly optimized minimum energy configuration, we first minimize the energy by using a simple gradient descent algorithm D. There are several ways to initialize walkers. The easiest way is to start with all walkers at the previously calculated initial configuration where the energy is E_{min}^e. This has the disadvantage, that it takes time for the walkers to spread out and start sampling the entire PES. A better way, is to start with the walkers distributed according to $\psi_T(\boldsymbol{x})$. In the case of Gaussian trial wave functions, this can be achieved by giving the walkers the initial position $\boldsymbol{x}_i = \eta\sigma_i$ for each dimension i with a normal distributed random number η. At the end of the simulation, the walkers will be distributed according to $\psi(\boldsymbol{x})\psi_T(\boldsymbol{x})$. This is not the exact wave function, but it still can be useful to get some information about the system. A more detailed description of the input and output files of the program is given in E

2.6.4   Parallelization

A big advantage of the DQMC Algorithm is that it can be easily parallelized to run on multiple CPU cores. By far the most expensive step of the algorithm is the evaluation of the PES for the calculation of $E_L(\boldsymbol{x})$ on line 7. This can be done in parallel for all the walkers within the same time step since their are independent of each other. We therefore close the for-loop starting on line 4 after line 7 and run that part in parallel. Then we loop over all walkers again for the split-join algorithm. This part can not be done in parallel but its computational effort is negligible compared to the energy evaluation.

2.7   Unguided DQMC

Now that we established the guided DQMC algorithm, it is trivial to go back to the unguided DQMC algorithm for comparison and validation of the results. For this, we simply set $\psi_T(\boldsymbol{x}) = 1$ or $\omega_i = 0$ for all dimensions i. This means that the drift velocity vanishes and the local energy is just the potential energy given by evaluating the PES. The walker update now only consists of random diffusion without any bias, which leads to slower convergence. Further, we can not use the accept-reject step described in 2.6.2 as in the absence of a trial wave function, the acceptance probability is always one. This means that we can only control the time-step error by actually choosing a smaller time

step, which again takes more time. Since the unguided DQMC algorithm does not use a trial wave function, the calculation can ether be performed in Cartesian coordinates or in the coordinate system of the normal modes $\boldsymbol{u}^i$. As a small improvement one can use the mass to scale the random diffusion to get slightly better results [3]. This leads to the update step

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \eta\sqrt{\Delta\tau/m_i}, \tag{2.28}$$

for all dimensions i with η being a normal distributed random number. The time step $\Delta\tau$ can now be chosen much larger, depending on the average of the masses.

Chapter 3

Results

3.1 Ethane

Ethane (C_2H_6) is a small and simple molecule to test our implemetation and do some performance tests. We compare our results to literature values[12] and to experimental and computational data from the Computational Chemistry Comparison and Benchmark Database (CCCBDB)[13]. Before starting the comptuations, one can get a feel for the vibrational modes of the molecule by visualizing it using for example the PyMol plugin PyVibMS [14]. The the optimized input geometry and the vibrational modes can be calculated directly at the beginning of the algorithm.

(a) Dimension 7
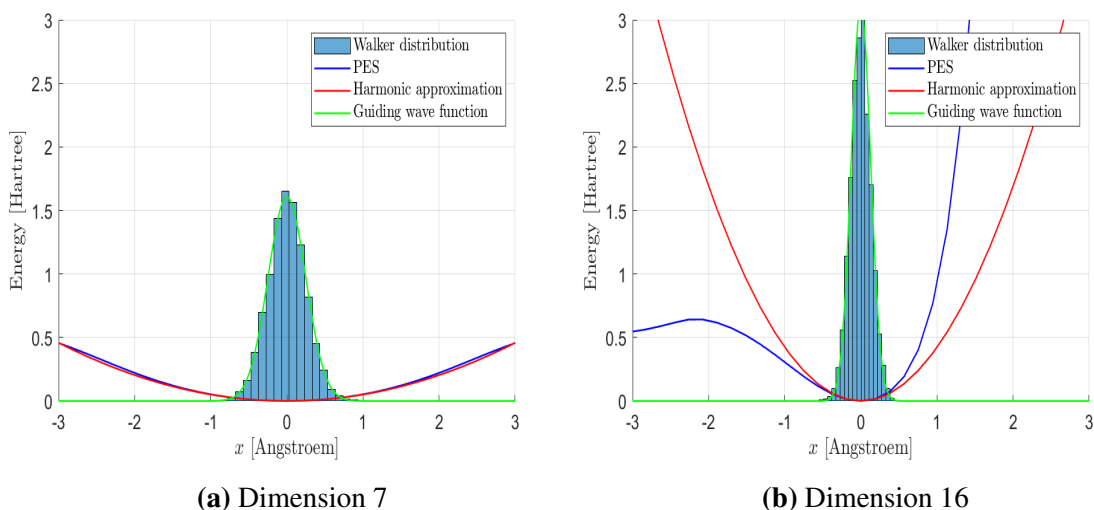
(b) Dimension 16

Figure 3.1. Comparison between the PES and the harmonic approximation for two example dimensions shows a good agreement up to the desired region of the harmonic approximation ZPE. The calculated trial wave function and the final distribution of the walkers also show excellent agreement.

Next, we can check the harmonic approximation by comparing it to the true PES and calculate the trial wave function. For the ethane molecule, the harmonic approximation gives a ZPE of 0.07264 Hartree. With the trial wave function calculated, we can start the main calculation of the ZPE. We start with 10 parallel simulations of 1000 walkers and simulate for 10000 time steps of size $\Delta\tau = 5 \cdot 10^{-4}$ with a damping parameter $\zeta = 0.01$ Hartree. After the simulation, we see that the walkers are distributed according to the trial wave function. Because the harmonic approximation is very good in this example, we expect the distribution of walkers to match the trial wave function. For the ethane molecule, the DQMC result changes the harmonic approximation by less than one percent. Therefore, it does not make much sense to run a long DQMC simulation to improve the accuracy of the ZPE compared to just taking the harmonic approximation.
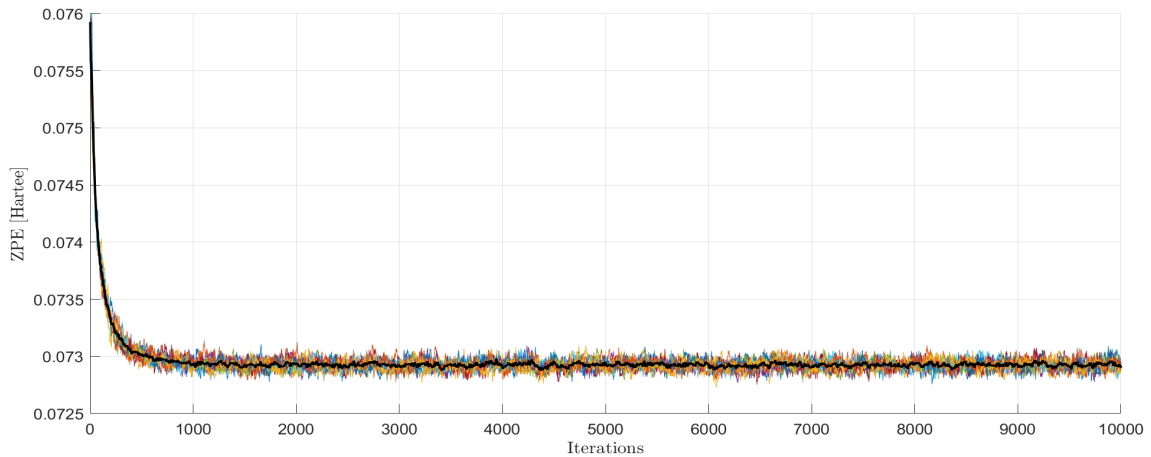
Figure 3.2. Evolution of the ZPE with time for the 10 different runs (in color) and the average over all the runs (in black)

From figure 3.2 we can see that the ZPE stays approximately constant after 2000 time steps. Therefore, we can say that the system has equilibrated after 2000 steps and we can start taking the average from there to the end of the simulation. This gives a value of 0.07293 Hartree with a statisitcal uncertainty of $2.4 \cdot 10^{-5}$ Hartree. We must keep in mind that DFTB+ is an approximate DFT method, so the result still lacks accuracy compared to a calculation with DFT energy calculations. We finally observe, that the harmonic approximation of 0.072641 Hartree is fairly close to the DQMC result, which makes sense considering the good quality of the harmonic approximation in 3.1. From the literature [12] we get a value of 0.07393 Hartree and from the CCCBDB [13] we get values forms 0.07 to 0.08 Hartree, depending on the method and basis set used and a experimental value of 0.072234 Hartree (from fundamental vibrations). Event though these values are obtained from different methods, it suggests that our method is working correctly.
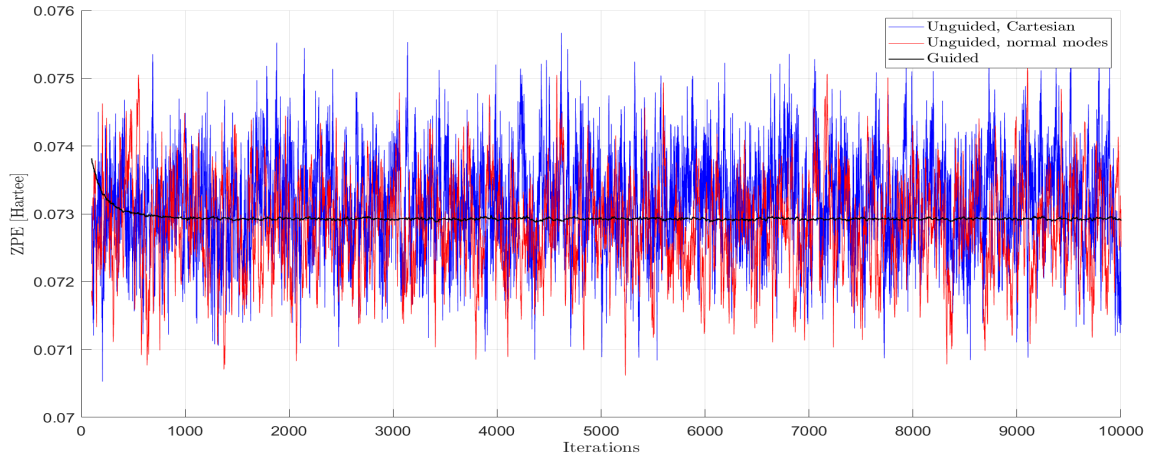
**Figure 3.3.** Comparison between guided and unguided DQMC for Cartesian and normal mode coordinates.

We can also compare our results with the unguided DQMC algorithm as described in 2.7. If we perform the calculation in Cartesian coordinates, we get a ZPE of 0.07322 Hartree with an uncertainty of 0.00076 Hartree and if we do the calculation in the coordinate system of the normal modes we get a ZPE of 0.07288 Hartree with an uncertainty of 0.00062 Hartree. Because the results are within each others uncertainties, we can conclude that both coordinate systems are equivalent in their DQMC result. The difference between guided and unguided DQMC in very noticeable. For the exact same parameters, we get about 30 to 40 times less statistical uncertainty with guided DQMC. Equivalently, one would need less than 2000 steps with guided DQMC to get the same uncertainty as with unguided DQMC for the full 10000 steps.

Method	$\overline{ZPE}$ [Hartree]	ΔZPE [Hartree]
Harmonic approximation	0.07264	
Unguided, Cartesian	0.07322	0.00076
Unguided, normal modes	0.07288	0.00062
Guided	0.07293	0.00002
Ref. [12]	0.07393	
CCCBDB experimental [13]	0.07223	
CCCBDB calculated [13]	$0.07 - 0.08$	

Table 3.1. Summary of the results for the ethane molecule and literature values for comparison.

17

3.2 Buckminsterfullerene

Due to the number of atoms, the Buckminsterfullerene (C_{60}) poses a bigger challenge to the algorithm. We start again by looking at the harmonic approximation and find, that it matches quite very well again. Visualizing the vibrations, we find that the modes with small frequencies correspond to bending and stretching movements of the entire molecule, while the modes with large frequencies correspond to stretching and compressing the bonds between the atoms. Similar to the ethane molecule, the harmonic approximation holds well for all the 176 dimension the coordinate system of the normal modes and we get a harmonic approximation of the energy of 0.381769 Hartree.
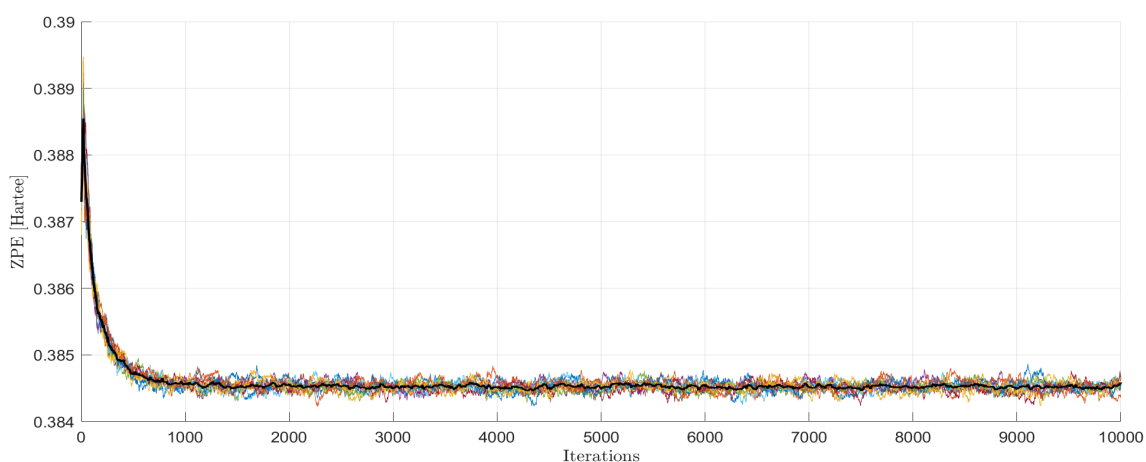
Figure 3.4. Evolution of the ZPE with time for the 10 different runs (in color) and the average over all the runs (in black). The parameters are exacly the same as for the C_2H_6

We can see that the DQMC algorithm has about the same converge behavior for the C_{60} than for the C_2H_6 but the fluctuations are a about 5 times larger and the energy fluctuates with a larger frequency. Again taking averages after 2000 simulation steps, we get a ZPE of 0.384524 Hartree with a statistical uncertainty of $1.4 \cdot 10^{-5}$. This shows the most powerful aspect of the DQMC algorithm. The convergence does not get worse with larger systems and the accuracy only decreases slightly. For comparison, we look again at the CCCBDB [13], where we find an experimental value of 0.369493 Hartree and calcuated values ranging from 0.35 to 0.40 Hartree.

The unguided DQMC algorithm does not work well at all for the C_{60}. The energy rises at the start of the simulation to about 0.6 Hartree and only decreases very slowly making it very inefficient and not worth the computer time.

Appendix A

The Box-Muller Algorithm

The Box-Muller algorithm transforms two $\mathcal{U}([0, 1])$ random numbers (u_1, u_2) to two $\mathcal{N}(0, 1)$ random numbers (x_1, x_2).

Algorithm 2 Box-Muller Algorithm

1: **procedure** BOXMULLER(u_1, u_2)
2: $\quad x_1 = sqrt(-2\log(u_1))\cos(2\pi u_2)$
3: $\quad x_2 = sqrt(-2\log(u_1))\sin(2\pi u_2)$
4: **end procedure**

Appendix B

The Metropolis Algorithm

The Metropolis algorithm generates random numbers distributed according to a (high-dimensional) probability distribution $p(\boldsymbol{x})$ using a Markov Chain. A Metropolis step generates a new random number $\boldsymbol{x}_{i+1}$ from a given $\boldsymbol{x}_i$ that follows the probability distribution $p(\boldsymbol{x})$. It can also be used as the accept-reject step after the move of a walker. In this case, f describes the propagation of the walker according to (2.11) and α becomes the acceptance probability (2.25).

Algorithm 3 Metropolis Step

1: **procedure** METROPOLIS($\boldsymbol{x}_i$)
2:     $\boldsymbol{x}' = f(\boldsymbol{x}_i,\text{"noise"})$ $\triangleright$ proposing a new random number
3:     $\alpha = \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x}_i)}$
4:     **if** $\alpha \geq 1$ **then**
5:         $\boldsymbol{x}_{i+1} = \boldsymbol{x}'$ $\triangleright$ Accepting the proposed number
6:     **else**
7:         **draw** $u \sim \mathcal{U}([0,1])$
8:         **if** $u < \alpha$ **then**
9:             $\boldsymbol{x}_{i+1} = \boldsymbol{x}'$ $\triangleright$ Accepting the proposed number
10:         **else**
11:             $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i$ $\triangleright$ Rejecting the proposed number
12:         **end if**
13:     **end if**
14: **end procedure**

Appendix C

Numerical Derrivatives for $E_L(x)$, $v(x)$ and the Hessian matrix

For a trial wave function $\psi(\boldsymbol{x})$ the first derivative used for the drift velocity $\boldsymbol{v}(\boldsymbol{x})$ is given by

$$\psi'(\boldsymbol{x}_i) \approx \frac{\psi(\boldsymbol{x}_i + h) - \psi(\boldsymbol{x}_i - h)}{2h}, \tag{C.1}$$

and the second derivative used for the local energy $E_L(\boldsymbol{x})$ is given by

$$\psi''(\boldsymbol{x}_i) \approx \frac{\psi(\boldsymbol{x}_i + h) - 2\psi(\boldsymbol{x}_i) + \psi(\boldsymbol{x}_i - h)}{h^2}, \tag{C.2}$$

where h has to be chosen sufficiently small. The Hessian can most efficiently be calculated by taking the derivatives of the forces

$$H(i, j) = \frac{\boldsymbol{f}_i(\boldsymbol{x}^{(j+)}) - \boldsymbol{f}_i(\boldsymbol{x}^{(j-)})}{2h} \quad \text{for all } i, \tag{C.3}$$

where $\boldsymbol{x}^{(j+)}$ is the vector $\boldsymbol{x}$ with the j-th element increased by h and $\boldsymbol{x}^{(j-)}$ is the vector $\boldsymbol{x}$ with the j-th element decreased by h.

Appendix D

Gradient Descent

The gradient descent algorithm takes an input configuration x with the forces f and optimizes it with respect to the minimal energy E_{min}. There is a lot of freedom in the choice of the parameters that we will not discuss here. The here given algorithm gives some reasonable choices for the parameters. The idea of the gradient descent is to move along the negative gradient or forces to a region where the forces and the energy are lower. The step size α is adjusted depending on the angle between the current forces and the previous iteration. This way the step size increases if we have consecutive moves in the same direction and decreases if we start moving in an opposite direction.

Algorithm 4 Gradient Descent

1: **procedure** GRADIENTDESCENT(x, f, E_{min}) ▷ calculate the energy and forces of a given configuration

2:   $\alpha = 10^{-3}$

3:   call energyandforces(x, f, E_{min})

4:   **while** ($||f||/N_{at} > 10^{-6}$) **do**   ▷ iterate up to a desired precision

5:     $f_{prev} = f$

6:     $x = x + \alpha f$

7:     call energyandforces(x, f, E_{min})

8:     $\beta = \arccos\left(< f, f_{prev} > / \sqrt{||f|| \, ||f_{prev}||}\right)$

9:     **if** ($\beta/N_{at} > 1.0472$) **then**   ▷ 60 degree in radians

10:       $\alpha = \alpha/2$

11:     **else**

12:       $\alpha = 1.05 \, \alpha$

13:     **end if**

14:   **end while**

15: **end procedure**

Appendix E

Input and Output Files

The DQMC program requires the following input files:

- **geometry.xyz** describes the initial positions of the atoms in the standard .xyz file format. The first line contains the number of atoms N_{at}. The second line can be used for comments and will not be read. From the third line on are the N_{at} atoms, one on each line with their atomic symbols and then their x, y, and z coordinates. The position coordinates must be in units of Angstrom.

- **masses.in** contains the masses of the atoms. The first line contains again the number of atoms and the following line contain the masses in atomic mass units in the same order as the atoms in **geometry.xyz**.

- **dqmc.in** is the input file for the parameters of the DQMC calculation. It contains the following lines:
  ```
  damping 0.01d0 #damping constant
  no_walkers_start 10000 #initial number of walkers
  delta_t 5.d-4 #discrete time step
  n_steps 10000 #number of simulation steps
  debug_extended .FALSE. #show extended information
  continue_run .FALSE. #continue a previous simulation
  grad_descent .TRUE. #do a gradient descent
  ```
  If restart is set to true, the files **et_noWalkers_e0.out** and **walker_positions.out** from the simulation that wants to be continued are also required as input files. The simulation then continues from where it finished for `n_steps` more steps. In the output file **et_noWalkers_e0.out** the new steps will be added to the end of the file.

If the simulation is run using DFTB+, the input files such as **dftb_in.hsd** and the Slater-Koster files are also required. Details can be found in the DFTB+ manual[15].

The DQMC program gives the following output files:

- **et_noWalkers_e0.out** contains the trial energy E_T, the number of walkers N_w and the current ground state estimate E_0 in its columns for each iteration. The energies are given in the Hartree energy unit.

- **walker_positions.out** contains the displacements of the walkers in the coordinate system of the normal modes at the end of the simulation. The values are all in one column starting with the $N_{at} - 6$ values for the first walker.

- **walker_positions_cartesian.out** contains the final positions of the walkers in Cartesian coordinates in the .xyz format (without the first two lines, the atomic symbol, but in the same order as in **geometry.xyz**.

- **energies_weights.out** contains the local energy E_L in column one and the weight w in column two for all the walkers.

- **eigenvalues.out** contains the eigenvalues of the Hessian matrix in ascending order. This order also corresponds to the order of the displacements in **walker_positions.out**.

- **modes.txt** can be directly loaded into the PyMol plugin PyVibMS [14] together with **geometry.xyz** to visualize the vibrational modes.

Bibliography

[1] Florian Pfeiffer, Guntram Rauhut, David Feller, and Kirk A. Peterson. Anharmonic zero point vibrational energies: Tipping the scales in accurate thermochemistry calculations? *The Journal of Chemical Physics*, 138(4):044311, 2013.

[2] Koichi Komatsu, Michihisa Murata, and Yasujiro Murata. Encapsulation of molecular hydrogen in fullerene c60 by organic synthesis. *Science*, 307(5707):238–240, 2005.

[3] Anne B. McCoy. Diffusion monte carlo approaches for investigating the structure and vibrational spectra of fluxional systems. *International Reviews in Physical Chemistry*, 25(1-2):77–107, 2006.

[4] Julien Toulouse, Roland Assaraf, and Cyrus J. Umrigar. Chapter fifteen - introduction to the variational and diffusion monte carlo methods. In Philip E. Hoggan and Telhat Ozdogan, editors, *Electron Correlation in Molecules ab initio Beyond Gaussian Quantum Chemistry*, volume 73 of *Advances in Quantum Chemistry*, pages 285 – 314. Academic Press, 2016.

[5] Apurba Nandi, Chen Qu, and Joel Bowman. Diffusion monte carlo calculations of zero-point energies of methanol and deuterated methanol. *Journal of computational chemistry*, 40, 10 2018.

[6] Cyrus J. Umrigar. Introduction to variational and projector monte carlo.

[7] C. J. Umrigar, M. P. Nightingale, and K. J. Runge. A diffusion monte carlo algorithm with very small timestep errors. *The Journal of Chemical Physics*, 99(4):2865–2890, 1993.

[8] George H. Booth, Alex J. W. Thom, and Ali Alavi. Fermion monte carlo without fixed nodes: A game of life, death, and annihilation in slater determinant space. *The Journal of Chemical Physics*, 131(5):054106, 2009.

[9] B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Y. Deshaye, T. Dumitric, A. Dominguez, S. Ehlert, M. Elstner, T. van der

Heide, J. Hermann, S. Irle, J. J. Kranz, C. Köhler, T. Kowalczyk, T. Kuba, I. S. Lee, V. Lutsker, R. J. Maurer, S. K. Min, I. Mitchell, C. Negre, T. A. Niehaus, A. M. N. Niklasson, A. J. Page, A. Pecchia, G. Penazzi, M. P. Persson, J. ezá, C. G. Sánchez, M. Sternberg, M. Stöhr, F. Stuckenberg, A. Tkatchenko, V. W.-z. Yu, and T. Frauenheim. Dftb+, a software package for efficient approximate density functional theory based atomistic simulations. *The Journal of Chemical Physics*, 152(12):124101, 2020.

[10] M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, Th. Frauenheim, S. Suhai, and G. Seifert. Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties. *Phys. Rev. B*, 58:7260–7268, Sep 1998.

[11] M. Gaus, A. Goez, and M. Elstner. Parametrization and benchmark of dftb3 for organic molecules. *Journal of Chemical Theory and Compuation*, 9(1):338354, 2013.

[12] Amir Karton, Branko Ruscic, and Jan M.L. Martin. Benchmark atomization energy of ethane: Importance of accurate zero-point vibrational energies and diagonal bornoppenheimer corrections for a simple organic molecule. *Journal of Molecular Structure: THEOCHEM*, 811(1):345–353, 2007. Computational Organic Chemistry.

[13] Russell Johnson. Computational chemistry comparison and benchmark database http://cccbdb.nist.gov. 04 2018.

[14] Yunwen Tao, Wenli Zou, and Sadisha Nanayakkara. Pyvibms: a pymol plugin for visualizing vibrations in molecules and solids. *Journal of Molecular Modeling*, 26, 09 2020.

[15] https://dftbplus.org/documentation/.