

# 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 1, 2021

# Contents

1	Introduction	2
2	Theoretical Background	3
2.1	The Born-Oppenheimer Approximation . . . . .	3
2.2	Monte Carlo Methods . . . . .	4
2.3	Diffusion Quantum Monte Carlo . . . . .	4
2.4	The DQMC Algorithm . . . . .	6
2.5	Gaussian Trial Wave Functions . . . . .	8
2.6	Implementation Details and Improvements . . . . .	9
2.6.1	Metropolis Step . . . . .	9
2.6.2	Walker Initialization . . . . .	9
2.6.3	Parallelization . . . . .	10
3	Results	11
3.1	Ethane . . . . .	11
3.2	The Quantum Harmonic Oscillator . . . . .	11
3.2.1	Comparison of guided and unguided DQMC . . . . .	12
A	The Box-Muller Algorithm	14
B	The Metropolis Algorithm	15
C	Numerical Derivatives for $E_L(x)$ and $v(x)$	16
	Bibliography	17

# Chapter 1

## Introduction

Monte Carlo methods are almost as old as computers. Its first application was to simulate neutron transport in fissionable material. Here, 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 to do 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 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 can be very useful 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 with the developed DQMC algorithm. In the first step, the algorithm is tested using the well known quantum harmonic oscillator. Then, it is applied to the molecule of ethane to calculate its zero point energy. The result is then compared to a reference calculation and we conclude that the algorithm performed very well.

# 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 the algorithm.

### 2.1 The Born-Oppenheimer Approximation

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

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

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)  $E_0^e(\mathbf{R}_i, \dots, \mathbf{R}_{N_{at}})$  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  $\mathbf{x} = \mathbf{R}_i, \dots, \mathbf{R}_{N_{at}}$  of length  $3N_{at}$ . The second part of the problem is the nucleonic Schrödinger equation

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

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 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.

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

or

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

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  $\Omega$  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  $\alpha$ . 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 two main ways to derive the DQMC algorithm but the most common one is by a transformation to of the Schrödinger equation to an imaginary time  $\tau = it$  [1, 2, ?]. The Schrödinger equation then becomes

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

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

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

where  $E_i$  and  $\psi_i(\mathbf{x})$  solve the time independent nucleonic Schrödinger equation  $\mathcal{H}\psi_i(\mathbf{x}) = E_i\psi_i(\mathbf{x})$  and  $E_0 \leq E_1 \leq E_2 \leq \dots$ . As we now increase our simulation time  $\tau$ , the sum only converges to the ground state wave function  $\psi_0(\mathbf{x})$  if we adjust  $E_T$  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 from position  $\mathbf{x}$  to position  $\mathbf{y}$  during a short time step  $\Delta\tau$

$$G(\mathbf{x}_i, \mathbf{y}_i; \Delta\tau) = \sqrt{\frac{m_i}{2\pi\Delta\tau}} e^{-(\mathbf{x}_i - \mathbf{y}_i)^2 m_i / (2\Delta\tau)} e^{-\Delta\tau(E_0(\mathbf{y}) - E_T)} + \mathcal{O}(\Delta\tau^2), \quad (2.7)$$

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

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  $\psi_T(\mathbf{x})$ .  $\psi_T$  should approximate  $\psi(\mathbf{x})$  as good as possible, without requiring too much effort to compute and evaluate. Our program will later need a lot of evaluations of the first and second derivative of  $\psi_T(\mathbf{x})$ . We now multiply the Schrödinger equation (??) with  $\psi_T(\mathbf{x})$ , define  $f(\mathbf{x}) = \psi(\mathbf{x})\psi_T(\mathbf{x})$  and after some calculation [3] we get

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

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

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

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

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

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  $\mathbf{x}$  towards a region where  $f(\mathbf{x})$  is large. 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. 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

$$\mathbf{y}_i = \mathbf{x}_i + \boldsymbol{\eta}_i + \mathbf{v}_i(\mathbf{x})\Delta\tau/m_i \text{ for } i = 1, \dots, N_{at}, \quad (2.11)$$

where  $\boldsymbol{\eta}$  is a vector of  $N_{at}$  normal distributed random numbers with mean 0 and standard deviation  $\sqrt{\Delta\tau/m_i}$  for dimension  $i$ .

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

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

With these non-integer weights, one can implement the split-join algorithm [5] 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 new weights  $w/\lfloor w \rfloor$ . For two walkers  $\alpha$  and  $\beta$  with weight  $w_\alpha < 1/2$  and  $w_\beta < 1/2$  one keeps walker  $\alpha$  with probability  $w_\alpha/(w_\alpha + w_\beta)$  and walker  $\beta$  otherwise. The surviving walker gets the new weight  $(w_\alpha + w_\beta)$ . All walkers with weight  $1/2 \leq w \leq 2$  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 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 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 [1, 4]. We will here use an approach [2] that makes use of the total weight  $W^{(k)}$  after iteration  $k$  and updates  $E_T$  according to

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

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

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

is the current estimation of the ground state energy  $E_0$ . At the end of the simulation, this estimate can be improved by averaging it over many iterations after the equilibration phase giving the final result of the calculation, the energy eigenvalue of the Schrödinger equation and therefore also the ZPE.

## 2.4 The DQMC Algorithm

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

---

**Algorithm 1** Guided DQMC

---

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

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

$$H_{i,j} = \left. \frac{\partial^2 E_0^e(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \right|_{\mathbf{x}=\mathbf{x}^0}, \quad (2.15)$$

where  $\mathbf{x}^0$  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. Then, we calculate the eigenvectors  $\mathbf{u}^i$  and eigenvalues  $\lambda_i$

$$H\mathbf{u}^i = \lambda_i \mathbf{u}^i = m_i \omega_i^2 \mathbf{u}^i, \quad (2.16)$$

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

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. This means that we now have  $3N_{at} - 6$  dimensions defined by the eigenvectors  $\mathbf{u}^i$ .

for the calculation of the local energy  $E_L$  in equation (2.22) we need the masses for all the new  $3N_{at} - 6$  dimensions. To do this, we calculate the mass-scaled Hessian matrix

$$\tilde{H}_{i,j} = \frac{1}{\sqrt{m_i m_j}} \left. \frac{\partial^2 E_0^e(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \right|_{\mathbf{x}=\mathbf{x}^0}. \quad (2.17)$$

and its eigenvalues and eigenvectors

$$\tilde{H}\tilde{\mathbf{u}}^i = \tilde{\lambda}_i \tilde{\mathbf{u}}^i = \omega_i^2 \tilde{\mathbf{u}}^i. \quad (2.18)$$

Comparing  $\tilde{\lambda}_i \tilde{\mathbf{u}}^i = \omega_i^2 \tilde{\mathbf{u}}^i$  with  $\lambda_i \mathbf{u}^i = m_i \omega_i^2 \mathbf{u}^i$  from (2.16), we get  $m_i = \lambda_i / \tilde{\lambda}_i$  for the masses of the  $3N_{at} - 6$  new dimensions.

With the  $\omega_i$ , we can now calculate the ground state energy in the harmonic approximation

$$E_0^{harm} = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \omega_i = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \sqrt{\tilde{\lambda}_i} = \frac{1}{2} \sum_{i=1}^{3N_{at}-6} \sqrt{\frac{\lambda_i}{m_i}}, \quad (2.19)$$

and the Gaussian trial wave function

$$\psi_T(\mathbf{x}) = \prod_{i=1}^{3N_{at}-6} \mathcal{N} e^{-\frac{1}{2} m_i \omega_i (\mathbf{x}_i - \mathbf{x}_i^0)^2} = \prod_{i=1}^{3N_{at}-6} \mathcal{N} e^{-\frac{1}{2} \frac{(\mathbf{x}_i - \mathbf{x}_i^0)^2}{\sigma_i^2}}, \quad (2.20)$$

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

$$\mathbf{v}_i(\mathbf{x}) = \nabla_{\mathbf{x}_i} \psi_T(\mathbf{x}) / \psi_T(\mathbf{x}) = \frac{-(\mathbf{x}_i - \mathbf{x}_i^0)}{\sigma_i^2} \quad (2.21)$$

for the drift velocity and

$$E_L(\mathbf{x}) = (H\psi_T(\mathbf{x})) / \psi_T(\mathbf{x}) = \sum_{i=1}^{N_{at}} \frac{-1}{2m_i\sigma_i^2} \left( \frac{(\mathbf{x}_i - \mathbf{x}_i^0)^2}{\sigma_i^2} - 1 \right) + E_0^e(\mathbf{x}) \quad (2.22)$$

for the local energy.

## 2.6 Implementation Details and Improvements

As we saw in the last section, it is very simple to make calculations in the coordinate system of the normal modes  $\mathbf{u}^i$ . It therefore makes sense to keep track of the position of each walker in this coordinate system and only transform back to the Cartesian representation when doing the energy evaluation for the local energy  $E_L(\mathbf{x})$ . The transformation back to Cartesian coordinates is done using the simple matrix-vector multiplication  $\mathbf{x}_{cart} = U\mathbf{x}$ , where  $U$  is the  $3N_{at}$  by  $3N_{at}$  matrix with the eigenvectors  $\mathbf{u}^i$  as its columns.

### 2.6.1 Metropolis Step

The short time propagator (2.10) introduces a time-step error. 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  $\mathbf{y}$ , we propose a new position  $\mathbf{x}'$  according to (2.12) and calculate a acceptance probability

$$P_{acc}(\mathbf{x}', \mathbf{x}) = \frac{P_{prop}(\mathbf{x}, \mathbf{x}') \psi(\mathbf{x}')^2}{P_{prop}(\mathbf{x}', \mathbf{x}) \psi(\mathbf{x})^2}, \quad (2.23)$$

with

$$P_{prop}(\mathbf{x}', \mathbf{x}) = \frac{1}{(2\pi\tau)^{(3N_{at}-6)/2}} e^{-\frac{(\mathbf{x}' - \mathbf{x} - \mathbf{v}(\mathbf{x})\tau)^2}{2\tau}}. \quad (2.24)$$

Even though a rejected step means that a expensive evaluation of the PES is wasted, it is generally worth it to add this step to the algorithm.

### 2.6.2 Walker Initialization

There are several ways to initialize walkers. The easiest way is to start with all walkers at the 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  $\psi_T(\mathbf{x})$ . In the case of Gaussian trial

wave functions, this can be achieved by giving the walkers the initial position  $\mathbf{x}_i = \eta\sigma_i$  for each dimension  $i$ . At the end of the simulation, the walkers will be distributed according to  $\psi(\mathbf{x})\psi_T(\mathbf{x})$ . This is not the exact wave function, but it still can be useful to get some information about the system.

### 2.6.3 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  $E_L(\mathbf{x})$  on line 7. This can be done in parallel for all the walkers since they 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 the time this takes is negligible.

In DMC algorithms, one can choose between moving along all the dimensions at once and then adjust the trial energy  $E_T$  or moving along all dimensions one by one and adjusting  $E_T$  after every move. Generally, the second option is slightly more efficient but it requires a lot more evaluations of the PES and would ruin the easy parallelization of the algorithm. Therefore, we stick with moving along all dimensions at once.

# Chapter 3

## Results

### 3.1 Ethane

To test and tune our algorithm, we first use the simple molecule of ethane.

### 3.2 The Quantum Harmonic Oscillator

To test the algorithm, we first let it run with the potential of the  $d$ -dimensional quantum harmonic oscillator

$$V(\mathbf{x}) = \sum_{i=1}^d \frac{1}{2} m \omega_i^2 x_i^2, \quad (3.1)$$

where  $m$  the mass and  $\omega_i = \sqrt{\frac{k_i}{m}}$  the angular frequencies with  $k_i$  the force constants. The quantum harmonic oscillator has a known analytical solutions for the quantum mechanical ground state energy in atomic units

$$E_0 = \sum_{i=1}^d \frac{1}{2} \omega_i, \quad (3.2)$$

and for the ground state wave function

$$\psi_0(\mathbf{x}) = \prod_{i=1}^d \left( \frac{m \omega_i}{\pi} \right)^{1/4} e^{-\frac{1}{2} m \omega_i x_i^2}. \quad (3.3)$$

These theoretical values allow us to compare and verify the results of our simulation.

In the following we present the results of the simulation for these parameters.

$$d = 12$$

$$\mathbf{k} = (2.0, 1.1, 0.4, 1.8, 3.2, 1.0, 3.4, 1.4, 1.1, 0.6, 1.9, 0.6)$$

$$m = 1$$

$$\text{Initial walker positions: } (0, 0, \dots, 0)$$

$$\Delta t = 0.1$$

$$\begin{aligned}
N_w^{(0)} &= 10^4 \\
E_T^{start} &= 8 \\
\xi &= 0.1 \quad A = 1 \text{ (using (2.8) to update the trial energy)}
\end{aligned}$$

With these parameters and equation (3.2), the theoretical value for the ground state energy  $E_0$  is calculated to be 7.11475.

### 3.2.1 Comparison of guided and unguided DQMC

For guided DQMC the choice of the trial wave function very much determines the convergence speed. The perfect trial wave function is the actual ground state wave function as in equation (3.3). This is a product of  $d$  Gaussian with means  $\boldsymbol{\mu} = \mathbf{0}$  and variances  $\sigma_i^2 = 1/(m\omega_i) = 1/\sqrt{mk_i}$  for  $i = 1, \dots, d$ . Using this perfect trial wave function, the DQMC algorithm converges in the first iteration step and gives the exact ground state energy up to machine precision. Obviously, for more realistic problems this the exact ground state wave function is not known and therefore cannot be used as a trial wave function. There, one needs to use the best possible approximation of the actual wave function as a trial wave function. To simulate this, we used trial wave functions with randomly shifted values for  $\boldsymbol{\mu}$  and  $\boldsymbol{\sigma}^2$ , namely

$$\tilde{\mu}_i = \mu_i + c\mathcal{U}([-1, 1]), \quad (3.4)$$

and

$$\tilde{\sigma}_i^2 = \sigma_i^2 + c\mathcal{U}([-1, 1]), \quad (3.5)$$

for  $i = 1, \dots, d$ . The parameter  $c$  determines the quality of the trial wave function.  $c = 0$  corresponds to the perfect trial wave function.

$c$	$\bar{E}_0$ (averaged over the last 8000 steps)
unguided	7.11159 standard deviation: $2.0666 \cdot 10^{-2}$
1	does not converge
0.1	7.12980 standard deviation: $3.7232 \cdot 10^{-3}$
0.01	7.11688 standard deviation: $3.8956 \cdot 10^{-4}$
0.001	7.11497 standard deviation: $3.6661 \cdot 10^{-5}$
0.0001	7.11475 standard deviation: $6.7776 \cdot 10^{-6}$

Table 1: Averaged ground state energy for different qualities of the trial wave function after a burn-in phase of 2000 time steps.

One can clearly see that the quality of the guiding wave function drastically improves the performance of the DQMC algorithm in terms of the accuracy but also in terms of the fluctuations. The gain in accuracy and the decrease in the standard deviation are roughly proportional to the quality of the trial wave function. However, if the guiding wave function is chosen poorly, the algorithm might not converge at all or one would get a better result using unguided DQMC.

# 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:    $x_1 = \text{sqrt}(-2 \log(u_1)) \cos(2\pi u_2)$   
3:    $x_2 = \text{sqrt}(-2 \log(u_1)) \sin(2\pi u_2)$   
4: end procedure
```

---

# Appendix B

## The Metropolis Algorithm

The Metropolis algorithm generates random numbers distributed according to a (high-dimensional) probability distribution  $p(\mathbf{x})$  using a Markov Chain. A Metropolis step generates a new random number  $\mathbf{x}_{i+1}$  from a given  $\mathbf{x}_i$  that follows the probability distribution  $p(\mathbf{x})$ .

---

### Algorithm 3 Metropolis Step

---

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

---



## Appendix C

### Numerical Derivatives for $E_L(x)$ and $v(x)$

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

$$\psi'(x) \approx \frac{\psi(x+h) - \psi(x-h)}{2h}, \quad (\text{C.1})$$

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

$$\psi''(x) \approx \frac{\psi(x+h) - 2\psi(x) + \psi(x-h)}{h^2}, \quad (\text{C.2})$$

where  $h$  has to be chosen sufficiently small.

# Bibliography

- [1] 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.
- [2] 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.
- [3] Cyrus J. Umrigar. Introduction to variational and projector monte carlo.
- [4] 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.
- [5] C. J. Umrigar, M. P. Nightingale, and K. J. Runge. A diffusion monte carlo algorithm with very small time-step errors. *The Journal of Chemical Physics*, 99(4):2865–2890, 1993.