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

Simon Neidhart

Supervisor: Prof. Dr. Stefan Goedecker

University of Basel
October 21, 2021

Contents

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

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

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) $E^e_0(\boldsymbol{R}_i, ..., \boldsymbol{R}_{N_{at}})$ with a minimum E^e_{min}. Since we do not have to worry about the electrons anymore, we will from now on write all atom coordinates in one vectctor $\boldsymbol{x} = \boldsymbol{R}_i, ..., \boldsymbol{R}_{N_{at}}$ of length $3N_{at}$. The second part of the problem is the nucleonic Schrödinger equation

$$\sum_{i=1}^{N_{at}}\frac{-1}{2m_i}\nabla^2_{\boldsymbol{R_i}}\psi^n + E^e_0\psi^n = E_0\psi^n, \tag{2.2}$$

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^e_{min} and E_0. Because the kinetic energy (first term in eq. (2.2)) is always positive, E_0 is always greater than E^e_{min} 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 two main ways to derive the DQMC algorithm. The first way is by a transformation to an imaginary time $\tau = it/\hbar$ [1, 2]. The second way exploits the similarity between the Schrödinger equation and the diffusion equation.

Here, we compare the Schrödinger equation (2.2) to the diffusion equation

$$\frac{\partial}{\partial t} \psi(\boldsymbol{x}, t) = \frac{1}{2} \nabla^2_{\boldsymbol{x}} \psi(\boldsymbol{x}, t) + (E - V(\boldsymbol{x})) \psi(\boldsymbol{x}, t) \tag{2.5}$$

with an additional growth or decay term on the right hand side. Comparing these two equations, we notice that for $\frac{\partial}{\partial t}\psi(\boldsymbol{x},t) = 0$ the diffusion equation satisfies the time independent Schrödinger equation and we get

$$\sum_{i=1}^{N_{at}} \frac{-1}{2m_i}\nabla^2_{\boldsymbol{R}_i}\psi(\boldsymbol{x}) + E_0^e(\boldsymbol{x})\psi(\boldsymbol{x}) = E_T\psi(\boldsymbol{x}). \tag{2.6}$$

Instead of E_0, we have now introduced the trial energy E_T. This trial energy will be adjusted during the simulation of the diffusion and will give us the solution of the Schrödinger equation E_0 once we have reached a stationary state, when $\frac{\partial}{\partial t}\psi(\boldsymbol{x},t) = 0$. To simulate this diffusion we use 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 Δt

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

for $i = 1, ..., 3N_{at}$. This propagator has two parts. The first part $\sqrt{\frac{m_i}{2\pi\Delta t}}e^{-(\boldsymbol{x}_i-\boldsymbol{y}_i)^2 m_i/(2\Delta t)}$ is pure diffusion for dimension i and corresponds to a Gaussian distribution centered around $\boldsymbol{x}_i$ giving a smaller value, the further $\boldsymbol{y}_i$ is away from $\boldsymbol{x}_i$. The second part $q = e^{-\Delta t(E_0^e(\boldsymbol{y})-E_T)}$ represents a branching process. If $E_0^e(\boldsymbol{y}) > E_T$ then $q < 1$ and if $E_0^e(\boldsymbol{y}) < E_T$ 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 $\psi_T(\boldsymbol{x})$. ψ_T should approximate $\psi(\boldsymbol{x})$ as good as possible, without requiring to much effort to compute and evaluate. Our program will later need a lot of evaluations of the first and second derivative of $\psi_T(\boldsymbol{x})$. We now multiply the Schrödinger equation (2.6) with $\psi_T(\boldsymbol{x})$, define $f(\boldsymbol{x}) = \psi(\boldsymbol{x})\psi_T(\boldsymbol{x})$ and after some calculation [3] we get

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

We have now introduced the drift velocity $\boldsymbol{v}(\boldsymbol{x}) = \nabla\psi_T(\boldsymbol{x})/\psi_T(\boldsymbol{x})$ and the local energy $E_L(\boldsymbol{x}) = (H\psi_T(\boldsymbol{x}))/\psi_T(\boldsymbol{x}) = \sum_{i=1}^{N_{at}} \frac{-1}{2m_i}\nabla^2_{\boldsymbol{R}_i}\psi_T(\boldsymbol{x}) + E_0^e(\boldsymbol{x})$ and similarly to the unguided case, we can get the short-time propagator

$$\tilde{G}(\boldsymbol{x_i},\boldsymbol{y}_i;\Delta t) = \sqrt{\frac{m_i}{2\pi\Delta t}}e^{-(\boldsymbol{x}_i-\boldsymbol{y}_i-\boldsymbol{v}(\boldsymbol{x})\Delta t)^2 m_i/(2\Delta t)}e^{-\Delta t(E_L(\boldsymbol{x})+E_L(\boldsymbol{y}))/2-E_T)} + \mathcal{O}(\Delta t^2). \tag{2.9}$$

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. 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

$$\boldsymbol{y}_i = \boldsymbol{x}_i + \boldsymbol{\eta}_i + \boldsymbol{v}_i(\boldsymbol{x})\Delta t/m_i \text{ for } i = 1, ..., N_{at}, \tag{2.10}$$

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

$$w^{(k+1)} = w^{(k)} e^{(-\Delta t(E_L(\boldsymbol{x})+E_L(\boldsymbol{y}))/2-E_T))}. \tag{2.11}$$

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 (2.11). Walkers with a weight $w > 2$ split into $\lfloor w \rfloor$ walkers with new weights $w/\lfloor w \rfloor$. For two walkers α and β with weight $w_\alpha < 1/2$ and $w_\beta < 1/2$ 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 $1/2 \leq w \leq 2$ 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

$$E_T^{(k+1)} = E_0^{(k)} + \xi \log(W^{(k)}/W^{(0)}), \tag{2.12}$$

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

$$E_0^{(k)} = \frac{\sum_{i=1}^{N_w^{(k)}} w_i^{(k)} E_L(\boldsymbol{x}_i^{(k)})}{\sum_{i=1}^{N_w^{(k)}} w_i^{(k)}}, \tag{2.13}$$

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 may 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

```
procedure DQMC(steps, Δt, E_0^e(x), ψ_T(x), m)
    initialize walkers
    for i = 1, steps do
        for all x in walkers do
            draw η ~ N(y, sqrt(Δt/m))
            x = x + η + v(x)                                   ▷ Moving the walker
            w = w · exp(−Δt(E_L(x) + E_L(x_prev))/2 − E_T))
            if (w > 2) then
                make ⌊q⌋ copies of x with weight w/⌊w⌋ for the next cycle
            end if
            if (w < 1/2) then
                find another walker y with w_y < 1/2
                draw r ~ U([0, 1])
                if (w/(w + w_y) > r)) then
                    make a copy of x with weight w + w_y for the next cycle
                else
                    make a copy of y with weight w + w_y for the next cycle
                end if
            end if
        end for
        update E_T                                             ▷ According to (2.13)
    end for
end procedure
```

2.4.1 Walker Initialization

There are several ways to initialize walkers. One is to distribute them randomly in a given simulation box. This can be done for every dimension individually by drawing uniform random numbers within the specified box bounds. Even better Alternatively, one can start with all walkers in the minimum of the PES. Since one need an equilibration phase at the start of the simulation, this gives enough time for the walkers to spread out and explore the PES.

At the end of the simulation, the walkers will be distributed according to $\psi(\boldsymbol{x})\psi_T(\boldsymbol{x})$. Unfortunately this is not the exact wave function, but it still can be useful to get some information about the system.

Chapter 3

Results

3.1 The Quantum Harmonic Oscillator

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

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

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

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

and for the ground state wave function

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

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.

$d = 12$
$\boldsymbol{k} = (2.0, 1.1, 0.4, 1.8, 3.2, 1.0, 3.4, 1.4, 1.1, 0.6, 1.9, 0.6)$
$m = 1$
Initial walker positions: $(0, 0, ..., 0)$
$\Delta t = 0.1$
$N_w^{(0)} = 10^4$
$E_T^{start} = 8$
$\xi = 0.1$ $A = 1$ (using (2.14) to update the trial energy)

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

3.1.1 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 $i = 1, ..., d$. Using this perfect trial wave function, the DQMC algorithm converges in the fist 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

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

and

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

for $i = 1, ..., d$. 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: $\quad x_1 = sqrt(-2\log(u_1))\cos(2\pi u_2)$
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 $p(\boldsymbol{x})$.

Algorithm 3 Metropolis Step

```
procedure Metropolis(x_i)
    x' = f(x_i, i, "noise")          ▷ proposing a new random number
    α = p(x') / p(x_i)
    if α ≥ 1 then
        x_{i+1} = x'                 ▷ Accepting the proposed number
    else
        draw u ~ U([0, 1])
        if u < α then
            x_{i+1} = x'             ▷ Accepting the proposed number
        else
            x_{i+1} = x_i            ▷ Rejecting the proposed number
        end if
    end if
end procedure
```

Appendix C

Numerical Derrivatives for $E_L(x)$ and $v(x)$

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

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

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

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

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.