Università degli Studi di Perugia

Dipartimento di Chimica, Biologia e Biotecnologie

Theoretical Chemistry and Computational Modelling (TCCM)

Quantum Monte Carlo Program Documentation

Gianluca Regni

Version 1.0

Contact Information:
Gianluca Regni
Email: *gianluca.regni@studenti.unipg.it*.

Contents

Chapter 1

Basic Information about QMC

The Monte Carlo (MC) method is a set of computational techniques that use random sampling to obtain numerical results. It is utilized to find solutions to mathematical problems involving numerous variables that are difficult to solve through conventional means. The efficiency of this method increases as the dimension of the problem increases.

Quantum Monte Carlo (QMC) is a computational approach which makes use of Monte Carlo techniques to study and solve complex problem in quantum mechanics field that are difficult or impossible to solve using conventional analytical methods. There are several variants of the QMC method, including:

- **Variational Monte Carlo (VMC)**: A QMC method that employs a variational approach to approximate the ground state energy of the system for arbitrary forms of ψ.
- **Diffusion Monte Carlo (DMC)**: A QMC method that simulates the diffusion of particles (or "walkers") in configuration space to approximate the ground state energy of the system.

1.1 Variational Monte Carlo

The VMC method employs a trial wavefunction, $\Psi_T(\mathbf{R})$, which is optimized to approximate the ground state of the system. The energy expectation value, which VMC aims to minimize, is given by:

$$\langle E \rangle = \frac{\int \Psi_T^*(\mathbf{R}) H \Psi_T(\mathbf{R}) d\mathbf{R}}{\int \Psi_T^*(\mathbf{R}) \Psi_T(\mathbf{R}) d\mathbf{R}} = \frac{\int |\Psi(\mathbf{R})|^2 E_L(\mathbf{R})\, d\mathbf{r}}{\int |\Psi(\mathbf{R})|^2 \, d\mathbf{R}}$$

where H is the Hamiltonian operator that represents the total energy of the system, $\mathbf{R}$ denotes the collective coordinates of all particles in the system, E_L is the local energy, and $|\Psi(\mathbf{r})|^2$ is the probability density of finding a particle at the configuration $\mathbf{R}$. The process involves adjusting the parameters of Ψ_T to find the minimum possible $\langle E \rangle$, which corresponds to the system's ground state energy.

If we can sample N_{MC} configurations $\{\mathbf{r}\}$, we can estimate E as the average of the local energy computed over these configurations. If N_{MC} is large, a Gaussian distribution is expected and the energy estimate will be calculated as:

$$E = \frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} E_L(r_i)$$

The variance of the average energies can be computed as:

$$\sigma^2 = \frac{1}{M-1} \sum_{i=1}^{M} (E_i - E)^2$$

The statistical error is then found by:

$$\delta E = \frac{\sigma}{\sqrt{M}}$$

1.2 Diffusion Monte Carlo

The DMC method stands out as a robust computational approach designed to address the quantum many-body problem. It simulates the ground state properties of quantum systems by allowing particles to "diffuse" in imaginary time, a process that inherently guides the system toward its lowest energy state. The DMC algorithm is grounded in the solution of the Schrödinger equation in imaginary time (τ), which can be represented as:

$$\frac{\partial \Psi(\mathbf{R}, \tau)}{\partial \tau} = (H - E_T)\Psi(\mathbf{R}, \tau)$$

In this equation, E_T is a shift energy, such that if we adjust it to the running estimate of E_0, we can expect that simulating the differential equation in imaginary time will converge to the exact ground state of the system. The evolution of $\Psi(\mathbf{R}, \tau)$ over imaginary time τ guides the system towards its ground state configuration. This process ensures a denser sampling in energetically favorable regions, enhancing the accuracy of the ground state estimation.

Instead of having a variable number of particles to simulate the branching process as in the DMC algorithm, a variant called Pure Diffusion Monte Carlo (PDMC) use a potential term considered as a cumulative product of weights:

$$W(r_n, \tau) = \prod_{i=1}^{n} \exp(-\delta t(E_L(r_i) - E_{\mathrm{T}})) = \prod_{i=1}^{n} w(r_i)$$

Further Reading

If you want to learn more about QMC and dive deeper into how it works, here are some easy-to-follow resources:

- *Monte Carlo Methods in Ab Initio Quantum Chemistry* by B. L. Hammond, W. A. Lester, Jr., P. J. Reynolds.
- *Quantum Monte Carlo Methods: Algorithms for Lattice Models* by James Gubernatis, Naoki Kawashima, Philipp Werner
- *Quantum Monte Carlo* on Wikipedia
- *QMC tutorial of the 2021 LTTC winter school* by Anthony Scemama, Vijay Gopal Chilkuri, Claudia Filippi
- Check out arXiv.org for the latest research papers on QMC.

Chapter 2

User Guide

The Quantum Monte Carlo program is a tool designed to estimate the average energy and its associated error of quantum systems through Monte Carlo simulations. It is specifically developed to accommodates five key cases: H, He, H_2, H_2^+, and H_3^+. This program leverages quantum mechanics principles and statistical methods to provide insights into the properties of quantum systems.

The software accommodates both VMC and PDMC techniques, offering a versatile tool for analyzing quantum systems. It processes input parameters and configurations of the system, conducts Monte Carlo simulations, and delivers outcomes including the system's average energy and acceptance ratio, alongside their corresponding errors. Structurally, the software integrates two separate components: `qmc.f90`, the main program, and `qmc_routines.f90`, a module containing essential functions and subroutines for executing the simulations.

Author and Credits

- **Author**: Gianluca Regni - *gianluca.regni@studenti.unipg.it*

2.1 QMC Program Components

The QMC Program is structured into two main components: the main program, identified as `qmc.f90`, and a supplementary module that contains essential routines for QMC simulations, named `qmc_routines.f90`. This chapter provides an overview of each component and its roles in performing Quantum Monte Carlo simulations.

2.1.1 Main Program (`qmc`)

The main program acts as the orchestrator for initiating and managing the flow of the QMC simulation process. Its responsibilities include:

- Reading input variables.
- Performing input validation for the simulation by ensuring that the number of electrons and atoms does not exceed predefined limits.
- Converting units of system geometry to atomic unit.
- Calculating the number of electrons in the system based on the provided configuration.
- Determining the simulation type (VMC or PDMC) from the input keyword and executing the corresponding method.
- Managing the overall simulation process, from initialization to the final output.

2.1.2 QMC Routines Module (`qmc_routines`)

This module is a collection of subroutines and functions essential for conducting the simulations. The routines include:

Wave Function (`psi`)

The function `psi` calculates and returns the wave function value for a specific system configuration. Given the inputs: wave function parameter `a`, electron positions `r`, number of atoms `natoms`, nuclei positions `geom`, and number of electrons `nelec`, the routine performs the following steps:

1. Initialize the wave function value ψ to 1.
2. For each electron, iterate over all nuclei to calculate the cumulative interaction effect. This is done by computing the Euclidean distance between the electron and each nucleus, applying an exponential function modulated by a, and summing these contributions to get a total interaction effect e for the electron.
3. Update ψ by multiplying it with e, effectively accumulating the interaction effect across all electrons.

The final value of ψ after completing these steps represents the wave function of the system, approximated as the product of one-electron wave functions. The mathematical expression for calculating the interaction effect e for an electron with all nuclei is given by:

$$e = \sum_{j=1}^{\texttt{natoms}} \exp\left(-a \cdot \sqrt{(r_{i,1} - \texttt{geom}_{j,1})^2 + (r_{i,2} - \texttt{geom}_{j,2})^2 + (r_{i,3} - \texttt{geom}_{j,3})^2}\right)$$

where i indexes over electrons, and j indexes over nuclei.

An example of trial wave function used for H_2 is:

$$\psi(H_2) = (e^{-\alpha r_{1,A}} + e^{-\alpha r_{1,B}}) \cdot (e^{-\alpha r_{2,A}} + e^{-\alpha r_{2,B}})$$

where $r_{1,A}$, $r_{1,B}$, $r_{2,A}$, and $r_{2,B}$ are the various electron-nucleus distances, and α is the sole variational parameter.

Potential Energy (`potential`)

The `potential` function calculates the local total potential energy of a quantum system composed of electrons and nuclei. It accounts for both attractive and repulsive interactions, modeled according to Coulomb's law. The attractive interactions occur between electrons and nuclei, while the repulsive interactions occur among electrons themselves and among the nuclei. The Coulomb interaction implies that these energies are inversely proportional to the distance between the interacting particles.

The potential energy V is calculated as the sum of three parts: attractive V_{att}, electron-electron repulsive $V_{elec-elec}$, and nucleus-nucleus repulsive $V_{nuc-nuc}$:

$$V = V_{att} + V_{elec-elec} + V_{nuc-nuc}$$

1. Attractive Part V_{att}:

$$V_{\text{att}} = \sum_{i=1}^{\text{natoms}} \sum_{j=1}^{\text{nelec}} \left(-\frac{z_i}{|\vec{r}_j - ge\vec{o}m_i|}\right)$$

where $|\vec{r}_j - ge\vec{o}m_i|$ is the Euclidean distance between electron j and nucleus i and $z(i)$ is the atomic number of nucleus i.

2. Electron-Electron Repulsive Part $V_{\text{elec-elec}}$:

$$V_{\text{elec-elec}} = \sum_{i=1}^{\text{nelec}-1} \sum_{j=i+1}^{\text{nelec}} \left(\frac{1}{|\vec{r}_i - \vec{r}_j|}\right)$$

This term accounts for the repulsion between different electrons, ensuring each pair is only counted once.

3. Nucleus-Nucleus Repulsive Part $V_{\text{nuc-nuc}}$:

$$V_{\text{nuc-nuc}} = \sum_{i=1}^{\text{natoms}-1} \sum_{j=i+1}^{\text{natoms}} \left(\frac{z_i z_j}{|ge\vec{o}m_i - ge\vec{o}m_j|} \right)$$

This term calculates the repulsion between different nuclei, with each pair being considered once.

The routine ensures that calculations are only performed when distances are greater than zero to avoid division by zero errors. Warnings are printed if a zero distance is encountered, indicating a divergence in the potential energy calculation.

1-electron-1-nucleus Wavefunction (`psi_1e1N`)

The function computes the contribution to the wavefunction of one electron and one nucleus using the formula $e^{-a \cdot |r_e - R_N|}$, where r_e and R_N are the positions of the electron and nucleus, respectively. This calculation is essential for determining the derivatives (or Laplacian) of the wavefunction, which are used in the calculation of local kinetic energy and drift vector in QMC simulations.

Kinetic Energy (`kinetic`)

The local kinetic energy is defined as:

$$T_L(r) = -\frac{1}{2} \frac{\Delta\psi(r)}{\psi(r)}$$

The routine calculates the local kinetic energy of a wave function for specific atomic and molecular systems: hydrogen atom (H), dihydrogen cation (H_2^+), helium atom (He), hydrogen molecule (H_2), and the trihydrogen cation (H_3^+). The function calculates the kinetic energy by first determining the distances between each electron and nucleus and then applying these distances to specific formulas depending on the type of system.

Local Energy (`e_loc`)

The `e_loc` function computes the local energy, E_{loc}, for a given configuration of a quantum system. This local energy is a sum of local kinetic energy, T_{loc}, and the local total potential energy, V_{loc}, of the system.

The local energy calculation is performed as follows:

$$E_{\text{loc}} = T_{\text{loc}} + V_{\text{loc}}$$

Average and Error Calculation (`ave_error`)

The `ave_error` subroutine calculates the average and statistical error of a dataset represented by an array. If the array contains only one element, it sets the average to that element and the error to zero, addressing the simplest case where no variability exists. For arrays with more than one element, the subroutine first computes the average as the sum of all elements divided by the number of elements:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

It then calculates the variance as the sum of squared differences between each element and the average, divided by one less than the number of elements:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$

The statistical error is derived from the variance, calculated as the square root of the variance divided by the number of elements:

$$e_S = \sqrt{\frac{\sigma^2}{n}}$$

Random Number Generation (`random_gauss`)

The subroutine `random_gauss` is designed to populate a matrix `z` with Gaussian-distributed random numbers. It operates by generating `nset` sets of numbers, each set containing `n` numbers. The generation process employs the Box-Muller transform, a method for producing normally distributed (Gaussian) random numbers from uniformly distributed random numbers.

The Box-Muller transform is mathematically represented as follows. Given two independent random numbers u_1 and u_2 uniformly distributed in the interval (0, 1], two independent normally distributed random numbers z_0 and z_1 can be generated as:

$$z_0 = \sqrt{-2\log(u_1)}\cos(2\pi u_2),$$
$$z_1 = \sqrt{-2\log(u_1)}\sin(2\pi u_2).$$

Drift Calculation (`drift`)

The subroutine `drift` is designed to compute the drift vector $\mathbf{D}$ ($\frac{\nabla\psi}{\psi}$). It is specific for hydrogen atom (H), dihydrogen cation (H_2^+), helium atom (He), hydrogen molecule (H_2), and the trihydrogen cation (H_3^+) systems. To achieve this, the subroutine first measures the distances between each electron and nucleus. Following this, it applies these measured distances within specific formulas tailored to the system in question.

Variational Monte Carlo (`vmc`)

The subroutine `vmc` is designed to implement the VMC method for estimating the average energy of a system. It takes as input parameters the variational parameter a, the step size δt, the maximum number of Monte Carlo steps $n_{\max}$, the geometry of the system `geom`, the number of atoms n_{atoms}, the number of electrons n_{elec}, and the atomic numbers of the atoms Z. The output parameters include the estimated average energy of the system `energy` and the acceptance ratio of the Monte Carlo simulation steps `accep`.

The subroutine starts by initializing variables, including the square root of the step size $\sqrt{\delta t}$, and setting the initial energy and acceptance count to zero. The initial positions of the electrons are generated using a Gaussian distribution, and the initial drift vector is calculated based on these positions.

For each Monte Carlo step, the subroutine performs the following actions:

1. Calculate the local energy at the current configuration and accumulate it into the total energy.
2. Propose a new configuration for the electrons by moving them according to a Gaussian-distributed random displacement and the drift vector, scaled by δt and $\sqrt{\delta t}$, respectively:
$$r' = r_n + \delta t\frac{\nabla\psi}{\psi} + \sqrt{\delta t}\ \chi$$
3. Calculate a new drift vector for the proposed configuration.
4. Evaluate the wave function for the new configuration.
5. Apply the Metropolis acceptance criterion. The algorithm is:
   - Calculate the acceptance probability $A = \frac{T(\mathbf{r'}\to\mathbf{r_n})P(\mathbf{r'})}{T(\mathbf{r_n}\to\mathbf{r'})P(\mathbf{r_n})}$
     where $\frac{T(\mathbf{r'}\to\mathbf{r_n})}{T(\mathbf{r_n}\to\mathbf{r'})} = \exp\left[(\mathbf{r_{n+1}} - \mathbf{r_n})\cdot\left(\frac{\nabla\Psi(\mathbf{r_{n+1}})}{\Psi(\mathbf{r_{n+1}})} + \frac{\nabla\Psi(\mathbf{r_n})}{\Psi(\mathbf{r_n})}\right) + \frac{1}{2}\delta t\left(\frac{(\nabla\Psi(\mathbf{r_{n+1}}))^2}{\Psi(\mathbf{r_{n+1}})^2} - \frac{(\nabla\Psi(\mathbf{r_n}))^2}{\Psi(\mathbf{r_n})^2}\right)\right]$
     and $P(r) = |\psi(r)|^2$
   - Generate a uniform random number $\nu \in [0,1]$
   - If $\nu \leq A$ accept the move: set $r_{n+1} = r'$
   - else, reject the move: set $r_{n+1} = r_n$

Finally, the subroutine computes the average energy by dividing the total accumulated energy by the number of Monte Carlo steps, and the acceptance ratio by dividing the number of accepted steps by the total number of steps.

Pure Diffusion Monte Carlo (`pdmc`)

The routine implements the PDMC method to estimate the average energy of a system. The algorithm integrates several parameters including the wave function parameter a, step size δt, imaginary time variable τ, system geometry *geom*, Monte Carlo steps n_{max}, number of atoms n_{atoms}, number of electrons n_{elec}, atomic numbers z, and the energy shift E_{ref}. The output of the routine consists of the estimated average energy of the system and the acceptance ratio of the Monte Carlo steps simulation.

The algorithm begins with initializing variables including the square root of the step size $\sqrt{\delta t}$, the initial energy, the number of accepted steps, and the normalization factor. Electron positions are initialized using a Gaussian distribution. The drift vector based on current positions is computed, followed by the calculation of the initial wave function value.

In the main Monte Carlo loop, the local energy e is evaluated at each step. The weight factor w is updated using the exponential of the negative product of step size δt and the difference between the local energy e and the reference energy E_{ref}:

$$w(r_n) = e^{-\delta t(E_L(r_n)-E_{\text{ref}})}$$

The weighted energy and normalization factor are accumulated to estimate the average energy. The algorithm also includes a mechanism to reset the weight and the current imaginary time when the specified τ is reached.

A new configuration of electron positions is proposed by adding a displacement consisting of the current drift vector $\frac{\nabla\psi}{\psi}$ and a random Gaussian displacement χ scaled by $\sqrt{\delta t}$:

$$r' = r_n + \delta t \frac{\nabla\psi}{\psi} + \sqrt{\delta t}\ \chi$$

The drift vector for the new positions is recalculated, and the square of the new drift vector is computed.

The wave function value at the new positions is evaluated, and the Metropolis acceptance criterion is applied to decide whether to accept the new configuration (refer to Paragraph VMC in Section 2.1.2 for Metropolis algorithm).

Finally, the acceptance ratio is calculated as the ratio of the number of accepted steps to the total number of Monte Carlo steps. The average energy of the system is computed by dividing the accumulated weighted energy by the normalization factor:

$$E = \frac{\sum_{k=1}^{N_{MC}} E_L(\mathbf{r}_k) W(\mathbf{r}_k, k\delta t)}{\sum_{k=1}^{N_{MC}} W(\mathbf{r}_k, k\delta t)}$$

Input Reader (`read_input`)

This subroutine, named `read_input`, is designed to read configuration parameters from the input file named *qmc.in* (see Section 2.4 to details about configuration) for initializing a quantum Monte Carlo (QMC) simulation. The routine begins by opening the file. It first reads the number of atoms involved in the simulation, stored as `natoms`, which is a critical parameter for allocating memory for subsequent data structures.

Following this, the subroutine allocates space for two arrays: `atom`, which holds the chemical symbols of the atoms involved, and `geom`, a two-dimensional array that stores the Cartesian coordinates (x, y, z) of each atom. The charge of the system, `charge`, is then read, indicating the net charge of the molecule.

For each atom, the subroutine reads its chemical symbol and coordinates from the file, filling the previously allocated arrays.

After initializing the atomic configuration, the subroutine proceeds to read the simulation parameters. These include the computational method (`method`), which determines the specific QMC algorithm to be used, and several numerical parameters critical for the simulation's execution: the variational parameter `a`, the step size `dt`, the maximum number of MC steps `nmax`, and the number of MC runs `nruns`.

If the selected method is *pdmc* (PDMC), additional parameters `E_ref` (reference energy) and `tau` (imaginary time) are read.

The subroutine concludes by closing the input file, having fully initialized the simulation's configuration and parameters. This process ensures that the QMC simulation is ready to proceed with a clear definition of the physical system, computational method, and operational parameters.

Output Management (`output`)

This subroutine is specifically designed to handle the output related to QMC simulations. It performs two primary tasks. First, it prints the input variables to the console, detailing the system's molecular charge, atomic composition including atomic numbers and their Cartesian coordinates, and various parameters governing the simulation such as the chosen method, wavefunction parameter, step size, and, if applicable, the energy shift and imaginary time variable specific to the PDMC method. It also displays the number of MC steps and runs being executed.

Second, the subroutine writes a comprehensive summary of both the input variables and the results of the simulation to an ouput file. This summary includes detailed information about the molecular system, the simulation parameters (similarly to what is printed to the console), and the final results (the computed average energy and acceptance ratio alongside their errors)

Atomic Number Finder (`find_atomic_number`)

The subroutine `find_atomic_number` is designed to map a set of chemical elements to their corresponding atomic numbers. It operates on the principle that each chemical element is uniquely identified by its atomic number, which is a sequential integer starting from 1 for Hydrogen, 2 for Helium, and so on, up to 118 for Oganesson, the heaviest element recognized as of this writing.

Given an input array `input_element` containing symbols of chemical elements and its size `nelem`, the subroutine searches through a predefined, ordered array `elements` that lists all 118 known chemical elements by their atomic number. For each element in the input array, the subroutine compares it against the `elements` array to find a match. Upon finding a match, it assigns the index of the matched element in the `elements` array, which corresponds to the atomic number of that element, to the output array `z` at the same position as the input element.

The process can be outlined mathematically as follows:

$$\text{For each } e_j \in \texttt{input\_element},\ \text{find } i \text{ such that } \texttt{elements}[i] = e_j,$$

and then set

$$\texttt{z}[j] = i,$$

where i and j are indices running from 1 to 118 and 1 to `nelem`, respectively. It is important to note that the subroutine initializes each position in the output array `z` to 0 before the search begins. This initialization serves as an indicator that if an element is not found within the `elements` array, its atomic number remains 0 in the output array, signaling an unsuccessful search.

2.2 Preparing The Environment

Before running the QMC Program, it is essential to ensure that your computational environment is correctly set up. This chapter walks you through the process of installing a Fortran compiler, if you haven't done so already, and verifying its installation to prepare for QMC simulations.

The QMC Program is written in Fortran 90, requiring a Fortran compiler to translate the source code into executable software. `gfortran`, part of the GNU Compiler Collection (GCC), is recommended for its compatibility and ease of installation.

For Linux Users

1. **Open your terminal.** This can usually be done through your system's application menu or by using a keyboard shortcut, often Ctrl+Alt+T.
2. **Update your package list** to ensure you can download the latest version of the compiler. Use:

```
sudo apt update
```

3. **Install `gfortran`.** Execute the following command:

```
sudo apt install gfortran
```

4. **Confirm the installation** by checking the version of `gfortran` installed:

```
gfortran --version
```

Other Fortran compilers may also work. For detailed instructions on installing `gfortran`, please refer to this guide. Further information about `gfortran` can be found on its official web page.

2.3 Running the QMC Program

For Linux users, the following steps outline how to compile and run the QMC program:

1. Create the input file named `qmc.in`. For details about the input configuration, refer to Section 2.4.
2. Compile the program using the commands below in your terminal:

```
gfortran -c qmc_routines.f90
gfortran qmc_routines.o qmc.f90 -o qmc
```

3. Run the executable:

```
./qmc
```

4. After running the program, a summary output file named `qmc.out` will be generated. This file contains the input variables you provided and the results from the QMC simulation.

If you are using a different operating system, such as macOS or Windows, the steps to compile and run the program are similar but may require specific adjustments. Ensure that you have a compatible Fortran compiler installed and refer to the compiler's documentation for precise command syntax. For Windows users, using a command-line environment like Cygwin or WSL (Windows Subsystem for Linux) may be necessary to execute Linux-like commands.

2.4 Configuration

Preparing the correct configuration for the QMC simulations is crucial for obtaining accurate and meaningful results. This section explore into the details of the input file, `qmc.in`, guiding you through the process of editing and understanding the various parameters you can control.

The `qmc.in` file serves as the configuration tool for your QMC simulations, allowing you to specify the system parameters, simulation settings, and methodological preferences. Each line in the file corresponds to a different parameter or set of parameters that influence the behavior of the QMC program.

All quantities must be given in *atomic units* (except for geometry, which should be in angstroms; see below for more details). The results will be presented in *atomic units* too.

The file is formatted as plain text, making it editable with any basic text editor software. When editing the file, ensure that each parameter is correctly specified according to the documentation. Incorrect values or formats can lead to errors in the simulation or inaccurate results.

The input file structure is divided into three main sections:

1. **Geometry Section:**
   - **Number of atoms** (*natoms*): An integer specifying how many atoms (or nuclei) are in the system.
   - **Molecular charge** (*charge*): An integer indicating the total charge of the molecule.
   - **Atom position** (*geom*): For each atom or nucleus, specify the chemical element symbol (as a string) followed by its position in three-dimensional space (X, Y, Z coordinates as real numbers), in angstroms.
2. **Method Section:**
   - **Monte Carlo method** (*method*): A string specifying the Monte Carlo method to be used, either `"vmc"` for Variational Monte Carlo or `"pdmc"` for Pure Diffusion Monte Carlo.

3. **Parameters Section:**

   - **Variational parameter** (a): A real number representing a parameter of the wavefunction, related to atomic number of the atoms of the molecule.
   - **Step size** (dt): A real number that specifies the step size used in the Monte Carlo simulation, influencing how the simulation explores the space of possible configurations.
   - **Number of MC steps** ($nmax$): An integer that defines the total number of Monte Carlo steps per run, determining the length of the simulation and its potential accuracy.
   - **Number of MC runs** ($nruns$): An integer that sets how many separate runs of the simulation to perform, which can help in averaging results to improve accuracy.
   - **Energy shift** (E_ref) **for PDMC**: A real number, specific to the PDMC simulation and is not relevant for VMC methods. It represents the constant offset to the energy.
   - **Imaginary time variable** (tau) **for PDMC**: A real number, specific to the PDMC simulation and not applicable for VMC methods. It indicates the imaginary time in the Schrödinger equation.

A sample input file might look like this:

```
! Geometry (angstrom)
2  ! number of atoms (or nuclei) [Integer]
0  ! molecular charge [Integer]
H -0.049222 0.000000 0.1000000  ! Atom type [String], X, Y, Z positions [Real]
H -0.049222 0.000000 0.8408481  ! Atom type [String], X, Y, Z positions [Real]

! Simulation method
vmc  ! Monte Carlo method used: "vmc" (Variational) or "pdmc" (Pure Diffusion) [String]

! Simulation parameters
1.2d0   ! Wavefunction parameter (a) [Real]
0.01d0  ! Step size (dt) [Real]
100000  ! Number of MC steps (nmax) [Integer]
30      ! Number of MC runs (nruns) [Integer]
-1.1d0  ! Energy shift (E_ref) for pdmc [Real] (Not applicable for vmc)
100.d0  ! Imaginary time variable (tau) for pdmc [Real] (Not applicable for vmc)
```

2.5 Output

When you run the example input file mentioned in the previous section, you will get an output on your screen that looks something like this:

```
--------- System -----------

- Molecular Charge:            0

Atom, Atomic Number (Z),  Cartesian Coordinates (ang)
H            1 -0.45060001247180542        0.0000000000000000       0.10000000276782189
H            1 -0.45060001247180542        0.0000000000000000       0.84084812327317771

-- Simulation Parameters --

- Method: vmc
- Wavefunction Parameter (a):    1.2000000000000000
- Step Size (dt):    1.0000000000000000E-002
- Number of MC Steps:                 100000
- Number of MC Runs:           30

--------- Results ---------

E =   -1.0487273097129339      +/-   2.6987596316455187E-003
A =   0.82891033333333330      +/-   2.7112599748716545E-004
```

Additionally, a summary file will be created in your local directory. This file will contain content similar to the following:

```
==================== Simulation Summary ====================

----------------------------
          System
----------------------------

- Molecular Charge:  0

Atom, Atomic Number (Z),  Cartesian Coordinates (ang)
H   1 -0.450600  0.000000  0.100000
H   1 -0.450600  0.000000  0.840848


----------------------------
         Simulation
----------------------------

- Method: vmc
- Wavefunction Parameter (a):   1.200
- Step Size (dt):  0.010
- Number of MC Steps:     100000
- Number of MC Runs:         30


----------------------------
          Results
----------------------------

Energy (Eh): -1.048727 +/- 0.3E-02
Acceptance Ratio:  0.828910 +/- 0.3E-03
```

The output simulation is divided into three sections: System, Simulation, and Results. The System section introduces the hydrogen molecule (H_2) that was specified in the input file. The Simulation section outlines the method and parameters used in the simulation. The method is the Variational Monte Carlo (vmc). Variational wavefunction parameter set to 1.2, the step size for the simulation at 0.01, the number of Monte Carlo steps as 100,000, and that the simulation was run 30 times. Finally, the Results section provides the outcomes of the simulation. It gives the energy of the system in Hartree units as -1.048727 with the relative error, and an acceptance ratio of about 0.829 with the relative error.