Theosamuele Signor
1230667
theosamuele.signor@studenti.unipd.it

Quantum Monte Carlo

06/03/2020

Abstract

In this work we will study the Diffusion Monte Carlo method (one variant of Quantum Monte Carlo) to find numerically the ground state of many-body quantum systems. The main theoretical steps for the development of this method will be given. Hereafter the actual algorithm will be developed using FORTRAN programming language, and the energy and wave function of the ground state obtained for different quantum systems will be compared with exact, experimental or other numerical solutions.
Such benchmark few-body systems used in this work are: 1D Harmonic Oscillator, Morse Oscillator, Hydrogen Atom, H^-, $H_2{}^+$, $H_3{}^+$ ions and H_2 molecule.

Theory

The Diffusion Monte Carlo method formulation starting point is the time-dependent Schrödinger equation. For simplicity, we consider a single particle in one dimension x.

$$i\hbar\frac{\partial}{\partial t}\psi(x,t) = \hat{H}\psi(x,t) \tag{1}$$

where

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x) \tag{2}$$

The general solution for this equation is:

$$\psi(x,t) = e^{-\frac{it}{\hbar}\hat{H}}\psi(x,0) \tag{3}$$

It is possible to expand Eq. 3 in eigenstates of $\hat{H}$:

$$\psi(x,t) = \sum_{n=0}^{\infty} c_n e^{-\frac{i}{\hbar}E_n t}\phi_n(x) \tag{4}$$

Where E_n is the eigenvalue of the eigenstate ϕ_n. The coefficients c_n describe the overlap of the initial state $\phi(x,o)$ with the correspondent eigenstates:

$$c_n = \int_{-\infty}^{+\infty} \phi_n^*(x)\psi(x,t)dx \tag{5}$$

In this way the solution of the Schrödinger equation is a linear combination of solutions to the time independent counterpart.

Imaginary Time Schödinger Equation

Coming back to Eq.1, we now introduce an arbitrary energy shift, i.e. $V(x) \to V(x) - E_R$ (where E_R is a reference energy) and perform the substitution $it \to \tau$ (Wick Rotation of Time) in Eq. 4, we get:

$$\hbar\frac{\partial}{\partial t}\psi(x,\tau) = \frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x,t) - [V(x) - E_R]\psi(x,t) \tag{6}$$

$$\implies \psi(x,t) = \sum_{n=0}^{\infty} c_n e^{-\frac{E_n - E_R}{\hbar}\tau}\phi_n(x) \tag{7}$$

expanding the sum,

$$\psi(x,\tau) = c_0 e^{-\frac{E_0 - E_R}{\hbar}\tau}\phi_0(x) + c_0 e^{-\frac{E_1 - E_R}{\hbar}\tau}\phi_1(x) + ... + c_n e^{-\frac{E_n - E_R}{\hbar}\tau}\phi_n(x) \tag{8}$$

it is easy to see that, if E_R is equal to the ground state energy E_0,

$$\lim_{\tau\to+\infty}\psi(x,\tau) = c_0\phi_0(x) \tag{9}$$

While, if $E_R > E_n$, the exponential diverges and if $E_R < E_n$ the exponential goes to zero.
We now need to evolve the wave function in imaginary time for some value of the reference energy. However, since it has to be equal to the ground state energy, which is unknown, we need also a way to update and modify the Reference energy during the time steps.
Given the probability amplitude (propagator) for the particle to move from x_{n-1} to x_n in a short time $\delta\tau$ ($G(x_n, \tau + \delta\tau | x_{n-1}, \tau)$), the solution of the Schrödinger equation (Eq 6) can be written in terms of the integral:

$$\psi(x_n, \tau + \delta\tau) = \int_{-\infty}^{+\infty} G(x_n, \tau + \delta\tau | x_{n-1}, \tau)\psi(x_{n-1}, \tau)dx_{n-1} \tag{10}$$

$G(x_n, \tau + \delta\tau | x_{n-1}, \tau)$ also has satisfy the imaginary Schröodinger equation. If the second term in Eq 6 can be neglected, the solution for $G(x_n, \tau + \delta\tau | x_{n-1}, \tau)$ of the Imaginary-time Schödinger equation is:

$$P(x_n, x_{n-1}) \coloneqq G_d(x_n, \tau + \delta\tau | x_{n-1}, \tau) = \sqrt{\frac{m}{2\pi\hbar\delta\tau}} exp[-\frac{m(x_n - x_{n-1})^2}{2\hbar\delta\tau})] \tag{11}$$

If instead the first term is neglected the solution is:

$$W(x_n) \coloneqq G_B(x_n, \tau + \delta\tau | x_{n-1}, \tau) = exp[-\frac{V(x_n) - E_R}{\hbar}\delta\tau] \tag{12}$$

The transition probability can be approximated to the second order of $\delta\tau$ by the product of these 2 different solutions.
Eq. 10 thus becomes:

$$\psi(x_n, \tau + \delta\tau) = \int_{\infty}^{+\infty} dx_{n-1} W(x_n) P(x_n, x_{n-1}) \psi(x_{n-1}, \tau) \tag{13}$$

It is important to notice that $P(x_n, x_{n-1})$ is related to the kinetic energy term of the Hamiltonian (2) and can be taught of a Gaussian probability distribution of x_n with mean μ and variance σ such that

$$\begin{aligned} \mu &= x_{n-1} \\ \sigma &= \sqrt{\frac{\hbar\delta\tau}{m}} \end{aligned} \tag{14}$$

The function $W(x_n)$ (called weight function) is instead related to the potential energy.
To compute the integral (13) it is necessary to apply some Monte Carlo algorithm.

Digression: Monte Carlo Integration

Any N-dimensional convergent integral of the form

$$I = \int_{-\infty}^{+\infty} dx_0 ... \int_{-\infty}^{+\infty} dx_{N-1} f(x_0, ..., x_{N-1}) \rho(x_0, ..., x_{N-1}) \tag{15}$$

being $\rho(x_0, ..., x_{N-1})$ a probability density, can be interpreded as an expected value of a random variable, i.e.

$$I = \frac{1}{M} \sum_{i=1}^{M} f(x_0^i, ..., x_{N-1}^i) \tag{16}$$

where x^i are generating randomly according to the distribution ρ. The larger M the better the approximation of the integral is. The error of this computation is proportional to $1/\sqrt{M}$.

If we interpret the wave function as a probability density, it is possible to compute the integral 13 by means of the Monte Carlo Method (Eq. 16):

$$\psi(x_n, \tau + \delta\tau) = \frac{1}{N} \sum_{\substack{\text{i=1} \\ x_i \in \psi(x,t)}} W(x_n) P(x_n, x_{n-1}) \tag{17}$$

However, this interpretation implies that the wavefunction must a be normalized and positive definite function. This limists the applicability of this method.

The Diffusion Monte Carlo Method

To obtain the wave function of the ground state we need to compute the time integral (Eq. 13) enough times to reach the limit $\tau \to \infty$.
In the Diffusion Monte Carlo algorithm this is achieved by a series of diffusion and branching processes, given respectively by Eq. 11 and 12: the idea is to evolve in imaginary time a large number ($\mathcal{N}$) of "replicas", each described by its position at time $\tau = k\delta\tau$. The ground state wave function is represented by the average density of walkers at large time.
It is convenient to to set

$$\psi(x_0, 0) = \delta(x - x_0) \tag{18}$$

i.e. all the replicas initially are in the same position; to have a good overlap with the ground state wave function, it is a good choice to set x_0 value in a region where the potential is small. Here are explained the processes taking place at every iteration:

- in the diffusion process the position of the new replica x_k is generated according to a Gaussian Distribution with mean $\mu = x_{k-1}$ with standard deviation $\sigma = \sqrt{\hbar\delta\tau/m}$. In a more computer-friendly way:

  $$x_n = x_{n-1} + g\sqrt{\frac{\hbar\delta\tau}{m}} \tag{19}$$

  where g is a gaussian random number with $\mu = 0$ and $\sigma = 1$.

- The potential part modifies the weight. Instead of accumulating the weights for every replica, it is convenient to replicate (or kill) particles according to their weight; each particle is replaced by a number of

  $$m_n = min\{int[W(x_n) + u], 3\} \tag{20}$$

  replicas, where u is a random number uniformly distributed in the interval $[0, 1]$; the new born replicas have the same position as their mother.
  The maximum number of copies is set to 3 to avoid numerical instabilities; this limitation, however results in a small error, since, if $\delta\tau << 1$, $W(x_n)$ (Eq. 12) can be approximated by

  $$W(x) \approx 1 - \frac{V(x) - E_R}{\hbar}\delta\tau \tag{21}$$

  which is never too larger than 1.

Summarizing the DMC method implementation, the following steps are taken:

- initialize a number $\mathcal{N}_0$ of replicas $x_o^{(j)}$, distributed according to a distribution $\psi(x_0, 0)$ which is chosen to be $\delta^D(x - x_0)$, i.e. all of them are in the same position. Every replica is caracterized by a position x_n^j where j counts the replicas and n the diffusive displacement.

- diffuse the replicas according to Eq 11,

  $$x_1^{(j)} = x_o^{(j)} + g_1^{(j)}\sqrt{\hbar\delta\tau/m} \tag{22}$$

- compute the weight:

  $$W(x_1^{(j)}) = exp[-\frac{V(x_n^{(j)}) - E_R}{\hbar}\delta\tau] \tag{23}$$

- determine $m_1^{(j)}$ (Eq. 20) and:

  - if $m_1^{(j)} = 0$ terminate the replica
  - if $m_1^{(j)} = 1$ the replica is kept alive
  - if $m_1^{(j)} = 2$ the replica is kept alive and a copy of it is added
  - if $m_1^{(j)} = 3$ the replica is kept alive and two copies of it are added

  in this way, the number of alive replicas changes at every diffusive displacement. This value is indicated as $\mathcal{N}_n$.

- adapt the reference energy E_R. As explained above, only for $E_R = E_0$ we expect a stable asymptotic distribution such that the number of alive replicas fluctuates around $\mathcal{N}_0$. To achieve this behaviour, we start from the first equation in (21) averaged over all replicas:

$$\overline{W}_1 \approx 1 - \frac{\overline{V}_1 - E_R}{\hbar}\delta\tau \tag{24}$$

  Since we want to keep the number of replicas approximately fixed, $\overline{W}$ has to be equal to 1. Then, we might set $E_R^1 = \overline{V}_1$.
  However, if the number of replicas changes too much, it is possible to compensate by another choice of the reference energy: in fact, if $\mathcal{N}_1/\mathcal{N}_0 > 1$ we would like to reduce the number of replicas; on the other hand, if $\mathcal{N}_1/\mathcal{N}_0 < 1$ we would like to increase their number. This leads us to the following choice of the reference energy:

$$E_R = \overline{V}_1 + \alpha(1 - \frac{\mathcal{N}_1}{\mathcal{N}_0}) \tag{25}$$

  where α is an arbitrary positive number, set empirically.

This procedure is iterated until, after a large enough number of (time τ) steps the reference energy becomes stationary.
Once the procedure is completed, the replicas will be spatially distributed according to the ground state wave function $\phi_0(x)$ and, since $\mathcal{N}_1/\mathcal{N}_0 \to 1$:

$$E_0 = \lim_{n\to\infty} \overline{V}_n \tag{26}$$

All what is said above regards a particle in one dimension; however, this algorithm can easily be generalized to S particles in D dimensions.

Particle in D dimension

The Hamiltonian of a particle in D dimension is:

$$\hat{H} = -\frac{\hbar^2}{2m}(\sum_{\nu=1}^{D} \frac{\partial^2}{\partial x_\nu^2}) + V(\{x_\nu\}) \tag{27}$$

where D is the spatial dimension of the system.
The kinetic energy is just a sum of kinetic energies of D particles having the same mass. Then, in practice, we assign the coordinates for each replica (for example, if $D = 3, (x, y, z)$) and then for every direction apply the diffusive displacement.

S Particles

In the case we have S particles in D dimensions, the Hamiltonian is (assuming no internal degrees of freedom are involved):

$$\hat{H} = \sum_{j=1}^{S}[-\frac{\hbar^2}{2m_j}(\sum_{\nu=1}^{D} \frac{\partial^2}{\partial x_{j,\nu}^2}) + V(\{x_{j,\nu}\})] \tag{28}$$

which, defining

$$x_{j,\nu} = x'_{j,\nu}\sqrt{\frac{m}{m_j}} \tag{29}$$

becomes equal in form as the one (Eq. 27) describing one particle moving in $S \times D$ dimensions.

Code Development

The described algorithm was implemented in FORTRAN as a main SUBROUTINE which takes as input:

- N_0 : the initial number of alive replicas;
- $Nmax$: the maximum number of alive replicas;
- τ : the number of time steps;
- $\delta\tau$: the time step size;
- D : the diimension of the system;

- *potential choice* : potential energy of the system to study;
- *initial position* : initial position of all replicas (as said we chose to initialize all the replicas at the same position;
- E_R : initial reference energy
- x_{min} : the lower limit of the spatial coordinates
- x_{max} : the upper limit of the spatial coordinates
- n_b :the number of intervals $[x_{min}, x_{max}]$ is divided in

Good results (see next section) are achieved with $N_0 = 1000$, $Nmax = 2000$, $\tau = 4000$, $\delta\tau = 0.05$.
The input parameters D, *initial position*, *potential choice* depend on the system we are willing to study. This choice is taken by the user, and, accordingly its dimensions, initial position of replicas, and reference energy.
To initialize the replicas, a matrix called psi is created with N_{max} rows and $D + 1$ columns. The first column indicates if the replica is dead or alive; i.e. if the replica j is alive $psi(j, 1) = 1$, if it is dead $psi(j, 1) = 0$. The other columns are its spatial coordinates.
The first N_0 replicas are thus set alive.

```
SUBROUTINE DMC(N0,Nmax,tau,delta_tau,D,initial_position,E_R,xmin,xmax,nb,potential_choice)
       INTEGER N0,Nmax,tau,potential_choice,D,m
       integer*4 w, nb
       REAL*8 delta_tau,u,E_R,v,vmean,ww,xmin,xmax,spatial_bin_step
       INTEGER ii,jj,DD,N,BB
       DOUBLE PRECISION, DIMENSION(:,:),ALLOCATABLE::psi,temp_psi
       DOUBLE PRECISION, DIMENSION(:)::initial_position
       DOUBLE PRECISION, DIMENSION(:),ALLOCATABLE::BUCKET,bin_centers !for 1D histograms

       ALLOCATE(psi(Nmax,D+1))
       ALLOCATE(temp_psi(Nmax,D+1))


       !spatial bin step: will be used for the 1d spatial distribution
       spatial_bin_step=(xmax-xmin)/(nb*1d0)
       IF (D .EQ. 1) THEN
           ALLOCATE(BUCKET(nb+1))
           BUCKET(:)=0
       END IF
       allocate(bin_centers(nb))

       !!!!STARTING DMC!!!!
       psi(:,:)=0
       !Setting first N0 replicas alive
       psi(:N0,1)=1

       !initial position
       DO DD=1,D
           psi(:N0,DD+1)=initial_position(DD)
       END DO
```

After initializing psi, we can enter the loop over time, and another one over replicas: if the replica is alive, it is applied first the diffusion process:

```
           DO ii=1,2*tau
               !number of alive replicas
               N=0
               vmean=0d0

               temp_psi=psi
               DO jj=1,Nmax
                   IF (ABS(psi(jj,1)-1d0) <= 1E-15) THEN
                       N=N+1 !enlarge the number of alive replicas
                       !walk
                       DO DD=1,D
                           CALL RANDOM_NORMAL(u)
                           u=u*sqrt(delta_tau)
```

```
                    temp_psi(jj,DD+1)=temp_psi(jj,DD+1)+u
                END DO
```

where the subroutine RANDOM NORMAL is user-defined to return a Gaussian-distributed random number using the $GASDEV$[4] method.

```
SUBROUTINE RANDOM_NORMAL(u)
    REAL*8 xx,yy,u
    CALL RANDOM_NUMBER(xx)
    CALL RANDOM_NUMBER(yy)
    u=sqrt(-2d0*log(xx))*cos(2*4*atan(1d0)*yy)
END SUBROUTINE RANDOM_NORMAL
```

Once the new coordinates are obtained, the weight function is computed and the birth-death process begins:

```
        !weight calculation
        V=POTENTIAL(temp_psi(jj,2:),potential_choice)
        vmean=((N-1)*vmean+v)/(N*1d0)!strange method to compute iteratively the mean of a
            sequence
        CALL RANDOM_NUMBER(u)
        w=int(exp((E_R-v)*delta_tau)+u)
        w=abs(w)
        m=min(w,3)
       !branching
       CALL BRANCH(m,temp_psi,jj)
```

Where the subroutine BRANCH, as the name might suggest, is dedicated to this task:

```
    SUBROUTINE BRANCH(m,branched_psi,j)
        integer m,j,ix
        DOUBLE PRECISION branched_psi(:,:)
        !finding where we have not born/dead replicas
        integer, allocatable:: indeces(:)
        indeces=pack([(ix,ix=1,size(branched_psi(:,1)))],abs(branched_psi(:,1))<=1e-15)
        !kill replica
        IF (M .EQ. 0) THEN
            branched_psi(j,1)=0.

        !if m=1 do nothing
        !if m=2 create a copy of the particle in the
        !first position we have a dead one
        ELSEIF (M .EQ. 2) THEN
            branched_psi(indeces(1),1)=1.
            branched_psi(indeces(1),2:)=branched_psi(j,2:)

        !if m=3 create 2 copies
        ELSEIF (M .EQ. 3) THEN
            !first copy
            branched_psi(indeces(1),1)=1.
            branched_psi(indeces(1),2:)=branched_psi(j,2:)

            !second copy
            branched_psi(indeces(2),1)=1.
            branched_psi(indeces(2),2:)=branched_psi(j,2:)
        END IF
        DEALLOCATE(indeces)

    END SUBROUTINE BRANCH
```

After τ iterations, the stationarity is reached and if $D = 1$ spatial distribution is computed.
This allows us to compute the distribution of replica in a cumulative way, not only for the last iteration.
This, since at this point the same distribution is provided, yields to better statistics for the wave function.
If the system under study is a ion or a molecule, even the positions of the replicas is stored on the file *positions.txt* If $D > 1$ the program computes the radial distribution of replica.

```
                IF (ii .gt. tau) THEN
```

```
                    IF (D .EQ. 1) THEN
                        DO BB=1,nb !loop over buckets
                            IF ((temp_psi(jj,2) .ge. xmin+(BB-1)*spatial_bin_step) .and. &
                            (temp_psi(jj,2) .lt. xmin+(BB)*spatial_bin_step)) THEN
                                Bucket(BB) = Bucket(BB) + 1
                            END IF
                        END DO
                    ELSE
                        Radius=norm2(temp_psi(jj,2:))
                        DO BB=1,nb !loop over buckets
                            IF ((Radius .ge. xmin+(BB-1)*spatial_bin_step) .and. &
                            (Radius .lt. xmin+(BB)*spatial_bin_step)) THEN
                                Bucket(BB) = Bucket(BB) + 1
                            END IF
                        END DO

                    END IF !way to make histogram depending on dimension

                END IF !if time iteration > tau, we start to evaluate psi
            END IF !if replics is alive
        END DO !loop over replicas
        WRITE(44,*) E_R
```

Since the replicas are distributed according to ϕ_0, the radial probability density is proportional to a spherical shell volume element; this implies that the radial distribution of replicas is proportional to

$$\mathcal{P}(r) = 4\pi r^2 \phi_0(r) \tag{30}$$

Thus, the radial distribution of the replicas is divided by r^2 to compute the radial wavefunction.
At the end of the time loop, this distribution is saved on the file *wavefunction.txt*.
As can be seen from the subroutines above, both the number of alive replicas and their mean potential are computed; this is useful to calculate the reference energy update, as defined in Eq. 25.

```
    !reference energy update
        E_R = vmean+(1-(N*1d0/N0))
        psi=temp_psi
        WRITE (2,*) N
```

I this work, we chose $\alpha = 1$, which provided a good behaviour of the number of alive replicas and E_R as function of tau.
When the stationarity is reached (i.e. after τ time steps), the positions of the alive replicas is written on a file.
When the simulation is completed (it takes $2 * \tau$ time steps), we end up with 3 files:

- '$N.txt$' which contains the number of alive replicas after every time iteration;
- '$ref\_energy.txt$' which contains the reference energy after every time iteration: the ground state energy will be its mean after the stationarity is reached;
- if $D = 1$, *wavefunction.txt* which contains the wavefunction of the system, otherwise it contains the radial wavefunction.
- in the case of ions or molecules, *positions.txt*, which contains the coordinates of the replicas

The FORTRAN program is launched via a python script.
This script, when the simulation is completed, reads the files described above to:

- plot the trace of the number of alive replicas;
- plot the trace of the reference energy;
- plot the groundstate (radial) wavefunction,

The ground state energy (and its standard deviation) is (are) computed by averaging the reference energy along the time iterations after τ iterations:

```
print('Estimated E_0 = %0.3f'%np.mean(Er[int(len(Er)/2):]), 'Std =
    %0.3f'%np.std(Er[int(len(Er)/2):]) )
```

Summarizing, to start the simulation an the subsequent result analysis, the user should run the python script *FinalProject-Signor-CODE.py*, which compiles and executes the Fortran Script *FinalProject-Signor-CODE.f90*; this first asks the user which quantum system, among the available ones, the user wants to compute the ground state of; according to this choice the dimension of the system is set and then the simulation launched; once completed, the python script reads the output and computes the energy and ground state wavefunction of the system and compares the results with theoretical or other numerical ones if available.

Results

The results obtained by the described algorithm generalized to $D \times N$ spatial dimensions, where N is the number of particles of the quantum system and D the actual dimension of the space where they are moving, for different quantum systems are given below.
Good results are achieved with the following values of the input parameters:
$N_0 = 1000,\ Nmax = 2000,\ \tau = 4000,\ \delta\tau = 0.05$.

1D Harmonic Oscillator

The first and easier system considered is the 1-dimensional harmonic oscillator, which is just 1 particle moving along 1 axis under the potential:

$$V(x) = \frac{m\omega^2}{2}x^2 \tag{31}$$

which, setting $\omega = 1,\ m = 1$, becomes:

$$V(x) = \frac{1}{2}x^2 \tag{32}$$

The Schrödinger equation corrisponding to this potential is analytically solvable: the ground state energy in dimensionless units is $E_0^{ex} = 0.5$ and its (unnormalized) wavefunction is

$$\phi_0(x) = e^{-x^2/2} \tag{33}$$

The behaviour of the mean reference energy and the ground state wavefunction provided by the DMC simulation is shown in Figure 1 And the delivered results for E_0 are:

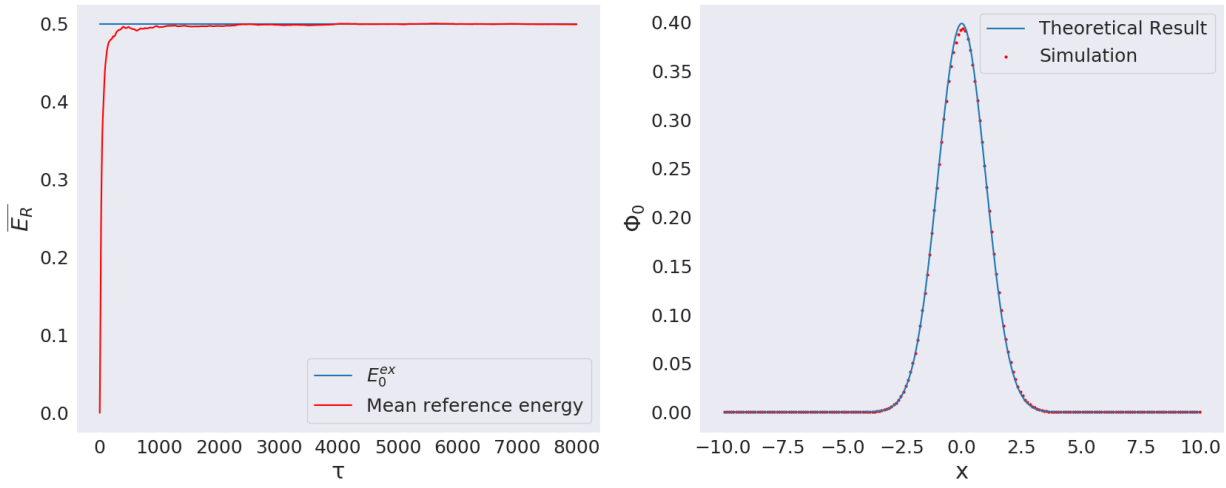

Figure 1: Simulation results for the Harmonic Oscillator. (***Left***) Reference energy E_R as function of the imaginary time τ. (***Right***) Estimated ground state wavefunction compared to the exact one.

$$E_0 = 0.500\ \ \sigma = 0.026 \tag{34}$$

Morse Oscillator

The Morse Oscillator is a particle moving in one dimension too, but under the potential

$$V(x) = -\alpha(2e^{-\beta x} - e^{-2\beta x}) \tag{35}$$

Choosing $\alpha = 1$ and $\beta = 1$, the exact ground state energy is $E_0^{ex} = -0.125$ and the corresponding wavefunction

$$\phi_0(x) = -\sqrt{2} exp[\frac{x}{2} + e^{-x}] \tag{36}$$

The behaviour of the mean reference energy and the ground state wavefunction provided by the DMC simulation is shown in Figure 2. The delivered results for E_0 are:

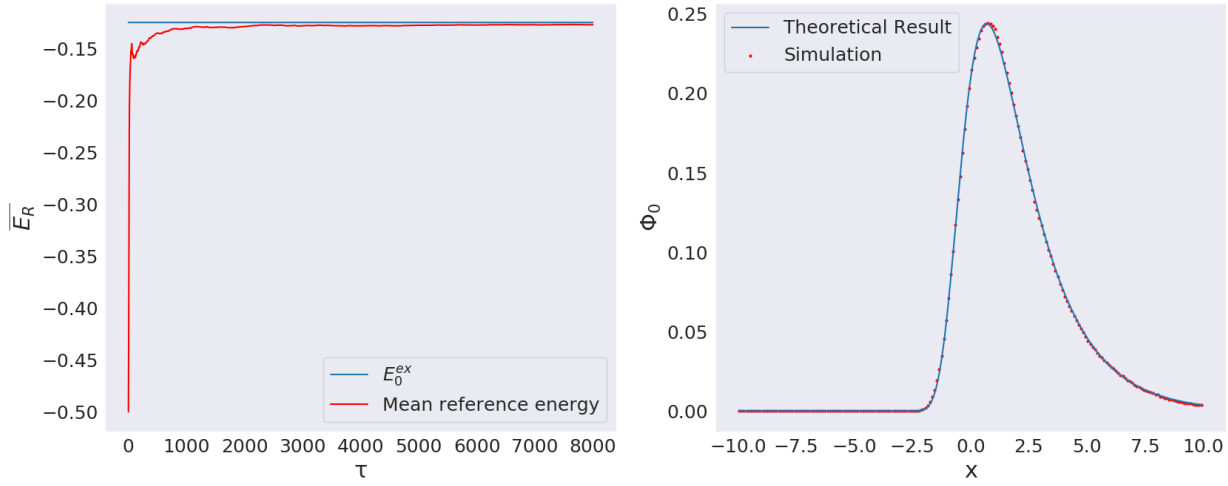

Figure 2: Simulation results for the Morse oscillator.(***Left***) Reference energy E_R as function of the imaginary time τ. (***Right***) Estimated ground state wavefunction compared to the exact one.

$$E_0 = -0.126 \ \ \sigma = 0.027 \tag{37}$$

Hydrogen Atom

The Hydrogen atom consists in an eletron orbiting in the 3-dimensional space around a proton under the potential

$$V(r) = -\frac{e^2}{4\pi\epsilon_0 r} \tag{38}$$

where e is the electron charge, ϵ_0 the vacuum permittivity and r the distance electron-proton.
Choosing $a_0 = 4\pi\epsilon_0\hbar^2/me^2 = 1$[1] (Bohr radius), since we have already chosen $\hbar = 1$ we get

$$V(r) = -\frac{1}{r} \tag{39}$$

The exact unnormalized ground state wavefunction is

$$\phi_0(x) = e^{-r} \tag{40}$$

and the corresponding energy:

$$E_0^{ex} = -0.5 \tag{41}$$

The delivered results for the ground state energy are:

$$E_0 = -0.495 \ \ \sigma = 0.030 \tag{42}$$

Results are shown in Fig 3. As can be seen in the right panel, the distribution of the radii simulated is slightly below the expected one. This can be explained by the fact that the maximum number of copies for replicas is set to 2 (Eq. 20). This is based on the fact that in general the weight approximated (Eq.21) is around unity. However, Coulomb potentials may diverge for $r \to 0$, leading to an infinite number of copies to make. Nonetheless, the results obtained are in good agreement with the expected ones.

Ground state energies

For some ions and molecules the ground state energy was computed

[1] $a_0 = 5.29 \times 10^{-11} m \implies E[eV] = 27.71 E[adim\ units]$

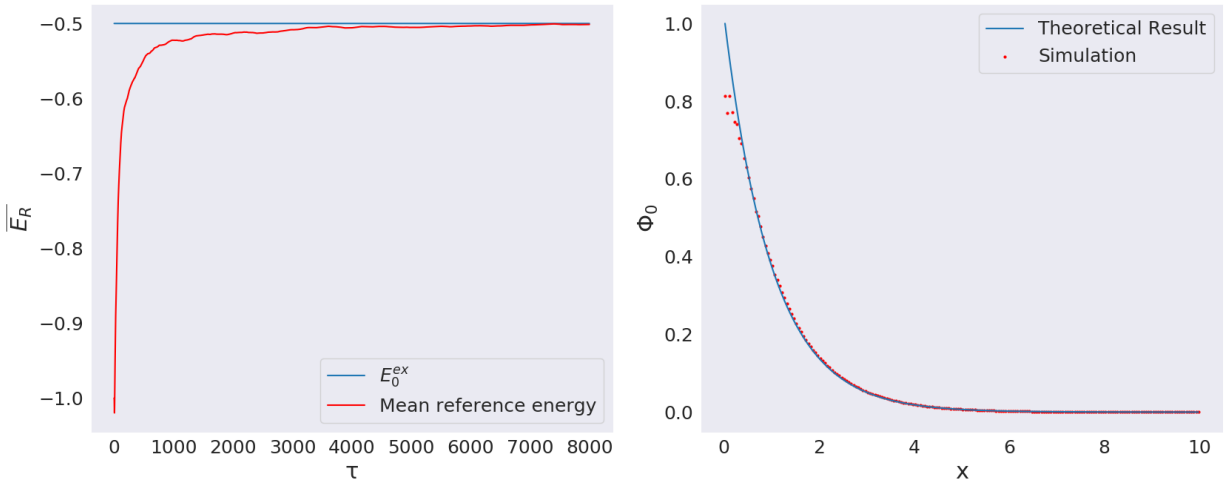

Figure 3: Simulation results for the Hydrogen atom.(***Left***) Reference energy E_R as function of the imaginary time τ. (***Right***) Estimated ground state wavefunction compared to the exact one.

H^- ion

The Hydrogen anion is composed by two electron orbitating around a proton (located at the origin). The potential is then composed by three components, one for every couple of particles, and takes the form (using the same units as before)

$$V(x) = -\frac{1}{r_A} - \frac{1}{r_B} + \frac{1}{|r_A - r_B|} \tag{43}$$

where r_i is the position of the electron i. The total ground state energy is computed to be

$$E_0 = -0.515 \ \ \sigma = 0.029 \tag{44}$$

in good agreement with the expected result of $E_0^{ex} = -0.528$

H_2^+ ion

The dihydrogen cation (H_2^+) is composed by one electron orbitating around two protons, the latters separated, at equilibrium by a distance $R = 1.06\mathring{A}$, corresponding to a ground state energy $E_0^{ex} = -0.597$[2]. Since we are working in units of the Bohr radius a_0, $R = 2.003a_0$, we set the first proton in $(0, 0, 1)$ and the second $(0, 0, -1)$. Results obtained:

$$E_0 = -0.603 \ \ \sigma = 0.037 \tag{45}$$

H_3^+ ion

The Trihydrogen cation is composed by three protons and two electrons orbiting around. The lowest energy found in literature ([1]) is $E_0^{ex} = -1.344$ for protons placed in an equilateral triangle with $R = 1.66a_0$. Thus, our simulation was run for protons in the following positions: $P_1 = (0, 0, 1)$, $P_2 = (0, 0, 2.66)$, $P_3 = (0, 1.4376, 1.83)$

$$E_0 = -1.347 \ \ \sigma = 0.067 \tag{46}$$

H_2 molecule

The H_2 molecule is composed by two protons and 2 electrons. At equilibrium, the protons distance is $R = 1.398a_0$ and for this configuration the ground state energy found is $E_0^{ex} = -1.161$ ([2]).
Then, for this simulations the two protons positions are: $P_1 = (0, 0, 0.699)$, $P_2 = (0, 0, -0.699)$. The results obtained are:

$$E_0 = -1.164 \ \ \sigma = 0.066 \tag{47}$$

Self-Evaluation

Self-Evaluation

Rating of the program

Correctness

As already discussed above, the results provided by this Quantum Monte Carlo implementations were compared to some exact, if available, or numerical ones to check their correctness.
In all cases presented, our DMC algorithm estimations are in good agreement with the expected ones.
It is also possible to improve the precision by increasing the initial number of replicas N_0, or the number of time steps τ, or increasing the time step size $\delta\tau$.
Regarding this last parameters choice, one has to keep in mind two factors: if a too large value leads to systematic errors (see approximation (13)), on the other hand a too small value makes time for convergence larger and the diffusion process less random (see Eq. 22).
The set of parameters chosen in this work is a good compromise between precision and computing time.

Numerical Stability

Although some precautions are taken, the algorithm itself is not very numerically stable; in fact, important fluctuation in the reference energy are still present even after the stationarity is ”reached”, as shown in Figure 4 for the Morse oscillator.
Even tough these fluctuation are simmetrical with respect to their mean (E_0), the reference energy is expected to converge exactly to the ground state energy. This convergence is observed only for its mean value (see, for example Fig. 2). This can be improved implementing an importance sampling.

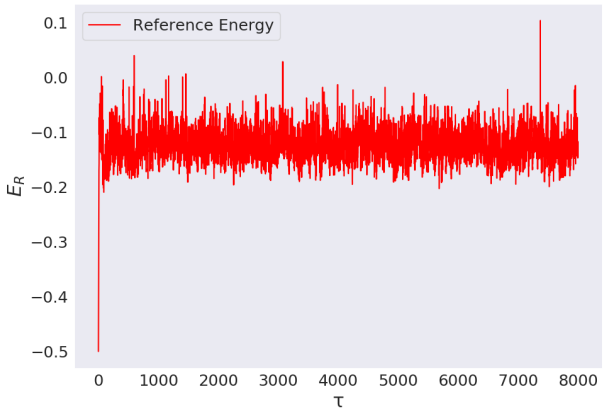

Figure 4: Example of trace of the Reference Energy (Morse Oscillator). The fluctuation are caused by the birth-death processes taking place even when the replicas are distributed according to the ground state wavefunction.

Flexibility

The method can be used to compute the ground state of different quantum systems, as it is sufficient to change the potential, but it is necessary to remember to convert it in adimensional units. The only limitation is that the wave function has to be always greater or equal to zero, since it is seen as a probability density function: this is the major limitation of the method.

Efficiency

Altough the code execution is pretty fast, there might be some performance improvement to implement: for example, there are many loop involved; the most heavy and avoidable is the one over the replicas, but for surpass this limitation a whole restructuration of the code is required.

Self-evaluation

The main goals of this work are achieved: implementing a Quantum Monte Carlo algorithm to compute the ground state of quantum systems. Even though I've been working with easy systems (up to 2 electrons), it might be possible to use the algorithm for larger system, even though spin interactions may arise, giving the need for approximations or theoretical backgrounds I'm not equipped with.
However, I've learned another way to compute ground states with a relatively efficient and straight-forward algorithmic implementation.

Future Developments

The numerical instability described above, as already mentioned, is due to fluctuations in the number of replicas, which is caused by the fact that we sample new positions even where the wave function is small. This inconvenience can be overtaken by implementing the importance sampling, which practically means changing the probability distribution of the replica in such a way the replicas spend more time on regions where the wave function is expected to be large.
Then, it is possible to use this algorithm to compute the proton separation at equilibrium for systems like $H_2{}^+$ ion or H_2 molecule, by running the simulation for different distances among protons: the minimum energy will correspond to the equilibrium separation.

Appendix

Other Results

When computing the ground state of ions or molecules, the spatial distribution of replicas is plotted, together with electrons radii distribution. Examples for H_2 and $H_3{}^+$ are shown in Fig. 6,5.

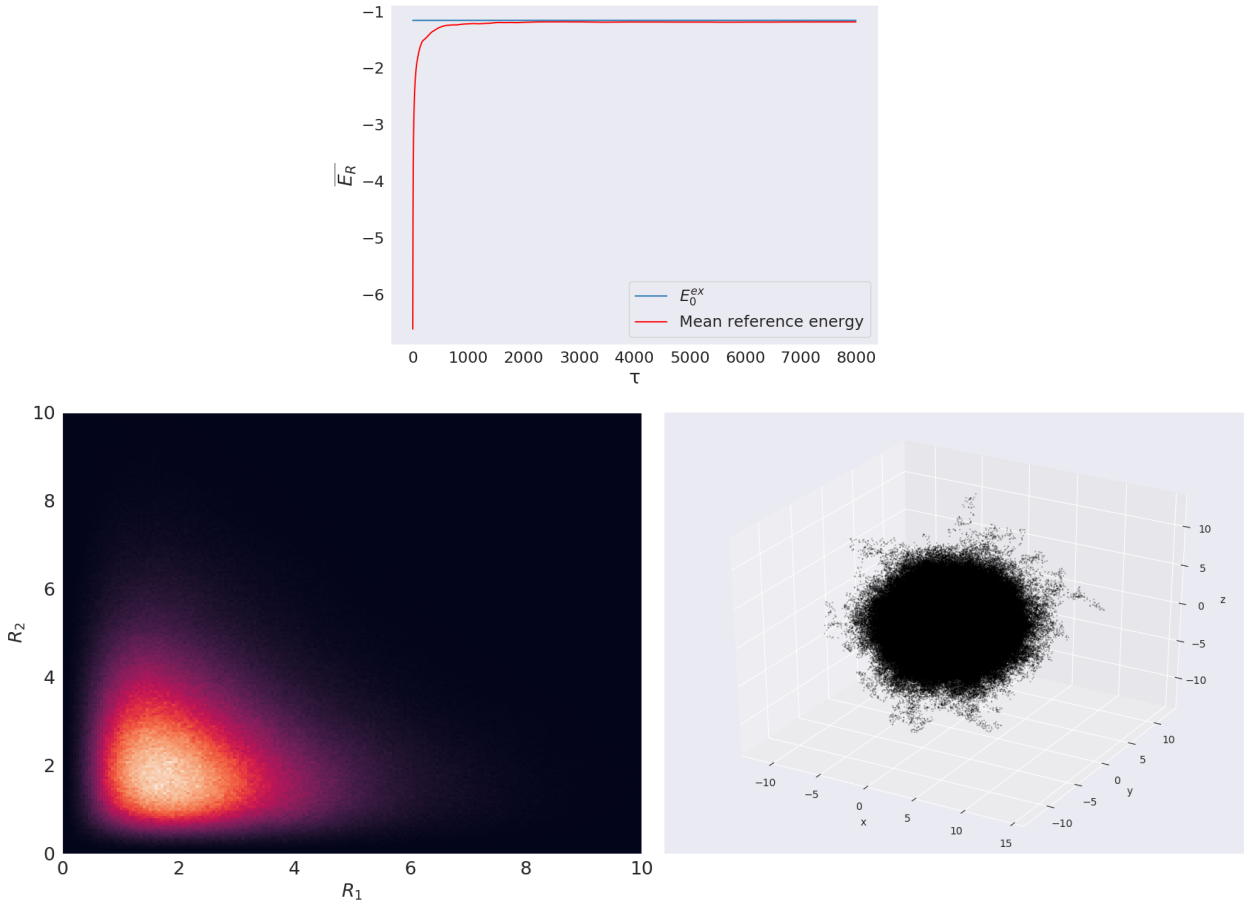

Figure 5: Simulation results for the H_2 molecule.(***Top***) Reference energy E_R as function of the imaginary time τ.(***Bottom Left***) 2D Density plot of the two electrons radii (***Bottom Right***) Distribution of replicas; the cloud is centered around the center of the distance vector between the protons.

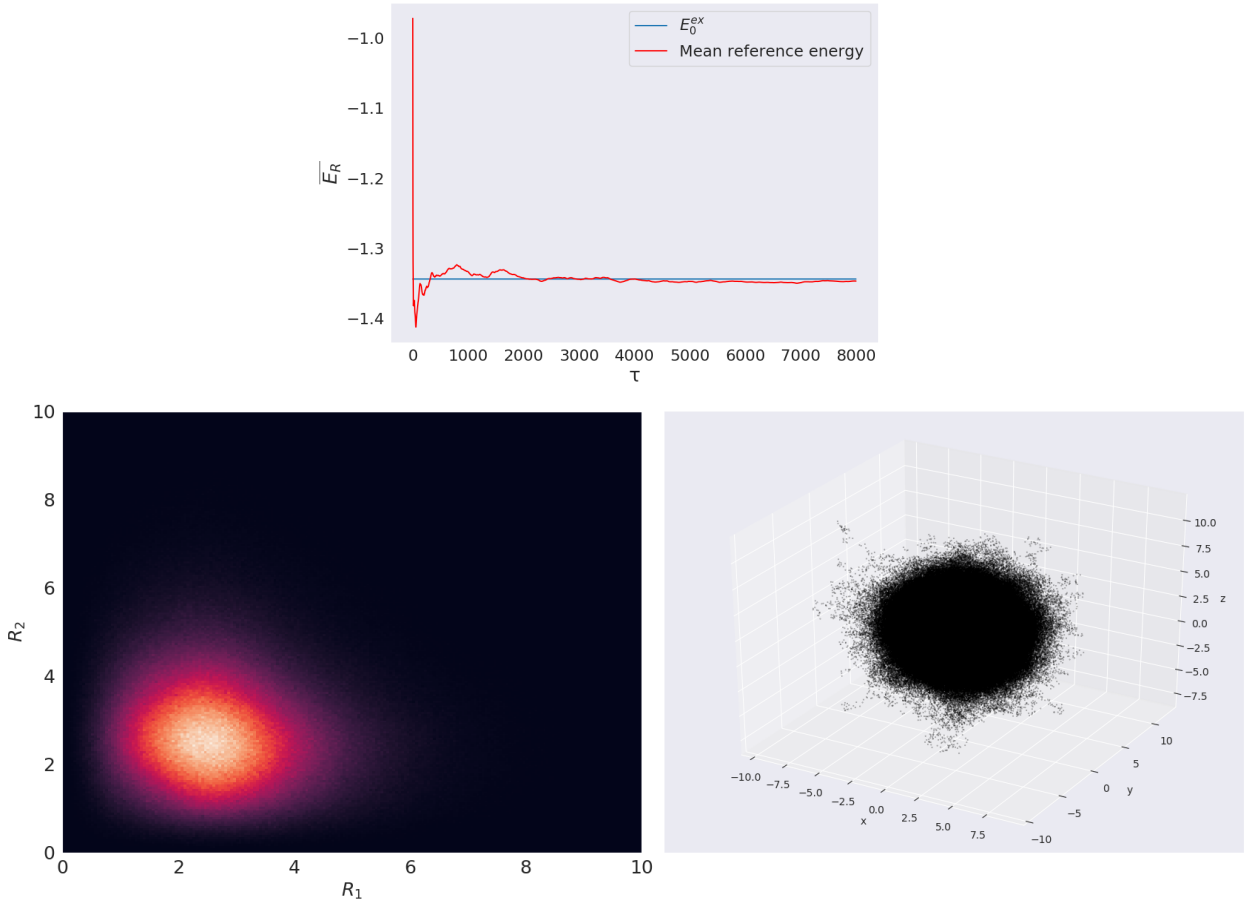

Figure 6: Simulation results for the $H_3{}^+$ ion.(***Top***) Reference energy E_R as function of the imaginary time τ.(***Bottom Left***) 2D Density plot of the two electrons radii (***Bottom Right***) Distribution of replicas; the cloud is centered around the center of the triangle whose vertices are the locations of protons.

Documentation

```
!###### DOCUMENTATION ######
!
! #REQUIREMENTS
!   To execute this program no particular packages or lybraries are needed.
!
! #EXECUTION
!   gfortran FinalProject-Signor-CODE.f90
!
! #Work report available https://www.overleaf.com/read/kxkysdhbwnjr
!
! ####MODULE DIFFUSION MONTE CARLO####
!   Module containing subroutines and functions useful to execute a
!   diffusion monte carlo simulation to compute the ground state of
!   many-body quantum systems, based on
!   "Introduction to the Diffusion Monte Carlo Method", Kosztin et al.
!   (https://arxiv.org/abs/physics/9702023).
!   It contains also some potentials used to test the code:
!   1d harmonic oscillator,1d Morse oscillator, Hydrogen atom, H- ion,
!   H2+ ion, H3+ ion, H2 molecule
!
!   #SUBROUTINE RANDOM_NORMAL
!       SCOPE
!           Generate random normal numbers from a uniform distribution
!           using gasdev. See
    http://gi.ics.nara-wu.ac.jp/~takasu/lecture/old/archives/L2_random_variables-2.pdf
```

```
!       ARGUMENTS
!           u: a double precision real number
!       OUTPUT
!           assigns to the input variable a gaussian random number
!           with mean 0 and variance 1
!
!   #FUNCTION POTENTIAL
!       SCOPE
!           Compute the potential energy (in dimensionless units) of some test systems
!       ARGUMENTS
!           position: DOUBLE PRECISION(:). Array of coordinates
!           potential choice: INTEGER. Number of the potential we want to study
!               1:1D Harmonic Oscillator
!               2: Morse potential
!               3: Hidrogen atom
!               4: H- ion
!               5: H2+ ion
!               6: H3+ ion
!               7: H2 molecule
!       OUTPUT
!           POTENTIAL: REAL*8. potential in dimensionless units
!
!   #SUBROUTINE BRANCH
!       SCOPE
!           Perform the branching process required by the DMC method
!
!       ARGUMENTS
!           m: INTEGER. computed within the DMC SUBROUTINE. given by m=min{int[W(x) +u],3} is the
    number of
!                     copies of the replica to keep. m=0,1,2.
!           branched_psi: DOUBLE PRECISION(:,:) first column: existence flag
!                                               second column: spatial distribution of replicas
!           j: INTEGER. replica index
!
!       OUTPUT
!           branched_psi with new born or killed replicas
!
!   #SUBROUTINE DMC
!       SCOPE
!           It is the where the DMC algorithm is performed.
!           1)initialize the matrix (psi) containing in the first column the
!             existence flag and in the others the spatial coordinates
!           2)loop over time steps to get tau->inf
!           3)compute the potential calling the FUNCTION POTENTIAL
!           4)compute the weight
!           5)call the branch subroutine for every replica
!           6)after tau iterations(stationarity reached) write on a file
!             the coordinates for every alive replica
!           7)update the reference energy
!           8) if d=1 computes directly the spatial distribution of replicas, i.e. the wavefunction
!              if d>1 computes the radial wavefunction

!       ARGUMENTS
!           NO:INTEGER. Initial number of alive replicas
!           Nmax: INTEGER. Maximum number of replicas; i.e. the length of
!                          vector psi
!           tau: INTEGER. Number of time steps
!           delta_tau: REAL*8. Size of the time steps
!           D: INTEGER. Dimension of the system (e.g. for the H atom D=3)
!           initial_position: REAL*8. initial position of every replica
!           E_R: REAL*8. initial reference energy. Suggested to set it as the value
!                        of the potential energy at the initial position
!           xmin: INTEGER. Lower limit for coordinates for the spatial distribution histogramming
!           xmax: INTEGER. upper limit for coordinates for the spatial distribution histogramming
!           nb: INTEGER. Number of bins for sorting replicas
!           potential_choice: INTEGER. Number of potential chosen. See above
!
```

```
!      OUTPUT
!          Different outputs are written on different files
!              'N.txt' number of alive replicas after every time iteration
!              'ref_energy.txt' Reference energy after every time iteration
!              -'wavefunction.txt' the wavefunction of the ground state if d=1,
!                                  if d>1 the radial wavefunction
!
!
!
!   ####PROGRAM QUANTUM_MONTE_CARLO####
!
!   MAIN to run the diffusion monte carlo.
!   Here the inputs for the different DMC subroutines and function are set.
!   The potential is chosen by terminal by the user, and according to this choice,
!   the D, initial_position and E_R are computed authomatically
!   Furthermore the potential choice is written on "potential_choice.txt' and
!   tau on 'time_steps.txt'.
!   All the resulting files are useful to easily compute the ground state, being:
!   E_0 the mean of the reference energy after tau time iterations, its error
!   the standard deviation;
!   the groundstate wavefunction the spatial distribution of alive replicas
!
!
!
!###AUTHOR###
!  Theosamuele Signor, Unipd
!  theosamuele.signor@studenti.unipd.it

!#########################################################################
```

Acknowledgments

This work is based on the paper by Kosztin,Faber and Schulten [3]

References

[1] ANDERSON, J. B. A random-walk simulation of the schrödinger equation: $H_3{}^+$. *The Journal of Chemical Physics 63*, 4 (1975), 1499–1503.

[2] HERZBERG, G. Molecular spectra and molecular structure. *Van Nostrand, New York* (1950).

[3] KOSZTIN, I., FABER, B., AND SCHULTEN, K. Introduction to the diffusion monte carlo method. *American Journal of Physics 64*, 5 (May 1996), 633–644.

[4] TAKASU, F. Generating random numbers.