QUACCS 2014: Quantum Monte Carlo for Electrons

Pierre-François Loos
Research School of Chemistry, Australian National University, Canberra, ACT 0200, Australia
`pf.loos@anu.edu.au`

Abstract

The purpose of this document is to give a general presentation of quantum Monte Carlo (QMC) methods for electronic systems. In particular, we give an overview of variational and diffusion Monte Carlo methods and we provide simple examples to illustrate these methods. For sake of general accessibility, there is absolutely no mathematical rigor in this document! We use atomic units throughout.

Contents

1 Monte Carlo method: a classic example!

Roughly speaking, Monte Carlo is a **numerical integration method**, or at least this is why it is used in quantum chemistry! It was named by Stanislaw Ulam, who in 1946 became the first mathematician to dignify this approach with a name, in honor of his uncle having a little issue with gambling [1]. Nicolas Metropolis also made important contributions to the development of such methods [2]. It is usually used in problems where it is too difficult or impossible to obtain analytical expressions or the dimensionality of the integral is large. The method consists in repeating random sampling many times to obtain numerical results: this is a non-deterministic method.

1.1 Calculating π

The calculation of π with Monte Carlo is probably one of the most didactic example. It can be done by throwing beans on a piece of paper,[1] but we can also use a computer which is more fun!

The idea is to draw a circle inscribed in a unit square. Given that the area of the circle and the square have a ratio of π, we can obtain a numerical estimate using Monte Carlo.

A possible `fortran77` code to calculate π with Monte Carlo could be:

```fortran
      program pi
c     Declare variables
      implicit none
      integer seed,i,n
      double precision one,four,x,y,r,res
c     Input data
      one=1.0d0
      four=4.0d0
      seed=1234
      res=0.0d0
      n=100000
c     Initialize random number generator
      call srand(seed)
c     Start loop
      do i=1,n
c       Get two random numbers
        x=rand(0)
        y=rand(0)
c       Calculate the radius
        r=x**2+y**2
c       Increment result if in circle
        if(r.le.1) res=res+one
      enddo
c     Print the result
      print*,'MC estimate of pi = ',four*res/dble(n)
      print*,'Exact value of pi = ',four*datan(one)
c     End of the program
      return
      end
```

To compile this code (using `gfortran` for example) and run it, just do:

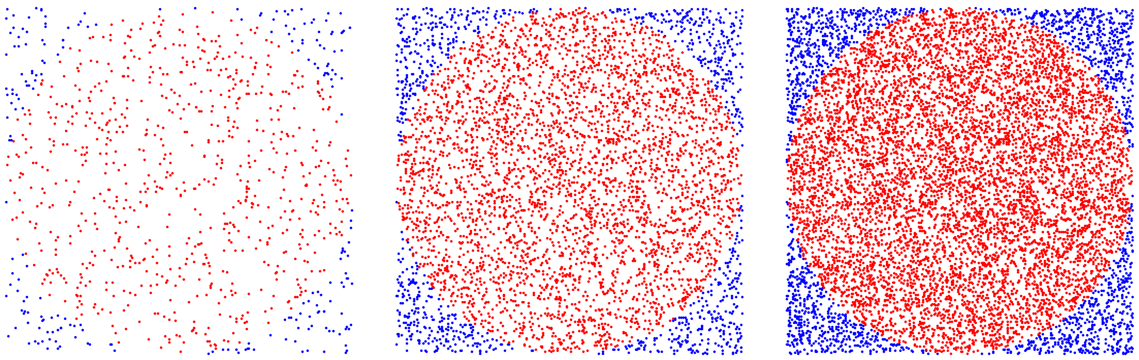
```
prompt$ gfortran pi.f -o pi
prompt$ ./pi
 MC estimate of pi =    3.1416400000000002
 Exact value of pi =    3.1415926535897931
```

Few remarks (see Fig. 1):

[1] At least if you're able to scatter them uniformly.

- The random numbers have to be really *random*![2]
- As you can see, one needs a *large* number of points N. Monte Carlo is known to converge as $1/\sqrt{N}$ which is quite slow sadly.[3]
- The result will improve when N increases (trust me!).
- A Monte Carlo result should never be reported without **error bars**!

Extra question #1: show the $1/\sqrt{N}$ convergence with the `pi.f` program



(a) $\pi \approx \frac{774}{1000} = 3.096$ (b) $\pi \approx \frac{3962}{5000} = 3.1696$ (c) $\pi \approx \frac{7948}{10000} = 3.1792$

Figure 1: Monte Carlo computation of π.

1.2 Average, variance and error

Suppose we want to determine the following quantity

$$I = \int_\Omega f(\boldsymbol{x}) d\boldsymbol{x}, \tag{1}$$

where Ω is a subset of $\mathbb{R}^n$, $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ and f is a function behaving well enough to be integrated in Ω. Let us generate N points $(\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \dots, \bar{\boldsymbol{x}}_N)$ uniformly sampled in Ω. Monte Carlo tells us that the estimate of the integral is given by the **average** (or **mean**):

$$I \approx \langle f \rangle = \frac{1}{N} \sum_{i=1}^{N} f(\bar{\boldsymbol{x}}_i). \tag{2}$$

The square of the error (or **variance**) associated with this average is

$$\sigma^2 = \frac{1}{N(N-1)} \sum_{i=1}^{N} \left[f(\bar{\boldsymbol{x}}_i) - \langle f \rangle \right]^2. \tag{3}$$

How can we figure out the physical meaning of these quantities?

[2] We recommend using `ranlux`.

[3] However, this result does not depend on the number of dimensions of the integral, which is the promised advantage of Monte Carlo integration against most deterministic methods that depend exponentially on the dimension. Moreover, Monte Carlo algorithms can be easily parallelized!

1.2.1 The normal distribution

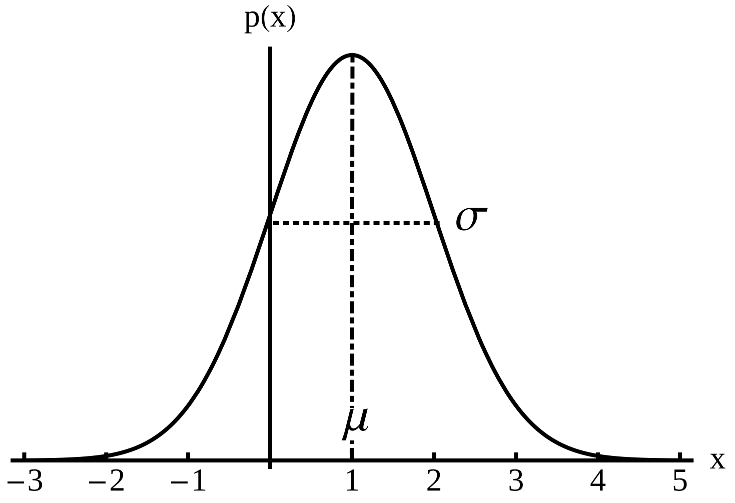


Figure 2: Normal distribution $p(x)$ with $\sigma = \mu = 1$.

Let us consider a particular kind of probability distribution called the normal distribution[4]

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{4}$$

which is properly normalized, i.e.

$$\int_{-\infty}^{\infty} p(x)\, dx = 1. \tag{5}$$

This function gives you, in most of the cases, the distribution of your results. The average/mean of this distribution (i.e. the most probable result you can get)

$$\mu = \int_{-\infty}^{\infty} x\, p(x)\, dx, \tag{6}$$

and the variance is

$$\sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\, dx. \tag{7}$$

In Figure 2, we have represented $p(x)$ for $\sigma = \mu = 1$. The mean tells you where is centered the curve and the variance gives you an idea of how spread the distribution is (i.e. the length of the tail). If you get a mean energy equal to μ with a **standard error**[5] of $\pm\sigma$, we have

$$\int_{\mu-\sigma}^{\mu+\sigma} p(x) = \text{erf}\left(\frac{1}{\sqrt{2}}\right) \approx 68\%, \tag{8}$$

which means that you still have 32% of chance that the *correct* result is outside the range $\mu \pm \sigma$.[6] This shows that, although the error bar gives you an idea of the error you are doing in your calculation, this is more than likely that your result is an **outlier**!

[4]... also called the drunkard's walk for obvious reasons. Peter Gill will probably give you an experimental demonstration during this week.

[5]Standard means that it is based on the normal distribution.

[6]This nasty guy is called an outlier and it happens all the time!

Take-home message #1: be careful with errors!

2 Variational Monte Carlo

2.1 Theory

Within quantum chemistry, variational Monte Carlo (VMC) is used to obtain expectation values (mainly energies[7]). Using more confusing words, we could say that, in VMC, the expectation value of the Hamiltonian with respect to a trial wave function is obtained using a stochastic integration technique.

2.1.1 Basics

Suppose we want to know the electronic energy of a system for which we know a **trial wave function** $\Psi_\mathrm{T}(\boldsymbol{r})$, where $\boldsymbol{r} = \{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_n\}$ are the coordinates of the electrons. The energy is given by

$$E_\mathrm{VMC} = \frac{\int \Psi_\mathrm{T}(\boldsymbol{r})\, \hat{H}\, \Psi_\mathrm{T}(\boldsymbol{r})\, d\boldsymbol{r}}{\int \Psi_\mathrm{T}(\boldsymbol{r})^2\, d\boldsymbol{r}}, \tag{9}$$

where $\hat{H}$ is the Hamiltonian of the system. Using a very involved mathematical trick, Eq. (9) can be recast

$$E_\mathrm{VMC} = \frac{\int \frac{\hat{H}\, \Psi_\mathrm{T}(\boldsymbol{r})}{\Psi_\mathrm{T}(\boldsymbol{r})} \Psi_\mathrm{T}(\boldsymbol{r})^2\, d\boldsymbol{r}}{\int \Psi_\mathrm{T}(\boldsymbol{r})^2\, d\boldsymbol{r}}, \tag{10}$$

where

$$E_\mathrm{L} = \frac{\hat{H}\, \Psi_\mathrm{T}(\boldsymbol{r})}{\Psi_\mathrm{T}(\boldsymbol{r})} \tag{11}$$

is the **local energy**. Equation (10) means that, if you are samplig the local energy (11) with the probablity distribution Ψ_T^2, you will get the variational energy of the wave function Ψ_T. In other words, the method is only as good as the variational trial wave function itself! Note that **the VMC energy is an upper bound to the exact ground state energy**: unless you are using a wave function which does not satisfy the right boundary conditions (and it's usually a bad idea), the energy you will obtain is always higher than the exact energy.

Take-home message #2: VMC gives the variational energy of Ψ_T, no more, no less!

2.1.2 How to optimize variational parameters in VMC?

Usually the trial wave function is of the form $\Psi_\mathrm{T}(\boldsymbol{r}, \boldsymbol{c})$, where $\boldsymbol{c} = (c_1, c_2, \ldots, c_M)$ are variational parameters.[8] Pratically, optimizing the variational parameters of the trial wave function is by far the trickiest part of the calculation and one has to be (most of the time) very patient. People are usually optimizing these coefficients by energy minimization or variance minimization [3, 4, 5].

[7] The energy corresponds to the expectation value of the Hamiltonian.

[8] The form of Ψ_T depends on the problem to be solved and $\boldsymbol{c}$ can be a mixture of linear or non-linear parameters.

2.2 Example

2.2.1 Helium atom

Let us consider the following simple trial wave function for the helium atom as an example:

$$\Psi_\mathrm{T}(r_1,r_2)=\exp\left[-2(r_1+r_2)\right]. \tag{12}$$

In terms of r_1, r_2 and $r_{12}=|\boldsymbol{r}_1-\boldsymbol{r}_2|$, the Hamiltonian of the He atom is

$$\hat{H}=-\frac{1}{2}\Big[\frac{\partial^2}{\partial r_1^2}+\frac{2}{r_1}\frac{\partial}{\partial r_1}+\frac{\partial^2}{\partial r_2^2}+\frac{2}{r_2}\frac{\partial}{\partial r_2}+2\frac{\partial^2}{\partial r_{12}^2}+\frac{4}{r_{12}}\frac{\partial}{\partial r_{12}}+\frac{r_1^2+r_{12}^2-r_2^2}{r_1 r_{12}}\frac{\partial^2}{\partial r_1 r_{12}} \\ +\frac{r_2^2+r_{12}^2-r_1^2}{r_2 r_{12}}\frac{\partial^2}{\partial r_2 r_{12}}\Big]-\frac{2}{r_1}-\frac{2}{r_2}+\frac{1}{r_{12}}. \tag{13}$$

The energy of the trial wave function (12) can be easily obtained in closed form $E=-11/4$, and we would like to reproduce this result using VMC. The local energy for this wave function is

$$E_\mathrm{L}=\frac{\hat{H}\Psi_\mathrm{T}}{\Psi_\mathrm{T}}=\frac{1}{r_{12}}-4. \tag{14}$$

2.2.2 VMC `fortran77` code

```
      program HeVMC
c     Declare variables
      implicit none
      integer seed,ie,ne,it,nEq,nAc
c     Number of electrons
      parameter(ne=2)
      double precision energy,error,elocal,dtvmc,psi,psip,w,r12,weight,
     $  accept,zero,pt5,one,two,four,x,y,z,xp,yp,zp,r,rp
      dimension x(ne),y(ne),z(ne),xp(ne),yp(ne),zp(ne),
     $  r(ne),rp(ne)
      save zero,pt5,one,two,four
      data zero,pt5,one,two,four
     $ /0.0d0,0.5d0,1.0d0,2.0d0,4.0d0/
666   format(' Acceptance ratio                  = ',F15.10,'  %',/
     $       ' Total energy                      = ',F15.10,' AU',/
     $       ' Standard error                  +/-',F15.10,' AU',/
     $       '                                                        ',/
     $       ' Exact energy                      = ',F15.10,' AU')
c     Input data
      seed=12345
      nEq=100000
      nAc=100000
      accept=zero
      energy=zero
      error=zero
      weight=zero
      dtvmc=0.4d0
c     Initialize random number generator
      call srand(seed)
c     Generate initial position of electrons
      do ie=1,ne
        x(ie)=two*rand(0)-one
        y(ie)=two*rand(0)-one
        z(ie)=two*rand(0)-one
      enddo
c     Compute r1 and r2
      do ie=1,ne
        r(ie)=x(ie)**2+y(ie)**2+z(ie)**2
        r(ie)=dsqrt(r(ie))
      enddo
```

```fortran
c      Calculate the wave function
       psi=dexp(-two*(r(1)+r(2)))
c      Start equilibration/accumulation
       do it=1,nEq+nAc
c        Propose a move
         do ie=1,ne
           xp(ie)=x(ie)+dtvmc*(two*rand(0)-one)
           yp(ie)=y(ie)+dtvmc*(two*rand(0)-one)
           zp(ie)=z(ie)+dtvmc*(two*rand(0)-one)
         enddo
c        Compute new values of r1 and r2
         do ie=1,ne
           rp(ie)=xp(ie)**2+yp(ie)**2+zp(ie)**2
           rp(ie)=dsqrt(rp(ie))
         enddo
c        Calculate the new wave function
         psip=dexp(-two*(rp(1)+rp(2)))
c        Compute the ratio w
         w=(psip/psi)**2
c        Calculate the energy if accumulation phase
         if(it.gt.nEq)then
           r12=(xp(1)-xp(2))**2+(yp(1)-yp(2))**2+(zp(1)-zp(2))**2
           r12=dsqrt(r12)
c          Calculate local energy
           elocal=one/r12-four
c          Accumulate energy
           energy=energy+w*elocal
           error=error+(w*elocal)**2
           weight=weight+w
         endif
c        Accept or reject the move
         if(rand(0).le.min(w,one))then
           accept=accept+one
           do ie=1,ne
             x(ie)=xp(ie)
             y(ie)=yp(ie)
             z(ie)=zp(ie)
             psi=psip
           enddo
         endif
c      End of equilibration/accumulation
       enddo
c      Print various results
       accept=accept/dble(nEq+nAc)*100.d0
       error=error-energy**2/dble(nAc)
       energy=energy/dble(nAc)
       error=dsqrt(error/(dble(nAc-1)*dble(nAc)))
       write(*,666)accept,energy,error,-11.d0/4.0d0
c      End of the program
       return
       end
```

Extra question #2: find the energy of $\Psi_T(r_1,r_2) = (1+r_{12}/2)\exp[-2(r_1+r_2)]$

2.2.3 Comments

- A rough estimate of the number of iterations during equilibration can be obtained using the formula

$$N_{\text{eq}} = \frac{\lambda^2}{D\,\Delta\tau_{\text{VMC}}\,A}, \tag{15}$$

where λ is the largest physically relevant length-scale of the system, D is the dimensionality, $\Delta\tau_{\text{VMC}}$ is the VMC **"time-step"** and A is the VMC **acceptance** [6]. The number of equilibration moves should be substantially larger than the above estimate of the correlation period.

- In VMC, the time-step $\Delta\tau_{\mathrm{VMC}}$ is usually chosen to have an acceptance around 50% i.e. $A = 1/2$ [6].
- It is usually a good idea to use more than one configuration during a VMC run.

3 Diffusion Monte Carlo

3.1 Theory

Diffusion Monte Carlo (DMC) is a stochastic projector technique for solving the many-body Schrödinger equation [7, 8, 9]. This is an **exact** method, within statistical errors.

3.1.1 Why is it called *diffusion* Monte Carlo?

The starting point here is the many-body time-dependent Schrödinger equation written in imaginary time:

$$\frac{\partial \Phi(\boldsymbol{r},\tau)}{\partial \tau} = (\hat{H} - S)\Phi(\boldsymbol{r},\tau). \tag{16}$$

For $\tau \to \infty$, the steady-state solution of Eq. (16) for S close to the ground state energy is the exact ground state wave function $\Phi(\boldsymbol{r})$ [10]. DMC generates **configurations** (or **walkers**) distributed according to the product of the trial wave function $\Psi_{\mathrm{T}}(\boldsymbol{r})$ and the exact ground state wave function $\Phi(\boldsymbol{r},\tau)$:

$$\rho(\boldsymbol{r},\tau) = \Psi_{\mathrm{T}}(\boldsymbol{r})\,\Phi(\boldsymbol{r},\tau). \tag{17}$$

Using (17), Eq. (16) can be splitted into three contributions

$$\frac{\partial \rho(\boldsymbol{r},\tau)}{d\tau} = \underbrace{\frac{1}{2}\nabla^2\rho(\boldsymbol{r},\tau)}_{\textbf{diffusion}} + \underbrace{\nabla\cdot[F(\boldsymbol{r})\rho(\boldsymbol{r},\tau)]}_{\textbf{drift}} - \underbrace{[E_{\mathrm{L}}(\boldsymbol{r}) - E_{\mathrm{T}}]\,\rho(\boldsymbol{r},\tau)}_{\textbf{branching}}, \tag{18}$$

where

$$F(\boldsymbol{r}) = \frac{\nabla\Psi_{\mathrm{T}}(\boldsymbol{r})}{\Psi_{\mathrm{T}}(\boldsymbol{r})} \tag{19}$$

is the **quantum force**.

When the trial wave function has the correct **nodes**, the DMC method yields the exact energy with only a statistical error that can be made arbitrarily small by increasing the number of Monte Carlo steps. Thus, the quality of the trial wave function is paramount to achieve high accuracy [11, 12]. This can be easily understood knowing that the statistical error decreases as $C/\sqrt{N}$, where C is a constant and N is the number of iterations. We can't really do anything about the $N^{-1/2}$ behavior but we can try to reduce C by improving Ψ_{T}. However, when Ψ_{T} gets more complicated, the calculation becomes more expensive and the parameters get harder to optimize.

3.1.2 DMC algorithm

Here, we present a simplified version of the DMC algorithm. See Refs. [9, 11] for an efficient and elegant implementation of DMC.

1. Generate initial configuration of walkers (usually using VMC).

2. Do **diffusion** and **drift**: move each walker $\boldsymbol{r}_i' \leftarrow \boldsymbol{r}_i + \chi + F(\boldsymbol{r}_i)\Delta\tau_{\text{DMC}}$, where $\Delta\tau_{\text{DMC}}$ is the DMC **time-step** and χ is a Gaussian random number of mean zero and variance $\Delta\tau_{\text{DMC}}$.[9]

3. Do **branching**: create $\bar{w}_i = \text{int}(w_i + \eta)$ copies of walker i, where

$$w_i = \exp\left[-\Delta\tau_{\text{DMC}}\left(\frac{E_{\text{L}}(\boldsymbol{r}) + E_{\text{L}}(\boldsymbol{r}')}{2} - S\right)\right] \tag{20}$$

is the **weight** of walker i and η is a random number between 0 and 1.

4. Accumulate samples for calculation of averages: $E \leftarrow E + w_i E_{\text{L}}(\boldsymbol{r}')$

5. Repeat steps 2. to 4. until the statistical error is sufficiently small and update sometimes S to make sure that the branching process doesn't go pear-shaped.

Few remarks:

- The exact energy is obtained only for $\Delta\tau_{\text{DMC}} = 0$ (**time-step error**). The limit is usually obtained by linear extrapolation using small values of $\Delta\tau_{\text{DMC}}$.
- Ouch! The statistical error increases when $\Delta\tau_{\text{DMC}}$ decreases.
- The number of iterations during equilibration can be estimated using (15) with $\Delta\tau_{\text{DMC}}$ and $A = 1$.

3.1.3 Fixed-node approximation

The major problem is that the previous algorithm doesn't work for fermions (hence electrons). Indeed, it always converges to the lowest-energy state which is usually the nodeless bosonic ground state (unless you're very lucky!). People call that the sign problem [13, 14, 15]. In terms of the DMC algorithm this means that a **fixed-node approximation** should be used; that is, the fermionic ground-state wave function is expressed as some non-negative function multiplied by a function that has a certain nodal surface [16].[10] In other words, each walker stays in its **nodal pocket** [17, 18] (see Fig. 3). If a walker crosses a nodal surface during the drift and diffusion processes, it is killed.

What is a **node**?
nodes = points in space $\boldsymbol{n}$ for which $\Psi(\boldsymbol{n}) = 0$.

What is a **nodal pocket**?
nodal pocket = region of space in which electrons can travel *without* crossing a node.

The DMC method finds the best energy for a given choice of nodal surface, providing an upper bound for the ground-state energy. The exact ground state energy is reached if and only if the nodal surface is exact. The energy difference between the fixed-node approximation and the exact value is called the **fixed-node error** and nobody knows how to quantity it accurately [19].

[9] In `fortran77`, this can be computed as `chi = dsqrt(-two*dtau*dlog(rand(0)))*dcos(pi*rand(0))`
[10] The nodes of the trial wave function are usually used.

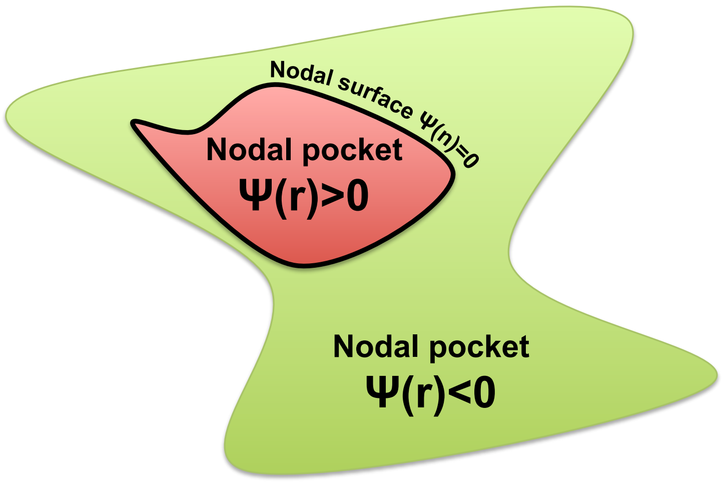


Figure 3: Nodal surfaces and nodal pockets.

Take-home message #3: DMC is exact if and only if the nodes are exact!

3.1.4 DMC recipe

1. Obtain a *good* trial wave function using VMC (see above).
2. Perform calculations for various values of $\Delta\tau_{\text{DMC}}$ (not too small).
3. Extrapolate to $\Delta\tau_{\text{DMC}} = 0$ using linear fit.
4. Published!!

3.2 Example

3.2.1 Helium atom

We propose to use the same example as above and try to improve the variational result ($E_{\text{VMC}} = -11/4$). Don't worry about fixed-node errors, the spatial part of the He ground-state wave function is totally symmetric! For information, the exact energy of He is approximately [20]

$$E_{\text{He}} = 2.903\,724\,377\,034\,119\,598\,311\,159\,245\,194\,404\,446\,696\,905\,37\ldots. \tag{21}$$

3.2.2 DMC fortran77 code

```
      program HeDMC
c     Declare variables
      implicit none
c     Some very smart coding
      bla bla bla
c     End of the program
      return
      end
```

4 QMC Sofwares

For my own research, I use a software called CASINO [21] but there are many other softwares available on the market. CASINO is developed by a group in Cambridge from the Cavendish laboratory. The CASINO user guide contains details about the various methods and algorithms used in QMC calculations. Each year, Mike Towler organizes a very nice conference/workshop in Tuscany. See `http://vallico.net/casinoqmc/` for more details.

References

[1] P. Hoffman. *The Man Who Loved Only Numbers: The Story of Paul Erdős and the Search for Mathematical Truth*, pages 238–239. Hyperion, New York, 1998.

[2] N. Metropolis and S. Ulam. *J. Amer. Stat. Assoc.*, page 335, 1949.

[3] C. J. Umrigar. *Quantum Monte Carlo Methods in Physics and Chemistry*, pages 129–160. NATO Science Series. Kluwer Academic Press, Dordrecht, 1999.

[4] C. J. Umrigar and Claudia Filippi. *Phys. Rev. Lett.*, 94:150201, 2005.

[5] J. Toulouse and C. J. Umrigar. *J. Chem. Phys.*, 126:084102, 2007.

[6] R. M. Lee, G. J. Conduit, N. Nemec, P. Lopez-Rios, and N. D. Drummond. *Phys. Rev. E*, 83:066706, 2011.

[7] M. H. Kalos, D. Levesque, and L. Verlet. *Phys. Rev. A*, 9:2178, 1974.

[8] D. M. Ceperley and M. H. Kalos. *Monte Carlo Methods in Statistical Physics.* Springer Verlag, Berlin, 1979.

[9] P. J. Reynolds, D. M. Ceperley, B. J. Alder, and W. A. Lester, Jr. *J. Chem. Phys.*, 77:5593, 1982.

[10] J. Kolorenc and L. Mitas. *Rep. Prog. Phys.*, 74:026502, 2011.

[11] C. J. Umrigar, M. P. Nightingale, and K. J. Runge. *J. Chem. Phys.*, 99:2865, 1993.

[12] C.-J. Huang, C. J. Umrigar, and M. P. Nightingale. *J. Chem. Phys.*, 107:3007, 1997.

[13] E. Y. Loh, J. E. Gubernatis, R. T. Scalettar, S. R. White, D. J. Scalapino, and R. L. Sugar. *Phys. Rev. B*, 41:9301, 1990.

[14] M. Troyer and U.-J. Wiese. *Phys. Rev. Lett.*, 94:170201, 2005.

[15] C. J. Umrigar, J. Toulouse, C. Filippi, S. Sorella, and R. G. Hennig. *Phys. Rev. Lett.*, 98:110201, 2007.

[16] D. M. Ceperley. *J. Stat. Phys.*, 63:1237, 1991.

[17] L. Mitas. Structure of fermion nodes and nodal cells. *Phys. Rev. Lett.*, 96:240402, 2006.

[18] D. Bressanini. *Phys. Rev. B*, 86:115120, 2012.

[19] S. K. M. Rasch, S. Hu, and L. Mitas. *J. Chem. Phys.*, 140:041102, 2014.

[20] H. Nakashima and H. Nakatsuji. *J. Chem. Phys.*, 127:224104, 2007.

[21] R. J. Needs, M. D. Towler, N. D. Drummond, and P. Lopez Rios. *J. Phys. Condens. Matter*, 22:023201, 2010.