Variational Monte Carlo for approximating ground states of Atoms and Molecules.

A. V. Solbrå
Department of Physics, University of Oslo.

(Dated: May 20, 2013)

An Object Oriented code for performing Variational Monte Carlo (VMC) has been produced for measuring energies for Atoms and Molecules in order to search for the ground state energy and function. I run the code for He-, Be- and Ne-atoms, as well as H_2 and Be_2 molecules, using a simple slater determinant with a Jastrow factor ansatz.

I. INTRODUCTION

Variatinal Monte Carlo is a simple and useful way to approximate the ground state using some guess with a set of free parameters $\boldsymbol{\alpha}$. Since the Hamiltonian is hermitian we can expand our trial function $\psi_T(\mathbf{r}, \boldsymbol{\alpha})$ in terms of the normalized eigenstates of the Hamiltonian as $\psi_T = \sum_i c_i\psi_i$, where for a normalized trial wave function, $\sum_i |c_i|^2 = 1$. The expectation value of the energy for the trial wave function is then $\langle\psi_T|\hat{H}|\psi_T\rangle = \sum_i c_i^2 E_i \geq E_0$, where E_0 is the energy of the ground state. We vary the parameters $\boldsymbol{\alpha}$ in order to find the lowest value of expectation value. In general our wave function is not normalized, and so our problem is to find

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \frac{\langle\psi_T(\boldsymbol{\alpha})|\hat{H}|\psi_T(\boldsymbol{\alpha})\rangle}{\langle\psi_T(\boldsymbol{\alpha})|\psi_T(\boldsymbol{\alpha})\rangle}. \quad (1)$$

And we then use $\psi_T(\mathbf{r}, \boldsymbol{\alpha}^*)$ as our approximation of the ground state, and we can use this state to calculate to an approximation to the ground state energy.

II. MONTE CARLO INTEGRATION AND THE METROPOLIS ALGORITHM

We want to calculate the integral

$$\langle E(\boldsymbol{\alpha})\rangle = \frac{\int \psi_T^*(\mathbf{r}, \boldsymbol{\alpha})\hat{H}(\mathbf{r})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}}{\int \psi_T^*(\mathbf{r}, \boldsymbol{\alpha})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}} \quad (2)$$

The idea of Monte Carlo methods is to use the fact that

$$\int_V f(\mathbf{r})\, d\mathbf{r} = \langle f\rangle V, \quad (3)$$

where V is the domain of integration and $\langle f\rangle$ is the average value of f in the domain. We then approximate the average as

$$\langle f\rangle \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad (4)$$

where the x_is are chosen by some rule.

Importance sampling is a method of choosing samples for the Monte Carlo method in a way such that the samples are as relevant as possible (e.g. in a way such that we avoid picking so many zero values in the above problem). Let's say that we want to find the integral of some function, $\int_a^b F(x)\, dx$ have some probability distribution $p(x)$, so that

$$\int_a^b p(x)\, dx = 1. \quad (5)$$

We can then rewrite our integral as

$$I = \int_a^b F(x)\, dx = \int_a^b p(x)\frac{F(x)}{p(x)}\, dx. \quad (6)$$

Random numbers are generated for the uniform distribution on the unit interval $p(y)$ we need to perform a change of variables $y \to x$ through

$$y(x) = \int_a^x p(x')\, dx', \quad (7)$$

If we can invert this equation, we can find $x(y)$. Now we can write our integral as

$$I = \int_a^b F(x)\, dx = \int_a^b p(x)\frac{F(x)}{p(x)}\, dx = \int_{\tilde{a}}^{\tilde{b}} \frac{F(x(y))}{p(x(y))}\, dy \quad (8)$$

Note in particular the new integration limits. There is no definite way to chose the sampling function, but a nice trick is to choose a distribution which is a factor of the function we want to integrate, as this distribution will often look at least a bit like the integrand.

For expectation values of quantum observables there is a standard way of choosing the distribution. By noting that we can write our expectation value as

$$\langle E(\boldsymbol{\alpha})\rangle = \frac{\int \psi_T^*(\mathbf{r}, \boldsymbol{\alpha})\hat{H}(\mathbf{r}, \boldsymbol{\alpha})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}}{\int \psi_T^*(\mathbf{r}, \boldsymbol{\alpha})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}} \quad (9)$$

$$= \frac{\int \psi_T^*(\mathbf{r}, \boldsymbol{\alpha})\frac{\psi_T(\mathbf{r}, \boldsymbol{\alpha})}{\psi_T(\mathbf{r}, \boldsymbol{\alpha})}\hat{H}(\mathbf{r}, \boldsymbol{\alpha})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}}{\int \psi_T^*(\mathbf{r}, \boldsymbol{\alpha})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}} \quad (10)$$

$$= \frac{\int |\psi_T(\mathbf{r}, \boldsymbol{\alpha})|^2\frac{1}{\psi_T(\mathbf{r}, \boldsymbol{\alpha})}\hat{H}(\mathbf{r}, \boldsymbol{\alpha})\psi_T(\mathbf{r}, \boldsymbol{\alpha})d\mathbf{r}}{\int |\psi_T(\mathbf{r}, \boldsymbol{\alpha})|^2 d\mathbf{r}}. \quad (11)$$

We now see that we can use the probability distribution

$$p(\mathbf{r}, \boldsymbol{\alpha}) = \frac{|\psi_T(\mathbf{r}, \boldsymbol{\alpha})|^2}{\int |\psi_T(\mathbf{r}, \boldsymbol{\alpha})|^2 d\mathbf{r}} \quad (12)$$

If we now define the *local energy* as

$$E_L(\mathbf{r},\boldsymbol{\alpha}) = \frac{1}{\psi_T(\mathbf{r},\boldsymbol{\alpha})}\hat{H}(\mathbf{r},\boldsymbol{\alpha})\psi_T(\mathbf{r},\boldsymbol{\alpha}) \tag{13}$$

Then the expectation value of the energy can be written as

$$\langle E\rangle = \int p(\mathbf{r},\boldsymbol{\alpha})E_L(\mathbf{r},\boldsymbol{\alpha})d\mathbf{r}. \tag{14}$$

This all seems very nice, but we have so far overlooked the part about being able to invert the change of variables function. For our trial function, the probability density is invertible, and we must find some other way of drawing our samples. This is where the metropolis algorithm comes into the picture.

The Metropolis algorithm is easy to state in its simplest form. Suppose we have found a first sample position $\mathbf{R}$. We draw a candidate for a new step as $\mathbf{R}' = \mathbf{R} + \Delta\mathbf{R}$, where $\Delta\mathbf{R}$ has component drawn from a uniform distribution. We must perform two tests:

1. If
$$\frac{p(\mathbf{R}')}{p(\mathbf{R})} > 1 \tag{15}$$
then the new step is accepted.
2. If not, the new step is accepted if
$$r \le \frac{p(\mathbf{R}')}{p(\mathbf{R})} \tag{16}$$
where r is a uniform random variable on the unit interval. A nice advantage of this is that we do not ever need to compute the norm of the function as we are only computing ratios between probabilites. There is no absolute answer for what is the optimal acceptance rate of movements, but a good rule of thumb is to choose the distribution $\Delta\mathbf{R}$ is drawn from such that we get an acceptance rate of 50%.

A. Importance sampling in the Metropolis algorithm

We want to keep the correlation between the Monte Carlo measurement low. An obvious way to do this would be to simply increase the step size, so that the wave function looks entirely different from one step to the next. The problem with this is that the acceptance rate would fall and we would keep selecting the same point over and over, which would give a high variance! This is where the acceptance rate "rule of thumb" of 0.5 comes from, but the optimal interval varies a lot from case to case.

If we proposed our new steps in a better way, we could perhaps still improve the efficency of the step length versus acceptance rate. One way to do this is to modify our symmetrical selection rule such that the samples move towards areas of the state-space where the distribution is large. One such procedure is the Fokker-Planck formalism where the walkers are moved according to the gradient of the distribution. The formalism "pushes" the samples in a "disireable" direction. The idea is to propose moves similarly to an isotropic diffusion process with a drift. A new position r_{new} is calculated from the old one, r_{old} as

$$r_{\text{new}} = r_{\text{old}} + \chi + DF(r_{\text{old}})\delta t. \tag{17}$$

Here, χ is a Gaussion randum number with mean 0 and $\sigma = \sqrt{2D\delta t}$. It accounts for the drift. F is a drift velocity dependent on the position of the samples and is derived from the quantum mechanical wave function ψ. The constant D, being the diffusion constant of χ, also adjusts the size of the drift. δt is a time step parameter whose presence will be investigated.

What is the optimal choice for the drift term? From statistical mechanics we know that a simple isotropic drift diffusion process obeys a Fokker-Planck equation of the form:

$$\frac{\partial f}{\partial t} = \sum_i D\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_i} - F_i(F)\right)f, \tag{18}$$

where f is the continuous distribution of samples. It can be shown that the drift velocity has to be chosen as follows:

$$F = \frac{1}{f}\nabla f, \tag{19}$$

or in our case, where $f = |\psi|^2$,

$$F = 2\frac{1}{\psi}\nabla\psi, \tag{20}$$

From our previous calculations, we note that

$$\frac{\partial}{\partial x_1}\psi = \left(-\frac{\alpha x_1}{r_1} + \frac{x_1 - x_2}{2r_{12}(\beta r_{12}+1)^2}\right)\psi \tag{21}$$

which lets us implement a closed form of the gradient.

With this new step selection rule, according to the Metropolis algorithm the requirement for accepting a step is also modified. The previus one,

$$P(r_{\text{new}}, r_{\text{old}}) = \min(1, q(r_{\text{new}}, r_{\text{old}})) \tag{22}$$

with

$$q(r_{\text{new}}, r_{\text{old}}) = |\psi(r_{\text{new}})|^2/|\psi(r_{\text{old}})|^2|\psi(r_{\text{new}})|^2/|\psi(r_{\text{old}})|^2 \tag{23}$$

is now replaced with

$$q(r_{\text{new}}, r_{\text{old}}) = \frac{G(r_{\text{old}}, r_{\text{new}}, \Delta t)|\psi(r_{\text{new}})|^2}{G(r_{\text{new}}, r_{\text{old}}, \Delta t)|\psi(r_{\text{old}})|^2} \tag{24}$$

where G is the Green's function

$$G(r_{\text{new}}, r_{\text{old}}, \Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left(- (\mathbf{r}_{\text{new}} - \mathbf{r}_{\text{old}} - D\Delta t F(r_{\text{old}}))^2 / 4D\Delta t \right) \tag{25}$$

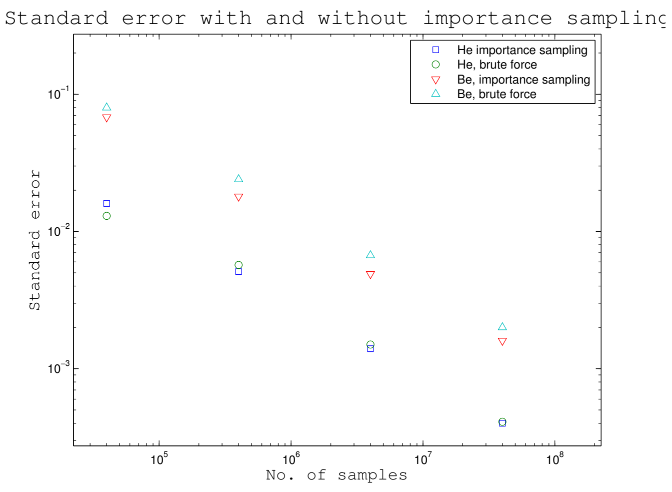


FIG. 1: The figure shows the standard errors for He and Be, with and without importance sampling. The standard errors are calculated as detailed in section V.

This produces the same results as our previous algorihm, however, by choosing a small value for Δt we can get a much higher acceptance rate, which means we will get a smaller error, as shown in figure 1

III. HAMILTONIAN AND ANSATZ FOR THE WAVE FUNCTION

The Helium atom is the simplest possible system besindes the hydrogen atom. If we let $\mathbf{r}_1$ describe the position of the the first electron relative to the nucleus, and $\mathbf{r}_2$ likewise describe the position of the second electron, the in potential energy is from the interactions between the electrons and the nucleus can be described as

$$V_{\text{nuc}}(\mathbf{r}_1, \mathbf{r}_2) = -\frac{Z}{4\pi\epsilon_0}\left(\frac{1}{r_1} + \frac{1}{r_2}\right), \tag{26}$$

where r_1 is the length of $\mathbf{r}_1$ and Z is the number of protons in the nucleus, e.g. 2 for Helium. It is preferrable to work with dimensionless units, so let us from this point on work in distance units of $\hat{r} = 4\pi\epsilon_0 r$. The hats will be cumbersome, so we will simply use r. In dimensionless units, the nucleus potential is written as

$$V_{\text{nuc}}(\mathbf{r}_1, \mathbf{r}_2) = -Z\left(\frac{1}{r_1} + \frac{1}{r_2}\right). \tag{27}$$

We should also include the interactions between the particles, making the full potential

$$V = -\frac{Z}{r_1} - \frac{Z}{r_2} + \frac{1}{r_{12}} \tag{28}$$

yielding the total Hamiltonian

$$H(\mathbf{r}_1, \mathbf{r}_2) = -\frac{\nabla_1^2}{2} - \frac{\nabla_2^2}{2} - \frac{Z}{r_1} - \frac{Z}{r_2} + \frac{1}{r_{12}} \tag{29}$$

For an atom with n particles, the general Hamiltonian is

$$H(\{\mathbf{r}_i\}) = \sum_{i=1}^{n}\left(\frac{-\nabla_i^2}{2} - \frac{Z}{r_i} + \sum_{j=i+1}^{n}\frac{1}{r_{ij}}\right) \tag{30}$$

In this project we will assume the wavefunction to be comprised of hydrogen-like orbitals, as well as a correlation term in the form of a Jastrow factor, i.e. we let our ansatz be

$$\Psi_T(\alpha, \beta, \{\mathbf{r}\}) = \Psi_S(\alpha, \{\mathbf{r}\})\Psi_C(\beta, \{\mathbf{r}\}) \tag{31}$$

where

$$\Psi_S(\alpha, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n) = \begin{vmatrix} \phi_1(\mathbf{r}_1) & \phi_2(\mathbf{r}_1) & \ldots & \phi_4(\mathbf{r}_1) \\ \phi_1(\mathbf{r}_2) & & & \vdots \\ \vdots & & \ddots & \\ \phi_1(\mathbf{r}_n) & \ldots & & \phi_n(\mathbf{r}_n) \end{vmatrix} \tag{32}$$

is the slater-determinant of the lowest orbitals with α as an effective charge, i.e.

$$\phi_{1s}(\mathbf{r}) = \exp(-\alpha r), \tag{33}$$
$$\phi_{2s}(\mathbf{r}) = (1 - \alpha r/2)\exp(-\alpha r/2), \tag{34}$$
$$\phi_{2p}(\mathbf{r}) = \alpha \mathbf{r} \exp(-\alpha r/2) \tag{35}$$

where we use solid harmonics for the different m_l states of the $2p$ orbital. Note that each function is used twice to account for the different spin orientations. The correlation term is

$$\Psi_C(\beta, \{\mathbf{r}_i\}) = \prod_{i<j} \exp\left(\frac{a_{ij} r_{ij}}{1 + \beta r_{ij}}\right), \tag{36}$$

where $a_{ij} = 1/2$ if particles i and j have opposite spin, and $a_{ij} = 1/4$ otherwise.

A. Efficient computation of the wave function properties

One complication is that the Slater Determinant is very expensive to compute for larger systems (the computation of a determinant of an $N \times N$ matrix goes as $O(N^3)$ in the leading term). It is worthwhile to find better methods of computing this determinant.

The first thing we do is to note that, as shown by for example [1], that for the variational energy we can approximate the determinant as

$$\phi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n) \propto \text{Det} \uparrow \text{Det} \downarrow \tag{37}$$

where we put the first half of the electrons in the first determinant, and the other half in the second determinant. This is valid for spin independent hamiltonians. Using this we can write our wave function as $\psi_T(\mathbf{r}) = D_\uparrow D_\downarrow \psi_C$. This is already a significant speedup! However, we will now look into a far more efficient way of calculating ratios of wavefunctions, like those we will need in the metropolis algorithm.

1. *Closed expressions for local energy*

When implementing the closed form expression for the energy, we calculate

$$\frac{1}{\psi_T} \hat{H} \psi_T \tag{38}$$

where the challenge is calculating the $\nabla_1^2 \psi$ term. Let us see if we are able to calculate this term in our more general case. Our wavefunction can be factorized as

$$\psi_T(\{\mathbf{r}_i\}) = \psi_S(\{\mathbf{r}_i\}) \psi_C(\{\mathbf{r}_i\}) \tag{39}$$

where ψ_S is the Slater Determinant, and ψ_C contains the jastrow factor. In this way, we can write the laplacian as

$$\nabla_i^2 \psi_T = \psi_C \nabla_i^2 \psi_S + 2 \nabla_i \psi_S \cdot \nabla_i \psi_S + \psi_S \nabla_i^2 \psi_C^2. \tag{40}$$

The expressions for the gradients and laplacians are given in Appendix B.

B. Hamiltonian and ansatz for molecules

The H_2 molecule consists of two protons and two electrons with a ground state energy. $E = -1.175$ a.u. and the equilibrium distance between the two hydrogen atoms of $r_0 = 1.40$ Bohr radii.

We define our systems using the following variables. Origo is chosen to be halfway between the two protons. The distance from proton 1 is defined as $\mathbf{R}/2$ whereas proton 2 has a distance $-\mathbf{R}/2$. Calculations are performed for fixed distances $\mathbf{R}$ between the two protons.

Electron 1 has a distance r_1 from the chose origo, while electron 2 has a distance r_2. The kinetic energy operator then becomes

$$-\frac{\nabla_1^2}{2} - \frac{\nabla_2^2}{2}. \tag{41}$$

The distance between the two electrons is $r_{12} = |\mathbf{r}_1 - \mathbf{r}_2|$. The repulsion between the two electrons results in a potential energy term given by

$$+\frac{1}{r_{12}}. \tag{42}$$

In a similar way we obtain a repulsive contribution from the interaction between the two protons given by

$$+\frac{1}{|\mathbf{R}|} \tag{43}$$

For general elemental gasses the term becomes $Z^2/|\mathbf{R}|$.

To obtain the final potential energy we need to include the attraction the attraction the electrons feel from the protons. To model this, we need to define the distance between the electrons and the two protons. If we model this along a chosen z-axis with electron 1 placed at a distance $\mathbf{r}_1$ from a chosen origin, one proton at $-\mathbf{R}/2$ and the other at $\mathbf{R}/2$, the distance from proton 1 to electron 1 becomes

$$\mathbf{r}_{1p1} = \mathbf{r}_1 + \mathbf{R}/2 \tag{44}$$

and

$$\mathbf{r}_{1p2} = \mathbf{r}_1 - \mathbf{R}/2 \tag{45}$$

from proton 2.

Similarly, for electron 2 we obtain

$$\mathbf{r}_{2p1} = \mathbf{r}_2 + \mathbf{R}/2 \tag{46}$$

and

$$\mathbf{r}_{2p2} = \mathbf{r}_2 - \mathbf{R}/2 \tag{47}$$

These four distances define the attractive contributions to the potential energy,

$$-\frac{1}{r_{1p1}} - \frac{1}{r_{1p2}} - \frac{1}{r_{2p1}} - \frac{1}{r_{2p2}}.$$

And the complete hamiltonian becomes

$$\hat{H} = -\frac{\nabla_1^2}{2} - \frac{\nabla_2^2}{2} - \frac{1}{r_{1p1}} - \frac{1}{r_{1p2}} - \frac{1}{r_{2p1}} - \frac{1}{r_{2p2}} + \frac{1}{r_{12}} + \frac{1}{|\mathbf{R}|} \tag{48}$$

For H_2, we will use a trial wave function of the form

$$\psi_T(\mathbf{r}_1, \mathbf{r}_2, \mathbf{R}) = \psi(\mathbf{r}_1, \mathbf{R}) \psi(\mathbf{r}_2, \mathbf{R}) \exp\left(\frac{r_{12}}{2(1 + \beta r_{12})}\right). \tag{49}$$

with the following trial wave function

$$\psi(\mathbf{r}_1, \mathbf{R}) = (\exp(-\alpha r_{1p1}) + \exp(-\alpha r_{1p2})), \tag{50}$$

for electron 1, and

$$\psi(\mathbf{r}_2, \mathbf{R}) = (\exp(-\alpha r_{2p1}) + \exp(-\alpha r_{2p2})) \tag{51}$$

for electron 2.

The question arises of how to construct the higher orbitals. One approach that ensures as many electrons in the bottom states as possible, is to let the next wavefunctions be

$$\psi(\mathbf{r}_1, \mathbf{R}) = (\exp(-\alpha r_{1p1}) - \exp(-\alpha r_{1p2})), \tag{52}$$

for electron 1, and

$$\psi(\mathbf{r}_2, \mathbf{R}) = (\exp(-\alpha r_{2p1}) - \exp(-\alpha r_{2p2})) \tag{53}$$

for electron 2. We then use sums and differences of the $|2,0,0\rangle$ states for the four next electrons, and so on. These first 8 wavefunctions allows us to simulate the beryllium molecule as well.

IV. MINIMIZATION ALGORITHM FOR THE ENERGY

Using brute force methods as described earlier is well and good for making energy plots such as in figures 1 and **??**. However, when we want to do a large scale simulation in order to calculate the energy with an as small error as possible, we would rather only use a large number of samples for the minimal energy. We need some method of finding the minimal energy. We will focus on *gradient methods*, iterative methods given by

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + c_k \mathbf{d}_k \tag{54}$$

where $\nabla E(\boldsymbol{\alpha})^T \mathbf{d}_k < 0$, i.e. we move in directions where the gradient is negative. There are many different wany to chose the direction $\mathbf{d}_k$:

- If we choose $\mathbf{d}_k = -\nabla E(\boldsymbol{\alpha}_k)$ we get the *steepest descent method*,

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - c_k \nabla E(\boldsymbol{\alpha}_k) \tag{55}$$

- The steepest descent method can be thought of as a first order approximation to the gradient of E. A second order approximation gives us Newtons method,

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - c_k (\nabla^2 E(\boldsymbol{\alpha}_k))^{-1} \nabla E(\boldsymbol{\alpha}_k) \tag{56}$$

  where ∇^2 here means the hessian matrix, not the laplacian.

- The conjugate gradient method chooses new direction orthogonal to the previous one. It can be sumarized in three steps. First you calculate the residual,

$$\mathbf{r}_k = -\nabla E(\boldsymbol{\alpha}) \tag{57}$$

  then you find the new direction by the gram-schmidt process as

$$\mathbf{d}_k = \mathbf{r}_k - \sum_{i<k} \frac{\mathbf{d}_i^T \nabla^2 E(\boldsymbol{\alpha}) \mathbf{r}_k}{\mathbf{d}_i^T \nabla^2 E(\boldsymbol{\alpha}) \mathbf{d}_i} \mathbf{d}_i \tag{58}$$

  and then finally we find the new values as

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + c_k \mathbf{d}_k \tag{59}$$

- Broyden's method is a quasi-Newton method, that uses updates the position as

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - B_k \nabla E(\boldsymbol{\alpha}_k) \tag{60}$$

  for some matrix B_k. The next matrix is then found as

$$B_{k+1} = B_k + \frac{1}{s_k^T s_k}(y_k - B_k s_k) s_k^T \tag{61}$$

  where $s_k = \boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k$ and $y_k = E(\boldsymbol{\alpha}_{k+1}) - E(\boldsymbol{\alpha}_k)$. This can be thought of as a multi-dimensional secant method, and it can be shown that B_k approximates the hessian well, given that the initial expression is good.

It turns out that all of these methods are quite dissappointing in our case, much because there are large uncertainties in our expressions for the gradient. We will implement an adaptive method, where the step direction is simply along the signs of the gradient, and the step rule is to half the step each time the derivative changes sign, and multiply the step with 1.1 when it does not. Such a minimization search is illustrated in figures 2 and **??**.

A. Expressions for the gradient

One can show the following expressions

$$\frac{\partial}{\partial \alpha_i}\langle E\rangle = 2\left(\left\langle \frac{1}{\psi(\boldsymbol{\alpha})}\frac{\partial \psi(\boldsymbol{\alpha})}{\partial \alpha_i} E_L(\boldsymbol{\alpha})\right\rangle - \langle E_L(\boldsymbol{\alpha})\rangle \left\langle \frac{1}{\psi(\boldsymbol{\alpha})}\frac{\partial \psi(\boldsymbol{\alpha})}{\partial \alpha_i}\right\rangle\right) \tag{62}$$

As for our spesific case,

$$\frac{1}{\psi}\frac{\partial \psi}{\partial \alpha} = \frac{1}{D}\frac{\partial D}{\partial \alpha} = \sum_{i,j} \frac{\partial \phi_{ij}}{\partial \alpha} D_{ji}^{-1} \tag{63}$$

and

$$\frac{1}{\psi}\frac{\partial \psi}{\partial \beta} = \frac{1}{\psi_C}\frac{\partial \psi_C}{\partial \beta} = \sum_{i=1}^{N}\sum_{j=i+1}^{N} a_{ij} r_{ij}^2/(1+\beta r_{ij})^2 \tag{64}$$

These expressions will make the gradient fast and more reliable to calculate, as opposed to numerical derivatives.

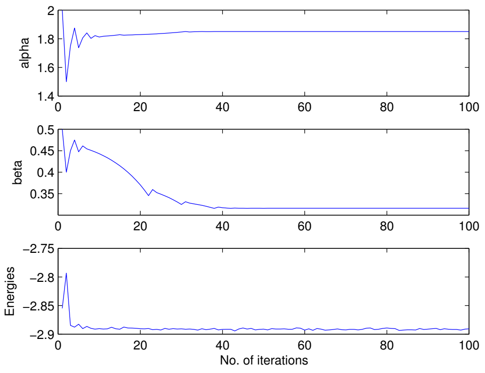


FIG. 2: The figure shows the evolution of the parameter values of α and β using the addaptive method detailed in section IV

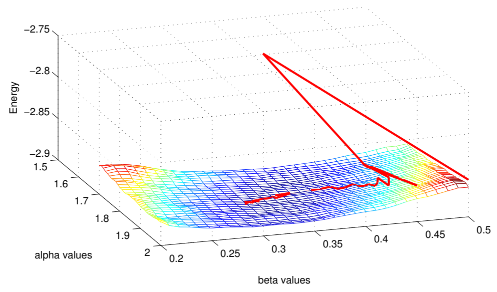


FIG. 3: in the energy space using the addaptive method detailed in section IV

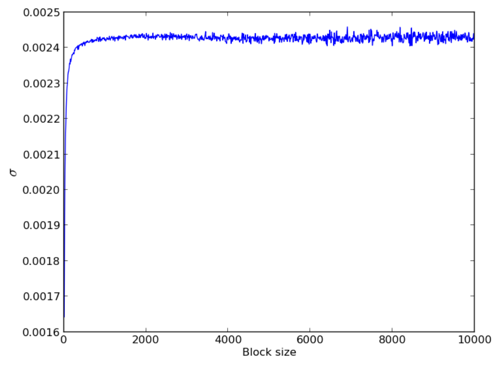


FIG. 4: The figure shows a typical example of a blocking analysis result.

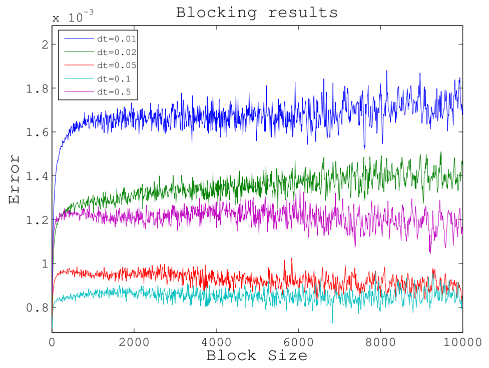


FIG. 5: The figure shows the blocking results for many different Δts. The results are obtained from runs of the Helium atom, using 10^6 Monte Carlo cycles.

V. STATISTICAL ANALYSIS USING BLOCKING

We want an estimate on how good our Monte Carlo result is. If we find an average of N uncorrolated measurements, then the standard deviation of the mean you found is given by the well known result

$$\sigma_N = \frac{1}{\sqrt{N}}\sigma \tag{65}$$

where σ is the standard deviation in the individual draws. But our draws using the metropolis algorithm are not uncorrolated. Far from it! Each step is absolutely dependent on the position of the previous step. In effect, this means that the above formula will give too nice an error estimate.

One step is affected by the previous step. But what about the step before that, or 10, 100 1000, 10 000 steps before that? The idea that chunks of steps far appart will be mostly uncorrelated forms the idea behind the blocking method. What we do is simply to collect blocks of m samples, find the average of these, and use the standard deviations in them to calculate σ_N as

$$\sigma_N = \frac{1}{\sqrt{m}}\sigma_m \tag{66}$$

where σ_m is the standard deviation of the averages in the blocks. The effectiveness of this method is shown in [3], we here simply note that if the samples were uncorrolated to begin with, this procedure will give the same result as calculating the error normally, regardless of m.

Figure 4 shows an example of a blocking result (The data is taken from a Neon run which is detailed in section VI). The error is read off as the maximum of the curve.

A. Error as a function of the Metropolis time step

The parameter Δt in the Metropolis algorithm sets the average trial step length. A too large step will give a small acceptance probability, while a too small step will move the walker an insignifigant amount. Both of these extremes lead to highly correlated data, and as such choosing the right Δt is important to obtain good results. Figure 5 shows the result of an experimental search for the right Δt.

Atom	α	β	$E_{\rm VMC}$	E_0	N	$\sigma_{\rm std}$	$\langle r_{ij}\rangle$	$\tilde{\epsilon}$
He	1.84	0.34	-2.8908	-2.9037	10^8	4e-4	1.363	4.44e-3
He	2.0	0.175	-2.84267	-2.9037	4×10^7	8e-4	1.284	...
He	2.0	-	-2.740	-2.9037	4×10^7	1e-3	1.085	...
He	1.69	-	-2.8426	-2.9037	4×10^7	9e-4	1.28	...
Be	3.88	0.12	-14.4827	-14.6664	10^8	5e-4	1.31	1.25e-2
Be	4.0	0.087	-14.4743	-14.6664	4×10^7	2e-3	2.25	...
Be	4.0	-	-13.696	-14.6664	4×10^7	3e-3	1.48	...
Be	3.37	-	-14.213	-14.6664	4×10^7	3e-3	1.76	...
Ne	10.33	0.073	-127.607	-128.884	10^8	3e-3	1.18	1.00e-2
Ne	10	0.104	-127.63	-128.884	4×10^6	2e-2	1.18	...
Ne	10	-	-112.20	-128.884	4×10^6	6e-2	0.68	...
Ne	7.81	-	-121.88	-128.884	4×10^6	9e-2	0.87	...

TABLE I: The table shows a summary of VMC results for He, Be and Ne. α and β are the variational parameters, E is the optimal energy attained, E_0 is the experimental value of the energy, N is the number of monte carlo cycles, $\sigma_{\rm std}$ is the standard error calculated as explained in section V, $\langle r_{ij}\rangle$ is the average electron separation and $\tilde{\epsilon}$ is the relative error of the VMC result compared to the experimental value.

Molecule	R	α	β	$E_{\rm VMC}$	E_0	N	$\sigma_{\rm std}$	$\langle r_{ij}\rangle$	$\tilde{\epsilon}$
H_2	1.4	1.29	0.39	-1.1584	-1.175	10^7	4e-4	2.17	1.41e-2
Be_2	4.63	3.65	0.55	-28.6927	-29.3385	10^7	3e-3	3.75	2.20e-2

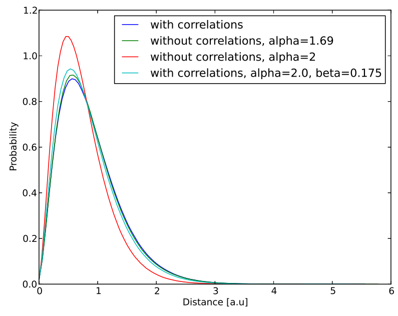
TABLE II: summary of VMC results for He_2 and Be_2. α and β are the variational parameters, E is the optimal energy attained, E_0 is the experimental value of the energy, N is the number of monte carlo cycles, $\sigma_{\rm std}$ is the standard error calculated as explained in section V, $\langle r_{ij}\rangle$ is the average electron separation and $\tilde{\epsilon}$ is the relative error of the VMC result compared to the experimental value.

VI. VMC CALCULATIONS FOR He, Be, Ne, H_2 AND Be_2

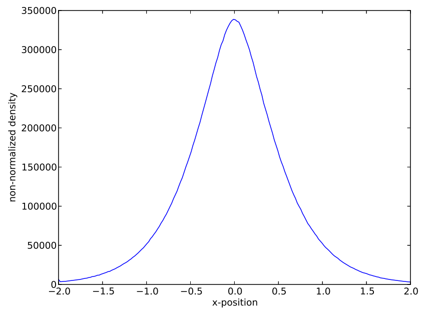
The adaptive method described in section IV was run for the different systems, after which a large scale Monte Carlo Simulation was run to calculate the optimal energies. The results for atoms are summarized in table I, and the results for molecules are summarized in table II.

A. One-body densities

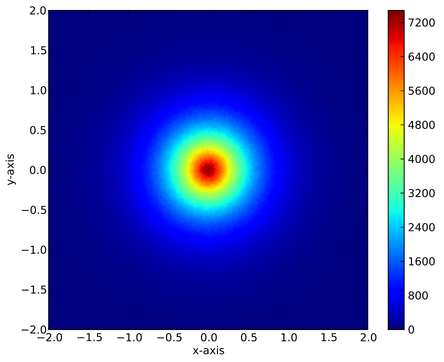
Using the optimal parameters detailed in the previous section, we can produce the one-body densities of the different systems. This is done by noting that the Metropolis algorithm creates the distribution defined by the wave function. So we simply store that walker positions for a large number of Monte Carlo cycles, and make histograms of the walker positions.



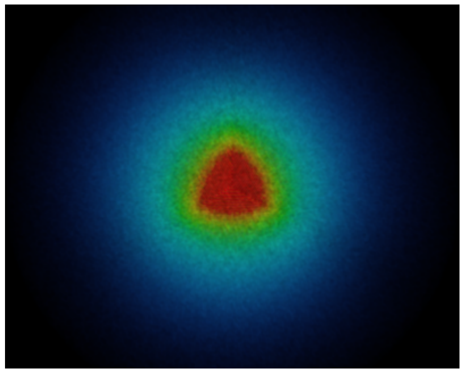
(a)radial distribution



(b)projection on the x-axis

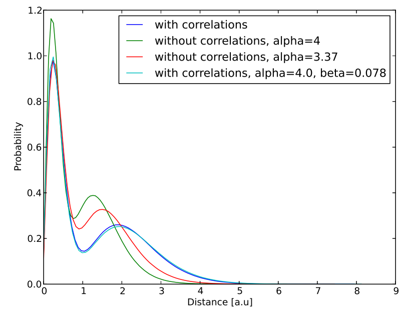


(c)projection on the xy-plane

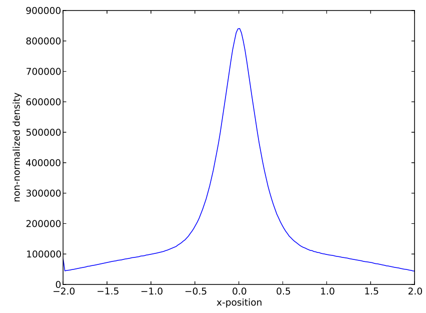


(d)3d distribution

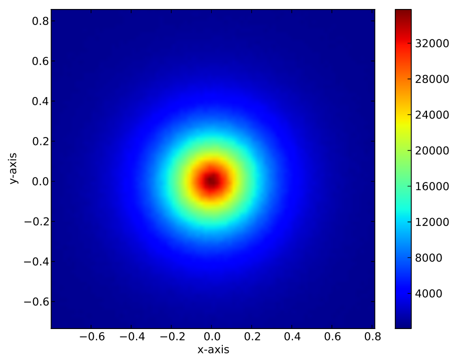
FIG. 6: The figures show various projections of the non-normalized one-body density of He. The radial distribution shows pure hydrogenic orbitals, optimization over α only, optimization over β only, and complete optimization. The rest show only the complete optimization.



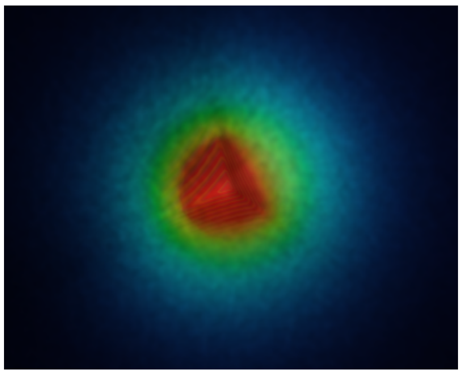
(a)radial distribution



(b)projection on the x-axis

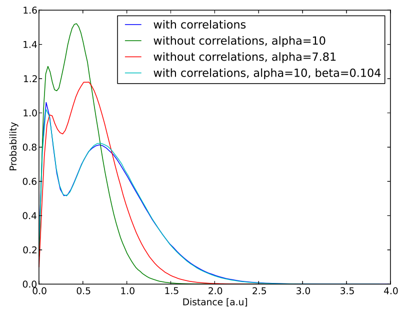


(c)projection on the xy-plane

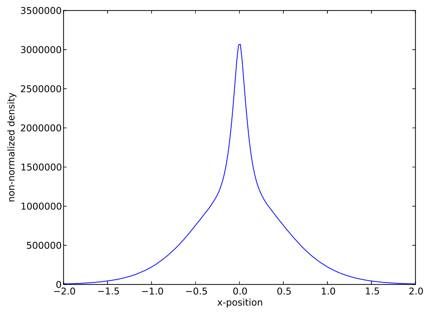


(d)3d distribution

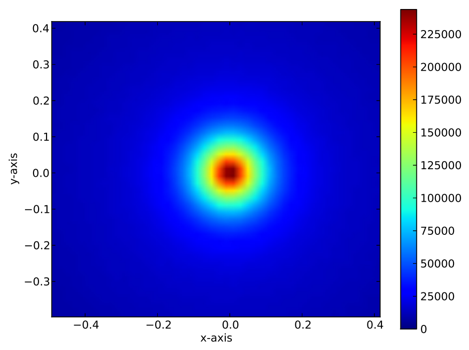
FIG. 7: The figures show various projections of the non-normalized one-body density of Be. The radial distribution shows pure hydrogenic orbitals, optimization over α only, optimization over β only, and complete optimization. The rest show only the complete optimization.



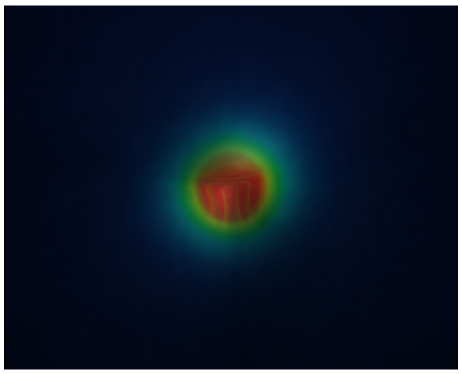
(a)radial distribution



(b)projection on the x-axis

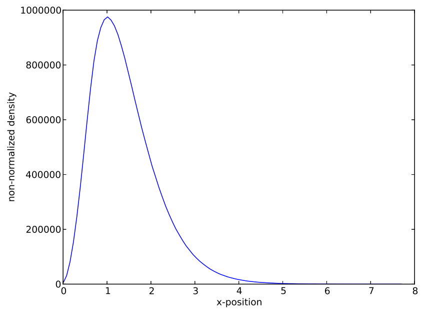


(c)projection on the xy-plane

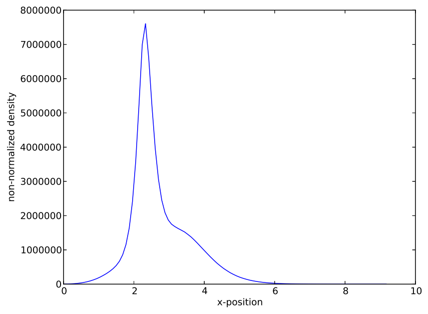


(d)3d distribution

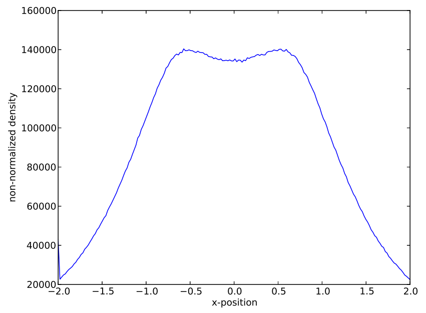
FIG. 8: The figures show various projections of the non-normalized one-body density of Ne. The radial distribution shows pure hydrogenic orbitals, optimization over α only, optimization over β only, and complete optimization. The rest show only the complete optimization.



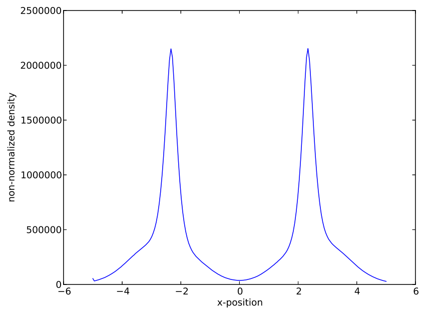
(a)radial distribution



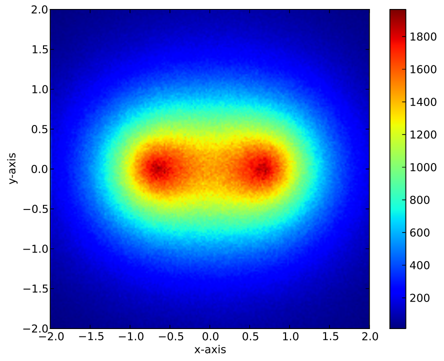
(a)radial distribution



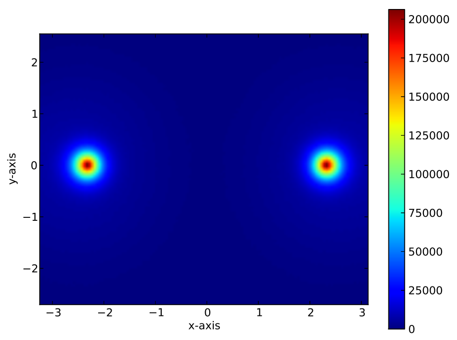
(b)projection on the x-axis



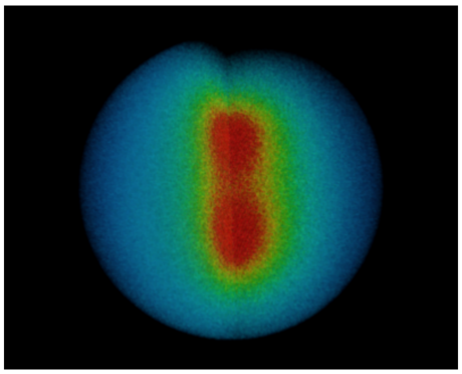
(b)projection on the x-axis



(c)projection on the xy-plane



(c)projection on the xy-plane


(d)3d distribution

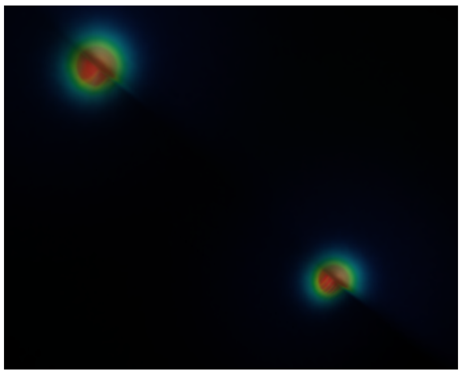

(d)3d distribution

FIG. 9: The figures show various projections of the non-normalized one-body density of H_2.

FIG. 10: The figures show various projections of the non-normalized one-body density of Be_2.

VII. CONCLUTIONS AND PERSPECTIVES

I have developed an object oriented code for doing VMC calculations of quantum systems. This includes parameter optimization search and statistical analysis, as well as the Metropolis algorithm with importance sampling for doing the main VMC calculations. The code runs the single Monte Carlo calculations in parallel using MPI, and the broad spectre brute force optimization using OpenMP.

Future work would involve generalizing the code further (see Appendix A for details on the current structure of the code). This would invole making a Hamiltonian class to allow for arbitrary potentials, as well as making the program easier to use (perhaps by including a wrapper in python or using some GUI). It would also by within reason to give the code a complete makeover, to ensure the different modules communicate better.

Acknowledgments

I would like to thank the instructors at the group for Computational Physics and the University of Oslo for many illuminating discussions, and in particular Jrgen Hgberget for providing useful results from VMC and DMC for benchmarking.

A. PROGRAM CODE

For the project I made an object oriented code, the program consists of the following classes:

- `Atom`. This essentially contains physical info such as the number of particles, the charge, whether the particle is an atom or a molecule etc.
- `VMCSetup`. The purpose of this function is to set up the different kinds of program runs, such as parameter optimization, production runs, etc.
- `VMCSolver`. This is created by `VMCSetup`, and does the actual Monte Carlo integration, using the Metropolis algorithm (with or without importance sampling)
- `WaveFunction`. This class creates the slater determinant and the jastrow (optional), and collects the values of wavefunctions, gradients and laplacians from these.
- `SlaterDeterminant`. Creates a vector of `Orbital` objects, and is used to calculate the value of the slater determinant, and its gradients and laplacians.
- `Jastrow`. Calculates the value, gradients and laplacians of the correlation part of the wave function.
- `Orbital`. This is called by `SlaterDeterminant`, and creates the single particles wave functions which makes up the determinant as well as the single particle gradients and laplacians used by functions in the SlaterDeterminant class.
- `MoleculeOrbital inherits Orbital`, the molecule version of `Orbital`.
- `Blocking`. Not a part of the rest of the program. Uses data collected from runs to calculate the standard error using the blocking method.

And overview of program calls is shown in figure 11. The code is available at https://github.com/andreavs/fys4411/tree/master/VMC.

B. EXPRESSIONS FOR GRADIENTS AND LAPLACIANS

When implementing the closed form expression for the energy earlier, we calculated

$$\frac{1}{\psi_T}\hat{H}\psi_T \tag{67}$$

where the tricky part was calculating the $\nabla_1^2\psi$ term. Let us see if we are able to calculate this term in our more general case. Our wavefunction can be factorized as

$$\psi_T(\{\mathbf{r}_i\}) = \psi_S(\{\mathbf{r}_i\})\psi_C(\{\mathbf{r}_i\}) \tag{68}$$

where ψ_S is the Slater Determinant, and ψ_C contains the jastrow factor. In this way, we can write the laplacian as

$$\nabla_i^2\psi_S\psi_T = \psi_C\nabla_i^2\psi_S + 2\nabla_i\psi_S\cdot\nabla_i\psi_S + \psi_S\nabla_i^2\psi_C^2 \tag{69}$$

This as the combined advantage of being close to how we would want to implement the closed form expressions, and making the derivations to come more digestable. Lets get to work. We'll start with the jastrow factor, and the results will be summarized in table III. We begin by noting that

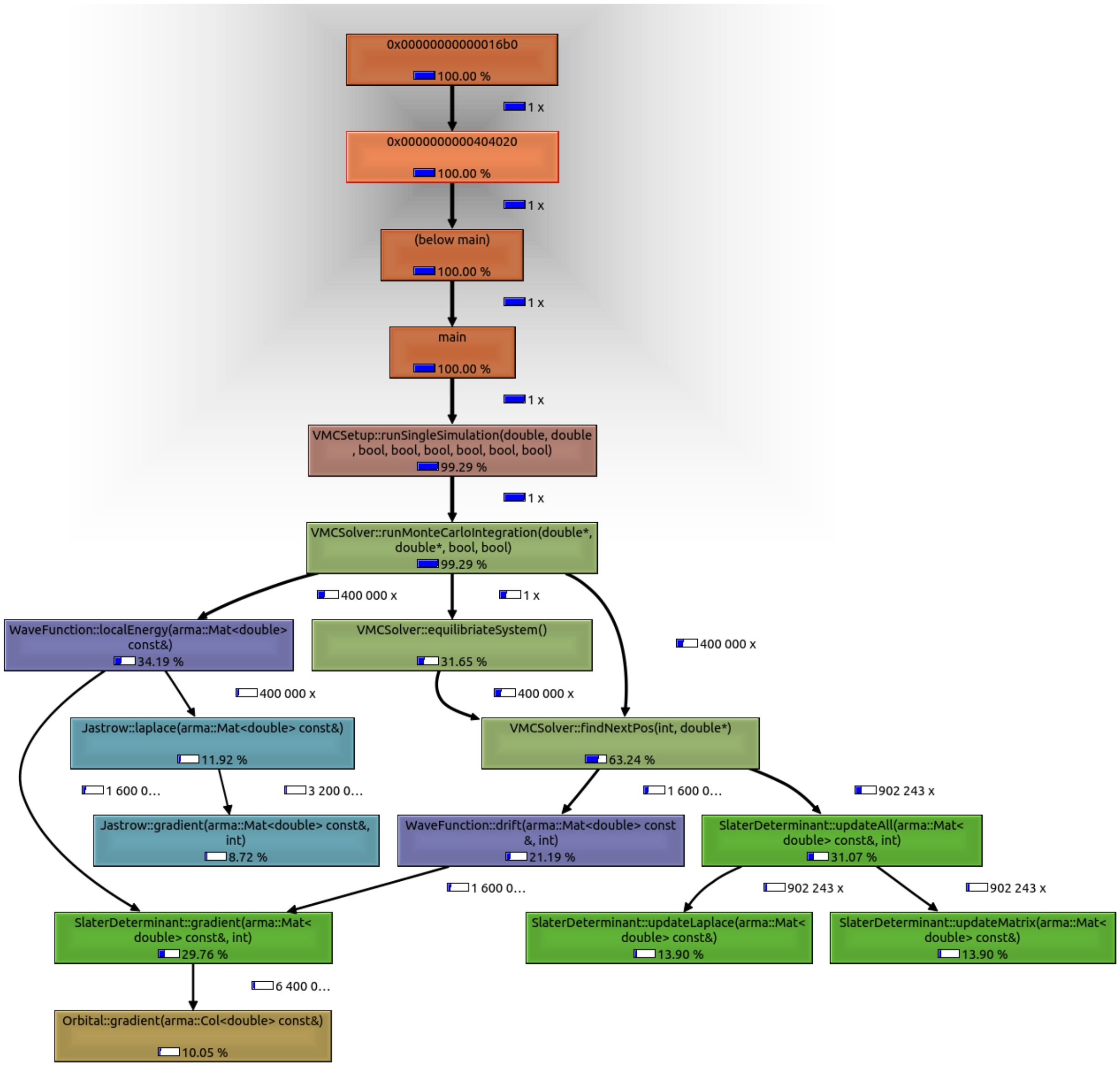

0x00000000000016b0
100.00 %
1 x
0x0000000000404020
100.00 %
1 x
(below main)
100.00 %
1 x
main
100.00 %
1 x
VMCSetup::runSingleSimulation(double, double , bool, bool, bool, bool, bool, bool)
99.29 %
1 x
VMCSolver::runMonteCarloIntegration(double*, double*, bool, bool)
99.29 %
400 000 x
1 x
400 000 x
WaveFunction::localEnergy(arma::Mat<double> const&)
34.19 %
VMCSolver::equilibriateSystem()
31.65 %
400 000 x
400 000 x
Jastrow::laplace(arma::Mat<double> const&)
11.92 %
VMCSolver::findNextPos(int, double*)
63.24 %
1 600 0...
3 200 0...
1 600 0...
902 243 x
Jastrow::gradient(arma::Mat<double> const&, int)
8.72 %
WaveFunction::drift(arma::Mat<double> const &, int)
21.19 %
SlaterDeterminant::updateAll(arma::Mat< double> const&, int)
31.07 %
1 600 0...
902 243 x
902 243 x
SlaterDeterminant::gradient(arma::Mat< double> const&, int)
29.76 %
SlaterDeterminant::updateLaplace(arma::Mat< double> const&)
13.90 %
SlaterDeterminant::updateMatrix(arma::Mat< double> const&)
13.90 %
6 400 0...
Orbital::gradient(arma::Col<double> const&)
10.05 %

FIG. 11: The figure shows the overview of the a typical program run

$$\frac{\partial}{\partial x_i}\psi_C = \frac{\partial}{\partial x_i}\prod_{j<k}^{n}\exp\left(\frac{a_{jk}r_{jk}}{1+\beta r_{jk}}\right) \tag{70}$$

$$= \frac{\partial}{\partial x_i}\exp\left(\sum_{j<k}^{n}\frac{a_{jk}r_{jk}}{1+\beta r_{jk}}\right) \tag{71}$$

$$= \left(\sum_{j\neq i}\frac{a_{ij}(x_i-x_j)}{r_{ij}(\beta r_{ij}+1)^2}\right)\psi_C \tag{72}$$

which can be generalized as

$$\nabla_i \psi_C = \left(\sum_{j \neq i} \frac{a_{ij}(\mathbf{r}_i - \mathbf{r}_j)}{r_{ij}(\beta r_{ij} + 1)^2} \right) \psi_C \tag{73}$$

Going further, using the product rule, we have that

$$\frac{\partial^2}{\partial x_i^2} \psi_C = \left[\left(\sum_{j \neq i} \frac{a_{ij}(x_i - x_j)}{r_{ij}(\beta r_{ij} + 1)^2} \right)^2 + \frac{\partial}{\partial x_i} \left(\sum_{j \neq i} \frac{a_{ij}(x_i - x_j)}{r_{ij}(\beta r_{ij} + 1)^2} \right) \right] \psi_C \tag{74}$$

$$= \left[\left(\sum_{j \neq i} \frac{a_{ij}(x_i - x_j)}{r_{ij}(\beta r_{ij} + 1)^2} \right)^2 + \sum_{j \neq i} \left(\frac{a_{ij}}{r_{ij}(\beta r_{ij} + 1)^2} - \frac{a_{ij}(x_i - x_j)^2(3\beta r_{ij} + 1)}{r_{ij}^3(\beta r_{ij} + 1)^3} \right) \right] \psi_C \tag{75}$$

$$= \sum_{j \neq i} \left[\frac{a_{ij}^2 (x_i - x_j)^2}{r_{ij}^2(\beta r_{ij} + 1)^4} + \frac{a_{ij}}{r_{ij}(\beta r_{ij} + 1)^2} - \frac{a_{ij}(x_i - x_j)^2(3\beta r_{ij} + 1)}{r_{ij}^3(\beta r_{ij} + 1)^3} \right. \tag{76}$$

$$\left. + 2 \sum_{k \neq j} \frac{a_{ij} a_{ik}(x_i - x_j)(x_i - x_k)}{r_{ij} r_{ik}(\beta r_{ij} + 1)^2(\beta r_{jk} + 1)^2} \right] \psi_C \tag{77}$$

and we can generalize this to

$$\nabla_i^2 \psi_C = \sum_{j \neq i} \left[\frac{a_{ij}^2}{(\beta r_{ij} + 1)^4} + \frac{3a_{ij}}{r_{ij}(\beta r_{ij} + 1)^2} - \frac{a_{ij}(3\beta r_{ij} + 1)}{r_{ij}(\beta r_{ij} + 1)^3} \right. \tag{78}$$

$$\left. + 2 \sum_{k \neq j} \frac{a_{ij} a_{ik}(\mathbf{r}_i - \mathbf{r}_j) \cdot (\mathbf{r}_i - \mathbf{r}_k)}{r_{ij} r_{ik}(\beta r_{ij} + 1)^2(\beta r_{jk} + 1)^2} \right] \psi_C \tag{79}$$

$$= \sum_{j \neq i} \left[\frac{a_{ij}^2}{(\beta r_{ij} + 1)^4} + \frac{2a_{ij}}{r_{ij}(\beta r_{ij} + 1)^3} + 2 \sum_{k \neq j} \frac{a_{ij} a_{ik}(\mathbf{r}_i - \mathbf{r}_j) \cdot (\mathbf{r}_i - \mathbf{r}_k)}{r_{ij} r_{ik}(\beta r_{ij} + 1)^2(\beta r_{jk} + 1)^2} \right] \psi_C \tag{80}$$

And so, to avoid redundant calculations, we can write the sum of the laplacians as

$$\sum_i \nabla_i^2 \psi_C = 2 \sum_i \sum_{j > i} \frac{a_{ij}}{(\beta r_{ij} + 1)^2} \left[\frac{a_{ij}}{(\beta r_{ij} + 1)^2} + \frac{2}{r_{ij}(\beta r_{ij} + 1)} + 4 \sum_{k > j} \frac{a_{ik}(\mathbf{r}_i - \mathbf{r}_j) \cdot (\mathbf{r}_i - \mathbf{r}_k)}{r_{ij} r_{ik}(\beta r_{jk} + 1)^2} \right] \psi_C \tag{81}$$

For the slater determinant, if we denote the matrix by $\mathcal{D}$, then its gradient is given by Jacobi's formula as

$$\nabla_i |\mathcal{D}(\mathbf{r})| = |\mathcal{D}(\mathbf{r})| \sum_{j=0}^{n} \nabla_i d_{ij}(\mathbf{r}) d_{ij}^{-1}(\mathbf{r}) \tag{82}$$

where d_{ij} is the ij-element of $\mathcal{D}$ and d_{ij}^{-1} is the ij-element of $\mathcal{D}^{-1}$. The laplacian is

$$\nabla_i^2 |\mathcal{D}(\mathbf{r})| = |\mathcal{D}(\mathbf{r})| \sum_{j=0}^{n} \nabla_i^2 d_{ij}(\mathbf{r}) d_{ij}^{-1}(\mathbf{r}) \tag{83}$$

We should calculate the gradients and laplacians of the relevant wave functions as well, we'll start with the 1s state, $\phi_{1s}(\mathbf{r}) = \exp(-\alpha r)$.

$$\nabla \phi_{1s}(\mathbf{r}) = \frac{-\alpha \mathbf{r} \exp(-\alpha r)}{r} \tag{84}$$

and

$$\nabla^2 \phi_{1s}(\mathbf{r}) = \alpha(\alpha r - 2)\exp(-\alpha r)/r, \tag{85}$$

These computations get tedious, and we should utilize some symbolic software such as sympy to avoid human errors. The rest of the wave function results are summarized in table III.

function	gradient	laplacian
$\phi_{1s}(\mathbf{r})$	$\alpha\mathbf{r}\phi_{1s}/r$	$\alpha(\alpha r-2)\phi_{1s}/r$
$\phi_{2s}(\mathbf{r})$	$\alpha\mathbf{r}(\alpha r-4)\exp(-\alpha r/2)/4r$	$-\alpha(\alpha^2r^2-10r\alpha+16)\exp(-\alpha r/2)/8r$
$\phi_{2px}(\mathbf{r})$	$-\alpha(\alpha x^2-2r,\alpha xy,\alpha xz)\exp(-\alpha r/2)/(2r)$	$\alpha^2x(\alpha r-8)\exp(-\alpha r/2)/4r$
$\phi_{2py}(\mathbf{r})$	$-\alpha(\alpha xy,\alpha y^2-2r,\alpha yz)\exp(-\alpha r/2)/(2r)$	$\alpha^2y(\alpha r-8)\exp(-\alpha r/2)/4r$
$\phi_{2pz}(\mathbf{r})$	$-\alpha(\alpha xz,\alpha yz,\alpha z^2-2r)\exp(-\alpha r/2)/(2r)$	$\alpha^2z(\alpha r-8)\exp(-\alpha r/2)/4r$
ψ_C	$\nabla_i\psi_C=\left(\sum_{j\neq i}\frac{a_{ij}(\mathbf{r}_i-\mathbf{r}_j)}{r_{ij}(\beta r_{ij}+1)^2}\right)\psi_C$	$2\sum_{j>i}\frac{a_{ij}}{(\beta r_{ij}+1)^2}\left[\frac{a_{ij}}{(\beta r_{ij}+1)^2}+\frac{2}{r_{ij}(\beta r_{ij}+1)} + 4\sum_{k>j}\frac{a_{ik}(\mathbf{r}_i-\mathbf{r}_j)\cdot(\mathbf{r}_i-\mathbf{r}_k)}{r_{ij}r_{ik}(\beta r_{ik}+1)^2}\right]\psi_C$
$\lvert\mathcal{D}(\mathbf{r})\rvert$	$\lvert\mathcal{D}(\mathbf{r})\rvert\sum_{j=0}^{n}\nabla_i d_{ij}(\mathbf{r})d_{ij}^{-1}(\mathbf{r})$	$\lvert\mathcal{D}(\mathbf{r})\rvert\sum_{j=0}^{n}\nabla_i^2 d_{ij}(\mathbf{r})d_{ij}^{-1}(\mathbf{r})$

TABLE III: The table summarizes the expression for the derivatives of our trial wave functions.

A. Expressions for molecules

as the molecule wave functions are made up of sums and differences of hydrogen orbitals, the gradients and laplacians will due to linearity simply be sums and differences of gradients and laplacians of the single particle states.

C. FAST COMPUTATION OF SLATER DETERMINANTS AND SLATER INVERSES

We have that as we are only changing one line of the matrix at the time, we can implement expressions which updates the different quantities of the slater matrix, rather than calculating them. This will gives us a large speed increase. Let us prove some of these.

First we have that the determinant of the slater matrix, $|D|$, can be written as

$$|D|=\sum_{j=1}^{N}D_{ij}C_{ij} \tag{86}$$

where C_{ji} are the cofactors of D (i.e. the determinants you get from crossing out the i-th row and $j-th$ column in D and then finding the determinant), for any choice of row i. In particular the row we update. Let us from now on call this row i. We have the the ratio between the one slater determinant and the next is

$$R_{SD}=\frac{|D(\mathbf{x}^{\mathrm{new}})|}{|D(\mathbf{x}^{\mathrm{cur}})|}=\frac{\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{new}})C_{ij}(\mathbf{x}^{\mathrm{new}})}{\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{cur}})C_{ij}(\mathbf{x}^{\mathrm{cur}})} \tag{87}$$

However, since only the i-th row is changed, we have $C_{ij}(\mathbf{x}^{\mathrm{cur}})=C_{ij}(\mathbf{x}^{\mathrm{new}})$, and also, $C_{ij}(\mathbf{x}^{\mathrm{cur}})=D_{ji}^{-1}|D|$ (this is not shown).

This means that

$$R_{SD}=\frac{\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{new}})C_{ij}(\mathbf{x}^{\mathrm{cur}})}{\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{cur}})C_{ij}(\mathbf{x}^{\mathrm{cur}})} \tag{88}$$

$$=\frac{\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{new}})D_{ji}^{-1}(\mathbf{x}^{\mathrm{cur}})|D|(\mathbf{x}^{\mathrm{cur}})}{\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{cur}})D_{ji}^{-1}(\mathbf{x}^{\mathrm{cur}})|D|(\mathbf{x}^{\mathrm{cur}})} \tag{89}$$

Now the denominator sums to 1 as D is invertible, and we end up with

$$R_{SD}=\sum_{j=1}^{N}D_{ij}(\mathbf{x}^{\mathrm{new}})D_{ji}^{-1}(\mathbf{x}^{\mathrm{cur}}) \tag{90}$$

and this gives us a quick way to evaluate the new determinant.

For the gradient, we have that

$$\nabla|D(\mathbf{r})|=\nabla_j\sum_{i=1}^{N}D_{ij}(\mathbf{r}_k)C_{ij} \tag{91}$$

where the cofactor matrix is independent of $\mathbf{r}_j$. Through similar arguments as above we find that

$$\frac{\nabla_j|D(\mathbf{r})|}{|D(\mathbf{r})|}=\sum_{i=1}^{N}\nabla_j D_{ij}(\mathbf{r})D_{ji}^{-1}(\mathbf{r})=\sum_{i=1}^{N}\nabla_j\phi_i(\mathbf{r}_j)D_{ji}^{-1}(\mathbf{r}) \tag{92}$$

and similarly

$$\frac{\nabla_j^2|D(\mathbf{r})|}{|D(\mathbf{r})|}=\sum_{i=1}^{N}\nabla_j^2\phi_i(\mathbf{r}_j)D_{ji}^{-1}(\mathbf{r}) \tag{93}$$

Finally, it can be shown (but this is more complicated) that

$$D^{-1}_{kj}(\mathbf{x}^{\mathrm{new}}) = \begin{cases} D^{-1}_{kj}(\mathbf{x}^{\mathrm{cur}}) - \frac{D_{ki}(\mathbf{x}^{\mathrm{cur}})}{R}\sum_{l=1}^{N} D_{il}(\mathbf{x}^{\mathrm{new}}) D^{-1}_{lj}(\mathbf{x}^{\mathrm{cur}}) & \mathrm{if} i \neq j \\ \frac{D_{ki}(\mathbf{x}^{\mathrm{cur}})}{R}\sum_{l=1}^{N} D_{il}(\mathbf{x}^{\mathrm{cur}}) D^{-1}_{lj}(\mathbf{x}^{\mathrm{cur}}) & \mathrm{if} i = j \end{cases} \tag{94}$$

[1] Moskowitz and Kalos, Int. Journal of Quantum Chemistry **20**, 1107 (1981).
[2] Filippi , Singh and Umrigar, J. Chemical Physics **105**, 123 (1996).
[3] H.Flyvbjerg and H. G. Petersen, J. Chemical Physics **91**, 91 (1989).