FYS4411 - Computational quantum mechanics

Spring 2016

Project 1; Variational Monte Carlo Studies of Bosonic systems

TEMPORARY REPORT

Sean Bruce Sangolt Miller
s.b.s.miller@fys.uio.no

Filip Henrik Lasren
filiphenriklarsen@gmail.com

Date: March 13, 2016

Abstract

Some text that is abstact

Contents

1 Introduction

2 Theory and methods

A brief on VMC

2.1 Preliminary derivations

2.1.1 Simplified problem

The local energy is defined as:

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} H\Psi_T(\mathbf{R}), \tag{1}$$

As a first approximation, it is assumed there is no interaction term in the Hamiltonian, which means the hard sphere bosons have no physical size (the hard-core diameter is zero). It is also assumed that no magnetic field is applied to the bosonic gas, leaving a perfectly spherically symmetrical harmonic trap. Inserting this new Hamiltonian into the local energy gives:

$$E_L(\mathbf{R}) = \frac{1}{\Psi_T(\mathbf{R})} \sum_i^N \left(\frac{-\hbar^2}{2m} \nabla_i^2 + \frac{1}{2} m\omega_{ho}^2 r_i^2 \right) \Psi_T(\mathbf{R}) \tag{2}$$

The potential term is trivial since this is a scalar, i.e. the denominator will cancel the wavefunction. A more challenging problem is to find an expression for $\nabla_i^2 \Psi_T(\mathbf{R})$. The trial wavefunction shown in equation (...), with the aforementioned approximations, is:

$$\Psi_T(\mathbf{R}) = \prod_i e^{-\alpha r_i^2} \tag{3}$$

where α is the variational parameter for VCM. The first derivative is:

$$\nabla_j \prod_i e^{-\alpha r_i^2} = -2\alpha \mathbf{r}_j e^{-\alpha r_j^2} \prod_{i \neq j} e^{-\alpha r_i^2} \tag{4}$$

$$= -2\alpha \mathbf{r}_j \prod_i e^{-\alpha r_i^2}. \tag{5}$$

The second derivative then follows:

$$\nabla_j^2 \prod_i e^{-\alpha r_i^2} = \nabla_j \left(-2\alpha \mathbf{r}_j \prod_i e^{-\alpha r_i^2} \right) \tag{6}$$

$$= \left(4\alpha^2 r_j^2 - 2d\alpha \right) \prod_i e^{-\alpha r_i^2}. \tag{7}$$

where d is the number of dimensions. Inserting this into back into the local energy (equation (2)), the final expression can be derived:

$$\begin{aligned} E_L(\mathbf{R}) &= \frac{1}{\Psi_T(\mathbf{R})} \sum_i^N \left(\frac{-\hbar^2}{2m} \nabla_i^2 + \frac{1}{2} m\omega_{ho}^2 r^2 \right) \Psi_T(\mathbf{R}) \\ &= \sum_{i=1}^N \left[\frac{-\hbar^2}{2m} \left(4\alpha^2 r_i^2 - 2d\alpha \right) + \frac{1}{2} m\omega_{ho}^2 r_i^2 \right] \end{aligned} \tag{8}$$

The drift force (quantum force), still with the approximations above, is defined by:

$$F = \frac{2\nabla \Psi_T}{\Psi_T} \tag{9}$$

The gradient here is defined as

$$\nabla \equiv (\nabla_1, \nabla_2, \ldots, \nabla_N)$$

i.e. a vector of dimension Nd. The gradient with respect to a single particle's position is already given in equation 5, so it's not too hard to realise:

$$\begin{aligned} F &= \frac{-4\alpha}{\Psi_T}\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right)\Psi_T \\ &= -4\alpha\left(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N\right) \end{aligned}$$

2.1.2 Full problem

The full local energy[1] is a bit more tedious to derive. The first step is to rewrite the trial wavefunction to the following form:

$$\Psi_T(\mathbf{R}) = \prod_i \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})} \tag{10}$$

where, in order for this to fit with the previous wavefunction, $u(r_{ij}) \equiv \ln[f(r_{ij})]$ and $\phi(\mathbf{r}_i) \equiv g(\alpha, \beta, \mathbf{r}_i)$. The gradient with respect to the k-th coordinate set is:

$$\nabla_k \Psi_T = \nabla_k \prod_i \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})} \tag{11}$$

$$= \nabla_k \phi_k \left[\prod_{i\neq k} \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})}\right] + \left[\prod_i \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})} \nabla_k \left(\sum_{i''<j''} u_{i''j''}\right)\right] \tag{12}$$

$$= \nabla_k \phi_k \left[\prod_{i\neq k} \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})}\right] + \left[\prod_i \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})} \left(\sum_{i''<j''} \nabla_k u_{i''j''}\right)\right] \tag{13}$$

The function u_{ij} is symmetric under permutation $i \leftrightarrow j$, as one can see from the definitions of u_{ij} and $f(r_{ij})$. Therefore, the last sum above can have a different indexing: All terms without an index k, will give zero when taking the derivative ∇_k, so only $i = k$ or $j = k$ remains (remember $i \neq j$). Due to the symmetry of u_{ij}, one can simply always say $i = k$ and let j be the summation index:

$$\nabla_k \Psi_T = \nabla_k \phi_k \left[\prod_{i\neq k} \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})}\right] + \left[\prod_i \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})} \left(\sum_{j''\neq k} \nabla_k u_{kj''}\right)\right] \tag{14}$$

The second derivative now becomes (where ∇_k only acts on the first parenthesis to its right):

$$\begin{aligned} \nabla_k^2 \Psi_T = {} & (\nabla_k^2 \phi_k) \left[\prod_{i\neq k} \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})}\right] + (\nabla_k \phi_k) \left[\prod_{i\neq k} \phi(\mathbf{r}_i) \nabla_k e^{\sum_{i'<j'} u(r_{i'j'})}\right] \\ & + \left[\nabla_k \left(\prod_i \left(\phi(\mathbf{r}_i)\right) e^{\sum_{i'<j'} u(r_{i'j'})}\right) \left(\sum_{j''\neq k} \nabla_k u_{kj''}\right)\right] \\ & + \left[\prod_i \phi(\mathbf{r}_i) e^{\sum_{i'<j'} u(r_{i'j'})} \left(\sum_{j''\neq k} \nabla_k^2 u_{kj''}\right)\right] \end{aligned}$$

While a bit of a nuisance to read, the expression above is simply the product rule for $\nabla_k(\nabla_k \Psi_T)$. Written in terms of Ψ_T, the above can be a bit simplified[2]:

[1] The "full local energy" means not making any assumptions on the particle interactions or the potential.

[2] For the more mathematically concerned nitpicker, the product symbol only runs over the functions $\phi(\mathbf{r}_i)$), not the following exponential. This is easy to realise by recalling how Ψ_T was defined, and is important to know when inserting Ψ_T as done now.

$$\nabla_k^2 \Psi_T = (\nabla_k^2 \phi_k) \frac{\Psi_T}{\phi(\mathbf{r}_k)} + (\nabla_k \phi_k) \left[\prod_{i \neq k} \phi(\mathbf{r}_i) \nabla_k e^{\sum_{i'<j'} u(r_{i'j'})} \right] + \left[(\nabla_k \Psi_T) \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) \right] + \left[\Psi_T \left(\sum_{j'' \neq k} \nabla_k^2 u_{kj''} \right) \right]$$

The second term above is equal to the second term in equation 14, divided by $\phi(\mathbf{r}_k)$. Furthermore, the gradient $\nabla_k \Psi_T$ is already calculated above, and in terms of Ψ_T is:

$$\nabla_k \Psi_T = \frac{\Psi_T}{\phi(\mathbf{r}_k)} \nabla_k \phi_k + \Psi_T \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) \tag{15}$$

Inserting all this back into the second derivative yields:

$$\nabla_k^2 \Psi_T = (\nabla_k^2 \phi_k) \frac{\Psi_T}{\phi(\mathbf{r}_k)} + (\nabla_k \phi_k) \left[\frac{\Psi_T}{\phi(\mathbf{r}_k)} \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) \right] + \left[\frac{\Psi_T}{\phi(\mathbf{r}_k)} \nabla_k \phi_k + \Psi_T \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) \right] \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) + \left[\Psi_T \left(\sum_{j'' \neq k} \nabla_k^2 u_{kj''} \right) \right]$$

Giving:

$$\frac{1}{\Psi_T} \nabla_k^2 \Psi_T = (\nabla_k^2 \phi_k) \frac{1}{\phi(\mathbf{r}_k)} + (\nabla_k \phi_k) \left[\frac{1}{\phi(\mathbf{r}_k)} \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) \right] + \left[\frac{1}{\phi(\mathbf{r}_k)} \nabla_k \phi_k + \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) \right] \left(\sum_{j'' \neq k} \nabla_k u_{kj''} \right) + \sum_{j'' \neq k} \nabla_k^2 u_{kj''}$$

Which can be rewritten to:

$$\frac{1}{\Psi_T} \nabla_k^2 \Psi_T = \frac{\nabla_k^2 \phi_k}{\phi(\mathbf{r}_k)} + \frac{2 \nabla_k \phi_k}{\phi(\mathbf{r}_k)} \left(\sum_{i \neq k} \nabla_k u_{ki} \right) + \left(\sum_{j \neq k} \nabla_k u_{kj} \right)^2 + \sum_{l \neq k} \nabla_k^2 u_{kl} \tag{16}$$

The gradients $\nabla_k u_{ki}$ can be rewritten using partial differentiation (where $r_{k,i}$ is coordinate i of $\mathbf{r}_k$):

$$
\begin{aligned}
\nabla_k u_{ki} &= \left(\frac{\partial}{\partial r_{k,1}}, \frac{\partial}{\partial r_{k,2}}, \dots\right) u_{ki} \\
&= \frac{\partial u_{ki}}{\partial r_{ki}} \left(\frac{\partial r_{ki}}{\partial r_{k,1}}, \frac{\partial r_{ki}}{\partial r_{k,2}}, \dots\right) \\
&= \frac{\partial u_{ki}}{\partial r_{ki}} \left(\frac{\partial}{\partial r_{k,1}} \left(\sqrt{\left[(r_{k,1} - r_{i,1})\hat{e}_1 + \dots\right]^2}\right), \dots\right) \\
&= \frac{\partial u_{ki}}{\partial r_{ki}} \left(\frac{r_{k,1} - r_{i,1}}{r_{ki}}, \dots\right) \\
&= \frac{\partial u_{ki}}{\partial r_{ki}} \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}} \\
&= \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}} u'_{ki} \ , \ u'_{ki} \equiv \frac{\partial u_{ki}}{\partial r_{ki}}
\end{aligned}
$$

While the second derivative of u_{ki} is:

$$
\begin{aligned}
\nabla_k^2 u_{ki} &= \nabla_k (\nabla_k u_{ki}) \\
&= u'_{ki} \nabla_k \left(\frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}}\right) + \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}} \nabla_k u'_{ki} \\
&= u'_{ki} \left(\frac{d}{r_{ki}} - \frac{r_{k,1} - r_{i,1}}{r_{ki}^3} - \frac{r_{k,2} - r_{i,2}}{r_{ki}^3} - \dots\right) + \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}} \cdot \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}} u''_{ki} \\
&= \frac{u'_{ki}}{r_{ki}} \left(d - \frac{r_{k,1} - r_{i,1}}{r_{ki}^2} - \frac{r_{k,2} - r_{i,2}}{r_{ki}^2} - \dots\right) + u_{ki} \\
&= \frac{u'_{ki}}{r_{ki}} (d - 1) + u_{ki}
\end{aligned}
$$

where d, as earlier, is the number of dimensions present, which in our world is usually $d = 3$. Finally, this gives:

$$
\frac{1}{\Psi_T} \nabla_k^2 \Psi_T = \frac{\nabla_k^2 \phi_k}{\phi(\mathbf{r}_k)} + \frac{2 \nabla_k \phi_k}{\phi(\mathbf{r}_k)} \left(\sum_{i \neq k} \frac{\mathbf{r}_k - \mathbf{r}_i}{r_{ki}} u'_{ki}\right) + \sum_{i,j \neq k} \frac{(\mathbf{r}_k - \mathbf{r}_i)(\mathbf{r}_k - \mathbf{r}_j)}{r_{ki} r_{kj}} u'_{ki} u'_{kj} + \sum_{i \neq k} u''_{ki} + \frac{u'_{ki}}{r_{ki}} (d - 1) \tag{17}
$$

As the exact forms of ϕ_k and u_{ki} are known, this can be written to a more recognisable, and calculable, expression. However, this expression will be quite long, so only the necessary variables will be derived. Inserting them into equation 17 is trivial. The u_{ki} derivatives are:

$$
\frac{\partial u_{ki}}{\partial r_{ki}} = \frac{\partial}{\partial r_{ki}} \left(\ln\left[1 - \frac{a}{r_{ki}}\right]\right) = \frac{a}{r_{ij}^2 - a r_{ij}} \tag{18}
$$

$$
\frac{\partial^2 u_{ki}}{\partial r_{ki}^2} = \frac{\partial}{\partial r_{ki}} u'_{ki} = -a \frac{2 r_{ki} - a}{(r_{ki}^2 - a r_{ki})^2} \tag{19}
$$

and the ϕ_k derivatives are:

$$
\nabla_k \phi_k = \nabla_k e^{-\alpha(x_k^2 + y_k^2 + \beta z_k^2)} = -2\alpha(x_k, y_k, \beta z_k) \phi_k \tag{20}
$$

$$
\nabla_k^2 \phi_k = \nabla_k (\nabla_k \phi_k) = \left[-2\alpha(2 + \beta) + 4\alpha^2 (x_k^2 + y_k^2 + \beta^2 z_k^2)\right] \phi_k \tag{21}
$$

2.2 Metropolis algorithm

The basic principle behind the Metropolis algorithm is to make an assumption on the transition probability for a system to move from setting to another, as an exact, or even approximate, expression is lacking.

If the probability distribution for a state i is given by w_i, then from Markov chain theory the time derivative is:

$$
\frac{\partial w_i(t)}{\partial t} = \sum_j W(j \to i) w_j(t) - W(i \to j) w_i(t) \tag{22}
$$

where $W_{i\to j}$ is the probability of moving from a state i to another state j, i.e the rate of change in w_i is given by the probability for a state j to go to i minus the probability of state i going to j, summed over all j. The most likely state will fulfil $\frac{\partial w_i(t)}{\partial t} = 0$, giving:

$$\begin{aligned} W(j \to i)w_j(t) &= W(i \to j)w_i(t) \\ \Rightarrow \quad \frac{W(j \to i)}{W(i \to j)} &= \frac{w_i}{w_j} \end{aligned} \tag{23}$$

Since the transition probability W is unknown, we approximate it by guessing its form:

$$W(j \to i) = T(j \to i)A(j \to i) \tag{24}$$

where T is the transition moving probability, while A is the probability of accepting such a move. Furthermore, in brute force Metropolis, one guesses $T_{i\to j} = T_{j\to i}$. Therefore:

$$\frac{A_{j\to i}}{A_{i\to j}} = \frac{w_i}{w_j} \tag{25}$$

Since the probability densities are known, it is known whether or not this ratio is larger than one. If it's larger than one, then the acceptance probability from j to i is the biggest, i.e we are more likely to accept the move than not. Therefore, we simply say the move is accepted. However, if the probability ratio is smaller than one, then we are more likely to move from the new state to the one the system is currently in. To check whether or not we should accept the move, we may compare it to, say, a "coin toss"; if it's bigger, the move is accepted.

2.2.1 Importance sampling

The standard Metropolis algorithm with brute force sampling accepts approximately half of the proposed moves. When a move is rejected we end up sampling the same state several times. This seems like an aweful waste of CPU time. A possible improvement is to propose "better" moves. From the Langevin equations one can propose new moves as

$$x_{new} = x_{old} + DF(x_{old})\Delta t + \xi\sqrt{\Delta t}, \tag{26}$$

where x is the position, D is a diffusion coefficient, F is a drift velocity biasing our new move toward more probable states, Δt is a time step length whose significance will be addressed later, and ξ is a normal distributed random number with mean value of 0 and standard deviation of $\sqrt{2D\Delta t}$. The force term is derived from the Fokker-Planck equation

$$\frac{\partial P}{\partial t} = D\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x} - F\right)P(x,t), \tag{27}$$

where $P(x,t)$ is a time-dependent probability density in one dimension. However, a stationary probability density will occur only when the left hand side is zero. This leave us with

$$\frac{\partial^2 P}{\partial x^2} = P\frac{\partial}{\partial x}F + F\frac{\partial}{\partial x}P. \tag{28}$$

Since we are seeking a stationary state, it can be shown that this leads F being on the form

$$F = \frac{1}{P}\frac{\partial P}{\partial x}. \tag{29}$$

In a quantum mechanical interpretation the probability distribution is given by the wavefunctions, so our drift force can be expressed as

$$\begin{aligned} F &= \frac{1}{|\Psi_T|^2}\nabla|\Psi_T|^2 \\ &= 2\frac{1}{\Psi_T}\nabla\Psi_T, \end{aligned} \tag{30}$$

which is known as the quantum force.

By utilizing equation (26) when proposing new moves, they will be to areas where the wavefunction is large, i.e. there is a large probability.

The Fokker-Planck equation has a solution in the form of the Green's function:

$$G(y,x,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left\{-\frac{(y-x-D\Delta t F(x))^2}{4D\Delta t}\right\} \tag{31}$$

giving the transition probability matrix T stated earlier. Therefore, the ratio test to be compared with the acceptance probability ratio is now:

$$q(y,x) = \frac{G(x,y,\Delta t)}{G(y,x,\Delta t)} \frac{|\Psi_T(y)|^2}{|\Psi_T(y)|^2} \tag{32}$$

with $q(y,x)$ being the ratio (or "coin toss") for accepting the move from setting x to y. In the same fashion as the toggle for use of analytical calculation of the Laplacian, a user may choose to apply importance sampling by setting

2.3 Optimizing parameters

The trial wavefunction discussed earlier is not exact for the interacting case. While the Jastrow factor includes a SOME NOTES ON JASTROW FACTOR, the variable α will have to be found. Obviously, in the VMC approach, the minimal energy is sought. Therefore, minimizing $\langle E_L\rangle$ with respect to the variational parameter (α) is the goal. There are several ways to minimize a value with respect to some parameter, and here we will use the method of steepest descent (SD). The SD method, in algorithm form, is:

$$x_{n+1} = x_n - \gamma_n \nabla f \tag{33}$$

where x is the variable with which one wishes to find the minimum of f. In application to the current problem, $f = \langle E_L\rangle$. However, since $\langle E_L\rangle$ is an expensive quantity to find numerically, and its derivative ($\bar{E}_\alpha \equiv \frac{d\langle E_L\rangle}{d\alpha}$) even more so, an analytical expression is desirable. This can be found as follows:

$$\begin{aligned}
\bar{E}_\alpha &= \frac{d}{d\alpha}\int dx P(x) E_L \\
&= \frac{d}{d\alpha}\int dx \frac{|\psi|^2}{\int dx'|\psi|^2}\frac{1}{\psi}H\psi \\
&= \frac{d}{d\alpha}\int dx \frac{\psi^* H\psi}{\int dx'|\psi|^2}
\end{aligned} \tag{34}$$

Since the Hamiltonian is hermitian, one has $\int dx\psi^* H\psi = \int dx H\psi^*\psi$, giving:

$$\begin{aligned}
&= \frac{d}{d\alpha}\int dx \frac{H\psi^*\psi}{\int dx'|\psi|^2} \\
&= \left[\int dx \frac{H\left(\psi^*\left(\frac{d\psi}{d\alpha}\right) + \left(\frac{d\psi^*}{d\alpha}\right)\psi\right)}{\int dx'|\psi|^2}\right] - \left[\int dx \frac{H\psi^*\psi}{\left(\int dx'|\psi|^2\right)^2}\int dx'\left(\psi^*\left(\frac{d\psi}{d\alpha}\right) + \left(\frac{d\psi^*}{d\alpha}\psi\right)\right)\right]
\end{aligned} \tag{35}$$

Again one may use the hermiticity of the Hamiltonian to get $\int dx H\psi^*\left(\frac{d\psi}{d\alpha}\right) = \int dx H\left(\frac{d\psi^*}{d\alpha}\right)\psi$. So:

$$\begin{aligned}
&= 2\left[\int dx \frac{H\psi^*\frac{d\psi}{d\alpha}}{\int dx'|\psi|^2}\right] - 2\left[\int dx \frac{H\psi^*\psi}{\left(\int dx'|\psi|^2\right)^2}\int dx'\psi^*\frac{d\psi}{d\alpha}\right] \\
&= 2\left[\int dx \frac{H\psi^*\frac{d\psi}{d\alpha}}{\int dx'|\psi|^2} - \int dx \frac{H\psi^*\psi}{\int dx'|\psi|^2}\int dx' \frac{1}{\int dx'|\psi|^2}\psi^*\frac{d\psi}{d\alpha}\right] \\
&= 2\left[\int dx \frac{\psi^*\left(\frac{E_L}{\psi}\frac{d\psi}{d\alpha}\right)\psi}{\int dx'|\psi|^2} - \int dx \frac{\psi^* E_L\psi}{\int dx'|\psi|^2}\int dx' \frac{\psi^*\left(\frac{1}{\psi}\frac{d\psi}{d\alpha}\right)\psi}{\int dx'|\psi|^2}\right] \\
&= 2\left(\langle\frac{\bar{\psi}_\alpha}{\psi}E_L\rangle - \langle\frac{\bar{\psi}_\alpha}{\psi}\rangle\langle E_L\rangle\right)
\end{aligned} \tag{36}$$

where $\bar{\psi}_\alpha \equiv \frac{d\psi}{d\alpha}$. In the non-interacting, simple harmonic oscillator potential, it is analytically shown $\alpha = \frac{1}{2}$, and due to the Jastrow factor, one would assume it doesn't differ too much. The equation above is a much better expression

to use since one only need one Monte Carlo cycle to find $\bar{E}_\alpha$. Therefore, only one Monte Carlo cycle[3] will give one value for $\bar{E}_L$.
Since α will only have to be determined once, it is permissible to do so with greater accuracy. Therefore, if the gradient at x_{n+1} is greater than that at x_n, we go back to x_n and halve the steplength γ_n, to calculate another x_{n+1}. This is repeated until either the gradiant becomes sufficiently small (around order 10^{-6}) or if the steplength becomes too small to continue (order 10^{-6} as well).

2.4 Blocking method

After having run a Monte Carlo simulation, the variation in estimated local energies are calculated as

$$\sigma = \sqrt{\frac{1}{n}\left(\langle E_L^2\rangle - \langle E_L\rangle^2\right)}. \tag{37}$$

However, these values will be much too low. This is because one assumes all data to be completely uncorrelated. Each energy is calculated by a small perturbation to the system setting[4], which means each new setting is very dependant on the previous setting. After sufficiently many perturbations, though, the system at step i will be so different from that of step j, that $\langle E_L\rangle_i$ is basically uncorrelated to $\langle E_L\rangle_j$, but not for all $\langle E_L\rangle_k$ between i and j. Ideally, one would like to find a *correlation time* τ such that i and j will be uncorrelated if a time greater than τ has passed. If Δt is the time between two Metropolis steps, then one would like to find $|i-j|$ in $\tau = |i-j|\Delta t$.
A method of dealing with this is the blocking technique. The set of $\langle E_L\rangle$ is grouped in blocks, each of which will give an average of average local energies. Afterwards, one can calculate the variance of these averages of averages. If then the standard deviation is plotted as a function of the number of blocks[5], one wants to find the lowest number of blocks where the curve approaches a plateau. Then one can calculate τ and find the correlation time.

$$\sigma = \sqrt{\frac{1+2\tau/\Delta t}{n}\left(\langle \mathbf{M}^2\rangle - \langle \mathbf{M}\rangle^2\right)} \tag{38}$$

2.5 Program structure

The program discussed is object oriented and written in C++. It is made up of several classes, each managing its own set of tasks. Below is an overview of the most central classes; their name and functionality.

`System`	The cornerstone of the structure. Every other class communicates with this class in some way. Stores the chosen `Hamiltonian`, `Wavefunction` and a vector `m_atoms` which holds all the atoms. Holds all global parameter such as number of particles, dimensions and metropolis steps. Actually does the Metropolis iterations.
`Sampler`	Stores the local energies, computes their average and variance at the end of the simulations. Prints results to screen and writes to file.
`Particle`	Objects of this class holds a vector for its position.
`Hamiltonian`	Contains a function to compute kinetic energy and constitutes that all specific types of hamiltonians (subclasses) must have a function `computeLocalEnergy`.
`Wavefunction`	Constitutes that all specific wavefunctions (subclasses) must have the functions `evaluate` and `computeLaplacian`.
`InitialState`	Adds `Particle` objects to the `m_atoms` vector in `System` and give them positions. A subclass can for instance distribute particles uniformly in space.
`Optimizer`	Optimizes variational parameters. Used when including the Jastrow factor in the wavefunction.

In `main.cpp` a user can set parameters of the system, such as number of particles, dimensions, Metropolis steps, alpha, beta, etc. Which subclass of `Hamiltonian`, `Wavefunction` and `InitialState` must be chosen here as well.

[3] As opposed to several needed to numerical derivation.
[4] In the program discussed here, only a single dimension of a single particle will be moved each step.
[5] Inversely proportional to the block size by block size $= \frac{\text{number of samples}}{\text{number of blocks}}$

An example may look like

```
System* system = new System();
system->setHamiltonian                  (new HarmonicOscillator(system, omegaHO));
system->setWaveFunction                 (new SimpleGaussian(system, alpha));
system->setInitialState                 (new RandomUniform(system, numberOfDimensions, numberOfParticles));
```

Also, one may choose to use an analytical computation of the laplacian of the wavefunction, use importance sampling, store local energies and store positions by setting the desired option to `true`.

```
system->setAnalyticalLaplacian          (false);
system->setImportanceSampling           (false);
system->setStoreLocalEnergy             (false);
system->setStorePositions               (false);
```

3 Results

3.1 Benchmarks

As in most computational work, we need to make certain our program produce reliable values. A very useful way to do this is to compare our numerical results with a known analytical solution to the problem.

3.1.1 Harmonic Oscillator without the Jastrow factor

Since we do know the analytical solution to the spherical harmonic oscillator, these may be used as benchmarks that we hope to reproduce. We have already derived the analytical expression in (8).

Table 1: Numerically calculated laplacian.

N	D	Analytical	Numerical	Variance	Time [s]
1	1	0.5	0.5	3.1e-20	0.14
1	2	1	1	9.4e-19	0.153
1	3	1.5	1.5	2.1e-18	0.115
10	1	5	5	3.6e-18	0.346
10	2	10	10	3.2e-17	0.378
10	3	15	15	3.2e-16	0.414
100	1	50	50	6.0e-15	3.41
100	2	100	100	1.2e-13	4.69
100	3	150	150	8.5e-13	6.02
500	1	250	250	9.5e-12	29.4
500	2	500	500	7.6e-11	55.8
500	3	750	750	2.3e-10	88.5

Table 2: Analytically calculated laplacian.

N	D	Analytical	Numerical	Variance	Time [s]
1	1	0.5	0.5	0	0.119
1	2	1	1	0	0.142
1	3	1.5	1.5	0	0.079
10	1	5	5	0	0.333
10	2	10	10	0	0.354
10	3	15	15	0	0.326
100	1	50	50	0	2.92
100	2	100	100	0	2.93
100	3	150	150	0	2.89
500	1	250	250	0	14.6
500	2	500	500	0	15.1
500	3	750	750	0	15.3

The results in table 1 and 2 are from simulations using $\alpha = 0.5$, an acceptance rate ≈ 0.5 and 10^4 metroplolis steps. In these testes we had the exact wavefunction and would expect no error at all. Yet, table 1 show that for large numbers of particles and dimensions, we might get minor errors. though these are very small compared to the energy. The reason we get these variations though, is the fact that we approximate the laplacian of the wavefunction numerically. The error in this approximation goes as h^2, where h is the steplength in the evaluation (1e-04 in this case). Also the variance will depend on the number of metropolis steps as shown in equation (37).

As can be seen in table 2, the variance in the local energies are zero when we compute the laplacian of the wavefunction analytically. This is because we do not do any approximations. By comparing two tables one can clearly see another advantages of computing the laplacian analytically. The time spent is much less. The reason is that when computing numerically, one has to evaluate the wavefunction 4 times, which is one of the most time consuming parts of the program. The analytically computed laplacian, equation (7), use only parameters already known.

Importance sampling as explained in section 2.2.1, may be used to increase the efficiency of the sampling. Running the same tests using this technique produced table 3.

Table 3: Imporance sampling and numerically computed laplacian

N	D	Analytical	Numerical	Variance	Time
1	1	0.5	0.5	3.4e-20	0.10
1	2	1	1	4.9e-19	0.09
1	3	1.5	1.5	1.8e-18	0.10
10	1	5	5	1.1e-17	0.36
10	2	10	10	2.8e-17	0.37
10	3	15	15	3.6e-16	0.41
100	1	50	50	7.0e-15	3.62
100	2	100	100	1.2e-13	4.56
100	3	150	150	4.2e-13	5.88
500	1	250	250	1.8e-12	29.0
500	2	500	500	7.4e-11	56.0
500	3	750	750	3.6e-10	89.9

In this case we did not see any significant differences in the results produced using importance sampling and not. However the acceptance rate when using it is about 0.999, in contrast to 0.5 when not using it. The acceptance rate is however dependent on the steplength Δt as in equation (26).

3.1.2 Parameter optimization

The harmonic oscillator may also be used to test if the parameter optimization function. Since, we know that for the simple harmonic oscillator without interactions, the parameter $\alpha = 0.5$ should ideally be produced. Running this test on the optimizer actually produced the value $\alpha = 0.5000000028$, which we can be very pleased with.

3.1.3 Conclutions

These benchmarks indicate that the program works as it should without the Jastrow factor, and from here we may add the complexity of the full interacting system.

3.2 Onebody density

After a few Monte Carlo interations, the system will knverge to a steady state. From the steady state, the probability density does not change with time. By sampling the position of all particles at all timesteps after this stady state is reached, one can easily make a histogram of what ranges of radial distances are the most probable. This is what we call the onebody density. SOMETHING ABOUT OUR WAVEFUNCTION BEING A PRODUCT OF UNDEPENDENT WAVEFUNCTIONS.

For the case of 10 particles in the eliptical trap, $\alpha =$ blah and $\beta = 2.82843$, the one body density was found to be as shown in figure blah4.

A three dimensional illustration of the density distribution in space is show in figure blah5, and a two dimensional projection is shown in figure blah6 and blah7.

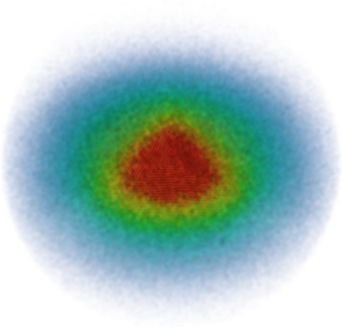

Figure 1

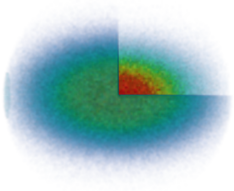

Figure 2

4 Conclusions

5 Appendix