FYS 4411

Andreas Leonhardt

May 29, 2013

Contents

1 The System

Given some arbitrary wavefunction $\left|\Psi\right\rangle$ and the wave function of the ground state $\left|\Psi_0\right\rangle$ we notice that

$$E_0 = \frac{\left\langle\Psi_0\middle|\hat{H}\middle|\Psi_0\right\rangle}{\left\langle\Psi_0\middle|\Psi_0\right\rangle} \leq \frac{\left\langle\Psi\middle|\hat{H}\middle|\Psi\right\rangle}{\left\langle\Psi\middle|\Psi\right\rangle} \tag{1}$$

For normalized wavefunctions the denominators will be 1. The inequality allows us to set a upper limit to the ground state energy using any wavefunction as a trial function.

The inner product in the RHS of equation 1 is defined as an integral and can be rewritten as

$$\left\langle\Psi\middle|\hat{H}\middle|\Psi\right\rangle = \int \mathrm{d}^3r \Psi\hat{H}\Psi = \int \mathrm{d}^3r |\Psi| \cdot \underbrace{\frac{1}{\Psi}\hat{H}\Psi}_{\epsilon(\vec{r})} \tag{2}$$

In this way it is written as an integral over the local energy ϵ multiplied with a probability density. We solve this using Monte Carlo Integration.

1.1 Single Atoms

We want to calculate the ground state energy of various atoms: Helium, Beryllium and Neon. They all have in common that they have closed shells and an even number of electrons, which simplifies things a bit.

The Hamilton operator in atomic units is given by

$$\hat{H} = \sum_{i=1}^{N} \left(\frac{1}{2}\nabla_i^2 + -\frac{Z}{r_i} + \sum_{j<i} \frac{1}{r_{ij}} \right), \tag{3}$$

where N is the number of electrons, Z the charge of the nucleus, $r_i = |\vec{r}_i|$ and $r_{ij} = |\vec{r}_i - \vec{r}_j|$. Since the first two terms alone have the hydrogen like wave functions as eigenstates, we use them for our ansatz. Taking into account that electrons are fermions, we can occupy every state with two electrons. Furthermore we have to antisymmetrize the ansatz. Since the Hamilton operator is independent of the spin of the particles, we can split the antisymmetrization and describe the wave function by a product of two independent Slater determinants, one for particles with spin up and one for all particles with spin down. Furthermore we include a factor, that provides an additional parameter to interactions, the so called Jastrow factor. The complete ansatz reads then

$$\Psi = |D_\uparrow| \cdot |D_\downarrow| \cdot \left(\prod_{i<j} \exp\left(-\frac{a r_{ij}}{1 + \beta r_{ij}} \right) \right) \tag{4}$$

with $a = \frac{1}{2}$ for electrons with equal spin and $a = \frac{1}{4}$ for opposite spins. The first two terms are the Slater determinants for spin up and down respectively, the last term is the Jastrow factor or correlation part, here also called Ψ_C. The Slater matrices are defined through

$$(D_\uparrow)_{ij} = \phi_j(\vec{r}_i), \quad (D_\downarrow)_{(k-N/2)j} = \phi_j(\vec{r}_k) \tag{5}$$

where $j = 0, \ldots, N-1$; $i = 0, \ldots, \frac{N}{2} - 1$ and $k = \frac{N}{2}, \ldots, N-1$.

orbital	ϕ_i	$\nabla\phi_i$	$\Delta\phi_i$
1s	$\mathrm{e}^{-\alpha r}$	$-\alpha\frac{\vec{r}}{r}\mathrm{e}^{-\alpha r}$	$\frac{\alpha}{r}(\alpha r - 2)\,\mathrm{e}^{-\alpha r}$
2s	$\frac{1}{2}(2 - \alpha r)\,\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\frac{\alpha\vec{r}}{4r}(\alpha r - 4)\,\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\frac{\alpha}{8r}(\alpha r - 8)(2 - \alpha r)\,\mathrm{e}^{-\frac{1}{2}\alpha r}$
2p	$\alpha x\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\left(\alpha e_x - \frac{\alpha^2}{2}x\frac{\vec{r}}{r}\right)\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\frac{\alpha^2 x}{4r}(\alpha r - 8)\,\mathrm{e}^{-\frac{1}{2}\alpha r}$
	$\alpha y\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\left(\alpha e_y - \frac{\alpha^2}{2}y\frac{\vec{r}}{r}\right)\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\frac{\alpha^2 y}{4r}(\alpha r - 8)\,\mathrm{e}^{-\frac{1}{2}\alpha r}$
	$\alpha z\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\left(\alpha e_z - \frac{\alpha^2}{2}z\frac{\vec{r}}{r}\right)\mathrm{e}^{-\frac{1}{2}\alpha r}$	$\frac{\alpha^2 z}{4r}(\alpha r - 8)\,\mathrm{e}^{-\frac{1}{2}\alpha r}$

Table 1: unnormalized single particle wave functions and derivatives

The hydrogen like wavefunctions and their derivatives are defined in table 1 up to the 2p level, so we can fill in at most 10 electrons (two for each state), which is sufficient for Neon. In the hydrogen case the Slater determinants reduce to the 1s function. Φ contains two variational paramaters, α in the single particle wavefunctions, β in the Jastrow factor. α determines the width of the hydrogen like wave functions. In the situation of only one electron, where we have analytical solutions, we get $\alpha = Z$ for the exact solution. Since the electron repell each other we expect α to be a little bit smaller. This can also be seen as an effect from screening, where the electrons reduce the effective charge of the nucleus due to their own charge. This effect might become more important for atoms with more electrons such as Neon. The second parameter β allow for correlations between the electrons.

1.2 Hydrogen Molecule

We now extend the program to one of the most simple molecules, H_2. We use the Born-Oppenheimer approximation, where the degrees of freedom for the nuclei are frozen, e.g. they are treated as fixed in space. We choose to align them along the z-axis, at $\pm\frac{1}{2}\vec{R} = \frac{1}{2}R\vec{e_z}$. The Hamilton operator reads then

$$\hat{H} = \sum_{i=1}^{N} \left(\frac{1}{2}\nabla_i^2 + -\frac{Z}{|\vec{r}_i - \frac{1}{2}\vec{R}|} - \frac{Z}{|\vec{r}_i + \frac{1}{2}\vec{R}|} + \sum_{j<i} \frac{1}{r_{ij}} \right) + \frac{1}{R}. \tag{6}$$

The ansatz for the wave function is again build on the hydrogen like wave functions and contains the Jastrow factor,

$$\begin{aligned} \Psi &= \tilde{\phi}_1 \cdot \tilde{\phi}_2 \cdot \Psi_C \\ \tilde{\phi}_i &= \mathrm{e}^{-\alpha|\vec{r}_i + \frac{1}{2}\vec{R}|} + \mathrm{e}^{-\alpha|\vec{r}_i - \frac{1}{2}\vec{R}|} \end{aligned} \tag{7}$$

The parameter R that fixes the distance between the nuclei alters the energy of the system. We can not treat it as a variational parameter, since it changes the system itself, not only the trial function. We could use another R in our ansatz, but we keep it the same as in the Hamiltonian. We later look how the ground state energy depends on the distance.

2 Monte Carlo Integration

2.1 Variational Monte Carlo

We calculate the integral by taking samples on positions, that are proposed by a random walker and checked against the propability distribution given by $|\Psi|$. A step from an old position to a new position is given by $\vec{r}_{\text{new}} = \vec{r}_{\text{old}} + t \cdot \vec{\eta}$, where t is a steplength, that has to be fixed to a reasonable value and $\vec{\eta}$ is a vector with random entries from the interval $(-\frac{1}{2}, \frac{1}{2})$. The move to the new position is accepted if

$$\frac{\Psi(\mathbf{r}_{\text{new}})}{\Psi(\mathbf{r}_{\text{old}})} < \chi, \tag{8}$$

for a random number $\chi \in [0, 1]$. The only quanitites we have to calculate are thus the ratio of wave functions and the local energy, both independet of the normalization of the wave function.

2.2 Importance Sampling

Here we change the way, we calculate a step and decide about the acceptance of a new position. For this we introduce the quantum force, that is defined by

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

This is again independet of the normalization. The suggested move to a new position is then given by

$$\vec{r}_{\text{new}} = \vec{r}_{\text{old}} + \sqrt{t}\cdot\vec{\eta}_{\text{g}} + \frac{1}{2}t\cdot F(\vec{r}_{\text{old}}) \tag{10}$$

Here $\vec{\eta}_{\text{g}}$ is filled with normal distributed random numbers with a variance of 1. We move only one particle and decide then if we accept the new position or keep the old one instead. The new position is accepted if

$$\frac{G(\mathbf{r}_{\text{old}},\vec{r}_{\text{new}},\Delta t)}{G(\mathbf{r}_{\text{new}},\vec{r}_{\text{old}},\Delta t)}\cdot\frac{\Psi(\mathbf{r}_{\text{new}})}{\Psi(\mathbf{r}_{\text{old}})} < \chi. \tag{11}$$

χ is again a random variable from the uniform distribution on the unit intervall $[0,1]$. The additional factor contains the Greeens function, given by

$$G(\vec{x},\vec{y},\Delta t) \propto \exp\left(-\frac{(\vec{y}-\vec{x}-D\Delta t F(\vec{x}))^2}{4D\Delta t}\right) \tag{12}$$

where we dropped constant prefactors since we are looking just for the ratio. After this has been done for all particles we evaluate the local energy.

2.3 Derivatives of Ratios

We can calculate closed form expressions for the local energy instead of calculating the derivatives numerically. We get

$$\epsilon = \sum_{i=1}^{N} -\frac{Z}{r_i} + \frac{\Delta_i|D|}{|D|} + \frac{\Delta_i\Psi_C}{\Psi_C} + 2\cdot\frac{\nabla_i|D|}{|D|}\cdot\frac{\nabla_i\Psi_C}{\Psi_C} + \sum_{j<i}\frac{1}{r_{ij}} \tag{13}$$

To increase the speed of the calculations, there are ways to avoid the calcuation of the Slater Matrix and its determinant every time we want to know the value of the function, for example after changing the postion of only one particle. Because we are often interested in ratios of wave functions, we can use another procedure. Therefore we the following relation for the the inverse of the Slater matrix D^{-1}, using the minors C_{ij}:

$$D^{-1}_{ji} = \frac{C_{ij}}{|D|}, \quad D^{-1}\cdot D = D\cdot D^{-1} = \mathbf{1}. \tag{14}$$

Putting this in the Laplace expansion of the determinant, we can write in situations, where the only one particle is moved the ratio as

$$\frac{\Psi(\mathbf{r}_{\text{new}})}{\Psi(\mathbf{r}_{\text{old}})} = \sum_i \phi_j(\vec{r}_{\text{new}})D^{-1}_{ji} \tag{15}$$

by expanding along the column with the new position and noting that then $C_{ij}(\text{old}) = C_{ij}(\text{new})$. This is done only for the part of the Slater determinant that contains the moved electron, either the spin up part or the spin down part. In a similar way we get for the expressions containing derivatives

$$\frac{\nabla_i \Psi}{\Psi} = \sum_j \left(\nabla_i \phi_j(\vec{r_i})\right) D_{ji}^{-1} \tag{16}$$

and similar for the laplacian. As before we calculate this only for the part containing particle i, since the derivative is not acting on the other determinant, a great advantage of having a factorized wave function as used above.

2.4 Parameter Optimization

As mentioned earlier, the energy calculated using the trial wave function sets a upper limit to the ground state energy. Therefore we want to set the parameters such, that we minimize the energy, knowing that this brings us closer to the ground state energy. This brings us to the problem of minimizing multiple parameters at the same time, where the energy depends non-linearily on those. The simplest approach would be to take the gradient in parameter space and go with a steplength proportional to the gradient into the opposite direction. This is known as steepest descend. However this has a few disadvantages, it converges slowly, since it is possible to jump over the minimum point several times and it becomes very sensitive to noise close to the minimum. To compensate for the later point we modify this method to the so called 'stochastic gradient' approach. Here we reduce the proportionality factor between gradient and steplenght for every step. The next set of parameters $\vec{\alpha}_n$ is then given by the recursive relation

$$\vec{\alpha}_{n+1} = \vec{\alpha}_n - k_n \cdot \nabla \vec{\alpha}_n. \tag{17}$$

(k_n) is a positive sequence that has to fullfill

$$\sum_{i=0}^{n} k_i \overset{(n\to\infty)}{\longrightarrow} \infty, \qquad \sum_{i=0}^{\infty} k_i^2 < \infty. \tag{18}$$

The first property ensures we can reach each point in parameter space, the second one tells us that the steps get smaller, so the procedure converges when multiplied with the gradient, who is supposed to get smaller as well. In practice we take only a finite number of steps. There are several choices for (k_n), including ones, that change the sequence dynamically whenever the gradient changes direction. The simplest possible choice would be $k_n = \frac{1}{n}$. This converges quite slow, because the step size is decreased drastically after only a few steps. Therefore we chose a modification, namely $k_n = \frac{A}{B+n}$. The parameters A and B can be chosen such, that we have a bigger stepsize in the beginning and the long term behavior of $\frac{1}{n}$. Depending on the inital values, $A = B = 50$ seems to be a good choice for our application. The gradient is calculated by Monte Carlo integration, but using much fewer samples than usually, getting therefore a result with big statistical fluctuations. Those cancel on average, and the multiplication with k_n prevents from jumping to far away again.

2.5 Data Analysis using Blocking

Since Monte Carlo integration is a statistical approach, we are interested in the statistical uncertainty of our result. As a first approach one can calculate

$$\Delta E = \left\langle (E - \langle E \rangle)^2 \right\rangle^{\frac{1}{2}} , \tag{19}$$

but this formula assumes uncorrelated data. But our samples are correlated due to the fact that we get them from a random walker. Therefore a simple use of the standard deviation would underestimate our error. However, there are not all samples correlated, they can be seen as independet after taking some steps. It is to expensive to calculate this so called correlation length, but we group them togheter by calculating the average over blocks with varying sizes. We put these averages in equation 19 and get an estimate for the error dependent on the block size. Doing this we expect the estimated error to grow, until the block size reaches the correlation length, where it reaches a plateau. The statistical uncertainty is given by the value of this plateau.

3 Programm Stucture

The programs source code and all necessary files can be found in the repository on GitHub: github.com/AndreasLeonhardt/Proj1.git. The structure of the program and of the most important function, mcint::integrate is shown in the flow charts 3 and 3.

3.1 Classes

The program is build on various classes in order to keep it modular and be able to switch between different scenarios. Members were kept private in all classes, but there are functions to read and write them if this is needed or useful.

3.1.1 positions

This class holds the information about the position of the electrons. The main part is a matrix, that contains all the vectors of the single electrons. Since we often just use the distance to the nucleus or between them, an object of this class hols additional vectors with these values, that are updated each time a particle moves. For numerical derivatives it contains a step function, to move a particle in one spatial direction.

3.1.2 function

This class is implemented as a virtual class, so one can define different classes that inherit all the properties. In this way it is easyer to switch between different functions. The classes that inherit from “function”, here calles “TrialFct” or similar define the actual trial fct. The most important functions take an object of the “positions” class and return the value of the wave function or some functions build of the wave function and its derivatives, among those the quantum force F, the gradient over the function, the parameter derivative over the function and for two different postions the ratio of Slater determinants and Jastrow factors.

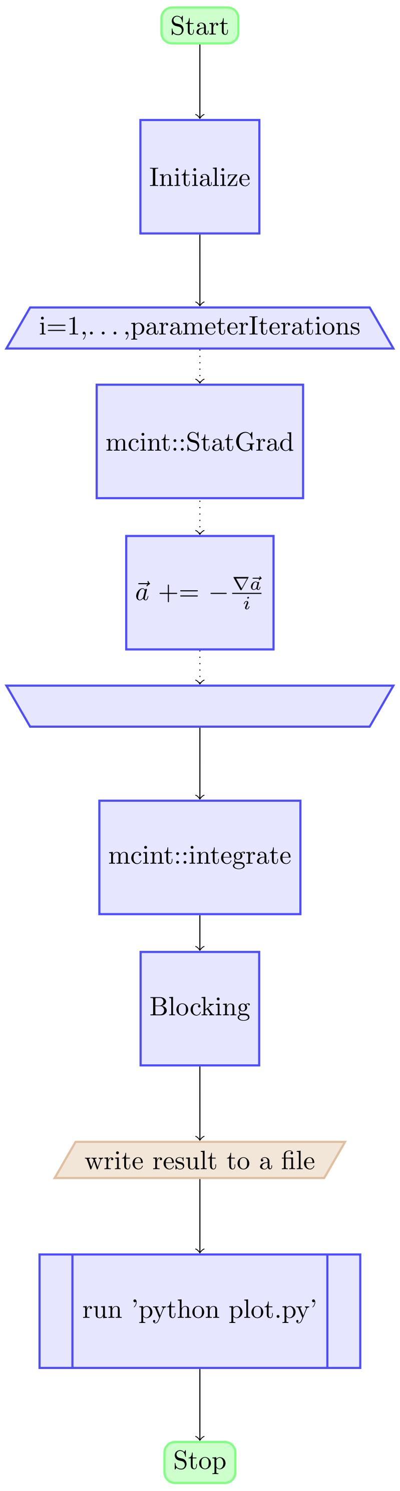
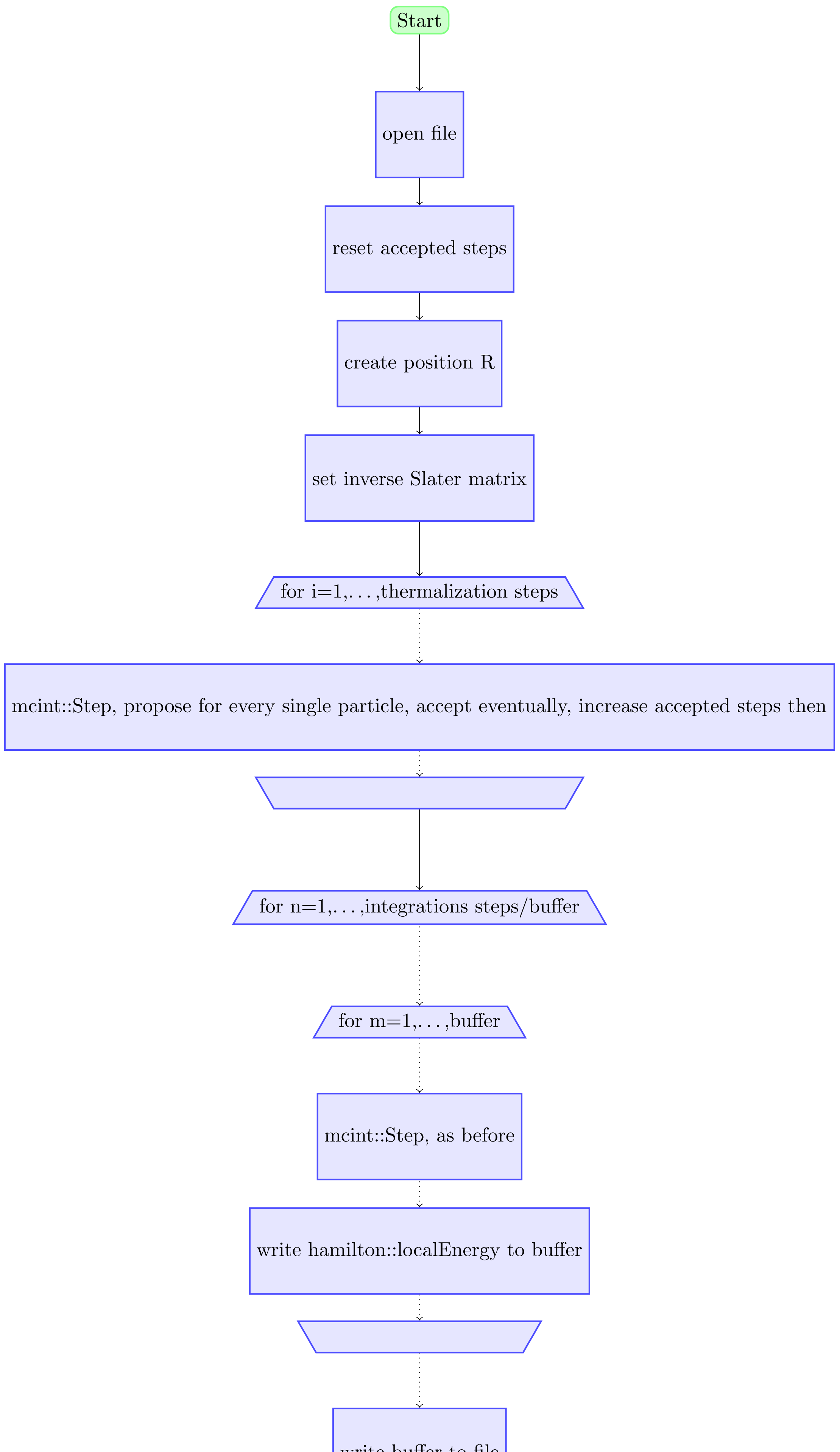


Figure 1: flow chart of main()


Start
open file
reset accepted steps
create position R
set inverse Slater matrix
for i=1,...,thermalization steps
mcint::Step, propose for every single particle, accept eventually, increase accepted steps then
for n=1,...,integrations steps/buffer
for m=1,...,buffer
mcint::Step, as before
write hamilton::localEnergy to buffer
write buffer to file

In addition it contains functions that update the inverse Slater matrix. These are sometimes empty functions, if this matrix is not used.

3.1.3 hamilton

Here we define only the Hamiltonian, of more precisely the energy density. In earlier versions there were two different versions, one that used numerical derivatives and one that had an analytical version. This was then moved inside the function class.

3.1.4 mcint

The "mcint" class contains the functions, that actually perform the calculations. The most important ones are the "step" function, that proposes a new position, based on an old one, and decides if it is accepted. This is used repeadetly in the thermalization and integration. "integrate" is another function, that first thermalizes according to number of steps in the parameter file, and than takes samples and writes them blockwise to a file. Afterwards these files are processed in the "blocking" function. There is a function, "StatGrad", that calculates derivatives of the energy with respect to the variational parameters. This is used in the parameter optimization via the stochastical gradient method. This is done by a Monte Carlo integration over only a few samples, about 1000. The derivative is calculated according to the formula

$$\partial_\alpha E[\alpha] = 2 \cdot \left\langle \hat{H} \frac{\partial_\alpha \Psi}{\Psi} \right\rangle - 2 \cdot \left\langle \hat{H} \right\rangle \left\langle \frac{\partial_\alpha \Psi}{\Psi} \right\rangle, \tag{20}$$

which an be found by tedious but straightforward calculation and holds for all parameters, as long as $\partial_\alpha \hat{H} = 0$. The result is much faster and more accurate than a numerical derivation would be, especially on data with statistical errors.

3.1.5 libconfig

Parameters are stored in a seperate configuration file, "parameters.cfg". To get a parameter the "libconfig[1]" class is used. The parameter has to be written in the format

```
keyword = parameter;
```

The parameter can then be looked up by a instance of the configuration class given the keyword as a string. The data type is detected automatically. The file contains all parameters like the charge of the nuclei, some initial values for α and β, the number of cycles for thermalization, integration, optimization and so on.

3.2 Structure

The Monte Carlo integration is function of the "mcint" class, that also has a function to suggest a new step and test it. It takes a function and a position from the classes "function" and "postions". A position contains the vectors of the particles, the length of the position vectors and the distance between them.

[1] hyperrealm.com/libconfig

The later ones are updated ones the position of one or all particles is changed. A object from the function class has several functions that return the value of the function at a given position, but also the gradient or the laplace of the function divided by the function, expressions we need often as seen above. In this branch is the local energy a class for itself, called "hamilton". This function is in later versions included in the function class. The function and hamilton classes are virtual, the subclasses implement the derivatives numerically or analyticaly. In this branch is the analytical version just defined for two particles. Parameters are stored in a seperate file, that is read in in the beginning, using the libconfig class.

The programm is in principle working the same way as in the previous case. The program is found at top on the master branch on the same repository as above, github.com/AndreasLeonhardt/Proj1.git. The local energy is now a function of the "function" class. In the numerical case the derivatives and ratios are calculated straight forward, what results in the Slater determinant beeing calculated many times. The analytical cases makes use of the simplifications described above and has to keep track of the inverese Slater matrix.