Tutorial

This section describes everything needed to set up the code and run the code for simple systems contained in the tutorial.

Installing/Running the code

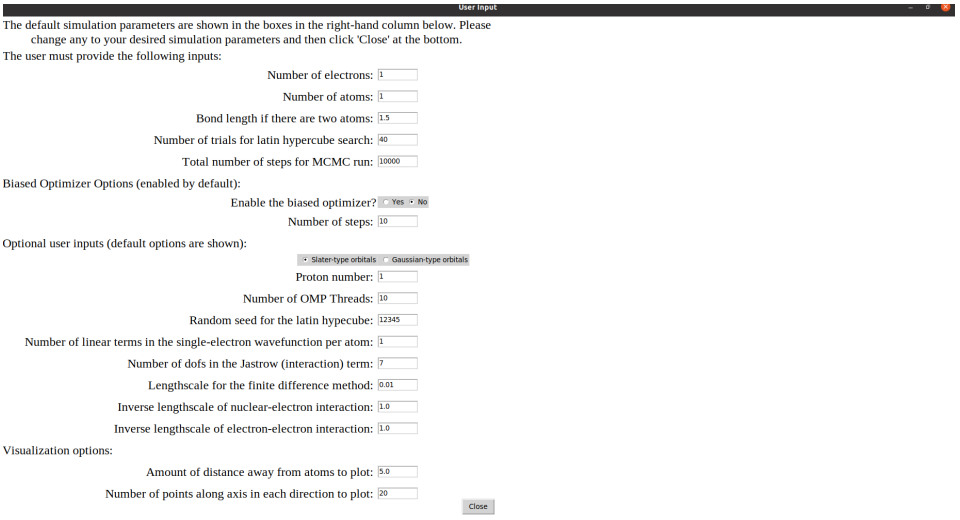
This code is compiled and ran in the terminal. YOu can build the code by running the following command for the bash script:

./build_latin_driver.sh

After that has successfully compiled, you can now run the following command code to execute the bash program:

./run_latin_driver.sh

Then the following interface will appear as shown in the figure below. This interface contains all of the variables which the user can change to define the system they want to look at. The interface already contains default values for the variables which the user can keep or change if they wish.



In order to then run the code, the user just needs to change the varibales they wish to change and then click close. This will start running the code with the progress and results displayed in the terminal. When the code has finished running, the output plots will automatically be generated and show each plot in a different pop-up window.

To complete another run of the code, make sure that all the ouputting windows containing the plots are closed, and then re-run the ./run_latin_driver.sh command. There is no need to recompile the code.

In the case that the code fails to run or has crashed after this step, you can easily exit the software by either closing the terminal or using Ctrl+c.

In order to demonstrate the varying functionality of the code, 4 problems have been defined in the subsections below for the user to try. There maybe slight differences in the quoted results due to the sampling and approximate numerical nature of the code, but none of the examples should obtain results which stray far from the results quoted here.

To Improve the accuracy of any the simulations, try increasing the number of trials and the number of steps in the MCMC runs. For 2 electron problems try increasing the number of Jastrow terms from 7 to 9.

Problem 1: Single atom and electron using Slater-type Orbitals

This part of the tutorial contains the most basic system consisting of 1 atom (nuclei) and 1 electron which represents a Hydrogen atom, and is described using Slater-type orbitals. In this example, the Slater-type orbitals provides the analytical result.

All the user has to do is close the user input window as this simulation run uses the default values already provided.

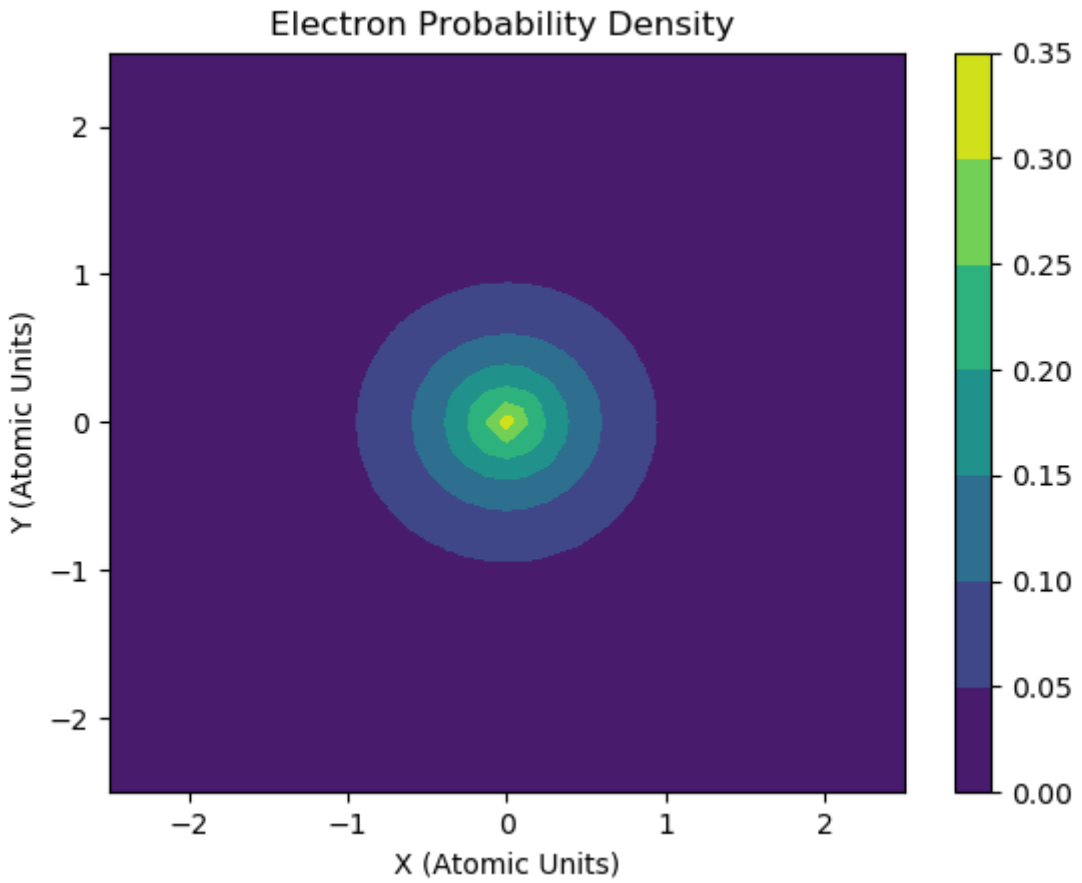
Once the user input window has been closed, the following outputs are printed in the terminal:

- The user defined input values are printed for reference
- Each latin hypercube trial and its corresponding energy result is printed

If the code has ran with no errors or crashes, the following will be printed everytime:

- The best latin hypercube trial
- The minimum energy found in both Hartree and eV units
- The best parameters (this is the optimal degrees of freedom found)
- Comparison of output density statement
- Success in writing output files results.nc4 and xyz.txt
- Total cpu runtime
- Total real runtime
- Upload of result.nc4 successfully
- Print of what is contained in the result.nc4 file
- The number of electrons, atoms

The output plotting script will automatically run, producing the following contour plot of the electron probability density.



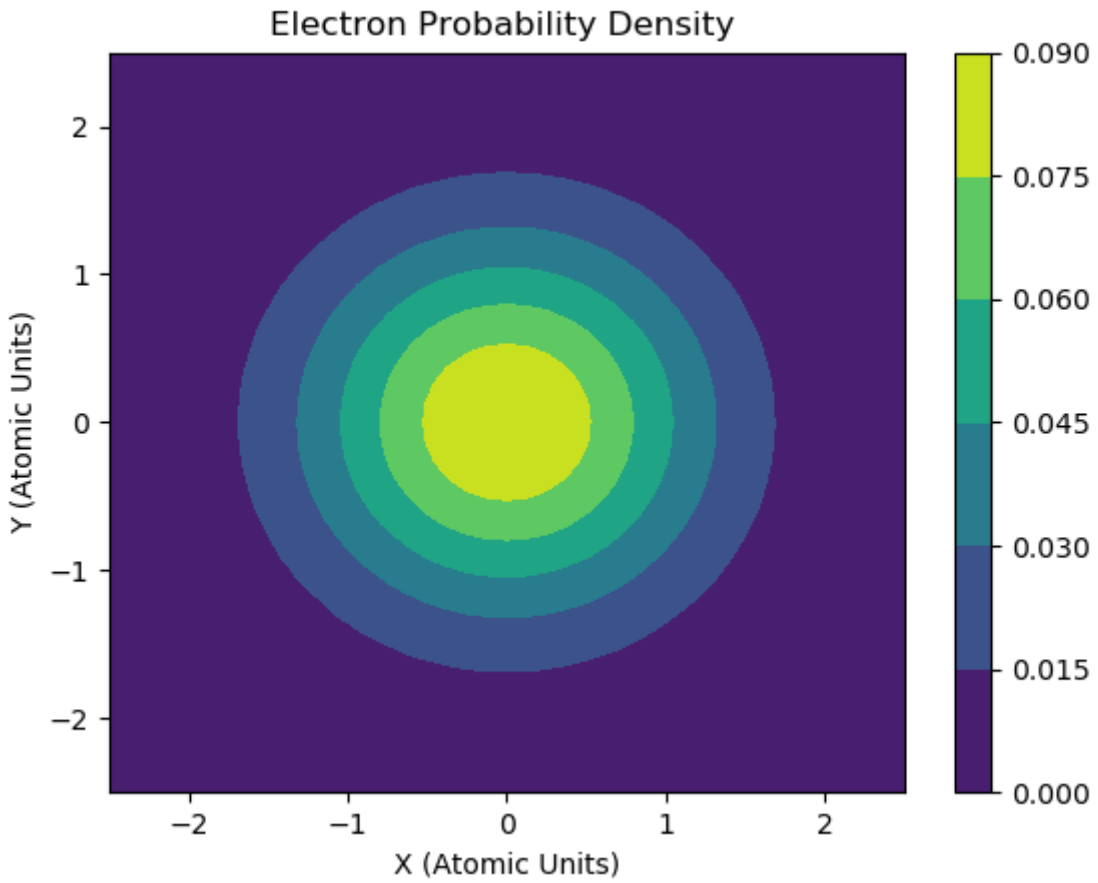
The minimum energy obtained for this system is -13.6 eV and -0.499 Hartree energy. The optimal degrees of freedom obtained is -1.42 and 0.997.

The code should take no longer than 3 seconds to run and complete.

Problem 2: Single atom and electron using Gaussian-type Orbitals

This problem is the same set up as the previous problem, however the user instead chooses Gaussian-type orbitals instead of Slater-type orbitals to describe the Hydrogen atom. Also, as the Gaussian-type orbitals only give an approximate result, the total number of steps for MCMC run is increased from 1000000 to 10000000.

This simulation results in the following contour plot as shown in the figure below for the electron probability density.



The energy obtained for this run is -11.8 eV and -0.434 Hartree energy. The energy results for this run may vary as it is an approximate result but the result the user obatins should not stray far from this value due to the increased MCMC steps.

This result is not far from the analytical result

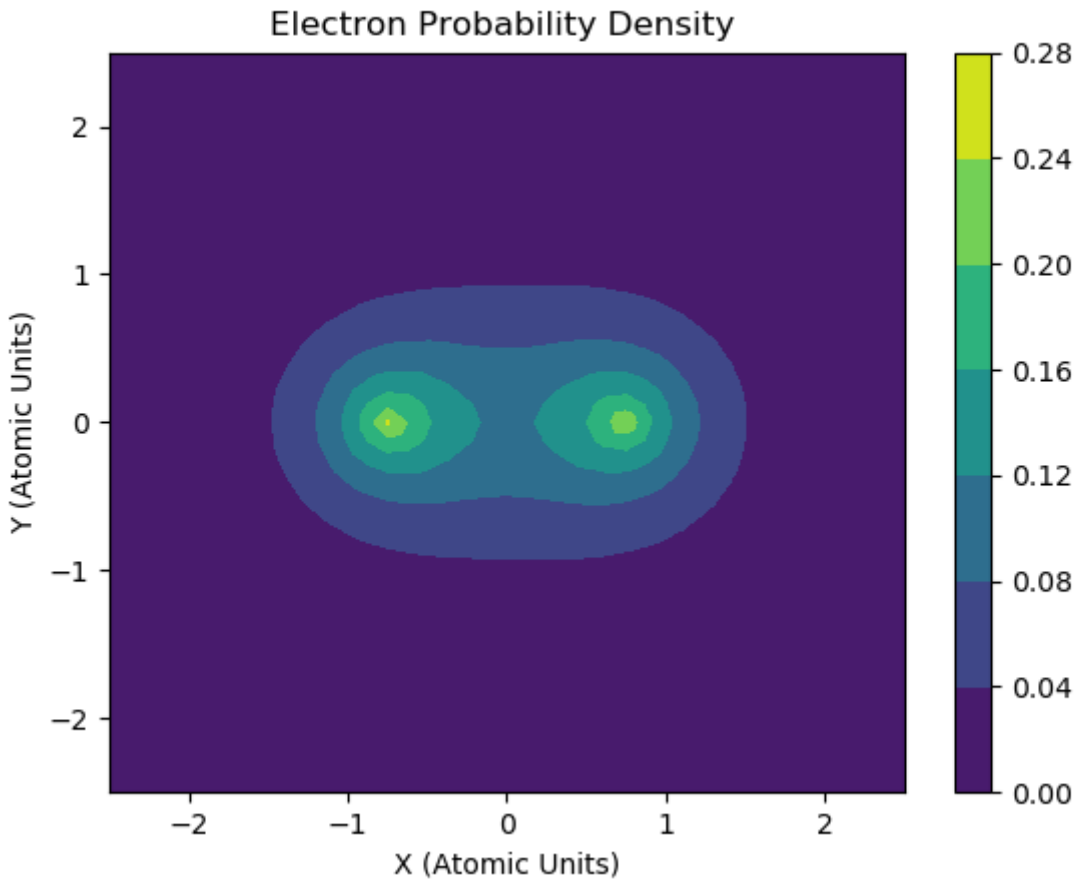
Due to the higher number of MCMC steps defined, this code will take slightly longer than the run completed in Problem 1. The code should not take longer 30 seconds.

Problem 3: Two atoms and a single electron

This problem investigates two atoms (or nuclei) and a single electron system. This corresponds to the H2+ ion.

The output of this system not only produces a contour plot of the electron probability density, but also allows for the evaluation of energy for varying bond lengths.

To first obtain the contour plot, the user uses the same procedure as previously defined in running the scripts, keeping all of the defualt values except the number of atoms is being changed from 1 to 2. Slater-type orbitals are used. The following contour plot should be obtained.



The minimum energy found for this system should be approximately -0.55 Hartree or -15.2 eV.

This code run should take no more than 5 seconds to run.

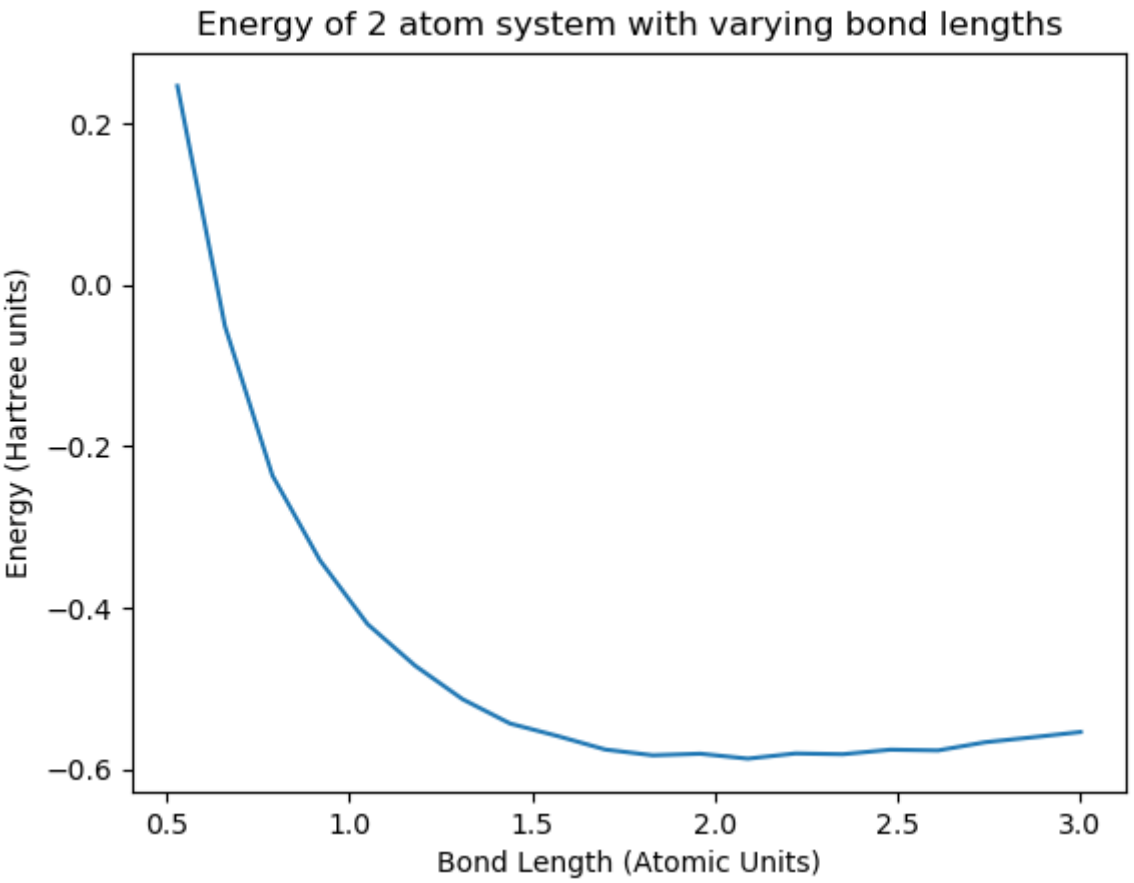
In order to obtain the energy against varying bond length plot, the user must run the following commands in the terminal:

./build_bond_driver.sh

Then run

./run_bond_driver.sh

Which will present the user input interface used previously. The only thing the user needs to change is the number of atoms from 1 to 2. The difference with this run is that the contour plot will not be outputted, only the energy against varying bond length will be plotted. The resulting output plot is shown in the figure below.



In the terminal, when the code has finished running, the minimum energy and its corresponding bond length is outputted. For this example, the bond length with the minimum energy was found to be 2.09 atomic units, with -0.586 Hartree energy.

The bond length driver is currently a proof-of-concept. The parameters for the search over bond length are hardcoded, but can be changed on lines 168-170 of **bond_driver.f90** file.

This code run should take no more than 75 seconds to fully run.

If the user wants to revert back to obtaining the contour plots, they will need to repeat the following commands in the terminal:

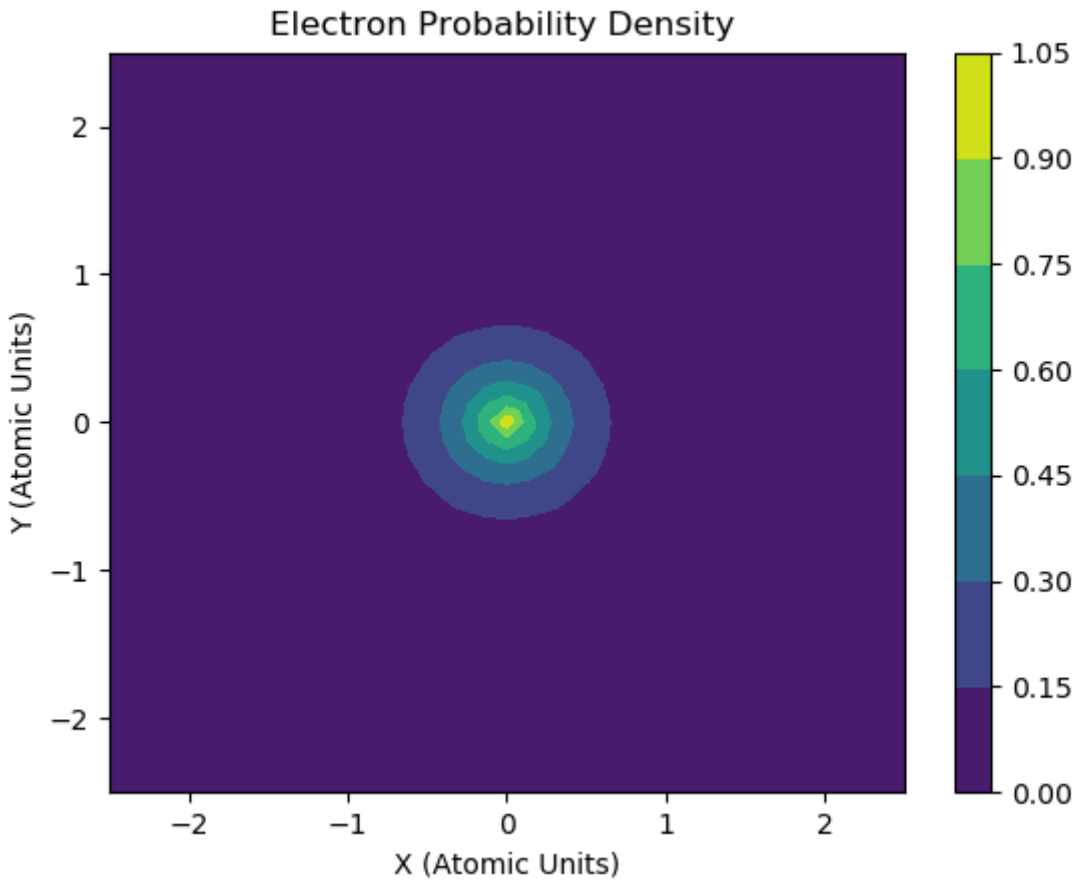
./build_latin_driver.sh

./run_latin_driver.sh

Problem 4: Helium+ ion and Helium atom

To simulate a Helium+ ion, 1 atom and 1 electron is simulated but the proton number is changed from 1 to 2, keeping all of the default variables the same for this case.

This results in the following contour plot for the electron density:



The minimum energy obtained is -1.87 Hartree energy or -50.9 eV. This is a good result when compared to expected value of -54.42 eV as the Slater-type orbitals uses hydrogen-like solutions. When using Gaussian-like orbitals, the resulting minimum energy found is -11.6 eV which does not agree well with the expected value.

For the Helium atom, the same procedure is done however the number of electrons is changed from 1 to 2 and the number of protons is changed from 1 to 2 from the default values.

The resulting minimum energy for the Helium atom for Slater-type orbitals was found to be -77.9 eV which is close to the expected value of -79.02 eV. When using Gaussian-like orbtials, the resulting minimum energy was found to be -69.5 eV which is slightly below the expected value, however as it only gives an approximate result, increasing the number of MCMC steps and latin hypercube trials should improve its accuracy.

For the 2 electron case, the contour plot for the resulting electron probability density will be outputted, however the results are wrong and should be ignored.

Using the default number of MCMC steps and latin hypercube trials, the runs should take no more 30 seconds to run each.

User options beyond the tutorial

The user can change the following variables if they want to look at 1 atom only:

- Number of electrons
- Number of atoms

- Number of trials for latin hypercube search
- Number of MCMC steps
- Optional user inputs (either Slater-type or Gaussian-type orbitals)

To investigate 2 atoms, all of the variables defined in the user input interface are relevant.

Plotting Options

These variables allow the user to change aspects of the resulting output such as the domain size and the resolution.

- Amount of distance away from atoms to plot
- Number of points in each direction

Increasing these options will cause the code to take slightly longer to run but will obtain better resolution images as a result.

Analyzing the results

Results are saved using the subroutine create_netcdf and are automatically plotted using either the **plotting.py** script or the **energy_plotting.py** script.

Note the user does not actually need to have python or a python IDE open in order for the plotting to appear as it does so automatically through a pop up window. The user also has the option to save the resulting image by clicking the save button with is the button on the far right at the bottom of the window.

Warnings

If the following warning has appeared in the print statements in the terminal: WARNING: Acceptance rate 0 at: 1

The user can ignore this as the code will still run and produce an output. This error relates to an unsuitable initial condition, but unless this error appears later than step 1 then there is no problem, and the MCMC has corrected itself.

Advanced Options

How to add a basis set

It is relatively easy to add a basis set to the program. Two new functions defining the corresponding single electron wavefunction and reduced hamiltonian must be added to **basis_functions.f90**, following the format of **wave_function_slater_1s()** and **reduced_hamiltonian_slater_1s()**. Pure subfunctions similar to **centered_slater_1s_laplacian()** can be added to **component_functions.f90**.

An integer code for the basis must be added to **constants.f90**. A corresponding check must be added to the case selection on **basis_functions::basis_type** in the initialisation routine (currently line 198 in **basis_functions.f90**). This must point the function pointers `wave_function_single` and `reduced_hamiltonian` to the new functions.

Future extensions

The following additions and extensions may be added to the code.

- Diffusion Monte Carlo
- Other elements