

Variational Monte Carlo for the Helium Atom

The Helium Atom

The hydrogen atom is a two-body system consisting of a proton and an electron. If spin and relativistic effects are ignored, then the Schrödinger equation for the hydrogen atom can be solved exactly. In this respect, it is similar to the Kepler problem in classical mechanics.

The classical three-body problem is nonintegrable and exact solutions exist only in very special cases. The analogous quantum mechanical system is the Helium atom, which consists of a nucleus and two electrons. This quantum mechanical problem cannot be solved exactly even if spin and relativistic effects are ignored.

The review article[1] gives an overview of an enormous amount of theoretical and numerical work that has been done on the Helium atom since the discovery of quantum mechanics almost 100 years ago.

TABLE I. Ground-state energies (in a.u.) of the helium atom: left, semiclassical; right, quantum-mechanical and experimental. The good agreement of the energy obtained from the Heisenberg-Sommerfeld model (see Fig. 3) must be considered as accidental (see text). The given result from Solov'ev's approach was extracted from Fig. 4 of Solov'ev (1985). The asymmetric stretch orbit (Ezra *et al.*, 1991) is shown in Fig. 13(b), Sec. IV.B. The semiclassical cycle expansion is described in Sec. IV. The theoretical data do not include finite-mass, relativistic, or QED corrections, except for the result by Drake (1993), which contains relativistic effects. The difference between the latter result and the latest experimental figures (Bergeson *et al.*, 1998) reflects QED effects. Some of the data are from Leopold and Percival, 1980, Table I.

Year	Semiclassical Method	$-E$	Year	Quantal/Experimental Method	$-E$
1913	orbit Fig. 2(a), Bohr (1913)	3.06	1927	first order pert. Unsöld (1927)	2.75
1919	orbit Fig. 2(c), Landé (1919)		1927	molecularlike, Slater (1927)	2.895
1921	orbit Fig. 2(b), Langmuir (1921)	2.17	1927	variational, Kellner (1927)	2.873
1921	orbit Fig. 2(d), Langmuir (1921)	2.31	1928	variational, Hylleraas (1928)	2.895
1922	"hybrid orbit," Van Vleck (1922)	2.765	1929	var., 38 param., Hylleraas (1929)	2.9037
1922	orbit Fig. 3, Heisenberg (1922)	2.904	1959	var., 38 param., Kinoshita (1959)	2.903 722
1923	orbit Fig. 2(e), Kramers (1923)	2.762	1959	perimetric coord., Pekeris (1959)	2.903 724 376
1980	1. order pert., Leopold <i>et al.</i> (1980)	2.7410	1988	Hylleraas type basis, Drake (1988)	2.903 724 377 03415
1980	variational, Leopold <i>et al.</i> (1980)	2.8407			
1985	perturb. theory, Solov'ev (1985)	3.05	1995	perimetric coordinates, Bürgers <i>et al.</i> (1995)	2.903 724 377 034 119 589
1991	as. stretch orbit, Ezra <i>et al.</i> (1991)	3.097	1993	relativ. Drake (1993)	2.903 700 023
1991	cycle expansion, Ezra <i>et al.</i> (1991)	2.932	1924	experimental, Lyman (1924)	2.9035
			1998	exp., Bergeson <i>et al.</i> (1998)	2.903 693 775

Table 1: From the review article by Tanner *et al.*[1]. The ground state energy is expressed in atomic units $e = \hbar = m_e = 1$. In these units, the ground state energy of the hydrogen atom is -0.5 .

Variational Monte Carlo for the Hydrogen Atom

Before tackling the Helium problem it will be instructive to apply the variational Monte Carlo method to the hydrogen atom.

The Hydrogen atom is a system with two particles, electron and proton. The configuration space in which the system moves is therefore six dimensional. By moving to the center-of-mass system, the problem

becomes effectively 3 dimensional, with Hamiltonian

$$H = -\frac{\hbar^2}{2m}\nabla^2 - \frac{e^2}{r}, \quad (1)$$

where $\mathbf{r} = \mathbf{r}_e - \mathbf{r}_p$ is the relative coordinate of the electron with respect to the proton, e is the magnitude of the electron's charge, and $m = m_e m_p / (m_e + m_p)$ is the reduced mass.

By using conservation of angular motion and the fact that the ground state is spherically symmetric, i.e., it has zero orbital angular momentum, the problem can be reduced to one dimension with Hamiltonian operator

$$H = -\frac{\hbar^2}{2m} \left[\frac{d^2}{dr^2} + \frac{2}{r} \frac{d}{dr} \right] - \frac{e^2}{r}, \quad (2)$$

which depends on the radial coordinate r .

The exact ground state energy and wavefunction are given by

$$E_0 = -\frac{e^2}{2a_0}, \quad \psi_0(r) \sim e^{-r/a_0}. \quad (3)$$

where the *Bohr radius*

$$a_0 = \frac{\hbar^2}{m e^2}. \quad (4)$$

It is convenient to use *atomic units* in which $\hbar = m = e = 1$ so

$$H = -\frac{1}{2} \left[\frac{d^2}{dr^2} + \frac{2}{r} \frac{d}{dr} \right] - \frac{1}{r}, \quad E_0 = -\frac{1}{2}, \quad \psi_0(r) \sim e^{-r}. \quad (5)$$

A simple trial wave function for the Hydrogen atom ground state is

$$\psi_{T,\alpha}(r) = e^{-\alpha r}. \quad (6)$$

The local energy for this choice can easily be computed:

$$E_L(r) = \frac{1}{\psi_{T,\alpha}} H \psi_{T,\alpha}(r) = -\frac{1}{2} \left[\alpha^2 - \frac{2\alpha}{r} \right] - \frac{1}{r}. \quad (7)$$

Note two important points about this local energy:

- It is minimum and also independent of r at $\alpha = 1$, which gives the exact ground state energy and eigenfunction.
- For $\alpha \neq 1$ it is *singular* at $r = 0$ where the potential diverges. For more complex problems, like the Helium atom to be considered next, these singularities can cause problems with the numerical calculation. To deal with these singularities, *cusp conditions* are used to restrict the variational parameters.

The harmonic oscillator program can be adapted to reproduce these results by changing the form of the trial wave function and local energy. However, there are two essential changes that must be made:

- Since $r \geq 0$, a one-dimensional Metropolis walker should not be allowed to cross the origin to $r < 0$.
- Also, the probability that the one-dimensional walker is found between r and $r + dr$ must be proportional to $4\pi r^2$, which is the surface area of a sphere of radius r .

To proceed, use a walker in 3 dimensional space. Given a walker position \mathbf{r} and a maximum step size δ , the next trial step is chosen uniformly at random within a cube of side 2δ centered on the point \mathbf{r} and aligned with the coordinate axes. This actually solves both of the problems above at the expense of making three calls to the random number generator for each trial move.

_____ Program 1: <http://www.physics.buffalo.edu/phy410-505/topic5/vmc-h.cpp> _____

```
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <vector>
using namespace std;

#include "linalg.hpp"
#include "random.hpp"
using namespace cpl;

Random rg;           // random number generator object

int dimensions = 3;  // dimensionality of space
int electrons = 2;  // number of electrons
int N;              // number of walkers
vector<              // vector of walkers
    vector<          // vector of electrons
        vector<double> > > r; // vector of coordinates

double alpha;       // Pade-Jastrow variational parameter
double delta;       // trial step size

void initialize() {  // function to initialize the calculation

    // initialize random number generator
    rg.set_seed_time();
    cout << " using " << rg.get_algorithm()
         << " and seed " << rg.get_seed() << endl;

    // initialize the positions of the walkers
    for (int n = 0; n < N; n++) {
        vector< vector<double> > exyz;
        for (int e = 0; e < electrons; e++) {
            vector<double> xyz;
            for (int d = 0; d < dimensions; d++)
                xyz.push_back(rg.rand() - 0.5);
            exyz.push_back(xyz);
        }
        r.push_back(exyz);
    }

    // set initial step size
    delta = 1;
}
```

```

double e_sum;           // accumulator for energy average
double e_sqd_sum;      // accumulator for variance in energy

void zero_accumulators() { // set accumulators to zero
    e_sum = e_sqd_sum = 0;
}

double Psi(             // compute trial wave function
    vector<double>& r_electron_1,
    vector<double>& r_electron_2
) {

    // value of trial wave function for walker n
    double r1 = 0, r2 = 0, r12 = 0;
    for (int d = 0; d < dimensions; d++) {
        r1 += r_electron_1[d] * r_electron_1[d];
        r2 += r_electron_2[d] * r_electron_2[d];
        r12 += (r_electron_1[d] - r_electron_2[d])
            * (r_electron_1[d] - r_electron_2[d]);
    }
    r1 = sqrt(r1);
    r2 = sqrt(r2);
    r12 = sqrt(r12);
    double Psi = - 2*r1 - 2*r2 + r12 / (2 * (1 + alpha*r12));

    return exp(Psi);
}

double e_local(        // compute contribution to local energy
    vector<double>& r_electron_1,
    vector<double>& r_electron_2
) {

    // value of trial wave function for walker n
    double r1 = 0, r2 = 0, r12 = 0;
    for (int d = 0; d < dimensions; d++) {
        r1 += r_electron_1[d] * r_electron_1[d];
        r2 += r_electron_2[d] * r_electron_2[d];
        r12 += (r_electron_1[d] - r_electron_2[d]) *
            (r_electron_1[d] - r_electron_2[d]);
    }
    r1 = sqrt(r1);
    r2 = sqrt(r2);
    r12 = sqrt(r12);
    double dot_prod = 0;
    for (int d = 0; d < dimensions; d++) {
        dot_prod += (r_electron_1[d] - r_electron_2[d]) / r12 *
            (r_electron_1[d] / r1 - r_electron_2[d] / r2);
    }
    double denom = 1 / (1 + alpha * r12);
}

```

```

double denom_2 = denom * denom;
double denom_3 = denom_2 * denom;
double denom_4 = denom_2 * denom_2;
double e = - 4 + alpha * (denom + denom_2 + denom_3)
           - denom_4 / 4 + dot_prod * denom_2;

return e;
}

int n_accept;           // keep track of number of accepted steps

void Metropolis_step(   // one Metropolis step
    int walker         // for this walker
) {

    // make a trial move of each electron
    vector<double> r_electron_1(dimensions), r_electron_2(dimensions),
        r_trial_1(dimensions), r_trial_2(dimensions);
    for (int d = 0; d < dimensions; d++) {
        r_electron_1[d] = r[walker][0][d];
        r_trial_1[d] = r_electron_1[d] + delta * (2 * rg.rand() - 1);
        r_electron_2[d] = r[walker][1][d];
        r_trial_2[d] = r_electron_2[d] + delta * (2 * rg.rand() - 1);
    }

    // Metropolis test
    double w = Psi(r_trial_1, r_trial_2) / Psi(r_electron_1, r_electron_2);
    if (rg.rand() < w * w) {
        for (int d = 0; d < dimensions; d++) {
            r[walker][0][d] = r_electron_1[d] = r_trial_1[d];
            r[walker][1][d] = r_electron_2[d] = r_trial_2[d];
        }
        ++n_accept;
    }

    // accumulate local energy
    double e = e_local(r_electron_1, r_electron_2);
    e_sum += e;
    e_sqd_sum += e * e;
}

void one_Monte_Carlo_step() {

    // do Metropolis step for each walker
    for (int n = 0; n < N; n++)
        Metropolis_step(n);
}

int main() {

    cout << " Variational Monte Carlo for Helium Atom\n"

```

```

        << " -----\n";
cout << " Enter number of walkers: ";
cin >> N;
cout << " Enter Pade-Jastrow parameter alpha: ";
cin >> alpha;
cout << " Enter number of Monte Carlo steps: ";
int MC_steps;
cin >> MC_steps;

initialize();

// perform 20% of MC_steps as thermalization steps
// and adjust step size so acceptance ratio ~50%
int therm_steps = int(0.2 * MC_steps);
int adjust_interval = int(0.1 * therm_steps) + 1;
n_accept = 0;
cout << " Performing " << therm_steps << " thermalization steps ..."
    << flush;
for (int i = 0; i < therm_steps; i++) {
    one_Monte_Carlo_step();
    if ((i+1) % adjust_interval == 0) {
        delta *= n_accept / (0.5 * N * adjust_interval);
        n_accept = 0;
    }
}
cout << "\n Adjusted step size delta = " << delta << endl;

// production steps
zero_accumulators();
n_accept = 0;
cout << " Performing " << MC_steps << " production steps ..." << flush;
for (int i = 0; i < MC_steps; i++)
    one_Monte_Carlo_step();

// compute and print energy
double e_ave = e_sum / double(N) / MC_steps;
double e_var = e_sqd_sum / double(N) / MC_steps - e_ave * e_ave;
double error = sqrt(e_var) / sqrt(double(N) * MC_steps);
cout << "\n <Energy> = " << e_ave << " +/- " << error
    << "\n Variance = " << e_var << endl;
}

```

Variational Monte Carlo for the Helium Atom

The Helium atom presents a 3-particle problem: two electrons orbit around a nucleus, which consists of two protons with charge e each and two neutral neutrons. The nucleus, which is $\sim 8,000$ times more massive than an electron, can be assumed to be at rest at the origin of the coordinate system. The

electrons have positions \mathbf{r}_1 and \mathbf{r}_2 . This is simpler than making a transformation to the center-of-mass system of the three particles, and it is sufficiently accurate.

If we use atomic units with $\hbar = m_e = e = 1$, the Hamiltonian for the motion of the two electrons can be written

$$H = -\frac{1}{2}\nabla_1^2 - \frac{1}{2}\nabla_2^2 - \frac{2}{r_1} - \frac{2}{r_2} + \frac{1}{r_{12}}, \quad (8)$$

where $r_{12} = |\mathbf{r}_{12}| = |\mathbf{r}_1 - \mathbf{r}_2|$. The terms $-2/r_i$ represent the negative (attractive) potential energy between each electron with charge -1 and the Helium nucleus with charge $+2$, and the term $+1/r_{12}$ represents the positive (repulsive) potential energy between the two electrons.

A simple choice of variational trial wave function

If the repulsive term $1/r_{12}$ were not present, then the Hamiltonian would be that of two independent Hydrogen-like atoms. It can be shown that the energy and ground state wave function of a Hydrogen-like atom whose nucleus has charge Z are given by

$$E_0 = -\frac{Z^2}{2}, \quad \psi_0 \sim e^{-Zr}. \quad (9)$$

The wave function of the combined atom with two non-interacting electrons would be the product of two such wave functions:

$$\psi(\mathbf{r}_1, \mathbf{r}_2) \sim e^{-2r_1} e^{-2r_2}. \quad (10)$$

This suggests a trial wave function of the form

$$\Psi_{T,\alpha} = e^{-\alpha r_1} e^{-\alpha r_2}, \quad (11)$$

similar to what was done for the Hydrogen atom.

If the electron-electron interaction is neglected, then the average energy with this wave function can be calculated

$$\left\langle -\frac{1}{2}\nabla_1^2 - \frac{1}{2}\nabla_2^2 - \frac{2}{r_1} - \frac{2}{r_2} \right\rangle = 2 \times \frac{\alpha^2}{2} - 2 \times \alpha, \quad (12)$$

which has a minimum at $\alpha = 1$, which gives $\langle E \rangle = -1$. The experimentally measured ground state energy is $E_0 = -2.904$.

In fact, the average energy can be evaluated exactly for this trial wave function even if the electron-electron interaction is included:

$$\left\langle -\frac{1}{2}\nabla_1^2 - \frac{1}{2}\nabla_2^2 - \frac{2}{r_1} - \frac{2}{r_2} + \frac{1}{r_{12}} \right\rangle = \alpha^2 - \frac{27}{8}\alpha, \quad (13)$$

which has a minimum at $\alpha = 27/16$, which gives $\langle E \rangle = -2.8477$.

This shows that the repulsion between the electrons is important and lowers the energy.

Padé-Jastrow wave function

A convenient choice of trial wave function is the Padé-Jastrow form:

$$\Psi(\mathbf{r}_1, \mathbf{r}_2) = e^{-2r_1} e^{-2r_2} e^{\frac{r_{12}}{2(1+\alpha r_{12})}}, \quad (14)$$

with α as a variational parameter.

The local energy is given by

$$E_L = \frac{1}{\Psi} \left[-\frac{1}{2} \nabla_1^2 - \frac{1}{2} \nabla_2^2 - \frac{2}{r_1} - \frac{2}{r_2} + \frac{1}{r_{12}} \right] \Psi. \quad (15)$$

Let's write

$$\Psi = f(r_1) f(r_2) g(r_{12}) \quad (16)$$

where $f(r) = e^{-2r}$ and $g(r) = e^{r/(2(1+\alpha r))}$. Then,

$$\begin{aligned} \nabla_1^2 f(r_1) g(r_{12}) &= (\nabla_1^2 f(r_1)) g(r_{12}) + f(r_1) (\nabla_1^2 g(r_{12})) \\ &\quad + 2(\vec{\nabla}_1 f(r_1)) \cdot (\vec{\nabla}_1 g(r_{12})). \end{aligned} \quad (17)$$

The Laplacian of a scalar function can be computed as follows:

$$\nabla_1^2 f(r_1) = \vec{\nabla} \cdot (\vec{\nabla} f(r_1)) = (\vec{\nabla} f'(r_1)) \cdot \hat{r}_1 + f'(r_1) \vec{\nabla} \cdot \hat{r}_1 = f''(r_1) + \frac{2}{r_1} f'(r_1), \quad (18)$$

where $\hat{r}_1 = \vec{r}_1/r_1$ is a unit vector. Similarly

$$\nabla_1^2 g(r_{12}) = g''(r_{12}) + \frac{2}{r_{12}} g'(r_{12}), \quad (19)$$

$$(\vec{\nabla}_1 f(r_1)) \cdot (\vec{\nabla}_1 g(r_{12})) = \hat{r}_1 \cdot \hat{r}_{12} f'(r_1) g'(r_{12}), \quad (20)$$

and

$$(\vec{\nabla}_2 f(r_2)) \cdot (\vec{\nabla}_2 g(r_{12})) = \hat{r}_2 \cdot \hat{r}_{21} f'(r_2) g'(r_{12}). \quad (21)$$

Note that $r_{12} = r_{21}$, but $\hat{r}_{12} = -\hat{r}_{21}$.

The derivatives of the scalar functions are easily computed:

$$\frac{f'(r)}{f(r)} = -2, \quad \frac{f''(r)}{f(r)} = 4, \quad (22)$$

$$\frac{g'(r)}{g(r)} = \frac{1}{2(1+\alpha r)^2}, \quad \frac{g''(r)}{g(r)} = \frac{1}{4(1+\alpha r)^4} - \frac{\alpha}{(1+\alpha r)^3}. \quad (23)$$

Collecting all terms, the local energy is given by

$$\begin{aligned} E_L &= -\frac{f''(r_1)}{2f(r_1)} - \frac{f''(r_2)}{2f(r_2)} - \frac{g''(r_{12})}{g(r_{12})} - \frac{f'(r_1)}{r_1 f(r_1)} - \frac{f'(r_2)}{r_1 f(r_2)} - \frac{2g'(r_{12})}{g(r_{12})} \\ &\quad - \frac{\hat{r}_1 \cdot \hat{r}_{12} f'(r_1) g'(r_{12})}{f(r_1) g(r_{12})} - \frac{\hat{r}_2 \cdot \hat{r}_{21} f'(r_2) g'(r_{12})}{f(r_2) g(r_{12})} - \frac{2}{r_1} - \frac{2}{r_2} + \frac{1}{r_{12}} \end{aligned} \quad (24)$$

$$\begin{aligned} &= -4 - \frac{1}{4(1+\alpha r_{12})^4} + \frac{\alpha}{(1+\alpha r_{12})^3} + \frac{2}{r_1} + \frac{2}{r_2} - \frac{1}{r_{12}(1+\alpha r_{12})^2} \\ &\quad + \frac{(\hat{r}_1 - \hat{r}_2) \cdot \hat{r}_{12}}{(1+\alpha r_{12})^2} - \frac{2}{r_1} - \frac{2}{r_2} + \frac{1}{r_{12}}. \end{aligned} \quad (25)$$

Using

$$\frac{1}{r_{12}} - \frac{1}{r_{12}(1+\alpha r_{12})^2} = \frac{\alpha}{(1+\alpha r_{12})} + \frac{\alpha}{(1+\alpha r_{12})^2} \quad (26)$$

gives the final form for the local energy

$$\begin{aligned} E_L(\mathbf{r}_1, \mathbf{r}_2) &= -4 + \frac{\alpha}{(1+\alpha r_{12})} + \frac{\alpha}{(1+\alpha r_{12})^2} + \frac{\alpha}{(1+\alpha r_{12})^3} \\ &\quad - \frac{1}{4(1+\alpha r_{12})^4} + \frac{\mathbf{r}_{12} \cdot (\hat{\mathbf{r}}_1 - \hat{\mathbf{r}}_2)}{(1+\alpha r_{12})^2}. \end{aligned} \quad (27)$$

VMC Program for the Helium Atom

The following program implements this trial function choice.

Program 2: <http://www.physics.buffalo.edu/phy410-505/topic5/vmc-he.cpp>

```
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <vector>
using namespace std;

#include "linalg.hpp"
#include "random.hpp"
using namespace cpl;

Random rg;           // random number generator object

int dimensions = 3;  // dimensionality of space
int electrons = 2;  // number of electrons
int N;              // number of walkers
vector<
  vector<
    vector<double> > > r; // vector of coordinates

double alpha;       // Pade-Jastrow variational parameter
double delta;       // trial step size

void initialize() { // function to initialize the calculation

  // initialize random number generator
  rg.set_seed_time();
  cout << " using " << rg.get_algorithm()
       << " and seed " << rg.get_seed() << endl;

  // initialize the positions of the walkers
  for (int n = 0; n < N; n++) {
    vector< vector<double> > exyz;
    for (int e = 0; e < electrons; e++) {
      vector<double> xyz;
      for (int d = 0; d < dimensions; d++)
        xyz.push_back(rg.rand() - 0.5);
      exyz.push_back(xyz);
    }
    r.push_back(exyz);
  }

  // set initial step size
  delta = 1;
}

double e_sum;       // accumulator for energy average
```

```

double e_sqd_sum;          // accumulator for variance in energy

void zero_accumulators() { // set accumulators to zero
    e_sum = e_sqd_sum = 0;
}

double Psi(                // compute trial wave function
    vector<double>& r_electron_1,
    vector<double>& r_electron_2
) {

    // value of trial wave function for walker n
    double r1 = 0, r2 = 0, r12 = 0;
    for (int d = 0; d < dimensions; d++) {
        r1 += r_electron_1[d] * r_electron_1[d];
        r2 += r_electron_2[d] * r_electron_2[d];
        r12 += (r_electron_1[d] - r_electron_2[d])
            * (r_electron_1[d] - r_electron_2[d]);
    }
    r1 = sqrt(r1);
    r2 = sqrt(r2);
    r12 = sqrt(r12);
    double Psi = - 2*r1 - 2*r2 + r12 / (2 * (1 + alpha*r12));

    return exp(Psi);
}

double e_local(            // compute contribution to local energy
    vector<double>& r_electron_1,
    vector<double>& r_electron_2
) {

    // value of trial wave function for walker n
    double r1 = 0, r2 = 0, r12 = 0;
    for (int d = 0; d < dimensions; d++) {
        r1 += r_electron_1[d] * r_electron_1[d];
        r2 += r_electron_2[d] * r_electron_2[d];
        r12 += (r_electron_1[d] - r_electron_2[d]) *
            (r_electron_1[d] - r_electron_2[d]);
    }
    r1 = sqrt(r1);
    r2 = sqrt(r2);
    r12 = sqrt(r12);
    double dot_prod = 0;
    for (int d = 0; d < dimensions; d++) {
        dot_prod += (r_electron_1[d] - r_electron_2[d]) / r12 *
            (r_electron_1[d] / r1 - r_electron_2[d] / r2);
    }
    double denom = 1 / (1 + alpha * r12);
    double denom_2 = denom * denom;
    double denom_3 = denom_2 * denom;
}

```

```

double denom_4 = denom_2 * denom_2;
double e = - 4 + alpha * (denom + denom_2 + denom_3)
           - denom_4 / 4 + dot_prod * denom_2;

return e;
}

int n_accept;           // keep track of number of accepted steps

void Metropolis_step(   // one Metropolis step
    int walker         // for this walker
) {

    // make a trial move of each electron
    vector<double> r_electron_1(dimensions), r_electron_2(dimensions),
        r_trial_1(dimensions), r_trial_2(dimensions);
    for (int d = 0; d < dimensions; d++) {
        r_electron_1[d] = r[walker][0][d];
        r_trial_1[d] = r_electron_1[d] + delta * (2 * rg.rand() - 1);
        r_electron_2[d] = r[walker][1][d];
        r_trial_2[d] = r_electron_2[d] + delta * (2 * rg.rand() - 1);
    }

    // Metropolis test
    double w = Psi(r_trial_1, r_trial_2) / Psi(r_electron_1, r_electron_2);
    if (rg.rand() < w * w) {
        for (int d = 0; d < dimensions; d++) {
            r[walker][0][d] = r_electron_1[d] = r_trial_1[d];
            r[walker][1][d] = r_electron_2[d] = r_trial_2[d];
        }
        ++n_accept;
    }

    // accumulate local energy
    double e = e_local(r_electron_1, r_electron_2);
    e_sum += e;
    e_sqd_sum += e * e;
}

void one_Monte_Carlo_step() {

    // do Metropolis step for each walker
    for (int n = 0; n < N; n++)
        Metropolis_step(n);
}

int main() {

    cout << " Variational Monte Carlo for Helium Atom\n"
         << " -----\n";
    cout << " Enter number of walkers: ";
}

```

```

cin >> N;
cout << " Enter Pade-Jastrow parameter alpha: ";
cin >> alpha;
cout << " Enter number of Monte Carlo steps: ";
int MC_steps;
cin >> MC_steps;

initialize();

// perform 20% of MC_steps as thermalization steps
// and adjust step size so acceptance ratio ~50%
int therm_steps = int(0.2 * MC_steps);
int adjust_interval = int(0.1 * therm_steps) + 1;
n_accept = 0;
cout << " Performing " << therm_steps << " thermalization steps ..."
    << flush;
for (int i = 0; i < therm_steps; i++) {
    one_Monte_Carlo_step();
    if ((i+1) % adjust_interval == 0) {
        delta *= n_accept / (0.5 * N * adjust_interval);
        n_accept = 0;
    }
}
cout << "\n Adjusted step size delta = " << delta << endl;

// production steps
zero_accumulators();
n_accept = 0;
cout << " Performing " << MC_steps << " production steps ..." << flush;
for (int i = 0; i < MC_steps; i++)
    one_Monte_Carlo_step();

// compute and print energy
double e_ave = e_sum / double(N) / MC_steps;
double e_var = e_sqd_sum / double(N) / MC_steps - e_ave * e_ave;
double error = sqrt(e_var) / sqrt(double(N) * MC_steps);
cout << "\n <Energy> = " << e_ave << " +/- " << error
    << "\n Variance = " << e_var << endl;
}

```

Results from running the program

The results can be compared with some famous variational calculations[2, 3], see Table. 2.

Method	Energy (Hartrees)
Wavefunction $e^{-2r_1-2r_2}$	-2.75
$e^{-\alpha r_1-\alpha r_2}$ with $\alpha = 27/16$	-2.84765
Hartree-Fock	-2.8617
Padé-Jastrow <code>vmc-he.cpp</code>	-2.878
Hylleraas (1929) 10 variational parameters	-2.90363
Pekeris (1959) 1,078 variational parameters	-2.90372
Experiment	-2.90372

Table 2: Comparison with other theoretical predictions and experiment.

Homework Problem

Simplify the `vmc-he.cpp` program to find the ground state energy and probability density of the hydrogen atom. Compare your results with analytic expressions from your quantum mechanics textbook. (If you prefer you can start with the harmonic oscillator program `vmc.cpp` and generalize it to study the hydrogen atom.)

References

- [1] G. Tanner, K. Richter and J.-M. Rost, “The theory of two-electron atoms: between ground state and complete fragmentation”, *Rev. Mod. Phys.* **72**, 497 (2000), <http://link.aps.org/doi/10.1103/RevModPhys.72.497>.
- [2] E.A. Hylleraas, *Z. Phys.* **54**, 347 (1929).
- [3] C.L. Pekeris, *Phys. Rev.* **126**, 1470 (1962), http://prola.aps.org/abstract/PR/v126/i4/p1470_1.