

Harmonic Oscillator: Variational Monte Carlo

Eigenstates of the Quantum Harmonic Oscillator

A basic problem in quantum mechanics is to find the energy eigenstates of quantum mechanical systems. Perhaps the simplest such system is the quantum harmonic oscillator in one dimension.

To find the energy eigenstates, we solve the time-independent Schrödinger equation

$$H\psi(x) = \left[-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} + \frac{1}{2}m\omega^2 x^2 \right] \psi(x) = E\psi(x), \quad (1)$$

subject to boundary conditions

$$\lim_{x \rightarrow \pm\infty} \psi(x) = 0. \quad (2)$$

The solution of this equation is the *wave function* of the particle. The interpretation of the wave function $\psi(x)$ is that

$$|\psi(x)|^2 dx \quad (3)$$

is the *probability* of finding the particle between position x and position $x + dx$.

Such a simple one-dimensional problem can easily be solved numerically using deterministic algorithms to solve ordinary differential equations. In fact, the harmonic oscillator problem can be solved exactly. Solutions which satisfy the boundary conditions exist only for discrete *eigenvalues* of the energy

$$E_n = \left(n + \frac{1}{2} \right) \hbar\omega \quad n = 0, 1, 2, 3, \dots \quad (4)$$

and the normalized energy *eigenfunctions* are given by

$$\psi_n(x) = \left(\frac{m\omega}{\pi\hbar} \right)^{\frac{1}{4}} \frac{1}{\sqrt{2^n n!}} H_n \left(x \sqrt{\frac{m\omega}{\hbar}} \right) e^{-m\omega x^2 / (2\hbar)}, \quad (5)$$

where H_n are Hermite polynomials

$$H_0(y) = 1, \quad H_1(y) = 2y, \quad H_2(y) = 4y^2 - 2, \quad \text{etc.} \quad (6)$$

Exact solutions have been found only for a very small number of problems which can essentially be reduced to one-dimensional ordinary differential equations. Another example is the hydrogen atom which consists of a proton and an electron interacting through a Coulomb force.

The Variational Theorem

The variational method is a powerful tool to estimate the ground state energy of a quantum system, and some of its low lying excited states, when an exact solution cannot be found. When combined with Monte Carlo methods, it can be used on a wide range of atomic and solid state systems, as discussed in the review article by Foulkes et al.[1].

The eigenfunctions of a quantum mechanical problem are *complete*. This means that any wave function $\Psi(x)$ can be expressed as a linear superposition

$$\Psi(x) = \sum_n c_n \psi_n(x), \quad (7)$$

where c_n are complex numbers. According to the rules of quantum mechanics, the average energy of a particle with this wave function is given by

$$\langle E \rangle = \frac{\int dx \Psi^*(x) H \Psi(x)}{\int dx \Psi^*(x) \Psi(x)}. \quad (8)$$

The *variational theorem* states that $\langle E \rangle \geq E_0$ for *any* Ψ , and $\langle E \rangle = E_0$ if and only if $\Psi(x) = c_0 \psi_0(x)$. It is easy to see this is we use the fact that the eigenfunctions $\psi_n(x)$ can be chosen to be *orthonormal*

$$\int dx \psi_n^*(x) \psi_{n'}(x) = \delta_{nn'}, \quad (9)$$

$$\begin{aligned} \langle E \rangle &= \frac{\sum_{n,n'} c_n^* E_{n'} c_{n'} \int dx \psi_n^*(x) \psi_{n'}(x)}{\sum_{n,n'} c_n^* c_{n'} \int dx \psi_n^*(x) \psi_{n'}(x)} \\ &= \frac{\sum_n |c_n|^2 E_n}{\sum_n |c_n|^2} = E_0 + \frac{\sum_n |c_n|^2 (E_n - E_0)}{\sum_n |c_n|^2}, \end{aligned} \quad (10)$$

we have used the eigenvalue equation $H \psi_{n'} = E_{n'} \psi_{n'}$. Because $E_n - E_0 > 0$, the second term in the last expression is positive and larger than zero *unless* all $c_n = 0$ for all $n \neq 0$.

The *variational method* is based on this important theorem: to estimate the ground state energy and wave function, choose a *trial wave function* $\Psi_{T,\alpha}(x)$ which depends on a parameter α .

The expectation value $\langle E \rangle$ will depend on the parameter α , which can be *varied to minimize* $\langle E \rangle$.

This energy and the corresponding $\Psi_{T,\alpha}(x)$ then provide the best estimates for the ground state energy and wave function.

Variational Monte Carlo

In the Variational Monte Carlo method, a trial wave function $\Psi_{T,\alpha}$, which depends on a set of variational parameters $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_S)$, is carefully chosen.

An efficient way must be found to evaluate the expected value of the energy

$$\langle E \rangle = \frac{\int dR \Psi_{T,\alpha}^* H \Psi_{T,\alpha}}{\int dR |\Psi_{T,\alpha}|^2}, \quad (11)$$

where $R = (\mathbf{r}_1, \dots, \mathbf{r}_N)$ are the positions of the particles in the system.

The problem is that this multi-dimensional integral must be evaluated many many times as the program searches the α parameter space for the minimum $\langle E \rangle$.

Monte Carlo methods can be used to evaluate multi-dimensional integrals much more efficiently than deterministic methods. The key to using a Monte Carlo method is to define a *positive definite weight function* which is used to sample the important regions of the multi-dimensional space.

In the VMC method, the weight function is taken to be

$$\rho(R) = \frac{|\Psi_{T,\alpha}(R)|^2}{\int dR |\Psi_{T,\alpha}|^2}. \quad (12)$$

The expected value of the energy can then be written

$$\langle E \rangle = \frac{\int dR |\Psi_{T,\alpha}|^2 \frac{H \Psi_{T,\alpha}}{\Psi_{T,\alpha}}}{\int dR |\Psi_{T,\alpha}|^2} = \int dR \rho(R) E_L(R), \quad (13)$$

where the *local energy* $E_L(R)$ is defined by

$$E_L(R) = \frac{H\Psi_{T,\alpha}(R)}{\Psi_{T,\alpha}(R)}. \quad (14)$$

The variational wave function $\Psi_{T,\alpha}(R)$ is usually chosen to be real and non-zero (almost) everywhere in the region of integration. In evaluating the ground state of the system, it can generally be chosen to be real and positive definite.

The VMC strategy is to generate a random set of points $\{R_i\}$, $i = 1, \dots, M$ in configuration space that are distributed according to $\rho(R)$. Then

$$\langle E \rangle = \frac{1}{M} \sum_{i=1}^M E_L(R_i). \quad (15)$$

VMC Program for the Harmonic Oscillator

A simple variational trial wave function for the Harmonic Oscillator is:

$$\Psi_{T,\alpha}(x) = e^{-\alpha x^2}. \quad (16)$$

Let's choose units so that $m = 1$, $\hbar = 1$, and $\omega = 1$. The Hamiltonian operator in these units is

$$H = -\frac{1}{2} \frac{d^2}{dx^2} + \frac{1}{2} x^2, \quad (17)$$

from which the local energy can be derived:

$$E_L(x) = \alpha + x^2 \left(\frac{1}{2} - 2\alpha^2 \right). \quad (18)$$

Note that when $\alpha = 1/2$ we obtain the exact ground state energy and eigen function.

The following program implements the VMC method outlined above.

```
_____ Program 1: http://www.physics.buffalo.edu/phy410-505/topic5/vmc.cpp _____  
  
#include <cmath>  
#include <cstdlib>  
#include <iostream>  
#include <fstream>  
using namespace std;  
  
#include "linalg.hpp"  
#include "random.hpp"  
using namespace cpl;  
  
Random rg; // random number generator object
```

The program uses N Metropolis random walkers

The weight function

$$\rho(x) \sim e^{-2\alpha x^2}, \quad (19)$$

can easily be generated using a single Metropolis random walker. However, in more complex problems, it is conventional to use a large number of independent random walkers that are started at random points in the configuration space.

This is because the weight function can be very complicated in a multi-dimensional space: a single walker might have trouble locating all of the peaks in the distribution; using a large number of randomly located walkers improves the probability that the distribution will be correctly generated.

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

```
int N; // number of walkers
vector<double> x; // walker positions
double delta; // step size
```

Variables to measure observables

Variables are introduced to accumulate E_L values and compute the Monte Carlo average and error estimate. The probability distribution is accumulated in a histogram with bins of size dx in the range $-10 \leq x \leq 10$.

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

```
double e_sum; // accumulator to find energy
double e_sqd_sum; // accumulator to find fluctuations in E
double x_min = -10; // minimum x for histogramming psi^2(x)
double x_max = +10; // maximum x for histogramming psi^2(x)
double dx = 0.1; // psi^2(x) histogram step size
vector<double> psi_sqd; // psi^2(x) histogram

void zero_accumulators() {
    e_sum = e_sqd_sum = 0;
    for (int i = 0; i < psi_sqd.size(); i++)
        psi_sqd[i] = 0;
}
```

Initialization

The following function allocates memory to hold the positions of the walkers and distributes them uniformly at random in the range $-0.5 \leq x \leq 0.5$. The step size δ for the Metropolis walk is set to 1.

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

```

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

    x = vector<double>(N);
    for (int i = 0; i < N; i++)
        x[i] = rg.rand() - 0.5;
    delta = 1;

    psi_sqd = vector<double>(int((x_max - x_min) / dx));

    zero_accumulators();
}

```

Probability function and local energy

The following function evaluates the ratio

$$w = \frac{\rho(x_{\text{trial}})}{\rho(x)}, \quad (20)$$

which is used in the Metropolis algorithm: if $w \geq 1$ the step is accepted unconditionally; and if $w < 1$ the step is accepted only if w is larger than a uniform random deviate between 0 and 1.

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

```

double alpha;                // trial function is exp(-alpha*x^2)

double p(double x_trial, double x) {

    // compute the ratio of rho(x_trial) / rho(x)
    return exp(- 2 * alpha * (x_trial*x_trial - x*x));
}

double e_local(double x) {

    // compute the local energy
    return alpha + x * x * (0.5 - 2 * alpha * alpha);
}

```

One Metropolis step

One Metropolis step is implemented as follows:

- Choose one of the N walkers at random
- The walker takes a trial step to a new position that is Gaussian distributed with width δ around the old position. The function `gasdev` defined in `random.hpp` returns a Gaussian deviate with unit width: multiplying this by a step size δ yields a Gaussian deviate with $\sigma = \delta$.

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

```
int n_accept;                // accumulator for number of accepted steps

void Metropolis_step() {

    // chose a walker at random
    int n = int(rg.rand() * N);

    // make a trial move
    double x_trial = x[n] + delta * rg.gasdev();

    // Metropolis test
    if (p(x_trial, x[n]) > rg.rand()) {
        x[n] = x_trial;
        ++n_accept;
    }

    // accumulate energy and wave function
    double e = e_local(x[n]);
    e_sum += e;
    e_sqd_sum += e * e;
    int i = int((x[n] - x_min) / dx);
    if (i >= 0 && i < psi_sqd.size())
        psi_sqd[i] += 1;
}
```

As usual, when we have multiple walkers, one Monte Carlo Step is conventionally defined as N Metropolis steps:

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

```
void one_Monte_Carlo_step() {

    // perform N Metropolis steps
    for (int i = 0; i < N; i++) {
        Metropolis_step();
    }
}
```

Steering the computation with the main function

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

```
int main() {

    cout << " Variational Monte Carlo for Harmonic Oscillator\n"
         << " -----\n";
    cout << " Enter number of walkers: ";
    cin >> N;
    cout << " Enter parameter alpha: ";
    cin >> alpha;
    cout << " Enter number of Monte Carlo steps: ";
    int MC_steps;
    cin >> MC_steps;

    initialize();
```

As in all Monte Carlo calculations, some number of steps are taken and discarded to allow the walkers to come to “equilibrium.” The thermalization phase is also used to adjust the step size so that the acceptance ratio is approximately 50%.

- If δ is too small, then too many steps will be accepted; and conversely, if δ is too large, then too many steps will be rejected.
- Multiplying δ by one half of the acceptance ratio will increase δ if the ratio is larger than 0.5, and decrease δ if the ratio is smaller than 0.5.

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

```
// 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 Gaussian step size = " << delta << endl;
```

Once the system has thermalized, the accumulators for observables are initialized and the production steps are taken.

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

```
// 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();
```

Finally the average value of the energy and the Monte Carlo error estimate are printed, and the probability distribution $\sim \psi_0^2(x)$ is written in the form of a histogram to a file.

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

```
// 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;

// write wave function squared in file
ofstream file("psi_sqd.data");
double psi_norm = 0;
for (int i = 0; i < psi_sqd.size(); i++)
    psi_norm += psi_sqd[i] * dx;
for (int i = 0; i < psi_sqd.size(); i++) {
    double x = x_min + i * dx;
    file << x << '\t' << psi_sqd[i] / psi_norm << '\n';
}
file.close();
cout << " Probability density written in file psi_sqd.data" << endl;
}
```

The following plots show results for the average energy $\langle E \rangle$ and its variance $\langle E^2 \rangle - \langle E \rangle^2$ as functions of the variational parameter α .

Runs were performed with $N = 300$ walkers and $MC_steps = 10,000$.

As might be expected, the average energy is minimum $\langle E \rangle = 1/2$, and the variance is zero, at $\alpha = 1/2$ which corresponds to the exact solution for the harmonic oscillator ground state.

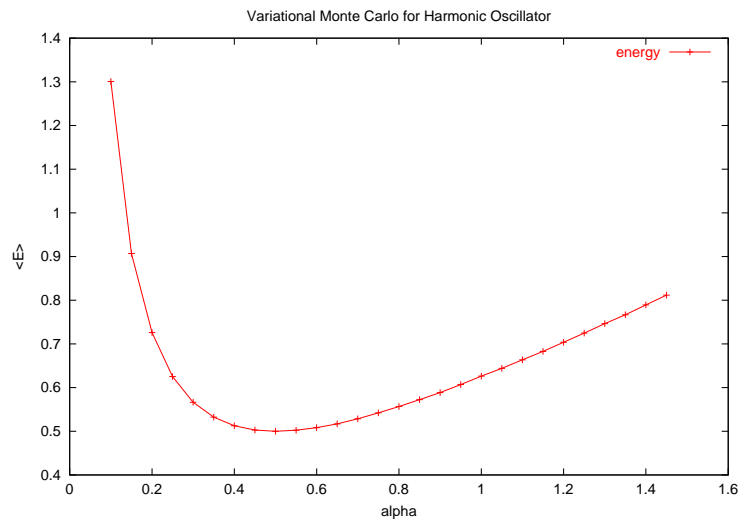


Figure 1: Energy $\langle E \rangle$ as a function of the variational parameter α .

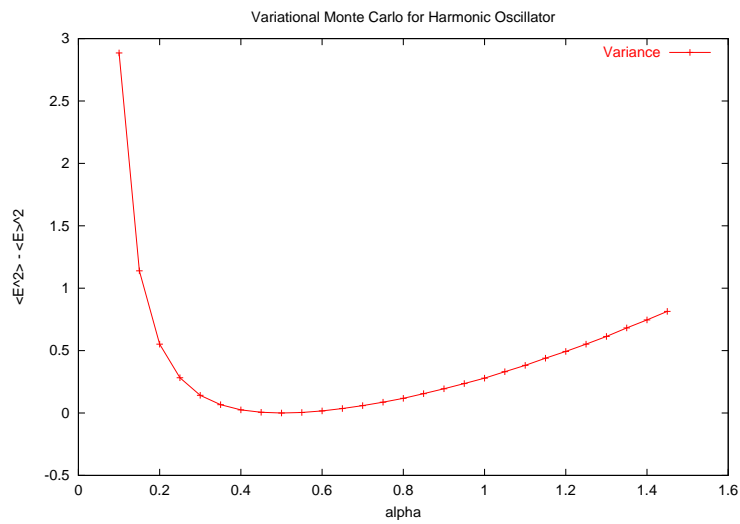


Figure 2: Variance $\langle E^2 \rangle - \langle E \rangle^2$ of the energy as a function of the variational parameter α .

Homework Problem

Add a quartic x^4 perturbation to the harmonic oscillator Hamiltonian. Estimate the ground state energy using `vmc.cpp` and compare with one other method, for example, first order perturbation theory, or the `schroedinger.cpp` program. Plot and compare the normalized probability densities of the unperturbed and perturbed oscillator.

References

- [1] W.M.C. Foulkes, “Quantum Monte Carlo simulations of solids”, *Rev. Mod. Phys.* **73**, 33 (2001), http://prola.aps.org/abstract/RMP/v73/i1/p33_1.