The Mathematica® Journal

Monte Carlo Simulation of Simple Molecules

Jan Vrbik

We show how a Monte Carlo procedure (based on random numbers) can generate a large sample of electron locations in any simple molecule. Based on this sampling, we can accurately estimate the molecule's ground-state energy and other properties of interest. We demonstrate this using the LiH molecule.

Trial Solution to Schrödinger Equation

A mathematical description of a molecule (say LiH) is provided by a solution to the Schrödinger equation [1] (a differential eigenvalue problem with smallest eigenvalue E_0):

$$-\frac{1}{2}\sum_i \nabla_i^2 \Phi + V\,\Phi = E_0\,\Phi. \tag{1}$$

Here ∇^2 is the Laplacian operator $\nabla^2 f \equiv \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$, the i summation is over all electrons, Φ is a function of their positions, and E_0 is the molecule's ground-state energy. The electrostatic potential V in the molecule is given by

$$V = \sum_{i,\alpha} \frac{-Z_\alpha}{|\mathbf{r}_i - \mathbf{R}_\alpha|} + \sum_{i<j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{\alpha<\beta} \frac{Z_\alpha\, Z_\beta}{|\mathbf{R}_\alpha - \mathbf{R}_\beta|}. \tag{2}$$

Here, the first summation is over all electrons and nuclei, where Z_α and $\mathbf{R}_\alpha$ are the nuclear charges and locations, respectively. The second summation is over all pairs of electrons, and the last one, analogously, is over all pairs of nuclei. The nuclear locations are kept fixed in accordance with the Born–Oppenheimer approximation [2]. We use atomic units, in which the electron's charge, mass, and Planck's constant are set equal to 1.

Thus, V for the LiH molecule is computed by the following functions, assuming that the Li nucleus is located at the origin.

```
m[q_] := √(q.q)
R_H = {0, 0, 3};
V[Q_] := Sum[-3/m[Q[[i]]] - 1/m[Q[[i]] - R_H], {i, 4}] +
   Sum[1/m[Q[[i]] - Q[[j]]], {i, 3}, {j, i + 1, 4}] + 3/m[R_H]
```

The argument `Q` consists of the four electron positions, each a list of three Cartesian coordinates.

The solution Φ to (1) is subject to two boundary conditions: that $\Phi = 0$ whenever one or more electrons approach infinity, and that Φ must change sign whenever two electrons of the same spin are interchanged, in accord with the Pauli exclusion principle [3].

There are several techniques for finding an approximate solution to (1), which we denote by Ψ, to distinguish it from the exact solution Φ. Monte Carlo is a method that "borrows" one of these solutions, called, in this context, a trial function [1], and seeks to improve its accuracy. This is done by generating, with the help of Ψ, a random statistical sample, called an ensemble in this context, representing the exact solution. Based on this ensemble of electron locations, or configurations, one can then easily find, within the "standard" statistical error, the value of the molecule's ground-state energy, and related properties such as dipole moment and polarizability, etc.

To build a trial solution for LiH, we start with two molecular orbitals, linear combinations of four simple atomic orbitals [2].

```
MO[q_] := {{1, 0, 0.05, 0}, {0, 1, 0.38, -0.22}}.
   {Exp[-2.89 m[q]], Exp[-0.87 m[q - R_H]],
 q[[3]] Exp[-2.85 m[q]], (q[[3]] - R_H[[3]]) Exp[-0.95 m[q - R_H]]}
```

The argument `q` is a list of three coordinates that describe a location of a single electron. The parameters of the `MO` functions and those of `J` in the next expression have been obtained by minimizing the variational energy (7).

Secondly, we define the so-called *Jastrow function* [2], its arguments being locations of each pair of electrons.

```
J[q1_, q2_] := Exp[0.5 m[q1 - q2]/(1 + 0.6 m[q1 - q2])]
```

The resulting trial function has the following form.

```
ψ[q1_, q2_, q3_, q4_] :=
 Det[{MO[q1], MO[q2]}] Det[{MO[q3], MO[q4]}] J[q1, q3]
  J[q1, q4] J[q2, q3] J[q2, q4]
```

The two factors are determinants of the two molecular orbitals, each evaluated at the location of the four electrons. Electrons 1 and 2 have spin "up," while 3 and 4 have spin "down." The remaining factors are Jastrow functions for each pair of opposite-spin electrons.

Next we estimate the ground-state energy of LiH based on this trial function.

Variational Estimate of the Smallest Eigenvalue

Let us rewrite (1) more compactly as

$$\mathbb{H}\,\Phi = E_0\,\Phi, \tag{3}$$

where $\mathbb{H}$ represents the sum of both operators on the left-hand side of (1). The variational principle tells us [2] that

$$\frac{\int \Psi\,\mathbb{H}\,\Psi\,d\mathbf{R}}{\int \Psi^2\,d\mathbf{R}} \geq E_0. \tag{4}$$

The integration is over the three coordinates of each of the four electrons, altogether a 12-dimensional problem—no mean task—and Ψ is any trial solution to (1). The limit of equality holds only for the exact solution Φ, but for approximate solutions, called variational estimates of the ground-state energy, the left-hand side of (4) is usually quite close to E_0. The main problem is how to evaluate the two 12-dimensional integrals; this is impossible to do analytically and not feasible even numerically. Well, Monte Carlo has the answer.

Let us first define the so-called *drift function* by

$$\mathbf{F} = \left\{\frac{\nabla_1\Psi}{\Psi}, \frac{\nabla_2\Psi}{\Psi}, \frac{\nabla_3\Psi}{\Psi}, \frac{\nabla_4\Psi}{\Psi}\right\} \tag{5}$$

and *local energy* by

$$E_L = \frac{\mathbb{H}\,\Psi}{\Psi} = -\frac{1}{2}\,\frac{\sum_i \nabla_i^2\Psi}{\Psi} + V. \tag{6}$$

Here are the corresponding commands.

```
Ψ = Apply[ψ, Partition[Table[xi, {i, 12}], 3]];
F = Table[D[Ψ, xi], {i, 12}] / Ψ;
EL = -1/2 Sum[D[Ψ, {xi, 2}], {i, 12}] / Ψ +
    V[Partition[Table[xi, {i, 12}], 3]];
```

The coordinates of the four electrons are now called $x_1, x_2, \ldots, x_{12}$, understanding that these are the x, y, z coordinates of the first electron, followed by the x, y, z coordinates of the second electron, etc. The left-hand side of (4) can now be rewritten as

$$\frac{\int E_L \Psi^2 \, d\mathbf{R}}{\int \Psi^2 \, d\mathbf{R}}. \tag{7}$$

Next, we randomly generate a large sample of 1000 configurations of $x_1, x_2, \ldots, x_{12}$ values denoted collectively as Ro and compute the corresponding Ψ, F, and E_L.

```
n = 1000;
Ro = Partition[RandomReal[{-3, 3}, 12 n], n];
Po = Ψ /. x_i_ :> Ro[[i]];
Fo = F /. x_i_ :> Ro[[i]];
Eo = EL /. x_i_ :> Ro[[i]];
```

By averaging the 1000 values of E_L, we get an estimate of E_0. Unfortunately, this estimate will be very inaccurate since our random sample of configurations bears, at this point, no relationship to Ψ^2 of (7).

To fix this, we move each configuration to a new location, specified by

$$\mathbf{R}_n = \mathbf{R}_o + \tau \mathbf{F}_o + \sqrt{\tau}\, \mathbf{N}, \tag{8}$$

where $\mathbf{F}_o$ is the drift function evaluated at the old location $\mathbf{R}_o$, $\mathbf{N}$ is a random vector of 12 independent components from the normal distribution (with mean 0 and standard deviation 1), and τ is an extra parameter called the step size, which controls the speed of this motion. This will bring us a step closer to the desired distribution of configurations whose probability density function is proportional to Ψ^2, but it will take dozens of such moves to reach it. Monitoring the consecutive sample averages of E_L, we find no systematic change but only random fluctuations after reaching a so-called equilibration. Once in equilibrium, we continue advancing our configuration for as many steps (called iterations) as feasible, to reduce the statistical error of the final estimate. This is computed by combining all the individual sample averages into one: the so-called grand mean).

There is only one little snag: the result will still have an error proportional to the step size τ. To correct for this, we would have to make τ impractically small and equilibration would take forever. Fortunately, there is another way, called Metropolis sampling [3]: for each proposed move (8) we compute a scalar quantity

$$T = \frac{{\Psi_n}^2}{{\Psi_o}^2} \exp\left((\mathbf{F}_o + \mathbf{F}_n) \cdot \left[\mathbf{R}_o - \mathbf{R}_n + \frac{\tau}{2} (\mathbf{F}_o - \mathbf{F}_n) \right] \right), \tag{9}$$

where the subscripts n and o mean that Ψ and $\mathbf{F}$ have been evaluated at the new or old location, respectively. The move is then accepted with a probability equal to T. When $T > 1$, the move is accepted automatically. When a move is rejected, the configuration simply remains at its old location $\mathbf{R}_o$. The step size τ should be adjusted to yield a reasonable proportion of rejections, say between 10% and 30%.

Rejecting configurations in this manner creates the last small problem: in our original random sample there is usually a handful of configurations which, because they have

landed at "wrong" locations, just would not move. To fix this, we have to monitor, for each configuration, the number of consecutive times a move has been rejected, and let it move, regardless of T, when this number exceeds a certain value, such as 10. After this is done and the sample equilibrates, the problem automatically disappears, and no configuration is ever refused its move more than six consecutive times (confirming that 10 consecutive rejections was a good indication of a "stuck" configuration).

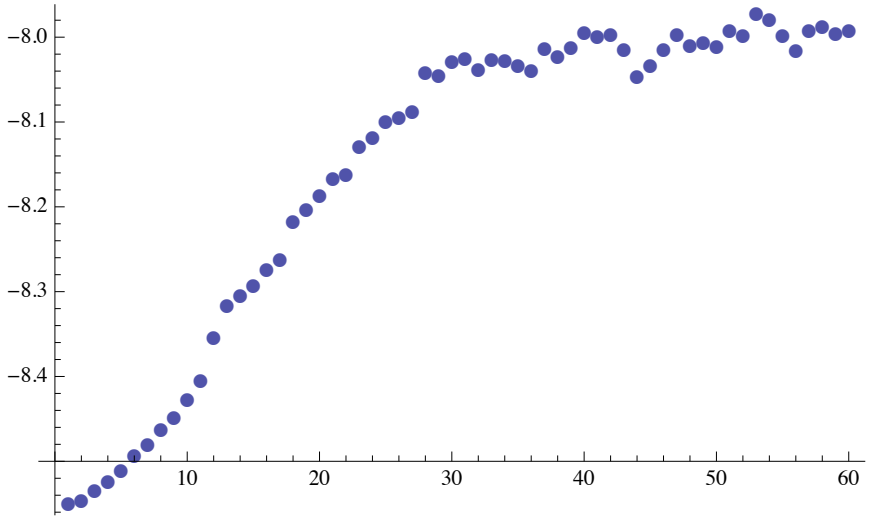
The following program carries this out (its execution will take a minute or two).

```
τ = 0.05; moni = Table[0, {n}]; res = {};
Do[
 Rn =
  Ro + τ Fo +
   Partition[RandomReal[NormalDistribution[0, Sqrt[τ]], 12 n],
    n];
Pn = Ψ /. x_i_ :> Rn[[i]]; Fn = F /. x_i_ :> Rn[[i]];
 En = EL /. x_i_ :> Rn[[i]];
T = Thread[Thread[10 < moni] || Thread[ RandomReal[{0, 1}, n] <
Pn^2 / Po^2 Exp[Sum[ (Fo + Fn)[[i]] (Ro - Rn + τ/2 (Fo - Fn))[[i]],
          {i, 12}]]]];
Ro = MapThread[If, {Table[T, {12}], Rn, Ro}, 2];
Fo = MapThread[If, {Table[T, {12}], Fn, Fo}, 2];
Eo = MapThread[If, {T, En, Eo}];
 Po = MapThread[If, {T, Pn, Po}];
moni = MapThread[If, {T, Table[0, {n}], moni + 1}];
res = Append[res, {Total[Eo] / n, Count[T, True] / n // N}],
 {60}]
```

Note that `moni` keeps track, for each configuration, of how many of its last consecutive proposed moves have been rejected, its value being reset to 0 as soon as a move is accepted. The program returns (in `res`) the value of all sample averages of the local energy E_L, together with the average acceptance rate.

This displays the former.

```
ListPlot[Transpose[res][[1]], PlotMarkers → Automatic]
```



We see that about 50 iterations are necessary to reach equilibrium.

To get an accurate estimate of E_0, we repeat the simulation with substantially more iterations, changing the `Do` loop's "count" from 60 to 1000, and then computing the grand mean of the E_L values by `Map[Total, Transpose[res]]`. In our case, this yields -8.0261 atomic units, with an average acceptance rate of about 85%.

The easiest way to find the corresponding statistical error is to execute the same program, independently, 5 to 10 times, and then to combine the individual results.

```
Mean[est = {-8.0261, -8.0320, -8.0296, -8.0272, -8.0314}]
StandardDeviation[est] / √(5 - 1)
```

```
-8.02926
```

```
0.0012853
```

This improves the estimate to -8.0293 atomic units, with the standard error of ± 0.0013. The "exact" ground-state energy of LiH is -8.0700 atomic units. The obvious discrepancy, well beyond the statistical error, between our estimate and this value is due to our use of a rather primitive trial function. In accordance with the variational principle, our estimate remains higher than the exact value.

Monte Carlo Estimate of the Smallest Eigenvalue

When, in (7), we replace Ψ^2 by $\Psi\Phi$, the expression then yields "nearly" the exact value of E_0, subject only to a small nodal error [1]. So, all we need to do is to modify our simula-

tion program accordingly, to get a sample from a distribution whose probability density function is proportional to $\Psi\,\Phi$ instead of Ψ^2. This can be achieved by assuming that each configuration carries a different weight, computed from

$$W = 1 - \tau \sum_j \exp\left(-\,2\,\tau^{3/2}\,j\right) \times [E_L\,(j) - E_c], \tag{10}$$

where $E_L\,(j)$ is the local energy of the configuration as computed j iterations ago, the summation is over all past iterations, and E_c is a rough estimate of E_0 (the variational result will do). The sum in (10) "depreciates" the past $E_L - E_c$ values at a rate that should resemble the decrease in serial correlation of the E_L sequence, which can be easily monitored during the variational simulation.

The new estimates of E_0 are then the correspondingly weighted averages, computed at each step and then combined in the usual grand-mean fashion. There are two slight problems with this algorithm, but both can be easily alleviated.

1. Occasionally (e.g., when an electron moves too close to a nucleus), $E_L\,(j) - E_c$ may acquire an unusually low value, making the corresponding W rather large, sometimes larger than all the remaining weights combined. We must eliminate "outliers" outside the $\pm\left(0.5 + \frac{0.03}{\tau}\right)$ range. It is better to do this in a symmetrical way by truncating the value to the nearest boundary of the interval.

2. The final (grand-mean) estimate may have a small, τ-proportional bias. This can be removed only by repeating the simulation, preferably more than once, at several (say 3 to 5) distinct values of τ, and getting an unbiased estimate of E_0 by performing a simple polynomial regression. It is the intercept of the resulting regression line (corresponding to $\tau = 0$) that yields the final answer.

This can all be achieved by the following simple modifications of the program from the previous section. Monte Carlo techniques in general require a long time to execute (this one may take several hours).

For this reason, we have made it (and the subsequent command, which processes its output) non-evaluatable.

```
τ = 0.025; S = moni = Table[0, {n}]; res = {}; L = Exp[-2 τ^(3/2)];
Do[
 Rn =
  Ro + τ Fo +
   Partition[RandomReal[NormalDistribution[0, √τ], 12 n],
    n];
Pn = Ψ /. x_i_ :> Rn[[i]]; Fn = F /. x_i_ :> Rn[[i]];
 En = EL /. x_i_ :> Rn[[i]];
T = Thread[Thread[10 < moni] || Thread[ RandomReal[{0, 1}, n] <
Pn^2 / Po^2 Exp[Sum[ (Fo + Fn)[[i]] (Ro - Rn + τ/2 (Fo - Fn))[[i]],
          {i, 12}]]]];
Ro = MapThread[If, {Table[T, {12}], Rn, Ro}, 2];
Fo = MapThread[If, {Table[T, {12}], Fn, Fo}, 2];
Eo = MapThread[If, {T, En, Eo}];
 Po = MapThread[If, {T, Pn, Po}];
moni = MapThread[If, {T, Table[0, {n}], moni + 1}];
Eh = Table[Min[Max[-0.5 - 0.03/τ, Eo[[i]] + 8.03], 0.5 + 0.03/τ],
   {i, n}];
S = S L + Eh; W = 1 - τ S; W = W / Total[W];
res = Append[res, {Total[W Eo], Count[T, True] / n // N}],
 {7000}]
```

A `ListPlot` of the iteration averages of E_L will show that equilibration now takes many more steps (about 500, when $\tau = 0.025$) than in the case of variational simulation. We have thus decided to discard the first 1000 results and partition the remaining 6000 into six blocks of 1000.

```
r025 =
 Drop[Map[Total, Partition[Transpose[res][[1]], 1000]] / 1000,
  1]
```

Similarly, we can produce six such values with $\tau = 0.050$ and $\tau = 0.075$, calling them `r050` and `r075`, respectively.

```
r025 = {-8.0675, -8.0666, -8.0676, -8.0662, -8.0668,
    -8.0669};
r050 = {-8.0722, -8.0728, -8.0727, -8.0736, -8.0741,
    -8.0723};
r075 = {-8.0815, -8.0799, -8.0818, -8.0778, -8.0795,
    -8.0813};
```

It is now easy to find the resulting intercept.

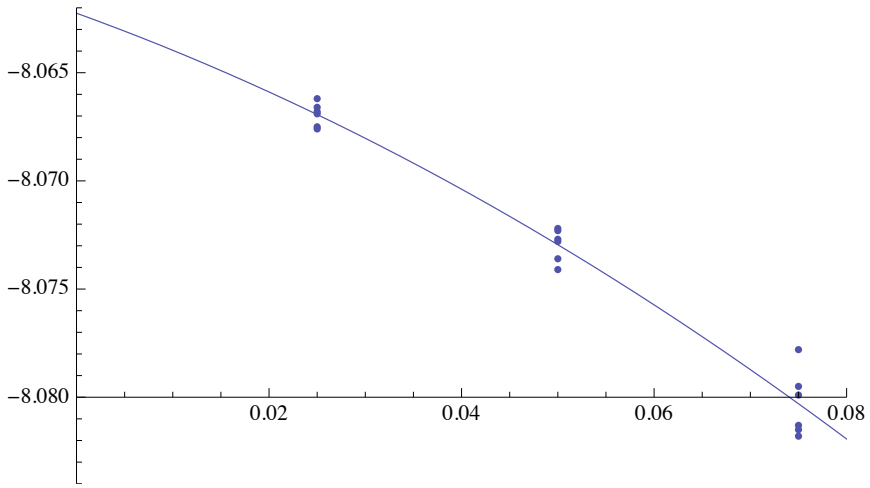
```
reg = Flatten[{Table[{.025, r025[[i]]}, {i, 6}],
     Table[{.05, r050[[i]]}, {i, 6}],
Table[{.075, r075[[i]]}, {i, 6}]}, 1];
LinearModelFit[reg, {x, x^2}, x]["ParameterTable"]
```

	Estimate	Standard Error	t Statistic	P-Value
1	−8.06225	0.0018372	−4388.33	3.11069×10^{-47}
x	−0.160667	0.0834494	−1.92532	0.0733643
x^2	−1.06667	0.825935	−1.29147	0.216089

This yields the value of -8.0622 ± 0.0018 for the corresponding intercept. This is in reasonable agreement, in view of the nodal error, with the exact value of -8.0700 atomic units.

This visualizes the regression fit.

```
Show[
 ListPlot[reg, PlotRange → {{0, .08}, {-8.084, -8.062}}],
Plot[-8.06225 - 0.160667 q - 1.06667 q^2, {q, 0, .11}]]
```



In a follow-up article, we will show how this procedure can be extended to estimate other significant molecular properties, including geometry and polarizability, etc. and how to optimize parameters of a trial function, to make the Monte Carlo method more "self-sufficient."

References

[1] J. B. Anderson, "Quantum Chemistry by Random Walk: Higher Accuracy," *The Journal of Chemical Physics*, **73**(8), 1980 pp. 3897–3899. doi:10.1063/1.440575.

[2] P. J. Reynolds, D. M. Ceperley, B. J. Alder, and W. A. Lester, Jr., "Fixed-Node Quantum Monte Carlo for Molecules," *The Journal of Chemical Physics*, **77**(12), 1982 pp. 5593–5603. doi:10.1063/1.443766.

[3] D. M. Ceperley and B. J. Alder, "Quantum Monte Carlo," *Science*, **231**(4738), 1986 pp. 555–560. doi:10.1126/science.231.4738.555.

J. Vrbik, "Monte Carlo Simulation of Simple Molecules," *The Mathematica Journal*, 2011. dx.doi.org/doi:10.3888/tmj.13–5.

About the Author

Jan Vrbik
Department of Mathematics, Brock University
500 Glenridge Ave., St. Catharines
Ontario, Canada, L2S 3A1
jvrbik@brocku.ca