Standard deviation in $f(x_1,x_2,\ldots,x_N)$

The difference between any function of N statistically independent points and its true value is due to the errors introduced by the points.

$$f\left(x_1,x_2,...,x_N\right)-f\left(x_1.x_2,...,\infty\right)=\sum_{i=1}^{N}\frac{\partial f}{\partial x_i}\delta x_i$$

(1.1)

The difference squared is

$$\left(f\left(x_1,x_2,...,x_N\right)-f\left(x_1.x_2,...,\infty\right)\right)^2=\sum_{i=1,j=1}^{N}\frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j}\delta x_i\delta x_j$$

(1.2)

In addition, its expectation value is

$$\sigma_f^2=\left\langle\left(f\left(x_1,x_2,...,x_N\right)-f\left(x_1.x_2,...,\infty\right)\right)^2\right\rangle$$

$$=\sum_{i=1,j=1}^{N}\frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j}\left\langle\delta x_i\delta x_j\right\rangle=\sum_{i=1}^{N}\left(\frac{\partial f}{\partial x_i}\right)^2\left\langle\delta x_i^2\right\rangle$$

(1.3)

The uncorrelated values for $i \neq j$ are always assumed to be zero in the statistical ensemble. It is this assumption that enables us to arrive at a reasonable approximation to the ensemble average using a single element of the ensemble.

Using the sum of the function differences

The term in the sum in (1.3) can be found by leaving out the i'th data point. That is

$$\frac{\partial f\left(x_1...x_N\right)}{\partial x_i}\delta x_i=f\left(x_1,...,x_i,...x_N\right)-f\left(x_1,...,nox_i,...x_N\right)$$

(2.1)

This yields

$$\sigma_f^2=\sum_{i=1}^{N}\left(f\left(x_1,...,x_i,...x_N\right)-f\left(x_1,...,nox_i,...x_N\right)\right)^2$$

(2.2)

Equation (2.2) is a sum over the result of omitting each term in turn. If a minimization method such as the Amoeba or Nlfit was used to find f it needs to be rerun with the point omitted. If Lagrange interpolation was used to find f, the interpolation needs to be made again with the point omitted. Since the loss of a single point does not move Nlfit or the Amoeba far from equilibrium, the new value of f – with a missing point - can usually be found with only a few steps from the original – with all points. In the case of the Amoeba the full output array should be used as input for each to the new sets. Convergence should be relatively rapid since one is nearly at the minimum in each case. As long as special points such as the ends are removed, it is always possible to "randomly" select M of the points and then to multiply by N/M to arrive at the final error estimate.

Replacing the points

If the distribution of the points x_i is known is possible to directly estimate

$$\sigma_f^2=\frac{1}{M}\sum_{i=1}^{M}\left(f\left(x_1,x_2,...,x_N\right)-f\left(\hat{x}_1.\hat{x}_2,...,\hat{x}_N\right)\right)^2$$

(3.1)

This is done by selecting all N x's as the expected values + a random number with a Gaussian distribution with the standard deviation of the point x_i. An easy means of making this selection is given below. It requires at least M=10 to cover the odd chances and most references recommend 25 to 30. Note that in this case the values are averaged, not summed as in the previous case.

Psuedo Random numbers

Random numbers are not wanted. What is desired is a set of completely reproducible numbers that are the same on all computers and in both C and Fortran. In addition, it must be true that the numbers give all averages the same as would be given by actual random numbers. In practice the numbers should be re-initialized with the numbers needed to recreate the desired set written out as

17894561 3678991
18345899 234567899
...

In this way, when the code bombs after 23 hours of running, the situation just before the bomb can be recreated by simply entering these numbers.

With this many random numbers, the "randomness" of the set can be a problem for some calculations[1]. The method used here, first suggested by MacLaren and Marsaglia[2], is to insert numbers into a table with a multiplicative congruential method with one multiplier and seed and to extract them from the table with a multiplicative congruential method with a different multiplier and seed.

random.for contains the common
COMMON/RAN/IX,IY,ITAB(128)
random.c random.h trandom.c
The file random.h contains

```
struct {int ix,iy,itab[128];} ran;
int rseed(int ix,int iy);
double rndmf();
```

The seeds are ix and iy. The random number set is initialized by calling rseed(ix,iy). This call causes 128 numbers to be fed into the table itab[128] with one random number generator. The call x=rndmf() gives a random number base on ix,iy and itab, between 0 and 1. It selects this by advancing ix and iy, pulling a value from a randomly selected table entry and replacing that with a new value.

---

[1] R. L. Coldwell, "Correlational Defects in the Standard IBM 360 Random Number Generator and the Classical Ideal Gas Correlation Function", J. Computational Phys. **14**, 223 (1974).

[2] M. D. MacLaren and G. Marsaglia, *J. Assoc. Comput. Mach.* **12** (1965),83

This can be re-initialized by printing out ix and iy and then calling rseed. This wastes 128 random numbers, but gives a completely determined new set of random numbers.

Gaussian distribution[3]

```
Rgauss=-6;
For(i=0;i<12;++i){Rgauss+=rndmf();}
```

$$\langle f \rangle = \int_0^1 x dx = \frac{1}{2}$$
$$\langle f^2 \rangle = \int_0^1 x^2 dx = \frac{1}{3} \quad (4.1)$$

Thus

$$\sigma^2 = \langle f^2 \rangle - \langle f \rangle^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12} \quad (4.2)$$

or

$$12\langle f \rangle - 6 = \sum_{i=1}^{12} rndmf(\,) - 6 \pm 1 \qquad (4.3)$$

This means that the above sum is 0 +- 1. If I want to produce a peak of height P = 36 +- 6 with a Gaussian distribution on the six. I simply take

$$P = 36 + 6 * \left\{ -6 + \sum_{i=1}^{12} rndmf(\,) \right\} \quad (4.4)$$

Simple Monte Carlo Integration

$$< f >= \int_a^b f(x) dx \qquad (5.1)$$

Change variables such that x=a+(b-a)*x'. dx = (b-a)dx'

$$< f >= (b-a) \int_0^1 f(a+(b-a)x) dx \qquad (5.2)$$

now select x positions uniformly and randomly between 0 and 1 and average them so that

$$x_i = Rand(1.)$$
$$< f >= \underset{N \to \infty}{(b-a)\mathcal{L}} \frac{1}{N} \sum_{i=1}^{N} f(a+(b-a)x_i) \quad (5.3)$$

"obvious" error analysis

The expected error in $< f >_N$ can be estimated by considering each random selection as an independent estimator of $< f >_N$ and then averaging the squares of the differences to yield

$$\delta_N^2 = \frac{1}{N} \sum_{i=1}^{N} \left(f(b+(b-a)x_i) - < f >_N \right)^2$$
$$= \frac{1}{N} \sum_{i=1}^{N} f^2(a+(b-a)x_i)$$
$$- \frac{2 < f >_N}{N} \sum_{i=1}^{N} f(a+(b-a)x_i) + \frac{< f >_N^2}{N} \sum_{i=1}^{N} 1 \quad (6.1)$$
$$= < h^2 >_N - 2 < f >_N^2 + < f >_N^2$$
$$= < h^2 >_N - < h >_N^2$$

as the rms difference between $< h >$ and any single estimator. The expectation value of the sum is then N times δ_N so that the deviation in the sum is

$$\varepsilon_{sum}^2 = N \delta_N^2 \qquad (6.2)$$

In addition, the expected error in the sum is

$$\varepsilon_{sum} = \sqrt{N} \delta_N \qquad (6.3)$$

while the expected error in the average is

$$\sigma_N = \frac{\varepsilon_{sum}}{N} = \frac{\delta_N}{\sqrt{N}} \qquad (6.4)$$

so that

$$< f >= < f >_N \pm \frac{< f^2 > - < f >^2}{\sqrt{N}} \qquad (6.5)$$

"more elegant" error analysis

$$\langle f \rangle_N - \langle f \rangle_{nok} = \frac{1}{N} \sum_{i=1}^{N} f_i - \frac{1}{N-1} \sum_{\substack{i=1 \\ i \neq k}}^{N} f_i \quad (7.1)$$

Then

$$\langle f \rangle_N - \langle f \rangle_{nok} = \frac{1}{N} \sum_{i=1}^{N} f_i - \frac{1}{N\left(1 - \frac{1}{N}\right)} \sum_{\substack{i=1 \\ i \neq k}}^{N} f_i$$
$$= \frac{1}{N} \sum_{i=1}^{N} f_i - \frac{1}{N} \left(1 + \frac{1}{N}\right) \sum_{\substack{i=1 \\ i \neq k}}^{N} f_i \qquad (7.2)$$
$$= \frac{1}{N} \sum_{i=1}^{N} f_i - \frac{1}{N} \left(1 + \frac{1}{N}\right) \sum_{i=1}^{N} f_i + \frac{1}{N} \left(1 + \frac{1}{N}\right) f_k + h.o.$$

Finally

$$\langle f \rangle_N - \langle f \rangle_{nok} = -\frac{1}{N} \left(\frac{1}{N}\right) \sum_{i=1}^{N} f_i + \frac{f_k}{N} = \frac{1}{N} \left(f_k - \langle f \rangle \right) \quad (7.3)$$

Then using equation (3.1)

$$\sigma_{\langle f \rangle}^2 = \frac{1}{N^2} \sum_{k=1}^{N} \left(f_k - \langle f \rangle \right)^2$$
$$= \frac{1}{N^2} \left\{ \sum_{k=1}^{N} f_k^2 \right\} - \frac{2\langle f \rangle}{N^2} \sum_{k=1}^{N} f_k + \frac{\langle f \rangle^2}{N^2} \sum_{k=1}^{N} 1 \quad (7.4)$$
$$= \frac{1}{N} \left\{ \langle f^2 \rangle - \langle f \rangle^2 \right\}$$

..\..\optimization\amoeba\for\trandom.f ..\for\RANDOM.FOR

Expectation value error estimates.

[3] J. M Hammersley and D. C. Handscomb, **Monte Carlo Methods,** John Wiley (1964) p.39

In general, we frequently do not really want an integral, but rather an expectation value. That is the quantity of interest is usually

$$< H >= \frac{\int \Psi H \Psi d\tau}{\int \Psi^2 d\tau} = \frac{\sum_{i=1}^{i=N} \Psi_i H \Psi_i / w_i}{\sum_{i=1}^{i=N} \Psi_i^2 / w_i} \quad (8.1)$$

which can easily be seen to have less error than that indicated by the two integrals. For example if neither the numerator or denominator samples a position where Ψ is large, the ration of the two still has some meaning. Especially is this true if HΨ/Ψ is on the order of -9, while Ψ itself varies by many orders of magnitude. We begin the error estimate by calculating the change in <H> caused by leaving out a single one of the Monte Carlo points.

$$\begin{aligned}
\delta_j &=< H > - \frac{\sum \Psi_i H \Psi_i / w_i - \Psi_j H \Psi_j / w_j}{Norm\left(1 - \Psi_j^{\,2} / \left(w_j Norm\right)\right)} \\
&=< H > - \frac{Norm < H > - \Psi_j H \Psi_j / w_j}{Norm} \times \\
&\quad \left(1 + \Psi_j^{\,2} / \left(w_j Norm\right) + \left(\Psi_j^{\,2} / \left(w_j Norm\right)\right)^2 + \cdots\right) \\
&= - < H > \Psi_j^{\,2} / w_j / Norm + \Psi_j H \Psi_j / w_j / Norm + O\left(\left(\Psi_j^{\,2} / \left(w_j Norm\right)\right)^2\right) \\
&= \frac{\Psi_j H \Psi_j / w_j - < H > \Psi_j^{\,2} / w_j}{Norm} + O\left(\left(\Psi_j^{\,2} / \left(w_j Norm\right)\right)^2\right)
\end{aligned}$$

(8.2)

and it looks basically the same as with the integral except that instead of a -1 in the denominator there is a $-\Psi_j^2/w_j$ which is assumed to be small compared to the sum over all terms so that we expand it and hold only those terms of order Ψ^2/w/sum to find.

$$\delta < H >_j = \frac{\left(\Psi_j / w_j\right)\left(H\Psi_j - < H > \Psi_j\right)}{\sum \Psi_j^2 / w_j} \quad (8.3)$$

and of course the total error squared in the expectation value of H is the sum of the squares of these terms so that

$$\sigma^2_{<H>} = \frac{\sum \left(\Psi_j / w_j\right)^2 \left(H\Psi_j - < H > \Psi_j\right)^2}{\left(\sum \Psi_j^2 / w_j\right)^2} \quad (8.4)$$

or in the limit N→very large so that we can make the sums integrals.

$$\sigma^2_{<H>} = \frac{\int (1/w(x)) \Psi^2 \left(H\Psi - < H > \Psi\right)^2 d\tau}{\left(N \int \Psi^2 d\tau\right)^2}$$

(8.5)

and we notice two items of interest.

A. If Ψ is an eigenfunction, the second term in the integral is zero. and there is no Monte Carlo error in the estimate of the eigenvalue.

B. The guiding function w is still present, even in the infinite limit (note however that (1/N) does drive the error slowly to zero even in this limit. The best choice of w will be one that most closely mimics the second term which is the positive definite miss^2 of the eigenfunction, but in general is merely the are which contributes to those parts different from the average expectation value of an operator.

Capping an infinity.

A common integrand of interest is $1/r_{ij}$ where it is not normally appropriate to use r_i as the location from which to sample the integrand.

We do know however that for short distances in the r_{ij} space the integrand is

$$I = \int_0^{\varepsilon} r_{ij}^{\,2} \frac{1}{r_{ij}} \Psi^2(r_i, r_j, r_{ij}) dr_{ij} \quad (9.1)$$

we want to replace the $1/r_{ij}$ by an integrand with the same integral, but without the singularity. For small ε, assume that Ψ is constant so that the integral is

$$I = \Psi^2(r_i, r_j, 2\varepsilon/3) \int_0^{\varepsilon} r_{ij}^{\,2} c dr_{ij} = \Psi^2(r_i, r_j, 2\varepsilon/3) \int_0^{\varepsilon} r_{ij}^{\,2} \frac{1}{r_{ij}} dr_{ij}$$

$$\Rightarrow c = \int_0^{\varepsilon} r_{ij}^{\,2} \frac{1}{r_{ij}} dr_{ij} \Big/ \int_0^{\varepsilon} r_{ij}^{\,2} dr_{ij} = \frac{\varepsilon^2/2}{\varepsilon^3/3} = \frac{3}{2\varepsilon}$$

(9.2)

which implies that $1/r_{ij}$ should be replaced by $1/(2\varepsilon/3)$ for any r_{ij} less than ε

Consider a 3d integral over all space

$$< f >= \iiint d^3 r f(\vec{r}) = 4\pi \int_0^1 d\bar{\mu} \int_0^1 d\bar{\varphi} \int_0^{\infty} r^2 f(\vec{r}) dr \quad \textbf{(10.1)}$$

$$\mu = 2\bar{\mu} - 1; \quad \varphi = 2\pi\bar{\varphi}$$

Change variables to

$$y = \int_0^r g(z) dz \Big/ \int_0^{\infty} g(z) dz \qquad \textbf{(10.2)}$$

Problem 1 – Monte Carlo

Calculate $<1/r_{ij}>$ and $<(1/r_{ij})^2>$ for electrons in 1s and 2s states hydrogen like orbitals of helium.

$\Psi(r_1,r_2)=\exp(-2r_1)(2-r_2)\exp(-r_2)$.

For p orbitals

$\Psi(r_1,r_2)=\exp(-2r_1)x_2\exp(-r_2)$

The choice of g can become involved. What we want is a crude estimate along with an estimate of the standard deviation. Note first that a crude estimate of <H> in (8.4) **is sufficient. Don't bother to calculate the error for the first 10000 points.**

Then Use 8.4 for the rest of the million points. Watch out for the infinity, it needs to be capped.

Problem 2 – Trap rule

Evaluate $<1/r_{ij}>$ and $<(1/r_{ij})^2>$ for electrons in 1s and 2s states of helium. Do the phi1, phi2 and theta1 and theta2 integrals by mid-point trap rule or by inspection. Use the Laurent transform followed by mid-point trap to evaluate the integrations in r1 and r2.

Wave function details are in **..\..\diffeqns\WAVEFUN.htm**