Montecarlo techniques

From random walk to simulated annealing

Summary

- A few reminders of statistics
  - Average, variance, correlation
  - Central limit theorem
- Pseudo-random numbers
- Metropolis algorithm
  - Random walk
  - Detailed balance
  - Self-correlation
- Simulated annealing
  - Travelling salesman

Mean and variance: one variable

$$\mu(X) = \int_{-\infty}^{+\infty} x \; f(x)dx$$

$$\sigma^2(X) = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)dx$$

Two variables

$$Y_2 = X_1 + X_2$$

$$\mu(Y_2) = \int_{-\infty}^{+\infty} dx_1 \int_{-\infty}^{+\infty} dx_2 (x_1 + x_2) f(x_1, x_2)$$

$$f_1(x_1) = \int_{-\infty}^{+\infty} dx_2 f(x_1, x_2)$$

$$f_2(x_2) = \int_{-\infty}^{+\infty} dx_1 f(x_1, x_2)$$

$$\begin{aligned}
\mu(Y_2) &= \int_{-\infty}^{+\infty} dx_1 \; x_1 \int_{-\infty}^{+\infty} dx_2 \; f(x_1, x_2) + \\
&\quad \int_{-\infty}^{+\infty} dx_1 \; \int_{-\infty}^{+\infty} dx_2 \; x_2 \; f(x_1, x_2) \\
&= \int_{-\infty}^{+\infty} dx_1 \; x_1 \; f_1(x_1) + \int_{-\infty}^{+\infty} dx_2 \; x_2 f_2(x_2) \\
&= \mu(X_1) + \mu(X_2).
\end{aligned}$$

$$\sigma^2(Y_2) = \sigma^2(X_1) + \sigma^2(X_2) + 2 \times \mathrm{cov}[X_1, X_2]$$

$$\mathrm{cov}(X_1, X_2) = \int_{-\infty}^{+\infty} dx_1 \; \int_{-\infty}^{+\infty} dx_2 \; [x_1 - \mu_1] [x_2 - \mu_2] f(x_1, x_2)$$

$$f(x_1, x_2) = f_1(x_1) \times f_2(x_2)$$

$$\mathrm{cov}(X_1, X_2) = 0$$

N independent variables sharing the same distribution

$Y_N = X_1 + X_2 + \ldots + X_N$

$Z_N = (X_1 + X_2 + \ldots + X_N)/N$

$W_N = (X_1 + X_2 + \ldots + X_N)/\sqrt{N}$

$\sigma^2_{Y_N} = N\,\sigma^2_X$

$\sigma^2_{Z_N} = \sigma^2_X/N$

$\sigma^2_{W_N} = \sigma^2_X$

Chebyshev inequality

$$\mathcal{P}\left\{|X-\mu| \geq k\sigma\right\} \leq \frac{1}{k^2}$$

$$\begin{aligned}
\sigma^2 &= \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)\, dx \\
&\geq \int_{-\infty}^{\mu-k\sigma} (x-\mu)^2 f(x)\, dx + \int_{\mu+k\sigma}^{+\infty} (x-\mu)^2 f(x)\, dx \\
&\geq k^2\sigma^2 \left[\int_{-\infty}^{\mu-k\sigma} f(x)\, dx + \int_{\mu+k\sigma}^{+\infty} f(x)\, dx \right] \\
&= k^2\sigma^2 \mathcal{P}\left\{|X-\mu| \geq k\sigma\right\} \quad .
\end{aligned}$$

$$\bar{y}_N \equiv \left(x_1 + x_2 + \cdots + x_N\right)/N$$

$$P\left\{\left|\bar{y}_N - \mu\right| \geq K\sigma_{\bar{y}_N}\right\} \leq \frac{1}{K^2}$$

$$\sigma_{\bar{y}_N} = \sigma/\sqrt{N}$$

$$\varepsilon \equiv K\sigma_{\bar{y}_N} = \frac{K\sigma}{\sqrt{N}} \Rightarrow \frac{1}{K} = \frac{\sigma}{\sqrt{N}\,\varepsilon}$$

$$P\left\{\left|\bar{y}_N - \mu\right| \geq \varepsilon\right\} \leq \frac{\sigma^2}{N\varepsilon^2}$$

Central limit theorem

Let $X_1, X_2, \cdots X_N$ be N independent and identically distributed random variables having a common Gaussian probability density with mean zero and variance σ^2, given by,

$$G_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{x^2}{2\sigma^2}\right]. \qquad (76)$$

$Y_N = X_1 + X_2 + \cdots X_N$

$$G_{Y_N}(x) = \frac{1}{\sigma\sqrt{2\pi N}} \exp\left[-\frac{x^2}{2N\sigma^2}\right]$$

with mean zero and variance $N\sigma^2$

The CLT states that a gaussian distribution is reached for N→∞ even starting from a non-gaussian distribution, as long as the original distribution has finite variance

Pseudo-random numbers

Goal: to generate points distributed according to a given function representing a density of probability.

It will then be «easy» to find the maximum of that function.

Linear congruential generators:

$$R_{i+1} = aR_i + b \pmod{\mathrm{m}}$$

Possible problems:
Short cycles
Marsaglia planes

A very short cycle

Table 3: Linear congruential recursion with $a = 5$, $b = 1$, $m = 100$, and $R_1 = 1$.

R_1								=	1
R_2	=	$(5\times$	1	$+1)$	(mod 100) =	6	(mod 100) =		6
R_3	=	$(5\times$	6	$+1)$	(mod 100) =	31	(mod 100) =		31
R_4	=	$(5\times$	31	$+1)$	(mod 100) =	156	(mod 100) =		56
R_5	=	$(5\times$	56	$+1)$	(mod 100) =	281	(mod 100) =		81
R_6	=	$(5\times$	81	$+1)$	(mod 100) =	406	(mod 100) =		6

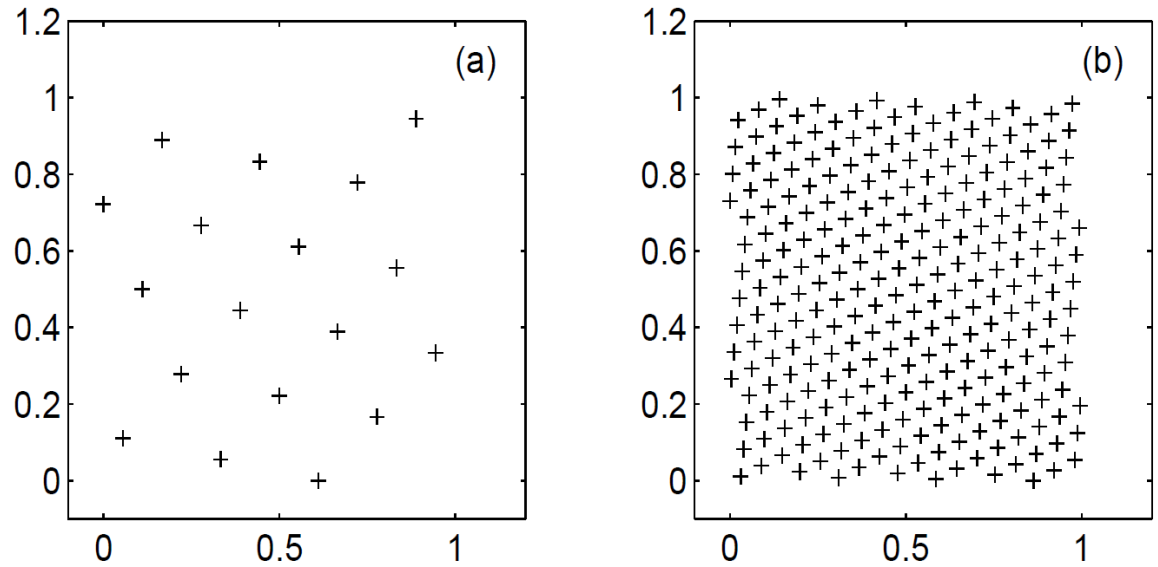


Figure 8: Random numbers of the linear congruential generator embedded in a two dimensional phase space. (a) depicts the results for the generator with $a = 7$, $b = 13$ and $m = 18$. (b) depicts the results for the generator with $a = 137, b = 187$ and $m = 256$. (Both these generators give full period)

If d-tuples $(\xi_1, \xi_2, \cdots, \xi_d)$, $(\xi_2, \xi_3, \cdots, \xi_{d+1})$, $(\xi_3, \xi_4, \cdots, \xi_{d+2})$, $\cdots$, of the random numbers produced by linear congruential generators are viewed as points in the unit hyper cube of d dimensions, then all the points will be found to lie in a relatively small number of parallel hyper planes. Such structures occur with any near congruential generator and in any dimension.

There is good evidence, both theoretical and empirical, that the simple multiplicative congruential algorithm

$$I_{j+1} = aI_j \pmod{m} \tag{7.1.2}$$

can be as good as any of the more general linear congruential generators that have $c \neq 0$ (equation 7.1.1) — *if* the multiplier a and modulus m are chosen exquisitely carefully. Park and Miller propose a "Minimal Standard" generator based on the choices

$$a = 7^5 = 16807 \qquad m = 2^{31} - 1 = 2147483647 \tag{7.1.3}$$

First proposed by Lewis, Goodman, and Miller in 1969, this generator has in subsequent years passed all new theoretical tests, and (perhaps more importantly) has accumulated a large amount of successful use. Park and Miller do not claim that the generator is "perfect" (we will see below that it is not), but only that it is a good minimal standard against which other generators should be judged.

It is not possible to implement equations (7.1.2) and (7.1.3) directly in a high-level language, since the product of a and $m - 1$ exceeds the maximum value for a 32-bit integer. Assembly language implementation using a 64-bit product register is straightforward, but not portable from machine to machine. A trick due to Schrage [2,3] for multiplying two 32-bit integers modulo a 32-bit constant, without using any intermediates larger than 32 bits (including a sign bit) is therefore extremely interesting: It allows the Minimal Standard generator to be implemented in essentially any programming language on essentially any machine.

Schrage's algorithm is based on an *approximate factorization* of m,

$$m = aq + r, \quad \text{i.e.,} \quad q = [m/a],\ r = m \bmod a \tag{7.1.4}$$

with square brackets denoting integer part. If r is small, specifically $r < q$, and $0 < z < m - 1$, it can be shown that both $a(z \bmod q)$ and $r[z/q]$ lie in the range $0, \ldots, m - 1$, and that

$$az \bmod m = \begin{cases} a(z \bmod q) - r[z/q] & \text{if it is} \geq 0, \\ a(z \bmod q) - r[z/q] + m & \text{otherwise} \end{cases} \tag{7.1.5}$$

The application of Schrage's algorithm to the constants (7.1.3) uses the values $q = 127773$ and $r = 2836$.

From “Numerical recipes”, Press et al.

The routine ran0 is a Minimal Standard, satisfactory for the majority of applications, but we do not recommend it as the final word on random number generators.

Our reason is precisely the simplicity of the Minimal Standard. It is not hard to think of situations where successive random numbers might be used in a way that accidentally conflicts with the generation algorithm. For example, since successive numbers differ by a multiple of only 1.6×10^4 out of a modulus of more than 2×10^9 very small random numbers will tend to be followed by smaller than average values. One time in 10^6 for example, there will be a value $< 10^{-6}$ returned (as there should be), but this will *always* be followed by a value less than about 0.0168. One can easily think of applications involving rare events where this property would lead to wrong results.

ran1, uses the Minimal Standard for its random value, but it shuffles the output to remove low-order serial correlations. A random deviate derived from the *j*th value in the sequence, *Ij*, is output not on the *j*th call, but rather on a randomized later call, *j* +32 on average.

Transformation method

$$p(x)dx = \begin{cases} dx & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

y=y(x) is a given function, then:

$$|p(y)dy| = |p(x)dx|$$

$$p(y) = p(x)\left|\frac{dx}{dy}\right|$$

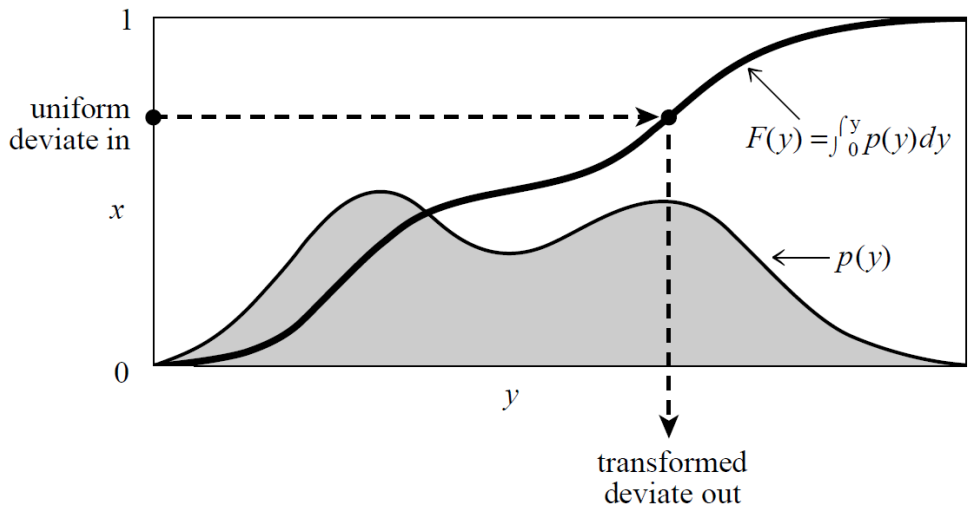
An example:

suppose that $y(x) \equiv -\ln(x)$

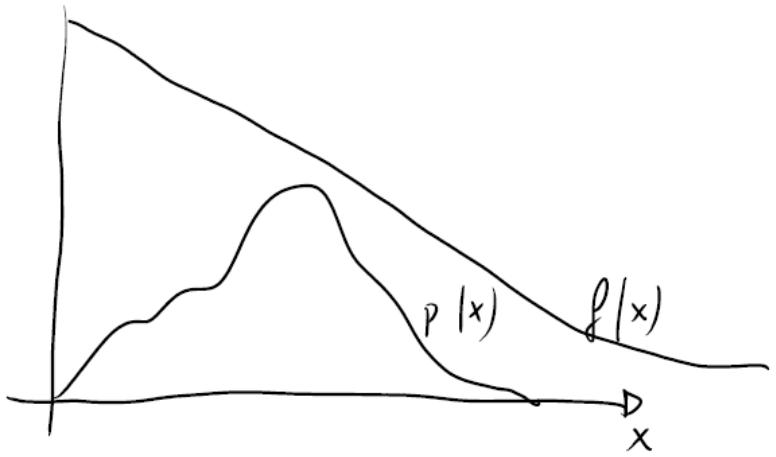
$$p(y)dy = \left|\frac{dx}{dy}\right| dy = e^{-y}dy$$

Viceversa: we want to find y=y(x) so that p(y) is a given f(y)

$$\frac{dx}{dy} = f(y) \qquad y(x) = F^{-1}(x)$$



Rejection method



f(x)>p(x)
Generate numbers x_i according to f(x)
Generate a number $0 \leq y_i \leq f(x_i)$
Take x_i if $y_i \leq p(x_i)$
The x_i are distributed as p(x)

See rejection.py

Special technique for gaussian distributions

Is based on Maxwell distribution of energies:

$$e^{-\frac{E}{kT}} = e^{-m\,v^2/(2\,kT)}$$

$$\begin{aligned} v_i(\text{new}) &= \frac{v_j(\text{old}) + v_i(\text{old})}{\sqrt{2}} \\ v_j(\text{new}) &= \frac{v_j(\text{old}) - v_i(\text{old})}{\sqrt{2}}. \end{aligned}$$

Metropolis

Let $f(x)$ be a function that is proportional to the desired probability distribution $P(x)$ (a.k.a. a target distribution).

1. Initialization: Choose an arbitrary point x_0 to be the first sample, and choose an arbitrary probability density $\boldsymbol{g(x|y)}$ (sometimes written $\boldsymbol{Q(x|y)}$) that suggests a candidate for the next sample value x, given the previous sample value y. For the Metropolis algorithm, g must be symmetric; in other words, it must satisfy $\boldsymbol{g(x|y) = g(y|x)}$. A usual choice is to let $\boldsymbol{g(x|y)}$ be a Gaussian distribution centered at y, so that points closer to y are more likely to be visited next—making the sequence of samples into a random walk. The function g is referred to as the *proposal density* or *jumping distribution*.
2. For each iteration t:
   - Generate a candidate x' for the next sample by picking from the distribution $\boldsymbol{g(x'|x_t)}$.
   - Calculate the *acceptance ratio* $\alpha = f(x')/f(x_t)$, which will be used to decide whether to accept or reject the candidate. Because f is proportional to the density of P, we have that $\alpha = f(x')/f(x_t) = P(x')/P(x_t)$.
   - If $\alpha \geq 1$, then the candidate is more likely than x_t; automatically accept the candidate by setting $x_{t+1} = x'$. Otherwise, accept the candidate with probability α; if the candidate is rejected, set $x_{t+1} = x_t$, instead.

$x = y + r \Delta$

An equilibrium is reached

$$\Delta n_i = n_j P_{j \to i} - n_i P_{i \to j}$$

$$= n_j P_{i \to j} \left[\frac{P_{j \to i}}{P_{i \to j}} - \frac{n_i}{n_j} \right]$$

$$\text{IF} \quad \frac{n_i}{n_j} > \frac{P_{j \to i}}{P_{i \to j}} \Rightarrow \Delta n_i < 0$$

The equilibrium corresponds to the given density of probability

$$P_{i \to j} = g_{i \to j} \, A_{i \to j}$$

assume $g_{i \to j} = g_{j \to i}$

$$A_{i \to j} = \min\left[1, \frac{f_j}{f_i}\right]$$

$$\frac{P_{j \to i}}{P_{i \to j}} = \frac{A_{j \to i}}{A_{i \to j}} = \frac{f_i}{f_j} \Rightarrow \frac{n_i}{n_j} = \frac{f_i}{f_j}$$

at equilibrium

If $f_j / f_i < 1$ then
$A_{i->j} = f_j / f_i$
$A_{j->i} = 1$
$A_{j->i} / A_{i->j} = f_i / f_j$

If $f_j / f_i > 1$ then
$A_{i->j} = 1$
$A_{j->i} = f_i / f_j$
$A_{j->i} / A_{i->j} = f_i / f_j$ again

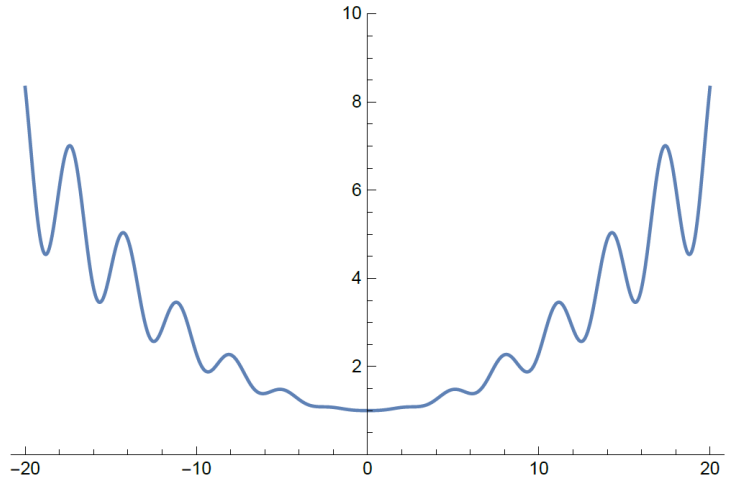
Simulated annealing

$$\text{Prob}\,(E) \sim \exp(-E/kT)$$

To minimize E means to maximize Prob(E)
Large T: big jumps
Small T: small jumps
→ Gradually reduce T after a given number of iterations

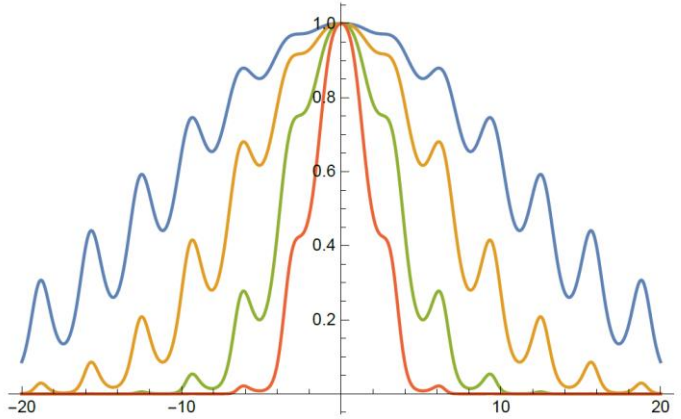
```
In[7]:= e[x_] := 0.01 x^2 (1 + Sin[x]^2) + 1
In[9]:= Plot[e[x], {x, -20, 20}, PlotRange → {0, 10}]
```

Out[9]=




```
In[5]:= f[x_, t_] := Exp[-e[x] / t]
In[14]:= Plot[{f[x, 3] / f[0, 3], f[x, 1] / f[0, 1],
    f[x, 0.3] / f[0, 0.3], f[x, 0.1] / f[0, 0.1]}, {x, -20, 20}]
```

Out[14]=



Travelling salesman

1. *Configuration.* The cities are numbered $i = 1 \ldots N$ and each has coordinates (x_i, y_i). A configuration is a permutation of the number $1 \ldots N$, interpreted as the order in which the cities are visited.

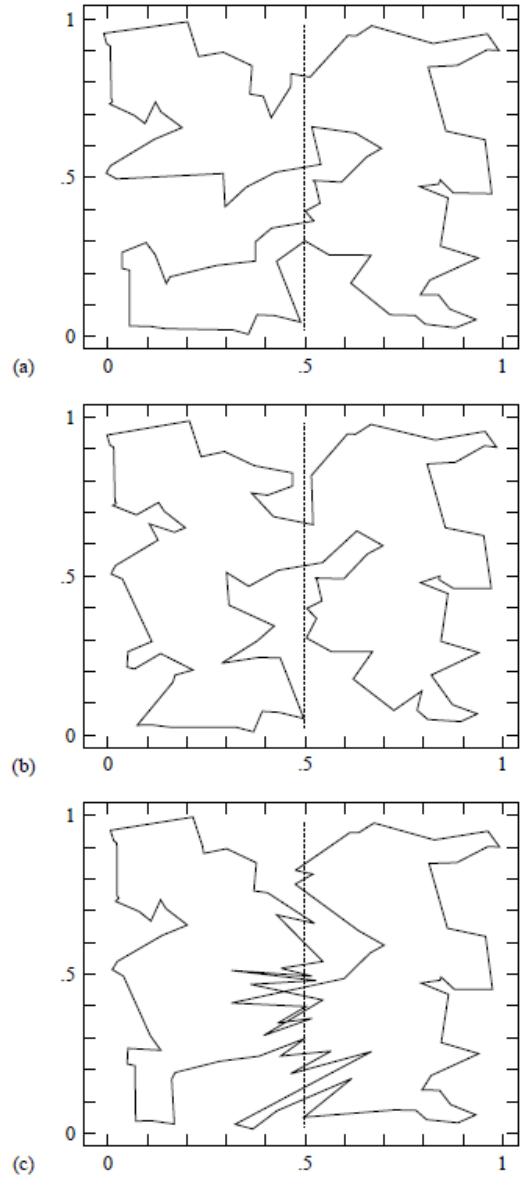
2. *Rearrangements.* An efficient set of moves has been suggested by Lin [6]. The moves consist of two types: (a) A section of path is removed and then replaced with the same cities running in the opposite order; or (b) a section of path is removed and then replaced in between two cities on another, randomly chosen, part of the path.

3. *Objective Function.* In the simplest form of the problem, E is taken just as the total length of journey,

$$E = L \equiv \sum_{i=1}^{N} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \qquad (10.9.2)$$

$$E = \sum_{i=1}^{N} \left[\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} + \lambda(\mu_i - \mu_{i+1})^2 \right]$$

4. *Annealing schedule.* This requires experimentation.

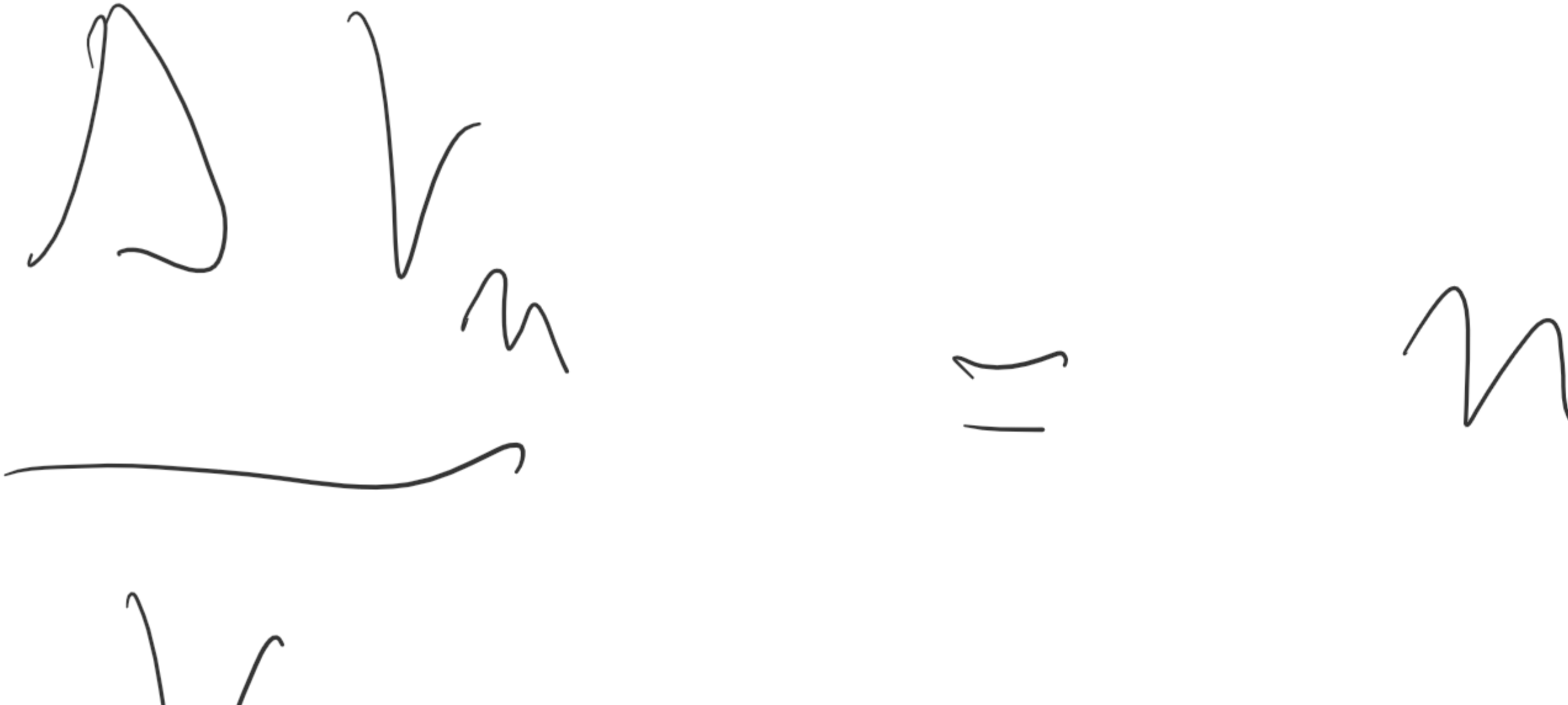

1
.5
0
0
.5
1
(a)
1
.5
0
0
.5
1
(b)
1
.5
0
0
.5
1
(c)

Simple Montecarlo integration

$$\int f \, dV \approx V \langle f \rangle \pm V \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}} \tag{7.6.1}$$

Here the angle brackets denote taking the arithmetic mean over the N sample points,

$$\langle f \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i) \qquad \langle f^2 \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f^2(x_i) \tag{7.6.2}$$



Autocorrelation function

For a discrete process with known mean and variance for which we observe n observations {X1, X2, … Xn}, an estimate of the autocorrelation may be obtained as:

$$\hat{R}(k) = \frac{1}{(n-k)\sigma^2} \sum_{t=1}^{n-k} (X_t - \mu)(X_{t+k} - \mu)$$

for any positive integer k<n. When the true mean μ and variance σ^2 are known, this estimate is unbiased. If the true mean and variance of the process are not known if μ and σ^2 are replaced by the standard formulae for sample mean and sample variance, then this is a biased estimate.