Computational Finance – The Monte Carlo Method for Pricing

Before even begining, we will mention that Monte Carlo methods in finance are a huge area of research and we only touch on this topic to give a taste of some techniques available. Naturally many more details may be obtained from the vast literature, and we suggest [**?**] as a good place to start.

1 Some Elements from Probability Theory

Monte Carlo is a probabilistic technique and so we summarize some basic facts from probability theory which are useful. For a random variable $\mathbf{X} \in \mathbb{R}^n$, we denote its probabiliy density function $p_{\mathbf{X}}(\mathbf{x})$. For a subset $S \subseteq \mathbb{R}^n$,

$$P[\mathbf{X} \in S] = \int_S d\mathbf{x}\ p_{\mathbf{X}}(\mathbf{x}).$$

Two random variables $\mathbf{X}$ and $\mathbf{Y}$ are independent if for any subsets $S_1 \subseteq \mathbb{R}^n$, $S_1 \subseteq \mathbb{R}^n$

$$P[\mathbf{X} \in S_1, \mathbf{Y} \in S_2] = P[\mathbf{X} \in S_1] \cdot P[\mathbf{Y} \in S_2].$$

Let $f : R^n \to \mathbb{R}$. Then $f(\mathbf{X})$ is itself a random variable with its own probability distribution. The expectation $E[f(\mathbf{X})]$ is given by

$$E[f(\mathbf{X})] = \int d\mathbf{x}\ p_{\mathbf{X}}(\mathbf{x}) f(\mathbf{x}).$$

We will focus on a 1-dimensional random variable X with density $p_X(x)$. The mean is given by setting $f(X) = X$,

$$E[X] = \int d\mathbf{x}\ p_X(x) x.$$

The variance is given by

$$Var[X] = E[(X - E[X])^2] = E[X^2] - E[X]^2,$$

where

$$E[X^2] = \int d\mathbf{x}\ p_X(x) x^2.$$

In the multi-dimensional case we define the i^{th} component of the mean vector $\boldsymbol{\mu}$ by selecting $f(\mathbf{X}) = X_i$, the i^{th} component of $\mathbf{X}$.

$$\mu_i = E[X_i] = \int d\mathbf{x}\ p_X(x) x_i.$$

Similarily, we define the *covariance matrix* Σ where

$$\Sigma_{ij} = Cov[X_i, X_j] = E[X_i X_j] - E[X_i]E[X_i].$$

Note that $\Sigma_{ii} = Var[X_i]$. One of the most important distributions in computational finance is the Normal distribution. $\mathbf{X} \sim N(\mu, \Sigma)$ is a normally distributed random vector with mean $\boldsymbol{\mu}$ and covariance matrix Σ if

$$p_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

It is not hard to show that $E[\mathbf{X}] = \boldsymbol{\mu}$ and that $Cov[X_i, X_j] = \Sigma_{ij}$.

Exercise 1.1

For a Normal random vector as described above, show that $E[\mathbf{X}] = \boldsymbol{\mu}$ and that $Cov[X_i, X_j] = \Sigma_{ij}$.

1.1 Sums of Random Variables

Suppose that $X_1, \ldots, X_n$ are 1-dimensional random variables, and let $X = \sum_{i=1}^n X_i$ be the random variable equal to the sum. Then

$$E[X] = \sum_{i=1}^{n} E[X_i].$$

If two random variables X_1, X_2 are independent, then $Cov[X_1, X_2] = 0$. In general if $Cov[X_1, X_2] = 0$ (which does not imply independence), the random variables are uncorrelated. Suppose that $X_1, \ldots, X_n$ are *independent* random variables, then

$$E\left[\prod_{i=1}^{n} f_i(X_i)\right] = E[f_1(X_1)f_2(X_2)\cdots f_n(X_n)] = \prod_{i=1}^{n} E[f_i(X_i)],$$

i.e. the expectation of products of functions of the random variables factorizes to a product of expectations. In particular, for independent random variables,

$$E[X_1 X_2] = E[X_1]E[X_2].$$

We can compute the variance of a sum as

$$Var[X] = \sum_{i=1}^{n} Var[X_i] + \sum_{i \neq j} Cov[X_i, X_j].$$

For independent random variables, the covarince term is zero among all pairs, and so

$$Var[X] = \sum_{i=1}^{n} Var[X_i] \qquad \text{(independent random variables)}.$$

The standard deviation $\sigma_X = \sqrt{Var[X]}$ is typically a good measure of the dispersion of X. For example, for the normal distribution, with probability greater than 0.997, the random variable X lies within 3σ of the mean. If X is a random variable then $Y = aX + b$, a linear transformation of X is also a random variable. Then,

$$\begin{aligned} E[Y] &= E[aX + b], \\ &= aE[X] + b, \\ Var[Y] &= Var[aX + b], \\ &= a^2 Var[X]. \end{aligned}$$

We can perform a similar calculation for a linear transformation of a random vector $\mathbf{X}$. Let $\mathbf{Y} = A\mathbf{X} + \boldsymbol{b}$, where A is a matrix and $\boldsymbol{b}$ is a vector. Let Σ_X and $\Sigma_{\mathbf{Y}}$ be the covariance matrix for $\mathbf{X}$ and $\mathbf{Y}$ respectively. Then,

$$\begin{aligned} E[\mathbf{Y}] &= E[A\mathbf{X} + \boldsymbol{b}], \\ &= AE[\mathbf{X}] + \boldsymbol{b}, \\ \Sigma_{\mathbf{Y}} &= Cov[A\mathbf{X} + \boldsymbol{b}], \\ &= A\Sigma_{\mathbf{X}} A^{\mathrm{T}}. \end{aligned}$$

If $\mathbf{X}$ is a normally distributed random vector, then any linear transformation $\mathbf{Y} = A\mathbf{X} + \boldsymbol{b}$ is also a normally distributed random vector, with the appropriately defined mean and covariance matrix. Note that if a random variable is bounded, so that $X \in [a, b]$, then $Var(X) \leq (b - a)^2$.

Exercise 1.2

Show that if $X \in [a, b]$ then $Var(X) \leq (b - a)^2$.

1.2 Probability Bounds

Various powerful inequality bounds exist which relate the probability of a random variable taking on certain values to quantities like its expectaiton. These are typically called tail inequalities.

Markov's Inequality

$$P[|X| > t] < \frac{E[|X|]}{t}.$$

Chebyshev's Inequality

$$P[|X - E[X]| > k\sigma_X] \leq \frac{1}{k^2}.$$

This bound shows that the standard deviation is a good measure of how good a random variable is at approximating the mean. Typically, in statistics, one uses 2σ as an error bar when using the realization of a random variable to approximate its mean.

Exercise 1.3

Prove Markov's inequality and use Markov's inequality to prove Chebyshev's inequality.

One application of Chebyshev's inequality is the *weak law of large numbers*. Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed random variables with mean μ and variance σ^2. Let $S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then

$$\begin{aligned} E[S_n] &= E[\frac{1}{n}\sum_{i=1}^{n} X_i], \\ &= \frac{1}{n}\sum_{i=1}^{n} E[X_i], \\ &= \mu, \end{aligned}$$

and

$$\begin{aligned} Var[S_n] &= Var[\frac{1}{n}\sum_{i=1}^{n} X_i], \\ &= \frac{1}{n^2}\sum_{i=1}^{n} Var[X_i], \\ &= \frac{\sigma^2}{n}. \end{aligned}$$

From Chebyshev's inequality, we then have that

$$P[|S_n - \mu| > k\sigma] \leq \frac{1}{nk^2}.$$

We see that the right hand side goes to zero as $n \to \infty$, and so a sample average converges to the mean as the number of samples increases, in a probabilistic sense. In the general case, one cannot improve on this bound. In the case when the X_i are bounded, there are many more precise tighter bounds which can be obtained. These types of bounds are typically called tail inequalities, for example the Bernstein inequalities (special cases of which are the Hoeffding inequality, the Azuma inequality, ...). The version presented here does not make use of the variance. Bennett's inequality can be used to give a tighter bound if the variance or a good bound on the variance is known (in addition to the random variable being bounded).

Hoeffdings Inequality Let $X_1, \ldots, X_n$ be independent random variables which are bounded, so $X_i \in [a_i, b_i]$, then

$$P[|S_n - \mu| > t] \leq 2 \exp\left(-\frac{2n^2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

For the special case when all the X_i are bounded by the same bounds, $X_i \in [a, b]$, letting $B = b - a$, we have that

$$P[|S_n - \mu| > t] \leq 2 \exp\left(-\frac{2nt^2}{B^2}\right).$$

For any fixed t, the right hand side exponentially decays to zero with respect to n. In fact for $t = \omega(\frac{1}{\sqrt{n}})$, the right hand side tends to 0 with $n \to \infty$ which indicates that the deviation of S_n from μ is of the order of $1/\sqrt{n}$ with high probability.

2 Derivative Pricing, Expectations and Integrals

In the previous discussions, we have seen that the price of a derivative f, for simplicity a state dependent derivative, is given by the expectation of discounted future cash flows in the *risk neutral world*. In particular, the risk neutral world dynamics typically specifies how to generate a stock price path according to the risk neutral probability,

$$S(t + \Delta t) = M(S(t), \Delta t, \boldsymbol{\eta}),$$

where $\boldsymbol{\eta}$ is a random variable and $M(\cdot)$ is some function. For example, for the geometric Brownian motion risk neutral world,

$$M(S(t), \Delta t, \eta) = S(t) e^{(r - \frac{1}{2}\sigma_R^2)\Delta t + \sigma_R \sqrt{\Delta t}\eta},$$

where $\eta \sim N(0, 1)$. The price of the derivative is given by

$$\text{Price}(f) = E_{\text{paths } p}[PV(\text{cash flows along path } p)].$$

A path p is specified by the stock price values along the path, in particular, $S_0, S_1, \ldots, S_n$ where $S_i = S(i\Delta t)$, for $i = 0, 1, 2, \ldots$. The risk neutral dynimacs gives a probabilistic prescritpion for generating these paths, which implicitly defines the joint probability distribution $P(S_1, S_2, \ldots, S_n)$ of the stock prices along the path. The cash flow at time $i\Delta t$ when the stock price is S_i is specified by $f(S_i, i\Delta t)$, and so its present value is $e^{-ir\Delta t} f(S_i, i\Delta t)$. Thus the present value of all cash flows along the path $p = S_0, S_1, \ldots, S_n$ is

$$C(S_1, \ldots, S_n) = \sum_{i=1}^{n} e^{-ir\Delta t} f(S_i, i\Delta t).$$

We thus have that

$$\text{Price}(f) = \int dS_1 dS_2 \cdots dS_n \; P(S_1, S_2, \ldots, S_n) C(S_1, \ldots, S_n),$$

where $C(S_1, \ldots, S_n)$ is the discounted cash flow function for the path $S_1, \ldots, S_n$ and $P(S_1, S_2, \ldots, S_n)$ is the *risk neutral* probability of the path $S_1, \ldots, S_n$. The main thing to notice here is that the price of the derivative is the integral of some function in some very high dimensional space n, where typical values of n may be of the order of $10,000$ (for example using an hour time step to simulate a 1 year (250 days) option already has $n > 5,000$, and a 1 minute time step is more desirable).

The point of this discussion was to show that computing the price of an option is essentialy an integral. We now abstract ourselves away from the pricing framework, and consider Monte Carlo for computing integrals.

3 Monte Carlo For Evaluating Integrals

We consider a d-dimensional integral in one of two forms,

$$I = \int_{[0,1]^d} d\mathbf{x} \; p(\mathbf{x}) f(\mathbf{x}), \qquad I = \int_{[0,1]^d} d\mathbf{x} \; g(\mathbf{x}),$$

where $p(\mathbf{x})$ is a probability density on $[0,1]^d$ which is the domain of the integration. Notice that through a change of variables, we can without much loss of generality assume that the domain of integration is $[0,1]^d$, the d-dimensional hypercube. Clearly these two forms of integral are equivalent. Indeed, the first is just a special case of the second for the choice $g(\mathbf{x}) = p(\mathbf{x}) f(\mathbf{x})$, and the second is a special case of the first for the choice $p(\mathbf{x}) = 1$, the uniform distribution on the unit hypercube. Without loss of generality, we can assume that $f(\mathbf{0}) = g(\mathbf{0}) = 0$.

Typically one makes some assumption on the function being integrated in order to ensure that any numerical technique for computing the integral has some good convergence properties. A reasonable assumption is that the function is bounded, so for example

$$\max_{\mathbf{x} \in [0,1]^d} |g(\mathbf{x})| \leq B.$$

A slightly stronger assumtion is that $g(\mathbf{x})$ has a gradient with bounded norm, so that

$$\max_{\mathbf{x}\in[0,1]^d} \| \nabla g(\mathbf{x}) \| \leq B.$$

This is a stronger assumption, for it immediately implies that the function itself is bounded by using Taylors theorem:

$$g(\mathbf{x}) = g(\mathbf{0}) + \nabla g(\mathbf{x}^*) \cdot \mathbf{x},$$

for some $\mathbf{x}^* = \lambda \mathbf{x}$ for $0 \leq \lambda \leq 1$. Therefore

$$\begin{aligned} |g(\mathbf{x})| &= \| \nabla g(\mathbf{x}^*) \cdot \mathbf{x} \|, \\ &\leq \| \nabla g(\mathbf{x}^*) \| \| \mathbf{x} \|, \\ &\leq B\sqrt{d}, \end{aligned}$$

because $\| \mathbf{x}^* \| \leq \sqrt{d}$ for $x \in [0,1]^d$. In fact, let $\mathbf{x}, \mathbf{x}'$ be any two points in $[0,1]^d$, then $|g(\mathbf{x}) - g(\mathbf{x}')| \leq B\| \mathbf{x} - \mathbf{x}' \|$.

3.1 Brute Force Numerical Integration

First we consider a simple brute force numerical approach to integration which serves as a benchmark to illustrate why multidimensional integration is a very hard problem. In particular, we consider the standard multi-dimensional generalization of the Riemann sum approximation to an integral. In particular, partition each dimension of the unit hypercube into intervals of length ϵ. This partitions the unit hypercube into ϵ-hypercubes each of volume ϵ^d. The distance between any two points in the same ϵ-hypercube is at most $\epsilon\sqrt{d}$. The number of such hypercubes is

$$n_\epsilon = \frac{1}{\epsilon^d}.$$

Let the hypercubes be numbered $V_1, V_2, \ldots, V_{n_\epsilon}$, and let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_\epsilon}$ each be a point in the corresponding ϵ-hypercube, so $\mathbf{x}_i \in V_i$. The Riemann sum approximation to the integral $I = \int_{[0,1]^d} d\mathbf{x}\ g(\mathbf{x})$ is

$$\hat{I} = \epsilon^d \sum_{i=1}^{n_\epsilon} g(\mathbf{x}_i).$$

One of the measures of a performance of a numerical integration routine is the number of times it has to evaluate the function (the sample complexity). This algorithm clearly has a sample complexity of $n_\epsilon = \frac{1}{\epsilon^d}$. Another measure of performance is the error between the

estimate and the true value of the integral, $\mathcal{E} = |I - \hat{I}|$. Indeed, we have that

$$\begin{aligned} I &= \int_{[0,1]^d} d\mathbf{x}\ g(\mathbf{x}), \\ &= \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ g(\mathbf{x}), \\ &= \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ (g(\mathbf{x}) - g(\mathbf{x}_i) + g(\mathbf{x}_i)), \\ &= \hat{I} + \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ (g(\mathbf{x}) - g(\mathbf{x}_i)). \end{aligned}$$

So we see that

$$\begin{aligned} \mathcal{E} &= \left| \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ (g(\mathbf{x}) - g(\mathbf{x}_i)) \right|, \\ &\leq \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ \left| g(\mathbf{x}) - g(\mathbf{x}_i) \right|, \\ &\leq B \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ \left| \mathbf{x} - \mathbf{x}_i \right|, \\ &\leq B\sqrt{d}\epsilon \sum_{i=1}^{n_\epsilon} \int_{V_i} d\mathbf{x}\ 1, \\ &= B\sqrt{d}\epsilon \sum_{i=1}^{n_\epsilon} \epsilon^d, \\ &= B\sqrt{d}\epsilon. \end{aligned}$$

Thus, the error in the estimate is decreasing linearly in the side length ϵ which appears to be a good thing because it means that by taking ϵ sufficiently small, which ammounts to making n_ϵ sufficiently large, we can obtain an arbitrarily accurate integral. However, there is a subtle problem hidden here, which appears when we view compute the sample complexity required to compute a desired error $\mathcal{E}$. Suppose that we have a target error e for the integral estimate which we would like to guarantee. If $B\sqrt{d}\epsilon \leq e$, then certainly $\mathcal{E} \leq e$. Since $n_\epsilon = \frac{1}{\epsilon^d}$, we find that to guarantee an error e it suffices that

$$n_\epsilon = \left(\frac{B\sqrt{d}}{\mathcal{E}} \right)^d.$$

Notice that the sample complexity based on this bound is growing exponentially in d. The severity of this problem is illustrated in the following exercise.

Exercise 3.1

Suppose that we are doing a 100 dimensional integral, $d = 100$, and that $B = 1$ and that we wish to guarantee an error of 0.01. What is the sample complexity implied by the bound.

A sample complexity which is exponential in d is not practically feasible for large d and as we already discussed, in the derivative pricing setting, we are interested in very high dimensional integrals. Unfortunately, this is the best that one can do in order to *deterministically* guarantee the error, and for this reason it is called the *curse of dimensionality*. However, if we only require of an algorithm that most of the time it be very accurate, then we can actually escape this curse of dimensionality, which is where Monte Carlo comes in.

3.2 Monte Carlo for Integral Computation

We will consider the form of the integral $I = \int_{[0,1]^d} d\mathbf{x}\ p(\mathbf{x}) f(\mathbf{x})$ because it will turn out that $p(\mathbf{x})$ can actually play a role in making the extimate of the integral more accurate. The prescription for computing the integral $\hat{I}$ is given in the following very simple algorithm. The input to the algorithm is n, the number of Monte Carlo samples to use. In describing the algorithm, we assume that an oracle exists to compute $f(\mathbf{x})$ in constant time, and an oracle exists to generate *iid* samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with each $\mathbf{x}_i \sim p(\mathbf{x})$

1: Assume that n is given.
2: Generate $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ *iid* according to $p(\mathbf{x})$.
3: **return** $\hat{I} = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)$.

In a nutshell,

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i),$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are *iid* random variates generated according to $p(\mathbf{x})$.

The sample complexity of this algorithm is n. There is another complexity in this algorithm which is the requirement to generate n random variates *iid* according to $p(\mathbf{x})$. This is a non-trivial task, even for $p(\mathbf{x})$ being the uniform distribution. For the uniform distribution, one has to generate nd uniform random variates. Luckily, random variate generation is a well studied topic in its own right and so there are many good pseudo-random number generators for a variety of distributions. These random number generators, together with variable transformations is typically enough to generate random variates for our purposes.

Applying The Monte Carlo Algorithm to Derivative Pricing. Remember that in the derivative pricing setting, we have

$$\text{Price}(f) = \int dS_1 dS_2 \cdots dS_n\ P(S_1, S_2, \ldots, S_n) C(S_1, \ldots, S_n),$$

where $S_1, S_2, \ldots, S_n$ is a sample path. We assume that the cashflow function for a sample path, $C(S_1, \ldots, S_n)$ can be computed in typically in $O(n)$ time. How do we generate a sample $\mathbf{x} = (S_1, S_2, \ldots, S_n)$ from the joint distribution $P(S_1, S_2, \ldots, S_n)$. Well, this is just a sample path, and we assume that we know the risk neutral dynamics, so, for example, for the geometric Brownian motion,

$$S((i+1)\Delta_t) = S(i\Delta t)e^{\eta_i},$$

where $\eta_i \sim N((r - \frac{1}{2}\sigma_R^2)\Delta T, \sigma_R^2 \Delta t)$. Starting from S_0 we can generate a sample path providing we can generate *iid* random variates η_i from a normal distribution, we can then generate a sample path. Generating a sample path is exactly equivalent to generating $\mathbf{x} = (S_1, S_2, \ldots, S_n)$ from the joint distribution $p(\mathbf{x}) = P(S_1, S_2, \ldots, S_n)$. We give the full algorithm below.

1: Let N be the number of Monte Carlo samples and assume that $n, \Delta t$ are given (the path discretization parameters). Suppose that S_0 the initial stock price is given.
2: **for** $i = 1$ to N **do**
3: Generate a sample path $\mathbf{x}_i = (S_1, S_2, \ldots, S_n)$ as follows,
4: **for** $j = 1$ to n **do**
5: generate an independent normal random variate $\eta \sim N(0, 1)$.
6: set $S_j = S_{j-1} e^{(r - \frac{1}{2}\sigma_R^2)\Delta T + \sigma_R \sqrt{\Delta t}\eta}$.
7: **end for**
8: Compute the cash flow along the sample path $\mathbf{x}_i$, $f_i = C(S_1, \ldots, S_n)$.
9: **end for**
10: **return** $Price = \frac{1}{N}\sum_{i=1}^{N} f_i$.

This prescription requires us to be able to generate standard normal random variates, and we will see how to do this later. In a nutshell,

$$Price(f) = \frac{1}{n} \sum_{iid \text{ paths } (S_1, \ldots, S_n)} C(S_1, \ldots, S_n),$$

where $C(S_1, \ldots, S_n)$ is the discounted cashflow function along a particlar path.

3.3 Analysis of the Monte Carlo Algorithm

Remember that

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i).$$

The first we will show is that $\hat{I}$ is an unbiased estimator of I.Indeed,

$$\begin{aligned} E[\hat{I}] &= E\left[\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)\right], \\ &= \frac{1}{n}\sum_{i=1}^{n} E[f(\mathbf{x}_i)], \\ &= E[f(\mathbf{x})], \\ &= \int_{[0,1]^d} d\mathbf{x}\; p(\mathbf{x}) f(\mathbf{x}), \\ &= I. \end{aligned}$$

We have used the fact that the $\mathbf{x}_i$ are *iid* which implies that the $f(\mathbf{x}_i)$ are also *iid* random variates, and so $E[f(\mathbf{x}_i)] = E[f(\mathbf{x})]$ for all i. Unlike with the brute force numerical integration, for Monte Carlo, $\hat{I}$ is a random quantity, and so no guarantees can be made about it. However, $\sigma = \sqrt{Var[\hat{I}]}$ gives an estimate of the typical deviation between $\hat{I}$ and $E[\hat{I}] = I$, which is therefore a measure of the typical error that one expects in estimating I by $\hat{I}$. Indeed,

$$\begin{aligned} Var[\hat{I}] &= Var\left[\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i)\right], \\ &= \frac{1}{n^2}\sum_{i=1}^{n} Var[f(\mathbf{x}_i)], \\ &= \frac{1}{n} Var[f(\mathbf{x})]. \end{aligned}$$

If we now assume that f is bounded on $[0,1]^d$ so $f(\mathbf{x}) \in [a,b]$ with $B = b - a$ (a weaker assumption than bounded derivative), then $Var[f(\mathbf{x})] \leq B^2$, and so

$$\sigma(f) \leq \frac{B}{\sqrt{n}}.$$

Thus, the "probabilistic error bar" in the estimate is $O(n^{-1/2})$, *independent of the dimension* d. This is quite a huge breakthrough. By giving up the deterministic guarantee on the error, we can avoid the curse of dimensionality. A more precise statement of this fact is obtained by using the Hoeffding inequality. Using the Hoeffding bound,

$$P[|\hat{I} - I| > \tfrac{A}{\sqrt{n}}] \leq e^{-\frac{2A^2}{B^2}}.$$

The right hand side probability bound can be made arbitrarily small by selecting A large enough. Fixing A to some such large value, we then have that with probability at least $1 - e^{-\frac{2A^2}{B^2}}$, $|\hat{I} - I| \leq \frac{A}{\sqrt{n}}$, or that with very high probability the error in the integral estimate is $|\hat{I} - I| = O(\frac{1}{\sqrt{n}})$.

3.4 A Hybrid Monte Carlo Algorithm

We can slightly improve the efficiency of the Monte Carlo by combining it with the numerical brute force approach as follows. The basic idea is that if we are going to use $n = k^d$ monte-carlo samples, then instead of randomly distributing each of them in the hypercube, we construct the ϵ-hypercubes as with the numerical brute force approach, with $\epsilon = \frac{1}{k}$ and distribute *one* point randomly in each of the n ϵ-hypercubes.

We consider the integral

$$I = \int_{[0,1]^d} d\mathbf{x}\ f(\mathbf{x}),$$

which corresponds to $p(\mathbf{x})$ being the uniform distribution. We will give the general algorithm for general $p(\mathbf{x})$, but as we will see, the most gain occurs for the uniform distribution. Let V_i be the i^{th} ϵ-hypercube, and let p_i be the probability of a point being generated in V_i,

$$\begin{aligned} p_i &= \int_{V_i} d\mathbf{x}\ p(\mathbf{x}), \\ &= \epsilon^d. \end{aligned}$$

We now generate the point $\mathbf{x}_i$ in V_i according to the density $\frac{1}{p_i}p(\mathbf{x})$. The estimate $\hat{I}$ is given by

$$\begin{aligned} \hat{I} &= \sum_{i=1}^{n} p_i f(\mathbf{x}_i), \\ &= \epsilon^d \sum_{i=1}^{n} f(\mathbf{x}_i), \end{aligned}$$

We compute the properties of this estimate as before, assuming that f has a bounded derivative, $\| \nabla f \| \leq B$.

$$\begin{aligned} E[\hat{I}] &= E\left[\sum_{i=1}^{n} p_i f(\mathbf{x}_i)\right], \\ &= \sum_{i=1}^{n} p_i E[f(\mathbf{x}_i)], \\ &= \sum_{i=1}^{n} p_i \int_{V_i} d\mathbf{x}\ \frac{p(\mathbf{x})}{p_i} f(\mathbf{x}), \\ &= \sum_{i=1}^{n} \int_{V_i} d\mathbf{x}\ p(\mathbf{x}) f(\mathbf{x}), \\ &= \int_{[0,1]^d} d\mathbf{x}\ p(\mathbf{x}) f(\mathbf{x}), \\ &= I. \end{aligned}$$

$$\begin{aligned} Var[\hat{I}] &= Var\left[\sum_{i=1}^{n} p_i f(\mathbf{x}_i)\right], \\ &= \sum_{i=1}^{n} p_i^2 Var\left[f(\mathbf{x}_i)\right], \\ &\leq \sum_{i=1}^{n} p_i^2 B^2 d\epsilon^2, \\ &= B^2 d\epsilon^2 \sum_{i=1}^{n} p_i^2, \end{aligned}$$

where we have used that $f(\mathbf{x}_i)$ is bound in a range of $B\sqrt{d}\epsilon$ as $\mathbf{x}_i$ varies over V_i. The summation $\sum_{i=1}^{n} p_i^2 \geq \frac{1}{n}$, with equality attained for the uniform distribution, which is why we consider the uniform distribution. For the uniform distribution, we have

$$Var[\hat{I}] \leq \frac{B^2\sqrt{d}}{n^{1+\frac{2}{d}}},$$

which in terms of the dependence on n is an improvement by a factor of $\frac{1}{n^{2/d}}$ over the standard Monte Carlo approach. For small d, this can be a significant gain. For example, for $d = 1$ and using σ as the error measure, we have $\sigma(\hat{I}) = O(n^{-3/2})$, as opposed to $O(n^{-1/2})$. One important requirement to run the hybrid algorithm effectively is to be able to generate a uniform point $\mathbf{x}_i$ in V_i. This is accomplished by the following algorithm,

1: For integer k, let $n = k^d$, and $\epsilon = \frac{1}{k}$.
2: **for** $i = 0$ to $n - 1$ **do**
3: Let $a_1 a_2 \ldots a_d$ be the base k representation of i, so $i = a_1 + a_2 k + a_3 k^2 + \cdots + a_d k^{d-1}$, where $a_i \in \{0, 1, \ldots, k-1\}$. Let $\mathbf{k}$ be the vector containing $a_1, \ldots, a_d$
4: Generate d uniform random variables, $u_1, \ldots, u_d \sim U[0, 1]$. Let $\mathbf{u}$ be the vector containing $u_1, \ldots, u_d$.
5: Set $\mathbf{x}_i = \epsilon \cdot (\mathbf{k} + \mathbf{u})$.
6: **end for**
7: **return** $\hat{I} = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)$.

In the above algorithm, it is necessary to compute the d-ary representation of a number. We used this approach because it is convenient to represent. What is actually going on is that we have d nested loops each from 0 to $k-1$. However such an algorithm is hard to encode with d as an input. The k-ary representation is easy to compute efficiently in $O(d)$ time. The basis of the computation is through the notions of the remainder and the quotient. Specifically, we can write any integer $x = qk + r$, where q is the quotient and r the remainder. Then,

$$q(x; k) = \left\lfloor \frac{x}{k} \right\rfloor, \qquad r(x; k) = x - q(x)k = x - \left\lfloor \frac{x}{k} \right\rfloor k.$$

Then, the following recursion constructs the k-ary representation of an integer x:

$$a_1(x) = r(x;k),$$
$$q_1(x) = q(x;k),$$

and for $i > 1$,

$$\begin{aligned} a_i(x) &= r(q_{i-1}(x);k), \\ q_i(x) &= q(q_{i-1}(x);k). \end{aligned}$$

$a_1, \ldots, a_n$ are the digits of x in base k.

Exercise 3.2

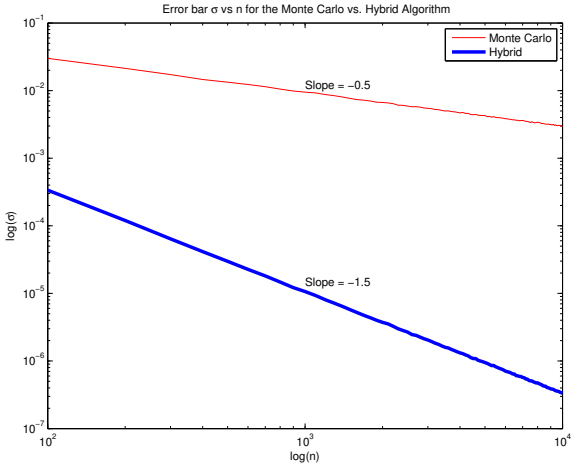
Implement efficiently the standard Monte Carlo algorithm, and the hybrid algorithm for $p(\mathbf{x})$ uniform, and test it for $d = 1$ using the following example.

(a) Let $I = \int_0^1 dx\ x^2$, so $g(x) = x^2$. Set $n = 100$ and compute an estimate $\hat{I}_1$ using the standard Monte Carlo $m = 5000$ times. What is the sample standard deviation in this estimate? This sample standard deviation is a good estimate of $\sigma(\hat{I}_1)$.

Now repeat this same same experiment to compute the Hybrid monte Carlo estimate $\hat{I}_2$ m times the Hybrid Monte Carlo and compare the sample standard deviations $\sigma(\hat{I}_1)$ and $\sigma(\hat{I}_2)$.

(b) Now repeat the previous experiment for n in the range $[100, 10,000]$ and give a plot on a log-log scale of $\sigma(\hat{I}_1(n))$ and $\sigma(\hat{I}_2(n))$ as n increases. Compute the slopes of the resulting curves and explain.

[Answer:



]

4 Random Variate Generation

A very hard problem brushed under the rug in the Monte Carlo algorithm is the generation of the random $\mathbf{x}_i$. This is in its own right an entire area of research in mathematics, and we do not propose to give anywhere near a complete characterization of this topic. Our goal is to give some basic tools which would prepare the reader for performing Monte Carlo, and if necessary further research through other literature.

The simplest of these problems is to generate a sequence $u_1, u_2, \ldots$ of *iid* uniform random variates, where each $u_i \sim U[0,1]$ is independent of every other one. Naturally there is no known algorithm (except possibly for harnessing "trully random" quantum physical processes) to generate a purely random sequence $u_1, u_2, \ldots$, so all the research has focussed on generating *pseudo-random* sequences, which are deterministic sequences that "look" random. The simplest of these is the linear congruential generator (LCG), which is typically the default random number generator in most math libraries such as rand() in C++. Linear congruential generators give a sequence of pseudo-random integers starting from a *seed* z_0, and following the recursion

$$z_{i+1} = az_i + b(mod\ m).$$

a, b, m are parameters and much work goes into determining good values for these parameters. For example, the standard C++ generator uses $m = 2^{32}$, $a = 1103515245$ and $b = 12345$. Note that any such generators are periodic, with a period of at most m.

Other non-linear aproaches to univariate generation also exist and tend to be slow in general, though perhaps slightly superior to the LCG. One such method which I recommend is the *Mersenne Twister* random number generator, which is available under the FreeBSD liscence off the www, for example at

http://www-personal.umich.edu/ wagnerr/MersenneTwister.html

The algorithm is very fast, and has superior properties to rand(), including the fact that its period is $2^{19937} - 1$

For our purposes, we will assume that a stream of uniform random variates $u_1, u_2, \ldots$ is available. We study how to generate random variates from other distributions, given such a stream of uniform random variates. Suppose we wish to generate $X \sim f_X(x)$ with cdf $F_X(x)$.

4.1 The Transformation Method

Let $X = F_X^{-1}(u)$ where $u \sim U[0,1]$. We show that $X \sim f_X$. Indeed,

$$\begin{aligned} P[X \leq x] &= P[F_X^{-1}(u) \leq x], \\ &= P[u \leq F_X(x)], \\ &= F_X(x). \end{aligned}$$

We have used the fact that F_X is monotonic, and that and that the cdf of the uniform distribution is given by $F_U(u) = u$ for $u \in [0,1]$.

The transformation method is generally efficient if F^{-1} is known in analytical form. In general, F_X itself is hard to compute (for example for the Normal distribution), so F_X^{-1} is not easy to compute. Thus the method is not general, however in the cases where it can be carried out, it is efficient.

Exercise 4.1

Using the transformation method, show how to generate random variates from the following distributions.

(a) The one sided exponential, $f_X(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$

(b) The two sided exponential, $f_X(x) = \frac{\lambda}{2} e^{-\lambda |x|}$.

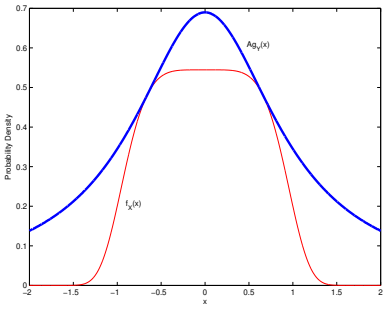
(c) The monomial distribution on $[0,1]$, $f_X(x) = (p+1)x^p$ for $x \in [0,1]$, and $p > -1$.

(d) The monomial distribution on $[0,1]$, $f_X(x) = (p+1)(1-x)^p$ for $x \in [0,1]$, and $p > -1$.

(e) The cauchy distribution, $f_X(x) = \frac{1}{\pi} \frac{1}{1+x^2}$.

4.2 The Acceptance-Rejection Method

In order to implement this technique it is necessary to have a second distribution g_Y from which one can generate random variables, and it is necessary that for any x such that $f_X(x) > 0$, $g_Y(x) > 0$. We let $A = \sup_{x: f_X(x) > 0} \frac{g_Y(x)}{f_X(x)}$. Then note that $A \geq 1$ and $\forall x$, $f_X(x) \leq A g_Y(x)$. Any distribution g_Y which is non-zero everywhere would do. A particularly useful such distribution is the Cauchy distribution $g_Y(x) = \frac{1}{\pi} \frac{1}{1+x^2}$ on the real line, for which we can generate random variates using the transformation method. On the unit interval, the polynomial distributions are useful. The situation is illustrated in the figure below,



The algorithm to generate a random variate according to f_X is then given in the following algorithm.

1: initialize a variable success=0.
2: **while** success=0 **do**
3: Generate a random variate x from g_Y, $x \sim g_Y$.
4: Generate a uniform random variate $u \sim U[0,1]$.
5: **if** $x \leq \frac{f_X(x)}{Ag_Y(x)}$ **then**
6: success=1
7: **end if**
8: **end while**
9: **return** x.

In a nut shell, the algorithm generates a random variate $x \sim g_Y$ and accepts it with probabilty $f_X(x)/Ag_Y(x)$.

Exercise 4.2

Show that the probability that a random variate is accepted is $\frac{1}{A}$.

Exercise 4.3

Show that $P[x \leq h | x \text{ is accepted}] = F_X(h)$ where F_X is the distribution function for f_X.

This exercise shows that the random variates generated have the right distribution.

The algorithm has been described in the 1-dimensional setting, but it is completely analogous in the multi-dimensional setting. Thus the algorithm is general, however it is not very

efficient, having an efficiency $\frac{1}{A}$ – for every A random variates generated according to g_Y, on average one will be accepted.

Exercise 4.4

Let $q(x)$ be any non-negative valued function not identically zero. Suppose you wanted to generate a random variate from the density $f(x) = kq(x)$.

(a) What is the value of k.

Suppose you have a bounding density $Ag(x) \geq q(x)$, which is a bounding density for $q(x)$. Note that $q(x)$ is not itself a density, however it has the right *shape* as $f(x)$. Suppose we use the following rejection sampling algorithm.

1: initialize a variable success=0.
2: **while** success=0 **do**
3: Generate a random variate x from $g(x)$, $x \sim g$.
4: Generate a uniform random variate $u \sim U[0,1]$.
5: **if** $x \leq \frac{q(x)}{Ag(x)}$ **then**
6: success=1
7: **end if**
8: **end while**
9: **return** x.

(b) Show that the acceptance rate of the above algorithm is $\frac{1}{kA}$.

(b) Show that the accepted samples have the density $f(x)$.

This exercise shows that as long as the shape of $f(x)$ is known and one has a bounding density for the shape function $q(x)$, we can generate from the density $f(x)$ without having to normalize the function $q(x)$ - i.e. without having to obtain k. We will be able to generate from $f(x)$, however we will not be able to compute the efficiency without k.

Exercise 4.5

Suppose that we wish to generate a random variable from the distribution distribution $f(x) \propto e^{-x^4}$. What will be the efficiency (acceptance rate) if we used as the bounding density

(a) The Normal density $g(x) = \frac{1}{2\pi} e^{-\frac{1}{2}x^2}$.

(b) The Exponential density $g(x) = \frac{1}{2} e^{-|x|}$.

(c) The Cauchy density $g(x) = \frac{1}{\pi}\frac{1}{1+x^2}$.

Exercise 4.6

[Restriction Sampling and Rejection Sampling] Sometimes one has a density $f(x)$ defined (say) on the real line and one wishes to generate the random variable restricted to some region I for example $I = [0, 1]$. For this exercise we will consider $f(x) = e^{-x}$. One natural way to generate such a restricted x is to generate $x \sim f(x)$ (assuming we can generate from $f(x)$) and then accept if $x \in I$ and reject otherwise. The efficiency of this process is $P[x \in I] = \int_I dx\ f(x)$. We call this restriction sampling.

1. Show that restriction sampling is equivalent to sampling from the density
$$f_r(x) = \begin{cases} kf(x) & x \in I, \\ 0 & x \notin I. \end{cases}$$
What is k?
2. Show that restriction sampling is equivalent to rejection sampling where the bounding density $g(x)$ is $f(x)$. What is A?
3. It is often possible to obtain a better efficiency for restriction sampling using a different bounding density than $f(x)$ itself. For our example of the restricted exponential density, consider the following two bounding densities on $[0, 1]$ (which are easy to generate from) and compare the efficiency of rejection sampling using these densities with the original restriction sampling.
   i. The uniform density, $g(x) = 1$
   ii. The linear density, $g(x) = 2(1 - x)$.

4.3 Normal Random Variates

From the financial perspective, the multi-dimensional Normal random variate is one of the most important. The multi-dimensional normal density $N(\boldsymbol{\mu}, \Sigma)$ is the following pdf,

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})},$$

where $\boldsymbol{\mu}$ is the mean of the density, Σ is the covariance matrix and $|\Sigma|$ is the determinant of Σ, and d is the dimensionality. The standard Normal distribution is obtained by setting $\boldsymbol{\mu} = \mathbf{0}$ and $\Sigma = I_{d\times d}$, where $I_{d\times d}$ is the $d \times d$ identity matrix. The standard Normal density has the functional form

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\|\mathbf{x}\|^2}.$$

Note that the standard normal density is spherically symmetric. Also note that uncorrelated normal random variates are independent as the density factors into a product of densities. We will use a theorem from probability theory which states that a linear combination of normal random variables is itself normal. Suppose that $\mathbf{x} \sim N(\mathbf{0}, I)$ is a standard Normal random variable and let $\mathbf{z} = A\mathbf{x} + \boldsymbol{\mu}$ be a linear transform of $\mathbf{x}$. Then

$$E[\mathbf{z}] \quad = \quad AE[\mathbf{x}] + \boldsymbol{\mu} = \boldsymbol{\mu}.$$

Exercise 4.7

Show that

$$E[\mathbf{z}\mathbf{z}^{\mathsf{T}}] = AA^{\mathsf{T}} + \boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}.$$

The previous exercise shows that the covariance matrix for $\mathbf{z}$ is given by

$$\Sigma_{\mathbf{z}} = AA^{\mathrm{T}}.$$

Given a symmetric positive definite Σ, we can construct the unitary decomposition of Σ,

$$\Sigma = UDU^{\mathrm{T}}$$

where the matrix U are the eigenvectors of Σ and D is the diagonal matrix of (positive) eigenvalues. The matrix U is orthogonal, i.e. $UU^{\mathrm{T}} = U^{\mathrm{T}}U = I_{d\times d}$. The matrices U and D are easily available in most mathematical packages, for example in matlab they may be obtained through the function eig. Setting $A = \Sigma^{1/2} = UD^{1/2}U^{\mathrm{T}}$, one can verify that

$$\Sigma_{\mathbf{z}} = \Sigma.$$

What this means is that given $\boldsymbol{\mu}, \Sigma$, we can define $A = \Sigma^{1/2}$ and $\boldsymbol{b} = \boldsymbol{\mu}$, and then $\mathbf{z} = A\mathbf{x} + \mathbf{b} \sim N(\boldsymbol{\mu}, \Sigma)$ if $\mathbf{x}$ is a standard Normal. To see this, we know from probability theory that the linear combination of a set of Normal random variables is Normal, so it suffices to verify that the mean and covariance matrix are correct. But this is what the entire previous discussion established. Thus it suffices to be able to generate from a standard Normal density.

Exercise 4.8

Show that $\Sigma_{\mathbf{z}} = \Sigma$.

We thus consider generating random variables from the standard Normal density. There are many very accurate approximations to the standard Normal cumulative distribution function,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} ds\ e^{-\frac{1}{2}s^2}.$$

There are also many approximations in terms of rational functions and Chebyshev approximations to ϕ^{-1}, and one can use these to generate a standard Normal random variate to good accuracy using the transformation method.

Perhaps the most famous method for generating normal random variates is the Box-Muller method which actually does not generate a single standard Normal, but it generates *two* independent standard Normal random variates from two uniform variates. Let u_1, u_2 be two uniform random variates, and define the transformed random variates x_1, x_2 by

$$\begin{aligned} x_1 &= \sqrt{-2\ln u_1}\cos 2\pi u_2, \\ x_2 &= \sqrt{-2\ln u_1}\sin 2\pi u_2. \end{aligned}$$

Then x_1, x_2 are independent standard Normal random variates.

Exercise 4.9

Show that x_1, x_2 are independent standard Normals.

[Hint: It is easier to start with two jointly independent Normal random variates x_1, x_2 with $f(\mathbf{x}) = \frac{1}{2\pi}e^{-\frac{1}{2}(x_1^2+x_2^2)}$ and show that $u_1 = \frac{1}{2\pi}arctan\frac{x_2}{x_1}$ and $u_2 = e^{-\frac{1}{2}(x_1^2+x_2^2)}$ are uniformly distributed. Polar coordinates may be useful and the following observation. Since (x_1, x_2) has a spherically symmetric distribution, the angle made by the vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and the x_1-axis and the radius of the vector $x_1^2+x_2^2$ must be independent. Further the angle must be uniformly distributed in $[0, 2\pi]$. The angle θ is exactly $arctan\frac{x_2}{x_1}$. Thus you may observe that u_1 is entirely a function of the angle and u_2 is a function entirely of the radius which are independent. Now show that u_1, u_2 have uniform distributions. Since the transformation is invertible, it follows that if u_1, u_2 have independent uniform distributions, then x_1, x_2 have independent standard Normal distributions.]

Thus the transform method can be used to generate a pair of independent standard Normal random variates. We may now generate the d-dimensional standard Normal density by generating $\lceil \frac{d}{2} \rceil$ pairs of independent Normal variates and selecting any d of these.

5 Variance Reduction in Monte Carlo

A good measure of the error of an estimate is $\sigma(\hat{I}) = \sqrt{Var[\hat{I}]}$. A good bit of work in Monte Carlo deals with effective techniques to reduce the variance. Of course, as we have already seen, $\sigma(\hat{I}) = O(\frac{1}{\sqrt{n}})$, where n is the number of samples, so one way to reduce the variance is to increase n. The question is, what is the best estimate that one can obtain given

i. A fixed budget of computation (fixed n).

ii. A fixed budget of uniform random variates.

5.1 Anti-Thetic Variates

Suppose that we wish to compute the integral $I = \int_0^1 dx\ f(x)$. The goal of variance reduction is to try to compute as accurate an estimate $\hat{I}$ with as few samples as possible.

Lets first consider the following simple case. Suppose that $f(x)$ is a linear function. In this case I can be computed with just *one* sample, as the following exercise demonstrates.

Exercise 5.1

Show that if $f(x)$ is linear, then $I = f(\frac{1}{2})$.

The same is true of an arbitrary continuous function $f(x)$, namely that $I = f(x^*)$ for some $x^* \in [0, 1]$. This is in fact the content of the mean value theorem. Thus one sample x^* suffices to compute I for any function $f(x)$. The problem is that we do not know x^*. For a linear f, we know that $x^* = \frac{1}{2}$. A generalization of this to higher order polynomials is essentially the theory behind integration by quadrature.

Sticking with the linear function we now consider an approach which has more chance to generalize. In particular, lets move up to two samples. What can we do with two samples? It turns out that we can get an exact estimate for I. In particular, let u be the first sample point, and let $1 - u$ be the second sample point, then for a linear function $f(x)$,

$$I = \frac{f(u) + f(1-u)}{2},$$

for *any* $u \in [0, 1]$.

Exercise 5.2

Show that $I = \frac{f(u)+f(1-u)}{2}$.

The nice thing about the previous little result is that we do not need to know exactly where to sample. It only matters that we use the sample points u and $1-u$ for any u. This is something that can generalize to an arbitrary function, and we know it is exact for the linear function. The function value $f(u)$ may mis-estimate the integral I, but then $f(1-u)$ mis-estimates in exactly the opposite direction, exactly compensating for the mis-estimation of $f(u)$.

If u were a uniform random variate, then $1-u$ is called the *anti-thetic* random variate because it has two very interesting properties. The first is that $1-u$ has exactly the same distribution as u, namely uniform on $[0,1]$. The second is that $1-u$ is perfectly anti-correlated with u. In terms of the error in the estimate $\hat{I} = \frac{1}{2}(f(u)+f(1-u))$, we have

$$\begin{aligned} Var[\tfrac{1}{2}(f(u)+f(1-u))] &= \frac{1}{4}(Var[f(u)]+Var[f(1-u)]+2Cov[f(u),f(1-u)]), \\ &= \frac{1}{2}(Var[f(u)]+Cov[f(u),f(1-u)]). \end{aligned}$$

Note that we used the fact that since u and $1-u$ are both uniform random variables, $f(u)$ and $f(1-u)$ are identically distributed (though not independent). Note also that if $Cov[f(u),f(1-u)] < 0$ then this estimate generated from *one* random variable u has a *lower* variance that the estimate generated from *two* independent random variables u_1, u_2. However note also that if $Cov[f(u),f(1-u)] > 0$, then this estimate is *worse*.

Exercise 5.3

Show that if $f(x)$ is linear, the $Cov[f(u),f(1-u)] = -Var[f(u)]$.

Certainly for the linear function, the anti-thetic variate $1-u$ in combination with the variate u gives a perfect estimate. What can we say in general, i.e. when is the anti-thetic variate useful. The relevent feature of the linear function is that it is monotonic, infact perfectly monotonic. Whenever f is monotonic, then $Cov[f(u),f(1-u)] < 0$, and n variates together with the n anti-thetic variates are better than $2n$ independent variates.

Exercise 5.4

Show that if $f(x)$ is monotonic (increasing or decreasing) and contiuous then $Cov[f(u),f(1-u)] < 0$.

[Hint: $Cov[f(u), f(1-u)] = \int_0^1 (f(x) - I)(f(1-x) - I)$. By the mean value theorem to write $I = f(x^*)$ for some $x^* \in [0,1]$. Now show that the integrand is always negative.]

In fact, the anti-thetic variate does not always help. In particular, when $f(x)$ is symmetric about $\frac{1}{2}$.

Exercise 5.5

Show that if $f(x) = f(1-x)$ then $Cov[f(u), f(1-u)] = Var(f(u))$ and in this case the anthi-thetic variate does not add any value.

Notice that the anti-thetic variate approach never hurts in the sence of worsening a sample. In the case of f perfectly symmetric about $\frac{1}{2}$ the variance of the anti-thetic estimate is the same as the variance of the original sample of size n. The only cost is that one has to do an additional function evaluations. Thus, when the primary cost is random-variate generation, and function computation is cheap, then the anti-thetic variates never hurt, and can considerably help. In this sense anti-thetic variates are almost a no-brainer.

Normal Anti-Thetic Variates. In general the anthithetic variate for a variate generated using the transformation method (eg. the Box-Muller method for Normal random variates) by setting each uniform random variate u used in the generation to $1-u$ may not work. For example in the Box-Muller algorithm,

$$x_1 = \sqrt{-2\ln u_1}\cos 2\pi u_2,$$

and the anthithetic variate would be

$$\begin{aligned} \bar{x}_1 &= \sqrt{-2\ln(1-u_1)}\cos 2\pi(1-u_2), \\ &= \sqrt{-2\ln(1-u_1)}\cos 2\pi u_2. \end{aligned}$$

It is true that x_1 and $\bar{x}_1$ have the same marginal distributions, however they are not perfectly anti-correlated.

Exercise 5.6

Compute the correlation coefficient between x_1 and $\bar{x}_1$.

[Numerical Answer: 0.5822]

In fact the prescription above gives positive correlation. Fortunately, for normal random variates, it is easy to generate the antithetic variate. For the normal random vector $\mathbf{z}$, the anti-thetic vector is $-\mathbf{z}$. It is clear that the two have the same marginal distributions, and that their correlation is -1.

Pricing Options Using Anti-Thetic Paths In the Monte Carlo framework for pricing options, we generate a path p over n time steps of length Δt. In the binomial tree model, to generate the path, at each step one generates a uniform random variate and compares with $\tilde{p}$ to determine if the move will be up or down. Thus to generate a path p in the binomial model, we use a stream of n uniform random variates $u_1, \ldots, u_n$. The anti-thetic path $\bar{p}$ is generated in exactly the same way using the stream of anti-thetic uniform random variates, $1-u_1, \ldots, 1-u_n$. Note taht the anti-thetic path is not necessarily the flipped path, in which every up move is converted to a down move and vice-versa.

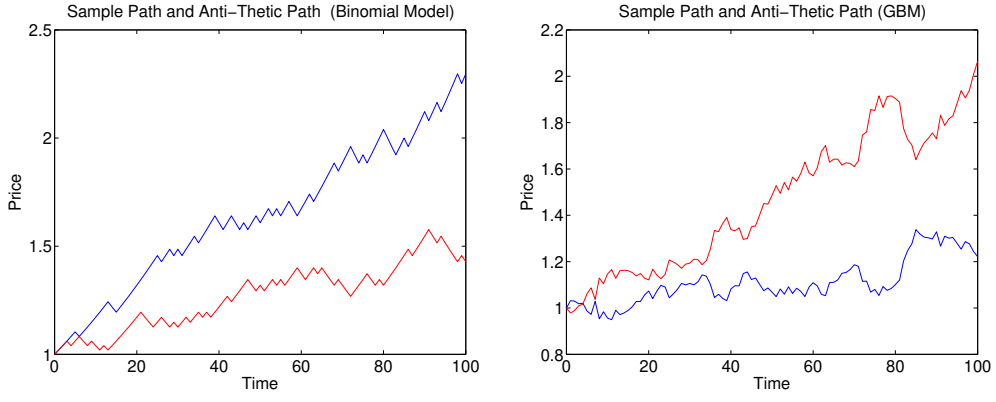
In the GBM model, the path p is generated according to the stochastic dynamics $S(t+\Delta t) = S(t)e^{\eta}$, where $\eta \sim N((r-\frac{1}{2}\sigma_R^2)\Delta t, \sigma_R^2\Delta t)$. The Normal variate η can be generated as follows,

$$\eta = (r-\tfrac{1}{2}\sigma_R^2)\Delta t + \sigma_R\sqrt{\Delta t}z,$$

where z is a standard Normal random variate. Thus the path p is generated using a stream of Normal variates $\eta_1, \ldots, \eta_n$, which in turn uses the stream of standard Normal random variates $z_1, \ldots, z_n$. The anti-thetic path $\bar{p}$ is generated using the anti-thetic stream of variates $\bar{\eta}_1, \ldots, \bar{\eta}_1$, which is generated using the anti-thetic standard normal random variates $-z_1, \ldots, -z_n$,

$$\begin{aligned} \eta_i &= (r-\tfrac{1}{2}\sigma_R^2)\Delta t + \sigma_R\sqrt{\Delta t}z_i, \\ \bar{\eta}_i &= (r-\tfrac{1}{2}\sigma_R^2)\Delta t - \sigma_R\sqrt{\Delta t}z_i. \end{aligned}$$

A sample of the path p and its anti-thetic path $\bar{p}$ are shown in the figure below. Notice that the anti-thetic path is not simply the flipped version of the original path, with every up move replaced by a down move and vice-versa. If $\tilde{p} = \frac{1}{2}$ or if $r = \frac{1}{2}\sigma_R^2$, then it is the case that the anti-thetic path is a flipped version of the original path.



5.2 Common Random Variates

Common random variates are useful when trying to estimate the difference between two expected values. This arises often in computational finance when one wishes to compute the sensitivity of the price of a derivative to some parameter (these sensitivities are often reffered to as the *Greeks* because they are usually denoted by Greek letters.) Let C be the price of the option. Then we know that C is a function of the risk neutral measure (r, σ_R or $\tilde{p}, \lambda_\pm$), and the initial stock price S_0, in addition to the parameters of the derivative (for example time to maturity, T, the strike price K, etc.). Actually it is the expectation,

$$C(r, \sigma_R, S_0, T, K) = E[f(\boldsymbol{z}; r, \sigma_R, S_0, T, K)],$$

where f is the cash flow function for a path which is determined by the standard Normal random vector $\boldsymbol{z}$, given the parameters (in the GBM model). If we were in the binomial model, the expectation would be w.r.t. a uniform random vector $\boldsymbol{u}$ which determines the path. Note that $\tilde{p}$ depends on r and $\lambda_\pm$. The typically used Greeks are the Delta Δ, the Gamma Γ, the Vega ν, the Theta Θ, the Rho ρ,

$$\begin{aligned}
\Delta &= \frac{\partial C}{\partial S_0}, \\
\Gamma &= \frac{\partial^2 C}{\partial S_0^2}, \\
\nu &= \frac{\partial C}{\partial \sigma_R}, \\
\Theta &= -\frac{\partial C}{\partial T}, \\
\rho &= \frac{\partial C}{\partial r},
\end{aligned}$$

One may estimate the sensitivities numerically using a finite differnence, for example

$$\begin{aligned}
\Delta &\approx \frac{1}{2h}(C(r, \sigma_R, S_0 + h, T, K) - C(r, \sigma_R, S_0 - h, T, K)), \\
&= \frac{1}{2h}(E[f(\mathbf{z}; \sigma_R, S_0 + h, T, K)] - E[f(\mathbf{z}; \sigma_R, S_0 - h, T, K)])
\end{aligned}$$

which is the difference of the expectation of two different functions depending on the random variable $\mathbf{z}$

Let f and g be two functions of a random variable u. Suppose that we wish to estimate the difference

$$E[f(u)] - E[g(u)].$$

Suppose that we generate two random variates u_1, u_2 and compute the quantity $f(u_1) - g(u_2)$, where u_1, u_2 have the same marginal distributions as u. We have that

$$Var[f(u_1) - g(u_2)] = Var[f(u_1)] + Var[g(u_2)] - 2Cov[f(u_1), g(u_2)].$$

If u_1 and u_2 are independent, then $Cov[f(u_1), g(u_2)] = 0$. However, if $Cov[f(u_1), g(u_2)] > 0$ then we may gain through dependence. In particular, one might home that if u_1 and u_2 have positive correlation the $f(u_1)$ and $f(u_2)$ will also have positive correlation. In particular, the extreme case is when $u_1 = u_2$ and the correlation is 1. In this case, u_1 and u_2 are called *common variates*.

As with anti-thetic variates, it is not always the case that positive correlation in the u's leads to positive correlation in f and g. If f and g are both monotonic in the same direction, then this will indeed be the case. however if they are monotonic in different directions, for example, then this will not be the case. In particular, take $g = -f$. In this case with common variates leads to a doubling of the variance compared to the case when the variates are independent.

Exercise 5.7

Let $f(u)$ and $g(u)$ be two continuous functions of a random variate with measure $d\mu(u)$.

(a) Show that if f and g are both monotonic in the same direction, then $Cov[f(u), g(u)] > 0$ for any random variable u.
[Hint: Use the fact that for some x, y $E[f] = f(x)$ and $E[g] = g(y)$

$$\begin{aligned} Cov[f(u), g(u)] &= \int d\mu(u)\ (f(u) - E[f])(g(u) - E(g)), \\ &= \int d\mu(u)\ (f(u) - f(x))(g(u) - g(y)), \\ &= \int d\mu(u)\ (f(u) - f(y) + f(y) - f(x))(g(u) - g(y)), \\ &= \int d\mu(u)\ (f(u) - f(y))(g(u) - g(y)), \end{aligned}$$

and now argue that the integrand is always positive.]

(b) Show that if f and g are both monotonic in opposite directions, then $Cov[f(u), g(u)] < 0$ for any random variable u.
In particular, show that if $g = -f$, then $Cov[f(u), g(u)] = -Var[f(u)]$ for any random variable u

---

In applying the common variate approach to computing an option sensitivity one should use the same stream of random variates to compute the expected price for one setting of the parameters and the perturbed set of parameters and then take the difference.

5.3 Control Variates

The anti-thetic and common variates techniques can be applied even when not much is known about the integrand functions. The use of control variates depends on some knowledge of the integrand.

Suppose that we wish to compute $I = \int_0^1 dx\ f(x)$, as usual. Suppose that there is some other function whose integral is known, $G = \int_0^1 dx\ g(x)$, where G is known. Suppose that we now generate a sample $x_1, \ldots, x_n$ and compute two sample estimates

$$\begin{aligned} \hat{I} &= \frac{1}{n}\sum_{i=1}^{n} f(x_i), \\ \hat{G} &= \frac{1}{n}\sum_{i=1}^{n} g(x_i). \end{aligned}$$

Since we know G, the question we ask is whether the difference $G - \hat{G}$ can convey any additional information about I, other than that already contained in $\hat{I}$. In particular, suppose that $f(x)$ and $g(x)$ have a positive correlation, and suppose that $G > \hat{G}$. We can qualitatively conclude that the $g(x_i)$'s were on average less than their expectation. This means that, since there is positive correlation between $g(x)$ and $f(x)$, the $f(x_i)$'s should also be on average less than their expectation, or that we expect that $I > \hat{I}$. The natural thing to do therefore is to somehow correct for this effect by incrementing $\hat{I}$. We consider a very simple form for this increment, namely a linear correction,

$$\hat{I}_1 = \hat{I} + \alpha(\hat{G} - G).$$

The variate $\hat{G}$ is called the control variate. Notice first that $E[\hat{I}_1] = E[\hat{I}]$, and so this new estimate is indeed unbiased because $\hat{I}$ is unbiased. For the variance, we have that

$$\begin{aligned} Var[\hat{I}_1] &= Var[\hat{I}] + \alpha^2 Var[\hat{G}] + 2\alpha Cov[\hat{I}, \hat{G}], \\ &= Var[\hat{G}]\left(\alpha + \frac{Cov[\hat{I}, \hat{G}]}{Var[\hat{G}]}\right)^2 + Var[\hat{I}]\left(1 - \frac{Cov[\hat{I}, \hat{G}]^2}{Var[\hat{I}]Var[\hat{G}]}\right), \\ &= Var[\hat{G}]\left(\alpha + \frac{Cov[\hat{I}, \hat{G}]}{Var[\hat{G}]}\right)^2 + Var[\hat{I}](1 - \rho^2_{\hat{I}\hat{G}}). \end{aligned}$$

It is clear from this expression that the variance is minimum when one choses for *alpha* the optimal value

$$\alpha^* = -\frac{Cov[\hat{I}, \hat{G}]}{Var[\hat{G}]}.$$

In this case, the variance is reduced by a factor $1 - \rho^2_{\hat{I}\hat{G}} < 1$. Noting that $\rho_{\hat{I}\hat{G}} = \rho_{fg}$, we have that, for the optimal choice of α,

$$Var[\hat{I}_1] = Var[\hat{I}](1 - \rho^2_{fg}).$$

Exercise 5.8

Show that $\rho_{\hat{I}\hat{G}} = \rho_{fg}$, and that $\alpha^* = -\frac{Cov[f,g]}{Var[g]}$

Notice that if $Cov[f,g] > 0$ then $\alpha^* < 0$ and the correction to $\hat{I}$ is in the direction we qualitatively argued for. There is, however, a problem with the above prescirption. In order to compute α^*, one needs to know $Var[g]$ as well as $Cov[f,g]$ which ammounts to the knowledge of the additional integrals

$$\begin{aligned} Var[g] &= \int_0^1 dx\ (g(x) - G)^2, \\ Cov[f,g] &= \int_0^1 dx\ (f(x) - I)(g(x) - G). \end{aligned}$$

Certainly, since I is not known, we cannot expect $Cov[f,g]$ to be known, and one probably does not even know $Var[g]$. This raises a problem with the approach, in principle, but in practice, one may overcome this problem by using some pilot estimates for $Cov[f,g]$ and $Var[g]$. Specifically, suppose that n samples $x_1, \ldots, x_n$ are available. One selects the first k samples for the pilot estimate, and over these k samples, one computes sample estimates $\hat{Var}[g]$ and $\hat{Cov}[f,g]$. Using these estimates one computes an estimate $\hat{\alpha}^*$. The remaining $n-k$ are used to compute $\hat{I}_1$.

One might try to use all the samples to compute $\hat{I}_1$ using the estimate $\hat{\alpha}^*$, but since the estimate $\hat{\alpha}^*$ was developed on part of the data, this results in a biased estimate. The bias becomes negligible as n becomes large.

Choosing an Optimal Pilot Sample Size. The analysis is difficult because α is not even an unbiased estimate of α^*. However, we will use a heuristic derivation, making any necessary approximations which are reasonable. Notice that $\hat{\alpha}^*$ is the regression coefficient of $f(x_i)$ onto $g(x_i)$. For simple linear regression with normal, independent $f(x_i)$, we know that the distribution of $\hat{\alpha}^*$ will also be Normal. We have independence, but we do not have Normality. However, we will use this approximation. In this case, the variance of $\hat{\alpha}^*$ is known (for example, see [**?**]). Thus, we have the approximation

$$Var[\hat{\alpha}^*] \approx \frac{Var[f]}{s_g^2},$$

where $s_g^2 = \sum_{i=1}^k (g(x_i) - m_g)^2$ and $m_g = \frac{1}{k}\sum_{i=1}^k g(x_i)$. Thus, $\frac{1}{k}s_g^2$ is the sample variance of the $g(x_i)$'s. Thus, $Var[\hat{\alpha}^*] \approx \frac{Var[f]}{kVar[g]}$ Since $E[\hat{\alpha}^*] \approx -\frac{Cov[f,g]}{Var[\hat{g}]}$, we see that

$$E\left[\left(\hat{\alpha}^* + \frac{Cov[f,g]}{Var[\hat{g}]}\right)^2\right] \approx Var[\hat{\alpha}^*] \approx \frac{Var[f]}{kVar[g]}.$$

Thus we can approximate $E[Var[\hat{I}_1]]$ taking into account the fact that the sample on which $\hat{\alpha}^*$ and $\hat{I}, \hat{G}$ are independent by,

$$\begin{aligned} E[Var[\hat{I}_1]] &\approx E[Var[\hat{G}]]Var[\hat{\alpha}^*] + E[Var[\hat{I}](1-\rho_{fg}^2)], \\ &\approx \frac{Var[g]}{n-k}\frac{Var[f]}{kVar[g]} + \frac{Var[f]}{n-k}(1-\rho_{fg}^2), \\ &= \frac{Var[f]}{n-k}\left(\frac{1}{k} + 1 - \rho_{fg}^2\right). \end{aligned}$$

We can try to minimize this function of k which we can approximately do by setting its derivative with respect to k to zero. This results in a quadratic equation whose positive solution is

$$k^* = \frac{\sqrt{1 + (1-\rho_{fg}^2)n} - 1}{1-\rho_{fg}^2} \approx \sqrt{\frac{n}{1-\rho_{fg}^2}}.$$

We thus see that the optimal pilot sample size is increasing proportional to $\sqrt{n}$, and inversely with $\sqrt{1-\rho_{fg}^2}$. This makes sense in the sense that the larger is ρ_{fg}^2, the more dividend it pays to have a better estimate of α^* and so you should use more samples to do so. The fact that the dependence on these two parameters is square root is an interesting consequence of this heuristic analysis.

The practical prescription is therefore to sample the pilot set maintaining an estimate of ρ_{fg}^2 and the moment $k > \sqrt{\frac{n}{1-\rho_{fg}^2}}$, stop sampling and use the estimate $\hat{\alpha}^*$ from this pilot sample.

Exercise 5.9

Show that (as claimed above), $k^* = \frac{\sqrt{1+(1-\rho_{fg}^2)n}-1}{1-\rho_{fg}^2} \approx \sqrt{\frac{n}{1-\rho_{fg}^2}}$.

5.4 Application to Derivatives Pricing

In pricing one derivative using Monte Carlo, the samples are the paths generated according to the risk neutral measure. In many cases one can compute analytically the price of certain derivatives. These derivatives may then be used as control variates to price a derivative for which one does not have an analytic price.

Example Control Variates We list a number of example control variates.

1. The stock price itself at any point in the path. Since the stock price is a martingale under the risk neutral measure, we have that $E[S(t)] = S_0$ for any time step in a path. In particular a useful control variate is $S(T)$.

2. The European Call option. We know that price of this option analytically:

   Thus The European call option with a variety of strikes and times to maturity can be used as control variates.

3. Barrier options which can also be priced analytically come in a variety of forms. These can be priced analytically. However the analytic price typically assumes a continuous Brownian motion, and so there will be a bias in the price, so one has to be careful to use a small enough time-step so that the bias is negligible.

4. The Asian options whos strikes or payoffs are based on the geometric mean of the stock price. One can show that the geometric mean of a set of log-normal random variables is itself log normal, and can use this to price such options in much the same way as the European call option.

5.4.1 Multiple Control Variates

There is no reason that the prior analysis could not be generalized to the situation where we have multiple functions $g_1, g_2, \ldots, g_d$ whose integrals (expectations) are known. Let $\boldsymbol{G}$ be the known vector of integrals and let $\hat{\boldsymbol{G}}$ be the estimated vector from the samples $x_1, \ldots, x_n$. Then we have the linear correction based on these control variates given by

$$\hat{I}_1 = \hat{I} + \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{G} - \hat{\boldsymbol{G}}).$$

5.5 Importance Sampling

Again lets come back to our usual integral $I = \int_0^1 dx\ f(x)$, which we estimate using the sample $x_1, \ldots, x_n$, each drawn *i.i.d* from $U[0,1]$. We have assumed that the x_i's have been given to us sampled from the uniform distribution. What if we did not have control over the sampling distribution. Suppose instead that $x_i \sim p(x)$, where $p(x)$ is some other density on

$[0,1]$. if we simply computed $\hat{J} = \sum_{i=1}^{n} f(x_i)$ we not be getting an estimate of I. Instead we have that

$$E[\hat{J}] = \int_0^1 dx\ p(x)f(x).$$

Instead of computing $f(x_i)$, suppose instead that we computed $g(x_i) = f(x_i)/p(x_i)$. Thus, lets construct the estimate

$$\hat{I} = \frac{1}{n}\sum_{i=1}^{n} g(x_i).$$

Then we have that $E[\hat{I}] = \int_0^1 dx\ p(x)g(x)$, but since $g(x) = f(x)/p(x)$, we have that

$$E[\hat{I}] = \int_0^1 dx\ f(x),$$

as desired. So it turns out that we can construct an estimate of I given samples from *any* distribution, as long as we know the distribution and as long as this distribution has support wherever f has support. We can thus turn this around and ask, is it possible to actually construct a better sampling distribution, $p(x)$, one that results in an estimate $\hat{I}$ with lower variance than would be obtained from the standard uniform samples? For simplicity, suppose that $f(x) > 0$. If not one can always decompose f into its positive ($f_+ = \max(f, 0)$) and negative ($f_- = -\min(f, 0)$) parts and compute two integrals $\int f = \int f_+ - \int f_-$. To show that there certainly exist such better distributions, consider the density

$$p(x) = \frac{f(x)}{\int_0^1 dx\ f(x)} = \kappa f(x).$$

i.e., the density which is proportional to $f(x)$ itself. For this density, $g(x_i) = \frac{1}{\kappa}$, and so $\hat{I} = \frac{1}{\kappa} = \int_0^1 dx\ f(x) = I$ with zero variance. Thus sampling according to this new ideal density results in an exact estimate with even just one point! Unfortunately, as always, there is a problem. In order to constructe/valuate $p(x)$, we need to know I, and knowing I defeats the whole purpose, which is to estimate I. Thus we immediately identify two important properties that any useful $p(x)$ should satisfy:

1. It should be easy to generate samples from $p(x)$.

2. It should be easy to compute $p(x)$ for any x, which is needed in order to obtain $g(x)$.

The idea behind importance sampling is therefore to construct an approximation to this ideal $p(x)$ which satisfies these two properties. The basic approach is as with control variates, to generate a pilot estimate to the ideal sampling density $p(x) = \kappa f(x)$.

5.5.1 The Bin/Histogram Approach in 1 Dimension

Suppose you have n random samples available. We use k of these to construct a pilot estimate $\hat{p}(x)$ of the ideal sampling density $p(x) = \kappa f(x)$. The estimate will be *piecewise constant*, obtained by sampling $f(x)$ at a few random points and then building $\hat{p}(x)$ from these samples.

Similar to the hybrid Monte Carlo discussed earlier, we bin the interval $[0, 1]$ into k bins of width $\frac{1}{k}$. Each bin b_i is an interval,

$$b_i = \left[\frac{i-1}{k}, \frac{i}{k}\right].$$

In each bin, generate a random point $x_i \in b_i$ uniformly distributed in the bin,

$$x_i = \frac{i - 1 + u_i}{k},$$

where $u_i \sim U[0, 1]$ is a uniform random variable. We now have a piecewise constant estimate to $f(x)$ given by

$$\hat{f}(x) = f(x_i) \qquad \text{for } x_i \in b_i.$$

Notice that $\hat{I}_k = \frac{1}{k}\sum_{i=1}^{k} f(x_i)$ is exactly the hybrid Monte Carlo estimate for I on k samples. Further

$$\begin{aligned}
\int_0^1 dx\ \hat{f}(x) &= \sum_{i=1}^{k} \int_{b_i} dx\ \hat{f}(x), \\
&= \sum_{i=1}^{k} \int_{b_i} dx\ f(x_i), \\
&= \frac{1}{k}\sum_{i=1}^{k} f(x_i), \\
&= \hat{I}_k,
\end{aligned}$$

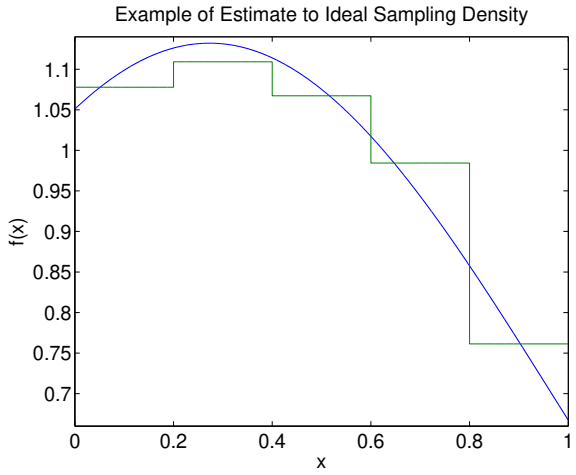
where the penultimate line follows because $f(x_i)$ is a constant over the bin b_i and the width of bin b_i is $\frac{1}{k}$. Thus, the approximate sampling density is given by

$$\hat{p}(x) = \frac{f(x_i)}{\hat{I}_k} \qquad \text{for } x_i \in b_i.$$

If $f(x) > 0$, then everything is fine, but if $f(x)$ can take on the value 0 on sets with positive measure, then it is posible that $f(x_i)$ may be sero for some i, and all future samples from this estimated sample density may be skewed. Thus it helps to *regularize* by pushing it a little toward the uniform density using a regularization parameter ϵ,

$$\hat{p}(x) = \frac{f(x_i) + \epsilon}{\hat{I}_k + \epsilon} \qquad \text{for } x_i \in b_i.$$

The figure below illustrates the ideal density and the estimate of the ideal density, with $\epsilon = 0$. We will take $\epsilon = 0$ in what follows.



We first note that $\hat{p}(x)$ can be evaluated easily for any $x \in (0, 1)$,

$$\hat{p}(x) = f(x_{i^*}),$$

where $i^* = \lfloor kx \rfloor + 1$ is the bin x falls into, i.e., $x \in b_{i^*}$. We now show that it is easy to generate from $\hat{p}(x)$, which is equivalent to generating a random variate by first picking a bin b_i randomly with probability $\frac{f(x_i)}{\hat{I}}$ and then picking a point in the bin uniformly. This can be achieved with a single uniform random variate as follows. First define the the cumulative distribution F_i which corresponds to $f(x_i)$ as follows. $F_0 = 0$ and

$$F_i = F_{i-1} + \frac{f(x_i)}{k\hat{I}_k}, \qquad i = 1, \ldots k$$

Note that $F_k = 1$. Given a uniform random variate u define i_u to be the index such that $F_{i_u} < u \leq F_{i_u+1}$. The random variate x is generated as follows

$$x = \frac{1}{k}\left(i_u + \frac{u - F_{i_u}}{F_{i_u+1} - F_{i_u}}\right).$$

Exercise 5.10

Show that x as generated above from u is distributed according to the estimate to the ideal sampling density, $x \sim \hat{p}(x)$.

The final algorithm is to sample $x_1, \ldots, x_k$ uniformaly. Using these samples we construct $x_{k+1}, \ldots, x_n$ according to $\hat{p}(x)$. The final estimate is

$$\hat{I}_{n-k} = \frac{1}{n-k} \sum_{i=k+1}^{n} \frac{f(x_i)}{\hat{p}(x_i)}.$$

Even with as few as 2 bins, one can get a significant improvement in the variance. Note that the samples yield an unbiased estimate of I, and so we may actually use all the samples in estimating I. It should be possible to get an even lower variance with a final estimate of the form

$$\hat{I} = \alpha \hat{I}_k + (1-\alpha)\hat{I}_{n-k}.$$

Exercise 5.11

Show that $E[\hat{I}] = I$.

Exercise 5.12

Show that $Var[\hat{I}]=$

One can choose the value of α and k to minimize the varianve above. We will approach this question heuristically. What is the optimal value of k? This is a classic case of an optimal stopping problem, in which we are faced with an exploration-exploitation dilemma. The larger k is (the more exploration we do) the smaller $n-k$ is and the less we can exploit our knowledge gained through exploration to sample more efficiently.

5.5.2 Kernel Based Approaches in Multi-Dimensions

Unfortunately the histogram/bin approach suffers from the curse of dimensionality in multi-dimensions. In d dimensions to have just two bins in each dimension requires 2^d samples in the $\hat{p}(\mathbf{x})$ estimation phase. An alternative is to generate k samples uniformly and somehow use these samples to estimate a suitable $\hat{p}(\mathbf{x})$. We will focus on the integral

$$I = \int_{\mathbb{R}^d} d\mathbf{x}\; p(\mathbf{x})g(\mathbf{x}),$$

where the integration domain is now over the entire space. This is more convenient and one can always transform the integral on the unit cube to this form of integral, so there is no

loss of generality. Let $f(\mathbf{x}) = p(\mathbf{x})g(\mathbf{x})$. We will also assume without loss of generality that $f(\mathbf{x}) \geq 0$.

Exercise 5.13

Give a way to transform an integral over the unit hyper-cube to an integral over the entire space.

In general our discussion will be applicable to any kernel function $K(\mathbf{x})$ which typically satisfies two properties:

1. $K(\mathbf{x}) > 0$ for all $\mathbf{x}$.
2. $\int_{\mathbb{R}^d} d\mathbf{x}\ K(\mathbf{x}) = 1$.

In our discussion we will choose the Gaussian kernel,

$$K(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x}}.$$

Suppose that k points $\mathbf{x}_1, \ldots, \mathbf{x}_k$ have been sampled *i.i.d* from $p(\mathbf{x})$. We estimate the function $\hat{f}(\mathbf{x})$ using a radial basis function type of estimate,

$$\begin{aligned} \hat{f}(\mathbf{x}) &= \sum_{i=1}^{k} w_i K\left(\frac{\mathbf{x} - \mathbf{x}_i}{\gamma}\right), \\ &= \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^{k} w_i e^{-\frac{1}{2\gamma^2}(\mathbf{x}-\mathbf{x}_i)^2}, \end{aligned}$$

where γ is a "smoothing" parameter and the w_i are to be estimated. Ideally, we would like $\hat{f}(\mathbf{x}_i) = f(\mathbf{x}_i)$. Define the matrix G by its entries G_{ij} given by

$$G_{ij} = K\left(\frac{\mathbf{x}_i - \mathbf{x}_j}{\gamma}\right).$$

Then, ideally we would like to have that

$$f(\mathbf{x}_i) = \sum_{j=1}^{k} G_{ij} w_j,$$

which is a linear system in $\{w_j\}$ which may be solved as long as G is invertible. Let $\boldsymbol{w}$ be a vector containing the w_j's, and $\boldsymbol{f}$ be a vector containing the function values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)$. Then we wish to have $\boldsymbol{f} = G\boldsymbol{w}$, which for invertible G gives

$$\boldsymbol{w} = G^{-1}\boldsymbol{f}.$$

The resulting estimate $\hat{f}(\mathbf{x})$ is given by

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{k} \sum_{j=1}^{k} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{\gamma}\right) G_{ij}^{-1} f(\mathbf{x}_j).$$

Unfortunately, there is no guarantee that this estimate is always positive, or that the weights w_i are positive. In general, we can obtain the weights by solving a linear program. We determine the w_i's as the solution to the optimization problem

$$\min_{\boldsymbol{w}} \max_{i} |f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i)|,$$
$$\boldsymbol{w} \geq 0.$$

This optimization problem may be solved efficiently by posing it as a linear program, and most mathematics packages (such as Matlab) have standard routines for solving linear programs.

Exercise 5.14

Show that the weights w_i may be obtained by solving the following linear program.

Assuming that we have the weights $\boldsymbol{w} \geq 0$, we get the estimate $\hat{I}_k$ using the fact that $\int_{\mathbb{R}^d} d\mathbf{x}\ K(\mathbf{x}) = 1$,

$$\hat{I}_k = \gamma^d \sum_{i=1}^{k} w_i,$$

and so we estimate the ideal sampling density by

$$\begin{aligned} \hat{p}(\mathbf{x}) &= \frac{\hat{f}(\mathbf{x})}{\hat{I}_k}, \\ &= \frac{1}{\gamma^d \sum_{i=1}^{k} w_i} \sum_{i=1}^{k} w_i K\left(\frac{\mathbf{x}-\mathbf{x}_i}{\gamma}\right). \end{aligned}$$

Note that it is easy to generate from $\hat{p}(\mathbf{x})$ as follows. First generate a center i with probability $w_i / \sum_i w_i$ using a uniform random variate, and then generate a point $\mathbf{x} \sim N(\mathbf{x}_i, \gamma^2)$.

Exercise 5.15

Show that the prescription above for generating a random variate $\mathbf{x} \sim \hat{p}(\mathbf{x})$ is correct.

We now sample $n-k$ points $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_n$ *i.i.d* according to $\hat{p}(\mathbf{x})$. These points are used to compute the final estimate $\hat{I}_{n-k}$,

$$\begin{aligned}\hat{I}_{n-k} &= \frac{1}{n-k}\sum_{i=k+1}^{n}\frac{f(\mathbf{x}_i)}{\hat{p}(\mathbf{x}_i)},\\ &= \frac{1}{n-k}\sum_{i=k+1}^{n}\frac{p(\mathbf{x}_i)g(\mathbf{x}_i)}{\hat{p}(\mathbf{x}_i)}.\end{aligned}$$

Exercise 5.16

Show that $E[\hat{I}_{n-k}] = I$.

Exercise 5.17

What is the computational complexity of computing $\hat{I}_{n-k}$, starting with the estimation of $\hat{p}(\mathbf{x})$ and generating samples from it and computing the final estimate $\hat{I}_{n-k}$.

Optimal Choice of γ. There is some work in choosing the optimal value of γ which depends on some of the properties of $f(\mathbf{x})$.

5.6 Low Discrepancy Sequences

We now describe quasi-Monte Carlo, which is based on the notion of replacing the *iid* random samples with samples from a low discrepancy sequence. The idea is based on delivering a sequence of points which has a sample distribution function close to the distribution function from which we are trying to generate, but in a deterministic way. Suppose we wish to generate from the uniform distribution on the unit hyper-cube in d-dimensions. Then one way to generate a sample of points with approximately the uniform distribution is to generate the points uniformly at random. Suppose that we generate n points, and compute the sample

distribution function F_n and compare with the distribution function of the uniform, F. One of the properties of a trully random sample is that

$$sup_{\mathbf{x}}|F_n(\mathbf{x}) - F(\mathbf{x})| = O(n^{-\frac{1}{2}}).$$

This error measure is called the discrepancy. There are deterministic sequences that one can generate for which the discrepancy between the sample distribution function and the true distribution function is $O(\log^d n^{-1})$, which is an asymptotically better discrepancy in any dimension. We will describe the simplest such low-discrepancy sequence, the Halton sequence. There are others with better properties, for example the Sobol sequence, which the reader may read more about in the references.

5.6.1 The Halton Sequence.

The Halton sequence $H_1, H_2, \ldots$ is relatively simple to generate. Actually one parameter to be specified in the Halton sequence H_n is the prime base p. In this case we define the base-p representation of n as

$$\begin{aligned} n &= d_0 + d_1 p + d_2 p^2 + \cdots, \\ &= \sum_{i=0}^{\infty} d_i p^i \end{aligned}$$

Note that the non-zero digits $d_0, d_1, \ldots$ of this base-p representation of n can be computed in $O(\log_p n)$ time. The Halton number $H_n(p)$ is then obtained by reflecting this representation and adding the decimal point at the begining,

$$\begin{aligned} H_n(p) &= 0.d_0 d_1 d_2 \cdots, \\ &= \sum_{i=0}^{\infty} d_i p^{-(i+1)}. \end{aligned}$$

In generating multi-dimensional Halton variates, one should use a different prime number for each dimension.