TMS150 & MSG400

STOCHASTIC DATA PROCESSING AND SIMULATION 2025

MONTE CARLO METHODS & SIMULATION OF STOCHASTIC PROCESSES WITH PYTHON, PROJECT 3

ANNIKA LANG

Before you start: Work on and answer all the questions which are included in the document and marked with boxes. Details on how to plot, interpret, and evaluate results are discussed in the lecture, and example figures can be found in the lecture slides. Program all tasks in Python.

This project contains two parts. We will first start with an introduction to Monte Carlo methods in the first week before we apply them to the analysis and approximation of stochastic processes.

1. Monte Carlo methods

Originally the Monte Carlo (MC) method was developed for the numerical integration of (deterministic) functions. However, we want to emphasize right away that for low dimensional integration — especially one dimensional integrals — MC methods are quite inferior to the standard methods from numerical analysis. Since integrals naturally appear in all branches of quantitative science, in particular in mathematics (and there especially in probability and statistics), physics, engineering and so forth, there is a huge amount of literature on MC methods and their applications. Thus this project can only give a very first taste of some of the ideas and methods of MC.

Despite the above mentioned fact that MC methods should be avoided for one-dimensional integrals, it is nevertheless so that for this case the ideas can be explained easily, and hence we consider here the problem of determining the (numerical value of the) integral of a bounded — say, piecewise continuous — real valued function f defined on a finite interval. By appropriately scaling and translating f, we may assume that it is defined on the unit interval $[0, 1]$ with values in $[0, 1]$, cf. Figure 1.

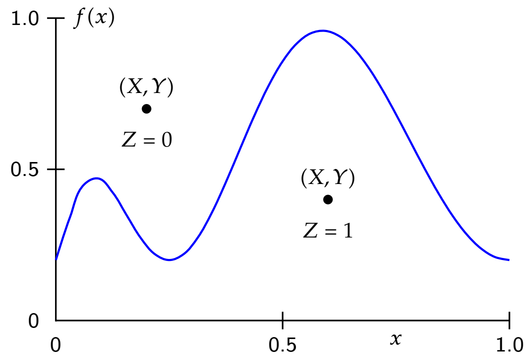

Figure 1. A function f mapping $[0, 1]$ into itself

In this project we consider the two most basic MC methods: *crude* or *naive MC* and *hit-or-miss MC*.

1.1. Crude Monte Carlo.

For crude MC one simply observes that for $U \sim \mathcal{U}([0, 1))$, i. e., U is uniformly distributed on $[0, 1)$,

$$\mathbb{E}\big[f(U)\big] = \int_0^1 f(x)\,\mathrm{d}x.$$

We appeal to the strong law of large numbers to conclude that

$$\frac{1}{M}\sum_{m=1}^{M} f(U^{(m)}) \underset{M\to\infty}{\longrightarrow} \int_0^1 f(x)\,\mathrm{d}x,$$

in the sense of a.s.-convergence, where $(U^{(m)},\, m \in \mathbb{N})$ is an independent sequence of $\mathcal{U}([0,1))$-distributed random variables. Thus we can choose

$$\frac{1}{M}\sum_{m=1}^{M} f(U^{(m)})$$

as an estimator of the integral we are interested in.

The standard toy problem is the estimation of $\pi/4$. Let us therefore observe that the function f given by

$$f(x) := \sqrt{1-x^2}$$

for $x \in [0,1]$ describes a quarter of a uniform circle, i. e., has area $\pi/4$ or more specifically

$$\int_0^1 f(x)\,\mathrm{d}x = \frac{\pi}{4}.$$

In pseudo-code we obtain as algorithm

Algorithm 1.1.

```
$\theta \leftarrow 0$
***for*** $k = 0, \dots, M-1$ ***do***
    *generate* $U \sim \mathcal{U}([0,1))$
    $\theta \leftarrow \theta + \sqrt{1-U^2}$
***end for***
***return*** $\theta/M$
```

Question 1 *(0 points)*

Consider throughout the questions of this project the function $f : [0,1] \to [0,1]$ given by

$$f(x) := x^4$$

and let

$$\theta := \int_0^1 f(x)\,\mathrm{d}x.$$

Compute θ analytically.
Write then a Python program that estimates for fixed sample size M an estimation of θ with crude Monte Carlo.

1.2. **Hit–Or–Miss Monte Carlo.** The idea of hit-or-miss MC is very simple, and somewhat similar to the acceptance-rejection method for generating random numbers: Let X, Y be independent random variables which are uniformly distributed in $[0,1)$, and consider the two-dimensional random variable (X,Y). We define another random variable Z which is equal to 1 if (X,Y) is below the graph of f, and 0 otherwise, cf. Figure 1:

$$Z = \mathbb{1}_{\{Y \le f(X)\}}.$$

Thus Z is a Bernoulli random variable with values 0, 1, and we determine its parameter p:

$$\begin{aligned} p &= P(Z=1) = P\big(Y \le f(X)\big) \\ &= \int_{y \le f(x)} \mathbb{1}_{[0,1]}(x)\,\mathbb{1}_{[0,1]}(y)\,\mathrm{d}x\,\mathrm{d}y = \int_0^1 \int_0^{f(x)} 1\,\mathrm{d}y\,\mathrm{d}x \\ &= \int_0^1 f(x)\,\mathrm{d}x = \theta. \end{aligned}$$

Therefore, θ is equal to the integral that we want to determine. On the other hand we know that for Bernoulli random variables Z with values 0 and 1 the relation $\theta = \mathbb{E}(Z)$ holds true:

$$\int_0^1 f(x)\,\mathrm{d}x = \mathbb{E}(Z).$$

Suppose that $(Z^{(m)}, m \in \mathbb{N})$ is an *iid* sequence with law equal to the one of Z, i. e., the random variables are independent, identically distributed. Then we can apply the strong law of large numbers to conclude that

$$\frac{1}{M}\sum_{m=1}^{M} Z^{(m)}$$

converges P–a.s. to the value of the integral we are interested in. Thus for large $M \in \mathbb{N}$ (P–a.s.)

$$\frac{1}{M}\sum_{m=1}^{M} Z^{(m)} \approx \int_0^1 f(x)\,\mathrm{d}x,$$

that is, we have good reasons to choose $\frac{1}{M}\sum_{m=1}^{M} Z^{(m)}$ as an *estimator* for $\int_0^1 f(x)\,\mathrm{d}x$. If we have sample values $z^{(1)}$, $z^{(2)}$,…, $z^{(M)}$ of $Z^{(1)}$, $Z^{(2)}$, …, $Z^{(M)}$, we shall speak of $\frac{1}{M}\sum_{m=1}^{M} z^{(m)}$ as an *estimate* of the integral.

A simple algorithm implementing this for the computation of an MC approximation of $\pi/4$ is given by

Algorithm 1.2.

```
count ← 0
for $k = 0, \dots, M-1$ do
  generate $U_1, U_2 \sim \mathcal{U}([0,1))$
  if $U_2 \leq \sqrt{1-U_1^2}$ then
    count ← count + 1
  end if
end for
return count/M
```

In this case we observe that the above random variable Z is equal to 1 if and only if $X^2+Y^2 \leq 1$. Therefore we can else reformulate our condition to $U_1^2 + U_2^2 \leq 1$.

Question 2 *(0 points)*

Let f and θ be as in Question 1.

Write a Python program that estimates for fixed sample size M an estimation of θ with hit-or-miss Monte Carlo.

1.3. **Comparison.** Now that we have two working methods, which are easily implemented, the question is, which one should we use?

Of course, the most naive way to reduce the variance of an MC estimator is to increase the sample size M. The formulae above show that the standard deviation of an estimator decreases like $M^{-1/2}$ when M is increasing, and this is also the typical rate of convergence of the strong law of large numbers. Thus, in order to get an estimate which will be one decimal digit better, we need to increase the sample size by a factor 100 — obviously not always the best choice.

The moral of our discussion above is that we should try to find an MC estimator based on a random variable X with small variance. However, we will not concentrate on variance reduction techniques in this project but work on implementations in python of the two methods and testing them.

Question 3 *(1 point)*

Let f and θ be as in Question 1.

Write a python program that computes the Monte Carlo estimates denoted by $\hat{\theta}_i^{\mathrm{MC}}$ and $\hat{\theta}_i^{\mathrm{HM}}$ based on both estimators for a sequence of samples $M = (2^i, i = 1, \dots, 20)$. Compute the errors $|\hat{\theta}_i^{\mathrm{MC}} - \theta|$ and $|\hat{\theta}_i^{\mathrm{HM}} - \theta|$ using your analytical result θ. Plot your results in a loglog plot and add the theoretical reference slope $M^{-1/2}$. What do you observe?

In order to estimate the root mean squared error, use another Monte Carlo estimate with $N = 10$, i. e., compute

$$\left(\frac{1}{N} \sum_{n=1}^{N} \left| \frac{1}{M} \sum_{m=1}^{M} f(U^{(m,n)}) - \theta \right|^2 \right)^{1/2}$$

where $(U^{(m,n)}, m, n \in \mathbb{N})$ is a sequence of iid $\mathcal{U}([0,1))$-distributed random variables. Observe that you need in addition to the loop over M, as you did in Question 5, one more loop over N now. Do the same for hit-or-miss Monte Carlo. Plot your results in a similar way as in the previous task. Is the result as expected? What happens if you change N?

2. Simulation of stochastic processes

We are unable to exactly predict the future, for example the weather or stock prices. A mathematical tool to describe this time dependent phenomenon are randomness and stochastic processes. These are as abstract objects often defined continuously in time and possibly in space. This poses a challenge when trying to compute such processes, since computers are able to perform only finitely many computations in finite time. Usually, one therefore approximates continuous time stochastic processes on discrete time grids.

The goal of this project is to generate discrete sample paths of time continuous stochastic processes and analyze how approximations on different time grids influence the quality of the simulation. We will start with shortly introducing the theory of stochastic processes before introducing the important example of a Brownian motion. This process will be used as building block for more advanced stochastic processes that we in the last part of the project approximate. To perform so-called strong and weak error analysis, we will build on our knowledge from the last project on Monte Carlo methods.

2.1. Stochastic processes and Brownian motion.

This first section of the project is devoted to the introduction of stochastic processes in discrete and continuous time and of Brownian motion together with its simulation. Let us start with the definition of a stochastic process.

Definition 2.1. Let $\mathbb{T} \subset \mathbb{R}$ be a set. A *stochastic process* X with *parameter space* $\mathbb{T}$ is a measurable function $X : \Omega \times \mathbb{T} \to \mathbb{R}$, where $(\Omega, \mathcal{A}, P)$ denotes the probability space.

In other words, a stochastic process is a collection of random variables indexed by the set $\mathbb{T}$. We use $X := (X(t), t \in \mathbb{T})$ as equivalent notation.

The parameter t is called the *time parameter*. We distinguish between *discrete time* for our simulated paths, where usually $\mathbb{T} = \mathbb{Z}$ or $\mathbb{T} = \mathbb{N}$, and *continuous time*, where usually $\mathbb{T} = \mathbb{R}$, $\mathbb{T} = \mathbb{R}_+$, or $\mathbb{T} = [0, T]$ for some $T < \infty$.

A plot of a stochastic process $X(\omega, t)$ as a function of $t \in \mathbb{T}$ for fixed $\omega \in \Omega$ is called a *sample path* of the process. When a stochastic process is plotted, it is a (hopefully typical) sample path that is depicted.

The probably most well-known example of a stochastic process in continuous time is Brownian motion. It describes fluctuations, going back to the observations of Brown of polls moving on a water surface. We will use it as basic driving noise for our simulations.

Let $\mathbb{T} = \mathbb{R}_+$ or $\mathbb{T} = [0, T]$ for some finite $T > 0$.

Definition 2.2. A *Brownian motion* $W : \Omega \times \mathbb{T} \to \mathbb{R}$ is a stochastic process that satisfies

- $W(0) = 0$,
- W is P-almost surely continuous,
- W has independent increments,
- $W(t) - W(s) \sim \mathcal{N}(0, t - s)$ for all $0 \leq s \leq t$.

As a mathematical object, Brownian motion is also known as *Wiener process*.

For $\mathbb{T} = [0, T]$ let us consider a partition of time into N intervals given by

$$0 = t_0 < t_1 < \cdots < t_N = T.$$

Typical partitions are equidistant partitions such that $h := N^{-1}$ and

$$t_n = n \cdot h.$$

In order to obtain nested grids of time discretizations which will be of help in the simulations in this project, we will typically set $N := 2^i$ for some $i \in \mathbb{N}$.

From the definition of Brownian motion we obtain that Brownian motion can be rewritten as a telescopic sum

$$W(T) = \sum_{n=1}^{N} W(t_n) - W(t_{n-1}),$$

where all *increments* $W(t_n) - W(t_{n-1})$ are independent $\mathcal{N}(0, h)$-distributed random variables. Therefore we can simulate the path of a Brownian by simulating in each time step a random number $\eta \sim \mathcal{N}(0, h)$ and setting

$$W(t_n) := W(t_{n-1}) + \eta.$$

In order to generate the same Brownian sample path on different resolutions, we observe that for grids with step size $N_1 := 2^i$ and $N_2 := 2^{i+1}$, the first two increments on the fine grid of the Brownian motion are summed together the increment of the Brownian motion on the coarser grid. More specifically, if η_1 and η_2 are the increments on the grid with respect to N_2 time steps, then

$$\tilde{\eta}_1 := \eta_1 + \eta_2$$

is the first increment of the same sample of the Brownian motion on the coarser grid with N_1 grid points. This observation extends to all grid points on the time interval and to all neighboring resolutions of the time grid.

The first task of the project is to simulate a Brownian sample path on a sequence of different time grids:

Question 4 *(1 point)*

Compute a sample path of a Brownian motion at all resolutions $(h_i, i = 1, \ldots, 10)$ with $h_i = 2^{-i}$ based on the same noise. Plot your result and explain it.

2.2. **Approximation of stochastic processes.** Let us in this section assume that $\mathbb{T} = [0, T]$, i. e., that we are working on a finite time interval, which is reasonable for computations in finite time. We are interested in simulating the stochastic process $X : \Omega \times \mathbb{T} \to \mathbb{R}$ given by

$$X(t) = \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W(t)\right),$$

where W denotes a Brownian motion. This example is motivated from the theory and approximation of stochastic differential equations. Although we know the process explicitly in this specific case, we will try to approximate it by a discrete time process and show how it converges to the theoretical process. We do this for simplicity in order to really control the errors explicitly. In applications, we will usually not know the process X explicitly but only have an implicit description of it. In the given case, we can argue that it makes sense to approximate the process since the computation of the exponential function is expensive if we need it at many times t compared to other operations on a computer.

Instead of computing X, let us compute $X_h : \Omega \times \{t_n := n \cdot h, n = 0, \dots, N\} \to \mathbb{R}$ by the recursion scheme

$$X_h(t_n) = (1 + h\mu)X_h(t_{n-1}) + \sigma X_h(t_{n-1})(W(t_n) - W(t_{n-1}))$$

and $X_h(0) = 1$.

If we simulate sample paths of the true process and its approximation, we see in pictures that the difference gets smaller for finer time grids:

Question 5 *(1 point)*

Let $\mu := 2$ and $\sigma := 1$ and $h_i = 2^{-i}$. Compute a sample path of X and sample paths $(X_{h_i}, i = 1, \dots, 10)$ based on the same noise. Plot all results in the same graph and describe your observations. Is there a difference to the plot in Question 1?

To quantify the error mathematically, let us introduce two different concepts of errors.

Definition 2.3. Let X be a stochastic process and $(X_h, h \in (0, 1])$ be a family of approximations of the process. Then

$$\mathbb{E}\left[(X(T) - X_h(T))^2\right]^{1/2}$$

is called the *strong error*, also known as L^2 or root mean squared error.

The family $(X_h, h \in (0, 1])$ is said to *converge strongly* to X if

$$\lim_{h \to 0} \mathbb{E}\left[(X(T) - X_h(T))^2\right]^{1/2} = 0.$$

It converges with *rate* γ if there exists a constant C such that for all h sufficiently small

$$\mathbb{E}\left[(X(T) - X_h(T))^2\right]^{1/2} \le Ch^\gamma.$$

Since we are usually not able to compute the expectation in this expression explicitly, one possibility is to approximate it by a Monte Carlo simulation. We therefore can approximate the strong error for sufficiently large $M \in \mathbb{N}$ by

$$\mathbb{E}\left[(X(T) - X_h(T))^2\right]^{1/2} \approx \left(\frac{1}{M}\sum_{m=1}^{M}((X(T)^{(m)} - X_h(T)^{(m)})^2\right)^{1/2},$$

which will is the next task of this project:

Question 6 *(2 points)*

Estimate the strong error with a Monte Carlo simulation based on $M = 1000$ (or more if you like) for all $(h_i, i = 1, \dots, 10)$. Plot your results with h in the x-axis and the strong error in the y-axis in a loglog plot and add a reference slope $h^{1/2}$. Describe the plot and how the simulated errors behave.

Let us continue with a second type of convergence which is of relevance if we are interested in the convergence of so-called *quantities of interest* related to the distribution of the process instead of properties of the sample paths.

Definition 2.4. Let X be a stochastic process and $(X_h, h \in (0, 1])$ be a family of approximations of the process. Furthermore let $\phi : \mathbb{R} \to \mathbb{R}$ be a function within a suitable class of test functions. Then

$$|\mathbb{E}[\phi(X(T))] - \mathbb{E}[\phi(X_h(T))]|$$

is called the *weak error*.

The family $(X_h, h \in (0, 1])$ is said to *converge weakly* to X if

$$\lim_{h \to 0} |\mathbb{E}[\phi(X(T))] - \mathbb{E}[\phi(X_h(T))]| = 0$$

for all test functions ϕ. It converges with *rate* γ if there exists a constant C such that for all h sufficiently small

$$|\mathbb{E}[\phi(X(T))] - \mathbb{E}[\phi(X_h(T))]| \leq Ch^{\gamma}.$$

For weak errors, if we assume that we know $\mathbb{E}[\phi(X(T))]$ explicitly, the computation of the error can be done using the Monte Carlo estimator

$$|\mathbb{E}[\phi(X(T))] - \mathbb{E}[\phi(X_h(T))]| \approx \left|\mathbb{E}[\phi(X(T))] - \frac{1}{M}\sum_{m=1}^{M} \phi(X_h(T))^{(m)}\right|$$

for a sufficiently large number of sample paths M.

As has been seen in the lecture and the previous project, the estimation of the error is very sensitive to the choice of M. Be aware of that in your simulations. You should especially keep in mind that the total error behaves additive, i. e., is given by

$$\frac{1}{\sqrt{M}} + h^{\gamma}.$$

This means that whenever h is very small, the Monte Carlo error will dominate and be visible in your simulated convergence rates.

We should mention that the weak error is bounded by the strong error if ϕ is Lipschitz continuous. This leads to the same rate of weak convergence as obtained for strong convergence. In this project we will observe that this is not optimal and a general rule of thumb holds that says that the weak rate of convergence is twice the strong one.

Question 7 *(2 points)*

Estimate the weak error of

$$|\mathbb{E}[X(1)] - \mathbb{E}[X_h(1)]|$$

with a Monte Carlo simulation based on $M = 1000$ (and as many more as you can compute in a reasonable time) for all $(h_i, i = 1, \dots, 10)$. Observe that

$$\mathbb{E}[X(1)] = \exp(\mu).$$

Plot your results with h in the x-axis and the weak error in the y-axis in a loglog plot and add a reference slope that behaves as h. Discuss the plotted results and compare them to your results in the previous task, where you simulated strong errors.

Having chosen the simplest test function for simulation, i.e., the identity, we new the quantity $\mathbb{E}[X(1)] = \exp(\mu)$ exactly. This automatically allows us to compute weak convergence errors for all affine linear test functions. To see this, let the affine linear function be given by

$$\phi(x) = ax + b$$

for some parameters $a, b \in \mathbb{R}$. Then, due to the linearity of the expectation and that the expectation of a constant is the constant, we obtain

$$\mathbb{E}[\phi(X(1))] = \mathbb{E}[aX(1) + b] = a\,\mathbb{E}[X(1)] + b = a\exp(\mu) + b.$$

Therefore, our simulation of weak errors in the previous task extends naturally to the class of affine linear test functions. At the same time, this method does not automatically enable us to compute expectations of nonlinear functions ϕ. In the last task of this project, take another (interesting) test function outside of the class of affine linear functions. You will need to compute or carefully approximate $\mathbb{E}[\phi(X(1))]$ to have a good "exact" solution to compute errors.

Question 8 *(1.5 points)*

Do the same as in Question 4 but choose another test function than $\phi = \mathrm{Id}$ or any linear affine transformation of it. Plot and describe your results and compare them to the error plots in the previous tasks.

Finally, writing a structured and clear report is highly appreciated and recommended. Therefore, we will reward you with an extra 0.5 points for a nice report and well-commented code. Please try to be to the point instead of submitting lengthy chaotic reports with fluffy formulations not related to the questions. It is one of the big challenges when learning writing reports to find the right balance between being brief and including enough background to be concise.

Question 9 *(0.5 points)*

Make sure that your report is clearly structured and good to read and that the code attached to the report is properly structured and commented.

Assignment A3 has a maximum of *9 points* and must be written as a single LaTeX report and be submitted to Canvas. Notice the "recommended deadline" on the course webpage.

Please use the recommended report template provided on the course page, and in case you worked out the exercises with some student, remember to write the name of said student as a footnote in the report front page. Also, recall that each submitted report and the code therein is an *INDIVIDUAL* submission, no group work.

Finally, since you have to write a proper report: as from the provided template, you are also asked to produce some background on the methodology you use. So do not just write answers to the exercise questions.

The report should not be longer than 10 pages including figures, but excluding appendix. Figures and axes labels should be big enough to be readable if printed. It is OK to use colors.

Full details on grading are of course available at `https://chalmers.instructure.com/courses/35741/pages/course-pm`