Technische Universität München
Lehrstuhl für Informatik V
Dr. Tobias Neckel
M. Sc. Friedrich Menhorn

SS 2018

May 09, 2018

Algorithms for Uncertainty Quantification

Tutorial 3: Standard Monte Carlo sampling

In this worksheet, we focus on standard Monte Carlo sampling.

Monte Carlo integration

In the lecture, we saw that we can use Monte Carlo sampling to approximate integrals. To this end, consider $f : [0, 1] \to \mathbb{R}$. The aim is to evaluate $I_f = \int_0^1 f(x)dx$.
The Monte Carlo approximation of I reads

$$I_f \approx \hat{I}_f = \frac{1}{N} \sum_{i=1}^{N} f(U_i), \tag{1}$$

where $U_i \sim \mathcal{U}(0, 1)$. The associated standard error is

$$\hat{\sigma}_{\hat{I}_f} = \frac{\hat{\sigma}_f}{\sqrt{N}}, \tag{2}$$

where $\hat{\sigma}_f^2 = \frac{1}{N-1} \sum_{i=1}^{N} (f(U_i) - \hat{I}_f)^2$.

Assignment 1

The root mean squared error (RMSE) of an approximation $\hat{X}$ is given as

$$\text{RMSE} = \sqrt{\mathbb{E}[(\hat{X} - X)^2]}, \tag{3}$$

where X is the exact result. Using the RMSE, show Eq. 2.
Hint: $\bar{X}$ is an unbiased estimator.

Solution 1

Given the Monte Carlo estimator

$$\hat{I}_f = \overline{f(U)} = \frac{1}{N}\sum_{i=1}^{N} f(U_i)$$

We compute the mean squared error (MSE) between the exact integral I_f and its approximation $\hat{I}_f$:

$$\text{MSE} = \mathbb{E}[(\hat{I}_f - I)^2] = \mathbb{E}[(\overline{f(U)} - \mathbb{E}[f(U)])^2].$$

Using that $\overline{f(U)}$ is unbiased, i.e. $\mathbb{E}[\overline{f(U)}] = \mathbb{E}[f(U)]$, we get

$$\begin{aligned}\mathbb{E}[(\overline{f(U)} - \mathbb{E}[f(U)])^2] &= \mathbb{E}[(\overline{f(U)} - \mathbb{E}[\overline{f(U)}])^2] \\ = Var[\overline{f(U)}] = Var[\frac{1}{N}\sum_{i=1}^{N} f(U_i)] &= \frac{1}{N^2}\sum_{i=1}^{N} Var[f(U_i)] \\ = \frac{1}{N^2} N\sigma_f^2 = \frac{\sigma_f^2}{N} &\approx \frac{\hat{\sigma}_f^2}{N}.\end{aligned}$$

Thus, the approximated RMSE is given as

$$\text{RMSE} = \sqrt{\text{MSE}} \approx \frac{\hat{\sigma_f}}{\sqrt{N}}$$

Short introduction to chaospy[1]

Import `chaospy` in your program as

```
import chaospy as cp
```

Define the distribution of interest as

```
distr = cp.chaospyDistribution(parameters)
```

where

```
chaospyDistribution = {Uniform, Normal, Beta, Gamma etc.}
```

[1] the online documentation is available at `http://chaospy.readthedocs.io/en/master/`

For example, the syntax to define an uniform distribution defined on $[a, b]$ is

```
distr = cp.Uniform(a, b)
```

To sample from the defined distribution `distr`,

```
samples = distr.sample(size=sampleSize),
```

where `sampleSize` is the desired number of samples. Therefore, if you want to write a `python` script to e.g. generate 1000 uniform random variables in $[0.2, 0.5]$, the syntax is

```
import chaospy as cp

distr = cp.Uniform(0.2, 0.5)
samples = distr.samples(size=1000)
```

Assignment 2

Write a `python` program to estimate $f(x) = \int_0^1 sin(x)dx$ using Eq. (1) and compute the associated standard error using Eq. (2) for $N = [10, 100, 1000, 10000, 100000, 1000000]$. Use the `chaospy` library to generate random variables. Plot the standard error in a loglog plot. What do you observe?
Optional: Integrate $f(x)$ exactly, compute the exact error $\epsilon = |f(x) - \hat{I}|$ and compare to Eq. (2).
Why does Monte Carlo quadrature using uniform samples $U_i \sim \mathcal{U}(0, 1)$ work for integration on $[0, 1]$ for an arbitrary function $g(x)$?

$$\mathbb{E}[g(x)] = \int_{-\infty}^{\infty} g(x)p_U(x)dx =$$
$$\int_0^1 g(x)p_U(x)dx = \int_0^1 g(x)\frac{1}{1-0}dx =$$
$$\int_0^1 g(x)dx = I_g$$

And we can approximate this **expected value** using Monte Carlo sampling from Eq. 1:

$$\mathbb{E}[g(x)] \approx \frac{1}{N}\sum_{i=1}^{N} g(U_i)$$

Assignment 3

Using the same approach as in Assignment 2, estimate $\int_2^4 sin(x)dx$ using Monte Carlo integration.
Two possible approaches:

1. Use integration by substitution and sample from $U_i \sim \mathcal{U}(0,1)$ to reuse approach of Assignment 2
2. Sampling from $\hat{U}_i \sim \hat{\mathcal{U}}(2,4)$ directly

1. Integration by substitution is given by:

$$\int_{T(a)}^{T(b)} f(x)dx = \int_a^b f(T(x))T'(x)dx$$

Our bounds are $a = 2$ and $b = 4$ and $T(x)$ is a linear mapping $T(x) : [0,1] \to [2,4]$. We found this mapping in Worksheet 2:

$$T(x) = (b-a)x + a.$$

Hence, we compute the integral in the following way:

$$\int_2^4 sin(x)dx = \int_0^1 sin(T(x))T'(x)dx =$$
$$\int_0^1 sin((b-a)x+a)(b-a)dx = (b-a)\int_0^1 sin((b-a)x+a)dx =$$
$$(4-2)\int_0^1 sin((4-2)x+2)dx = 2\int_0^1 sin(2x+2)dx \approx 2\frac{1}{N}\sum_{i=0}^{N} sin(2U_i+2)$$

2. We take a look at the expected value:

$$\mathbb{E}[sin(x)] = \int_{-\infty}^{\infty} sin(x)p_U(x)dx =$$
$$\int_a^b sin(x)\frac{1}{b-a}dx = \int_2^4 sin(x)\frac{1}{4-2}dx =$$
$$\frac{1}{4-2}\int_2^4 sin(x)dx$$

Therefore:

$$\int_2^4 sin(x)dx = (4-2)\mathbb{E}[sin(x)] \approx 2\frac{1}{N}\sum_{i=0}^{N} sin(\hat{U}_i)$$

Monte Carlo sampling for UQ

In the lecture, we saw that we can employ Monte Carlo sampling in uncertainty propagation. To this end, assume that we have a model $G : \mathbb{A} \to \mathbb{B}$, where $\mathbb{A}$, $\mathbb{B}$ are some domains (usually G is given in terms of an ordinary or partial differential equation).

In standard deterministic setting, the inputs x of G (physical parameters, geometry, initial conditions, boundary conditions etc.) are considered to be known with certainty. Therefore, the problem consists in computing $y = G(x)$ i.e., evaluating G once.

However, in most realistic scenarios, the inputs of a physical or engineering model are not know with certainty. Therefore, let $x_r \subset x$ denote the uncertain input(s). Usually, uncertainty is modeled in terms of random variables/vectors. Hence, in this setting, the problem consists in propagating the uncertainty in x_r through G and computing a quantity of interest (QoI) $F(G(x_r, x \setminus x_r))$, where F is a given functional. Whatever method we employ for uncertainty propagation, G needs be evaluated multiple times.

Monte Carlo sampling is one of the most basic algorithms for uncertainty quantification. Assuming the previously defined setting, Monte Carlo sampling reads

- model x_r, i.e. the uncertain input(s), as a random variable/vector
- generate $\{x_r^i\}_{i=1}^N$ random samples
- evaluate $y_i = G(x_r^i, x \setminus x_r), i = 1, \ldots, N$
- $q = F(y_1, \ldots, y_N) = \text{QoI}$

Monte Carlo sampling for UQ

Consider the linear damped oscillator from Tutorial 1

$$\begin{cases} \frac{d^2y}{dt^2}(t) + c\frac{dy}{dt}(t) + ky(t) = f\cos(\omega t) \\ y(0) = y_0 \\ \frac{dy}{dt}(0) = y_1, \end{cases} \tag{4}$$

where c is the damping coefficient, k the spring constant, f the forcing amplitude, ω the frequency, y_0 represents the initial position, whereas y_1 is the initial velocity.

Considering $t \in [0, 20]$, $\Delta t = 0.01$, $c = 0.5$, $k = 2.0$, $f = 0.5$, $\omega = 1.0$, $y_0 = 0.5$, $y_1 = 0.0$, use the `odeint`[2] function from `scipy.integrate` to discretize the model from Eq. (4). *Hint: re-use the implementation from Tutorial 1.*

Assignment 4

Compute $y(10)$.

Assignment 5

Assume that $\omega \sim \mathcal{U}(0.95, 1.05)$. Write a `python` program and employ the `chaospy` library to propagate the uncertainty in ω through the model in Eq. (4) using Monte Carlo sampling with N = [100, 10000, 100000] samples. Compute the mean and variance of $y(10)$ and compare the mean value to the deterministic result. What do you observe? Perform now a Monte Carlo sampling simulation assuming that $\omega \sim \mathcal{U}(0.8, 1.2)$. What do you observe?

[2] see `https://docs.scipy.org/doc/scipy-0.17.0/reference/generated/scipy.integrate.odeint.html`