Problem Set 7

Exercises for course Fundamentals of Simulation Methods, WS 2021

Prof. Dr. Mario Flock, Prof. Dr. Friedrich Röpke
Tutors: Brooke Polak (brooke.polak@uni-heidelberg.de | ITA), Glen Hunter (glen.hunter@uni-heidelberg.de | ITA), Jan Henneco (jan.henneco@h-its.org | HITS)
Offices: HITS: Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg; ITA: Albert-Ueberle-Strasse 2, 69120 Heidelberg

Hand in until Wednesday, 08.12.2021, 23:59
Tutorials times: 09.12 - 10.12.21
Group 1: Brooke | Thursday 11:00 - 13:00
Group 2: Glen | Thursday 14:00 - 16:00
Group 3: Jan | Friday 11:00 - 13:00

Random Numbers and Monte Carlo Integration [8pt]

1. Pitfalls of pseudo-random number generation

Consider the linear congruential random number generator RANDU, introduced by IBM in System/360 mainframes in the early 1960s. (Donald Knuth called this random number generator "really horrible", and it is indeed notorious for being one of the worst generators of all time.) The recursion relation of RANDU is defined by

$$I_{i+1} = (65539\, I_i) \bmod 2^{31}, \tag{1}$$

and needs to be started from an odd integer. The obtained integer values can be mapped to pseudo-random floating point numbers $u_i \in [0, 1]$ through

$$u_i = \frac{I_i}{2^{31}}. \tag{2}$$

(a) Implement this number generator. Make sure that you do not use 32-bit integer arithmetic, otherwise overflows will occur. (Use 64-bit integer arithmetic instead, or double precision for simplicity – its precision is sufficient to represent the relevant integer range exactly.)

(b) Now generate 2-tuples of successive random numbers from the sequence generated by the generator, i.e. $(x_i, y_i) = (u_{2i}, u_{2i+1})$. Generate 1000 points and make a scatter plot of the points in the unit square. Does this look unusual? How does this look in 3D (using 3-tuples)?

(c) Now zoom in by a large factor onto a small region of the square, for example $[0.2, 0.201] \times [0.3, 0.301]$, and generate enough points that there are again 1000 points within the small region as before. Interpret the result.

(d) Repeat a) - c) for your favorite standard random number generator.

2. Performance of Monte Carlo integration in different dimensions [8pt]

We would like to compare the performance of the Monte Carlo integration technique with the regular midpoint method. To this end, consider the integral

$$I = \int_V f(\vec{x})\, \mathrm{d}^d\vec{x}, \tag{3}$$

where the integration domain V is a d-dimensional hypercube with $0 \leq x_i \leq 1$ for each component of the vector $\vec{x} = (x_1, x_2, \ldots, x_d)$. The function we want to integrate is given by

$$f(\vec{x}) = \prod_{i=1}^{d} \frac{3}{2}\left(1 - x_i^2\right). \tag{4}$$

This has an analytic solution of course, which is $I = 1$ independent of d, but we want to ignore this for the moment and use the problem as a test of the relative performance of Monte Carlo integration and ordinary integration techniques. To this end, calculate the integral in dimensions $d = 1, 2, 3, \ldots, 10$, using

(a) the midpoint method, where you divide the volume into a set of much smaller hypercubes obtained by subdividing each axis into n intervals, and where you approximate the integral by evaluating the function at the centers of the small cubes.

(b) standard Monte Carlo integration in d dimensions, using N random vectors (don't use the "wrong" random number generator from the previous problem!).

For definiteness, adopt $n = 6$ and $N = 20000$. For both of the methods, report the numerical result for I and the CPU-time needed for each of the dimensions $d = 1, 2, \ldots, 10$. There are timing modules for the different programming language that you can use. (If you manage, you can also go to slightly higher dimensions.)

3. Propability Transformation and Metropolis Monte Carlo [4pt]

In this exercise you are asked to transform a flat probability distribution

$$p(x) = 1 \;\; if \; x \in [0, 1]$$
$$p(x) = 0 \;\; otherwise$$

into a distribution of the form $p_{new}(x) = 1/x^2$, realised on the interval $[1, 20]$

1. Derive an *exact inversion*, so that $y(x)$ is distributed according to $p_{new}(x)$ if x is drawn from the flat distribution.

2. Use the Metropolis Monte Carlo formalism to achieve the same goal numerically, that is: Create a set of one million random numbers, distributed according to $p_{new}(x)$ by starting at an arbitrary point in $[1, 20]$ and accepting/dumping new points based on the acceptance criterion you learned about in the script/lecture. To generate new points we suggest using a normal distribution: $x_{i+1} = N(x_i, \sigma)$ with step size $\sigma = 0.1$, but you also can choose other options.

3. Plot your results from 1.) and 2.) in the form of histograms and compare to the target distribution $p_{new}(x)$. Can you observe pathological aberrations, if you set the step size too small for the Metropolis Monte Carlo scheme?