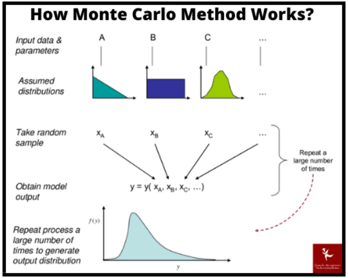
Algorithms and Data Structures

Quadrature
Monte Carlo Integration



Learning goals

- Simple Monte Carlo
- Hit-or-Miss approach

SIMPLE MONTE CARLO

Goal: Calculate $I(f) = \int_a^b f(x)\, dx$

- We define

$$I(f) = (b-a)\int_a^b f(x) \cdot \frac{1}{b-a}\, dx = (b-a) \cdot \mathbb{E}[f(x)]$$

  with $x \sim U(a, b)$

- With $x_i \overset{iid}{\sim} U(a, b), i = 1, ..., n$ the Monte Carlo estimation is given by

$$Q_{MC}(f) = \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$$

- By "sampling" n independent random numbers from $U(a, b)$ an estimate for the integral can be calculated.

SIMPLE MONTE CARLO / 2

Monte Carlo is a **non-deterministic** approach. The estimation for the integral $\int_a^b f(x)\,dx$ is subject to randomness:

- The strong law of large numbers states that $Q_{MC}(f)$ converges almost certainly towards $I(f) = \mathbb{E}[f(x)]$ for $n \to \infty$
- We can derive the variance from $Q_{MC}(f)$:

$$\begin{aligned}\mathrm{Var}\left(Q_{MC}(f)\right) &= \mathrm{Var}\left(\frac{b-a}{n}\sum_{i=1}^{n} f(x_i)\right) \\ &\overset{iid}{=} \frac{(b-a)^2}{n^2}\cdot n \cdot \mathrm{Var}(f(x)) = \frac{(b-a)^2}{n}\sigma^2\end{aligned}$$

where $\sigma^2 = \mathrm{Var}(f(x))$ denotes the variance of the random variable $f(x)$ with $x \sim U(a,b)$ which can be empirically estimated by

$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(x_i)\right)^2, x_i \overset{iid}{\sim} U(a,b)$.

SIMPLE MONTE CARLO / 3

- If $\sigma^2 < \infty$ the variance of the estimate (and thus also the worst case error of the procedure) approaches 0 for $n \to \infty$
- Monte Carlo also works well in multidimensional settings:
  - The Monte Carlo integration can simply be generalized to multidimensional integrals $\int_\Omega f(\boldsymbol{x})\, d\boldsymbol{x}$ with $\Omega \subset \mathbb{R}^d$ by drawing the random variables uniformly distributed in the d-dimensional space Ω.
  - The variance is then

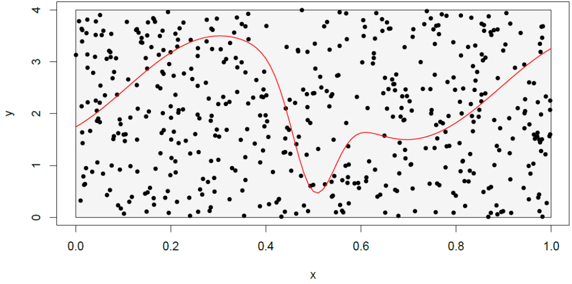
$$\mathrm{Var}\left(Q_{MC}(f)\right) = \frac{V^2}{n}\sigma^2, \quad V = \int_\Omega d\boldsymbol{x}$$

  - In particular, the speed of convergence for the variance **does not** depend on the dimension of the function to be integrated.

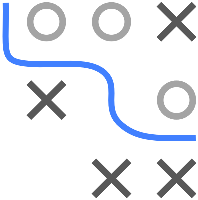
HIT-OR-MISS

Idea:

We draw *n* independently uniformly distributed data points from a rectangle enclosing our function:



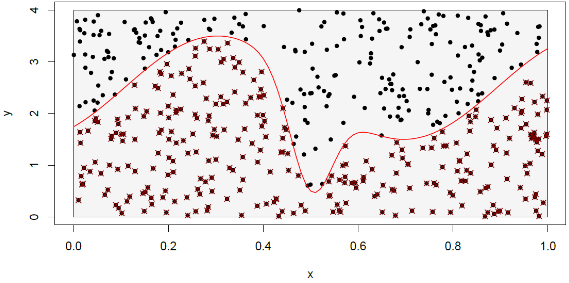
HIT-OR-MISS / 2



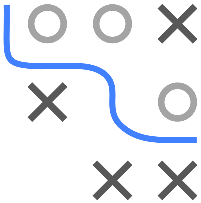
"Hit-or-Miss" Approach:

We still consider the integral $\int_a^b f(x)dx$. We assume that $0 \leq f(x) \leq c$. If we count the number of hits (the points underneath the curve), we obtain the integral by:

$$I(f) \approx \frac{Hits}{n} \cdot \text{area of the rectangle} = \frac{\sum_{i=1}^{n} \mathbf{1}_{y_i \leq f(x_i)}}{n} \cdot c \cdot (b-a)$$



HIT-OR-MISS / 3

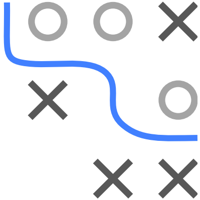


```
m$estimate # Estimation of area
## [1] 2.288
m$hits # Number of points underneath the curve
## [1] 286
```

$$I(f) \approx \frac{286}{500} \cdot 1 * 4 = 2.288$$

This naive method works well for simple examples, but error rates are high for more complex applications.

HIT-OR-MISS / 4



Advantages:

- Monte Carlo integration does not require continuity for f
- Error does not depend on the dimension (in contrast to deterministic quadrature formulas), but only on the variance of the function f and the number of simulations n
  - $\rightarrow$ Improve precision through high number of simulations
  - $\rightarrow$ Improve precision by reducing variance

Disadvantages:

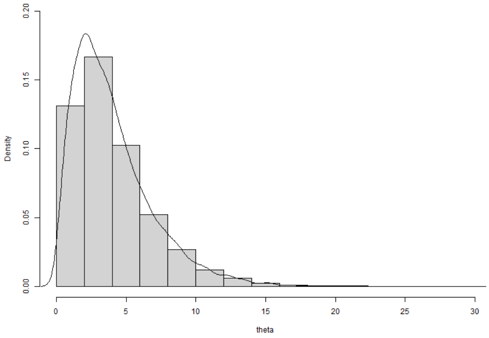
- Relatively slow convergence rates

HIT-OR-MISS / 5

```
set.seed(333)

T = 10000; shape = 2; rate = 1 / 2
theta = rgamma(T, shape = shape, rate = rate)

hist(theta, freq = FALSE, ylim = c(0, 0.2), main = "")
lines(density(theta))
```



HIT-OR-MISS / 6

```
(Etheta = mean(theta)) # MC estimator
## [1] 4.007281
(se.Etheta = sqrt(var(theta) / T)) # variance
## [1] 0.02841768
shape * 1 / rate # Theoretical expectation
## [1] 4

(Ptheta = mean(theta > 5)) # MC Estimator
## [1] 0.2863
(se.Ptheta = sqrt(var(theta > 5) / T)) # variance
## [1] 0.004520539
1 - pgamma(5, shape = shape, rate = rate) # theo value
## [1] 0.2872975
f = function(x) {dgamma(x, shape = shape, rate = rate)}
integrate(f, 5, Inf) # Numerical integration in R
## 0.2872975 with absolute error < 9.3e-05
```