Stochastik-Praktikum **WS 2023-2024**

Prof. Dr. Matthias Birkner
Jan Lukas Igelbrink

Blatt 4

Aufgabe 1 [Box-Muller-Methode] (3+3=6 Punkte)
Die Box-Muller-Methode[1] ist ein Verfahren, um aus zwei auf $[0,1]$ gleichverteilten, unabhängigen Zufallsvariablen U, V einen zweidimensionalen standardnormalverteilten Zufallsvektor $Z = (X, Y)$ zu erzeugen, wobei

$$X = \sqrt{-2\log(U)}\cos(2\pi V) \text{ und } Y = \sqrt{-2\log(U)}\sin(2\pi V). \tag{1}$$

a) Erstellen Sie eine Funktion `box_muller(n)`, die unter Verwendung der Box-Muller-Methode n unabhängige zweidimensionale standardnormalverteilte Zufallsvektoren Z erzeugt.

b) Simulieren Sie mit Hilfe der Funktion aus a) $N = 10^5$ Paare $(X_i, Y_i)_i$ unabhängiger normalverteilter Zufallsvariablen.

Erstellen Sie ein Histogramm der Werte $X_1, X_2, \ldots, X_N, Y_1, Y_2, \ldots, Y_N$. Stellen Sie die N Punktpaare $(X_i, Y_i)_i$ auch mittels `plot` wie auch schon auf Blatt 2 als Scatterplot dar.

Aufgabe 2 $(3 + 3 = 6$ Punkte)
Sei X eine Zufallsvariable, die Werte in den reellen Zahlen annimmt. Wir rekapitulieren kurz Definition 1.37 und Beobachtung 1.38 aus den Vorlesungsnotizen: Mit

$$F_X(x) := \mathbf{P}(X \leq x) \tag{2}$$

bezeichnen wir die Wahrscheinlichkeit, dass X einen Wert kleiner oder gleich $x \in \mathbb{R}$ annimmt. F_X nennen wir die Verteilungsfunktion von X. Die Funktion

$$F_X^{-1} : [0,1] \to \mathbb{R} \cup \{\infty\}, \qquad F_X^{-1}(t) := \inf\{x : F_X(x) \geq t\} \tag{3}$$

nennen wir Quantilfunktion. Diese erfüllt

$$F_X^{-1}(t) \leq x \Leftrightarrow t \leq F_X(x). \tag{4}$$

Sei nun U eine auf $[0,1]$ uniform verteilte Zufallsvariable, d.h

$$\mathbf{P}(U \in [a,b]) = b - a \qquad \forall 0 \leq a < b < 1,$$

oder auch informell geschrieben als

$$\mathbf{P}(U \in \mathrm{d}a) = \mathrm{d}a \qquad \forall\, a \in [0,1], \tag{5}$$

[1] Nach George E.P. Box und Mervin E. Muller, A Note on the Generation of Random Normal Deviates. Ann. Math. Stat. 29, 610-611, 1958.

dann besitzt die Zufallsvariable $X := F_X^{-1}(U)$ die Verteilungsfunktion F_X, denn aufgrund von (4) gilt,

$$\mathbf{P}\left(X \leq x\right) = \mathbf{P}\left(U \in \left\{t : F_X^{-1}(t) \leq x\right\}\right) = \mathbf{P}\left(U \in [0, F_X(x)]\right) = F_X(x).$$

Somit erlaubt uns die Relation (4) Zufallsvariablen mit Verteilungsfunktion F_X aus uniformverteilten Zufallsvariablen zu simulieren, dies wird auch Inversionsmethode genannt. (Siehe z.B. Beobachtung 1.38 im Skript.)

a) Verwenden Sie die Inversionsmethode um exponentialverteilte Zufallsvariablen zu simulieren. Schreiben Sie dazu zuerst eine Funktion *quantil_exp(t,λ)*, welche für $t \in [0, 1)$ and $\lambda > 0$ den Wert $F_X^{-1}(t)$ mit

$$F_X(x) = 1 - e^{-\lambda x}$$

berechnet (Wenn Sie es sich einfach machen möchten, können Sie natürlich auch die Definition von *qexp* nachschlagen und diese Funktion passend einsetzen.).

b) Simulieren Sie 1000 Zufallsvariablen für $\lambda = \frac{1}{4}$ und vergleichen Sie die daraus gewonnene empirische Verteilungsfunktion mit der theoretischen Verteilungsfunktion F_X, in dem Sie beide zusammen in einen Plot mit unterschiedlichen Farben zeichnen lassen. Verwenden Sie hierbei als x-Achse die Werte von 0 bis 10 mit Schrittlänge 10^{-1}. Ergänzen Sie den Plot um eine passende Beschriftung der Achsen, eine Legende und einen Titel. Verwenden Sie für die Plots *type* = " *l* ".

Aufgabe 3 [Poisson-Prozesse](3+3=6 Punkte)
Es sei $X \sim \text{Exp}(1)$-verteilt und die $X_1, X_2, \ldots$ unabhängige Kopien von X. Wir setzen

$$N(t) := \max\left\{k : \sum_{i=1}^{k} X_i < t\right\}. \tag{6}$$

a) Simulieren Sie sowohl für $t = 1$ als auch $t = 3$ die Zufallsvariable $N(t)$ jeweils 10^5 mal und stellen Sie die empirischen Verteilungsfunktionen jeweils in einem Plot der Verteilungsfunktion der Pois(t)-Verteilung gegenüber.

b) Nutzen Sie den untenstehenden `R`-Code, der beispielhaft zeigt, wie man Code parallelisiert ausführen kann, um Aufgabenteil **a)** mit 10^7 Simulationen durchzuführen.

B) Diese **B**onusaufgabe kann bis zu drei fehlende Punkte auf diesem Blatt ausgleichen (die maximale Punktzahl ändert sich nicht): Wenn Sie mögen, nutzen Sie das Paket `tictoc` (siehe bspw. `https://cran.r-project.org/web/packages/tictoc/index.html`), um die Laufzeit des Codes aus **a)** und **b)** für $1, 2, 3, \ldots, 10, 10^2, 10^3, 10^4, \ldots, 10^5$ viele Simulationen zu vergleichen. Veranschaulichen Sie dies in einem Plot, den Sie als `png` exportieren und an Ihre Abgabe anhängen.

```
library(parallel)
```

```
nsim <- 10
cores <- 4
mySeed <- 422283782

myFunction <- function() {
  return("Test")
}

#create cluster
parReplicate <- function(cl, n, expr, simplify=TRUE, USE.NAMES=TRUE) {
  parSapply(cl, integer(n), function(i, ex) eval(ex, envir=.GlobalEnv),
            substitute(expr), simplify=simplify, USE.NAMES=USE.NAMES)
}


cl <- makePSOCKcluster(cores)
clusterSetRNGStream(cl, iseed = mySeed)
clusterExport(cl=cl, varlist=c("myFunction"), envir=environment())
result <- parReplicate(cl, nsim, myFunction())
stopCluster(cl)

print(result)
```