Introduction to Simulation with Variance Estimation

Exercise 3

Bosse Behrens, st.id 12347333

2024W

Task 1

part 1

We simply follow the example code form the lecture with using the specified function and uniform distribution first with the Monte Carlo integration method and the using integrate() from R.

```
set.seed(12347333)
usample <- runif(100000, 1, 6)

integ <- mean(exp(-(usample)^3)*(6-1))
x <- integrate(f = function(x){exp(-x^3)}, lower = 1, upper = 6)

results <- data.frame(
  type = c("Monte Carlo integral method", "integrate function", "absolute difference"),
  value = c(integ, as.numeric(x[1]), abs(as.numeric(x[1]) - integ))
)

print(results)
```

```
##                          type       value
## 1 Monte Carlo integral method 0.084633806
## 2          integrate function 0.085468329
## 3         absolute difference 0.000834523
```

part 2

The distribution we will get samples from for the Monte Carlo integral method needs to have the same domain as the integrand $f(x)$. Therefore using an uniform distribution is not possible anymore since the domain is now $[1, \infty]$. A distribution that has that same domain is the pareto distribution with location parameter $x_0 = 1$. Additionally we choose $\alpha = 1$ as the shape parameter so we get the simple pdf x^{-2} of the $Pa(1, 1)$ distribution. This distribution is not part of base R, so we load the VGAM package to get pareto-distributed samples. After this we follow the same steps as in part 1. Then we again use the Monte Carlo method and R's integrate.

```
#install.packages("VGAM")
library(VGAM)
```

```
## Lade nötiges Paket: stats4
```

```
## Lade nötiges Paket: splines
```

```r
set.seed(12347333)
psample <- rpareto(100000, 1, 1)

integ2 <- mean(exp(-(psample)^3)/(psample^(-2)))
y <- integrate(f = function(x){exp(-x^3)}, lower = 1, upper = Inf)

results2 <- data.frame(
  type = c("Monte Carlo integral method", "integrate function", "absolute difference"),
  value = c(integ2, as.numeric(y[1]), abs(as.numeric(y[1]) - integ2))
)

print(results2)
```

```
##                          type        value
## 1 Monte Carlo integral method 0.0856060462
## 2          integrate function 0.0854683294
## 3         absolute difference 0.0001377167
```

part 3

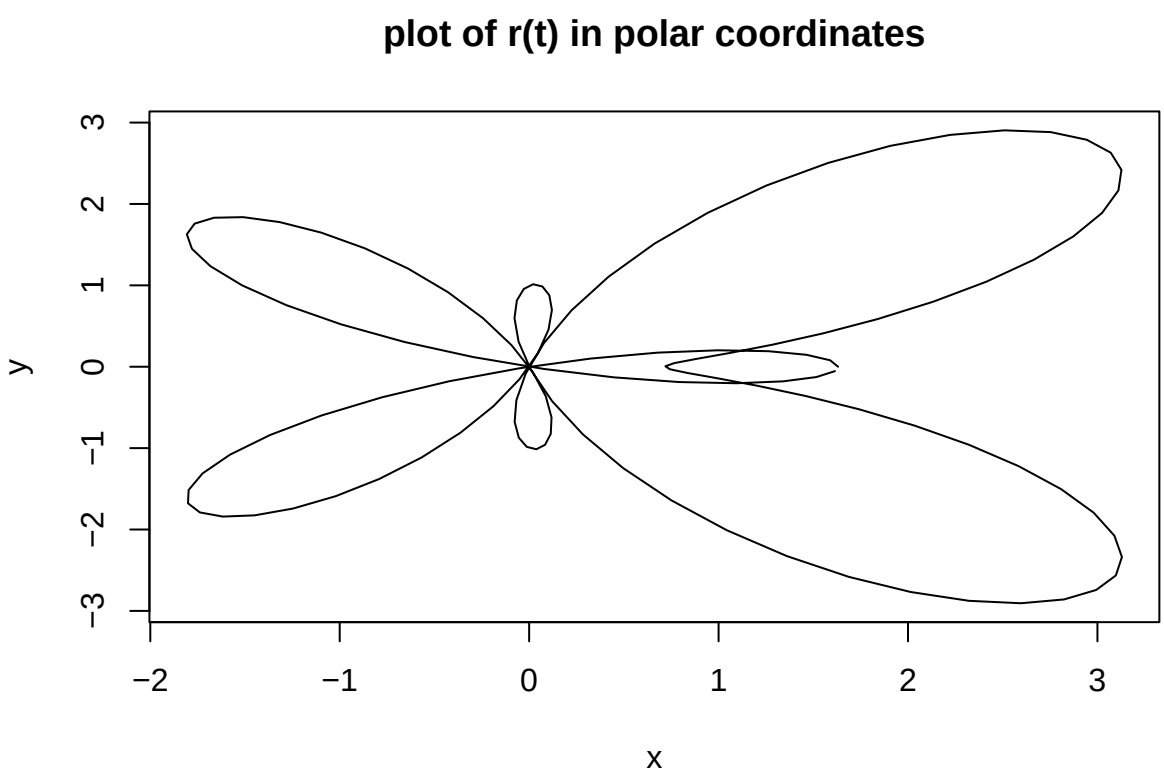
The absolute difference between the Monte Carlo integral and integrate function is less in part 2 than 1. This might be due to the sample distributions used. While the uniform distribution is of course easy to sample and implement in this case the integrand $f(x) = e^{-x^3}$ behaves very different. It starts at 1 with higher values and rapidly decreases, quickly having function values near zero. The pdf of of the $Pa(1, 1)$ distribution, which is x^{-2} behaves more similar to f. It also starts higher at $x = 1$ and then quickly decreases with asymptotic behavior of values converging to zero. While it has a heavier right tail than $f(x)$ it is still far more similar than the uniform distribution. This is called Importance sampling which means points are sampled more frequently where the integrand is large, instead of just random sampling as in part 1. To keep in mind, there might also be further reasons for the differences in part 1 and 2 that were not clear in this example.

Task 2

part 1

We write the function and transform x and y into polar coordinates to be able to plot the function.

```r
r <- function(t){
  exp(cos(t))-2*cos(4*t)-sin(t/12)^5
}
t <- seq(-pi, pi, 0.05)
x <- r(t)*cos(t)
y <- r(t)*sin(t)
plot(x,y, type = "l", lwd = "1", main = "plot of r(t) in polar coordinates")
```



part 2

We implement the function that checks if a point is inside the function area or outside of it. First we have to transform the point into polar coordinates: The radius $r = \sqrt{x^2 + y^2}$ and the angle φ for which we have to implement the different cases since *arctan* is not continous on $(-\pi, \pi)$. Then we need to get the previous functions value at angle φ to get the curves radius, as well as the one shifted by pi, since we also need the curve of the function at $\varphi + \pi$ that is shifted by 180 degrees to check on the opposite side for different behavior. In the next if-functions we have to check first at positive angles of $r(\varphi)$ if a points radius is less than the curves absolute radius value. Then we also need to check on the opposite side that is shifted by φ if at a negative angle of the curve at $\varphi + \pi$ the points radius is again less than the curves absolute radius at that value. In both cases TRUE will be returned else FALSE.

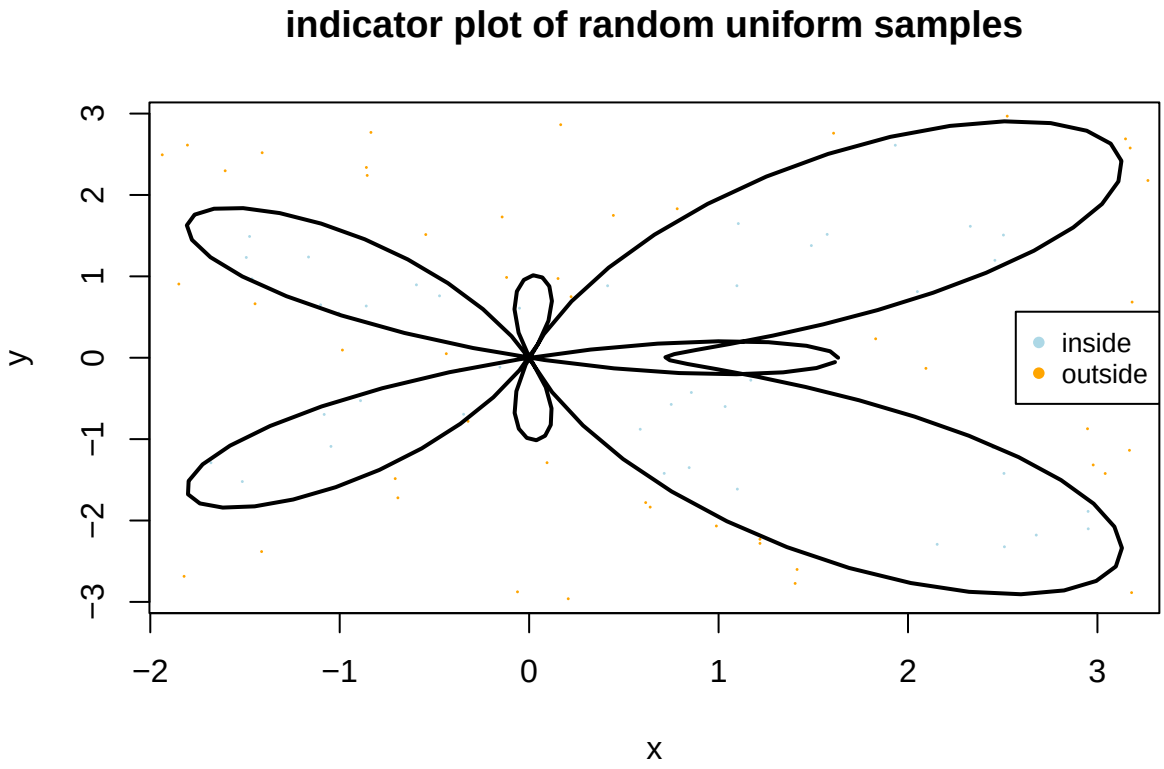
```
inside_test <- function(x, y){
  r_value <- sqrt(x^2 + y^2) #calculate radius
  if (x > 0){phi <- atan(y/x)}  #calculate phi for dif. cases
  if (x < 0){
    if (y >= 0){phi <- atan(y/x)+pi}
    if (y < 0){phi <- atan(y/x)-pi}
  }
  if (x == 0 & y > 0){phi <- pi/2}
  if (x == 0 & y > 0){phi <- -pi/2}

  r_func_value <- r(phi)   #radius of the curve of our func
  r_func_pi <- r(phi + pi) #shifted by pi for the 180 degree shift
```

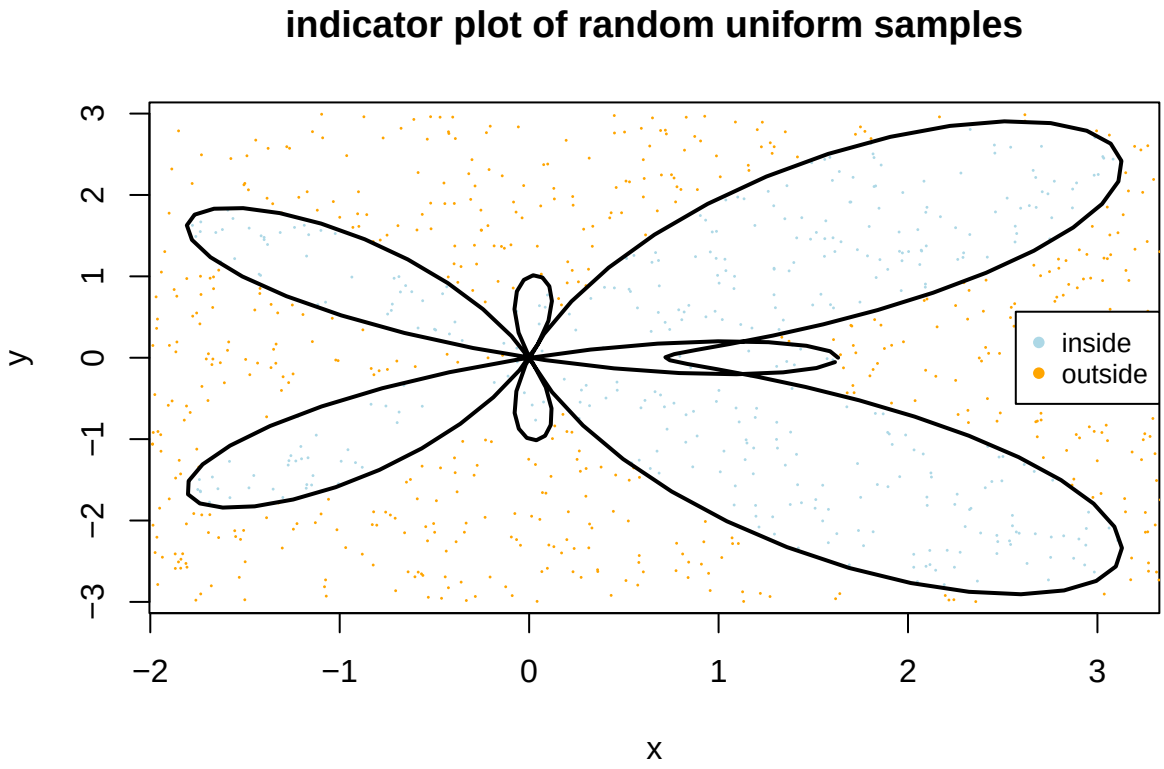
```
  if (r_value < abs(r_func_value) & r_func_value > 0){
    return(TRUE)
  }
  if (r_value < abs(r_func_pi) & r_func_pi < 0){
    return(TRUE)
  }
  else {
    return(FALSE)
  }
}
```

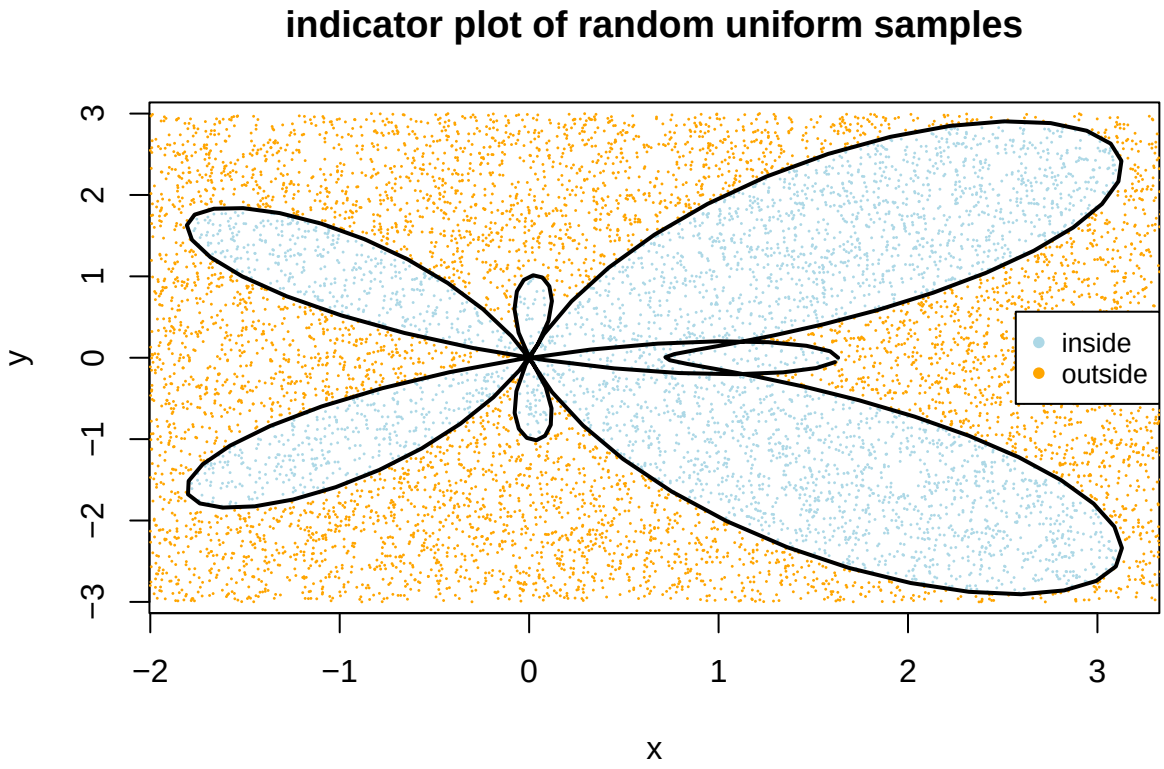
part 3

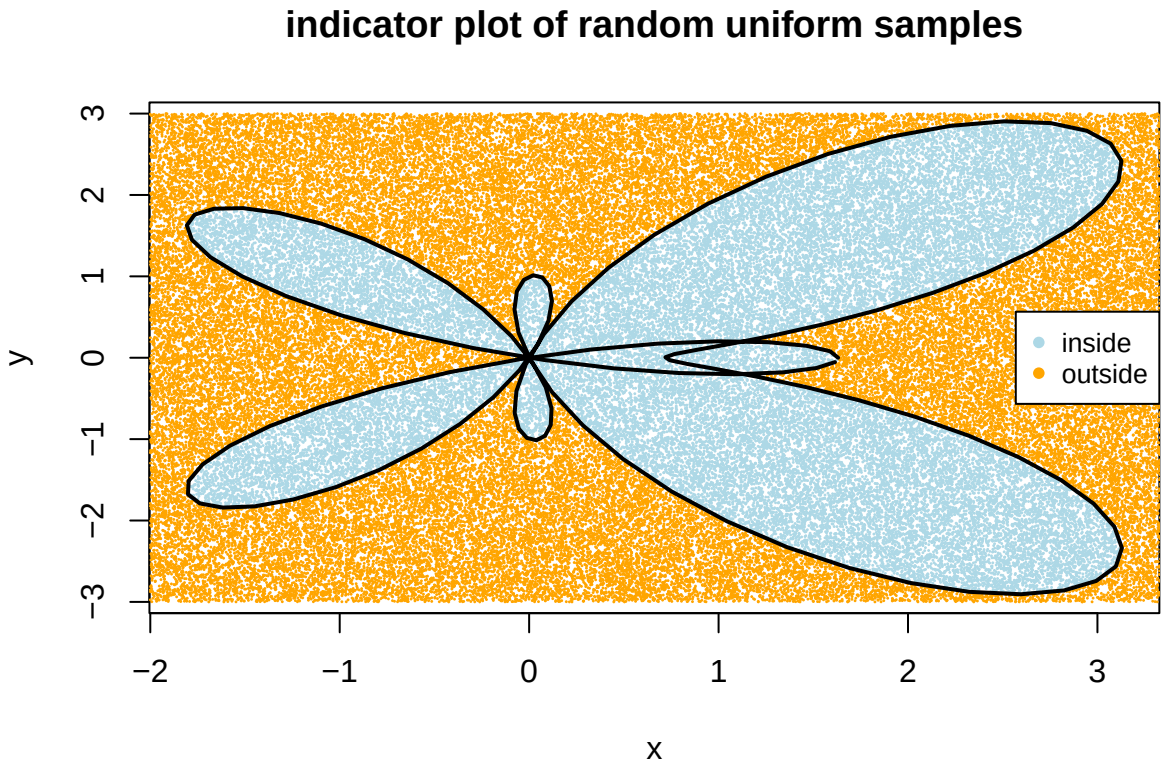
Now we generate the specified uniform samples of $100, 1000, 10000$ and 100000 samples in $[-2, 3.5]x[-3, 3]$ respectively and apply our implemented test to check if a coordinate is inside the curves' limit for those samples. The ratios of $\frac{number\ of\ points\ inside}{total\ points}$ is calculated and multiplied by the sample rectangles area to get an estimation of the integral. Furthermore the plots containing the function curve and the random points (distinguished by inside/outside) are plotted.

```
set.seed(12347333)
ratios <- c()
for (i in 1:4){

  x_1 <- runif(10^(1+i), -2, 3.5)
  y_1 <- runif(10^(1+i), -3, 3)
  inside <- mapply(inside_test, x_1, y_1)

  #calculating the ratios, adjusting for area of rectangle and o. of samples
  ratios <- c(ratios, sum(inside == TRUE)*(5.5*6)/(10^(1+i)))

  plot(x,y, type = "n", main = "indicator plot of random uniform samples")
  points(x_1[inside], y_1[inside], col = "lightblue", pch = 16, cex = 0.2)
  points(x_1[!inside], y_1[!inside], col = "orange", pch = 16, cex = 0.2)
  lines(x, y, lwd = 2)
  legend("right", legend = c("inside", "outside"),
         col = c("lightblue", "orange"), pch = 16, cex = 0.8, bg = "white")
}
```


indicator plot of random uniform samples
inside
outside
y
x
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


indicator plot of random uniform samples
inside
outside
y
x
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


indicator plot of random uniform samples
inside
outside
y
x
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3



Now we are interested in the estimations of the integral for the different sample sizes:

```
names(ratios) <- c("sample size: 100", "sample size: 1000",
                   "sample size: 10000", "sample size: 100000")
print(ratios)
```

```
##    sample size: 100   sample size: 1000  sample size: 10000 sample size: 100000
##            14.85000            13.33200            13.34520            13.18944
```

Considering these values we can make the estimation that the exact integral (area of the function) is somewhere in $[13.1, 13.4]$.

part 4

Monte Carlo simulation means first generating a large amount of samples from a distribution to use as inputs for a model, in this case a function in the 2-dimensional space. By simulating the outcome many times using the random samples a distribution of possible outcomes is built, in this example we gain the amount of randomly sampled points that are inside the curve of the function. Since the inputs are randomly distributed in this case we can make an estimate of the area/integral by getting the ratio of the points inside to the ones outside relative to the specified experiment area. This gets more accurate the larger the samples are but of course also more computerintensive. When having the results of the simulations, they can be analyzed statistically to gain information and calculate statistic values such as means or probabilities of outcomes. The Monte Carlo method in general is therefore a valid choice when trying to solve problems that cannot be solved in analytical or explicit form and need to be calculated numerical or using statistical methods just like in this task.