Monte Carlo Simulation of areas

Kirill Medovshchikov

2022-11-04

Contents

1 Setup

In this section the necessary library is attached and the seed is set to the student number.

```
library(ggplot2)
library(tidyr)

set.seed(12144024)
```

2 Computation of integrals with the Monte Carlo integration

2.1 For b = 6

Here the uniformly distributed random variables are created to create an integral for b which is equal to 6.

```
a <- 1
b <- 6
n <- 1000000

unif_dist <- runif(n, a, b)
```

Then the function for calculating the exponent of the integrated value is created.

```
exponential <- function(n){
  return(exp(-n^3))
}
```

And the exponential for the random variables is calculated.

```
expo <- exponential(unif_dist)
```

Here the function for calculating the monte carlo integration is implemented.

```
monte_carlo_integration <- function(n){
  return(mean(expo) * (b - a))
}
```

And the result of the integration is displayed.

```
mci_my <- monte_carlo_integration(unif_dist)
print(mci_my)
```

```
## [1] 0.08533867
```

As well as the result of integration performed by the R's native function.

```
mci_native <- integrate(exponential, a, b)
print(mci_native)
```

```
## 0.08546833 with absolute error < 3.2e-07
```

The two results differ very lightly, with the difference being just in mere thousands to the right from the decimal point.

2.2 For b = *infinity*

Here the new b variable equal to infinity is defined.

```
b_inf = Inf
```

After that a new monte carlo integration function is created for the integration of an infinite b variable.

```
monte_carlo_integration_inf <- function(n){
  expo_inf <- rexp(n)
  return(mean(exponential(expo_inf + 1) / dexp(expo_inf)))
}
```

Whose output can be found below.

```
mci_my_inf <- monte_carlo_integration_inf(n)
print(mci_my_inf)
```

```
## [1] 0.08545798
```

And that is also very close to the native R's integrate function's output. The tryCatch condition is used due to the previously occurring error in the exponent calculation, that had been resolved by now. In any case it had been decided to leave the construct be due to the inpredictability of the program's behavior due to working with infinite number in this particular case.

```
tryCatch({
  mci_native_inf <- integrate(exponential, a, b_inf)
  print(mci_native_inf)
},
error = function(error_message){
  print(error_message)
})
```

```
## 0.08546833 with absolute error < 6.2e-06
```

2.3 Why Monte Carlo integration agrees in 2. with integrate but not so much in 1.?

While the difference in between these two cases is not very large, it is still present in both of the tests.

```
mci_native_clean <- extract_numeric(mci_native)[1]
```

```
## extract_numeric() is deprecated: please use readr::parse_number() instead

## Warning in extract_numeric(mci_native): NAs introduced by coercion
```

```
mci_native_inf_clean <- extract_numeric(mci_native_inf)[1]
```

```
## extract_numeric() is deprecated: please use readr::parse_number() instead

## Warning in extract_numeric(mci_native_inf): NAs introduced by coercion
```

```
mci_my
```

```
## [1] 0.08533867
```

```
mci_native_clean
```

```
## [1] 0.08546833
```

```
mci_my_inf
```

```
## [1] 0.08545798
```

```
mci_native_inf_clean
```

```
## [1] 0.08546833
```

As it can be seen from the code chunks above, contrary to the question, the difference is indeed in favor of the second experiment. This could be due to the fact that the infinite b variable allows for more samples to be processed and therefore have a more precise estimation.

3 Monte Carlo enclosure by the graph

3.1 Visualise the function and the area

Here the function is used to calculate the r(t).

```
rt_func <- function(t){
  exp(cos(t)) - 2 * cos(4 * t) - sin(t / 12)^5
}
```

Then the sequence from minus pi to the positive pi is initialized.

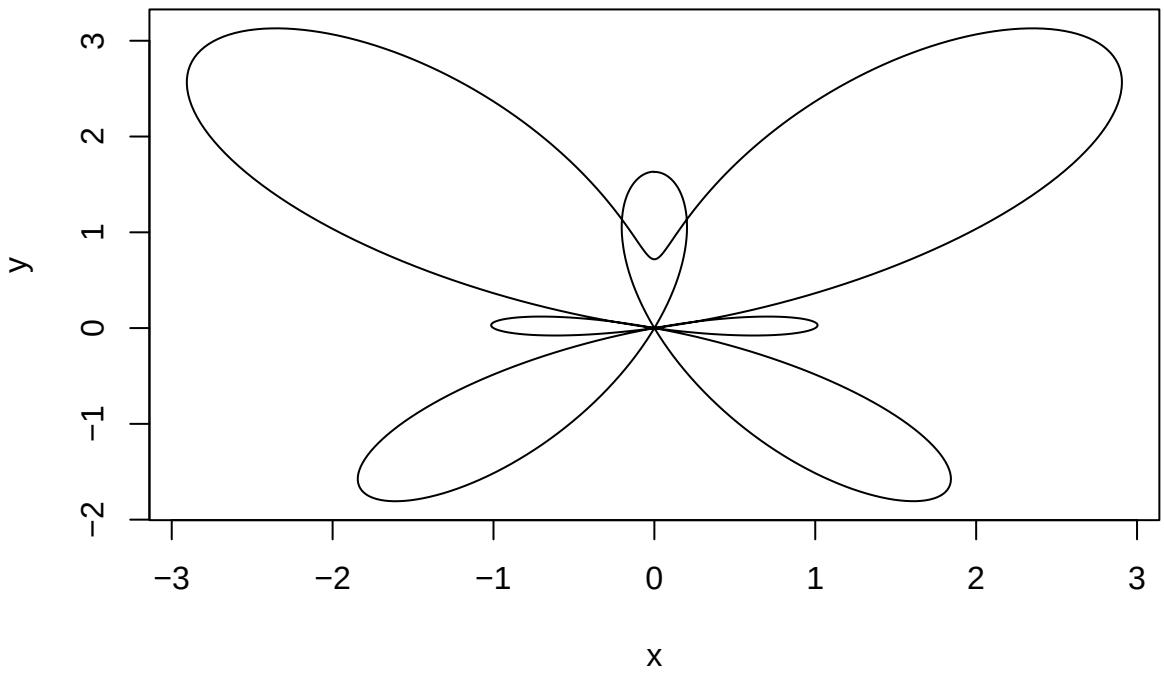
```
t <- seq(by=0.01,from = -pi, to = pi)
```

Then the function's output is utilized in the multiplication with the sin of t and cos of t for x and y accordingly.

```
x <- rt_func(t) * sin(t)
y <- rt_func(t) * cos(t)
```

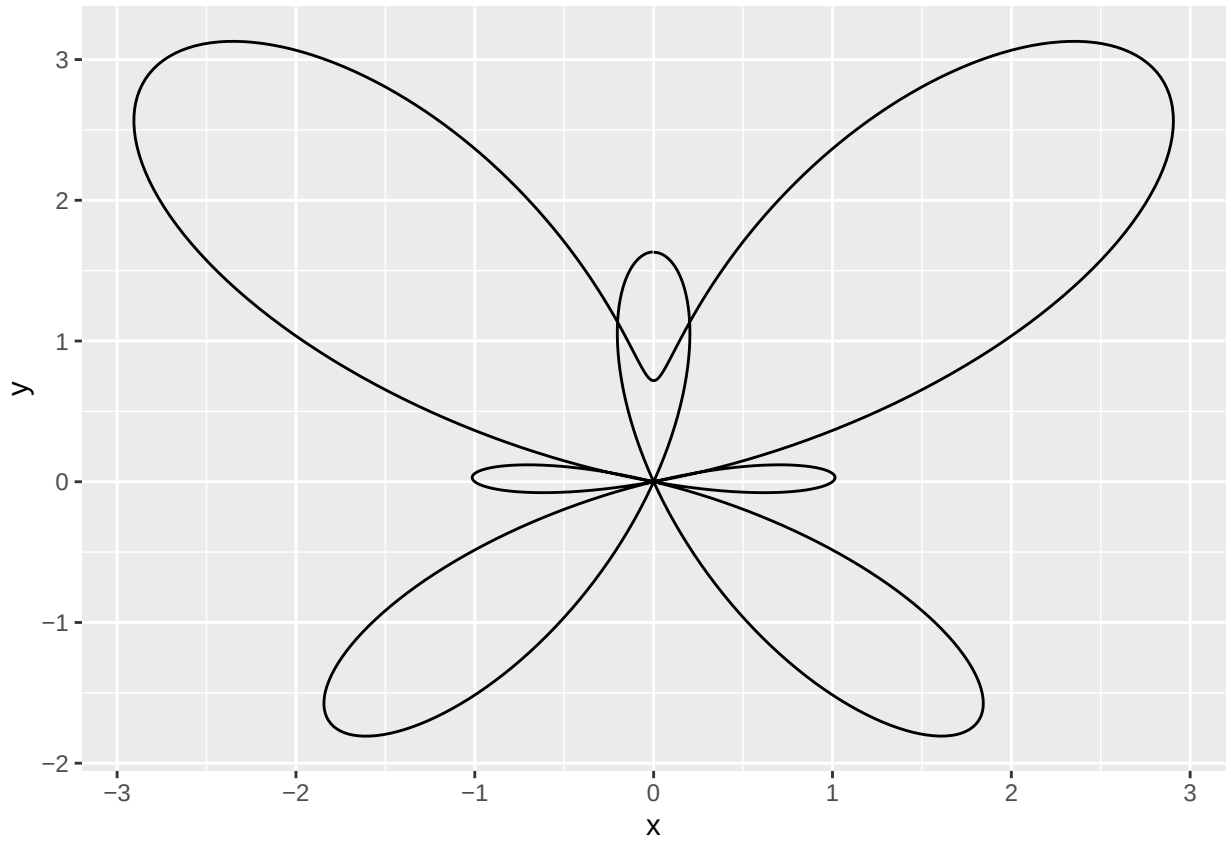
When the computed x and y values are plotted, the shape similar to a butterfly can be observed.

```
plot(x, y, 'l')
```



The same here, although now with the use of the ggplot() library.

```
ggplot() + geom_path(aes(x=x, y=y))
```


3
2
1
0
−1
−2
y
−3
−2
−1
0
1
2
3
x