

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

Symulacje Komputerowe

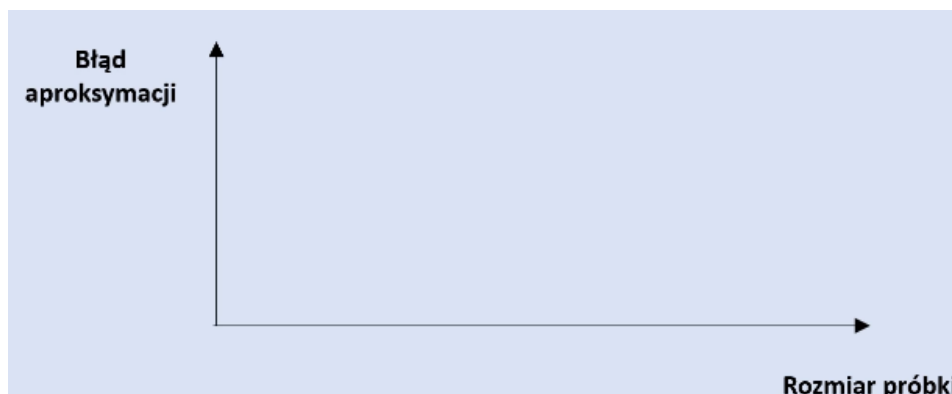
Sprawozdanie “Symulacja Monte Carlo”

Adam Talarczyk, Mateusz Wrzoł

Uniwersytet Śląski, Sosnowiec, 2021

1 Zadanie 1

Należy zmodyfikować kod dla aproksymacji stałej π , aby sprawdzić jak rozmiar próbki wpływa na błąd aproksymacji. Błąd aproksymacji obliczamy jako wartość bezwzględną różnicy, pomiędzy aproksymacją π i wartością rzeczywistą π (3.14159265). Należy przygotować wykres [Rysunek 1].



Rysunek 1: Przykład wykresu

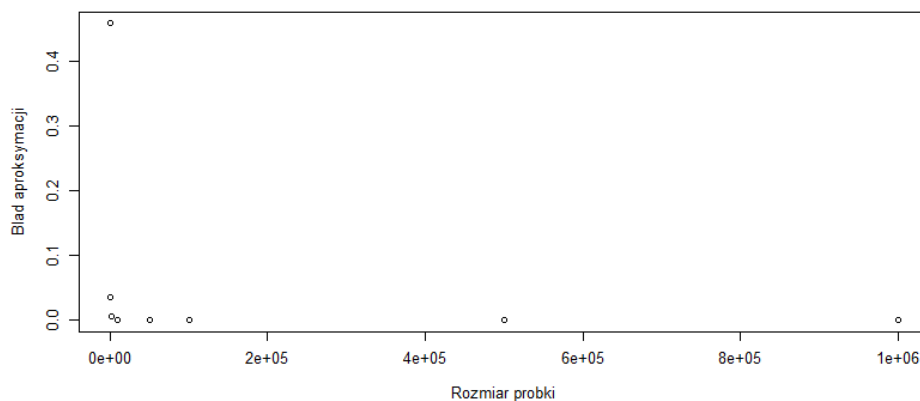
1.1 Rozwiązanie

Do uzyskania odpowiednich wyników konieczne było wyliczenie aproksymacji π dla wielu próbek. W zadaniu wykorzystano próbki z zakresu od 1 do 1000000. Dodatkowo, konieczne było kilkukrotne powtórzenie wykonywanej aproksymacji w celu uśrednienia otrzymanego przybliżenia π . W opisywanym przykładzie zastosowano 10 powtórzeń. Błąd aproksymacji został wyliczony jako różnica bezwzględna pomiędzy zdefiniowaną stałą 3,14159265 oraz uzyskanym uśrednionym przybliżeniem. Wyniki zaprezentowane są w tablicy 1.

Próbka	Średnie przybliżenie π	Błąd aproksymacji
1	3,6	0,45840735
10	3,04	0,10159265
100	3,064	0,07759265
1000	3,124	0,01759265
10000	3,13392	0,00767265
100000	3,140084	0,00150865
1000000	3,1409996	0,00059305

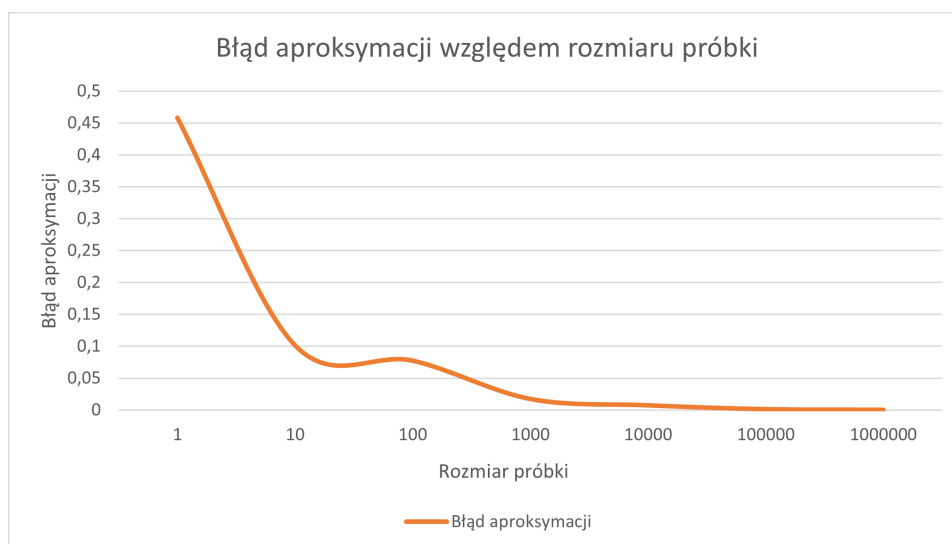
Tablica 1: Tablica z wynikami uzyskanymi w symulacji

Na podstawie powyższych danych wygenerowany został wykres za pomocą funkcji `plot()` (Rysunek 2).

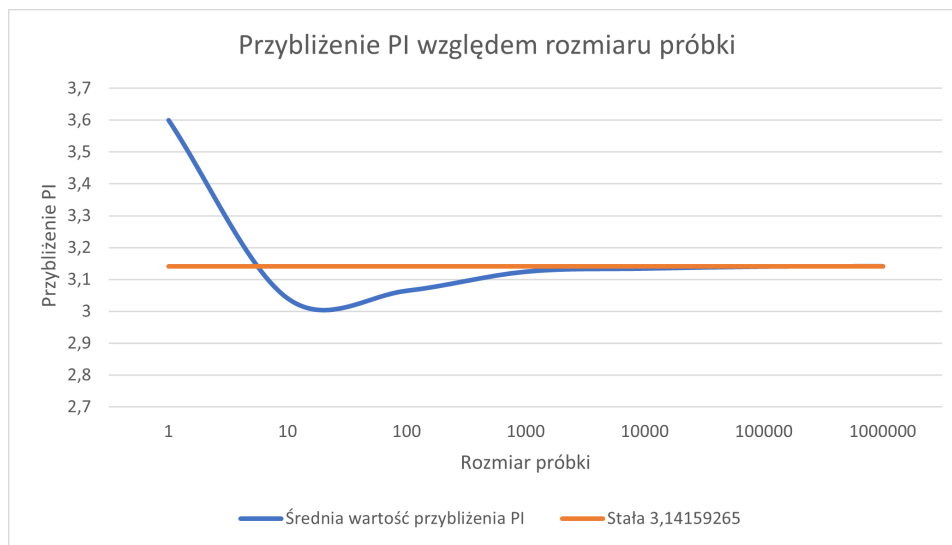


Rysunek 2: Wygenerowany wykres przedstawiający błąd aproksymacji względem rozmiaru próbki

Przedstawiony na rysunku 2 wykres jest mało czytelny, dlatego wyeksportowano dane do pliku o rozszerzeniu `xlsx` i na ich podstawie utworzono kolejne wykresy: Błąd aproksymacji względem rozmiaru próbki (Rysunek 3) oraz przybliżenie PI względem rozmiaru próbki z zaznaczoną wartością 3,14159265 (Rysunek 4).



Rysunek 3: Wykres przedstawiający błąd aproksymacji względem rozmiaru próbki



Rysunek 4: Wykres przedstawiający przybliżenie PI względem rozmiaru próbki, w porównaniu do wartości 3,14159265

Na podstawie przedstawionej symulacji można stwierdzić, że wielkość błędu aproksymacji zależy od wielkości próbki. Im większa próbka, tym mniejszy jest błąd aproksymacji, a przybliżenie liczby π jest bardziej dokładne.

1.2 Kod źródłowy

```

1 # Title      : Monte Carlo Simulation
2 # Objective  :
3 # Created by: Adam Talarczyk, Mateusz Wrzol
4 # Created on: 16.04.2021
5
6 library("xlsx")
7 source('pi/avarage_difference.R')
8
9 calculate <- function(steps, sequences) {
10   diff.vector <- NULL
11   dataset <- data.frame()
12
13   for (runs in steps)
14   {
15     diff <- avarage_difference(runs, sequences)
16     dataset <- rbind(dataset, diff)
17     diff.vector <- append(diff.vector, diff[1, 'diff'])
18   }
19
20   export_dataset(dataset)

```

```

21 draw_plot(steps, diff.vector)
22 }
23
24 export_dataset <- function(dataset){
25   write.xlsx(dataset, file = "pi/export/data.xlsx",
26             sheetName="PI")
27 }
28 draw_plot <- function(steps, results) {
29   plot(steps, results, xlab = 'Rozmiar próbki', ylab = 'Błąd
    aproksymacji', col = 'black')
30 }
31
32 sequence <- c(1,10,100,1000,10000,100000,1000000)
33 calculate(sequence, 10)

```

Listing 1: Plik main.R- wywołanie głównej funkcji

```

1 # Title      : Avarage absolute difference
2 # Objective  :
3 # Created by: Adam Talarczyk, Mateusz Wrzol
4 # Created on: 10.04.2021
5
6 source('pi/approximation.R')
7
8 avarage_difference <- function(runs, sequences = 100) {
9   pi.vector <- NULL
10  for (i in seq(1, sequences, by = 1))
11  {
12    mc.pi <- approximation(runs)
13    pi.vector <- append(pi.vector, mc.pi)
14  }
15
16  avarage_pi <- mean(pi.vector, trim = 0, na.rm = FALSE)
17  difference <- abs(3.14159265 - avarage_pi)
18
19  dataset <- data.frame(
20    step = runs,
21    pi_value = pi.vector,
22    avg_pi = avarage_pi,
23    diff = difference
24  )
25 }

```

Listing 2: Plik avarage_difference.R- odpowiedzialny za wyliczanie błędu aproksymacji PI

```

1 # Title      : PI approximation
2 # Objective  :
3 # Created by: Adam Talarczyk, Mateusz Wrzol

```

```

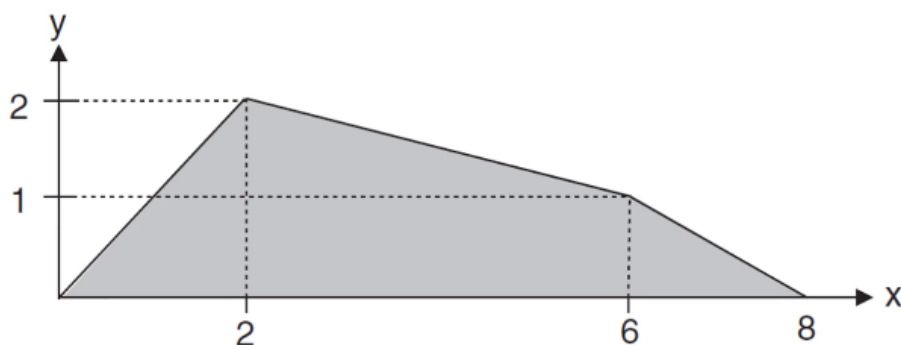
4 # Created on: 10.04.2021
5
6 approximation <- function(runs = 1000) {
7   xs <- runif(runs, min = -0.5, max = 0.5)
8   ys <- runif(runs, min = -0.5, max = 0.5)
9   in.circle <- xs^2 + ys^2 <= 0.5^2
10  mc.pi <- (sum(in.circle) / runs) * 4
11 }

```

Listing 3: Plik approximation.R- odpowiedzialny za obliczanie przybliżenia liczby π

2 Zadanie 2

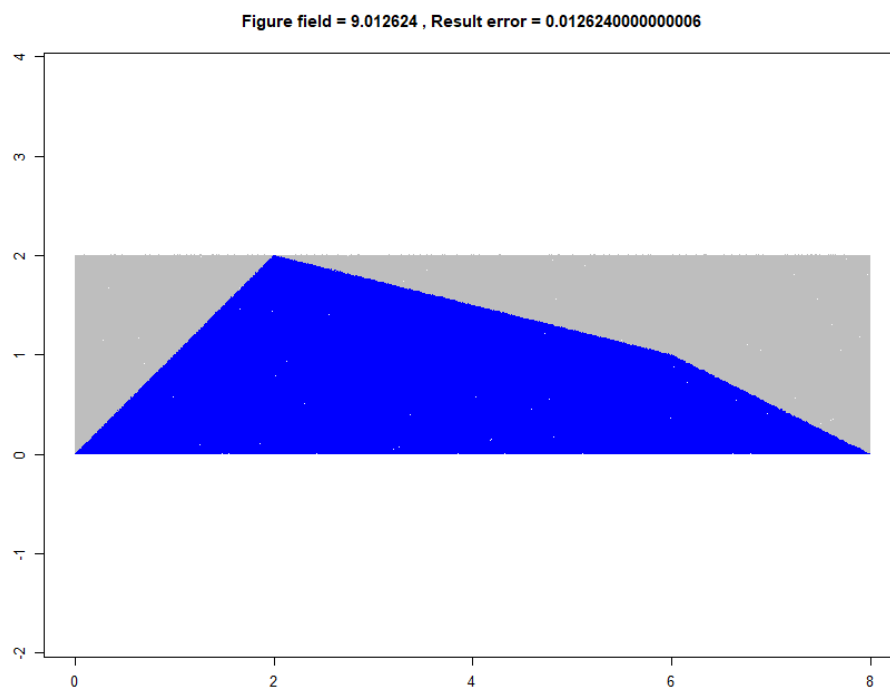
Zaprogramować symulację Monte Carlo (np. w języku R), która pozwoli obliczyć pole powierzchni szarego obszaru, przedstawionego na poniższym rysunku [Rysunek 5]. Obliczyć błąd uzyskanego wyniku.



Rysunek 5: Figura

2.1 Rozwiązanie

W celu uzyskania odpowiedniego wyniku, w pierwszej kolejności generowane są punkty dla współrzędnych x (od 0 do 8) i y (od 0 do 2). Następnie sprawdzane jest, czy punkty znajdują się w przestrzeni odpowiedniej figury (z podziałem na mniejsze figury, takie jak prawy trójkąt, lewy trójkąt, środkowy trójkąt oraz środkowy prostokąt). Kolejnym krokiem jest połączenie wyników dla poszczególnych figur w jeden wielokąt z treści zadania. Dokładne pole figury zostało wyliczone jako suma poszczególnych elementów wielokąta, zastosowano działanie $(2 \cdot 2 / 2) + (1 \cdot 4 / 2) + (1 \cdot 2 / 2) + (1 \cdot 4)$, którego wynik wynosi 9. Wynik symulacji to iloraz sumy punktów w figurze i liczby wygenerowanych punktów, pomnożony przez pole badanego obszaru ($2 \cdot 8 = 16$). Błąd uzyskanego wyniku został wyliczony jako wartość bezwzględna z różnicy uzyskanego wyniku i dokładnego pola figury. Na sam koniec wygenerowany został wykres przy użyciu funkcji `plot()` (Rysunek 6).



Rysunek 6: Figura utworzona przy użyciu funkcji `plot()` dla próbki równej 1000000

Wyniki dla poszczególnych próbek zamieszczone są w tablicy 2. Dodatkowo, wygenerowane wykresy przedstawione są na rysunku 7 w formie porównania względem wielkości próbki.

Próbka	Pole figury	Błąd
100000	9.0352	0.0351999999999997
1000000	9.012624	0.0126240000000006

Tablica 2: Tablica z wynikami uzyskanymi w symulacji


```

29 mc.error <- abs(mc_exact_figure_field - mc_figure_field)
30 plot(xs, ys,
31       col = ifelse(in.all_figures, "blue", "grey"),
32       xlab = '', ylab = '', asp = 1, pch = '.',
33       main = paste("Figure field =", mc_figure_field, ",
34                     Result error =", mc.error))
35 }
36 figure(100000)

```

Listing 4: Plik figure.R- odpowiedzialny za obliczenie pola i wygenerowanie wykresu