Mohamed BOUDIAF University – M'Sila
Faculty of Mathematics and Computer Science
Department of Mathematics
3rd Year Bachelor's in Applied Mathematics 2025–2026

Simulation and Software Practice
Monte Carlo method

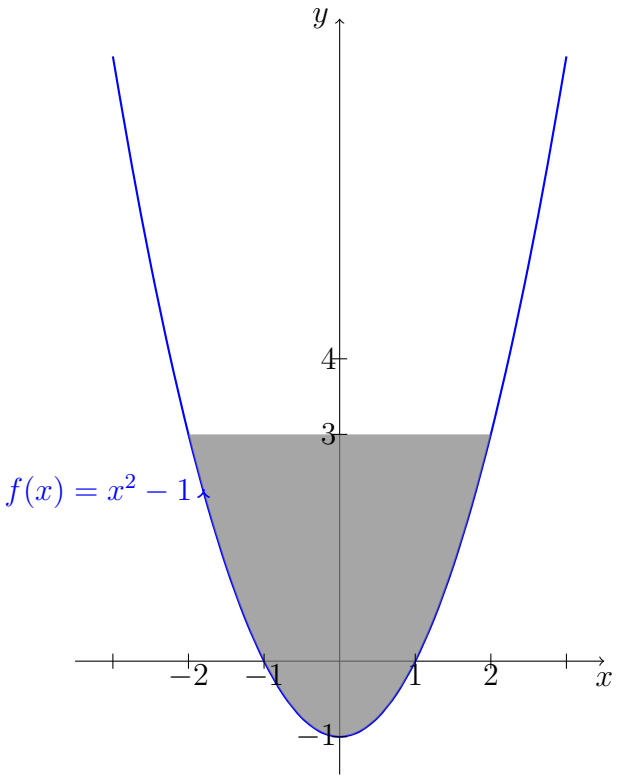
Exercise 1 :

(Use the Monte Carlo method)

1. Write a program in R that estimates the value of the following integral:

$$I = \int_{-2}^{2} f(x)\,dx.$$

2. Write a program in R to estimate the shaded area shown in the figure below and plot a graph showing the corresponding curve and the points generated for the estimation.



Exercise 2 :

(Use R)

1. Use the Monte Carlo method to calculate the area of a circle of radius r=2 and plot a graph showing the curve of the circle and the points that were created for estimation.
2. Write a function that estimates the area of a circle with radius r using the Monte Carlo method, with $n = 1000$ points by default.

Solution Ex 1

```
  integralsquare<- function(N) {
# Initialize counters for each region
n1 <- 0
n2 <- 0
n3 <- 0

# Monte Carlo simulation
for (i in 1:N) {
  # Region 1: x in [-2, -1], y in [0, 3]
  xi <- runif(1, min = -2, max = -1)
  yi <- runif(1, min = 0, max = 3)
  if (yi <= (xi^2 - 1)) {
    n1 <- n1 + 1
  }

  # Region 2: x in [-1, 1], y in [-1, 0]
  xi <- runif(1, min = -1, max = 1)
  yi <- runif(1, min = -1, max = 0)
  if (yi >= (xi^2 - 1)) {
    n2 <- n2 + 1
  }

  # Region 3: x in [1, 2], y in [0, 3]
  xi <- runif(1, min = 1, max = 2)
  yi <- runif(1, min = 0, max = 3)
  if (yi <= (xi^2 - 1)) {
    n3 <- n3 + 1
  }
}

# Compute the integrals for each region
I1 <- 3 * (n1 / N) # Region 1 area
I2 <- 2 * (n2 / N) # Region 2 area
I3 <- 3 * (n3 / N) # Region 3 area

# Combine the results
I <- I1 - I2 + I3

# Return the estimated integral
return(I)
}
```

```
# Example usage:
N <- 10000  # Number of random samples
result <- integralsquare(N)
cat("Estimated integral I:", result, "\n")
____________________Surface_____________________
S <- function(N) {
  n <- 0  # Counter for points above the curve

  # Vectors to store the coordinates of the random points
  x_points <- numeric(N)
  y_points <- numeric(N)

  for (i in 1:N) {
    xi <- runif(1, min = -2, max = 2)  # Random x in [-2, 2]
    yi <- runif(1, min = -1, max = 3)  # Random y in [-1, 3]
    x_points[i] <- xi
    y_points[i] <- yi

    # Check if the point is above or on the curve
    if (yi >= (xi^2 - 1)) {
      n <- n + 1
    }
  }

  # Calculate the estimated surface
  S <- 16 * n / N  # Area of the rectangle [-2, 2] x [-1, 3] is 16

  # Plotting
  plot(x_points, y_points, pch = 16, col = ifelse(y_points >= (x_points^2 - 1), "green",
       "blue"), cex = 0.6, xlab = "x", ylab = "y", main = "Monte Carlo estimation of the
       area", xlim = c(-2, 2), ylim = c(-1, 3))
  curve(x^2 - 1, from = -2, to = 2, col = "red", lwd = 2, add = TRUE)
  legend("topright", legend = c("Points above the curve", "Points off the curve",
         "Curve f(x)"), col = c("green", "blue", "red"), pch = c(16, 16, NA),
         lty = c(NA, NA, 1), bty = "n")

  return(S)
}

# Example usage
N <- 10000  # Number of random points
result <- S(N)
```

```
cat("Estimated surface:", result, "\n")
```

Solution of EX 2 First question

```
    n <- 500  # Number of points
r <- 2    # Radius

# Start timing
start_time <- Sys.time()

# Generate random points
x <- runif(n, -r, r)  # Random points in range [-r, r]
y <- runif(n, -r, r)  # Random points in range [-r, r]

# Check if points are inside the circle
inside <- (x^2 + y^2 <= r^2)  # Logical vector for points inside

# Estimate the area of the circle
area_estimated <- sum(inside) / n * (2 * r)^2  # Monte Carlo area estimate

# Real area of the circle
area_real <- pi * r^2

# End timing
end_time <- Sys.time()
time_taken <- end_time - start_time

# Print results
cat(sprintf("Estimated Circle Area: %.4f\n", area_estimated))
cat(sprintf("Real Circle Area: %.4f\n", area_real))
cat(sprintf("Time Taken: %.4f seconds\n", as.numeric(time_taken)))

# Plotting
plot(
  x, y, col = ifelse(inside, "green", "red"), pch = 16,
  xlab = "x", ylab = "y", main = "Monte Carlo Simulation: Circle within Square"
)
symbols(0, 0, circles = r, add = TRUE, inches = FALSE, lwd = 2, col = "blue")  # Circle bo
legend(
  "topright", legend = c("Points inside", "Points outside", "Circle boundary"),
  col = c("green", "red", "blue"), pch = c(16, 16, NA), lty = c(NA, NA, 1), bty = "n"
)
```

R (easy) there is no estimate pi see next page

```
    n <- 500  # Number of points
r <- 2    # Radius
#or we use
#n <- as.integer(readline(prompt = "Enter the number of points (n): "))
#r <- as.numeric(readline(prompt = "Enter the radius (r): "))
# Start timing
start_time <- Sys.time()

# Generate random points
x <- runif(n, -r, r)  # Random points in range [-r, r]
y <- runif(n, -r, r)  # Random points in range [-r, r]

# Initialize counter
inside_count <- 0

# Loop through each point
for (i in 1:n) {
  if (x[i]^2 + y[i]^2 <= r^2) {  # Check if point is inside the circle
    inside_count <- inside_count + 1
  }
}

# Estimate the area of the circle
area_estimated <- (inside_count / n) * (2 * r)^2  # Monte Carlo area estimate

# Estimate the pi
pi_estimated <- 4*(inside_count / n)

# Real area of the circle
area_real <- pi * r^2

# End timing
end_time <- Sys.time()
time_taken <- end_time - start_time

# Print results
cat(sprintf("Estimated Circle Area: %.4f\n", area_estimated))
cat(sprintf("Real Circle Area: %.4f\n", area_real))
cat(sprintf("Estimated pi: %.4f\n", pi_estimated))
cat(sprintf("Real pi: %.4f\n", pi))
cat(sprintf("Time Taken: %.4f seconds\n", as.numeric(time_taken)))
```

```
# Plotting
plot(
  x, y, col = ifelse((x^2 + y^2 <= r^2), "green", "red"), pch = 16,
  xlab = "x", ylab = "y", main = "Monte Carlo Simulation: Circle within Square"
)
symbols(0, 0, circles = r, add = TRUE, inches = FALSE, lwd = 2, col = "blue")  # Circle bo
legend(
  "topright", legend = c("Points inside", "Points outside", "Circle boundary"),
  col = c("green", "red", "blue"), pch = c(16, 16, NA), lty = c(NA, NA, 1), bty = "n"
)
```

Second question

```
    estimer_aire_cercle <- function(r) {
  n <- 1000  # Nombre de points
  x <- runif(n, -r, r)  # Générer des points aléatoires pour x dans [-r, r]
  y <- runif(n, -r, r)  # Générer des points aléatoires pour y dans [-r, r]

  cercle <- 0

  for (i in 1:n) {
    if (x[i]^2 + y[i]^2 <= r^2) {
      cercle <- cercle + 1
    }
  }

  S_cercle <- (cercle / n) * (2 * r)^2

  cat("Estimated area of the circle :", S_cercle, "\n")

  return(S_cercle)
}
```