Tarea 1
Modelamiento Estocástico y Simulación

Universidad Técnica Federico Santa María
Departamento de Informática

Daniel San Martín Reyes
<daniel.sanmartinr@sansano.usm.cl>

Marzo del 2018

Tarea

Usando el método de Monte Carlo

1. Proponer un generador de Números aleatorios provenientes de la $\mathcal{U}(0,1)$ justifique como determina la calidad del generador.

2. Usando los resultados del problema 1 evaluar las integrales

   - $\displaystyle\int_{-\infty}^{\infty} e^{-x^2} dx$

   $$\theta = \int_{-\infty}^{\infty} e^{-x^2} dx. \tag{1}$$

   Dado que e^{-x^2} es simétrica con respecto a $x = 0$, entonces

   $$\theta = 2\int_{0}^{\infty} e^{-x^2} dx, \tag{2}$$

   Sea $y = \frac{1}{1+x}$, $dy = -\frac{dx}{(1+x)^2} = -y^2 dx$, entonces

   $$\theta = 2\int_{0}^{1} \frac{e^{-(\frac{1}{y}-1)^2}}{y^2} dy \tag{3}$$

   De esta forma, podemos estimar θ como $\hat{\theta} = \mathbb{E}[h(y)]$ con $h(y) = 2\frac{e^{-(\frac{1}{y}-1)^2}}{y^2}$.

   Ejecutando el código adjunto obtenemos $\hat{\theta} \approx 1{,}772264$.

- $\int_0^1 \int_0^1 e^{-(x+y)^2} dxdy$

$$\int_0^1 \int_0^1 e^{-(x+y)^2} dxdy \tag{4}$$

En este caso, θ puede ser estimado por $\hat{\theta} = \mathbb{E}[g(u_1, u_2)]$, con U_1, U_2 v.a.i.i.d $\mathcal{U}(0, 1)$, por lo tanto

$$\hat{\theta} = \frac{1}{k} \sum_{i=1}^{k} e^{-(x+y)^2}, \tag{5}$$

donde $X, Y \sim U(0, 1)$.

Ejecutando el código adjunto obtenemos $\hat{\theta} \approx 4{,}891916$.

3. Defina su Problema de simulación y su impacto , Conceptualización Modelo, los elementos constructivos, constructos del modelo, forma Recolección de Datos ; Construcción del Modelo, Verificación y Validación, forma de conducción delos experimentos y Análisis de los Resultados

Referencias

Ross, S. (2013). *Simulation* (5 ed.). Academic Press.

Código

```
# Question 1 - Randon Number Generator

# Using CPU frequency
RNG_freq <- function(N) {
  num <- vector(mode="numeric", length=N)
  #cat(num)
  for (i in 1:N) {
    cmd <- system("lscpu␣|␣grep␣'CPU␣MHz'", intern = TRUE)
    freq <- as.double(sapply(strsplit(cmd, "␣"), tail, 1))
    num[i] <- freq %% 1
  }
  return (num)
}

# Using Wichmann & Hill algorithm
RNG_WH <- function(N) {
  x <- vector(mode="numeric", length=N)
  y <- vector(mode="numeric", length=N)
  z <- vector(mode="numeric", length=N)
  u <- vector(mode="numeric", length=N)
```

```
  x[1] = 1
  y[1] = 2
  z[1] = 3

  for (i in 2:N) {
    x[i] = (171*x[i-1]) %% 30269
    y[i] = (172*y[i-1]) %% 30307
    z[i] = (170*z[i-1]) %% 30323

    u[i] = (x[i]/30269 + y[i]/30307 + z[i]/30323) %% 1
  }
  return(u)
}

# Using linear congruential generator
RNG_cong <- function(N) {
  m <- 9
  x <- vector(mode="numeric", length=N)
  u <- vector(mode="numeric", length=N)

  x[1] = 3
  x[2] = 2
  x[3] = 1

  u[1] = x[1] / m
  u[2] = x[2] / m
  u[3] = x[3] / m

  a <- c(8, 6, 2)

  for (n in 4:N) {
   x[n] = (a[1]*x[n-1] + a[2]*x[n-2] + a[3]*x[n-3]) %% m
   u[n] = x[n] / m
  }

  return(u)
}



# Exercise 1
f1 <- function(x) {
  y <- 5*(1+25*x^2)^(3/2)
}

# Exercise 2
f2 <- function(x) {
```

```
  y <- (1/x - 1) / (x*(1+(1/x-1)^2))^2
}

# Exercise 3
f3 <- function(x) {
  y <- 2*exp(-(1/x-1)^2)/x^2
}

# Exercise 4
f4 <- function(x, y) {
  y <- exp((x + y)^2)
}

# Monte Carlo Integration for 1D
montecarlo = function(f, k) {
  #X <- runif(k, 0, 1)
  #X <- RNG_freq(k)
  X <- RNG_WH(k)
  #X <- RNG_cong(k)
  int <- sum(f(X)) / k
}

# Monte Carlo Integration for 2D
montecarlo2D <- function (f, k) {
  X <- runif(k, 0, 1)
  Y <- runif(k, 0, 1)
  int <- sum(f(X, Y)) / k
}

# Number of repetitions
k = 1000000

# Compute integrals
r1 <- montecarlo(f1, k)
r2 <- montecarlo(f2, k)
r3 <- montecarlo(f3, k)
r4 <- montecarlo2D(f4, k)

# Print results
cat(sprintf("Integral 1: %f \n", r1))
cat(sprintf("Integral 2: %f \n", r2))
cat(sprintf("Integral 3: %f \n", r3))
cat(sprintf("Integral 4: %f \n", r4))
```