Tarea 1
Modelamiento Estocástico y Simulación

Universidad Técnica Federico Santa María
Departamento de Informática

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

28 de Marzo de 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.

   Para el desarrollo de este trabajo se propone utilizar la combinación de algoritmos congruenciales, en específico, basándose en el algoritmo combinado de Wichmann y Hill (Wichmann & Hill, 1982).

   La determinación de la calidad del generador puede ser realizado mediante: Contraste de Kolmogorov-Smirnov y con el Test χ^2.

2. Usando los resultados del problema 1 evaluar las integrales

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

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

   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 $\hat{\theta} \approx 1{,}772264$.

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

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

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 de los experimentos y Análisis de los Resultados.

   - **Problema:** Modelar la propagación de incendios forestales para ayudar a mejorar procesos de combate y control del fenómeno. En Chile este problema es de alto impacto debido al efecto dañino que tiene a nivel ambiental, económico y social. Es por esto que el desarrollo de modelos de buena calidad, coherentes con la realidad, se hacen importantes para el estudio, análisis y predicción del fenómeno. El principal objetivo es implementar un modelo que ayude a profesionales de instituciones que trabajen con este problema ya sea CONAF, ONEMI, etc.
   - **Conceptualización del Modelo:** Para el desarrollo del modelo se busca establecer la relación entre el comportamiento del fuego con la interacción entre el combustible y las condiciones meteorológicas del escenario a simular.
   - **Recolección de Datos:** Los datos necesarios para el desarrollo del trabajo deben ser obtenidos de fuentes gubernamentales, específicamente de instituciones como CONAF para el modelar el combustible y la DMC para conocer el modelo relacionado a los datos meteorológicos.
   - **Construcción del Modelo:** Dado que el problema es el manejo y/o control de la propagación del fuego en un incendio forestal, la idea es construir un modelo cualitativamente coherente con la realidad. Para iniciar, se busca comenzar con un modelo basado en autómata celular dada que la implementación es considerablemente más simple que comenzar con uno basado en ecuaciones diferenciales. El modelo final espera ser construido mediante ecuaciones diferenciales estocásticas, siempre que el tiempo así lo permita.
   - **Verificación y Validación:** Para verificar el modelo se intentará contactar con profesionales expertos en el ámbito. Para la validación se espera contar con datos de las instituciones pertinentes y que nos permitan contrastar resultados.
   - **Conducción de los Experimentos:** Por definir.
   - **Análisis de los Resultados:** Por realizar.

Referencias

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

Wichmann, B. A. & Hill, I. D. (1982). Algorithm as 183: An efficient and portable pseudo-random number generator. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *31*(2), 188–190.

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] = 6
  y[1] = 6
  z[1] = 6

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

  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)
}
```

```r
# 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)
}

validation <- function(sample) {
  # Test Chi-squared
  intervals <- seq(0, 1, length.out=10)
  sample.counts <- hist(sample, breaks=intervals, plot=F)$counts
  chsq <- chisq.test(rbind(sample.counts, mean(sample.counts)))

  # Test de Kolmogorov - Smirnov
  ks <- ks.test(sample, 'punif')

  print(ks)
  print(chsq)

}

# Chi squared
N <- 1000#0000
X <- RNG_WH(N)
#X <- RNG_cong(N)
#X <- runif(N, 0, 1)

validation(X)
```

```r
# Question 2

# 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))
```