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 una combinación de algoritmos congruenciales. La combinación de algoritmos congruenciales se basa en los siguientes resultados:

   - Si $U_1, ..., U_k$ son variables aleatorias iid $\mathcal{U}(0,1)$ la parte fraccionaria de $U_1 + ... + U_k$ también sigue una distribución $\mathcal{U}(0,1)$, es decir,

     $$U_1 + U_2 + ... + U_k - [U_1 + U_2 + ... + U_k] \sim \mathcal{U}(0,1). \tag{1}$$

     Notar que $U_1 + U_2 + ... + U_k - [U_1 + U_2 + ... + U_k]$ es equivalente a $(U_1 + U_2 + ... + U_k) \bmod 1$
   - Si $u_1, u_2, ..., u_k$ están generados por algoritmos congruenciales con ciclos de periodo $c_1, c_2, ..., c_k$, respectivamente, entonces la parte fraccionaria de $u_1 + u_2 + ... + u_k$ tiene un ciclo de periodo m.c.m.$\{c_1, c_2, ..., c_k\}$.

   En este trabajo utilizamos el algoritmo combinado de Wichmann y Hill (Wichmann & Hill, 1982). El generador de este algoritmo es:

   $$\begin{aligned} x_i &= 171\ x_{i-1} \bmod 30269 \\ y_i &= 172\ y_{i-1} \bmod 30307 \\ z_i &= 170\ z_{i-1} \bmod 30323, \end{aligned} \tag{2}$$

   luego

$$u_i = \left(\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323}\right) \bmod 1. \tag{3}$$

Para intentar mejorar la implementación, las semillas del algoritmo fueron escogidas utilizando la parte decimal de la frecuencia del procesador del equipo en el momento que se ejecuta el código.

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

- Test χ^2: Dada una muestra $X_1, X_2, ..., X_n$ de una $F_x(x)$ desconocida, se desea contrastar
$$H_0 : F_x(x) = F_0(x) \qquad H_1 : F_x(x) \neq F_0(x). \tag{4}$$
Realizando una partición del soporte de X en k subconjuntos disjuntos $I_1, I_2, ..., I_k$ tal que $\bigcup I_i = X$ y $I_i \bigcap i_j = \emptyset$. Luego, se considera el estadístico
$$T = \sum_{i=1}^{k} \frac{(f_i - e_i)^2}{e_i} \ \sim \ \chi^2_{(k-1)}, \tag{5}$$
donde f_i corresponde a la frecuncia absoluta del conjunto i-ésimo (I_i) y e_i es el número de observaciones esperadas en I_i bajo H_0. Este test considera la aleatoriedad de $F_0 = \mathcal{U}(0, 1)$.

  El problema de este test de bondad de ajuste es que nos permite justificar el rechazo de la hipótesis, pero no ayuda mucho en la aceptación de esta.

- Contraste de Kolmogorov-Smirnov: Dada una muestra aleatoria simple $x_1, ..., x_n$, la función de distribución empírica de la muestra es
$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{x_i \leq x} = \frac{|\text{valores } x_i \leq x|}{n}. \tag{6}$$
Las hipótesis son:
$$H_0 : F_n = F_0 \qquad H_1 : F_n \neq F_0, \tag{7}$$
donde F_0 es la hipotética función de distribución. El estadístico de contraste para el test K-S es:
$$D_n = \sup_{x \in \mathbb{R}} |F_n(x) - F_0(x)|. \tag{8}$$
La distribución exacata de D_n está tabulada para valores de $n \leq 40$ y distintos niveles de significación α. Para muestras grandes se utiliza la deistribución asintóticas de D_n dada por
$$\lim_{n\to\infty} P(\sqrt{n}D_n \leq z) = L(z) = 1 - 2\sum_{i=1}^{\infty}(-1)^{i-1}e^{-2i^2z}. \tag{9}$$

Para el caso particular de que F_0 es la función de distribución de una v.a $\mathcal{U}(0,1)$, por lo tanto se puede comprobar que el estadístico de Kolmogorov-Smirnov para contraster la uniformidad de la muestra $u_1, ..., u_n$ viene dado por:

$$D_n = \underset{1\leq i\leq n}{\text{máx}} \left\{ \text{máx} \left\{ \left| \frac{i}{n} - u_{(i)} \right|, \left| \frac{i-1}{n} - u_{(i)} \right| \right\} \right\} \tag{10}$$

siendo $u_{(i)}$ el i-ésimo menor valor de la muestra.

Utilizando las técnicas mencionadas anteriormente obtenemos que el generador propuesto sería al menos útil, porque los resultados nos permiten concluir que no existe evidencia suficiente para rechazar la hipótesis nula, que en este caso particular considera la aleatoriedad de $F_0 = \mathcal{U}(0,1)$ dada una muestra de números aleatorios creados con nuestro generador. La implementación puede ser revisada en el código adjunto.

2. Usando los resultados del problema 1 evaluar las integrales.

Mediante la ley de los grandes números, podemos estimar las integrales con

$$\sum_{i=1}^{k} \frac{g(u_i)}{k} \;\rightarrow\; \mathbb{E}[g(u)] \;\rightarrow \theta \text{ cuando } k \rightarrow \infty, \tag{11}$$

con $U \sim \mathcal{U}(0,1)$ generados con la propuesta del problema 1.

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

$$\theta = \int_{-\infty}^{\infty} e^{-x^2} dx. \tag{12}$$

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

$$\theta = 2 \int_{0}^{\infty} e^{-x^2} dx. \tag{13}$$

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

$$\theta = 2 \int_{0}^{1} \frac{e^{-(\frac{1}{y}-1)^2}}{y^2} dy \tag{14}$$

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 $\hat{\theta} \approx 1{,}77$.

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

$$\theta = \int_0^1 \int_0^1 e^{(x+y)^2} dxdy \tag{15}$$

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

$$\hat{\theta} = \frac{1}{k} \sum_{i=1}^{k} e^{(x+y)^2}, \tag{16}$$

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

Ejecutando el código adjunto obtenemos $\hat{\theta} \approx 4{,}89$.

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. En el caso que no exista la posibilidad de acceder a esta información, se puede utilizar fuentes de datos abiertas como lo son Google Maps API, Open Weather Map API, NOAA, entre otros.
   - **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 autómatas celulares 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

# Get CPU frequency decimal part for seed
CPU_seed <- function(N) {
  num <- vector(mode="numeric", length=N)

  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 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 <- CPU_seed(3)

  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)
}

# 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)

  # Using CPU for seed
  cpu = CPU_seed(3)
  x[1] = cpu[1]
  y[1] = cpu[2]
  z[1] = cpu[3]

  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)
}

# Validation with chi-squared and K-S
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)
}

# Quality of RNG
N <- 10000000
X <- RNG_WH(N)
```

```
validation(X)


# 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) {
  z <- exp((x + y)^2)
}

# Monte Carlo Integration for 1D
montecarlo = function(f, k) {
  #X <- runif(k, 0, 1)
  X <- RNG_WH(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)
  X <- RNG_WH(k)
  Y <- RNG_WH(k)
  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))
```