Laboratory Session 04 : April 27, 2020

Exercises due : May 13, 2020

Exercise 1

- The triangular distribution, in the interval (a, b), is given by the following:

$$f(X) = \begin{cases} \dfrac{2(x-a)}{(b-a)(c-a)} & a \leq x < c \\[2ex] \dfrac{2(b-x)}{(b-a)(b-c)} & c \leq x \leq b \\[2ex] 0 & \text{otherwise} \end{cases}$$

  where $c \in [a, b]$.

a) plot the function, given the interval (a, b)

b) and write an algorithm to generate random numbers from the triangular distribution

c) generate 10^4 random number from the distribution, show them in an histogram and superimpose the analytical curve

Exercise 2

- given a discrete probability distribution, defined by the following probabilities: `0.05, 0.19, 0.14, 0.17, 0.02, 0.11, 0.06, 0.05, 0.04, 0.17`

a) plot the probability density function and the cumulative density function

b) write an algorithm to generate random numbers from the discrete probability distribution

Exercise 3

- Generate random variables from the following distribution

$$f(X) = \frac{2}{\pi R^2} \sqrt{R^2 - x^2}$$

- where $-R \leq x \leq R$

a) using the acceptance-rejection algorithm, assume $M = 2/(\pi \mathrm{R})$ and generate 10^4 random variables, plotting them in an histogram

Exercise 4

- An important property of the gamma distribution is the so-called *reproductive property*
- given a sequence of independent random variable $X_j \sim \text{Gamma}(\alpha_j, \beta)$, it follows that

$$Y = \sum_{j=1}^{n} X_j \rightarrow Y \sim \text{Gamma}(\alpha, \beta) \quad \text{where} \quad \alpha = \sum_{j=1}^{n} \alpha_j$$

- if $\alpha = m$ is an integer, a random variable from gamma distribution $\text{Gamma}(m, \beta)$ (also known as Erlang distribution) can be obtained by summing m independent exponential random variables $X_j \sim \text{Exp}(\beta)$:

$$Y = \beta \sum_{j=1}^{n} (-\ln U_j) = -\beta \ln \prod_{j=1}^{n} U_j$$

a) write an algorithm to sample variables from an Erlang distribution $\text{Gamma}(m, \beta)$

Exercise 5

- one of the first random number generator was proposed by von Neumann, the so-called *middle square* algorithm
- write R code to implement this type of generator and, given a fixed digit number input, square it an remove the leading and trailing digits, in order to return a number with the same number of digits as the original number
- *Suggestion :* after having squared the number, convert it to a list of characters
(`number <- unlist(strsplit(as.character(x.squared),""))`)
and, after having removed the head and tail of the list, convert it back to a number
(`as.numeric(paste(number.after.trimming, collapse=""))`)