UTN: FACULTAD REGIONAL ROSARIO

DEPARTAMENTO DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Cátedra: Simulación

Trabajo Práctico Integrador

Autores:

Andrés J. Botello	Legajo: 43697
Santiago Ciriaci	Legajo:43828
Joaquín Romero	Legajo:43740

Supervisado por:
Ing. Torres, Juan Ignacio

Trabajo Práctico de Simulación

Andrés Julián Botello, Joaquín Romero, Santiago Ciriaci

Andrés Botello: 43697, Joaquín Romero: 43740, Santiago Ciriaci: 43828

Abstracto

En el siguiente informe presentaremos no sólo conceptos relacionados a la simulación, sino que además explicaremos el porqué de esta, sus ventajas y desventajas. Para esto necesitaremos utilizar conocimientos previos de probabilidades y estadísticas, que se mencionarán brevemente y más tarde en secciones subsecuentes aplicaremos un caso real con el objetivo de obtener mayor claridad, es el caso de la ruleta de un casino. Además, veremos cómo generar valores de variables estocásticas o aleatorias para diversas distribuciones de probabilidad usando métodos como la "transformada inversa". Finalmente haremos uso de los tests de Chi cuadrado y de Corridas para verificar uniformidad e independencia(y por lo tanto, aleatoriedad).

Palabras Clave:
Simulación, Ruleta, Distribuciones de Probabilidad, Generadores, Chi cuadrado, Corridas.

Correos Electronicos: andresjbotello@gmail.com (Andrés Julián Botello), romero.joaquin98@gmail.com (Joaquín Romero), santiagociriaci@gmail.com (Santiago Ciriaci)

Índice

I Simulación - Fundamentos 3

1. Simulación 4
1.1. ¿Qué es simulación? 4

II Ruleta 5

2. Ruleta - Aplicación de conceptos de Probabilidad y Estadística 6
2.1. Introducción. 6
2.2. Frecuencia Absoluta. 6
2.3. Frecuencia Relativa. 6
2.4. Esperanza. 6
2.5. Varianza. 6
2.6. Histograma. 6

3. Metodología 6
3.1. Hipótesis 6

4. Casos de estudio 7
4.1. Población. 7
4.2. Experimento 1 - Frecuencia Relativa 7
4.3. Experimento 2 - Esperanza. 7
4.4. Experimento 3 - Esperanza de la esperanza . . 7
4.5. Experimento 4 - Varianza 8
4.6. Experimento 5 - Histograma. 8
4.7. Conclusiones 8

III Distribuciones de Probabilidad y Generadores 9

5. Marco Teórico de Distribuciones de Probabilidad y Generadores 10
5.1. Distribución Uniforme 10
5.2. Distribución Exponencial 10
5.3. Distribución Gamma 11
5.4. Distribución Normal 11
5.4.1. Procedimiento del Límite Central . . . 12
5.4.2. Procedimiento Directo 12
5.5. Distribución Binomial 12
5.6. Distribución de Poisson 13
5.7. Distribución Empírica 14
5.7.1. Distribución empírica discreta 14
5.7.2. Distribución empírica continua 14

6. Metodología 16

7. Conclusiones 16
7.1. Distribución Uniforme 16
7.2. Distribución Exponencial 16
7.3. Distribución Gamma 16
7.4. Distribución Normal 17
7.5. Distribución Binomial 18
7.6. Distribución de Poisson 18
7.7. Distribución empírica discreta 18
7.8. Distribución empírica continua 19

IV GCL 20

8. Generadores Congruenciales Lineales - GCL 21

V Pruebas de independencia y aleatoriedad 22

9. Tests de Chi cuadrado y de Corridas 23
9.1. Enunciado 24

VI Anexo 25

10. Anexo 26
10.1. Codificación - Uniforme 26
10.2. Codificación - Exponencial 27
10.3. Codificación - Gamma 28
10.4. Codificación - Normal 29
10.4.1. Normal - Procedimiento del límite central 29
10.4.2. Normal - Procedimiento Directo 30
10.4.3. Normal - Python 30
10.5. Codificación - Binomial 31
10.6. Codificación - Poisson 31
10.7. Codificación - Empírica 32
10.7.1. Empírica Discreta 32
10.7.2. Empírica Continua 32
10.8. Codificación de Tests (Chi cuadrado y Corridas) 33
10.8.1. Ejercicio 1 33
10.8.2. Ejercicio 2 34
10.8.3. Ejercicio 3 35
10.8.4. Ejercicio 4 36
10.8.5. Ejercicio 5 37
10.8.6. Ejercicio 6 38

Parte I

Simulación - Fundamentos

1. Simulación

Probablemente uno se pregunte, porqué es que deberíamos o necesitaríamos realizar una simulación. Si bien existen infinidades de aplicaciones de esta técnica, el objetivo de primordial importancia que nosotros perseguimos es el de resolver problemas que carecen de una solución analítica. Aunque como todo, tiene sus ventajas y desventajas. De estas últimas, las principales son su elevado costo y la no obtención de resultados exactos (estimadores).

1.1. ¿Qué es simulación?

Thomas T. Goldsmith Jr. y Estle Ray Mann la definen así: "Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos períodos".
Una definición más formal, formulada por R. E. Shannon, es: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema".
El proceso de simulación conlleva una serie de pasos a seguir.

1. **Definición del sistema**.
   Consiste en estudiar el contexto del problema, identificar los objetivos del proyecto, especificar los índices de medición de la efectividad del sistema, establecer los objetivos específicos del modelamiento y definir el sistema que se va a modelar un sistema de simulación.
2. **Formulación del modelo**.
   Una vez definidos con exactitud los resultados que se espera obtener del estudio, se define y construye el modelo con el cual se obtendrán los resultados deseados. En la formulación del modelo es necesario definir todas las variables que forman parte de él, sus relaciones lógicas y los diagramas de flujo que describan en forma completa el modelo.
3. **Colección de datos**.
   Es importante que se definan con claridad y exactitud los datos que el modelo va a requerir para producir los resultados deseados.
4. **Implementación del modelo en la computadora**.
   Con el modelo definido, el siguiente paso es decidir qué lenguaje de programación (como Fortran, Algol, Lisp, etc.) o qué paquete de software se va a utilizar para procesar el modelo en la computadora y obtener los resultados deseados.
5. **Verificación**.
   El proceso de verificación consiste en comprobar que el modelo simulado cumple con los requisitos de diseño para los que se elaboró. Se trata de evaluar que el modelo se comporta de acuerdo a su diseño.
6. **Validación del sistema**.
   A través de esta etapa se valoran las diferencias entre el funcionamiento del simulador y el sistema real que se está tratando de simular. Las formas más comunes de validar un modelo son:
   - La opinión de expertos sobre los resultados de la simulación.
   - La exactitud con que se predicen datos históricos.
   - La exactitud en la predicción del futuro.
   - La comprobación de falla del modelo de simulación al utilizar datos que hacen fallar al sistema real.
   - La aceptación y confianza en el modelo de la persona que hará uso de los resultados que arroje el experimento de simulación.
7. **Experimentación**.
   La experimentación con el modelo se realiza después que este haya sido validado. La experimentación consiste en comprobar los datos generados como deseados y en realizar un análisis de sensibilidad de los índices requeridos.
8. **Interpretación**.
   En esta etapa del estudio, se interpretan los resultados que arroja la simulación y con base a esto se toma una decisión. Es obvio que los resultados que se obtienen de un estudio de simulación colabora a soportar decisiones del tipo semi-estructurado.
9. **Documentación**.
   Dos tipos de documentación son requeridos para hacer un mejor uso del modelo de simulación. La primera se refiere a la documentación del tipo técnico y la segunda se refiere al manual del usuario, con el cual se facilita la interacción y el uso del modelo desarrollado.

Parte II

Ruleta

2. Ruleta - Aplicación de conceptos de Probabilidad y Estadística

Este trabajo consiste en realizar cinco experiencias distintas, y su posterior análisis, utilizando como caso de estudio una ruleta que comprenda los números enteros de 0 a 36. Finalmente se debe demostrar que esta posee propiedades equiprobables.

2.1. Introducción.

La estadística como disciplina se relaciona con las técnicas y los métodos que se han desarrollado para planear experiencias, recopilar, organizar, resumir, analizar, interpretar y comunicar la información proveniente de datos tanto cuantitativos como cualitativos.

Si dichos datos son un subconjunto de una población mas grande, se dice que son ”datos muestrales”. En cambio, si este conjunto de datos abarca a toda la población, son ”datos poblacionales”.

Para que estos datos resulten comprensibles es necesario organizarlos, representarlos gráficamente y definir medidas descriptivas que sinteticen la información.

2.2. Frecuencia Absoluta.

La frecuencia absoluta es el número de veces que aparece un determinado valor en un estudio estadístico. La suma de las frecuencias absolutas es igual al número total de datos, que se representa por N.

$$\sum_{i=1}^{n} F_i = N$$

2.3. Frecuencia Relativa.

La frecuencia relativa es el cociente entre la frecuencia absoluta de un determinado valor y el número total de datos. La suma de las frecuencias relativas es igual a 1.

$$f_i = \frac{F_i}{N}$$

$$\sum_{i=1}^{n} f_i = 1$$

2.4. Esperanza.

En matemáticas y estadística, la esperanza, también llamada promedio o media, de un conjunto finito de números es el valor característico de una serie de datos cuantitativos, objeto de estudio que parte del principio de la esperanza matemática o valor esperado

Si $x_1, x_2, x_3, ..., x_N$ es una población finita de tamaño N entonces la esperanza de la población es:

$$\mu = \frac{x_1 + x_2 + x_3 + ... + x_N}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2.5. Varianza.

La varianza es una medida de dispersión que representa la variabilidad de una serie de datos respecto a su media.

Si $x_1, x_2, x_3, ..., x_N$ es una población finita de tamaño N se define la varianza poblacional como:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

2.6. Histograma.

Un histograma es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia, ya sea absoluta o relativa, de los valores representados.

Su función es exponer gráficamente números, variables y cifras de modo que los resultados se visualicen más clara y ordenadamente.

Normalmente, las frecuencias son representadas en el eje vertical mientras que en el horizontal se representan los valores de cada una de las variables.

3. Metodología

Consideramos una ruleta con 37 variables discretas (los números enteros de 0 a 36) siendo estos los valores que puede tomar la población que nos dedicaremos a analizar. Para lograr este análisis, generamos aleatoriamente 2700 valores que conformaran nuestra población.

Dentro de cada uno de los cinco experimentos, utilizamos dichos valores para analizar como evolucionan los mismos a medida que van aumentando las iteraciones.

Para facilitar el análisis de estos datos, utilizamos el lenguaje de programación *python* el cual tiene una amplia variedad de librerías. De todas ellas, utilizaremos únicamente *random* (utilizada para generar los valores aleatorios) y *matplotlib* (para graficar la evolución de los datos).

La forma en la que realizamos los experimentos la desarrollaremos en la sección casos de estudio.

3.1. Hipótesis

Nuestra hipótesis consiste en el supuesto de que a medida que aumentan las iteraciones al evaluar una mayor cantidad de datos, el resultado final que toman los experimentos de frecuencia relativa, esperanza, esperanza de la esperanza y varianza de la esperanza tienden al valor que le correspondería según las fórmulas definidas en el marco teórico evaluado.

En el caso de las esperanzas, estas deben converger en el valor 18. Ya que, al ser equiprobable nuestra ruleta, los valores convergen al valor promedio entre sus extremos.

Para la varianza podemos intuir que a medida que aumentan las iteraciones y la esperanza converge, los valores que tome la varianza de la esperanza van a tender a cero, ya que la esperanza no varía significativamente.

Por otro lado, el experimento que corresponde a la representación de los valores en un histograma debería tener una forma simil rectangular ya que las frecuencias absolutas de todas las variables tendrían que ser similares, debido a que son equiprobables.

4. Casos de estudio

4.1. Población.

Para generar la población de una manera aleatoria utilizamos una lista, la cual rellenamos con valores enteros en un rango de 0 a 36 utilizando la función *randint*.

4.2. Experimento 1 - Frecuencia Relativa

Primero elegimos un valor al azar de la población, siete en este caso, para posteriormente tomar este valor y analizar como evoluciona su frecuencia relativa a medida que generamos nuevos valores de la ruleta.

Para analizar dicha frecuencia, utilizamos la fórmula descripta en la sección 2.3.

Para obtener la frecuencia absoluta del elemento elegido dentro de la población, utilizamos la función *count* la cual nos provee la cantidad de veces que dicho elemento se manifiesta en la población.

Para conseguir el valor de *N* utilizamos la función *len* que nos devuelve el total de los elementos generados en esta iteración.

Luego de las 2700 iteraciones utilizamos la función *plot* de la librería *Matplotlib*, la cual gráfica el valor analizado para cada iteración.

4.3. Experimento 2 - Esperanza.

Para realizar este experimento tomamos de a uno los elementos de la lista definida en la sección 4.1, y evaluamos la esperanza para cada nuevo valor que tomamos y la depositamos en una nueva lista. Para esto, utilizamos la fórmula definida en la sección 2.4.

La obtención de los datos requeridos la hacemos mediante la función *sum* la que calcula automáticamente la sumatoria de los valores en la lista. Al igual que en el experimento anterior, repetimos dicho procedimiento 2700 veces para finalmente graficar el resultado de la nueva lista obtenida.

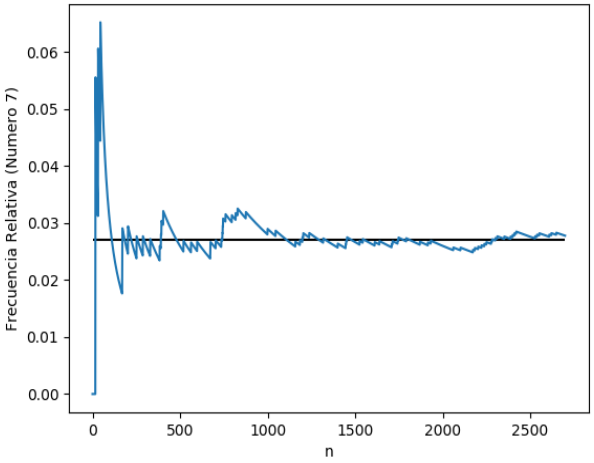
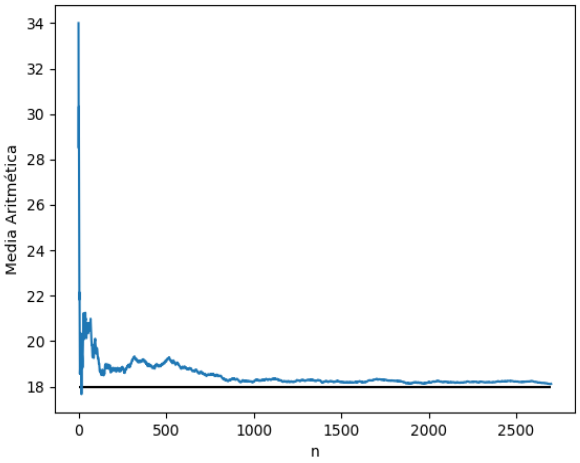
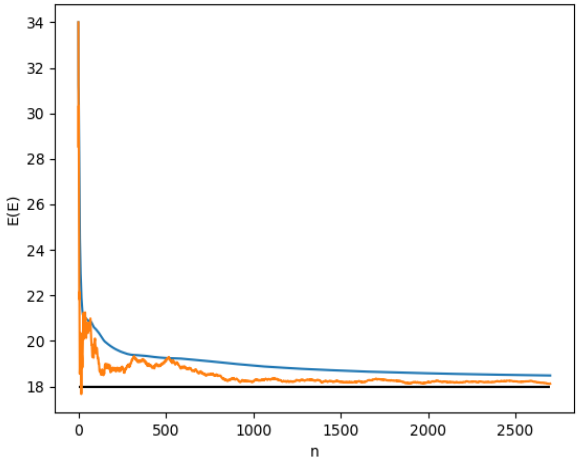


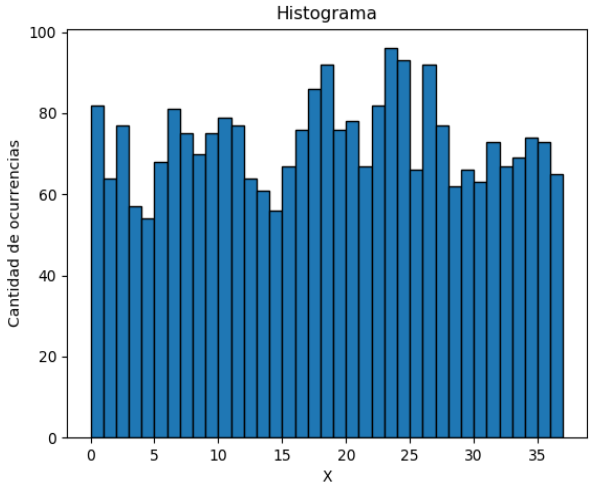
Figura 1: Frecuencia Relativa.



4.4. Experimento 3 - Esperanza de la esperanza

Al igual que el experimento anterior calculamos los valores de la esperanza pero utilizando como dato de entrada la lista de esperanzas calculada anteriormente. Para una mejor comparación, incluimos en la gráfica los valores de la esperanza del experimento anterior (representados en color naranja) junto con los de este (representados en color azul).



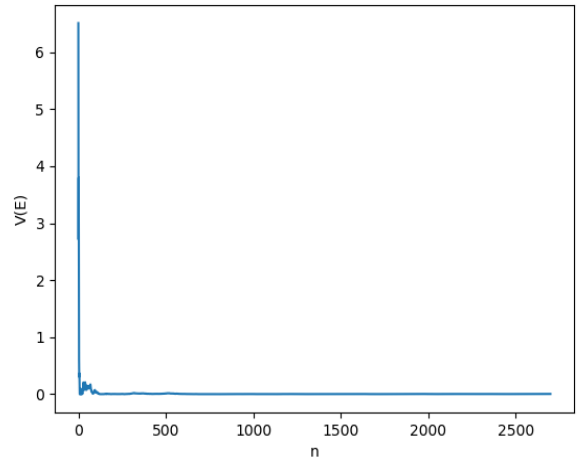


4.5. *Experimento 4 - Varianza*

El objetivo de este experimento es calcular los valores de la varianza utilizando como datos de entrada los valores de la esperanza calculados en el experimento 2.

Para realizarlo, empleamos la fórmula desarrollada en la sección 2.5. Para la cual, inicialmente, calculamos la esperanza total utilizando todos los datos de la lista. Y posteriormente, los almacenamos en una tercer lista.

Finalmente, al igual que en los demás experimentos, utilizamos la función del lenguaje para graficar los valores que toma la varianza mientras recorremos la lista.



4.6. *Experimento 5 - Histograma.*

Para el último experimento tomamos nuevamente la lista de la ruleta inicial y colocamos en la función *hist*, perteneciente a la librería *matplotlib*, junto con la descripción de la distribución de los intervalos y otros parámetros estéticos, como el color, transparencia, etcétera.

4.7. *Conclusiones*

Como podemos ver en las gráficas incluidas en los casos de estudio, los valores de los primeros cuatro experimentos cumplen satisfactoriamente con la hipótesis planteada en la sección 3.1. Para comprobarlos utilizamos, en cada una de estas gráficas, una recta de referencia sobre los valores a los cuales deben converger.

También podemos observar que se satisface la condición definida para el experimento numero 5, ya que este posee valores de frecuencias absolutas similares, por lo tanto presenta una forma aparentemente rectangular.

Si considerásemos una población aún mas grande podríamos notar estos detalles con mayor claridad.

A partir de los análisis realizados en los experimentos podemos afirmar que nuestra población se compone de valores equiprobables.

Parte III

Distribuciones de Probabilidad y Generadores

5. Marco Teórico de Distribuciones de Probabilidad y Generadores

Muchas veces los indicadores de la estadística descriptiva no nos proporcionan una imagen clara de nuestros datos. Por esta razón, siempre es útil complementarlos con gráficos de las distribuciones de los datos, que describan con qué frecuencia aparece cada valor. La representación más común de una distribución es un histograma, que es un gráfico que muestra la frecuencia o probabilidad de cada valor. El histograma muestra las frecuencias como un gráfico de barras que indica cuan frecuente un determinado valor ocurre en el conjunto de datos. El eje horizontal representa los valores del conjunto de datos y el eje vertical representa la frecuencia con que esos valores ocurren. En este informe, se trabajará con distribuciones continuas. Entre ellas veremos la distribución uniforme, exponencial y gamma.

5.1. *Distribución Uniforme*

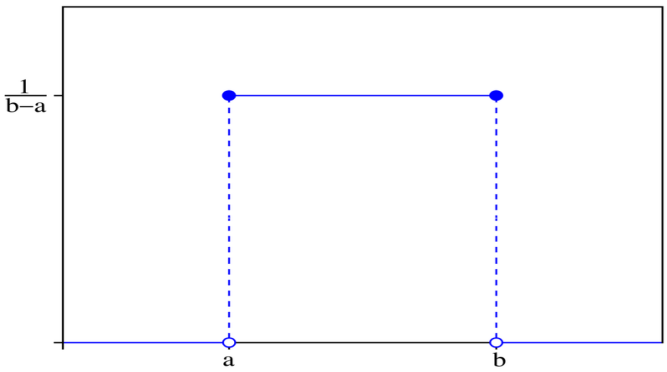


Figura 2: Función de densidad de probabilidad, distribución uniforme

En teoría de probabilidad y estadística, la distribución uniforme continua es una familia de distribuciones de probabilidad para variables aleatorias continuas, tales que para cada miembro de la familia, todos los intervalos de igual longitud en la distribución en su rango son igualmente probables. El dominio está definido por dos parámetros, a y b, que son sus valores mínimo y máximo. La distribución es a menudo escrita en forma abreviada como $U(a, b)$.

La **función de densidad de probabilidad** de la distribución uniforme continua es:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{a} \leq \text{x} \leq \text{b} \\ 0 & \text{fuera del intervalo (a,b)} \end{cases} \tag{1}$$

Los valores en los dos extremos a y b no son por lo general importantes porque no afectan el valor de las integrales de $f(x)dx$ sobre el intervalo, ni de $xf(x)dx$ o expresiones similares. A veces se elige que sean cero, y a veces se los elige con el valor $\frac{1}{b-a}$. Este último resulta apropiado en el contexto de estimación por el método de máxima verosimilitud. En el contexto del análisis de Fourier, se puede elegir que el valor de $f(a)$ o $f(b)$ sean $\frac{1}{2(b-a)}$, para que entonces la transformada inversa de muchas transformadas integrales de esta función uniforme resulten en la función inicial, de otra forma la función que se obtiene sería igual .[en] casi todo punto", o sea excepto en un conjunto de puntos con medida nula. También, de esta forma resulta consistente con la función signo que no posee dicha ambigüedad.

La esperanza viene dada por:

$$E(X) = \frac{a+b}{2} \tag{2}$$

y la varianza por:

$$V(X) = \frac{(b-a)^2}{12} \tag{3}$$

Las siguienes expresiones, provenientes del método de momentos, permiten calcular los valores a y b:

$$a = EX - \sqrt{3VX} \tag{4}$$

$$b = 2EX - a \tag{5}$$

Finalmente, para simular una distribución uniforme sobre cierto intervalo (a, b) deberemos, en primer lugar, obtener la transformación inversa para la ecuacion

$$F(x) = \int_a^x \frac{1}{b-a} \cdot dt = \frac{x-a}{b-a}, \qquad 0 \leq F(x) \leq 1 \tag{6}$$

5.2. *Distribución Exponencial*

En estadística la distribución exponencial es una distribución de probabilidad continua con un parámetro $\lambda > 0$ cuya función de densidad es:

$$f(x) = P(x) = \begin{cases} \lambda e^{-\lambda x} & \text{para } x \geq 0 \\ 0 & \text{en caso contrario} \end{cases} \tag{7}$$

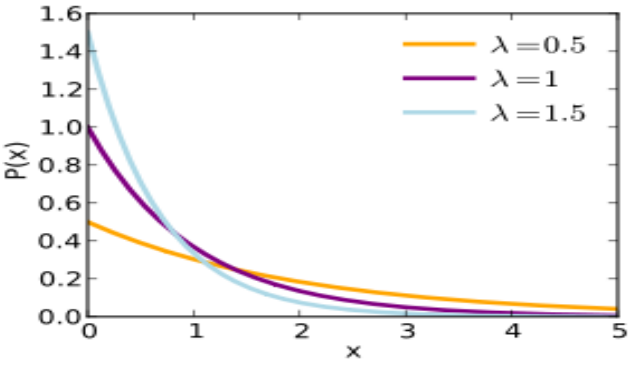


Figura 3: Función de densidad de probabilidad, distribución exponencial

Su función de distribución acumulada es:

$$F(x) = P(X \leq x) = \begin{cases} 0 & \text{para } x < 0 \\ 1 - e^{-\lambda x} & \text{para } x \geq 0 \end{cases} \tag{8}$$

La esperanza y la varianza de una variable aleatoria X con distribución exponencial son:

$$E[X] = \frac{1}{\lambda}, \qquad V(X) = \frac{1}{\lambda^2} \tag{9}$$

La distribución exponencial es un caso particular de distribución gamma con $k = 1$, la cual veremos próximamente.

Existen muchas maneras para lograr la generación de valores de variables aleatorias exponenciales. Puesto que $F(x)$ existe implícitamente, la técnica de la transformación inversa nos permite desarrollar métodos directos para dicha generación. Debido a la simetría que existe entre la distribución uniforme sigue que la intercambiabilidad de $F(x)$ y $1 - F(x)$. Por lo tanto

$$r = e^{-\alpha x} \tag{10}$$

y consecuentemente

$$x = -\frac{1}{\alpha} \log r = -EX \log r \tag{11}$$

5.3. Distribución Gamma

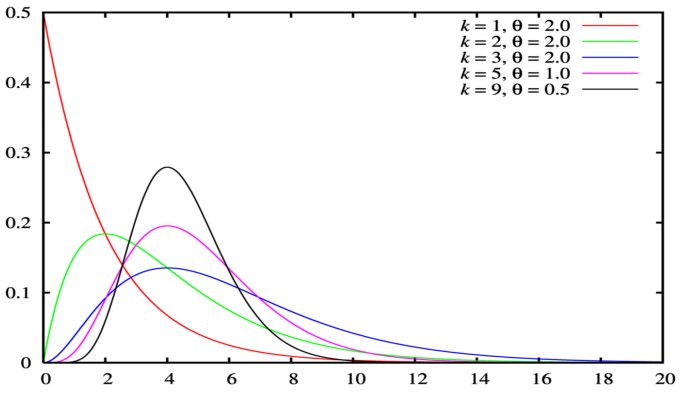


Figura 4: Distribución Gamma

Si un determinado proceso consiste de k eventos sucesivos y si el total del tiempo transcurrido para dicho proceso se puede considerar igual a la suma de k valores independientes de la variable aleatoria con distribución exponencial, cada uno de los cuales tiene un parámetro definido α, la distribución de esta suma coincidirá con una distribución gamma con parámetros α y k. La suma de los k (donde k es un número positivo) valores de la variable aleatoria con distribución exponencial con un mismo parámetro α, también recibe el nombre de distribución de Erlang.

La función gamma está descrita mediante la siguiente función de densidad:

$$f(x) = \frac{\alpha^k x^{k-1} e^{-\alpha x}}{(k-1)!} \qquad \alpha > 0, k > 0, x > 0 \tag{12}$$

Respecto a la media y la varianza de esta distribución, sus correspondientes expresiones están formuladas como sigue:

$$EX = \frac{k}{\alpha} \tag{13}$$

$$VX = \frac{k}{\alpha^2} \tag{14}$$

Cabe aclarar que si $k = 1$, entonces la distribución gamma resulta ser idéntica a la distribución exponencial; mientras que si k es un entero positivo, la distribución gamma coincide con la distribución de Erlang.

Debido a que para una distribución gamma no se puede formular explícitamente una función de distribución acumulativa, debemos considerar un método alternativo para generar valores de variable aleatoria con distribución gamma. En relación a tales valores que satisfacen la distribución Erlang, éstos se pueden generar con sólo reproducir el proceso aleatorio sobre el cual se basa la distribución de Erlang. Para lograr este resultado se debe tomar la suma de los k valores de variable aleatoria con distribución exponencial $x_1, x_2, ..., x_k$, cuyo valor esperado es el mismo e igual a $\frac{1}{\alpha}$. En consecuencia, el valor de la variable aleatoria (Erlang) x se puede expresar como

$$x = \sum_{i=1}^{k} x_i = -\frac{1}{\alpha} \sum_{i=1}^{k} \log r_i \tag{15}$$

luego, se simplifica a una forma computacional equivalente y más rápida, que la substituye. Esta expresión corresponde a la siguiente fórmula:

$$x = -\frac{1}{\alpha} (\log \prod_{i=1}^{k} r_i) \tag{16}$$

5.4. Distribución Normal

La más conocida y más ampliamente utilizada ditribución de probabilidad es sin duda la distribución normal. Ella basa su utilidad en el teorema del límite central el cual no es objeto de explicación en este informe. Este teorema permite el empleo de distribuciones normales para representar medidas globales operadas sobre los efectos de causas (errores) aditivas distribuidas en forma independiente, sin importar la distribución de probabilidad a que obedezcan las mediciones de causas individuales. Otro razgo valiosamente práctico de la distribución normal, es su utilidad para aproximar distribuciones de Poisson y binomiales para mencionar solo dos entre muchas. Si la variable aleatoria X tiene una función de densidad $f(x)$ dada como

$$f(x) = \frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_x}{\sigma_x})^2}, \quad x \in \mathbb{R} \tag{17}$$

con σ_x positiva, entonces se dice que X tiene una distribución normal o Gaussiana, con parámetros μ_x y σ_x. Si los parámetros de la distribución normal tienen los valores $\mu_x = 0$ y $\sigma_x = 1$, la función de distribución recibirá el nombre de *distribución normal estándar*, con función de densidad

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, \quad z \in \mathbb{R} \tag{18}$$

Cualquiera distribución normal se puede convertir a la forma estándar, mediante la siguiente substitución:

$$z = \frac{x - \mu_x}{\sigma_x} \tag{19}$$

A continuación se muestra la conocida gráfica con su forma de campana, de la función de densidad normal. Existen varios métodos para generar en una computadora, valores de variable aleatoria distribuidos en forma normal. Debido a su popularidad solo discutiremos el procedimiento llamado del límite central.

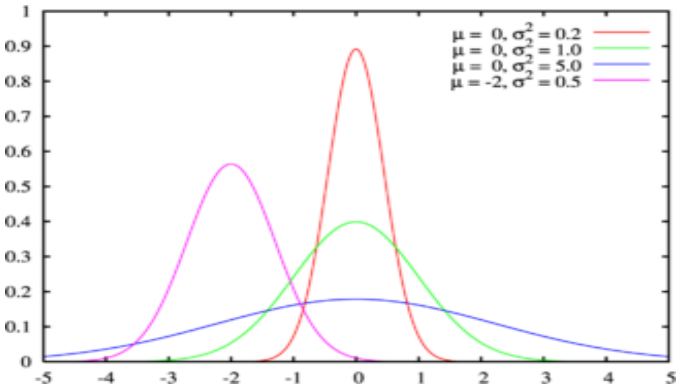


Figura 5: Distribución normal

5.4.1. *Procedimiento del Límite Central*

El procedimiento para simular valores normales utilizando computadoras requiere el uso de la suma de K valores de variable aleatoria distribuidos uniformemente; esto es, la suma de $r_1, r_2, ..., r_K$, con cada r_i definida en el intervalo $0 < r_i < 1$. Aplicando la convención notacional de la forma matemática del teorema del límite central, así como nuestros conocimientos previos de la distribución uniforme, encontramos que

$$\theta = \frac{a+b}{2} = \frac{0+1}{2} = \frac{1}{2}, \tag{20}$$

$$\sigma = \frac{b-a}{\sqrt{12}} = \frac{1}{\sqrt{12}} \tag{21}$$

$$z = \frac{\sum_{i=1}^{K} r_i - \frac{K}{2}}{\sqrt{\frac{K}{12}}} \tag{22}$$

Pero, por definición, z es un valor de variable aleatoria con distribución normal estándar que se puede escribir en la forma sugerida por la ecuación 19, donde x es un valor de variable aleatoria distribuido en forma normal que se va a simular, con media μ_x y variancia σ_x^2. Igualando las ecuaciones 22 y 19 y resolviendo para x, se tiene que

$$x = \sigma_x \left(\frac{12}{K}\right)^{1/2} \left(\sum_{i=1}^{K} r_i - \frac{K}{2}\right) + \mu_x \tag{23}$$

En código Fortran resultaría:

1. SUBROUTINE NORMAL (EX,STDX,X)
2. SUM = 0.0
3. DO 5 I = 1,12
4. R = RND(R)
5. SUM = SUM+R
6. X = STDX * (SUM - 6.0) + EX
7. RETURN

5.4.2. *Procedimiento Directo*

Sean r_1, r_2 valores de variable aleatoria independientes distribuidos de modo uniforme y definidos en el intervalo $(0, 1)$, entonces:

$$x_1 = (-2 \log_e r_1)^{1/2} \cos 2\pi r_2 \tag{24}$$

$$x_2 = (-2 \log_e r_1)^{1/2} \sin 2\pi r_2 \tag{25}$$

serán dos valores de variable aleatoria obtenidos a partir de una distribución normal estándar. Con este método se logran resultados exactos y su velocidad de cómputo se puede comparar con la del método del límite central.

5.5. *Distribución Binomial*

Las variables aleatorias definidas por el número de eventos exitosos en una sucesión de eventos exitosos en una sucesión de n ensayos independientes de Bernoulli, para los cuales la probabilidad de éxito es p en cada ensayo, siguen una distribución Binomial. Este modelo también se puede aplicar al proceso de muestreo aleatorio con reemplazo, cuando los elementos muestreados tienen sólo dos tipos de atributos (por ejemplo sí o no, o respuestas como defectuoso o aceptable). El diseño de una muestra aleatoria de n elementos es análoga a n ensayos independientes de Bernoulli. En los que x es un valor binomial que está denotando al número de elementos de una muestra de tamaño n con atributos idénticos. Es ésta la analogía que sitúa a la distribución Binomial como uno de los modelos más imporantes en las áreas de muestreo estadístico y del control de calidad.

La distribución binomial proporciona la probabilidad de que un evento o acontecimiento tenga lugar x veces en un conjunto de n ensayos, donde la probabilidad de éxito está dada por p. La función de probabilidad para la distribución binomial se puede expresar de la manera siguiente:

$$f(x) = \frac{n!}{x!(n-x)!} p^x q^{n-x} \tag{26}$$

donde x se toma como un entero definido en el intervalo finito $0, 1, 2, ...n$, y al que se le asocia el valor $q = (1 - p)$. El valor esperado y la variancia de la variable binomial X son

$$EX = np \tag{27}$$

$$VX = npq. \tag{28}$$

La segunda expresión implica que la variancia de las variables binomiales siempre tienen un valor menor al de la media. Cuando se conocen la media y la variancia, resulta inmediata la determinación de p y de n, las cuales pueden calcularse como sigue:

$$p = \frac{(EX - VX)}{EX} \tag{29}$$

$$n = \frac{(EX)^2}{(EX - VX)} \tag{30}$$

La distribución normal porporciona, cuando n es muy grande, una buena aproximación para la distribución binomial. Puesto que con la distribución normal resulta posible manipular valores negativos, a fin de hacer un buen uso de tal distribución la probabilidad de registrar observaciones negativas deberá ser despreciablemente pequeña. En la práctica, esto significa que el valor esperado deberá ser por lo menos tres veces meyor que la desviación estándar, o sea

$$np \geq 3(npq)^{1/2} \tag{31}$$

lo cual implica que $n \geq 9\frac{q}{p}$.

Los valores de la variable aleatoria con distribución binomial se pueden generar de muy diversos modos, aunque uno de los métodos más simples, que en el caso de que el valor de n sea moderado resulta uno de los métodos más eficientes, es el basado en la reproducción de los ensayos de Bernoulli, siguiendo el método de rechazos. Este método empieza con valores conocidos de p y de n y consiste en generar n números aleatorios después de fijar $x_0 = 0$.

Para cada número aleatorio $r_i (1 \leq i \leq n)$ se efectúa una prueba y la variable x_i se incrementa de acuerdo con el siguiente criterio:

$$x_i = x_{i-1} + 1, \quad r_i \leq p \tag{32}$$

$$x_i = x_{i-1}, \quad r_i > p \tag{33}$$

Después de haberse generado n números aleatorios, el valor de x_n será igual al valor de la variable aleatoria con distribución binomial x. Este procedimiento se puede repetir tantas veces como valores binomiales se requieran.

En pseudocódigo Fortran resultaría:

```
1. SUBROUTINE BINOM (N,P,X)
2. X = 0.0
3. DO 7 I = 1,N
4. R = RND(R)
5. IF (R - P) 6,6,7
6. X = X + 1.0
7. CONTINUE
8. RETURN
```

5.6. *Distribución de Poisson*

Si tomamos una serie de n ensayos independientes de Bernoulli, en cada uno de los cuales se tenga una probabilidad p muy pequeña relativa a la ocurrencia de un cierto evento, a medida que n tiende a infinito, la probabilidad de x ocurrencias está dada por la distribucipon de Poisson

$$f(x) = e^{-\lambda}\frac{\lambda^x}{x!}; \quad x = 1, 2, ...; \quad \lambda > 0, \tag{34}$$

siempre y cuando permitamos que p se aproxime a cero de manera que se satisfaga la relación $\lambda = np$ consistentenmente. Sabemos que np es el valor esperado de la distribución binomial y se puede demostrar que λ es el valor esperado de la distribución de Poisson. De hecho, tanto el valor esperado como la variancia de la distribución de Poisson coinciden en el valor λ. También se puede demostrar que si x es una variable de Poisson con parámetro λ entonces para valores muy grandes de $\lambda (\lambda > 10)$ se puede utilizar la distribución normal con $EX = \lambda$ y $VX = \lambda$ para aproximar la distribución de x. Los eventos que se distribuyen en forma poissoniana ocurren frecuentemente en la naturaleza; por ejemplo, el número de aeroplanos que descienden en un aeropuerto en un período de veinticuatro horas puede ser considerablemente grande. Aun así, resulta muy pequeña la probabilidad de que un avión aterrice durante un segundo determinado. Por lo tanto, podemos esperar que en un período determinado la probabilidad de que desciendan $0, 1, 2, ...$ aviones, obedecerá a las leyes de la distribución de Poisson. Esta distribución es particularmente útil cuando tratamos con problemas en los que se da la ocurrencia de eventos aislados sobre un intervalo continuo de tiempo, o bien cuando resulta posible prescribir el número de veces que ocurre un evento aunque no el número de veces que no ocurre. Para simular una distribución de Poisson con parámetro λ, nos podemos servir ventajosamente de la relación conocida entre las distribuciones exponenciales y de Poisson. Se puede justificar que si 1) el número total de eventos que ocurren durante un intervalo de tiempo dado es independiente del número de eventos que ya han ocurrido proviamente al inicio del intervalo y 2) la probabilidad de que un evento ocurra en el intervalo de t a $t + \triangle t$ es aproximadamente $\lambda \triangle t$ para todos los valores de t, entonces: a) la función de densidad del intervalo t entre las ocurrencias de eventos consecutivos es $f(t) = \lambda e^{-\lambda t}$,la probabilidad de que ocurran x eventos durante el tiempo t es

$$f(x) = e^{-\lambda t}\frac{(\lambda t)^x}{x!} \tag{35}$$

para toda x y toda t. Considérese un horizonte de tiempo que se inicia en el origen 0 como punto de referencia y que se ha dividido en intervalos unitarios, como se ilustra en la figura de abajo: Supóngase también que los eventos ocurren a lo largo

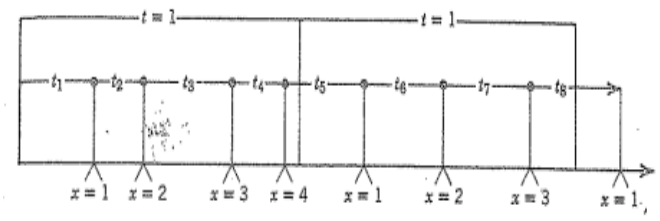


Figura 6: Eventos distribuidos en forma poissoniana, sobre una escala de tiempo

del mencionado horizonte y que se denotan por el símbolo $\wedge$. Se supondrá que el intervalo t entre eventos obedece a una distribución exponencial cuyo valor esperado es igual a $\frac{1}{\lambda}$. Con estos antecedentes se implica que el número de eventos x que ocurren durante un tiempo unitario seguirán una distribución de Poisson cuyo valor esperado estará dada por λ. Un método para generar valores de variable aleatoria con distribución de Poisson deberá considerar la generación de intervalos $t_1, t_2, t_3, ...$, distribuidos en forma exponencial con un valor esperado igual a 1. Una vez generados esos intervalos aleatorios, se acumulan hasta que su suma exceda al valor de λ. En términos matemáticos el valor poissoniano de x se determina haciendo uso de la siguiente desigualdad:

$$\sum_{i=0}^{x} t_i \leq \lambda < \sum_{i-0}^{x+1} t_i \quad (x = 0, 1, 2, ...), \tag{36}$$

donde los valores de la variable aleatoria t_i se generan por medio de la fórmula

$$t_i = -\log r_i \tag{37}$$

con una media unitaria. Un método más rápido para generar los valores poissonianos x es el que consiste en reformular la

ecuación anterior de la manera siguiente:

$$\prod_{i=0}^{x} r_i \geq e^{-\lambda} > \prod_{i=0}^{x+1} r_i \tag{38}$$

En pseudocódigo Fortran resultaría:

1. SUBROUTINE POISSN (P,X)
2. X = 0.0
3. B = EXP (-P)
4. TR = 1.0
5. R = RND (R)
6. TR = TR * R
7. IF (TR - B) 10, 8, 8
8. X = X + 1.0
9. GO TO 5
10. RETURN

5.7. *Distribución Empírica*

En las secciones anteriores nos hemos enfocado en los métodos para generar ciertas distribuciones particulares de prabildiad, como uniforme, exponencial, normal, etc. Ahora trataremos un método más general que puede emplearse para simular cualquiera de las siguientes distribuciones: 1) empírica discreta y 2) continua. En este método generamos valores de variable aleatoria a partir de los datos de una tabla de frecuencias. En un principio veremos cómo se generan los valores para una distribución empírica discreta para luego tratar la simulación de una empírica continua.

5.7.1. *Distribución empírica discreta*

Sea X una variable aleatoria discreta con $P(X = b_i) = p_i$, nuestro método para generar x es aquel que genera un valor de variable aleatoria r sujeto a una distribución uniforme, en el intervalo $(0, 1)$ y un conjunto de valores $x = b_i$ siempre que se satisfaga:

$$p_1 + ... + p_{i-1} < r \leq p_1 + ... + p_i \tag{39}$$

A modo de ejemplo, generamos valores discretos basándonos en la siguiente tabla de frecuencias:

$x = b_i$	$fr = P(x = b_i) = p_i$	$FA = p_1 + ... + p_i$
1	0.10	0.10
2	0.10	0.20
3	0.20	0.40
4	0.20	0.60
5	0.10	0.70
6	0.10	0.80
7	0.05	0.85
8	0.05	0.90
9	0.05	0.95
10	0.05	1.00

Tabla 1: Tabla de frecuencias.

Los valores generados se muestran en el siguiente histograma: y ahora el resultado del método:

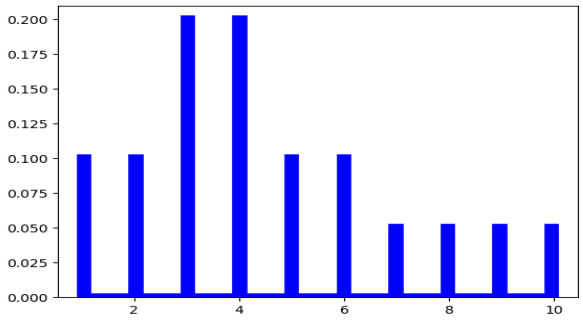


Figura 7: Histograma correspondiente a la tabla

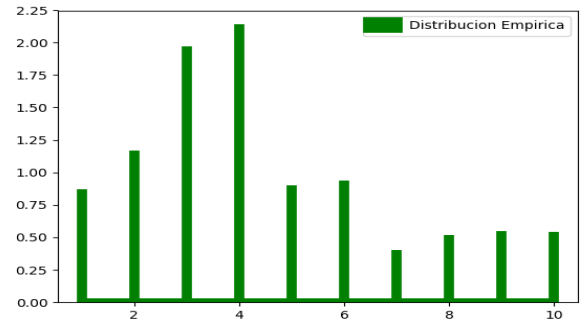


Figura 8: Histograma generado a través del método

5.7.2. *Distribución empírica continua*

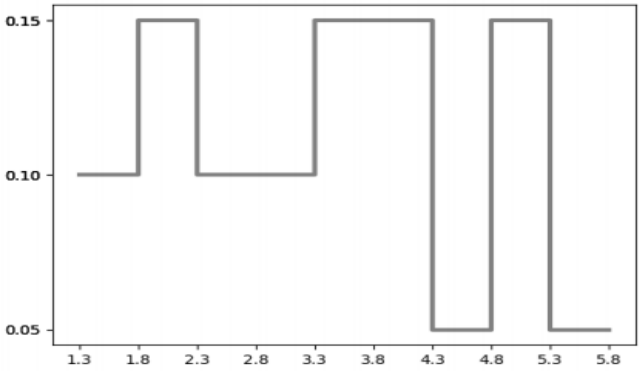
El método para generar valores aleatorios cuando se posee una tabla de frecuencias con intervalos continuos, es muy similar al método utilizado anteriormente para la distribución empírica discreta. La diferencia principal está en que una vez que se encuentra el intervalo en el que cae el número generado, se genere un número aleatorio entre 0 y 1, y se lo utiliza, de la misma manera para conseguir un número aleatorio entre a y b. Siendo a y b el límite inferior y superior, respectivamente, del mencionado intervalo. Supongamos que se posee la siguiente tabla de frecuencias:

Intervalo	Frecuencia Relativa	Frecuencia Acumulada
[1.3 - 1.8)	0.10	0.10
[1.8 - 2.3)	0.15	0.25
[2.3 - 2.8)	0.10	0.35
[2.8 - 3.3)	0.10	0.45
[3.3 - 3.8)	0.15	0.60
[3.8 - 4.3)	0.15	0.75
[4.3 - 4.8)	0.05	0.80
[4.8 - 5.3)	0.15	0.95
[5.3 - 5.8)	0.05	1.00

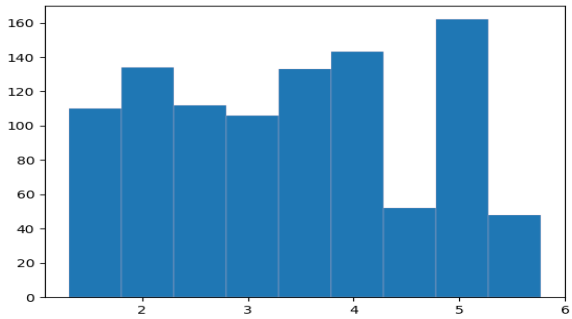
Tabla 2: Tabla de frecuencias.

La gráfica correspondiente a la tabla de frecuencias se mues-

tra a continuación:



Luego el histograma obtenido ejecutando el código que se encuentra en el anexo:



6. Metodología

Hemos desarrollado, mediante el uso del lenguaje Python y varias librerías (scipy.stats,matplotlib, numpy, etc.), códigos capaces de generar valores de variables aleatorias que respondan a las distribuciones mencionadas anteriormente. Se podrá observar, además, la comparación entre los generadores "teóricos"de valores de variables aleatorias y los propios de las funciones del lenguaje.

7. Conclusiones

7.1. Distribución Uniforme

En las pruebas en esta distribución podemos observar cambios relativamente pequeños. Los valores del intervalo que utilizamos fueron $(2, 5)$. El histograma quedó de la siguiente manera:

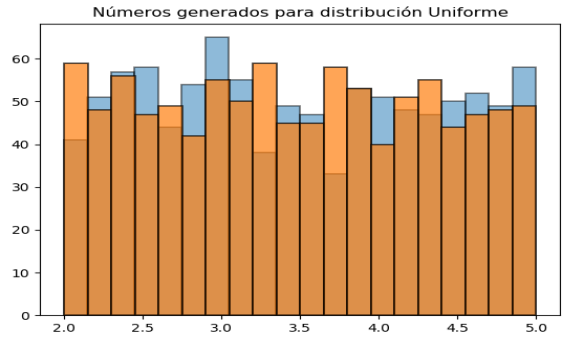


Figura 9: Distribución Uniforme

Podemos observar que los bastones celestes, producidos por nuestro propio código, difieren muy poco con respecto a los bastones naranjas, producidos por la función propia de Python.
Con respecto a los valores de la media, la esperanza y la varianza, son los siguientes:
Esperanza teórica: 3.49
Varianza teórica: 0.77
Desvío Estándar teórico: 0.88
Esperanza de Python: 3.48
Varianza de Python: 0.75
Desvío Estándar de Python: 0.87
Como podemos observar las diferencias son muy mínimas, lo que podemos concluir que las pruebas en ambos códigos fueron satisfactorias. Que ambos son lo suficientemente buenos para utilizarlos dentro de pruebas de muestreo.

7.2. Distribución Exponencial

En las pruebas de esta distribución también obtuvimos valores cercanos pero con diferencias más amplias. El valor utilizado es $\alpha = 1$.

El histograma resultó como se muestra en la siguiente imagen.
Los bastones azules representan nuestro propio código, mientras que los naranjas son los representados por la función de

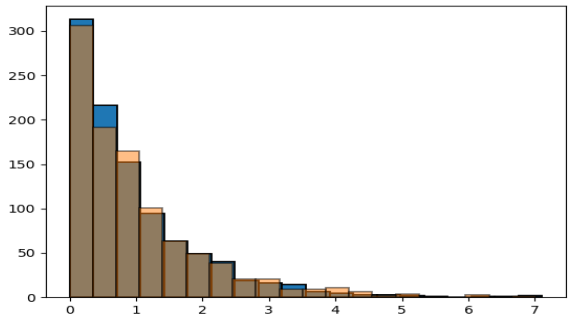


Figura 10: Distribución Exponencial

Python. Tomamos el valor de Alpha como 1, ya que para cualquier otro valor, el histograma realizado por Python, alejaba demasiado ambos histogramas. La razón de este problema es que la función de Python no deja incorporar valores de Alpha, toma por defecto el valor de este en 1. Provocando una diferencia, no solo gráfica, sino también en los valores obtenidos, lo que no nos permitía poder realizar buenas observaciones.
Con respecto a los valores de la media, la esperanza y la varianza, son los siguientes:
Esperanza teórica: 0.96
Varianza teórica: 0.96
Desvío Estándar teórico: 0.98
Esperanza de Python: 1.00
Varianza de Python: 1.02
Desvío Estándar de Python: 1.01
Como podemos observar, estas diferencias entran en un intervalo de (0.03, 0.06). Podemos concluir que utilizando un Alpha=1, podemos obtener valores muy cercanos, pudiendo utilizar estas funciones para muestreos posteriores.

7.3. Distribución Gamma

En las pruebas de esta distribución vamos a observar una gran variedad de cambios. Los valores de los eventos fueron $k = 3$ y $\alpha = 2$. El histograma quedó de la siguiente manera:

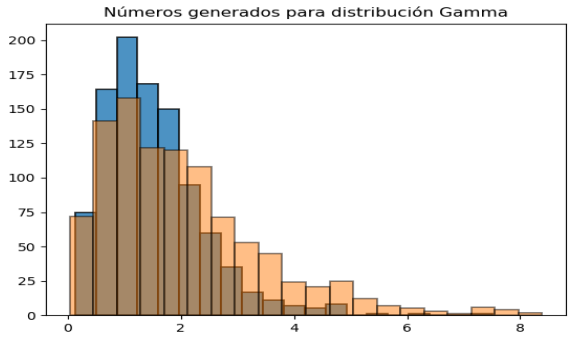


Figura 11: Distribución Gamma

Donde, nuevamente, los bastones azules representan nuestro código, mientras que los naranjas representan los generados

por la función de Python.
Durante las pruebas en esta distribución tuvimos un percance parecido al de la función Exponencial. El problema se presentó al momento de los eventos, ya que la función propia de Python para esta distribución solo dejaba incorporarle el valor α, quitándonos la posibilidad de indicarle la cantidad de eventos requerida.

Luego de un par de pruebas, pudimos indicar, por lo visto en distintas gráficas, que el valor de $k = 3$ utilizado en nuestro código, era muy similar al generado por Python, para el mismo α en ambos. Por lo que concluimos realizar el gráfico con los valores de k y α, previamente nombrados.

Con respecto a los valores de la media, la esperanza y la varianza, son los siguientes:

Esperanza teórica: 1.51
Varianza teórica: 0.77
Desvío Estándar teórico: 0.88
Esperanza de Python: 2.02
Varianza de Python: 2.05
Desvío Estándar de Python: 1.43

Como podemos observar, los valores anteriores para ambos códigos difieren en números relativamente grandes. Lo que podemos concluir que para utilizar la distribución Gamma, para un futuro muestreo, sería conveniente utilizar la función generada por nosotros, ya que podríamos indicarle la cantidad de eventos que de verdad necesitáramos, sin tener que limitarnos.

7.4. *Distribución Normal*

En las pruebas de esta distribución, tenemos muchas variantes a observar. Vamos a analizar distintos casos. Los valores utilizados fueron una media de 0 y una desviación de 2. Tanto las gráficas generadas por nuestro propio código como las de Python tienen los mismos valores. Empezaremos mostrando la gráfica de la Normal generada por Python:

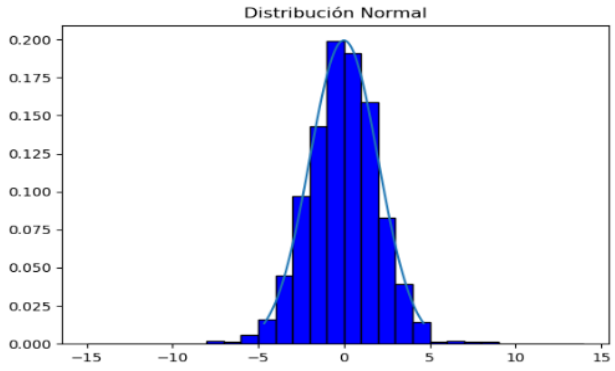


Figura 12: Distribución Normal generada con Python

Proseguimos con la gráfica de nuestro código. Acá tenemos dos casos, hemos generado una gráfica utilizando el valor predeterminado de $K = 12$ y otra cambiando este mismo valor por 24.

Gráfica con $K = 12$:

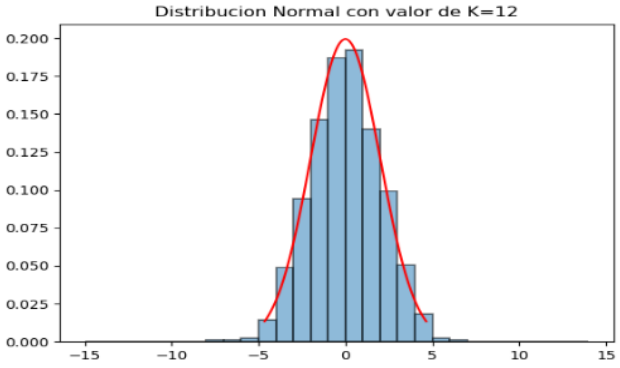


Figura 13: Distribución Normal K=12

Gráfica con $K = 24$:

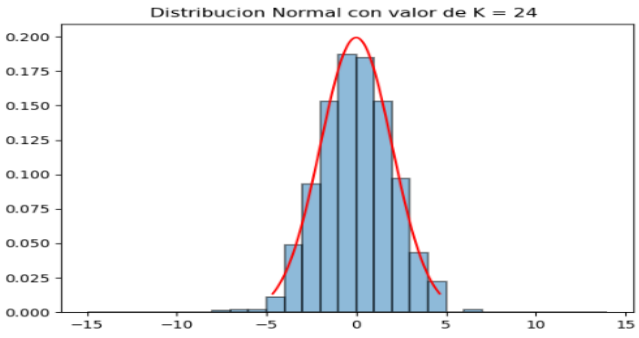


Figura 14: Distribución Normal K=24

Obviamente, al tener las tres gráficas la misma media y desviación, sólo observamos variaciones en los valores aleatorios generados.

Por un lado, la gráfica generada por Python utiliza, técnicamente, un valor de K que desconocemos. Mientras que en nuestro código variamos el valor de K para poder observar y analizar cómo cambia la gráfica.

En ambas gráficas (K valiendo 12 y 24) tenemos cambios relativamente pequeños, tomando solo los valores aleatorios. Por último, también realizamos la gráfica de la Normal que obtiene dos valores utilizando dos valores aleatorios y las funciones seno y coseno. El llamado *Procedimiento Directo*.

La gráfica quedó de la siguiente manera:

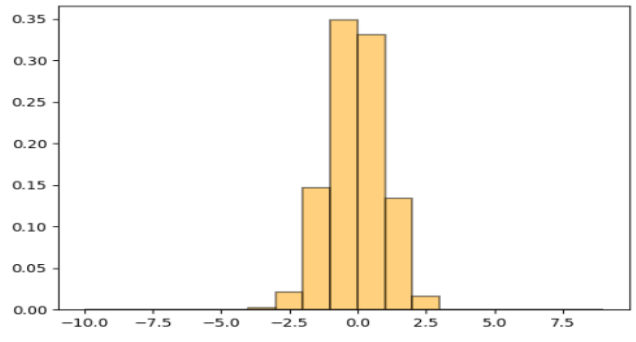


Figura 15: Distribución Normal con Procedimiento Directo

Esta gráfica es la que más distinta está con respecto a las demás. Y tiene su significado: los valores obtenidos de la media y la desviación son completamente distintos a los valores de las demás gráficas. A las otras tres gráficas se les asignó estos valores, mientras que en esta última, los calculamos con los valores obtenidos de las variables aleatorias. Estos valores fueron:

Media: -0.05

Varianza: 1.00

Desviación Estándar: 1.00

Que aunque no varíen en un intervalo muy grande, hace que las gráficas tengan una variación bastante considerable.

7.5. *Distribución Binomial*

En las pruebas de esta distribución notamos, nuevamente, valores aproximados. Los parámetros utilizados fueron n=15 (cantidad de eventos) y una probabilidad de éxito p=20El histograma, junto también con la función de densidad, quedó de la siguiente manera:

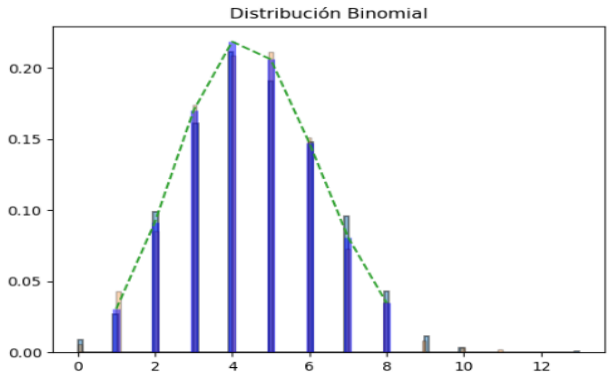


Figura 16: Figura 13: Distribución Binomial

Los cambios en el histograma son mínimos respecto a ambos códigos (azul nuestro, naranja Python). En esta distribución tanto como en la de Poisson, utilizamos un histograma con bastones para identificar mejor la variable aleatoria discreta. Con respecto a los valores de la esperanza, la varianza y la desviación, son los siguientes:

Media teórica: 4.55

Varianza teórica: 3.44

Desviación estándar teórica: 1.85

Media de Python: 4.45

Varianza de Python: 3.24

Desviación estándar de Python: 1.80

Como podemos observar, los cambios en las estadísticas de ambas son bastante cercanas, con un intervalo de diferencia de $(0,05; 0,20)$ siendo aún buenas estimaciones.

7.6. *Distribución de Poisson*

En las pruebas de esta distribución notamos cambios pequeños en ambos histogramas.

El valor de λ es 5. El histograma, junto con la función de densidad, quedó de la siguiente manera:

Los bastones celestes, producidos por nuestro código, y los naranjas, producidos por Python, difieren en muy pocos valores,

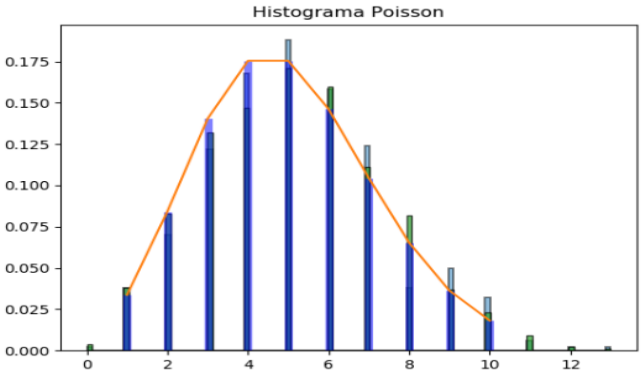


Figura 17: Figura 14: Distribución Poisson

siendo mayores los primeros en ciertas partes, y mayores los segundos en otra.

Con respecto a los valores de la esperanza, varianza y desviación, son los siguientes:

Esperanza teórica: 5.16

Varianza teórica: 4.96

Desviación estándar teórica: 2.23

Esperanza de Python: 5.13

Varianza de Python: 5.11

Desviación estándar de Python: 2.26

Con respecto a estos valores, podemos concluir que sus diferencias son mínimas. Ambos códigos han sido satisfactorios en nuestras pruebas y cumplen perfectamente para un muestreo posterior.

7.7. *Distribución empírica discreta*

Para verificar el grado de similitud entre los valores de la tabla y los generados por computadora procederemos a realizar una analogía visual y de los estadísticos de las mismas.

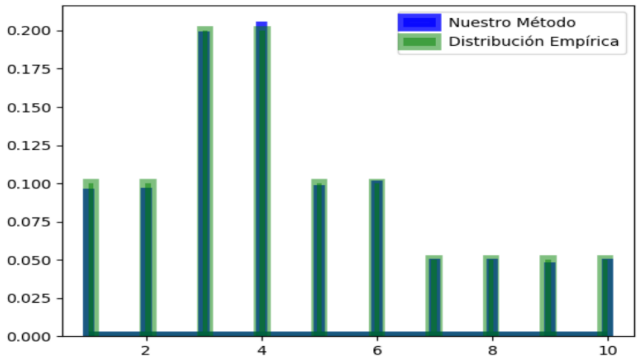


Figura 18: Comparación de ambos histogramas

Tabla de Frecuencias	Distribución Generada
$\overline{x} = 4{,}5$	$\overline{x} = 4{,}481$
$s^2 = 6{,}111$	$s^2 = 6{,}118$

Tabla 3: Tabla comparativa.

Realizando una análisis de las comparativas se observa que existe una gran similitud a nivel gráfico y en relación de los valores de las medias y varianzas muestreales obtenidas. A partir de esto podemos concluir que el método utilizado para generar valores aleatorios empíricos es sumamente confiable.

7.8. Distribución empírica continua

Para verificar el grado de similitud entre los valores de la tabla y los generados por computadora procederemos a realizar una analogía visual y de los estadísticos de las mismas.

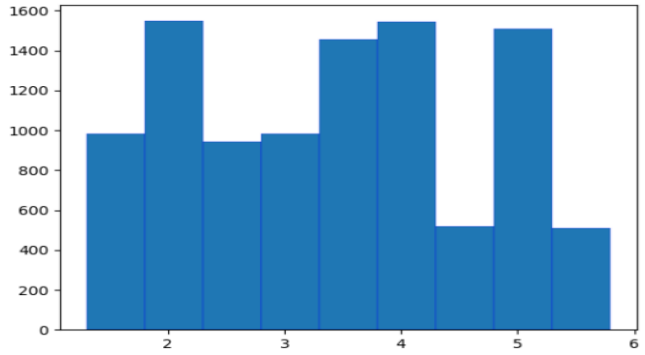


Figura 19: Histograma generado

Tabla de Frecuencias	Distribución Generada
$\overline{x} = 3{,}425$	$\overline{x} = 3{,}435$
$s^2 = 1{,}472$	$s^2 = 1{,}501$

Tabla 4: Tabla comparativa.

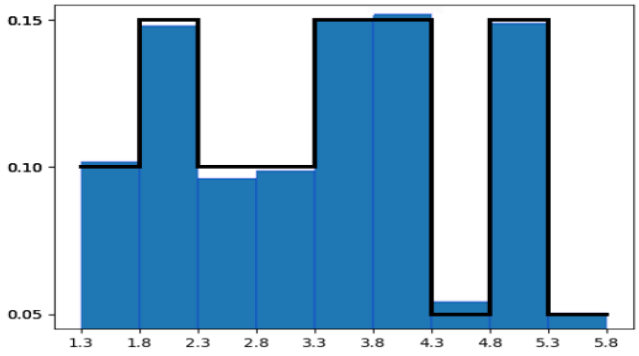


Figura 20: Gráfica comparativa

Realizando un análisis visual entre ambas gráficas, podemos observar que hay una gran similitud. Además, la media y la varianza muestrales obtenidas también tienen un gran parecido.

Se puede concluir que, tanto el método utilizado para generar valores aleatorios empíricos discretos, como el método utilizado recientemente para generar valores aleatorios empíricios discretos son sumamente confiables.

Parte IV
GCL

8. Generadores Congruenciales Lineales - GCL

1. Aplicar el método del cuadrado de los medios con un $Z_0 = 1009$ (en papel).
2. Generar los valores U_1 a U_20 para identificar un comportamiento significativo ($Z_0 = 7, m = 16, a = 5, c = 3$).
3. Chequear si se comprueba el teorema para el ejercicio anterior.
4. Cambiar los valores de a y c ($a = 4yc = 3; a = 5yc = 4; a = 4yc = 4$) generar los valores de U_i con estas 3 alternativas y comprobar que $p = m$ y además el teorema no se cumple.
5. Implementar un GCL en Python con parámetros conocidos.

Ejercicio 1

i	Z_i	U_i	Z_i^2
0	1009	0,1009	01018081
1	0180	0,0180	00032400
2	0324	0,0324	00104976
3	1049	0,1049	01100401
4	1004	0,1004	01008016
5	0080	0,0080	00006400
6	0064	0,0064	00004096
7	0040	0,0040	00001600
8	0016	0,0016	00000256
9	0002	0,0002	00000004
10	0000	0,0000	00000000

Tabla 5: Resultados obtenidos del método “cuadrado de los medios”.

No podemos utilizar este método con el valor $Z_0 = 1009$ ya que no cumple con la propiedad de uniformidad en el intervalo $(0, 1)$.

Ejercicio 2

Codificación en Python:

```
z0 = 7
m = 16
a = 5
c = 3

x = []
for i in range(20):
    z = (a*z0+c)%m
    x.append(z)
    z0 = z

print(x)
```

Los resultados obtenidos son: [6, 1, 8, 11, 10, 5, 12, 15, 14, 9, 0, 3, 2, 13, 4, 7, 6, 1, 8, 11]

Ejercicio 3

Para que se cumpla el teorema, se deben dar las 3 condiciones simultáneamente:

- El único entero que divide a m y c en forma simultánea (primos entre sí) es 1.
- Si q es un número primo que divide a m, entonces q divide a $a - 1$.
- Si 4 divide a m, entonces 4 divide a $a - 1$.

Las tres condiciones se cumplen. En la primera, el único número que divide a 3 y a 16 de forma simultánea (primos entre sí) es el 1. En la segunda, con un $q = 2$ se verifica la condición (2 es primo y a su vez divide a $m = 16$ y también a $a - 1 = 4$). Mientras que la tercera condición también se cumple ya que 4 divide a $m = 16$ y a $a - 1 = 4$. Además de que es un GCL de período completo (los números repetidos son $0, 1, 8, 11$, y al restarlos de los 20 generados, obtenemos 16 que es igual a m).

Ejercicio 4

A) [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]

B) [7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]

C) [0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]

A) No es un GLC de período completo ya que $p = 1$ y $m = 16$. No cumple con el teorema ya que la segunda y tercera condición no se satisfacen.

B) No es un GLC de período completo ya que $p = 1$ y $m = 16$. No cumple con el teorema ya que no cumple la primera condición.

C) No es un GLC de período completo ya que $p = 2$ y $m = 16$. No cumple con el teorema ya que ninguna de las condiciones se satisface.

Ejercicio 5

El GCL fue generado en el apartado 2.

Parte V

Pruebas de independencia y aleatoriedad

9. Tests de Chi cuadrado y de Corridas

El test de **Chi cuadrado** (χ^2) se considera una prueba no paramétrica que mide la discrepancia entre una distribución observada y otra teórica (bondad de ajuste), indicando en qué medida las diferencias existentes entre ambas, de haberlas, se deben al azar en el contraste de hipótesis. También se utiliza para probar la independencia de dos variables entre sí, mediante la presentación de los datos en tablas de contingencia.

Existen una serie de pasos a seguir para desarrollar el algoritmo capaz de realizar este test. Los pasos son los siguientes:

1. Dividir el intervalo $(0, 1)$ en k subintervalos de igual longitud con $k \geq 5$.
2. Generar U_i con $i = 1, ..., n$ con n determinado de manera que $\frac{n}{k} \geq 5$.
3. Calcular f_j donde f_j es la frecuencia absoluta de la cantidad de U_i que cayeron en el intervalo j.
4. Calcular la variable de Chi Cuadrado $\chi^2 = \frac{k}{n} \sum_{j=1}^{k} (f_j - \frac{n}{k})^2$.
5. Comparar $\chi^2 > \chi^2_{k-1,1-\alpha}$.
   Donde $\chi^2_{k-1,1-\alpha}$ es el valor de tabla de una Chi Cuadrado con $k-1$ grados de libertad y $1-\alpha$ de nivel de confianza.
6. Si la desigualdad se comprueba, entonces se rechaza la hipótesis nula (H_0) de que los números aleatorios se comportan uniformes (es decir que no son uniformes).

Un **test de Corridas** es un método que nos ayuda a evaluar el carácter de aleatoriedad de una secuencia de números estadísticamente independientes y números uniformemente distribuidos. Es decir dado una serie de números determinar si son o no aleatorios.

Específicamente, a este test que desarrollamos se lo denomina **"Prueba de corridas arriba y abajo para números estadísticamente independientes"**.
Si tenemos una secuencia de números de tal manera que a cada uno de los números siga otro mayor la secuencia dada será ascendente (*arriba*).Si cada número va seguido por otro menor, la secuencia será descendente (*abajo*).

Pasos para evaluar una prueba de corridas:

1. Primeramente le asignaremos un signo a cada número de la secuencia ya sea + o − , eso dependerá de lo siguiente.
2. Si a un número le sigue otro mayor, se le asigna +. Esto es si $X_i < X_{i+1}$ el signo asignado será (+). Siendo X_i un número de la muestra o secuencia de números.
3. Si el número siguiente es menor, se le da un signo −. Esto es si $X_i > X_{i+1}$ el signo asignado será (−).
4. Se continuará con la comparación de los números y la asignación de su signo correspondiente hasta $N - 1$. Es decir hasta el penúltimo número de la secuencia, ya que al último número le sigue un evento nulo (no es posible compararlo con otro número).

Una vez encontrados los signos de cada número de la secuencia dada se procede a calcular el total de corridas que resulta de la suma de suma de corridas ascendentes con las descendentes. Una corrida se define como una sucesión de eventos similares, precedidos y seguidos por un evento diferente.

Tomando como ejemplo que $a = 33$ es el número total de corridas en una secuencia y $N = 50$ el tamaño de la muestra. La media μ_a y la varianza σ_a^2 están dadas por:

$$\mu_a = \frac{2N-1}{3} = \frac{2(50)-1}{3} = 33 \tag{40}$$

$$\sigma_a^2 = \frac{16N-29}{90} = \frac{16(50)-29}{90} = 8,57 \tag{41}$$

Para $N > 20$, es posible aproximarse razonablemente a la distribución de a mediante una distribución normal con la media y la varianza que se dan en las anteriores ecuaciones.

Por lo común, esa aproximación sería apropiada para comprobar la aleatoriedad de los números generados por un generador de números aleatorios, puesto que se pueden producir varios centenares de números antes de aplicar una prueba.
Podemos rechazar una hipótesis de que una secuencia de números es aleatoria, porque hay un número excesivo o demasiado bajo de corridas. Por ende se requiere una prueba de colas para determinar si se ha presentado alguno de esos extremos. Como estadística de la prueba utilizaremos:

$$Z = \frac{a_1 - \mu_a}{\sigma_a} \tag{42}$$

H_0 **Hipótesis nula:**criterio de aceptación $|Z| \leq Z_{1-\alpha/2}$. La secuencia de números es independiente y por lo tanto la secuencia es aleatoria.

H_1 **Hipótesis alternativa:** criterio de rechazo $|Z| > Z_{1-\alpha/2}$. La secuencia de números **NO** es independiente y por lo tanto la secuencia **NO** es aleatoria.

Sustituyendo la media μ_a y el desvío estándar σ_a, tenemos que:

$$Z = \frac{a - [(2N-1)/3]}{\sqrt{(16N-29)/90}} \tag{43}$$

$$Z = \frac{33-33}{\sqrt{8,57}} = 0 \tag{44}$$

Si se define el nivel de significancia por medio de $\alpha = 0{,}05$, entonces $Z_{1-\alpha/2}$ será igual a $1 - 0,05/2 = 0,975$, buscando este valor en las tablas de la distribución normal encontramos que tiene un valor de $1,96$ entonces, si el valor absoluto de Z calculada es mayor o igual a la Z de las tablas se rechazará la hipótesis de la independencia de los números (propiedad de los números pseudoaleatorios). Esto es:

$$Z_{calculada} = 0 < Z_{0,975} = 1,96 \tag{45}$$

Estaremos rechazando la hipótesis de que los números dados no son estadísticamente independientes. Debido a la falsedad de la comparación llegamos a la aceptación de la hipótesis alternativa.

9.1. Enunciado

1. Aplicar el test de chi-cuadrado a 10000 valores generados por la función aleatorio().
2. Idem anterior a valores generados por el GCL implementado en punto anterior.
3. Idem anterior a los valores generados por Random.org.
4. Aplicar el test de corridas a 10000 generados por la función aleatorio().
5. Idem anterior a valores generados por el GCL implementado en punto anterior.
6. Idem anterior a los valores generados por Random.org.

Toda la codificación correspondiente se ha realizado en Python y podrá observarla en el Anexo al final del documento.

1. Los resultados obtenidos al correr el programa, fueron los siguientes:

   f1= 7 f2= 2 f3= 1 f4= 9 f5= 6
   χ^2 = 2.8000000000000003
   valorTabla = 7.779440339734858

   Se concluye que: Son uniformes.
   Dependiendo de cada corrida del programa, las muestras inevitablemente varían por lo que se obtienen distintos valores de χ^2.

2. Los resultados obtenidos al correr el programa, fueron los siguientes:

   f1= 3 f2= 8 f3= 6 f4= 4 f5= 4
   χ^2 = 3.2
   valorTabla = 7.779440339734858

   Se concluye que: Son uniformes.

3. Los resultados obtenidos al correr el programa, fueron los siguientes:

   f1= 5 f2= 6 f3= 3 f4= 5 f5= 6
   χ^2 = 1.2000000000000002
   valorTabla = 7.779440339734858

   Se concluye que: Son uniformes.

4. Los resultados obtenidos al correr el programa, fueron los siguientes:

   Total Cadenas: 6675
   Media: 6666.333333333333
   Varianza: 1777.4555555555555
   Desvío: 42.159880876913725
   Z calculado: 0.20556667823539176
   Z en 1 - (alpha/2): 1.959963984540054

   La secuencia de números es independiente y por lo tanto la secuencia es aleatoria.

5. Los resultados obtenidos al correr el programa, fueron los siguientes:

   Total Cadenas: 6250
   Media: 6666.333333333333
   Varianza: 1777.4555555555555
   Desvío: 42.159880876913725
   Z calculado: 9.875106965999812
   Z en 1 - (alpha/2): 1.959963984540054

   La secuencia de numeros NO es Independiente y por lo tanto la secuencia NO es aleatoria.

6. Los resultados obtenidos al correr el programa, fueron los siguientes:

   Total Cadenas: 6678
   Media: 6666.333333333333
   Varianza: 1777.4555555555555
   Desvío: 42.159880876913725
   Z calculado: 0.27672437454764026
   Z en 1 - (alpha/2): 1.959963984540054

   La secuencia de numeros es independiente y por lo tanto la secuencia es aleatoria.

   En conclusión, se puede observar, habiendo aplicado el test de Corridas, que los valores obtenidos del sitio web random.org son los mejores y que realmente son aleatorios como predican.

Parte VI

Anexo

10. Anexo

10.1. Codificación - Uniforme

```
import random
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt

a = float(input("Ingresa \"a\": "))
b = float(input("Ingresa \"b\": "))

#con formula teorica
x = []
for i in range(1000):
    r = random.random()
    x.append((a + (b - a) * r))
print("la media es: ",np.mean(x))
print("la varianza es: ",np.var(x))
plt.title("Distribucion Uniforme")
plt.hist(x, bins=40, alpha=0.5,
edgecolor="black", linewidth=1.3)
plt.show()

#con la funcion incluida en python
a = float(input("Ingresa \"a\": "))
b = float(input("Ingresa \"b\": "))

x = []
for i in range(1000):
    x.append(random.uniform(a,b))
print("la media es: ",np.mean(x))
print("la varianza es: ",np.var(x))
plt.title("Distribucion Uniforme")
plt.hist(x, bins=40, alpha=1,
edgecolor="black", linewidth=1.3)
plt.show()

#comparando de ambos metodos
a = float(input("Ingresa \"a\": "))
b = float(input("Ingresa \"b\": "))
print('\n','\n')
x = []
for i in range(1000):
    r = random.random()
    x.append((a + (b - a) * r))
print("Esperanza teorica: ",
"%.2f" %np.mean(x))
print("Varianza teorica: ",
"%.2f" %np.var(x))
print("Desvio Estandar teorico: ",
"%.2f" %sqrt(np.var(x)),
'\n','\n')

y = []
for i in range(1000):
    y.append(random.uniform(a,b))
print("Esperanza de Python: ",
"%.2f" %np.mean(y))
print("Varianza de Python: ",
"%.2f" %np.var(y))
print("Desvio Estandar de Python: ",
"%.2f" %sqrt(np.var(y)))
plt.title("Distribucion Uniforme")
plt.hist(x, bins=20, alpha=0.5,
edgecolor="black", linewidth=1.3)
plt.hist(y, bins=20, alpha=0.7,
edgecolor="black", linewidth=1.3)
plt.show()
```

10.2. Codificación - Exponencial

```
import random
import matplotlib.pyplot as plt
import numpy as np
import math as m
import scipy.stats as sp

print("Dist. Exponencial")
alpha = 0
while alpha <= 0:
    alpha = float(input("\"alpha\": "))

#con formula teorica
x = []
for i in range(1000):
    r = random.random()
    x.append(((-1/alpha)*m.log(r)))
#print("lista:",x)
print("la media es: ",np.mean(x))
print("la varianza es: ",np.var(x))
plt.title("Distribucion Exponencial")
plt.hist(x, bins=40, alpha=0.5,
edgecolor="black", linewidth=1.3)
plt.show()


#con funcion de Python
print("Dist. Exponencial")
alpha = 0
while alpha <= 0:
    alpha = float(input("\"alpha\": "))

x = []
for i in range(1000):
    x.append(sp.expon(alpha).rvs())
print("la media es: ",np.mean(x))
print("la varianza es: ",np.var(x))
plt.title("Distribucion Exponencial")
plt.hist(x, bins=40, alpha=0.5,
edgecolor="black", linewidth=1.3)
plt.show()


#comparacion de ambos histogramas
print("Dist. Exponencial")
alpha = 0
while alpha <= 0:
    alpha = float(input("\"alpha\": "))

x = []
for i in range(1000):
    r = random.random()
    x.append(((-1/alpha)*m.log(r)))
print("Esperanza teorica: ",
"%.2f" %np.mean(x))
print("Varianza teorica: ",
"%.2f" %np.var(x))
print("Desvio Estandar teorico: ",
"%.2f" %m.sqrt(np.var(x)),
'\n','\n')

y = sp.expon().rvs(1000)
print("Esperanza de Python: ",
"%.2f" %np.mean(y))
print("Varianza de Python: ",
"%.2f" %np.var(y))
print("Desvio Estandar de Python: ",
"%.2f" %m.sqrt(np.var(y)))
plt.hist(x, bins=20, alpha=1,
edgecolor="black", linewidth=1.3)
plt.hist(y, bins=20, alpha=0.5,
edgecolor="black", linewidth=1.3)
plt.show()
```

10.3. Codificación - Gamma

```
import random
import matplotlib.pyplot as plt
import numpy as np
import math as m
import scipy.stats as sp


k = int(input("Ingrese n  k de eventos: "))
a = int(input("Ingrese alpha: "))
x=[]
for i in range(1000):
    tr = 1.0
    for j in range(k):
        r = random.random()
        tr = tr * r
        print(tr)
    x.append(-(m.log(tr))/a)
print("lista:",x)
print("la media es: ",np.mean(x))
print("la varianza es: ",np.var(x))
plt.title("Distribucion Gamma")
plt.hist(x, bins=40, alpha=0.5,
edgecolor="black", linewidth=1.3)
plt.show()



#Comparacion con funcion de Python

a = float(input("alpha: "))
aleatorios = sp.gamma.rvs(a,size=1000)
plt.hist(aleatorios ,bins=40,alpha=1,
edgecolor="black", linewidth=1.3)
plt.ylabel('frecuencia')
plt.xlabel('valores')
plt.title('Histograma Gamma')
plt.show()


#comparacion de ambos histogramas

k = int(input("Ingrese n  k de eventos: "))
a = float(input("Ingrese alpha: "))
x=[]
for i in range(1000):
    tr = 1.0
    for j in range(k):
        tr = tr * random.random()
    x.append(-(m.log(tr))/a)
print("Esperanza teorica: ",
        "%.2f" %np.mean(x))
print("Varianza teorica: ",
        "%.2f" %np.var(x))
print("Desvio Estandar teorico: ",
"%.2f" %m.sqrt(np.var(x)),
'\n','\n')


aleatorios = sp.gamma(a).rvs(size=1000)
print("Esperanza de Python: ",
    "%.2f" %np.mean(aleatorios))
print("Varianza de Python: ",
    "%.2f" %np.var(aleatorios))
print("Desvio Estandar de Python: ",
"%.2f" %m.sqrt(np.var(aleatorios)))
plt.title("Distribucion Gamma")
plt.hist(x,bins=20,alpha=0.8,
edgecolor="black",
linewidth=1.3,color="green")
plt.hist(aleatorios ,bins=20,alpha=0.5,
edgecolor="black",
linewidth=1.3,color="red")
plt.show()
```

10.4. Codificación - Normal

10.4.1. Normal - Procedimiento del límite central

```
import random as ran
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sp
import math as m

#procedimiento del limite central

x=[]
u=0.0
sigma=0.0

u=float(input("Ingrese media: "))
sigma=float(input("Ingrese desviacion: "))

#eligiendo el valor de K resulta...
k = int(input("Ingrese valor de K:"))
for i in range(1000):
    sum = 0.0
    for j in range(k):
        r = ran.uniform(0,1)
        sum += r
    x.append(round(sigma * ((12/k)**(1/2)) *
            (sum - (k/2)) + u ,2))
plt.title(f"Distribucion Normal K = {k} ")
mybins = np.arange(-15,15,1)
plt.hist(x, bins= mybins, alpha=0.5,
weights=np.zeros_like(x)+1./len(x),
edgecolor="black", linewidth=1.3)
x_1 = np.linspace(sp.norm(u, sigma).ppf(0.01),
        sp.norm(u,sigma).ppf(0.99), 1000)
plt.plot(x_1,
        sp.norm(u, sigma).pdf(x_1),
        color="red")
plt.show()


#valor fijo... K = 12

for i in range(1000):
        sum=0.0
        for j in range(12):
                r=ran.uniform(0,1)
                sum=sum + r
        x.append(round(sigma *(sum-6.0) + u , 2))

plt.title("Distribucion Normal K=12")
mybins = np.arange(-15,15,1)
plt.hist(x, bins= mybins, alpha=0.5,
weights=np.zeros_like(x)+1./len(x),
edgecolor="black", linewidth=1.3)
x_1 = np.linspace(sp.norm(u, sigma).ppf(0.01),
        sp.norm(u,sigma).ppf(0.99), 1000)
plt.plot(x_1, sp.norm(u, sigma).pdf(x_1),
        color="red")
plt.show()
```

10.4.2. Normal - Procedimiento Directo

```
import random as ran
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sp
import math as m

#mediante procedimiento directo
x = []
for i in range (1000):
    r1 = ran.uniform(0,1)
    r2 = ran.uniform(0,1)
    x1 = (m.sqrt(-2*m.log(r1,m.e))
            *m.cos(2*m.pi*r2))
    x2 = (m.sqrt(-2*m.log(r1,m.e))
            *m.sin(2*m.pi*r2))
    x.append(x1)
    x.append(x2)

print("La media es: ",
"%.2f" %np.mean(x))
print("La varianza es: ",
"%.2f" %np.var(x))
print("La desviacion estandar es: ",
"%.2f" %m.sqrt(np.var(x)))
mybins = np.arange(-10,10,1)
plt.hist(x, bins=mybins,
weights=np.zeros_like(x)+1./len(x),
alpha=0.5 ,edgecolor='black',
color ="orange",
linewidth = 1.3)
plt.show()
```

10.4.3. Normal - Python

```
import random as ran
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sp
#con la funcion incluida en python

u=float(input("Ingrese media: "))
sigma=float(input("Ingrese desviacion: "))
normal = sp.norm(u, sigma)
x = np.linspace(normal.ppf(0.01),
    normal.ppf(0.99), 100)
data = np.random.normal(loc=u, scale=sigma,
                        size=1000)
mybins = np.arange(-15,15,1)
plt.hist(data,bins=mybins,
weights=np.zeros_like(data)+1./len(data),
edgecolor='black',color='blue')
fp = normal.pdf(x)
plt.plot(x, fp)
plt.title('Distribucion Normal')
plt.show()
```

10.5. Codificación - Binomial

```
import random as ran
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sp
import math as m

#con formula teorica

n=int(input("Ingrese n eventos \"n\": "))
p=float(input("Ingrese la prob. \"p\": "))

x=[]

for i in range(1000):
    prob=0
    for j in range(n):
        r=ran.uniform(0,1)
        if((r-p) <= 0):
            prob+= 1
    x.append(prob)

print("La media es: ",
"%.2f" %np.mean(x))
print("La varianza es: ",
"%.2f" %np.var(x))
print("La desviacion estandar es: ",
"%.2f" %m.sqrt(np.var(x)))

plt.title("Distribucion Binomial")
plt.hist(x, bins=100,
weights=np.zeros_like(x)+1./len(x),
alpha=0.5,edgecolor="black", linewidth=1.3)

#con la funcion incluida en python

valores = sp.binom.rvs(n, p, size=1000)
print("La media es: ",
"%.2f" %np.mean(valores))
print("La varianza es: ",
"%.2f" %np.var(valores))
print("La desviacion estandar es: ",
"%.2f" %m.sqrt(np.var(valores)))
plt.hist(valores, bins=100,
weights=np.zeros_like(x)+1./len(x),
alpha=0.3,edgecolor="black", linewidth=1)
plt.title('Distribuci n Binomial')

b = sp.binom(n, p) # Distribucion
x = np.arange(b.ppf(0.01),
              b.ppf(0.99))
fmp = b.pmf(x) # Funcion de Probabilidad
plt.plot(x, fmp, '--')
plt.vlines(x, 0, fmp, colors='b', lw=5,
           alpha=0.5)
plt.show()
```

10.6. Codificación - Poisson

```
import random as ran
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as sp
import math as m


#con formula teorica

p=float(input("Ingrese el valor de p: "))
x=[]

for i in range(500):
    acu=0
    tr=1
    b=np.exp(-p)
    r=ran.uniform(0,1)
    tr=tr*r
    while((tr-b)>=0):
        acu+= 1
        r=ran.uniform(0,1)
        tr=tr*r
    x.append(acu)

print("La media es: ",
"%.2f" %np.mean(x))
print("La varianza es: ",
"%.2f" %np.var(x))
print("La desviacion estandar es: ",
"%.2f" %m.sqrt(np.var(x)))

plt.title("Distribucion Poisson")
plt.hist(x, bins=100, alpha=0.5,
weights=np.zeros_like(x)+1./len(x),
edgecolor="black", linewidth=1.3)


#con la funcion incluida en python

poisson = sp.poisson(p)
w = np.arange(poisson.ppf(0.01),
              poisson.ppf(0.99))
fmp = poisson.pmf(w)
plt.plot(w, fmp)
plt.vlines(w, 0, fmp, colors='b',
           lw=5, alpha=0.5)



y = np.random.poisson(p, 1000)
plt.hist(y,alpha=0.7, bins=100,
weights=np.zeros_like(y)+1./len(y),
edgecolor='black')
plt.ylabel("frequencia")
plt.xlabel("valores")
```

```
plt.title("Histograma Poisson")
plt.show()

print("La media es: ",
"%.2f" %np.mean(y))
print("La varianza es: ",
"%.2f" %np.var(y))
print("La desviacion estandar es: ",
"%.2f" %m.sqrt(np.var(y)))
```

10.7. Codificación - Empírica

10.7.1. Empírica Discreta

```
import random
import matplotlib.pyplot as plt
import numpy as np

x = [1,2,3,4,5,6,7,8,9,10]
fa = [0.1,0.2,0.4,0.6,0.7,0.8,
        0.85,0.9,0.95,1]

arreglo = []

for i in range(10):
    arreglo.append(1)
    arreglo.append(2)
    arreglo.append(6)
    arreglo.append(5)

for i in range(20):
    arreglo.append(3)
    arreglo.append(4)

for i in range(5):
    arreglo.append(7)
    arreglo.append(8)
    arreglo.append(9)
    arreglo.append(10)

valores = []
for i in range(1000):
    r = random.uniform(0,1)
    c = 0
    b = True
    while (b):
        if (r<=fa[c]):
            valores.append(x[c])
            b = False
        else:
            c += 1

plt.hist(valores,bins=len(valores),
edgecolor='green',linewidth = 7,
color = 'green',
weights=np.zeros_like(valores)+1. /len(arreglo),
label = 'Metodo')
plt.legend()
plt.show()
```

10.7.2. Empírica Continua

```
import random
import matplotlib.pyplot as plt
import numpy as np

z = [1.3,1.8,2.3,2.8,3.3,3.8,4.3,
    4.8,5.3,5.8,6.3]
fra = [0.1,0.25,0.35,0.45,0.6,0.75,
        0.8,0.95,1]
a = 0
b = 0
arreglo = []

for i in range(1000):
    cont = 0
    flag = True

    r1 = random.uniform(0,1)
    while (flag):
        if (r1 >= fra[cont]):
            cont += 1
        else:
            a = z[cont]
            b = z[cont+1]
            flag = False
    r2 = random.uniform(0,1)
    x = a + (b-a)*r2
    arreglo.append(x)

print("E",np.mean(arreglo))
print("V",np.var(arreglo))
plt.hist(arreglo,bins=9,
    linewidth=0.2,edgecolor='pink')
plt.show()
```

10.8. Codificación de Tests (Chi cuadrado y Corridas)

10.8.1. Ejercicio 1

```
import random as rnd
from scipy import stats

m = 10000
aleatorios = []
for i in range(m):
    aleatorios.append(rnd.random())

k = 5
n = 25
confianza = 0.90

u = []
f1 = 0
f2 = 0
f3 = 0
f4 = 0
f5 = 0

for i in range(n):
    p = rnd.randint(0,9999)
    u.append(aleatorios[p])
    if p in range(0,1999):
        f1 += 1
    elif p in range(2000,3999):
        f2 += 1
    elif p in range(4000,5999):
        f3 += 1
    elif p in range(6000,7999):
        f4 += 1
    elif p in range(8000,9999):
        f5 += 1

print("f1=",f1,"f2=",f2,"f3=",f3,"f4="
        ,f4,"f5=",f5)


chicuadrado = (1/5)*((f1-5)**2+(f2-5)**2
                +(f3-5)**2+(f4-5)**2+(f5-5)**2)

print("X^2 = ",chicuadrado)
valorTabla = stats.chi2.ppf(confianza,k-1)
print(valorTabla)

if chicuadrado > valorTabla:
    print("No son uniformes")
else:
    print("Son uniformes")
```

10.8.2. Ejercicio 2

```
from scipy import stats
import random as rnd


z0 = 7
m = 16
a = 5
c = 3

n = 25
k = 5
confianza = 0.90


aleatorios = []
for i in range(10000):
    z = (a*z0+c)%m
    aleatorios.append(z)
    z0 = z

u = []
f1 = 0
f2 = 0
f3 = 0
f4 = 0
f5 = 0

for i in range(n):
    p = rnd.randint(0,9999)
    u.append(aleatorios[p])
    if p in range(0,1999):
        f1 += 1
    elif p in range(2000,3999):
        f2 += 1
    elif p in range(4000,5999):
        f3 += 1
    elif p in range(6000,7999):
        f4 += 1
    elif p in range(8000,9999):
        f5 += 1
print(u)
print("f1=",f1,"f2=",f2,"f3=",f3,"f4="
        ,f4,"f5=",f5)


chicuadrado = (1/5)*((f1-5)**2+(f2-5)**2
            +(f3-5)**2+(f4-5)**2+(f5-5)**2)

print("X^2 = ",chicuadrado)

valorTabla = stats.chi2.ppf(confianza,k-1)
print(valorTabla)

if chicuadrado > valorTabla:
    print("No son uniformes")
else:
    print("Son uniformes")
```

10.8.3. Ejercicio 3

```
from scipy import stats
import random as rnd


f = open('randomOrg.txt',"r")
aleatorios = []
for i in range(10000):
    string = f.readline()
    num = float(string[0:18])
    aleatorios.append(num)

f.close()

n = 25
k = 5
confianza = 0.90

u = []
f1 = 0
f2 = 0
f3 = 0
f4 = 0
f5 = 0

for i in range(n):
    p = rnd.randint(0,9999)
    u.append(aleatorios[p])
    if p in range(0,1999):
        f1 += 1
    elif p in range(2000,3999):
        f2 += 1
    elif p in range(4000,5999):
        f3 += 1
    elif p in range(6000,7999):
        f4 += 1
    elif p in range(8000,9999):
        f5 += 1
print(u)
print("f1=",f1,"f2=",f2,"f3=",f3,"f4="
        ,f4,"f5=",f5)


chicuadrado = (1/5)*((f1-5)**2+(f2-5)**2
        +(f3-5)**2+(f4-5)**2+(f5-5)**2)

print("X^2 = ",chicuadrado)

valorTabla = stats.chi2.ppf(confianza,k-1)
print(valorTabla)

if chicuadrado > valorTabla:
    print("No son uniformes")
else:
    print("Son uniformes")
```

10.8.4. Ejercicio 4

```
import random as rnd
import math as ma
from scipy import stats

m = 10000
aleatorios = []
for i in range(m):
    aleatorios.append(rnd.random())

subcadenas = []

for i in range(0,m-1):
    if aleatorios[i] < aleatorios[i+1]:
        subcadenas.append('+')
    else:
        subcadenas.append('-')

#print(aleatorios,subcadenas)
total_cadenas = 1

ini = subcadenas[0]
for i in range(0,len(subcadenas)-1):
    if ini != subcadenas[i+1]:
        total_cadenas = total_cadenas + 1
        ini = subcadenas[i+1]

print("Total Cadenas: ",total_cadenas)

media = (2*m - 1)/3
varianza = (16*m - 29)/90
desvio = ma.sqrt(varianza)
print("media: ",media)
print("varianza: ",varianza)
print("desvio: ",desvio)

#calculo la Z
z_calculado = ma.fabs((total_cadenas - media)/desvio)
print("Z calculado: ",z_calculado)

alpha = 0.05
c = 1-(alpha/2)
#Z_1-alpha/2
z_alpha2 = stats.norm.ppf(c)
print("Z en 1 - (alpha/2): ",z_alpha2)

if z_calculado <= z_alpha2:
    print("La secuencia de numeros es
     independiente y por lo tanto la
     secuencia es aleatoria")
else:
    print("La secuencia de numeros NO
     es Independiente y por lo tanto
     la secuencia NO es aleatoria")
```

10.8.5. Ejercicio 5

```
from scipy import stats
import math as ma
import random as rnd

z0 = 7
m = 16
a = 5
c = 3

n = 25
k = 5
confianza = 0.90


aleatorios = []
for i in range(10000):
    z = (a*z0+c)%m
    aleatorios.append(z)
    z0 = z


subcadenas = []

for i in range(0,len(aleatorios)-1):
    if aleatorios[i] < aleatorios[i+1]:
        subcadenas.append('+')
    else:
        subcadenas.append('-')

#print(aleatorios,subcadenas)
total_cadenas = 1

ini = subcadenas[0]
for i in range(0,len(subcadenas)-1):
    if ini != subcadenas[i+1]:
        total_cadenas = total_cadenas + 1
        ini = subcadenas[i+1]

print("Total Cadenas: ",total_cadenas)

media = (2*len(aleatorios) - 1)/3
varianza = (16*len(aleatorios) - 29)/90
desvio = ma.sqrt(varianza)
print("media: ",media)
print("varianza: ",varianza)
print("desvio: ",desvio)

#calculo la Z
z_calculado = ma.fabs((total_cadenas - media)/desvio)
print("Z calculado: ",z_calculado)



alpha = 0.05
con = 1-(alpha/2)
#Z_1-alpha/2
z_alpha2 = stats.norm.ppf(con)
print("Z en 1 - (alpha/2): ",z_alpha2)

if z_calculado <= z_alpha2:
    print("La secuencia de numeros es
        independiente y por lo tanto
            la secuencia es aleatoria")
else:
    print("La secuencia de numeros NO
        es Independiente y por lo tanto
            la secuencia NO es aleatoria")
```

10.8.6. Ejercicio 6

```
from scipy import stats
import math as ma
import random as rnd


f = open('randomOrg.txt',"r")
aleatorios = []
for i in range(10000):
    string = f.readline()
    num = float(string[0:18])
    aleatorios.append(num)

f.close()



subcadenas = []

for i in range(0,len(aleatorios)-1):
    if aleatorios[i] < aleatorios[i+1]:
        subcadenas.append('+')
    else:
        subcadenas.append('-')

#print(aleatorios,subcadenas)
total_cadenas = 1

ini = subcadenas[0]
for i in range(0,len(subcadenas)-1):
    if ini != subcadenas[i+1]:
        total_cadenas = total_cadenas + 1
        ini = subcadenas[i+1]

print("Total Cadenas: ",total_cadenas)

media = (2*len(aleatorios) - 1)/3
varianza = (16*len(aleatorios) - 29)/90
desvio = ma.sqrt(varianza)
print("media: ",media)
print("varianza: ",varianza)
print("desvio: ",desvio)

#calculo la Z
z_calculado = ma.fabs((total_cadenas - media)/desvio)
print("Z calculado: ",z_calculado)


alpha = 0.05
con = 1-(alpha/2)
#Z_1-alpha/2
z_alpha2 = stats.norm.ppf(con)
print("Z en 1 - (alpha/2): ",z_alpha2)

if z_calculado <= z_alpha2:
    print("La secuencia de numeros es
        independiente y por lo tanto
        la secuencia es aleatoria")
else:
    print("La secuencia de numeros NO
    es Independiente y por lo tanto
    la secuencia NO es aleatoria")
```