Contents

Clase 2: Generación de Números Aleatorios y Distribuciones de Probabilidad

Objetivos de la clase:

- Comprender la importancia de la generación de números aleatorios en la simulación.
- Conocer y aplicar diferentes métodos para la generación de números aleatorios uniformes.
- Introducir el concepto de distribuciones de probabilidad y su relevancia en la simulación.
- Aprender a generar números aleatorios con distribuciones de probabilidad específicas (discretas y continuas).

Contenido Teórico Detallado:

1. **La Importancia de los Números Aleatorios:**
   - La base de la simulación de Monte Carlo reside en la generación de números aleatorios.
   - La calidad de los números aleatorios afecta directamente la precisión y confiabilidad de los resultados de la simulación.
   - Propiedades deseables de un generador de números aleatorios: uniformidad, independencia, largo período, reproducibilidad y eficiencia.
2. **Generación de Números Aleatorios Uniformes:**
   - **Generadores Congruenciales Lineales (GCL):**
     - Descripción del método: `X_(i+1) = (aX_i + c) mod m`
     - Parámetros clave: módulo (m), multiplicador (a), incremento (c) y semilla (X_0).
     - Consideraciones para la elección de los parámetros (Hull-Dobell Theorem).
     - Ejemplo: Implementación de un GCL simple y cálculo de los primeros números generados.
     - Limitaciones de los GCLs: patrones y autocorrelación.
   - **Otros métodos:** (Mención breve)
     - Registros de desplazamiento de retroalimentación lineal (LFSR).
     - Generadores basados en funciones criptográficas.
3. **Distribuciones de Probabilidad:**
   - Definición de distribución de probabilidad: función que describe la probabilidad de que una variable aleatoria tome un valor específico o caiga dentro de un rango de valores.
   - **Distribuciones Discretas:**
     - Distribución de Bernoulli: Modelado de eventos binarios (éxito/fracaso).
     - Distribución Binomial: Número de éxitos en una secuencia de ensayos independientes de Bernoulli.
     - Distribución de Poisson: Número de eventos que ocurren en un intervalo de tiempo o espacio.
   - **Distribuciones Continuas:**
     - Distribución Uniforme: Todos los valores dentro de un rango tienen la misma probabilidad.
     - Distribución Exponencial: Tiempo entre eventos en un proceso de Poisson.
     - Distribución Normal (Gaussiana): Común en muchos fenómenos naturales.
4. **Generación de Números Aleatorios con Distribuciones Específicas:**
   - **Método de la Transformada Inversa:**
     - Explicación del método: encontrar la inversa de la función de distribución acumulativa (CDF) y aplicar un número aleatorio uniforme a ella.
     - Ejemplo: Generación de números aleatorios exponenciales usando la transformada inversa: `X = - ¹ ln(1 - U)`, donde `U` es un número aleatorio uniforme entre 0 y 1, y es el parámetro de la distribución exponencial.
     - Aplicación a la distribución uniforme.

- **Método de Aceptación y Rechazo:**
  - Descripción del método: generar puntos aleatorios dentro de un área que contiene la distribución deseada y aceptar solo los puntos que caen debajo de la curva de la distribución.
  - Ejemplo: Generación de números aleatorios a partir de una distribución beta.
  - Consideraciones para la elección de la función de propuesta.

Ejemplos y Casos de Estudio:

- **Ejemplo 1: Simulación de una Cola de Espera con Distribución Exponencial.**
  - Simular una cola de espera donde el tiempo entre llegadas de clientes sigue una distribución exponencial.
  - Generar números aleatorios exponenciales utilizando el método de la transformada inversa para simular los tiempos entre llegadas.
  - Generar números aleatorios exponenciales para simular los tiempos de servicio.
  - Analizar el comportamiento de la cola (longitud promedio, tiempo de espera promedio).
- **Ejemplo 2: Simulación del Lanzamiento de una Moneda Sesgada.**
  - Modelar el lanzamiento de una moneda donde la probabilidad de obtener cara es diferente de 0.5.
  - Generar números aleatorios uniformes entre 0 y 1.
  - Si el número aleatorio es menor que la probabilidad de obtener cara, registrar "cara"; de lo contrario, registrar "cruz".
  - Simular un gran número de lanzamientos y calcular la frecuencia de "cara" y "cruz".

Problemas Prácticos y Ejercicios con Soluciones:

- **Problema 1:** Implementar un GCL con parámetros a=1664525, c=1013904223, m=2^32 y semilla X_0 = 12345. Generar los primeros 10 números aleatorios y verificar si parecen uniformemente distribuidos (visualmente).
  - **Solución:** Código en Python: "'python def gcl(a, c, m, x0, n): numbers = [] x = x0 for _ in range(n): x = (a * x + c) % m numbers.append(x / m) # Normalizar a [0, 1] return numbers

  a = 1664525 c = 1013904223 m = 2**32 x0 = 12345 n = 10 random_numbers = gcl(a, c, m, x0, n) print(random_numbers) "'
- **Problema 2:** Generar 1000 números aleatorios con una distribución exponencial con = 0.5 utilizando el método de la transformada inversa. Calcular la media y la desviación estándar de los números generados y compararlos con los valores teóricos (media = 1/ , desviación estándar = 1/).
  - **Solución:** Código en Python:

  "'python import numpy as np import math

  def exponential_inverse_transform(lambd, n): u = np.random.uniform(0, 1, n) x = -1/lambd * np.log(1 - u) return x

  lambd = 0.5 n = 1000 exponential_numbers = exponential_inverse_transform(lambd, n)

  mean = np.mean(exponential_numbers) std = np.std(exponential_numbers)

  print(f"Media: {mean}, Desviación estándar: {std}") print(f"Media teórica: {1/lambd}, Desviación estándar teórica: {1/lambd}") "'
- **Problema 3:** Simular 100 lanzamientos de un dado. Calcular la frecuencia relativa de cada número (1 al 6) y compararlos con la probabilidad teórica (1/6).
  - **Solución:** Código en Python:

  "'python import numpy as np

  def simulate_dice_rolls(n): rolls = np.random.randint(1, 7, n) # Genera enteros aleatorios entre 1 y 6 (inclusive) return rolls

def calculate_frequencies(rolls): frequencies = {} for i in range(1, 7): frequencies[i] = np.sum(rolls == i) / len(rolls) return frequencies

n = 100 rolls = simulate_dice_rolls(n) frequencies = calculate_frequencies(rolls)

print(frequencies) “‘

Materiales Complementarios Recomendados:

- **Libros:**
  - ”Simulation Modeling and Analysis” de Averill M. Law.
  - ”Discrete-Event System Simulation” de Jerry Banks et al.
- **Artículos:**
  - Artículos sobre generadores de números aleatorios y pruebas de aleatoriedad.
- **Recursos en línea:**
  - Documentación de bibliotecas de simulación en Python (SimPy, Scipy).
  - Tutoriales sobre generación de números aleatorios y distribuciones de probabilidad.