FACULTAD DE INGENIERIA
Universidad de Buenos Aires

UBA
Universidad de Buenos Aires

Simulación
(7526-9519)

Trabajo Práctico 1

Grupo 5

Integrantes:

Apellido y nombre	Padrón	Email
Brizueña, María Eva	74620	mebrizue@gmail.com
Giordano, Franco	100608	francogior98@gmail.com
Nitz, Ignacio	100710	nachonitz@gmail.com

Números al azar

Ejercicio 1

Utilizando Matlab, Octave o Python implementar un Generador Congruencial Lineal (GCL) de módulo 2^{32}, multiplicador 1013904223, incremento de 1664525 y semilla igual a la parte entera de la suma ponderada (0,15-0,25-0,6) de los números de padrón de los integrantes del grupo, ordenados ascendentemente.

- Informar los primeros 5 números de la secuencia.
- Modificar el GCL para que devuelva números al azar entre 0 y 1, y realizar un histograma sobre 100.000 valores generados.

Utilizando un GCL dado por la ecuación:

Método Lineal Congruente [Lehmer, 1949]

$$X_{n+1} = (aX_n + c) \bmod m, \qquad n \geq 0.$$

$$\begin{array}{lll} m, & \text{the modulus;} & 0 < m. \\ a, & \text{the multiplier;} & 0 \leq a < m. \\ c, & \text{the increment;} & 0 \leq c < m. \\ X_0, & \text{the starting value;} & 0 \leq X_0 < m. \end{array}$$

Generando los primeros 5 valores de la secuencia se encuentra lo siguiente:

2294318634, 4243656099, 3179884170, 3833243971, 2803027690

Luego se ajustó el generador para arrojar valores entre 0 y 1. Corriendo ahora el algoritmo hasta 100.000 valores se encuentra el siguiente histograma con los resultados:

Se observa que las realizaciones respetan la distribución uniforme esperada, entre 0 y 1.

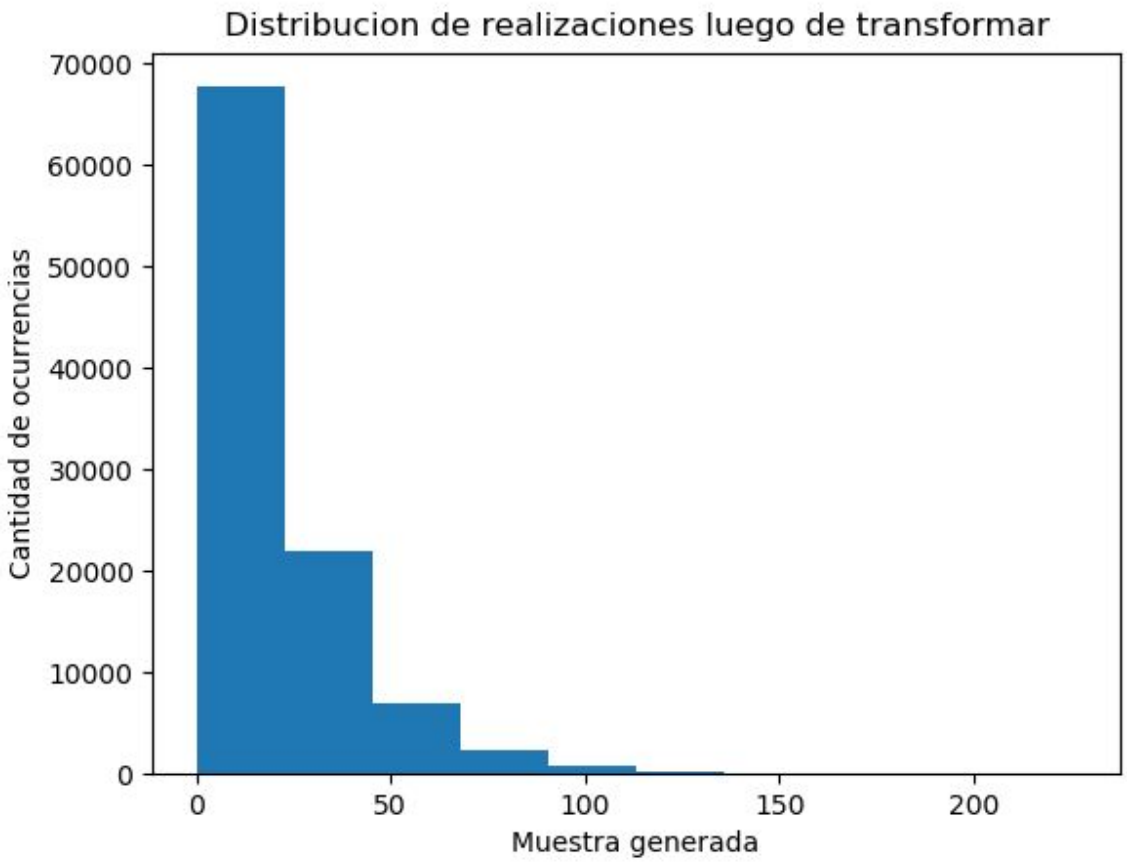
Ejercicio 2

Utilizando el generador de números aleatorios con distribución uniforme [0,1] implementado en el ejercicio 1 y utilizando el método de la transformada inversa genera números pseudoaleatorios con distribución exponencial negativa de media 20.

- Realizar un histograma de 100.000 valores obtenidos.
- Calcular la media y varianza de la distribución obtenida y compararlos con los valores teóricos.

El método de la transformada inversa propone igualar las funciones de probabilidad acumulada de las dos distribuciones que deseo trabajar: la que puedo generar (A) y la que deseo generar (B). Así se encuentra la forma de transformar una realización (de A), de tal forma que la realización transformada actúa como si hubiera sido muestreada de la distribución B.

Con esto en cuenta, se generó una distribución exponencial de parámetro 1/20 a partir de una uniforme 0-1 (con GCL). Graficando los resultados de 100.000 muestras en un histograma:



Como era esperado, se encuentra una tendencia exponencial entre las realizaciones transformadas.

Para verificar que realmente se trate de una exponencial de media 20, se puede estimar la media de las muestras como:

$$\widehat{E[X]} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Siendo $\widehat{E[X]}$ la media estimada, n la cantidad de muestras, y Xi la muestra i.
De esta forma se halló que el parámetro estimado es 20.0505 el cual se aproxima bastante a 20.

Por otro lado se puede hacer un cálculo similar para la varianza:

$$\widehat{\sigma^2} = \frac{n}{\sum_{i=1}^{n} X_i}^2$$

Se encuentra que el valor estimado resulta 0.002487 lo cual se aproxima al esperado: (1/20)^2 = 0.002500

Ejercicio 3

Utilizando el método de Box-Muller genere números aleatorios con distribución normal standard.

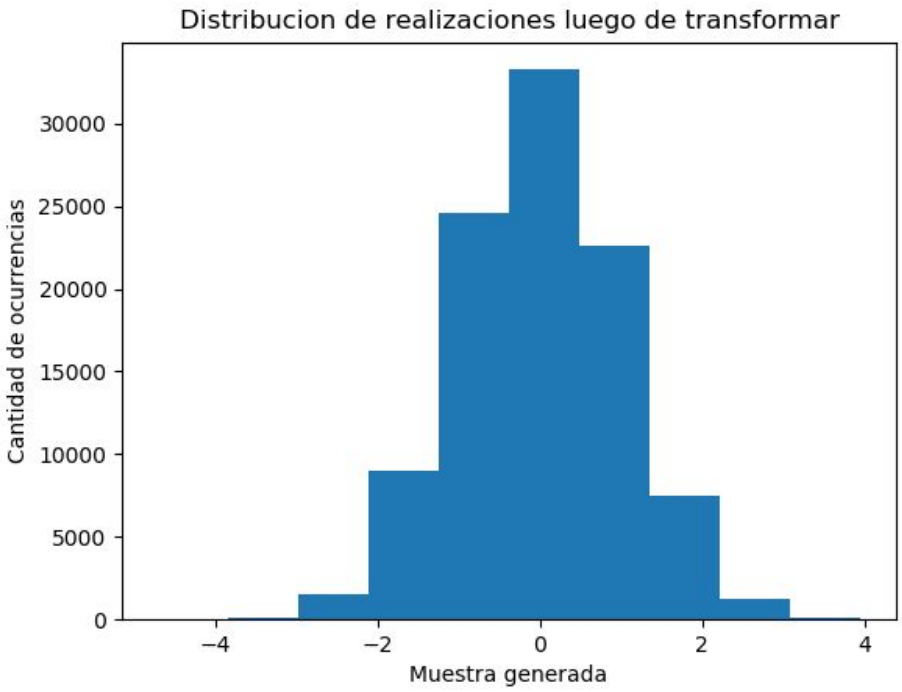
- Realizar un histograma de 100.000 valores obtenidos.
- Calcular la media y varianza de la distribución obtenida y compararlos con los valores teóricos.

El método de Box-Muller busca generar dos variables aleatorias normales estándar independientes. Para ello propone, a partir de dos uniformes independientes 0-1, la siguiente transformación:

$$z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2) = \sqrt{-2\ln s}\left(\frac{u}{\sqrt{s}}\right) = u\cdot\sqrt{\frac{-2\ln s}{s}}$$

$$z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2) = \sqrt{-2\ln s}\left(\frac{v}{\sqrt{s}}\right) = v\cdot\sqrt{\frac{-2\ln s}{s}}.$$

Como basta con generar una sola variable, se calcularán realizaciones solo para z0. Así, trabajando con 100.000 valores de U1 y otros 100.000 de U2, se generaron realizaciones para Z0, obteniendo el siguiente histograma:



Como se observa, la muestra obedece una distribución normal estándar.

Nuevamente, para concluir con mayor certeza, pueden calcularse los valores estimados vs. los teóricos.

Para la media (o esperanza) se halla que en la muestra resulta 0.004949 cuando se esperaba 0 teóricamente.

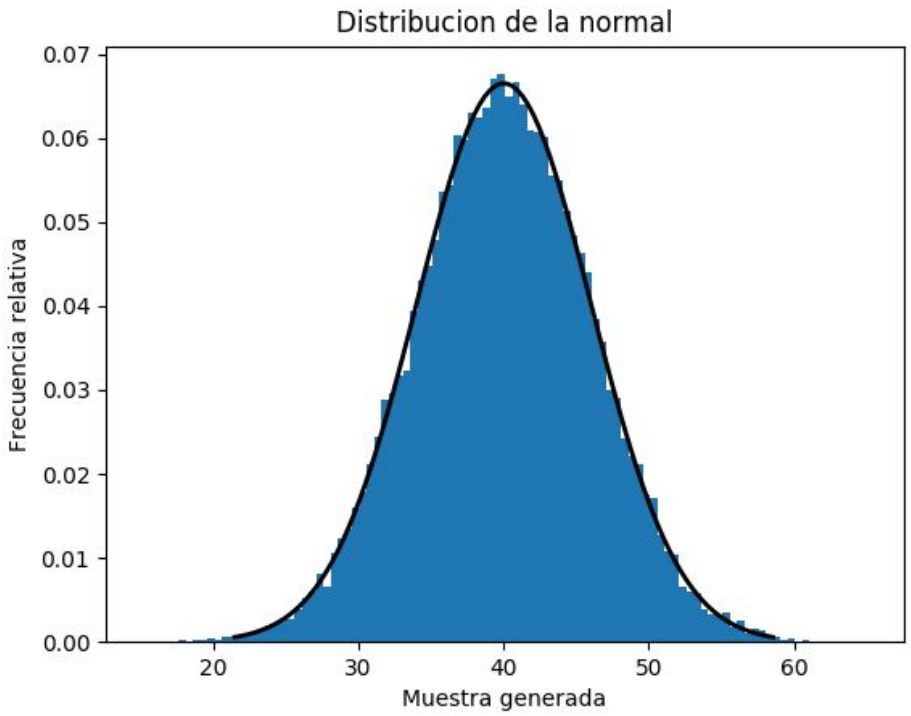
En la varianza, se halla el valor 1.00455 donde teóricamente se esperaba 1.

Ejercicio 4

Genere 100.000 número aleatorios con distribución Normal de media 40 y desvío estándar 6 utilizando el algoritmo de Aceptación y Rechazo.

- Realizar un histograma de frecuencias relativas con todos los valores obtenidos.
- Comparar, en el mismo gráfico, el histograma realizado en el punto anterior con la distribución normal brindada por Matlab, Octave o Python.
- Calcular la media y la varianza de la distribución obtenida y compararlos con los valores teóricos.

Luego de generar 100.000 muestras con este método se procede a generar un histograma de valor arrojado vs frecuencia relativa. A modo de comparación, se graficó superpuesta la distribución esperada (Normal media 40, desvio 6):



Claramente, el método arrojó correctamente muestras que obedecen a una Normal de media 40 y desvío 6.

Podemos ahora caracterizar la muestra con su media y varianza, para luego compararla a sus valores teóricos:

	Valor Empirico	Valor Experimental
Esperanza	40.009600	40
Varianza	35.963395	36

Nuevamente, se llega a un resultado esperado.

Ejercicio 5

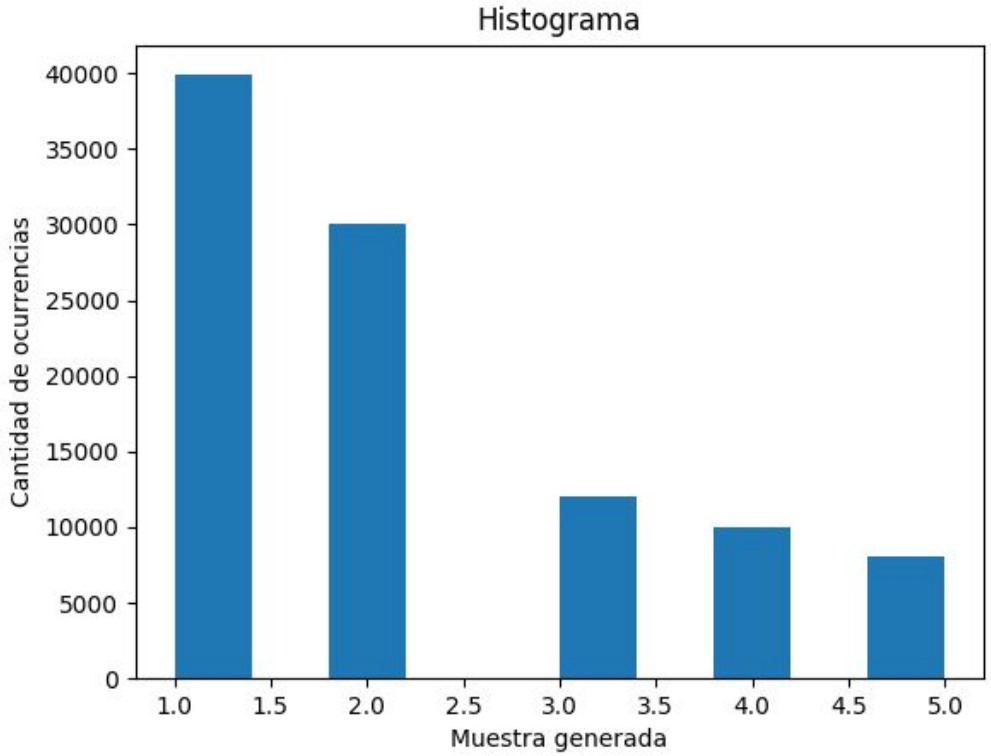
Utilizando el método de la transformada inversa y utilizando el generador de números aleatorios implementado en el ejercicio 1 genere números aleatorios siguiendo la siguiente función de distribución de probabilidad empírica.

Probabilidad	Valor generado
.4	1
.3	2
.12	3
.1	4
0.08	5

Muestre los resultados obtenidos en un histograma.

Se dividió al intervalo [0,1] en distintos subintervalos, donde, por ejemplo, si la realización de la uniforme estándar resultaba caer entre [0, 0.4] se le asignaba el valor 1, si caía entre [0.7, 0.82] seria el valor 3, etc.

De esta forma, se generaron las muestras y se las representó en un histograma:

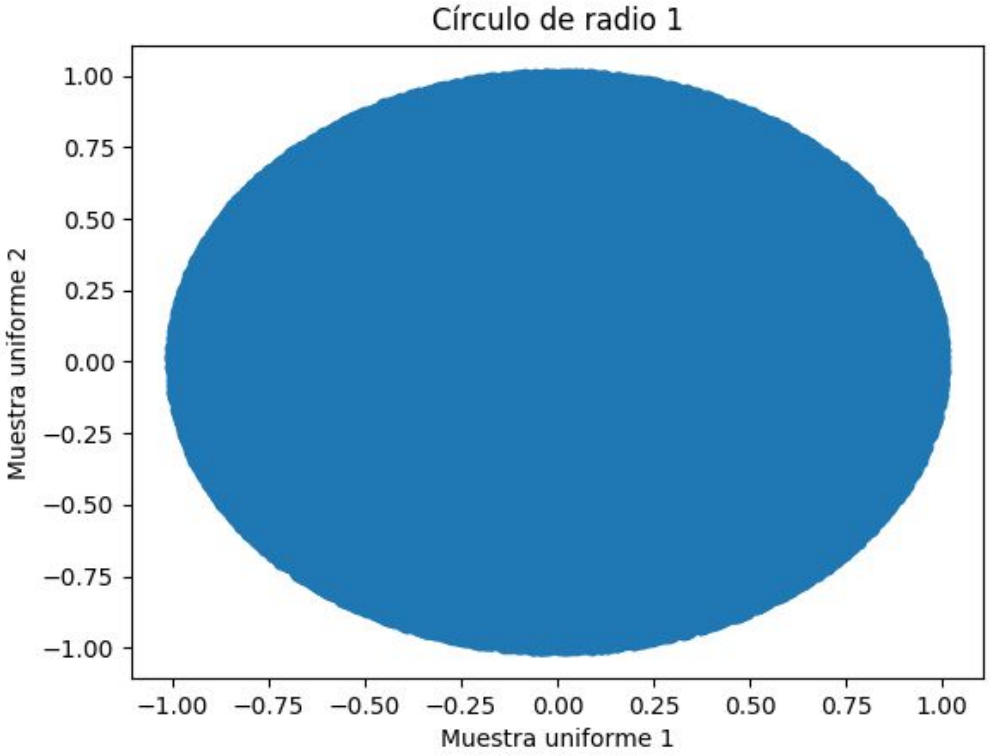


Evidentemente las muestras respetan la función de probabilidad dada.

Ejercicio 6

Utilizando 2 generadores de números al azar, provistos por el lenguaje elegido para resolver el tp, con distribuciones uniformes en [-1,1] genere números aleatorios en un círculo de radio 1 centrado en el origen.
Muestre el resultado en un gráfico de 2 dimensiones.

Para este ejercicio basto generar dos uniformes entre [-1, 1] e interpretarlo como un punto en el plano. Si el punto se encuentra dentro del círculo unitario se lo acepta, sino se lo descarta y se continúa generando, hasta acumular 100.000 muestras.
Asi, realizando un gráfico de 2 dimensiones resulta:



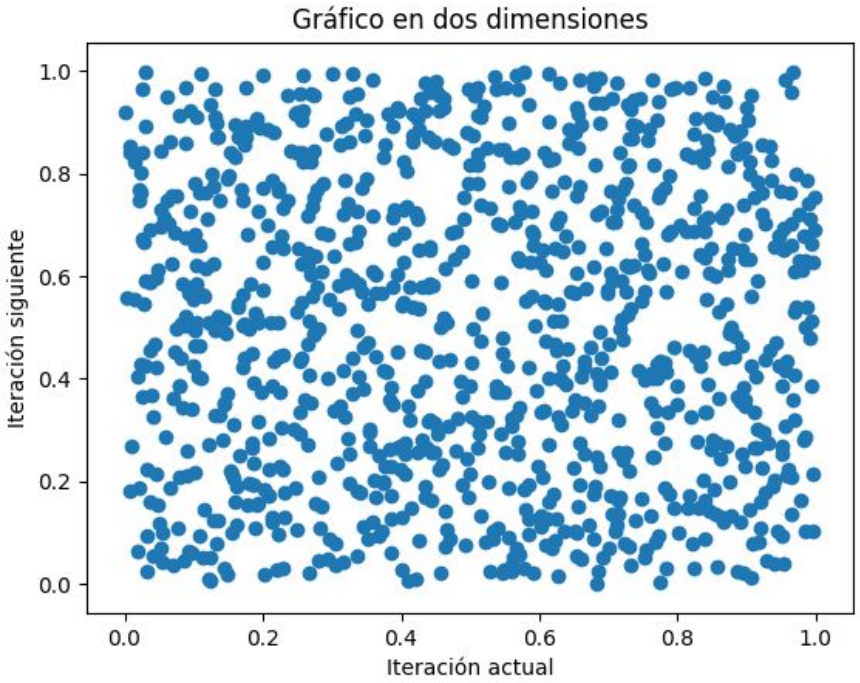
Se puede observar que todos los puntos pertenecen al círculo unitario centrado en el origen.

Test estadísticos

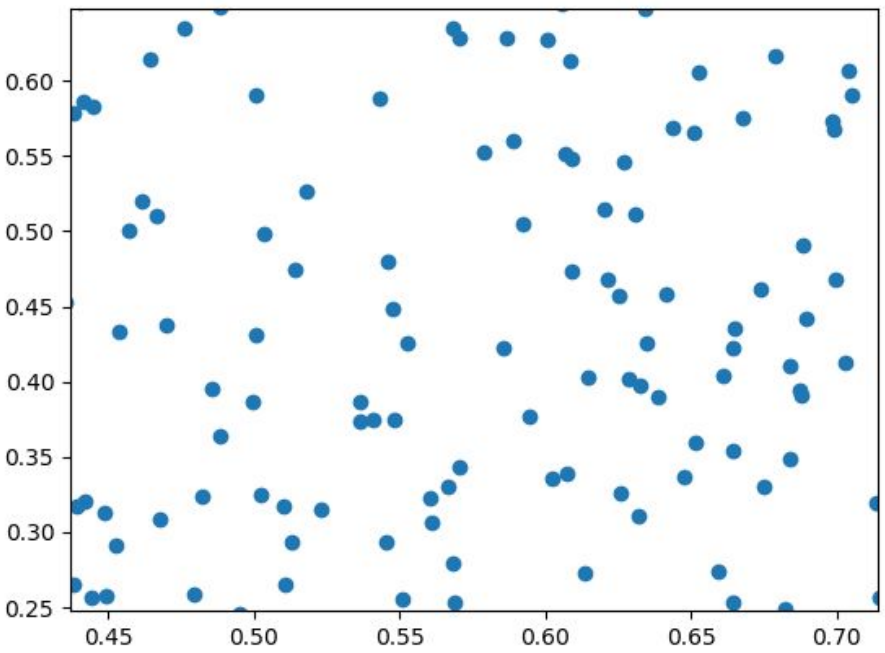
Ejercicio 7

Realizar, sólo gráficamente, un test espectral en 2 y 3 dimensiones al generador congruencial lineal implementado en el ejercicio 1. ¿Cómo se distribuyen espacialmente los puntos obtenidos?

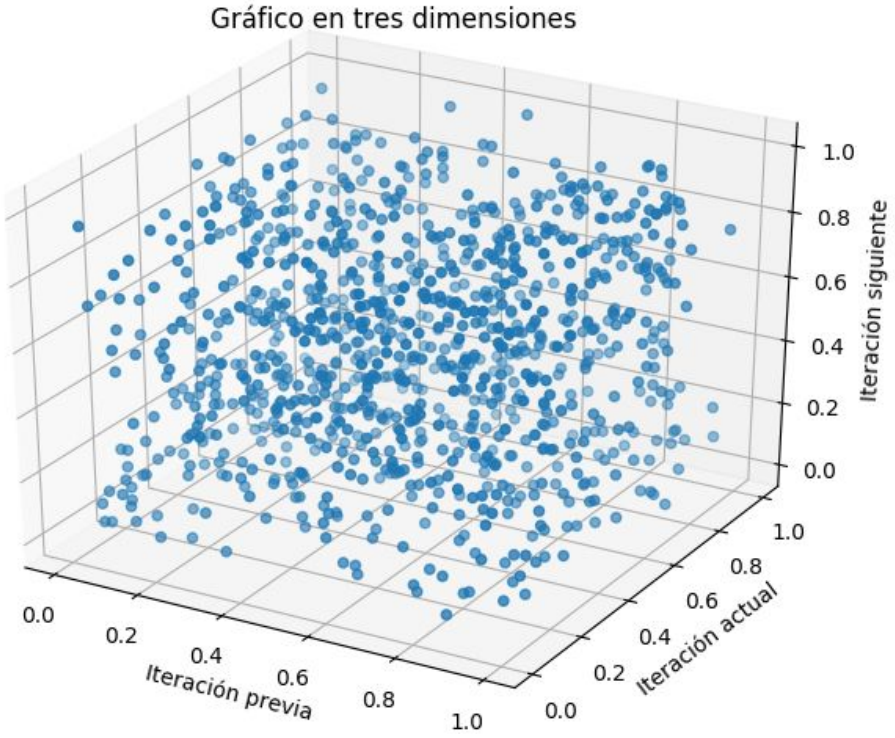
Se procedieron a generar 1.000 valores para obtener una mejor visualización y distinguir distancias. Así, graficando realizaciones en la iteración actual vs. la iteración siguiente resulta:



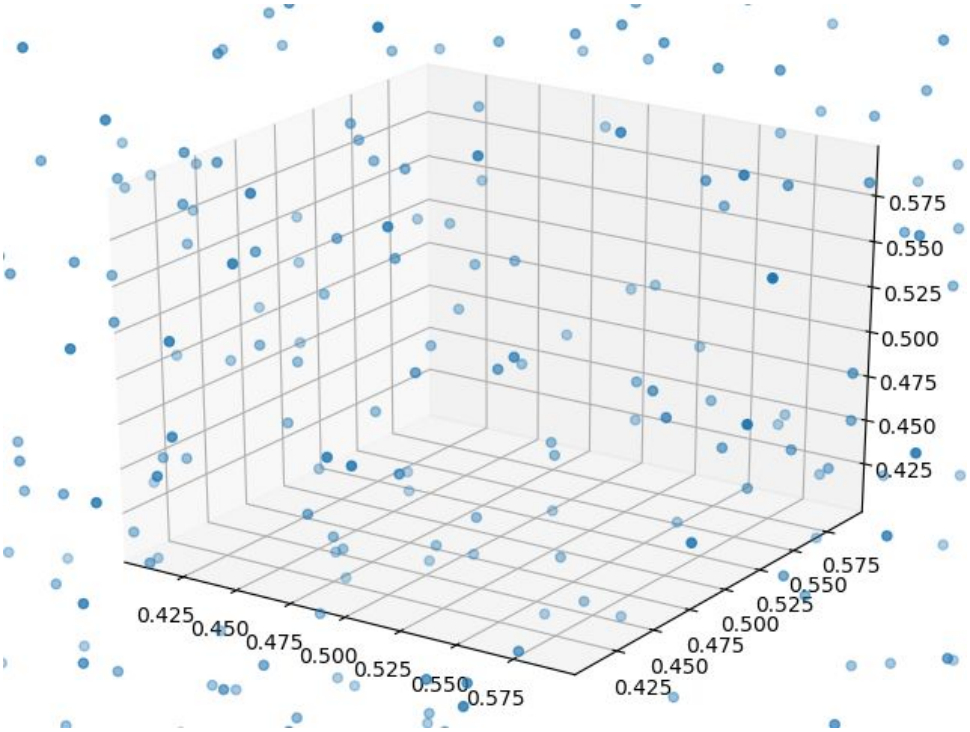
Analizando ahora con mayor aumento a una zona:



No se distingue ningún patrón significativo a simple vista. Viendo ahora en tres dimensiones (realización anterior vs. realización actual vs. realización siguiente):



Ocurre algo similar, analizando con mayor detalle:



Siguen sin distinguirse patrones. En principio, se concluye que GCL con estas configuraciones no forma patrones obvios lo cual es beneficioso al tratarse de números (pseudo) aleatorios.

Ejercicio 8

> Realizar un test Chi 2 a la distribución empírica implementada en el Ej 5, analizar el resultado indicando si la distribución puede o no ser aceptada.

Utilizando un nivel de significación de 0.01, se llevó a cabo el test Chi2. Se optó por recurrir a una función implementada en la librería SciPy.

Así, el test sobre la muestra resultó en un p-valor de 0.8064 y como resulta p-valor > 0.01 (nivel de significación) NO se rechaza la hipótesis nula. En otras palabras, las distribuciones coinciden.

Ejercicio 9

Al generador congruencial lineal implementado en el ejercicio 1 realizarle un gap test para el intervalo [0,2 - 0,5], analizar el resultado indicando si pasa el test.

Para el gap test se tomaron los extremos 0.2 y 0.5 y se procedió a calcular las probabilidades asociadas. Se exigió calcular probabilidades (las cuales resultan descendientes) hasta que la computadora no pueda distinguir más la diferencia entre una probabilidad y la siguiente. En otras palabras, continuó hasta que se hizo cero, devolviendo 2048 probabilidades distintas.
Por otro lado, se simularon 100.000 muestras del GCL y se contó la cantidad de veces que: no se cayó afuera entre dos veces que se cayó adentro, se cayó 1 vez afuera entre dos veces adentro, se cayo 2 veces afuera… etc. Luego se halló la frecuencia para cada uno (valor dividido situaciones totales) y se procedió con el test Chi2.

El test arrojó el p-valor de 1.0, lo cual al ser mayor (y mucho mayor) que el nivel de significación (0.01) se concluye que el GCL está respetando las probabilidades esperadas. Esto sugiere que el método está efectivamente generando según una uniforme estándar.

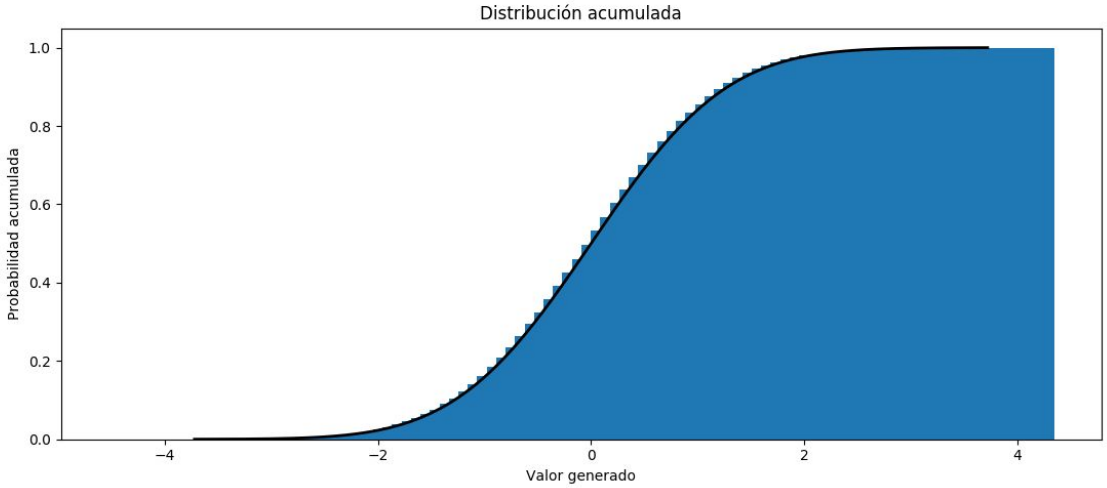
Ejercicio 10

Aplicar el test de Kolmogorov-Smirnov al generador de números al azar con distribución normal generado en el ejercicio 3, y analizar el resultado del mismo.
Graficar la distribución acumulada real versus la distribución empírica.

En este caso, se utilizó una función implementada por la librería SciPy, donde dadas las realizaciones y la distribución contra la que se desea comparar devuelve el estadístico y p-valor asociados.

De esta forma, utilizando un nivel de significación 0.01, se halló un p-valor de 0.9542. Finalmente, siendo p-valor > 0.01, se concluye que la realización proviene de una normal estándar, como era esperado.

Graficando ahora la distribución acumulada real (negro) vs la distribución empírica (azul):



Se observa la esperada coincidencia entre ambas graficas.

Anexo - Código en Python

Todo el código está también disponible en GitHub:
https://github.com/Franco-Giordano/simulacion-TP1-1C2019

Ejercicio 1

```
1.
2.  from matplotlib import pyplot as plt
3.
4.
5.  #GCL
6.
7.  def GCL(multi, inc, mod, x0):
8.      #x0 = valor anterior
9.
10.     return ((multi * x0) + inc) % mod
11.
12. def realizacionesGCL(cantidad=100000):
13.     x0 = 96771 # = 0.15*74620 + 0.25*100608 + 0.6*100710
14.
15.     multiplicador = 1013904223
16.
17.     incremento = 1664525
18.
19.     modulo = 2**32
20.
21.     resultados = []
22.
23.     iteracion_actual = x0
24.
25.     for i in range(cantidad):
26.
27.         iteracion_actual = GCL(multiplicador, incremento, modulo, iteracion_actual)
28.         resultados.append(iteracion_actual)
29.
30.     return resultados
31.
32. def realizacionesEstandarGCL(cantidad=100000):
33.     return [i/(2**32) for i in realizacionesGCL(cantidad=cantidad)]
34.
35.
36.
37. if __name__ == "__main__":
38.
39.     resultados_A = realizacionesGCL(cantidad=5)
40.
```

```
    print(resultados_A)

    resultados_B = realizacionesEstandarGCL()

    plt.title('Distribucion de realizaciones con GCL')
    plt.ylabel('Cantidad de ocurrencias')
    plt.xlabel('Muestra generada')
    plt.hist(resultados_B)
    plt.show()
```

Ejercicio 2

```
from ejsTP1_1 import realizacionesGCL

from matplotlib import pyplot as plt
import math

uniformes = realizacionesGCL()

resultados = [-20*math.log(1-u/(2**32)) for u in uniformes]


plt.title('Distribucion de realizaciones luego de transformar')
plt.ylabel('Cantidad de ocurrencias')
plt.xlabel('Muestra generada')

plt.hist(resultados)
plt.show()

#calculo de media, se espera 20
media = (sum(resultados) / len(resultados))
print(media)

#calculo de varianza, se espera 1/20^2
varianza = 1 / (sum(resultados) / len(resultados)) ** 2
print(varianza)
```

Ejercicio 3

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


def realizacionesNormalEstandar(cant=100000):
    resultados = []

    for i in range(cant):
```

```
11.
12.         u1, u2 = random.random(), random.random()
13.
14.         z = math.sqrt(-2 * math.log(u1)) * math.cos(2 * math.pi * u2)
15.
16.         resultados.append(z)
17.
18.     return resultados
19.
20. if __name__ == "__main__":
21.
22.     resultados = realizacionesNormalEstandar()
23.
24.
25.
26.     plt.title('Distribucion de realizaciones luego de transformar')
27.     plt.ylabel('Cantidad de ocurrencias')
28.     plt.xlabel('Muestra generada')
29.
30.     plt.hist(resultados)
31.     plt.show()
32.
33.     #media
34.     media = sum(resultados) / len(resultados)
35.     print("ESPERANZA: ",media)
36.
37.     #varianza
38.     print("VARIANZA: ",np.var(resultados))
```

Ejercicio 4

```
1.  import random
2.  import math
3.  from matplotlib import pyplot as plt
4.  from scipy.stats import norm
5.
6.  import numpy as np
7.
8.
9.  c = 1.3 #haciendo fX(t)/fY(t) se halla que este valor (aproximado) de t maximiza el
    cociente
10.
11. realizaciones_std = []
12.
13. def cociente(t):
14.     return (1/math.sqrt(2*math.pi)) * math.exp(t - (t**2 / 2)) * 1/c
15.
16. for i in range(100000):
17.
18.     t = random.expovariate(1) #genero mi realizacion exponencial de media 1
```

```
19.
20.     u = random.random()
21.
22.     if u < cociente(t):
23.         #acepto la realizacion, ahora decido si es por encima de la media o por debajo
24.
25.         u2 = random.random()
26.
27.         if u2 < 0.5:
28.             realizaciones_std.append(t)
29.
30.         else:
31.             realizaciones_std.append(-t)
32.
33. #hago el cambio de variable para que tenga una normal con media 40,desvio 6
34. realizaciones = [z*6 + 40 for z in realizaciones_std]
35.
36. x = np.linspace(norm.ppf(0.001, 40,6), norm.ppf(0.999,40,6), 100) #la normal "real" la
    grafico entre los cuantiles z0.001 y z0.999
37. fig, ax = plt.subplots(1, 1)
38.
39. rv = norm(40,6)
40. ax.plot(x, rv.pdf(x), 'k-', lw=2)
41.
42. ax.hist(realizaciones,100, density=True)
43. plt.show()
44.
45.
46. print('ESPERANZA: {}, VARIANZA: {}'.format(np.mean(realizaciones),
    np.var(realizaciones)))
```

Ejercicio 5

```
1.  from ejsTP1_1 import realizacionesEstandarGCL
2.
3.  from matplotlib import pyplot as plt
4.  import math
5.
6.  def realizacionesEJ5(cantidad=100000):
7.
8.      realizacionesUniformes = realizacionesEstandarGCL(cantidad)
9.
10.     resultados = []
11.
12.     for u in realizacionesUniformes:
13.
14.         realizacion = 0
15.
16.         if u <= 0.4:
17.             realizacion = 1
```

```
18.
19.         if 0.4 < u <= 0.7:
20.             realizacion = 2
21.
22.         if 0.7 < u <= 0.82:
23.             realizacion = 3
24.
25.         if 0.82 < u <= 0.92:
26.             realizacion = 4
27.
28.         if 0.92 < u:
29.             realizacion = 5
30.
31.         resultados.append(realizacion)
32.
33.     return resultados
34.
35. if __name__ == "__main__":
36.
37.     plt.hist(realizacionesEJ5())
38.     plt.show()
39.
```

Ejercicio 6

```
1.  import random
2.  from matplotlib import pyplot as plt
3.
4.  xs, ys = [],[]
5.
6.  puntos = 0
7.
8.  while puntos < 100000:
9.
10.     u1 = random.uniform(-1,1)
11.
12.     u2 = random.uniform(-1,1)
13.
14.     if u1**2 + u2**2 <= 1: #si pertenece al circulo unitario...
15.
16.         xs.append(u1)
17.         ys.append(u2)
18.         puntos += 1
19.
20. plt.scatter(xs,ys)
21. plt.show()
```

Ejercicio 7

```
1.  from ejsTP1_1 import realizacionesEstandarGCL
```

```
2.
3.  from matplotlib import pyplot as plt
4.  from mpl_toolkits.mplot3d import Axes3D
5.
6.
7.  resultados_previos = realizacionesEstandarGCL(1002)
8.
9.  resultados = resultados_previos[1:]
10. resultados_previos = resultados_previos[:-2]
11.
12. resultados_siguientes = resultados[1:]
13. resultados = resultados[:-1]
14.
15. plt.scatter(resultados_previos, resultados)
16.
17. fig = plt.figure()
18. ax = Axes3D(fig)
19.
20. ax.scatter(resultados_previos, resultados, resultados_siguientes)
21.
22. plt.show()
```

Ejercicio 8

```
1.  from scipy import stats as sp
2.  import matplotlib.pyplot as plt
3.
4.  from ejsTP1_5 import realizacionesEJ5
5.
6.  realizaciones = realizacionesEJ5()
7.
8.  #cuento las apariciones
9.  obs_values = [0 for i in range(5)]
10. for i in realizaciones:
11.     obs_values[i-1] += 1
12.
13.
14. expected_p_values = [0.4, 0.3, 0.12, 0.1, 0.08]
15. expected_values = [100000*i for i in expected_p_values]
16.
17.
18. statistic, pvalue = sp.chisquare(obs_values, f_exp=expected_values)
19.
20. print(statistic, pvalue)
21.
22.
23. #si pvalue <= alfa entonces rechazo H0 (digo que son distintas)
24.
25. if pvalue <= 0.01:
26.     print("las distribuciones NO son las mismas")
```

```
27.
28. else:
29.     print("las distribuciones coinciden")
```

Ejercicio 9

```
1.  from ejsTP1_1 import realizacionesEstandarGCL
2.
3.  from matplotlib import pyplot as plt
4.
5.  from scipy import stats as sp
6.
7.
8.  a,b = 0.2,0.5
9.
10. #calculo las probabilidades, corto cuando ya no distingo diferencia
11. probs = [b-a]
12. termino = False
13. i = 1
14. while not termino:
15.     nuevaProb = probs[0] * ((1-probs[0]) ** i)
16.
17.     if nuevaProb == probs[i-1]:
18.         termino = True
19.
20.     else:
21.         probs.append(nuevaProb)
22.         i += 1
23.
24.
25.
26. muestras = realizacionesEstandarGCL()
27.
28. #calculo cantidad de veces que cai 0 afuera, 1 afuera, 2 afuera...
29. primero = True
30. contador_local = 0
31. numeros = {}
32. for x in muestras:
33.     if (x>=0.2 and x<=0.5 and primero):
34.         primero = False
35.     if (x>=0.2 and x<=0.5 and not primero):
36.         veces = numeros.get(contador_local,0)
37.         numeros[contador_local] = veces + 1
38.         contador_local = 0
39.     if ((x<0.2 or x>0.5) and not primero):
40.         contador_local+=1
41.
42.
43. #ajusto para que sean frecuencias
44. total = sum(numeros.values())
```

```
for k in numeros.keys():
    numeros[k] /= total

#lo paso a lista y relleno con 0 donde no tuve apariciones
lista = []
for i in range(2084):
    lista.append(numeros.get(i,0))

print(probs, lista)

print(sp.chisquare(lista, f_exp=probs))
```

Ejercicio 10

```
from ejsTP1_3 import realizacionesNormalEstandar

from matplotlib import pyplot as plt
from scipy.stats import norm
from scipy.stats import kstest


import numpy as np

realizaciones = realizacionesNormalEstandar()

stat, pvalue = kstest(realizaciones, 'norm', N=100000)

print(pvalue)

if pvalue <= 0.01:
    print("Hay evidencia suficiente para decir que las muestras NO vienen de una normal")

else:
    print("Las muestras provienen de una normal estandar")


fig, ax = plt.subplots(1, 1)
x = np.linspace(norm.ppf(0.0001), norm.ppf(0.9999), 100000) #la normal "real" la grafico entre los cuantiles z0.0001 y z0.9999

rv = norm()
ax.plot(x, rv.cdf(x), 'k-', lw=2)

ax.hist(realizaciones,100, density=True, cumulative=True)
plt.show()
```