FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1

[75.26 - 95.19] Simulación
Segundo cuatrimestre de 2018

Grupo 5

Alumno	Número de Padrón	Email
Bataglia, Matías	95354	matubat91@gmail.com
Beroch, Santiago	101135	sberoch@gmail.com
Ruiz, Francisco	99429	fran7ruiz9@gmail.com

Repositorio GitHub: https://github.com/FranR96/Simulacion_fiuba

Índice

1. Introducción **2**

2. Ejercicio 1 **2**
2.1. Enunciado . . . 2
2.2. Resolución . . . 2
2.3. Código . . . 3

3. Ejercicio 2 **3**
3.1. Enunciado . . . 3
3.2. Resolución . . . 4
3.3. Código . . . 4

4. Ejercicio 3 **5**
4.1. Enunciado . . . 5
4.2. Resolución . . . 5
4.3. Código . . . 6

5. Ejercicio 4 **6**
5.1. Enunciado . . . 6
5.2. Resolución . . . 6
5.3. Código . . . 8

6. Ejercicio 5 **8**
6.1. Enunciado . . . 8
6.2. Resolución . . . 8
6.3. Código . . . 9

7. Ejercicio 6 **9**
7.1. Enunciado . . . 9
7.2. Resolución . . . 10
7.3. Código . . . 10

8. Ejercicio 7 **10**
8.1. Enunciado . . . 10
8.2. Resolución . . . 11
8.3. Código . . . 11

9. Ejercicio 8 **12**
9.1. Enunciado . . . 12
9.2. Resolución . . . 12
9.3. Código . . . 12

10. Ejercicio 9 **13**
10.1. Enunciado . . . 13
10.2. Resolución . . . 13
10.3. Código . . . 13

11. Ejercicio 10 **14**
11.1. Enunciado . . . 14
11.2. Resolución . . . 14
11.3. Código . . . 16

1. Introducción

En el siguiente informe, presentaremos la resolución del trabajo práctico nº 1 de la materia Simulación para las carreras de Ingeniería en Informática y Licenciatura en Sistemas. Para poder resolver los ejercicios planteados se utilizo Octave.

2. Ejercicio 1

2.1. Enunciado

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 del promedio de los números de padrón de los integrantes del grupo.

- Informar los primeros 6 números al azar 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.

2.2. Resolución

Para la resolución de este ejercicio, utilizamos los padrones de los integrantes de este grupo, 95354, 99429, 101135 para hallar la semilla con la cual el Generador Congruencial Lineal debe iniciar.

La semilla obtenida, es el promedio de los padrones y es 98639.

Los primeros 6 números obtenidos por el GCL son: 2186829662, 1137640448, 1656218112, 1239172096, 3043140096, 2339996672. Para obtenerlos, simplemente se modificó el código que se adjunta en la sección 2.3, eliminando la normalización que se genera al dividir y por 2^32.

En el siguiente gráfico se puede observar el histograma del GCL bastante desparejo, con varios picos, generando una sensación de que el generador no es realmente uniforme, esto se puede producir debido a la cantidad de bins utilizados (100).

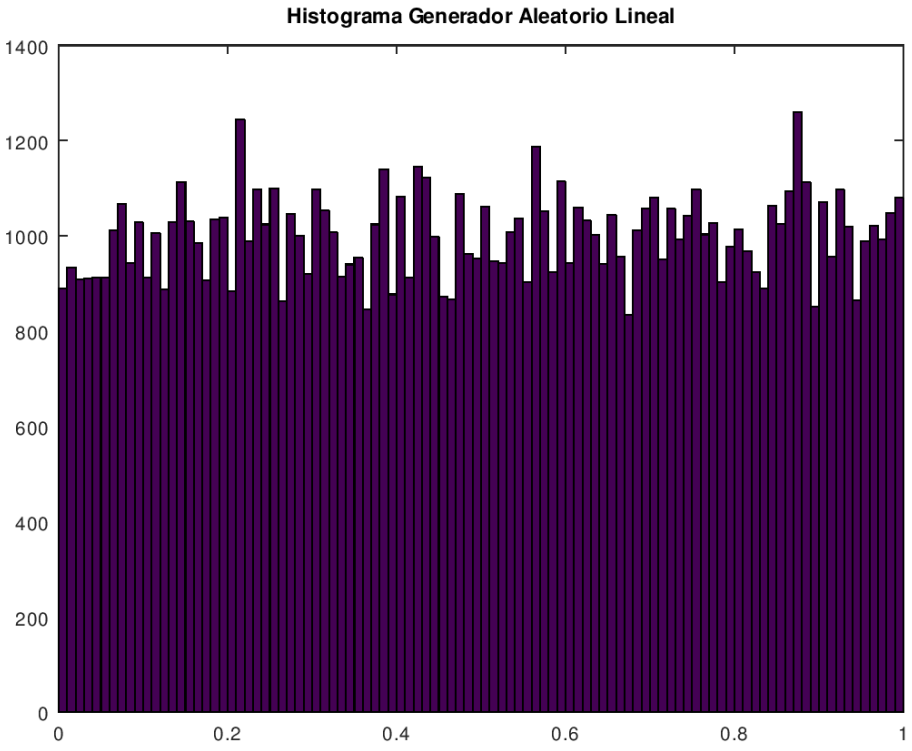

Figura 1: Histograma del Generador Congruencial Lineal

Ahora, comparémoslo con un histograma que tenga los mismos datos, distribuidos en una menor cantidad de bins (20).

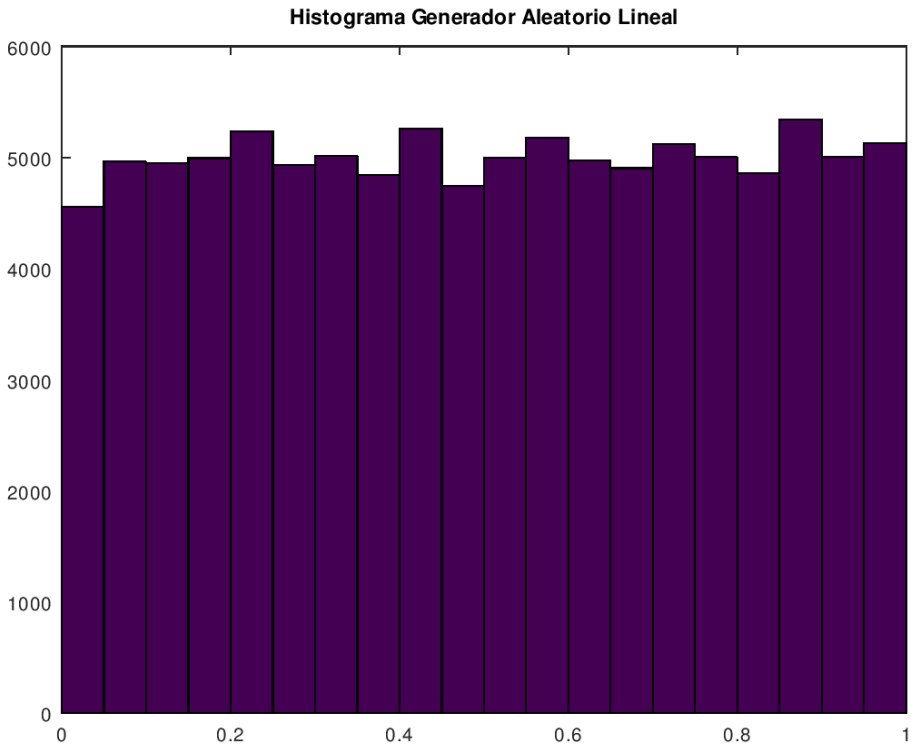

Figura 2: Histograma del Generador Congruencial Lineal con 20 bins

En este plot, podemos observar que la distribución se puede apreciar mucho más uniforme de la que se veía en la Figura 1, esto se debe a que los bins son más anchos y tienen mayor probabilidad de captar números.

2.3. Código

```
function z = generadorAleatorioLineal(n)
  semilla = 98639;
  for i = 1:n
    y = mod((1013904223*semilla+1664525),2^32);
    semilla = y;
    z(end + 1) = y/(2^32);
  endfor
endfunction
```

El parámetro n, nos indica la cantidad de números que se quieren obtener con el generador.

Podemos ver como se aplica la función del GCL y luego los números obtenidos, son normalizados y se van almacenando en un vector z.

3. Ejercicio 2

3.1. Enunciado

Utilizando el generador de números aleatorios con distribución uniforme [0,1] implementado en el ejercicio 1 y utilizando el método de la transformada inversa genere números pseudoaleatorios con distribución exponencial negativa de media 15.

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

3.2. Resolución

En este ejercicio, debemos calcular la inversa de la distribución exponencial negativa, esto es despejar x de la siguiente ecuación: $1 - e^{-lambda*x} = u$ y lo que obtengo es la función en la cual voy a evaluar a u, siendo estos valores generados por el GCL del punto anterior, para obtener los valores de la exponencial negativa. En la sección 3.3, se puede observar el código implementado para realizar la inversa de la distribución ya mencionada.

En el siguiente histograma, podemos ver la distribución exponencial negativa representada con los datos obtenidos previamente.

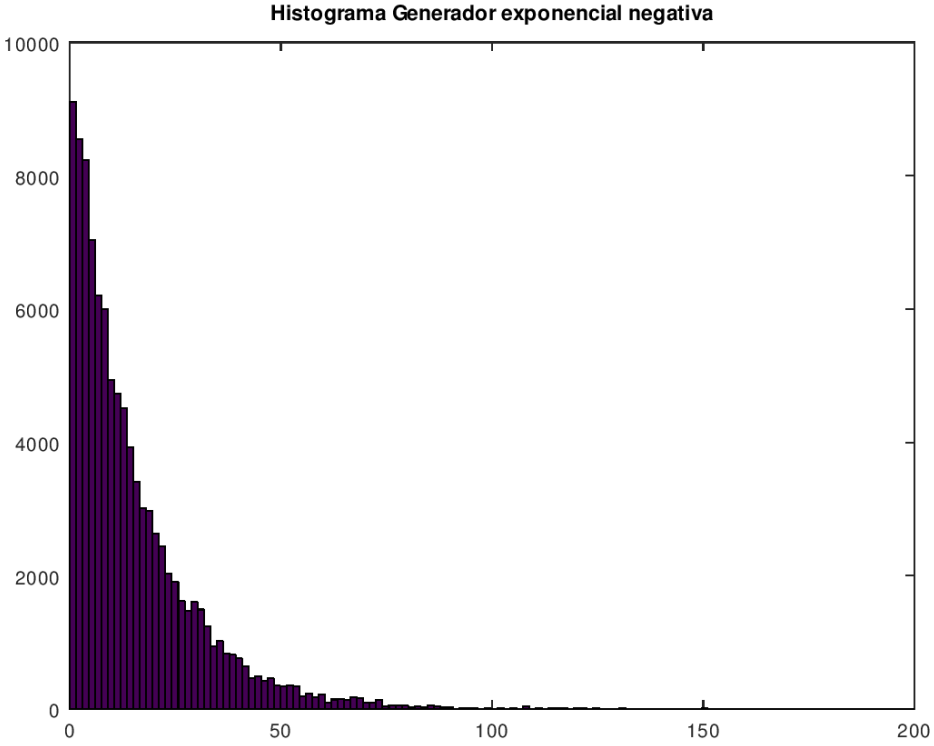

Figura 3: Histograma del Generador Exponencial Negativa

Ahora analizaremos la media, varianza y moda de la distribución obtenida.

- La media teórica de la exponencial es *lambda* y como ya sabemos por enunciado, esta es 15. La esperanza obtenida es de 15.179, por lo cual es bastante aproximada.
- La varianza teórica de la distribución dada en el enunciado es de 225, mientras que la obtenida es de 230.54, que comparado al item anterior, se trata de una diferencia un poco más grande.
- Por último, la moda de una Función de densidad de probabilidad de la distribución exponencial siempre es 0, que resulta muy similar a la moda que se alcanzó a través del método de la inversa que fue 0.00037551.

3.3. Código

```
function y = generadorExponencialNegativa()
 i = 1;
 lambda =1/15;
 numerosGenerados = generadorAleatorioLineal(100000);
 while i <= 100000
   y(end+1) = -log(1-numerosGenerados(i))/(lambda);
   i = i+1;
 endwhile
endfunction
```

En el código, se puede apreciar como se ha utilizado el generador congruencial lineal para obtener las *ues* que se serán evaluadas en la ecuación que se encuentra en el while y así generar aquellos números que siguen una distribución exponencial negativa.

4. Ejercicio 3

4.1. Enunciado

Utilizando el generador de números aleatorios con distribución uniforme [0,1] implementado en el ejercicio 1 genere números pseudoaleatorios con distribución Normal Standard utilizando el método de la transformada inversa (realizando un muestreo de función de distribución acumulada e interpolándolos linealmente).

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

4.2. Resolución

Para este ejercicio se quiere generar una distribución Normal Standard con el método de la transformada inversa. Se presenta un problema con la inversa de la normal, por lo que para obtener la distribución buscada se utilizaron 24 puntos coincidentes con el gráfico de la función de probabilidad acumulada inversa y se los interpolo para lograr una aproximación a dicha función. La interpolación resulta certera al utilizar la función de octave "splinefit", que permite interpolar por splines. En nuestro caso serán 8 polinomios pegados de orden 4 para acercarse lo mejor posible al resultado. Se podría haber aumentado el orden o el numero de splines utilizados, pero el coste computacional es mayor y no se considero necesario.

Finalmente se obtiene para cada uniforme generado, al evaluarlo en la función inversa aproximada, un número que obedece a la distribución Normal Standard, resultando el siguiente gráfico:

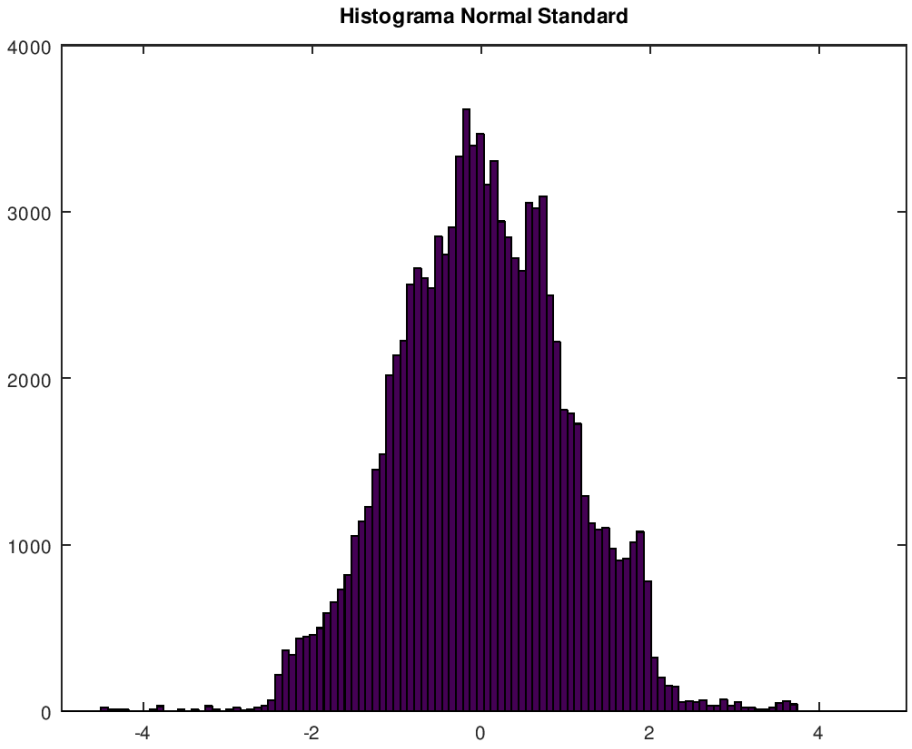

Figura 4: Histograma de la Normal Standard

- La media de la Normal Standard es 0, y obtuvimos 0.01481 como esperanza de la variable aleatoria generada.
- La varianza teórica es 1 y la Normal Standard que fue generada por el método de la transformada inversa tiene como varianza 0.99526. Un valor muy aproximado al real.

- La moda es el elemento que más frecuencia tiene en un conjunto de datos, y para la distribución Normal Standard es 0. En comparación con la obtenida que fue 0.017, donde nos vimos obligados a bajarle la precisión a los float para que no calcule cualquier moda y tenga más sentido.

4.3. Código

En M definimos una matriz donde cada columna representa un punto a interpolar, y las filas representan su coordenada en x y su coordenada en y, respectivamente.

Luego, evaluamos los valores que nos devuelve nuestro generador realizado en el ejercicio 1, en el polinomio que se utilizó para interpolar los puntos que estaban en la matriz.

```
function z = generadorNormalInversa(n)

M = [0,0.00003,0.00135,0.00621,0.02275,0.06681,0.11507,0.15866,0.21186,...
+0.27425,0.34458,0.42074,0.5,0.57926,0.65542,0.72575,0.78814,0.84134,...
+0.88493,0.93319,0.97725,0.99379,0.99865,0.99997;-5,-4,-3,-2.5,-2,-1.5,...
+ -1.2,-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1,1.2,1.5,2,2.5,3,4];

x = M(1,:);
y = M(2,:);

p = splinefit(x,y,8,"order",4);
u = generadorAleatorioLineal(n);

for i = 1:n
  z(end+1) = ppval(p,u(i));
endfor
endfunction
```

5. Ejercicio 4

5.1. Enunciado

Genere 100.000 número aleatorios con distribución Normal de media 35 y desvío estándar 5 utilizando el algoritmo de Aceptación y Rechazo.

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

5.2. Resolución

Al querer utilizar el método de Aceptación-Rechazo, debemos tener una función Y(t) que se superponga con la distribución que queremos generar en algún sector y que en otros no. Justamente, este método elije aquellos puntos que caigan en la zona de solapamiento y descarta aquellos que no estén ahí.

Al pensar este ejercicio, nos encontramos con la dificultad de que la Normal estaba corrida y no hallábamos ninguna función que cumpla las características mencionadas arriba.

Por lo tanto, decidimos generar la distribución Normal Standard, con el método de Aceptación-Rechazo, siendo Y(t) una función con distribución exponencial, para luego aplicar la propiedad de transformación lineal $X = a * Z + b$, donde a es el desvío estándar, b la media de la distribución a la cual se quiere ir y Z la variable aleatoria con distribución Normal Standard, para así obtener los puntos generados que sigan la distribución pedida N(35,25). El resultado es el siguiente:

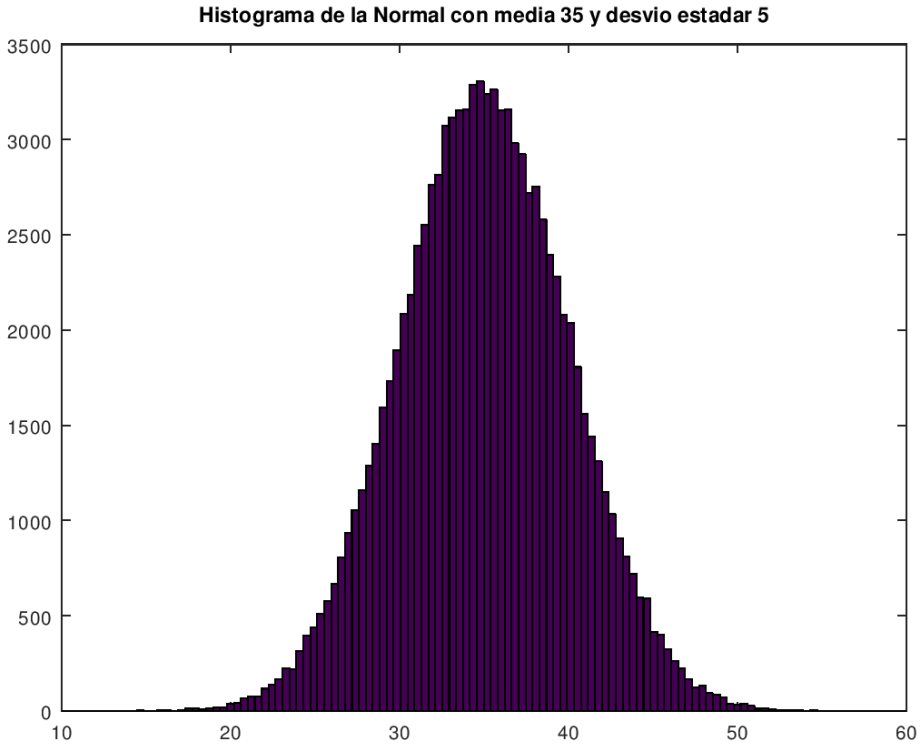

Figura 5: Histograma de la Normal con media 35 y desvió estándar 5

En la figura 6, podemos ver que la distribución generada previamente se aproxima de una manera muy precisa a la distribución Normal que nos brinda Octave, siendo ésta la linea roja en el gráfico.

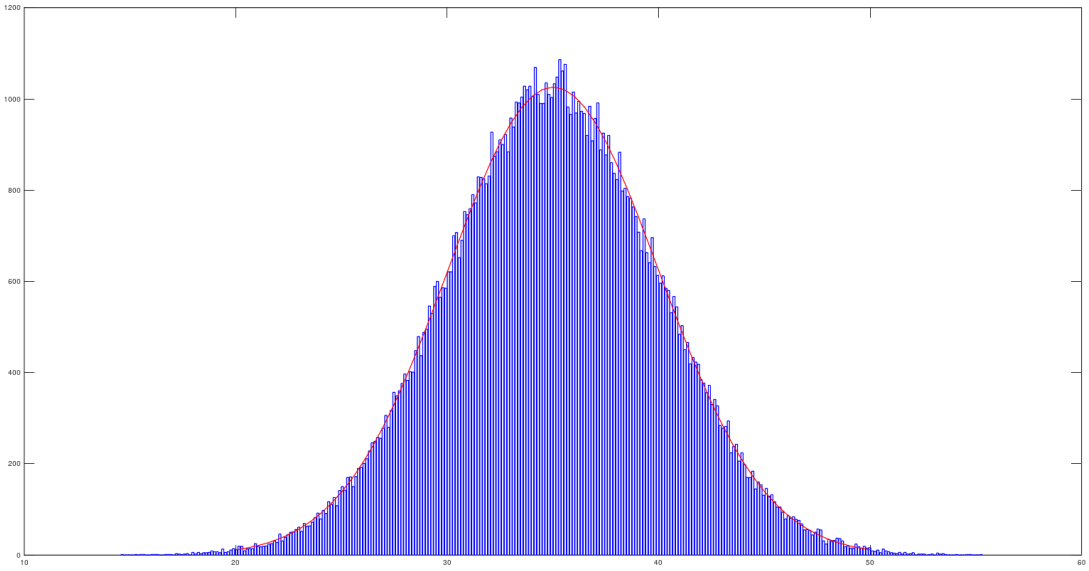

Figura 6: Comparación de lo obtenido con la generada por normpdf

Respecto al análisis de la media, varianza y moda teóricas comparadas contra las obtenidas mediante la simulación tenemos que:

- La esperanza teórica de una variable aleatoria que sigue una distribución normal es μ, en este caso es 35, y la obtenida es 35.011, bastante aproximada al valor esperado.
- La varianza teórica es el desvío estándar elevado al cuadrado, es decir 25. Mediante la generación de esta variable aleatorio por el método de Aceptación-Rechazo el resultado fue 25.163, siendo otro valor bastante cercano al teórico.

- Por último, la moda de la distribución Normal es igual a la media, como ya hemos mencionado esta es 35. Pero ahora, la moda obtenida nos obliga a generar un redondeo a 3 dígitos porque el tratamiento del punto flotante de Octave nos generaba un error grosero, entonces el resultado obtenido es 35.365 que tiene una pequeña diferencia con la moda teórica.

5.3. Código

Como se puede observar en el código, generamos valores aleatorios para las distribuciones exponenciales y uniformes, chequeando que cumplan la condición de aceptación. Para aquellos valores aceptados, con una probabilidad de 0.5, se dejaba tal cuál estaba o se los multiplicaba por -1 y así obtener la simetría de la campana de Gauss, además se genera un redondeo a tres decimales, para que la moda sea más exacta.

Una vez generados todos los números de la Normal Standard, se procedió a aplicarle a cada unos de los elementos del vector z, la transformación lineal mencionada en la sección 5.2.

```
function z= generadorNormal(media,desvioEstandar)

  c = sqrt(2*exp(1)/pi);
  i = 0;
  while i <100000
    j = rande();
    u = rand();
    if u < ((exp(-j.^2/2)/sqrt(2*pi))/(c*exp(-j)))
      p = rand();
      if p<0.5
        z(end + 1) = round(1000*(desvioEstandar*j+media))/1000;
      else
        z(end+1) = round(1000*(desvioEstandar*(-j)+media))/1000;
      endif
      i = i+1;
    endif
  endwhile

endfunction
```

6. Ejercicio 5

6.1. Enunciado

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
0.5	1
0.2	2
0.1	3
0.2	4

6.2. Resolución

Para realizar la función inversa de una función de probabilidad empírica, lo que hacemos es dividir el espacio de la uniforme [0,1], de acuerdo a las probabilidades de que ocurra cada evento. Es decir, para el intervalo [0; 0.5] será para el valor 1, en el intervalo [0.5; 0.7] se genera el valor 2, al 3 le corresponde el [0.7; 0.8] y por último al 4, el [0.8;1].

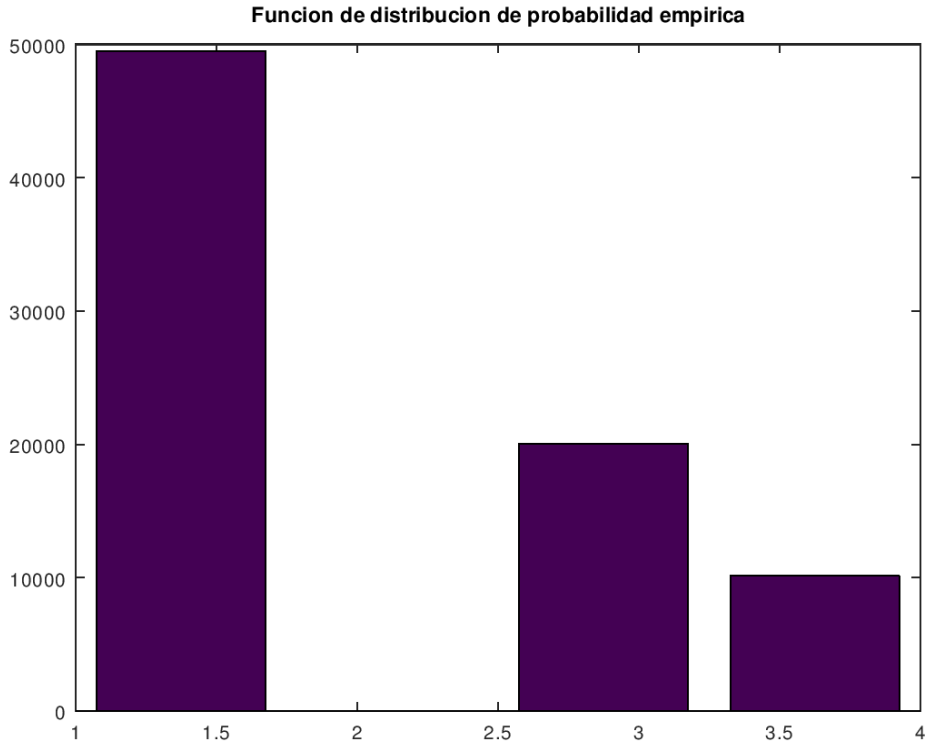

Figura 7: Función de distribución de probabilidad empírica

6.3. Código

Como vemos, generamos los números con distribución uniforme y luego dependiendo en que intervalo cae, se genera el valor que corresponde y se almacena en el vector y.

```
function y= generadorFuncionEmpirica()
   numerosGenerados= generadorAleatorioLineal(100000);
   for i = 1:100000
     if numerosGenerados(i)<0.5
       y(end+1)=1;
     elseif 0.5<numerosGenerados(i)<0.7
       y(end+1)=2;
     elseif 0.7<numerosGenerados(i)<0.8
       y(end+1)=3;
     elseif 0.8<numerosGenerados(i)<1
       y(end+1)=4;
     endif

   endfor
endfunction
```

7. Ejercicio 6

7.1. Enunciado

Considerar el siguiente experimento: Lanzar una moneda tantas veces como sea necesario hasta obtener cara. Realizar el experimento 10000 veces indicando:

- ¿A qué tipo de proceso corresponde cada uno de los lanzamientos?
- Con qué distribución conocida se puede modelar cada uno de los experimentos.

- Realizar un histograma mostrando la distribución obtenida

7.2. Resolución

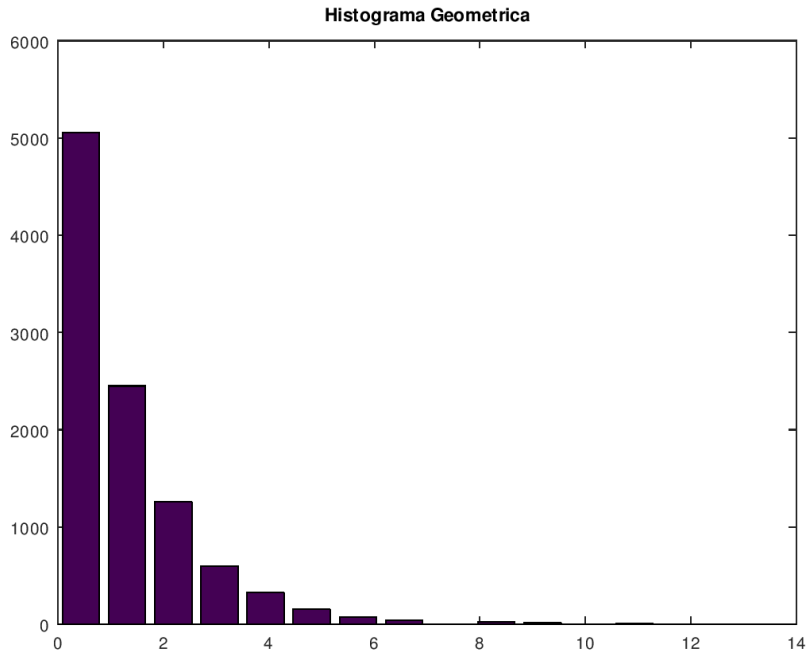

Figura 8: Histograma Geométrica

7.3. Código

```
function y = geometricaMonedas(n)

  p = 0.5;
  for i = 1:n

    cont = 0;
    u = rand();

    while (u < p)
      cont = cont + 1;
      u = rand();
    endwhile

    y(end+1) = cont;

  endfor
endfunction
```

8. Ejercicio 7

8.1. Enunciado

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?

8.2. Resolución

En los gráficos que están a continuación, podemos observar que no se encuentra ningún patrón, tanto para el test espectral de 2 dimensiones como para el de 3 dimensiones.

Por lo tanto, podemos decir que los números generados en el ejercicio 1 son realmente aleatorios porque se puede ver que están distribuidos en una nube de puntos.

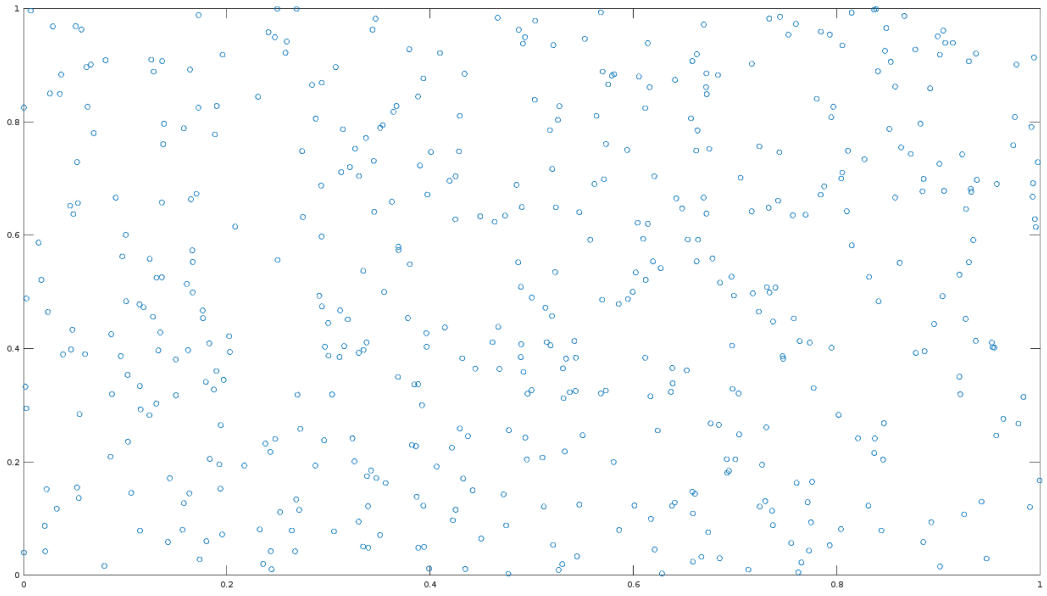

Figura 9: Test espectral 2D

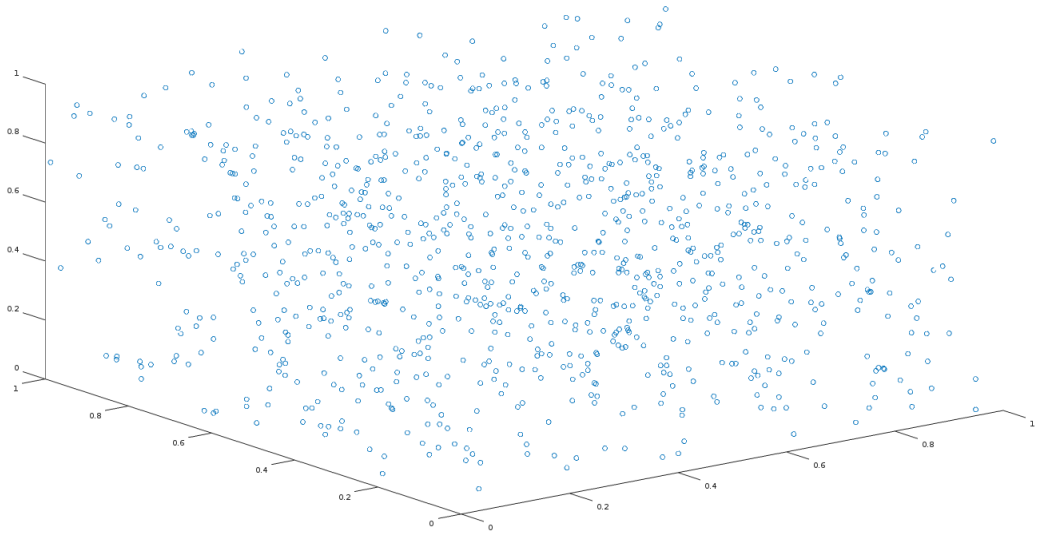

Figura 10: Test espectral 3D

8.3. Código

En los códigos inferiores, se puede observar que se utiliza el GCL y se toman de a 2 o 3 elementos, según corresponda y esos puntos son los que se plotearon en los gráficos anteriores.

```
function testEspectral2D()

  vec = generadorAleatorioLineal(2000);
```

```
  for i = 1:2:1000

    x(end+1) = vec(i);
    y(end+1) = vec(i+1);

  endfor

  plot(x,y,'o')

endfunction

function testEspectral3D()

  vec = generadorAleatorioLineal(9000);

  for i = 1:3:3000

    x(end+1) = vec(i);
    y(end+1) = vec(i+1);
    z(end+1) = vec(i+2);

  endfor

  plot3(x,y,z,'o')

endfunction
```

9. Ejercicio 8

9.1. Enunciado

Realizar un test Chi 2 a la distribución empírica implementada en el Ej 6. Analizar el resultado para un nivel de significación 0,01.

9.2. Resolución

Como se advirtio con antelacion para el ejercicio 6, la distribución generada (esperada) es de tipo geometrica, con p = 0.5. Se comprueba esto aplicando un test Chi Cuadrado con un nivel de significacion del 0.01 que contraste la distribucion generada contra la esperada, dada por la funcion de distribucion

$$P(X = x) = (1 - p)^{x-1}p \tag{1}$$

Octave no cuenta con una funcion para aplicar el mencionado test (al menos con la forma en que disponiamos los datos), por lo que se lo implemento por su definicion, como se ve en el codigo. Se debe comparar el resultado D cuadrado con el cuantil correspondiente a una distribución chi cuadrado, con 9 grados de libertad y significación del 1 % cuyo valor es 21.7 (ver tabla de chi cuadrado en Anexo). Los 9 grados de libertad son el resultado de tomar 10 bins para agrupar los resultados.

Finalmente al realizar el test se obtiene un D cuadrado de 7.69, mucho menor al cuantil mencionado y no se puede rechazar la hipotesis nula de que lo obtenido en el ejercicio 6 es una geometrica con p = 0.5

9.3. Código

El parámetro vec, es un vector de números aleatorios que siguen una distribución geométrica, el parámetro p es la probabilidad de que caiga en el gap indicado y cuant es el nivel de significación con el cual se quiere evaluar el test.

```
function y = chi2test(vec, cuant, p)

  bins = 9;
  N = length(vec);

  % Vector de ocurrencias de cada numero
```

```
  observed = histc(vec, 0:bins)

  % Esperado
  for i = 0:bins
    expected(end+1) = N*((1-p)^i)*p;
  endfor

  expected

  % Formula
  for i = 1:bins
    parcial(end+1) = ((observed(i) - expected(i))^2)/expected(i);
  endfor

  dcuadrado = sum(parcial)

  %Para 9 grados de libertad:
  % 1% ------- 21.7
  % 5% ------- 16.9
  cuantil = cuant;

  if (dcuadrado < cuantil)
    y = "No rechazo hipotesis";
  else
    y = "Rechazo hipotesis";
  end

endfunction
```

10. Ejercicio 9

10.1. Enunciado

Al generador congruencial lineal implementado en el ejercicio 1 realizarle un gap test para los siguientes intervalos:

- Intervalo 1: 0.2-0.6.
- Intervalo 2: 0.5-1.

Analizar los resultados con un nivel de significación del 5 %.

10.2. Resolución

Aplicar un gap test consiste en contar la cantidad de ocurrencias de un generador de números aleatorios desde que un resultado cae dentro de un intervalo hasta que uno de los siguientes lo vuelve a hacer. Se puede observar se tendra un vector resultado cuya distribucion sera nuevamente una geometrica, y la probabilidad de exito es el tamaño relativo del intervalo elegido contra la recta (0,1). Para analizar la validez de esto, se aplica el mismo test Chi Cuadrado del ejercicio anterior con una significacion del 5 % y 9 grados de libertad nuevamente, es decir, se comparara contra un cuantil de valor 16.9 (Ver tabla de chi cuadrado en Anexo).

Aplicado el test se obtiene:

- **Intervalo 1:** $D^2 = 7.21$, no rechazo hipotesis nula de que sea una geometrica con p = 0.4
- **Intervalo 2:** $D^2 = 7.41$, no rechazo hipotesis nula de que sea una geometrica con p = 0.5

10.3. Código

Para cada gap test, se le pasaron los parametros necesarios, en el primer caso, 0.2 y 0.6 y el segundo 0.5 y 1 respectivamente.

```
function y = gaptest(inicio,fin)

  vec = generadorAleatorioLineal(100000);
  j = 0;
  for i = 1:3000
    if ((vec(i) <= fin) && (vec(i) >= inicio))
      y(end +1)=j;
      j=0;
    else
      j=j+1;
    end
  endfor

endfunction
```

El codigo para el test Chi Cuadrado es el mismo que el usado para el ejercicio anterior.

11. Ejercicio 10

11.1. Enunciado

Aplicar el test de Kolmogorov-Smirnov al generador de números al azar con distribución normal generado en el ejercicio 4, y analizar el resultado del mismo para un nivel de significación 0,01. Graficar la distribución acumulada real versus la distribución empírica.

11.2. Resolución

El test de Kolmogorov-Smirnov se basa en calcular la máxima diferencia entre la función acumulativa real y la aproximación empírica a esta función acumulativa, que ésta es obtenida a través de la sumatoria de la cantidad de observaciones que sean menor o igual a un x dividido la cantidad total de muestras que se tienen de la V.A.

El valor obtenido luego de ejecutar el test, a un nivel de significación de 0.01, nos da que se tiene que rechazar H0, lo que nos indica que la aproximación empírica es buena y se encuentra bastante cerca de la función de probabilidad acumulada de la distribución Normal de media 35 y desvío estándar 5.

Por lo tanto, la distribución generada en el ejercicio 4 es buena al nivel de significación que nos indica el test.

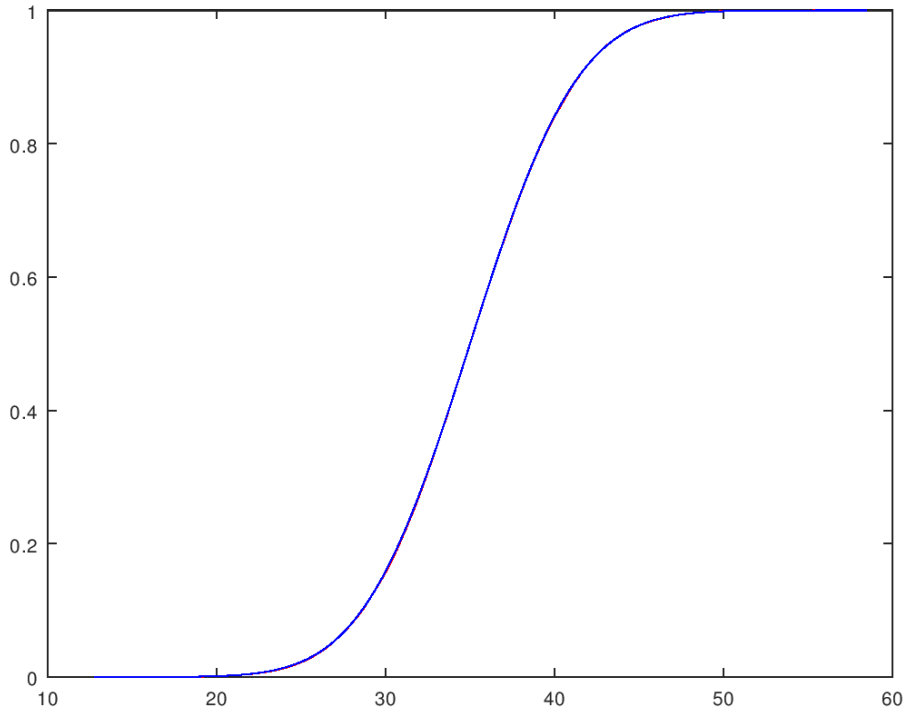

Figura 11: Comparación Distribución acumulada real vs la empírica

Como se puede ver en el gráfico superior, la comparación entre la distribución acumulada real y la empírica es muy exacta, casi que a simple vista no se pueden apreciar diferencias entre ambas funciones, por eso el test de Kolmogorov-Smirnov nos da que la aproximación es buena, para ver en más detalle, la siguiente figura nos muestra en rojo, la distribución acumulada empírica y en azul la real:

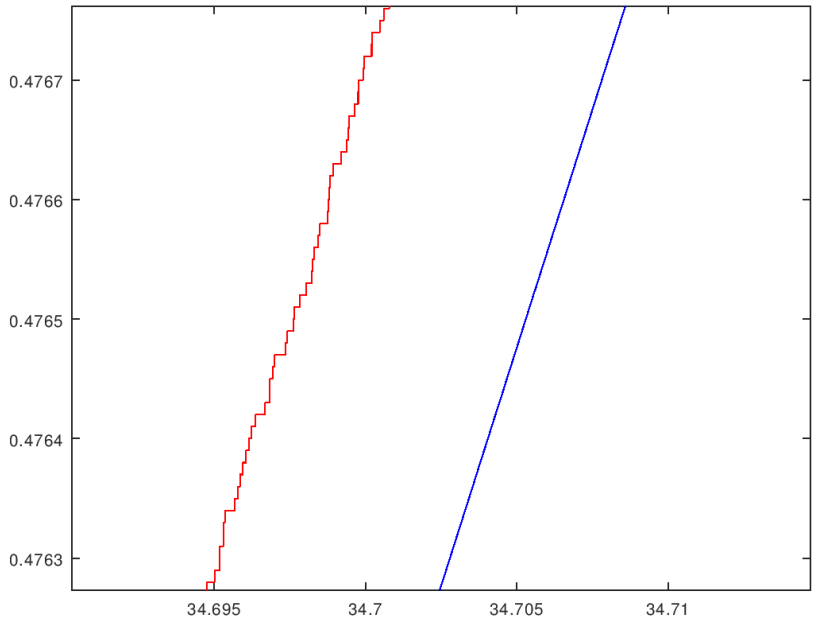

Figura 12: Zoom sobre la comparación

11.3. Código

El parámetro alfa representa el nivel de significación para el test de Kolmogorov-Smirnov, en este caso es 0.01.

Los números generados en el ejercicio 4 son ordenados en orden ascendente, y luego se obtiene un vector que contiene el valor de la función de probabilidad acumulada de la Normal para cada elemento que se encuentra en el vector x.

La variable q representa la máxima distancia entre la función de probabilidad acumulada y la aproximación empírica. Si q es menor a la ecuación dada, se rechaza H0 y la aproximación empírica es buena.

```
function y = testKolmogorovSmirnov(alfa)
  x = generadorNormal(35,5);
  x = sort(x);

  acumulada = normcdf(x,35,5);
  q = 0;
  for i = 1:100000
    q = max(q,abs(acumulada(i)-(i/100000)));
  endfor
    if q > sqrt((-1/200000)*log(alfa/2))
      y = "Acepto H0";
    else
      y = "Rechazo H0";
    endif
endfunction
```