N°1

Trabajo Práctico

75.26 - 95.19

Nicolas Continanza 97576
Mauro Di Pietro 93956
Matias Merlo Gonzalez 104093
Eleonora Luna 96444
Aldana Rastrelli 98408

Tabla de contenidos

01 Generador

Cómo funciona el generador

02 Tests

Evaluación de resultados de los distintos tests

03 Aplicaciones

Distintos usos del generador

$N \to R \quad n \geq n_0 : (x_n - g) < \varepsilon$

01

Generador Xorshift RNGs

Introducción al funcionamiento del generador

Xorshift RNGs

Generar una secuencia de números a partir de la aplicación sucesiva de la operación XOR entre una palabra y desplazamientos de la misma.

Esto es eficiente ya que con pocas líneas de código se pueden generar secuencias de hasta periodo $2^{128}-1$

Detalles de implementación

Para implementar el generador usamos Python. El algoritmo en el paper está especificado para lenguaje C, el cual tiene un rango limitado para sus variables de tipo enteras.
Dado que Python no tiene limitación para valores enteros, simulamos el comportamiento de variables enteras en C al aplicar módulo de 64 bits a cada variable luego de todas las operaciones.
Esto es con la finalidad de simular el overflow que sufren esas variables en C.
Implementamos el generador para tener una secuencia de 2^{64} -1

```python
class xorshift:
 max64bit = (2**64)

 def __init__(self,seed):
   self.seed = seed % self.max64bit

 def rand(self):
   # a=13 b=7 c=17
   self.seed = (self.seed ^ (self.seed << 13)) % self.max64bit
   self.seed = (self.seed ^ (self.seed >> 7)) % self.max64bit
   self.seed = (self.seed ^ (self.seed << 17)) % self.max64bit

   return self.seed

 # Dividimos por el módulo para obtener valores en [0,1]
 def uniform_rand(self):
   return self.rand()/self.max64bit
```

Generador GCL

Generador Congruencial Lineal (GCL) de módulo 232, multiplicador 1013904223, incremento de 1664525 y semilla igual a la parte entera del promedio de los números de padrón de los integrantes del grupo

Generador GCL

$$X_{n+1} = (aX_n + c) \; mod \; m, \quad n >= 0$$

$$m, \; the \; modulus, \; 0 < m$$

$$a, \; the \; multiplier, \; 0 <= a < m$$

$$c, \; the \; increment, \; 0 <= c < m$$

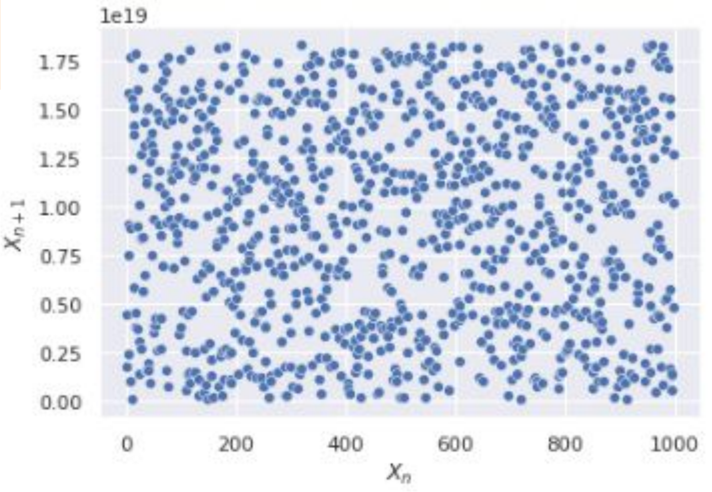
$$X_0, \; the \; starting \; value, \; 0 <= X_0 < m$$

En nuestro caso:

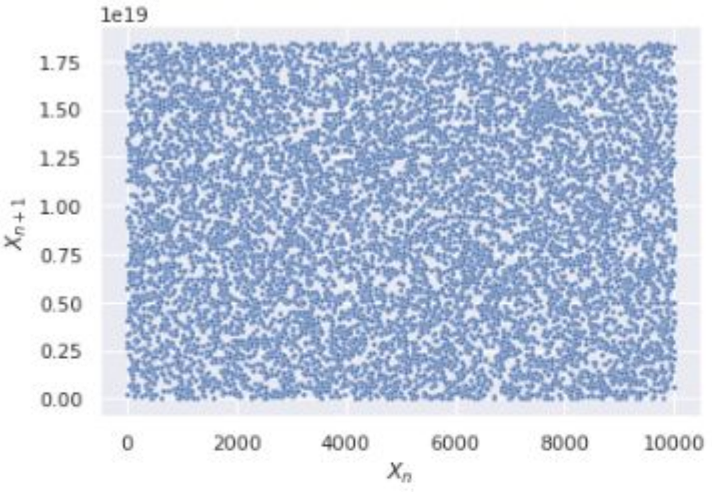
m=232, a=1013904223, c=1664525

Xo=parte entera del promedio de los números de padrón de los integrantes del grupo

Construimos un Xorshift RNG y generamos dos escenarios

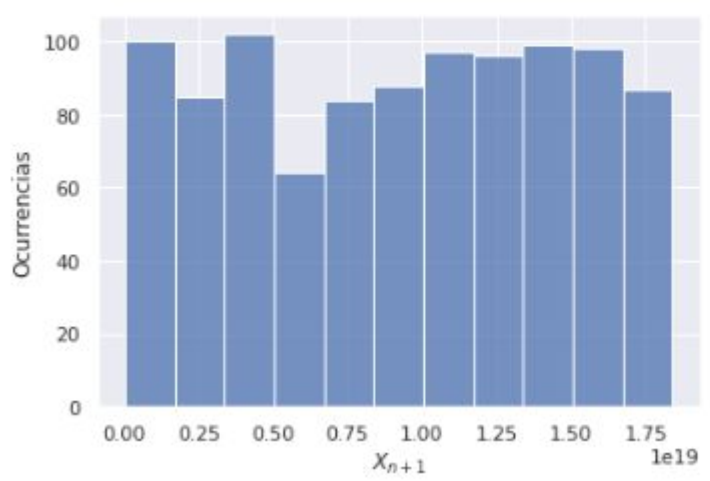


1000 muestras

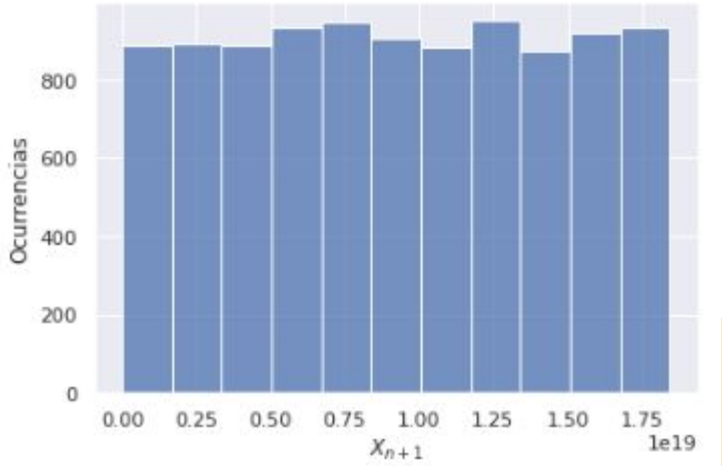


10000 muestras

Construimos un Xorshift RNG y generamos dos escenarios

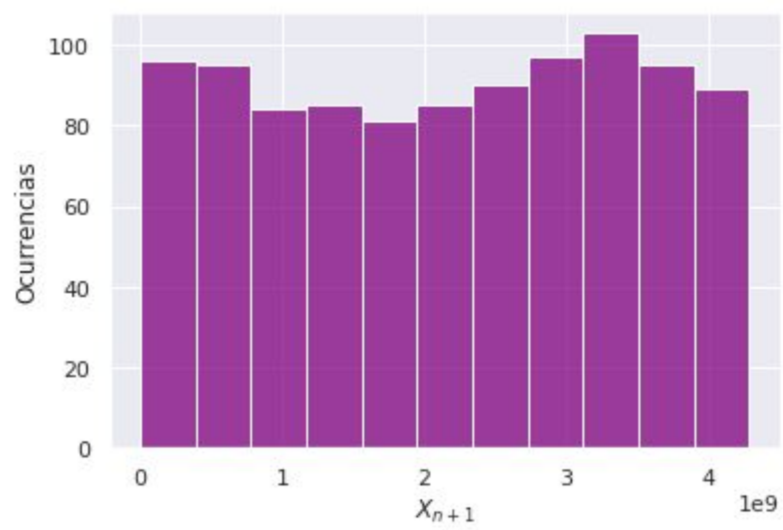


1000 muestras

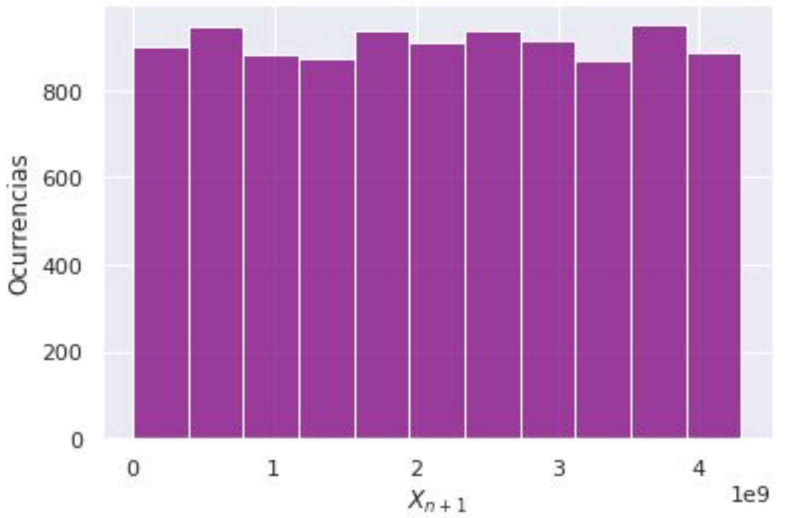


10000 muestras

Construimos un generador GCL y generamos dos escenarios



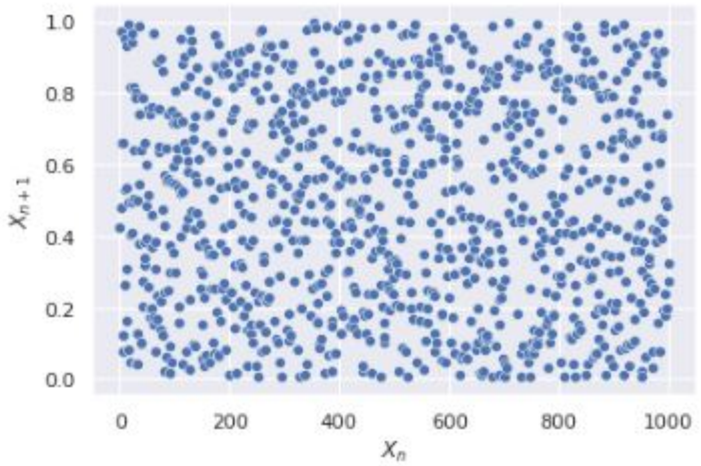
1000 muestras



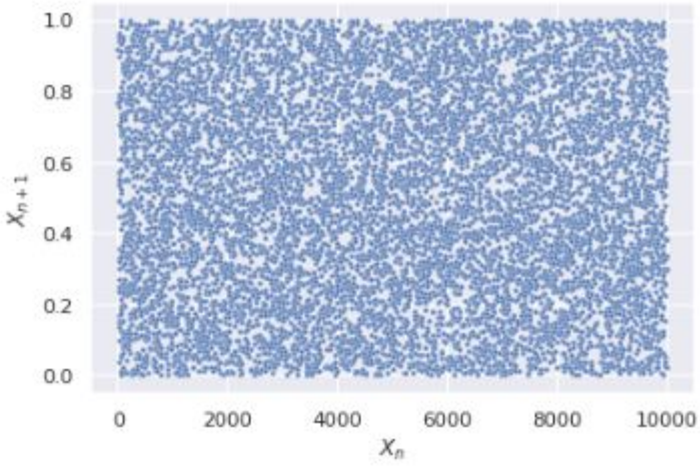
10000 muestras

Construimos un Xorshift RNG y generamos dos escenarios

Esta vez, lo modificamos para obtener números en [0,1]



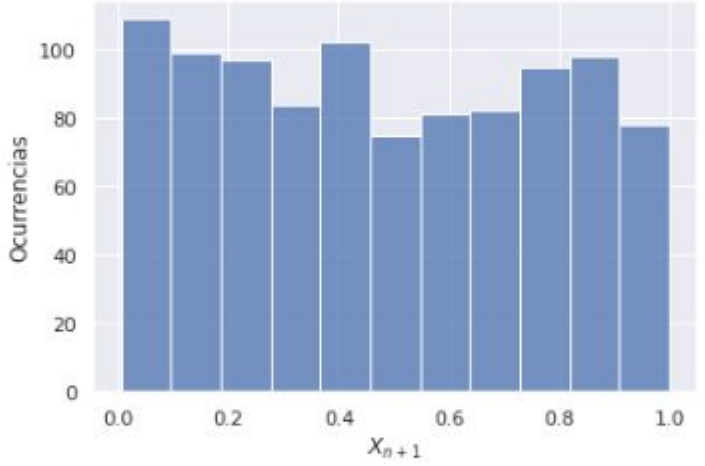
1000 muestras



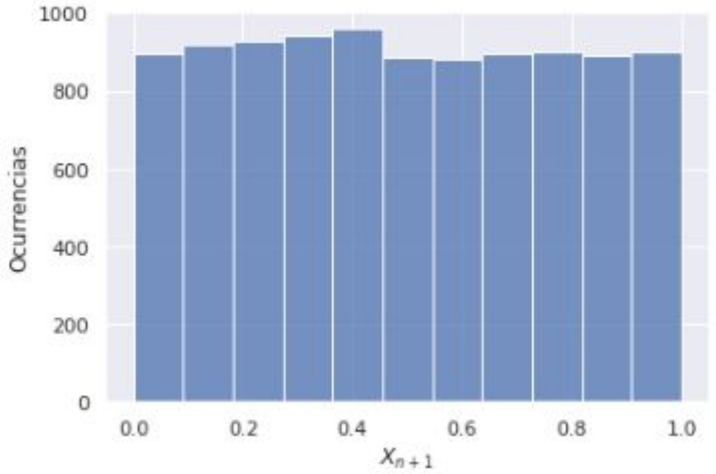
10000 muestras

Construimos un Xorshift RNG y generamos dos escenarios

Esta vez, lo modificamos para obtener números en [0,1]



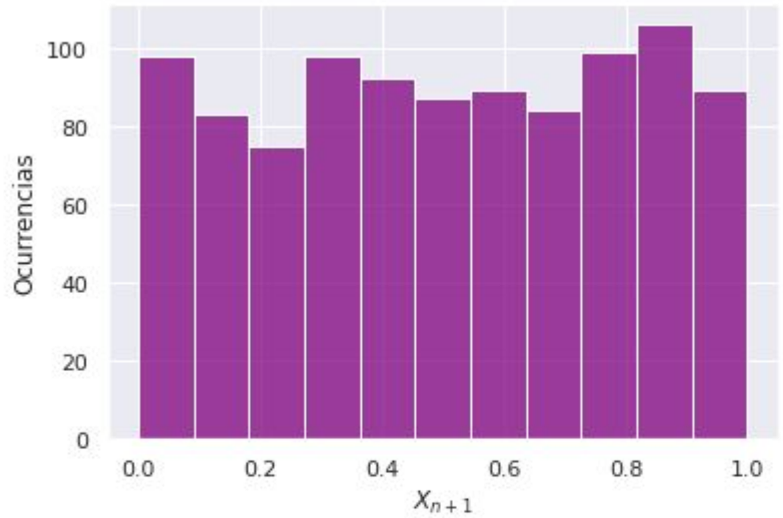
1000 muestras



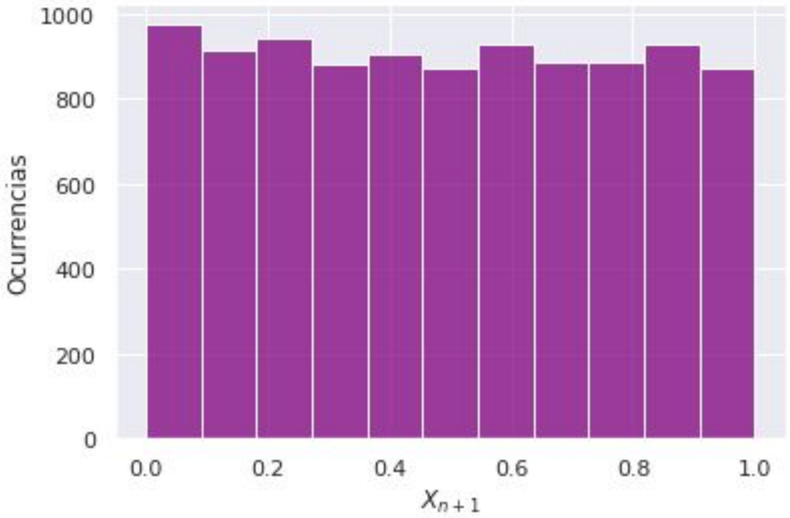
10000 muestras

Construimos un generador GCL y generamos dos escenarios

Esta vez, lo modificamos para obtener números en [0,1]



1000 muestras



10000 muestras

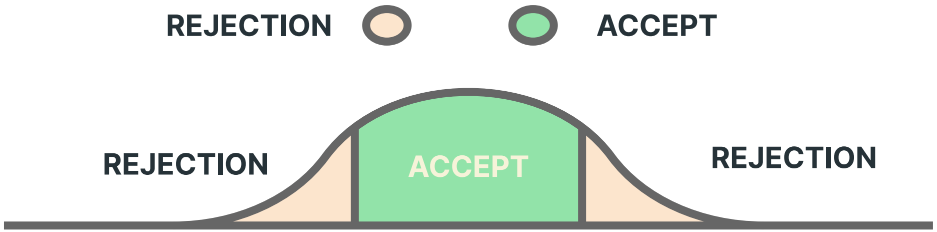
02

Tests de hipótesis

Test de Chi2

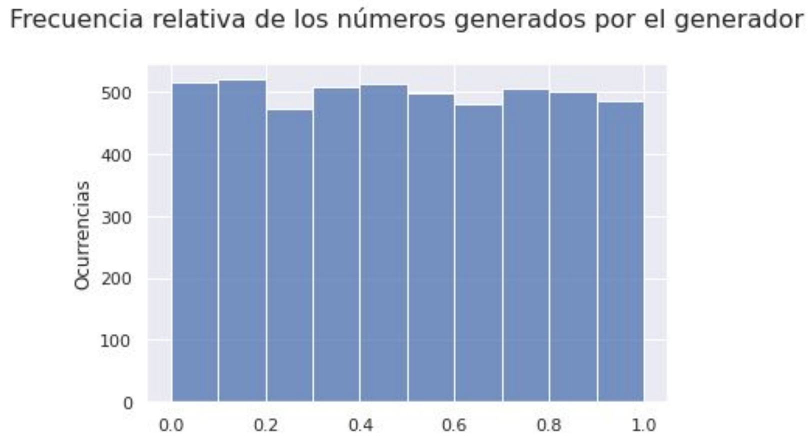
Discretizar la muestra

Para el Test de Chi2 lo que hacemos es discretizar la muestra, de esta forma vamos a tener valores por rangos.



Test de Chi2 (con el generador Xorshift RNG)

500 muestras



5000 muestras

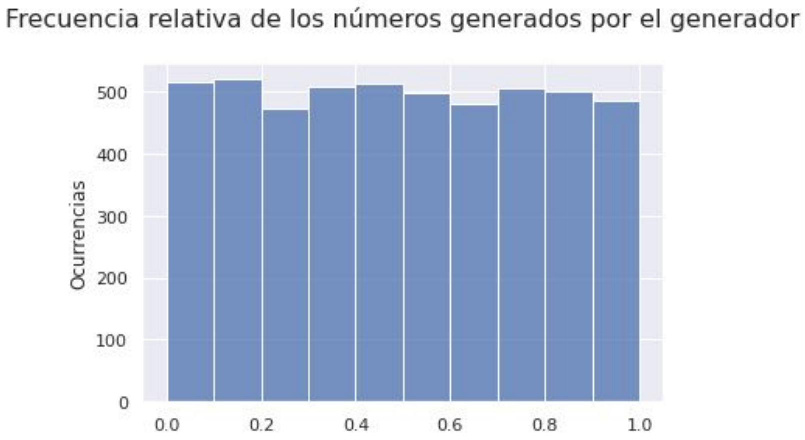
Test de Chi2 (con el generador GCL (opcional))

500 muestras

5000 muestras

Test de Chi2 (con el generador Xorshift RNG)

Realizamos el test de Chi2 con un grado de significación de 0.05



Con 500 muestras

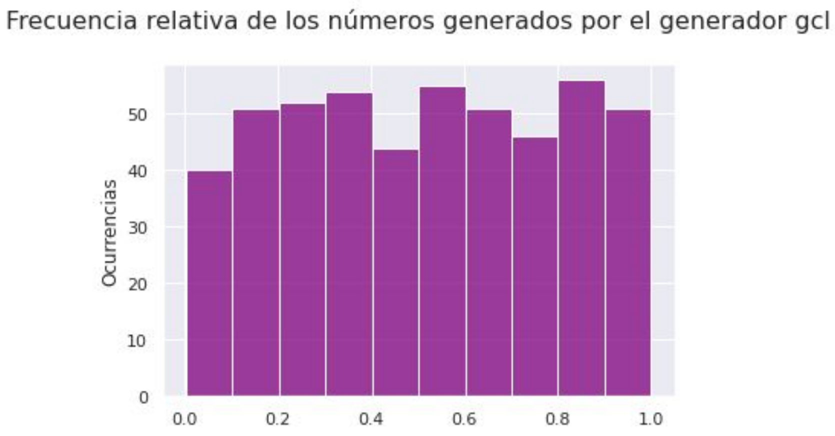
Estadistico: 8.41
El test acepta la hipotesis nula.

Con 5000 muestras

Estadistico: 3.83
El test acepta la hipotesis nula.

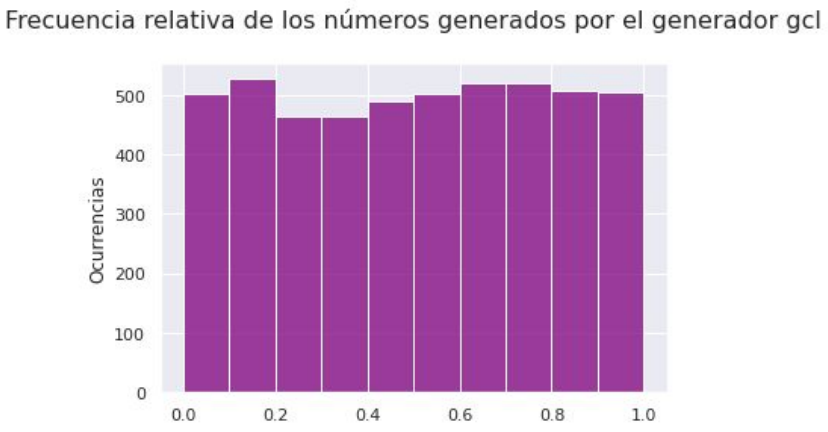
Test de Chi2 (con el generador GCL (opcional))

Realizamos el test de Chi2 con un grado de significación de 0.05



Con 500 muestras

Estadistico: 5.32
El test acepta la hipotesis nula.



Con 5000 muestras

Estadistico: 8.41
El test acepta la hipotesis nula.

Test de Kolmogorov (con el generador Xorshift RNG)

Realizamos el test de Kolmogorov usando una Distribución Uniforme, con un grado de significación de 0.05

Con 500 muestras

El test acepta la hipotesis nula.

Estadistico: 0.04
P-Valor: 0.50

Con 5000 muestras

El test acepta la hipotesis nula.

Estadistico: 0.01
P-Valor: 0.27

Test de Kolmogorov (con el generador GCL (opcional))

Realizamos el test de Kolmogorov usando una Distribución Uniforme, con un grado de significación de 0.05

Con 500 muestras

El test acepta la hipotesis nula.

Estadistico: 0.04
P-Valor: 0.57

Con 5000 muestras

El test acepta la hipotesis nula.

Estadistico: 0.01
P-Valor: 0.27

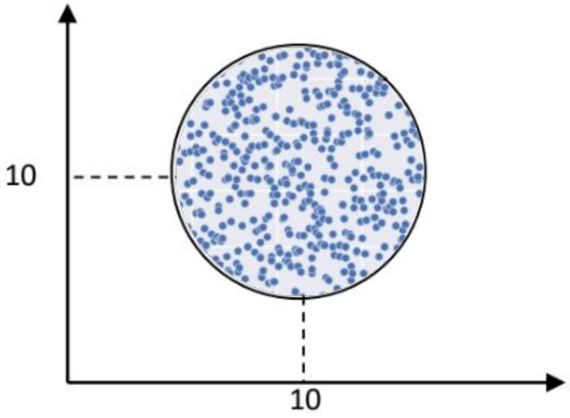
03

Aplicaciones

Distintos usos del generador

Distribución Circular (Ej3)

Se desea generar puntos al azar con distribución uniforme dentro del área descrita en el gráfico utilizando los siguientes generadores de números al azar



Distribución Circular

La forma circular se puede expresar como:

$$(x - 10)^2 + (f_X(x) - 10)^2 = 5^2$$

Y se puede descomponer en dos funciones:

$$f_{1X}(x) = +\sqrt{25 - (x - 10)^2} + 10$$

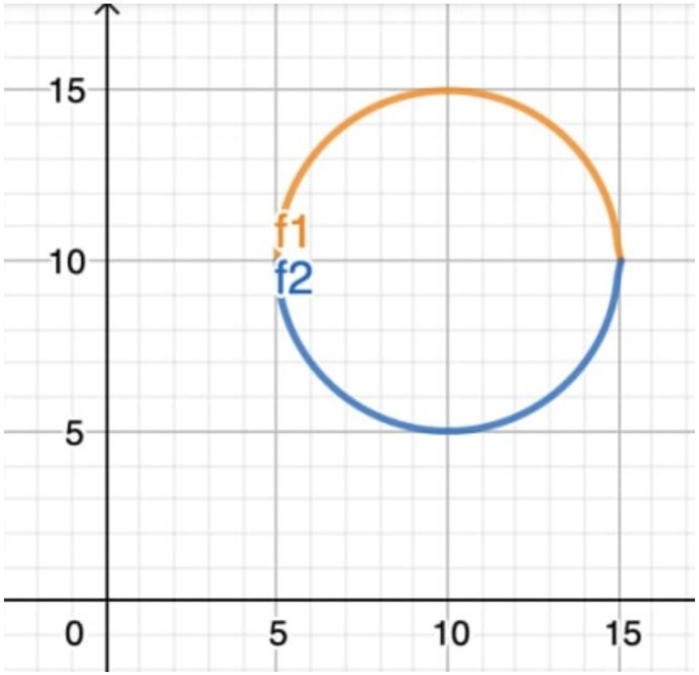
$$f_{2X}(x) = -\sqrt{25 - (x - 10)^2} + 10$$

Distribución Circular

$$f_{1X}(x) = +\sqrt{25 - (x - 10)^2} + 10$$

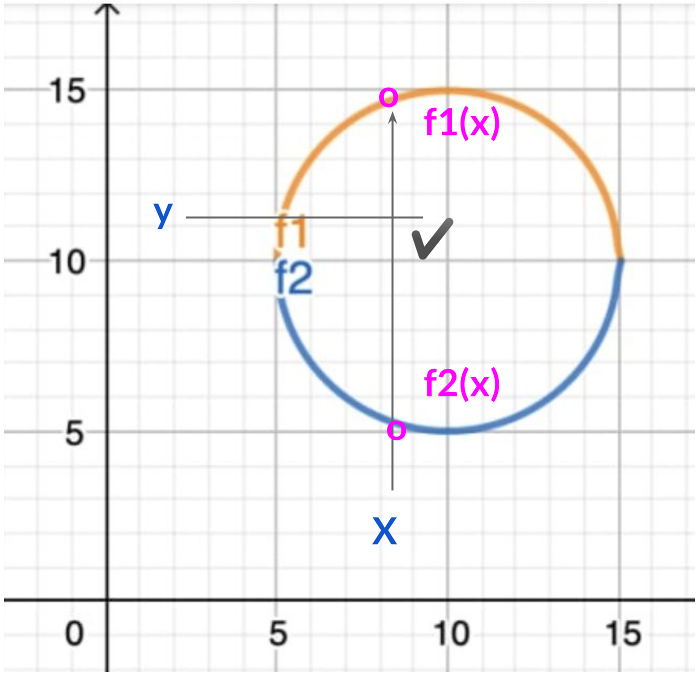
$$f_{2X}(x) = -\sqrt{25 - (x - 10)^2} + 10$$

$$f_{1_X}(x) \leq y \leq f_{2_X}(x)$$



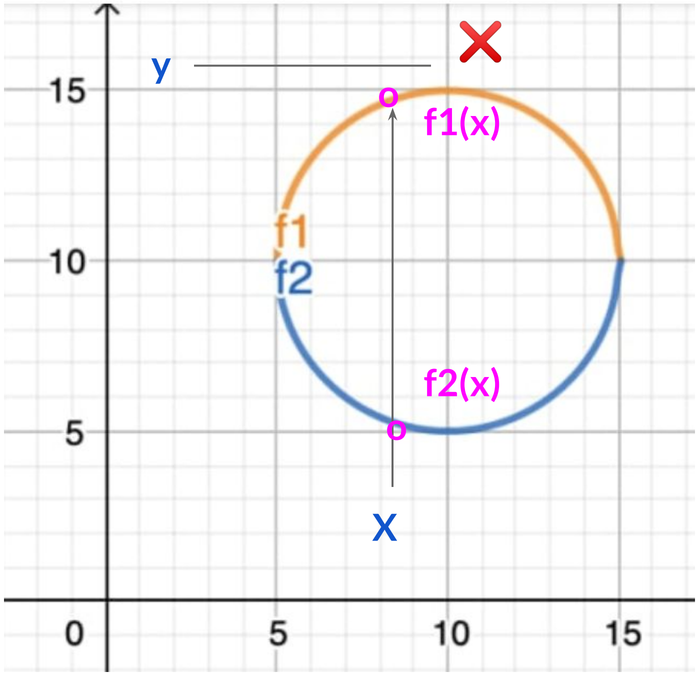
Distribución Circular

```
x = random_func()
y = random_func()

if f1(x) >= y and y >= f2(x):
      guardar(x)
```



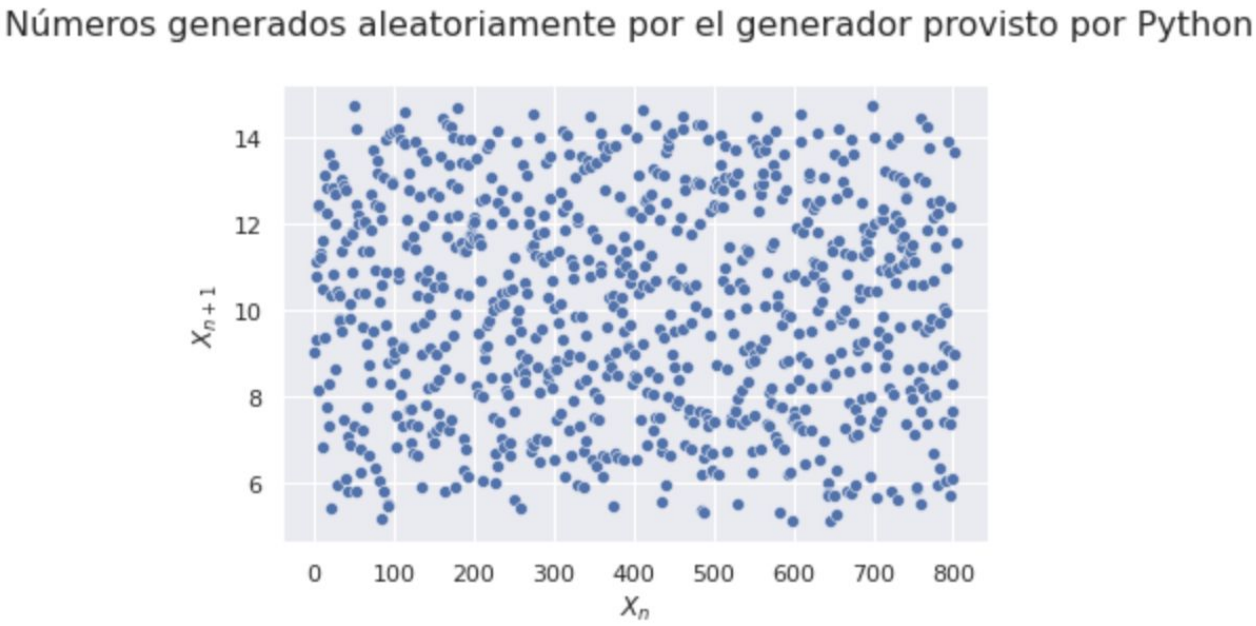
Distribución Circular

```
x = random_func()
y = random_func()

if f1(x) >= y and y >= f2(x):
      guardar(x)
```



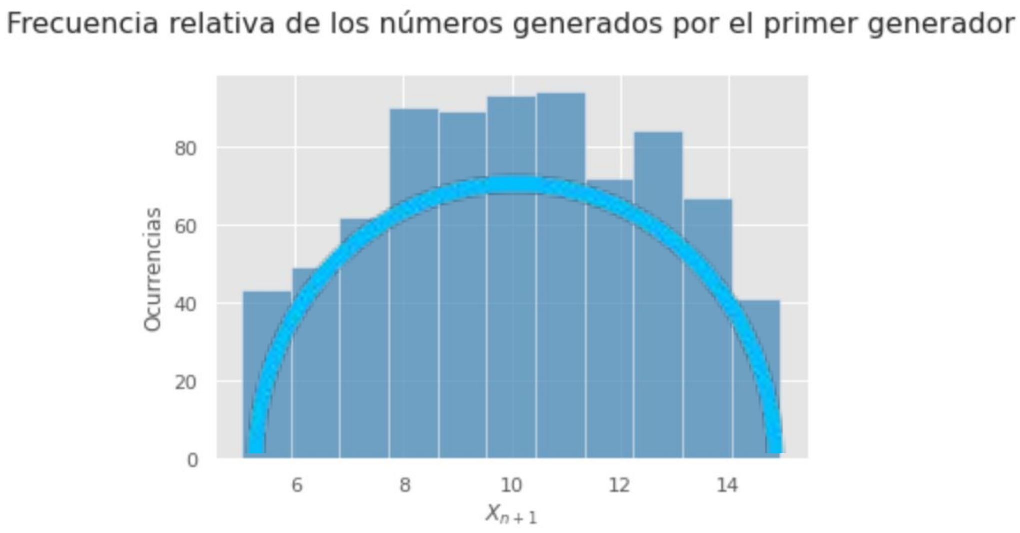
Distribución Circular

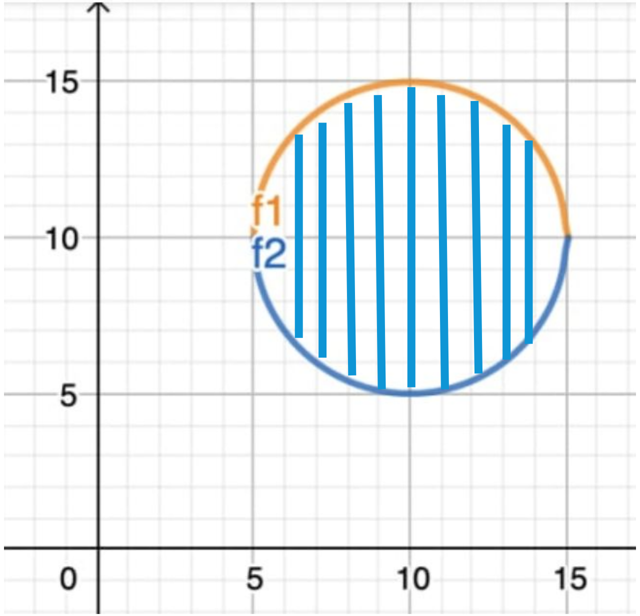
Generador provisto por Python



Distribución Circular

Generador provisto por Python





Distribución Circular

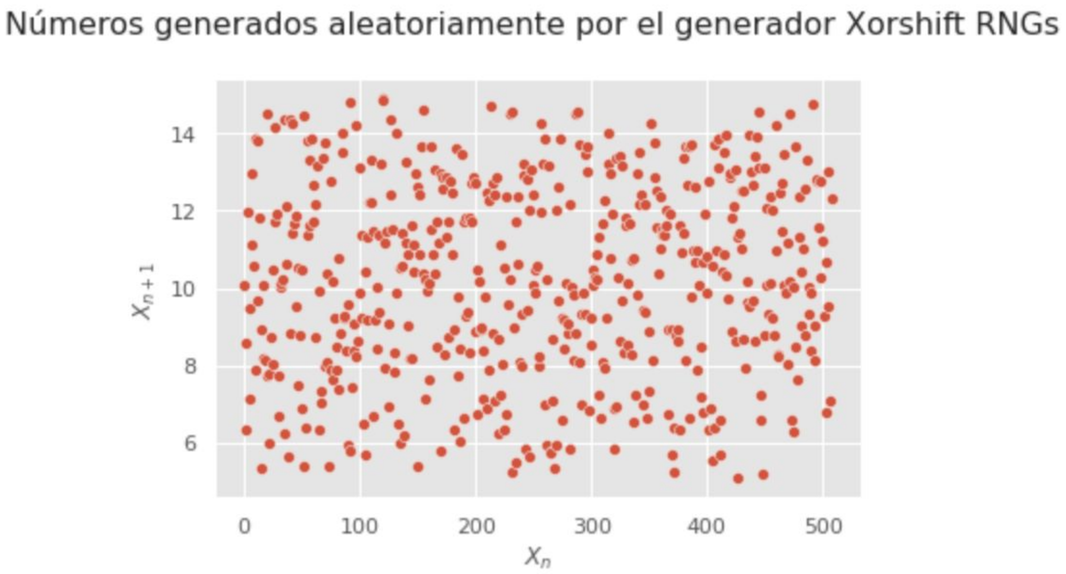
Generador provisto por Python

Por lo tanto, $Factor\ de\ rendimiento =$

```
[163] num_val / 1000

      0.784
```

Distribución Circular

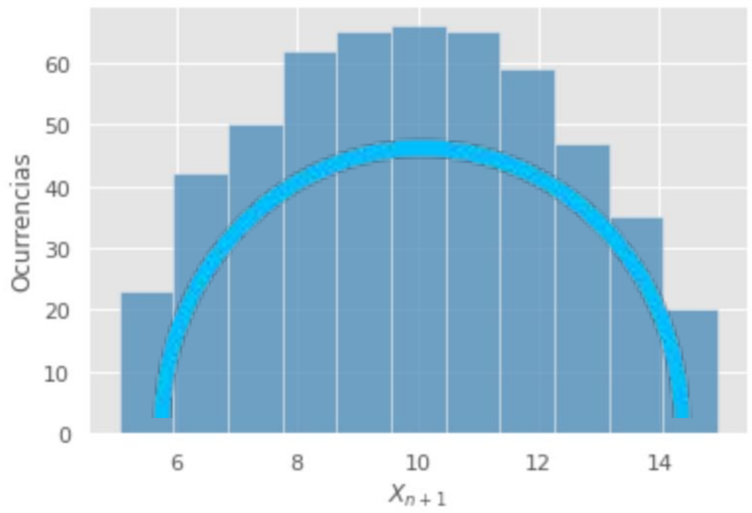
Generador Xorshift RNGs



Distribución Circular

Generador Xorshift RNGs

Frecuencia relativa de los números generados por el segundo generador



Distribución Circular

Generador Xorshift RNGs

Por lo tanto, $Factor\ de\ rendimiento =$

```
[169] num_val2 / 10000

0.0514
```

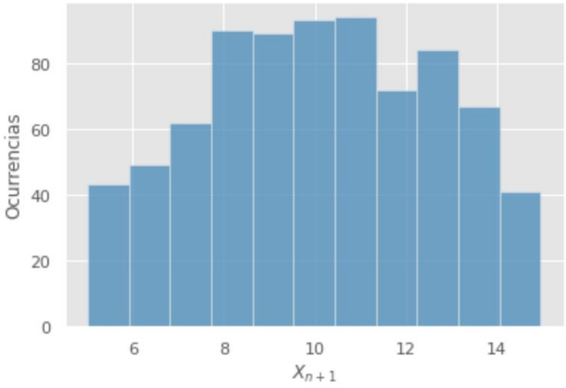
Distribución Circular

Generador Python

Por lo tanto, $Factor\ de\ rendimiento =$

```
[163] num_val / 1000
```

```
0.784
```



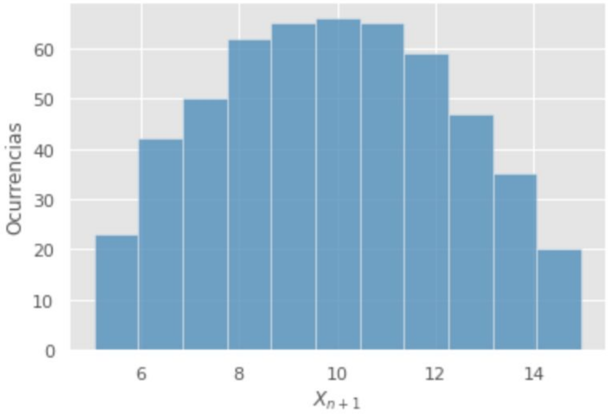
Generador Xorshift

Por lo tanto, $Factor\ de\ rendimiento =$

```
[169] num_val2 / 10000
```

0s

```
0.0514
```



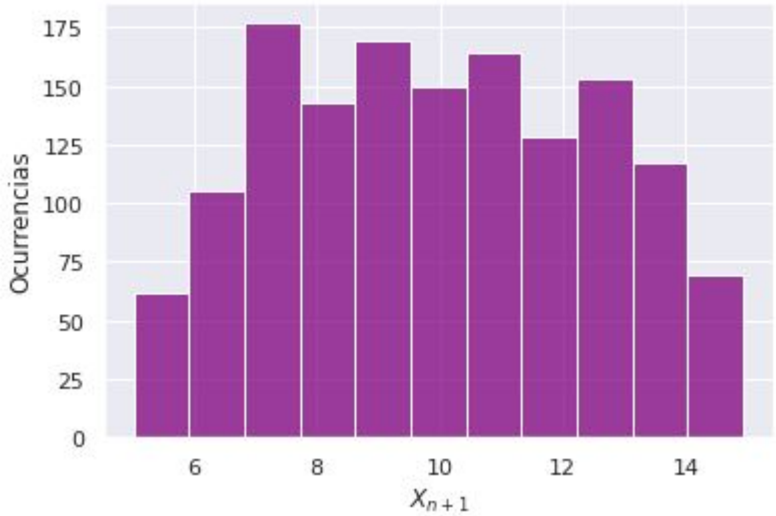
Distribución Circular

Generador GCL (10000 muestras)

Por lo tanto, $Factor\ de\ rendimiento =$

```
[ ] num_val_gcl / 10000
```

```
0.1437
```

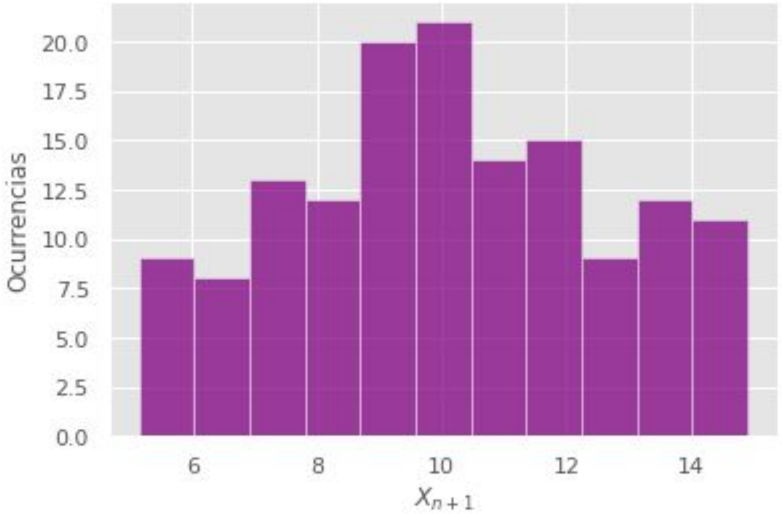


Generador GCL (1000 muestras)

Por lo tanto, $Factor\ de\ rendimiento =$

```
num_val_gcl / 1000
```

```
0.144
```



Distribución Circular

Generador Python

Por lo tanto, $Factor\ de\ rendimiento =$

```
[163] num_val / 1000
```

```
0.784
```

Generador Xorshift

Por lo tanto, $Factor\ de\ rendimiento =$

```
[169] num_val2 / 10000
```

```
0.0514
```

Generador GCL (1000 muestras)

Por lo tanto, $Factor\ de\ rendimiento =$

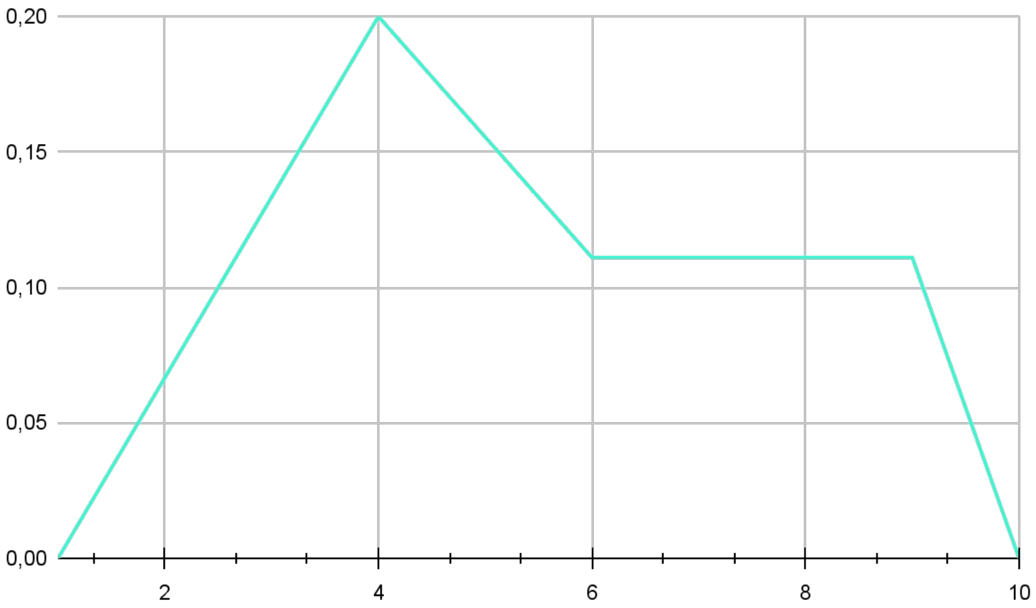
```
num_val_gcl / 1000
```

```
0.144
```

El generador GCL logra casi el mismo factor de rendimiento con 1000 que con 10000 muestras. A diferencia del generador Xorshift que necesita muchas más muestras para lograr un resultado apenas mejor.
El generador provisto por Python resulta ser significativamente mejor que los mencionados anteriormente

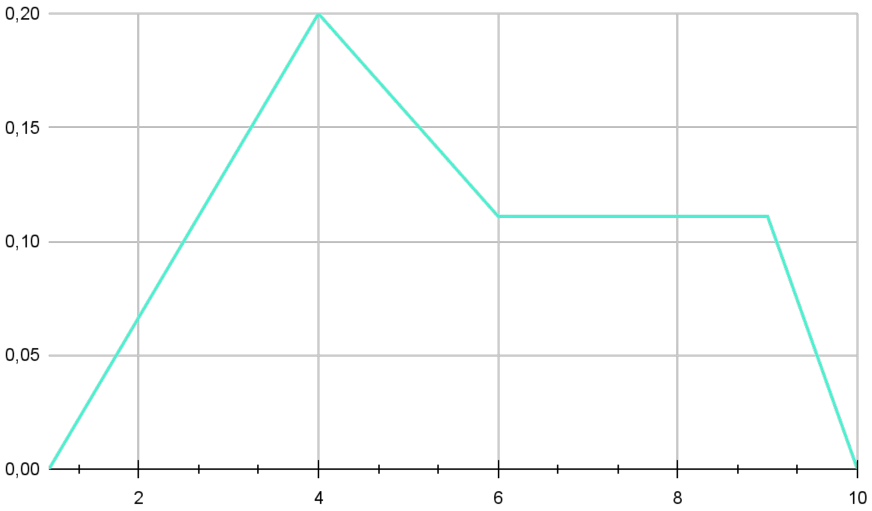
Distribución particular

Función de densidad $f(x)$



Distribución particular

Función de densidad $f(x)$



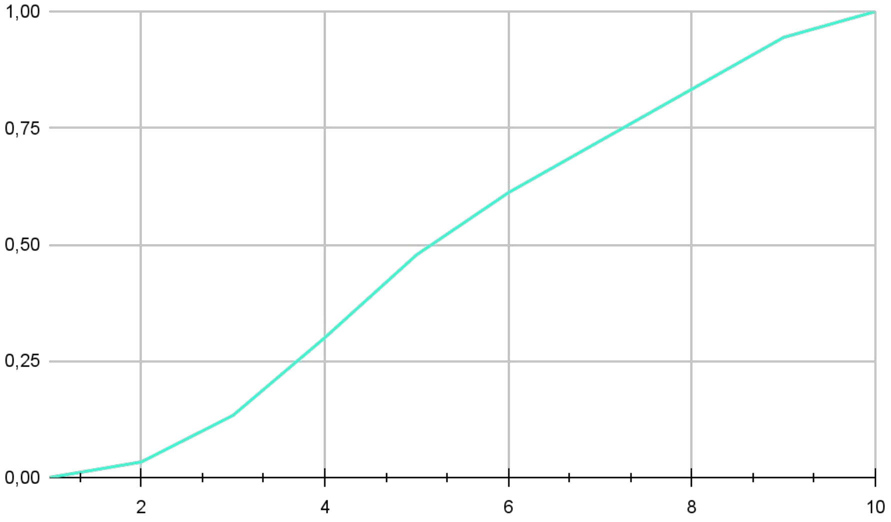
$$f_X(x) = \begin{cases} 1/15x - 1/5, & 1 <= x < 4 \\ -2/45x + 17/45, & 4 <= x < 6 \\ 1/9, & 6 <= x < 9 \\ -1/9x + 10/9, & 9 <= x <= 10 \\ 0 & \textit{en otro caso} \end{cases}$$

Distribución particular

$$\lim_{n \to \infty} \frac{n^2 - x}{3}$$

Función de prob acum

$$F_X(x) = \begin{cases} 0 & x < 1 \\ 1/30x^2 - 1/15x + 1/30 & 1 <= x < 4 \\ -1/45x^2 + 17/45x - 77/90 & 4 <= x < 6 \\ 1/9x - 1/18 & 6 <= x < 9 \\ -1/18x^2 + 10/9x - 41/9 & 9 <= x <= 10 \\ 1 & x > 10 \end{cases}$$

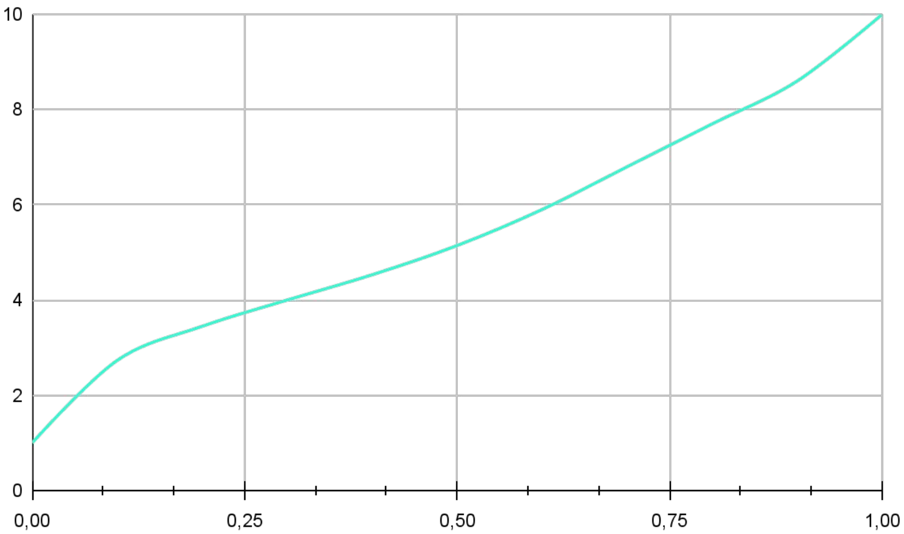


Distribución particular

Transformada inversa

$$\lim_{n\to\infty}\frac{n^2-x}{3}$$

$$\frac{(|a|-5)^2}{5|a|}$$



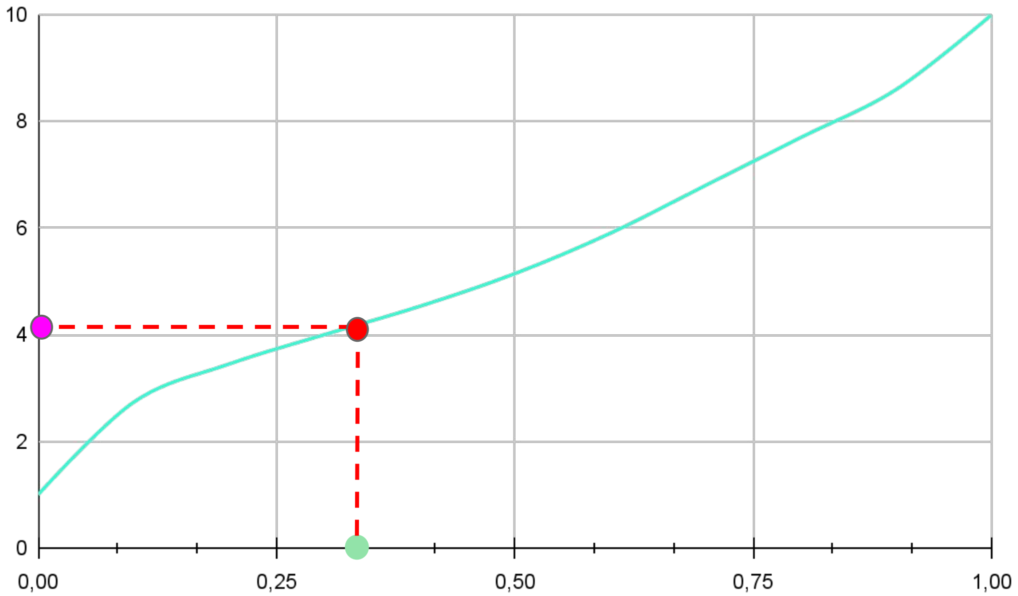
$$F_X^{-1}(u) = \begin{cases} \sqrt{30u} + 1 & 0 <= u < 3/10 \\ \frac{-\sqrt{-720u+540}+34}{4} & 3/10 <= u < 11/18 \\ 9u + 1/2 & 11/18 <= u < 17/18 \\ 3\sqrt{-2u+2} + 10 & 17/18 <= u <= 1 \end{cases}$$

Distribución particular

$$\lim_{n \to \infty} \frac{n^2 - x}{3}$$

$$\frac{(|a|-5)^2}{5|a|}$$

Transformada inversa



Método de la transformada inversa

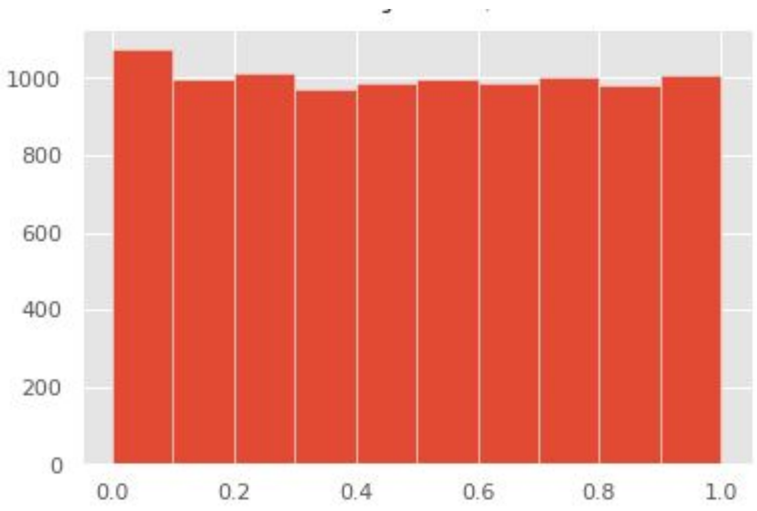
1- Generamos números al azar distribuidos uniformemente en el [0,1] con el generador del punto 1

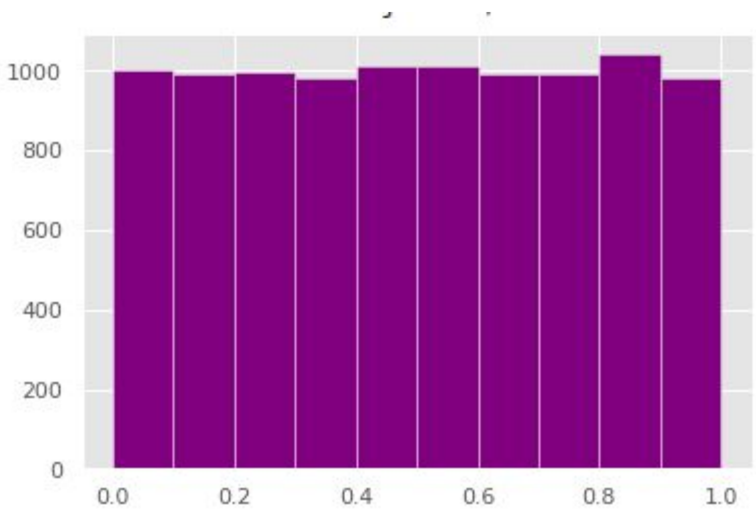
2- Aplicamos la transformada inversa a esos números para que tengan la distribución deseada

$f(x)$

Distribución particular

Distribución de los números aleatorios [0,1] generados por cada generador





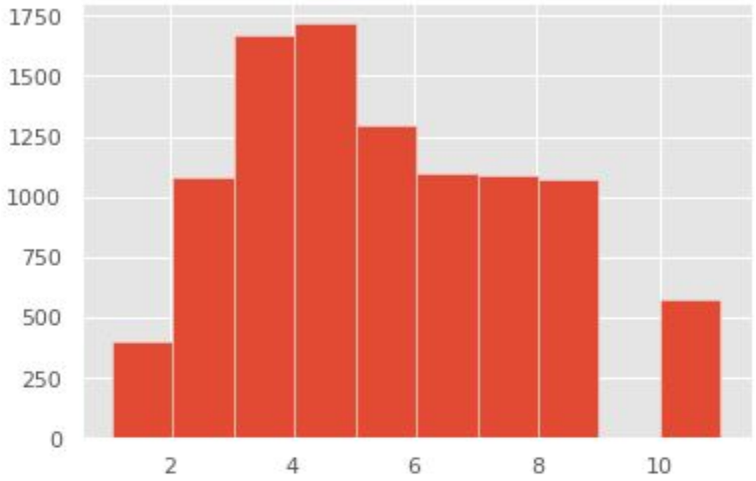
Generador Xorshift

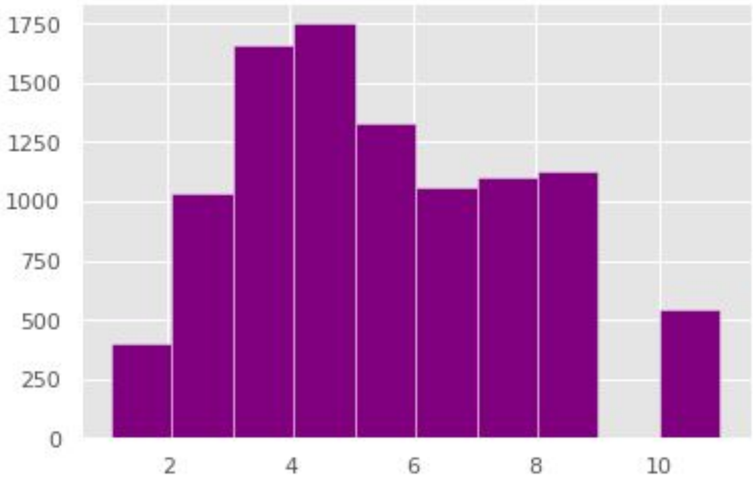
Generador GCL (opcional)

Distribución particular

Distribución de los números aleatorios generados por cada generador luego de aplicar la transformada inversa

No se evidencian diferencias significativas utilizando uno u otro generador





Generador Xorshift

Generador GCL (opcional)

Simulación de Distribución normal

$$\lim_{n \to \infty} \frac{n^2 - x}{3}$$

$$\frac{(|a|-5)^2}{5|a|}$$

Usamos el generador para simular variables aleatorias con distribución normal.
Para esto decidimos usar el método de superposición.
Sabemos que la esperanza de una suma de variables aleatorias será la suma de las esperanzas y que lo mismo ocurre con las varianzas. A su vez conocemos la varianza y esperanza de una variable aleatoria uniforme continua.

$$E[x] = \frac{a+b}{2}$$

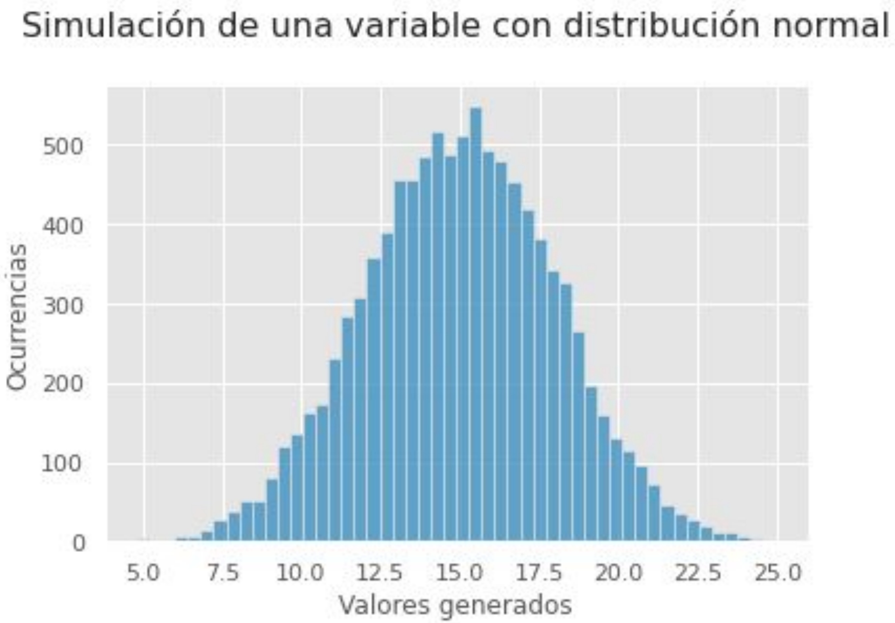
$$Var[x] = \frac{(b-a)^2}{12}$$

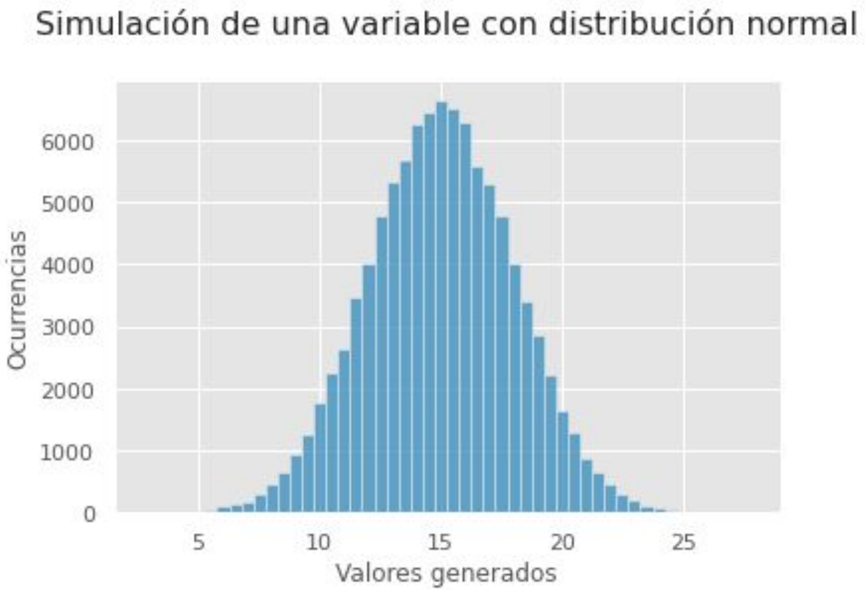
$$E[X] = \sum_{i=1}^{n} E[x_i] = n\,E[x]$$
$$Var[X] = \sum_{i=1}^{n} Var[x_i] = n\,Var[x]$$

$$E[X] = n\,\frac{a+b}{2} = 15$$
$$Var[X] = n\,\frac{(b-a)^2}{12} = 9$$

$$E[X] = 12\,\frac{a+b}{2} = 15 \Rightarrow 6\,(a+b) = 15$$
$$Var[X] = 12\,\frac{(b-a)^2}{12} = 9 \Rightarrow (b-a)^2 = 9 \Rightarrow (b-a) = 3$$

$$a = -\frac{1}{4} \;\; b = \frac{11}{4}$$





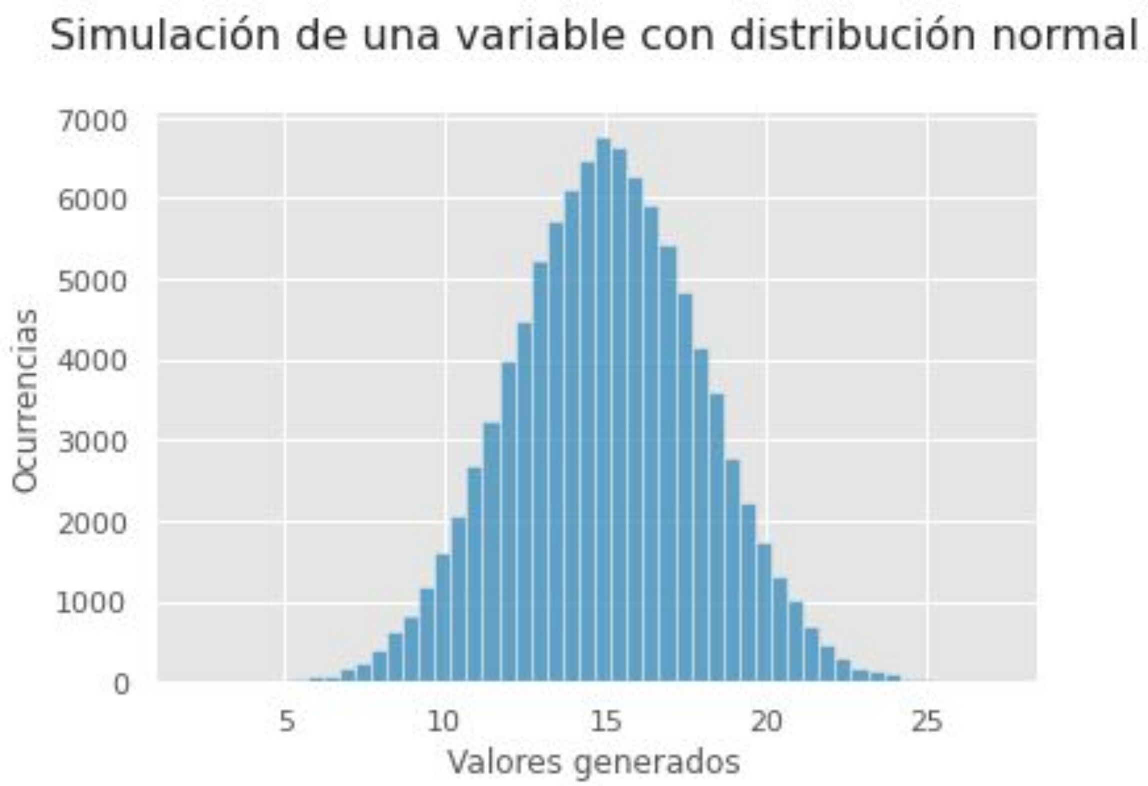
10.000 muestras
Pasa el test chi2 y el test de Anderson-Darling

100.000 muestras
No pasa test chi2 ni el de Anderson-Darling

Probemos sumando 30 variables en vez de 12

$$a = -\frac{\sqrt{\frac{18}{5}+1}}{2}+1$$

$$b = \frac{\sqrt{\frac{18}{5}+1}}{2}$$



Muestra de tamaño 100.000
Pasa el test chi2 y el de Anderson-Darling

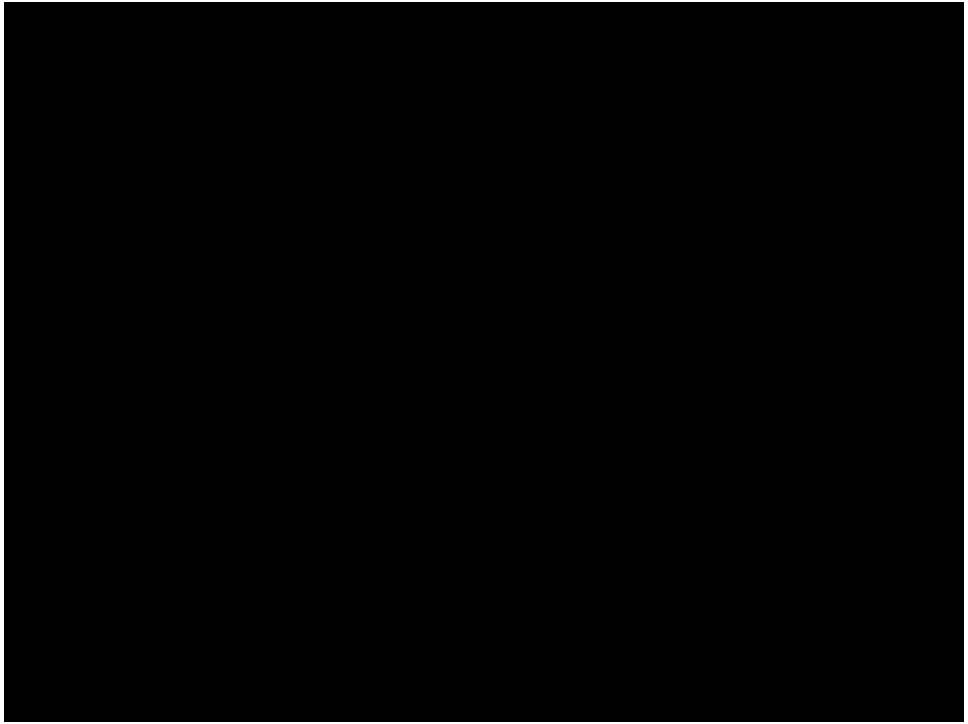
Ejercicio de aplicación

Simulación de desplazamiento de partículas

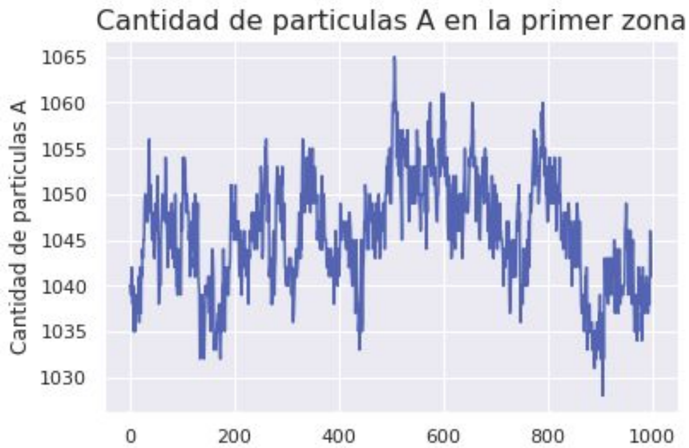
Tipo	%	Características
A	**70**	Se desplazará 1 celda por instante de tiempo
B	**25**	Se desplazará 1 celda cada 2 instantes de tiempo
C	**5**	Se desplazará 1 celda cada 4 instantes de tiempo

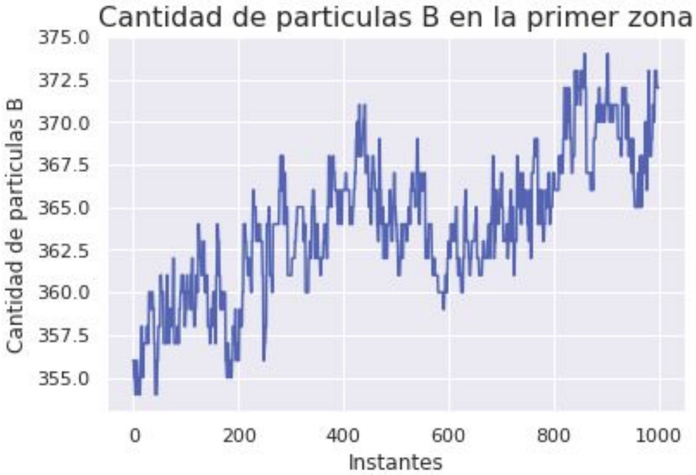
Ejercicio de aplicación

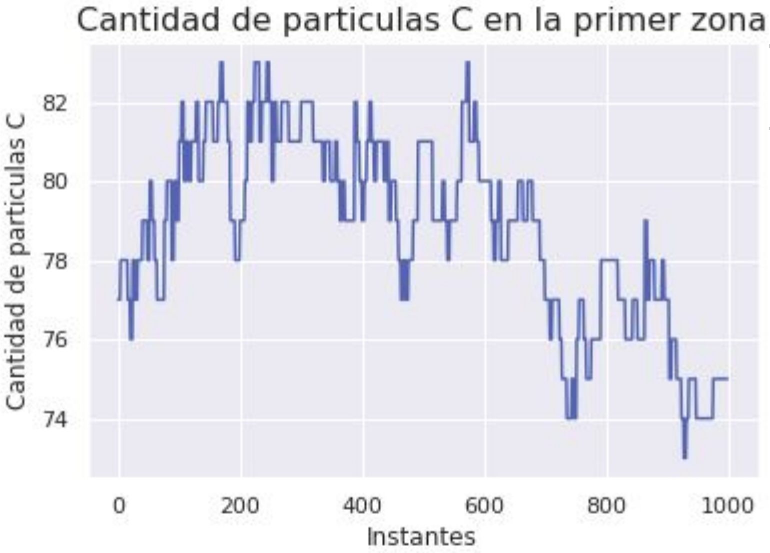
Simulación de desplazamiento de 3000 partículas



Ejercicio de aplicación







Gracias