Random Number Generator with Schrage's Algorithm

2017135002 최성윤

목차

- 목표
- 기본원리(Schrage’s Algorithm)
- 실행
- 결과
- 토의사항

목표

Schrage’s algorithm을 사용하여 난수를 발생 시키는

Pseudo-Random Number Generator를 제작한다

기본원리(Schrage’s Algorithm)

Linear Congruence Method 식에서 overflow를 발생시키지 않기 위하여 일부 변형이 된 Schrage’s Algorithm을 사용한다

(overflow란? 연산의 결과가 컴퓨터에서 다르는 범위를 초과 했을 때)
Ex)32bit를 다루는 컴퓨터에서는 수가 2^{31}을 초과하는 경우

$$X_{n+1} = (A\ X_n + B) \bmod C$$

Linear Congruence Method

$$A\ X_n \bmod C = A(X_n \bmod Q) - R[X_n/Q]$$

그런데 결과가 0 보다 작으면,

Schrage’s Algorithm

$$A\ X_n \bmod C = A(X_n \bmod Q) - R[X_n/Q] + C$$

C=A*Q+R이다. (A=16807 Q=127773 R=2836 C=2147483647)
초기값 X=1로 계산을 진행한다

실행

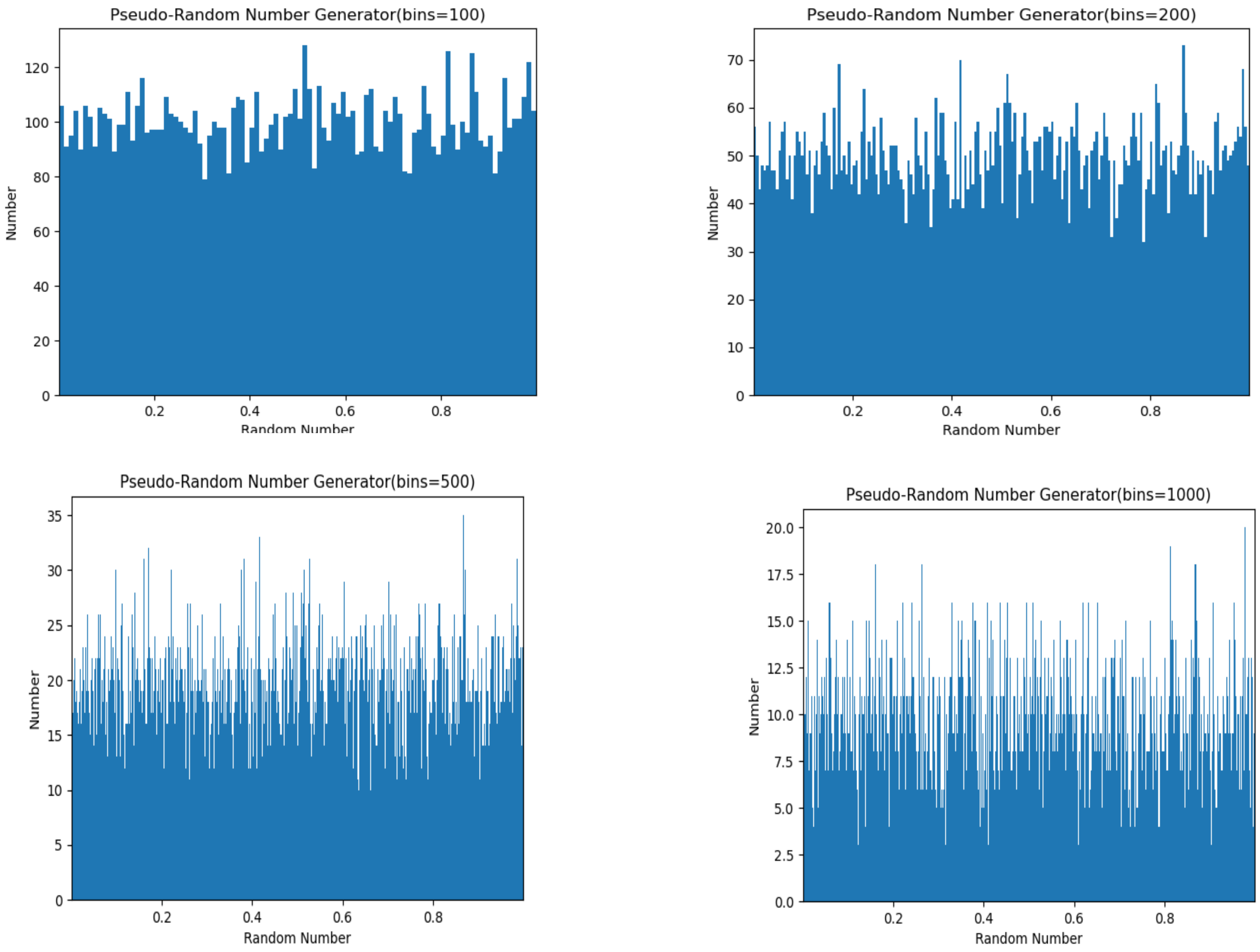
```python
import matplotlib.pyplot as plt
A=16807
C=2147483647
Q=127773
R=2836
X=1
N=[]
T=[]
for i in range(10000): # 10000번 반복하는 for문
    X=A*(X%Q)-R*(X//Q)  # %은 나머지를 반환하는 연산자, //은 몫을 반환하는 연산자
    if X<0:     # X값이 음수인경우 C를 더하여 양수를 만들어준다
        X=X+C
    N.append(X)   # N리스트에 X값 추가
M=max(N) # 0과 1사이의 난수값을 발생시키위하여 N리스트에 있는 값을 최대값으로 나누어준다
for i in N:
    T.append(i/(M+1))  # N리스트를 0과 1사이의 변수로 만들어 새로운 리스트에 넣어준다

plt.hist(T,bins=200)
plt.title("Pseudo-Random Number Generator")
plt.xlabel("Random Number")
plt.ylabel("Number")
plt.xlim(min(T),max(T))  # x축 범위를 T리스트내의 최소값 최대값으로 설정한다
plt.show()
```

→난수 발생을 위하여 Schrage's Algorithm을 사용하였다

→ Schrage's Algorithm으로 발생한 난수들을 히스토그램으로 표현한다

bins값(막대의 영역)을 바꾸어 가며 다양한 히스토그램을 표현한다

결과


Pseudo-Random Number Generator(bins=100)
Pseudo-Random Number Generator(bins=200)
Pseudo-Random Number Generator(bins=500)
Pseudo-Random Number Generator(bins=1000)
Number
Random Number
0.2
0.4
0.6
0.8

토의사항

1. 충분히 균일한 (uniform)한 분포가 나타났는가?

1)

```
A=0
for i in T:
    A=A+i
print(A/10000)
```

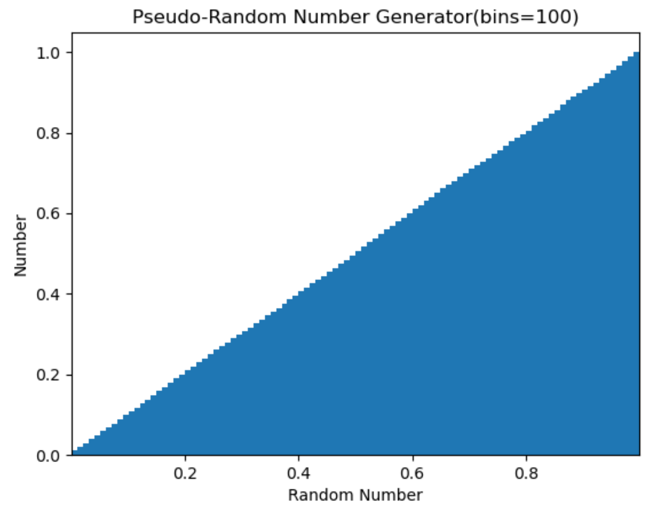
옆의 식을 추가하여 T 리스트 내부에 있는 값의 평균을 구해보면 0.5018268이 나옴을 알 수 있다.

2)

```
plt.hist(T,bins=100,cumulative=True,density=True)
```

'cumulative=True'는 누적 히스토그램을 나타낸다
'density=True'는 함수를 밀도함수로 바꾸어 주어 아래면적이 1이 되게한다

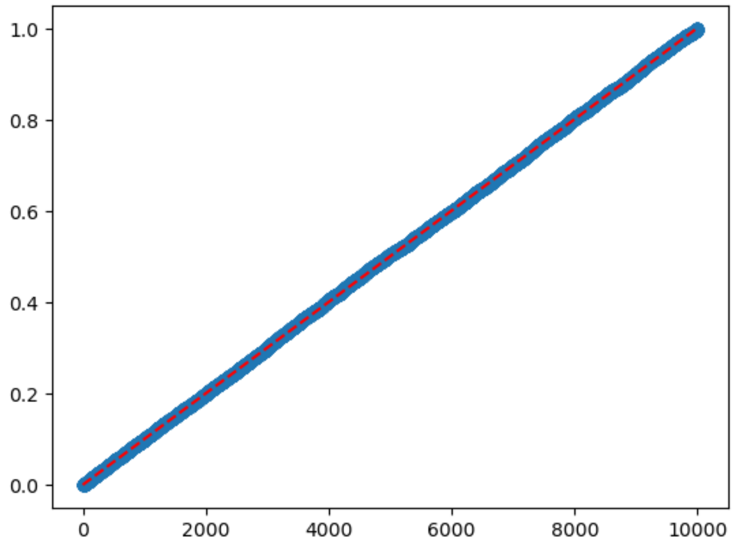
히스토그램을 확률누적분포함수로 바꾸면 y=x와 유사함을 알 수 있다.
(균일분포의 누적함수 성질)



3)

```python
import matplotlib.pylab as pylab
import numpy as np
A=16807
C=2147483647
Q=127773
R=2836
X=1
N=[]
T=[]
T1=[]
for i in range(10000): # 10000번 반복하는 for문
    X=A*(X%Q)-R*(X//Q)  # %은 나머지를 반환하는 연산자, //은 몫을 반환하는 연산자
    if X<0:    # X값이 음수인경우 C를 더하여 양수를 만들어준다
        X=X+C
    N.append(X)  # N리스트에 X값 추가
M=max(N) # 0과 1사이의 난수값을 발생시키위하여 N리스트에 있는 값을 최대값으로 나누어준다
for i in N:
    T.append(i/(M+1))  # N리스트를 0과 1사이의 변수로 만들어 새로운 리스트에 넣어준다

for i in range(10000):  #산점도를 그리기 위하여 T1리스트에 1~10000까지 넣어준다
    T1.append(i)

T.sort()  #T리스트를 크기 순서대로 정렬

#산점도를 그리기면서 추세선을 나타내는 식
z=np.polyfit(T1,T,1)
p=np.poly1d(z)
pylab.plot(T1,T,'o')
pylab.plot(T1,p(T1),"r--")
pylab.show()
print("y=%.6fx+(%.6f)"%(z[0],z[1]))
```

X축을 횟수(1~10000)
Y축을 순서대로 정렬한 난수로 설정하였다
(x값 1부터 발생한 난수들 중 작은 값이 대입된다.)



y=0.000100x+(0.000957) 추세선 함수

이를 통하여 Schrage's Algorithm을 통하여 발생한 난수들은 어느정도 균일하게 분포하였음을 알 수 있다.

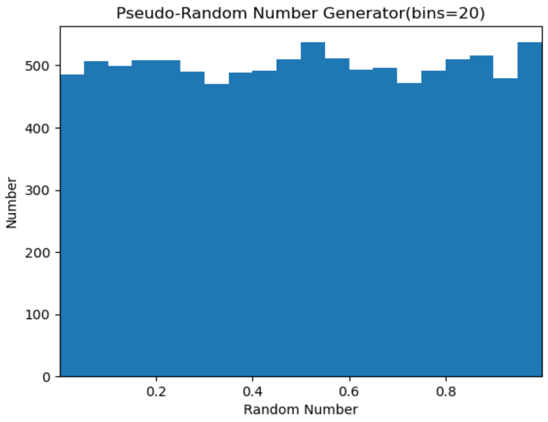
* 이를 통하여 결과가 균등하게 나타난 것을 알 수 있다

토의사항

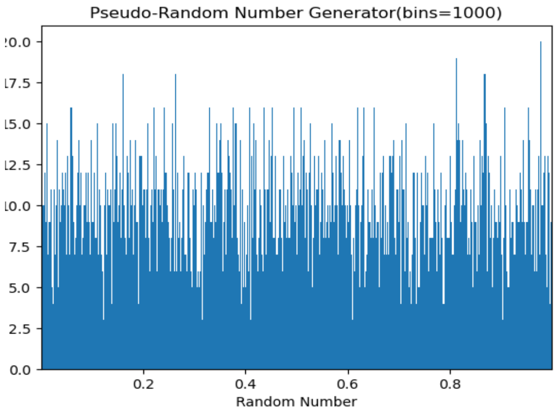
2.히스토그램의 간격(bin size)은 어떤 값이 최적일까?

위의 결과값에도 보았듯이 bins값이 증가 할수록 Numbers(횟수)의 격차가 심해짐을 볼 수 있다.

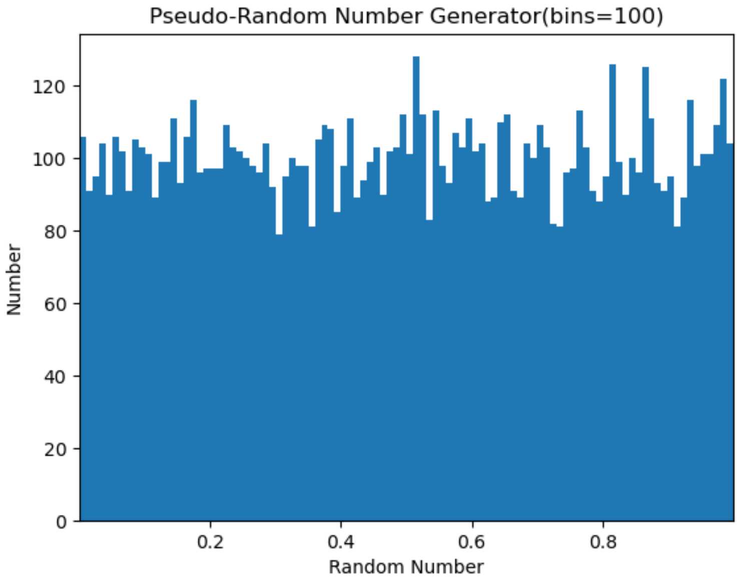
Bins=20



Bins=1000



따라서 유의미한 난수의 빈도를 나타내면서 극심한 격차를 나타내지 않기 위해서 Bins값이 100이 적당하고 판단



토의사항

3.이 난수 발생기의 성능에 대해 만족하는가? 이것의 약점은 무엇일까?

Schrage's Algorithm를 통하여 난수를 발생 시킨다면 식 내부에 있는 변수나 초기값(시드값)을 변경 시키지 않는다면 발생되는 난수가 항상 같다는 것을 알 수 있다. (주기가 일정하다)

-> 다음 수에 대해 예측이 가능해진다.

-> 'True' Random Number Generator가 아닌

"Pseudo-Random Number Generator"(유사난수발생기)의 성질

Schrage's Algorithm은 기본적으로 Linear Congruence Method을 기본으로 하기 때문에

$$X_{n+1} = (A\ X_n + B) \bmod C$$

A,B,C, X(초기값)에 이후 발생하는 난수들이 크게 의존하게 된다.

초기값(시드값)을 계속 변경 할 수 있다면 이후 발생되는 난수들이 예측 불가능 하지 않을까?
->초기값(시드값)을 계속 변화는 시간으로 설정한다. (time.time()함수 적용)

```python
import matplotlib.pyplot as plt
import time  #시간과 관련된 모듈을 사용하기 위하여 time 모듈
A=16807
C=2147483647
Q=127773
R=2836
X=time.time() # X초기값을 현재 시간을 실수로 설정한다
N=[]
T=[]

for i in range(10000): # 10000번 반복하는 for문
    X=A*(X%Q)-R*(X//Q)  # %은 나머지를 반환하는 연산자, //은 몫을 반환하는 연산자
    if X<0:    # X값이 음수인경우 C를 더하여 양수를 만들어준다
        X=X+C
    N.append(X)  # N리스트에 X값 추가
M=max(N) # 0과 1사이의 난수값을 발생시키위하여 N리스트에 있는 값을 최대값으로 나누어준다
for i in N:
    T.append(i/(M+1))  # N리스트를 0과 1사이의 변수로 만들어 새로운 리스트에 넣어준다

print(X)   # 초기값이 계속 바뀌면서 리스트 난수 값이 바뀜을 볼 수 있다.
print(T)
```

실행 마다 초기값이 변화하여 리스트내의 변수가 계속 바뀜을 알 수 있다.
(True Random Number Generator에 근접하였다고 생각 할 수 있다.)

하지만 이 또한 특정한 환경(변수, 시드) 내에서는 일정한 주기를 가진다는 Linear Congruence Method의 한계를 벗어나지 못하였다.

토의사항

4.x_n값이 0이 되는 경우 이후 발생하는 값들이 계속 0이 됨을 볼 수 있다.

현재 주어진 환경(10000회 반복, A=16807, Q=127773, R=2836, C=2147483647) 내에서는 발생한 난수들 중에서 0이 없음을 볼 수 있다.
하지만 X(초기값)에 0이 넣을 경우 이후 발생하는 값들이 모두 0임을 볼 수 있다.
->이는 Schrage’s Algorithm에서 난수 발생도중 한번이라도 ‘0 ' 이 발생하면 이후 값들이 다 0임을 알 수 있다.

```python
import matplotlib.pyplot as plt
A=16807
C=2147483647
Q=127773
R=2836
X=0
N=[]
T=[]
for i in range(10000): # 10000번 반복하는 for문
    X=A*(X%Q)-R*(X//Q)  # %은 나머지를 반환하는 연산자, //은 몫을 반환하는 연산자
    if X<0:    # X값이 음수인경우 C를 더하여 양수를 만들어준다
        X=X+C
    N.append(X)  # N리스트에 X값 추가
M=max(N) # 0과 1사이의 난수값을 발생시키위하여 N리스트에 있는 값을 최대값으로 나누어준다
for i in N:
    T.append(i/(M+1))  # N리스트를 0과 1사이의 변수로 만들어 새로운 리스트에 넣어준다
print(T)
```

초기값을 0으로 설정 시 이후 값들이 다 0이다.

발생한 난수(X)값이 0이 되었을 때 다른 값으로 바꾸어 주면 어떨까?
(python내에서 X값이 0 일때 다른 값으로 바꾸어주는 IF문 작성)

```python
for i in range(10000): # 10000번 반복하는 for문
    X=A*(X%Q)-R*(X//Q)  # %은 나머지를 반환하는 연산자, //은 몫을 반환하는 연산자
    if X<0:    # X값이 음수인경우 C를 더하여 양수를 만들어준다
        X=X+C
    if X==0:
        X=1
    N.append(X)  # N리스트에 X값 추가
```

X값이 0 일때 X를 다른 수(1)로 변경해준다

이때 X값이 1로 바뀌면서 다시 난수가 발생됨을 볼 수 있다.
하지만 이는 다시 0이 발생시 또 다시 1로 반환되어 계속 반복되는 규칙성을 보여준다.
->0이 계속 발생되는 문제는 해결하였지만 ‘난수의 규칙성(주기성) ' 은 해결하지 못하였다.

Reference

· *How to add trendline in python matplotlib dot (scatter) graphs?, stackoverflow,2014/10/19*

https://stackoverflow.com/questions/26447191/how-to-add-trendline-in-python-matplotlib-dot-scatter-graphs

·*"Uniform Distribution (Continuous)" , MathWorks, 2019*

https://www.mathworks.com/help/stats/uniform-distribution-continuous.html

·*'Matplotlib 히스토그램 그리기' ,Codetorial*

https://codetorial.net/matplotlib/index.html

·*Pseudo-Random Number Generators(PRNGs), Dr Mads Haahr, Random.org,*

https://www.random.org/randomness/