LAB 4

OBJECTIVE

To generate pseudo-random numbers using middle-square method and implement Kolmogorov-Smirnov Test to test the uniformity of random numbers

THEORY

In mathematics, the middle-square method is a method of generating pseudorandom numbers. To generate a sequence of n-digit pseudorandom numbers, an n-digit starting value is created and squared, producing a 2n-digit number. If the result has fewer than 2n digits, leading zeroes are added to compensate. The middle n digits of the result would be the next number in the sequence, and returned as the result. This process is then repeated to generate more random numbers.

The testing for uniformity of random numbers can be achieved through different frequency tests. The Kolmogorov-Smirnov test is one of them.

The procedure for the test follows the following steps :

Step 1 : Rank the data from smallest to largest. Let Ri denote i^{th} smallest observation, so that R1 <= R2 <= R3 <= ... <= RN

Step 2 : Compute D+ = max { i/N – Ri } and D- = max { Ri – (i-1)/N}

Step 3 : Find D = max {D+, D-}

Step 4 : Determine the critical value, D_{α}, from Table for the specified value of α and the sample size N.

Step 5 : If the sample statistic D is greater than the tabulated value of D_{α}, the null hypothesis that the data are a sample from uniform distribution is rejected and alternate hypothesis is accepted.

CODE

```
# Python program to generate pseudo-random numbers using the middle
square method

seed = int(input("Enter a 4 digit number : "))
number = seed
print(f"The seed is {seed}.")

print("The random numbers are : ")
already_generated = set()

while(True):

    already_generated.add(number)

    Xi = str(int(number)**2)
    # print(f"Square : {Xi}")

    # Builtin function to fill zeros / prepend 0s.
    # Xi.zfill(8)

    zeros_to_be_added = 8 - len(Xi)
    for i in range(zeros_to_be_added):
        Xi = '0' + Xi
    # print(f"After padding : {Xi}")

    number = str(Xi)[2:6]

    if(number in already_generated):
        break

    if (int(number)==0):
        break

    print(int(number))
```

OUTPUT

```
Enter a 4 digit number : 9967
The seed is 9967.
The random numbers are :
3410
6281
4509
3310
9561
4127
321
1030
609
3708
7492
1300
6900
6100
2100
4100
8100
```

CODE

```
# Python program to implement Kolmogorov-Smirnov Test

data=[0.44,0.81,0.14,0.05,0.93]
N = len(data)
α = 0.05
D_critical = 0.565
print("Data = {}".format(data))
print("N = {}".format(N))
print("Level of significance (α) = {}".format(α))
print("Critical value of D for α = {} and N = {} is {}."
      .format(α,N,D_critical))

Ri = data.copy()
Ri.sort()
print("\nAfter sorting :")
print("Ri = {}".format(Ri))

i = list(range(1, len(data)+1))
print("i = {}".format(i))

i_N = []
for index in i:
    i_N.append(index/N)
print("i/N = {}".format(i_N))

i_N_Ri = []
for index in i:
    i_N_Ri.append(round(i_N[index-1] - Ri[index-1],2))
print("i/N - Ri = {}".format(i_N_Ri))

Ri_i_1_N = []
for index in i:
    Ri_i_1_N.append(round(Ri[index-1] - (index-1)/N,2))
print("Ri - (i-1)/N = {}".format(Ri_i_1_N))

D_p = max(i_N_Ri)
print("D+ = {}".format(D_p))
D_n = max(Ri_i_1_N)
print("D- = {}".format(D_n))
```

```
D_calc = max(D_p,D_n)
print("D = {}".format(D_calc))

if (D_calc < D_critical):
    print("\nCalculated value of D < Tabulated value of D.")
    print("Therefore, Null Hypothesis is accepted.")
else:
    print("\nCalculated value of D > Tabulated value of D.")
    print("THerefore, Null Hypothesis is rejected.")
```

OUTPUT

```
Data = [0.44, 0.81, 0.14, 0.05, 0.93]
N = 5
Level of significance (α) = 0.05
Critical value of D for α = 0.05 and N = 5 is 0.565 .

After sorting :
Ri = [0.05, 0.14, 0.44, 0.81, 0.93]
i = [1, 2, 3, 4, 5]
i/N = [0.2, 0.4, 0.6, 0.8, 1.0]
i/N - Ri = [0.15, 0.26, 0.16, -0.01, 0.07]
Ri - (i-1)/N = [0.05, -0.06, 0.04, 0.21, 0.13]
D+ = 0.26
D- = 0.21
D = 0.26

Calculated value of D < Tabulated value of D.
Therefore, Null Hypothesis is accepted.
```

CONCLUSION

In this lab, I implemented the middle-square method to generate pseudo-random numbers and also tested the uniformity of given random numbers using the Kolmogorov-Smirnov test using python programming language.