Random Number Generation Using Combined Linear Congruential Generator

Sarbajit Mazumdar, Roll-1911147[1*]

[1]*School of Physical Sciences*

[1] *National Institute of Science Education and Research, Bhubaneswar, HBNI, P.O. Jatni, Khurda-752050, Odisha, India.*

November 21, 2021

1 Introduction

Linear Congruential Generator is a computational method by which we can produce Random Numbers. Random number generation is a process by which, a sequence of number that cannot be reasonably predicted better than by random chance is generated. True random number generations is almost an impossible task to model practically that's why in contrast to so-called "random number generations" done by pseudo-random number generators that generate numbers that only look random but are in fact a sequence which has a large period of repetitions. In this Project I will try to build up a combined Linear Congruential Generator which is in fact combination of two or more Linear Congruential Generators.

2 Linear Congruential Generator

A linear congruential generator (LCG) is a method by which we can generate a sequence $\{x_n\}$ of pseudo-random numbers calculated with a discontinuous piece wise linear modulo equation. Linear Congruential generator produces sequence $\{x_n\}$ defined by relations

$$\begin{aligned} &0 \leq x_0 < M \\ &x_{n+1} = (\mathcal{A}x_n + C) \bmod M, \quad n \geq 0 \end{aligned} \tag{1}$$

Modulus M, seed value is x_0, multiplier and increment is $\mathcal{A}$ and C respectively. where mod M means the arithmetic mod M. Without loss of generality we take the value of $\mathcal{A}, C$ in $\{0, \cdots, M-1\}$. In this method the random integers x_n are being generated in $[0, M-1]$ and by defining $\mathcal{R}_n = \frac{x_n}{M}$ we can generate random numbers between 0 and 1, where $\mathcal{R}_n \in \left[0, \frac{M-1}{M}\right]$.

If $C = 0$, the generator is often called a multiplicative congruential generator. If $C \neq 0$, the method is called a Mixed congruential generator.

*sarbajit.mazumdar@niser.ac.in

2.1 Choice of Parameters

The choice of parameters($\mathcal{A}$, M and C) are very much important to build up a good Random Number Generator(RNG), specially it is known that with appropriate choice of parameters, the period can be made longer. The following are several properties that are known to us.

M is prime and $C = 0$:

In this choice the maximum Period ($\mathscr{P}$) that can be achieved is $P = M - 1 \iff$ multiplier $\mathcal{A}$ has property that smallest integer k such that

$$\mathcal{A}^k - 1 \equiv 0 \ (\ mod\ M\ ) \quad \text{and } k = M - 1 \tag{2}$$

$M = 2^r$, $r \in \mathbb{N}$ and $C = 0$:

Most often we choose $M = 2^{32}$ or $M = 2^{64}$, which produces a efficient LCG. Longest possible period that can be achieved by choosing this particular choice of parameter is

$$\mathscr{P} = \frac{M}{4} = 2^{r-2} \tag{3}$$

In this choice the seed x_0 has to be odd and the multiplier has to be in the form of either $\mathcal{A} = 3 + 8k$ or $\mathcal{A} = 5 + 8k$, for $k = 0, 1, \cdots$

$C \neq 0$:

For this case there goes a very famous theorem called **Hull-Dobell Theorem** which suggests The length of period $\mathscr{P}$ is maximum, i.e. equal to M, if and only if it holds these three following conditions :

1. C and M should be co prime.
2. $\mathcal{A} - 1$ is divisible by each prime factor of M.
3. If 4 divides M then also 4 divides $\mathcal{A} - 1$.

2.2 Examples of LCG

1. I will give an example which follows the **Hull-Dobell Theorem**,

Let's take

$$\begin{gathered} M = 18, \mathcal{A} = 7, C = 5 \\ x_0 = 4 \\ x_{n+1} = (7x_n + 5) \mod 18, n \geq 0 \\ \{x_n\} = \underbrace{15, 2, 1, 12, 17, 16, 9, 14, 13, 6, 11, 10, 3, 8, 7, 0, 5, 4,}_{\text{Periodicity, length} = 18 = M} 15, 2, 1, 12, 17, 16, \cdots \end{gathered} \tag{4}$$

Hence we can see, we achieved the $\mathscr{P} = M = 18$.

2. Another very well known LCG goes as $\mathcal{A} = 1664525$, $C = 1013904223$ $M = 4294967296 = 2^{32}$ and $x_0 = 1$.

We will generate $\mathcal{R}_n = \frac{x_n}{M}$ as random numbers which guarantees $\mathcal{R}_n \in \left[0, \frac{M-1}{M}\right]$.

I've plotted the generator by setting $N = 2500$

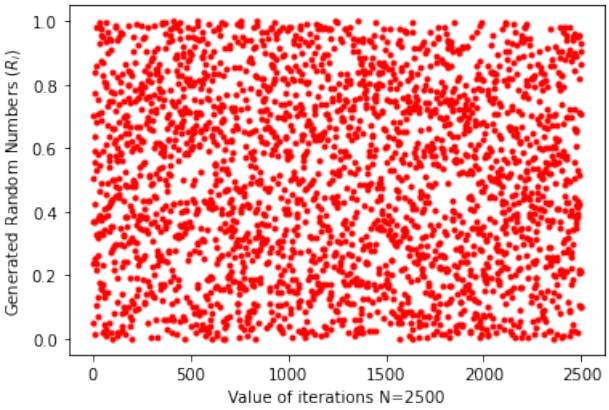

Figure 1: : Scatter Plot for Random Numbers; $\mathcal{R}_n$ for $N = 2500$

3 Combined Linear Congruential Generator(CLCG)

A combined linear congruential generator is a pseudo-random number generator algorithm based on combining two or more linear congruential generators. It is defined as

$$x_i \equiv \left(\sum_{j=1}^{k} (-1)^{j-1} Y_{i,j} \right) \pmod{M_1 - 1} \tag{5}$$

where: M_1 is the "modulus" of the first LCG $Y_{i,j}$ is the i^{th} input from the j^{th} LCG x_i is the i^{th} generated random integer with:

$$\mathcal{R}_i = \begin{cases} \frac{x_i}{M_1} & \text{for } x_i > 0 \\ \frac{(M_1 - 1)}{M_1} & \text{for } x_i = 0 \end{cases} \tag{6}$$

where $\mathcal{R}_i$ is a uniformly distributed random number between 0 and 1 .

The maximum period that can be attainable for combining k-many LCG is :

$$\mathscr{P} = \frac{1}{2^{k-1}} \prod_{i=1}^{k} (M_i - 1) \tag{7}$$

Hence we can use such Combined LCG to produce Random numbers which have longer periods.

3.1 Combining 2 LCGs ($k = 2$):

There is a very famous RNG that had been built using **2** LCGs using the conditions:

$$\begin{aligned} \mathcal{A}_1 &= 40014 & \mathcal{A}_2 &= 40692 \\ M_1 &= 2147483563 & M_2 &= 2147483399 \\ C_1 &= 0 & C_2 &= 0 \end{aligned}$$

As following our choice of parameters:

1. The seed value for the first LCG, $Y_{0,1}$ should be selected in the range of **[1, 2147483562]**.

2. The seed value for the second LCG, $Y_{0,2}$ should be selected in the range of $[1, 2147483398]$, Setting: $i = 0$

$$\begin{aligned} k = 2, \quad 1 \le Y_{0,1} \le 2147483562, \quad 1 \le Y_{0,2} \le 2147483398 \\ Y_{n+1,1} = (40014 Y_{n,1} + 0) \bmod 2147483563, \quad n \ge 0 \\ Y_{n+1,2} = (40692 Y_{n,2} + 0) \bmod 2147483399, \quad n \ge 0 \\ x_n = (Y_{n,1} - Y_{n,2}) \bmod 2147483562, \quad n \ge 0 \end{aligned} \tag{8}$$

As following our definition of CLCG, The generated Random numbers between 0 and 1 will be in the form of:

$$\mathcal{R}_i = \begin{cases} \frac{x_i}{2147483563} & \text{for } x_i > 0 \\ \frac{2147483562}{2147483563} & \text{for } x_i = 0 \end{cases} \tag{9}$$

3.2 Combining 3 LCGs ($k = 3$):

3.2.1 Wichman-Hill Generator

The Wichman-Hill generator goes as follows. The state space is $\{0, 1, 2 \cdots, M_1 - 1\} \times \{0, 1, 2 \cdots, M_2 - 1\} \times \{0, 1, 2 \cdots, M_3 - 1\}$ and three generators are (X_n, Y_n, Z_n). Then the generator is

$$\begin{aligned} X_n &= 171 X_{n-1} \bmod M_1 \\ Y_n &= 172 Y_{n-1} \bmod M_2 \\ Z_n &= 170 Z_{n-1} \bmod M_3 \end{aligned}$$

with $M_1 = 30269, M_2 = 30307, M_3 = 30323$. The output function is

$$U_n = \frac{X_n}{M_1} + \frac{Y_n}{M_2} + \frac{Z_n}{M_3} \bmod 1$$

These guarantees generation between $(0, 3)$ and the plot for $N = 3000$ is:

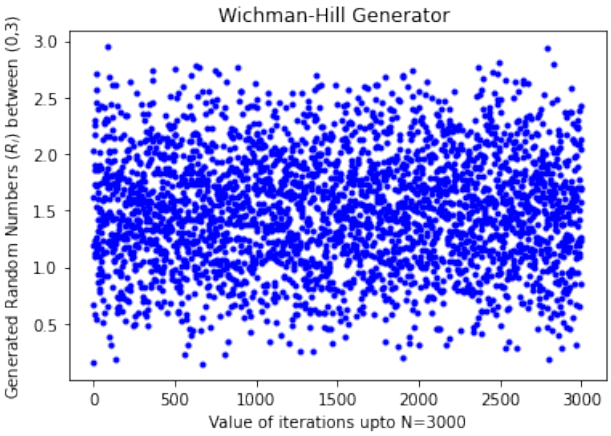

Figure 2: : Wichman-Hill Generator ; $\mathcal{R}_n$ for $N = 3000$

3.2.2 My Attempt:

Now I will try to build up a CLCG using 3 LCG. I will try to first select the choice of parameters. As in the previous section 2.1, I discussed about the different choices of parameters, using that I am going to choose the case where $M = 2^r, r \in \mathbb{N}$ and

$C = 0$. We already know that for this particular case the seed x_0 has to be odd and the multiplier has to be in the form of either $\mathcal{A} = 3 + 8k$ or $\mathcal{A} = 5 + 8k$, for $k = 0, 1, \cdots$

$$\begin{array}{lll} \underline{\mathrm{LCG}-I} & \underline{\mathrm{LCG}-II} & \underline{\mathrm{LCG}-III} \\ \mathcal{A}_1 = 811 & \mathcal{A}_2 = 819 & \mathcal{A}_3 = 827 \\ M_1 = 2^{64} & M_2 = 2^{32} & M_3 = 2^{16} \\ C_1 = 0 & C_2 = 0 & C_3 = 0 \\ \text{seed} = 7 & \text{seed} = 9 & \text{seed} = 11 \end{array} \tag{10}$$

$\Longrightarrow$

$$\begin{aligned} Y_{n+1,1} &= 811 Y_{n,1} \bmod 2^{64} \\ Y_{n+2,2} &= 819 Y_{n,2} \operatorname{mad} 2^{32} \\ Y_{n+3,3} &= 827 Y_{n,3} \bmod 2^{16} \\ x_n &= (Y_{n,1} - Y_{n,2} + Y_{n,3}) \bmod (2^{64} - 1) \end{aligned} \tag{11}$$

Hence following the criteria mentioned in the definition equation 6 we got the generated random numbers $\mathcal{R}_i$ between 0 and 1:

$$\mathcal{R}_i = \begin{cases} \frac{x_i}{2^{64}} & \text{for } x_i > 0 \\ \\ \frac{2^{64}-1}{2^{64}} & \text{for } x_i = 0 \end{cases} \tag{12}$$

I've plotted the generator by setting $N = 3000$.

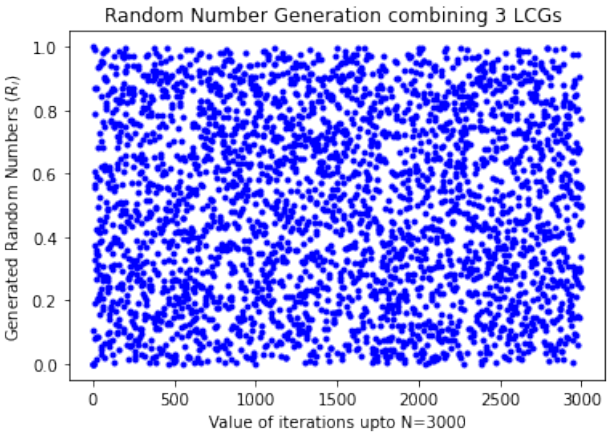

Figure 3: : Scatter Plot for Random Numbers produced by CLCG ; $\mathcal{R}_n$ for $N = 3000$

I've attached the entire Python code for generation of the CLCG.
A very interesting fact about this, is the period ($\mathcal{P}$) of this generator is very high :

$$\mathcal{P} = \frac{1}{2^{3-1}} \prod_{i=1}^{3} (M_i - 1) = O(2^{110}) \tag{13}$$

This represents a tremendous improvement over the period of the individual LCGs. Although it may not reach it's entire strength. Also the histogram for $N = 3000$ will be:

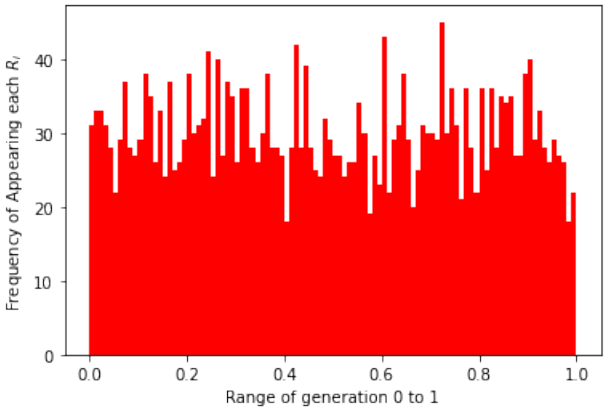

Figure 4: : Histogram Plot for Random Numbers produced by CLCG ; $\mathcal{R}_n$ for $N = 3000$

4 Test For Randomness

The **Kolmogorov-Smirnov Test** of Normality is one of the best statistical testing which ensures that whether our data is normally distributed or not. The test statistic (D) provides a measurement of the divergence of your sample distribution from the normal distribution. The higher the value of D, the less probable it is that your data is normally distributed. The ***p*-value** quantifies this probability, with a low probability indicating that your sample diverges from a normal distribution to an extent unlikely to arise merely by chance. Put simply, high D, low p, is evidence that your data is not normally distributed.

The Null hypothesis used here assumes that the numbers follow the normal distribution.

The functioning of the function remains exactly same. Again it returns statistics and p-value. If the p-value is $< D$, we reject the Null hypothesis and proves that our generated data is random.

I checked in an online Kolmogorov-Smirnov Test calculator whether my CLCG $x_n = (Y_{n,1} - Y_{n,2} + Y_{n,3})$ is generating random numbers or not, by taking $N = 300$ which is the calculator's maximum limit of input.

4.1 Distribution Summary

Count: **300**
Mean: **0.47442**
Median: **0.447017**
Standard Deviation: **0.295662**
Skewness: **0.150193**
Kurtosis: **−1.229855**

4.2 Result

From the online calculator we got:

The value of the K-S test statistic (D) is **0.08319**. The p-value is **0.02964**. This provides good evidence that your data is **not** normally distributed. The datasheet for $N = 300$ will be attached.

5 Conclusion

In this project I've generated random numbers using Combined Linear Congruential Generator obeying all the mathematical intricacies. Also I've performed a test which gives the output as the generated random numbers are in fact non uniform and random. Also I've shown some pure mathematical definition and building steps of congruential generator by scattered plots and python algorithm using large number of data sets.

6 Acknowledgement

I would like to thank our P **346** course instructor **Dr. Subhasis Basak** for assigning this project which has enormous computational values, I learned plenty about random number generation and mathematical theory behind the congruential generators. I tried my best to reflect my acquired knowledge in this project.

References

1. Linear Congruential Generators https://en.wikipedia.org/wiki/Linear_congruential_generator
2. Lehmer random number generator https://en.wikipedia.org/wiki/Lehmer_random_number_generator
3. Kolmogorov-Smirnov Test https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test
4. Random Number Generators by Professor Karl Sigman, Columbia University http://www.columbia.edu/~ks20/4106-18-Fall/Simulation-LCG.pdf
5. Kolmogorov-Smirnov Calculator Online https://www.socscistatistics.com/tests/kolmogorov/default.aspx

The Kolmogorov-Smirnov Test of Normality

Success!

Interpreting the Result

The test statistic (D), which you'll see below, provides a measurement of the divergence of your sample distribution from the normal distribution. The higher the value of D, the less probable it is that your data is normally distributed. The *p*-value quantifies this probability, with a low probability indicating that your sample diverges from a normal distribution to an extent unlikely to arise merely by chance. Put simply, high D, low *p*, is evidence that your data *is not* normally distributed.

It's also worth taking a look at the figures provided for skewness and kurtosis. The nearer both these are to zero, the more likely it is that your distribution is normal.

Your Data

```
4.013174
34145509
47e-16,
0.999999
99999992
52,
1.672225
74491959
07e-10,
1.639722
83245795
38e-07,
0.000133
13184134
07085,
0.107970
08471370
063,
0.563738
83676364
5,
0.192196
67571897
```

Distribution Summary

Count : 300

Mean: 0.47442

Median: 0.447017

Standard Deviation: 0.295662

Skewness: 0.150193

Kurtosis: -1.229855

Result: The value of the K-S test statistic (D) is .08319.

The *p*-value is .02964. This provides good evidence that your data is *not* normally distributed.

Calculate Reset

Data set for N=300 which used for Kolmogorov-Smirnov Test:

4.0131743414550947e-16, 0.9999999999999252, 1.6722257449195907e-10, 1.6397228324579538e-07, 0.0001331318413407085, 0.10797008471370063, 0.563738836763645, 0.19219667571897647, 0.8715041897645636, 0.789898082119466, 0.6073446477627992, 0.5565095062387594, 0.32920962188153824, 0.9890035020120458, 0.0818401967847648, 0.37239967590135176, 0.016137339747194526, 0.08738268137486856, 0.8673546367453047, 0.4246105858077931 5, 0.3591851099448423, 0.2991243255025576, 0.5898280106978362, 0.35051685992616216, 0.26917353107067815, 0.29973378154826663, 0.08409692094695294, 0.2026029403127135, 0.310984756612152, 0.20863761775476322, 0.2051081448215485, 0.34270561903993524, 0.9342570690534093, 0.6824830246779151, 0.49373303038748145, 0.41748767652287316, 0.5825057817819264, 0.41218914134062695, 0.2853936332410263, 0.45423658521433263, 0.3858706392541715, 0.9410885856842356, 0.2228430088165934, 0.7256802775218621, 0.5267051398560034, 0.15786843843426795, 0.03130362713608629, 0.38724163572342957, 0.05296658530008323, 0.9559008231545228, 0.2355677295605378, 0.04542873584401514, 0.8427048907649408, 0.43366659158224874, 0.7036058004073966, 0.6243041516081138, 0.31066707844566016, 0.951000775975779, 0.26162942909656733, 0.18146716031734175, 0.16986717644571112, 0.7622801793176217, 0.20922546278725337, 0.6818503223042522, 0.9806114535339208, 0.2758888641940863, 0.7458688623222887, 0.899647452657955, 0.6140841858314207, 0.022274835164997422, 0.06489137160706522, 0.626902445030045, 0.4178830394850579, 0.9031451162145477, 0.4506894201461533, 0.5091198339777948, 0.8961854509760899, 0.8064009101432562, 0.9911382829564327, 0.8131476372097409, 0.4627339343373721, 0.2772208714154228, 0.8261267245921843, 0.9887737009756709, 0.8954716169201123, 0.22748136970422278, 0.4873909218845208, 0.27403772073613136, 0.24459169752400833, 0.36386682015704047, 0.09599129560522468, 0.8489409048343045, 0.4910738286893629, 0.26087513715870975, 0.5697364222320838, 0.056238496846869566, 0.6094211228725377, 0.24053071326037517, 0.0704084857482037, 0.10128210986577942, 0.1397912208041565, 0.37068020635181287, 0.6216474105701401, 0.15605002241260205, 0.5565682031308601, 0.3768129129644566, 0.5952724314619173, 0.7659420500247563, 0.17900273492112256, 0.17121814137950772, 0.8579126698453932, 0.767175302019862, 0.1791701080257934, 0.30695774286835525, 0.9427296306188547, 0.5537305271077909, 0.07545756949249355, 0.19608890936235812, 0.028105505549787368, 0.7935651161540749, 0.5813093185360967, 0.44185736735345993, 0.3463251080968805, 0.8696628134318437, 0.296541765388012, 0.49537174120235516, 0.7464821752820857, 0.3970442091007392, 0.0028537513080899982, 0.31439249576650363, 0.9723140765464693, 0.5467162322752164, 0.38686443768071654, 0.7470589846507893, 0.8648365623934962, 0.3824521407766818, 0.16868635732821227, 0.8046359068432825, 0.5597204810224726, 0.9333102595464923, 0.9146205256332269, 0.7572463588637457, 0.12679707100384532, 0.8324246765692277, 0.09641278848069329, 0.1907716259157297, 0.7157886771378695, 0.5046172982958822, 0.24462907635030826, 0.39418096620650683, 0.6807637523262582, 0.09940328806904083, 0.6160666721773749, 0.6300711780414426, 0.9877254287282865, 0.045322719158341296, 0.7567253017375665, 0.7042197854782988, 0.12224609898239899, 0.14158634757921398, 0.826528015851603, 0.31422094694886454, 0.8331880133378476, 0.7154788607977953, 0.2533561328317833, 0.4718237295699129, 0.6490447738807114, 0.3753117724204558, 0.37784744635884643, 0.4342791415789455, 0.20038394824947883, 0.5113821142477423, 0.7308947826449527, 0.7556688848293983, 0.8474656764128189, 0.2946636747761764, 0.9722403089533527, 0.48689072001916295, 0.8683740185382385, 0.251329221722158, 0.8279988607038861, 0.5070760490628542, 0.23867594582849982, 0.5661920961937483, 0.18179001589446214, 0.43170295934347047, 0.11110012738225883, 0.10220343335409476, 0.8869846318459003, 0.34453661446532213, 0.41919443005268936, 0.9666828435084869, 0.9797861501660684, 0.6065679433016643, 0.9266020575350998, 0.4742687049997172, 0.6319197992665734, 0.4869572270916051, 0.9223112971722537, 0.9944621740769785, 0.5088233309012938, 0.6557214222740465, 0.7900735101298003, 0.7496168164811919, 0.939238203594692, 0.7221832086685286, 0.6905822532287049, 0.062207524100911564, 0.4503022023822931, 0.19508630149702133, 0.21499058555540018, 0.3573649672749223, 0.8229885729315622, 0.44373272934189445, 0.867243514489207, 0.33449026065660437, 0.2716015103171679, 0.2688250306866559, 0.017099910392069338, 0.8680274093517338, 0.9702291456425138, 0.8558371361357743, 0.08391741026212197, 0.057019758544169345, 0.2430242731557467, 0.0926856662583023, 0.1680754772742242, 0.30921208483964646, 0.7710008754995031, 0.28171019747663273, 0.46697028450296907, 0.7129008047603489, 0.16255270536753405, 0.8302442186079866, 0.3280613941330833, 0.057790667059447146, 0.8682310091875797, 0.13534853389298296, 0.7676611093997512, 0.5731598811248908, 0.8326637479117627, 0.2902995714248889, 0.43295248714047724, 0.12446715161897158, 0.942860050364504, 0.6595009488984509, 0.8552697069641366, 0.6237323608249254, 0.8469446988709951, 0.8721509102587118, 0.31438825647159724, 0.9688761026742868, 0.758

519303427479, 0.15915521363568033, 0.07487831640215956, 0.7263147819845656, 0.041288250116129036, 0.4847709396290203 5, 0.14923214219293218, 0.027267362270672717, 0.11383096428518734, 0.316912123124727, 0.01573199433010316, 0.7586474927807163, 0.2631168143858432, 0.38773660086969897, 0.45438336688049175, 0.5049106320697802, 0.4825226867490718, 0.32589904594839664, 0.304126346455328, 0.6464670317549291, 0.28476290195376763, 0.9427135024876321, 0.5406506196020403, 0.4676525954690639, 0.26625509855542157, 0.9328850082171759, 0.5697416881027332, 0.06050923276293941, 0.0729877896465031, 0.1930974787055074, 0.6020552324688597, 0.2667936030233732, 0.36961218060379797, 0.7554785559051349, 0.6931089990681456, 0.1113984266336256, 0.34412410454018233, 0.08464882589156851, 0.6501978709144463, 0.3104733734034739, 0.7939059378841086, 0.8577157522011835, 0.6074750443786671, 0.6622610913868295, 0.09374528924661053, 0.02742972932220872, 0.2455105540867939, 0.1090593715334586