Investigating whether the inputs of a Multiplicative Congruential Generator can be determined using the outputs

Yohwllo

March 10th 2024

Contents

1 Introduction

Ever since I was a young child the concept of random has amazed me. Randomness could at times not seem random, for example, number generators more often than not are far from random, instead they are pseudo-random number generators, which are not truly random. These pseudo-random number generators are not truly random, but rather just a mathematical equation which is being done iteratively, and if the developer was forward-thinking, it used a different seed (a seed is the first value) every time too.

Now among random number generators, there is a particularly common one called the Multiplicative Congruential Generator (MCG for short), it has been extensively used for various tasks that could require a lot of random numbers, that only need to be "random enough". For example, creating test data, or for things such as dice rolling in games. There is also another type of random number generator similar to it called the Linear Congruential Generator which is very similar in terms of how it functions.

2 background

2.1 Random Number Generators

2.1.1 Multiplicitve Random Number Generator

Usually an MCG works based off of the equation $x_{i+1} = (a \cdot x_i) \bmod m$. Where the next value is equal to the current value multiplied by constant, modulus a constant. [1] [5]

2.1.2 Linear Congruential Generators

In contrast an LCG (Linear Congruential Generator) works in a very similar way, just adding a constant before the modulus operation. Its equation looks like $x_{i+1} = (a \cdot x_i + b) \bmod m$ [1]

2.1.3 How LCG/MCG's are used

Linear/Multiplicative congruential generators are used in a variety of settings, from Monte Carlo simulations [5], to the shuffle function in music players [2], these are just the implementations that are public and known about. Considering that these types of random number generators are also used in glibc's [6] random function, no one knows just how many things use these types of generators for their random numbers. So the implication of these random number generators not being random would have an unknown impact on the world.

2.1.4 Flaws of LCG/MCG's

Now these types of linear congruential generators have some flaws that were found by Gorge Marsaglia [5]. That is, if you know how to arange these numbers on a 3d object, all of said numbers will fall upon $\sqrt[n]{n! \cdot m}$ hyper planes [5]. A hyperplane is a plane of $n-1$ dimensions, with n being the number of diemensions. As George Marsaglia put it, "the points are about as randomly spaced in the unit n-cube as the atoms in a perfect crystal at absolute zero". Later on in his paper he goes over a method that can be used to determine the inputs to such an equation.

2.2 Modular Arithmetic

Modular arithmetic, as well as its properties play an important roll in both the generation, and solving of linear/multiplictive congruential generators. Infact the equations themselves use modular arithmetic, so knowing how it works, as well as its properties will prove useful when solving for the variables. [7]

2.2.1 Residues

An important concept in modular arithmetic is residues, a residue is what is left when you subtract the modulo value as much as you can, before subtracing anymore would result in a negative number. It is also called the remainder in the contest of division. An example would be, 10 mod 3, would

have a residue of 1, as $10-(3+3+3)=1$, subtracing anymore would result in a negative number. Thus the residue is 1. It is also important to note that a residue can be 0, such is the case of 12 mod 3, where $12-(3+3+3+3)=0$, this is a case where the residue is equal to 0. [7]

2.2.2 Congruence

We say a number, or equation is congruent to one another when all of residues/remainders of that value modulo a constant are the same. It is often shown through the sign $\equiv$. An example of it being used correctly would be the equation; $2 \equiv 7 \equiv 12 (\mathrm{mod} 5)$, as each of the values 5 mod 5 will have the same result, meaning they are congruent. [7]

2.2.3 Relations

The relation between $x \equiv b (\mathrm{mod} m)$ and $x = b (\mathrm{mod} m)$. The first one is for equivalence. In comparision the second equation is equality. As a note from Cornell University put it;
"$x = b (\mathrm{mod} m)$ is the smallest positive solution to the equation $x \equiv b (\mathrm{mod} m)$" [8]

2.2.4 Rules to note:

Sum Rule: if $a \equiv b (\mathrm{mod} n)$, and $c \equiv d (\mathrm{mod} n)$ then $a + c \equiv b + c (\mathrm{mod} m)$ [8]

Multiplication Rule: if $a \equiv b (\mathrm{mod} m)$ and if $c \equiv d (\mathrm{mod} m)$ then $a \cdot c \equiv b \cdot d (\mathrm{mod} m)$ [8]

2.2.5 Otherways It Can Be Written

Another way which $a \equiv b (\mathrm{mod} m)$ can be written is $a = k \cdot m + b$, where k is an arbitary integer. Yet another way it can be written is $n|(a-b)$, which means, $a - b$ is a multiple of n. This becomes very useful when solving for the variables later on.

2.2.6 Chinese Remainder Theorm

Chinese remainder theorm is a suprise tool that will help us later on. Chinese remainder therom is a method to solve for x in a system of equations which are written as such; $x = q_n (\mathrm{mod} m_n)$. Ben Lynn of Standford University states it;

$$x = q_1 (\mathrm{mod} m_1)$$

$$x = q_n (\mathrm{mod} m_n)$$

[**?**] For our purposes, only q_n changes, as the value of m stays constant. Meaning our equations will resemble;

$$x = q_1 (\mathrm{mod} m)$$

$$x = q_n (\mathrm{mod} m)$$

But because q_n is simply a constant multiplied by the previous number, like;

$$q_n = a \cdot g$$

Where g is the last value, and a is a constant whole number integer.
So this means our equation really looks like;

$$q_i = a \cdot g (\mathrm{mod} m)$$

From this it has been said that we can use some special trickery (sum rule) to figure out the value of a. However, brute forcing is a valid method, and in many cases is much simplier.

2.2.7 Brute Forcing

Brute force is a method of trial and error, where different values are continously tried until a value works.

2.3 Method Of Solving

The original paper by George Marsaglia [5] mentioned a method which could be used in order to solve for the original inputs given a sufficent amount of input. This was then expanded upon by Haldir [4]. The expanded upon method showcased by Haldir will be used, although first how it works will be explained in this section.

2.3.1 Forming The Matrix

To begin solving for the input values you need to obtatin 6 values from the generator, let these values be $\{1 \leq i \leq 6\}, \{1 \leq x_i | x_i \in \mathbb{Z}^+\}$. Letting i be the index of the value, and x_i being the value.

The method that was used in this investigation, and the method used in Marsaglia's, and Haldir's paper differ here. In their papers they setup the matrix as; [5] [4]

$$\begin{bmatrix} x_1 & x_2 & 1 \\ x_2 & x_3 & 1 \\ x_3 & x_4 & 1 \end{bmatrix}$$

However, during my inital calculations I had made a critical error, and had arranged it as such instead;

$$\begin{bmatrix} x_1 & x_2 & 1 \\ x_3 & x_4 & 1 \\ x_5 & x_6 & 1 \end{bmatrix}$$

At the time I did not relise my error, yet once I reached the end I had gotten the same answer, this shows arranging the matrix as I did by accident also worked. Thus the calculations will continue with the arrangment that was done by accident.

2.3.2 Finding m

Now that the matrix has been arranged as;

$$\begin{bmatrix} x_1 & x_2 & 1 \\ x_3 & x_4 & 1 \\ x_5 & x_6 & 1 \end{bmatrix}$$

We need to find the determinate, as Haldir, and Marsaglia found, the determinte of this matrix is an integer multiple of the value of m in the equation of the LCG/MCGs.

Before finding the determinte, for the sake of simplicty we will assign variable names to each of the points in the matrix before replacing them witht he actual numbers. They will be labeled as such;

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

$$\det(A) = a_1 \cdot \begin{vmatrix} b_2 & b_3 \\ c_2 & c_3 \end{vmatrix} - a_2 \cdot \begin{vmatrix} b_1 & b_3 \\ c_1 & c_3 \end{vmatrix} + a_3 \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix}$$

$$\det(A) = a_1 \cdot (b_2 \cdot c_3 - b_3 \cdot c_2) - a_2 (b_1 \cdot c_3 - b_3 \cdot c_1) + a_3 (b_1 \cdot c_2 - b_2 \cdot c_1)$$

Then after replacing the letters a,b, and c, you end up with the equation below.

$$= x_1 \cdot ((x_3 \cdot 1) - (x_5 \cdot 1)) - x_2 \cdot ((x_2 \cdot 1) - (x_4 \cdot 1)) + ((x_2 \cdot x_5) - (x_3 \cdot x_4))$$

After simplifying this equation one will end up with;

$$\det(A) = x_1 \cdot (x_3 - x_5) - x_2 \cdot (x_2 - x_4) + ((x_2 \cdot x_5) - (x_3 \cdot x_4))$$

Now that we have the determinte for one set of numbers, we need to repeat it a lot of times and record all of the determinates. The theoretical minimum number of determinates needed is found in George Marsaglia's paper [5]. It has a table of the number needed, for the purposes of this investigation the process will just be repeated an abritary amount of times.

Now that we have obtained an arbitary amount of determinates, we will need to find their largest common factor. I suggest using a compute program to find the factors of each number, and then save them. After all of the factors have been found, find the largest common number between them all. An example will be shown below with arbitary numbers 500, 525, 450, 700

$$500 : 1, 2, 4, 5, 10, 20, 25, 50, 100, 125, 250, 500 \quad (1)$$

$$525 : 1, 3, 5, 7, 15, 21, 25, 35, 75, 105, 175, 525 \quad (2)$$

$$450 : 1, 2, 3, 5, 6, 9, 10, 15, 18, 25, 30, 45, 50, ..., 450 \quad (3)$$

$$700 : 1, 2, 4, 5, 7, 10, 14, 20, 25, 28, 35, 50, ..., 700 \quad (4)$$

Some of the factors were ommited because they could not be displayed in an organised manner, and/or they have no affect on the end result. The greatest common factor between all 4 numbers/ In the case of the numbers above, the greatest common factor is 25. Thus the value of m in the equation of an MCG/LCG is 25.

2.3.3 Solving for a in an MCG

To solve for the remaining variables of an LCG, we will use the equation

$$x_i = (x_{i-1} \cdot a) \bmod m$$

Since we have m, and have x_i, we just have to solve for a, which is trivial, it can be done using Chinese Remainder Therom, or using brute force. In addition a is the value which the previous integer is multiplied by before being put through a modulo operation. k is a constant value

3 Investigation

To begin, a multiplictive congruential generator will be used to generate 72 numbers. Then those numbers will be used to find the original inputs to the generator, to check if the inputs are correct, the values which were found will be used as inputs to the multiplictive congruential generator.

3.1 Values

The 72 numbers which were found are stated below;
547, 91, 73, 200, 19, 159, 109, 157, 170, 90, 209, 61, 144, 39, 182, 146, 189, 38, 107, 7, 103, 129, 180, 207, 122, 77, 78, 153, 81, 167, 76, 3, 14, 206, 47, 149, 203, 33, 154, 156, 95, 162, 123, 152, 6, 28, 201, 94, 87, 195, 66, 97, 101, 190, 113, 35, 93, 12, 56, 191, 188, 174, 179, 132, 194, 202, 169, 15, 70, 186, 24, 112
These values will be used for the calculation of the input values of the MCG.

3.2 Setting up the matrixes

Now we will need to setup the 12 matrixes we will be using to find the value of m. The matrixes will be setup as such

$$A = \begin{bmatrix} 547 & 91 & 1 \\ 73 & 200 & 1 \\ 19 & 159 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 109 & 157 & 1 \\ 170 & 90 & 1 \\ 209 & 61 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 144 & 39 & 1 \\ 182 & 146 & 1 \\ 189 & 38 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 107 & 7 & 1 \\ 103 & 129 & 1 \\ 180 & 207 & 1 \end{bmatrix}$$

$$E = \begin{bmatrix} 122 & 77 & 1 \\ 782 & 153 & 1 \\ 81 & 167 & 1 \end{bmatrix}$$

$$F = \begin{bmatrix} 76 & 3 & 1 \\ 14 & 206 & 1 \\ 47 & 149 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 203 & 33 & 1 \\ 154 & 156 & 1 \\ 95 & 162 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 123 & 152 & 1 \\ 6 & 28 & 1 \\ 201 & 94 & 1 \end{bmatrix}$$

$$I = \begin{bmatrix} 87 & 195 & 1 \\ 66 & 97 & 1 \\ 101 & 190 & 1 \end{bmatrix}$$

$$J = \begin{bmatrix} 113 & 35 & 1 \\ 93 & 12 & 1 \\ 56 & 191 & 1 \end{bmatrix}$$

$$K = \begin{bmatrix} 118 & 174 & 1 \\ 179 & 132 & 1 \\ 194 & 202 & 1 \end{bmatrix}$$

$$L = \begin{bmatrix} 169 & 15 & 1 \\ 70 & 186 & 1 \\ 24 & 112 & 1 \end{bmatrix}$$

3.3 Getting the Determinates of the Matrixes

Now that we've arranged all the matrixes we need to get the absolute value of their determinates, absolute values for simplicties sake later on when we need to find the factors. The function $\det(A)$ shall represent it, letting A be the matrix.

$$\det(A) = 547 \cdot \begin{vmatrix} 200 & 1 \\ 159 & 1 \end{vmatrix} - 91 \cdot \begin{vmatrix} 73 & 1 \\ 19 & 1 \end{vmatrix} + \begin{vmatrix} 547 & 91 \\ 19 & 159 \end{vmatrix}$$

$$\det(A) = 25320$$

$$\det(B) = 109 \cdot \begin{vmatrix} 90 & 1 \\ 61 & 1 \end{vmatrix} - 157 \cdot \begin{vmatrix} 170 & 1 \\ 209 & 1 \end{vmatrix} + \begin{vmatrix} 109 & 157 \\ 209 & 61 \end{vmatrix}$$

$$\det(B) = 844$$

$$\det(C) = 144 \cdot \begin{vmatrix} 146 & 1 \\ 38 & 1 \end{vmatrix} - 39 \cdot \begin{vmatrix} 182 & 1 \\ 189 & 1 \end{vmatrix} + \begin{vmatrix} 144 & 39 \\ 189 & 38 \end{vmatrix}$$

$$\det(C) = 4853$$

$$\det(D) = 107 \cdot \begin{vmatrix} 129 & 1 \\ 207 & 1 \end{vmatrix} - 7 \cdot \begin{vmatrix} 103 & 1 \\ 180 & 1 \end{vmatrix} + \begin{vmatrix} 107 & 7 \\ 180 & 207 \end{vmatrix}$$

$$\det(D) = 9706$$

$$\det(E) = 122 \cdot \begin{vmatrix} 153 & 1 \\ 167 & 1 \end{vmatrix} - 77 \cdot \begin{vmatrix} 78 & 1 \\ 81 & 1 \end{vmatrix} + \begin{vmatrix} 122 & 77 \\ 81 & 167 \end{vmatrix}$$

$$\det(E) = 844$$

$$\det(F) = 76 \cdot \begin{vmatrix} 206 & 1 \\ 149 & 1 \end{vmatrix} - 3 \cdot \begin{vmatrix} 14 & 1 \\ 47 & 1 \end{vmatrix} + \begin{vmatrix} 76 & 3 \\ 47 & 149 \end{vmatrix}$$

$$\det(F) = -3165$$

$$\det(G) = 203 \cdot \begin{vmatrix} 156 & 1 \\ 162 & 1 \end{vmatrix} - 33 \cdot \begin{vmatrix} 154 & 1 \\ 95 & 1 \end{vmatrix} + \begin{vmatrix} 203 & 33 \\ 95 & 162 \end{vmatrix}$$

$$\det(G) = 6963$$

$$\det(H) = 123 \cdot \begin{vmatrix} 28 & 1 \\ 94 & 1 \end{vmatrix} - 152 \cdot \begin{vmatrix} 6 & 1 \\ 201 & 1 \end{vmatrix} + \begin{vmatrix} 123 & 152 \\ 201 & 94 \end{vmatrix}$$

$$\det(H) = 16458$$

$$\det(I) = 87 \cdot \begin{vmatrix} 97 & 1 \\ 190 & 1 \end{vmatrix} - 195 \cdot \begin{vmatrix} 66 & 1 \\ 101 & 1 \end{vmatrix} + \begin{vmatrix} 87 & 195 \\ 101 & 190 \end{vmatrix}$$

$$\det(I) = 1477$$

$$\det(J) = 113 \cdot \begin{vmatrix} 12 & 1 \\ 191 & 1 \end{vmatrix} - 35 \cdot \begin{vmatrix} 93 & 1 \\ 56 & 1 \end{vmatrix} + \begin{vmatrix} 113 & 35 \\ 56 & 191 \end{vmatrix}$$

$$\det(J) = 4431$$

$$\det(K) = 188 \cdot \begin{vmatrix} 132 & 1 \\ 202 & 1 \end{vmatrix} - 174 \cdot \begin{vmatrix} 179 & 1 \\ 194 & 1 \end{vmatrix} + \begin{vmatrix} 188 & 174 \\ 194 & 202 \end{vmatrix}$$

$$\det(K) = 0$$

$$\det(L) = 169 \cdot \begin{vmatrix} 186 & 1 \\ 112 & 1 \end{vmatrix} - 15 \cdot \begin{vmatrix} 70 & 1 \\ 24 & 1 \end{vmatrix} + \begin{vmatrix} 169 & 15 \\ 24 & 112 \end{vmatrix}$$

$$\det(L) = 15192$$

Excluding the determinate(s) which are 0, we have the determinates, 25320, 844, 4853, 9706, 844, 3165, 6963, 16458, 1477, 4431, 15192.

3.4 Solving for the inital values

From the determinates of the matrix we can solve for m, using the method which was previously states. That is, to find the greatest common factor between all of the determinates. After solving for the greatest common factor, a value of 211 is obtained. That means that the equation of the MCG that is generating these values should look similar to $x_{i+1} \equiv (a \cdot x_i)(\text{mod}211)$.
Using some of the equations previously mentioned, we can re-arrange them to craft the equation. [4]

$$a \cdot x_{i-1} \equiv x_i(\text{mod}m)$$

Since we know the value of m is 211, we can insert it into the equation, getting

$$a \cdot x_{i-1} \equiv x_i(\text{mod}211)$$

To determine a we shall use 6 different values of x, if the solved values of a are all the same, we will have found a. The values used will be; 109, 157, 73, 200, 19, and 159. Putting these into equations we get the equations;

$$157 \equiv 109 \cdot a(\text{mod}211)$$
$$200 \equiv 73 \cdot a(\text{mod}211)$$
$$157 \equiv 109 \cdot a(\text{mod}211)$$

By brute forcing these equations for the value of a, and using wolframAlpha, a value of 75 is obtained.

Thus the values of a is 75, and m is 211. You only need one value from the generator to predict all the values which come after it, so we have everything that is needed.

3.5 Checking The Values

To check our values, we will need one value from the MCG to act as the seed, and then we'll need to solve for the next values, if the values line up, we'll know we had the correct values. Using the equation $x_i = x_{i-1} \cdot a(\text{mod}m)$ we can solve for them. The value of x_1 will be 547, as that is the first value to come out of the generator, the values of m, and a, have already been solved for, and were shown to be 211, and 75 respectively. This means we get an equation of $x_i \equiv x_{i-1} \cdot 75(\text{mod}211)$. Inserting in 547, the number we get out is 91. Repeating that 10 times, we get the sequence; 547, 91, 73, 200, 19, 159, 109, 157, 170, 90.Which perfectly matches the order, and values that were extracted from the actual generator, meaning our values are correct.

3.6 Applications/Implications of Knowing The Inputs

There are many implications of being able to predict the next "random" number, infact there are so many implications that we do not really know just how far reaching the consequences are. One of the implciations as stated before is about about cryptography, as quite a few badly built cryptographyic programs use Linear, and Multiplicative Congruential Generators as randomness generators. Another consequence is the use of random in video games, and computer graphics. In video games they are often used to place objects, or to randomise behaviours of characters. Whilst in computer graphics random generators are used to seed the rendering equations to make renders seem more realistic, as there is "random". So these random number generators being predictable means that, these applications are not getting truely random values.

Now some of the applications of this would be; being able to predict the next drops in a gacha game, the outcomes of certain actions in games which have some element of random, or being able to decrypt communications.

3.7 Limitations

However, there many limitations of this method to determine the input variables using the outputs. The first and largest limitation is, determing whether an LCG/MCG is being used, as you would need access to the code of the application in order to determine whether an LCG/MCG is being used. The alternative method of a spectral anal-

ysis to determine whether an LCG/MCG would require a significant number of data points, which may be unrealistic to collect.

The next limitation would be the amount of numbers needed, as to extract the required number of values needed would take a significant amount of time and energy.

Then there is the time it would take to find the GCD of larger numbers, as computing the factors of numbers takes a sufficiently long time, and as the numbers grow larger, the time taken also grows. So if a lot of values were extracted, and thus determinates found, it would take quite a while to find the value of m. However the use of Chinese Remainder Therom, and Stein's algorithim could potentially decrease the time taken. As these algorithims would allow you to find the factors, and the value of m with far less brute forcing than otherwise.

3.8 Extensions

There are many methods to extend this investigation, such as; investigating the use of Steins's algorithim, and Chinese Remainder Therom. Steins's algorithim is often used to find the greatest common denominator between two non-negative numbers u, and v by repeatedly applying these identities;

1. Let $gcd(a, b)$ be a function which returns the greatest integer by which both values can be divided by which results in a posititive integer

2. $gcd(0, b) = b$, and symmetrically; $gcd(a, 0) = a$

3. $gcd(2a, 2b) = 2 \cdot gcd(a, b)$

4. $gcd(2a, b) = gcd(a, b)$ if b is odd,
   then $gcd(a, b) = gcd(a, 2b)$ if a is odd

5. $gcd(a, b) = gcd(|a - b|, min(a, b))$,
   if a and b are both odd

(Note: Symmetrically in this context, means the other way around, as in the values swapped) Stein's algorithim is simply the continous application of these rules until the gcd is found. [**?**]. The use of Stein's algorithim would be a great step forwards when it comes to decreasing the time taken to find the value of m.

Continuing, Chinese Remainder Therom as previously mentioned could be used in order to make the calculations done by hand much faster. It could also be applied to solve for a Linear Congruential Generator. However the existence of fast computers makes brute forcing a much more reasonable operation. Until the values are sufficently large, where solving by hand is quicker. [5]

References

[1] University Of Waterloo Math Department. generating random numbers. https://wiki.math.uwaterloo.ca/statwiki/index.php?title=generating_Random_Numbers#Inverse_Transform_Method, 2009.

[2] Juk Developers. playlist.cpp. https://github.com/KDE/juk/blob/master/playlist.cpp#L676, 2024.

[3] Michel Goossens, Frank Mittlebach, and Alexander Samarin. *The Latex Companion A*. Addison-Wesley, Reading, Massachusetts, 1993.

[4] HaldirRET. How to crack a linear congruential generator. *reteam.org*, 2004. https://www.reteam.org/papers/e59.pdf.

[5] George Marsaglia. Random numbers fall mainly in the planes. *Mathematics research Laboratory, Boeing Scientific Research Laboratories*, 1968.

[6] Huzaifa Sidhurwala. Understanding random number generators and their limitations, in linux. *Red Hat Blog*, 2019.

[7] Art Of Problem Solving. Modular arithmetic/introduction. https://artofproblemsolving.com/wiki/index.php/Modular_arithmetic/Introduction, 2024.

[8] Cornell University. Everything you need to know about modular arithmetic. https://pi.math.cornell.edu/~morris/135/mod.pdf, 2006.