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. For randomness could at times not seem random, for example, number generators more often than not are far from random, instead they are psuedo random number generators, which are not truely random. These pesudo random number genrators are not truely random, but rather just a mathematical equation which is being done itteratively, and if the developer was forward thinking, it used a different seed 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 genreatofr 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 $x_{i+1} = (a * x_i) \bmod m$. Where the next value is equal to the current value multiplied by a constant, and then its modulus is set as the remainder. [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. Its equation looks like $x_{i+1} = (a * x_i + b) \bmod m$ [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 jus the implmentations that are public and known about. Considering that these types of random number generators are also used in glibc's [7] 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. [8]

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. [8]

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; $2 \equiv 7 \equiv 12 (\text{mod} 5)$, as each of the values 5 mod 5 will have the same result, meaning they are congruent. [8]

2.2.3 Relations

The relation between $x \equiv b(\text{mod} m)$ and $x \equiv b(\text{mod} m)$. The first one is for equivalence, which i sthe same as equality. In comparision the second equation is equality. As a note from Cornell University put it;
"$x = b(\text{mod} m)$ is the smallest positive solution to the equation $x \equiv b(\text{mod} m)$" [9]

2.2.4 Rules to note:

Sum Rule: if $a \equiv b(\bmod m)$ then $a + c \equiv b + c(\bmod m)$ [9]

Multiplication Rule: if $a \equiv b(\text{mod} m)$ and if $c \equiv d(\text{mod} m)$ then $a \cdot c \equiv b \cdot d(\text{mod} m)$ [9]

2.2.5 Otherways It Can Be Written

Another way which $a \equiv b(\text{mod} m)$ 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.

2.3 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;

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

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

Some asdclaisdm [3]
Department [1]
noi [6]

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

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

$$= x_1 \cdot ((x_3 \cdot 1) - (x_5 \cdot 1)) - x_2 \cdot ((x_2 \cdot 1) - (cal_4 \cdot 1)) + 1 \cdot ((x_2 \cdot x_5) - (x_3 \cdot x_4))$$

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

$$500 : 1, 2, 4, 5, 10, 20, 25, 50, 100, 125, 250, 500 \tag{1}$$

$$525 : 1, 3, 5, 7, 15, 21, 25, 35, 75, 105, 175, 525 \tag{2}$$

$$450 : 1, 2, 3, 5, 6, 9, 10, 15, 18, 25, 30, 45, 50, ..., 450 \tag{3}$$

$$700 : 1, 2, 4, 5, 7, 10, 14, 20, 25, 28, 35, 50, ..., 700 \tag{4}$$

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 MCG, we will use the equation

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

Since we have m, and have x_i, we just have to solve for a, which is trivial. In addition a is the value which the previous integer is multiplied by before being put through a modulo operation.

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] noi. *the whack man.* nomanssky, reading Massachusetts, 2001.

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

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

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