Random Numbers & the Graphing Calculator

Ryan Williamson

April 13, 2018

Abstract

As random as a subject this could be, it's all about random numbers, the graphing calculator and how the graphing calculator uses random numbers to perform certain functions for the user such as finding the minimum and maximum of functions. Calculus and algebra will be used to compare results obtained using the graphing calculator and MatLab. In addition, we used other coding languages to aid our research such as MatLab and AutoHotkey.

1 Introduction

A Random Number is an unbiased choice for a number and is not more likely to be one number than another. A random number must be given some sort of boundaries for which the random number is chosen. Specific qualities must be given to the number such as: real, whole, rational, natural, positive, greater than 73.35... etc. Truly random numbers are chosen once and then forgotten so that no other random numbers can be chosen based off of that previously chosen random number. Some uses of random numbers include: choosing a random winner for a contest, encryption and decryption, selection of jurors, as well as any other situation in which fairness is required.

Imagine you are extremely rich and would like to give away $1,000 to 1 household in your neighborhood of about 30 houses. How would you decide which house to pick? You would have some bias when driving or walking down the road. Some biases include: "Wow that house is nice!" "My feet are killing me.." "I've never met the people who live there before" "Hey that family is outside now".. The list could go on forever. One of the best methods to give to a truly random household would be to label the houses 1-30 on a map. Then write down 1-30 on slips of paper and remove your house number. Then put the 29 other pieces into a hat. Give your hat a shuffle. Finally pick a slip of paper out of the hat. Find the house with that number. Then don't even think about doing a re-do. Just go give them their money. That person won. This was an example of a discrete small sample size for a random choice.

Some cases require truly random selection from a much larger group of people. Say, for example, the entire population of United States citizens who are 18 years old or older. You may have wondered in the past how courts choose who will be summoned as a juror. They use your Social Security Number.

2 Random Number Generators(RNGs)

The goal of a Random Number Generator is to provide numbers to be independent of each other and not identically distributed. They use a Random Number Generator to generate 9 numbers per try from 1-9 inclusive. They then have requirements that their RNG knows such as all digits cannot be the same digit (E.g. 333-33-3333). The court must also have the SSNs of residents close to the area in which they are located. If the RNG generates one of these numbers and they have not previously served, then they summon this person.

There are two main types of Random Number Generators. Neither of which actually are truly Random. The first of the two is a Pseudo RNG. With a pseudo RNG, numbers are generated based on the previous number. The first number, a seed, produces the numbers with an iteration in the following form:

$$x_i = ax_{i-1} + b(\text{mod } m)$$
$$u_i = \frac{x_i}{m}$$

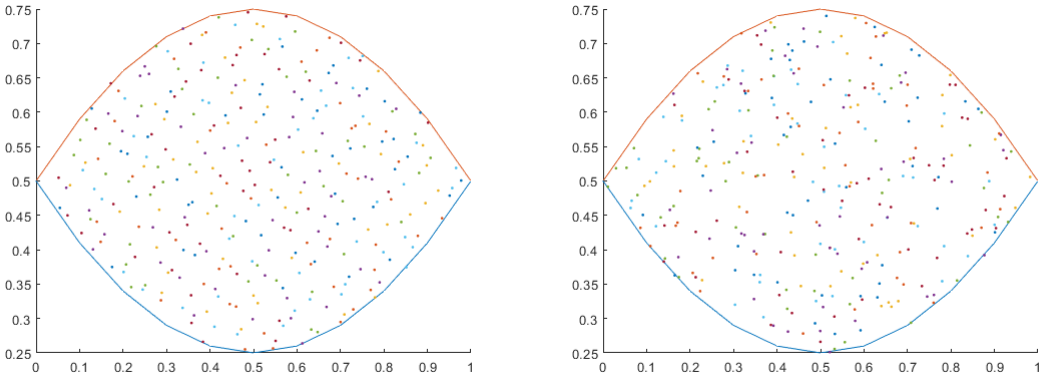
Where a is a multiplier; b is an offset; m is the modulus; m-1 is the period of the generator; and u is the number generated. An example of a pseudo-random number generator is below. The first 10 numbers generated with $x_0 = 7$ (our seed) is below.

$$x_i = 17x_{i-1}(\text{mod } 33)$$
$$u_i = \frac{x_i}{33}$$

x	u
20	0.6061
10	0.3030
5	0.1515
19	0.5758
26	0.7879
13	0.3939
23	0.6970
28	0.8485
14	0.4242
7	0.2121

With a Quasi RNG, numbers are generated to be "equally distributed" or fill in the missing holes that a pseudo RNG would not fill.

The difference between Quasi RNG (left) and Pseudo RNG (right) is shown below where the x and $f(x)$ values are both generated with their respective generator:

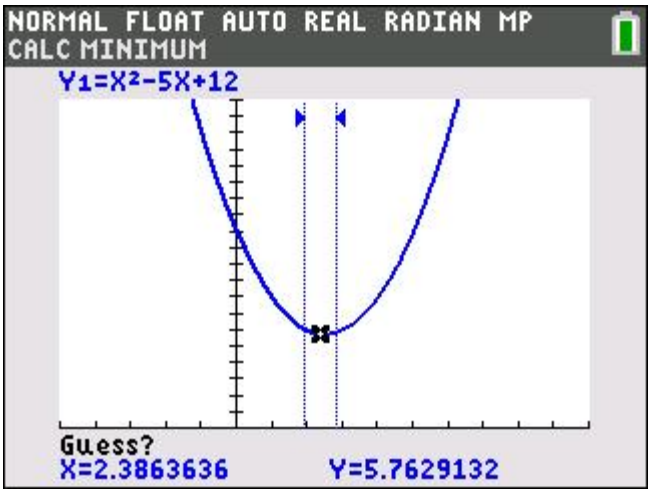


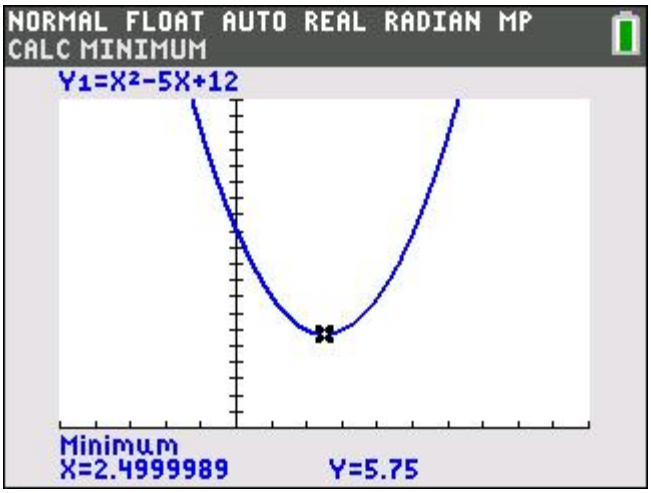
I used three different coding methods to showcase how helpful Random Numbers can be. Two of these methods were written through coding languages, and the other was by hand: Calculus. The Calculus method, if done correctly and if able to use Calculus for the certain problem, will be 100% accurate. Sometimes Calculus cannot be used to determine the minimum or maximum or a function. An example of a polynomial which its minimums and maximums cannot be found with calculus is: $f(x) = 3x^8 - 7x^5 + 15x + 3$. Taking the derivative we find $f'(x) = 24x^7 - 35x^4 + 15$. This is where random numbers come into play since there isn't a formula we have learned nor any way to factor. One way would be to use a Graphing calculator.

3 Minimum and Maximum Problems

The MatLab code on the next page is how the Graphing calculator finds the minimums and maximums of functions. This can be changed to accept an array of coefficients instead of begin limited to a quadratic. For simplicity I chose to include a set number of coefficients as just a quadratic would suffice for now. Feel free to copy, paste and save as either Minimum.m or Maximum.m respectively for each function in order to test the code. You will of course need MatLab installed to run these codes.

```
function maximum = Maximum(A,B,C,D,a,b,g,n)
%   Max: Find the Maximum of a function
%   Given the interval [a,b]
%   Generate Random Numbers for the x-values between the interval,
%    starting with the guess, and using these x-values
%    to find the y-values of the function
%   The largest value in the array of y-values will be our Minimum
%   A,B,C,D refers to f(x)=Ax^3+Bx^2+Cx+D
%   [a,b]:interval
%   g:guess
%   n:number of random numbers
%   Example usage:
%       Maximum(1, 3, -9, 7, -3.841463, -1.890244, -3.109756, 10000)
X=1:n;Y=1:n;
x=g;
x=mod(16807.*x,2^31-1);
for i=1:n
    X(i)=a+(b-a).*(x/(2^31-1));
    Y(i)= A*X(i)^3+B*X(i)^2+C*X(i)+D;
    x=mod(16807.*x,2^31-1);
end
[M,I] = max(Y);
maximum=[X(I) M];
end

function minimum = Minimum(A,B,C,a,b,g,n)
%   Min: Find the Minimum of a function
%   Given the interval [a,b]
%   Generate Random Numbers for the x-values between the interval,
%    starting with the guess, and using these x-values
%    to find the y-values of the function
%   The lowest value in the array of y-values will be our Minimum
%   A,B,C refers to f(x)=Ax^2+Bx+C
%   [a,b]:interval
%   g:guess
%   n:number of random numbers
%   Example usage:
%       Minimum(1, -5, 12, 1.9318182, 2.8409091, 2.3863636, 10000)
X=1:n;Y=1:n;
x=g;
x=mod(16807.*x,2^31-1);
for i=1:n
    X(i)=a+(b-a).*(x/(2^31-1));
    Y(i)= A*X(i)^2+B*X(i)+C;
    x=mod(16807.*x,2^31-1);
end
[M,I] = min(Y);
minimum=[X(I) M];
end
```





MatLab Function call:

```
>> Minimum(1, -5, 12, 1.9318182, 2.8409091, 2.3863636, 10000)

ans =

    2.5000    5.7500
```

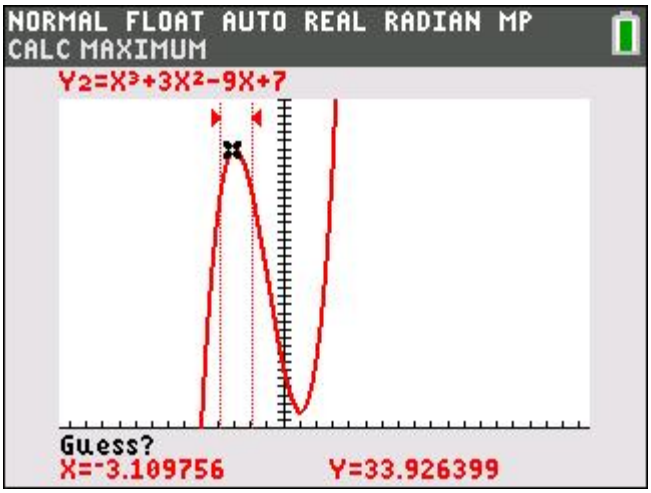
To find the Minimum with Calculus, we use the formula:

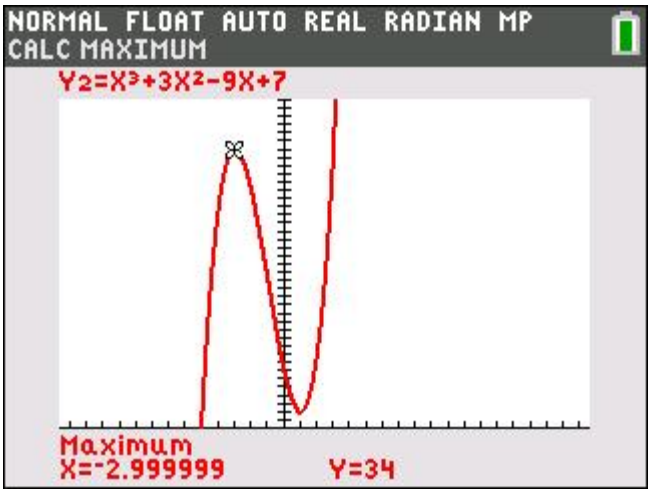
$$x = \frac{-b}{2a}$$

where

$$f(x) = ax^2 + bx + c$$

is the standard form of a quadratic equation. We then find the function's value at the x value from the $x =$ formula.
In our example, $f(x) = x^2 - 5x + 12$, so $x = \frac{5}{2}$ and $f(x) = (\frac{5}{2})^2 - 5(\frac{5}{2}) + 12 = \frac{23}{4}$
We then found the following for our errors:
Actual Minimum: (2.5, 5.75)
Error for Graphing Calculator: 0.0000011
Error for MatLab Code: 0





MatLab Function call:

```
>> Maximum(1, 3, -9, 7, -3.841463, -1.890244, -3.109756, 10000)

ans =

   -2.9999   34.0000
```

To find the Maximum with calculus, we must take the derivative, then find the zeroes of the quadratic.
In our example, $f(x) = x^3 + 3x^2 - 9x + 7$, so $f'(x) = 3x^2 + 6x - 9$, then setting $f'(x) = 0$ we have $0 = 3(x^2 + 2x - 3) = 3(x + 3)(x - 1)$. Then the critical points are below. We then divide the real number line into intervals where the critical points are the dividers. Next determine the sign (+ or -) of $f'(x)$. Next, a maximum occurs from the transition of a positive $f'(x)$ to a negative $f'(x)$. Then we find the function value at $x = -3$
$f(-3) = (-3)^3 + 3(-3)^2 - 9(-3) + 7 = 34$

Critical Points: -3 and 1					
x	left	-3	middle	1	right
$f'(x)$	+	0	-	0	+

We then found the following for our errors:
Actual Maximum: (-3, 34)
Error for Graphing Calculator: 0.000001
Error for MatLab Code: 0.0001

There are times in which the BEST and the ONLY way to find minimums and maximums are with Random Numbers. An example would be:

$$f(x) = 3x^8 - 7x^5 + 15x - 3$$
$$f'(x) = 24x^7 - 35x^4 + 15$$

Notice how there is no way to factor the derivative nor any formula we can use to solve for the extrema. This is where we must use random numbers to find an interval in which they occur, then generate random numbers in that interval for the x-values to then find the smallest(min) or the greatest(max) y-value.

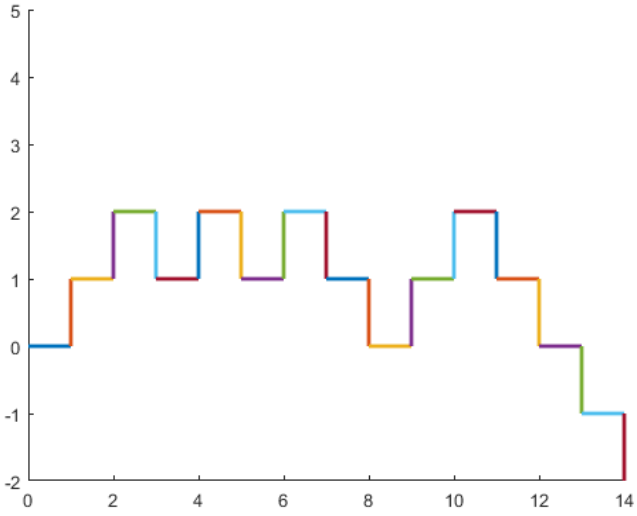
4 Other Random Number Applications

Another program I created in the language of AutoHotkey, open source located at https://autohotkey.com/. My goal for the program was to create a program to type random digits with random font sizes. I first randomly generated 35 sizes 8-32. Then randomly generated 35 digits 0-9. With the help of https://www.random.org/ this was an easy task. My favorite part about AutoHotkey is that the code is very easy to understand. I gave comments before each of the steps the code takes denoted by a semicolon at the front of the line, then a large arrow to get the reader's attention. The .ahk file is below. Feel free to copy, paste and save as literally anything with the file extension "ahk" such as "RN.ahk" in order to test the code. Also you will need to download and install AHK from the above link in order to run ahk files. More than likely the "default" position for your Font button will not work, so you must set it with hovering over the font button then pressing (without pressing shift) the key to the left of "1" (named backquote or backtick). You must have the file running when you do this with of course Google Slides with a new slide up with a text field. Then you press F1 to Start/resume, and if needed during the program, press Esc to Pause.

```
;------------------------------>set random sizes and digits into an array
Sizes:=[20,16,29,26,24,9,19,26,10,25,19,25,22,31,23,30,29,27,8,13,31,14,18,29,
    21,20,11,15,20,16,24,13,20,13,22]
Digits:=[9,5,3,0,2,3,8,9,2,7,7,7,7,4,5,0,4,1,4,1,4,8,0,6,7,4,7,4,7,4,2,9,6,2,4,6,0]
x:=799 ;Default x position for font button
y:=180 ;Default y position for font button
Pause
Loop 35
{
size:=Sizes.pop()
digit:=Digits.pop()
;------------------------------>activate window then click
SetTitleMatchMode, 2
WinWait, Google Slides,
IfWinNotActive, Google Slides, , WinActivate, Google Slides,
WinWaitActive, Google Slides,

MouseClick, left, %x%, %y%
Sleep, 1000
;------------------------------>type size
Send, %size% {Enter}
Sleep, 1000
;------------------------------>type number
Send, %digit%
}
;------------------------------>gives the ability to Pause/resume with
    Esc/F1 keys
Esc::Pause, on
F1::Pause, off
;------------------------------>set position of the font button
`::MouseGetPos, x, y ;key to the left of "1"
```

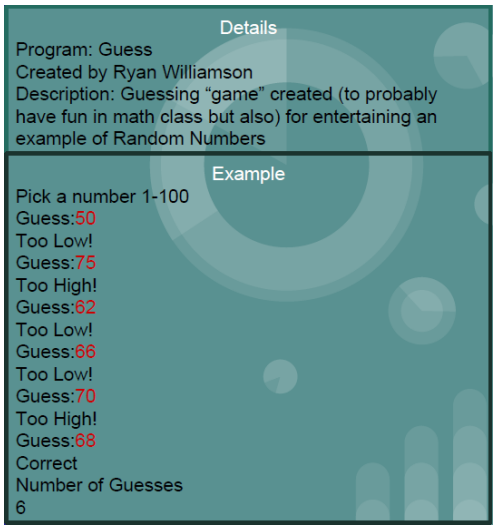
The following code with provided picture is from a random walk which this code will execute 10,000 random walks to determine the average time it takes for the walk to exceed the y-boundaries. A random walk is where, in this case a graph on the integer number line, we start at 0 and then either increase or decrease by 1 with equal likelihood until we exceed some pre-determined boundaries (below and above the x-axis such as [-2, 5]). Feel free to copy, paste and save as RandomWalk.m in order to test the code. You will of course need MatLab installed to run this code.

```
function [A] = RandomWalk(b,a)
% Exercise 9.3 #2 in Numerical Analysis by Tim Saur
% use Monte Carlo to estimate how many steps to reach
% -b or a of given interval [-b,a] with a random walk
%   (a) [-2,5] (b) [-5,3] (c) [-8,3]
% Example usage: (input -b, not b!)
% RandomWalk(2,5)
% RandomWalk(5,3)
% RandomWalk(8,3)
Exits=0;
EscTime=1:10000;
for n=1:10000
    w=0;
    Count=0;
    while(w~=a && w~=-b)
        Count=Count+1;
        if rand>1/2
            w=w+1;
        else
            w=w-1;
        end
        if (w==a || w==-b)
            Exits=Exits+1;
            EscTime(n)=Count;
        end
    end %while
AvgEscTime=sum(EscTime)/10000;
err = abs(a*b-AvgEscTime);
A=[AvgEscTime err];

end %for

end %function
```



The following code with provided example and details demonstrates a guessing game coded in the Graphing Calculator in which the user inputs a number between 1 and 100 and the calculator tells the user if his or her guess was too low, high, or correct displaying the number of guesses afterwards.

```
100->C
0->N
randInt(1,C)->I
Disp "Pick a number 1-100"
Lbl L
0->G
Input "Guess:", G
If G=I: Then
N+1->N
Disp "Correct", "Number of Guesses:", N
Stop
End
If G<I: Then
Disp "Too Low!"
1+N->N
End
If G>I: Then
Disp "Too High!"
1+N->N
End
Goto L
```



5 Conclusion

Throughout history, if one person said to another "hey, pick up a random rock for me", whether it be Babylonians in 1800 BCE (probably on Earth's ground..) or Astronauts in year 3000 (on some celestial object), there will always be a use for the word "Random". Random numbers can be helpful in many different applications such as keeping credit card information safe, helping calculate minimums and maximums, and filling your want for a random natural number between a certain bounded interval. I decided to look more in depth on Random numbers and the Graphing Calculator because I found them interesting as I use random numbers every day in coding, in gaming, and even for things I didn't know used Random Numbers such as Credit Card Encryption.

6 Acknowledgements

Without Dr. Cazacu's assistance with the research of random numbers and concepts to show the reader, I would not have been able to write this project. I really appreciate all of her guidance.

7 Bibliography

References

[1] Diehm, Adam. "Free Online Comma Separating Tool". http://Delim.co/

[2] Frenkel, Edward. "An Excerpt from the Book Love and Math." https://math.berkeley.edu/~frenkel/RSA.pdf

[3] Frenkel, Edward. "Prime Numbers and Pierre Fermat Keep Your Secrets Safe Online." Slate Magazine, Slate, 3 June 2013, http://www.slate.com/articles/health_and_science/science/2013/06/online_credit_card_security_the_rsa_algorithm_prime_numbers_and_pierre_fermat.html.

[4] Haahr, Mads. "True Random Number Service." RANDOM.ORG - True Random Number Service, www.random.org.

[5] Sauer, Tim. Numerical Analysis. Pearson Addison Wesley, 2006.