# The Ngazi Method of Exponential Function Calculation.

Kuria Tony Kimani[1]

[1]Affiliation not available

May 5, 2020

The Ngazi Method of Exponential Function Calculation

Msc. Student, Research Methods

Jomo Kenyatta University of Agriculture and Technology.

Tel: +254-713 172 555

e-mail: tkuria19@gmail.com



**Abstract**

This paper presents an entirely new way of calculating exponential functions. This method uses the 2n sequence at its foundation. An example of the 2n sequence is 2,4,8,16,32,64 and so on. This method is vastly more time saving and energy saving because one performs very few multiplication operations once an exponent has been broken down into its 2n components. This method can be coded into a computer software and this will improve the speed with which computer libraries calculate exponential functions. This paper also explains how this new method of exponential function calculation can partly help to solve the discrete exponential and discrete logarithm problem through easier calculation of exponential functions.

## 1.Introduction.

"Neural network simulations often spend a large proportion of their time computing exponential functions. Since the exponentiation routines of typical math libraries are rather slow, their replacement with a fast approximation can greatly reduce the overall computation time." [1]

It is true that computer math libraries that deal with the exponential functions are rather slow but this is a function of using very archaic and long routine calculations which are time-consuming and energy wasting. If a more time-efficient method for calculating exponential functions can be found and then coded into a computer math library then this will save a lot of time for many mathematicians, engineers and computer scientists who use these exponential function libraries a lot. This author presents a new method of calculating exponential functions which is both accurate and time-saving. We no longer need to rely on time-saving approximations of exponential functions. We can now use time-saving accurate calculations of exponential functions and this could indeed change the field computational science.

There is a relationship between exponential functions and logarithms. The power of logarithms as a computational device lies in the fact that by them multiplication and division are reduced to the simpler operations

of addition and subtraction.[2]

"There are many applications of exponential functions and logarithmic functions in science and technology. The voltage in a given circuit can be expressed using exponents. The value of money in an investment can be determined through the use of exponents. The intensity of earthquakes is measured by a logarithmic scale. The intensity of light related to the thickness of the material through which it passes can be expressed using exponents. The distinction between acids and bases in chemistry is measured in terms of logarithms." [3]

When it comes to exponential functions, the word *exponent* is often used instead of index, and functions in which the variable is in the index (such as $2^x$, $10^{\sin x}$) are called exponential functions. [4] If b is a real number greater than zero, then for each real exponent x we assume $b^x$ is a unique real number. Since for each real x there is one and only one $b^x$, the equation $y=b^x,(b>0)$

defines a function. We call such an equation an exponential function.[3]

**2. Body** .

To solve the exponential function $f(x)=m^x$, the most common method of calculating exponential functions has been to directly multiply m, x number of times. This method of calculation is extremely slow especially if the exponent has a large value. For, example calculating

$f(x) = 35^{123000}$ involves multiplying 35, 123,000 times. This is an extremely difficult task and is frankly almost impossible if calculated manually. Computers are best suited to calculate this function because computers do not get tired of performing repetitive tasks. However, there is a need to come up with a more efficient way of calculating the exponential function. This is important because exponential functions are very important in mathematics and engineering fields.

This paper introduces a new, novel method of calculating the exponential function. This method is extremely efficient and it uses the $2^n$ sequence at its foundation. One can solve the exponential function quicker if one uses the $2^n$ table. A sample of the table will be displayed towards the end of the paper.

I have decided to call this method the Ngazi method. Ngazi is a swahili word for ladder or stairs. Ngazi is a two syllable word which is pronounced as (nga-zi) where /n/ and /g/ are pronounced as one syllable.[5] I named this method ngazi or stairs because the results of the first multiplication are used in the second multiplication and the results of the second multiplication are used in the third multiplication and so on. Therefore the first multiplication is linked to the second multiplication and the second multiplication is linked to the third multiplication and this reminds the author of a flight of stairs where one stair leads to the next stair up to the final stair.

*2.1 Introduction to the ngazi method*

Suppose you are trying to calculate the function $f(x)= m^{16}$.

The method you would use is probably the normal method we always use for calculating exponential functions. You would multiply m by itself 15 times to get $m^{16}$.

$f(x)= m\times m\times m\times m\times m\times m\times m\times m\times m\times m\times m\times m\times m\times m\times m\times m = m^{16}$.

However, this method is tedious, time-consuming and inefficient.

Now let us use the ngazi method to calculate this function. I will calculate it first, then I will explain after the calculation.

**Example 1**

$f(x)= m^{16}$

$f(x)= m\times m = m^2$.

$m^2\times m^2 = m^4$

$m^4 \times m^4 = m^8$

$m^8 \times m^8 = m^{16}$

Notice that this new method exploits in its calculation, the fact that $m^n \times m^n = m^{2n}$. This is a well known property of the exponential function and in this paper, this property will be applied to all cases of exponential functions including functions that have exponents that do not belong to the $2^n$ sequence.

When two exponential functions are multiplied together, and the bases are similar, then we can add the exponents together. Therefore, instead of multiplying m by itself 15 times, we can multiply the results of every multiplication by itself and we will end up with only four distinct multiplication operations. Therefore we have reduced the number of multiplication operations from 15 to 4. We can calculate the percentage of reduction in inefficiency in multiplication operations. 15-4 = 11. Therefore, we have reduced the number of multiplication operations by 11. Therefore, in percentage terms we have reduced inefficient multiplication operations by $(11 \div 15) \times 100 = 73.33\%$. This is a 73.33% reduction in inefficient multiplication operations and it is a significant improvement in efficiency. What is magical about this method is the fact that the reduction in inefficiency actually increases as the value of the exponent keeps on increasing. I will give a second example with a slightly larger exponent value to prove this salient point.

**Example 2**

Calculate $f(x) = m^{256}$

Using the normal method

$f(x) = m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m$

$m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m \times m = m^{256}$.

Notice that each row has 16 operands of m and we have a total of 16 rows therefore we have a total of

$16 \times 16 = 256$ operands of m. We also have 255 (256-1) multiplication operations. To get the number of multiplication operations, subtract one from the total number of operands. 255 multiplication operations is unacceptably large and a more efficient method ought to be used instead. Let us use the ngazi method to calculate the function $f(x) = m^{256}$

The ngazi method

Calculate $f(x) = m^{256}$

$f(x) = m \times m = m^2$.

$m^2 \times m^2 = m^4$

$m^4 \times m^4 = m^8$

$m^8 \times m^8 = m^{16}$

$m^{16} \times m^{16} = m^{32}$

$m^{32} \times m^{32} = m^{64}$

$m^{64} \times m^{64} = m^{128}$

$m^{128} \times m^{128} = m^{256}$

Thus we have used only eight multiplication operations instead of the 255 multiplication operations that we used in the normal calculation. Therefore, this new method of exponential function calculation is extremely efficient. Let us calculate how efficient the calculation of the second example was. We used only 8 multiplication operations instead of 255 multiplication operations and the difference between the two is 255-8 = 247. Therefore, we have reduced the number of multiplication operations by 247. Therefore, in percentage terms we have reduced inefficient multiplication operations by $247 \div 255 \times 100 = 96.86\%$. This is an extremely impressive level of improvement in efficiency. This improvement can be increased up to 99% for very large exponent values. I will give an example of such a large exponent value with the following function $f(x) = m^{65536}$ however due to space constraints, I will not list the 65,535 multiplication operations because that will fill several pages of paper. I will jump straight to the ngazi method.

The ngazi method

**Example 3**

Calculate $f(x) = m^{65536}$

$f(x) = m \times m = m^2$.

$m^2 \times m^2 = m^4$

$m^4 \times m^4 = m^8$

$m^8 \times m^8 = m^{16}$

$m^{16} \times m^{16} = m^{32}$

$m^{32} \times m^{32} = m^{64}$

$m^{64} \times m^{64} = m^{128}$

$m^{128} \times m^{128} = m^{256}$

$m^{256} \times m^{256} = m^{512}$

$m^{512} \times m^{512} = m^{1024}$

$m^{2048} \times m^{2048} = m^{4096}$

$m^{4096} \times m^{4096} = m^{8192}$

$m^{8192} \times m^{8192} = m^{16384}$

$m^{16384} \times m^{16384} = m^{32768}$

$m^{32768} \times m^{32768} = m^{65536}$

Thus we have used only 15 multiplication operations instead of the 65,535 multiplication operations that we would have used in the normal method. Let us calculate how efficient the calculation of this third example was. We used only 15 multiplication operations instead of 65,535 multiplication operations and the difference is 65,535-15 = 65,520. Therefore, we have reduced the number of multiplication operations by 65,520. Therefore, in percentage terms we have reduced inefficient multiplication operations by 65,520÷65,535×100= 99.98%. I do not think that there is any method of exponential calculation out there that is more efficient than this method.

Notice that there is a relationship between the exponents in the 3 examples. The three exponents are 16, 256 and 65,536. The peculiar thing about these three numbers is that they all belong to the same sequence. They belong to the $2^n$ sequence. This is an important fact.

Despite how efficient this method is, this method tends to be slightly more inefficient if the value of the exponent does not belong to the $2^n$ sequence. I will give several examples of exponential functions whose exponents do not belong to the $2^n$ sequence. However, this increase in inefficiency is only slight and will not greatly affect the overall efficiency of this new method. Let us calculate the following function using the ngazi method, $f(x) = m^{310.}$

**Example 4**

Calculate $f(x) = m^{310}$

When we are dealing with exponents that do not belong to the $2^n$ sequence, the first step is to multiply the results of each operation by itself as many times as it takes for the result to be greater than 310. This simply means continue with the multiplication operation until the result is greater than the value of the function you are trying to calculate.

*Step 1*

*Table 1*

| Operation | Number of multiplication operations required |
|---|---|
| $m \times m = m^2$ | 1 |
| $m^2 \times m^2 = m^4$ | 2 |
| $m^4 \times m^4 = m^8$ | 3 |
| $m^8 \times m^8 = m^{16}$ | 4 |
| $m^{16} \times m^{16} = m^{32}$ | 5 |
| $m^{32} \times m^{32} = m^{64}$ | 6 |
| $m^{64} \times m^{64} = m^{128}$ | 7 |
| $m^{128} \times m^{128} = m^{256}$ | 8 |
| $m^{256} \times m^{256} = m^{512}$ | 9 |

As you can see, we have multiplied the results of each operation by itself until we got a result that was greater than the value of the exponent (310). Next, we need to take into consideration the exponent value 256. This is the penultimate exponent value in the $2^n$ sequence table above and it is less than the exponent value 310 that we are calculating and it is also less than the exponent value 512 (which is the exponent value in the$^{2n}$ sequence greater than the value we are calculating).

This exponent should always fall in the penultimate row. We have no use for the ultimate row other than the fact that the final row acts as a place holder. It marks the place where the operation needs to stop but the final row itself is not involved in any mathematical calculations.

The second step will involve subtracting the penultimate exponent from the exponent that we are calculating.

*Step 2.*

Thus, we subtract 256 from 310 which is 310-256 = 54. Obviously the value 54 does not belong to the $2^n$ sequence. Therefore, the function $f(x)=m^{310}$ can be rewritten as:

$f(x)=m^{256}\times m^{54}$. This is because in a multiplication operation, when the two bases are the same, we can add the exponents together and so $m^{256}\times m^{54} = m^{310}$.

So how do we calculate $f(x)=m^{54}$ yet 54 does not fall on the $2^n$ sequence? We do this by engaging in the same multiplication operation involving exponents of the $2^n$sequence until we get a result greater than 54. This will be step three.

*Step 3*

$f(x) = m^{54}$

$f(x)= m\times m = m^2$.

$m^2\times m^2 = m^4$

$m^4\times m^4 = m^8$

$m^8 \times m^8 = m^{16}$

$m^{16}\times m^{16} = m^{32}$

$m^{32}\times m^{32} = m^{64}$

Since 64 is greater than 54, we stop the multiplication operation here.

In step 4, we will have to subtract the penultimate exponent (32) from the exponent that we are calculating (54).

*Step 4* .

Hence, 54-32=22.

Therefore we can rewrite $f(x)=m^{310}$ as:

$f(x)=m^{256}\times m^{32}\times m^{22}$

So how do we calculate $f(x)=m^{22}$ yet 22 does not fall on the $2^n$ sequence? We do this by engaging in the same multiplication operation involving exponents of the $2^n$sequence until we get a result greater than 22. This will be step five.

*Step 5*

$f(x) = m^{22}$

$f(x)= m\times m = m^2$.

$m^2\times m^2 = m^4$

$m^4\times m^4 = m^8$

$m^8 \times m^8 = m^{16}$

$m^{16}\times m^{16} = m^{32}$

Since 32 is greater than 22, we stop the multiplication operation here.

In step 6, we will have to subtract the penultimate exponent (16) from the exponent that we are calculating (22).

*Step 6* .

Hence, 22-16=6.

Therefore we can rewrite $f(x)=m^{310}$ as:

$f(x)=m^{256}\times m^{32}\times m^{16}\times m^{6}$.

Notice that 256, 32 and 16 are all penultimate exponents and they also belong to the $2^n$ sequence. However, 6 is not a penultimate exponent, it is a remainder exponent because it is what remains after we have removed all possible $2^n$ sequence exponents. This author stopped when the exponent of the value that does not belong to the $2^n$ sequence reduced to a single digit value. The assumption is that $m^6$ is easy to calculate no matter how large the value of m is. The final step which is step 7, involves solving for $f(x)=m^{256}\times m^{32}\times m^{16}\times m^{6}$.

The beauty of this method is that once we have calculated the first round of multiplication for $m^{310}$ in table 1, then we already obtained the values for $m^{256}$, $m^{32}$ and $m^{16}$. This means that we do not need to calculate these values again. We can just go back to table 1 in example 4 and pick them up. Therefore $m^{256}$, $m^{32}$ and $m^{16}$ need zero new multiplication operations. However, notice that $m^6$ is not found in table 1 and we will have to calculate it manually. However it is expected that it will be trivial to calculate $m^6$for mathematicians or computers regardless of the size of m. We know that $m^6$ will take 5 multiplication operations to calculate it. From table1, we can see that $m^{256}$, $m^{32}$ and $m^{16}$ require 9 multiplication operations to calculate. That is, table 1 requires 9 multiplication operations to create and once it is created, we can obtain values for $m^{256}$, $m^{32}$ and $m^{16}$without further calculations. Therefore how many multiplication operations does it take to calculate $m^{256}\times m^{32}\times m^{16}\times$m6 if we already have the values for each operand separately? It takes 3 multiplication operations to calculate it. Therefore the total multiplication operations necessary to solve this problem is 9+5+3=17. Therefore instead of multiplying m by itself 309 times, we can just multiply it 17 times to get the answer if we use this new method. We will reduce the number of multiplication steps needed by 309-17=292. This is a reduction in inefficiency of $292\div309\times100=$ 94.5%

**Example 5**

When one is given a scary equation like this: $f(x)=3^{19}$one should not panic. This equation can be sorted out in a few steps if you know that $f(x)=3^{19}$ = $f(x)=3^{16}\times3^{3}$. Always reduce the exponents into $2^n$ sequence exponents and single-digit remainder exponents that do not belong to the $2^n$sequence. The value $3^{16}$ can be calculated in 4 multiplication operations as shown in table 1 in example 4. The value $3^3$ will be calculated in 2 multiplication operations. Therefore the total number of multiplication operations will 4+2=6. The value $3^{16}\times3^3$ itself will take 1 multiplication operation to calculate. Therefore if you know that $m^{19}$ can be divided into $m^{16}$ and $m^3$ then you will know that it takes only 4+2+1=7 multiplication operations to get the answer instead of doing all the 16 multiplication operations that is used in the normal calculations. Knowing how a particular number is divided into its $2^n$sequence components is extremely important and one can also use a table of $2^n$ values to make work easier. For example, with enough practice or with a $2^n$ table nearby, it is not difficult at all to realize that 163 can be broken into its $2^n$ components like this 128,32 and 3. The number 3 is not a $2^n$ component, it is just the remainder. Therefore with enough practice or with a table nearby it is not difficult for one to discover that $m^{163}=m^{128}\times m^{32}\times m^3$. $m^{128}$ requires 7 multiplication operations. $m^{32}$ requires 5 multiplication operations, however we do not have to recalculate $m^{32}$ since we already calculated it when we were calculating $m^{128}$. This means we can just pick the values for $m^{32}$ from our calculations of $m^{128}$. Therefore zero multiplication operation is necessary to calculate $m^{32}$ because it is already calculated. The value $m^3$ requires 2 multiplication operations. Remember that $m^{128}\times m^{32}\times m^3$ itself requires 2 multiplication operation. Therefore the total number of multiplication operations required is 7+2+2=11. So instead of multiplying m by itself 162 times, we can just use the ngazi method which requires only 11 multiplication operations to get the answer.

Next, The author will go ahead and carry out the actual calculations to prove that one need only 7 multiplication operations to calculate $f(x)=m^{19}$.

First of all, we assume that the mathematician who is trying to use this method is proficient in converting $m^{19}$ to its $2^n$ components. Now let us solve an actual problem.

**Example 6**

Solve f(x)= $3^{19}$

*Step 1*

Convert it into its $2^n$ components. This becomes

f(x)= $3^{16}\times3^3$.

*Step 2*

$3^{16}$ requires 4 multiplication operations as shown below

f(x)= m×m = $m^2$.

$m^2\times m^2 = m^4$

$m^4\times m^4 = m^8$

$m^8 \times m^8 = m^{16}$

*step 3*

Replacing m with 3 we get

f(x)= 3×3=9 (First, square 3 to get 9)

9×9=81 (Then square 9 to get 81)

81×81=6,561 (Then square 81 to get 6,561)

6,561×6,561=43,046,721 (Finally square 6,561 to get 43,046,721)

$3^3$ requires 2 multiplication operations as shown below

3×3×3=27

*step 4*

$3^{16}\times3^3$ requires 1 multiplication operation.

Therefore the total number of multiplication operations required to perform this calculation using this method is 4+2+1=7.

*2.2 Using $2^n$ Tables to quickly solve exponential functions*

If one has a $2^n$ table then one can quickly break down any number into its $2^n$ components.

A table will make work easier for any mathematician. Below is a sample $2^n$ table that will be used to calculate another example for demonstration purposes.

*Table 2*

| Number of multiplication operations required | Value of the $2^n$ exponent |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |
| 8 | 256 |
| 9 | 512 |

| Number of multiplication operations required | Value of the $2^n$ exponent |
|---|---|
| 10 | 1,024 |
| 11 | 2,048 |
| 12 | 4,096 |
| 13 | 8,192 |
| 14 | 16,384 |
| 15 | 32,768 |
| 16 | 65,536 |

If you are given f(x)= $m^{243}$, use table 2 to solve the problem.

Let us break up 243 into its $2^n$ components.

First of all we can see from table 2 that 243 falls between 128 and 256. Therefore, we must subtract 128 from 243.

243-128= 115

115 is the remainder but 115 itself can be further broken down into other $2^n$ components. 115 falls between 64 and 128 therefore we can subtract 64 from 115.

115-64=51

51 is the remainder but 51 itself can be further broken down into other $2^n$ components. 51 falls between 32 and 64 therefore we can subtract 32 from 51.

51-32=19

19 is the remainder but it can be further broken down into other $2^n$ components. 19 falls between 16 and 32 therefore we can subtract 16 from 19.

19-16=3.

3 is the remainder and since it is a single digit remainder, we stop our operation here.

Therefore the final equation will look like this

f(x)=$m^{243}$ = $m^{128}\times m^{64}\times m^{32}\times m^{16}\times m^3$

That is how the table can be used quickly to break down any number into its $2^n$ components

*3. Discussion*

If this ngazi method is coded into a computer software then we will create computers that can calculate exponential functions much faster than they currently do. This method is very important because exponential function calculations are done everyday in fields such as engineering, mathematics and computer science. In fact since this method will vastly improve computer efficiency in calculating exponential functions then this method will contribute greatly towards finding the solution to the discrete logarithm problem through the use of discrete exponentiation. Let me explain what the discrete logarithm problem is. Unsolved problems website explains this problem very well.

"Suppose that we have an equation

$y=g^x \bmod p$.

Where all four variables are integers, and g and p are very large prime numbers, say, greater than 100 digits. Given g, p and x, it takes a modern-day computer only a fraction of a second to calculate y. The Discrete Logarithmic Problem (DLP), is given g, p and y, can we find x, even if we have a million computers and

a million years at our disposal?[6] The discrete logarithm is the inverse of the discrete exponentiation in a finite cyclic group.[7] This concept is important in

internet security. A recent finding is that Diffie-Hellman security is not as watertight as was initially believed.[8] Even though digital computers are efficient in computing, they still have challenges in solving two problems, factoring integers and finding discrete logarithms.[9] In common culture, In the famous maths drama television programme called Numbers, discrete logarithm were mentioned in the season 2 episode "In plain sight".[10]

This author's reply to the discrete logarithm problem is that it is possible to find x in finite time using ngazi method and rearranging the equation as will be shown later. This author believes that we can solve this problem even if we are given only g,p and y. This is so because this new method reduces tremendously the number of multiplication operations that are needed to solve this problem. The following procedure should be followed in order to make finding solutions to discrete exponentiation and by extension discrete logarithm problems easier.

Since we don't have x in $g^x$, we have no choice but to use the trial and error method. Some trial and error methods are more efficient than others in predicting what the value of x would be. However, whatever method is used, one must be prepared to carry out several rounds of calculations until one obtains the value of x. In the simplest example, suppose we assume that x is 1 then

$y=g^1 \bmod p$. We then calculate $g^1 \div p$. We then check whether the remainder of this operation is equal to y. If it is, then we have found the solution to x. If it is not, then we must proceed with our calculations. We can set x to be 2. $y=g^2 \bmod p$. We can then calculate $g^2 \div p$ and then we can check whether the remainder of this operation is y. If it is not then we can set x to be 3 and then proceed to calculate $g^3 \bmod p$. Of course we can go on and on setting x to any value and each time using the ngazi method to calculate $g^x$. We can repeat this operation until we find x. Of course since the ngazi method is vastly more efficient in calculating exponential functions, then it will take a finite amount of time to find x in the equation $y=g^x \bmod p$ as long as x is not an infinite value. Of course there are more efficient methods of guessing x instead of just following the number line 1,2,3,4,5, we can use more efficient methods of guessing x and this will save time. It will definitely take a finite period of time to get the answer to that equation.

One last thing that has to be said about the discrete logarithm problem is that it is a problem now better solved via the route of discrete exponentiation. The method of discrete exponentiation that I have explained above can be improved and made more efficient. The problem with the method above is that we require the computer to calculate $g^x \bmod p$ and then find out whether the remainder of this operation has the same value as y. However, this method can be extremely inefficient and time-consuming if the remainder itself is a big number. This big number will take up a lot of precious computer memory and will also consume a lot of computing power and time during its calculation. This author has found a novel and more efficient method of calculating the discrete exponential function in such a way that the remainder does not need to be calculated in full.

Given $y=g^x \bmod p$ and we know the values for g,p and y, is it possible to find x in finite time?

To avoid calculating and manipulating the remainder, we shall have to rearrange the equation above.

The equation $g^x \bmod p$ has a remainder value of y (where $y>0$). Therefore this means that

$g^x$ is not divisible by p since $g^x \div p$ produces a remainder y. $g^x$ can be understood to be the sum of a number t which is divisible by p and a remainder y which is not divisible by p. Therefore we can subtract the remainder y from $g^x$ to obtain the number which is divisible by p.

$y=g^x \bmod p$

$g^x-y=t$

Now, the value $g^x$-y is divisible by p because it does not have the remainder y. Since $g^x$-y is divisible by p, the next step would be divide $g^x$-y by p.

$(g^x\text{-}y)\div p$

So we can confidently say that the value $(g^x\text{-}y)\div p$ is definitely an integer and that means it does not have a remainder or decimal points. The problem that remains however, is that we do not know what the value of x is. This will force us to do trial and error to get x. Therefore we can feed in arbitrary values for x, we can either follow a systematic approach, for example, 1,2,3,4,5 for values of x or any other suitable approach can be used. This author prefers the systematic approach where the values of x are input starting from 1,2,3,4,5 up to x without skipping any number until x is found. So how will we know when we have found x?

Let us suppose we set x to be 1 as shown $(g^1\text{-}y)\div p$. If the result of this calculation has a remainder or a decimal then 1 is not the value for x and we must set x to be 2.

If the computer calculates the equation $(g^2\text{-}y)\div p$ and then gives an output with decimals then it is clear that this equation has a remainder and therefore we must set x to be 3 and this process goes on and on as long as this equation produces decimal points. It is prudent to program the computer to stop the calculation at the $1^{st}$ decimal point to save computer power and time. We no longer need to calculate the whole remainder because we no longer need to compare the remainder with y. This act of comparing, after every calculation, whether the remainder is equal to y would have consumed immense computer power and time but because we no longer need to compare the remainder with y then computer resources are conserved. So how will we know when we have found x? It is very simple. We will know that we have found x when the calculation $(g^x\text{-}y)\div p$ does not give us any decimals. Therefore when we feed in a certain value of x and fail to get a remainder then that value of x is the one we were looking for. Therefore the computer can be programmed to stop the calculation only when a given value of x in the equation $(g^1\text{-}y)\div p=h$ produces an integer value for h. In conclusion, if h is an integer value then we have found x, however if it a decimal or a fraction then we haven't found x. Just to summarize, this new approach is more efficient than the previous one because

1. The entire remainder does not need to be calculated. In fact when the first decimal has been calculated then the operation is stopped and the computer moves on to the next value of x. For computers that produce results of division operations in fraction form, the computer can be programmed to terminate calculation after the first digit of the remainder has been calculated. The computer does not need to calculate all the digits of the remainder. Therefore there will be no computer power and time spent calculating this large remainder.

2. The second advantage is that because the entire remainder is not calculated, this means that it does not need to be stored or retrieved at a later date for comparison with y. Because the remainder will not be stored then there will be no memory used for this process.

3. The third advantage is that after every round of calculation, the remainder was supposed to be compared with y to find out whether they have the same value, this would have been an inefficient process. However, we no longer need to compare the remainder with y and therefore we save on computer power and time that would have been used to compare these two large numbers.

This new method can also be used in the creation of new and more efficient computer math libraries that calculate exponential functions. It is important that computer libraries that deal with mathematical calculations should be as fast as possible.

Indeed, this new method can be used in any area that uses exponential functions in their calculations.

*4. Conclusion*

The conclusion is that this paper presents an entirely new way of calculating exponential functions. This method uses the $2^n$ sequence at its foundation. This method is vastly more time saving and energy saving because one performs very few multiplication operations once an exponent has been broken down into its $2^n$ components.

*5. References*

1. Schraudolph NN. A Fast, Compact Approximation of the Exponential Function. Technical Report IDSIA-07-98 to appear in Neural Computation 11(3). March 10,1998.

2. Eves H. An Introduction to the history of Mathematics. Sixth Edition. The Saunders Series.

ISBN 0-03-029558-0. pg 308.

3. Lyng MJ, Meconi LJ, Zwick EJ. Applied Technical Mathematics with Calculus.

Wm.C.Brown Publishers. 1992, ISBN 0-697-05970-7. pg 414.

4. Backhouse JK, Houldsworth SPT, Cooper BED, Horril PJF. Longman Group UK Limited.

ISBN 0-582-35387-4. pg 19.

5. Iribemwangi PI. Kiswahili Phonology and Pronounciation Guidelines. Rosetta Stone Language Workshop. Dubai, UAE. January 25th 2010.

https://www.researchgate.net/publication/324730793_Kiswahili_Phonology_and_Pronounciation_Guidelines

6. Roberts TS. Unsolved Problems: In Number Theory, Logic, and Crystography. timro21@gmail.com,

http://unsolvedproblems.org/index.htm

7. Studholme C. The discrete Log Problem. June 21, 2002

http://www.cs.toronto.edu/~cvs/dlog/

8. Adrian D, Bhargavan K, Durumeric Z, Gaudry P, Green M, Halderman JA et al. Imperfect Forward Secrecy: How Deffie-Hellman Fails in Practice. 22nd ACM Conference on Computer and Communications Security. CSS 2015, December 2015

9. Shor PW. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. Arxiv. 25 Jan 1996.

doi.10.1137/s0097539795293172.

10. Weisstein EW. "Discrete Logarithm." From MathWorld-A Wolfram Web Resource.

http://mathworld.wolfram.com/DiscreteLogarithm.html