Computing cube root of a positive number

YumnamKirani Singh
C-DAC, IIPC Building, NIT Campus, Silchar, Assam,
Assam, India-781039.
yumnam.singh@cdac.in

Abstract:***Proposed here is a new algorithm to compute the cube root of large positive integer. The algorithm is based on the implementation of long division method also known as manual method we usually use to find the square root of a number. To implement the long division method, the given number is first represented in a radix-10 representa and then Bino's Model of Multiplication is used to systematically implement the long division method. A representa is a special array to represent a number in the form of an array so as to enable us to treat the representas in the same way as we treat numbers. This simplifies the difficulty of dealing large numbers in a computer. Also, at the same time it simplifies the implementation of long division method to find the cube root of positive number, ranging from single digit number to arbitrarily large positive number such as RSA challenge numbers. The algorithm can be used to compute cube root of a non-perfect cube number up to desired precision and each computed digit of cube root gives the best precision. Cube root of 2, 5, 10 up to 30 digits and integer parts of cube roots of first few and last few RSA challenge numbers are also provided in the experimental result to show that the algorithm works perfectly to compute the cube root of any positive integer, however small or large it may be.***

Keywords:Bino's Model of Multiplication, Convolution, Cube of a large number, Large number manipulation, Long division method, RSA challenge numbers, Representa, Cube root computation.

1. Introduction

Submit Dealing in large number in a personal computer is problematic. This is because dealing of large numbers is to be done in special ways as such large number cannot be represented as a value in a variable of a programming language and the programs or algorithm written for small numbers cannot be directly applied to such large numbers. This means that algorithm for computing square root or cube root of small number like 1234565 cannot be used to compute the square root or cube root of large number such as 30 or more digit number like RSA (Rivest, Shamir and Adleman) challenge number. This problem of handling large number can be solved by using the notion of representa [7]. Representa is a special way to represent number in the form of an array, which can be treated in the same way as we treat numbers. Representa simplifies the development of algorithms for arithmetic operations for large numbers. Finding square root or cube root of a number in a computer or calculator is done in totally different ways as we do to compute the square or cube root manually. The manual method of computing square root gives the best possible result for any digit computed as compared to numerical approximation methods. However, so far no one has implemented successfully the manual method to compute the square root in a computer and hence the method is regarded as dead end method of computing square root in computer science community.
The most popular method for finding square root electronically is the iterative method based on Babylonian algorithm or Newton-Rapson method or its variants [1, 2, 3, 4]. But these algorithms are suitable for small numbers which are within the byte size limitations. These algorithms iteratively approximate the root until a specified precision is achieved and require division of the number by newly approximated root at each iteration. Division of two large numbers is computationally difficult task which requires special algorithms. So, the direct implementation of iterative approximation method for finding square root of a large number is not possible. A new algorithm based on long division method to compute the square root of an arbitrarily large number has been successfully developed in [8]. This paper extends the square root method to find cube root using long division method. This method is similar to the method computing cube root suggested by Aryabhatta[5] in theory but the way of implementation is totally different. In [6], another method to compute the cube root is explained in terms of expansion of a cube of a polynomial. Some online sites which shows or tires to explains the manual method of computing cube roots are listed in [9 --14]. This paper implements the long division method of computing cube root of an arbitrarily large number using the concepts of representa and Bino's Model of Multiplication.
The rest of the paper is organized into four sections. Section-II describes the Cube of a large number using Bino's model of Multiplication (BMM). This can be used to test the correctness of the result of cube root and is used to explain why and how Long Division Method (LDM) works to find the cube root of large number. Section III describes the long division method of computing cube root of large integer. An algorithm to compute the cube root based on representa is presented. The experimental result is given in section IV and conclusion in section V.

2. Cubing a large number using BMM

Set Bino's model of multiplication is generalized multiplication model for multiplication of numbers, polynomials and arrays. To multiply two numbers, numbers are represented in the form of special arrays called representa depending on a specific radix or base, which is a power of 10. In a representa of radix-10, each element of the representa must be a reminder of 10. In a representa of radix-100, each element is a reminder of 100. In this way, in a representa of radix-1000, the elements are reminder of 1000. Representing a number in higher radix, saves, significant amount of memory and processing time. However, in this paper, we will be dealing with representa of radix-10, for easy understanding of the explanation. Once the numbers to be multiplied are represented in representa of same radix (10, here), multiplication terms are computed. The actual result of multiplication can be easily obtained from the multiplication terms adjusting the carries in the multiplication terms. More on representa arithmetic and Bino's model of multiplication can be found in [13].

According to the Bino's model of multiplication, when two representas each of length m and n are multiplied the number of multiplication terms is given by m+n-1. The number of multiplication terms of square two m-digit number is 2m-1. The cube of an m-digit number can be considered as multiplication of the square of the m-digit number and the number itself. So, the number of multiplication terms of cube of m-digit number is (2m-1)+m-1=3m-2. When m=1, the number of multiplication terms is 3*1-2=1, i.e., the number of multiplication terms for cubing a single digit number is 1. When m=2, 3m-2=4, i.e., the number of multiplication terms when a 2-digit number is cubed is 4. Similarly, when m=3, 3m-2=7, i.e., the number of multiplication terms when a 3-digit number is cubed is 7 and so on. The number of multiplication terms also indicates the minimum number of digits when a number is cubed.

Let us consider some examples of finding cube using Bino's Model of Multiplication.

A. *Cube of 2-digit number*

Suppose b1,b2 is a two digit number for which we want to compute cube. We first represent in a radix-10 represented as B1=[b1,b2]. Then, cube of the 2-digit number will consist of three multiplication terms. Let us represent the array representing a multiplication term by T, and each element in the array by Ti, (denoting i-th element).

B1^3=T

Where T=[T1,T2,T3,T4] and the respective terms are

$$T_1 = b_1^3, T_2 = 3b_1^2 b_2, T_3 = 3b_1 b_2^2, T_4 = b_2^3$$

The result of cube of 2-digit number will be given by

$$(b_1 b_2)^3 = 1000 \times T_1 + 100 \times T_2 + 10 \times T_3 + T_4$$

Depending on the values of the two digits, the square of a 2-digit number will have minimum of 3 digits and maximum of 4- digits.

Example-1: Cube of 23,

Multiplication Terms are

T

$T_1 = b_1^3 = 2^3 = 8$

$T_2 = 3b_1^2 b_2 = 3 \times 4 \times 3 = 36$

$T_3 = 3b_1 b_2^2 = 3 \times 2 \times 3^2 = 54$

$T_4 = b_2^3 = 3^3 = 27$

So, $23^3 = 1000 \times 8 + 100 \times 36 + 10 \times 54 + 27 = 2167$, which is a 5- digit number.

Example-2: Square of 67

Multiplication Terms are

$\mathrm{T}_1 = \mathrm{b}_1^3 = 6^3 = 216$

$\mathrm{T}_2 = 3\mathrm{b}_1^2 \mathrm{b}_2 = 3 \times 36 \times 7 = 756$

$\mathrm{T}_3 = 3\mathrm{b}_1 \mathrm{b}_2^2 = 3 \times 6 \times 7^2 = 882$

$\mathrm{T}_4 = \mathrm{b}_2^3 = 7^3 = 343$

So,

$$67^3 = 1000 \times 216 + 100 \times 756 + 10 \times 882 + 343$$
$= 300763$, which is a 6- digit number.

From these examples, it can be seen that any 4, 5 or 6-digit number can be thought of as$b_1^3,\ 3b_1^2 b_2,\ 3b_1 b_2^2,\ b_2^3$.

B. *Cube of a 3-digit number*

Let $B_1 = [b_1, b_2 b_3]$ be a radix-10 representa corresponding to a 3- digit number$b_1 b_2 b_3$. Then, the cube of B_1 will have 7 multiplication terms, which are as follows.

$$B_1^3 = [T_1, T_2, T_3, T_4, T_5, T_6, T_7]$$
$$Where T_1 = b_1^3, T_2 = 3b_1^2 b_2, T_3 = 3b_1^2 b_3 + 3b_1 b_2^2,$$
$$T_4 = 6b_1 b_2 b_3 + b_2^3, T_5 = 3b_1 b_3^2 + 3b_2^2 b_3,$$
$$T_6 = 3b_2 b_3^2, T_7 = b_3^3$$

The actual result of the squaring the 3-digit number is given by

$$(b_1 b_2 b_3)^3 = 1000000 \times T_1 + 100000 \times T_2 + 10000 \times T_3 + 1000 \times T_4 + 100 \times T_5 + 10 \times T_6 + T_7$$

As there are seven multiplication terms, the number of digits in resulting square will be 7, 8 or 9-digit number depending on the values of first two multiplication terms. So, any 7, 8 or 9- digit number can be thought of as if obtained from the above seven multiplication terms cubing a 3-digit number.

C. *Cube of an m-digit number:*

Let $B_1 = [b_1, b_2, b_3, \ldots, b_m]$ be an m-digit number represented in a representa of base 10. Then, there will be 3m-2 multiplication terms.

$T_1 = b_1^3,$

$T_2 = 3b_1^2 b_2,$

$T_3 = 3b_1^2 b_3 + 3b_1 b_2^2$

$T_4 = 3b_1^2 b_4 + 6b_1 b_2 b_3 + b_2^3$

$T_5 = 3b_1^2 b_5 + 6b_1 b_2 b_4 + 3b_1 b_3^2 + 3b_2^2 b_3$

$T_6 = 3b_1^2 b_6 + 6b_1 b_2 b_5 + 6b_1 b_3 b_4 + 3b_2^2 b_4 + 3b_2 b_3^2$

$T_7 = 3b_1^2 b_7 + 6b_1 b_2 b_6 + 6b_1 b_3 b_5 + 3b_1 b_4^2 + 3b_2^2 b_5 + 6b_2 b_3 b_4 + b_3^3$

And so on.

Giving direct formulation of cube of multi-digit number is difficult without using over-bracket summation. However, describing over-bracket summation will make content longer and deviate from the main point of the paper. Interested readers are referred to [7].

Another approach to compute cube of multi-digit number is to multiply the square of the multi-digit number with number itself using Bino's model of multiplication. Finding the square of an m-digit number can be done easily using BMM. Following is the description of an algorithm to compute the square of an m-digit number given in [8].

Let B be radix-10 representa of an m-digit number. That is,

```
m=Length of the representa of base length 10.
T= an array of length L to store L multiplication terms
L=2*m-1;
For i=1 to m-1
lowerhalf=0;
upperhalf=0;
if (i%2)
        for j=1 to (i-1)/2
                lowerhalf=lowerhalf + B(j)*B(i+1-j);
                upperhalf = upperhalf + B(m+1-j)*B(m-1+j)
        end
T(i)=lowerhalf+B((i+1)/2)*B((i+1)/2);
T(L-i)=upperhalf +B((i+1)/2)*B((i+1)/2);
Else
        For j=1 to (i/2)
                lowerhalf=lowerhalf+B(j)*B(i+1-j);
                upperhalf=upperhalf + B(m+1-j)*B(m-1+j)
        end
```

T(i)=lowerhalf;
T(L-i)=upperhalf;
End

The actual result of squaring is obtained from multiplication terms T by successively adding the carry from the last term to successive terms on the left till the first term.
Once we find the square of an m-digit number using the above algorithm, the cube of number can be easily obtained by multiplying the square with the m-digit number itself. Bino's model of multiplication shows that convolution operation is nothing but a multiplication operation. So, if T is the square of an m-digit number and B is a representa of base 10 of m-digit number, then the cube of m-digit number Q can be written as

Q=T⊗B, where ⊗ denotes the convolution operation.
The elements of Q correspond to the multiplication terms of cube the m-digit number represented by representa B.

3. Computing cube root using LDM

The long division method also known as manual method is the method, we generally use, to compute the square root of a number. This method has not been implemented as an algorithm to compute square or cube root of a number. Instead other approximation or estimation method such as Newton's or Halley's method has been used. The main reason why long division method has not been implemented as an algorithm is because the underlying theory why and how this method works has not been explained properly. Some tries to explain it using squaring of polynomials. But the polynomials representation and number representation are different and so, the explanation of the process is not clearly understandable for implementation purpose. BMM can be used to explain how LDM works for finding the cube root of a number.
Before explaining LDM, let us examine the following multiplication terms of cubing a 1, 2, and 3-digit numbers in their representa forms.

$[\,b_1]^3 = [b_1^3]$,

(only one multiplication term, 3*1-2=1)

$[\,b_1, b_2]^3 = [b_1^3, 3b_1^2 b_2, 3b_1 b_2^2, b_2^3]$,

(Four multiplication terms, 3*2-2= 4)

$$[b_1, b_2, b_3]^3 = [b_1^3, 3b_1^2 b_2, 3b_1^2 b_3 + 3b_1 b_2^2, 6b_1 b_2 b_3 + b_2^3, 3b_1 b_3^2 + 3b_2^2 b_3, 3b_2 b_3^2, b_3^3\,]$$

(Seven multiplication terms, 3*3-2=7)

f we observe carefully, we see that the cube of digits occur after every 3 multiplication terms. That is, 1st, 4th, 7th etc. multiplication terms contain b_1^3, b_2^3, b_3^3 etc. Finding cube root from the multiplication terms of a cube of a number requires a systematic process to eliminate the cube of the digits in their order of occurrence i.e. from left to right in long division method.

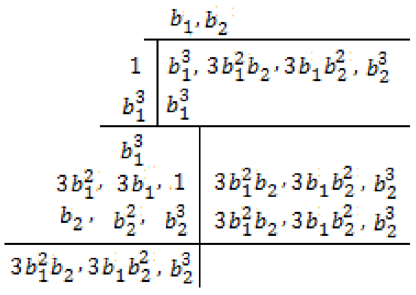
Let us examine the some cases of finding square roots of some numbers.

Finding the square root of 1, 2 or 3-digit number is trivial. It will be any digit from 0 to 9.

A. *Cube root of 4, 5 or 6-digit number*

Cube root of 4, 5 or 6-digit number will be a 2 digit number. We know that cubing a 2-digit number results in four multiplication terms. So, the minimum number of digits when a 2-digit number is cubed is 4. The maximum number of digits when a 2-digit number is 6. This is because, the number of multiplication terms when a 3-digit number is cubed is 7. So, the problem of finding the cube root of a 4,5 or 6-digit number is finding the two digits from the four multiplication terms.

Let us assume that the 4, 5 or 6-digit number is represented by four multiplication terms obtained when a 2-digit number is cubed. The first term is b_1^3, which can be obtained by multiplying b_1 with itself three times, as shown in the upper left part of Figure-1. Then, b_1^3 is subtracted from the multiplication terms and the first multiplication term is eliminated. So, b_1 becomes the first digit of the cube root, which is written at the top. After eliminating the first multiplication term, we are left with three multiplication terms, $3b_1^2 b_2, 3b_1 b_2^2, b_2^3$. They are brought down in the next line. Then 3 times the square of first digit of the root, 3 times the first of the root and 1 (that is $3a_1^2, 3a_1, 1$) is multiplied element-wise by b_2, b_2^2, b_2^3, where $b_2$is the next probable digit of the cube root. When $(3a_1^2, 3a_1, 1)$ and (b_2, b_2^2, b_2^3) are multiplied we get the remaining three multiplication terms. Thus, we get b_1, b_2as the cube root from the four multiplication terms obtained by cubing of 2-digit number.



b_1, b_2
1
$b_1^3, 3b_1^2 b_2, 3b_1 b_2^2, b_2^3$
b_1^3
b_1^3
b_1^3
$3b_1^2, 3b_1, 1$
$3b_1^2 b_2, 3b_1 b_2^2, b_2^3$
b_2, b_2^2, b_2^3
$3b_1^2 b_2, 3b_1 b_2^2, b_2^3$
$3b_1^2 b_2, 3b_1 b_2^2, b_2^3$

Figure-1: Process for finding cube root of a 4, 5 or 6-digit number.

B. *Finding cube root of 7,8 or 9- digit number*

To find the cube root of 7, 8 or 9-digit number, we can consider the number as the seven multiplication terms when a 3-digit number is cubed. The process remains the same as described in section III.A. First eliminate the first cube term to get the first digit of the cube root. Bring down the next three multiplication terms separated by commas, the last of which contains the cube of the next digit of the root. Find the partial divisor multiplying $(3a_1^2, 3a_1, 1)$ and (b_2, b_2^2, b_2^3) element wise i.e to get $3b_1^2 b_2, 3b_1 b_2^2, b_2^3$, Subtract the result from the three multiplication terms brought down earlier as sown in Figure-2. The difference $3b_1^2 b_3, 6b_1 b_2 b_3$is brought down and b_2 is written at top as the next digit of the cube root. The remaining three multiplication terms brought down and appended to the difference. To find the next digit of the cube root we form the partial divisor by multiplying $[3(b_1, b_2)^2, 3(b_1, b_2), 1]$and $[b_{3,} b_3^2, b_3^3]$. Note here that multiplying $3(\,b_1, b_2)^2$ by b_3 is equivalent to multiplication of $3(b_1^2, 2b_1 b_2, b_2^2)$by b_3, which results in $3b_1^2 b_3, 6b_1 b_2, 3b_2^2 b_3$. Similarly multiplication of $3(b_1, b_2)$by

b_3^2 results in $3b_1b_3^2, 3b_2b_3^2$. The terms are subtracted from the corresponding terms brought down earlier which results in no-remainder. The third digit of the cube root i.e., $b_3$is written at top as the last digit of the root. The described process is shown in figure-2.

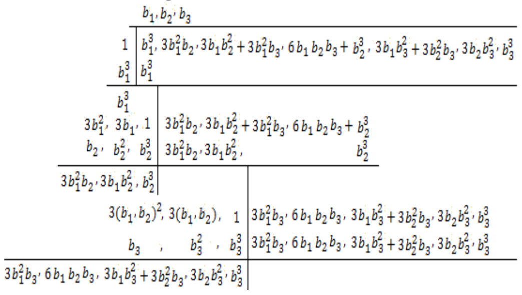

b_1, b_2, b_3
1
$b_1^3, 3b_1^2b_2, 3b_1b_2^2+3b_1^2b_3, 6b_1b_2b_3+b_2^3, 3b_1b_3^2+3b_2^2b_3, 3b_2b_3^2, b_3^3$
b_1^3
b_1^3
b_1^3
$3b_1^2, 3b_1, 1$
b_2, b_2^2, b_2^3
$3b_1^2b_2, 3b_1b_2^2+3b_1^2b_3, 6b_1b_2b_3+b_2^3$
$3b_1^2b_2, 3b_1b_2^2, \quad b_2^3$
$3b_1^2b_2, 3b_1b_2^2, b_2^3$
$3(b_1,b_2)^2, 3(b_1,b_2), 1$
b_3, b_3^2, b_3^3
$3b_1^2b_3, 6b_1b_2b_3, 3b_1b_3^2+3b_2^2b_3, 3b_2b_3^2, b_3^3$
$3b_1^2b_3, 6b_1b_2b_3, 3b_1b_3^2+3b_2^2b_3, 3b_2b_3^2, b_3^3$
$3b_1^2b_3, 6b_1b_2b_3, 3b_1b_3^2+3b_2^2b_3, 3b_2b_3^2, b_3^3$

Figure-2: Process for finding cube root of a 7, 8 or 9-digit number.

C. Algorithm for computing cube root

We have seen that while computing cube root of 1,2 or 3-digit number, 4,5 or 6-digit number and 7,8 or 9-digit number from their respective multiplication terms, except the first term, we brought down three consecutive multiplication terms to find the next digit of the cube root. So, we will mark the digits of multi-digit number in group of three digits starting from the right end before computing cube root using long division method. Grouping the digits in a group of 3 can be done by representing the given number as radix-1000 representa. So, the algorithm is as follows.

```
Let  X= radix-1000 representa representing the multi-digit
number whose cube root is to be computed.
R=radix-10, representa representing result, i.e., cube root.
C=radix-10, representa to act as partial divisor to find the
next digit of the root.
D=radix-10 representa to act as partial dividend.
Set D=X(1)
#-Finding the first digit of the cube root
For i=1 to 10
   if (i*i*i) >D
          R(1)=i-1;
   End if
End for
C=[3*R*R];
While (last element of X not processed)
          Compute D=Subtract( D, C);
          Append the next element of X to D.
          C=[3*R*R, 3*R, 1];
          For I=1 to 10
                  P=[I, I*I, I*I*I];
                  #-Multiply C and P element-wise
                  C=Multiply(C,P);
                  if C > D
                          Append I-1 to R
                          break;
                  End if
End for
End while
```

In the algorithm, the first element of R, i.e., the first digit of cube root is obtained separately using a for-loop which runs 1 to 10.The remaining digits of the cube root are computed inside the while-loop. To compute next digit of cube root, the for-loop which runs 1 to 10 is used. So, to find the cube root of 3N digit number, the operations inside the for-loop is used 10N times in the worst case, which happens when the digits of cube root are all 9's. That is, the algorithm takes linear time, O(N).The algorithm can be easily extended to compute the cube root of floating point number or a non-perfect cube number up-to desired number of precision after decimal point. For computing cube roots of floating point numbers the digits in the integer part are marked in a group of 3 from right to left from the decimal point and the digits in the fractional part are marked in a group of 3 from left to right starting from the decimal point. For cube root of non-perfect cube number up to a desired precision, after the last digit of the given number is used, three zeros are appended to the non-zero remainder to form the next partial dividend for the computation of next digit of the cube root. The process continues till the desired length of cube root after decimal point is obtained.

4. Experimental Result

Computing cube root manually has been tried by many mathematicians including Aryabhatta. But the approaches they followed are difficult to understand and cannot be conveniently converted into a computer algorithm. Consequently, iterative approximation methods have been used to compute square roots and cube roots. But the approximation methods have problems in finding the square roots or cube roots of large number or to find cube root of a non-perfect cube number to a desired precision. The proposed method of finding cube root of a positive number can be used to compute cube root of a number manually as well and computationally and the result obtained is the best result. The proposed algorithm is linear time algorithm and has been applied to find the cube root of single digit numbers 2 and 5 and a double digit number 10 to generate cube root of these small numbers up to 30 digits. The same algorithm has been applied to find the cube root of a perfect cube number having 44 digits. The cube root is a 15 digit number, which when cubed get the same number. Also, the algorithm has been applied to find cube roots of some of the RSA challenge numbers (RSA-100, RSA-130, RSA-200, RSA-240, RSA-500 and RSA-2048). Only the cube root is shown for these RSA-challenge numbers as the original RSA challenge numbers are quite lengthy and can be easily obtained from the internet. The experimental results are shown below.

1. 2.0
   Cube root:1.259921049894873164767210607 27
2. 5.0
   Cube root: 1.912931182772389101199116839 54
3. 10.0
   Cube root: 2.154434690031883721759293566 51
4. 83546779665562342421950171719070872029440703
   Cube root: 437162839514687
5. RSA-100
   Cube root:
   115043588465166611052453297 4697442
6. RSA-130
   Cube root:
   2180336704012072496674149365688606022854061
7. RSA-200
   Cube root:
   303651066603140363846612365833841917629022073
   8388049847839865563 50
8. RSA-240
   Cube root:

49949330907769107012091952756571724638721703869724829082950155417925671261854 5

9. RSA-500
   Cube root:
   266708746156832291959937828840942579395522786453390618174092022297018187081433615265030371909302503641598652891528601872082090354244567354955068355092098424315352921 08

10. RSA-2048
    Cube root:
    293163573842459603088498719693125067331655209341030427155133342825706703197029117256512284576875150002893373800771754015461523203279536765914662709813307811863363635808983475806412154612602400850152200432 94

5. Conclusion

A generalized algorithm for computing the cube root of a large number is proposed. Also, a new algorithm for computing the cube of a positive integer is proposed. The algorithm for computing cube root is based on the long division method, implemented using representa and Bino's model of multiplication. The general notion that long division method is manual method for computing square root and cube root is proved incorrect. The proposed algorithm is simple and fast. It can used to find the cube root of a single digit number or to an arbitrarily large number such as RSA-challenge number. The long division method can be extended to compute the nth root of any positive real number giving the best precision.

References

[1] David Fowler and Eleanor Robson, "Square Root Approximations in OldBabylonian Mathematics," Historia Mathematica, 25 (1998), 366-378.

[2] Liang-Kai W, Schulte MJ. Decimal Floating-Point Square Root Using Newton-Raphson Iteration. 16th IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP). 2005: 309-315.

[3] Kosheleva O. Babylonian Method of Computing The Square Root: Justifications Based on Fuzzy Techniques and on Computational Complexity. Annual Meeting of the NorthAmerican Fuzzy Information Processing Society (NAFIPS). 2009: 1-6.

[4] Thomas J. Osler, "Extending the Babylonian algorithm", Mathematics and Computer Education, Vol. 33, No. 2, 1999, pp. 120-128.

[5] Abhishek Parakh, "Aryabhatta's Root Extraction Methods", Indian Journal of History of Science, Vol. 42, No. 2, 2007, pp. 149-161.

[6] Brian J. Shelburne, " Another method for extracting cube roots", http://www4.wittenberg.edu/academics/mathcomp/bjsdir/ExtractingCubeRoots.pdf.

[7] YumnamKiraniSingh , "On Some Generalized Transforms for Signal Decomposition and Reconstruction " Ph.D. dissertation, CVPR Unit, ISI, Kolktata. 2006.

[8] YumnamKirani Singh, "Computing Square Root of a Large Positive Integer", Submitted to NCC-2016.

[9] http://www.had2know.com/academics/compute-cube-root-by-hand.html

[10] http://www.careerbless.com/maths/speedmaths/cuberoot1.php

[11] http://www.mathpath.org/Algor/cuberoot/algor.cube.root.why.htm

[12] http://www.ijird.com/index.php/ijird/article/viewFile/35915/29097

[13] http://www.decodedscience.org/potential-new-lgorithm-to-calculate-the-cube-root-of-a-number/10336/2 (Newton's and Halley's methods)

[14] https://xlinux.nist.gov/dads//HTML/cubeRoot.html

Authors Profile

Yumnam Kirani Singh, has completed Master's Degree in Electronics Science from Guwahati University in 1997 and got Ph. D. degree from Indian Statistical Institute, Kolkata in 2006. Served as a lecturer in Electronics in Shri Shankaracharaya College of Engineering & Technology from Jan, 2005 to May, 2006. Joined CDAC Kolkata in May 2006 and worked there before coming to CDAC Silchar, in March 2014. Developed Bino's Model of Multiplication, ISITRA, YKSK Transforms and several other image binarization and edge detection techniques.Interested to work in the application and research areas of Signal Processing, Image Processing, Pattern recognition and Information Security. Published several papers in national and international journals and conference.