

## Another Method for Extracting Cube Roots

Brian J. Shelburne  
Dept of Math and Computer Science  
Wittenberg University

Probably the easiest way (?) to compute the cube root of  $a$  is to use the iteration formula

$x_{n+1} = \frac{2x_n^3 + a}{3x_n^2}$ . However, while checking out an article in the December 1958 issue of the

*Communications of the ACM* I ran across a paper that demonstrated another method for extracting cube roots. The method was useful (?) since it was done using fixed point arithmetic, it required no division and could be modified so only minimal multiplication was needed (both “expensive” operations for computers in the 1950’s). It’s similar to the method of extracting a square root by hand.

## The Theory

Let  $X$  be the number whose cube root we want

Represent  $\sqrt[3]{X}$  by the sum  $\sum_{i=1}^{\infty} d_i 10^{n-i}$  where  $d_i$  is a digit between 0 and 9 and  $n$  is the number of digits above the decimal point. Letting  $a_i = d_i 10^{n-i}$  we can write  $X = (a_1 + a_2 + \dots + a_n + \dots)^3$ .

For positive integer  $k$ ,  $(a_1 + a_2 + \dots + a_k)^3$  approximates  $X$  and so  $(a_1 + a_2 + \dots + a_k)$  approximates  $\sqrt[3]{X}$ .

We will show how to compute each term  $a_k$  or more exactly each digit  $d_k$  in an iterative fashion such that at each stage the error  $X - (a_1 + a_2 + \dots + a_{k-1} + a_k)^3$  decreases.

This is done by expanding  $(a_1 + a_2 + \dots + a_{k-1} + a_k)^3$  in terms of  $(a_1 + a_2 + \dots + a_{k-1})$  and  $a_k$ ; that is

$$(a_1 + a_2 + \dots + a_{k-1} + a_k)^3 = (a_1 + a_2 + \dots + a_{k-1})^3 + \left[ 3 \cdot (a_1 + a_2 + \dots + a_{k-1})^2 \cdot a_k + 3 \cdot (a_1 + a_2 + \dots + a_{k-1}) \cdot a_k^2 + a_k^3 \right]$$

Then if  $X_{k-1} = X - (a_1 + a_2 + \dots + a_{k-1})^3$  is the error at the  $k-1$ st stage we choose the largest  $a_k$  so that  $X - (a_1 + a_2 + \dots + a_{k-1})^3 - (3 \cdot (a_1 + a_2 + \dots + a_{k-1})^2 \cdot a_k + 3 \cdot (a_1 + a_2 + \dots + a_{k-1}) \cdot a_k^2 + a_k^3)$  is non-negative or equivalently  $X_k = X_{k-1} - (3 \cdot (a_1 + a_2 + \dots + a_{k-1})^2 \cdot a_k + 3 \cdot (a_1 + a_2 + \dots + a_{k-1}) \cdot a_k^2 + a_k^3)$  is non-negative.

**An Example** - Find  $\sqrt[3]{79201}$

**Stage 1:** Begin by finding the *largest* value of  $a_1$  such that  $a_1^3$  is less than or equal to X, or equivalently finding the largest value of  $a_1$  such that the *error*  $X_1 = X - a_1^3$  is non-negative.

We partition the digits into groups of three (i.e. **79** 201.000) and start with the left-most group. In this case the largest integer (digit) whose cube is less than or equal to 79 is 4.

$$\begin{array}{r} \phantom{4} \\ +----- \\ | \mathbf{79} \ 201 \ 000 \\ \underline{4^3} \quad -64 \\ \phantom{4^3} \quad - \\ \phantom{4^3} \quad 15 \ 201 \quad = X_1 \end{array}$$

We then bring down the next group of three digits. This has the effect of scaling the answer by 10.



**Stage 3 -** In general given  $(a_1 + a_2 + \dots + a_{k-1})$  find the *largest* value of  $a_k$  such that the error  $X_k = X_{k-1} - [3 \cdot (a_1 + a_2 + \dots + a_{k-1})^2 a_k + 3 \cdot (a_1 + a_2 + \dots + a_{k-1}) a_k^2 + a_k^3]$  is non-negative. Note that  $X_k = X - (a_1 + a_2 + \dots + a_k)^3$

Find the largest digit  $d_3$  such that  $3 \cdot 176400 \cdot d_3 + 3 \cdot 420 \cdot d_3^2 + d_3^3 = 529200 \cdot d_3 + 1260 \cdot d_3^2 + d_3^3$  is less than or equal to 5113000. Because of scaling  $(a_1 + a_2) = 420$

$$\begin{array}{r}
 3 \times 1600 \times 2 + 3 \times 40 \times 2^2 + 2^3 \\
 4800 \times 2 + 120 \times 2^2 + 2^3 \\
 \hline
 3 \times 176400 \times d_3 + 3 \times 420 \times d_3^2 + d_3^3 \\
 = 529200 \times d_3 + 1260 \times d_3^2 + d_3^3
 \end{array}$$

$4^3$

$4 \quad 2 \quad d_3$   
 +-----  
 | 79 201 000  
 -64  
 ---  
 15 201  
 -10 088  
 -----  
 5 113 000

We estimate  $d_3$  to be 9 which works

$$\begin{array}{r}
 3 \times 1600 \times 2 + 3 \times 40 \times 2^2 + 2^3 \\
 4800 \times 2 + 120 \times 2^2 + 2^3 \\
 \hline
 3 \times 176400 \times d_3 + 3 \times 420 \times d_3^2 + d_3^3 \\
 = 529200 \times 9 + 1260 \times 9^2 + 9^3
 \end{array}$$

$4^3$

$4 \quad 2 \quad 9$   
 +-----  
 | 79 201.000  
 -64  
 ---  
 15 201  
 -10 088  
 -----  
 5 113 000  
 -4 865 589  
 -----  
 247 411 000

## Reducing Computational Complexity - I

We can simplify the squaring of  $(a_1+a_2+\dots+a_{k-1})^2$  at the  $k^{\text{th}}$  step by using previously computed values. Let  $u_k$  be the value of  $(a_1+a_2+\dots+a_{k-1})$  and  $v_k = u_k^2$  be the value of  $(a_1+a_2+\dots+a_{k-1})^2$  at the  $k^{\text{th}}$  step. It's not difficult to derive a pair of interlocking formulas to compute  $u_k$  and  $v_k$ .

$$u_k = \begin{cases} 0 & \text{if } k = 0 \\ (u_{k-1} + d_{k-1}) \cdot 10 & \text{if } k > 0 \end{cases}$$

and

$$v_k = \begin{cases} 0 & \text{if } k = 0 \\ (v_{k-1} + 2 \cdot u_{k-1} \cdot d_{k-1} + d_{k-1}^2) \cdot 100 & \text{if } k > 0 \end{cases}$$

Given  $u_{k-1}$  and  $v_{k-1}$  the computational cost of computing  $v_k = (a_1+a_2+\dots+a_{k-1})^2$  is reduced to an addition followed by a multiplication by 10 to obtain  $u_k$  plus *two* additions and *three* simple multiplications (by 2, by the digit  $d_k$  and by 100) to obtain  $v_k$  (assuming a simple table lookup to find the square of the digit  $d_k$ ).

	4	2	9	
	+-----			
	79 201.000			
$4^3$	-64			
	---			
	15 201			
	-10 088			
	-----			
	5 113 000			
	-4 865 589			
	-----			
	247 411 000			

  

$$3 \times 1600 \times 2 + 3 \times 40 \times 2^2 + 2^3$$

$$4800 \times 2 + 120 \times 2^2 + 2^3$$
  

$$3 \times 176400 \times d_3 + 3 \times 420 \times d_3^2 + d_3^3$$

$$= 529200 \times 9 + 1260 \times 9^2 + 9^3$$

$$4^3$$

For example  $(1600 + 2 \times 40 \times 2 + 2^2) \times 100 = 176400$

### Stage 4

			4	2 . 9 4
		+-----		
		79 201.000 000		
	$4^3$	-64		
		---		
		15 201		
		-10 088		
		-----		
		5 113 000		
		-4 865 589		
		-----		
		247 411 000		
		-221 055 184		
		-----		
		26 355 816		

$$u_4 = (u_3 + d_3) \times 10 = 4290$$

$$v_4 = (v_3 + 2 \times u_3 \times d_3 + d_3^2) \times 100 = 18\,404\,100$$

## Reducing Computational Complexity - II

The *difference* between adjacent terms of the form  $3 \cdot v_k \cdot m + 3 \cdot u_k \cdot m^2 + m^3$  for  $m = 1, 2$ , etc can be written as

$$3 \cdot v_k \cdot m + 3 \cdot u_k \cdot m^2 + m^3 - (3 \cdot v_k \cdot (m-1) + 3 \cdot u_k \cdot (m-1)^2 + (m-1)^3) = 3 \cdot v_k + 3 \cdot u_k \cdot (2m-1) + \Delta m^3$$

Let  $\Delta m^3 = m^3 - (m-1)^3$  denote the difference between adjacent cubes

The computational cost of estimating the largest  $d_k$  so that  $X_{k-1} - 3 \cdot v_k \cdot d_k + 3 \cdot u_k \cdot d_k^2 + d_k^3$  is non-negative can be simplified by subtracting  $3 \cdot v_k + 3 \cdot u_k \cdot 1 + 1$  from  $X_{k-1}$  and continuing to subtract expressions of the form  $3 \cdot v_k + 3 \cdot u_k \cdot (2m-1) + \Delta m^3$  until a negative value is obtained (then restoring the previous value). The difference between adjacent cubes (i.e. 1, 7, 19, 37, 61, 91, 127, 169, 217, 271) is easily found by table look up.

For example, the calculation for  $k = 4$  above (where  $3 \cdot v_4 = 55,212,300$  and  $3 \cdot u_4 = 12,870$ ) can be redone using the following table.

$d_4$	$55212300 + 12870 \times (2k-1) + \Delta d_4^3$	$X_3 \times 1000 = 247411000$
1	$55212300 + 12870 \times 1 + 1$	192185829
2	$55212300 + 12870 \times 3 + 7$	136934912
3	$55212300 + 12870 \times 5 + 19$	816582434
4	$55212300 + 12870 \times 7 + 37$	<b>26355816</b>
5	$55212300 + 12870 \times 9 + 61$	-28972375
6	$55212399 + 12870 \times 11 + 127$	...

Computationally for each row you need only one multiplication (by an odd integer), two additions and one table look up for the first difference of the cube.

## References

Sugai, Iwao; "Extraction of Roots by Repeated Subtractions for Digital Computers"; *Comm. of A.C.M.*; Vol 1 No 12; December 1958; pp 6 – 8.

Asimov, Isaac; "The Feeling of Power"; *The Mathematical Magpie*, Clifton Fadiman, ed.; Simon and Schuster; New York; 1962.

My Home Page: <http://www4.wittenberg.edu/academics/mathcomp/shelburn.shtml>