**A faster version of the Welton J. Crook cube root method**

This is a description of a faster version of the Welton J. Crook cube root algorithm, along the same lines as the speeded up version of the Crook square root algorithm which I have described in a previous paper. As one would expect, this cube root algorithm is a good deal more complicated than the square root method, but may still be useful for those who really want to use the abacus or soroban to find cube roots.

I assume the reader is already familiar with the basic Crook cube root algorithm. If not, it is described here: http://webhome.idirect.com/~totton/soroban/cuberoot2.html . For each new digit of the cube root, I begin the computation just as Crook does, with the same active cube number, the same square number, and the same root number. I have developed some equations which allow one to home in on the next digit without going thru the tedious iterative process of the original algorithm, and which then allow prompt updating of the "root number" and the "square number". One other change I have made is to skip the iteration at the beginning; instead, simply subtract the largest cube possible from the first digit group. This assumes that one knows, or has at hand a table of the cubes of the numbers one through nine. These cubes will also be needed for plugging into the equations which follow. Note that in the following description I reverse the positions of the "cube number" and the "root number" from Crook's description, as I prefer to keep the cube on the left, the square in the middle, and the "root number" on the right - you may assign them whatever position you like, using as many rods on the soroban as needed.

Briefly described, my method to determine a given digit begins at the point in Crook's algorithm where the previous digit has just been determined and a new group of three digits has just been pulled down and tacked onto the "active" part of the remainder of the cube. Following Crook, we add a one to the current root number, but rather than add that sum into the square number, we subtract it, then we subtract two more from the square number. The reason for this is that the square number we have remaining from the previous digit computation is larger than Crook's square number at this point (the reason is given later) and we must get back to the same beginning square number before computing the next digit. Now again following Crook, we tack on two zeroes to the square number, then tack on a single zero to the root number and add 11 to it. At this point, we do <u>not</u> add this into the square number as Crook does. The new procedure is to use an equation to determine the largest of several possible discrete numbers which may be subtracted from the active part of the cube number. This will give us the new digit of the true cube root, because each of the above discrete numbers is calculated from a possible value, one through nine, of the new trial cube root digit. Having found the cube root digit and subtracted its corresponding number or "decrement" from the cube, we then use two more equations to update the root number to the same value it would have at this point in the original Crook algorithm, and to update the square number to the value it would have in Crook's method if the root digit were one larger than it is. We then proceed to find the next digit of the root. When we have as many digits as we need, we can find the true cube root just as in the original method by adding two to the root number and dividing by three, though that isn't necessary if we've been keeping a record of the true cube root digits obtained in the foregoing procedure.

Symbols used in the equations:

K = the currently "active" part of the cube or its remainder
$S_0$ = the square number obtained just after tacking on the two zeroes as described above
$R_1$ = the root number obtained just after adding 11 as described above
d = the value of the digit of the cube root which is being worked on (zero to nine)
$D_d$ = the decrement number, calculated from d, to subtract from K
$S_d$ = the square number updated after finding d
$R_d$ = the root number updated after finding d
$S_d$inc = the amount to increment $S_d$ if d is incremented from d-1 to d to correct an undershoot, or alternately the amount to decrement $S_d$ if d is decremented from d to d-1 to correct an overshoot

The equations are:

$D_d = d*S_0 + d^2*(R_1 - 1) + d^3$
$S_d = S_0 + [(d + 1)^2 - d^2]*(R_1 - 1) + (d + 1)^3 - d^3$
$R_d = R_1 + 3*(d - 1)$
$S_d$inc $= 2*(R_1 - 1) + 6*d$

Notice that when we compute $S_d$, we are actually computing the square number which, in Crook's algorithm, would be subtracted from K if d were one greater than it is. The reason for doing it this way is to allow easy checking for undershoot by simply comparing K and $S_d$.

The most difficult part of this method is determining what is the largest value of $D_d$ which may be subtracted from K. Simply dividing K by $S_0$ is not a reliable method because the square and cube parts of $D_d$ are often large compared to the linear part. It is probably best to make an estimate of $D_d$ by choosing trial values of d, mentally computing the estimate using only the first digit or two (rounded up) of $S_0$, $d^2$, and $R_1$, and comparing this to K. This should ensure that we will tend to undershoot, rather than overshoot the correct value of d. Once a likely value has been determined for d, we can then compute, and simultaneously subtract the exact value of $D_d$ from K, and update $S_d$ and $R_d$.

If K is larger than the new value of $S_d$, then we have undershot the value of d, so we increment d, add three to $R_d$, and subract $S_d$ from K. We then correct the value of $S_d$ and check that $S_d$ is now larger than K. In the rare event of overshoot, where K goes negative when we subtract $S_d$, we decrement d, subtract three from Rd, correct the value of $S_d$, and then add this back to K.

Here's an example to illustrate the process:

Cube root of 52,313,624 = 374

| K | $S_d$ | $R_d$ |
|---|---|---|

Start by setting $S_0 = 0$ and $R_1 = 1$

53 313 624 0 1

Remember that the first group is handled differently; just subtract the largest possible cube from the first group

```
- 27      (d = 3)
-----
  25
```

and bring down the next group and compute the new values of $S_d$ and $R_d$ from the equations

$$S_d = 0 + (4^2 - 3^2)*(1 - 1) + 4^3 - 3^3 = 37$$
$$R_d = 1 + 3*(3 - 1) = 7$$

25 313 37 7

Add one to $R_1$ as in the Crook algorithm

```
      + 1
      -----
        8
```

but now, subtract this number and an additional 2 from $S_d$

```
      -  8
      -  2
      ------
        27
```

and now follow Crook again to get the beginning root and square numbers

```
        80
      +  11
      ------
2700    91
```

At this point we do <u>not</u> add $R_1$ to $S_0$

25 313 2700 91

Now we must determine the value of d which yields the largest value of $D_d$ which can be subtracted from K.. If we guess at d = 8, we can estimate $D_d$ (using only the first one or two digits rounded up) as 8*2700 + 64*100 + 500 = 28500, which is too large. If d is 7, then $D_d$ is approximately 7*2700 + 50*100 + 400 = 24300, so d really is 7.

$D_d = 7*2700 + 49*(90) + 343$

```
  25 313
- 14
 -  4 9
  - 3 6
   -   81
    -  343
------------
    1 660
```

Now update $S_d$ and $R_d$

$S_d = 2700 + (64 - 49)*(91 - 1) + 512 - 343 = 2700 + 15*90 + 512 - 343$

```
  2700
+ 09
 + 45
 + 512
 -  343
---------
  4219
```

$R_d = 91 + 3*(7 - 1) = 91 + 3*6$

```
  91
+ 18
 ----
 109
```

We now have:

| | | |
|---|---|---|
| 1660 | 4219 | 109 |

Again, follow our previous procedure to get the new $S_0$ and $R_1$

```
                  +  1
                  ------
                   110
- 110
-    2
-------
 4107              1100
                   +  11
                  --------
                   1111
```

and bring down the next group in K, but don't add $R_1$ to $S_0$

| | | |
|---|---|---|
| 1 660 624 | 410700 | 1111 |

Now we start the process all over again to determine the next digit:

Find a value for d which yields the maximum value of $D_d$ which can be subtrated from K
If we guess at d = 4, then $D_d$ is approximately 4*410,000 + 16*1200 + 64 =
approximately 1,660,000 so try d = 4:
$D_d = 4*410700 + 16*(1111 - 1) + 64 = 4*410700 + 16*1110 + 64$

```
    1 660 624
-   1 6
 -    04
  -     2 8
   -   16
    -    1 6
      -   16
        -  64
----------------
```

0 000 000

There is no need to update $S_d$ now because K = 0. $R_d$ only needs to be updated if we haven't been keeping a record of the root digits and thus have to find the root from the root number $R_d$. In this case, $R_d = 1111 + 3*(4 - 1) = 1120$ and the root is 1122/3 = 374.

Roundoff

I don't know a simple way to determine when to round up the last digit of the root. Probably the best that can be done is to find the new $S_0$ and $R_1$ for the following digit and then try to estimate d for that digit - if it is five or greater then round up the last digit of the root and/or add three to the root number. Estimating d generally gets easier as the number of digits previously determined increases, because the $S_0$ term begins to dominate in the equation for $D_d$, so we get better estimates by simply dividing $S_0$ into K.

Undershoot

As an example of what to do when our estimate undershoots the digit, let's go back to the point in the above example where we are trying to find the second digit and choose d = 6 instead of the actual value of 7.

```
  25 313                          2700                    91
D_d = 6*2700 + 36*(90) + 216
```

```
  25 313
- 12
 -  4 2
  - 2 7
  -   54
   -  216
-----------
   5 657
```

Now update $S_d$ and $R_d$ for d = 6

$S_d = 2700 + (49 - 36)*(91 - 1) + 343 - 216 = 2700 + 13*90 + 343 - 216$

```
   2700
+ 09
    27
  + 343
  -  216
---------
  3997
```

$R_d = 91 + 3*(6 - 1) = 91 + 3*5$

```
    91
+ 15
  ----
  106
```

Because $S_d$ is less than K, we know that we have undershot. We must increment d and subtract Sd from K.
d = d+1 = 6 + 1 = 7

```
   5657
 -  3997
 ---------
   1660
```

Now update Sd and Rd
Rd = Rd + 3 = 106 + 3 = 109
$S_d$inc = 2*($R_1$ - 1) + 6*d = 2*90 + 6*7

```
    3997
  + 18
   +  42
 -------
   4219
```

Overshoot

If we overshoot the value of d so that K goes negative (this should be rare using the estimation method described earlier) then we must decrement d and correct $S_d$. We then add the corrected $S_d$ to K to make it positive and decrement $R_1$ by 3.

If, in the previous example we had taken d as 8, then we would have:
$D_d$ = 8*2700 + 64*90 + 512

```
   25 313
-   16
 -    5 600
  -  5 400
  -     360
  -     512
------------
  997 441    (the leading 9's mean this is a negative number)
```

At this point, recognizing that we have overshot, we could just compute $S_d$ and $R_d$ for d = 7 and add $S_d$ back into K. However, if we didn't recognize this in time and went ahead and computed $S_d$ for d = 8:
$S_d$ = 2700 + (81 - 64)*90 + 729 - 512 = 4447
we could then correct it by subtracting $S_d$inc = 2*90 + 6*8 = 228
The corrected value of $S_d$ is 4447 - 228 = 4219 which is the same as $S_d$ for d = 7. We then add it back into K

```
  997 441
+   4  219
--------------
     1 660
```

$R_d$ for d = 8 is 91 + 13*7 = 112, so we would need to decrement that by 3 to go back to d = 7

When one of the digits of the root is zero

Here's another example which shows what to do when one of the digits of the root is zero:

Cube root of 8,869,743 = 207

| K | $S_d$ | $R_d$ |
|---|---|---|
| 8,869,743 | 0 | 1 |
| - 8 (d = 2) | | |
| ---- | | |
| 869 | | |

$S_d = 0 + (3^2 - 2^2)*(1 - 1) + 3^3 - 2^3 = 19$
$R_d = 1 + 3*(2 - 1) = 4$

Go to the next digit as before

| | | |
|---|---|---|
| 869 | 19 | 4 |
| | | + 1 |
| | | ----- |
| | | 5 |
| | - 5 | |
| | - 2 | |
| | ----- | |
| | 12 | 50 |
| | | + 11 |
| | | ------ |
| 869 | 1200 | 61 |

Because $S_d$ is larger than K, d must be zero, so pull down another group to K, tack on two more zeroes to $S_d$, and change $R_d$ = 50 to $R_d$ = 500 and add 101 instead of 11.

| | | |
|---|---|---|
| 869 743 | 120000 | 601 |

Try d = 7
Estimate $D_d$ approximately = 7*120000 + 50*600 + 340 = 870340 - this is close enough to try. Actual $D_d$ is 7*120000 + 49*600 + 343 = 869743.

| |
|---|
| - 869 743 |
| ---------- |
| 000 000 |

No need to update $S_d$. $R_d$ = 601 + 3*6 = 619. The root is 621/3 = 207.

Steve Treadwell
April 1, 2009