Balanced (Signed) Ternary Notation

(Talk given at Spring 2006 Ohio Section MAA meeting – Revised Summer 2009)

Brian J. Shelburne
Department of Mathematics and Computer Science
Wittenberg University
Springfield, Ohio 45501

Abstract: In the 1840's Thomas Fowler, a self taught mathematician and inventor from Devonshire, England invented a calculating machine that used *balanced ternary notation* to perform its calculations. Fowler observed that the mechanics of calculation were simplified by using balanced ternary notation, its use being analogous to the use of binary notation in modern computers. However, balanced ternary notation has some advantages over binary notation, especially in the way negative numbers are handled. This talk is an introduction to balanced ternary notation and balanced ternary calculations.

1. Positional Notation: Everyone is familiar the base 10 and base 2 (binary) notation where the position of each digit determines the power of 10 or 2 used to assign the weight to that digit

Example $1725 = 1\times10^3 + 7\times10^2 + 2\times10^1 + 5\times10^0$

$$1101_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 13_{decimal}$$

Ternary notation is base 3 notation using only the three digits 0, 1 and 2. Instead of having units, tens, hundreds, etc. columns (i.e. power of ten) it has units, threes, nines, twenty-sevens, etc. columns (i.e. powers of three).

$$1202_3 = 1\times3^3 + 2\times3^2 + 0\times3^1 + 2\times3^0 = 47_{decimal}$$

Balanced ternary notation uses the digits -1, 0, and 1 instead of the digits 0, 1, and 2. Note the *negative* valued digit. While this might seem unusual at first, this is what makes balanced ternary calculations so effective. The three digit balanced ternary number 1(-1)1 is one times nine minus one times three plus one or

$$1(-1)1 = 1\times3^2 + (-1)\times3^1 + 1\times3^0 = 9-3+1 = 7$$

The use of parentheses and -1 is notationally awkward; it's easier to use the symbols "+" and "−" in place of "1" and "-1" along with the 0 in balanced ternary notation. From now on we'll adopt this "signs only" balanced ternary representation as the standard representation

Counting from -1 to 10 (decimal) in Balanced Ternary

decimal	ternary	balanced ternary	"signs only" balanced ternary
-1	-1	(−1)	−
0	0	0	0
1	1	1	+
2	2	1(−1)	+ −
3	10	10	+ 0
4	11	11	+ +
5	12	1(−1)(−1)	+ − −
6	20	1(−1)0	+ − 0
7	21	1(−1)1	+ − +
8	22	10−1	+ 0 −
9	100	100	+ 0 0
10	101	101	+ 0 +

There is a trick to converting *ternary notation* to *balanced ternary notation*. Since $2_3 = +-$ (i.e. 2 equals 3 minus 1) replace each 2 in a ternary expansion with a − and add 1 to the next column over recalling that 2 plus 1 equals 0 carry 1!

Example:

$$47_{decimal} = 1202_3 = 121- = 12+-- = 1+--- = 2---- = +----- = 81-27-9+3-1 = 47_{decimal}$$

Note the four transitions (red to blue) used to convert 1202_3 to $+--+-$

Positive values always have + as their left-most or most significant digit; *negative* values always have – is the left-most digit.

To *negate* any balanced ternary representation simply change all + digits to – digits, all – digits to + digits and leave 0 alone.

Example: $+--+- = +81-27-9+3-1 = 47$

$-++-+ = -81+27+9-3+1 = -47$

Aside: Balanced Ternary notation carries over in a natural way to represent fractional values. One third is obviously $0.+$ (zero point plus) while two thirds is $+.-$ (plus point minus!). Not surprisingly, one half is the repeating balanced ternary expression $0.\overline{+-}$ and one tenth is $0.\overline{0+0-}$

2. Decimal to Balanced Ternary Conversions

Balanced Ternary to Decimal conversion is easily done by adding/subtracting the powers of three in the polynomial expansion.

$$+--+-=+81-27-9+3-1=47$$

Decimal to Balanced Ternary conversion can be done by either subtracting out the proper powers of 3 to first obtain the ternary representation.

For example there are no 81's in 47 but we can subtract out a 27 leaving 20. From the 20 we can subtract out two 9's leaving 2.

```
81   27    9    3    1  <- powers of 3
--   --   --   --   --
 0    1    2    0    2  <- count of powers of 3

     47   20         2
    -27   -9        -1
    ---  ---        --
     20   11         1
         - 9        -1
         ---        --
           2         0
```

or by using integer division

```
47 / 3 = 15 r 2
15 / 3 = 5  r 0
 5 / 3 = 1  r 2
 1 / 3 = 0  r 1     topple the remainders: 1 2 0 2
```

Then converting ternary to balanced ternary

```
1 2 0 2 => + - - + -
```

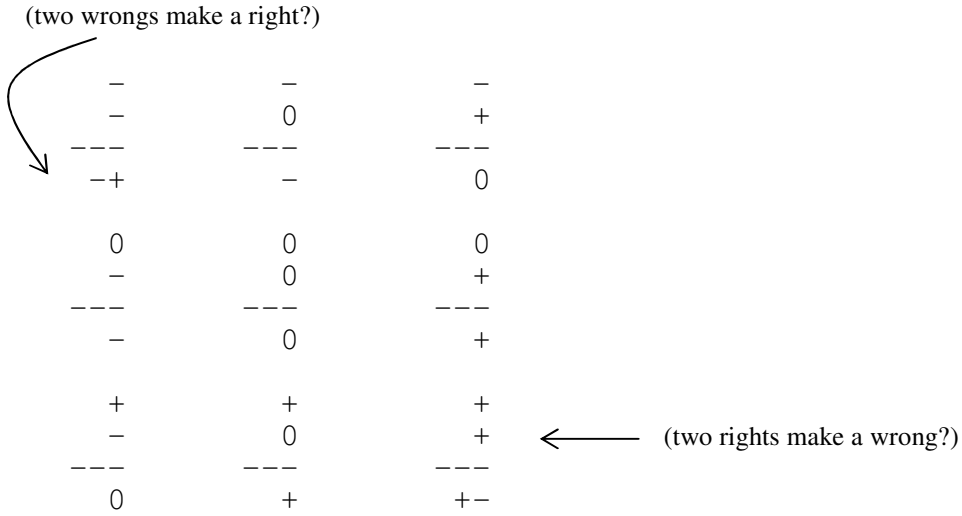
3. Where's the Advantage? Addition and Subtraction

The rules for arithmetic for balanced ternary are simple. First of all, to negate a number simply reverse the digits.

For example, to negate $+0-+-0$ you obtain $-0+-+0$

Whether a representation denotes a positive or negative value depends on the leading digit; in a sense one does not distinguish between positive and negative values.

There are only nine rules which cover both addition and subtraction, the latter done by adding a negative value. There is no borrowing, only a carry into the next position over.



Note that only the first (two wrongs!) and last (two rights!) rules require a carry.

4. Where's the Advantage? Multiplication and Division

The addition-only type rules for rules for balanced ternary notation make multiplication and division easy to do

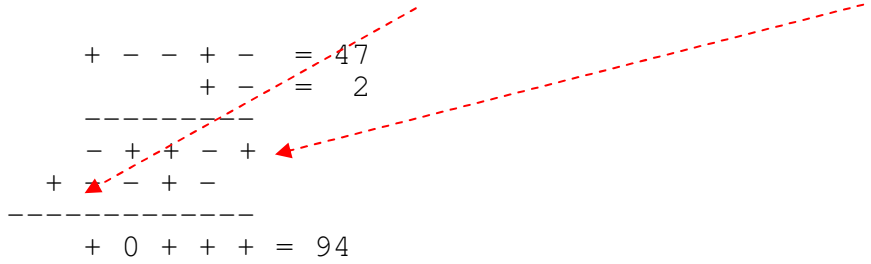
Example

```
          + - - + - = 47
            + - 0 - = 17
          ---------
          - + + - +
      - + + - + 0
    + - - + -
 ----------------
      + 0 + 0 - - + = 799
```

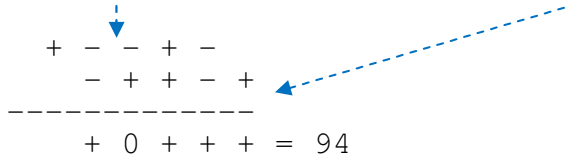
Observe how partial products are obtained. A "+" multiplier digit simply copies the multiplicand (after shifting), a 0 multiplier digit does nothing but shift the multiplicand, and a "−" multiplier digit inverts the multiplicand (after shifting).

Adding columns of digits is simplified by *first* canceling pairs of +'s and −'s. This reduces the number of carries and the amount of carry propagation which are problems with calculations in binary notation.

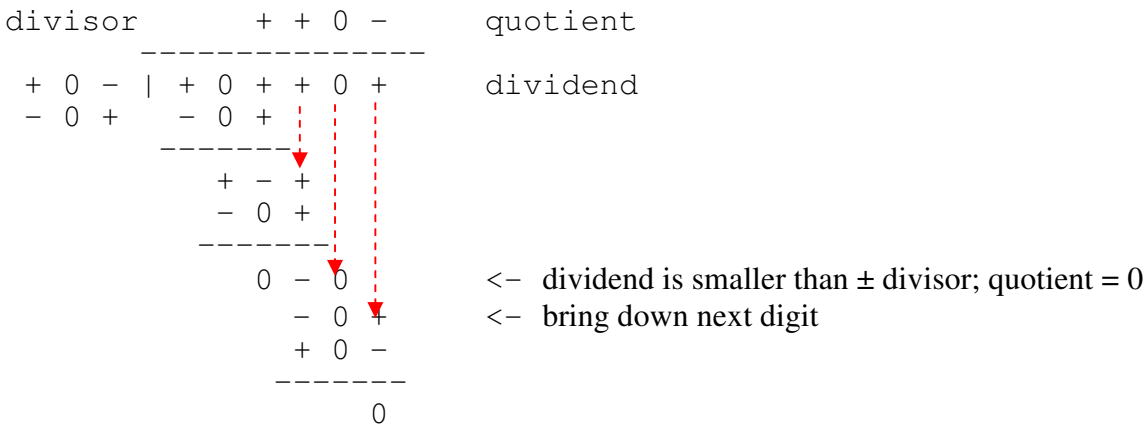
Doubling a number is fairly easy: left shift the multiplicand and add the inverted multiplicand!



Or copy the multiplicand, *invert and right shift the multiplicand* and add



Division is done the normal way but it makes effective use of both the positive and negative divisor (make copies of both).



If the current dividend is positive (leading digit is +) add the *negative* divisor and use + as your quotient digit. Bring down the next digit.

If the dividend is negative (leading digit is –) add the *positive* divisor and use – as the quotient digit. Bring down the next digit.
If after bringing down the next digit, the dividend is too small to add either the negative or positive divisor, use 0 as your quotient digit. Bring down the next digit.

Like binary division, balanced ternary has none of the *estimating* the quotient digit of decimal long division. Also since binary subtraction is somewhat difficult due to borrow propagation, the balanced ternary use of addition of the negative divisor, makes the calculation easier.

One minor point: Integer division might not result in a positive remainder. For example 17 divided by 3 should be 5 remainder 1 not 6 remainder -1 as computed below. The latter is correct but by convention remainders are usually positive.

```
              + - 0   : 6
           ----------
 3:   + 0 | + - 0 -   : 17
      - 0   - 0
            -----
              - 0
              + 0
              -----
                 0 -   : -1  remainder
```

If the remainder is negative add the divisor to the negative remainder (+− = 2) and subtract 1 from (or add -1 to) the quotient (+−− = 5).

5. Where's the Advantage? Taking Square Roots?

Recall the technique used to take a square root by hand

a. Starting from the decimal point partition the digits into pairs. Call the leading unpaired digit or paired digits, the "current dividend".

b. Starting with the "current dividend" estimate the largest digit ("current quotient") whose square is less than or equal to the "current dividend". Subtract the "quotient squared" from the "current dividend" and bring down the next two digits for the next "current dividend". Call the current quotient "a".

c. Double the current quotient `a` and left shift it to obtain `2a<<1`[1]. Find the largest digit `b` such that `(2a<<1) + b` *times* `b` is less than the current dividend. Subtract. Repeat [2]

Note: Estimate digit b by finding the largest digit b such that 2a<<1 alone times b is less than the current dividend.

[1] Notation: Let `x<<1` denote the value x *left shifted* 1 place. So `2a<<1` denotes 2 times `a` left shifted one place. If a = 17 then `2a<<1` would equal 170.

[2] The underlying mathematical method being used here is completing the square.

Example; Compute $\sqrt{273}$

```
        1   6 .  5  2
      +---------------
  1   | 2  73 . 00 00    1 is largest integer whose square is less than 2
        1
       -----------
 2 b    1  73            current quotient is 173; double 1 and left shift plus b to obtain 2b
 2 6    1  56            try b = 6; subtract 26 times 6
       ----------
 32 b      17  00        current quotient is 1700; double 16, left shift plus b to obtain  32b
 32 5      16  25        try b = 5; subtract 325 times 5
           ------
 330 b         75  00    current quotient is 7500; double 165, left shift plus  b
 330 2         66  04    try b = 2; subtract 3302 times 2
               ------
                8  96    etc.
```

These steps are much easier to do in balanced ternary notation.

To double a value (multiply by 2 or + −) left shift it and add the negative (obtained by inverting the digits of the original). Your choice for the digit "b" will be +, −, or 0 so multiplying by digit "b" will result in no change (multiplying by 1), 0 or negating (multiplying by -1).

The *current quotient* is always positive. If the "current dividend" is negative your next digit b will be either – or 0; if positive b will be + or 0. Use the non zero value of b and if the absolute value of the result is less than the absolute value of the "current dividend", use that value for b; otherwise use b = 0.

Example: Find the square root of + − + 0 + = 64

```
                 +    0    −    = 8
               -------------
+              | +, − +, 0 +
− ---------->    −
               ----
                 0  − +          <- bring down next two digits (current dividend)
+ − b                            <- double current quotient + ; left shift plus b
+ − −                            <- current dividend negative - try b = −
− + +                            <- × b (negate)
+ − − ------>    +  − −          <- add
                 -------
                    + 0          <- bigger in absolute value than − + so b = 0
                    − +  0 +     <- bring down next 2 digits; again "b" is – or 0
+ − 0 b                          <- double + 0, left shift plus b
+ − 0 −                          <- negative so try b = −
− + 0 +                          <- × b (negate)
+ − 0 − -------->  + −  0 −      <- add
                   ---------
                   0 0  0 0
```

6. Balanced Ternary in Retrospect

There is some evidence that balanced ternary notation is the most efficient of numbering systems, the so called "Goldilocks" of numbering systems in that "base 2 is too small", "base 10 is too big", and "base 3 is just right".[3]

The main advantage of balanced ternary representation over binary is much less carry propagation. Also negating numbers is easier to do in balanced ternary than it is in twos complement binary representation (in balanced ternary you just invert +'s and –'s where as in two complement binary you complement 0's and 1's but then *add 1*). And as we have seen, arithmetic operations including square roots are easy to do in balanced ternary. Also fewer digits are needed in balanced ternary to represent large values than in binary.

Of course in terms of the *number* of digits needed to represent large values, base ten is better than both binary and balanced ternary. However the rules for arithmetic are much more complicated. Recall how long division works. With ten-digit decimal you have to guess which of the ten digits is the largest digit such that when you multiply it by the divisor, the result is less than the current dividend. In balanced ternary you choose either + or – depending on the sign of the current dividend; there is no guessing.

Interestingly enough because of the advantages balanced ternary has over binary notation, there have been a few attempts to construct *ternary* computers. In the late 1950's Nikolai Brousentsov at the Moscow State University designed a ternary computer called the "Setun", approximately 50 of which were built between 1958 and 1965. It had a word length of 18 trit's (similar to the contraction for bit - *binary digit*. An 18 trit word would have the same range as a 28 bit word. (So instead of a "flip-flop" you have a "flip-flap-flop"!)

In 1973 G. Frieder at SUNY Buffalo designed a base-3 machines called the "Ternac" along with a software emulator for it.

In closing here are two quotes, the first from Thomas Fowler, the 19th century pioneer in mechanical computing whose work on building a balanced ternary computing device was the inspiration for this talk, and the second from Donald Knuth, a 20th century computer scientist.

> "*I often reflect that had the Ternary instead of the denary Notation been adopted in the infancy of Society, Machines something like the present would long ere this been common, as the transition from mental to mechanical calculation would have been so very obvious and simple*" – Thomas Fowler, 8 May 1841
>
> "*Perhaps the prettiest number system of all … is the balanced ternary notation*" – Donald Knuth, *The Art of Programming*

[3] If r is the *radix* and w the *width* of an integer (number of digits), you want to minimize the product $y = r \cdot w$ for some fixed value of $C = r^w$. Thus $w = \ln C / \ln r$ so if you minimize $y = r \cdot \ln C / \ln r$, the critical point occurs at e which is closer to 3 than 2.

7. References

1 M. Glusker, D. Hogan & P. Vass, "The Ternary Calculating Machine of Thomas Fowler," *IEEE Annals of the History of Computing*, vol. 27, no. 3, 2005, pp 4 – 22

2 B. Hayes, "Third Base," *American Scientist*, vol. 89, no. 6, 2001, pp 490-494

3 A. Stakhov, "Brousentsov's Ternary Principle, Bergman's Number System and Ternary Mirror-symmetrical Arithmetic", *The Computer Journal*, vol. 45, no. 2, 2002, pp. 221 -236