BALANCER SOLIDITY NUMERICAL APPROXIMATIONS

FERNANDO MARTINELLI

1. Original formulas to be approximated

When a user sends to Balancer tokens i to get tokens o, we can calculate A_o (the amount of tokens o) a user gets when sending A_i (the amount of tokens i):

$$A_o = \left(1 - \left(\frac{B_i}{B_i + A_i}\right)^{\frac{W_i}{W_o}}\right).B_o \tag{1}$$

It is also very useful for traders to know how much they need to send of the input token A_i to get a desired amount of output token A_o. We can calculate the amount A_i as a function of A_o similarly as follows:

$$A_i = \left(\left(\frac{B_o}{B_o - A_o}\right)^{\frac{W_o}{W_i}} - 1\right).B_i \tag{2}$$

Since solidity does not have fixed point algebra or more complex functions like fractional power we use the following binomial approximation:

$$(1+x)^\alpha = 1 + \alpha x + \frac{(\alpha)(\alpha-1)}{2!}x^2 + \frac{(\alpha)(\alpha-1)(\alpha-2)}{3!}x^3 + \cdots = \sum_{k=0}^{\infty}\binom{\alpha}{k}x^k \tag{3}$$

which converges for $|x| < 1$

1.1. Derivation for A_o.

We derive the solidity approximation of the first equation using the binomial approximation above:

$$A_o = \left(1 - (1+x)^\alpha\right).B_o \tag{4}$$

where

$$x = \frac{B_i}{B_i + A_i} - 1 = \frac{-A_i}{B_i + A_i} \tag{5}$$

Date: July 2019.

and

$$\alpha = \frac{W_i}{W_o} \tag{6}$$

A_i is by design limited to 10% of B_i. That is, no trade can exchange/sell more than 10% of Balancer's balance of those tokens. This prevents excessive slippage loss for traders.

$|x|$ is then by design always lower than 0.1.

Since calculations in solidity are done in integers, the order of the operations we choose is fundamental for the calculation to be correct. E.g. $(99/100) * 100 = 0$. This happens because $99/100$ is truncated to 0 and then multiplied by 100.

To avoid dividing two numbers that are close to each other (which truncates all the precision as in the example above), we multiply by B_o all terms in the binomial expansion used for approximating A_o:

$$\begin{gathered} A_o = \left(1 - (1+x)^{\alpha}\right).B_o = \\ B_o - \left(B_o + B_o\alpha x + B_o\frac{(\alpha)(\alpha-1)}{2!}x^2 + B_o\frac{(\alpha)(\alpha-1)(\alpha-2)}{3!}x^3 + \cdots\right) = \\ -\left(B_o\alpha x + B_o\frac{(\alpha)(\alpha-1)}{2!}x^2 + B_o\frac{(\alpha)(\alpha-1)(\alpha-2)}{3!}x^3 + \cdots\right) \end{gathered} \tag{7}$$

To make the solidity implementation simpler and more elegant using recursive functions, we can rewrite A_o as:

$$A_o = -\sum_{k=1}^{n} T_k \tag{8}$$

where:

$$T_0 = B_o \tag{9}$$

and

$$T_k = \frac{(\alpha - (k-1))\, x}{k} T_{k-1} \tag{10}$$

The binomial approximation described above is especially accurate for small values of α. When $\alpha > 1$ we split the calculation into two parts for increased accuracy:

$$A_o = \left(1 - \left(\frac{B_i}{B_i + A_i}\right)^{int\left(\frac{W_i}{W_o}\right)} \left(\frac{B_i}{B_i + A_i}\right)^{\frac{W_i}{W_o}\%1}\right).B_o \tag{11}$$

1.2. **Derivation for A_i.** Similarly to A_o, we have for A_i:

$$A_i = \left((1+x)^{\alpha} - 1\right).B_i \tag{12}$$

where

$$x = \frac{B_o}{B_o - A_o} - 1 = \frac{A_o}{B_o - A_o} \tag{13}$$

and

$$\alpha = \frac{W_o}{W_i} \tag{14}$$

To avoid dividing two numbers that are close to each other (which truncates all the precision as in the example above), we multiply by B_i all terms in the binomial expansion used for approximating A_i:

$$\begin{gathered} A_i = \left((1+x)^{\alpha} - 1\right).B_i = \\ \left(B_i + B_i\alpha x + B_i\frac{(\alpha)(\alpha-1)}{2!}x^2 + B_i\frac{(\alpha)(\alpha-1)(\alpha-2)}{3!}x^3 + \cdots\right) - B_i = \\ \left(B_i\alpha x + B_i\frac{(\alpha)(\alpha-1)}{2!}x^2 + B_i\frac{(\alpha)(\alpha-1)(\alpha-2)}{3!}x^3 + \cdots\right) \end{gathered} \tag{15}$$

To make the solidity implementation simpler and more elegant using recursive functions, we can rewrite A_i as:

$$A_i = \sum_{k=1}^{n} T_k \tag{16}$$

where:

$$T_0 = B_i \tag{17}$$

and

$$T_k = \frac{(\alpha - (k-1))\,x}{k} T_{k-1} \tag{18}$$