Anyhedge Numerical Error Analysis

Karol Trzeszczkowski

May 26, 2020

AnyHedge applies integer mathematics to resolve a contract. The goal of this paper is to locate sources of error, calculate upper limits for the error and prescribe safe ranges of contract parameters.

1 Introduction

1.1 Intent

AnyHedge contract is a futures contract with two positions, hedge and short, as described in the whitepaper [1]. For complete context, it is important to note that this paper follows the calculation steps in the document "Detailed execution of AnyHedge on Bitcoin Cash" and not the abstract steps in the whitepaper "High level execution of AnyHedge". From the perspective of the user the contract is described by three intent parameters introduced in part A) of the detailed execution.:

- hedge_value (denoted h),
- start_price_units_per_bch (p_1),
- largest_allowed_price_drop (L), $L \in [0\%, 100\%)$.

We will use these parameters as the starting point for numerical analysis of the contract. Two relevant parameters are obtained from the intent: `total_satoshis_in` that we will denote as T and `hedge_value_X_sats_per_bch`, later denoted as H_s which is a value of $h\,[\text{units}] \times 10^8 \frac{\text{sats}}{\text{฿}}$.

The following assumptions have been made:

- positive numbers,
- arbitrary precision,
- $T < H_s$.
- $0 <= \texttt{n\_hightrunc\_bytes} - \texttt{n\_lowtrunc\_bytes}) <= 3$

1.2 Error sources

There are three sources of error - calculating contract values from the intent parameters, mathematical operations in the contract and divisibility limit of Bitcoin.

BCH VMB (Bitcoin Cash Virtual Machine Bytecode, aka Bitcoin Script) can do numerical operation on 4 byte long numbers while the transaction values can be up to 8 byte long. BCH VMB also lacks operation of multiplication which requires replacing it with division operation instead. Because of this AnyHedge contract using oracle values necessarily deals with numbers larger than the 4 byte limit for numerical operations.

To avoid precision loss a special procedure of division and truncation is employed. The parameters derived from section 1.1 are truncated to two different levels of truncation and only then mathematical operations are performed on them.

This initial truncation of the parameters is a primary source of the error. This error propagates through all operations to the final result of the contract.

The mathematical operations performed by the contract are designed to minimize the information loss along the way. Nevertheless because there is a limit on the integer size, some information is lost in the process for values exceeding it.

Finally, Bitcoin Cash is divisible up to 10^8 satoshi units. Limited divisibility is only a problem when the total number of satoshis in the contract approaches one. This type of error source will be taken into account by describing errors in relation to the total amount in contract.

The goal of this paper is to estimate those errors and narrow safe ranges of parameters.

2 Notation

Different truncation levels will be denoted with the upper index T for high truncation and upper index t for low truncation, for example H_s^T means a `hedge_value_X_sats_per_bch` at a high truncation. H_s is related to H_s^T from 18.B) by multiplication $2^{8\ \texttt{n\_hightrunc\_bytes}}$. T is related to T^t from 18.B) by multiplication $2^{8\ \texttt{n\_lowtrunc\_bytes}}$. `hedge_value` will be denoted with h and `short_sats` with S. Price in $\frac{[\text{units}]}{\text{Ƀ}}$at the beginning of the contract will be denoted as p_1 and at the end as p_2. Other symbols will be defined over the course of the paper.

3 Error analysis

3.1 Initial truncation

In the section 18.B) of AnyHedge whitepaper two `truncated` values are calculated based on the intent: H_s^T which represents the amount that when divided by the current price of BCH in terms of desired units gives the amount of satoshis that will be paid to the person hedging in the contract, and T^t that represents the total amount of satoshis on the contract. If any of them exceeds the limit dictated by BCH VMB they are truncated. It is assumed that H_s is always larger than T therefore the number of truncated bytes from T^t is equal or lower than the number of truncated bytes from H_s^T. They can both have from 0 to 4 bytes of truncation, but the difference between them cannot be equal to 4 bytes, or the contract will fail.

In this step the truncation rejects some number of bits. The error introduced this way is smaller than the worst case - if all the bits rejected by the truncation are set to 1. To limit the error from above we put:

$$\Delta_\tau = \sum_{i=0}^{n} 2^i \text{sats} \tag{1}$$

where n is the number of bits truncated and $n \in \{, 8, 16, 24, 32\}$. By the symbol of Δ we will denote absolute worst case error. Subscript τ marks that it comes from truncation.

3.2 Mathematical Operations

Procedure

Mathematical operations executed in the contract, as described in section 18.D-1) of AnyHedge whitepaper, are the following:

1. $\frac{H_s^T}{p_2} = H_D^T$,
2. $H_s^T \bmod p_2 = H_{mod}^T$,
3. $2^{8\ n\_\texttt{delta}} H_D^T = H_1^t$,
4. $2^{8\ n\_\texttt{delta}} H_{mod}^T = H_{mod}^t$,
5. $\frac{H_{mod}^t}{p_2} = H_2^t$,
6. $H_1^t + H_2^t = H^t$,
7. $T^t - H^t = S^t$,
8. $H = 2^{8\ \texttt{n\_hightrunc\_bytes}} H^t$,
9. $S = 2^{8\ \texttt{n\_hightrunc\_bytes}} S^t$.

H is the value of hedge position payout in sats and S is the value of short position payout in sats.

Hedge

Division that happens during the resolution of the contract is executed in two steps corresponding to two levels of truncation. At a higher level the first division is done.

$$\frac{H_s^T}{p_2} = \mathtt{H_D^T}$$

and the remainder is preserved as:

$$H_s^T \bmod p_2 = H_{mod}^T$$

at this point no information is lost because we can reverse this operation.

$$H_D^T \times p_2 + H_{mod}^T = H_s^T$$

In the next step both values are untruncated to the level of `lowtrunc` by the difference between the two initial truncation levels, what is equivalent to multiplying both values by $2^{8\ n_{\mathtt{delta}}}$. At this level operations involving T^t can be performed. Now the remnder after the division is divided by the p_2. This is the point when information is lost. As before we have an equation:

$$H_2^t \times p_2 + H_{2\,mod}^t = H_{mod}^t \quad \Rightarrow \quad H_2^t = \frac{H_{mod}^t}{p_2} - \frac{H_{2\,mod}^t}{p_2}$$

$H_{2\,mod}^t$ is not recovered by the contract. It is lost. The relation $H_{2\,mod}^t \leq p_2 - 1$ is preserved by the definition of modulo, so the value of $\frac{H_{2\,mod}}{p_2} < 1$ and this is the part that is being lost from the result. The error at this truncation level is less than 1. Our goal is to limit the contract error from above so we will assume that it is equal to 1. This is an overestimation, but we aim to restrict the error from above. After untruncating the lower level we get the final result

$$\Delta_M = 2^{8\ \mathtt{n\_lowtrunc\_bytes}}\text{sats} \tag{2}$$

Δ_M is the absolute error originating from the mathematical operations in the contract. Similarly the error is being made on the short payout.

Short

Short position is calculated by subtracting the H from T. The errors originating from those two must be added. It will be shown in section 4.2. The absolute error affecting S is the sum of $\Delta_{\tau\,H}$ and $\Delta_{\tau\,T}$ where $\Delta\tau$ denote the errors originating from truncation of respectively H and T.

3.3 Divisibility of Bitcoin

The final error we want to calculate should be relative to the total payout in the unit of satoshi. Total contract payout is calculated as follows:

$$T = \frac{10^8\,h}{(1-L)p_1}$$

where L is the `largest_allowed_price_drop` and h is `hedge_value` . Later, in section 4.2, we will use this value to divide the total error. Considering relative error will let us take into account the finite granularity of bitcoin.

3.4 Upper limit

There is an upper limit for the parameters, and it is determined by the minimum payout of the hedge, due to price growth.

Hedge payout is calculated as follows:

$$\left(\frac{10^8 h}{p_2}\right).$$

We express it with x how many times price has grown.

$$x \equiv \frac{p_2}{p_1},$$

Hedge payout depending on how many times the price went up reads:

$$\left(\frac{10^8 h}{x\ p_1}\right).$$

Putting a restriction on this number will give us a safe range for how many times the price can grow, without causing additional errors.

4 Propagating and Adding Errors

4.1 Error propagation

The values used to calculate contract hedge and short positions are already affected by errors, there is the need to propagate this error to the result. It will be done by the standard formula of error propagation in products and quotients. To derive this formula, let us consider function of parameter with a small error. From Taylor expansion of this function we approximate:

$$f(y \pm \Delta y) \approx f(y) \pm f'(y)\Delta y.$$

The error propagated over the function reads:

$$\Delta f = f'(y)\Delta y.$$

In our case $f(y) = \frac{y}{p_2}$ so

$$\Delta f = f'(y)\Delta y = \frac{1}{p_2}\Delta y$$

$$f \equiv H = \frac{H_s^T}{p_2},$$

therefore we propagate the error as follows:

$$\Delta_H = \frac{\Delta_{H_s^T}}{p_2}.$$

The p_2 parameter should be replaced by the worst case scenario - largest allowed price drop.

4.2 Adding errors

Since we consider worse case scenario, not a statistical error, we shall add error terms with a regular addition, not using a square summation, as we would do with statistical error. We will also divide the sum of absolute errors by the total number of units in the contracts, as mentioned earlier.

$$\delta y = \frac{\sum_{\{i\}} \Delta_i}{T}.$$

4.3 Result

As a result we achieve two formulas: for relative error of hedge and for relative error of short positions.

$$\delta_H = \frac{\Delta_M + \Delta_\tau H}{T},$$

$$\delta_S = \frac{\Delta_M + \Delta_\tau H + \Delta_\tau T}{T}.$$

δ_S is strictly larger than δ_H. To restrain the error from above we chose the larger of two values: δ_S.

5 Intent parameters restrictions

We can now plot the safe ranges for parameters under the condition, that the error shall be smaller than arbitrary fraction.

5.1 Expression for errors

The final expression describing errors is:

$$\delta_S = \frac{\Delta_M + \Delta_{\tau H} + \Delta_{\tau T}}{T}.$$

Based on this equation we will appoint restrictions required to keep the error below a certain level.

5.2 Safe regimes of intent parameters

The safe region is appointed by the relation $\delta_S < 0.15\%$. This means that the difference between the actual and the naive expected satoshis output for short will be less than 0.15%. Because δ_S is strictly larger than δ_H, we can also say that the error for hedge will be less than this value.It is plotted on the figure 1 in terms of 3 parameters of the intent.

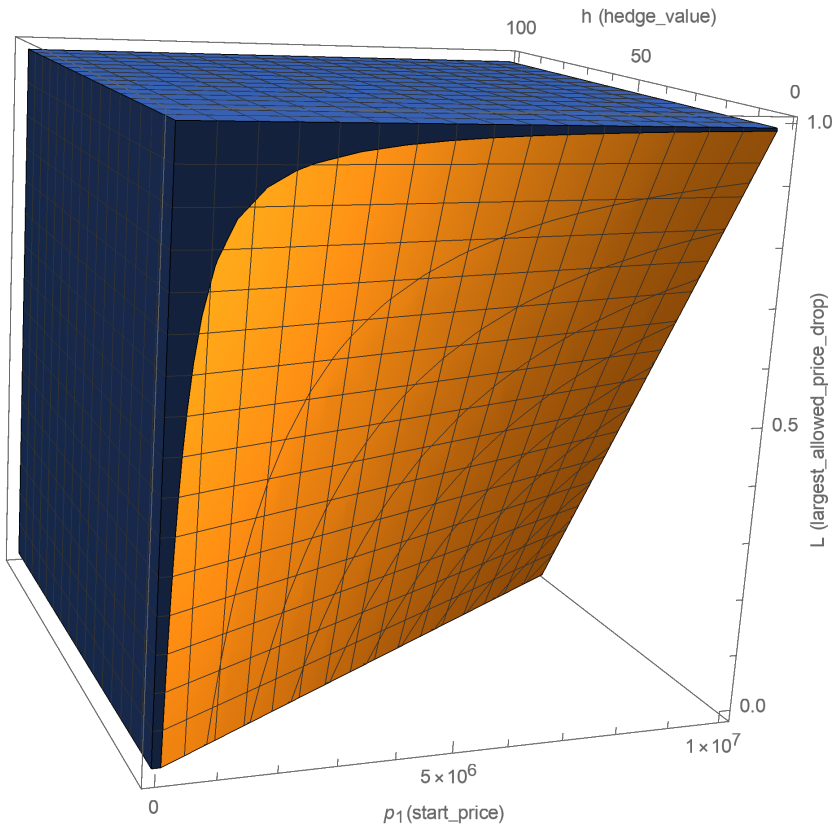

Figure 1: A region of parameters restricting the contract outcome error below 0.15%.

As the largest_allowed_price_drop goes up the safe regime is growing. The worst case scenario is largest_allowed_price_drop = 0. This case is shown on the figure 2 for two dimensions of remaining intent parameters.

Example

To determine a safe range of p_1 based on the figure 2 for the hedge of 200 units, we have to read the shaded range, which is $(0 - 1.6 \times 10^7)\frac{\text{units}}{\text{฿}}$.

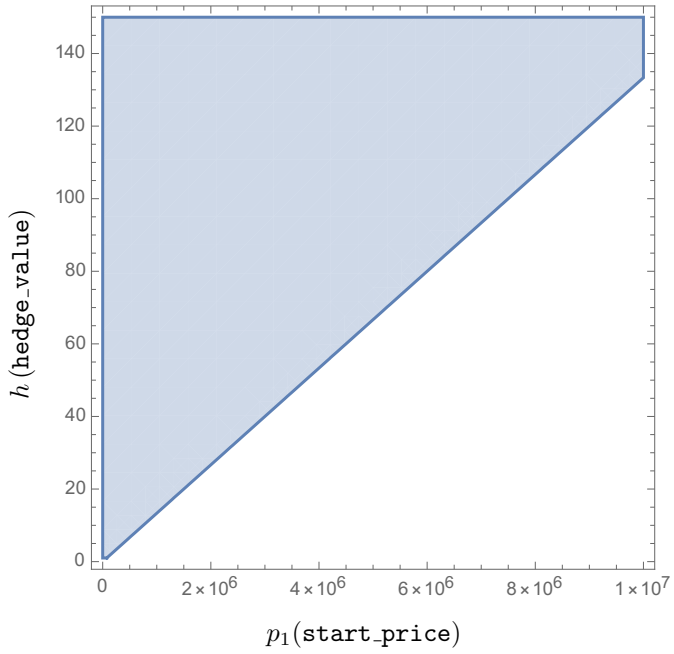

Figure 2: Shaded region with short error below 0.15% of the contract

5.3 Upper limit

For the upper restriction for price growth we have to agree on what is the minimum outcome in terms of satoshis that we are willing to approve. If the price of an asset grows to be big enough, the output could be too small. We chose to have an outcome not smaller than $\alpha = 10$ sats.

The formula for the upper limit is as follows:

$$\left(\frac{10^8 h}{p_1}\right) < \alpha x$$

the valid maximum price growth range is shown on the Figure 3.

Example

For the hedge of 50 units and p_1 of $5 \times 10^6 \frac{\text{units}}{฿}$ the price can grow up to 100 times before the hedge payout will go below 10 sats.

5.4 Intent restriction

As mentioned in section 1.1, parameters should follow the assumption we made, namely:

$$T < H_s.$$

6 Comparison with experiment

To evaluate prediction of this model, it was compared with a simulated outcome of the contract. The result reflect the upper boundary of the error distribution. It was shown on the figure 4 The model overestimates the error about two times due to overestimation as described in 3.2 and 4.3.

References

[1] AnyHedge: A Decentralized Hedge Solution against Arbitrary Assets on Bitcoin Cash, imaginary_username (im_uname#102) John Nieri (emergent_reasons#100) Jonathan Silverblood (Jonathan#100).

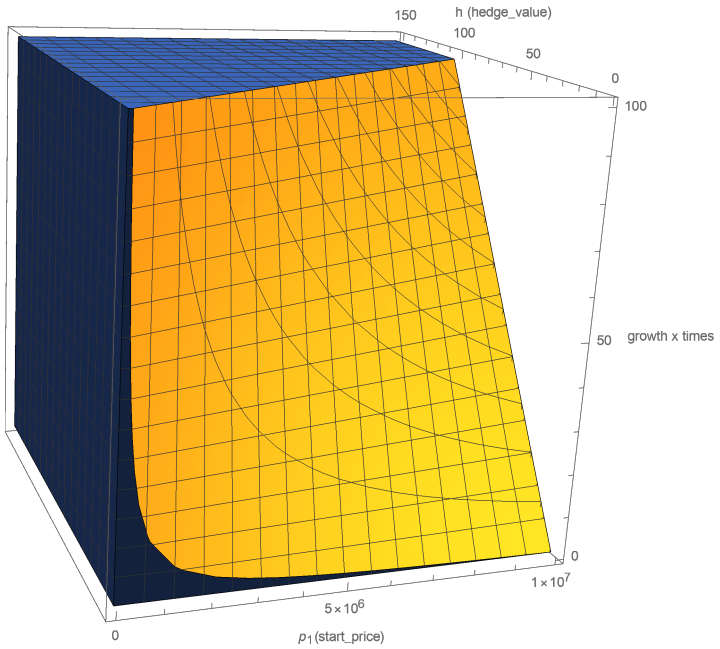

Figure 3: The shaded region indicates safe parameters regarding the upper price limit.

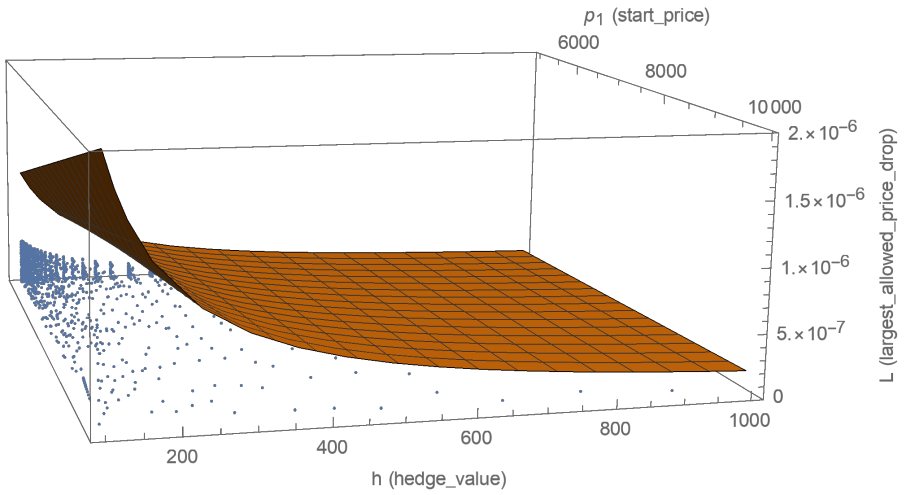

Figure 4: Comparison of of theoretical prediction to simulated contract data. Points represent simulated contract error and the plane is the theoretical prediction of the upper limit for error.