Unlockd Finance Protocol

Data Science Department

Mathematical Risk Framework

Model Proposal

by Alejandro Roige Vázquez

July 2022

Contents

1 Introduction

The present document will define the **Unlockd Finance Risk Framework**. We will begin by defining the Loan To Value Model by interpolating an Exponential Decay function to ensure a limit selling pressure is set should extreme volatility events occur.A formal definition of the functions will be written below and a full comprehensive explanation will follow.

1.1. Important considerations:

1.- The *low* boundary of *Upshot's Price Object*[8] will be taken as the price value for any asset to consider the *Median Relative Error* and therefore reduce the protocol risk.

2.- The *confidence* parameter on the *Price Object* should not be considered as it displays redundant information already shown in the *MRE* and considered in the lower boundary in the *low* attribute.

3.- All parameters are subject to further discussion and are not to be treated as definitive. Many of them which will be fixed in this document should be voted as Unlockd transitions to a DAO.

2 Loan To Value - LTV

We want to express the Loan To Value as a variable function of the number of assets we have in reserves and how volatile the market is.

$$\boxed{LTV(n_i) = \Psi(p) \cdot 0.4 \cdot e^{-\frac{log(\frac{0.4}{0.01})}{N} \cdot n_i}}$$

where

$$\begin{cases} n_i & = \text{the new i-th element you want to know the LTV for.} \\ N & = \text{total number of assets in a certain collection.} \\ \Psi_i & = \text{Unlockd's confidence function.} \\ p & = \text{latest } \textit{low} \text{ price available from appraisal protocols.} \end{cases}$$

To get n_i would be useful to keep track a **counter** of hoy many assets of a collection we have accepted in reserves. A pseudo-code proposal follows next:

```
function ltv(ni: number, N: number){
    return 0.4 * exp(((Math.log(0.4/0.01)) / N)*ni);
}
```

To do: Add the $\Psi_i(p)$ function proposal in Typescript code.

2.1.- Loan To Value Calculation - Assets Under Management Component:

There are a few considerations to make before jumping straight to the LTV formula construction:

- As NFT's of a single collection increase in reserves, new collection based loans should be heavily discouraged to limit the selling pressure in case of a massive liquidation black swan event.
- The initial LTV should be sufficient to attract new users above our competitors.
- Last item LTV should be practically zero.

As we pointed out previously, we want our model to have two main components. First let's see the variable number of asset decaying model that helps us limit the selling pressure in case of an unexpected volatility event. Fist let's have a look at a typical exponential decay model:

$$N(t) = N_0 \cdot e^{-r \cdot t}$$

Let N_0 be the initial value at $t = 0$, that is, our initial Loan To Value for a given asset. We can arbitrarily initially set this value at:

$$\boxed{N_0 = 0.4}$$

The only parameter left to discuss is the decay rate which can be calculated if we set a final value of N_k, where k is the number of elements in a certain collection, e.g:

$$k_{cryptopunks} = 10,000$$

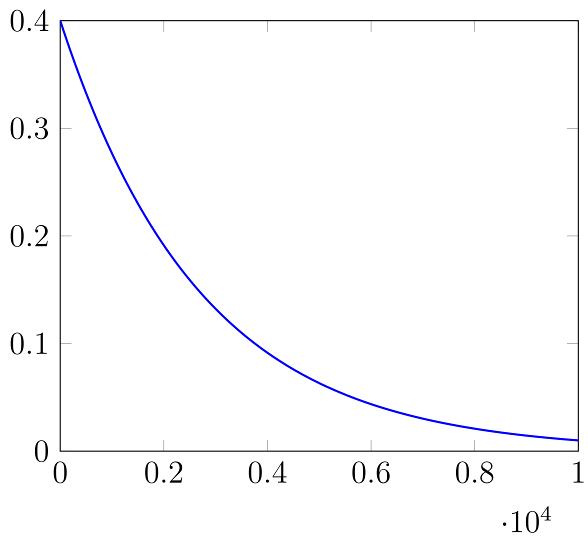
And then by interpolation the exponential decay function we can find r to be:

$$\begin{cases} N_1(t_1) = N_0 \cdot e^{-r \cdot t_1} \\ N_2(t_2) = N_0 \cdot e^{-r \cdot t_2} \end{cases} \implies r = \frac{log(\frac{N_1}{N_2})}{t_2 - t_1}$$

Then, using the data fixed for our *Cryptopunk's* example we obtain:

$$r = \frac{log(\frac{0.4}{0.01})}{10,000} \approx 3.69 \cdot 10^{-4}$$

A visual representation of the model can be seen in the following graph:



A more comprehensive understanding of the above function can be seen in the table:

Loan to Value by number of assets in reserves		
Assets in reserve	LTV	LTV (%)
0	0.4	40
1000	0.27657	27.66
2000	0.19123	19.12
3000	0.13222	13.22
4000	0.0914	9.14
5000	0.0632	6.32
6000	0.0437	4.37
7000	0.0302	3.02
8000	0.0208	2.08
9000	0.0144	1.44
10000	0.0099	0.99

2.2.- Loan To Value Calculation - Volatility Component:

For the volatility component we would like to build a model that takes *Implied Volatility Rank/Percentile* into consideration; nevertheless, the ecosystem is not yet liquid enough to provide these metrics accurately.

As such, an *Historic Volatility* calculation with adjusted error metrics will be used. As can be seen in [3] a formal definition for volatility is:

Volatility is a statistical measure of the dispersion of returns for a given security or market index. In most cases, the higher the volatility, the riskier the security. Volatility is often measured as either the standard deviation or variance between returns from that same security or market index.[3]

The steps for a full calculation with examples will be provided below as well as a discussion on why this process should ideally not be within the protocol smart-contracts.

Steps to calculate Historical Volatility (σ) for any NFT asset:

$$\begin{cases} \text{1.- Find the mean of the dataset.} \\ \text{2.- Calculate the deviation (difference between each value and the mean).} \\ \text{3.- Square the deviations (this eliminates negative values).} \\ \text{4.- Add all of squared deviations toghether.} \\ \text{5.- Divide the sum of squared deviations by number of data values.} \\ \text{6.- Take the square root of the variance (Standard Deviation).} \end{cases}$$

Let's see a worked example to make things a little bit more clear with a *Cryptopunk* NFT. Imagine we have an array of historic prices for a certain asset i. For the sake of the example and further implementation concerns let's also suppose we have this array within a *prices* matrix where the first element filters by collection and the second by token ID. In the example the *Cryptopunks* collection will have $index = 1$. All the prices will be in ETH, not in wei, this is an important remark.

$$prices[1][i] = [100, 98, 97, 101, 102]$$

$$\begin{cases} \text{1.- } \mu = \frac{\sum_{i=1}^{n} x_i}{n} = 99.6 \\ \\ \text{2.- } S = (x_i - \mu) = [0.4, -1.6, -2.6, 1.4, 2.4] \\ \\ \text{3.- } S^2 = (x_i - \mu)^2 = [0.16, 0.0256, 6.76, 1.96, 5.76] \\ \\ \text{4.- } \sum_{i=1}^{n} S^2 = 14,6656 \\ \\ \text{5.- } \sigma^2 = \frac{\sum_{i=1}^{n} S^2}{n} = 2.93312 \\ \\ \text{6.- } \sigma = \sqrt{\sigma^2} = 1.7126354 \end{cases}$$

This result $\boxed{\sigma \approx 1.71}$ is a measure of risk and shows how values are spread out around the average price. It gives us an idea of how much far price may deviate from the average. We can now normalize any value we get from *Upshot*[8] as:

$$Z_i = \frac{x_i - \mu}{\sigma}$$

For example, let's supose we make an API Call to Upshot in order to get the Latest Price and it returns the following Price Object:

```
{
    "assetId": "0x64f...6d",
    "estimatedPrice": 101.5,
    "low": 101,
    "high": 102,
    "confidence": 0.8,
    .
    .
    .
    "medianRelativeError": 0.5,
    "currency" :(Object),
    "asset": (Object),
}
```

As mentioned on the introduction the *low* parameter will be used for the model in order to reduce the risk. The *confidence* parameter is redundant with the *MRE.*

Then Unlockd's confidence function will be defined by the normalized price of the asset provided by the external appraisal protocols as:

$$\Psi_i(p) = Z_{(x_i=p)} = \frac{p - \mu}{\sigma} = \frac{101 - 99.6}{1.71} \approx 0.82$$

That will be our own *confidence* parameter $\boxed{\Psi_i(p) \approx 0.82}$ to evaluate how much *LTV* we provide our clients with.

Note that $\Psi_i(p) \in [0, 1]$.

A detailed explanation on why data centralization is needed for the deviation process can be added here if necessary with all due saved gas fee costs estimations.

3 Liquidation Grace Period

The Grace Period should be a decending function that decreases as the difference between the original price we gave a loan at and the liquidation price grows.

$$\boxed{GP(p_{loan}, p_{liq}) = \frac{24}{p_{loan}} \cdot p_{liq}}$$

$$\begin{cases} p_{loan} & = \text{the original price we gave a loan at for a specified asset.} \\ p_{liq} & = \text{the current price at we have to begin a liquidation process.} \end{cases}$$

It should be noted that p_{loan} is the price taking into account the LTV calculations, not just the value return by our appraisal providers.

```
function gp(ploan: number, pliq: number){
    return (24/ploan) * pliq;
}
```

Grace Period Calculation:

An interval between 24h and 0h will be set depending on the price delta between the actual liquidation price and the original price at which the loan was given. Let Δ_p be the difference of prices and $\Delta_{p(\%)}$ the percentual difference with the initial loan price.

$$\Delta_p = p_{loan} - p_{liq}$$
$$\Delta_{p(\%)} = \frac{p_{liq}}{p_{loan}} \cdot 100$$

The basic idea behind the function we want to build is the following: we have at maximum a grace period of 24h and it descends as the liquidation price does, that is, we want to get rid of an asset faster as its price gets lower.

Let $X = \mathbb{R}$, and $\Delta_p \in X$ we need to create a function that maps the difference between prices into $Y = [0, 24]$.

$$\begin{aligned} \varphi\colon \quad X &\longrightarrow Y \\ x &\longmapsto y \end{aligned}$$

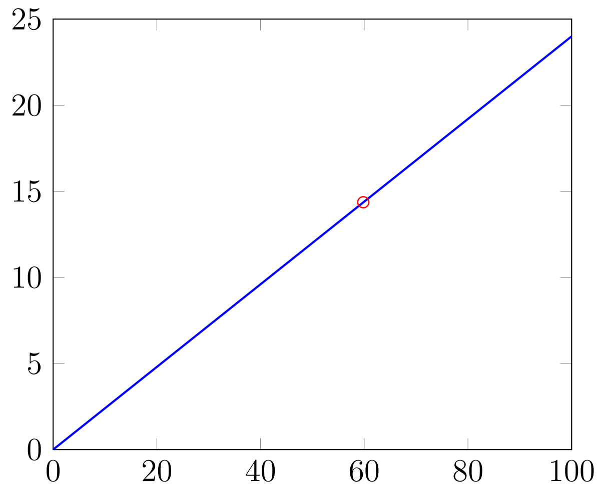
The idea is that this morphism might be a composition of two simpler ones $\varphi(\Delta_p) = f \circ g$ such that

$$[0, +\infty) \xrightarrow{f} [0, 100] \xrightarrow{g} [0, 24]$$

In f we are just trying to limit the price in a set interval and then in g converting it into the liquidation grace period. Therefore, the simple linear interpolation with p_{liq} as the independent variable should be enough.

$$\boxed{\varphi(p_{liq}) = \frac{24}{p_{loan}} \cdot p_{liq}}$$

Let's continue with the *Punk* example so it is easier to understand. Let's say we gave a loan for $100ETH$ and we had to liquidate it because the price is down a 40% so it is now $p_{liq} = 60ETH$. With the function defined above we would have:



So we would have a grace period of $\varphi(p_{liq}) = \frac{24}{100} \cdot 60 = \boxed{14.4h}$

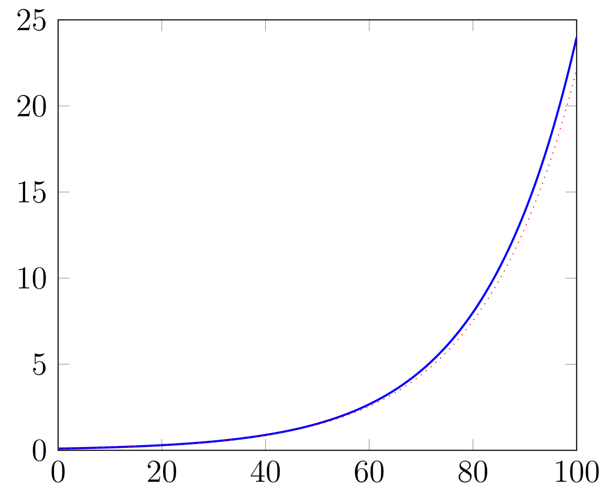
Note that $\Delta_{p(\%)} = 60\%$.

Grace Period as a function of $\Delta_{p(\%)}$	
$\Delta_{p(\%)}$	Grace Period (h)
0	0
10	0.27657
20	2.4
30	4.8
40	9.6
50	12
60	14.4
70	16.8
80	19.2
90	21.6
100	24

An exponential decay function in this case would not make sense as can be seen below. For this purpose we only need to do a basic exponential decay interpolation with a positive decay rate.

This calculations can be explicitly written but I don't think it is necessary.

$$g(p_{liq}) = 0.09945288962 \cdot e^{\frac{ln(\frac{24}{0.1})}{100-0.1} \cdot p_{liq}} \approx 0.1 \cdot e^{0.054 \cdot p_{liq}}$$



As can be seen an exponential approach would liquidate the assets too soon leaving no room for users to repay their loans in time.

References:

[1] **Estimating the Volatility in the Black-Scholes Formula**, *Rebecca Keenan, Rachel Lane, et al.*: https://www.valpo.edu/mathematics-statistics/files/2015/07/Estimating-the-Volatility-in-the-Black-Scholes-Formula.pdf.

[2] **IV Rank vs. IV Percentile: Which is Better?**, *ProjectFinance, April 14, 2022*: https://www.projectfinance.com/iv-rank-percentile/

[3] **Volatility**, *Adam Hayes, Investopedia, October 30, 2021*: https://www.investopedia.com/terms/v/volatility.asp

[4] **An Historical Perspective On The EURUSD Volatility Surface**, *Kevin Mulhern*: https://seekingalpha.com/article/445121-an-historical-perspective-on-the-eurusd-volatility-surface

[5] **What is Implied Volatility**, *Fidelity International*: https://www.fidelity.com.sg/beginners/what-is-volatility/implied-vs-historical-volatility

[6] **Calculate Exponential Decay**, *Mathematics Exchange*: https://math.stackexchange.com/questions/728530/calculate-exponential-decay

[7] **Interpolation of exponential-type functions on a uniform grid by shifts of a basis function** , *Jeremy Levesley, et al.*: https://www.aimsciences.org/article/exportPdf?id=684c7ace-4d7b-46b9-a71b-d99a75a73a5f

[8] **Upshot Analytics**, *Price Object*: https://docs.upshot.xyz/upshot-api/api/response-objects/price-object

[9] **Normalization Formula: How To Use It on a Data Set**, *Indeed*: https://www.indeed.com/career-advice/career-development/normalization-formula

[10] **Volatility - Finance**, *Wikipedia*: https://en.wikipedia.org/wiki/Volatility$_(finance)$

[11] **Rate of return (logarithmic returns)**, *Wikipedia*: https://en.wikipedia.org/wiki/Rate$_of_returnLogarithmic$