Uniswap Math with Fees

Intro

The purpose of this document is to provide derivation of the math involving fees in the Uniswap V1 code, although the same principals apply to Uniswap V2. I have seen many explainers on the basic math behind constant product AMMs, but rarely any covering the math involving fees.

I will not be providing an explainer on the basics of constant product AMMs here. If you need an overview of the basics, I recommend watching the AMM video from one of my favorite YouTube channels, WhiteboardCrypto: https://www.youtube.com/watch?v=1PbZMudPP5E&t=6s&ab_channel=WhiteboardCrypto

Fair warning, I do break down much of the algebra step by step in order to reach the broadest audience. The math may be unnecessarily detailed for those with math backgrounds.

The Formalized Model

Overview

Here is a snippet taken from the paper that formalizes the mathematical model on which Uni V1 and other CPAMMs are based. It is technically not the whitepaper, but the formalized model of the constant product AMM. The Uniswap whitepaper lists it as a resource under “Formalized Model”. I have provided a link to the paper in the sources section.

Here are the equations that do not involve a fee:

$$x' = x + \Delta x = (1+\alpha)x = \frac{1}{1-\beta}x$$

$$y' = y - \Delta y = \frac{1}{1+\alpha}y \quad = (1-\beta)y$$

where $\alpha = \frac{\Delta x}{x}$ and $\beta = \frac{\Delta y}{y}$. Also, we have:

$$\Delta x = \frac{\beta}{1-\beta}x$$

$$\Delta y = \frac{\alpha}{1+\alpha}y$$

Allow me to elaborate where the variables are coming from. There is a liquidity pool of X tokens and Y tokens. There are x amount of X tokens in the pool, and y amount of Y tokens. After trading Δx X tokens for Δy Y tokens, there are x' amount of X tokens in the pool, and y' amount of Y tokens.

Let's break it down:

- X tokens are going into the pool, Y tokens are leaving the pool
- x and y are the amount of tokens in the pool BEFORE a trade
- x' and y' are the amount of tokens in the pool AFTER a trade
- Δx is the amount of X tokens being traded INTO the pool
- Δy is the amount of Y tokens being traded OUT of the pool

With a fee, things get a bit more complicated. Here are the equations involving fees, on which the code is based.

Now consider a fee for each token trade. Let $0 \leq \rho < 1$ be a fee, e.g., $\rho = 0.003$ for 0.3% fee schedule.

$$x'_\rho = x + \Delta x = (1+\alpha)x = \frac{1+\beta(\frac{1}{\gamma}-1)}{1-\beta}x$$

$$y'_\rho = y - \Delta y = \frac{1}{1+\alpha\gamma}y = (1-\beta)y$$

where $\alpha = \frac{\Delta x}{x}$, $\beta = \frac{\Delta y}{y}$, and $\gamma = 1-\rho$. Also, we have:

$$\Delta x = \frac{\beta}{1-\beta} \cdot \frac{1}{\gamma} \cdot x$$

$$\Delta y = \frac{\alpha\gamma}{1+\alpha\gamma} \cdot y$$

Here, all definitions of variables are the same, but there is one extra term - γ (pronounced gamma). γ is simply equal to 1 minus the fee. I personally would prefer to use the term "f" for "fee," which is set to .003 (0.3%) for Uniswap. However, I see why they did this, as the math looks slightly neater when using γ. Here, γ is equal to .997 (99.7%). This is the percentage of Δx that is actually being used to trade.

Axiomatic Equations

Where did these equations come from? Let's start by discussing the 3 equations on which all of the math is based

(1) $$xy = (x + \gamma\Delta x)y'$$

(2) $$x' = x + \Delta x$$

(3) $$y' = y - \Delta y$$

(2) and (3) are pretty simple to explain. It's important to remember that X tokens are defined as the tokens being deposited into the pool, and Y tokens are defined as the tokens leaving the pool.

(2) is expressing that the amount of X tokens in the pool after the trade (x') is equal to the amount of X tokens in the pool before the trade (x) plus the X tokens deposited into the pool (Δx).

(3) is expressing that the amount of Y tokens in the pool after the trade (y') is equal to the amount of Y tokens in the pool before the trade (y) minus the Y tokens released from the pool (Δy).

Now let's go over (1), it's a bit more complicated. In a constant product market maker without fees, the common expression that trades are based on is

$$xy = x'y' = k$$

with x' calculated using (2), y' calculated using (3). *With fees involved, 99.7% of the deposit is being used to trade, although the fee is still being deposited into the pool.* So although 99.7% of the deposit is used to trade, 100% of the deposit is still being put into the pool. The trade is calculated as if the X pool will be increasing by only 99.7% of the deposit, hence (1). Basically, during a trade, instead of using (2) to calculate x', this is used:

$$x' = x + \gamma\Delta x$$

using only 99.7% of the deposit to calculate the new amount of X tokens in the pool before making the trade based off of the constant product formula.

Now that we understand where these equations come from, let's derive the equations in the formalized model for calculating Δx and Δy

Derivation of Δx

First we will derive the equation for Δx. This is when the user knows the quantity of token Y they would like to receive, and need to know how much of token X it will require.

We are essentially trying to find how much of token X we will need (Δx) to deposit to receive Δy. In other words, we must find Δx in terms of Δy:

$$\Delta x(\Delta y) = ?$$

To reference the model, we need to derive

$$\Delta x = \frac{\beta}{1 - \beta}\frac{1}{\gamma}x$$

where

$$\beta = \frac{\Delta y}{y}$$

Let's start with equation (1)

(1) $$xy = (x + \gamma\Delta x)y'$$

rearranging (1), we get

$$x + \gamma\Delta x = \frac{xy}{y'}$$

rearranging further

$$\gamma\Delta x = \frac{xy}{y'} - x$$

rearranging further

$$\Delta x = (\frac{xy}{y'} - x)\frac{1}{\gamma}$$

simplifying, we get

(4) $$\Delta x = (\frac{y}{y'} - 1)\frac{x}{\gamma}$$

Now we can plug in y' from (3) into (4)

(5) $$\Delta x = (\frac{y}{y - \Delta y} - 1)\frac{x}{\gamma}$$

Next we will use the following relationship to further simplify this

(6) $$1 = \frac{y - \Delta y}{y - \Delta y}$$

plugging in 1 from (6) into (5), we get

$$\Delta x = \left(\frac{y}{y - \Delta y} - \frac{y - \Delta y}{y - \Delta y}\right)\frac{x}{\gamma}$$

$$= \left(\frac{y - y + \Delta y}{y - \Delta y}\right)\frac{x}{\gamma}$$

Which simplifies to

(7) $$\Delta x = \frac{\Delta y}{y - \Delta y}\frac{1}{\gamma}x$$

Next, we'll make use of the following relationship:

(8) $$1 = \frac{\frac{1}{y}}{\frac{1}{y}}$$

Multiplying (7) by 1 in the form of (8), we get

$$\Delta x = \frac{\Delta y \frac{1}{y}}{(y - \Delta y)\frac{1}{y}} \frac{1}{\gamma} x$$

simplifying, we get

$$\text{(9)} \qquad \Delta x = \frac{\frac{\Delta y}{y}}{(1 - \frac{\Delta y}{y})} \frac{1}{\gamma} x$$

if we let

$$\text{(10)} \qquad \beta = \frac{\Delta y}{y}$$

plugging (10) into (9), we get

$$\text{(11)} \qquad \Delta x = \frac{\beta}{1 - \beta} \frac{1}{\gamma} x$$

Which is the equation from the formalized model

Derivation of Δy

Next we'll derive the equation for Δy. This is when the user knows the quantity of token X they would like to deposit, and would like to know how much of token Y they will receive.

We are essentially trying to find how much of token Y we will receive (Δy) after depositing Δx. In other words, we must find Δy in terms of Δx:

$$\Delta y(\Delta x) = ?$$

To reference the formalized model, we need to derive

$$\Delta y = \frac{\alpha\gamma}{1 + \alpha\gamma} y$$

where

$$\alpha = \frac{\Delta x}{x}$$

Looking back at equation (1) and (3):

(1) $$xy = (x + \gamma\Delta x)y'$$

(3) $$y' = y - \Delta y$$

rearranging (3), we get

(12) $$\Delta y = y - y'$$

rearranging (1), we get

(13) $$y' = \frac{xy}{x + \Delta x\gamma}$$

plugging y' from (13) into (12), we get

$$\Delta y = y - \frac{xy}{x + \Delta x\gamma}$$

Which simplifies to

(14) $$\Delta y = (1 - \frac{x}{x + \Delta x\gamma})y$$

To simplify this, we will need to make use of the following relationship

(15) $$1 = \frac{x + \Delta x\gamma}{x + \Delta x\gamma}$$

plugging (15) into (14), we get

$$\Delta y = \left(\frac{x + \Delta x\gamma}{x + \Delta x\gamma} - \frac{x}{x + \Delta x\gamma}\right)y$$

$$= \frac{x + \Delta x\gamma - x}{x + \Delta x\gamma}y$$

Which simplifies to

(16) $$\Delta y = \frac{\Delta x\gamma}{x + \Delta x\gamma}y$$

Next we'll make use of the following relationship

(17) $$1 = \frac{\frac{1}{x}}{\frac{1}{x}}$$

Multiplying (16) by 1 in the form of (17), we get

$$\Delta y = \frac{\Delta x\gamma\frac{1}{x}}{(x + \Delta x\gamma)\frac{1}{x}}y$$

simplifying, we get

(18) $$\Delta y = \frac{\frac{\Delta x}{x}\gamma}{(1 + \frac{\Delta x}{x}\gamma)}y$$

if we let

(19) $$\alpha = \frac{\Delta x}{x}$$

plugging (19) into (18), we get

$$(20) \qquad \Delta y = \frac{\alpha\gamma}{1 + \alpha\gamma} y$$

Which is the equation from the formalized model

Relation of Model and Code

Overview

The two equations that appear in the code are the calculations for Δx and Δy. During a trade, a user deposits Δx X tokens into the pool, and receives Δy Y tokens. A user usually falls under 1 of 2 situations:

A. The user knows how much of token X they want to deposit(Δx), and would like to know how much of token Y they will receive (Δy) for depositing Δx.

B. The user knows how much of token Y they want to receive(Δy), and would like to know how much of token X they must deposit (Δx) to receive Δy.

In the code, these 2 situations are handled with the functions getInputPrice and getOutputPrice. Here is are the snippets of code involving those functions:

```
def getInputPrice(input_amount: uint256, input_reserve: uint256, output_reserve: uint256) -> uint256:
    assert input_reserve > 0 and output_reserve > 0
    input_amount_with_fee: uint256 = input_amount * 997
    numerator: uint256 = input_amount_with_fee * output_reserve
    denominator: uint256 = (input_reserve * 1000) + input_amount_with_fee
    return numerator / denominator
```

```
def getOutputPrice(output_amount: uint256, input_reserve: uint256, output_reserve: uint256) -> uint256:
    assert input_reserve > 0 and output_reserve > 0
    numerator: uint256 = input_reserve * output_amount * 1000
    denominator: uint256 = (output_reserve - output_amount) * 997
    return numerator / denominator + 1
```

The getInputPrice function addresses situation A. It takes an input the amount of X tokens the user would like to deposit (Δx), and outputs the amount of Y tokens that the user will receive (Δy) for depositing Δx.

The getOutputPrice function addresses situation B. It takes an input the amount of token Y the user would like to receive (Δy), and outputs the amount of token X that the user must deposit (Δx) to receive Δy.

Broken down:
- getInputPrice calculates Δy, given Δx
- getOutputPrice calculates Δx, given Δy

getInputPrice

Let's take a look at a smaller portion of the getInputPrice function and create an equation represented by the variables given in the model.

```
input_amount_with_fee: uint256 = input_amount * 997
numerator: uint256 = input_amount_with_fee * output_reserve
denominator: uint256 = (input_reserve * 1000) + input_amount_with_fee
```

Let's break down the variables in the code in term of the variables in the model:
- $input\ amount = \Delta x$
- $997 = 1000 \cdot \gamma$
- $output\ reserve = y$
- $input\ reserve = x$

You may be wondering why there is 997 instead of .997, and where 1000 is coming from. The EVM does not support decimals. To avoid decimals, there is a factor of 1000 applied to the numerator and denominator. This measure was taken simply to apply the fee, it does not effect the final calculation because it is essentially the same as multiplying by 1:

$$1 = \frac{1000}{1000}$$

Recreating the math in the code in terms of the variables from the model, we get:

$$\Delta y = \frac{997 \cdot \Delta x \cdot y}{1000x + 997\Delta x}$$

dividing out the 1000, we get

(a) $$\Delta y = \frac{\gamma \Delta x \cdot y}{x + \gamma \Delta x}$$

This is the same equation in the model, just not in terms of α and β.

In the model, we have

$$\Delta y = \frac{\alpha \gamma}{1 + \alpha \gamma} y$$

where

$$\alpha = \frac{\Delta x}{x}$$

getting rid of “α” and instead expressing in terms of “x” and “Δx”, we get

(b) $$\Delta y = \frac{\frac{\Delta x}{x} \gamma}{1 + \frac{\Delta x}{x} \gamma} y$$

using the relationship

(c) $$1 = \frac{x}{x}$$

we multiply (b) by 1 in terms of (c) and get

$$\Delta y = \frac{\Delta x \gamma}{x + \Delta x \gamma} y$$

which is equivalent to (a), the equation used in the code.

(a) $$\Delta y = \frac{\gamma \Delta x \cdot y}{x + \gamma \Delta x}$$

getOutputPrice

Let's take a look at a smaller portion of the getOutputPrice function and create an equation represented in the variables given in the model.

```
numerator: uint256 = input_reserve * output_amount * 1000
denominator: uint256 = (output_reserve - output_amount) * 997
```

Let's break down the variables in the code in term of the variables in the model:

- $input\ reserve = x$
- $output\ amount = \Delta y$
- $output\ reserve = y$
- $997 = 1000 \cdot \gamma$

Recreating the math in the code in terms of the variable from the model, we get:

$$\Delta x = \frac{1000 \cdot x \cdot \Delta y}{(y - \Delta y)997}$$

dividing out the 1000, we get

$$\text{(d)} \qquad \Delta x = \frac{x \cdot \Delta y}{(y - \Delta y)\gamma}$$

This is the same equation in the model, just not in terms of α and β.

In the model, we have

$$\Delta x = \frac{\beta}{1 - \beta}\frac{1}{\gamma}x$$

where

$$\beta = \frac{\Delta y}{y}$$

getting rid of “β” and instead expressing in terms of “y” and “Δy”, we get

(e) $$\Delta x = \frac{\frac{\Delta y}{y}}{1 - \frac{\Delta y}{y}} \frac{1}{\gamma} x$$

using the relationship

(f) $$1 = \frac{y}{y}$$

we multiply (e) by 1 in terms of (f) and get

$$\Delta x = \frac{\Delta y}{y - \Delta y} \frac{1}{\gamma} x$$

which is equivalent to (d), the equation used in the code

(d) $$\Delta x = \frac{x \cdot \Delta y}{(y - \Delta y)\gamma}$$

Extra Derivations

Overview

I have provided this section for those interested in the derivation of the remaining equations in the snippet of the formalized model paper. However, these equations are not used in the code. I will only provide derivation of the following 2 equations, as the others are quite self explanatory given the values of α and β.

(21) $$x' = \frac{1 + \beta\,(\frac{1}{\gamma} - 1)}{1 - \beta} x$$

(22) $$y' = \frac{1}{1 + \alpha\gamma} y$$

Deriving x' in terms of Δy

Looking back at equation (2) and (13):

(2) $$x' = x + \Delta x$$

(11) $$\Delta x = \frac{\beta}{1 - \beta}\frac{1}{\gamma}x$$

Plugging Δx from (13) into (2), we get

$$x' = x + \frac{\beta}{1 - \beta}\frac{1}{\gamma}x$$

$$= (1 + \frac{\beta}{(1 - \beta)\gamma})x$$

$$= (\frac{(1 - \beta)\gamma}{(1 - \beta)\gamma} + \frac{\beta}{(1 - \beta)\gamma})x$$

$$= \frac{(1 - \beta)\gamma + \beta}{(1 - \beta)\gamma}x$$

$$= \frac{\gamma - \beta\gamma + \beta}{(1 - \beta)\gamma}x$$

dividing out the γ, we get

$$= \frac{1 - \beta + \frac{\beta}{\gamma}}{1 - \beta}x$$

simplifying, we get

$$x' = \frac{1 + \beta(\frac{1}{\gamma} - 1)}{1 - \beta}x$$

which is the equation in the formalized model

Deriving y' in terms of Δx

Looking back at equation (3) and (14):

(3) $$y' = y - \Delta y$$

(20) $$\Delta y = \frac{\alpha\gamma}{1 + \alpha\gamma} y$$

Plugging Δy from (20) into (3), we get

$$y' = y - \frac{\alpha\gamma}{1 + \alpha\gamma} y$$

$$= (1 - \frac{\alpha\gamma}{1 + \alpha\gamma}) y$$

$$= (\frac{1 + \alpha\gamma}{1 + \alpha\gamma} - \frac{\alpha\gamma}{1 + \alpha\gamma}) y$$

$$= (\frac{1 + \alpha\gamma - \alpha\gamma}{1 + \alpha\gamma}) y$$

which simplifies to

$$y' = \frac{1}{1 + \alpha\gamma} y$$

which is the equation in the formalized model

Conclusion

I hope with was insightful and helped clear up the basics of the Uniswap math involving the fee. Feel free to reach out if you have any comments, questions, or suggestions.

Sources

1. Formalized Model Paper

https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf

2. Uniswap V1 Exchange Code

https://github.com/Uniswap/v1-contracts/blob/master/contracts/uniswap_exchange.vy