Uniswap V3 Pricing Review for Lenders

In[1]:=
```
SeedRandom["0xf2ecf6f0aaf635b6df6404485e749dcad5be4dd1e5bf7b9aac3f06f1245da0f1"]
```

Out[1]=
```
RandomGeneratorState[ Method: ExtendedCA
                      State hash: 4599521046699082755 ]
```

Motivation

With the creation of Uniswap, thorough stochastic pricing analysis has been reviewed by Bardoscia and Milionis describing it's spot pricing dynamics. Since it's release, only a few protocols have been created to address leveraged perpetual options using Uniswap V3 pricing dynamics, effectively creating a "loan" based on backed collateral for position holders. Below is a review of simulation using Mathematica to generate empirical risk profiles for V3 positions with standard techniques in Stochastic Calculus. If users are to appropriately price loans for Uniswap V3 positions, simulating worst case lower bounds on losses is important and should be taken into consideration with other methods such as backtesting.

Stochastic Calculus Review

Ito's Lemma

Ito' s Lemma allows us to model functions whose variables are random values . For a function $F(X, t)$ where X is a random variable and t is time, Ito's lemma gives us a way to model a probability density function for the future time t. We call the variables a "process", and the result of using Ito's Lemma a new process. The rest of the system utilizes Ito's Lemma under the hood to derive functions of random variables. It's not essential to understand these Lemmas, but they are important in stochastic modeling.

Single Variable Ito's Lemma

Below is the main statement for Ito's Lemma.

In[2]:=

14.2.3 Ito's Lemma

The central tool in stochastic differential equations is **Ito's lemma**, which basically says that a smooth function of an Ito process is itself an Ito process.

THEOREM 14.2.1 *Suppose that* $f: R \to R$ *is twice continuously differentiable*[1] *and that* $dX = a_t\, dt + b_t\, dW$. *Then* $f(X)$ *is the Ito process*

$$f(X_t) = f(X_0) + \int_0^t f'(X_s)\, a_s\, ds + \int_0^t f'(X_s)\, b_s\, dW + \frac{1}{2} \int_0^t f''(X_s)\, b_s^2\, ds$$

for $t \geq 0$.

In differential form, Ito's lemma becomes

$$d f(X) = f'(X)\, a\, dt + f'(X)\, b\, dW + \frac{1}{2} f''(X)\, b^2\, dt. \qquad (14.9)$$

Out[2]=

14.2.3 Ito's Lemma

The central tool in stochastic differential equations is **Ito's lemma**, which basically says that a smooth function of an Ito process is itself an Ito process.

THEOREM 14.2.1 *Suppose that* $f: R \to R$ *is twice continuously differentiable*[1] *and that* $dX = a_t\, dt + b_t\, dW$. *Then* $f(X)$ *is the Ito process*

$$f(X_t) = f(X_0) + \int_0^t f'(X_s)\, a_s\, ds + \int_0^t f'(X_s)\, b_s\, dW + \frac{1}{2} \int_0^t f''(X_s)\, b_s^2\, ds$$

for $t \geq 0$.

In differential form, Ito's lemma becomes

$$d f(X) = f'(X)\, a\, dt + f'(X)\, b\, dW + \frac{1}{2} f''(X)\, b^2\, dt. \qquad (14.9)$$

Multivariable Ito's Lemma

When modeling over multiple variables including time, we need the Multivariable Ito's Lemma.

In[3]:=

THEOREM 14.2.3 *Let* $W_1, W_2, \ldots, W_m$ *be Wiener processes and let* $X \equiv (X_1, X_2, \ldots, X_m)$ *be a vector process. Suppose that* $f : R^m \to R$ *is twice continuously differentiable and* X_i *is an Ito process with* $dX_i = a_i\, dt + b_i\, dW_i$. *Then* $df(X)$ *is the following Ito process,*

$$df(X) = \sum_{i=1}^{m} f_i(X)\, dX_i + \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{m} f_{ik}(X)\, dX_i\, dX_k,$$

with the following multiplication table:

$\times$	dW_i	dt
dW_k	$\rho_{ik}\, dt$	0
dt	0	0

Here, ρ_{ik} *denotes the correlation between* dW_i *and* dW_k.

Out[3]=

THEOREM 14.2.3 *Let* $W_1, W_2, \ldots, W_m$ *be Wiener processes and let* $X \equiv (X_1, X_2, \ldots, X_m)$ *be a vector process. Suppose that* $f : R^m \to R$ *is twice continuously differentiable and* X_i *is an Ito process with* $dX_i = a_i\, dt + b_i\, dW_i$. *Then* $df(X)$ *is the following Ito process,*

$$df(X) = \sum_{i=1}^{m} f_i(X)\, dX_i + \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{m} f_{ik}(X)\, dX_i\, dX_k,$$

with the following multiplication table:

$\times$	dW_i	dt
dW_k	$\rho_{ik}\, dt$	0
dt	0	0

Here, ρ_{ik} *denotes the correlation between* dW_i *and* dW_k.

Uniswap V3 Stochastic Analysis

Now that we have Ito’s Lemma defined, we can use the program to determine some average case and variances for Uniswap’s value functions.

Uniswap State Equations

Below, are some necessary equations for Uniswap V3 analysis. In these equations, treat Y as a cash-like stable numeraire. Prices are in terms of the numeraire per X token (USD per ETH). p_a represents the lower bound of the position, and p_b is the upper bound.

In[4]:=

$$\mathtt{liquidityEquation} = \left(x + \frac{L}{\sqrt{p_d}}\right)\left(y + L\sqrt{p_c}\right) == L^0;$$

$$\mathtt{tokensGivenLiquidity} = \left\{x \to L\,\frac{\left(\sqrt{p_d} - \sqrt{p}\right)}{\sqrt{p} * \sqrt{p_d}},\ y \to L\left(\sqrt{p} - \sqrt{p_c}\right)\right\};$$

$$\mathtt{liquidityGivenTokens} = \left\{L_z \to x * \frac{\left(\sqrt{p} * \sqrt{p_d}\right)}{\sqrt{p_d} - \sqrt{p}},\ L_{\grave{A}} \to \frac{y}{\sqrt{p} - \sqrt{p_c}}\right\};$$

In[7]:=

$$\begin{aligned}
&\mathtt{Liquidity}[\mathtt{lowerPriceBound_},\ \mathtt{upperPriceBound_},\ \mathtt{currentPrice_},\ \mathtt{total_}] := \\
&\quad \mathtt{Solve}\Big[\\
&\qquad \Big(L_z == L_{\grave{A}}\ /.\ \mathtt{liquidityGivenTokens}\ /.\ \{x \to \mathtt{holdX},\ y \to \mathtt{holdY}, \\
&\qquad\qquad p \to \mathtt{currentPrice},\ p_c \to \mathtt{lowerPriceBound},\ p_d \to \mathtt{upperPriceBound}\}\Big)\ \&\& \\
&\qquad \mathtt{holdX} > 0\ \&\&\ \mathtt{holdY} > 0\ \&\& \\
&\qquad \mathtt{total} == \mathtt{holdX} * \mathtt{currentPrice} + \mathtt{holdY},\ \{\mathtt{holdX},\ \mathtt{holdY}\}\Big]\ // \\
&\quad \left(\left\{x \to \mathtt{holdX},\ y \to \mathtt{holdY},\ L \to \frac{\mathtt{holdY}}{\sqrt{\mathtt{currentPrice}} - \sqrt{\mathtt{lowerPriceBound}}}\right\}\ /.\ \#\right)\&;
\end{aligned}$$

In[8]:=

```
Liquidity[1600, 1700, 1628, 10 000] // N
1628 * x + y /. %
```

Out[8]= {{x → 4.37661, y → 2874.87, L → 8249.71}}

Out[9]= {10 000.}

In[10]:=

$$\begin{aligned}
&\mathtt{originalValue} = \\
&\quad x * p + y\ /.\ \mathtt{tokensGivenLiquidity}\ /.\ \{p_c \to \mathtt{lower},\ p_d \to \mathtt{higher},\ p \to \mathtt{startPrice}\}; \\
&\mathtt{currentValue} = x * p + y\ /.\ \mathtt{tokensGivenLiquidity}\ /.\ \{p_c \to \mathtt{lower},\ p_d \to \mathtt{higher}\}; \\
&\mathtt{IL} = \frac{\mathtt{currentValue} - \mathtt{originalValue}}{\mathtt{originalValue}}; \\
&\mathtt{humanReadable} = \{\mathtt{lower} \to p_c,\ \mathtt{higher} \to p_d,\ \mathtt{startPrice} \to p_{fl}\}; \\
&\mathtt{IL}\ /.\ \mathtt{humanReadable}\ //\ \mathtt{Simplify};
\end{aligned}$$

Pricing Derivations for Impermanent Loss

Taking a look at a particular example, we can derive some interesting results with the Uniswap math. Below is a close enough situation to reality using the curves. The block directly below is just a setup

In[15]:=
```
ethDailyVol = 0.0034;
ethMeanYearly = 0.1;
dailyETHmean = ethMeanYearly / 365;
currentPrice = 1628;
lowerBound = 1600;
upperBound = 1700;
initialValue = 10 000;
currentLiquidityParams =
 Liquidity[lowerBound, upperBound, currentPrice, initialValue] // N
```

Out[22]= {{x → 4.37661, y → 2874.87, L → 8249.71}}

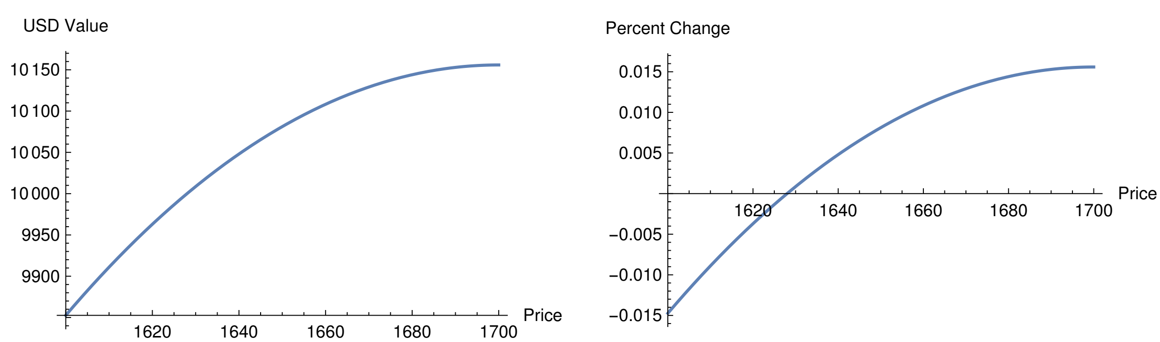
Plotting to Understand Value Curves

Let’s load the example into the curves and take a look at them to determine value action across price.

In[23]:=
```
valueCurve = currentValue /. currentLiquidityParams;
```

In[24]:=
```
GraphicsGrid[{{
   Plot[valueCurve /.
      {lower → lowerBound, higher → upperBound , startPrice → currentPrice},
     {p, lowerBound, upperBound}, AxesLabel → {"Price", "USD Value"}],
   Plot[
    IL /. {lower → lowerBound, higher → upperBound , startPrice → currentPrice},
    {p, lowerBound, upperBound}, AxesLabel → {"Price", "Percent Change"}]
  }}]
```

Out[24]=



Closed Form Analysis

Because Wolfram acts symbolically, we can derive some processes directly and compute means and variances.

In[25]:=
```
procPVOpen = TransformedProcess[currentValue /. {p → p[t]},
   p ≈ GeometricBrownianMotionProcess[μ, σ, S],
   t];
```

In[26]:=
```
meanProcPVOpen = Mean[procPVOpen[t]];
varianceProcPVOpen = Variance[procPVOpen[t]];
```

In[28]:=
```
TableForm[{
  {"Mean Function", meanProcPVOpen /. {higher → p_d, lower → p_c}},
  {"Variance", varianceProcPVOpen /. {higher → p_d, lower → p_c}}
 }]
```

Out[28]//TableForm=

Mean Function	$L\left(2 e^{\frac{1}{8} t\left(4 \mu-\sigma^2\right)} \sqrt{S}-\sqrt{p_a}-\frac{e^{t \mu} S}{\sqrt{p_b}}\right)$
Variance	$\frac{e^{t\left(\mu-\frac{\sigma^2}{4}\right)}\left(-1+e^{\frac{t \sigma^2}{4}}\right) L^2 S\left(\left(e^{t\left(\mu+\frac{3 \sigma^2}{4}\right)}+e^{t\left(\mu+\sigma^2\right)}+e^{\frac{1}{4} t\left(4 \mu+\sigma^2\right)}+e^{t \mu+\frac{t \sigma^2}{2}}\right) S-4 e^{\frac{1}{8} t\left(4 \mu+\sigma^2\right)}\left(1+e^{\frac{t \sigma^2}{4}}\right) \sqrt{S} \sqrt{p_b}+4 p_b\right)}{p_b}$

Above supplies a closed form mean and variance of the values.

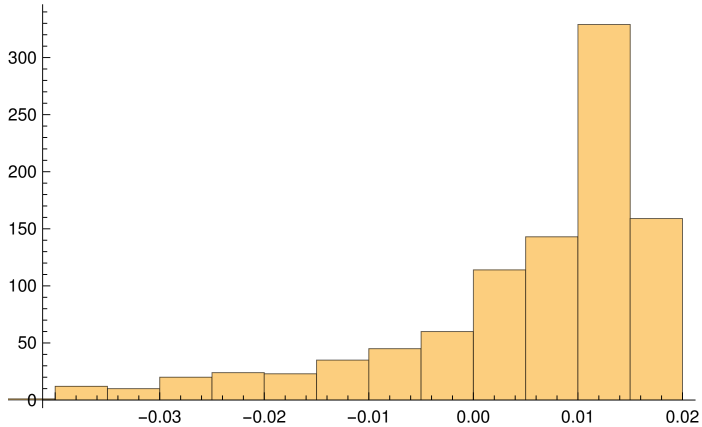
In[29]:=
```
realizedDistribution =
  procPVOpen /. {μ → ethMeanYearly/365, σ → ethDailyVol, S → currentPrice,
     higher → upperBound, lower → lowerBound} /. currentLiquidityParams[[1]];
realizedSamples = SliceDistribution[realizedDistribution, 90] //
    RandomVariate[#, 1000] & //
   Map[(# - 10 000)/10 000 &];
```

In[31]:=
```
realizedSamples // {Mean[#], Median[#], Quantile[#, {1/20, 1/4, 3/4, 19/20}]} &
```

Out[31]= {0.00344573, 0.00957961, {−0.0296404, −0.000634804, 0.0141588, 0.0155399}}

In[32]:=
```
Histogram[realizedSamples]
```

Out[32]=



From this we have a clear closed form solution for the mean and variance, so one can produce 95% confidence intervals.

Further Discussions

Much of this was inspired by the papers below. Since this is more about sampling and not mathematical proofs, readers should take a look at the analysis below for better metrics.

In[33]:=
```
TableForm[{
  {"Impermanent Loss in Uniswap V3", Hyperlink[
    "https://lambert-guillaume.medium.com/an-analysis-of-the-expected-value-of-the-
      impermanent-loss-in-uniswap-bfbfebbefed2"]},
  {"Uniswap Liquidity V3 Math",
   Hyperlink["http://atiselsts.github.io/pdfs/uniswap-v3-liquidity-math.pdf"]}
 }]
```

Out[33]//TableForm=

Impermanent Loss in Uniswap V3	https://lambert-guillaume.medium.com/an-analysis-of-impermanent-loss-in-uniswap-bfbfebbefed2
Uniswap Liquidity V3 Math	http://atiselsts.github.io/pdfs/uniswap-v3-liquidity

Perpetual Lending Stochastic Analysis

Mean-Reverting Additional Value Term

Below we use the same techniques to add an additional interest rate using the Ornstien Uhlenbeck proccess. This model is similar to a perpetual option where it does fluxuate, but is mean reverting assuming a standard 2% per year interest rate.

In[34]:=
```
100 (1 - ℯ^(-fl8fl0))
```

Out[34]=
```
1.98013
```

In[35]:=
```
borrowedCapital = 10 000;
borrowDailyRate = 0.02/365;
borrowRateDailyVol = borrowDailyRate/20;
orResponseTerm = 0.2;
mrModel =
 currentValue - borrowedCapital (1 - ℯ^(-t [v]*v)) /. {p → p[t], higher → upperBound,
     lower → lowerBound} /. currentLiquidityParams[[1]] // Simplify
```

Out[39]=
```
-339 988. + 10 000. ℯ^(-t r[t]) + 16 499.4 Sqrt[p[t]] - 200.085 p[t]
```

We can see we have a viable closed form model. Below is a histogram of values.

In[40]:=
```
realizedProcMR = TransformedProcess[ mrModel , {
    p ≈ GeometricBrownianMotionProcess[ethMeanYearly/365, ethDailyVol, currentPrice],
    r ≈ OrnsteinUhlenbeckProcess[borrowDailyRate,
      borrowRateDailyVol, orResponseTerm, borrowDailyRate]
  }, t]
```

Out[40]=
```
TransformedProcess[-339 988. + 10 000. ℯ^(-t p2[t]) + 16 499.4 Sqrt[p1[t]] - 200.085 p1[t],
 {p1 ≈ GeometricBrownianMotionProcess[0.000273973, 0.0034, 1628],
  p2 ≈ OrnsteinUhlenbeckProcess[0.0000547945, 2.73973 × 10^-6, 0.2, 0.0000547945]}, t]
```

In[41]:=
```
samplesMRDistribution = SliceDistribution[realizedProcMR, 90] //
    RandomVariate[#, 1000] & // Map[(# - 10000)/10000 &];
stats = samplesMRDistribution // {Mean[#], Median[#], Quantile[#, {1/20, 1/4, 3/4, 19/20}]} &

Histogram[samplesMRDistribution]
```

Out[42]= {-0.00082362, 0.00523417, {-0.0308813, -0.00557974, 0.0094681, 0.010742}}

Out[43]=