Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Cryptocurrency Statistical Arbitrage on Decentralised Exchanges

Author:
Devam Savjani

Supervisor:
Dr. Thomas E. Lancaster

Second Marker:
Mr. Ivan Procaccini

June 8, 2023

Abstract

SOME ABSTRACT

Acknowledgements

SOME ACKNOWLEDGEMENTS

Contents

Chapter 1

Introduction

1.1 Motivation

Since the introduction of Bitcoin, a peer-to-peer payment network and cryptocurrency, there have been countless new cryptocurrencies that are used and traded. Along with this, their high volatility has piqued a lot of retail investor's interest with some investors gaining a high return of interest and others losing a lot of money [1]. In addition to this, larger investment institutions have also sought to gain profits from this new type of tradable asset [2]. This has lead to more sophisticated forms of cryptocurrency trading.

However, trading cryptocurrencies was initially very difficult for people without technical know-how, and the first recorded transaction was on 12^{th} October 2009 via a paypal transaction [3]. Since then, other cryptocurrency exchanges have emerged, many of which are centralized, which provide a more traditional trading terminal and support for investors with little technical know-how, and others are decentralised, which operate on blockchain networks and allow users to directly with each other using smart contracts. Centralized exchanges are predominant due to their easy-to use and familiar interface for traders. We can see in Figure 1.1 the proportion of trades on decentralised exchanges compares to the number of trades on centralized exchanges, we can also see that the volume traded in DEXes have been near 0% until the summer of 2020 and has been increasing since.

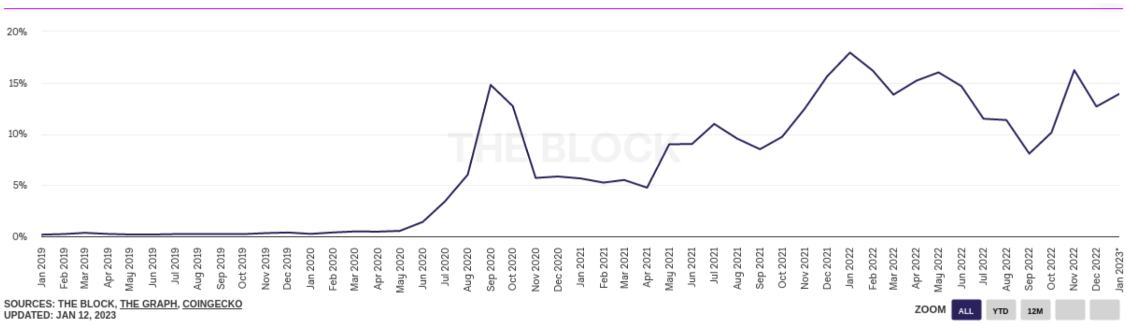

Figure 1.1: DEX to CEX Spot Trade Volume [4]

This poses the question that about which trading strategies can exploit arbitrage opportunities on decentralised exchanges. There has been some research on this topic, mainly focussing on triangular and cyclic arbitrage on DEXes such as Uniswap and SushiSwap, however there has been no research into analysing the performance of statistical arbitrage methods on decentralised exchanges.

1.2 Contributions - TODO

Chapter 2

Background

2.1 Cryptocurrencies

Before delving into the financial side of the project, it is important to understand the underlying assets and the technology that drives them.

2.1.1 Blockchain

The building blocks of cryptocurrencies come from blockchain technology. Blockchain is a distributed ledger that stores data, in blocks, in a chain, comprising the data itself as well as a full transaction history [5]. They can be thought of as a State Transition System where the state is the store of data, i.e. the owners of each token, and the state-transition function is a function of state and a transaction [6]. The transition function defines how the transaction, T, should affect the state, σ.

$$\sigma' = transition\_func(\sigma, T)$$

Thus the state, σ, and transition function $transition\_func$ are defined by the blockchain implementation, which tends to be more complex.

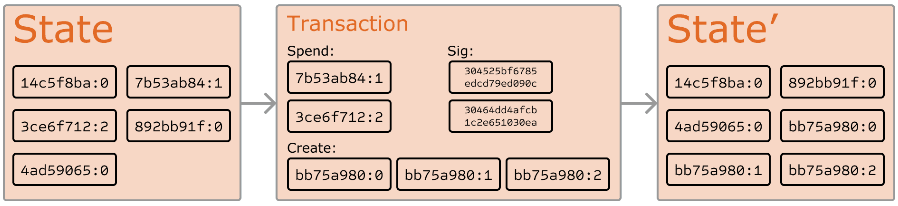

Figure 2.1: State Transition Diagram [6]

In reality, a collection of transactions are batched together into a block which is then commited to the network. The components of a block can be seen below.

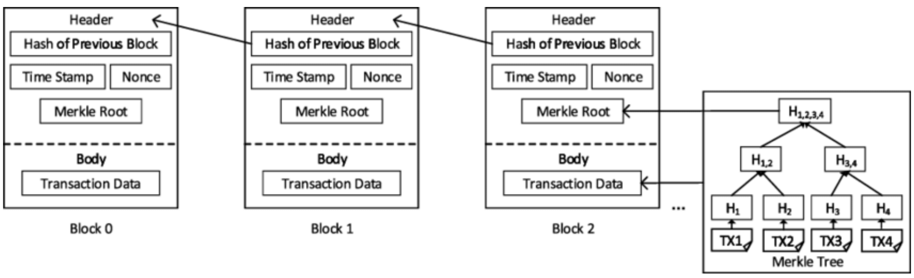

Figure 2.2: Blockchain Diagram [7]

A block can two significant components, the header and the body. The body of the block contains the list of transactions that are to be applied. Whereas, the components of the header may depend on the blockchain's implementation however they typically have the following key components.

- **Block number** - A unique identifier assigned to each block in the chain. It serves as a chronological ordering mechanism, indicating the position of the block within the blockchain.
- **Timestamp** - Timestamp of the block's creation time.
- **Merkle Root** - A Merkle root is also stored in each block to validate transactions in the body of the block efficiently, in terms of storage and searching [8]. A Merkle tree is a tree of hashes where each leaf node is its data hash and its parent node is the hash of their children's hashes. In storing the Merkle root, we do not need to directly store each transaction in each block, and also allows a quick search for any malicious alterations in differing blocks.
- **Hash of Previous Block** - Acts as a pointer to the predecessing block. Hence, the term "Blockchain" is coined as one would be able to track back to the first block from the latest block in a chain like manner.

In addition to these components, a major feature of blockchain technology is its property of being decentralised. The state of the blockchain is managed by a distributed network of nodes that are maintained by voluntary node operators. Node operators maintain the blockchain by synchorizing with the network, validating, propogating transactions and blocks and finally participating in consensus mechanisms to ensure all of the nodes agree on the current state before updating the blockchain. For this blockchain networks are typically built on a peer-to-peer (P2P) architecture. Each participant in the network, or node, has an equal status and communicates directly with other nodes. Transactions and data are shared among participants without the need for intermediaries or central servers. When transitioning to the next state, the node broadcasts the transaction, T, to the blockchain network and each node applies the transition function to obtain the new and same state, $\sigma' = transition\_func(\sigma, T)$. All nodes arrive to the same state as the transition function is deterministic. Once each node has applied the transition function, the network uses a consensus mechanism to the globally agreed σ' and the transactions become actualised.

Consensus mechanisms are used to achieve agreement among participants on the validity of transactions and the order in which they are added to the blockchain. Examples of consensus mechanisms include Proof of Work (PoW), Proof of Stake (PoS), and Delegated Proof of Stake (DPoS), however current blockchains predominantly use Proof of Work or Proof of Stake [9]. In addition to reaching consensus, the consensus algorithms also determine which node should be the create of a new block, which node should be granted a monetary reward.

The Proof of Work (PoW) consensus algorithm was first used in Bitcoin and is where the nodes compete to solve complex mathematical problems to verify that the transactions from a new block are valid in order to add them to the blockchain. These nodes are called miners. Once the first miner obtains the solution of the mathematical puzzle, it broadcasts the new block on the network so that the other nodes verify the solution and update their local replica. Consensus is then reaches when 51% of the nodes agree on the new state of the blockchain. The main drawback of this algorithm is that it requires miners to perform extensive computational calculations repeatedly until they find the solution to the mathematical problem, hence once the solution has been found by a miner, after expensive work, the solution is sent along in the broadcast to the other nodes to easily and cheaply verify the solution of the problem.

To address this issue, some blockchains have opted to use the Proof of Stake (PoS) consensus algorithm as it is more energy efficient and scalable. Unlike Proof of Work, Proof of Stake use validators to create new blocks. Validators stake an amount of cryptocurrency as collatoral and the creator of a block is chosen at random from the validators, with the probability of selection based on the amount they have staked. Due to the advantages that PoS has over PoW, Ethereum has transitioned from PoW to PoS in Ethereum 2.0.

2.1.2 Ethereum

One of them was proposed by Vitalik Buterin, the co-founder of Ethereum, in a whitepaper that proposed the idea of using smart contracts to create financial products and services that could operate independently of traditional financial institutions, hence decentralised finance was birthed [10].

Smart Contracts

Smart contracts are programs that are self-executing contracts between buyers and sellers that deploy on a blockchain. The programs are reactive in the sense that they are only executed on the blockchain providing that certain conditions are met [11]. These conditions are set in the lines of code in the smart contract. To posses the reactive property smart contracts are stored and executed on every participating node on the network as part of the state σ. They are also immutable and decentralised which make smart contract advantageous as they can be used to facilitate, verify, and enforce the negotiation or performance of a contract [12, 13].

Ethereum's Architecture

Ethereum's architecture is similar to bitcoin's but has a few differences, one of which is rather than managing a distributed ledger, it uses a distributed state machine. The Ethereum Virtual Machine (EVM) defines the rules of changing states from block to block. Each node on the Ethereum blockchain contains an immutable instance of the EVM [14].

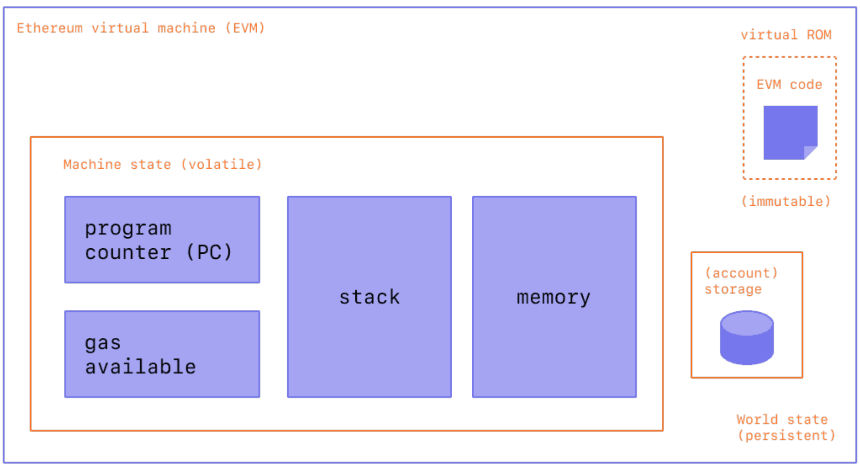

Figure 2.3: EVM components [14]

Ethereum's state σ consists of the states of all accounts where the state of an account is represented by four components: balance of ETH (the native cryptocurrency on Ethereum), a nonce, hash of the smart contract code (if account is a smart contract) and the Merkle root of smart contract storage which maintains the contracts variables (again, if account is a smart contract) [15]. A nonce, 'Number only used once', is a number that is added to a hashed block to make the transaction more secure. It is randomly generated and used to validate a transaction. A miner first guesses a nonce and appends the guess to the hash of the current header. The miner then rehashes the value and compares this to the target hash. If the guess was correct, the miner is granted the block [16].

Ethereum supports two types of transactions: contract creation and message calls. Message calls are used to transfer ETH from account to account and also for invoking smart contract. The process of how transactions are validated is below:

1. Validate the parent block
2. Validate that the current timestamp is greater than the previous timestamp
3. Check that the Ethereum concepts are valid
4. Perform Proof of Stake on the block
5. Check for errors and gas
6. Validate the final state

Gas Fees

Due to the capability of expressing loops in EVM bytecode, there is a possibility that smart contract execution could continue indefinitely. This presents a challenge as invoking a contract with an infinite loop would cause all Ethereum network nodes to become stuck executing it, resulting in a denial of service. To mitigate this issue, Ethereum has implemented a pricing mechanism. Every computation performed by a smart contract requires the payment of a fee called gas, which is denominated in ether.

To measure how much computational effort is required to execute operations on the Ethereum network, gas is used [17]. Every block has a base fee, derived from the demand for the block space, which is burnt. Therefore, users of the network are expected to set a tip (priority fee) to reimburse miners for adding their transaction in blocks, thus the higher the tip, the greater the incentive for miners to validate the transaction. Using gas means that the Ethereum network is tolerant to spam and also has a maximum gas fee to make Ethereum tolerant to malicious code that would be used to waste resources.

2.1.3 Decentralised Finance

One of the applications of Ethereum and smart contracts is Decentralised Exchanges (DEXes). Before delving into DEXes it is important to understand centralized exchanges.

Centralized Exchanges

Centralized exchanges(CEXes) allow agents to discover and trade assets. CEXes facilitate trading between buyers and sellers by proving an online platform that manages and maintains an order book. An order book aggregates buy and sell orders and execute matching buy and sell orders. The order book and transactions are typically managed on a central database. When trading, exchanges charge trading fees for the maker and the taker to operate the exchange and do not charge any gas fees as there is no interaction with the blockchain.

Decentralized Exchanges

In contrast, DEXes utilize blockchain technology and smart contracts to execute trades thus providing a high level of determinism, by nature of the technology. These trades are executed on the blockchain via smart contracts and on-chain transactions. There are two types of DEX: order book DEXes and Automated Market Makers (AMMs). An order book DEX is less common and is akin to a CEX, an order book is stored on the blockchain rather than on a central database. This means each order placed requires the order book to be posted on the blockchain at each transaction. Automated Market Makers are more common and provide instant liquidity by using liquidity pools so that users can swap their tokens for a price that is determined by the portions within the liquidity pool [18]. DEXes have multiple pros including lower transaction fees, privacy, diversity and trustless transactions but they also have their drawbacks such as scalability and poor liquidity a lot of the DEXes are quite new [19].

2.2 Arbitrage

Arbitrage is the process in which a trader simultaneously buys and sells an asset to take advantage of a market inefficiency [20]. Arbitrage is also possible in other types of securities by finding price inefficiencies in the prices of options, forward contracts and other exotics.

Sources have shown that the word "*Arbitrage*" has been used as early as the Renaissance era when surviving documents showed a large number of bills being exchanged [21]. There has also been some evidence to suggest that arbitrage was used as early as the Greek and Roman eras. Early forms of arbitrage would likely have been purchasing a commodity then transporting them to a foreign land and selling them at a higher price. This type of arbitrage is called commodity arbitrage and is still applicable today. With the example above, transporting the goods takes a significant amount of to the merchant, or trader, which could cause variations in the price, however, in the modern day this has been reduced and with electronic exchanges, this time to buy and sell is very small. This means inefficiencies in the market, where a trader can profit purely by buying and selling, should not exist. This is called the "Law of One Price". The "Law of One Price" states that every identical commodity or asset should have the same price regardless of exchange or location, given there are no transaction costs, no transportation costs, no legal restrictions, the exchange rates are the same and no market manipulation occurs [22]. This is because if this were not the case, an arbitrage opportunity would arise and someone would take advantage of the scenario causing the prices on both markets to converge due to the market forces. In the real world arbitrage opportunities are tremendously common, thus allowing a risk-free investment [23, 24].

There are countless types of arbitrage such as spatial arbitrage, which profits off of different prices on exchanges in different locations, temporal arbitrage, which takes advantage of price differences at different times, risk arbitrage, which profits from perceived discrepancies in their risk-return profiles and finally market arbitrage which takes advantages of different prices on different exchanges/markets. Statistical methods include pairs trading, which involves buying and selling assets that are believed to be mispriced relative to one another, momentum trading, which identifies if assets have a strong momentum (either up or down) and profiting off of that, and finally, algorithmic trading which uses algorithms to analyze data and trades based on statistical analysis. This project shows how these opportunities can be exploited both in a pure manner as well as using statistical methods.

2.3 Pure Arbitrage Techniques

Research into cryptocurrency arbitrage is still in its infancy and previous research has mainly focussed on the economics of cryptocurrencies, i.e. miner/trader behaviour and influence of cryptocurrency trading [25, 26, 27, 28, 29, 30, 31]. Furthermore, there has been very limited research comparing statistical strategies and pure methods of arbitrage of cryptocurrencies. Despite this, there has been plentiful research on arbitrage as a whole as it is immensely profitable; as a result of this people/institutions tends to keep their newly found research secret. Of the published research, I have looked into the arbitrage techniques that are used. As arbitrage can is highly profitable, it can be found in countless types of assets, such as options, stocks, bonds and many other types of products. Research into all types of products exist going into the theory and practical aspects of each [32, 33]. The most similar type of asset class to cryptocurrencies is fiat currencies, traditional currencies that are issued by governments, such as the US Dollar (USD) and the Great British Pound Sterling (GBP). The research in arbitrage in foreign exchanges shows that using a triangular/cyclic arbitrage is highly profitable and effective [34, 35, 36]. This trading strategy takes advantage of price discrepancies between three or more different currencies in the foreign exchange market. It involves executing a series of trades to profit from the imbalance in exchange rates between the currencies involved. An example of this can be seen in Figure 2.4, given the exchange rates are \$1 = €0.85, €1 = £0.75, £1 = \$1.20, we can make a series of exchanges (trades) such that by starting with \$100, the result of this cyclic arbitrage I am left with \$130.72, hence a \$30.72 risk-free profit.

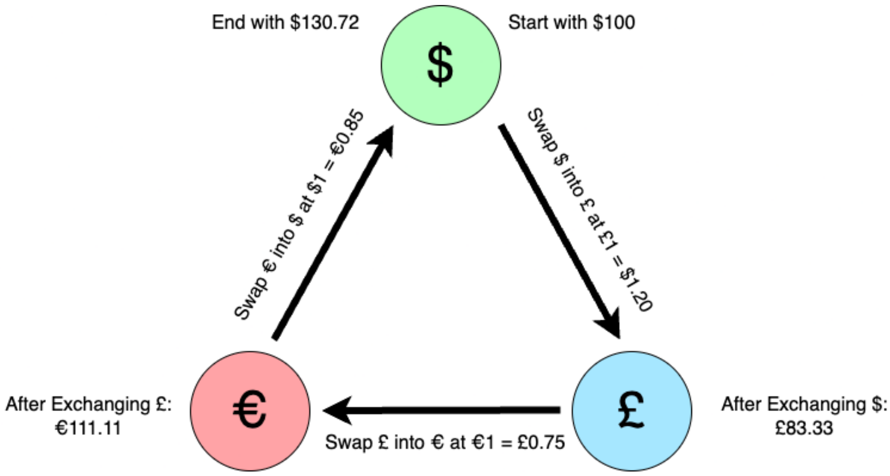

Figure 2.4: Triangular Arbitrage Diagram

Although this project aims to focus on statistical arbitrage techniques, it is still important to look at research into purer forms of arbitrage, i.e. triangular and cyclic arbitrage.

As previously mentioned, research into this topic is still in its infancy thus which means a very thin slice of exploration on the subject matter. The majority of the research has been into the arbitrage on centralized exchanges [37, 38, 39]. Cristian Pauna investigates and implements an arbitrage strategy in [39]. The paper details the technical details of arbitrage trading from the data and the system architecture used. Pauna finds complications such as requesting data from multiple exchanges, converting the data such that it is homogeneous and also managing server load. Pauna presents the architecture such that the servers request data from the necessary exchanges, aggregating prices in a relational database which then triggers a server that is used to generate trading signals.
As previously mentioned triangular and cyclic arbitrage is one of the most used and purest forms of arbitrage to implement and analyse, [40] explores triangular arbitrage on decentralised exchanges. Algorithm 1 is the algorithm used to find the most profitable arbitrage route on a particular platform, once this is calculated, it is compared with other routes on other platforms. Initially, the system converts the base token into another token and converts it back into the base token, using only one token is used as a middle route, then using the algorithm below, increases the number of middle tokens.
On evaluating the performance of the strategy on differing platforms depended on three main features of each exchange:

1. Portion size - Depending on how much the "trader" invested revenues differed and with the larger portion size, the revenue decreases as the token pair prices are adjusted based on supply/demand.

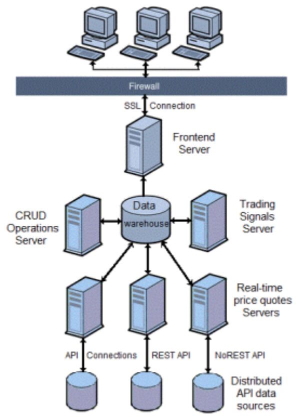

Figure 2.5: Arbitrage system architecture [39]

Algorithm 1 Maximum Profit Route Searching (R)

Input: T (token list), P (price graph), n (current route)

  for $i = 1, ..., T$ **do**
    $r = get\_profit(n + i)$
    for $j = 1, ..., P[i]$ **do**
      $p = max(r, R(T, P, n_j))$
    end for
  end for
  return p

---

2. Transaction fees - Each exchange has its own transaction fee.
3. Other considerations such as price slippage - Exchanges have different liquidity levels which depend on the usage and liquidity providers that the exchange employs.

Figure 2.6 displays the revenues obtained by same trading token route, ETH $\rightarrow$ MKR $\rightarrow$ OMG $\rightarrow$ USDT $\rightarrow$ ETH. As we can see upon applying the strategy on multiple exchanges; Uniswap, 1inch, Kyberswap and Bancor, 1 inch was the only exchange that generated a profit whereas the others lose money.

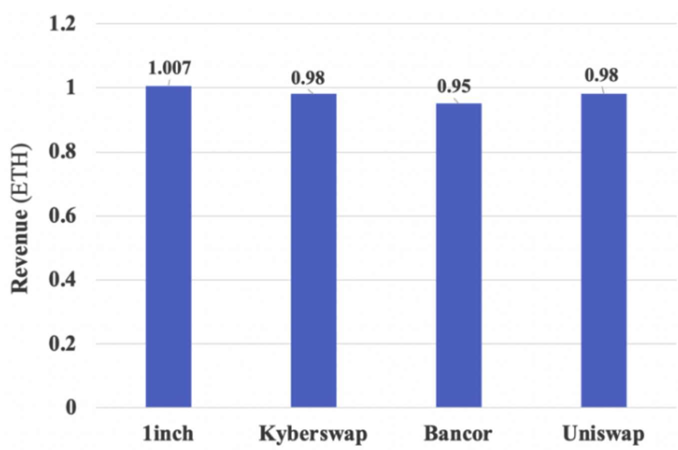

Figure 2.6: Trading profits same token routes within different exchanges [40]

Another paper that implemented and evaluated a cyclic arbitrage opportunity is [41]. The research consists of proposing a theoretical arbitrage model and further evaluation of real transactional data. The arbitrage model used is simple to understand, as it searches for a cyclic transaction between n tokens, $A_1, A_2, ..., A_n$ is a sequence of n trades:

$$\textit{Trade 1:}\ \text{Exchange } \delta_1 \text{ of } A_1 \text{ to } \delta_2 \text{ of } A_2$$

$$\textit{Trade 2:}\ \text{Exchange } \delta_2 \text{ of } A_2 \text{ to } \delta_3 \text{ of } A_3$$

$$\dots$$

$$\textit{Trade n:}\ \text{Exchange } \delta_n \text{ of } A_n \text{ to } \delta_1' \text{ of } A_1$$

It is important to note that $\delta_i = \delta_{i+1}$, i.e. the output of trade is equivalent to the input of the next. The revenues within a cycle are defined as $\delta_{i+1} - \delta_i$, and the overall profit is $\delta_1' - \delta_1$. This is not as simple as the revenues depend on how liquid the exchange is, thus the liquidity pools of each possible trading pair are hugely important. Therefore, the paper proposes a theorem, below:

Theorem 1 *For a given cycle $A_1 \to A_2 \to \cdots \to A_n \to A_1$ with n tokens, there exists an arbitrage opportunity for the cyclic transaction if the product of exchange rates $\frac{a_{2,1}a_{3,2}\cdots a_{1,n}}{a_{1,2}a_{2,3}\cdots a_{n,1}} > \frac{1}{r_1^n r_2^n}$ where $a_{i,j}$ denotes the liquidity of token A_i in the liquidity pool with token A_j. [41]*

In addition to the theorem, to obtain an optimal strategy we need to compute the optimal trading volume of a cycle, $A_1 \to A_2 \to \cdots \to A_n \to A_1$. The paper proposes the optimal trading volume to be $\delta_a^{op} = \frac{\sqrt{r_1 r_2 a' a} - a}{r_1}$ where $a = \frac{a'_{1,n} a_{n,1}}{a_{n,1} + r_1 r_2 a'_{n,1}}$ and $a' = \frac{r_1 r_2 a'_{1,n} a_{n,1}}{a_{n,1} + r_1 r_2 a'_{n,1}}$. Thus to calculate such arbitrage opportunities knowing the liquidity of tokens in other tokens' liquidity pools, algorithm 2 infers the direction and volumes to trade to get the optimal revenue.

Algorithm 2 Computing the equivalent liquidity of the cycle

$a'_{1,n} \leftarrow a_{1,2}$
$a'_{n,1} \leftarrow a_{2,1}$
for i from 2 to $n-1$ **do**
  $a'_{1,n} \leftarrow \frac{a'_{1,n} a_{i,i+1}}{a_{i,i+1} + r_1 r_2 a'_{n,1}}$
  $a'_{n,1} \leftarrow \frac{r_1 r_2 a'_{1,n} a_{i+1,i}}{a_{i,i+1} + r_1 r_2 a'_{n,1}}$
end for

After analyzing Ethereum block data and applying this strategy to identify the number of arbitrage opportunities, it was found that between May 4, 2020, and April 15, 2021, there were numerous exploitable and profitable arbitrage opportunities. These opportunities grew consistently to reach 1,750 in 11 months, as depicted in Figure 2.7. Only cycles with length 3 were experimented with and only cycles including ETH as 80% of the liquidity pools on Uniswap include ETH and another cryptocurrency [42]. Furthermore, it is found that 287,241 of the 292,606 arbitrages executed started with ETH, and 85% of the arbitrages used a cycle of length 3. The total revenue of the cyclic arbitrage was 34,429 ETH. However, gas fees account for 24.6% of the total revenue leaving an approximate 25,971 ETH profit.
The paper then delves into the implementation of the smart contract, and explores how both *sequential* and *atomic* implementations would affect the revenue and execution of the contracts. It was found that 52.3% of the arbitrages that were executed sequentially generated a loss, likely due to the fact that, when one submits n orders, the n blockchain transactions are executed sequentially, meaning some external transactions can be inserted between these transactions. Thus using atomic transactions avoids this issue of external transactions does not affect the market price that may affect the outcome of the arbitrage.

Furthermore, the authors of the paper also investigated the performance differences between using private smart contracts and public contracts. Deploying a smart contract that

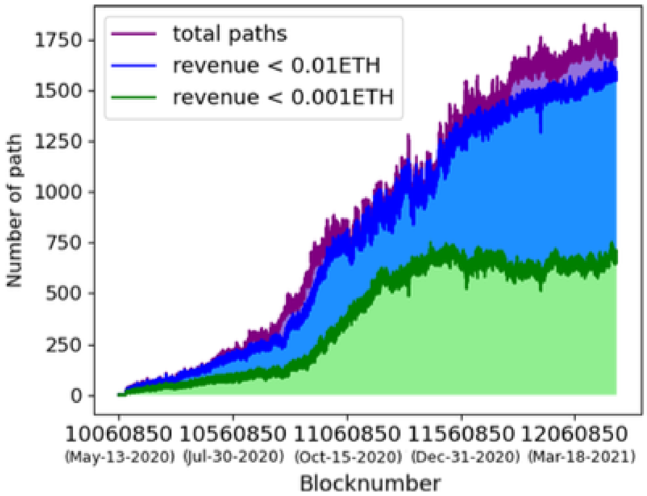

Figure 2.7: Number of exploitable opportunities in Uniswap V2 over time. The purple line represents the number of cycles that provide revenue higher than 0.0001 ETH. The green represents the number of cycles whose revenue is under 0.001 ETH. The blue line represents the number of cycles whose revenue is under 0.01 ETH. [41]

calls Uniswap functions, i.e. a private smart contract, is intuitively better and achieves a higher success rate of a lower bound of 52% and a higher bound of 90% in comparison to calling a public Uniswap smart contract which has a success rate of 27.3%. Overall the paper provides an insightful look into cyclic arbitrage in DEXes and highlights important decisions made such as liquidity calculations and smart contracts while comparing the performance of different options available.

2.4 Statistical Arbitrage Techniques

As mentioned previously mentioned and the subject of the project is to optimize statistical arbitrage methods to be able to compete with a purer form of arbitrage, i.e. cyclic arbitrage. As previously mentioned there are many methods of stat arb, pairs trading, momentum trading and algorithmic trading. Within these methods there are countless strategies to adopt and profit from, thus to limit the scope, this project I will be investigating strategies within pair trading. Research within Pair trading has been vast with many streams of approaches emerging; distance approach, cointegration approach, time-series approach, stochastic approach and some others, including using machine learning [43]. However, for this project, we will only look at cointegration/co-correlation approaches.

2.4.1 Mean Reversion

The cointegration approach follows three key steps. The first is the selection of pairs based on similarity measures, the next is assessing the tradability and finally, thresholds are set for trading. The spread is defined as

$$\varepsilon_{ij,t} = P_{i,t} + \gamma P_{j,t}$$

where $P_{i,t}$ and $P_{j,t}$ denote the $I(1)$ non-stationary price processes of the assets i and j, γ is the cointegration coefficient, also referred to in literature as the hedge ratio. $\varepsilon_{ij,t}$ is the linear combination of the non-stationary prices and is $I(0)$ stationary and hence mean-reverting, note that stationary processes are those of which have a constant mean. Rad's implementation of this approach on stocks results in a 0.83% return before considering transaction costs [44]. Another paper, [45], looked into setting the thresholds and setting a minimum profit, MP_{ij,t_c}:

$$MP_{ij,t_c} = \frac{n(\varepsilon_{ij,t_0} - \varepsilon_{ij,t_c})}{\mid \gamma \mid}$$

Where t_0 and t_c are the opening and closing times, n is the volume longed of asset j.

2.4.2 Optimal Portfolio Design for Mean Reversion

There has been further research into optimizing mean reversion, one of which was to use the successive convex approximation method on the mean reverting portfolio design [46]. The paper initially proposes the mean reversion portfolio:

- For each asset, the price at time t is denoted as p_t and its correspondeing log-price $y_t \triangleq log(p_t)$, its vector form of M assets $\mathbf{y}_t \triangleq \left[y_{1,t}, \ldots, y_{M,t}\right]^T$.
- The log-price spread is given by $y_t \triangleq \beta^T \mathbf{y}_t$, where $\beta \triangleq \left[\beta_1, \ldots, \beta_M\right]^T$ denotes the hedge ratios.
- The cointegration space with N relations is defined by $\mathbf{B} \triangleq \left[\beta_1, \ldots, \beta_N\right]$, thus the N spreads are $s_t \triangleq \mathbf{B}^T \mathbf{y}_t$.
- For these N spreads, the portfolio weight matrix is denoted as $\mathbf{w} \triangleq \left[w_1, \ldots, w_N\right]^T$.
- The auto-covariance matrix for the spreads s_t is defined as $M_i \triangleq Cov(s_t, s_{t+i}) = \mathbb{E}\left[\left(s_t - \mathbb{E}\left[s_t\right]\right)\left(s_{t+i} - \mathbb{E}\left[s_{t+i}\right]\right)^T\right]$

Now that we have defined everything required, we can now formalize the problem. The general problem of mean reversion portfolio design problem is formalized by:

$$\begin{aligned} &\underset{\mathbf{w}}{\text{minimize}} && F(\mathbf{w}) \triangleq U(\mathbf{w}) + \mu V(\mathbf{w}) + \gamma S(\mathbf{w}) \\ &\text{subject to} && \mathbf{w} \in \left\{\mathbf{w} \mid \left\|\mathbf{B}\mathbf{w}\right\|_0 \leq L\right\}, \qquad \text{where } L \text{ is the total leveraged investment} \end{aligned}$$

- μ defines the trade-off between the mean reversion measure and the variance preference.
- γ defines the regularization parameter of how sparse we would like the cointegration space to be.

Where the Mean Reversion term:

$$U(\mathbf{w}) \triangleq \xi \frac{\mathbf{w}^T \mathbf{H} \mathbf{w}}{\mathbf{w}^T \mathbf{M}_0 \mathbf{w}} + \zeta \left(\frac{\mathbf{w}^T \mathbf{M}_1 \mathbf{w}}{\mathbf{w}^T \mathbf{M}_0 \mathbf{w}}\right)^2 + \eta \sum_{i=2}^{p} \left(\frac{\mathbf{w}^T \mathbf{M}_i \mathbf{w}}{\mathbf{w}^T \mathbf{M}_0 \mathbf{w}}\right)^2$$

And the variance term:

$$V(\mathbf{w}) \triangleq \begin{cases} 1/\mathbf{w}^T \mathbf{M}_0 \mathbf{w} & \text{VarInv}(\mathbf{w}) \\ 1/\sqrt{\mathbf{w}^T \mathbf{M}_0 \mathbf{w}} & \text{StdInv}(\mathbf{w}) \\ -\mathbf{w}^T \mathbf{M}_0 \mathbf{w} & \text{VarNeg}(\mathbf{w}) \\ -\sqrt{\mathbf{w}^T \mathbf{M}_0 \mathbf{w}} & \text{StdNeg}(\mathbf{w}) \end{cases}$$

The variance term can be represented in any of the four forms.

And the asset selection term:

$$S(\mathbf{w}) \triangleq \left\|\mathbf{B}\mathbf{w}\right\|_0 = \sum_{m=1}^{M} sgn(\mid \left[\mathbf{B}\mathbf{w}\right]_m \mid)$$

This asset selection criterion is not necessary however as trading incurs a cost, selecting all of the assets is costly, thus selecting a subset of assets to trade is more profitable. To formalize this goal, we would like to minimize the cointegration space thus we use the ℓ_0 norm.

The paper then goes on to solve the optimization problem using the successive convex approximation (SCA) method [47]. The SCA method takes an optimization problem in the form of:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned}$$

Where $\mathcal{X} \subseteq \mathbb{R}^N$ is convex and $f(\mathbf{x})$ is non-convex. The SCA method involves starting at an initial point $\mathbf{x}^{(0)}$ and solving a series of subproblems of surrogate funtions $\tilde{f}(\mathbf{x};\mathbf{x}^{(k)})$ over the set $\mathcal{X}$. The sequence $\{\mathbf{x}^{(k)}\}$ is generated by:

$$\begin{cases} \hat{\mathbf{x}}^{(k+1)} = \underset{\mathbf{x}\in\mathcal{X}}{argmin} \tilde{f}(\mathbf{x};\mathbf{x}^{(k)}) \\ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \gamma^{(k)}(\hat{\mathbf{x}}^{(k+1)} - \mathbf{x}^{(k)}) \end{cases}$$

The first step is to generate a descent direction and then update the variable with a step size of $\gamma^{(k)}$. After applying this method to the MRP problem and further analysis of the paper, the following algorithm is proposed and used to solve the MRP design problem:

Algorithm 3 SCA-Based Algorithm for The Optimal MRP Design Problem

Require: $\mathbf{H}, \mathbf{M}_i, \mu, \gamma, \mathbf{B}, L$ and τ

1: Set $k = 0, \gamma^{(0)}$ and $\mathbf{w}^{(0)}$
2: **repeat**
3: Compute $\mathbf{A}^{(k)}$ and $\mathbf{b}^{(k)}$
4: $\hat{\mathbf{w}}^{(k+1)} = \underset{\mathbf{w}\in\mathcal{W}}{argmin}\, \mathbf{w}^T\mathbf{A}^{(k)}\mathbf{w} + \mathbf{b}^{(k)T}\mathbf{w}$
5: $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \gamma^{(k)}(\hat{\mathbf{w}}^{(k+1)} - \mathbf{w}^{(k)})$
6: $k \leftarrow k + 1$
7: **until** convergence

However, 4 line is a convex problem and has no closed-form solution thus to solve this subproblem using the ADMM method, this is done by introducing an auxiliary variable $\mathbf{z} = \mathbf{B}\mathbf{w}$.

$$\begin{aligned} \underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad & \mathbf{w}^T\mathbf{A}\mathbf{w} + \mathbf{b}^T\mathbf{w} \\ \text{subject to} \quad & \|\mathbf{z}\|_1 \leq B, \mathbf{B}\mathbf{w} - \mathbf{z} = \mathbf{0} \end{aligned}$$

This is then summarized into Algorithm 4:

Algorithm 4 An ADMM-Based Algorithm for Problem on line 4 in Algorithm 3

Require: $\mathbf{A}, \mathbf{b}, \mathbf{B}, B, \rho$

1: Set $\mathbf{w}^{(0)}, \mathbf{z}^{(0)}, \mathbf{u}^{(0)}$ and $k = 0$
2: **repeat**
3: $\mathbf{w}^{(k+1)} = -(2\mathbf{A} + \rho\mathbf{B}^T\mathbf{B})^{-1}(\mathbf{b} + \rho\mathbf{B}^T(\mathbf{u}^{(k)} - \mathbf{z}^{(k)}))$
4: $\mathbf{h}^{(k)} = \mathbf{B}\mathbf{w}^{(k+1)} + \mathbf{u}^{(k)}$
5: $\mathbf{z}^{(k+1)} = \Pi_{\mathcal{C}}(\mathbf{h}^{(k)})$
6: $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{B}\mathbf{w}^{(k+1)} - \mathbf{z}^{(k+1)}$
7: $k \leftarrow k + 1$
8: **until** convergence

After all of this analysis, the authors of the paper, [48, 46], ran simulations on real data comparing underlying spread. It found that it resulted in consistent profits as shown in figure 2.8:

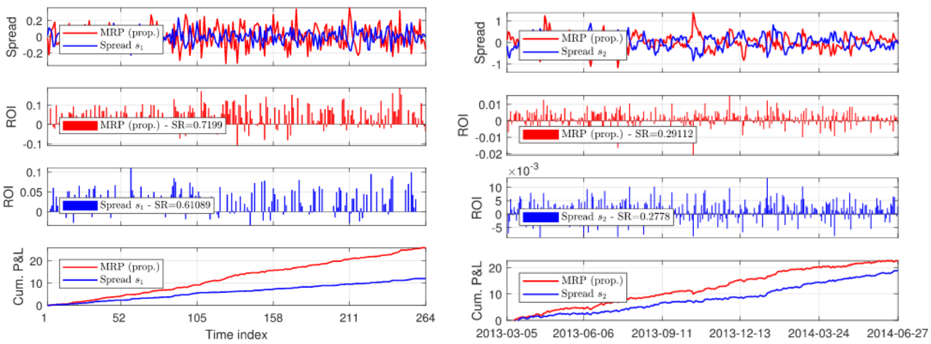

Figure 2.8: A mean-reversion trading based on real data [46]

Overall, this research successfully formulizes, solves the optimization problem mathematically, and goes further to implement the algorithms to solve the problem programmatically. In addition, the author compares the implementation with other benchmark algorithms, showing that it results in a greater P&L and Sharpe ratio.

2.4.3 Statistical Arbitrage using the Kalman Filter

Another method that is used in statistical arbitrage is using the Kalman Filter. Recall the equation for spread:

$$\varepsilon_{ij,t} = P_{i,t} + \gamma P_{j,t}$$

The Kalman filter is a recursive algorithm for estimating the state of noisy data and that needs to be filtered to be able to estimate the state of a system based on a sequence of observations, taking into account both current measurements and the system dynamics [49]. This makes it very useful to estimate the hedge ratio γ. Initially, a book by Vidyamurthy discusses best practices for choosing cointegrated equities and found that the Kalman filter was found optimal when the state-space and observation equations are linear and the noise is Gaussian [50]. Since then there have been many extensions of the filter such as the Extended Kalman Filter (EKF) and Unscented KF aimed to handle when the state-space and observation equations are non-linear and the noise is not Gaussian.

The Kalman Filter works in 2 phases, prediction and update. The prediction phase is as follows

$$\hat{\mathbf{x}}_k = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} + \mathbf{B}_k \vec{\mathbf{u}}_k + \mathbf{w}_k$$

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

Where $\hat{\mathbf{x}}_k$ is the new best estimate (prediction) that is derived from $\hat{\mathbf{x}}_{k-1}$, the previous estimate and the prediction function $\mathbf{F}_k$. $\vec{\mathbf{u}}_k$ is the correction term, called the control vector, that is used when it is known that there are external influences in combination with $\mathbf{B}_k$ which is called the control matrix. In addition to this, the new uncertainty (covariance matrix), $\mathbf{P}_k$, is calculated using the previous uncertainty and additional uncertainty from the environment, $\mathbf{Q}_k$, $\mathbf{w}_k$ is called the state noise. The update is as follows

$$\hat{\mathbf{x}}'_k = \hat{\mathbf{x}}_k + \mathbf{K}'(\vec{\mathbf{z}}_k - \mathbf{H}_k \hat{\mathbf{x}}_k)$$

$$\mathbf{P}'_k = \mathbf{P}_k - \mathbf{K}' \mathbf{H}_k \mathbf{P}_k$$

$$\mathbf{K}' = \mathbf{P}_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

Where $\mathbf{K}'$ is defined as the Kalman gain, $\mathbf{H}_k$ is the measurement matrix, $\vec{\mathbf{z}}_k$ is mean of the observed values, which is also calculated by $\vec{\mathbf{z}}_k = \mathbf{H}_k \hat{\mathbf{x}}_k + \mathbf{v}_k$ where $\mathbf{v}_k$ is the measurement noise, and $\mathbf{R}_k$ is the covariance of the uncertainty of the observed values [51].

A paper that investigated the use of the Kalman Filter on ETFs found that the strategy it employed worked well for in-sample data points and worse, but still profitable, results for out-of-sample data [52]. The paper adapted the Kalman Filter to be able to use it for pairs trading to the following:

$$\mathbf{y}_t = \mathbf{x}_t \beta_t + \epsilon_t$$

$$\beta_t = \mathbf{I}\beta_{t-1} + \omega_t$$

Then calculating the Kalman Gain:

$$\text{Kalman Gain} = \frac{\text{Error in the estimate}}{\text{Error in the estimate} + \text{Error in the measurement}}$$

Then to calculate the estimate:

$$\text{Estimate}_t = \text{Estimate}_{t-1} + \text{Kalman Gain} \times (\text{Measurement - Estimate}_{t-1})$$

And finally, calculating the new error:

$$E_{\text{estimate}_t} = \frac{E_{\text{measurement}} \times E_{\text{estimate}_{t-1}}}{E_{\text{measurement}} + E_{\text{estimate}_{t-1}}}$$

$$E_{\text{estimate}_t} = E_{\text{estimate}_{t-1}} \times (1 - \text{Kalman Gain})$$

The author states "pairs trading strategies have gained widespread acceptance thus making profitability much more elusive" to justify the disappointing results, however, the author fails to find evidence or provide sufficient evidence to justify the claim [52].

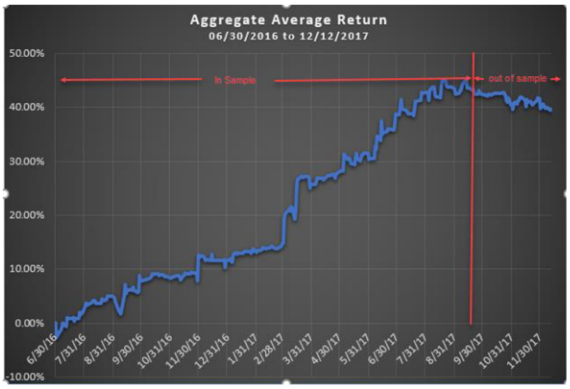

Figure 2.9: Aggregate average return of using the Kalman filter for pairs trading on ETFs [52]

Another paper used the combination of the Kalman Filter and Machine Learning, more specifically Extreme Learning Machine and Support Vector Regression (SVR) to build a statistical arbitrage strategy on the Brazilian Stock Exchange [53]. The strategies can simply be explained as using SVR and ELM to forecast returns and using the Kalman Filter to improve the forecast.

The paper also compares methods, such as LASSO, BMA, and GRR, to benchmark the performance of the Kalman Filter. The research found that using simply ELM and SVR forecasts results in a return of 20.19% and 21.32% respectively for out-of-sample data points and using a combination with the Kalman Filter gives a return of 26.13% for out-of-sample data points. The full results can be seen below in Figure 2.11. In addition to this, it can be seen that the volatility of the return also decreases which is ideal for investment managers.

Other papers/articles such as [54, 55, 56] have designed, compared and analysed other

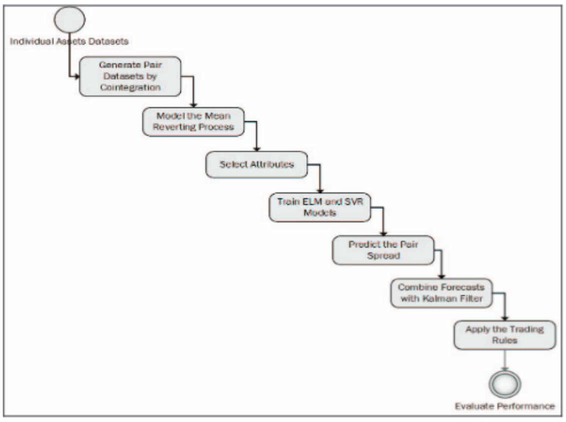

Figure 2.10: Visualisation of the trading strategy used in [53]

TABLE IV. ECONOMETRIC PERFORMANCE – ELM AND SVR – IN-SAMPLE

MODEL	MAX. DD	SHARPE	VOLATILITY	RETURN
ELM	-2.31%	1.80	**3.05%**	5.83%
SVR	**-2.18%**	1.73	3.20%	5.39%

TABLE V. ECONOMETRIC PERFORMANCE ELM AND SVR – OUT-OF-SAMPLE

MODEL	MAX. DD	SHARPE	VOLATILITY	RETURN
ELM	-2.80%	3.83	5.64%	20.18%
SVR	-2.72%	4.31	5.29%	21.32%

TABLE VI. ECONOMETRIC PERFORMANCE – COMBINATION MODELS– IN-SAMPLE

MODEL	MAX. DD	SHARPE	VOLATILITY	RETURN
BMA	-2.24%	1.38	**3.05%**	4.94%
GRR	-2.34%	**2.07**	3.20%	6.37%
KALMAN	-2.30%	1.97	3.57%	**6.89%**
LASSO	-2.43%	2.06	3.39%	6.30%

TABLE VII. ECONOMETRIC PERFORMANCE – COMBINATION MODELS – OUT-OF-SAMPLE

MODEL	MAX. DD	SHARPE	VOLATILITY	RETURN
BMA	-2.97%	3.83	**5.12%**	19.33%
GRR	**-2.53%**	4.76	5.49%	23.69%
KALMAN	-2.64%	**5.29**	5.17%	**26.13%**
LASSO	-2.64%	4.76	5.49%	23.79%

Figure 2.11: Econometric results [53]

statistical arbitrage techniques using Machine Learning algorithms and revealed that some algorithms are profitable. The majority of research on machine learning trading strategies has been on assets such as stocks on centralized exchanges. The little research that has been done on statistical arbitrage on cryptocurrencies has all been on analysing arbitrage on centralized exchanges and not decentralised exchanges. One of the research projects that analysed machine learning methods of statistical arbitrage on cryptocurrencies on a centralized exchange, compared a logistic regression approach with a random forest approach [56].

2.4.4 Analysis on Cryptocurrency Arbitrage on Centralized Exchanges

Although the research in the papers previously mentioned does not investigate the cointegration approach on cryptocurrencies, the takeaways are the mathematical fundamentals that are used in statistical arbitrage. Kristoufek and Bouri researched the sources of stat. arb. of bitcoin in multiple centralized exchanges. The Grey correlation is built on top of the Grey system theory [57], and can capture non-linear correlations without assuming a Gaussian distribution, thus using the Grey correlation provides a more robust metric to understand correlations between both series. The Grey correlation $\gamma(X_0, X_i)$ is defined with two steps:

1. $\gamma(x_0(k), x_i(k)) = \frac{min_i min_k|x_0(k)-x_i(k)|+\varepsilon max_i max_k|x_0(k)-x_i(k)|}{|x_0(k)-x_i(k)|+\varepsilon max_i max_k|x_0(k)-x_i(k)|}$

2. $\gamma(X_0, X_i) = \frac{1}{n}\sum_{i=1}^{n} \gamma(x_0(k), x_i(k))$

With $\varepsilon \in [0,1]$, the standard is set to $\varepsilon = 0.5$.

The DCC-GARCH(1,1), [58], model is also used to obtain conditional correlations for Bitcoin exchanges. The model was designed to use a combination of parameters such as the standard deviation of Bitcoin returns, traded volume, the volume of on-chain transactions, fees paid to miners, the ratio of current price and recent price history and internet hype/trends.

Upon analysis of Grey and DCC-GARCH(1,1) correlations, it is found that the DCC correlations show a little variability whereas the Grey correlations are a lot more variable ranging from 0.29 to 1. In addition, the paper then further investigates these sources and finds that these opportunities are introduced when there is a large number of inter-exchange transfer requests, i.e. the network is congested, and high price volatility. In contrast, the high volume of exchanges and on-chain activity cause the arbitrage opportunities to decrease. This paper finds and explains these sources of statistical arbitrage however does not implement or devise an algorithm that uses statistical arbitrage to generate a profit from price discrepancies of Bitcoin on different exchanges.

A paper that investigates statistical arbitrage on multiple cryptocurrencies is [59]. The authors of this paper analysed co-movements and cointegration of different cryptocurrencies on a centralized exchange using Augmented Dickey-Fuller (ADF) and Kwiatkowsky-Phillips-Schmidt-Shin (KPSS), Ljung-Box autocorrelation tests on both stationary forms ($I(0)$) and the original form ($I(1)$). The paper then develops a dynamic factor model based on the assumption that the price dynamics of cryptocurrencies are driven by Bitcoin [60], this is then evidenced by similar paths found in cryptocurrencies shown in Figure 2.12.

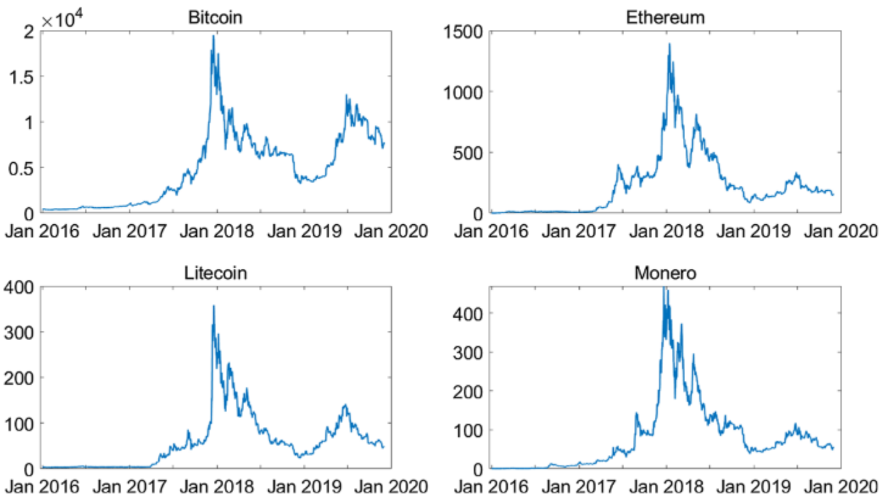

Figure 2.12: Price behaviour of Bitcoin, Ethereum, Litecoin, Monero [59]

For simplicity the authors set the number of hidden factors to 2 and upon analysis f_1 is a $I(1)$ process and the second factor f_2 is a stationary process that is independent of f_1. It is also found after overlaying f_1 with the price of Bitcoin, that the first factor strongly correlates with the price of Bitcoin.

The paper then uses this model to build an investment strategy, using forecasting using the estimated parameters:

$$\hat{p}_{i,\tau+1} = \mathbb{E}_\tau(p_{i,\tau+1}) = \hat{\alpha}_i + \hat{\beta}_{i1}\mathbb{E}_\tau(f_{1,\tau+1}) + \hat{\beta}_{i2}\mathbb{E}_\tau(f_{2,\tau+1})$$

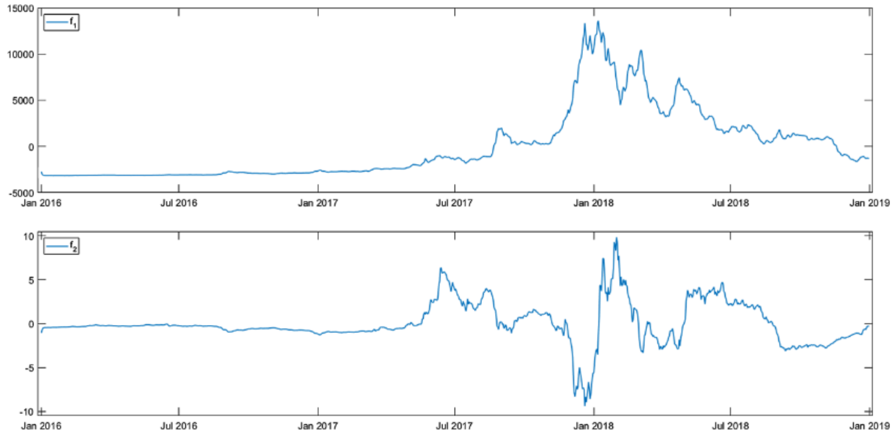

Figure 2.13: Hidden factors f_1 and f_2 from Jan 2016 to Dec 2018 [59]

Where

$$f_{1,t} = \lambda_1 f_{1,t-1} + \eta_{1,t}$$

$$f_{2,t} = \lambda_2 f_{2,t-1} + \eta_{2,t}$$

The expected gains one day ahead are given by:

$$g_{\tau+1} = \mathbb{E}_\tau[v_{\tau+1}] = \sum_{i=1}^{\lfloor I/2 \rfloor} \hat{p}_{\tau+1}^{(i)} - \sum_{i=\lceil I/2 \rceil+1}^{I} \hat{p}_{\tau+1}^{(i)}$$

Using this and a threshold which is calculated by the combination of the current price and standard deviation of the trading position value:

- if $g_{\tau+1} > v_\tau + c\sigma_\tau^v$, go long
- if $g_{\tau+1} < v_\tau - c\sigma_\tau^v$, go short
- if $v_\tau - c\sigma_\tau^v \leq g_{\tau+1} \geq v_\tau + c\sigma_\tau^v$, no trade

The researchers of the paper evaluated their trading strategy for 334 days and a moving window of 3 years, 1096 observations, every day to estimate the parameters for the dynamic factor model. We can see in Figure 2.14 that the strategy was able to consistently generate a profit even when considering transaction costs.

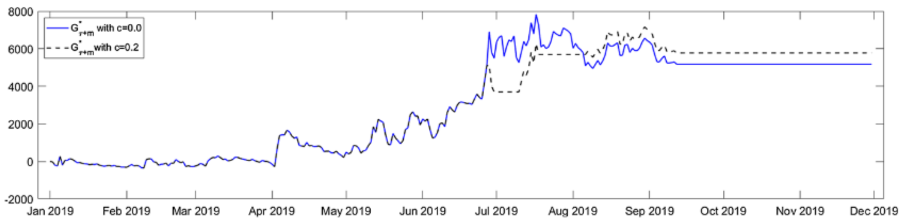

Figure 2.14: Net gains taking transaction fees into account [59]

Chapter 3

Protocols of Interest

3.1 Uniswap

3.1.1 Overview

Uniswap is a decentralized exchange protocol built on the Ethereum blockchain. It allows users to trade ERC-20 tokens directly from their wallets without the need for intermediaries or traditional order books. Uniswap utilizes automated market-making, where liquidity providers contribute funds to liquidity pools, earning fees on trades made in the pool. The protocol employs a mathematical formula called the constant product formula to maintain balanced token ratios in the pool. When users want to make a trade, Uniswap calculates the conversion based on the pool's token ratios and executes the trade through a smart contract.

3.1.2 How it works

Automated Market Maker (AMM) Model and Liquidity Pools

The AMM model is a system that replaces traditional order books with liquidity pools to facilitate trading between different tokens. Automated Market Makers do this with the aid of liquidity pools. They are pools of tokens contributed by liquidity providers (LPs) to facilitate trading between different tokens within the exchange. One paper describes pools as "a smart contract that holds at least two cryptoassets and allows trading through depositing a token of one type and thereby withdrawing tokens of the other type" [61]. These liquidity pools are maintained by smart contracts on Ethereum and Liquidity Providers (LPs). Liquidity providers are individuals that voluntarily contribute an equal amount of cryptocurrencies liquidity pools, for example, in a pool for trading ETH and DAI, LPs would contribute an equal value of ETH and DAI tokens. LPs are incentivised to provide liquidity in exchange for a share of the trading fees that occur in the liquidity pool. LPs can later withdraw their shares along with the accumulated fees. On Uniswap V3, the fee is 0.3% on Ethereum, however they can be any of 0.01%, 0.05%, 0.3%, or 1% depending on blockchain network.

Constant Product Formula

To maintain a balanced ratio between the tokens in the pool and price any swaps, the AMM model relies on a mathematical formula called the constant product formula ($xy = k$). As trades occur, the product of the token balances remains constant. When one token's value increases, its proportion in the pool decreases, ensuring an automatic adjustment in prices. When a trade is executed, the change in prices can be described in this formula $(x + \Delta x)(y + \Delta y) = k$. Hence, after rearranging: $\Delta y = \frac{k}{x+\Delta x} - y$. Under this model, the balance of the tokens in a liquidity pool can never be depleted as the token will get infinitely more expensive as the reserves approach 0.

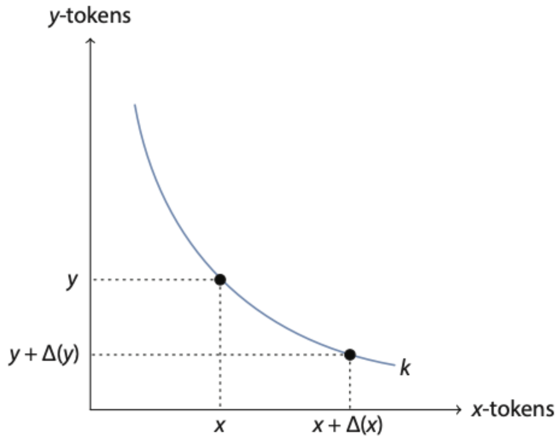

Figure 3.1: Constant Product Formula [61]

Slippage

As seen in Figure 3.2, we can see how the constant product formula is used in Uniswap. Uniswap also allow users to set slippage tolerance levels, which determine the maximum acceptable difference between the expected and executed prices. Slippage refers to the difference between the expected price of a trade and the executed price due to market volatility and liquidity conditions. Slippage happens because the constant product formula adjusts prices based on the ratio of tokens in the pool. As trades are executed, the token balances change, and the prices change accordingly. Thus, larger trades can cause more significant price impact, resulting in slippage.

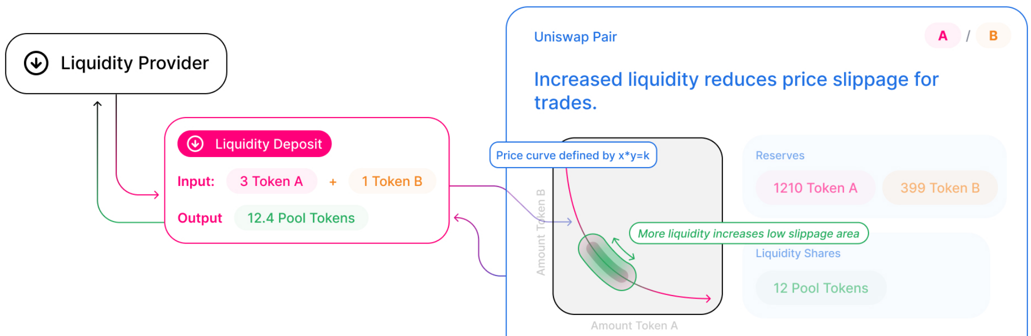

Figure 3.2: How Uniswap works [62]

3.2 Aave

3.2.1 Overview

Aave is a decentralized lending and borrowing protocol built on the Ethereum blockchain. It enables users to lend and borrow a wide range of cryptocurrencies directly, without the need for intermediaries such as banks. Aave operates through liquidity pools and smart contracts, providing a secure, transparent, and efficient platform for decentralized finance (DeFi) activities.

Users can deposit their cryptocurrency assets into Aave's liquidity pools and earn interest on their deposits. These funds contribute to the available liquidity for borrowers. On the other hand, borrowers can use their deposited assets as collateral to borrow other assets from the pool. The amount they can borrow is determined by the value of their collateral and specific borrowing parameters set by the protocol.

3.2.2 How it works

Liquidity Pools

The fundamental mechanism that enables Aave's functionality of lending and borrowing is liquidity pools. Users deposit their cryptocurrency assets into Aave's liquidity pools. These assets serve as collateral and contribute to the overall liquidity of the protocol. The deposited assets in the liquidity pools create reserves of available liquidity. These reserves are utilized to fulfill borrowing demands from other users within the Aave ecosystem.

Lending and Borrowing

Lending works by lenders depositing their cryptocurrency assets into Aave's liquidity pools. These assets act as collateral and are represented by interest-bearing tokens called aTokens. The aTokens represent the user's share of the deposited assets. Interest is earnt immediately and is accrued in real-time and reflected in an increase in the quantity of aTokens held by the depositor.

Borrowing works by borrowers using their deposited assets as collateral to borrow other assets from the liquidity pools. The value of the collateral determines the borrowing capacity. The borrowing capacity is calculated by paramters set by Aave, one of which is the maximum loan-to-value (LTV) ratio. Once the borrower requests a loan, if the borrower's collateral meets the necessary requirements, they can proceed with the loan and the borrowed funds are transferred to the borrower's wallet. In addition to this, borrowers pay interest on the borrowed amount based on the prevailing interest rates. Aave offers both variable and stable interest rates for borrowers. Variable rates fluctuate based on market dynamics, while stable rates remain fixed. This flexibility allows users to choose the borrowing option that best suits their needs. The interest on a loan is accrued in real-time, second by second, and the borrower decides when to repay it, as long as the loan is safe from liquidation. Liquidation is what happens if the value of a borrower's collateral falls below the liquidation threshold due to market volatility or other factors, the collateral may be liquidated. Liquidators can purchase the collateral at a discounted price to ensure the solvency of the liquidity pool.

Chapter 4

Implementation of the Trading Systems

This section provides a comprehensive overview of the two trading system implementations. The first one focuses on the backtesting system, which involves simulating and evaluating strategies in a controlled environment. In this system, historical market data is utilized to test and analyze the performance of various strategies. The second implementation is the live system, designed to execute trades on the Ethereum blockchain using smart contracts. This system operates in real-time and interacts with the live market, allowing for actual trades to be executed based on predefined strategies. Together, these two implementations provide a comprehensive framework for testing, evaluating, and deploying trading strategies, both in simulated environments and on the live blockchain.

4.1 Buying and Selling

The process of buying and selling in traditional markets is relatively straightforward as brokers and exchanges play a crucial role in executing orders on behalf of individuals. However, when it comes to trading cryptocurrencies, the responsibility falls directly on the trader, such as myself. Therefore, it becomes essential to clearly define what buying and selling entail in the context of cryptocurrency trading.

4.1.1 Buying

Prices on Uniswap are represented as ratios for example 1 USDC $= P$ WETH. With this understanding, let's delve into the process of buying one unit of USDC/WETH on Uniswap:

- Opening a Buy position:
  - Starts with P_0 WETH
  - Swaps the WETH for USDC, hence ends with 1 USDC
- Closing a Buy position:
  - Starts with 1 USDC
  - Swaps the USDC for WETH, hence ends with P_1 WETH

Consequently, if the price of USDC/WETH increases from the moment the buy position is opened to the time it is closed, the trader realizes a profit. On the other hand, if the price declines during this period, the trader incurs a loss.

4.1.2 Selling

Selling an asset is a more complex process compared to buying because it involves trading an asset that the trader doesn't initially possess. In traditional markets, this is facilitated by the trader borrowing the desired asset from a broker or another party. When the trader decides to close the position, they repurchase the same amount of the borrowed asset and return it to the lender, hoping that its value has declined. This borrowing and returning process is typically managed automatically by brokers and exchanges. However, in decentralized exchanges (DEXes), such mechanisms are not in place.

To simplify this process, I have opted to utilize Aave as a lending platform to borrow any required assets. It is worth noting that Aave supports only a limited range of tokens available for borrowing, namely DAI, EURS, USDC, USDT, AAVE, LINK, WBTC, and WETH. Therefore, shorting would be feasible only if I focus on liquidity pools that involve these cryptocurrencies. By leveraging Aave as a lending platform, I can access the necessary assets for shorting. The process of selling is as follows:

- Opening a Sell position:
  - Borrow 1 USDC
  - Swaps the USDC for WETH, hence ends with P_0 WETH
- Closing a Sell position:
  - Starts with P_0 WETH
  - Swaps the WETH for USDC, hence ends with $\frac{P_0}{P_1}$ USDC
  - Return the borrowed USDC, leaving $\frac{P_0}{P_1} - 1$ USDC

Consequently, if the price of USDC/WETH decreases, i.e. $P_1 < P_0$ from the moment the sell position is opened to the time it is closed, the trader realizes a profit. On the other hand, if the price increases during this period, the trader incurs a loss.

Note that this is is merely an illustrative example and does not account for any potential fees that the trader might incur.

4.2 Backtesting System

To develop a resilient backtesting system, a dedicated class was constructed to streamline the process of testing trading strategies. Upon evaluating a particular strategy the `backtest` function requires `cointegrated_pair`, a tuple of the names of the liquidity pools that the strategy is to be evaluated on, `strategy`, the strategy, and finally, the `initial_investment` in ETH. The first argument is required and used to retrieve the relevant historical prices. The second parameter is self-explanatory, the strategy to test. Finally, the inclusion of the initial investment amount as an input parameter enables users to simulate the performance of their strategies with a specific starting capital, it also helps analyze how the various transaction costs affect the ability to trade if any trades do result in a loss. The `backtest` function iterates through historical data calling the strategy's `generate_signal` and executing the trade orders it receives at each timestep. However, to do this historical data is required hence, the first step is to collect the historical data.

4.2.1 Data Collection and Storage

In order to simulate the market as accurately as possible the system should have access to reliable and accurate historical market data, including price, volume, and other relevant indicators. Therefore, I retrieve all of my data from the Uniswap and Aave protocols' subgraph using the Graph [63]. The Graph is a decentralized protocol for indexing and querying blockchain data hence making the data provided 100% reliable as it indexes directly on the Ethereum blockchain. Subgraphs serve as GraphQL APIs designed to facilitate querying and extracting data from the blockchain. These APIs adhere to a specific schema outlined by the protocol, enabling seamless communication between the protocol and the underlying blockchain. Therefore, I employ the Uniswap V3, Aave V2, and Aave V3 subgraphs to obtain the data required for the backtesting system.

The database has the following tables: The `liquidity_pools` contains data about the

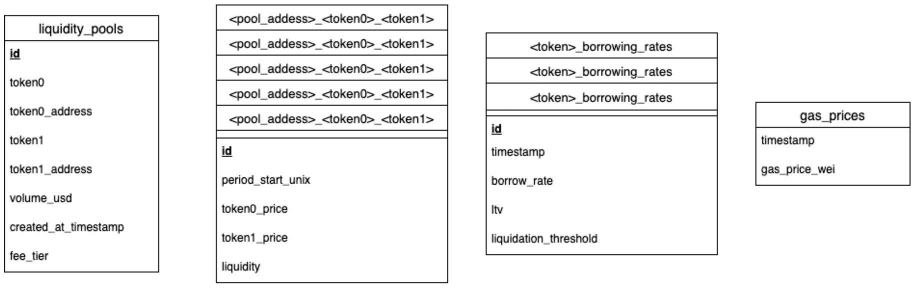

Figure 4.1: Tables contained in the database

liquidity pools that exist on Uniswap V3. After obtaining all of the data it is found that it possesses 12,182 liquidity pools. However, a substantial portion of these pools exhibit minimal or negligible trading volume. As a result, a criterion is established to selectively include only those liquidity pools that involve tokens supported by Aave and possess a trading volume exceeding $10,000,000 (or $10 million). This filtering condition ensures that the collected and stored data holds significance and relevance for research purposes, as these pools would allow for short selling.

Once these pools have been identified, pricing data about each pool that meets this condition is collected again using the Uniswap V3 subgraph. The following shows the GraphQL query:

```
query ($id: ID!, $prev_max_time: Int!) {
    pool(id: $id) {
        poolHourData(where: {periodStartUnix_gt: $prev_max_time}, orderBy:
    periodStartUnix, first: 1000) {
            id
            token0Price
            token1Price
            periodStartUnix
            liquidity
            feesUSD
        }
    }
}
```

This query returns an array of dictionaries containing the pre-specified pricing datapoints at a frequency of every hour. Due to the limitations imposed by the subgraph, the results are constrained to a maximum size of 1000 entries. To overcome this limitation and obtain the complete dataset, the query is executed iteratively. The previous maximum time, referred to as `prev_max_time`, is passed as an argument in subsequent queries to fetch the remaining data, i.e. `prev_max_time = hourlyData[-1]['periodStartUnix'] if len(hourlyData) > 0 else prev_max_time`. This data is stored in tables of the form *<pool_address>_<token0>_<token1>*.

In a similar manner, obtaining the interest rates for borrowing necessitates the utilization of two of Aave's subgraphs. This requirement arises due to Aave's migration to Version 3 in March 2022, resulting in a transitional period where both Uniswap V3 and Aave V2 were concurrently utilized. The schemas for the two are different, however for the data we require, the borrowing rate, loan-to-value and liquidity threshold, the schema is consistent and the same query can be used:

```
query ($symbol: String!, $prev_max_time: Int!) {
    reserves(where: {symbol: $symbol}) {
        id
        symbol
        lifetimeBorrows
        baseLTVasCollateral
        reserveLiquidationThreshold
        borrowHistory(
            where: {timestamp_gt: $prev_max_time}, first: 1000, orderBy: timestamp, orderDirection: asc) {
            id
            timestamp
            borrowRate
        }
    }
}
```

During the table initialization process, requests are made to both the V2 and V3 GraphQL endpoints. However, when updating the table, only the V3 endpoint is utilized for sending requests. The data is stored in tables of the form *<symbol>*_borrowing_rates, where *<symbol>* are all of the tokens that are present in the liquidity pools that are of interest.

To collect the gas price history, the transaction history is queried at each hour since Uniswap migrated to V3. This is because querying in the same as the pricing data and borrowing rate history is too exhaustive as transactions occur every second. Therefore, it is more efficient to query at each hour with a window to retrieve the gas price at the closest hour as follows:

```
query ($min_time: Int!, $max_time: Int!) {
    transactions(where: {timestamp_gt: $min_time, timestamp_lt: $max_time}, first:1000, orderBy: timestamp, orderDirection: asc) {
        id
        timestamp
        gasPrice
    }
}
```

The following pseudocode shows how to fetch the gas price at each hour:

4.2.2 Types of Orders and Execution

There are numerous order types that the backtesting supports due to measures to enusre a positive balance and avoid liquidation. The types of orders are; `BUY ETH`, `WITHDRAW`, `DEPOSIT`, `SWAP`, `OPEN BUY`, `CLOSE BUY`, `OPEN SELL`, `CLOSE SELL`. The precedures of each are outlined below.

`BUY ETH`

The `BUY ETH` order is used to swap a percentage of the accounts balance from WETH to ETH. The arguments is an float between 0 and 1. The logic of the execution, excluding

Algorithm 5 Retrieval of hourly gas prices where min_time & max_time are arguments

$rows\_set \leftarrow \{\}$
for $timestamp$ from min_time to $max\_time + (60 \times 60), step\_size = (60 \times 60)$ **do**
  $found\_result \leftarrow False$
  $window\_size\_in\_minutes \leftarrow 5$
  while *not* $found\_result$ **do**
    $transaction\_data \leftarrow$ GraphQL Query with arguments: $\{``min\_time" : timestamp - (60 * window\_size\_in\_minutes), ``max\_time" : timestamp + (60 * window\_size\_in\_minutes)\}$
    if $transaction\_data! = 0$ **then**
      $transaction\_data\_sorted = sorted(transaction\_data, key = lambda\ x : abs(timestamp - int(x['timestamp'])))$
      $rows\_set.update(timestamp : (timestamp, transaction\_data\_sorted[0]['gasPrice']))$
      $found\_result \leftarrow True$
    else
      $window\_size\_in\_minutes \leftarrow window\_size\_in\_minutes + 5$
    end if
  end while
  insert $rows\_set[timestamp]$ into `gas_prices` table
end for

gas fees, of the order is outlined below.

```
amount_to_swap = self.account['WETH'] * order[1]
self.account['WETH'] = self.account['WETH'] - amount_to_swap
self.account['ETH'] = self.account['ETH'] + amount_to_swap
```

SWAP

The `SWAP` order is to execute a simple swap for either token0 or token1 of the liquidity pool. This order is only used as a precautionary order, if the balance of the WETH balance gets too low. It takes 2 arguments, the first indicating whether it is swapping for token0 or token1 and the second parameter is a list of swaps, tuples containing with pool and the quantity, that would like to be swapped. As can be seen in the code excerpt below, the amount received in the account is multiplied by `(1 - swap_fees[swap_token])`, this is because Uniswap charges a percentage of the swap specified by `swap_fees[swap_token]`.

```
if is_for_token0:
    for swap_for_token0 in swaps:
        swap_token, swap_volume = swap_for_token0
        self.account['WETH'] = self.account['WETH'] - (swap_volume * prices
    [f'P{swap_token[1]}'])
        self.account[swap_token] = self.account[swap_token] + (swap_volume
    * (1 - swap_fees[swap_token]))
else:
    for swap_for_token1 in swaps:
        swap_token, swap_volume = swap_for_token1
        self.account[swap_token] = self.account[swap_token] - (swap_volume
    / prices[f'P{swap_token[1]}'])
        self.account['WETH'] = self.account['WETH'] + (swap_volume * (1 -
    swap_fees[swap_token]))
```

OPEN BUY

Similar to the SWAP order, OPEN BUY opens a buy position. As mentioned above opening a buy order is simply swapping token1 for token0. The parameters of buying are the target token and the volume.

```
self.account['WETH'] = self.account['WETH'] - (volume * buy_price)
self.account[token] = self.account[token] + (volume * (1 - swap_fees[token
    ]))
```

CLOSE BUY

Closing a buy position is similar to opening a buy position, however, the swap is in the other direction, i.e. from token to WETH. The only argument is the id of the buy order. To account for Uniswap's fees, the volume that was received, after the initial swap, is calculated then this volume is swapped back to WETH.

```
buy_token, bought_price, buy_volume, buy_timestamp = self.open_positions['
    BUY'][buy_id]
volume_bought = buy_volume * (1 - swap_fees[buy_token])
self.account[buy_token] = self.account[buy_token] - volume_bought
self.account['WETH'] = self.account['WETH'] + (prices[f'P{buy_token[1]}'] *
     (volume_bought * (1 - swap_fees[buy_token])))
```

OPEN SELL

Opening a sell position involves borrowing a token and then swapping the token. The parameters of selling are the target token and the volume. It also consists of depositing the required amount to borrow the volume ordered, calculated by the value to loan ratio.

```
# Borrow token and Deposit collateral
amount_to_move_to_collateral_WETH = ((volume * sell_price) / vtl_eth)
self.account[token] = self.account[token] + volume
self.account['WETH'] = self.account['WETH'] -
    amount_to_move_to_collateral_WETH
self.account['collateral_WETH'] = self.account['collateral_WETH'] +
    amount_to_move_to_collateral_WETH

# Swap borrowed tokens to WETH
self.account[token] = self.account[token] - volume
self.account['WETH'] = self.account['WETH'] + (volume * (1 - swap_fees[
    token]) * sell_price)
```

CLOSE SELL

Closing a sell position is more complicated that as in requires to swap back from the token to WETH, repay the loan with interest and finally return collateral. Swapping back to the required amount of tokens is the first step. For this the variable Annual Yield Rates between the timestamp of the opening of the position and closing positon is used to calculate the amount of interest required to be paid as this is cumulated every second. Once the volume of tokens required to be returned is calculated, the equivelent amount of WETH plus accounting for Uniswap fees is swapped to obtain these tokens. Finally, the tokens are repayed and the collateral is returned.

```
sell_token, sold_price, sell_volume, sell_timestamp = self.open_positions['SELL'][sell_id]

# Swap WETH back to Token
volume_required_to_return = sell_volume
previous_timestamp = sell_timestamp

for apy_idx in apy[sell_token].index:
    local_apy = apy[sell_token].loc[apy_idx]['borrow_rate']
    number_of_seconds = apy[sell_token].loc[apy_idx]['timestamp'] - previous_timestamp
    secondly_yield = (1 + local_apy)**(1 / (365*24*60*60))
    volume_required_to_return *= secondly_yield ** number_of_seconds
    previous_timestamp = apy[sell_token].loc[apy_idx]['timestamp']

self.account['WETH'] = self.account['WETH'] - (sold_price * volume_required_to_return / (1 - swap_fees[sell_token]))
self.account[sell_token] = self.account[sell_token] + volume_required_to_return

# Return Borrowed Tokens and Collatoral
self.account[sell_token] = self.account[sell_token] - volume_required_to_return
self.account['WETH'] = self.account['WETH'] + self.account['collateral_WETH']
self.account['collateral_WETH'] = 0
```

`WITHDRAW`

The `WITHDRAW` order's function is to withdraw some collateral that is stored in Aave. It is not used in the strategies however is implemented in case of developing further strategies that may require this functionality. The only parameter is the amount that would like to be withdrawn.

```
self.account['WETH'] = self.account['WETH'] + withdraw_amount
self.account['collateral_WETH'] = self.account['collateral_WETH'] - withdraw_amount
```

`DEPOSIT`

The `DEPOSIT` order's function is to deposit some additional collateral that is stored in Aave. This order is used when the collateral value not properly covering the loan value, hence avoiding liquidation. The only parameter is the amount that would like to be depositted.

```
self.account['WETH'] = self.account['WETH'] - deposit_amount
self.account['collateral_WETH'] = self.account['collateral_WETH'] + deposit_amount
```

4.2.3 Gas Fees - TODO

4.2.4 Validating Balance Health

4.3 Live Trading - TODO

Chapter 5

Key Decisions and Outline of Trading Strategies

5.1 Liquidity Pools

As previously mentioned, Uniswap is a decentralized exchange protocol that facilitates swapping of cryptocurrencies through the use of liquidity pools. One of the remarkable features of Uniswap is its support for even the smallest cryptocurrencies. In fact, it currently possesses 12,182 liquidity pools, as obtained via the Uniswap V3 subgraph. The subgraph provides valuable insights into the liquidity pools within the Uniswap ecosystem.

Table 5.1 showcases the diversity of these pools. Some pools exhibit high trading volumes, indicating significant activity and demand for those specific pairs. On the other hand, there are also pools with zero liquidity, which could be attributed to various reasons. It could be due to a lack of demand for one of the token pairs offered in the pool, or it may indicate that the pool is relatively new and has not attracted significant participation yet.

To enhance the chances of successful swaps and maximize potential profitability, my strategy is centered on liquidity pools with a trading volume surpassing \$10,000,000 (or \$10 million) and pools that include tokens supported by AAVE. Additionally, I exclude any liquidity pools that do not involve WETH, which is a tokenized form of ETH (Ethereum's native cryptocurrency). WETH's key advantage lies in its ability to align with ERC-20 standards, enabling broader compatibility and utilization across the Ethereum ecosystem.

5.1.1 Correlated and Cointegrated Liquidity Pools

Once the liquidity pools of interested in have been identified, it is crucial to filter the pool pairs based on their correlation. This is because correlated pairs tend to exhibit similar pricing movements thus would be possible to apply the mean reversion trading strategies. However, it is important to strike a balance between a highly correlated pair and a low correlated pair, as excessively high correlation can result in minimal price deviations, leading to fees that exceed the deviations and resulting in overall losses instead of profits. To address this, I have set a condition where the correlation coefficient (ρ) should fall within the range of $0.990 \leq \rho \leq 0.997$.

To determine which liquidity pools are cointegrated, I employed the Engle-Granger approach. The Engle-Granger test for cointegration involves several steps. Firstly, a unit root test is performed individually on each time series using methods like the Augmented Dickey-Fuller (ADF) test or the Phillips-Perron (PP) test. These tests assess whether the time series are stationary or exhibit unit roots (non-stationary) individually. For cointegration to hold, both time series must be non-stationary.

pool address	token0	token1	volume in USD	created at timestamp	feetier
0x88e6a0c2d...	USDC	WETH	375230561243.465	1620250931	500
0x8ad599c3...	USDC	WETH	70454095868.0967	1620169800	3000
0x11b815efb...	WETH	USDT	62385006691.8387	1620251172	500
0x3416cf6c7...	USDC	USDT	57192593471.8346	1636825557	100
0x4585fe772...	WBTC	WETH	49170385539.9928	1620246230	500
0x4e68ccd3e...	WETH	USDT	30135014933.0963	1620232628	3000
0x60594a40...	DAI	WETH	26075053939.434	1620237823	500
0xcbcdf9626...	WBTC	WETH	21870989841.1326	1620158974	3000
0x5777d92f2...	DAI	USDC	16143305036.8948	1636771503	100
0x7858e59e0...	USDC	USDT	15473402409.0591	1620159478	500
0x99ac8ca70...	WBTC	USDC	12568187132.1649	1620241995	3000
0xc2e9f25be...	DAI	WETH	12519316091.9979	1620159368	3000
0xe0554a476...	USDC	WETH	9381529300.20357	1636926269	100
0x6c6bc977e...	DAI	USDC	7219493916.70291	1620158293	500
0xac4b3dac...	APE	WETH	6621032721.87519	1647516735	3000
0x8c54aa2a3...	FEI	USDC	6206853090.73714	1621839430	500
0x53dd58b3...	sOHM	gOHM	0	1652914688	10000
0xbc90c4de...	SHIB	NSTIC	0	1652910391	100
0xaa1297b0...	BUSD	DPC	0	1674655715	100
0x75087e533...	DAI	ICAP	0	1652899566	10000
0xc65a68019...	stkAAVE	FRAX	0	1652817927	10000
0x94589b18...	VVV	SOL	0	1674701603	10000
0xf42f0def92...	APEFI	ApeUSD	0	1652808878	3000
0xb9ba65f15...	FRAX	ApeUSD	0	1652808680	500
0x1dfb167f1...	GHD	WETH	0	1652784327	3000

Table 5.1: A selection of liquidity pools on Uniswap

Once it is established that both time series are non-stationary, an estimation of the cointegration equation is derived. Typically, a linear regression model is employed to capture the long-term relationship between the time series variables. This equation provides an understanding of how the variables are linked over a long period.

Subsequently, a unit root test is conducted on the residuals obtained from the regression analysis. If the residuals are stationary (indicating the absence of unit roots), it suggests the presence of cointegration between the time series. This implies that the variables move together in the long run, even if short-term deviations occur.

By following the aforementioned sequence of steps in the Engle-Granger test, it enables us to determine which liquidity pools demonstrate cointegration, thereby emphasizing their interconnectedness and mutual relationship over time. The outcome of the cointegration tests for all possible combinations of liquidity pools is presented in Table 5.2. This table provides a comprehensive list of the liquidity pool pairs along with the results of their respective cointegration tests and the correlation coefficient of each combination.
Based on the results obtained, we focus our further investigation solely on the pairs of liquidity pools that exhibit cointegration under a 1% confidence level. This filtering process allows us to narrow down our analysis to those pairs where a long-term relationship exists, suggesting that they move together in a concerted manner.

By concentrating on the cointegrated pairs, we can delve deeper into exploring their dynam-

pool1	pool2	t-statistic of unit-root test on residuals	Critical Values			Corr Coeff
			1%	5%	10%	
USDC_WETH_0x88e6...	USDC_WETH_0xe055...	-11.280314	-3.898076	-3.337043	-3.045083	0.995244
USDC_WETH_0x8ad...	USDC_WETH_0xe055...	-11.183921	-3.898089	-3.33705	-3.045089	0.995049
WETH_USDT_0x11b...	USDC_WETH_0xe055...	-9.988616	-3.898076	-3.337043	-3.045083	0.995075
WETH_USDT_0x4e68...	USDC_WETH_0xe055...	-9.890807	-3.898088	-3.337049	-3.045088	0.994951
DAI_WETH_0x60594...	USDC_WETH_0xe055...	-11.369096	-3.898076	-3.337043	-3.045083	0.995187
DAI_WETH_0xc2e9f2...	USDC_WETH_0xe055...	-10.398429	-3.898317	-3.337177	-3.045177	0.995004
USDC_WETH_0xe055...	USDC_WETH_0x7be...	-7.561677	-3.901185	-3.338775	-3.046286	0.994488
USDC_WETH_0xe055...	WETH_USDT_0xc5af...	-9.76086	-3.90143	-3.338911	-3.04638	0.994551
USDC_WETH_0xe055...	DAI_WETH_0xa8096...	-6.91588	-3.905801	-3.341344	-3.048068	0.994107

Table 5.2: Cointegration test on Liquidity Pool pairs

ics, assessing their behavior, and leverage the interconnectedness between these liquidity pools. This approach enables us to prioritize our analysis and concentrate on the most relevant liquidity pool combinations that offer potential opportunities for profitable trading or other related activities.

5.2 Strategy

As previously mentioned, the strategy I employ is the mean reversion trading strategy. The basic concept of a mean reversion strategy identifying two closely related assets or securities, typically referred to as a pair, and taking advantage of deviations from their historical price relationship. This works by initially identifying a pair of assets that have a historically stable relationship as I have done in Section 5.1. Then as prices change you calculate the spread, which is the difference between their prices. Look for deviations from the historical mean spread, upon a deviation one buys the undervalued asset and sells the other. The positions are then closed once the spread reverts back to its historical mean.

5.2.1 Hedge Ratio

In the mean reversion strategy, the hedge ratio refers to the ratio or proportion between the positions of two assets involved in a pairs trading strategy. It determines the optimal allocation of capital between the assets to minimize risk and create a market-neutral position. The hedge ratio represents the number of units or shares of one asset that should be held for each unit or share of the other asset in order to create a balanced or hedged position. It is derived through statistical techniques such as regression analysis.

Mean Reversion Strategy

In the simple mean reversion strategy, the approach relies on a given historical dataset, assuming that the hedge ratio between the paired assets remains consistent over the long term. To determine this hedge ratio, the Ordinary Least Squares (OLS) regression method is employed. In Python, the calculation can be carried out using the following steps:

```
model = sm.OLS(history_p1, sm.add_constant(history_p2))
results = model.fit()
# Gradient of the OLS i.e. X = results.params[0] + results.params[1] * '
    p2_token1_price'
```

```
hedge_ratio = params[1]
```

In Figure 5.1, the observed trend reveals that the gradient, representing the rate of change, exhibits a proximity to the value of 1. This indicates a relatively balanced relationship between the prices of each pair. However, it is important to note that as time progresses, any fluctuations or deviations from the exact value of 1 are minimal and have a negligible impact on the overall trend. The stability of the gradient over time suggests a consistent and relatively stable relationship between the liquidity pools, reinforcing the notion that their interdependence remains relatively constant.

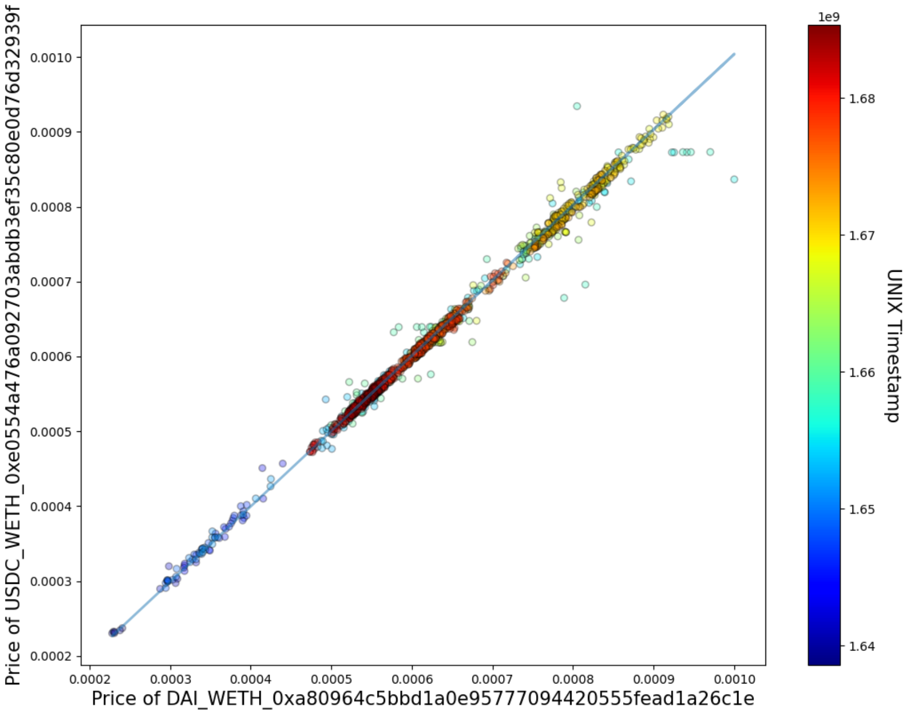

Figure 5.1: Conducting OLS to obtain the Hedge Ratio

Kalman Filter

As mentioned earlier, the hedge ratio plays a crucial role in minimizing risk and achieving a market-neutral position in trading strategies. However, relying on a fixed hedge ratio may not be ideal, as market dynamics and the relationship between the paired assets can change over time. To address this challenge, I employ the Kalman Filter as a dynamic tool to update and adapt the hedge ratio, taking into account the changing market conditions and the evolving relationship between the paired assets. This allows for a more responsive and adaptive approach to maintaining a market-neutral position. The Kalman Filter takes into consideration not only the current price data but also the historical information and the underlying dynamics of the assets. It provides a more robust and accurate estimation of the optimal hedge ratio, which aligns with the mean reversion strategy's objective of capturing potential price divergences and profiting from their eventual convergence.

The Kalman Filter works by iteratively updating and refining its estimate of the system state using a prediction step and an update step. For this an initial 'guess' of the parameters that I use is the using the Ordinary Least Squares from the historical information provided. The remainder of the parameters that are selected can be seen below:

```
model = sm.OLS(price_history_2, sm.add_constant(price_history_1))
```

```
initial_state = model.fit().params[::-1]

kf = KalmanFilter(
    n_dim_state=2,
    initial_state_mean=initial_state,
    transition_matrices=np.eye(2),
    observation_matrices=obs_mat,
    transition_covariance=1e-5 * np.eye(2)
    )
```

Below describes the reasoning behind the choices of parameters:

- `n_dim_state` - This parameter to the number of elements in the state. In our scenario, the state consists of two components: the y-intercept and the gradient.
- `initial_state_mean` - As mentioned earlier, the Kalman Filter utilizes past estimates to iteratively estimate the true states. Therefore, I employ the OLS method to calculate the initial state of the price history's regression, which returns both the y-intercept and gradient.
- `observation_matrices` - The observation matrix defines the relationship between the observed measurements and the hidden state variables. Thus takes the historical data that is provided zipped along with 1 as the observed measurements directly correspond to the hedge ratio. This can be represented as $\begin{bmatrix} y_1 & 1 \\ y_1 & 1 \\ \vdots & 1 \\ y_n & 1 \end{bmatrix}$.
- `transition_matrices` - In the context of our strategy, we anticipate a consistent long-term hedge ratio. Therefore, we set the `transition_matrices` parameter to $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
- `transition_covariance` - Finally the transition_covariance specifies the covariance matrix of the process noise. It represents the uncertainty or variability in the state transition, hence it is set to be very small, $\begin{bmatrix} 10^{-5} & 0 \\ 0 & 10^{-5} \end{bmatrix}$.

Figures 5.2 and 5.3 provide visual representations of the hedge ratio's evolution over time. These figures illustrate that the hedge ratio, while exhibiting subtle variations, does indeed change over the observed period. One notable event is the significant drop that occurs at the timestamp 1.655e9 or June 2022. This particular shift in the hedge ratio is likely attributed to the migration from the Proof of Work (PoW) consensus mechanism to the Proof of Stake (PoS) consensus mechanism on the Ethereum network.

During the transition from PoW to PoS, there are fundamental changes in how the Ethereum network reaches consensus and validates transactions. This change in consensus mechanism can impact various aspects of the network, including the dynamics of the hedge ratio. It is plausible that the observed drop in the hedge ratio during the migration period reflects the adjustments and reconfiguration of the underlying mechanisms in response to the shift to PoS.

5.2.2 Fees

As previously mentioned, when engaging in trading activities, there are several fees that are deducted. These fees serve multiple purposes, including compensating the liquidity providers and addressing the potential lack of access to assets from the lender. Therefore,

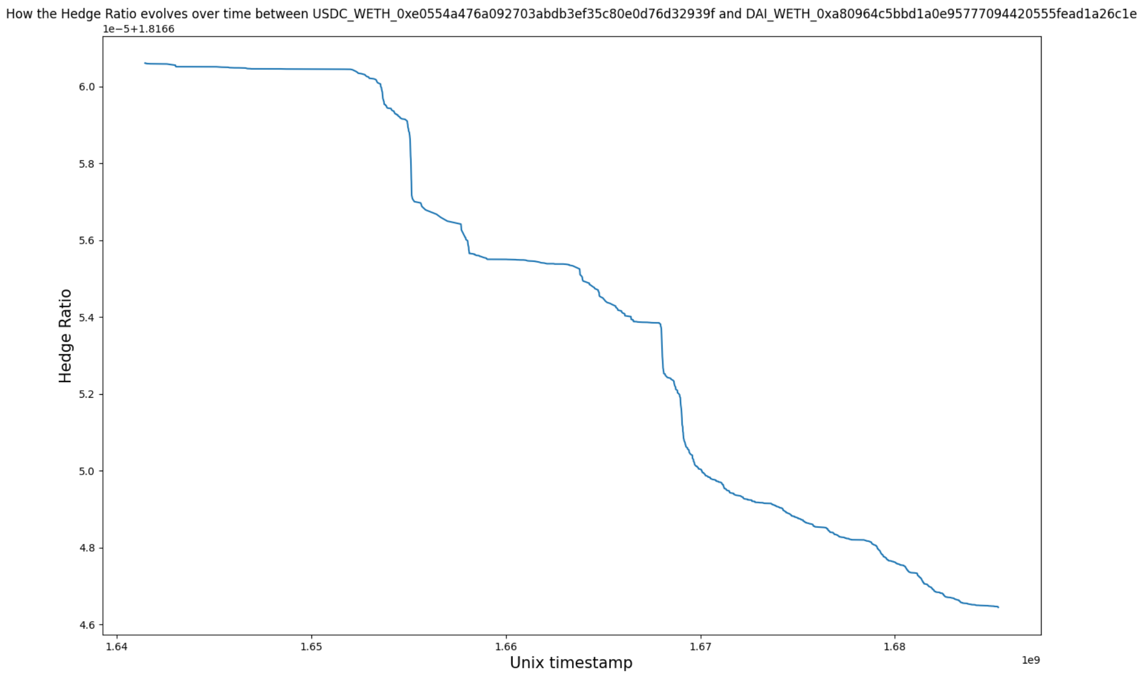

Figure 5.2: Plot of how the Hedge Ratio using the Kalman Filter

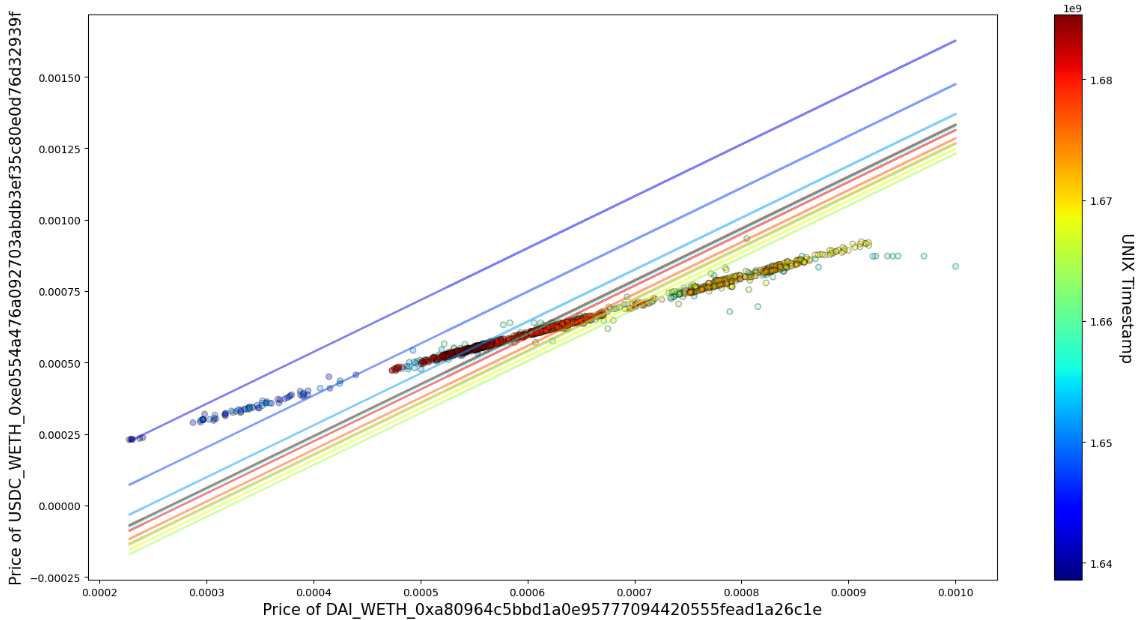

Figure 5.3: The plot shows the Kalman Filter estimating the regression line over time,

in this section, we will delve into the various costs associated with implementing and executing the strategy.

Gas Fees

Gas fees in Ethereum are a crucial component of the network's operation. They serve two main purposes: to prevent spam and denial-of-service attacks and to incentivize miners to include transactions in the blockchain. They are a measure of computational effort required to execute a transaction or perform a smart contract operation on the Ethereum network. Each operation, such as sending tokens, executing a smart contract function, or interacting with decentralized applications, consumes a certain amount of gas.

The fee is calculated by multiplying the gas price (expressed in Gwei, where 1 Gwei is equal to 0.000000001 ETH) by the gas used [17]. As mentioned in previous sections, when a transaction is initiated by a user, the user specifies the gas limit which defines the maximum amount of gas that a user is willing to pay for a transaction, if the gas limit is too low and the gas used is greater than its limit, the transaction will fail and be reverted. This means that the state changes made by the transaction will not be recorded on Ethereum, and the fees paid for that transaction will be lost.

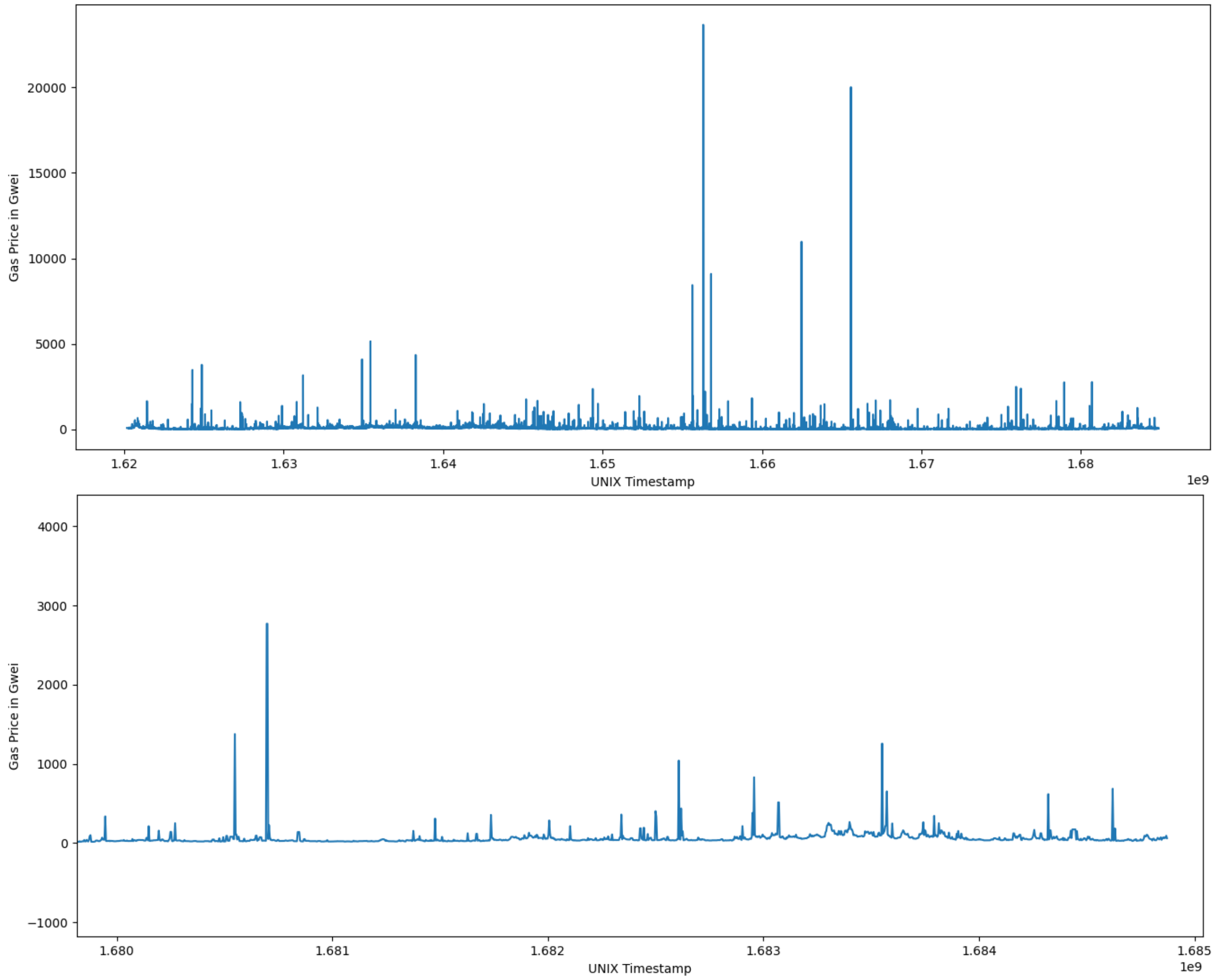

Figure 5.4: The top picture shows the gas price over time, while the bottom picture is a zoomed-in plot.

Figure 5.4 the gas price history over time, showing the price surges and trophs. The reason for these fluctuations that it is determined by the market forces of supply and demand. It is the amount of Ether (ETH) that a user is willing to pay for each unit of gas consumed in a transaction. Miners have the option to prioritize transactions with higher gas prices, and thus are incentivized to include transactions with higher gas prices because they receive the gas fees as a reward for their mining efforts. Therefore, setting a higher gas price increases the chances of a transaction being executed.

The second part of the gas fee is how much gas is used, it is calculated by multiplying the gas cost of each operation in the transaction by the corresponding gas price. Each operation has a predefined gas cost associated with it, which reflects the complexity and resource requirements of that operation. When a transaction or smart contract execution is initiated, the Ethereum Virtual Machine (EVM) processes each operation and deducts the corresponding gas cost from the gas limit. In addition to this, Ethereum also has a base fee with all of their transactions thus having multiple small transaction becomes more costly than using a large combined transaction [17].

Uniswap Fees

In addition to transaction fees, there is another fee mechanism in place within the system when swapping on Uniswap. This fee is designed to incentivize and reward liquidity providers on the platform. When a swap occurs, a fee is deducted from the amount of tokens expected to be returned to the user.

The fee is calculated as a percentage of the swap volume, meaning that the larger the swap, the higher the fee. For example, let's consider a scenario where the exchange price is 1 TKNA = 100 TKNB and the fee is set at 0.3%. If someone exchanges 1 TKNA for TKNB, instead of receiving the full 100 TKNB based on the exchange rate, they will receive $100 \times (1 - 0.3\%)$ TKNB = 99.7 TKNB. The fee reduces the total amount of tokens received in the swap.

It's important to note that the specific fee percentage may vary between different liquidity pools on Uniswap. Each liquidity pool can have its own fee tier, and the fees associated with each interested liquidity pools can be seen in the table below.

Pool Address	Token0	Token1	Fee Tier as a %
0xcbcdf9626bc03e24f779434178a73a0b4bad62ed	WBTC	WETH	0.30
0xc2e9f25be6257c210d7adf0d4cd6e3e881ba25f8	DAI	WETH	0.30
0x8ad599c3a0ff1de082011efddc58f1908eb6e6d8	USDC	WETH	0.30
0x4e68ccd3e89f51c3074ca5072bbac773960dfa36	WETH	USDT	0.30
0x60594a405d53811d3bc4766596efd80fd545a270	DAI	WETH	0.05
0x4585fe77225b41b697c938b018e2ac67ac5a20c0	WBTC	WETH	0.05
0x88e6a0c2ddd26feeb64f039a2c41296fcb3f5640	USDC	WETH	0.05
0x11b815efb8f581194ae79006d24e0d814b7697f6	WETH	USDT	0.05

Aave Fees & Collatoral

Aave facilitates lending and borrowing transactions among users, and as part of its operations and incentive mechanisms, it imposes certain fees. In the context of this trading strategy, the only fees applicable would be the interest rates on the loan. The interest rate is typically expressed as an Annual Percentage Yield (APY) and is accrued continuously. Currently, Aave supports variable interest rates, which fluctuate based on market conditions, the borrowed asset, and supply and demand dynamics within the Aave.

Another aspect of Aave for the trading strategy is collateralization and the avoidance of liquidation. Liquidation occurs when a borrower's position is forcibly closed, and their collateral assets are sold off in cases of default or insufficient collateral. When borrowing from Aave, borrowers are required to allocate a certain percentage of the loan value as collateral, known as the Loan-to-Value (LTV) ratio [64]. For instance, if a token has a 75% LTV, borrowers can borrow 0.75 ETH worth of the corresponding token for every 1 ETH worth of collateral provided. However, as token prices fluctuate, the ratio between the borrowed token value and the collateral value changes, posing risks for both lenders and Aave. To safeguard lenders and maintain protocol solvency, a liquidation threshold is established. If the collateral value falls below this threshold, the borrower's position becomes vulnerable to liquidation. In such cases, the collateral is auctioned off to repay the outstanding debt to the lenders. Typically, the liquidation threshold is set 5-10% higher than the asset's LTV. For WETH, the Loan-to-Value ratio is 82.5%, and the liquidation threshold is 86%.

5.2.3 Overall Strategy

The implementation of these strategies is intricate, as it is designed to be scalable and highly customizable with various parameters. Additionally, since the strategies being investigated are based on mean reversion, an abstract strategy is created to allow for quick exploration and research into different approaches and the impact of hedge ratio calculations on returns. This abstract strategy encompasses the core logic of generating trading signals, determining optimal trade timing and volumes, and transmitting them to the live or backtesting system. The strategy-specific classes contain operations that calculate the hedge ratio and establish the thresholds that trigger the generation of trading signals. Below shows the functions that the abstract strategy possesses.

```python
class Abstract_Strategy():
    def __init__(self, number_of_sds_from_mean, window_size_in_seconds,
   percent_to_invest, strategy_name, gas_price_threshold,
   rebalance_threshold_as_percent_of_initial_investment):
        ...

    def initialise_historical_data(self, history_p1, history_p2):
        ...

    def recalculate_thresholds(self, has_trade=False):
        raise NotImplementedError("recalculate_thresholds not implemented")

    def new_tick(self, price_of_pair1, price_of_pair2, has_trade):
        ...

    def generate_signal(self, ctx, prices):
        ...
```

The functions `__init__`, `initialise_historical_data`, `new_tick` and `generate_signal` are all inheritted by each strategy and `recalculate_thresholds` is implemented in each instance of the strategy. `__init__`, `initialise_historical_data` are self explanatory. `new_tick` is called in `generate_signal` to update the strategy's knowledge of historical prices, this function also triggers a call to `recalculate_thresholds` which re-calculates the hedge ratio and determines the thresholds at which an arbitrage opportunity becomes apparent. `generate_signal` is the function that is called by the trading system at each price update, it first invokes `new_tick` which in turn updates the thresholds, finally the updated hedge ratio and thresholds are used in the process of generating a signal. The steps of `generate_signal` are outlined below: TODO fishis after backtesting system

1. The first step is to check whether, a position is currently held, for this the open positions

Volume of Trades - TODO

Chapter 6

Evaluation - TODO

6.1 Number Arbitrage opportunities found

One of the ways I will be evaluating the strategies I implemented will be finding the number of opportunities that each strategy finds as well as highlighting opportunities that resulted in both profits and loss. Furthermore, I will show how the P&L is effected by transaction/gas fees by plotting both the P&L including and excluding any fees.

6.2 Return on theoretical trading

One major problem with this project is the timeline, from background research, I can see that most papers evaluate their strategies over at least a 6 month period. As this limits development time and evaluation time, the majority of the evaluation will be run by a back-testing system and evaluating each transaction cost at each time for any transaction that I theoretically make to simulate the trade environment. Using this trading environment, I will run each strategy and plot it's revenue and return.

6.3 Return on actual trading

In addition to the trading on the in-sample data points, I plan to trade and evaluate the performance and returns of each strategy in real time.

6.4 Sharpe Ratio

Another metric I will use is the Sharpe Ratio, the reason for this is that it is important to evaluate how risky the strategies are compared to the risk free rate. The metric is given by the formula below:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

Where R_p is the return of a portfolio, R_f is the risk-free rate and σ_p is the standard deviation of the portfolio. I will calculate the Sharpe Ratio for both in-sample and out-of-sample results.

6.5 Performance of Signal Generation

I also plan to evaluate how long it takes for each strategy to generate a signal when prices change. In analysing this metric, we may be able to find times where once the signal is created, the opportunity could be lost hence by looking at this metric, we may be able to find processes that may decrease the signal creation time.

Chapter 7

Ethical Issues

The ethics of cryptocurrencies are widely debated for reasons such as anonymity, leading it to be the choice of currency used by criminals and illegal institutions, volatility and lack of regulation. The high volatility makes cryptocurrencies and decentralised finance very risky for retail investors that don't have the technical or financial know-how making investing in cryptocurrencies.

Another aspect of cryptocurrencies that has raised ethical questions is the energy consumption and carbon dioxide emission from the mining of cryptocurrencies. Formal research about this has also been completed and found that 'approximately 69 million metric tons of CO2 (Carbon dioxide) emission as a result of bitcoin mining' [65]. Thus, this is an ethical concern that I have thought about when designing the strategies so that the number of transactions that don't result in a profit, i.e. do not add value to the project, is limited.

In addition to the concerns above, although this project aims to find riskless profits, *'free lunches'*, it is not, in any form, of financial advice, and those who use the research or software that used in the development and research process to attempt to get favourable results, are liable for the losses or gains.

Chapter 8

Conclusion - TODO

Bibliography

[1] "'I lost millions through cryptocurrency trading addiction'," *BBC News*, May 2021. [Online]. Available: https://www.bbc.com/news/uk-scotland-57268024

[2] C. Gondek, "What Big Companies Are Investing In Cryptocurrency?" [Online]. Available: https://originstamp.com/blog/what-big-companies-are-investing-in-cryptocurrency/

[3] "History of bitcoin exchanges and trading - Bit2Me Academy." [Online]. Available: https://academy.bit2me.com/en/history-exchanges-bitcoin-trading/

[4] "DEX to CEX Spot Trade Volume." [Online]. Available: https://www.theblock.co/data/decentralized-finance/dex-non-custodial/dex-to-cex-spot-trade-volume

[5] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Business & Information Systems Engineering*, vol. 59, no. 3, pp. 183–187, 2017.

[6] "Ethereum Whitepaper." [Online]. Available: https://ethereum.org

[7] Y.-C. Liang, *Blockchain for Dynamic Spectrum Management*, 01 2020, pp. 121–146.

[8] "Merkle Tree in Blockchain: What is it and How does it work | Simplilearn." [Online]. Available: https://www.simplilearn.com/tutorials/blockchain-tutorial/merkle-tree-in-blockchain

[9] "Consensus mechanisms." [Online]. Available: https://ethereum.org

[10] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[11] "What are smart contracts on blockchain? | IBM." [Online]. Available: https://www.ibm.com/topics/smart-contracts

[12] "Introduction to smart contracts." [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/

[13] "Smart Contracts - Industrial IoT Use Case Profile | IoT ONE Digital Transformation Advisors." [Online]. Available: https://www.iotone.com/usecase/smart-contracts/u88

[14] "Ethereum Virtual Machine (EVM)." [Online]. Available: https://ethereum.org/en/developers/docs/evm/

[15] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger." [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[16] "Components of Blockchain Network," Apr. 2021. [Online]. Available: https://www.geeksforgeeks.org/components-of-blockchain-network/

[17] "Gas and fees." [Online]. Available: https://ethereum.org/en/developers/docs/gas/

[18] "What Is a DEX (Decentralized Exchange)? | Chainlink." [Online]. Available: https://chain.link/education-hub/what-is-decentralized-exchange-dex

[19] "Decentralized Exchange Platforms in Crypto Trading." [Online]. Available: https://www.gemini.com/cryptopedia/decentralized-exchange-crypto-dex

[20] "What is arbitrage?" Jul 2021. [Online]. Available: https://online.hbs.edu/blog/post/what-is-arbitrage

[21] G. Poitras, "Origins of arbitrage," *Financial History Review*, vol. 28, no. 1, p. 96–123, 2021.

[22] "Law of One Price: Definition, Example, Assumptions." [Online]. Available: https://www.investopedia.com/terms/l/law-one-price.asp

[23] P. Isard, "How far can we push the "law of one price"?" *The American Economic Review*, vol. 67, no. 5, pp. 942–948, 1977. [Online]. Available: http://www.jstor.org/stable/1828075

[24] J. Richardson, "Some empirical evidence on commodity arbitrage and the law of one price," *Journal of International Economics*, vol. 8, no. 2, pp. 341–351, 1978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022199678900272

[25] I. Eyal, "The miner's dilemma," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 89–103.

[26] Z. Avarikioti, L. Heimbach, Y. Wang, and R. Wattenhofer, "Ride the lightning: The game theory of payment channels," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 264–283.

[27] G. Huberman, J. D. Leshno, and C. Moallemi, "Monopoly without a monopolist: An economic analysis of the bitcoin payment system," *The Review of Economic Studies*, vol. 88, no. 6, pp. 3011–3040, 2021.

[28] S. Athey, I. Parashkevov, V. Sarukkai, and J. Xia, "Bitcoin pricing, adoption, and usage: Theory and evidence," 2016.

[29] D. Easley, M. O'Hara, and S. Basu, "From mining to markets: The evolution of bitcoin transaction fees," *Journal of Financial Economics*, vol. 134, no. 1, pp. 91–109, 2019.

[30] C. R. Harvey, "Cryptofinance," *Available at SSRN 2438299*, 2016.

[31] E. Pagnotta and A. Buraschi, "An equilibrium valuation of bitcoin and decentralized network assets," *Available at SSRN 3142022*, 2018.

[32] J. Mo, "A Theoretical Model of Cross-market Arbitrage." [Online]. Available: https://w4.stern.nyu.edu/finance/docs/pdfs/PhD/mo-job-market.pdf

[33] N. Spiricheva, V. Baklykov, M. Tuhbadshin, and O. Mukhnovskiy, "Implementation of stock market arbitrage," in *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 2019, pp. 0055–0058.

[34] Q. F. Akram, D. Rime, and L. Sarno, "Arbitrage in the foreign exchange market: Turning on the microscope," *Journal of International Economics*, vol. 76, no. 2, pp. 237–253, 2008.

[35] Y. Aiba, N. Hatano, H. Takayasu, K. Marumo, and T. Shimizu, "Triangular arbitrage as an interaction among foreign exchange rates," *Physica A: Statistical Mechanics and its Applications*, vol. 310, no. 3-4, pp. 467–479, 2002.

[36] T. Ito, K. Yamada, M. Takayasu, and H. Takayasu, "Free lunch! arbitrage opportunities in the foreign exchange markets," National Bureau of Economic Research, Tech. Rep., 2012.

[37] I. Makarov and A. Schoar, "Trading and arbitrage in cryptocurrency markets," *Journal of financial economics*, vol. 135, no. 2, pp. 293–319, 2020.

[38] T. Crépellière, M. Pelster, and S. Zeisberger, "Arbitrage in the Market for Cryptocurrencies," Rochester, NY, Dec. 2022. [Online]. Available: https://papers.ssrn.com/abstract=3606053

[39] C. PAUNA, "Arbitrage trading systems for cryptocurrencies. design principles and server architecture," *Informatica economica*, vol. 22, no. 2/2018, pp. 35–42, 2018.

[40] N. Boonpeam, W. Werapun, and T. Karode, "The arbitrage system on decentralized exchanges," in *2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, 2021, pp. 768–771.

[41] Y. Wang, Y. Chen, H. Wu, L. Zhou, S. Deng, and R. Wattenhofer, "Cyclic Arbitrage in Decentralized Exchanges," Jan. 2022, arXiv:2105.02784 [cs, q-fin]. [Online]. Available: http://arxiv.org/abs/2105.02784

[42] L. Heimbach, Y. Wang, and R. Wattenhofer, "Behavior of liquidity providers in decentralized exchanges," *arXiv preprint arXiv:2105.13822*, 2021.

[43] C. Krauss, "Statistical arbitrage pairs trading strategies: Review and outlook," *Journal of Economic Surveys*, vol. 31, no. 2, pp. 513–545, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/joes.12153

[44] H. Rad, R. K. Y. Low, and R. Faff, "The profitability of pairs trading strategies: distance, cointegration and copula methods," *Quantitative Finance*, vol. 16, no. 10, pp. 1541–1558, 2016. [Online]. Available: https://doi.org/10.1080/14697688.2016.1164337

[45] Y.-x. Lin, M. Michael, and G. Chandra, "Loss protection in pairs trading through minimum profit bounds: A cointegration approach," *Journal of Applied Mathematics and Decision Sciences*, vol. 2006, 08 2006.

[46] Z. Zhao, R. Zhou, and D. P. Palomar, "Optimal mean-reverting portfolio with leverage constraint for statistical arbitrage in finance," *IEEE transactions on signal processing*, vol. 67, no. 7, pp. 1681–1695, 2019.

[47] Q. Tran-Dinh, W. Michiels, and M. Diehl, "An inner convex approximation algorithm for bmi optimization and applications in control," *Proceedings of the IEEE Conference on Decision and Control*, 02 2012.

[48] Z. Zhao, R. Zhou, Z. Wang, and D. P. Palomar, "Optimal portfolio design for statistical arbitrage in finance," in *2018 IEEE Statistical Signal Processing Workshop (SSP)*, 2018, pp. 801–805.

[49] B. Alsadik, "Chapter 10 - kalman filter," in *Adjustment Models in 3D Geomatics and Computational Geophysics*, ser. Computational Geophysics, B. Alsadik, Ed. Elsevier, 2019, vol. 4, pp. 299–326. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128175880000106

[50] G. Vidyamurthy, *Pairs Trading: quantitative methods and analysis.* John Wiley & Sons, 2004, vol. 217.

[51] "How a Kalman filter works, in pictures." [Online]. Available: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

[52] H. E. Dempsey, "Market Inefficiency: Pairs Trading with the Kalman Filter," Dec. 2017. [Online]. Available: http://www.hedempsey.com/papers/Pairs%20Trading%20with%20a%20Kalman%20Filter.pdf

[53] J. P. Nóbrega and A. L. I. Oliveira, "A combination forecasting model using machine learning and kalman filter for statistical arbitrage," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014, pp. 1294–1299.

[54] C. Krauss, X. A. Do, and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500," *European Journal of Operational Research*, vol. 259, no. 2, pp. 689–702, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221716308657

[55] S. Moraes Sarmento, *A Machine Learning based Pairs Trading Investment Strategy*, 1st ed., ser. SpringerBriefs in Computational Intelligence. Cham: Springer International Publishing, 2021.

[56] T. G. Fischer, C. Krauss, and A. Deinert, "Statistical arbitrage in cryptocurrency markets," *Journal of Risk and Financial Management*, vol. 12, no. 1, 2019. [Online]. Available: https://www.mdpi.com/1911-8074/12/1/31

[57] D. Ju-Long, "Control problems of grey systems," *Systems & Control Letters*, vol. 1, no. 5, pp. 288–294, 1982. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016769118280025X

[58] R. Engle, "Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models," *Journal of Business & Economic Statistics*, vol. 20, no. 3, pp. 339–350, 2002.

[59] G. Figá-Talamanca, S. Focardi, and M. Patacca, "Common dynamic factors for cryptocurrencies and multiple pair-trading statistical arbitrages," *Decisions in economics and finance*, vol. 44, no. 2, pp. 863–882, 2021.

[60] B. M. Blau, T. Griffith, and R. J. Whitby, "Comovement in the cryptocurrency market," *Economics Bulletin*, vol. 40, no. 1, p. 1, 2020.

[61] F. Schär, "Decentralized finance: On blockchain-and smart contract-based financial markets," *FRB of St. Louis Review*, 2021.

[62] "How Uniswap works | Uniswap." [Online]. Available: https://docs.uniswap.org//contracts/v2/concepts/protocol-overview/how-uniswap-works

[63] "The Graph." [Online]. Available: https://thegraph.com

[64] "Risk Parameters." [Online]. Available: https://docs.aave.com/risk/asset-risk/risk-parameters

[65] M. A. Egiyi and G. N. Ofoegbu, "Cryptocurrency and climate change: An overview," *International Journal of Mechanical Engineering and Technology (IJMET)*, vol. 11, no. 3, pp. 15–22, 2020.