commodities

MDPI

Article

Optimisation of Cryptocurrency Trading Using the Fractal Market Hypothesis with Symbolic Regression

Jonathan Blackledge [1,2,3,4,5,6,7,8,*] **and Anton Blackledge** [9,10]

[1] Science Foundation Ireland, Three Park Place, Hatch Street Upper, Saint Kevin's, D02 FX65 Dublin, Ireland
[2] Centre for Advanced Studies, Warsaw University of Technology, PI. Politechniki 1, 00-661 Warsaw, Poland
[3] Department of Computer Science, University of Western Cape, Robert Sobukwe Rd, Bellville, Cape Town 7535, South Africa
[4] School of Electrical and Electronic Engineering, Technological University Dublin, D07 EWV4 Dublin, Ireland
[5] School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, University Rd, Westville, Durban 3629, South Africa
[6] Chandigarh Engineering College, Chandigarh Group of Colleges, Kharar Banur Hwy, Sector 112, Sahibzada Ajit Singh Nagar 140307, India
[7] Office of the Vice Chancellor for Research & Innovation, INTI International University, Persiaran Perdana BBN Putra Nilai, Nilai 71800, Malaysia
[8] Barnsley College University Centre, Church Street, Barnsley, South Yorkshire S70 2YW, UK
[9] Department of Electronic and Electrical Engineering, Faculty of Engineering and Design, University of Bath, Claverton Down, Bath BA2 7AY, UK; buster9001@googlemail.com
[10] Department of Computing, Imperial College London, Exhibition Road, South Kensington, London SW7 2AZ, UK
* Correspondence: jonathan.blackledge@tudublin.ie; Tel.: +44-7584-676960

Abstract

Cryptocurrencies such as Bitcoin can be classified as commodities under the Commodity Exchange Act (CEA), giving the Commodity Futures Trading Commission (CFTC) jurisdiction over those cryptocurrencies deemed commodities, particularly in the context of futures trading. This paper presents a method for predicting both long- and short-term trends in selected cryptocurrencies based on the Fractal Market Hypothesis (FMH). The FMH applies the self-affine properties of fractal stochastic fields to model financial time series. After introducing the underlying theory and mathematical framework, a fundamental analysis of Bitcoin and Ethereum exchange rates against the U.S. dollar is conducted. The analysis focuses on changes in the polarity of the 'Beta-to-Volatility' and 'Lyapunov-to-Volatility' ratios as indicators of impending shifts in Bitcoin/Ethereum price trends. These signals are used to recommend long, short, or hold trading positions, with corresponding algorithms (implemented in Matlab R2023b) developed and back-tested. An optimisation of these algorithms identifies ideal parameter ranges that maximise both accuracy and profitability, thereby ensuring high confidence in the predictions. The resulting trading strategy provides actionable guidance for cryptocurrency investment and quantifies the likelihood of bull or bear market dominance. Under stable market conditions, machine learning (using the 'TuringBot' platform) is shown to produce reliable short-horizon estimates of future price movements and fluctuations. This reduces trading delays caused by data filtering and increases returns by identifying optimal positions within rapid 'micro-trends' that would otherwise remain undetected—yielding gains of up to approximately 10%. Empirical results confirm that Bitcoin and Ethereum exchanges behave as self-affine (fractal) stochastic fields with Lévy distributions, exhibiting a Hurst exponent of roughly 0.32, a fractal dimension of about 1.68, and a Lévy index near 1.22. These findings demonstrate that the Fractal Market Hypothesis and its associated indices provide a robust market model capable of generating investment returns that consistently outperform standard Buy-and-Hold strategies.

check for updates

Academic Editor: Julien Chevallier

Received: 19 July 2025
Revised: 20 September 2025
Accepted: 22 September 2025
Published: 3 October 2025

Citation: Blackledge, J.; Blackledge, A. Optimisation of Cryptocurrency Trading Using the Fractal Market Hypothesis with Symbolic Regression. *Commodities* **2025**, *4*, 22. https://doi.org/10.3390/commodities4040022

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Keywords: cryptocurrencies; fractal market hypothesis; symbolic regression; optimisation; return on investments

1. Introduction

Starting with the introduction of Bitcoin (BTC) in 2009 [1], digital currencies, known as cryptocurrencies, have dramatically increased in value and popularity. Founded on the idea of decentralisation and anonymity, they facilitate peer to peer transactions without the need for any central bank. Transactions are recorded anonymously on the Blockchain, a publicly accessible digital ledger. Verification of the transactions are completed by 'Miners', who solve complex mathematical problems requiring significant amounts of power before the record is added to the Blockchain. These 'miners' are compensated for their efforts with a 'Gas Fee' (unrelated to fuel), which is added to every crypto transaction. The price of 'Gas' can widely fluctuate as it depends on the current traffic on the Blockchain. As an example, gas fees at night can be a minute fraction of the cost during peak day times.

Starting with the introduction of Bitcoin (BTC) in 2009 [1], digital currencies—collectively known as cryptocurrencies—have risen dramatically in both value and popularity. Built on the principles of decentralisation and anonymity, these currencies enable peer-to-peer transactions without the involvement of a central bank. All transactions are recorded on the 'blockchain', a publicly accessible digital ledger that ensures transparency while preserving user anonymity.

Transaction verification is carried out by so-called 'miners', who solve complex mathematical problems that require significant computational power before a record can be added to the blockchain. In return for their efforts, miners receive a 'gas fee' (unrelated to fuel), which is attached to every cryptocurrency transaction. The price of 'gas' can fluctuate widely, as it depends on the level of activity on the blockchain. For example, gas fees late at night may be only a small fraction of those charged during peak daytime periods.

1.1. Context: Cryptocurrencies

Due to cryptocurrencies' early reputation as facilitators of black-market transactions and money laundering, their value has historically experienced extreme fluctuations and speculation. However, as the concept of a decentralised financial network has matured—and its applications have expanded to include Non-Fungible Tokens (NFTs), unique cryptographic tokens that exist on a blockchain and cannot be replicated, as well as computer games and the emerging Metaverse—Bitcoin's price has soared, reaching a current market capitalisation of around $750 billion [2].

Recently, Jurrien Trimmer, a director at Fidelity Investments, predicted that a single Bitcoin could reach a price of $100,000 by the end of 2022 [3]. This optimism has spurred developments such as the creation of the UK's first 'Decentralised Digital Economy Platform' [4] and El Salvador's adoption of cryptocurrency as legal tender to reduce dependence on the U.S. dollar [5]. More recently, the UK government announced its intention to support the growing cryptocurrency industry, recognising stable coins as a valid form of payment as part of a broader plan to position Britain as a global hub for crypto-asset technology [6].

Nevertheless, cryptocurrency valuations remain highly sensitive to public sentiment. For example, Bitcoin gained approximately 10% in just 24 h following a U.S. executive order encouraging 'responsible innovation' within the industry [7], only to drop about 8% shortly thereafter after Elon Musk posted negative comments on Twitter regarding China's crackdown on crypto services [8] (cryptocurrency trading is currently illegal in China).

Such rapid, sentiment-driven price swings make Bitcoin difficult to predict. As traditional financial institutions begin opening cryptocurrency divisions and offering crypto-based investment products, there is growing interest in developing trading algorithms tailored to crypto exchanges. These algorithms must account for the unique characteristics of cryptocurrency markets and provide robust trend analyses to guide investment decisions. Although many cryptocurrencies are now in circulation, Bitcoin and Ethereum (ETH) remain the most advanced and supply the most comprehensive historical datasets, making them ideal for back-testing any proposed trading hypotheses.

1.2. Market Models

For some time, a large body of research has shown that financial investment instruments do not exhibit Gaussian-distributed returns (i.e., the normal distribution of price changes) [9,10]. The assumption that asset returns follow a normal distribution stems from the foundations of the Efficient Market Hypothesis (EMH), a principal model and standard analytical tool in finance since the mid-1980s. The EMH is built upon the Random Walk Model (RWM), which itself originates from Louis Bachelier's 1900 study of French government bonds [11]. In his work, Bachelier departed from conventional 'fundamental' analysis—where predictions are based purely on research and information—and instead sought to determine the probability that a price would move up or down.

Drawing on the phenomenon of Brownian motion, first theorised by Albert Einstein in 1905 [12], Bachelier proposed that market changes were independent and identically distributed variables [11], enabling the use of the 'Gaussian' distribution originally derived by Carl Friedrich Gauss. Building on Bachelier's concepts, Eugene Fama developed the EMH, which posits that financial markets are 'informationally efficient,' meaning that prices fully reflect all available information [13]. Based on the RWM, the EMH assumes independently random events characterised by a Gaussian distribution: the more efficient the market, the more random the price changes. However, the assumption of complete and instantaneous information availability is neither realistic nor philosophically sound—particularly when information is intentionally withheld [14]. Critics argue that reliance on the EMH and its derivatives, such as the Black–Scholes model, contributed to both the 2001 'Dot-Com' crash and the global financial crisis of 2008 [15,16].

Traditional analytical methods for currencies and commodities include the RWM and Monte Carlo simulations [17]. These approaches, however, have notable limitations. For example, Monte Carlo analysis cannot yield accurate results without a well-defined underlying statistical distribution for the stochastic field. Attempts to define cryptocurrency return distributions have proven inaccurate [18], revealing the shortcomings of many current models and methodologies.

An evolution of 'technical' market theory has led to the adoption of the Fractal Market Hypothesis (FMH), which analyses financial markets using the principles of fractal geometry. Empirical evidence that financial stochastic fields deviate from Gaussian distributions—exhibiting skewed and heavy-tailed probability density functions (PDFs)—has profound implications for risk management. The FMH adopts a non-Gaussian perspective, allowing for time dependence and serial correlations within the time series. Under this model, one can develop market metrics that capture the instantaneous time evolution of a cryptocurrency's non-stationary behaviour and can thus be used to analyse market trends.

Much of this deeper understanding of financial time-series behaviour has emerged only in recent decades, supported by advances in statistical modelling, computational methods and Econophysics [19]. Contemporary research consistently demonstrates that the empirical distributions of asset returns across diverse markets are fat-tailed, implying a higher probability of extreme events than predicted by a normal distribution [20]. These

tails, however, are not necessarily heavy enough to satisfy the strict mathematical properties of Lévy-stable distributions, as was once proposed. Instead, return distributions tend to exhibit semi-heavy tails, necessitating more nuanced models that balance empirical realism with analytical tractability [21].

In this paper, the FMH provides the foundation for developing market metrics based on polarity changes in the so-called Beta-to-Volatility Ratio (BVR) and Lyapunov-to-Volatility Ratio (LVR). These metrics are used to signal trend reversals in the Bitcoin–U.S. Dollar and Ethereum–U.S. Dollar exchange rates and to recommend long or short trading positions. The approach is tested using a Matlab-based backtesting system with daily opening price data from 12 February 2016 to 12 February 2022 and hourly opening price data from 12 November 2021 to 12 February 2022. A broader analysis of these exchanges is conducted to determine whether they behave as fractal stochastic fields and are therefore consistent with the FMH. In addition, a machine-learning (ML) method based on symbolic regression (SR) is employed to complement short-term price predictions during periods of high trend stability. This approach aims to enhance profitability over given event horizons and to improve very short-term forecasts of actual price levels.

1.3. Structure of the Paper

This paper is structured as follows: Section 1 provides background on the world of cryptocurrencies and introduces standard market hypotheses in economics. Section 2 reviews the main cryptocurrency trading platforms, summarises previous research, and derives two trend-analysis metrics based on the Fractal Market Hypothesis (FMH). Section 3 analyses Ethereum and Bitcoin exchanges to determine their underlying statistical distributions before presenting the functions and methods used for individual financial time-series analysis. Section 4 explores the use of Symbolic Regression to generate short-term estimates of the actual price of a cryptocurrency time series, rather than merely predicting its trend. However, the stability of the trend is used as a key factor in assessing the potential accuracy of such predictions. Section 5 presents the analysis and results obtained, followed by a discussion of their implications in Section 6. Finally, Section 8 summarises the main findings and outlines possible directions for future research. An appendix provides all relevant Matlab functions developed during the project, enabling readers to reproduce the results and extend the work as desired.

1.4. Original Contributions

The principal original contributions of this paper are as follows:

(i) Evaluation of key FMH parameters for the analysis of cryptocurrency time series.
(ii) Application of the FMH to cryptocurrencies through the development of the Beta-to-Volatility Ratio (BVR) and Lyapunov-to-Volatility Ratio (LVR) metrics.
(iii) Development and optimisation of Matlab software to compute these metrics and perform backtesting.
(iv) Parameter optimisation for BVR and LVR calculations to maximise trading profitability.
(v) Application of Symbolic Regression for short-term cryptocurrency price prediction.
(vi) Optimisation of input data to improve the accuracy of Symbolic Regression-based price forecasts.

2. Background and Theory

This section reviews previous research on cryptocurrency markets and outlines the historical development of the Fractal Market Hypothesis (FMH). It also introduces key metrics used to characterise the underlying nature of the market under investigation.

Finally, methods for assessing the direction and stability of time-series trends are derived, based on the assumption of a self-affine field following a Lévy distribution.

2.1. Previous Research

There is now a substantial body of academic literature examining the price dynamics and return characteristics of cryptocurrencies. Much of this research has focused on determining whether high-market-cap cryptocurrencies, such as Bitcoin (BTC) and Ethereum (ETH), exhibit long-term dependence in their price signals or their derivatives. Most studies report evidence of cryptocurrency market inefficiency, suggesting the existence of profitable trading opportunities. However, a parallel line of research indicates that BTC is gradually becoming more efficient as the market evolves.

Urquhart (2016) analysed daily BTC price data from 2010 to 2016 to test for informational efficiency [22]. Using the Ljung–Box test, Runs test, Hurst exponent, and Bartels test, he assessed long-term memory and concluded that the market was inefficient, with the Hurst exponent showing strong anti-persistence. Nevertheless, by around 2013, BTC began to show signs of maturity.

Building on this work, Nadarajah and Chu (2017) [23] examined similar datasets from 2010 to 2016, split into two three-year samples, and applied additional methods such as the generalised spectral and Portmanteau tests. Their findings supported Urquhart's conclusion of return independence. Bariviera (2017) [24] studied long-term dependence in BTC returns and volatility using daily data from 2011 to 2017 and De-trended Fluctuation Analysis (DFA) with sliding windows to estimate the Hurst exponent. Results indicated that BTC returns showed persistence that declined toward efficiency after 2014, although volatility remained persistently long-range up to 2017. Further work by Bariviera et al. (2017) [25] demonstrated that alternative time scales did not affect long-term dependence. Lahmiri et al. (2018) [26] employed Fractionally Integrated GARCH models and rejected the Efficient Market Hypothesis (EMH), finding consistent evidence of market memory under various distributional assumptions.

Following the dramatic price surges of 2017, Al-Yahyaee et al. (2018) [27] used daily data from 2010 to 2017 to compare BTC efficiency with that of other asset classes, including foreign exchange markets. This was the first application of Multifractal Detrended Fluctuation Analysis (MF-DFA), developed by Kantelhardt in 2002, to the BTC market. Their results showed that BTC exhibited the strongest long-term persistence and the lowest efficiency among the markets examined. Alvarez-Ramirez et al. (2018) [28], also using DFA, reported alternating periods of efficiency and inefficiency, with the latter associated with anti-persistence. In the same year, Jiang et al. (2018) [29] tested for long-term dependence using the Hurst exponent on a 14-day rolling basis, along with the Ljung–Box and Automatic Variance Ratio (AVR) tests, again finding strong evidence of inefficiency and long memory. Zhang et al. (2018) [30] analysed daily data from 2013 to 2018 for a basket of cryptocurrencies using kurtosis, skewness, autocorrelation, and DFA, showing that BTC was gradually approaching market efficiency (Hurst exponent = 0.5), although volatility continued to display clear long-term dependence. Later in 2018, Caporale [31] applied Harold Hurst's Re-scaled Range (R/S) analysis to the largest cryptocurrencies by market capitalisation, finding that BTC's Hurst exponent was trending toward efficiency but remained unstable, implying that trend-following strategies could still generate excess profits.

Research in 2019 further strengthened these conclusions. Celeste et al. [32] used R/S analysis and continuous wavelet transforms to examine fractal dynamics in BTC and ETH prices, reporting strong market memory and increasing memory effects in ETH. Hu et al. (2019) [33] rejected the EMH using a panel framework that revealed cross-sectional

dependence among high-cap cryptocurrencies. Chu (2019) [34] tested the Adaptive Market Hypothesis for BTC and ETH with hourly data from July 2017 to September 2018, employing the Ljung–Box and Kolmogorov–Smirnov tests. His results showed time-varying efficiency in both BTC and ETH and suggested that market sentiment and news events were not necessarily decisive factors in determining efficiency.

More recent studies have confirmed and extended these findings. Al-Yahyaee et al. (2020) [35] analysed multifractal characteristics of several cryptocurrencies using a rolling MF-DFA and quantile regression approach. They found long-term memory effects and time-varying inefficiency, concluding that high liquidity and low volatility enhance efficiency and create arbitrage opportunities for active traders. During the COVID-19 pandemic, Kakinaka and Umeno (2021) [36] examined daily data from January 2019 to December 2020 using MF-DFA, observing an increase in short-term multifractality but a decline in long-term multifractal behaviour. David et al. (2021) [37] applied Auto-Regressive Integrated Moving Average (ARIMA), Auto-Regressive Fractionally Integrated Average models, and DFA to daily BTC data from July 2016 to March 2019. Their analysis of the Hurst exponent, fractal dimension, and Lyapunov exponents revealed persistent long-term dependence and chaotic dynamics.

Complementary work has advanced the use of high-frequency data. Drożdż et al. (2018) [38] applied advanced MF-DFA methods to capture subtle variations in multifractality that were not detectable in earlier studies. Watorek et al. (2021) [39] confirmed these findings using surrogate datasets to verify that observed multifractal structures were genuine and not artefacts of noise or methodology. This combination of high-frequency analysis and rigorous validation strengthens the evidence for complex scaling dynamics in cryptocurrency markets.

Taken together, this extensive body of research strongly challenges the EMH and supports the view that BTC, while becoming more efficient over time, continues to exhibit long-term memory effects. Such memory implies that future prices depend partly on past values, allowing investors to predict returns and reduce market risk. Because the Fractal Market Hypothesis (FMH) explicitly incorporates memory within a stochastic price field, it provides a natural and powerful framework for modelling cryptocurrency markets in future research.

2.2. Trading with Cryptocurrencies

Cryptocurrencies can be traded on a variety of platforms, each offering distinct advantages. Examples such as Coinbase and Binance are dedicated exclusively to cryptocurrency markets, whereas platforms like eToro also facilitate trading in traditional assets such as stocks and commodities. Each platform applies its own fee structure for executing trades. For instance, Coinbase charges a transaction fee of approximately 0.5% [40], while eToro charges around 1% [41]. However, these platform fees are not the only costs associated with cryptocurrency trading. An intrinsic cost, known as a 'Gas Fee', is required to add a transaction to the blockchain, as discussed in the introduction. Gas fees depend heavily on current blockchain traffic and the complexity of the transaction, and they can fluctuate widely, making trades significantly less cost-effective and potentially disrupting a trading strategy.

For example, while eToro charges a higher base transaction fee than Coinbase, it does not require payment of a gas fee. Given that the average gas fee is approximately $20, with peaks that can reach several thousand dollars during periods of network congestion, eToro can offer a more stable and predictable cost structure for traders.

At present (i.e. at the time of undertaken the research discussed in this article) the Ethereum blockchain is transitioning to *Ethereum 2.0*, an upgrade designed to introduce a more energy-efficient mechanism for validating transactions, thereby reducing gas fees. If

this transition is completed as projected by the end of 2022, it could significantly influence platform preferences among traders. Currently, eToro remains the only widely used and internationally trusted platform that supports crypto-based options and derivatives.

For the purposes of this study, the eToro model is adopted, under which no gas fee is applied. Accordingly, all Return on Investment (ROI) calculations are presented on a pre-fee basis and therefore represent gross profit.

2.3. The Fractal Market Hypothesis

Research dating back to the 1960s suggested that commodity price-change distributions were too sharply peaked to be Gaussian, a phenomenon later termed leptokurtosis. This characteristic was also observed by Mandelbrot [42,43]. These and other studies indicate that the assumption of normally distributed price changes is inadequate for capturing the fundamental statistics of Financial Time Series (FTS), motivating the development of the Fractal Market Hypothesis (FMH).

The FMH was formally introduced by Edgar Peters in his 1994 book *Fractal Market Analysis: Applying Chaos Theory to Investment and Economics* [44], building on earlier work by Mandelbrot and Ralph Elliott. It draws heavily on fractal geometry, a field first proposed by Felix Hausdorff in 1918 [45] and popularised by Mandelbrot in his 1982 book *The Fractal Geometry of Nature* [46]. Fractal geometry studies how fractured objects exhibit self-similarity, a property in which geometric features are preserved at all scales [47]. This property is closely related to chaotic signals, which can be generated through the iteration of strictly nonlinear functions. Modelling FTS as chaotic signals has found practical applications in various trading scenarios [48].

FTS are now recognised as examples of stochastic self-affine fields. The self-affine properties of financial series were first recorded by Ralph Elliott in 1938 [49]. Unlike conventional calculus, fractional calculus introduces the concept of memory into derivatives. Within this framework (i.e., modelling FMH signals as fractional differential equations), the FMH provides a model for financial time series that incorporates memory, meaning that the price of a given currency or commodity depends on past values. This represents a major advancement over the EMH, which assumes that markets are efficient and therefore unpredictable. Moreover, because FTS appear statistically similar at different scales, one can assume that the probability density functions (PDFs) of price values are approximately scale-invariant.

Many self-affine functions exist. Early examples include the 'Lévy Curve', introduced by Paul Lévy in his 1938 paper *Plane or Space Curves and Surfaces Consisting of Parts Similar to the Whole* [50]. Early FMH developments also relied on the Lévy distribution. Mandelbrot observed that cotton price changes followed a Lévy distribution [51] with a Lévy index (γ) of 1.7 [43]. The Lévy index quantifies the deviation of a distribution from Gaussian behaviour. Within the FMH, $\gamma \in [1, 2]$, where lower values indicate heavier tails and a higher peak, higher values indicate the opposite, and $\gamma = 2$ corresponds to a Gaussian distribution. As γ decreases further from 2, the probability of rare but extreme events—so-called 'Lévy flights'—increases.

Other metrics capture a signal's fractal characteristics. Mandelbrot, applying principles of Brownian motion to asset prices, proposed that markets exhibiting long-term dependence tend to maintain trends. He also developed a metric to quantify a field's persistence (trend continuation) or anti-persistence, known as the Hurst exponent (H). This metric is named after Harold Edwin Hurst, who, in 1906, discovered that the range of water levels in the Nile River scaled according to a fractional power law [52]. The Hurst exponent measures long-term memory: $0 < H < 0.5$ indicates anti-persistent behaviour (negative correlation), whereas $0.5 < H < 1$ indicates persistent behaviour (positive correlation). In the context

of FTS, a high H implies that prices are biased in a particular trend direction over time, meaning that in a Random Walk Model (RWM), price deviations from the origin are larger.

The parameters γ and H are related to the fractal dimension (D_F), which quantifies a field's complexity and self-affinity [53]. These can also be estimated using the Spectral Decay Coefficient (α), as discussed later in this report.

Because fractals are inherently iterative and related to chaos theory, there is a natural connection between chaos and the FMH. Another important parameter is the Lyapunov exponent (λ), which characterises the level of chaotic behaviour in a system, i.e., the transition from regular to chaotic dynamics. Within this framework, λ can be used to analyse the evolution of a financial time series and support trend prediction.

Taken together, these metrics—H, γ, D_F, and λ—provide a robust framework for analysing irregular financial time series within the context of the FMH.

2.4. The FMH vs. EMH

The Fractal Market Hypothesis and the Efficient Market Hypothesis offer contrasting perspectives on how financial markets function. The EMH posits that asset prices fully reflect all available information, rendering it impossible for investors to consistently achieve abnormal returns through prediction or arbitrage. It assumes rational agents, normally distributed returns, and market equilibrium driven by information efficiency. In this framework, price dynamics are largely random and memoryless, conforming to a Gaussian model.

By contrast, the FMH, draws on fractal geometry and chaos theory to argue that financial markets exhibit long-range dependence, heterogeneity of investor horizons, and self-similar structures across time scales. Rather than assuming equilibrium, FMH emphasises persistence and clustering in volatility, as well as the influence of structural changes and collective behaviour. It suggests that markets remain stable when investors with diverse time horizons interact, but instability arises when this diversity diminishes, leading to crises or bubbles.

In the context of cryptocurrency markets, FMH offers a more realistic framework than EMH. Cryptocurrencies are highly volatile, exhibit frequent periods of speculative bubbles, and are shaped by heterogeneous participants ranging from algorithmic traders to retail investors with divergent horizons. Moreover, empirical studies demonstrate heavy tails, autocorrelation in volatility, and fractal-like scaling in crypto price series, all of which contradict Gaussian assumptions underpinning EMH. FMH, by accommodating these nonlinear and long-memory features, provides a stronger theoretical basis for analysing cryptocurrency markets and assessing risks in environments characterised by structural instability and speculative dynamics.

2.4.1. Characteristic Indices

The observation that price changes in an FTS are inefficient [9] allows the introduction of a number of further indices to analyse the variability in stochastic field distributions. These act as a metric of the data's deviation from the normal distribution and characterise the fields susceptibility to undergo Lévy flights and exhibit long-term market memory. Moreover, these metrics can all be defined through linear relationships of the spectral decay coefficient, α, for a self-affine stochastic fields' power spectrum. Therefore, definitions of these indices can be reduced to the calculation of the α.

The power spectrum $P(\omega)$ of a self-affine signal decays according to a power law [54].

$$P(\omega) = \frac{c}{|\omega|^{2\alpha}}, \; |\omega| > 0$$

where ω is the spectral frequency (specifically the temporal angular frequency) and c is a real constant. Using logarithms, the above equation can be re-written as

$$\ln P(\omega) = \ln c - 2\alpha \ln |\omega|$$

from which the value of α can be obtained by determining the gradient of a linear regression applied to the power spectral log-log plot. In computational terms, application of the least squares regression formula provides a value of α when applied to the entire field. The Matlab code for calculating α is shown in Appendix A.8. Using this result, the following standard algebraic formulae for the following measures can be obtained:

$$H = \frac{2\alpha - 1}{2} \tag{1}$$

where H is the Hurst Exponent [55].

$$D_F = \frac{5 - 2\alpha}{2} \tag{2}$$

where D_F is the Fractal Dimension of the signal [46].

$$\gamma = \frac{1}{\alpha} \tag{3}$$

where γ is the Levy Index [56]. These measures and their meanings with respect to stochastic field distributions and the relevant cryptocurrencies are discussed in Section 2.3.

An FTS commonly consists of a set of discrete values, representing the price of a currency or commodity. There is value in developing deterministic models to analyse economic systems, providing such a quantitative analysis can aid the management of risk. However, large markets are usually functions of random variables characterised by external influences which are complex and therefore difficult to define. These influences on the system are compounded due to their non-linear nature caused by feedback, a market reacting to itself, or its sensitivity to shocks under 'market memory' conditions.

For these reasons, the models are often relatively complicated. By dispensing with the efficient market view of economic systems in favour of a fractal base, an index to predict the future price trend behaviour can be developed as outlined in the following discussion. The goal is to develop a single measure capable of predicting the start of an upward (bull) or downward (bear) trend in a cryptocurrency price.

A standard model for a signal $s(t)$, in this case a continuous, time-dependent currency price, is embodied in the equation

$$s(t) = h(t) \otimes f(t) + r(t) \tag{4}$$

where $\otimes$ denotes the convolution integral,

$$h(t) \otimes f(t) \equiv \int h(t - \tau) f(\tau) d\tau$$

In Equation (4), $h(t)$ is the Impulse Response Function, $f(t)$ is the Source Information Function and $r(t)$ is the stochastic function of time—residual and additive noise [57]. An inherent problem of signal processing is the extraction of the relevant information from the 'noisy' signal when only an estimate of $f(t)$ is possible. This remains the case with financial signal processing. However, in this case, $f(t)$ consists of a wide density of global transactions that occur throughout time. The wide and effectively random nature of these transactions produce such a broad spectrum that they can be interpreted as 'White Noise',

even if this 'Noise' represents real transactions. Under this assumption, Equation (4) can be modified to

$$s(t) = h(t) \otimes n(t) \tag{5}$$

where $n(t)$ now represents the white noise of the 'system'.

Consider a financial price signal where the system noise is in fact a valid price. Then the Signal-to-Noise Ratio can be considered to be high enough that residual noise can be ignored. Due to the human element of the transaction system, the convolution is where the feedback, or transaction history, is introduced into the model. This is a linear stationary model. However, financial signals are inherently non-stationary. Thus, a metric relating to a stationary model must be applied on a moving window basis to create an 'index signal' which represents the time evolution of the metric. This signal can then be used to indicate the instantaneous nature of the signal but not used to 'predict' the future, which is a common misconception in financial signal processing.

Under the assumptions of the fractal market displaying a Lévy distribution, the standard Fourier space signal signature, $S(\omega)$, can be defined as (using convolution theorem)

$$S(\omega) = H(\omega)N(\omega) \tag{6}$$

where $N(\omega)$ is the noise present in the system and $H(\omega)$ is the transfer function of the signal given by

$$H(\omega) = \frac{1}{(i\omega)^{\alpha}}$$

The temporal form of the transfer function is obtained by taking the Inverse Fourier Transform (IFT) giving [58]

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{e^{i\omega t}}{(i\omega)^{\alpha}} d\omega = \frac{\theta(t)}{\Gamma(\alpha)}\frac{1}{t^{1-\alpha}}$$

where Γ is the Gamma function and $\alpha \in (0,1)$—a metric that quantifies the self-affine characteristics of the signal—and $\theta(t)$ is the Heaviside step function given by

$$\theta(t) = \begin{cases} 1, & t > 0; \\ 0, & t < 0. \end{cases}$$

The signal is thus given by (using the convolution theorem)

$$s(t) = \frac{1}{\Gamma(\alpha)}\frac{1}{t^{1-\alpha}} \otimes n(t) = \frac{1}{\Gamma(\alpha)}\int_{-\infty}^{\infty} \frac{\theta(\tau)}{(t-\tau)^{1-\alpha}} n(\tau) d\tau \tag{7}$$

The metric α is related to the Fractal Dimension (D_F), Hurst Exponent (H) and Levy Index (γ), and hence, by association, measures the signals persistence or anti-persistence, i.e., whether the signal is convergent (bear market trend) or divergent (bull market trend).

The self-affine characteristics of the signal $s(t)$ for an arbitrary scaling factor, λ, can be analysed as follows. Suppose we let $\xi = \lambda\tau$ and compute $s_{\lambda}(t)$ for the same convolutional transform but for function $n(\lambda\tau)$ instead of $n(t)$. Given Equation (7), this yields

$$s_{\lambda}(t) = \frac{1}{\Gamma(\alpha)}\int \frac{1}{(t-\xi/\lambda)^{1-\alpha}} n(\xi)\frac{d\xi}{\lambda} = \frac{1}{\Gamma(\alpha)}\frac{1}{\lambda^{\alpha}}\int \frac{1}{(\lambda t-\xi)^{1-\alpha}} n(\xi) d\xi$$

and thus, we can write,

$$s_\lambda(t) = \frac{1}{\lambda^\alpha} s(\lambda t) \tag{8}$$

However, because $n(t)$ is being taken to be a stochastic function, there will be differences between the time signatures observed over different time scales but, on the basis of Equation (8), the distribution of amplitudes will remain the same over all scales. In other words, we can consider $s(t)$ to be statistically self-similar, a property that is compounded in the equation

$$\lambda^\alpha \Pr[s(t)] = \Pr[s(\lambda t)]$$

This is the defining characteristic of a self-affine stochastic function and therefore shows that Equation (7) is a valid representation of a fractal signal—a self-affine stochastic field.

This equation provides a method of calculating the metric α (by applying regression method to the power spectrum for $s(t)$ as discussed earlier) over an entire data set. However, this is not useful for estimating the instantaneous self-affine characteristic of the signal over time.

Suppose we consider a small 'look-back' window of data, where $\tau << t$, is used to compute the index on a rolling basis. In this case, the resulting time varying metric will provide an indicator of the base field's persistence such as a currency exchange rate.

Applying this approach to Equation (7), for the case when $\tau << t$, the convolution integral can be approximated as

$$s(t) \sim \frac{c}{t^{1-\alpha}}, \; t > 0 \tag{9}$$

where

$$c = \frac{1}{\Gamma(\alpha)} \int_0^T n(\tau) d\tau, \; T << 1$$

In practice, given Equation (9), T is the windowed data length, which is a fraction of the full data complement. Defining $\beta = 1 - \alpha$, creates a single calculable measure when $\beta > 0$ and the signal $s(t)$ is convergent, indicating an impending decay in value of the underlying currency. When $\beta < 0$ there is an indication of a rise in price.

Applying logarithms to Equation (9), we obtained the linear equation

$$\ln[s(t)] = \ln[c] - \beta \ln[t]$$

and thus, once again, using the least squares regression algorithm, the log-log line gradient is given by the value of β. Implemented in code, this is a faster computational procedure than performing rolling convolutions, especially when the FTS consists of many data points. The Matlab function for computing β is provided in Appendix A.3.

2.4.2. The Lyapunov Exponent

To develop another metric capable of indicating the time evolution of a stochastic field, other properties of an FTS can be utilised. For example, suppose we model an FTS as

$$u_{n+1} = u_n + \varepsilon_n$$

where u_n is the amplitude of the signal (the price value) for the n^{th} iteration and ε_n is a small n^{th}-order error. This is the basis for the RWM whether it be consistent for classical or fractional Brownian diffusion. However, self-affine stochastic fields do not exhibit Gaussian properties, being more susceptible to rare and extreme events and displaying longer distribution tails, which is indicative of a Lévy distribution. Under this model,

fractal fields exhibit fractional diffusion where the error in each time step can have wide and fluctuating values.

Chaos theory is a field of applied mathematics where seemingly insignificant differences in initial conditions produce significant system outputs changes. This makes long-term deterministic models and predictions impossible [59]. Fractional diffusion is in itself a chaotic iterative process where the next step is the result of some form of non-linear function.

A basic form for a chaotic iterative signal is

$$u_{n+1} = f(u_n) \tag{10}$$

where $f(u_n)$ is some non-linear function that depends critically on the value of a parameter or parameter set. In coupled systems, a single non-linear element will render the entire system as chaotic. These systems are Iterated Function Systems (IFS). Introduced in 1981 by John E. Hutchinson, they provide methods of constituting self-affine fractal geometries and signals [60].

In regard to Equation (10), the stability of an iterative process must be quantified. This can be done by considering the following iteration:

$$u_{n+1} = f(u_n) = u_n + \varepsilon_n$$

In this case, the error ε_n at any iteration n is a measure of the rate of separation of values between two iterations. The Lyapunov exponent (λ) is based on modelling this error as

$$\varepsilon_{n+1} = \varepsilon_n e^{\lambda} \tag{11}$$

Formally, this exponent is the measure of the rate of separation of infinitesimally close trajectories [61]. In this context, it can be used to examine how sensitive a signal is to an initial condition, i.e., how chaotic (unstable) the signal is. This is because, if $\lambda >> 0$, the error at each iteration will increase rapidly. Similarly, if $\lambda < 0$, the error will undergo exponential decay as the iteration progresses. From Equation (11), it is clear (summing over ε_n) that we can write the Lyapunov exponent in the form

$$\lambda = \frac{1}{N} \sum_{n=1}^{N} \ln\left(\frac{\varepsilon_{n+1}}{\varepsilon_n}\right) \tag{12}$$

In the case of an FTS, ε_n is taken to the price value u_n.

For a cryptocurrency time series, λ is calculated computationally as the sum of the log changes in price. From Equation (12), it is clear that if $\lambda < 0$, then the iterative process is stable; i.e., it converges. Moreover, if $\lambda > 0$, the process will diverge. Based on this observation, a positive value of λ denotes a signal with a chaotic characteristic and a negative value signifies a convergent, stable system. As values of λ increase, so does the rate at which the signal will converge or diverge [62].

Using this simple formula applied to an FTS assumed to be of the form of Equation (5), the Lyapunov exponent can be calculated for a small 'look-back' window of size N to once again create a metric signal that will represent a time evolution of the nature of the time series. As $1/N$ is merely a normalisation factor, if $\varepsilon_{n+1} < \varepsilon_n$, then $\lambda < 0$ and visa versa. Therefore, a change in polarity of the metric is a sign of the gradient of the time series, and thereby an indicator of the growth or decay of the currency price. The Matlab code used to implement this calculation for an input time series is presented in Appendix A.1.

2.4.3. The Lyapunov-to-Volatility Index

The accuracy of an FMH trend indicator, be it the Lyapunov or Beta indexes, relies on the size of the moving window used during the calculation. This moving window introduces a trading delay into the system directly proportional to its size. In the case of λ and β, transitioning from positive to negative, or visa-versa, is an indication of a transition in price trend. This transition manifests as the metric's graphical zero-crossings. However, this in itself, is not a measure of the stability of this trend.

The volatility, σ, is a measure of FTS stability. A trend indicator can be scaled using the volatility to produce not only a predictor of future trends, but also as a measure of the stability of that trend. In the case of the Lyapunov exponent, the resulting index is the Lyapunov-to-Volatility Ratio;

$$\lambda_\sigma = \frac{\lambda}{\sigma} \tag{13}$$

where the volatility is defined as

$$\sigma = \left[\sum_{n=1}^{N}\left|\ln\left(\frac{u_{n+1}}{u_n}\right)\right|^2\right]^{\frac{1}{2}}$$

where u_n is the price value. The Matlab function for computing the volatility is presented in Appendix A.2.

The same scaling can be applied to β, i.e., $\beta_\sigma = \beta/\sigma$, which now provides a different method, from a different theoretical base, to indicate changing price trends and their stability. These can be compared during system testing to observe any changes in accuracy or returns.

Although the LVR can now double as a stability indicator, the position of the zero-crossings, which recommend long or short positions, will depend on the accuracy of λ_σ. This itself depends on the inherent 'noise' present within the FTS. An FMH model assumes stationarity, which is not the case in Lévy distributed stochastic fields. Their tendency to exhibit Lévy flights create 'micro-trends' that can have a dramatic effect on the system. This can yield errors in the exact position of the zero-crossings (buy/sell positions), especially with respect to small term trend deviations.

To mitigate this effect, a moving average filter can be applied to the FTS, u_n, reducing the effect of the noise. This can be represented in the following continuous form:

$$s(t) = w(t) \otimes u(t)$$

where $w(t)$ calculates the mean of the windowed data of length W. This method is also applied to the resulting LVR signal so that

$$\lambda_\sigma(t) = w(t) \otimes \lambda_\sigma(t)$$

In this case, however, the windowed data is of length T, the financial index calculation period. Using the LVR, the zero-crossings are simply defined by the following Kronecker delta functions:

$$z_c(t) = \begin{cases} +1, & \lambda_\sigma(t) < 0 \;\;\&\;\; \lambda_\sigma(t+\tau) \geq 0 \\ -1, & \lambda_\sigma(t) > 0 \;\;\&\;\; \lambda_\sigma(t+\tau) \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where t is the current position of the time series and τ is a small time step. Essentially, $z_c = 1$ represents the end of a bear trend and the beginning of a bull trend and the opposite transition is represented by $z_c = -1$. Combined with the previous Matlab functions, this provides a method of trend prediction that can be used to analyse BTC-USD/ETH-USD

FTS and recommend long or short trades. Exploration into the optimisation of the W and T parameters for a specific FTS is conducted in Section 5.1.

3. Methodology

To perform trend prediction on an FTS, a suitable model for the series must be chosen. This determines the nature of the functions used to calculate any market metrics and conduct historic backtesting. The following section presents an overall analysis of the BTC-USD and ETH-USD markets to determine a 'distribution of best fit' before outlining the code implementing the equations derived in Section 2.3, the back-tesing strategy, and ML short-term prediction.

3.1. Cryptocurrency—An Efficient or Fractal Market

Before any trend prediction approach can be adopted, there must be a determination of whether cryptocurrencies display efficient or fractal behaviour. To achieve this, 7 years of opening prices (00:00:00 GMT) on both the BTC-USD and ETH-USD markets were collected. For the purpose of this report, only BTC-USD will be described in detail, with the accompanying results for ETH-USD presented to readers.

Figure 1 shows the price of Bitcoin from 12 February 2016 to 12 February 2022. The data was obtained from https://www.CryptoDataDownload.com (accessed on 17 March 2022) using the Gemini database for Coin-base hourly prices. The extraction of this data was achieved using a Matlab function which will be introduced in Section 3.3.2. All work prepared for this paper was conducted in Matlab and Excel.

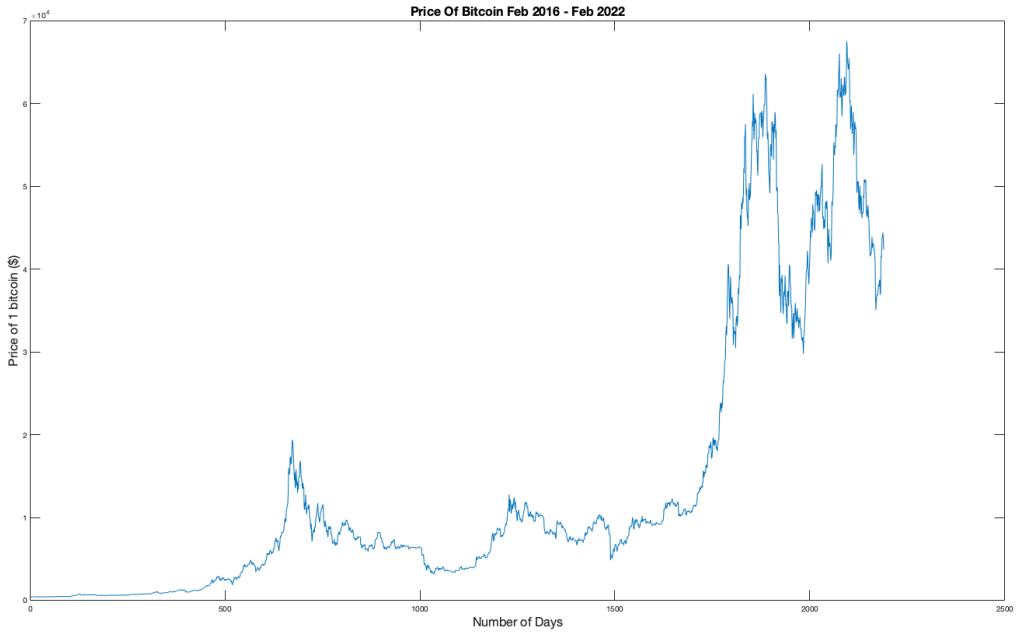


Figure 1. Daily opening prices for BTC-USD from 12 February 2016 to 12 February 2022.

Figure 2 displays the Probability Density Function (PDF) of BTC-USD prices. The distribution clearly does not conform to the traditional 'Bell Curve' synonymous with a Normal distribution. The standard deviation was calculated to be 1.68×10^4 with a Kurtosis of 4.28, indicating a Leptokurtic curve with a narrow peak. The Skew is 1.61 indicating a longer tail to the right.

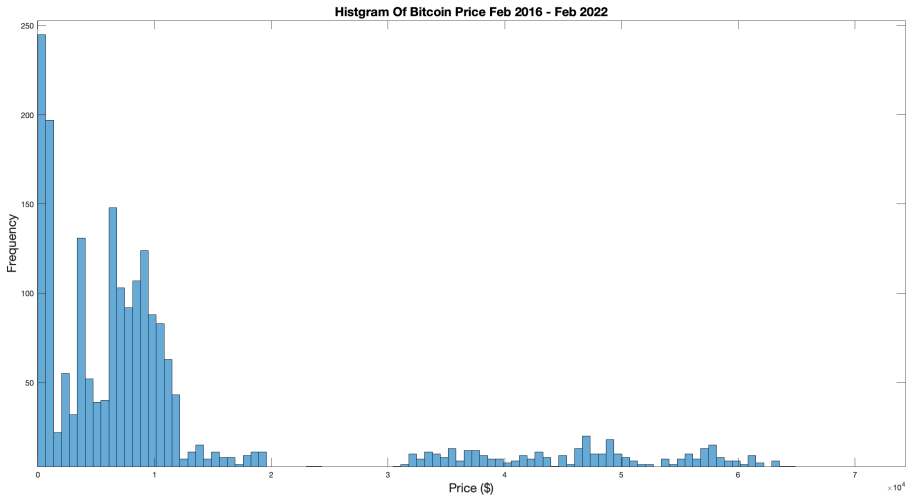


Figure 2. Frequency histogram for BTC-USD opening prices (2016–2022).

However, this work is concerned with the distribution of price changes between daily opening prices. In this respect, Figure 3 shows the PDF of the absolute log of BTC-USD price changes. Figure 4 displays the actual price changes. The forward difference is used to purely display price changes and logs taken to remove exponential elements within the signal.

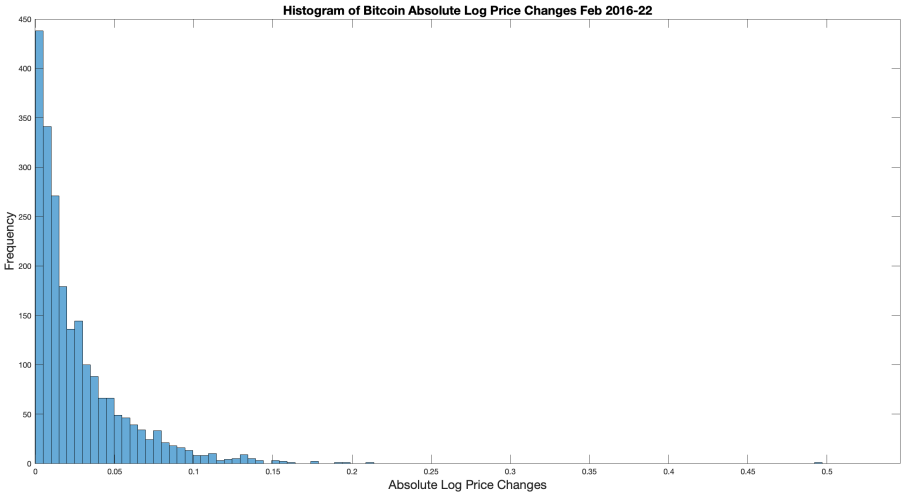


Figure 3. Frequency histogram of log price changes for BTC-USD (2016–2022).

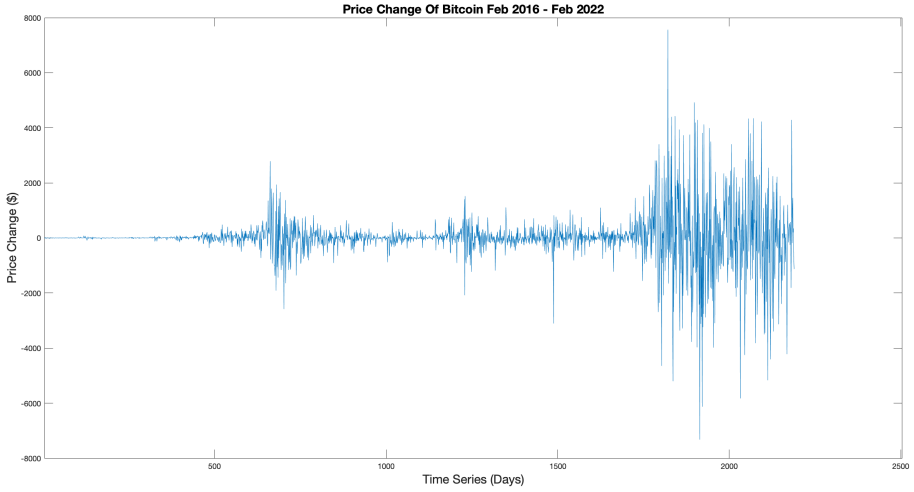


Figure 4. Actual price changes for opening daily BTC-USD prices (2016–2022).

An efficient market is considered to follow an RWM where each event is random and independent. Under this consideration, price changes should follow a Gaussian distribution. However, it is clear from Figure 3 that the peaked nature of the PDF is far more indicative of a Lévy distribution. This provides some initial evidence that BTC-USD is not an efficient market. Figure 5 depicts the log price PDF with a standard Gaussian bell curve overlaid to enforce the difference.

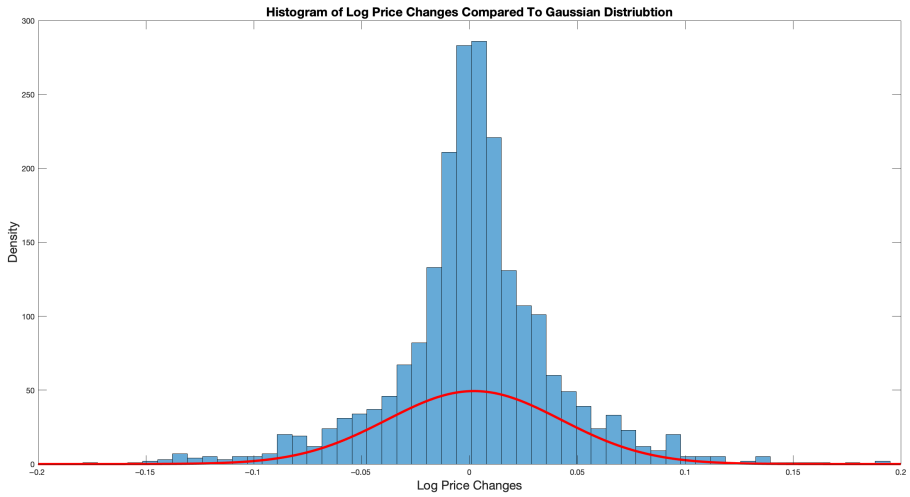


Figure 5. Log price change probability density function for opening daily BTC-USD prices (2016–2022) vs. standard Gaussian distribution.

As outlined in Section 2, the limitations of the EMH has led to the introduction of new analytic methods. The Hurst exponent (H), resulting from Re-Scaled Range Analysis (R/S), is a measure of a signal's persistence or anti-persistence, i.e., the likelihood that the next change will be in line or opposed to the current trend direction. $H = 0.5$ would signify an independently distributed signal, indicating a classical Brownian diffusion model, whilst $1 > H > 0.5$ indicates a persistent system and $0 < H < 0.5$ indicates an anti-persistent system. Using Equation (1), the Hurst Exponent was calculated, yielding a value of $H = 0.3185$, confirming that the BTC-USD exchange displays anti-persistence characteristics and does not conform to conventional Brownian motion, i.e., classical diffusion.

Another measure relevant to the BTC-USD time series is the Fractal Dimension (D_F). A measure of the self-similarity of a stochastic self-affine field, it can provide insight into the power spectrum range. With $D_F = 2$ being the maximum value, any value below this ($1 < D_F < 2$) will provide a quantitative representation of the fields deviation from normality. Using Equation (3) for the case of BTC-USD, $D_F = 1.682$. This is further evidence that this exchange is not an efficient market.

Finally the Lévy index gives a representation of the signals variability, with $\gamma = 2$ recovering a Gaussian distribution of price changes and $\gamma = 1$ signifying a Cauchy distribution. As γ decreases from 2, the distribution becomes increasingly peaked. Using Equation (3) the Levy index was calculated to be $\gamma = 1.222$, clearly indicating that the BTC-USD should be modelled as a Lévy distribution where the stochastic field is fractal in nature and susceptible to more frequent rare but extreme events called 'Lévy flights'.

The power spectrum of a signal can be used to characterise any correlation present within the signal [63]. This correlation is an indicator of any long- or short-term memory, another characteristic of fractal self-affine fields. In the spectrum, increased intensity at lower frequencies indicate a long-term market memory effect, with high intensity at higher frequencies indicating short-term memory. The function to determine the correlation between two time prices for a financial time series $u(t)$ is given by

$$u(t) \otimes u(t) \leftrightarrow |U(\omega)|^2$$

where $\leftrightarrow$ represents the transformation to Fourier space and the spectrum $U(\omega)$ is given by

$$U(\omega) = \mathcal{F}[u(t)] = \int_{-\infty}^{\infty} u(t) e^{-i\omega t} dt$$

The Auto-Correlation Function (ACF) is produced by taking the IFT of the power spectrum. This provides a measure of how present price data is correlated with future prices. According to the EMH, the time series should be uncorrelated and therefore exhibit no market memory. This would visually be rendered as a flat power spectrum and a peaked ACF around the origin.

The resulting power spectrum of the absolute log price changes is displayed in Figure 6, and shows that there is increased intensity at the lower frequencies, indicating long-term market memory and correlation within the data set. The corresponding filtered ACF is displayed in Figure 7 and shows that there is long-term market memory introduced at 425 days, with peaks of increasing amplitude at 550 and 800 days. Both of these results differ from that expected under the EMH. Therefore, evidence that the BTC-USD exchange does not conform to an efficient market view.

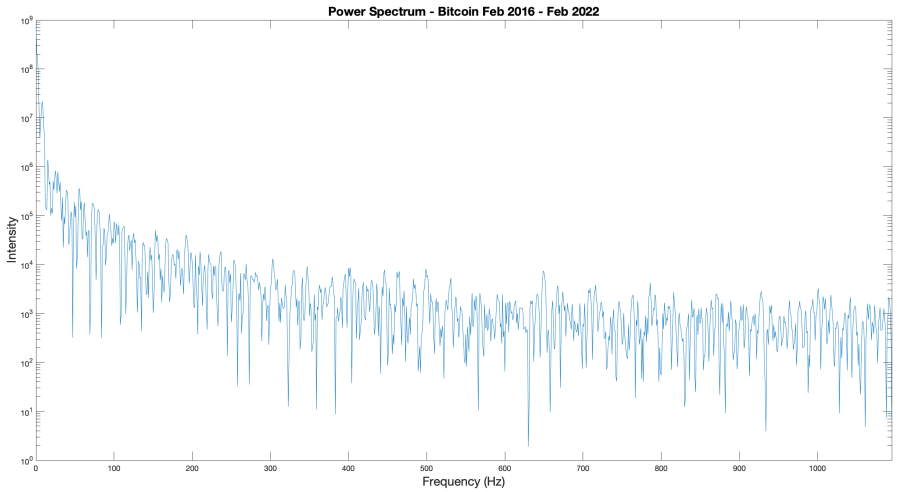


Figure 6. Power spectrum of absolute log price change of the BTC-USD (2016–2022).

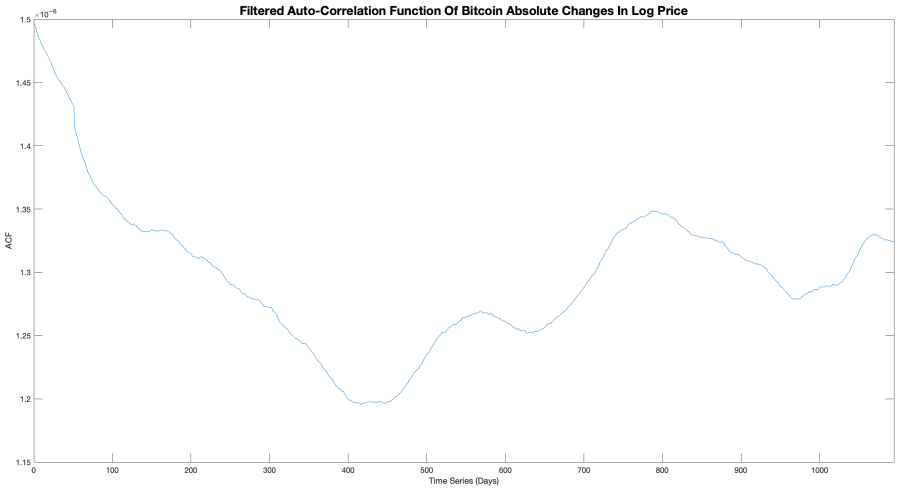


Figure 7. Auto-correlation function output for absolute log price changes of the BTC-USD (2016–2022).

Finally a quick comparison between Daily and Hourly price changes provides further evidence of self-affinity. In a fractal signal, the PDFs would be invariant of scale and therefore the same. Table 1 shows a compilation of all of the results for Ethereum and Bitcoin indexes. The results are all very similar confirming that both the markets are fractal.

Table 1. Comparison of BTC-USD and ETH-USD fractal indexes for both daily and hourly time series (2016–2022).

Index	BTC-USD Daily	ETH-USD Daily	BTC-USD Hourly	ETH-USD Hourly
Spectral Decay (α)	0.8185	0.8232	0.8714	0.8895
Hurst Exponent (H)	0.3185	0.3232	0.3732	0.3895
Fractal Dimension (D_F)	1.6815	1.6768	1.6286	1.6105
Levy Index (γ)	1.2218	1.2142	1.1452	1.1242

This body of analysis provides clear indication that the BTC-USD and ETH-USD exchanges are not efficient markets. The absolute log price changes are shown to not conform to a Gaussian distribution, with a peaked and skewed PDF observed showing that price changes are not independent. They display clear long-term correlation and market memory effects as well as being subject to Lévy flights. Therefore, both cryptocurrency fields have a highly non-linear, fractal, self-affine nature and require an FMH approach.

3.2. Analysis

To determine if the LVR (λ_σ) and BVR (β_σ) are useful metrics for recommending trading positions they must be tested on real-life markets. Backtesting is a well-established process, allowing traders to determine how effective a trading strategy is on historical data, thereby allowing them to assess its benefits or otherwise.

Using an evaluator function, the accuracy of the trading positions, represented by the zero crossings of the metric signal, can be quantified. Operating on the fact that when $z_c(t) > 0$, indicating a buy position, the following price trend is predicted to be positive. The next position (sell) should, therefore, be at a higher price. This case would be seen as a valid result; however, if the sell position was indicated at a lower price, this would be considered a failure as it has resulted in a net loss. This concept can be used in reverse for when $z_c(t) < 0$, indicating a short position and a predicted downward trend.

Applied to all trading positions in the backtesting output, a combined accuracy for the data set can be determined, quantifying how many recommended positions resulted in positive returns. The function for backtesting is presented in Appendix A.4.

For this body of work, individual time series for each year of BTC-USD/ETH-USD daily opening price data were made from 2016 to 2022, as well as time series representing individual months of hourly data from November to February 2022. This allowed a wide range of data sets to be analysed on different time scales.

The backtesting function receives parameters W, filtering window width, and T, the financial index calculation window. It also normalises the raw price data so that it can be displayed on the same scale as the resulting metric signal. Finally, the function produces a split graphic displaying the metric signal λ_σ (LVR), the filtered/unfiltered price data and z_c (left plot), and the non-normalised FTS (right plot). An example of which is displayed in Figure 8.

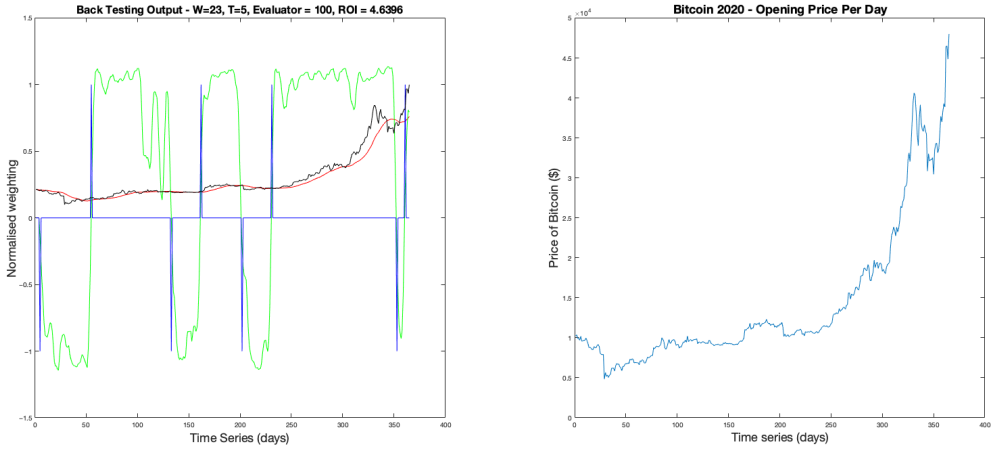


Figure 8. Example graphical output of the backtesting function displaying—λ_σ in green, z_c in blue, filtered price data in red and the unfiltered price data in black. The right hand plot shows the unfiltered data on its original scale for comparison.

This example uses parameters of $W = 23$ and $T = 5$. It shows 8 trades in total resulting in positive returns with a total ROI of ∼464%.

Another advantage of backtesting is the freedom to perform parameter analysis to find the optimum range that provides the greatest accuracy and returns. As eluded to in Section 2.4.3, the trading delay caused by pre-filtering coupled with micro-trends can cause errors in the recommended trading positions. Therefore, logically, there will be a 'sweet spot' of parameters where the delay does not heavily effect the system output and still adequately reduces the noise present in the signal.

Using the function $Optimisation(start, size, file)$, presented in Appendix A.5, backtesting was performed on an iterative basis, where each iteration increments the parameters W and T. The function produces two mesh representations of parameter combinations against evaluator results and returns, allowing comparison between parameters that achieve optimum accuracy and profit to determine if the two are correlated in any way. Due to possible micro-trends and over filtering, 100% position accuracy can result in a loss, as the evaluator operates using the filtered data set, not the real time series. Figure 9 shows an example evaluator mesh for the BTC-USD 2020-21 time series, where the x and z axes represent the parameter combinations and the y axis represents the accuracy percentage.

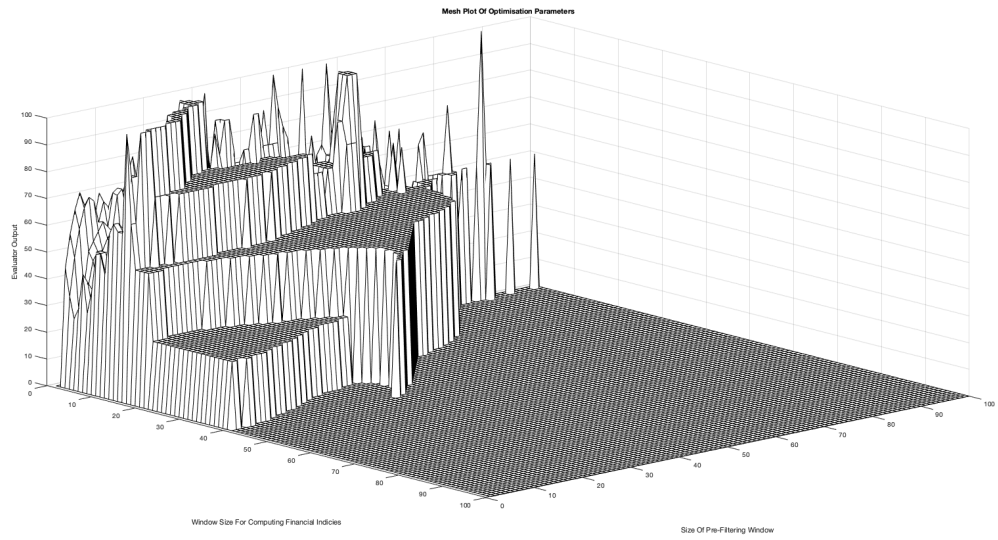


Figure 9. Parameter optimisation for BTC-USD 2020-21 where x, z axis are W, T parameter combinations and the y axis is the evaluator output accuracy.

3.3. Code Development

Previous studies into FMH have supplied portions of code that can be used as a base to implement the equations presented in Section 2.3 [64]. A number of modifications and additional features were added for speed and ease of use. All of these relevant functions are presented in the appendix. The main functions discussed in this section of the paper are: $Backtesting(W, T, time_series, 1)$, which is applied to the signal before calling the $Evaluator()$ function to assess the results.

3.3.1. System Optimisation

On initial testing, the $optimisation()$ function was taking ∼30 minutes to complete for a 100×100 parameter sweep (W, T). This put time limitations on progress. For this reason, the first step was to increase the performance of the code.

On an initial view, the iterative functions for calculating the metric signals, such as the Lyapunov Exponent (Listing 1), were using *for* loops, which take a total time of $N \times O(1)$, where N is the total number of iterations and $O(1)$ is the time required to complete the operations within one loop. The total time taken is increased, due to the function residing within another loop, to $n \times N \times O(1)$, where n is the total number of iterations of the outer loop.

Listing 1. Lyapunov Function with A *for* Loop.

```
%Compute the log differences of the data.
for n=1:N - 1
    d(n)=log(data(n + 1)/data(n));
end
d(N)=d(N - 1); %Set end point value.
L=sum(d); %Return the exponent.
```

Using vectorisation improves the speed of the function by a factor of N, as only 1 operation is required, therefore the new total time would be $n \times O(1)$. This is a significant improvement when there are multiple functions being called on an iterative basis, all using data sets of ∼1000 data points. At this number of values, the performance is increased by a factor of $a \times 1000$, where a is the number of functions called within the loop that required a *for* loop. In this case, a ∼30 minute runtime was reduced to < 6 seconds. A vectorised version of the function above is shown in Listing 2 given below.

Listing 2. Vectorised Lyapunov Function.

```
%-------------------------------------------------------
% Create a vector of the data points 1 step ahead
step_ahead_data=[data(2:end),1];
% Calculate the log of the ratio of data points.
log_dif=log(step_ahead_data./data);
% Append differences log with the final value.
log_dif(end)=log_dif(end-1);
L_Exponent=sum(log_dif); %Return the Lyapunov exponent.
```

Further reading of the code provided reveals the use of a proprietary moving average function. On inspection, this performed in the same fashion as the native Matlab filter function $movmean()$ with the exception that it modified the length of the input data, choosing to discard the first n elements that do not give a 'fully loaded' filter, instead of truncating the values. This has a dramatic knock-on-effect, as now the system outputs must be re-assigned a new length, using another *for* loop. Therefore, using the Matlab function

removes another loop and decreases the complexity, as there is no longer a need to sync the system outputs together.

This is a significant benefit as the initial method of assignment relied on an x-axis array and not an index position, meaning that the output arrays had T leading 0's before allocating the $T + 1$ element as the $T + W - 1$ value when compared to the input data. This made comparison very complicated. These changes dramatically improved the readability and speed of the functions which allowed for faster testing and therefore a more comprehensive analysis.

3.3.2. Additional Functions

In conjunction with the improvements to the system performance, a number of additional functions were created to ease analysis and handling of the cyrpto-currency historic data set. The first of these was a function capable of creating the required FTS from 7 years of hourly historical data. For this report, individual years of daily opening time data and months of hourly opening time data were required. Creating the $readfile(input, time_step, time_length, end_date)$ function, presented in Appendix A.9, allowed the raw full historical archive data set to be used. With the time step, where 1 is equal to 1 h, and the end date of the time series, in this case 12 February 2022 at 00:00:00, the function is capable of finding the first element and taking the next element separated by the time step for the *time_length* in days.

The ROI function, shown in Appendix A.6, is capable of applying the resulting long/short position indicators to the real-time data and determining the price change, be it negative or positive. The summation of individual trade returns can be compared to the initial investment, the price of the first position, allowing the total percentage return on investment to be presented to the user for comparison to other market indices and trading methods. It achieves this by receiving arrays of both data and trade positions, then indexing the unfiltered price data at said positions, calculating the difference between elements. Price changes are then appended to an array before a summation and the ratios are calculated. This function was incorporated into the backtesting system so that returns are calculated concurrently with the system metrics, also allowing a returns mesh plot to be compiled when performing optimisation. There were a number of other functions created during the course of this work; however, as they are not essential to the presentation of the results, they are not presented here.

3.4. Short-Term Prediction

The methods outlined in this paper so far attempt only to predict a future movement in an FTS and use this prediction to recommend trading positions. It makes no attempt to predict the scale of said movement. Considering the variability of volatility within cryptocurrencies, attempts of prediction at arbitrary time points would be futile. However, the underlying assumption in this fractal approach is that a 'strong' BVR or LVR amplitude signifies relatively stable market dynamic behaviour, and are therefore less likely to undergo a Lévy flight. This window of stability can be exploited to conduct short-term price predictions in the effort to assist a trader achieving an optimum trade position over a short future time horizon. Setting a threshold to define this 'stable period' will assign a certain degree of confidence to any predictions made. With this approach, the BVR signal now represents not only a trend indicator but also an opportunity for short-term price prediction.

4. Time Series Modelling Using Symbolic Regression

A period of trend stability empirically states that tomorrow's price will be similar to today's. Thus, formulating equations to represent the data up to that point in time (i.e., within the 'stable period') would allow discretised progressions into the future for a short number of time steps. This process can continue for as long as the stability period persists on a moving window basis.

This methodology suggests the use of a non-linear trend matching algorithm. In this respect, Symbolic Regression is a method of Machine Learning (ML) that iterates combinations of mathematical expressions to find non-linear models of data sets. Randomly generated equations using primitive mathematical functions are iteratively increased in complexity until the regression error is close to 0, or terminated by user subject to a given tolerance, for a pre-defined set of historical data.

4.1. Symbolic Regression

Symbolic Regression (SR) is a data-driven approach that identifies mathematical expressions describing relationships between input and output variables. It is a machine learning approach that aims to discover underlying mathematical expressions that best describe a dataset. Unlike traditional regression methods, which require the researcher to specify a model structure in advance, SR searches over a space of symbolic mathematical expressions (e.g., polynomials, logarithms, trigonometric functions) to identify both the functional form and parameters that optimally fit the data [65,66]. This is often achieved using evolutionary algorithms such as genetic programming, though recent work also integrates deep learning and reinforcement learning techniques to improve efficiency.

Key aspects of SR include:

(i) **Representation of Models**: Models are represented as expression trees, where nodes correspond to mathematical operators (e.g., $+$, $-$, $\times$, $\div$) and functions (e.g., sin, cos, exp, log), while leaves correspond to variables or constants.
(ii) **Search Space**: The search space includes all possible mathematical functions, expressions and combinations thereof within a specified complexity limit.
(iii) **Fitness Evaluation**: Fitness is based on criteria such as accuracy (e.g., mean squared error) and simplicity (e.g., number of nodes in the expression tree).
(iv) **Parsimony**: Balancing accuracy and simplicity prevents overfitting.

One major advantage of symbolic regression lies in its interpretability. Whereas models such as neural networks or ensemble methods (e.g., random forests) typically operate as "black boxes," SR produces explicit mathematical formulas that can be easily interpreted, analysed, and validated by domain experts. This feature is particularly valuable in scientific research and engineering, where understanding the governing dynamics of a system is as important as accurate prediction. Furthermore, SR can capture nonlinear relationships without presupposing the form of the dependency, making it highly flexible and capable of uncovering novel insights.

However, symbolic regression also presents notable limitations. The search space of possible symbolic expressions is vast and often computationally expensive to explore, especially for high-dimensional data. This can lead to long training times and difficulties in scaling SR methods compared to more established algorithms such as neural networks, which are highly optimised for large-scale data. Moreover, SR is prone to overfitting, as complex symbolic models may capture noise rather than true structure, particularly in small or noisy datasets. By contrast, decision trees or neural networks often incorporate regularisation techniques to mitigate this risk.

SR provides a transparent and flexible alternative to traditional machine learning approaches, excelling in domains where interpretability and discovery of governing equa-

tions are paramount, but facing challenges in scalability and robustness. The applications of this approach using a specific system—TuringBot—is discussed further detail in the following section.

4.2. Symbolic Regression Using TuringBot

Symbolic Regression is based on applying biologically inspired techniques such as genetic algorithms or evolutionary strategies. These methods evolve a population of candidate transformation rules over successive generations to maximise a fitness function. By introducing mutations, crossovers, and selections, the approach can explore a vast space of mathematical configurations. This approach is particularly useful in generating non-linear function to simulate complicated time series. In this section, we consider the use of the TuringBot to implement this approach in practice.

In this work, we use the *TuringBot* [67], which is a symbolic regression tool for generating trend fits using non-linear equations. Based on Python's mathematical libraries [68], the TuringBot uses simulated annealing, a probabilistic technique for approximating the global maxima and minima of a data field [69]. For this reason, we now provide a brief overview of the TuringBot system.

4.3. TuringBot

TuringBot [67] is an innovative AI-powered platform designed to streamline the process of algorithm creation and optimisation. By leveraging cutting-edge artificial intelligence, TuringBot enables users to generate, test, and refine algorithms for a variety of applications, such as data analysis, automation, and machine learning, without requiring extensive programming expertise. Its user-friendly interface and advanced capabilities make it an invaluable tool for professionals, researchers, and students seeking efficient solutions to complex computational problems. TuringBot stands out as a versatile and accessible resource in the ever-evolving landscape of artificial intelligence and technology. TuringBot.com is a symbolic regression software tool (version 3.1.4) designed to automatically discover analytical formulas that best fit a given dataset. Symbolic regression differs from traditional regression techniques by not assuming a predefined model structure; instead, it searches the space of possible mathematical expressions to find the one that best explains or 'fits' the data. TuringBot was developed to make this process efficient, intuitive, and accessible, especially for those without a background in machine learning or advanced data science.

Launched in the late 2010s, TuringBot was created in response to the growing need for interpretable artificial intelligence models. While most machine learning tools rely on complex neural networks or ensemble models that are difficult to understand and verify, TuringBot emphasised simplicity, transparency, and mathematical interpretability. By combining evolutionary algorithms with equation simplification techniques, TuringBot is able to produce human-readable formulas that describe complex datasets. This makes it particularly valuable in scientific, engineering, and academic contexts, where understanding the model structure is as important as prediction accuracy. The software is available for Windows and Linux and comes with a straightforward graphical user interface, allowing users to load data, configure parameters, and generate formulas with minimal setup. TuringBot also offers command-line integration for advanced users and supports exporting results for further analysis. As of the mid-2020s, TuringBot has gained a user base across various domains, from physics and finance to biology and control systems. It continues to evolve with improvements in computational performance and integration with modern workflows. The name "TuringBot" pays homage to Alan Turing, reflecting the tool's focus on combining algorithmic intelligence with human-understandable outputs.

In an era increasingly focused on explainable AI, TuringBot stands out as a lightweight, focused solution for data modelling grounded in classical mathematical reasoning. In this context, the system uses symbolic regression to evolve a formula that represents a simulation of the data (a time series) that is provided, subject to a Root Mean Square Error (RMSE) between the data and the formula together with other 'solution information' and 'Search Options'.

TuringBot automates the search for analytical models from data. At its core, the system employs a genetic programming-based algorithm that evolves candidate mathematical expressions through iterative recombination, mutation, and selection processes. This evolutionary approach allows the system to explore a wide variety of functional forms, including polynomials, logarithmic, exponential, and trigonometric structures, without requiring the user to predefine the shape of the model.

The fitness criterion used by TuringBot is primarily based on measures of predictive accuracy, most often the Mean Squared Error (MSE) between the model's predictions and observed target values. By minimising this error, the algorithm promotes the survival of symbolic expressions that better capture the underlying data structure. In addition to error-based metrics, the software also applies parsimony pressure, a complexity penalty that discourages overly large or convoluted formulas. This balance between accuracy and simplicity reflects the principle of Occam's razor and reduces the likelihood of overfitting.

Hyper-parameter tuning is handled through a combination of automated search and user-defined configuration. Parameters such as population size, mutation rate, crossover rate, and maximum tree depth play a central role in guiding the exploration of the symbolic space. In practice, TuringBot provides flexible controls that allow users to adjust these settings depending on the complexity of the dataset and the trade-off between exploration and exploitation. Iterative experimentation, often supported by built-in heuristics, ensures that hyper-parameters converge toward values that produce both accurate and interpretable symbolic models.

4.4. Comparison of TuringBot with Standard Machine Learning Models for Time Series Prediction

Time series prediction has traditionally been approached using statistical or machine learning models such as ARIMA, LSTM neural networks, and tree-based algorithms like XGBoost. In recent years, symbolic regression tools such as TuringBot have emerged as promising alternatives, offering distinct advantages in interpretability and adaptability. A comparison of these approaches highlights their relative strengths and limitations.

4.4.1. ARIMA

Autoregressive Integrated Moving Average (ARIMA) is a widely used classical model for time series forecasting. It performs well when data exhibit strong autocorrelations and stationarity. However, ARIMA requires significant pre-processing, including stationarity checks and differencing, and struggles to model nonlinear dynamics. In contrast, TuringBot, through symbolic regression, can capture nonlinearities directly by searching for functional relationships, reducing the reliance on strict assumptions about the data.

4.4.2. LSTM

Long Short-Term Memory (LSTM) networks, a deep learning extension of recurrent neural networks, are designed to model long-range dependencies in sequential data. LSTMs have achieved state-of-the-art results in complex prediction tasks, particularly when large datasets are available. Nevertheless, they are computationally expensive, require extensive hyper-parameter tuning, and produce models that are essentially "black boxes." TuringBot offers a counterpoint: its outputs are interpretable mathematical expressions that reveal underlying dynamics, which is particularly valuable in domains like finance where trans-

parency is critical. However, TuringBot may underperform compared to LSTMs in highly noisy or large-scale datasets where neural networks excel.

4.4.3. XGBoost

XGBoost, a gradient boosting algorithm, is known for its efficiency and predictive power across structured data problems. For time series, XGBoost is often used after feature engineering (e.g., lags, rolling statistics). While highly accurate, XGBoost models lack inherent interpretability and require manual feature design. TuringBot automates the discovery of functional relationships, potentially reducing the need for extensive feature engineering. Yet, XGBoost can handle large datasets more efficiently than symbolic regression, which may struggle with scalability.

4.4.4. Discussion

Table 2 provides a comparison of TuringBot with some standard models for time series prediction focusing on their strengths and limitations.

Table 2. Comparison of TuringBot with Standard Models for Time Series Prediction.

Model	Strengths	Limitations
ARIMA	• Well-established, and interpretable statistical framework. • Effective for stationary and linear series. • Good for short-term forecasts with clear autocorrelation.	• Requires stationarity and differencing. • Poor at capturing nonlinear dynamics. • Limited scalability for complex datasets.
LSTM	• Captures long-range dependencies. • Handles nonlinear and sequential dynamics. • High predictive accuracy on large datasets.	• Computationally expensive. • Requires large training data. • Functions as a 'black box' with low interpretability.
XGBoost	• High predictive accuracy across structured data. • Handles missing values and irregular data well. • Efficient and scalable.	• Requires manual feature engineering for time series. • Limited interpretability. • May overfit without careful tuning.
TuringBot (Symbolic Regression)	• Produces interpretable mathematical formulas. • Captures nonlinearities without strict assumptions. • Reduces feature engineering needs.	• Scalability challenges with very large datasets. • May be less accurate on noisy or highly complex sequences. • Search process can be computationally intensive.

Overall, TuringBot distinguishes itself by combining predictive modelling with interpretability, a rare feature among traditional models. While ARIMA, LSTM, and XGBoost

may outperform it in specific scenarios—such as stationary linear series (ARIMA), long memory dependencies (LSTM), or high-dimensional data (XGBoost)—TuringBot's ability to generate concise symbolic formulas provides unique insights into system behaviour. For applications where understanding model structure is as valuable as forecasting accuracy, TuringBot offers a compelling alternative to standard time series models.

In this work, we are interested in using the system to simulate a cryptocurrency time series of the types discussed earlier in the paper. For this purpose, a range of mathematical operations and functions are available including basic operations (addition, multiplication and division), trigonometric, exponential, hyperbolic, logical, history and 'other' functions. While all such functions can be applied, their applicability is problem specific. This is an issue that needs to be 'tempered', given that a TuringBot generated function will require translation to a specific programming language. This requirement necessitates attention regarding compatibility with the mathematical libraries that are available to implement the function in a specific language and the computational time required to compute such (nonlinear) functions. There is also an issue of how many data points should be used for the evolutionary process itself. In this context, it is noted that the demo version used in the case studies provided in the work only allows a limited number of data points to be used; i.e., quoting from the demo version (2024): 'Only the first 50 lines—data points—of an input file are considered'.

4.5. Example Case Study

Figure 10 shows an example of a set of 50 BTC-USD data points from 10 August 2021 to 5 September 2021 (red) and the trend match outcome of the TuringBot ML system (blue). These results were acquired by manually entering 50 data points and running the system. Here, the equation of the line was achieved after ∼103,000,000 iterations with an RMS error of 570.594 and mean absolute error of 0.983128%.

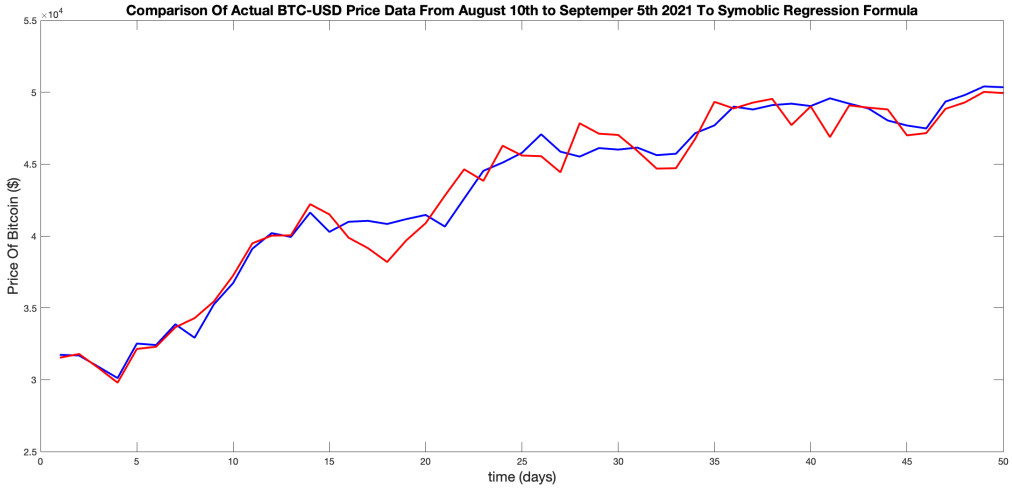


Figure 10. RealBTC-USD prices compared to TuringBot trend fit equation for 5 August 2021 to 10 September 2021.

The non-linear equation for the 'best fit' shown in Figure 10 is given by

$$\begin{aligned} f(t) = 29320.4 + (((-8.56537) \times (t + \operatorname{atanh}(\cos(t - 1.67383))) + \\ 821.732) \times \operatorname{round}(2.19581 \times \cos(\operatorname{round}((-898.977) \times (-0.978131 + t))) \\ + t - \tan(\cos(0.461446 - \exp(0.324379 + t))))) \end{aligned} \tag{15}$$

Figure 10 shows no future predictions; it is purely a trend match for a period of relative 'trend stability'. The principal point is that Equation (15) can be used to evolve a small

number of time steps into the future to estimate price fluctuation. By coupling the results of doing this with the scale of the LVR, for example, a confidence measure can be associated with the short-term forecast that are obtained. This is because a large positive or negative values of the LVR reflects regions in the time series where the log-term volatility is low, thereby providing confidence the forecast that is achieved. This is the principal associated with the financial time series analysis provided in the following section.

5. Bitcoin and Ethereum—Financial Time Series Analysis

With the derivation of market indexes and subsequent implementation in Matlab, an analysis of BTC and ETH FTS can be performed. Using the $readfile()$ function, a collection of series were created for two time scales, yearly opening day prices ranging from February 2016–February 2022, and monthly opening hour prices from November–February. A List of these series for BTC-USD is given in Table 3, the same collection was created for ETH-USD.

Using these, the analysis can be done on separate time scales, to not only determine the systems overall performance, but also whether the system can capitalise on the self-affine nature of the crypto-markets. For the purpose of the report, the BVR β_σ was used for analysis with the corresponding LVR results presented later in the paper for comparison.

Table 3. List of financial time series.

Name	Start	End	Time Step	Data Points
BTCUSD2021	12 February 2021	12 February 2022	24	365
BTCUSD2020	12 February 2020	12 February 2021	24	365
BTCUSD2019	12 February 2019	12 February 2020	24	365
BTCUSD2018	12 February 2018	12 February 2019	24	365
BTCUSD2017	12 February 2017	12 February 2018	24	365
BTCUSD2016	12 February 2016	12 February 2017	24	365
BTCUSD_janfeb	12 February 2016	12 February 2017	1	744
BTCUSD_decjan	12 February 2016	12 February 2017	1	744
BTCUSD_novdec	12 February 2016	12 February 2017	1	720

5.1. Daily Backtesting and Optimsiation

The first step in the analysis was to not only find the optimum parameter combinations for highest accuracy and profit, but to define a rule for selecting these parameters from a set range. Starting by running the optimisation function for the BTCUSD2021 time series, the range of parameter combinations that resulted in a 100% evaluator accuracy was observed in a three dimensional mesh plot. For the remainder of this section, the Filtering Window Width and Financial Calculation Window will be referred to as W and T, respectively. Figure 11 shows the resulting mesh graphic produced by the backtesting system.

It displays a broad range of W, T combinations. Due to the delay caused by the filtering process, it is logical to choose low values of W. From the mesh, there are W values in excess of 90 data points, which is equivalent to over 3 months of delay in the analysis. However, small values (i.e., $W < 10$) may not smooth the data enough to make the system perform well. This is confirmed by the lack of a 100% accuracy result with $W <\sim 20$.

The effect of different sizes of T, however, is not yet fully clear. Backtesting for two W, T combinations, one with a low T value and one with a high value, is presented in Figure 12 for BTCUSD2021. It shows that for $T = 2$ the metric signal becomes a binary representation, alternating between ± 1. This gives the trader no indication of a fluctuation in β_σ leading to a trade position.

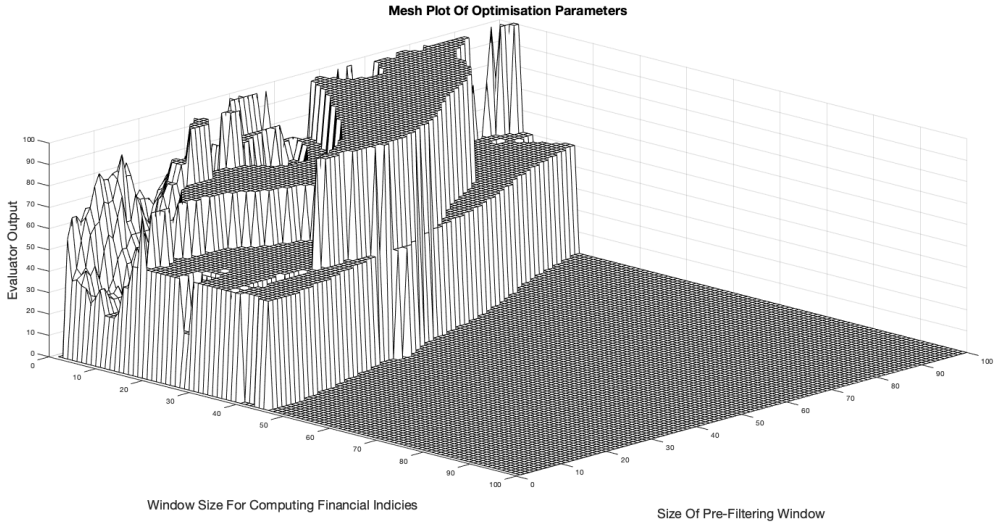


Figure 11. Mesh plot for parameter optimisation of BTCUSD 2021-22.

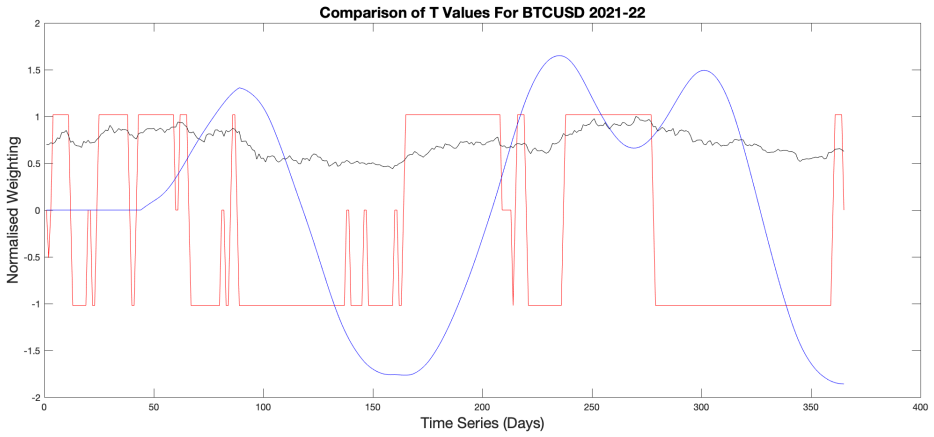


Figure 12. Comparison of backtesting metric signals for financial calculation windows, $T = 2$ (red) $T = 45$ (blue) with normalised price signal (black).

High T values result in sinusoidal fluctuations in β_σ making it hard to define periods of high stability and fast movements in trends. This is due to the assumption of stationarity within the windowed data, used to approximate the convolution integral in Equation (9), not being feasible for such a large value of T. Given that W values should minimise delay whilst providing enough filtering to reduce noise, a suitable limit for the values of T would be $T < W$.

From this comparison, T values should aim to be $2 < T < W$. W values should aim to be small enough to reduce system delay whilst maintaining a smooth enough price signal for good system performance.

To further study the optimum WT range, the array of WT combinations that returned 100% accuracy (WT_{op}) from the $Optimisation()$ function can be used to generate a new mesh plot of ROIs. As presented in Figure 13, this proves that not all optimum positions result in profitable trade positions. High-value combinations of WT generally result in a loss over the year. However, from the topology in Figure 13, it is clear that low values of $T \sim 6$ result in profitable trades irrespective of W.

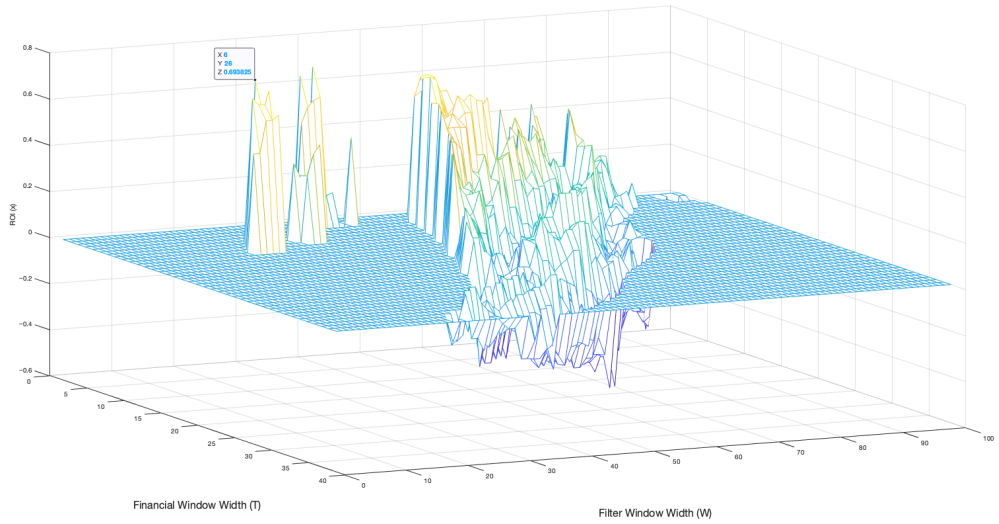


Figure 13. Mesh plot of Return On Investment (ROIs) For BTC-USD 2021-22 optimum parameter positions where x, z axis represents W, T values, respectively, and y axis represents the % ROI.

This provides evidence to suggest that the highest returns are achieved when a low value of T is chosen for the smallest W value, in this instance $W = 26, T = 6$ (For WT_{op} combinations only). To see how the returns for WT_{op} positions compare to non-optimum positions, i.e., WT combinations that produced less than 100% accuracy, a separate mesh plot was generated where all combinations are considered. This is presented in Figure 14, where yellow represents high ROIs and dark blue low (negative). For the mesh topology, this mesh provides evidence that the WT_{op} combinations do create high returns relative to all combinations. Interestingly, it also displays that very small W, T values create large losses and parameter sets where $T > W$ also create losses.

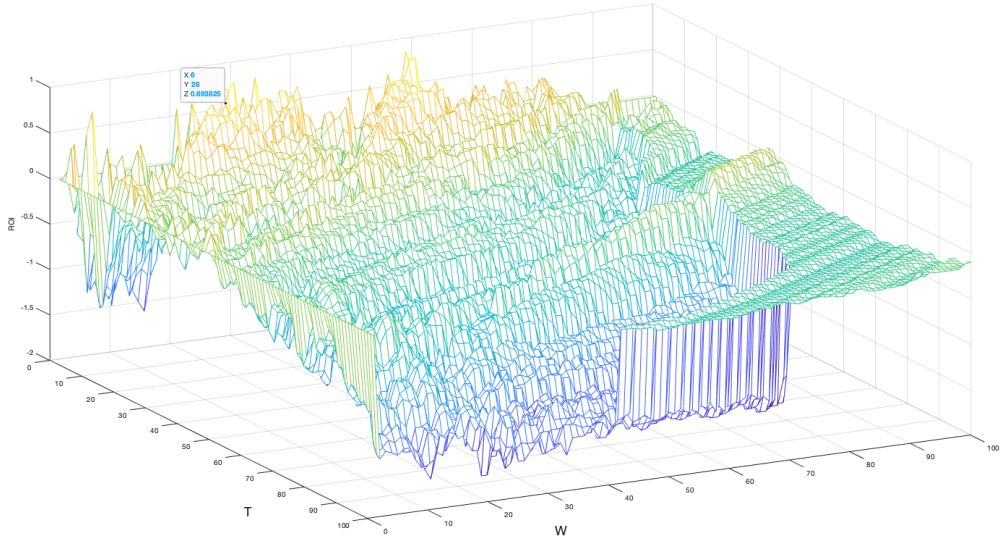


Figure 14. Mesh plot of percentage Return On Investment (ROIs) for BTCUSD 2021-22 including all W, T parameter combinations.

A significant discovery extracted from this mesh plot is that the highest possible returns do not occur at the optimum positions. The highest ROI from Figure 14 is selected and the surrounding peaks do rise higher. This can be interpreted as the result of micro-trends in the time series where non-optimum parameters have fortuitously recommended trades during a local peak or trough that has yet to influence the windowed data. The aim of

the system is to ensure accuracy and confidence in the trading strategy; therefore, optimum evaluator parameter combinations are preferred to highest profit achieving combinations. This priority definition warrants another evaluation of the optimum positions. Figure 15 presents a different perspective of the data displayed in Figure 11 where only the 100% accuracy combination are displayed on a two dimensional 'Top Down' view.

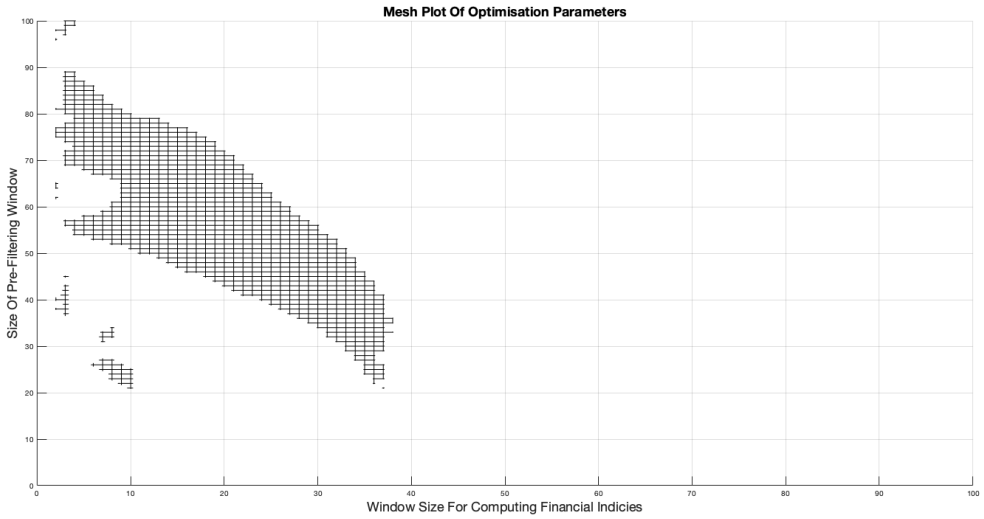


Figure 15. 'Top Down' view of optimised parameter mesh plot showing only W, T combinations that achieve 100% accuracy for BTC-USD 2021-22.

Figure 15 identifies a small grid at the bottom left hand corner where W values are within the lowest range and T values are $2 < T < W$. Knowing that this grid achieves the highest ROIs as shown in Figure 14, it is therefore preliminarily proposed that this 'Grid of Choice' (GOC) represents the best range of WT to be chosen from. To provide more evidence of this theory, the same approach was taken for the other yearly time series of BTC-USD. Each year displayed the same properties lending weight to the GOC theory and evidencing that the assumption of a fractal stochastic field is constant throughout the data. Figure 16 shows the resulting 'Top Down' optimum parameter mesh (left) and the ROI mesh (right) for BTC-USD 2020-21. It shows results consistent with that of BTC-USD 2021-22. The ridge of high (yellow) returns visible in Figure 16 are in some cases 20% higher than the returns achieved under WT_{op}. However, most of these parameter combinations violate the required conditions, in this case $T > W$ and high W values. This could be attributed to the fact that in 2020 BTC-USD had an almost constant upward trend.

With optimum parameters for 2021-22 ($W = 26, T = 6$), backtesting was performed. Figure 17 shows the graphical output from the function with, β_σ in green, z_c in blue, the filtered data in red and raw price signal in black. The same colour format will be used for all backtesting outputs in this report. It displays the 7 trades, resulting in a 69.3% return in a year when Bitcoin's value against the dollar widely fluctuated and lost value overall.

The nature of the trading delay is clear, with the filtered data (red) lagging behind the raw price data (black). The β_σ signal shows that there are periods of general trend stability in both bear and bull directions. By inspection, it can be seen that although some trade indications occur in the trough of the filtered data, when applied to the raw signal, the difference between the two price signals results in an overall loss for that transaction. This is a result of the micro trends Bitcoin displays coupled with the trading delay and inherent volatility.

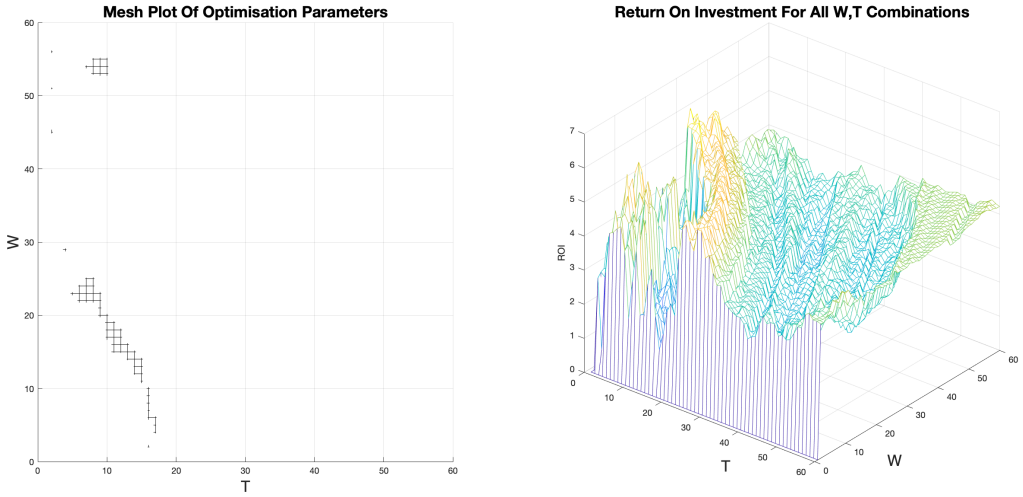


Figure 16. Optimisation graphical output depicting 'Top Down' view of optimum parameter combinations (**left**) and Return On Investment (ROIs) for BTS-USD 2020-21.

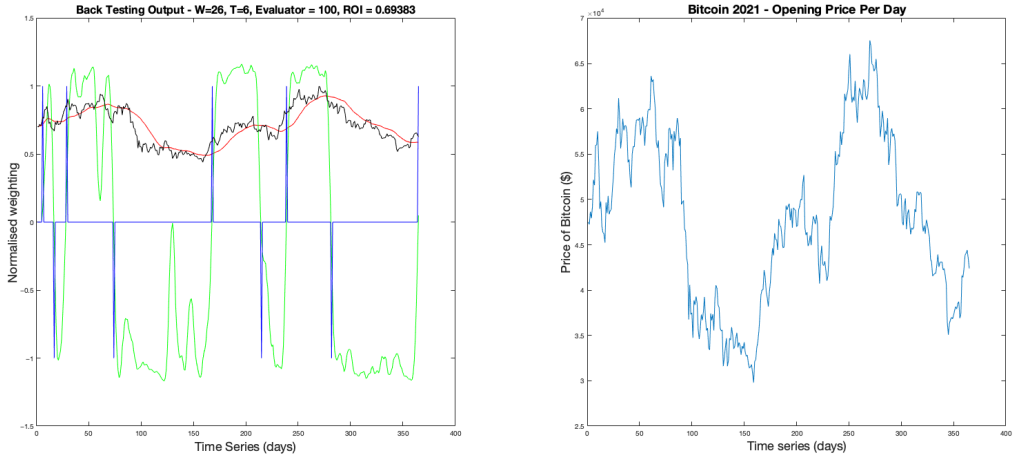


Figure 17. Backtesting for BTCUSD 2021-22 with parameters $W = 26$, $T = 6$.

The backtesting was completed on the same basis for the rest of the BTC-USD financial time series as well as ETH-USD. Results are displayed in Table 4. From these results, it is not always possible to achieve 100% accuracy. However, this does not lead to a loss for the year. It should be noted that during the analysis of ETH-USD data, the correlation between optimum parameters and high returns, including the GOC, was observed to provide further evidence in favour of the parameter selection theory.

Table 4. Percentage return on investment of BTC-USD and ETH-USD for daily financial time series using Beta-to-Volatility ratio.

Year	**BTC-USD**				**ETH-USD**			
	W	T	Evaluator (%)	ROI (%)	W	T	Evaluator (%)	ROI (%)
2016–2017	33	8	100	194	N/A	N/A	N/A	N/A
2017–2018	23	8	100	2487	17	3	100	11,093
2018–2019	20	3	87.5	38.5	21	3	100	76
2019–2020	43	4	100	267	17	3	94.4	108
2020–2021	23	5	100	464	15	5	100	1207
2021–2022	26	6	100	69	27	6	100	264

5.2. Hourly Backtesting and Optimisation

Given that the BTC-USD market has been shown to be a self-affine fractal signal exhibiting scale invariance, backtesting over a different scale, in this case hourly prices, should return similar results. However, as can be seen from Table 3 the monthly time series have double the number of data points. For this reason, the field is expected to have a higher level of detail and therefore noise. To inspect this, Figure 18 shows a 20-day extraction from the January to February 2022 time series. The high volatility and wild price fluctuations are more prevalent than for the daily data, with micro-trends occurring faster with bigger relative movements.

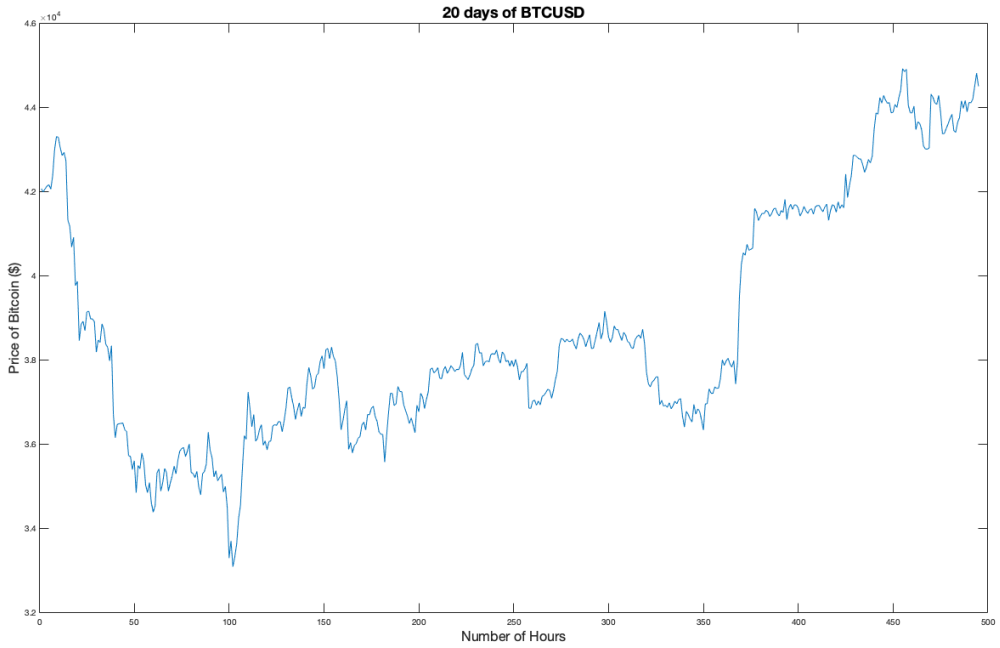


Figure 18. Twenty days of hourly opening price data for BTC-USD.

Due to the increased noise content, a higher level of filtering was expected to maintain an acceptable level of accuracy and therefore confidence in the recommended trade positions. This results in a GOC where T levels remain consistent but W values rise significantly.

As with the daily time series, the first step is to run the $Optimisation()$ function for the BTCUSD January–February 2022 field. Figure 19 shows the resulting mesh plot for all parameter combinations. Compared to the daily data, the general topology is far lower and as excepted, 100% accuracy is achieved with much higher values of W, in this case $W \sim 90$. T values remain consistently low, an expected result due to the self-affinity of the underlying price signal.

Taking a further look at the 'Top Down' view of the optimisation mesh plot in Figure 20, few accurate combinations of parameters exist. However, even with the sparsity of the results, there is still a clear grid containing the small range of W values for low T values. This is consistent with the expected findings. Producing the mesh plot for parameter returns, Figure 21, confirms that the GOC remains a source of strong returns.

Analysing the mesh plot of ROIs, the topology suggests an ideal location for parameters with returns around WT_{op} being a peak surrounded by low and negative results. This lends further evidence that the GOC is a valid theory. An interesting outcome is the plateau of high returns for high T values, irrespective of what filtering is applied. Many of these WT combinations are invalid due to $T > W$ or $W << 90$, the ideal range of filtering for this data. An explanation for this could be that for high values of T, the metric signal β_σ becomes heavily sinusoidal, containing low frequencies. This could result in low numbers

of trades operating at heavily delayed trade positions that are fortuitously executed. Other anomalous peaks in returns surrounding the origin also violate the $T < W$ rule. Such small filtering sizes increases the expected number of trades to high and infeasible values, due to the fees synonymous with trading cryptocurrencies. A comparison of these two invalid WT combinations is shown in Figure 22. In the $W = 80, T = 80$ backtest, a 0% accuracy still gives a positive return, confirming the anomalous nature of these combinations.

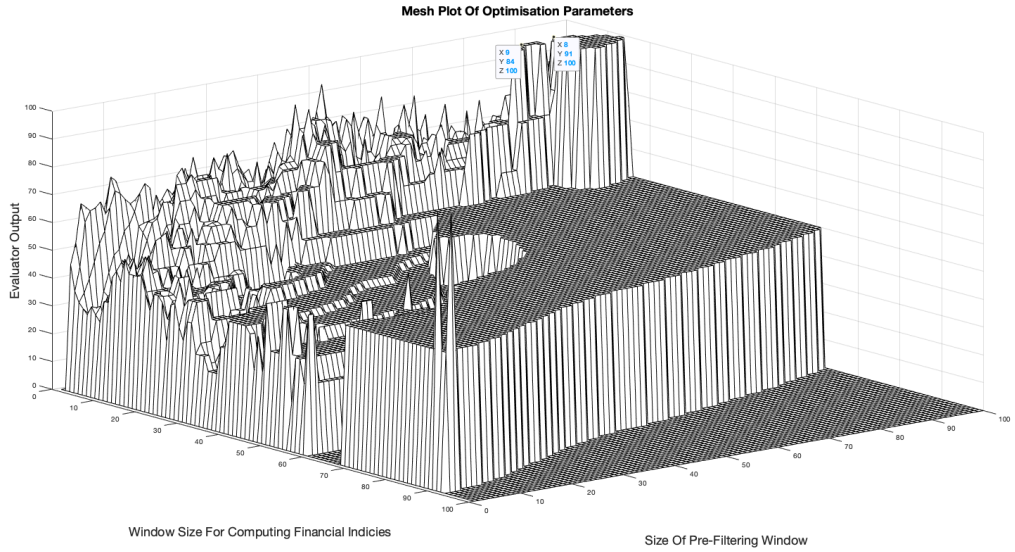


Figure 19. Parameter optimisation mesh for BTC-USD January–February 2022.

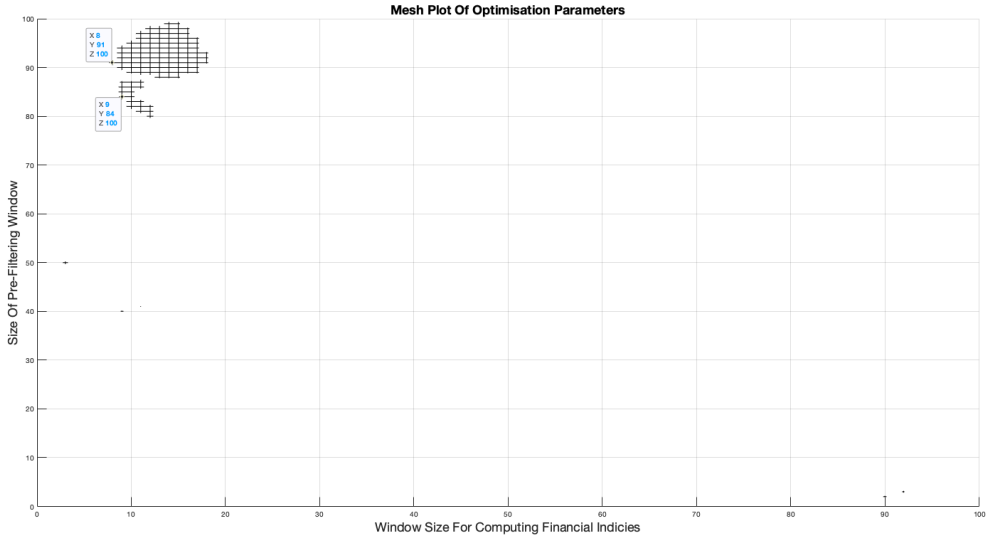


Figure 20. 'Top Down' view of optimised parameter mesh plot showing only W, T combinations that achieve 100% accuracy for BTC-USD January–February 2022.

The sharp and focused nature of the WT_{op} peak suggests that the effect of micro trends in the hourly time series is greater. Returns are reduced rapidly at small deviations from optimum combinations. The backtesting output for $W = 91, T = 8$ is shown in Figure 23. 9 trades are executed resulting in a 31.1% ROI. The increased volatility in the time series is reflected in the corresponding volatility in β_σ.

When applied to the other monthly time series, an interesting observation is the increase of filtering required as the fields evolve in time, suggesting that both BTC and ETH are entering a phase of high volatility. Table 5 displays the results for each monthly financial time series used in backtesting.

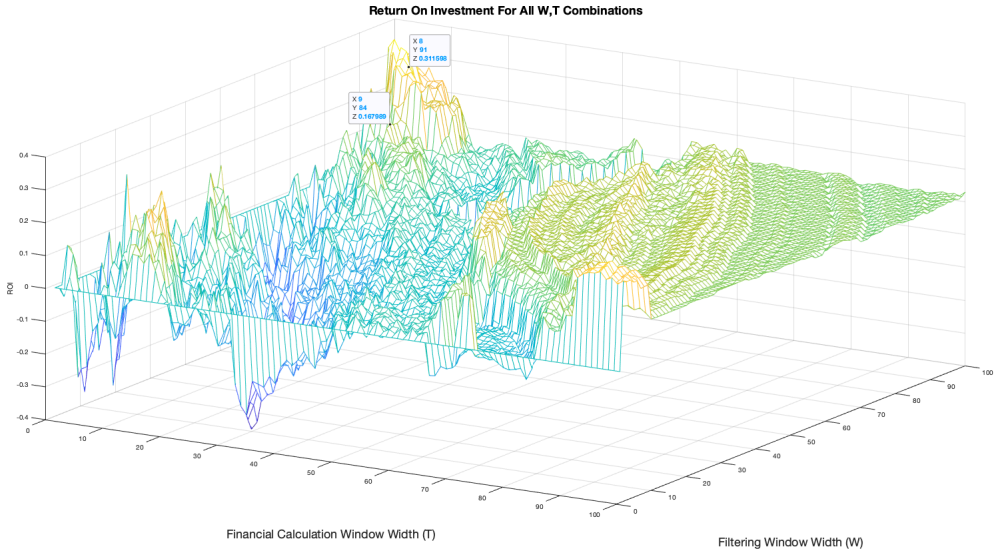


Figure 21. Mesh plot of percentage Return On Investment (ROIs) for BTC-USD January–February 2022.

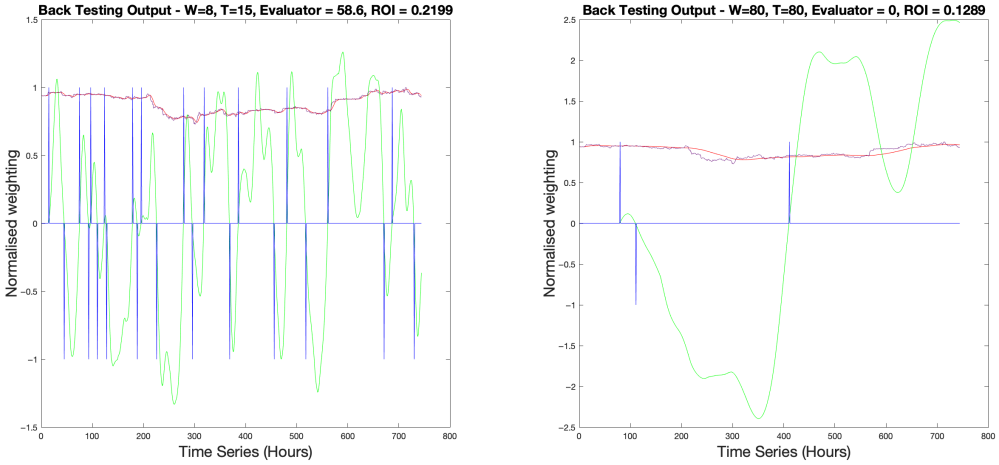


Figure 22. Backtesting output comparison of two invalid WT combinations—$W = 8, T = 15$ (**left**) and $W = 80, T = 80$ (**right**).

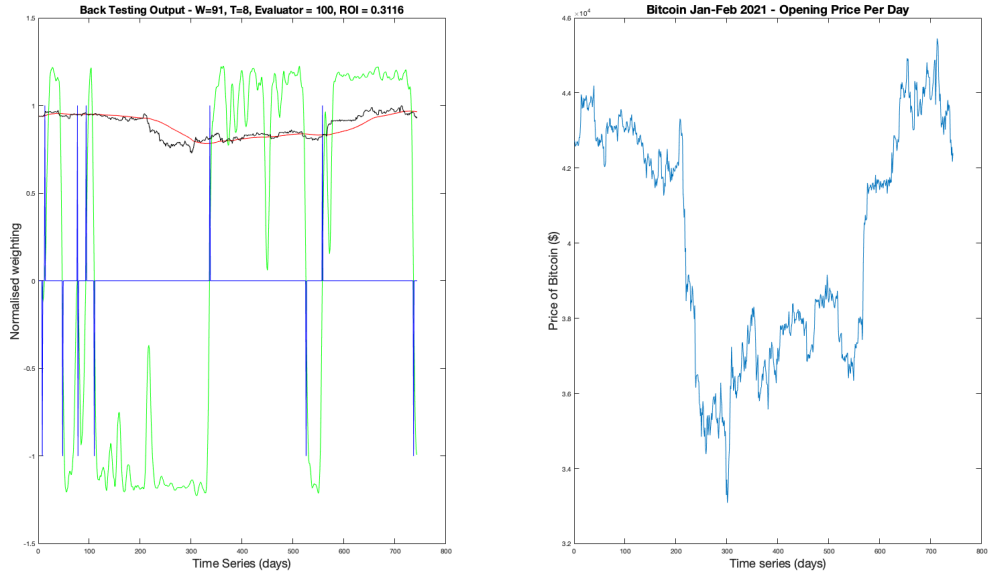


Figure 23. Backtesting for BTCUSD January–February 2022 with parameters $W = 91, T = 8$.

Table 5. Percentage return on investment of BTC-USD and ETH-USD for hourly financial time series using Beta-to-Volatility ratio.

Month	BTC-USD				ETH-USD			
	W	T	Evaluator (%)	ROI (%)	W	T	Evaluator (%)	ROI (%)
November–December	44	10	100	11.8	50	7	100	27.2
December–January	90	7	83.3	20	73	9	100	15.5
January–February	91	8	100	31.1	85	10	100	51.1

5.3. Analysis Using LVR

The backtests performed in previous sections were repeated for yearly financial time series using the LVR to observe any changes in results, The equivalent graphical LVR output for BTCUSD2021 is displayed in Figure 24. Results were consistent with the BVR metric, confirming that both ratios are valid for trend analysis. The LVR produced a metric signal with a greater amplitude than β_σ which provides more flexibility to change the conditions on which the trading positions are recommended. Currently trades are considered viable only when λ_σ crosses the axis. However, if the z_c signal was re-programmed to produce a delta peak when the signal reaches a certain threshold, say ± 2, this could reduce trading delay. A full set of ROI results for all BTC and ETH time series is presented in Table 6.

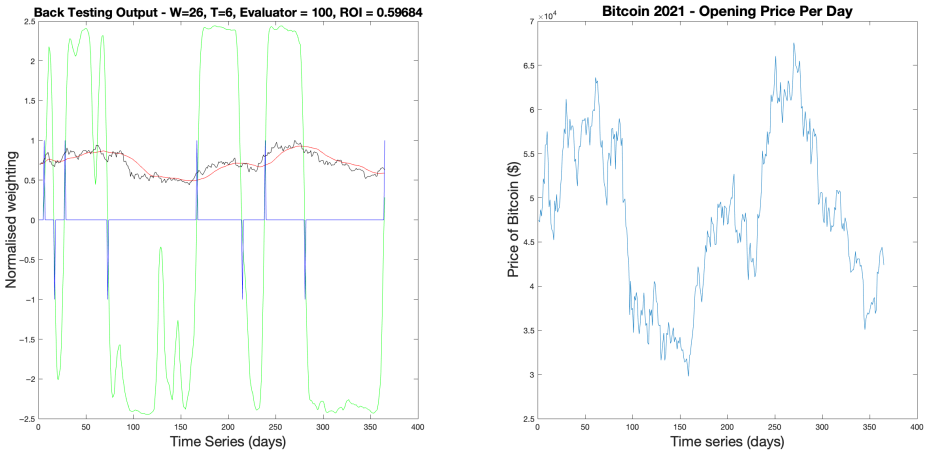


Figure 24. Backtesting for BTCUSD 2021-22 using LVR with parameters $W = 26, T = 6$.

Table 6. Percentage return on investment of BTC-USD and ETH-USD for daily financial time series using the Lyapunov-to-Volatility (LVR) ratio.

Year	BTC-USD (%)	ETH-USD (%)
2016–2017	137	N/A
2017–2018	2248	10,425
2018–2019	40.6	62.7
2019–2020	248	119.9
2020–2021	456	1158
2021–2022	59.7	248

5.4. Returns on Investment—Pre-Prediction

A full comparison of λ_σ and β_σ returns compared to the standard 'Buy and Hold' (B& H) strategy, where the price change over the whole time series is taken, is shown in Table 7.

Table 7. Percentage return on investment of BTC-USD and ETH-USD for daily financial time series using both LVR and BVR indicators compared to Buy & Hold strategy (B&H).

Year	BTC-USD			ETH-USD		
	ROI—$\lambda_\sigma(\%)$	ROI—$\beta_\sigma(\%)$	B&H (%)	ROI—$\lambda_\sigma(\%)$	ROI—$\beta_\sigma(\%)$	B&H (%)
2016–2017	137	194	162	N/A	N/A	N/A
2017–2018	2248	2487	700	10,425	11,093	7021
2018–2019	40.6	38.5	−60	62.7	76	−86.2
2019–2020	248	267	185	119.9	108	95.1
2020–2021	456	465	369	1158	1207	565
2021–2022	59.7	69	−10.7	248	264	58.7

It shows that the proposed system outperforms B&H for every financial time series under consideration, both for BTC and ETH coins, with the exception of BTCUSD 2016-17 LVR. In bear dominant years of high market loss, such as BTC 2018-19, the system was capable of producing a positive return. In other cases, where the year saw high overall gains, the system was able to improve further.

These returns are high when compared to other stock market indexes, with average returns considered to be 10%, including; S & P Commodity index returning an average 8.8% between 2009 and 2019, and the S & P 500 an average of 10.48% between 2005 and 2019. Overall, returns for the hourly data sets also proved to beat the B&H strategy.

5.5. Short-Term Price Prediction Using Machine Learning

As discussed in Section 3.4, periods of high trend stability, indicated by a 'strong' BVR amplitude, signify the opportunity for short-term prediction using Symbolic Regression (SR). As this period continues, non-linear formulas can be re-generated every day based on historical opening prices on a rolling window basis. Once generated for a time point t_n, the formula can be evolved for short-term future time horizons t_{n+1}, t_{n+2}, t_{n+3},... where n is the total number of data points used to create the formula. The hypothesis is that data points within the stable trend period can be used to generate non-linear formulas capable of guiding a trader to the optimum position execution, with the prices preceding the period being volatile and therefore detrimental to the SR algorithm.

Applying this to the BTCUSD2021 time series, a high-BVR period can be defined as $\beta_\sigma > 1$ or $\beta_\sigma < -1$ and designated $\mathrm{SR}_{\mathrm{period}}$. The BVR signal reaches this threshold on the 21st of August, indicating the ability to utilise SR. Advancing forward 34 days to September 10th, still within the $\mathrm{SR}_{\mathrm{period}}$, the TuringBot is used to generate a trend formula using the previous 34 days opening prices ($n = 34$). The resulting solution is

$$f(t) = 45020.3 + ((t/\tan(\tan(t))) + (146.767 \times (t - ((86.1446/(\mathrm{erf}(\cos(t)) + t)) + (6.92974 \times (\cos(2.49212 + t) - \cos(-3135.7 - \mathrm{floor}(\mathrm{gamma}(t) - 0.00100955))))))))) \quad (16)$$

Using this equation, data points $f(n_{35})$, $f(n_{36})$, $f(n_{37})$, ...can estimate price fluctuations over a short time horizon. Figure 25 shows the actual BTC-USD price data from 21 August to 20 September in black, the trend 'fit' for historical data up to 10 September, shown in blue, and then future estimated prices for +5 days to 15 September in red and +10 to 20 September in green. Each price estimation figure will use the same format.

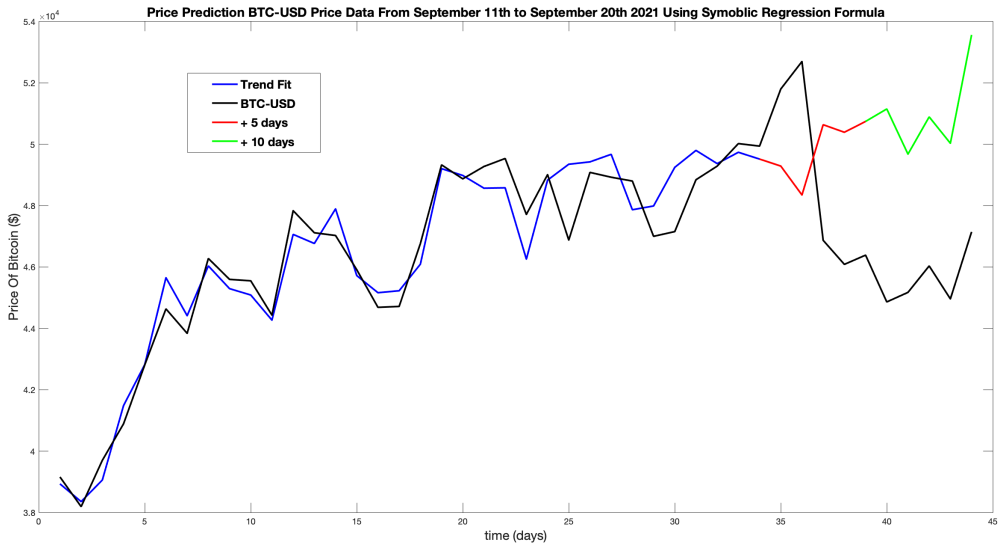


Figure 25. Price prediction using symbolic regression for BTC-USD 21 August to 10 September with price estimation from 11 September to 20 September (+10 days).

Observing the SR output graph, it is clear that the prediction provides no useful guidance. It does not predict the large drop in price on day 36, nor the increase in the preceding days. An optimal profit would have been achieved by exiting the long position on day 36 at $52,697. However, the prediction estimates $48,347, a price gap of nearly $5000. During the backtesting for this data set, a short position was recommended on 20 September at $47,144. Using Figure 25 as a reference, no increased profit would have been created. On 16 September (+6) β_σ drops below +1 and $\mathrm{SR}_{\mathrm{period}}$ ceases. This proves that no additional profit could have been made by the ML system. Formulas were generated for days 30–34, to see if an earlier prediction would have yielded better results. In every case, the estimated future prices gave no accurate guidance and failed to optimise the sell position. Figure 26 contains prediction plots for $n = 30$ (top) and $n = 32$ (bottom). The latter of which shows widely fluctuating prices and provides no confidence in its accuracy.

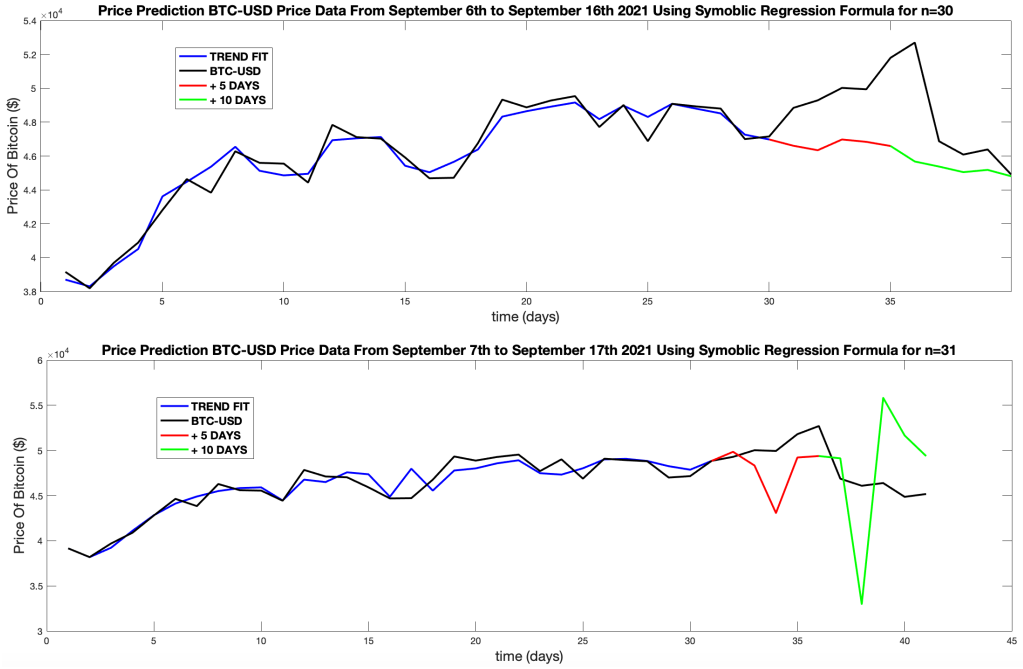


Figure 26. Price prediction using symbolic regression for BTC-USD for look-back window lengths of $n = 30$ (**top**) and $n = 32$ (**bottom**).

As a test, Figure 27 shows the non-linear formula generated at 7 September (Day 34) using the preceding 31 data points within SR_{period} and an additional 3 points from before the stable period began (i.e., $\beta_\sigma < +1$). This test goes against the ML hypothesis.

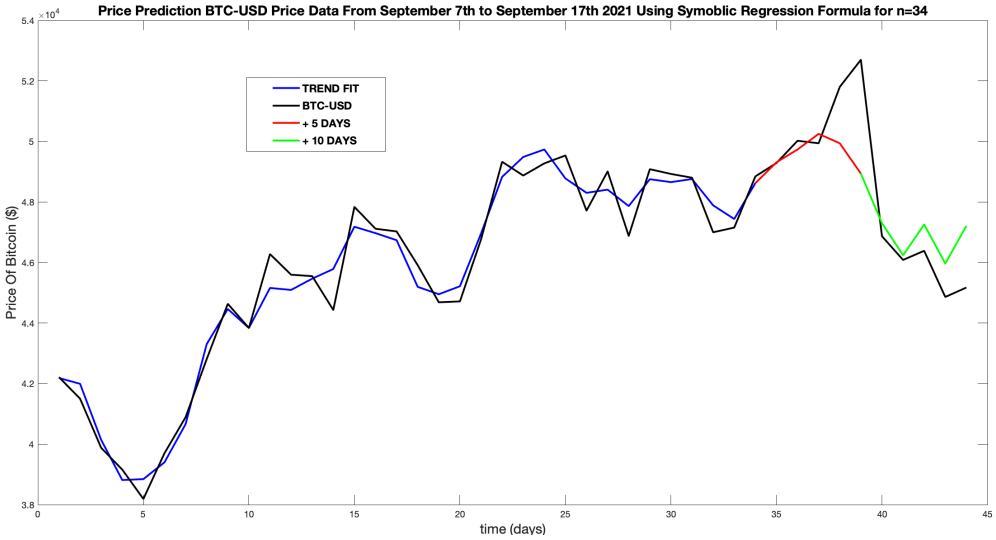


Figure 27. Price prediction using symbolic regression for BTC-USD 18 August to 7 September with price estimation from 7 September to 17 September (+10 days).

This output does provide credible guidance, indicating an exit of the long position on 10 September (Day 37) for $49,940. Compared to the zero-crossing recommendation on 20 September, this new trade position increased profit by $2800 or 6%, a significant increase in profit. To examine this further, Figure 28 shows a formula generated for 2 August to 7 September, now using 20 data points from outside the SR_{period} ($n = 50$).

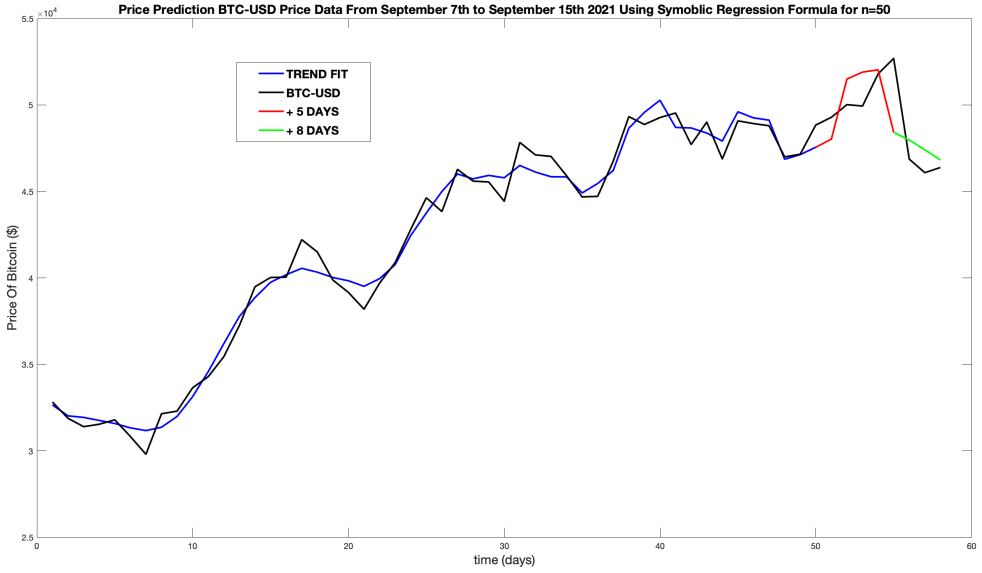


Figure 28. Price prediction using symbolic regression for BTC-USD 2 August to 7 September with price estimation from 7 September to 15 September (+8 days).

This result is based on using the following equation:

$$f(t) = 27510 - ((-159.209 - ((-1.54563) \times (t - 0.0670519 \times t \times \text{sign}(\cos(7.19577 \times t + 1.92497))))) \times (4.39689 \times (3.1596 \times \cos(0.486145 \times (-2.26983 + t)) + t) + (\text{abs}(t - 3.78218) - ((-16.5963)/\sinh(t))))) \quad (17)$$

giving a very accurate prediction. It correctly estimates the short price rise before the sharp fall. An indicated exit on 11 September at $51,800 increases profit by $4656 or 10%. The interesting observation here is that more precise price estimations came from extending the 'look-back' window beyond the SR_{period}. Due to the manual nature of the TuringBot, this process was only completed for the BTCUSD2021 time series and not for all time series.

6. Discussion

The analysis of both the cryptocurrencies considered in this research, has shown clear indications of non-normality (i.e., non-Gaussian behaviour). This is a defining characteristic of a fractal stochastic field. The peaked and broader side bands of the PDF for these financial signals deviate from a Gaussian PDF model, violating a core principle of the EMH. Disregarding the assumption of an efficient cryptocurrency market allows various indicators to be utilised to determine the financial fields nature. All these indicators were calculated as linear functions associated with spectral decay of the signal which was obtained through linear regression of the log power spectral plot. However, this method can lead to inaccuracies due to the erratic nature of the log power spectrum. As seen in Figure 29, the gradient of the log-log regression line could have a range of values.

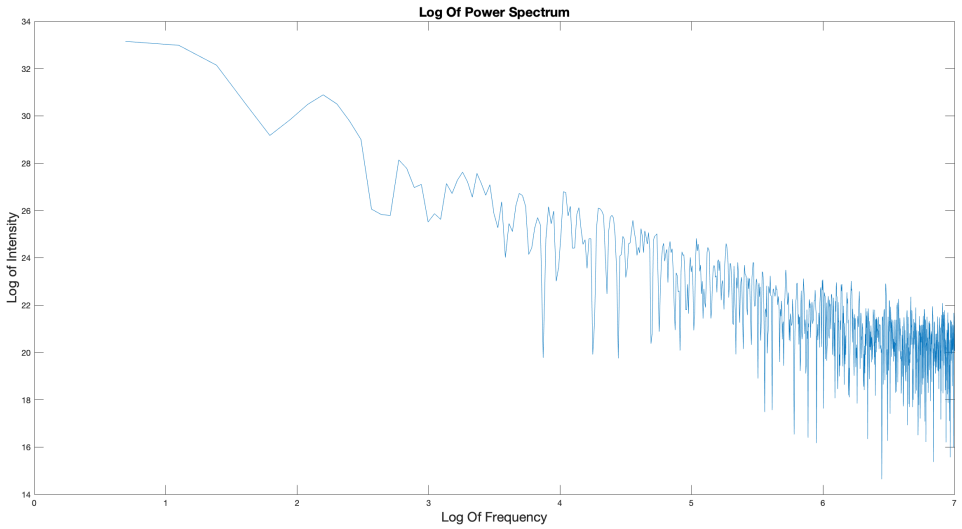


Figure 29. Log-Log power spectrum for BTC-USD February 2016–2022.

In order to obtain an accurate value of α, precise calculations of the spectrum and optimum region for fitting the regression line are required, which in most cases is not available [70]. As all of the indicators considered are linearly related, more precise methods of calculating their values are available, such as the 'Higuchi method' for determination of D_F and the algorithms for computing the Hurst exponent [71,72]. However, in the case of crypto-markets, the collection of indicators used showed such high deviations from normality that their inaccuracy would have made no difference to the conclusions in association with the application of a self-affine field model.

Each index showed a different aspect of deviation. The Hurst exponent (H) showed a level of anti-persistence not consistent with RWMs. The Levy index (γ) indicated a peaked PDF, indicative of a Lévy distribution, not a Gaussian distribution. The value of the fractal dimension (D_F) showed that the field has a narrower spectrum consistent with a self-affine signal. The ACF showed clear signs of data correlation, long-term market memory. This provided an overwhelming amount of evidence that the standard market hypothesis is not applicable and therefore the potential inaccuracy in the computation of α can be ignored.

During backtesting and optimisation, a range of ideal values for the W, T parameters was alluded to. The need to reduce trading delay, whilst minimising the noise in the signal,

proved to be dependent on the time scale of the time series, with larger filtering windows being required for the more detailed hourly data fields. Financial calculations had to be undertaken over small time windows relative to the length of data being analysed, whilst staying above the limit of 2. Unlike W, T values did not increase with an increase in scaling. Observing the WT_{op} positions, the location of the GOC can be acquired. This is related to the smallest range of W values for which T is minimised. This GOC was consistent, through all fields of the same currency and scale, leading to the guidelines for the parameter choice as follows: W—Smallest values for which noise is sufficiently removed; T—$2 < T < W$. Although this range, for the two parameters, did not always result in the most profitable trades, the high accuracy allowed high confidence in a profitable strategy. The fluctuation in ROI for neighbouring WT combinations can be explained by rapid micro-trends making an 'ideal' position recommendation impossible given the level of trading delay.

Another method to help overcome the trading delay is to redefine the conditions for which the position indicator z_c produces a Kronecker delta. Currently, positions are only recommended when either β_σ or λ_σ change polarity. However, if this was altered so that the Kronecker deltas were produced when a given threshold is passed, the system would react faster to changing trends. The implications of this are that price changes opposed to the current trend direction require less impact on the filter window to produce a Kronecker delta indicator.

The only down side to this change, is the increased risk due to parabolic metric flights caused by fast trend sweeps large enough to influence BVR/LVR results into recommending buy and sell positions in quick succession. However, looking at backtesting graphical outputs, it can be seen that a reduction in trading delay occurs more frequently. This change also depends on the metric used. The LVR produces a signal with a higher amplitude, giving more flexibility to choose a threshold range. Performances for both metrics were similar, as were returns in all backtests. As the metrics are derived from a different theoretical basis, one from a fractal model and the other from a chaotic model, their similar effectiveness further proves that cryptocurrency exchanges adhere to the FMH.

The self-affine behaviour of the cypto-markets under consideration allow for different time scales to be analysed. For example, Figure 30 shows an extraction from the daily backtesting output for BTC-USD 2021-22 for January–February (left) and compares it with the same backtesting time period for hourly data.

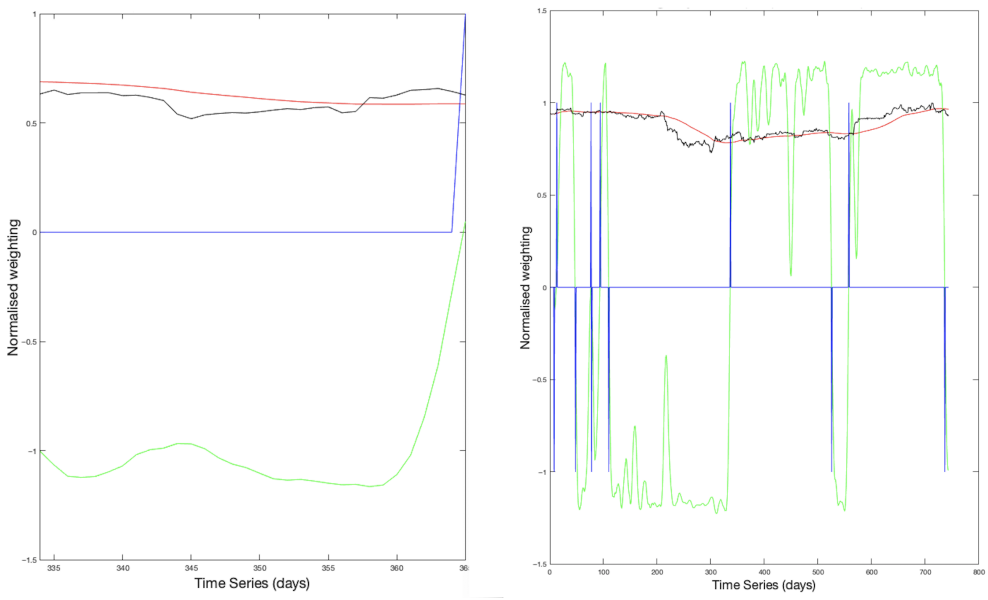


Figure 30. Comparison between daily and hourly system outputs for BTC-USD January–February 2022.

The hourly data (right) shows a number of trades being recommended with a 30% return over a period of overall loss in value. This is in stark contrast to the daily backtest, which showed no positions, therefore resulting in a loss. The most interesting comparison, is the opposite trade positions recommended for the last few data points, with daily data suggesting a purchase and hourly data suggesting a short market position.

The higher detail in the hourly data would be expected to produce more accurate positions. However, hourly data is particularly susceptible to micro-trends, requiring a large increase in the filtering window. This makes system outputs, based on hourly data, riskier; as evidenced from the reduced returns. Even though actual price changes over a month are less than over a year, the relative movement over the time period as a percentage is directly comparable over any scale.

Coupled with an increased number of trades per unit time, which has an effect due to the fees charged by the trading platform, the hourly trading system has its limitations. A combination of both strategies, whereby long-term trend analysis positions are complimented by short-term data, will increase the likelihood that the most profitable position can be achieved.

The application of ML to aid optimum trade executions provides price estimations that were highly inaccurate when using 'look-back' data contained within the SR_{period}. A range of non-linear equations, of increasing length, were created for sequential time steps within the FTS, each of which failed to predict the impending change of a trend, suggesting that increasing the window length within the SR_{period} has no effect on accuracy. However, accurate price estimations were produced when including a range of 'unstable' data points that proceed the window. During testing of look-back windows, that included values outside of the SR_{period}, it was observed that longer window lengths produce more precise results that correctly predict the magnitude and direction of the next 5–8 days within an accuracy of $\sim$90%. For example, using a window consisting of 50 data points ($n = 50$), including 20 unstable values, increased the profit for a single trade by $\sim$10%.

On re-evaluation of the approach, it becomes clear that any non-linear function based on a window of 'stable' data will only continue to display this trend when evolved forward in time. This is clearly a fundamental flaw, as, by definition, this approach will not achieve the goal of predicting a future change in trend. Extending the window beyond the SR_{period} creates an equation that accounts for both the low volatility trend and the high volatility movements, therefore giving a more accurate representation of the data and a better basis for future estimation. Tests on a window length of $\sim$30 produced vastly improved results, suggesting that increasing the window length is not the primary way to improve estimations. Nevertheless, increases did improve prediction accuracy by $\sim$20%. Thus, it can be concluded that an increase in the window length, using stable and unstable data, increases future price prediction accuracy.

Considering that the BVR and LVR are both calculated using their own rolling windows of length T, this extension should be at least T steps beyond the SR_{period}.

The manual nature of the TuringBot results in highly inefficient calculations. The lack of SR integration within the system is a significant flaw, if ML is to be recommended as a strategy to reduce trading delay. A proprietary SR algorithm or existing library will greatly increase usability. However, this is outside the scope of this work. Extensive manual testing of the ML method was not able to be completed over the time frame available for this work. A beneficial evolution would be an implementation in Python where large ML and evolutionary computing libraries are currently available. However, more recently, TuringBot.com has released an API for their software, allowing remote access to the SR algorithm from within the system code. This provides another approach to increasing efficiency.

During code development, using truncation to preserve data length and vectorisation to improve performance proved vital to conducting an analysis, the slow nature of the original functions making the program non-viable for continued and efficient trading. The creation of the $readfile()$ function made the system universal, where any .csv database of financial data could be uploaded and converted into a compatible time series.

Optimisation of the code resulted in a system that could generate an output in $\sim$0.0001 seconds. This was a significant improvement, making its use in conjunction with a live trading system viable. This decrease in computational time allows a continuous live data stream to be used, where fast tick times of a few seconds are implementable. However, this level of data requires a marked increase in filtering or new methods of creating price data. Opening daily prices can be replaced by an average of the underlying prices of the minimum time step.

The example results presented in Table 8 provide a useful comparison of percentage return on investment (ROI) for BTC-USD and ETH-USD when using LVR and BVR indicators against a simple Buy and Hold strategy. These figures clearly demonstrate the potential for improved profitability when applying indicator-based strategies. However, ROI alone is a limited measure: it captures absolute gains or losses but neglects the variability of outcomes, prediction accuracy, and the risk-adjusted performance of a strategy. For a more robust evaluation, the incorporation of performance metrics such as the Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and the Sharpe Ratio is essential.

Table 8. Percentage return on investment of BTC-USD and ETH-USD for hourly financial time series using both LVR and BVR indicators compared to Buy & Hold strategy (B&H).

Month	BTC-USD			ETH-USD		
	ROI—$\lambda_\sigma(\%)$	ROI—$\beta_\sigma(\%)$	B&H (%)	ROI—$\lambda_\sigma(\%)$	ROI—$\beta_\sigma(\%)$	B&H (%)
November–December	16.4	11.8	−23.8	27.5	27.2	−13.5
December–January	16.7	20	−13.4	12.8	15.5	−20.3
January–February	29.7	31.1	−0.78	47.8	51.1	−9.4

RMSE provides a measure of the deviation between predicted and actual values in the underlying financial time series. By penalising larger errors more heavily, RMSE highlights whether indicator-driven forecasts reliably track asset price movements. Similarly, MAPE offers an interpretable, scale-independent measure of forecast accuracy by quantifying the average percentage error. Together, these metrics ensure that strategies are not only profitable but also accurate and consistent in predicting price dynamics.

The Sharpe Ratio, in contrast, extends evaluation beyond forecast accuracy to financial viability. By calculating excess returns per unit of volatility, the Sharpe Ratio assesses how efficiently risk is converted into returns. A strategy with a higher ROI but excessive volatility may ultimately be less attractive than one with modest returns but a superior Sharpe Ratio. While the ROI results in Table 8 highlight profitability, supplementing them with RMSE, MAPE, and Sharpe Ratios provides a more comprehensive assessment. These metrics collectively capture predictive precision, error magnitude, and risk-adjusted efficiency, offering a balanced framework for evaluating trading strategies in volatile cryptocurrency markets.

7. Symbolic Regression and Fractal Dynamics: A Critical Appraisal

The study of complex systems characterised by fractal dynamics presents substantial methodological challenges for traditional machine learning models. Financial markets, turbulent flows, ecological systems, and even certain physiological processes often display

self-similarity, long-range dependence, and nonlinear feedback loops that resist representation by simple parametric models. Symbolic regression (SR), which searches for closed-form mathematical expressions directly from data, has emerged as a promising alternative to black-box methods such as neural networks or decision trees. This discussion evaluates why SR may outperform traditional models in the context of fractal dynamics, focusing on interpretability, functional discovery, scalability, and limitations.

7.1. Capturing Nonlinear and Self-Similar Structures

Fractal dynamics are typically defined by power-law relationships and recursive self-similar patterns across time scales. Traditional machine learning models, while capable of fitting highly nonlinear data, often do so in a statistical rather than structural manner. For example, a neural network may approximate a power-law curve through nonlinear activations, but it does not explicitly reveal the mathematical law underlying the data. In contrast, SR is specifically designed to uncover analytic functions, such as power laws or fractional differential equations, that are directly interpretable in fractal contexts. This functional discovery is crucial because fractal systems are often governed by compact mathematical expressions whose elegance is obscured by black-box models.

7.2. Interpretability and Theoretical Integration

A defining strength of SR lies in its interpretability. Whereas decision trees or neural networks produce models that must be treated as predictive artefacts, SR delivers explicit formulas. In fractal dynamics, interpretability is particularly important, as researchers seek not merely to predict but to understand the governing processes. For example, SR may reveal a scaling law consistent with fractional Brownian motion or uncover equations consistent with multifractal cascades. Such outcomes allow direct integration with existing theories of chaos and fractals, enabling models to serve as both predictive tools and theoretical insights. Traditional models rarely provide this level of epistemic value.

7.3. Adaptability to Multi-Scale Phenomena

Fractal systems exhibit multi-scale behaviour, where statistical properties at small scales resemble those at large scales. Symbolic regression naturally accommodates this property by exploring mathematical forms that explicitly capture scaling and recursive dynamics. In contrast, neural networks and ensemble methods are constrained by their architecture. While deep networks can approximate multi-scale patterns, they require large amounts of data and extensive training, often resulting in models that are computationally costly and prone to overfitting. SR, by seeking parsimonious mathematical rules, can often generalise more effectively across scales.

7.4. Robustness in Data-Limited Regimes

A further advantage of SR in fractal contexts is its efficiency in data-limited environments. Many fractal systems are difficult to observe over long horizons, and the available datasets may be noisy or sparse. Traditional machine learning methods thrive on large datasets, where their statistical approximations average out irregularities. By contrast, SR can leverage the inherent self-similarity of fractals to infer governing laws from relatively small samples, provided the symbolic search space is well managed. The explicit inclusion of parsimony criteria also reduces the risk of overfitting to noise, which is especially problematic in high-frequency fractal signals such as market tick data or turbulence measurements.

7.5. Limitations and Challenges

Despite these strengths, SR is not without limitations. The search space for symbolic expressions is combinatorially large, and naive implementations can be computationally intractable for high-dimensional data. While genetic programming and other heuristic algorithms alleviate this issue, they do not guarantee convergence to the globally optimal expression. Moreover, SR remains vulnerable to overfitting when model complexity is insufficiently penalised, especially in fractal systems where noise can mimic structural patterns. Traditional models such as random forests or gradient boosting often exhibit greater robustness to noise by design.

Another challenge is the interpretive bias introduced by SR's reliance on a predefined library of functions. If fractal dynamics in a given system are best described by a function outside the candidate set (for example, certain fractional calculus operators), the algorithm may converge to an approximation that captures behaviour without revealing the true generative law. Neural networks, by contrast, are universal approximations and may perform better when theoretical constraints are weak or unknown.

7.6. Comparative Value in Practice

The relative value of SR over traditional models depends on the **research objective**. If prediction accuracy is the sole criterion, ensemble methods or deep networks may sometimes outperform SR, particularly on high-dimensional datasets with weak theoretical grounding. However, when the objective is to uncover interpretable, generalisable laws consistent with fractal dynamics, SR is superior. Its ability to generate human-readable models offers explanatory power, a feature increasingly demanded in both academic and applied domains such as finance, physics, and environmental science.

8. Conclusions: Summary, Discussion and Future Directions

Symbolic Regression (SR) offers a powerful approach for modelling fractal dynamics, with strengths in interpretability, adaptability to scaling phenomena, and robustness in data-limited environments. Although computational challenges and the risk of overfitting remain, SR's ability to uncover underlying mathematical relationships sets it apart from traditional black-box methods. In situations where understanding the generative structure of fractal systems is as important as making accurate predictions, SR provides a methodological advantage that can complement, and in some cases outperform, neural networks or decision trees. Its primary value lies not only in prediction but also in linking empirical data to theoretical insight, making it a particularly suitable tool for advancing the study of fractal dynamics.

The fundamental analysis of BTC and ETH cryptocurrency markets is consistent with previous research. While most studies suggest that BTC is evolving toward a more mature and efficient market, the analysis presented here demonstrates that it remains far from fully efficient. Because no prior research has applied similar approaches for price and trend prediction in cryptocurrency markets, direct comparisons with other studies are not possible.

The analysis further indicates that cryptocurrency price data do not conform to a normal distribution. For Bitcoin, the Hurst exponent was calculated as 0.3185, indicating anti-persistence and short-term dependence. The Lévy index was measured at 1.2218, with a fractal dimension of 1.6815. Long-term market memory was also evident from the autocorrelation functions. These results confirm that cryptocurrency markets are inefficient, validating the relevance of a fractal modelling approach. In this context, the self-affine properties of the markets were confirmed by observing similar probability density functions (PDFs) across scaled time series, a pattern that is also reflected in Ethereum markets.

8.1. Summary

Based on the principles of the FMH, two fundamental indicators were derived from distinct theoretical frameworks, each scaled by volatility to produce a pair of trend analysis metrics. The effectiveness of these metrics is reflected in the zero-crossings of their signals, which indicate potential changes in trend. Positive returns were observed across all analysed time series, even during bear-dominated periods.

Parameter sweep analysis provided a basis for selecting optimal parameter values. Filter window sizes must be minimised to reduce trading delays while still removing noise to ensure accurate system outputs. For a given range of filter widths (W), the window length (T) should satisfy $2 < T < W$. This parameter selection rule was shown to yield high accuracy and profitable returns. Short time steps produce time series with elevated noise levels, reducing trend prediction accuracy and increasing susceptibility to micro-trends, which can lead to positions that incur losses. Consequently, a combination of long- and short-term analyses is recommended, where short-term results identify the most opportune moments to enter positions as guided by the associated long-term trends.

The application of machine learning (ML) to predict short-term price fluctuations and thereby reduce trading delays has been shown to be unreliable unless the data window extends at least T steps beyond the $\mathrm{SR}_{\mathrm{period}}$. Longer windows containing more data points improve predictive accuracy. In general, the TuringBot did not provide an efficient implementation of SR. Future system improvements should therefore focus on fully integrating ML-based approaches.

The analysis demonstrates that this methodology consistently outperforms traditional 'Buy & Hold' strategies across all time series considered and surpasses benchmark returns set by conventional stock market indices. Given the overall performance of cryptocurrencies over the past five years, this outcome is unsurprising; however, profitable returns were still achieved during bear-dominated periods.

Future research should investigate scale-combination strategies to further enhance profitability. A natural progression would be the development of a fully integrated SR algorithm. Additionally, further studies could explore whether a broader set of cryptocurrencies exhibit fractal properties and examine potential price correlations among them.

8.2. Discussion

The aim of this publication is to provide readers with a comprehensive background on the algorithms developed in this study. Apart from the introductory material presented in Sections 1 and 2, the results reported here are, to the best of the authors' knowledge, novel and original, particularly in terms of their application, the numerical outcomes obtained, and, more specifically, the type of data analysed—namely, cryptocurrencies.

A key feature of this work is the provision of the Matlab code used in the investigation. This enables readers to reproduce the results and extend the methodology, which the authors consider an important contribution, particularly for those interested in applying these algorithms to commodities markets or other financial domains.

It is worth noting that TuringBot is only one of several emerging tools in the field of genetic programming capable of evolving nonlinear functions to simulate real-world noise. Several Python-based alternatives allow the development of fully integrated programs without reliance on external applications such as TuringBot. One example is *gplearn*, a Python library implementing genetic programming with an API inspired by and compatible with scikit-learn [73].

8.3. Future Directions

The approach considered in this work is algorithmic, in that both long-term trends and short-term price values are derived from a set of quantifiable algorithms and their optimisation. For long-term trends, these algorithms are based on functions that compute metrics associated with the Fractal Market Hypothesis (FMH). For short-term price predictions, the algorithms are derived from nonlinear functions generated iteratively using symbolic regression. In this sense, the methods presented here integrate conventional time series modelling with machine learning techniques.

This approach contrasts with deep learning models, which can capture complex relationships between features in time series data and account for long-term dependencies [74]. While deep learning models have the potential to improve prediction accuracy, they require large volumes of training data to operate effectively.

A natural avenue for future research is to compare the algorithmic approach presented in this paper with deep time series forecasting models, using data specific to cryptocurrency trading and other commodities.

Author Contributions: Conceptualisation: J.B.; Methodology: J.B.; Software development: A.B.; Validation: A.B.; Formal analysis: A.B.; Investigation: A.B. All authors have read and agreed to the published version of the manuscript.

Funding: The research reported in this article was partly funded by the Science Foundation Ireland.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The software developed to present the results given in this work was based solely on the use of Matlab and the TuringBot symbolic regression software.

Acknowledgments: The authors acknowledges the support of the Technological University Dublin, Ireland, the University of Bath and Imperial College London, UK.

Conflicts of Interest: The authors declare no conflicts of interest.

Acronyms

List of abbreviations used in the text.

Acronym	Meaning
ACF	Autocorrelation Function
ARIMA	Autoregressive Integrated Moving Average
AVR	Automatic Variance Ratio
B&H	Buy and Hold
BTC	Bitcoin
BVR	Beta-to-Volatility Ratio
CEA	Commodity Exchange Act
CFTC	Commodity Futures Trading Commission
DFA	De-trended Fluctuation Analysis
EMH	Efficient Market Hypothesis
ETH	Ethereum
FMH	Fractal Market Hypothesis
FT	Fourier Transform
FTS	Financial Time Series
GARCH	Generalised Autoregressive Conditional Heteroskedasticity
GOC	Grid of Choice

IFS	Iterated Function System
IFT	Inverse Fourier Transform
LSTM	Long Short-Term Memory
LVR	Lyapunov-to-Volatility Ratio
MAPE	Mean Absolute Percentage Error
MF-DFA	Multifractal De-trended Fluctuation Analysis
ML	Machine Learning
MSE	Mean Squared Error
NFT	Non-Fungible Tokens
PDF	Probability Density Functions
R/S	Re-scaled Range Analysis
ROI	Return on Investment
RMSE	Root Mean Square Error
RWM	Random Walk Model
SR	Symbolic Regression
USD	United States Dollar

Appendix A. Matlab Functions

Appendix A.1. Lyapunov

```
function L_Exponent = Lyapunov(data)
%-------------------------------------------------------

% Function to compute the Lyapunov Exponent 'L' for data
% length N of a time period defined by tau.
% INPUTS:
% data -     Input data vector for the calculation

% OUTPUTS:
% L_Exponent -  Output of Lyapunov calculation
%-------------------------------------------------------
% Create a vector of the data points 1 step ahead
step_ahead_data=[data(2:end),1];
% Calculate the log of the ratio of data points.
log_dif=log(step_ahead_data./data);
% Append differences log with the final value.
log_dif(end)=log_dif(end-1);
L_Exponent=sum(log_dif); %Return the Lyapunov exponent.
end
```

Appendix A.2. Volatility

```
function Vol=Volatility(data)
%-------------------------------------------------------------
%Function to compute the Volatility for data of size N.
% INPUTS:
% data -     Data for computation
% N -        Length of the data

% OUTPUTS:
% Vol -      Result vector for the calculation of volatility
```

```
%-------------------------------------------------------------

% create a vector of the data points 1 step ahead of data
%using a dummy variable '1' for last value
step_ahead_data=[data(2:end),1];

% calculate the log of the ratio of data points
log_dif=log(step_ahead_data./data);

% append differences log with the final value to negate
%final value calculated with dummy variable
log_dif(end)=log_dif(end-1);

% Return the Volatilty
Vol=sqrt(sum(abs(log_dif).^2));
end
```

Appendix A.3. Beta

```
function beta=Beta(data,N)
%-------------------------------------------------------------
% Computation of the Beta Index using the least squares
%algorithm on 'data' of length N.
% INPUTS:
% data (array) -    input window of data for calculation
% N (int) -       Length of Data

% OUTPUTS:
% beta -   Result of beta computation
%-------------------------------------------------------------

% compute logarithm of the data.
xdata=log(1:1:N);
ydata=log(data);

% Compute each term of the least squares formula
%associated with log scaling law beta*log(data)
term1=sum(ydata).*sum(xdata);
term2=sum(ydata.*xdata);
term3=sum(xdata)^2;
term4=sum(xdata.^2);

% Compute beta
beta=(term1-(N*term2))/(term3-(N*term4));
end
```

Appendix A.4. Backtesting

```
function [Result,ROI] = Backtesting(W,T,time_series,show_plot)
%-------------------------------------------------------------
% FUNCTION: Back-testing procedure to compute accuracy of
```

```
%           trend analysis
% INPUT PARAMETERS:
% (int) W   - Size of window for pre-filtering using moving
%             average of data
% (int) T   - Size of window for computing financial indeces.
% (array) time_series - Array of close/opening prices to be
%                       used, be it hourly, daily or weekly
% (int) show_plot   - Flag to show figure output or supress
%                     during optimisation

% OUTPUT:
% Result - Array containing all the output of the evaluator.
%-------------------------------------------------------------



% Create a row vector of the time_series and normalise
data=transpose(time_series);
data=data./max(data);

% Apply a moving average filter
Fdata=movmean(data,[W-1,0]);
M=length(Fdata);

% Pre-define variables
S = zeros(1,M); V = zeros(1,M);
R = zeros(1,M); F = zeros(1,M);
ZC = zeros(1,M); G = zeros(1,M);
x = [1:1:M];

%Start moving window proceses with look-back period of W.
for m=T:M

    % Window the prior data.
    s=Fdata(m-T+1:m);

    % Compute the Lyapunov Exponent and Volatility
    %B(m) = Beta(s,T);       % Beta index
    S(m) = Lyapunov(s);      % Lyapunov Exponent
    V(m) = Volatility(s);    % Volatilty
    R(m) = S(m)/V(m); %Lyapunov to Volatiloty Ratio (LVR).

    %%% Compute zero crossings %%%
    % Compute mean LVRs (post filtering then take mean
    %of prior data using look-back window T).
    F(m)=mean(R(m-T+1:m));

    % Evaluate zero crossings from negative to
    %positive half-space
    if F(m)>0 && F(m-1)<=0
        ZC(m) =+ 1; %Zero-crossing given positive flag.
```

```
        G(m) = Fdata(m);%Assignment for later evaluation.
    end

    % Evaluate zero crossings from positive
    % to negative half-space
    if F(m)<0 && F(m-1)>=0
        ZC(m) = -1;% Zero-crossing given negative flag.
        G(m) = Fdata(m);% Assignment for later evaluation.
    end
end % Repeat process and update plot

% Evaluate accuracy of strategy.
Result = Evaluator(ZC(W+T-1:end),G(W+T-1:end),M-W-T-1,T);

% Calculate the return on investment
ROI=Returns(ZC(W+T:end),time_series(W+T:end));

% plot time-series and output
if show_plot == 1
    figure(3); subplot(1,2,1);
    plot(x,Fdata,'r-',x,F,'g-',x,ZC,'b-');
    hold on, plot(data,'k-'); hold off;
    subplot(1,2,2); plot(time_series);
end
end
```

Appendix A.5. Optimisation

```
function Op_Pos = Optimisation(start, size, file)
%-------------------------------------------------------------
% Function for optimising the parameters of the FMH system
% INPUT PARAMTERES:
% Start (int):  Start position of the optimisation window,
%               must have a min size of 2 to allow
%               calculations within functions called
%               within function
% Size (int):   Size of the filtering window
% File:         The financial time series

% OUTPUTS:
% Op_Pos:       List of all optimum parameters which
%               give the maximum efficiency.
%-------------------------------------------------------------

% ensure start has a min size of 2
if start<2
    start=2;
end

% pre-define loop varibales
A = zeros(size,size);
```

```
B = zeros(size,size);

% loop starting at 2 as this is the min no.
% of points required to calculate fin indicies.
for n = start:size
    for m = start:size
        [Result,roi] = Backtesting(n,m,file,0);
        %0 used to supress output plot
        A(n,m) = Result;
        B(n,m) = roi;
    end
end

% Locate the position of the maximum evaluator outputs.
[row,col] = find(A==max(A,[],'all'));
Op_Pos = [row,col];
disp("Max Evaluator Value = " + A(row(end),col(end)));

% Display mesh of optimised parameters and returns.
figure(20), mesh(A,'edgecolor','k');
figure(21), mesh(B);
end
```

Appendix A.6. Returns

```
function ROI = Returns(Pos_idx,data)
%-------------------------------------------------------
% Function to calucalte the Return On Investment (ROI) of
% the system under set parameters.
% INPUTS:
% Pos_idx - Array of the indicators to buy or sell
% data -    Array of the financial time series (unfiltered)

% OUTPUTS:
% ROI -          Output of the return on investment
%-------------------------------------------------------



data=transpose(data);
k=1;

% if there is a buy/sell indication, store the indicator,
%its position and price
for i=1:length(Pos_idx)
    if Pos_idx(i) == 1 || Pos_idx(i) == -1
        tick(k,:)=[i,Pos_idx(i),data(i)];
        k=k+1;
    end
end

if exist('tick','var')
```

```
    % pre define price change variable
    price_change=zeros(1,length(tick));

    % initial investment is the first buy price
    initial_investment = tick(1,3);

    % for all elements of the buy/sell array,
    % calculate price difference of long & short positions
    for i=1:size(tick,1)-1
        if tick(i,2)==1
            price_change(i)=tick(i+1,3)-tick(i,3);
        elseif tick(i,2)==-1
            price_change(i)=tick(i,3)-tick(i+1,3);
        end
    end

    % If the final indicator is a buy, calculate
    % change to final data point.
    if tick(end,2) == 1
        price_change(end)=(data(end)-tick(end,3));
    elseif tick(end,2)==-1
        price_change(end)=(tick(end,3)-data(end));
    end

    % calculate the overall change
    Return_captial=sum(price_change);

    % Calcualte the percentage change from the
    % initial investment in FIAT currency.
    ROI=((Return_captial+initial_investment)...
        /initial_investment)-1;
else
    ROI=0;
end
end
```

Appendix A.7. Evaluator

```
function Result = Evaluator(ZC,G,M,T)
%-------------------------------------------------------------
% FUNCTION: Evaluates accuracy of a short time trend analysis
% indicator in terms of actual price differences.
% INPUTS:
% ZC - Array composed of zeros crossing point indicators.
% G - Array composed of the time series at zero crossings.
% M - Length of Time Series analysed
% T - Period (moving window size used for data analysis).

% OUTPUT:
% Result: combined accuracy of the sustem.
%-------------------------------------------------------------
```

```
% Initiate counters.
n=1; N=1;

for m = T+1:M-T-1
    if G(m)>0.0
        P(n) = G(m);
        n = n+1;
        Q(N) = ZC(m);
        N = N+1;
    end
end

% Initiate counters
up_good=0; up_bad=0;

% Count the number of times that an indication of an
% upward trend led to a price increase (up_good) and
%the number of times it didn't (up_bad).
for n=1:N-2
    if  Q(n)>0 && P(n+1) - P(n)>0
        up_good=up_good+1;
    end
    if  Q(n)>0 && P(n+1) - P(n)<0
        up_bad=up_bad+1;
    end
end

%Initiate counters
down_good=0; down_bad=0;

% Count the number of times that an indication of a
% downward trend led to a price decrease (down_good)
%and the number of times it didn't (down_bad).
for n=1:N-2
    if  Q(n)<0 && P(n+1)-P(n)<0
        down_good=down_good+1;
    end
    if  Q(n)<0 && P(n+1)-P(n)>0
        down_bad=down_bad+1;
    end
end

% Provide outputs on the percentatge accuracy of
% - Successfully upward trend 'Entries_Accuracy'
% - Successfully downward trend 'Exits_Accuracy'
% - The combined accuracy of both success rates
%   (the 'Combined_Accuracy').
if (double(up_good) + double(up_bad)) > 0
    Entries_Accuracy = 100 * double(up_good)...
```

```
        / (double(up_good) + double(up_bad));
else
    Entries_Accuracy = 0.0;
end

if (double(down_good) + double(down_bad)) > 0
    Exits_Accuracy = 100 * double(down_good)...
        / (double(down_good) + double(down_bad));
else
    Exits_Accuracy = 0;
end

Combined_Accuracy = ...
    (Exits_Accuracy+Entries_Accuracy) / 2;
Result = Combined_Accuracy;
end
```

Appendix A.8. Levy Index

```
function output = LevyIndex(data,N)
%-------------------------------------------------------------
% Function: Computation of the Levy Index (gamma) using the
% least squares algorithm for 'data' of length N.
% INPUTS:
% data -     Array of input data
% N -        Length of data
%
% OUTPUT:
% output -  Levy index
%-------------------------------------------------------------

%Compute the Amplitude Spectrum
data = (abs(fft(data))).^2;

%take log of data
xdata=log(2:1:round(N/2));
ydata=transpose(log(data(2:round(N/2))));

%Compute each term of the least squares formula associated
%with log scaling law alpha*log(data).
term1=sum(ydata).*sum(xdata);
term2=sum(ydata.*xdata);
term3=sum(xdata)^2;
term4=sum(xdata.^2);
N=round(N/2);    % define N for half-space

%Compute alpha
alpha=(term1-(N*term2))/(term3-(N*term4));

%Compute Levy Index
```

```
output=-alpha;
end
```

Appendix A.9. Read File

```
function [output_price,output_timestamp] = ...
    readfile(input_file, time_step, time_length, end_date)
%-------------------------------------------------------------
% Function for importing the data and creating the
% finanaical time series.

% INPUTS:
% input_file:   contains the historical arcive data
% time_step:    contains the no. of hours between each
%               data point (1 day = 24)
% time_length:  total time of the series in days
% end_date:     end date of the series to be created in form
%               - 2022-02-12 00:00:00

% OUTPUTS:
% output_price: financial time series for use in system
%-------------------------------------------------------------

% import the file
price_table = readtable(input_file,'NumHeaderLines',2);

% extract the prices and time stamp from the table of data
time_stmp = table2array(price_table(:,2));
prices = table2array(price_table(:,4));

% how many data points required
no_points = (time_length*24)+1;
total_output_points=round(no_points/time_step);

% initiate output variable
output_price=zeros(total_output_points,1);
output_timestamp=string(zeros(total_output_points,1));


% find index of the first data point
idx=find(contains(string(time_stmp),end_date));
output_price(1)=prices(idx);

k=2;
for i=(idx+time_step):time_step:(no_points+idx-time_step)
    output_price(k)=prices(i);
    output_timestamp(k)=time_stmp(i);
    k=k+1;
end

output_price=flip(output_price);
```

```
output_timestamp=flip(output_timestamp);
```

References

1. En.wikipedia.org. Bitcoin—Wikipedia. 2022. Available online: https://en.wikipedia.org/wiki/Bitcoin (accessed on 25 April 2022).
2. Coinmarketcap.com. Bitcoin—Market Cap. 2022. Available online: https://coinmarketcap.com/currencies/bitcoin/ (accessed on 25 April 2022).
3. Farooqui, J.B. Will Bitcoin Hit USD 100,000 or Go Even Higher in 2022? 2021. Available online: https://www.proactiveinvestors.co.uk/companies/news/969323/will-bitcoin-hit-us-100-000-or-go-even-higher-in-2022-969323.html (accessed on 13 April 2022).
4. Portdex. Decentralised Digital Economy Platform. 2022. Available online: https://portdex.com/ (accessed on 23 March 2022).
5. Euklidiadas, M.M. Living with Bitcoin as a Currency: The Case of el Salvador. 2021. Available online: https://tomorrow.city/a/el-salvador-bitcoin-legal-tender/ (accessed on 21 April 2022).
6. Treasury, H. Government Sets out Plan to Make UK a Global Cryptoasset Technology Hub. 2022. Available online: https://www.gov.uk/government/news/government-sets-out-plan-to-make-uk-a-global-cryptoasset-technology-hub (accessed on 2 May 2022).
7. Bambrough, B. Leak Reveals Biden's Crypto Plans—Sending the Price of Bitcoin, Ethereum, BNB, Solana, Cardano, XRP, Terra's Luna and Avalanche Higher. 2022. Available online: https://www.forbes.com/sites/billybambrough/2022/03/09/leak-reveals-bidens-crypto-plans--sending-the-price-of-bitcoin-ethereum-bnb-solana-cardano-xrp-terras-luna-and-avalanche-higher/?sh=46e0131a30f9 (accessed on 10 March 2022).
8. Puterbaugh, J.; Haar, R. Bitcoin Dips Below USD40,000 Again. Here's How Investors Should React. 2022. Available online: https://time.com/nextadvisor/investing/cryptocurrency/bitcoin-crash-continues/ (accessed on 30 April 2022).
9. Mandelbrot, B.B. The Variation of Some Other Speculative Prices. *J. Bus.* **1967**, *40*, 393–413. [CrossRef]
10. Fama, E.F. The Behavior of Stock—Market Prices. *J. Bus.* **1965**, *38*, 34–105. [CrossRef]
11. Bachelier, L. Théorie de la spéculation. *Ann. Sci. L'École Norm. Supérieure* **1900**, *3*, 21–86. Available online: http://archive.numdam.org/article/ASENS_1900_3_17__21_0.pdf (accessed on 14 March 2022). [CrossRef]
12. Einstein, A. The Motion of Small Particles Suspended in Liquids at Rest Required by the Molecular-Kinetic Theory of Heat. *Ann. Der Phys.* **1905**, *17*, 549–560. [CrossRef]
13. Fama, E. Efficient Capital Markets: A Review of Theory and Empirical Work. *J. Financ.* **1970**, *25*, 383–417. [CrossRef]
14. Grossman, S.J.; Stiglitz, J.E. On the Impossibility of Informationally Efficient Markets. *Am. Econ. Rev.* **1980**, *70*, 393–408.
15. Malkiel, B.G. The Efficient Market Hypothesis and its Critics. *J. Econ. Perspect.* **2003**, *17*, 59–82. [CrossRef]
16. Guardian, T. The Mathematical Equation That Caused the Banks to Crash. 2012. Available online: https://www.theguardian.com/science/2012/feb/12/black-scholes-equation-credit-crunch (accessed on 13 April 2022).
17. Scott, F. What Monte Carlo Methods Cannot Do. *Hum. Ecol. Risk Assess. Int. J.* **1996**, *2*, 990–1007. [CrossRef]
18. Chu, J.; Nadarajah, S.; Chan, S. Statistical Analysis of the Exchange Rate of Bitcoin. *PLoS ONE* **2015**, *10*, e0133678. [CrossRef]
19. Lampiere, M.; Blackledge, J. Kearney, D. Carbon Futures Trading and Short-Term Price Prediction: An Analysis Using theFractal Market Hypothesis and Evolutionary Computing. *Mathematics* **2021**, *9*, 1005. [CrossRef]
20. Cont, R.; Kotlicka, M.; Valdivia, I. Financial Return Distributions: Past, Present, and COVID-19. *Entropy* **2021**, *23*, 884. [CrossRef]
21. Drożdż, S.; Kwapień, J.; Wątorek, M. What Is Mature and What Is Still Emerging in the Cryptocurrency Market? *Entropy* **2023**, *25*, 772. [CrossRef]
22. Urquhart, A. The inefficiency of Bitcoin. *Econ. Lett.* **2016**, *148*, 80–82. [CrossRef]
23. Nadarajah, S.; Chu, J. On the inefficiency of Bitcoin. *Econ. Lett.* **2017**, *150*, 6–9. [CrossRef]
24. Bariviera, A.F. The inefficiency of Bitcoin revisited: A dynamic approach. *Econ. Lett.* **2017**, *161*, 1–4. [CrossRef]
25. Bariviera, A.F.; Basgall, M.J.; Hasperué, W.; Naiouf, M. Some stylized facts of the Bitcoin market. *Phys. A Stat. Mech. Its Appl.* **2017**, *484*, 82–90. [CrossRef]
26. Lahmiri, S.; Bekiros, S.; Salvi, A. Long-range memory, distributional variation and randomness of bitcoin volatility. *Chaos Solitons Fractals* **2018**, *107*, 43–48. [CrossRef]
27. Al-Yahyaee, K.H.; Mensi, W.; Yoon, S.M. Efficiency, multifractality, and the long-memory property of the Bitcoin market: A comparative analysis with stock, currency, and gold markets. *Financ. Res. Lett.* **2018**, *27*, 228–234. [CrossRef]
28. Alvarez-Ramirez, J.; Rodriguez, E.; Ibarra-Valdez, C. Long-range correlations and asymmetry in the Bitcoin market. *Phys. A Stat. Mech. Its Appl.* **2018**, *492*, 948–955. [CrossRef]
29. Jiang, Y.; Nie, H.; Ruan, W. Time-varying long-term memory in Bitcoin market. *Financ. Res. Lett.* **2018**, *25*, 280–284. [CrossRef]
30. Zhang, W.; Wang, P.; Li, X.; Shen, D. The inefficiency of cryptocurrency and its cross-correlation with Dow Jones Industrial Average. *Phys. A Stat. Mech. Its Appl.* **2018**, *510*, 658–670. [CrossRef]

31. Caporale, G.M.; Gil-Alana, L.; Plastun, A. Persistence in the cryptocurrency market. *Res. Int. Bus. Financ.* **2018**, *46*, 141–148. [CrossRef]
32. Celeste, V.; Corbet, S.; Gurdgiev, C. Fractal dynamics and wavelet analysis: Deep volatility and return properties of Bitcoin, Ethereum and Ripple. *Q. Rev. Econ. Financ.* **2020**, *76*, 310–324. [CrossRef]
33. Hu, Y.; Valera, H.G.A.; Oxley, L. Market efficiency of the top market-cap cryptocurrencies: Further evidence from a panel framework. *Financ. Res. Lett.* **2019**, *31*, 138–145. [CrossRef]
34. Chu, J.; Zhang, Y.; Chan, S. The adaptive market hypothesis in the high frequency cryptocurrency market. *Int. Rev. Financ. Anal.* **2019**, *64*, 221–231. [CrossRef]
35. Al-Yahyaee, K.H.; Mensi, W.; Ko, H.U.; Yoon, S.M.; Kang, S.H. Why cryptocurrency markets are inefficient: The impact of liquidity and volatility. *N. Am. J. Econ. Financ.* **2020**, *52*, 101168. [CrossRef]
36. Kakinaka, S.; Umeno, K. Cryptocurrency market efficiency in short- and long-term horizons during COVID-19: An asymmetric multifractal analysis approach. *Financ. Res. Lett.* **2021**, *46*, 102319. [CrossRef]
37. David, S.; Inacio Jr., C.; Nunes, R.; Machado, J. Fractional and fractal processes applied to cryptocurrencies price series. *J. Adv. Res.* **2021**, *32*, 85–98. [CrossRef]
38. Drozdz, S.; Gebarowski, R.; Minati, L.; Oswiecimka, P.; Watorek, M. Bitcoin market route to maturity? Evidence from return fluctuations, temporal correlations and multiscaling effects. *Chaos Interdiscip. J. Nonlinear Sci.* **2018**, *28*, 071101. [CrossRef]
39. Watorek, M.; Drożdż, S.; Kwapień, J.; Minati, L.; Oświęcimka, P.; Stanuszek, M. Multiscale characteristics of the emerging global cryptocurrency market. *Phys. Rep.* **2021**, *901*, 1–82. [CrossRef]
40. Elliott, J. Coinbase vs. Coinbase Pro. 2021. Available online: https://www.investopedia.com/coinbase-vs-coinbase-pro-5120704 (accessed on 25 April 2022).
41. Crypto Fees. 2022. Available online: https://www.etoro.com/trading/fees/#crypto (accessed on 25 April 2022).
42. Mitchell, W.C. *The Making and Using of Index Numbers*. Bulletin No. 656; Bureau of Labour Statistics, United States Department of Labour, 1938. 2023. Available online: https://fraser.stlouisfed.org/files/docs/publications/bls/bls_0656_1938.pdf (accessed on 5 April 2025).
43. Mandelbrot, B.B. Forecasts of future prices, unbiased markets, and martingale models. *J. Bus.* **1966**, *39*, 242–255. [CrossRef]
44. Peters, E.E. *Fractal Market Analysis: Applying Chaos Theory to Investment and Economics*; Wiley: Hoboken, NJ, USA, 1994; ISBN 0471585246.
45. Britannica, E. Fractal Mathematics. 2022. Available online: https://www.britannica.com/science/fractal (accessed on 10 March 2022).
46. Mandelbrot, B. *The Fractal Geometry of Nature*; Einaudi Paperbacks; W. H. Freeman: New York, NY, USA, 1983.
47. Soltanifar, M. A Generalization of the Hausdorff Dimension Theorem for Deterministic Fractals. *Mathematics* **2021**, *9*, 1546. [CrossRef]
48. Williams, B. *Trading Chaos: Applying Expert Techniques to Maximise Your Profits*; Wiley: Hoboken, NJ, USA, 1995. Available online: https://c.mql5.com/3/133/Bill.Williams.Trading.Chaos.Applying.Expert.Techniques.To.Maximize.Your.Profits.pdf (accessed on 17 August 2025).
49. Elliott, R. *The Wave Principle*; Igal Meirovich: Dubai, United Arab Emirates, 1938; ISBN 1607964961.
50. Lévy, P. *Plane or Space Curves and Surfaces Consisting of Parts Similar to the Whole*; Addison-Wesley Publishing: Hoboken, NJ, USA, 1938; ISBN 0-201-58701-7.
51. Lévy, P. *Théorie de l'addition des Variables aléatoires*, Gauthier-Villars, Paris, 1937. Available online: https://www.cambridge.org/core/journals/mathematical-gazette/article/abs/theorie-de-laddition-de-variables-aleatoires-by-paul-levy-pp-xx-385-second-edition-1954-1200f-gauthiervillars-paris/73294A1332D0CC9727CF681F859605D1?utm_source=chatgpt.com (accessed on 13 August 2025).
52. Mandelbrot, B.B. *Gaussian, Self-Affinity and Fractals*; Springer Publishing: Berlin/Heidelberg, Germany, 2002; ISBN 978-0-387-98993-8.
53. Mandelbrot, B.B. How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science* **1967**, *156*, 636–638. [CrossRef] [PubMed]
54. Greis, N.P.; Greenside, H.S. Implication of a power-law power-spectrum for self-affinity. *Phys. Rev. A* **1991**, *44*, 2324–2334. [CrossRef] [PubMed]
55. Mandelbrot, B.B. Self-Affine Fractals and Fractal Dimension. *Phys. Scr.* **1985**, *32*, 257–260. [CrossRef]
56. Belomestny, D. Spectral Estimation of the Fractional Order of a Lévy Process. *Ann. Stat.* **2010**, *38*, 317–351. [CrossRef]
57. Blackledge, J.; Lamphiere, M. A Review of the Fractal Market Hypothesis for Trading and Market Price Prediction. *Mathematics* **2022**, *10*, 117. [CrossRef]
58. Fourier Transform; Wikipedia, The Free Encyclopedia, 2025. Available online: https://en.wikipedia.org/wiki/Fourier_transform (accessed on 25 September 2025).
59. Baker, G.L.; Gollub, J.P. *Chaotic Dynamics: An Introduction*; Cambridge University Press: Cambridge, UK, 1996.

60. Hutchinson, J. Fractals and Self-Similarity. *Indiana Univ. Math. J.* **1981**, *30*, 713–747. [CrossRef]
61. Brown, R.; Bryant, P.; Abarbanel, H. Computing the Lyapunov Spectrum of a Dynamical System From an Observed Time Series. *Phys. Rev. A* **1991**, *43*, 2787. [CrossRef]
62. Bryant, P.; Brown, R.; Abarbanel, H. Lyapunov Exponents From Observed Time Series. *Phys. Rev. Lett.* **1990**, *65*, 1523. [CrossRef]
63. Blackledge, J.M. *Digital Signal Processing*, 2nd ed.; Horwood Publishing: Chichester, UK, 2006; ISBN 1-904275-26-5. Available online: https://arrow.tudublin.ie/engschelebk/4/ (accessed on 12 July 2025).
64. Blackledge, J.; Kearney, D.; Lamphiere, M.; Rani, R.; Walsh, P. Econophysics and Fractional Calculus: Einstein's Evolution Equation, the Fractal Market Hypothesis, Trend Analysis and Future Price Prediction. *Mathematics* **2019**, *7*, 1057. [CrossRef]
65. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimisation over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
66. Dorigo, M.; Gambardella, L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. Evol. Comput.* **2002**, *1*, 53–66. [CrossRef]
67. TuringBot. Turing Bot Homepage. 2022. Available online: https://turingbotsoftware.com/ (accessed on 11 March 2022).
68. TuringBot. Turing Bot Documentation. 2022. Available online: https://turingbotsoftware.com/documentation.html (accessed on 11 March 2022).
69. En.wikipedia.org. Simulated Annealing—Wikipedia. 2022. Available online: https://en.wikipedia.org/wiki/Simulatedannealing (accessed on 25 April 2022).
70. Krakovská, H.; Krakovská, A. Fractal Dimension of Self-Affine Signals: Four Methods of Estimation. *arXiv* **2016**, arXiv:1611.06190. [CrossRef]
71. Higuchi, T. Approach to an irregular time series on the basis of the fractal theory. *Phys. D Nonlinear Phenom.* **1988**, *31*, 277–283. [CrossRef]
72. Hurst, H.E. Long-term storage capacity of reservoirs. *Trans. Am. Soc. Civ. Eng.* **1951**, *116*, 770–808. [CrossRef]
73. Python. Gplearn: Genetic Programming in Python, with a Scikit-Learn Inspired and Compatible API. Available online: https://gplearn.readthedocs.io/en/stable/ (accessed on 6 July 2025).
74. Liu, X.; Wang, W. Deep Time Series Forecasting Models: A Comprehensive Survey. *Mathematics* **2024**, *12*, 1504. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.