Ehlers Loops: Theoretical and Mathematical Explanation

Ehlers Loops are a technical analysis tool designed to capture cyclic behaviors in financial markets using principles from digital signal processing (DSP). John Ehlers, a pioneer in this field, introduced methods to analyze market data mathematically, focusing on extracting useful information from noisy price movements. The term "loops" arises from the visualization of price dynamics in phase space, where cyclic or trending behavior appears as continuous trajectories.

1. Theoretical Foundation

Ehlers Loops leverage the idea that financial markets are not purely random but exhibit cycles and trends driven by recurring economic, psychological, and structural factors. By transforming price data into a mathematical representation in phase space, analysts can better understand these dynamics.

The key goals of Ehlers Loops are to distinguish between trending and cyclical market behavior, detect momentum shifts and potential turning points, and reduce noise in the data for clearer analysis.

Ehlers builds on techniques such as the Hilbert Transform, which allows the decomposition of market data into orthogonal components. These components can reveal the underlying cyclical nature of price movements.

2. Mathematical Basis

a. Price Oscillation Model
Market prices are treated as oscillatory signals that can be expressed mathematically as:

$$P(t) = A\ \sin(2\pi f t + \phi) + \epsilon(t)$$

where $P(t)$ is the price at time t, A represents market volatility (amplitude), f is the frequency of the cycle (cycles per unit of time), ϕ is the phase shift indicating the position within the cycle, and $\epsilon(t)$ accounts for random noise or non-cyclic components.

b. Hilbert Transform
The Hilbert Transform computes the analytic signal, decomposing it into:

$$z(t) = x(t) + i\, y(t)$$

where $x(t) = P(t)$ is the original price series (in-phase component), and $y(t) = \mathrm{H}[P(t)]$ is the Hilbert Transform of $P(t)$(quadrature component).

The Hilbert Transform produces the quadrature component $y(t)$ by shifting the phase of the price series by 90 degrees, creating a signal orthogonal to the original.

c. Instantaneous Phase and Frequency
Using the analytic signal, the instantaneous phase $\phi(t)$ and frequency $f(t)$ can be derived:

$$\phi(t) = \tan^{-1}\frac{y(t)}{x(t)}$$

$$f(t) = \frac{1}{2\pi}\frac{d\phi(t)}{dt}$$

The instantaneous phase ($\phi(t)$) indicates the position within the cycle at time t, and the instantaneous frequency ($f(t)$) measures the rate of change of the phase, indicating how quickly the cycle progresses.

d. Phase Space Representation

By plotting $x(t)$ (in-phase) against $y(t)$ (quadrature), we generate a trajectory in phase space. These trajectories form loops, which are interpreted as follows: circular or elliptical loops indicate cyclic behavior with consistent frequency and amplitude, expanding loops suggest increasing volatility or momentum, and flattening loops suggest declining momentum or a potential reversal.

3. Signal Processing Techniques

To refine the analysis and reduce noise, several preprocessing steps are typically applied. Smoothing filters, such as moving averages or low-pass filters, reduce high-frequency noise. Detrending removes long-term trends to focus on cyclic behavior. Normalization ensures consistent amplitude for better comparison across time periods.

4. Interpretation of Ehlers Loops

The loops provide insights into market behavior. A trending market is indicated when the trajectory moves consistently in one direction with little looping. A cyclic market is indicated when the trajectory forms well-defined loops. The size of the loops reflects changes in market volatility. Sharp changes in the direction of the trajectory signal momentum shifts or potential reversals.

5. Advanced Concepts

Lag-free indicators minimize lag, providing more timely insights. Cycle period identification analyzes the phase and frequency components to identify dominant cycle periods. Adaptive filtering dynamically adjusts filters based on the detected cycle period, ensuring that the analysis remains relevant in varying market conditions.

6. Visualization and Analysis

To visualize Ehlers Loops, a phase-space plot is generated, where the x-axis represents the in-phase component $x(t)$, and the y-axis represents the quadrature component $y(t)$. Directional arrows can be added to indicate the temporal progression, helping analysts understand the flow of market behavior.

7. Practical Applications

Ehlers Loops are used for trend identification to detect when a market is trending versus oscillating, volatility analysis to measure changes in market dynamics, reversal detection to spot turning points and shifts in momentum, and market timing to use the phase information to time entries and exits.

8. Limitations

While powerful, Ehlers Loops are not without limitations. High noise levels in price data can distort loops, requiring careful preprocessing. Understanding and implementing the mathematical concepts may be challenging for some users. The method assumes markets exhibit cyclic behavior, which may not always hold true.

9. Conclusion

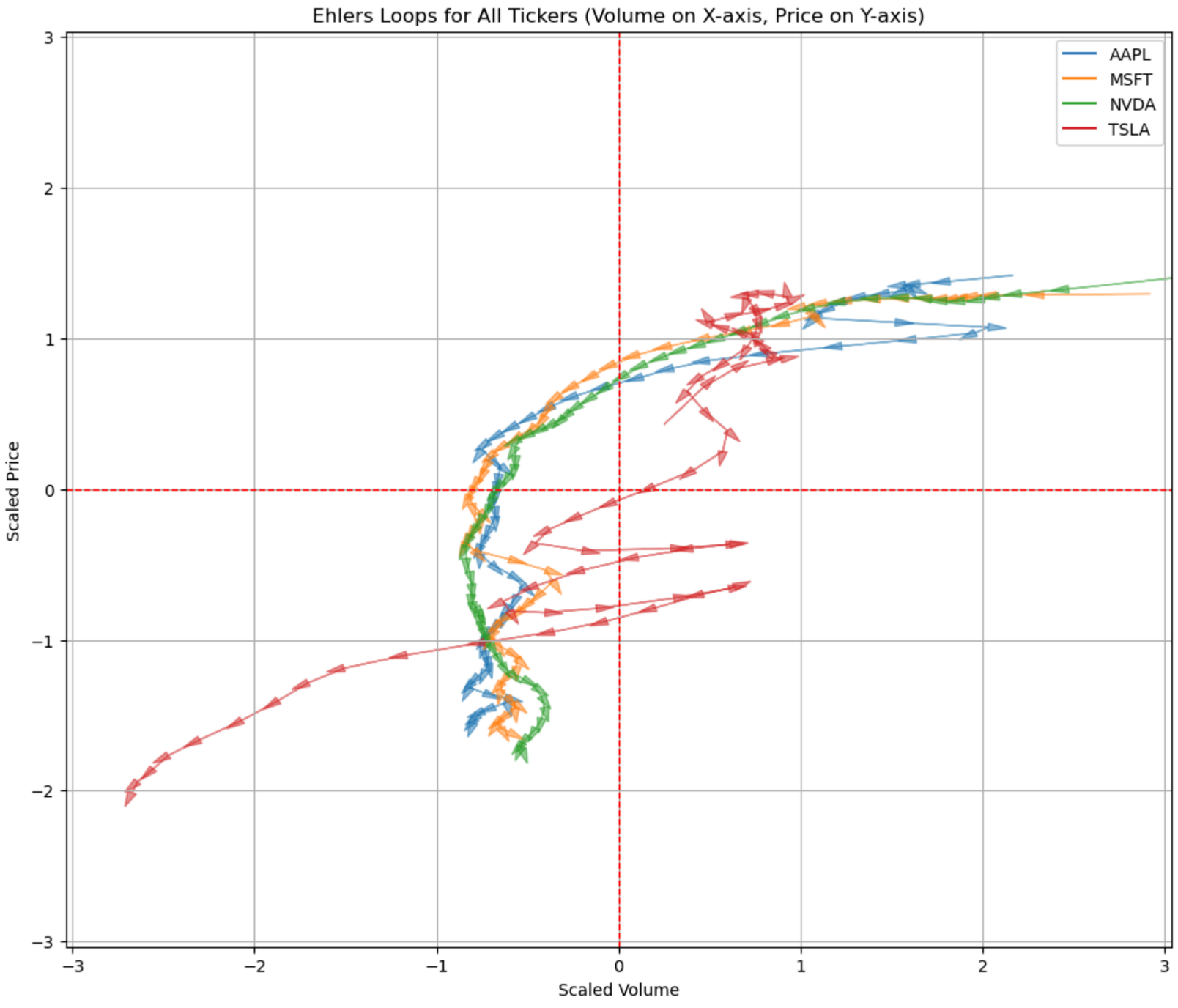
Ehlers Loops provide a sophisticated framework for analyzing market behavior using advanced DSP techniques. By visualizing price data in phase space, they uncover hidden cyclic patterns, offering unique insights into trends, momentum, and potential reversals. Despite their complexity, Ehlers Loops are a valuable tool for traders and analysts seeking to enhance their understanding of market dynamics.

In [23]:

```
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt

# Example Usage
tickers = ['AAPL', 'MSFT', 'NVDA', 'TSLA']  # List of financial instruments
ehlers = EhlersLoops(tickers=tickers, start_date='2024-09-01', end_date='2024-12-06', lp_period=10, hp_period=125)
ehlers.fetch_data()
ehlers.process_data()
ehlers.plot_2d_loops()

class EhlersLoops:
    def __init__(self, tickers, start_date, end_date, lp_period=10, hp_period=125):
        self.tickers = tickers
        self.start_date = start_date
        self.end_date = end_date
        self.lp_period = lp_period
        self.hp_period = hp_period
        self.data = {}
        self.filtered_data = {}

    def fetch_data(self):
        """Fetch financial data for each ticker from Yahoo Finance."""
        for ticker in self.tickers:
            self.data[ticker] = yf.download(ticker, start=self.start_date, end=self.end_date)

    def roofing_filter(self, series, high_pass, low_pass):
        """Applies high-pass and low-pass filters to the data."""
        hp_alpha = np.exp(-np.sqrt(2) * np.pi / high_pass)
        high_passed = series.astype(float).copy()  # Ensure data type compatibility
        for i in range(2, len(series)):
            high_passed.iloc[i] = (
                (1 + hp_alpha * (1 - hp_alpha)) / 2
            ) * (series.iloc[i] - 2 * series.iloc[i - 1] + series.iloc[i - 2]) + 2 * hp_alpha * high_passed.iloc[i - 1] - (hp_alpha ** 2) * high_passed.iloc[i - 2]

        lp_alpha = np.exp(-np.sqrt(2) * np.pi / low_pass)
        low_passed = high_passed.copy()
        for i in range(2, len(high_passed)):
            low_passed.iloc[i] = (
                (1 - lp_alpha) ** 2
            ) * (high_passed.iloc[i] + high_passed.iloc[i - 1]) / 2 + 2 * lp_alpha * low_passed.iloc[i - 1] - (lp_alpha ** 2) * low_passed.iloc[i - 2]
        return low_passed

    def process_data(self):
        """Process data by applying filters and normalizing price and volume."""
        for ticker, df in self.data.items():
            df['Price'] = df['Close']
            df['Volume'] = df['Volume']

            # Apply roofing filter to price and volume
            df['Filtered_Price'] = self.roofing_filter(df['Price'], self.hp_period, self.lp_period)
            df['Filtered_Volume'] = self.roofing_filter(df['Volume'], self.hp_period, self.lp_period)

            # Normalize data
            df['Price_Scaled'] = (df['Filtered_Price'] - df['Filtered_Price'].mean()) / df['Filtered_Price'].std()
            df['Volume_Scaled'] = (df['Filtered_Volume'] - df['Filtered_Volume'].mean()) / df['Filtered_Volume'].std()
```

```
            # Store processed data
            self.filtered_data[ticker] = df.dropna()

    def plot_2d_loops(self):
        """Plot 2D Ehlers Loops for all tickers on a single graph."""
        plt.figure(figsize=(12, 10))

        # Calculate global limits for consistent scaling
        all_x = []
        all_y = []
        for ticker, df in self.filtered_data.items():
            all_x.extend(df['Volume_Scaled'].values)
            all_y.extend(df['Price_Scaled'].values)

        max_abs_x = max(abs(min(all_x)), abs(max(all_x)))
        max_abs_y = max(abs(min(all_y)), abs(max(all_y)))
        limit = max(max_abs_x, max_abs_y)

        for ticker, df in self.filtered_data.items():
            x = df['Volume_Scaled']  # Volume is now the X-axis
            y = df['Price_Scaled']   # Price is now the Y-axis

            # Plot arrows to indicate movement direction for each ticker
            for i in range(1, len(x)):
                plt.arrow(x.iloc[i - 1], y.iloc[i - 1], x.iloc[i] - x.iloc[i - 1], y.iloc[i] - y.iloc[i - 1],
                          color=plt.cm.tab10(self.tickers.index(ticker) % 10), alpha=0.5,
                          head_width=0.05, head_length=0.1, label=None)

            # Add label for the ticker
            plt.plot([], [], label=ticker, color=plt.cm.tab10(self.tickers.index(ticker) % 10))

        # Customize plot
        plt.axhline(0, color='red', linestyle='--', linewidth=1)  # Y-axis
        plt.axvline(0, color='red', linestyle='--', linewidth=1)  # X-axis
        plt.xlim(-limit, limit)
        plt.ylim(-limit, limit)
        plt.title("Ehlers Loops for All Tickers (Volume on X-axis, Price on Y-axis)")
        plt.xlabel("Scaled Volume")
        plt.ylabel("Scaled Price")
        plt.legend()
        plt.grid()
        plt.show()
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```


Ehlers Loops for All Tickers (Volume on X-axis, Price on Y-axis)
AAPL
MSFT
NVDA
TSLA
Scaled Price
Scaled Volume
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

In [ ]: