Empirical Limits of Model-Free Feedback Trading Strategies under Discrete-Time and Self-Financing Constraints

Nicola Brandolini

December 2025

Abstract

Model-free feedback trading strategies have attracted attention in control and finance due to their robustness properties and minimal modeling assumptions. In particular, Simultaneous Long-Short (SLS) strategies have theoretical guarantees under idealized continuous-time settings. This paper evaluates SLS strategies on real equity data in a discrete-time, self-financing framework and tests practical extensions (volatility scaling via GARCH, RSI filtering, and sector-level filtering). Results suggest that feedback-based strategies can reduce volatility and drawdowns but do not deliver consistent excess returns relative to Buy-and-Hold in this setting.

1 Introduction

Efficient market arguments imply that systematic excess returns are difficult to obtain without predictive structure (Fama, 1970). This motivates strategies that do not forecast returns directly. Model-free feedback trading, developed in the intersection of control and finance, provides one such framework (Barmish & Primbs, 2015). However, the translation from continuous-time theory to discrete-time empirical trading can reveal limitations (Lo, 2004). We provide an empirical assessment of SLS strategies and simple extensions.

2 Data

We use daily adjusted close prices for 25 large-cap equities (five sectors) from 2015-05-04 to 2017-05-04. Log-returns are computed as:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right).$$

3 Trading Framework

At time t, the portfolio holds q_t shares and cash c_t, with value:

$$V_t = q_t P_t + c_t.$$

Trading is discrete-time at daily closes. Short-selling is permitted, but the portfolio is constrained to remain self-financing and non-negative ($V_t \geq 0$). Trades that would violate feasibility are rescaled to the maximum admissible size.

4 Benchmarks

Buy-and-Hold invests all initial capital at $t = 0$ and holds the position.
Random trading samples target positions from a symmetric distribution and applies the same self-financing constraint; this is a sanity-check baseline.

5 SLS Strategy

The discrete-time SLS update is:

$$q_{t+1} = q_t + k\,(P_t - P_{t-1}),$$

with feedback gain $k > 0$ (Barmish & Primbs, 2015).

6 Volatility Scaling (GARCH)

Conditional volatility is estimated with a GARCH(1,1) model (Bollerslev, 1986):

$$\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2.$$

The gain is scaled as:

$$k_t = \frac{k}{\sigma_t},$$

motivated by volatility-managed portfolio ideas (Moreira & Muir, 2017).

7 RSI Filtering

We compute RSI using Wilder's smoothing (Wilder, 1978) and use it as a modulation mechanism (risk gating), not as a return predictor. Momentum effects are well documented in the literature (Jegadeesh & Titman, 1993).

8 Sector-Level Filtering

To reduce idiosyncratic noise, we combine each asset's return with the average return of its sector peers, consistent with documented co-movement in equity returns (Campbell et al., 1997).

9 Results

Across assets, SLS and extensions typically reduce volatility and drawdowns relative to Buy-and-Hold, but do not yield consistent positive excess returns in this discrete-time self-financing setting. Volatility scaling and filters improve stability but do not fundamentally change average excess performance.

10 Discussion

The findings highlight a gap between continuous-time theoretical guarantees and discrete-time empirical trading behavior. Feedback rules regulate exposure effectively but do not create predictive edge. Extensions based on volatility targeting and filters can improve risk characteristics without reliably generating excess return.

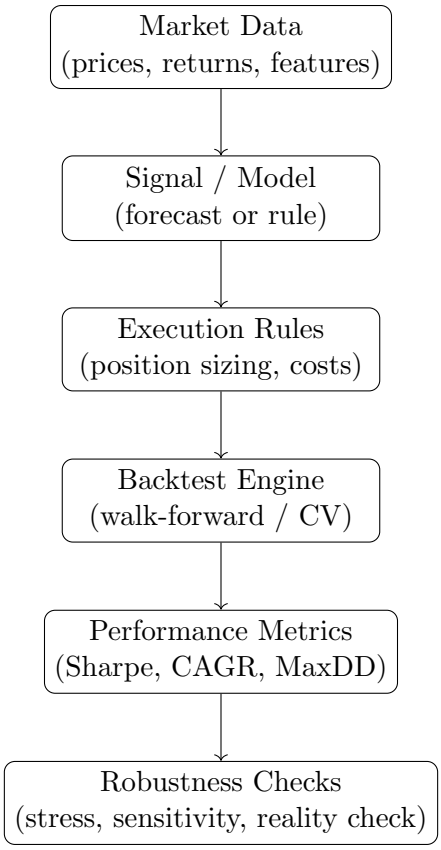

Figure 1: Experimental pipeline for strategy development and evaluation in algorithmic trading.

11 Conclusion

Under the tested assumptions and sample, model-free feedback trading is more effective as a risk-control mechanism than as a consistent alpha generator. Future work may explore hybrid schemes combining feedback control with predictive signals or different execution frequencies.

References

Barmish, B. R., & Primbs, J. A. (2015). On the control of stock trading using feedback. *IEEE Transactions on Automatic Control*, *60*(9), 2403–2410.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Campbell, J. Y., Lo, A. W., & MacKinlay, A. C. (1997). The econometrics of financial markets. *Princeton University Press*.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383–417.

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, *48*(1), 65–91.

Lo, A. W. (2004). The adaptive markets hypothesis. *The Journal of Portfolio Management*, *30*(5), 15–29.

Moreira, A., & Muir, T. (2017). Volatility-managed portfolios. *The Journal of Finance*, *72*(4), 1611–1644.

Wilder, J. W. (1978). *New concepts in technical trading systems*. Trend Research.

A Algorithmic Implementation Details

This appendix provides pseudo-code descriptions of the trading algorithms analyzed in the paper. The goal is to ensure full reproducibility of the experimental setup independently of any specific programming language.

B Self-Financing Trading Engine

All strategies rely on a common self-financing execution engine.

Algorithm A1: Self-Financing Execution

Input: Current price P_t, desired position $\tilde{q}_{t+1}$, current position q_t, current cash c_t
Output: Updated position q_{t+1}, updated cash c_{t+1}

1. Compute proposed trade:
$$\Delta q = \tilde{q}_{t+1} - q_t$$
2. Compute required cash:
$$\Delta c = -\Delta q \cdot P_t$$
3. If $c_t + \Delta c \geq 0$:
   - Execute full trade: $q_{t+1} = \tilde{q}_{t+1}$, $c_{t+1} = c_t + \Delta c$
4. Else:
   - Rescale Δq so that $c_{t+1} = 0$
   - Set $q_{t+1} = q_t + \Delta q_{\max}$

This mechanism ensures that the portfolio value remains non-negative at all times.

C Buy-and-Hold Strategy

Algorithm A2: Buy-and-Hold

Input: Initial capital V_0, initial price P_0

1. Set $q_0 = V_0/P_0$
2. Set $c_0 = 0$
3. Hold $q_t = q_0$ for all t

D Random Trading Strategy

Algorithm A3: Random Trading

Input: Distribution $\mathcal{D}$ over target positions

At each time step t:

1. Sample target position:
$$\tilde{q}_{t+1} \sim \mathcal{D}$$
2. Apply Algorithm A1 to enforce self-financing

E Simultaneous Long-Short (SLS) Strategy

Algorithm A4: Discrete-Time SLS

Input: Feedback gain k

At each time step t:

1. Observe price change:
$$\Delta P_t = P_t - P_{t-1}$$
2. Compute target position:
$$\tilde{q}_{t+1} = q_t + k \cdot \Delta P_t$$
3. Apply Algorithm A1

F Volatility-Scaled SLS

Algorithm A5: GARCH-Scaled SLS

Input: Base gain k, GARCH volatility estimate σ_t

At each time step t:

1. Compute scaled gain:
$$k_t = \frac{k}{\sigma_t}$$
2. Compute target position:
$$\tilde{q}_{t+1} = q_t + k_t \cdot \Delta P_t$$
3. Apply Algorithm A1

G RSI Filtering

Algorithm A6: RSI Computation (Wilder Smoothing)

Input: Period n, price series $\{P_t\}$

1. Initialize average gain and loss over first n observations
2. For $t > n$:

$$\text{avg\_gain}_t = \frac{(n-1)\,\text{avg\_gain}_{t-1} + \max(P_t - P_{t-1}, 0)}{n}$$

$$\text{avg\_loss}_t = \frac{(n-1)\,\text{avg\_loss}_{t-1} + \max(P_{t-1} - P_t, 0)}{n}$$

3. Compute:

$$RS_t = \frac{\text{avg\_gain}_t}{\text{avg\_loss}_t}, \quad RSI_t = 100 - \frac{100}{1 + RS_t}$$

Algorithm A7: RSI-Filtered SLS

At each time step t:

1. Compute RSI_t
2. Define modulation factor $m_t \in [0, 1]$ based on RSI thresholds
3. Update position:

$$\tilde{q}_{t+1} = q_t + m_t \cdot k \cdot \Delta P_t$$

4. Apply Algorithm A1

H Sector-Level Complementary Filter

Algorithm A8: Sector-Filtered Signal

Input: Asset return $r_t^{(i)}$, sector peer returns $\{r_t^{(j)}\}_{j \neq i}$

1. Compute sector average:

$$\bar{r}_t^{(-i)} = \frac{1}{N-1} \sum_{j \neq i} r_t^{(j)}$$

2. Construct filtered signal:

$$\hat{r}_t^{(i)} = \lambda r_t^{(i)} + (1 - \lambda)\bar{r}_t^{(-i)}$$

3. Use $\hat{r}_t^{(i)}$ in place of ΔP_t in Algorithm A4

I Summary of Algorithmic Structure

All strategies can be viewed as instances of the following generic update:

$$\tilde{q}_{t+1} = q_t + g_t \cdot s_t,$$

where s_t is a signal derived from observed prices and g_t is a possibly time-varying gain. The self-financing constraint (Algorithm A1) enforces feasibility and ensures comparability across strategies.

J Statistical Evaluation and Robustness

This appendix describes the statistical metrics and aggregation procedures used to evaluate strategy performance.

J.1 Return and Risk Metrics

For each asset and strategy, the following quantities are computed over the full sample horizon:

- **Cumulative return**:
$$R = \frac{V_T - V_0}{V_0}$$
- **Annualized volatility**:
$$\sigma_{\text{ann}} = \sqrt{252}\,\text{std}(r_t)$$
- **Maximum drawdown (MDD)**:
$$\text{MDD} = \max_{t \le s} \left(\frac{V_s - V_t}{V_s} \right)$$
- **Sharpe ratio (uncorrected)**:
$$\text{Sharpe} = \frac{\mathbb{E}[r_t]}{\text{std}(r_t)} \sqrt{252}$$

No risk-free rate adjustment is applied, as the analysis focuses on relative comparisons across strategies.

J.2 Excess Performance

Performance is primarily evaluated through excess return relative to the Buy-and-Hold benchmark:

$$\alpha = R_{\text{strategy}} - R_{\text{BH}}.$$

This quantity measures incremental performance attributable to active trading rather than market exposure.

J.3 Cross-Sectional Aggregation

Results are aggregated across the cross-section of assets using simple averages:

$$\bar{\alpha} = \frac{1}{N} \sum_{i=1}^{N} \alpha^{(i)}.$$

The distribution of $\alpha^{(i)}$ across assets is also analyzed to assess heterogeneity and robustness.

J.4 Bootstrap Confidence Intervals

To assess statistical uncertainty without relying on parametric assumptions, block bootstrap resampling is employed. Daily returns are resampled in contiguous blocks of fixed length L to preserve serial dependence.

For each bootstrap sample $b = 1, \ldots, B$:

1. A resampled return series is constructed.
2. Strategy performance metrics are recomputed.
3. The empirical distribution of $\alpha^{(b)}$ is obtained.

Confidence intervals are computed as percentile intervals of the bootstrap distribution. This procedure is used exclusively for robustness assessment and not for formal hypothesis testing.

J.5 Multiple Testing Considerations

Given the evaluation of multiple assets and parameter configurations, no formal hypothesis testing is conducted at the individual asset level. Reported results emphasize consistency of qualitative behavior rather than statistical significance of isolated outcomes.

This conservative approach reduces the risk of spurious findings due to multiple comparisons.

K Replication Statement

This study is fully reproducible using publicly available data and open-source software.

K.1 Data Availability

All price data used in this study consist of daily adjusted closing prices obtained from Yahoo Finance. The complete list of assets and the sample period are specified in Section 2. No proprietary datasets are used.

K.2 Code Availability

The complete source code used to generate all results in this paper, including data preprocessing, strategy implementation, and performance evaluation, is available upon request and will be released in a public repository upon publication.

The code is written in Python and relies exclusively on widely used open-source libraries.

K.3 Computational Environment

All experiments were conducted using a standard desktop computing environment. No specialized hardware or proprietary software is required. Randomized components of the experiments (e.g., random trading strategies and bootstrap procedures) are controlled via fixed random seeds to ensure reproducibility.

K.4 Reproducibility Notes

All trading strategies are implemented under explicitly stated assumptions regarding self-financing constraints and execution timing. No look-ahead bias or future information is used in signal construction.

Minor numerical discrepancies may arise across computing environments due to floating-point arithmetic, but these do not affect the qualitative conclusions of the study.