# A Cybernetics-Inspired Adaptive Trading System: Mathematical Methods and Implementation

July 20, 2025

## Contents



## 1 Introduction

This document details the theoretical and practical implementation of a cybernetic trading and forecasting framework, inspired by Norbert Wiener's *Cybernetics: Or Control and Communication in the Animal and the Machine*. The approach integrates core concepts from information theory, control theory, adaptive systems, and quantitative finance—most notably, rolling entropy, mutual information, Wiener filtering, feedback-based adaptive control, and a robust backtesting protocol, all implemented with awareness of look-ahead bias.

## 2 Data and Notation

Let $P_t$ denote the price (e.g., close) of an asset at time $t$. Assume a time series $P_1, P_2, \ldots, P_T$ sampled at regular intervals (e.g., daily). Let $X_t$ denote a vector of features or indicators constructed using the series up to time $t$.

## 3 Feature Engineering

Features are designed to capture both statistical properties and market regimes. All series and operations are computed using **only past and present values** (never future), to avoid forward-looking bias.

### 3.1 Returns

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

### 3.2 Rolling Mean and Standard Deviation

Define, for window length $w$:

$$\text{Mean}_t = \frac{1}{w} \sum_{i=t-w+1}^{t} P_i$$

$$\text{Std}_t = \sqrt{\frac{1}{w} \sum_{i=t-w+1}^{t} (P_i - \text{Mean}_t)^2}$$

### 3.3 Rolling Entropy

Given a vector $V = (P_{t-w+1}, ..., P_t)$ of length $w$, construct a histogram with $b$ bins, yielding empirical probabilities $p_j$.

$$\text{Entropy}_t = -\sum_{j=1}^{b} p_j \log_2 p_j$$

where $p_j$ is the normalized frequency of samples in bin $j$, and we set $0 \log_2 0 = 0$ by convention.

## 4 Information-Theoretic Feature Selection: Mutual Information

To quantify how much information a candidate feature carries about the future return, we use mutual information. Given feature matrix $X$ (shape $n \times m$) and future asset return $y$ (length $n$):

$$\text{MI}(X_j; y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

where $X_j$ is the $j$-th feature. Features with the highest MI are selected as primary predictors for the next window.

# 5 Signal Denoising: Wiener Filtering

For a time series $z_t$ with additive noise, the Wiener filter estimates the underlying "clean" signal using past and present values.[1]

$$\hat{z}_t = \mathcal{W}(z_{t-k}, \ldots, z_t; \sigma_n^2)$$

where $\mathcal{W}$ denotes the Wiener filter (e.g., implemented as a moving window, minimizing mean squared error), and $\sigma_n^2$ is the estimated noise variance.

# 6 Adaptive Weights: Online Linear Learning

Let each signal $X_t$ (for $t$ in the look-back window) be an $m$-dimensional feature vector. We aim to estimate a weight vector $\boldsymbol{w}$ for predicting the future target $y_t$ (such as next step return):

$$\hat{y}_t = \boldsymbol{w}^\top X_t$$

Weights are updated using **online gradient descent** (stochastic gradient):

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta(y_k - \hat{y}_k)X_k$$

where $\eta > 0$ is a learning rate, $y_k$ is the target (historical observed return), and $\hat{y}_k$ is the predicted value using current weights.

# 7 Feedback-Based Control Law

Inspired by regulatory mechanisms in cybernetics and control engineering, the trading action is based on the feedback between the desired and predicted signal. For proportional control (P-controller):

$$\text{TradeSignal}_t = K_p(T - S_t)$$

where $T$ is a target threshold (typically zero), $S_t$ is the predicted signal (e.g., predicted return), and $K_p$ is proportional gain (often set to 1). The trade decision at time $t$ is $\pm 1$ (long/short) based on the sign of $\text{TradeSignal}_t$.

# 8 Backtesting Engine

## 8.1 Simulation Protocol

At each step $t$:

[1] See Wiener, Norbert, *Cybernetics*, Ch. III.

**Step 1:** Use only data up to (and including) time $t-1$ to compute features, filter, adapt weights, select predictors, and generate trade signal.

**Step 2:** Observe price $P_t$ and determine P&L from holding previous position over $[t-1, t]$.

**Step 3:** Execute trade at $P_t$, taking into account transaction costs:

$$\text{Cost}_t = \gamma \left|\text{Position}_t - \text{Position}_{t-1}\right| \cdot P_t$$

where $\gamma$ is the proportional trading fee.

**Step 4:** Update equity by

$$\text{Equity}_t = \text{Cash}_{t-1} - \text{Cost}_t + \text{Position}_{t-1}(P_t - P_{t-1})$$

and $\text{Cash}_t = \text{Equity}_t - \text{Position}_t \cdot P_t$.

### 8.2 Performance Metrics

Backtest results are evaluated via:

- **Total Return**: $\dfrac{\text{Equity}_T}{\text{Equity}_0} - 1$
- **Sharpe Ratio**: $\dfrac{\mu_R}{\sigma_R}\sqrt{N}$, where $R$ is the periodic return
- **Maximum Drawdown**: $\min_t \left( \dfrac{\text{Equity}_t}{\max_{s \leq t} \text{Equity}_s} - 1 \right)$

## 9 Implementation Notes and Bias Avoidance

- **No Data Leakage**: All computations for trading decisions at time $t$ use features and targets computed from $[t - \text{lookback}, t-1]$ only—i.e., strictly up to, not including, time $t$'s outcome.
- **Synchronizing Indices**: Feature vectors and price series are aligned and dropped where missing data could introduce misaligned targets.
- **Online Adaptation**: Weights are updated per time window, based on rolling, in-sample past data only.
- **Transaction Costs**: Simulated on all position changes, to reflect realistic implementation.

## 10 Summary

This cybernetics-inspired trading system provides an integrated workflow for interpretable, adaptive algorithmic trading. Methods include:

- Rolling and information-theoretic feature engineering,
- Predictive signal denoising via Wiener filters,

- Adaptive online model learning via feedback control,
- Bias-avoiding and reproducible backtesting.

Such systems reflect Wiener's vision of robust, feedback-driven control and inference under uncertainty, applied here to modern algorithmic financial markets.