

TinyCTA: An Analytically Scaled Trend Signal and Correlation-Aware Position Sizing in Nine Modules

Thomas Schmelzer*

September 29, 2026

Abstract

TinyCTA is a deliberately small library for building trend-following strategies: roughly 1,300 lines covering signal generation, robust volatility estimation, and a position-sizing engine that solves a shrunk correlation system at every timestamp. This paper documents the pipeline and derives the one piece of it that is stated in the code without proof — the analytic scale factor of the oscillator. We show it is exactly the standard deviation of the EWMA difference under a driftless random walk with unit-variance increments, because the difference filter’s coefficients sum to zero and its cumulative representation telescopes; the derivation takes four lines, and a simulation over 2×10^5 steps confirms unit output standard deviation across a decade of (fast, slow) pairs. We then run the full engine over a 49-asset, 53-year futures panel and report what the shrinkage parameter actually does. It is a numerical necessity rather than a performance lever: at $\lambda = 1$, where the correlation matrix is used unmodified, the engine raises `SingularMatrixError` — the empirical correlation matrix over 47 assets has condition number 2.8×10^{18} — and as $\lambda \rightarrow 1$ mean gross exposure grows elevenfold while the Sharpe ratio stays inside $[0.21, 0.33]$. We also flag that the parameter reads backwards: `shrink=1.0` means *no* shrinkage, and it is the setting that fails.

1 Introduction

A trend-following CTA is a short pipeline. Prices become a signal; the signal is treated as an expected return; expected returns become positions, sized by volatility and by the correlation structure of the universe; positions become P&L. Each stage is a few lines of arithmetic, and the difficulty is not in any one of them but in the accumulated conventions — which returns, which lookback, which normalisation, what happens to an asset that has not started trading yet.

TinyCTA is an attempt to write that pipeline down at minimum size and to keep every convention explicit. It is nine modules (Schmelzer, 2025): two signal generators, volatility and matrix utilities, a validated configuration object, a Polars-facing engine (Vink and Polars contributors, 2025), a pure-NumPy numeric kernel (Harris et al., 2020), and an optional Optuna (Akiba et al., 2019) layer for parameter search. The core has four runtime dependencies and no plotting, no data loading, and no backtest framework.

The empirical literature on time-series momentum is large (Moskowitz et al., 2012; Hurst et al., 2017; Baltas and Kosowski, 2013) and this paper adds nothing to it. The contribution here is narrower and, we think, more useful: a precise account of what this implementation computes, a derivation of the one formula it asserts, and a measurement of the one parameter whose effect is not obvious.

*<https://github.com/tschm/TinyCTA>

2 Signals

Both signal generators are Polars expressions, so they compose inside a single `with_columns` call and run column-wise over a wide price frame.

2.1 Moving-average crossover

The discrete signal is the sign of an EWMA difference,

$$s_t = \text{sign}(\text{EWMA}_{\text{fast}}(x)_t - \text{EWMA}_{\text{slow}}(x)_t) \in \{-1, 0, +1\}, \quad (1)$$

with the lengths interpreted as centres of mass, `com = length - 1`, so that the decay factor is $1 - 1/\text{length}$. It is computed with `adjust=False`, the recursive form, and emits nulls until `min_samples` observations have accumulated.

2.2 The oscillator and its scale factor

The continuous signal is the same difference, divided by a constant:

$$\text{osc}_t = \frac{\text{EWMA}_{\text{fast}}(x)_t - \text{EWMA}_{\text{slow}}(x)_t}{s}, \quad s = \sqrt{\frac{1}{1-f^2} - \frac{2}{1-fg} + \frac{1}{1-g^2}}, \quad (2)$$

where $f = 1 - 1/\text{fast}$ and $g = 1 - 1/\text{slow}$. Here the EWMA uses `adjust=True`, the normalised form, whose steady-state weights are $(1-f)f^k$.

The code states s without derivation. It is worth having, because it is what makes oscillator magnitudes comparable across parameter choices — the property the whole design rests on.

Claim. Let x be a driftless random walk, $x_t = \sum_{j \leq t} \varepsilon_j$ with ε_j uncorrelated of unit variance. Then the numerator of Eq. (2) has standard deviation exactly s .

Proof. In steady state the numerator is a linear filter of the price history,

$$y_t = \sum_{k \geq 0} [(1-f)f^k - (1-g)g^k] x_{t-k}, \quad (3)$$

whose coefficients sum to $1 - 1 = 0$. A filter with coefficients summing to zero annihilates the unit root, so y_t is stationary even though x_t is not. Rewriting in terms of the increments and collecting the coefficient of ε_{t-m} gives a telescoping partial sum,

$$c_m = \sum_{k=0}^m [(1-f)f^k - (1-g)g^k] = (1-f^{m+1}) - (1-g^{m+1}) = g^{m+1} - f^{m+1}, \quad (4)$$

so that $y_t = \sum_{m \geq 0} c_m \varepsilon_{t-m}$ and

$$\text{Var}(y_t) = \sum_{m \geq 1} (g^m - f^m)^2 = \frac{g^2}{1-g^2} - \frac{2fg}{1-fg} + \frac{f^2}{1-f^2}. \quad (5)$$

Adding and subtracting the $m = 0$ term, $(1 - 2 + 1) = 0$, turns each $z^2/(1-z^2)$ into $1/(1-z^2)$ and recovers s^2 exactly as written in Eq. (2). $\square$

So the oscillator is a t -statistic of trend: numerator in units of price moves, denominator the standard deviation those moves would have if the price were a random walk. Under the null of no trend its output has unit variance, whatever `fast` and `slow` are, which is what allows a signal parameterised at (8, 24) and one at (64, 192) to be added, compared or blended without a per-parameter rescaling.

Table 1 is the numerical check. The realised standard deviation sits within 1.4% of unity for every pair tested, including the deliberately mismatched (8, 96). The small positive bias is a finite-sample effect: the steady-state weights assumed above are only approached after the slower EWMA has warmed up.

Table 1: Analytic scale factor against the realised standard deviation of the oscillator on a simulated driftless random walk with unit-variance Gaussian increments (2×10^5 steps, seed 0). The analytic factor varies by more than a factor of five across these parameter pairs; the output does not.

fast	slow	s (analytic)	sd(osc)
2	6	1.0851	0.9983
4	12	1.4651	1.0008
8	24	2.0334	1.0053
16	48	2.8513	1.0106
32	96	4.0159	1.0133
64	192	5.6680	1.0083
8	96	6.1323	1.0117

3 Volatility and matrix utilities

Volatility-adjusted returns. `vol_adj` standardises log returns by an EWMA standard deviation and clips the result,

$$\tilde{r}_t = \text{clip}\left(\frac{r_t}{\hat{\sigma}_t}, -c, +c\right), \quad r_t = \log x_t - \log x_{t-1}, \quad (6)$$

with $\hat{\sigma}$ an EWMA of r at centre of mass `vola` – 1. The clip is what keeps a single gap or a bad print from dominating everything downstream; the default in the engine is $c = 4.2$. `adj_log_prices` integrates $\tilde{r}$ by cumulative sum, giving a price-like series of roughly unit volatility — the natural input when a signal should see moves in risk units rather than in currency.

Note the warmup: an EWMA standard deviation is undefined for a single observation, so the first log return produces no value and the output starts at the second, irrespective of `min_samples`.

Robust volatility. `moving_absolute_deviation` is a doubly-robust alternative: both the centre and the dispersion are rolling medians of log returns,

$$\hat{\sigma}_t^{\text{MAD}} = \frac{1}{0.6745} \text{med}_w(|r_t - \text{med}_w(r_t)|), \quad w = 2 \text{ com} - 1. \quad (7)$$

The constant 0.6745 is the third quartile of the standard normal, so the estimator is consistent for the standard deviation under normality while retaining the median’s breakdown point of 50% (Huber and Ronchetti, 2009). Chaining two rolling medians over a differenced series costs $2w - 1$ returns of warmup before the first value appears.

Shrinkage towards the identity.

$$\text{shrink2id}(M, \lambda) = \lambda M + (1 - \lambda)I. \quad (8)$$

For a correlation matrix this preserves the unit diagonal and pulls every off-diagonal entry towards zero in proportion to $1 - \lambda$; it is the identity-target special case of the classical shrinkage estimator (Ledoit and Wolf, 2004).

A warning about the parameter’s direction. Equation (8) reads $\lambda = 1 \Rightarrow$ no shrinkage and $\lambda = 0 \Rightarrow$ complete shrinkage, and the engine’s configuration field carrying λ is named `shrink`. So `shrink=1.0` applies *no* shrinkage and `shrink=0.0` discards the correlations entirely — the opposite of how the name reads. Section 5.1 shows this is not a cosmetic complaint: `shrink=1.0` is exactly the setting that fails on real data.

4 The position engine

4.1 Configuration

Four parameters, validated by a frozen Pydantic model that rejects unknown keys: `vola` and `corr` (the two lookbacks, both positive), `clip` (positive), and `shrink` $\in [0, 1]$. One cross-field constraint is enforced: $\text{corr} \geq \text{vola}$, on the grounds that estimating a correlation matrix over a shorter window than the volatilities it is built from is numerically unstable. Freezing the model means a validated configuration cannot drift between construction and use.

4.2 Per-timestamp sizing

The engine consumes two aligned wide frames — `prices` and `mu`, the expected returns, with identical shapes and columns and a `date` column in each — and produces a frame of cash positions of the same shape.

Internally it derives clipped volatility-adjusted returns, an EWMA covariance matrix per timestamp over a window of $2\text{corr} + 1$ with `corr` observations of warmup, and normalises each to a correlation matrix C_t by its own diagonal. A zero-variance asset yields NaN rather than a division by zero. The forward walk is then delegated to a pure-NumPy kernel.

At timestamp t , writing $\mathcal{A}_t$ for the set of assets with a finite price (so an instrument that has not started trading is absent, not zero), μ_t for the expected returns restricted to $\mathcal{A}_t$ with NaNs zeroed, and $\rho_t = \text{shrink2id}(C_t, \lambda)$ restricted likewise:

$$u_t = \frac{\rho_t^{-1} \mu_t}{\sqrt{\mu_t^\top \rho_t^{-1} \mu_t}}, \quad (9)$$

$$\pi_t = \frac{u_t}{v_t}, \quad (10)$$

$$h_{t,i} = \frac{\pi_{t,i}}{\hat{\sigma}_{t,i}}. \quad (11)$$

Equation (9) is the informative step. It is a Markowitz direction (Markowitz, 1952) in correlation space, normalised so that $u_t^\top \rho_t u_t = 1$ — one unit of risk under the metric ρ_t , exactly. The normaliser is computed as $\sqrt{\mu_t^\top \rho_t^{-1} \mu_t}$, and the position is zeroed rather than scaled when that quantity is non-finite, below 10^{-12} , or when μ_t vanishes.

Equation (10) is a feedback loop. Realised profit $p_t = h_{t-1}^\top r_t$ drives an EWMA of squared profit,

$$v_t = \kappa v_{t-1} + (1 - \kappa) p_t^2, \quad \kappa = 0.99, \quad v_0 = 1, \quad (12)$$

and dividing by it shrinks the book after volatile P&L and expands it after quiet P&L — a volatility target expressed in realised profit rather than in forecast risk. Note that κ and v_0 are hard-coded, not configuration.

Equation (11) converts risk units to cash by dividing by each asset’s own EWMA volatility of simple returns, so a given risk allocation buys fewer units of a volatile instrument.

Scale is not pinned down. Equation (12) carries units. p_t is a currency amount, so v_t is a squared currency amount, while u_t in Eq. (9) is dimensionless; dividing one by the other leaves the overall size of the book determined jointly by $v_0 = 1$ and the numerical scale of the price series. There is no target-volatility or AUM parameter. The gross exposures in Table 2 should therefore be read as relative, not as leverage against a stated capital base — a caller who needs a specific risk level must rescale the output.

Table 2: The engine over the 49-asset panel as λ (**shrink**) varies. Recall $\lambda = 1$ means the correlation matrix is used unmodified. Gross exposure is $\sum_i |h_{t,i}|$, in the units of the price series.

λ	Sharpe	Mean gross	Max gross
0.00	0.265	128.9	706.3
0.25	0.225	428.4	4,436.8
0.50	0.220	629.7	7,423.6
0.75	0.255	889.2	9,891.2
0.90	0.327	1,168.7	12,229.1
0.99	0.206	1,415.4	219,767.6
1.00	SingularMatrixError		

5 Empirical behaviour

We run the pipeline end to end on the futures panel shipped with the test suite: 13,909 daily observations of 49 instruments with hashed identifiers, spanning 1970-01-02 to 2023-04-26, with instruments entering over time (the median instrument has about 9,100 observations; all 49 are live at the end). Expected returns are $\text{osc}(8, 24)$ per asset, nulls filled with zero. The configuration is $\text{vola} = 32$, $\text{corr} = 64$, $\text{clip} = 4.2$, and λ varies. P&L is $p_t = \sum_i h_{t-1,i} r_{t,i}$ with r simple returns; the Sharpe ratio is $\text{mean}(p) / \text{sd}(p) \times \sqrt{252}$, gross of all costs.

5.1 What shrinkage does

Table 2 contains the paper’s main empirical finding, and it is about numerics rather than returns.

At $\lambda = 1$ the engine does not produce a worse result — it produces no result. The solve behind Eq. (9) raises **SingularMatrixError**: the Cholesky factorisation fails as not positive-definite and the fallback general solve reports a singular matrix. This is not a fluke of one timestamp. The final EWMA correlation matrix over the 47 assets with finite entries has condition number 2.79×10^{18} and a smallest eigenvalue of 9.16×10^{-17} — numerically rank-deficient at double precision, notwithstanding an EWMA window (129) considerably longer than the cross-section (49). Clipping and the overlapping decay weights leave the effective sample far smaller than the nominal window.

Shrinkage is therefore load-bearing infrastructure in this engine, not a tuning knob: any $\lambda < 1$ makes ρ_t invertible, because Eq. (8) adds $(1 - \lambda)$ to every eigenvalue.

The route to that failure is visible in the exposure columns, and it is the more practically useful observation. Mean gross exposure rises elevenfold from $\lambda = 0$ to $\lambda = 0.99$, and maximum gross exposure rises by a factor of 311, reaching 2.2×10^5 at $\lambda = 0.99$ against 706 at $\lambda = 0$. Ill-conditioning does not announce itself as an exception until the very end; long before that it shows up as leverage, as ρ_t^{-1} amplifies small differences between nearly collinear assets (Michaud, 1989). A practitioner monitoring positions would see it; one monitoring only the Sharpe ratio would not.

Because the Sharpe ratio does not see it. Across the whole feasible range it stays inside $[0.206, 0.327]$, non-monotone in λ , with the best value at $\lambda = 0.90$ — and we would caution explicitly against reading that as a recommendation. The spread is well inside what 53 years of one gross-of-costs configuration can distinguish (Harvey and Liu, 2018), and the $\lambda = 0.99$ column shows how a nominally similar Sharpe can be produced by a book two orders of magnitude larger at its peak. The honest summary is that correlation shrinkage buys invertibility and exposure control here, and that this naive configuration — one signal, no costs, no turnover control, in-sample parameters — has no evidence to offer about its effect on returns.

5.2 Parameter search

The optional `tinycta.hyper` layer wraps Optuna (Akiba et al., 2019) with a tree-structured Parzen estimator (Bergstra et al., 2011), maximising the Sharpe ratio of a caller-supplied portfolio-suggestion function. Trials returning NaN are pruned rather than failed, which keeps the reported statistics clean. The layer is behind an extra (`pip install "tinycta[hyper]"`) and imports nothing that the core needs; the numeric kernel deliberately avoids even a logging dependency so that a core install stays at four packages.

Given the width of the Sharpe band in Table 2, we note that the same caution applies with more force to any search over more parameters: an optimiser that maximises in-sample Sharpe over a four-dimensional configuration will find something, and this panel cannot tell whether it found a strategy.

6 Limitations

No trading frictions. No transaction costs, no slippage, no market impact, no financing, and no turnover penalty. Positions can change arbitrarily between consecutive timestamps, and Eq. (9) has no term that discourages it. Any Sharpe ratio quoted here is gross.

No risk limits. There are no position caps, no gross or net exposure limits, no drawdown control, and no leverage constraint. The only mechanism restraining size is the profit-variance feedback of Eq. (12), which reacts to realised P&L after the fact and, as Table 2 shows, permits gross exposure to move over two orders of magnitude.

Memory. The engine materialises one correlation matrix per post-warmup timestamp in a dictionary. On this panel that is roughly 13,800 matrices of 49×49 doubles — about 265 MB — and it grows as $O(Tn^2)$. The design is fine at CTA scale and will not survive a large cross-section or intraday data without being reworked into a streaming walk.

Hard-coded constants. The profit-variance decay $\kappa = 0.99$, its initial value $v_0 = 1$, and the degeneracy floor 10^{-12} are not configurable. The first two set the book’s scale and its responsiveness, so they are choices a user might reasonably want to make.

In-sample, single panel. Every number in Section 5 comes from one dataset with parameters chosen before looking at the result but evaluated on the whole history. Nothing here is a backtest with an out-of-sample period, and nothing here should be read as evidence that this configuration works.

7 Conclusion

The pipeline is small enough to state completely, which is the point of the package and the point of this paper. Two results are worth carrying away.

The oscillator’s scale factor is exact, not heuristic: it is the standard deviation of the EWMA difference under a random walk, provable in four lines because the difference filter’s coefficients telescope, and confirmed to within 1.4% in simulation across parameter pairs whose raw magnitudes differ by more than five times. Signals at different speeds really are comparable without further normalisation.

Correlation shrinkage is not optional. At $\lambda = 1$ — which, confusingly, is what `shrink=1.0` means — the engine cannot run at all on a 49-asset panel whose correlation matrix is numerically singular, and as λ approaches 1 the deterioration shows up first as an eleven-fold growth

in gross exposure rather than as an error or a worse Sharpe ratio. The parameter’s name inverts its meaning, and its safe region is bounded away from the value the name suggests is safest.

A Reproducibility

All numbers were produced with TinyCTA 0.14.0 and `tests/tinycta/resources/prices_hashed.csv`. Columns with long leading gaps are inferred as strings by default, hence the explicit cast.

```
import numpy as np, polars as pl
from tinycta.config import Config
from tinycta.engine import Engine
from tinycta.osc import osc

raw = pl.read_csv("tests/tinycta/resources/prices_hashed.csv",
                  try_parse_dates=True, infer_schema_length=None)
raw = raw.rename({raw.columns[0]: "date"})
assets = [c for c in raw.columns if c != "date"]
prices = raw.with_columns(pl.col(a).cast(pl.Float64) for a in assets)
mu = prices.with_columns(osc(pl.col(a), fast=8, slow=24).alias(a)
                        for a in assets)
mu = mu.with_columns(pl.col(a).fill_null(0.0).fill_nan(0.0) for a in assets)

P = prices.select(assets).to_numpy()
ret = np.zeros_like(P); ret[1:] = P[1:] / P[:-1] - 1.0

for shrink in (0.0, 0.25, 0.5, 0.75, 0.9, 0.99, 1.0):
    cfg = Config(vola=32, corr=64, clip=4.2, shrink=shrink)
    engine = Engine(prices=prices, mu=mu, cfg=cfg)
    H = engine.cash_position.select(assets).to_numpy()
    p = np.nansum(np.nan_to_num(H[:-1]) * np.nan_to_num(ret[1:]), axis=1)
    gross = np.nansum(np.abs(H), axis=1)
    print(shrink, p.mean() / p.std() * np.sqrt(252),
          np.nanmean(gross), np.nanmax(gross))
```

The $\lambda = 1.0$ row raises `cvx.linalg.core.exceptions.SingularMatrixError` from inside `Engine.cash_position`. Condition numbers are of the last correlation matrix returned by `Engine.cor`, restricted to rows and columns that are finite throughout. Table 1 simulates `np.random.default_rng(0).standard_normal(200_000)`, takes its cumulative sum, and compares `np.std` of the oscillator against `s` evaluated directly.

References

- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019.
- Nick Baltas and Robert Kosowski. Momentum strategies in futures markets and trend-following funds. *SSRN Electronic Journal*, 2013.

- James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyperparameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011.
- Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. Array programming with numpy. *Nature*, 585:357–362, 2020.
- Campbell R. Harvey and Yan Liu. Lucky factors. *SSRN Electronic Journal*, 2018.
- Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley, 2nd edition, 2009.
- Brian Hurst, Yao Hua Ooi, and Lasse Heje Pedersen. A century of evidence on trend-following investing. *The Journal of Portfolio Management*, 44(1):15–29, 2017.
- Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004.
- Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.
- Richard O. Michaud. The markowitz optimization enigma: Is ‘optimized’ optimal? *Financial Analysts Journal*, 45(1):31–42, 1989.
- Tobias J. Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012.
- Thomas Schmelzer. Tinycta: A lightweight python package for commodity trading advisor strategies. <https://github.com/tschm/TinyCTA>, 2025. Version 0.14.0.
- Ritchie Vink and Polars contributors. Polars: Dataframes for the new era. <https://pola.rs>, 2025.