Adapted Turtle Trading Strategy - Implementation Guide for SIP & Monte Carlo (SciPy)

Purpose

This document specifies the exact logic, mathematics, and data contracts required to implement the adapted Turtle Trading strategy on your SIP modelling and Monte Carlo simulation website. All formulas are presented both plainly and as Excel expressions, and pseudocode / Python (NumPy/SciPy-ready) snippets are provided for a no-code tool or developer to implement faithfully. The design assumes SciPy is available for random variate generation and distribution fitting, and (optionally) NumPy/Pandas for vectorized calculations.

1) Strategy Overview (Adapted Turtle)

Trend-following with Donchian-channel breakouts, volatility-normalized position sizing using ATR (Wilder), fixed initial stop at 2N, profit-taking target at 4N (optional) and/or trailing exits, and scaling-in additional units every 0.5N in favorable movement up to 4 total units per market. Risk per trade is a fixed percentage of account equity.

2) Inputs and Data Contracts

Required time series per instrument:
- Date, Open, High, Low, Close, Volume (daily as default).
- Contract specifications: point value/tick value, contract size (e.g., 1 lot = units), min tick.
- Trading costs: commission per side, estimated slippage (in ticks or bps).

Global parameters (per backtest/simulation run):
- Account_Start (e.g., 10,000), RiskPct (e.g., 0.01 or 0.02), MaxUnitsPerMarket = 4, MaxUnitsCorrelated = 6, MaxUnitsDirection = 12.
- N_Period for ATR (Wilder 20 default); Donchian lookbacks: DC_Short = 20/10 exit; DC_Long = 55/20 exit.
- Scaling increment = 0.5N. Profit target = 4N (optional if you favor pure trailing exits).

3) Volatility (ATR) and Breakouts (Donchian Channels)

Average True Range (Wilder, length = N):
TR_t = max(High_t - Low_t, |High_t - Close_{t-1}|, |Low_t - Close_{t-1}|)
ATR_t (Wilder) = EMA(TR, alpha = 1/N) with ATR_0 = mean(TR first N).
We denote N = ATR_t at entry. In all formulas below, N is the ATR at the time of each action.

Donchian Channels:
- DC_high_L = rolling max(High, L)
- DC_low_L = rolling min(Low, L)

Entries:
- System 1 (short-term): go long when Close breaks above DC_high_20, short when below DC_low_20.
- System 2 (long-term): go long when Close breaks above DC_high_55, short when below DC_low_55.

Exits (classic):
- System 1 exit on 10-day opposite breakout; System 2 exit on 20-day opposite breakout.

Your adaptation also allows a fixed take-profit at 4N and 2N initial stop.

4) Position Sizing (Core Math)

Risk per trade (in account currency): Risk$ = Equity * RiskPct.
Initial stop distance = 2N (price units). Dollar risk per 1 unit (share/oz/contract) = 2N * PointValue.

Units = Risk$ / (2N * PointValue). Convert to lots with ContractSize: Lots = Units / ContractSize.
Cap adds so total units <= 4x initial unit risk per market.

Excel (cells):
- ATR in A1, Entry in D2, Account in A2, Risk% in B2, ContractSize in E2, PointValue in E3 (if needed)
- Risk$ = A2*(B2/100)
- Units = Risk$ / (2*A1*PointValue) (if PointValue=1, Units = Risk$/(2*A1))
- Lots = Units / E2
- StopLoss_long = D2 - (2*A1) ; StopLoss_short = D2 + (2*A1)
- TakeProfit_long = D2 + (4*A1) ; TakeProfit_short = D2 - (4*A1)
- Add1_long = D2 + 0.5*A1 ; Add2_long = D2 + 1.0*A1 ; Add3_long = D2 + 1.5*A1
  (mirror signs for shorts).

5) Scaling Logic

When in profit by k*0.5N from the last filled unit price, add a new unit of the same size as the initial unit (or re-compute using current ATR if desired). Stop adding after 3 adds (4 total units). For each add, raise the composite stop using either: (A) the worst-case stop across all units (conservative), or (B) a pyramiding stop method (e.g., last fill price - 2N). The classic Turtles maintained a uniform 2N stop from the most recent add.

6) Portfolio Risk Limits

- MaxUnitsPerMarket = 4; MaxUnitsCorrelated = 6 (define correlation groups, e.g., equity indices); MaxUnitsDirection = 12 (sum of all long or all short risk units across the book).
- Skip new entries that would breach limits; allow exits anytime.

7) Trading Costs & Slippage

Include: commission_per_side, spread or slippage (fixed ticks or proportional bps), financing (CFD swaps). In simulation, deduct costs at entry/exit and for each add. Optionally model slippage as a distribution with SciPy (e.g., normal with mean=expected_slippage, sd=sigma_slippage).

8) SIP (Systematic Investment Plan) Cashflows

If your website models SIP contributions/withdrawals, inject cashflows into equity at a fixed interval (e.g., monthly contribution C). Apply contributions before risk sizing on that day so Risk$ uses updated equity. In Monte Carlo, allow C to vary (e.g., lognormal or normal with truncation) using scipy.stats to sample.

9) Monte Carlo Design (SciPy)

Goal: sample return/price paths, run the strategy path-wise, aggregate performance distributions.

Price model options:
- Geometric Brownian Motion (GBM): dS/S = mu dt + sigma dW. Discretize: S_{t+1} = S_t * exp((mu-0.5*sigma^2)dt + sigma*sqrt(dt)*Z).
- Bootstrapped blocks of historical returns (preserves fat tails/clustering).
- Regime-switching or t-distributed returns via scipy.stats.t.

ATR during simulation: recompute rolling TR/ATR along each path using Wilder EMA. Donchian highs/lows also recomputed along path using rolling window logic.

10) Outputs and Performance Metrics

Per path: equity curve, trade log, total return, CAGR, volatility, Sharpe (or Sortino), max drawdown, win rate, profit factor, average trade, average hold days, percent time invested, exposure by asset class. Distribution of these statistics across paths yields confidence intervals. Present percentile bands (5/50/95).

11) End-to-End Pseudocode (Single Instrument)

```
Given: time series {O,H,L,C}, parameters (N_ATR=20, L_entry=20 or 55, L_exit=10 or 20),
RiskPct, ContractSize, PointValue.

Initialize:
  equity = Account_Start
  units_held = 0
  last_entry_price = NaN
  filled_prices = []

Compute ATR_t (Wilder) and Donchian bands DC_high_L, DC_low_L.

For each day t from max(N_ATR, L_entry) to T:
  # Update risk budget with SIP contribution if today is contribution day
  if is_contribution_day(t): equity += C_t

  N = ATR_t
  entry_long  = (C_t > DC_high_L[t]) and units_held == 0
  entry_short = (C_t < DC_low_L[t])  and units_held == 0

  # Sizing
  Risk$ = equity * RiskPct
  unit_size = Risk$ / (2 * N * PointValue)

  if entry_long:
     price = close[t]
     stop  = price - 2*N
     tp    = price + 4*N   # optional
     enter +unit_size @ price; units_held += 1; filled_prices.append(price)
     last_fill_price = price

  elif entry_short:
     price = close[t]
     stop  = price + 2*N
     tp    = price - 4*N   # optional
     enter -unit_size @ price; units_held -= 1; filled_prices.append(price)
     last_fill_price = price

  # Scaling-in (if already in a position and less than 4 units total)
  if position_is_long and units_held < 4:
      if close[t] >= last_fill_price + 0.5*N:
          add +unit_size; filled_prices.append(close[t]); units_held += 1
          stop = close[t] - 2*N  # or keep global worst-case stop
          last_fill_price = close[t]

  if position_is_short and units_held > -4:
      if close[t] <= last_fill_price - 0.5*N:
          add -unit_size; filled_prices.append(close[t]); units_held -= 1
          stop = close[t] + 2*N
          last_fill_price = close[t]

  # Exits: opposite Donchian breakout or stop/take-profit hit
```

```
  if position_is_long:
      exit_signal = (close[t] < DC_low_exit[t]) or (low[t] <= stop) or (close[t] >= tp)
  else:
      exit_signal = (close[t] > DC_high_exit[t]) or (high[t] >= stop) or (close[t] <= tp)

  if exit_signal:
      close all at close[t]; units_held = 0; filled_prices = []; last_entry_price = NaN
      update equity with P&L - costs
```

12) SciPy / NumPy Implementation Hints

```
# Random draws (SciPy)
from scipy import stats
Z = stats.norm.rvs(size=(n_paths, n_steps))  # standard normal shocks

# GBM path generator (vectorized)
import numpy as np
def gbm_paths(S0, mu, sigma, dt, Z):
    drift = (mu - 0.5*sigma**2)*dt
    shock = sigma*np.sqrt(dt)*Z
    log_returns = drift + shock
    S = np.empty_like(Z)
    S[:,0] = S0
    for t in range(1, Z.shape[1]):
        S[:,t] = S[:,t-1] * np.exp(log_returns[:,t])
    return S

# Wilder ATR (1D; extend per path)
def wilder_atr(high, low, close, n=20):
    tr = np.maximum(high-low, np.maximum(np.abs(high-np.roll(close, 1)), np.abs(low-np.roll(close,
1))))
    tr[0] = high[0]-low[0]
    atr = np.empty_like(tr, dtype=float)
    atr[:n] = np.nan
    atr[n-1] = np.nanmean(tr[:n])
    alpha = 1.0/n
    for i in range(n, len(tr)):
        atr[i] = atr[i-1] + alpha*(tr[i] - atr[i-1])
    return atr

# Donchian channels
def donchian_high(high, L):
    out = np.full_like(high, np.nan, dtype=float)
    for i in range(L-1, len(high)):
        out[i] = np.max(high[i-L+1:i+1])
    return out

def donchian_low(low, L):
    out = np.full_like(low, np.nan, dtype=float)
    for i in range(L-1, len(low)):
        out[i] = np.min(low[i-L+1:i+1])
    return out
```

13) Excel Mapping (Cells)

Use the following cells for a universal sheet:

A2=AccountSize, B2=Risk%, C2=ATR (N), D2=EntryPrice, E2=ContractSize (units per lot), F2=Long/Short text, G2=PointValue (1 if quote is $ per unit).

Formulas:
- Risk Per Trade (H2): =A2*(B2/100)
- Units (I2): =H2/(2*C2*G2)
- Stop-Loss (J2): =IF(F2="Long", D2-(2*C2), D2+(2*C2))
- Take-Profit (K2): =IF(F2="Long", D2+(4*C2), D2-(4*C2))
- Lot Size (L2): =I2/E2
- First Add (M2): =IF(F2="Long", D2+(0.5*C2), D2-(0.5*C2))
- Second Add (N2): =IF(F2="Long", D2+(1*C2), D2-(1*C2))
- Third Add (O2): =IF(F2="Long", D2+(1.5*C2), D2-(1.5*C2))
- Max Positions (P2): =I2*4 ; Max Lots (Q2): =L2*4

14) Tradable Instruments (Liquid & Trend-Friendly)

Select instruments with deep liquidity, clear long-term trends, and reliable price data.

FX Majors & Crosses: EUR/USD, GBP/USD, USD/JPY, USD/CHF, AUD/USD, NZD/USD, USD/CAD, EUR/GBP, EUR/JPY, GBP/JPY.
Metals: XAU/USD (Gold), XAG/USD (Silver), XPT/USD (Platinum), XPD/USD (Palladium), Copper (HG).
Energy: WTI Crude (CL/USOIL), Brent (BZ/UKOIL), Natural Gas (NG), Heating Oil (HO), Gasoline (RB).
Agricultural (liquid): Corn (ZC), Soybeans (ZS), Wheat (ZW), Sugar (SB), Coffee (KC), Cotton (CT), Cocoa (CC).
Equity Indices: S&P 500 (ES/SPX), Nasdaq 100 (NQ/NDX), Dow (YM/DJIA), Russell 2000 (RTY), DAX 40, EURO STOXX 50, FTSE 100, CAC 40, Nikkei 225, Hang Seng, ASX 200.
Rates & Bonds: US 2Y/5Y/10Y/30Y futures (ZT/ZF/ZN/ZB), German Bund (FGBL), Bobl (FGBM), Schatz (FGBS), STIRs (Euribor/SONIA).
Crypto (if allowed & sufficiently liquid): BTC/USD, ETH/USD, XRP, SOL.

Group correlated markets (e.g., equity indices together) to enforce MaxUnitsCorrelated.

15) Validation, Edge Cases, and Practical Notes

- Use walk-forward or Monte Carlo with bootstrapped return blocks to validate robustness.
- Recompute ATR and Donchian on every step; avoid look-ahead bias.
- Respect exchange holidays and gaps; stops may slip - model gap risk.
- For CFDs, confirm broker contract size and point value before converting units -> lots.
- Record every fill with price, N at fill, and fees; reconcile equity to trade log.