Testing Audit Policies Specification

Universal Stochastic Predictor (USP)

Adaptive Meta-Prediction Development Consortium

Document Version: 1.0
Last Updated: 2026-02-20

Abstract

This document defines the complete set of 45 testing policies derived from the authoritative specification corpus. All policies are strict, verifiable, and aligned with mathematical guarantees specified in the theory and implementation documents.

Source Documents:

- `doc/latex/specification/Stochastic_Predictor_Test_Cases.tex`
- `doc/latex/specification/Stochastic_Predictor_Tests_Python.tex`

Contents

1 Unit Tests: Kernels and Fundamental Algorithms

1.1 Category: Entropy and Random Variable Generation

1.1.1 TESTING POLICY #1: Validation of α-Stable Distributions (CMS Algorithm)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 32-38

Statement: Validate that the Chambers-Mallows-Stuck (CMS) algorithm generates α-stable random variables with parameters $(\alpha, \beta, \gamma, \delta)$ matching theoretical distributions on large sample sizes.

Criteria:

- Sample size $N \geq 10^4$
- Empirical moments must match theoretical properties within 95% confidence interval
- No NaN or Inf values detected
- For $\alpha \geq 1.8$, variance must not exceed expected variance $\times 1.5$
- For $|\beta| < 0.9$, empirical mean must be within $3\sigma/\sqrt{N}$

1.1.2 TESTING POLICY #2: Mersenne Twister/PCG64 Integrity

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 40-48

Statement: Verify the pseudo-random number generator (PRNG) has no serial correlations and meets long period guarantees.

Criteria:

- Apply standard statistical test batteries (TestU01, Diehard)
- No randomness tests fail
- Generator period $\geq 2^{127}$ (Mersenne Twister) or 2^{128} (PCG64)
- Zero serial correlations detected
- Deterministic seed initialization guarantees reproducibility

1.1.3 TESTING POLICY #3: WTMM Validation (Hölder Exponent Detection)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 52-60

Statement: Use synthetic signals with known Hölder exponent H to validate that the Wavelet Transform Modulus Maxima (WTMM) algorithm recovers singularity spectrum $D(h)$ accurately.

Criteria:

- Estimated Hölder exponent error: $|\hat{H} - H_0| < 0.05 \times H_0$ (5% relative error)
- Singularity spectrum recovered with $< 5\%$ error
- Multiscale analysis performs correctly across all scales

1.1.4 TESTING POLICY #4: Cone of Influence (Besov Influence Radius)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 62-71

Statement: Verify that maxima linking in scale space respects the influence radius defined by Besov constant C_{besov}.

Criteria:

- For consecutive maxima at scales $s_1 < s_2$, temporal positions satisfy: $|t_2 - t_1| \leq C_{\text{besov}} \times (s_2 - s_1)$
- Temporal coherence preserved across scales
- No spurious linkages beyond Besov radius

1.1.5 TESTING POLICY #5: Soft Nyquist Limit Validation (Multifractal Aliasing)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 73-136

Statement: Gradually reduce sampling frequency and validate system detects Nyquist undersampling before irreversible spectrum degradation, triggering `FreezingTopologicalBranchEvent`.

Criteria:

- Preventive detection: Emit `FreezingTopologicalBranchEvent` when relative error $\varepsilon_k > 0.05$ (5%)
- Acceptance criterion: Holder error $< 10\%$ at moment of freezing
- Critical frequency formula: $f_{\min} \geq (10/s_{\min}) \times (1 + H_{\min}^{-1})$
- Corrective action: Freeze topological branch weight $w_C \to w_C^{\text{frozen}}$ with no dynamic updates

1.2 Category: Algebraic Structures (Branch D)

1.2.1 TESTING POLICY #6: Signature Concatenation (Chen Identity)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 140-150

Statement: Validate that concatenating signatures of two path segments via tensor product equals signature of full path (Chen's identity).

Criteria:

- For paths γ_1, γ_2: $\text{Sig}(\gamma_1 \star \gamma_2) = \text{Sig}(\gamma_1) \otimes \text{Sig}(\gamma_2)$
- Numerical error $< 10^{-6}$ in Euclidean norm
- Identity holds for all path lengths and dimensions

1.2.2 TESTING POLICY #7: Temporal Reparametrization Invariance

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 152-161

Statement: Check that level-M truncated signature is invariant under strictly increasing time reparametrization.

Criteria:

- For strictly increasing $\varphi : [0,1] \to [0,1]$ with $\varphi(0) = 0, \varphi(1) = 1$
- $\text{Sig}^{(M)}(\gamma) = \text{Sig}^{(M)}(\gamma \circ \varphi)$
- Error $< 10^{-8}$ for all reparametrizations
- Negative control: Non-monotone reparametrizations must show different signatures

2 Integration Tests and Stochastic Convergence

2.1 Category: Stochastic Differential Equation (SDE) Solvers

2.1.1 TESTING POLICY #8: Euler-Maruyama vs Milstein Convergence

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 167-184

Statement: For diffusion with non-constant volatility, verify Milstein achieves strong convergence order 1.0 versus 0.5 for Euler-Maruyama.

Criteria:

- Euler-Maruyama error: $\mathbb{E}[|X_T - X_T^{EM}|^2] = O(\Delta t^{0.5})$
- Milstein error: $\mathbb{E}[|X_T - X_T^{M}|^2] = O(\Delta t^{1.0})$
- Convergence rates verified across multiple time step sequences
- Method order determined empirically from error decay rates

2.1.2 TESTING POLICY #9: CFL Condition Violation (Mixed CFL Stability)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 186-200

Statement: Force time step Δt violating Courant-Friedrichs-Lewy restriction and confirm numerical instability as expected behavior.

Criteria:

- $\Delta t > 10 \times C$ where $C =$ CFL bound
- Divergence detection: NaN or Inf emergence in trajectories
- Instability alert emitted by safety module
- System gracefully halts rather than silently diverging

2.2 Category: Transport Optimization (Orchestrator)

2.2.1 TESTING POLICY #10: Sinkhorn Algorithm Stability (Log-Domain Convergence)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 204-215

Statement: Evaluate Sinkhorn convergence in log domain with decreasing regularization ε, ensuring no underflow down to $\varepsilon \geq 10^{-4}$.

Criteria:

- Convergence guaranteed when $\varepsilon \geq 10^{-4}$
- System detects underflow risk and emits warning for $\varepsilon < 10^{-4}$
- Convergence measured by: $\|K\text{diag}(u)K^T\text{diag}(v) - \mu\|_1 < 10^{-6}$
- Log-domain arithmetic prevents numerical underflow

2.2.2 TESTING POLICY #11: Probabilistic Mass Conservation (Simplex Normalization)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 217-230

Statement: Confirm that after JKO update, sum of kernel weights is strictly 1.0 (probability simplex constraint).

Criteria:

- After JKO iteration: $\sum_i \rho_i^{(n+1)} = 1.0$
- All weights $\rho_i^{(n+1)} \geq 0$
- Numerical tolerance: $|\sum_i \rho_i^{(n+1)} - 1.0| < 10^{-10}$
- Simplex projection automatic in Sinkhorn algorithm

2.3 Category: HJB Solution via DGM (Branch B)

2.3.1 TESTING POLICY #12: Gradient Stability (Gradient Explosion Under Volatility)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 236-268

Statement: Monitor gradient norms during PDE training to detect and mitigate gradient explosions in high-volatility regimes.

Criteria:

- Monitor loss gradient norm: $\|\nabla_\theta L_{DGM}(\theta)\|_2$
- Gradient clipping threshold: $C_{\text{clip}} = 10.0$
- In high-volatility ($\sigma > 2\sigma_0$): if clipping occurs ≥ 5 consecutive iterations $\rightarrow$ emit `GradientInstabilityEv`
- Adaptive learning rate reduction: $\eta \rightarrow 0.5\eta$
- Stabilization required within next 20 epochs

2.3.2 TESTING POLICY #13: Crandall-Lions Comparison Principle (Viscosity Solution Validation)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 270-306

Statement: Validate neural solution respects comparison principle for viscosity solutions, ensuring uniqueness and consistency for non-linear PDEs.

Criteria:

- Time monotonicity for finite horizon (non-negative costs): $V(x, t_1) \geq V(x, t_2)$ for $t_1 < t_2$
- Sub/supersolution constraints: $\|\text{PDE}[V_\theta]\|_{L^\infty} \leq 10^{-3}$
- Compare with reference solution: $\|V_\theta - V_{\text{ref}}\|_{L^\infty} < 0.05 \times \|V_{\text{ref}}\|_{L^\infty}$ (5% error)
- Comparison principle verifies uniqueness

2.3.3 TESTING POLICY #14: Mode Collapse Detection (Training Entropy Test)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 308-355

Statement: Verify neural network does not collapse to trivial constant solutions during DGM training.

Criteria:

- Variance ratio: $\kappa_{\text{low}} \leq \text{Var}_x[V_\theta(x,t)]/\text{Var}[g(\xi)] \leq \kappa_{\text{high}}$ with $\kappa_{\text{low}} = 0.3$, $\kappa_{\text{high}} = 1.2$
- Acceptance: Variance ratio in range for $\geq 90\%$ of $t \in [0, T]$
- Differential entropy: $H[V_\theta] > H_{\min}$
- Spatial gradient norm: $\mathbb{E}_x[\|\nabla_x V_\theta\|_2] > \varepsilon_{\text{grad}} > 0$
- False positives avoided: only activate mode collapse alert on sustained variance loss

2.3.4 TESTING POLICY #15: Mesh Refinement Convergence (DGM Consistency)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 357-374

Statement: Verify DGM solution converges to exact solution as collocation point density increases.

Criteria:

- Convergence required across nested meshes with densities $N_1 < N_2 < N_3$
- Error at mesh k: $e_k = \|V_{\theta_k} - V_{\text{ref}}\|_{L^2}$
- Monotone convergence: $e_1 > e_2 > e_3$
- Empirical convergence rate: $r \geq 0.5$ (order $N^{-0.5}$)
- DGM loss $< 10^{-4}$ at convergence

2.3.5 TESTING POLICY #16: Minimum Variance Threshold (Mode Collapse Alternative)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 376-407

Statement: Complement entropy monitoring with direct variance ratio check to detect mode collapse.

Criteria:

- Minimum variance ratio: $R_{\text{var}}(t) \geq 0.10$ for all $t \in [0, 0.9T]$ (failure if < 0.10)
- Median variance ratio: $\text{median}_t R_{\text{var}}(t) \geq 0.50$
- Variance scaling captures $\geq 10\%$ of true variability
- If $R_{\text{var}} < 0.10$: interrupt training and adjust hyperparameters
- Warning levels: $R_{\text{var}} < 0.10$ (critical), 0.10-0.30 (warning), ≥ 0.50 (normal)

3 Robustness Tests and Circuit Breakers

3.1 Category: Outlier and Regime Handling

3.1.1 TESTING POLICY #17: Outlier Injection (Extreme Values)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 415-428

Statement: Inject extreme values ($> 20\sigma$) and verify system rejects point, emits alert, and preserves weights.

Criteria:

- Detect observations with $|y_t - \mu_t| > 10\sigma_t$
- Reject observation: NOT update meta-state Ξ_t
- Emit `OutlierDetectedEvent` with rejection metadata
- Keep weights $\{w_i\}_{i=A}^{D}$ unchanged
- Preserve system state integrity

3.1.2 TESTING POLICY #18: CUSUM Change Detection (Regime Change)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 430-442

Statement: Simulate regime change (structural drift) and validate change event emitted exactly when G_t^+ exceeds threshold h.

Criteria:

- CUSUM accumulation: $G_t^+ = \max(0, G_{t-1}^+ + (y_t - \mu_0) - k)$
- When $G_t^+ > h$: emit `RegimeChangedEvent`
- Reset $G_t^+ = 0$ after detection
- Detection delay $\leq \tau$ observations from true change point
- False positive control in stationary regimes

3.1.3 TESTING POLICY #19: Emergency Mode Activation (Critical Singularity)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 447-462

Statement: When Hölder exponent drops below $H_{\min}$, force $w_D \to 1.0$ and switch cost to Huber metric.

Criteria:

- Activation condition: $\hat{H}_t < H_{\min}$ (typically $H_{\min} = 0.25$)
- Kernel weights: $w_A = w_B = w_C = 0$, $w_D = 1.0$
- Cost function switch: Wasserstein $\to$ Huber with $\delta =$ default
- Emit `CriticalSingularityEvent`
- Maintain state until $\hat{H}_t > H_{\min} + \varepsilon_{\text{hysteresis}}$
- Hysteresis prevents oscillation

4 I/O and Persistence Tests

4.1 Category: Snapshot Protocol and Atomicity

4.1.1 TESTING POLICY #20: Hot-Start State Continuity

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 469-485

Statement: Serialize meta-state Ξ_t, restart system, load it. First prediction post-restart must match uninterrupted prediction.

Criteria:

- Full state capture: $\{w_i\}$, $\{\theta_i^*\}$, H_t, Sig_t, $G_t^{\pm}$, μ_t, σ_t^2
- Prediction parity: $|\hat{y}_{\text{original}} - \hat{y}_{\text{restored}}| < 10^{-12}$
- Bit-exact agreement on all state variables
- No numerical drift from save/load cycle

4.1.2 TESTING POLICY #21: Checksum Validation (Cryptographic Integrity)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 487-500

Statement: Corrupt snapshot file bit and verify SHA-256 validation rejects load, forcing cold start.

Criteria:

- Each snapshot includes SHA-256 hash
- Load: compute $H' = \text{SHA256(content)}$, compare to stored H
- If $H' \neq H$: reject load, emit `CorruptedSnapshotEvent`, cold start
- Single bit corruption reliably detected
- No false negatives in integrity check

4.1.3 TESTING POLICY #22: Write Interruption (Atomicity via Write-Then-Rename)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 510-547

Statement: Simulate power loss during snapshot serialization and verify partially written files handled safely.

Criteria:

- Three-step protocol: write to temp → `fsync()` → rename
- Atomic rename: `snapshot_{timestamp}.tmp` → `snapshot_{timestamp}.bin`
- Interruption handling at progress $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$
- Recovery: detect missing `.bin`, ignore corrupted `.tmp`, load previous valid
- Recovery time $<$ 30 seconds
- No infinite restart loops

4.1.4 TESTING POLICY #23: Silent Disk Corruption Detection

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 549-577

Statement: Detect and handle silent data corruption (bit rot) between snapshot write and read.

Criteria:

- Verification metadata: SHA-256 hash, creation timestamp, format version, CRC32
- Double-check: CRC32 if available, then full SHA-256
- Fallback: if current corrupt, search for previous valid snapshots
- Retention: keep last $N = 5$ valid snapshots
- Automatic recovery from t_{-k} if t_0 corrupt

4.1.5 TESTING POLICY #24: Disk Space Exhaustion Handling

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 579-605
Statement: Validate behavior when storage full during snapshot write.
Criteria:

- Pre-check free space: FreeSpace $\geq 2 \times$ EstimatedSize(Ξ_t)
- If insufficient: emit `InsufficientStorageEvent`, don't write
- Continue operation in memory until space available
- Mid-write failure: catch I/O exception, delete temp, preserve last valid
- Graceful degradation vs crash

5 Adaptive and Topological Robustness

5.1 Category: Dynamic Regularization and Level 4 Autonomy

5.1.1 TESTING POLICY #25: Dynamic Regularization Under Volatility Shocks

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 613-658

Statement: Validate dynamic regularization equation: $\varepsilon_t = \max(\varepsilon_{\min}, \varepsilon_0(1 + \alpha\sigma_t))$.

Criteria:

- Baseline volatility $\sigma_0 = 0.01$, then shock $\sigma_t = 100 \times \sigma_0$
- Convergence without divergence to NaN/Inf
- Wasserstein distance stability: $W_2(\mu, \nu) \leq C \times (1 + \sigma_t)$
- Log-domain arithmetic throughout
- Regularization lower bound: $\varepsilon_t \geq \varepsilon_{\min}$ at all iterations

5.1.2 TESTING POLICY #26: Kurtosis-Coupled CUSUM Adaptive Threshold

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 667-729

Statement: Validate Adaptive Threshold Lemma: $h_t = h_0 \times (1 + \beta \times (\kappa_t - 3)/(\kappa_0 - 3))$ adapts correctly.

Criteria:

- Phase 1 (Gaussian): $\kappa \approx 3$, $h_t \approx h_0$
- Phase 2 (Student-t $\nu = 3$): $\kappa \approx 9$, $h_t \approx 2h_0$
- Type I Error constancy: $|\text{FPR}_{\text{Phase1}} - \text{FPR}_{\text{Phase2}}| < 0.05$
- Threshold scaling: $h_{t=1500}/h_{t=500} \in [1.5, 2.5]$ for $\beta = 0.5$
- No spurious alarms during distribution shifts
- Grace period between threshold adaptations

5.1.3 TESTING POLICY #27: Entropy-Driven Capacity Expansion (DGM Architecture Scaling)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 731-787

Statement: Validate Entropy-Topology Coupling: $\log(W \times D) \geq \log(W_0 \times D_0) + \beta \times \log(\kappa)$.

Criteria:

- Mode collapse with fixed architecture ($W = 64$, $D = 4$): entropy loss $< \gamma \times H[g]$
- Entropy preservation with adaptive: $H_{\text{solution}} \geq 0.9 \times H[g']$
- Capacity scaling: $W_{\text{new}} \times D_{\text{new}} \geq (W_0 \times D_0) \times \kappa^{\beta}$ with $\beta \in [0.5, 1.0]$
- XLA recompilation budget: ≤ 1 per regime transition
- Cache hit rate $> 95\%$ after warmup

5.1.4 TESTING POLICY #28: Meta-Optimization Determinism (TPE Checkpoint Persistence)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 792-856

Statement: Validate TPE State Persistence ensuring bit-exact resumption after interruption.

Criteria:

- Uninterrupted run 50 trials $\rightarrow$ best parameters θ^*_A, best objective f^*_A
- Interrupted run: save at trial 25, resume $\rightarrow$ best θ^*_B, f^*_B
- Bit-exact equivalence: $\theta^*_A = \theta^*_B$ (all 14 hyperparameters)
- Objective parity: $|f^*_A - f^*_B| < 10^{-12}$

- Trial history identical: all 50 trials match
- PRNG state preserved: trial 26 identical after resumption
- SHA-256 integrity verification
- Failure modes: hash mismatch, missing sidecar, study name mismatch

6 Hardware Parity and Cross-Platform Tests

6.1 Category: Bit-Consistency and Numerical Parity

6.1.1 TESTING POLICY #29: Multi-Architecture Equivalence (CPU/GPU/FPGA)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 867-880

Statement: Verify critical algorithms produce equivalent results across architectures within precision limits.

Criteria:

- Platforms: CPU (IEEE 754 64-bit), GPU (32/64-bit), FPGA (fixed-point)
- Relative difference bounds: $\varepsilon_{\text{GPU}} = 10^{-6}$, $\varepsilon_{\text{FPGA}}$ = quantization error
- Components tested: random generation, signatures, SDE integration
- Error metrics: $(\|x_{\text{CPU}} - x_{\text{GPU}}\|_2)/\|x_{\text{CPU}}\|_2$

6.1.2 TESTING POLICY #30: Fixed-Point Error Accumulation (Branch D FPGA)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 888-941

Statement: Compare Branch D signatures on FPGA (fixed-point) vs CPU (64-bit floating-point).

Criteria:

- Primary: $\Delta_{\text{accum}} \leq N \times \varepsilon_{\text{quant}}$ after 10,000 iterations
- Norm preservation: relative error $< 1\%$
- Sign preservation: $\text{sgn}(s_i^{(k)}, \text{CPU}) = \text{sgn}(s_i^{(k)}, \text{FPGA})$ for all i, k
- Angular distance: $\cos(\theta) > 0.9999$ ($< 0.81°$ deviation)
- Relative error per level: $\tau_k = 0.05 \times k$
- Topological properties preserved

6.1.3 TESTING POLICY #31: Deterministic Reproducibility (Controlled Seed Initialization)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 943-950

Statement: Guarantee identical state sequences across platforms given same PRNG seed.

Criteria:

- Seed s_0 initialization
- 1,000 simulation steps
- Bit-for-bit equality for same-representation platforms
- Fixed-point to float conversion for FPGA comparison

6.1.4 TESTING POLICY #32: Cross-Platform Performance Benchmark

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 952-968

Statement: Measure execution time and throughput across architectures to identify bottlenecks.

Criteria:

- GPU performance: $T_{\text{GPU}} < 0.3 \times T_{\text{CPU}}$
- FPGA performance: $T_{\text{FPGA}} < 0.1 \times T_{\text{CPU}}$
- Throughput measurement: predictions/second
- Batch size $N = 1{,}000$

7 Final Validation Protocol (Causality)

7.1 Category: Generalization and Temporal Integrity

7.1.1 TESTING POLICY #33: Rolling Walk-Forward (Zero Look-Ahead Bias)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 975-990

Statement: Ensure training uses only data strictly prior to test horizon.

Criteria:

- For each test window T_k, train only with $D^k_{\text{train}} = \{(t_i, y_i) : t_i < t_{nk}\}$
- Rolling window advancement
- No future data used at time t
- Out-of-sample metrics aggregation (RMSE, MAE, Sharpe)

7.1.2 TESTING POLICY #34: Bayesian Optimization Efficiency

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 992-1004

Statement: Iterative hyperparameter improvement via Gaussian Process must beat random search.

Criteria:

- Expected Improvement (EI) vs random sampling
- Acceptance: $\min_i L(\theta_i^{\text{BO}}) < \min_i L(\theta_i^{\text{random}})$
- Significance: Mann-Whitney p-value < 0.05

7.1.3 TESTING POLICY #35: Temporal Integrity (TTL Staleness Metric)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 1006-1018

Statement: Cancel JKO update if target signal delay exceeds Δ_{max}.

Criteria:

- Time-to-live: $\text{TTL}(y_t) = t_{\text{current}} - t$
- If $\text{TTL}(y_t) > \Delta_{\text{max}}$: discard signal
- NOT perform JKO update
- Emit `StaleDataEvent`
- Typical $\Delta_{\text{max}} = 5$ seconds

7.1.4 TESTING POLICY #36: Degraded Inference Mode (Lag Injection)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 1020-1063

Statement: Artificially delay y_{target} beyond Δ_{max} and validate degraded mode activation and JKO suspension.

Criteria:

- TTL injection: $\text{TTL}(\tilde{y}_t) = \Delta_{\text{max}} + \delta$ ($\delta > 0$)
- Detect TTL violation
- Activate `DegradedInferenceMode` = True
- Suspend JKO transport
- Freeze orchestrator weights at last valid value
- Emit `StaleDataEvent` + `DegradedInferenceModeActivated`
- Detection time < 100 ms
- Predictions continue with frozen configuration
- Recovery: restore when $\text{TTL} < 0.8 \times \Delta_{\text{max}}$

8 Edge Cases and Operational Limits

8.1 Category: Boundary Conditions and Extreme Scenarios

8.1.1 TESTING POLICY #37: CUSUM Adaptive Dynamic Threshold (Volatility Adaptation)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 1071-1104

Statement: Validate CUSUM threshold adapts correctly to low/high volatility via $h_t = k \times \sigma_{\text{resid},t}$.

Criteria:

- Low volatility: $h = k \times 0.01$, detects small drifts
- High volatility: $h = k \times 0.50$, threshold scales proportionally
- Transition: smooth rolling window adaptation
- No spurious activations during volatility shifts
- Threshold ratio matches volatility ratio within 10%

8.1.2 TESTING POLICY #38: Maximum Entropy Convergence (Uniform Weights)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 1106-1131

Statement: Confirm convergence to uniform weights $[0.25, 0.25, 0.25, 0.25]$ as Sinkhorn $\varepsilon \to \infty$.

Criteria:

- Entropy regularization: $\varepsilon_k = 10^k$ for $k \in \{0, 1, 2, 3, 4\}$
- Limit: $\lim_{\varepsilon\to\infty}\{w_i\} = \{0.25, 0.25, 0.25, 0.25\}$
- Numerical criterion $\varepsilon = 10^4$: $\max_i |w_i - 0.25| < 0.01$
- Reflects maximum uncertainty principle (Jaynes)

8.1.3 TESTING POLICY #39: Path Reparametrization Invariance (Time Warping)

Source: `Stochastic_Predictor_Test_Cases.tex`, Lines 1133-1161

Statement: Test signal and time-stretched variants; rough path signature must be identical under reparametrizations.

Criteria:

- Reparametrization functions: $\varphi_1(t) = t^2$, $\varphi_2(t) = \sqrt{t}$, $\varphi_3(t) = \frac{1}{2}(1 - \cos(\pi t))$
- Signature invariance: $\|S_i - S_0\|_2 < 10^{-8}$
- Negative control: non-monotone φ produces different signature
- Captures intrinsic path geometry independent of execution speed

8.1.4 TESTING POLICY #40: Extreme Kurtosis Handling (Kurtosis > 20)

Source: `Stochastic_Predictor_Tests_Python.tex`, Lines 1480-1510

Statement: Kurtosis > 20 must generate critical alert and trigger adaptive threshold elevation.

Criteria:

- Detection: $\kappa > 15.0$
- Emit critical alert
- Adaptive threshold elevated: $h_{\text{adapt}} > 2.0 \times h_{\text{fixed}}$
- CUSUM remains calibrated despite heavy tails
- False positive control maintained

8.1.5 TESTING POLICY #41: Degraded Mode with TTL Violation (Operational Limits)

Source: `Stochastic_Predictor_Tests_Python.tex`, Lines 1424-1463

Statement: TTL counter exceeds limit $\rightarrow$ activate degraded mode, freeze weights, suspend JKO.

Criteria:

- Degraded mode flag: True when staleness detected
- JKO transport: SUSPENDED
- Weights: frozen at last valid value
- Hysteresis recovery: $0.8 \times \mathrm{TTL}_{\mathrm{max}}$ threshold for reactivation
- Predictions continue using frozen configuration

9 XLA VRAM and JIT Cache Assertions

9.1 Category: JAX Compilation and Asynchronous Dispatch

9.1.1 TESTING POLICY #42: Prevention of Host-Device Synchronization

Source: `Stochastic_Predictor_Tests_Python.tex`, Lines 1885-1950

Statement: Ensure orchestration returns unbacked DeviceArray objects without forcing host synchronization.

Criteria:

- Prediction type: `jax.Array` or `jnp.ndarray` (never Python `float`)
- Valid `.device()` attribute indicating XLA backend placement
- No explicit/implicit conversion to host types (`float()`, `.item()`, `.tolist()`)
- Telemetry uses `jax.lax.stop_gradient()` on diagnostic metrics
- Performance impact avoidance: prevents 100-500ms latency spikes per sync

9.1.2 TESTING POLICY #43: Vectorized Multi-Tenancy Bit-Exactness

Source: `Stochastic_Predictor_Tests_Python.tex`, Lines 1956-2022

Statement: Validate batched `jax.vmap` execution bit-exact to sequential loop execution for multi-tenant workloads.

Criteria:

- Sequential vs vectorized batch size $N = 128$
- Bit-exact prediction parity: `numpy.array_equal()`
- State update equivalence across all clients
- PRNG state advancement consistency
- Memory scaling sub-linear (config sharing prevents N-fold duplication)
- Compilation time: first call may be slow, subsequent < 5ms

9.1.3 TESTING POLICY #44: Load Shedding Without XLA Recompilation

Source: `Stochastic_Predictor_Tests_Python.tex`, Lines 2030-2079

Statement: Swapping Kernel D signature depths ($M \in \{2, 3, 5\}$) executes in $O(1)$ without cache miss.

Criteria:

- Warmup phase precompiles all depths $M \in \{2, 3, 5\}$
- Load shedding execution: < 10ms (cached)
- No recompilation (avoids 200ms stall)
- Cache hit rate $\geq 99\%$ after warmup
- Memory overhead: ≤ 3 entries per variant
- Failure mode test: verify early warmup prevents latency spikes

9.1.4 TESTING POLICY #45: Atomic TOML Mutation (POSIX Guarantees)

Source: `Stochastic_Predictor_Tests_Python.tex`, Lines 2084-2136

Statement: Ensure config mutation compliance with POSIX atomic write semantics.

Criteria:

- Three-step protocol: write tmp $\rightarrow$ `fsync()` $\rightarrow$ `os.replace()`
- `os.replace()` called with temp file as arg 1, target as arg 2
- Concurrent mutations detected and rejected (temp exists)
- Audit trail in `io/mutations.log` (JSON Lines format)

- Rollback capability: `config.toml.bak` backup created
- POSIX atomicity prevents partial config visibility

10 Summary Matrix

10.1 Test Classification and Coverage

Category	Count	Test Type	Coverage Target
Random Generation & Entropy	2	Unit	95%
Wavelet Analysis (WTMM)	4	Unit	92%
Algebraic Structures (Branch D)	2	Unit	90%
SDE Solvers	2	Integration	91%
Transport & Orchestration	2	Integration	93%
DGM/HJB	5	Integration	88%
Robustness & Circuit Breaking	3	Robustness	89%
I/O & Persistence	5	I/O	97%
Dynamic Adaptation	4	Feature	91%
Hardware Parity	4	Cross-platform	85%
Causality & Validation	4	Acceptance	92%
Operational Limits	3	Edge Case	88%
XLA/JAX Specific	4	Performance	89%
TOTAL	**45**	**Mixed**	**91%**

10.2 Acceptance Criteria Summary

10.2.1 Global Requirements

- **Code Coverage:** $\geq 90\%$ in all critical modules
- **Pass Rate:** 100% before merge
- **Performance:** Full suite < 5 minutes (no GPU, no Optuna)
- **Reproducibility:** Fixed-seed tests produce identical results
- **Numerical Parity:** CPU vs GPU error $< 10^{-5}$ (float32)

10.2.2 Mathematical Guarantees

- **Convergence:** Proven for all SDE and optimization schemes
- **Stability:** No NaN/Inf propagation under stress
- **Causality:** Zero look-ahead bias verified
- **Atomicity:** I/O operations POSIX-atomic
- **Determinism:** Bit-exact reproducible with fixed seed

10.2.3 Operational SLAs

- **Latency (p99):** < 50ms per prediction
- **Throughput:** $\geq 1{,}000$ predictions/second
- **Reliability:** 99.95% uptime
- **Recovery Time:** < 30 seconds from any failure
- **Data Integrity:** 100% snapshot fidelity

11 Policy Application Workflow

11.1 Phase 1: Pre-Merge Validation

1. Run unit tests (fast, < 1 min)
2. Check 90% coverage threshold
3. Fix VSCode errors
4. Commit with policy reference codes

11.2 Phase 2: Integration Testing

1. Run integration tests (3-5 min)
2. Validate cross-platform parity
3. Confirm causality invariants
4. Generate coverage reports

11.3 Phase 3: Production Deployment

1. Full walk-forward validation
2. Hardware parity confirmation
3. Snapshots integrity check
4. Meta-optimizer determinism verify

12 Documentation References

Related Specifications:

- `doc/latex/tests/code_audit_policies.tex` (36 policies)
- Mathematical specification: `doc/latex/specification/`
- Implementation guide: `Python/`

Key Theorems Underlying Tests:

- Chen's identity (algebra)
- Crandall-Lions viscosity theory (PDE)
- Universal approximation (neural networks)
- Jaynes maximum entropy principle
- POSIX atomic operations specification

Document Approved For: Implementation
Last Updated: 2026-02-20
Maintainer: Adaptive Meta-Prediction Development Consortium