# Protocolo de Validación y Pruebas del Predictor Estocástico Universal

Consorcio de Desarrollo de Meta-Predicción Adaptativa

19 de febrero de 2026

# Índice

# Capítulo 1

# Pruebas de Unidad: Núcleos y Algoritmos Fundamentales

Estas pruebas verifican la implementación aislada de los algoritmos críticos sin depender del estado global del sistema.

## 1.1 Generación de Entropía y Variables Aleatorias

### 1.1.1 Algoritmo CMS

**Caso de Prueba 1.1** (Validación de Distribuciones $\alpha$-Estables)**.** *Validar que la generación de variables $\alpha$-estables mediante el método de Chambers-Mallows-Stuck produzca distribuciones con los parámetros $(\alpha, \beta, \gamma, \delta)$ deseados.*

**Criterio 1.1.** *Para una muestra de tamaño $N \geq 10^4$, los momentos empíricos deben coincidir con las propiedades teóricas de la distribución estable dentro de un intervalo de confianza del 95%.*

### 1.1.2 Integridad de Generadores Pseudo-Aleatorios

**Caso de Prueba 1.2** (Mersenne Twister/PCG64)**.** *Verificar la ausencia de correlaciones seriales y el cumplimiento de periodos largos en el generador de números pseudo-aleatorios.*

**Criterio 1.2.** *Aplicar las baterías de pruebas estadísticas estándar (TestU01, Diehard) y verificar que no se detecten fallos en los tests de aleatoriedad. El periodo del generador debe ser $\geq 2^{127}$ para Mersenne Twister y $\geq 2^{128}$ para PCG64.*

## 1.2 Análisis de Singularidad (SIA)

### 1.2.1 Detección del Exponente de Hölder

**Caso de Prueba 1.3** (Validación de WTMM)**.** *Utilizar señales sintéticas con exponente de Hölder conocido ($H$ determinado) para validar que el algoritmo WTMM (Wavelet Transform Modulus Maxima) recupere el espectro de singularidades $D(h)$ con un error $< 5\%$.*

**Criterio 1.3.** *Sea $f(t)$ una señal sintética con $H = H_0$ conocido. El exponente de Hölder estimado mediante WTMM debe satisfacer:*

$$|\hat{H} - H_0| < 0.05 \cdot H_0$$

*donde $\hat{H}$ es el valor estimado a través del análisis multiescala.*

### 1.2.2 Cono de Influencia

**Caso de Prueba 1.4** (Radio de Influencia Besov)**.** *Verificar que el enlace de máximos locales en el espacio de escalas respete el radio de influencia definido por $C_{besov}$.*

**Criterio 1.4.** *Para dos máximos consecutivos en escalas $s_1 < s_2$, sus posiciones temporales $(t_1, t_2)$ deben satisfacer:*

$$|t_2 - t_1| \leq C_{besov} \cdot (s_2 - s_1)$$

*donde $C_{besov}$ es la constante del espacio de Besov asociado a la wavelet utilizada.*

### 1.2.3 Validación del Límite de Nyquist Suave

El documento de especificación de I/O establece una frecuencia mínima de inyección de datos para mantener la integridad del análisis multifractal. Este test valida explícitamente ese límite operacional.

**Caso de Prueba 1.5** (Aliasing Multifractal)**.** *Reducir gradualmente la frecuencia de los datos de entrada hasta que el espectro $D(h)$ colapse, validando que el sistema detecte la condición de submuestreo antes de que se produzca degradación irreversible de la estimación.*

**Criterio 1.5.** *Considerar una señal multifractal de referencia con espectro de singularidades conocido $D_0(h)$ y frecuencia de muestreo nominal $f_0$. Realizar el siguiente protocolo:*

1. *Reducir la frecuencia de muestreo mediante diezmado: $f_k = f_0/2^k$ con $k = 0, 1, 2, \ldots$*
2. *Para cada $f_k$, calcular el espectro estimado $\hat{D}_k(h)$ mediante WTMM*
3. *Calcular el error relativo de forma:*

$$\varepsilon_k = \frac{\|D_0(h) - \hat{D}_k(h)\|_{L^2}}{\|D_0(h)\|_{L^2}}$$

4. *Monitorear el exponente de Hölder dominante: $\hat{H}_k = \arg\max_h D_k(h)$*

*El sistema debe cumplir:*

- ***Detección preventiva:*** *Emitir señal* `FreezingTopologicalBranchEvent` *cuando $\varepsilon_k > 0.05$ (error $> 5\%$)*
- ***Criterio de aceptación:*** *La señal debe activarse antes de que el error en la estimación de Hölder supere el $10\%$:*

$$\frac{|\hat{H}_k - H_0|}{H_0} < 0.10 \quad \textit{en el momento de la congelación}$$

- ***Acción correctiva:*** *Tras la señal, el peso de la Rama Topológica debe fijarse en su último valor confiable: $w_C \to w_C^{frozen}$ sin actualizaciones dinámicas*

**Nota 1.1.** *El límite de Nyquist en análisis multifractal no es una frontera rígida como en el muestreo de señales periódicas. La naturaleza autosimilar de los procesos multifractales permite cierta tolerancia al submuestreo, pero existe un soft-limit por debajo del cual las estructuras de escala fina colapsan y el espectro $D(h)$ se degrada. Este test garantiza que el sistema opere siempre en el régimen de resolución adecuada.*

**Nota 1.2.** *La frecuencia crítica $f_{critical}$ depende de:*

1. *El rango de escalas $[s_{min}, s_{max}]$ de la transformada wavelet*

2. *El soporte temporal de la wavelet madre $\psi(t)$*

3. *La regularidad de Hölder mínima esperada $H_{min}$*

*Como regla heurística, se recomienda:*

$$f_{min} \geq \frac{10}{s_{min}} \cdot (1 + H_{min}^{-1})$$

*Este criterio asegura al menos 10 puntos por escala mínima, ajustados por la rugosidad del proceso.*

## 1.3 Estructuras Algebraicas (Rama D)

### 1.3.1 Identidad de Chen

**Caso de Prueba 1.6** (Concatenación de Signaturas)**.** *Validar que la concatenación de signaturas de dos segmentos de camino mediante el producto tensorial ($\otimes$) sea igual a la signatura del camino completo.*

**Criterio 1.6.** *Sean $\gamma_1 : [0, T_1] \to \mathbb{R}^d$ y $\gamma_2 : [0, T_2] \to \mathbb{R}^d$ dos caminos. La identidad de Chen establece:*

$$Sig(\gamma_1 \star \gamma_2) = Sig(\gamma_1) \otimes Sig(\gamma_2)$$

*donde $\gamma_1 \star \gamma_2$ denota la concatenación de caminos. El error numérico debe ser $< 10^{-6}$ en norma euclidiana.*

### 1.3.2 Invariancia de Escala

**Caso de Prueba 1.7** (Reparametrización Temporal)**.** *Comprobar que la signatura truncada al nivel $M$ sea invariante ante reparametrizaciones temporales del camino de entrada.*

**Criterio 1.7.** *Para una reparametrización estrictamente creciente $\phi : [0, 1] \to [0, 1]$ con $\phi(0) = 0$ y $\phi(1) = 1$, se debe cumplir:*

$$Sig^{(M)}(\gamma) = Sig^{(M)}(\gamma \circ \phi)$$

*donde $Sig^{(M)}$ denota la signatura truncada al nivel $M$.*

# Capítulo 2

# Pruebas de Integración y Convergencia Estocástica

Validan que la interacción entre componentes respete las leyes de la probabilidad continua y las condiciones de estabilidad numérica.

## 2.1 Solvers de Ecuaciones Diferenciales Estocásticas (SDE)

### 2.1.1 Convergencia de Esquemas Numéricos

**Caso de Prueba 2.1** (Euler-Maruyama vs. Milstein)**.** *Para una difusión con volatilidad no constante, verificar que el esquema de Milstein logre una convergencia fuerte de orden* 1.0 *frente al orden* 0.5 *de Euler-Maruyama.*

**Criterio 2.1.** *Considerar la EDE:*

$$dX_t = \mu(X_t, t)\, dt + \sigma(X_t, t)\, dW_t$$

*con $\sigma(x,t)$ no constante. Para una secuencia de pasos temporales $\Delta t_n = 2^{-n}\Delta t_0$, el error fuerte debe satisfacer:*

$$E[|X_T - X_T^{EM}|^2] = O(\Delta t^{0.5}) \quad \textit{(Euler-Maruyama)}$$
$$E[|X_T - X_T^{M}|^2] = O(\Delta t^{1.0}) \quad \textit{(Milstein)}$$

*donde $X_T^{EM}$ y $X_T^M$ son las aproximaciones numéricas.*

### 2.1.2 Condición CFL Mixta

**Caso de Prueba 2.2** (Violación de Estabilidad)**.** *Forzar un paso de tiempo $\Delta t$ que viole la restricción de Courant-Friedrichs-Lewy y confirmar la aparición de inestabilidad numérica o valores no numéricos (NaN) como comportamiento esperado para calibrar el monitor de seguridad.*

**Criterio 2.2.** *Para un esquema explícito, la condición CFL requiere:*

$$\Delta t \leq \frac{C}{\sup_x |\mu'(x)| + \sigma^2(x)/2}$$

*Forzar $\Delta t > 10C$ y verificar que el sistema detecte la divergencia mediante:*

- *Aparición de NaN o Inf en las trayectorias simuladas*
- *Emisión de alerta de inestabilidad por el módulo de seguridad*

## 2.2 Optimización del Transporte (Orquestador)

### 2.2.1 Estabilidad del Algoritmo de Sinkhorn

**Caso de Prueba 2.3** (Convergencia en Dominio Logarítmico)**.** *Evaluar la convergencia del algoritmo de Sinkhorn en el dominio logarítmico frente a un parámetro de regularización $\varepsilon$ decreciente, asegurando que no ocurra underflow hasta el límite de $\varepsilon \geq 10^{-4}$.*

**Criterio 2.3.** *El algoritmo de Sinkhorn-Knopp en escala logarítmica debe converger cuando:*

$$\varepsilon \geq 10^{-4}$$

*Para $\varepsilon < 10^{-4}$, el sistema debe detectar riesgo de underflow numérico y emitir advertencia. La convergencia se mide mediante:*

$$\|K\,diag(u)K^T\,diag(v) - \mu\|_1 < 10^{-6}$$

*donde $K_{ij} = \exp(-C_{ij}/\varepsilon)$ es el kernel de Gibbs.*

### 2.2.2 Normalización del Simplex

**Caso de Prueba 2.4** (Conservación de Masa Probabilística)**.** *Confirmar que, tras cualquier actualización del flujo JKO (Jordan-Kinderlehrer-Otto), la suma de los pesos de los núcleos sea estrictamente $\sum_i \rho_i = 1.0$.*

**Criterio 2.4.** *Después de cada iteración del esquema JKO:*

$$\rho^{n+1} = \arg\min_{\rho \in \mathcal{P}(\mathcal{X})} \left\{ W_2^2(\rho, \rho^n) + \tau \mathcal{F}[\rho] \right\}$$

*se debe verificar que $\rho^{n+1}$ pertenece al simplex probabilístico:*

$$\sum_{i=1}^{N} \rho_i^{n+1} = 1.0, \quad \rho_i^{n+1} \geq 0 \quad \forall i$$

*con tolerancia numérica $|\sum_i \rho_i^{n+1} - 1.0| < 10^{-10}$.*

## 2.3 Solución de la Ecuación HJB mediante DGM (Rama B)

La Rama B del predictor estocástico utiliza el Deep Galerkin Method (DGM) para resolver la ecuación de Hamilton-Jacobi-Bellman asociada a problemas de control óptimo estocástico. Estas pruebas validan la convergencia y estabilidad del método neuronal.

### 2.3.1 Estabilidad de Gradiente

**Caso de Prueba 2.5** (Explosión de Gradiente en Alta Volatilidad)**.** *Monitorear la norma de los gradientes durante el entrenamiento de la PDE para detectar explosiones de gradiente en condiciones de alta volatilidad del proceso estocástico subyacente.*

**Criterio 2.5.** *La ecuación HJB en su forma general:*

$$\frac{\partial V}{\partial t} + \sup_{u \in \mathcal{U}} \{\mathcal{L}^u V(x,t) + f(x,u,t)\} = 0$$

*se aproxima mediante una red neuronal $V_\theta(x,t)$ entrenada con el método DGM. Durante el entrenamiento, se debe monitorear:*

1. *La norma del gradiente de la pérdida respecto a los pesos de la red:*

$$\|\nabla_\theta \mathcal{L}_{DGM}(\theta)\|_2 = \left\| \frac{\partial}{\partial \theta} \mathbb{E}[|PDE[V_\theta]|^2 + |BC[V_\theta]|^2] \right\|_2$$

2. *Aplicar gradient clipping cuando* $\|\nabla_\theta \mathcal{L}\|_2 > C_{clip}$ *con* $C_{clip} = 10.0$

3. *En escenarios de alta volatilidad* ($\sigma(x,t) > 2\sigma_0$)*, el sistema debe:*

   - *Detectar si* $\|\nabla_\theta \mathcal{L}\|_2$ *excede* $C_{clip}$ *durante más de 5 iteraciones consecutivas*
   - *Emitir* `GradientInstabilityEvent`
   - *Reducir adaptativamente la tasa de aprendizaje:* $\eta \to 0.5\eta$
   - *Verificar estabilización dentro de las siguientes 20 épocas*

**Nota 2.1.** *Las explosiones de gradiente en DGM suelen originarse en la interacción entre los términos de derivadas de segundo orden (curvatura de la solución) y los coeficientes de difusión. La detección temprana y el clipping adaptativo son esenciales para la convergencia en régimenes de alta volatilidad.*

### 2.3.2 Validación de Soluciones de Viscosidad

**Caso de Prueba 2.6** (Principio de Comparación de Crandall-Lions)**.** *Validar que la solución de la red neuronal* $V_\theta(x,t)$ *respete el principio de comparación para soluciones de viscosidad, garantizando unicidad y consistencia con la teoría de PDEs no lineales.*

**Criterio 2.6.** *El principio de comparación establece que, si* $u$ *es una supersolución de viscosidad y* $v$ *es una subsolución de viscosidad de la misma ecuación HJB con* $u \geq v$ *en la frontera, entonces* $u \geq v$ *en todo el dominio.*

*Para validar numéricamente este principio:*

1. *Construir una solución de referencia* $V_{ref}(x,t)$ *mediante un método establecido (esquema de diferencias finitas upwind, método de líneas)*

2. *Entrenar la red DGM para obtener* $V_\theta(x,t)$

3. *Verificar monotonicidad en el tiempo (para problemas de horizonte finito):*

$$V_\theta(x,t_1) \geq V_\theta(x,t_2) \quad \forall t_1 < t_2, \forall x \in \mathcal{D}$$

   *(suponiendo costos no negativos)*

4. *Validar la propiedad de subsupersolución en una malla de puntos de test* $\{(x_i,t_j)\}_{i,j}$*:*

$$\begin{aligned} \textit{Subsolución:} \quad & PDE[V_\theta](x_i,t_j) \leq \epsilon_{tol} \\ \textit{Supersolución:} \quad & PDE[V_\theta](x_i,t_j) \geq -\epsilon_{tol} \end{aligned}$$

   *con tolerancia* $\epsilon_{tol} = 10^{-3}$

5. *Comparar con solución de referencia:*

$$\|V_\theta - V_{ref}\|_{L^\infty(\mathcal{D})} < \delta_{viscosity}$$

   *donde* $\delta_{viscosity} = 0.05 \cdot \|V_{ref}\|_{L^\infty}$ *(error relativo* $< 5\%$*)*

**Nota 2.2.** *La teoría de soluciones de viscosidad de Crandall-Lions es el marco riguroso para ecuaciones HJB no diferenciables en sentido clásico. Las redes neuronales DGM, al ser funciones suaves* ($C^\infty$)*, deben aproximar estas soluciones generalizadas. La validación mediante el principio de comparación es crítica para garantizar que la solución neuronal no exhibe artefactos espurios como oscilaciones no físicas o violaciones de causalidad.*

### 2.3.3 Test de Entropía de Entrenamiento (Colapso de Modo)

Durante el entrenamiento de redes neuronales para PDEs, existe el riesgo de que la red colapse a soluciones triviales (por ejemplo, una constante) que satisfacen formalmente la ecuación diferencial pero carecen de contenido informativo. Este test detecta y previene dicho comportamiento patológico.

**Caso de Prueba 2.7** (Detección de Colapso de Modo)**.** *Verificar que la red neuronal no sufra de colapso de modo, donde predice una solución constante o degenerada que satisface trivialmente la PDE pero no captura la estructura real de la solución.*

**Criterio 2.7.** *El colapso de modo ocurre cuando la red aprende una solución de varianza mínima (típicamente constante) que minimiza la pérdida de la PDE sin resolver realmente el problema de valor en la frontera.*

*Para detectarlo, se debe verificar la proporcionalidad entre la varianza de la solución neuronal y la condición terminal:*

1. *Calcular la varianza espacial de la solución en tiempo $t < T$:*

$$Var_x[V_\theta(x,t)] = \mathbb{E}_x[(V_\theta(x,t) - \bar{V}_t)^2]$$

   *donde $\bar{V}_t = \mathbb{E}_x[V_\theta(x,t)]$ es el valor medio sobre el dominio espacial*

2. *Calcular la varianza de la condición terminal (payoff):*

$$Var[g(\xi)] = \mathbb{E}_\xi[(g(\xi) - \bar{g})^2]$$

   *donde $g(\xi)$ es la condición de frontera terminal y $\xi$ representa los estados terminales relevantes*

3. *Verificar proporcionalidad:*

$$\kappa_{low} \leq \frac{Var_x[V_\theta(x,t)]}{Var[g(\xi)]} \leq \kappa_{high}$$

   *con umbrales típicos: $\kappa_{low} = 0.3$ y $\kappa_{high} = 1.2$*

4. *Si el ratio cae por debajo de $\kappa_{low}$: detectar colapso de modo*

5. *Monitorear la entropía diferencial de la distribución de valores predichos:*

$$H[V_\theta] = -\int p(v) \log p(v)\, dv$$

   *Una solución colapsada tendrá $H[V_\theta] \to -\infty$ (distribución delta)*

*Criterios de aceptación:*

- *El ratio de varianzas debe estar dentro de $[\kappa_{low}, \kappa_{high}]$ en al menos el 90% de los tiempos $t \in [0,T]$*
- *La entropía diferencial debe satisfacer: $H[V_\theta] > H_{min}$ con $H_{min}$ calibrado según la complejidad del problema*
- *La norma del gradiente de $V_\theta$ respecto a $x$ no debe colapsar:*

$$\mathbb{E}_x[\|\nabla_x V_\theta(x,t)\|_2] > \epsilon_{grad} > 0$$

  *(evitar soluciones planas espacialmente)*

**Nota 2.3.** *El colapso de modo es particularmente común en:*

1. *Problemas con condiciones de frontera uniformes o simétricas*

2. *Tasas de aprendizaje excesivamente altas que llevan a soluciones de mínima norma*

3. *Arquitecturas de red sub-dimensionadas que no pueden capturar la complejidad de la solución verdadera*

4. *Inicializaciones de pesos que sesgan hacia valores constantes*

*La detección temprana mediante monitoreo de varianza permite reinicializar el entrenamiento o ajustar hiperparámetros antes de que el colapso se consolide.*

**Nota 2.4.** *Desde el punto de vista de teoría del control, una solución colapsada corresponde a una política de control degenerada que no responde a las variaciones del estado. Matemáticamente, si* $V_\theta(x,t) \approx c$ *(constante), entonces la política óptima derivada* $u^*(x,t) = \arg\max_u\{\cdots + \nabla_x V \cdot b(x,u)\}$ *se vuelve indeterminada o trivial, perdiendo toda utilidad práctica.*

### 2.3.4 Test de Convergencia en Refinamiento de Malla

**Caso de Prueba 2.8** (Consistencia con Refinamiento)**.** *Verificar que la solución DGM converja a la solución exacta (o de referencia) cuando se incrementa la densidad de puntos de colocación en el dominio espacio-temporal.*

**Criterio 2.8.** *Para una secuencia de mallas anidadas* $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \mathcal{M}_3$ *con densidades* $N_1 < N_2 < N_3$ *de puntos de colocación:*

1. *Entrenar DGM en cada malla hasta convergencia de la pérdida:* $\mathcal{L}_{DGM}^{(k)} < 10^{-4}$

2. *Calcular el error en una malla fina de evaluación independiente:*

$$e_k = \|V_{\theta_k} - V_{ref}\|_{L^2(\mathcal{D}_{eval})}$$

3. *Verificar convergencia monótona:*

$$e_1 > e_2 > e_3$$

4. *Estimar la tasa de convergencia empírica:*

$$r = \frac{\log(e_1/e_2)}{\log(N_2/N_1)}$$

*Se espera* $r \geq 0.5$ *(convergencia al menos de orden* $N^{-0.5}$*)*

### 2.3.5 Test Simplificado de Varianza de Salida (Mode Collapse Detection)

Este test complementa el análisis de entropía de entrenamiento con un criterio cuantitativo directo sobre la varianza de la salida de la red neuronal, proporcionando una métrica de alerta temprana para colapso de modo.

**Caso de Prueba 2.9** (Umbral de Varianza Mínima)**.** *Comparar la varianza de la salida de la red neuronal* $V_\theta(x,t)$ *frente a una solución de referencia* $V_{ref}(x,t)$ *y verificar que la red capture al menos el* 10% *de la variabilidad de la solución verdadera.*

**Criterio 2.9.** *Sea* $V_{ref}(x,t)$ *una solución de referencia obtenida mediante un método establecido (diferencias finitas, elementos finitos, o solución analítica cuando esté disponible). Para la red DGM entrenada* $V_\theta(x,t)$*:*

1. *Calcular la varianza de la solución de referencia sobre el dominio espacial en un instante de tiempo* $t \in [0, T]$*:*

$$Var_{ref}(t) = \frac{1}{|\mathcal{X}|} \sum_{x_i \in \mathcal{X}} (V_{ref}(x_i, t) - \bar{V}_{ref}(t))^2$$

*donde* $\bar{V}_{ref}(t) = \frac{1}{|\mathcal{X}|} \sum_{x_i} V_{ref}(x_i, t)$ *es el valor medio.*

2. *Calcular la varianza de la salida neuronal:*

$$Var_{\theta}(t) = \frac{1}{|\mathcal{X}|} \sum_{x_i \in \mathcal{X}} (V_{\theta}(x_i, t) - \bar{V}_{\theta}(t))^2$$

3. *Evaluar el ratio de varianzas:*

$$R_{var}(t) = \frac{Var_{\theta}(t)}{Var_{ref}(t)}$$

4. ***Criterio de Fallo:*** *El test falla (detecta colapso de modo) si:*

$$R_{var}(t) < 0.10 \quad \text{para algún } t \in [0, 0.9T]$$

*Esto indica que la red ha colapsado a una solución de varianza mínima (típicamente constante o casi-constante) que no captura la estructura de la PDE.*

5. ***Criterio de Aceptación:*** *El test pasa si:*

$$R_{var}(t) \geq 0.10 \quad \forall t \in [0, 0.9T]$$

*y además:*

$$mediana_{t \in [0,T]} R_{var}(t) \geq 0.50$$

*garantizando que la red captura al menos el* 50% *de la variabilidad en la mayoría de los tiempos.*

**Nota 2.5.** *Este test es especialmente útil durante el entrenamiento iterativo:*

- *Si* $R_{var}(t) < 0.10$ *se detecta antes de la convergencia, se debe interrumpir el entrenamiento y ajustar hiperparámetros (tasa de aprendizaje, arquitectura de red, inicialización de pesos)*
- *El umbral del* 10% *es conservador y refleja que cualquier solución que capture menos del* 10% *de la variabilidad es prácticamente una constante*
- *La restricción a* $t \in [0, 0.9T]$ *excluye la vecindad de la condición terminal donde la varianza puede naturalmente reducirse al acercarse al payoff*

**Nota 2.6.** *Este criterio es complementario al test de entropía de la subsección anterior. Mientras que el test de entropía usa el ratio* $\kappa_{low} = 0.3$*, este test usa un umbral más estricto (*0.10*) para detección conservadora de colapsos severos. En la práctica:*

- $R_{var} < 0.10$*: colapso crítico (fallo inmediato)*
- $0.10 \leq R_{var} < 0.30$*: advertencia de varianza baja (requiere monitoreo)*
- $R_{var} \geq 0.50$*: operación normal*

# Capítulo 3

# Pruebas de Robustez y Circuit Breakers

Verifican la capacidad del sistema para protegerse ante anomalías del mercado o fallos en los datos.

## 3.1 Gestión de Outliers y Régimen

### 3.1.1 Inyección de Outliers

**Caso de Prueba 3.1** (Valores Atípicos Extremos)**.** *Introducir valores* $> 20\sigma$ *en el flujo de entrada y verificar que el sistema descarte el dato, emita la alerta de validación y mantenga el estado inercial de los pesos.*

**Criterio 3.1.** *Dado un flujo de observaciones* $\{y_t\}$ *con estadísticas móviles* $\mu_t$, $\sigma_t$, *inyectar una observación:*

$$\tilde{y}_t = \mu_t + 20\sigma_t$$

*El sistema debe:*

1. *Detectar que* $|\tilde{y}_t - \mu_t| > \theta\sigma_t$ *con* $\theta = 10$
2. *Rechazar la observación y NO actualizar el meta-estado* $\Xi_t$
3. *Emitir evento* `OutlierDetectedEvent` *con metadata del valor rechazado*
4. *Mantener los pesos* $\{w_i\}_{i=A}^{D}$ *sin cambios*

### 3.1.2 Disparo de CUSUM

**Caso de Prueba 3.2** (Cambio de Régimen Estructural)**.** *Simular un cambio de régimen (deriva estructural) y validar que el evento de cambio de régimen se emita exactamente cuando el estadístico* $G_t^+$ *exceda el umbral dinámico* $h$*.*

**Criterio 3.2.** *El detector CUSUM (Cumulative Sum) para detectar cambios en la media opera mediante:*

$$G_t^+ = \max(0, G_{t-1}^+ + (y_t - \mu_0) - k)$$

*donde* $k$ *es el slack de tolerancia y* $h$ *es el umbral de alarma. Se debe verificar:*

1. $G_t^+ > h \implies$ *emitir* `RegimeChangedEvent`
2. *Reiniciar* $G_t^+ = 0$ *tras la detección*
3. *La detección debe ocurrir con máximo* $\tau$ *observaciones de retardo desde el verdadero punto de cambio*

## 3.2 Modo de Emergencia (Postulado de Robustez)

### 3.2.1 Singularidad Crítica

**Caso de Prueba 3.3** (Régimen de Rugosidad Extrema)**.** *Reducir artificialmente el exponente de Hölder por debajo de $H_{min}$ y verificar que el Orquestador fuerce automáticamente el peso $w_D \to 1.0$ y cambie la función de costo a la métrica de Huber.*

**Criterio 3.3.** *Definir un umbral crítico $H_{min}$ (típicamente $H_{min} = 0.25$). Cuando el análisis WTMM detecta:*

$$\hat{H}_t < H_{min}$$

*el sistema debe:*

1. *Activar modo de emergencia: $w_A = w_B = w_C = 0$, $w_D = 1.0$*
2. *Cambiar la función de costo del Orquestador de Wasserstein a Huber:*

$$C(x, y) = \begin{cases} \frac{1}{2}|x - y|^2 & si\ |x - y| \leq \delta \\ \delta(|x - y| - \frac{\delta}{2}) & si\ |x - y| > \delta \end{cases}$$

3. *Emitir evento* `CriticalSingularityEvent`
4. *Mantener este estado hasta que $\hat{H}_t > H_{min} + \epsilon_{histeresis}$*

# Capítulo 4

# Pruebas de I/O y Persistencia

Garantizan la continuidad operativa y la integridad del estado latente.

## 4.1 Protocolo de Snapshots

### 4.1.1 Hot-Start

**Caso de Prueba 4.1** (Continuidad de Estado)**.** *Serializar el meta-estado $\Xi_t$, reiniciar el sistema y realizar una carga (Load). La primera predicción tras el reinicio debe ser idéntica a la que se habría generado sin interrupción.*

**Criterio 4.1.** *El meta-estado completo se define como:*

$$\Xi_t = \left\{ \{w_i\}_{i=A}^{D}, \{\theta_i^*\}_{i=A}^{D}, \mathcal{H}_t, Sig_t, G_t^{\pm}, \mu_t, \sigma_t^2 \right\}$$

*Procedimiento de validación:*

1. *En tiempo $t_0$, serializar $\Xi_{t_0}$ a archivo binario*
2. *Generar predicción $\hat{y}_{t_0+1}^{original}$*
3. *Reiniciar el sistema (liberar memoria)*
4. *Cargar $\Xi_{t_0}$ desde archivo*
5. *Generar predicción $\hat{y}_{t_0+1}^{restored}$*
6. *Verificar: $|\hat{y}_{t_0+1}^{original} - \hat{y}_{t_0+1}^{restored}| < 10^{-12}$*

### 4.1.2 Validación de Checksum

**Caso de Prueba 4.2** (Integridad Criptográfica)**.** *Corromper un solo bit en el archivo binario del snapshot y verificar que el sistema rechace la carga mediante el hash SHA-256, forzando un Cold-Start.*

**Criterio 4.2.** *Cada archivo de snapshot debe incluir un hash SHA-256 del contenido. El proceso de carga debe:*

1. *Leer el archivo binario*
2. *Calcular $H' = SHA256(contenido)$*
3. *Comparar con el hash almacenado $H$*
4. *Si $H' \neq H$: rechazar carga, emitir `CorruptedSnapshotEvent`, inicializar en modo Cold-Start*
5. *Si $H' = H$: proceder con la deserialización*

*Para validar, modificar 1 bit en posición aleatoria del archivo y verificar el rechazo.*

## 4.2 Recuperación Ante Fallos de I/O

La persistencia del estado $\Xi_t$ es crítica para la continuidad operativa. Estos tests validan la robustez del sistema ante fallos durante operaciones de escritura, lectura y almacenamiento.

### 4.2.1 Interrupción de Escritura (Atomicidad)

**Caso de Prueba 4.3** (Simulación de Caída de Energía)**.** *Simular una caída de energía o desconexión abrupta del sistema durante la serialización del estado $\Xi_t$, validando que el sistema maneje archivos parcialmente escritos sin entrar en estados inconsistentes.*

**Criterio 4.3.** *El protocolo de snapshotting debe garantizar atomicidad mediante el patrón de escritura temporal:*

1. *Serializar $\Xi_t$ a un archivo temporal:* `snapshot_{timestamp}.tmp`
2. *Calcular $H = SHA256(\Xi_t)$ y agregarlo al archivo*
3. *Realizar* `fsync()` *para forzar escritura a disco*
4. *Renombrar atómicamente:* `snapshot_{timestamp}.tmp` $\rightarrow$ `snapshot_{timestamp}.bin`

*Test de validación:*

1. *Iniciar proceso de snapshotting en tiempo $t_0$*
2. *Interrumpir el proceso en un punto aleatorio $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ del progreso de escritura*
3. *Simular terminación abrupta (enviar señal* `SIGKILL` *o cortar I/O)*
4. *Reiniciar el sistema*
5. *El sistema debe:*
   - *Detectar ausencia del archivo principal* `.bin` *(nunca fue renombrado)*
   - *Detectar presencia de archivo* `.tmp` *corrupto*
   - *Ignorar el archivo temporal (no intentar cargarlo)*
   - *Buscar el snapshot válido más reciente anterior a $t_0$*
   - *Si no existe snapshot previo válido: ejecutar* `ColdStartEvent`
   - *NO entrar en bucle de reinicio infinito*

*Criterio de éxito:*

$$\textit{Tiempo de recuperación} < T_{recovery} = 30 \textit{ segundos}$$

*Sin degradación de la funcionalidad del sistema tras la recuperación.*

**Nota 4.1.** *La atomicidad en la escritura de snapshots es fundamental para evitar el torn write problem, donde una escritura parcial deja el archivo en un estado inconsistente que no puede detectarse mediante checksum si la interrupción ocurre antes del cálculo del hash. El patrón write-to-temp-then-rename aprovecha la garantía de atomicidad que la mayoría de sistemas de archivos modernos (ext4, XFS, NTFS, APFS) ofrecen para la operación de renombrado.*

### 4.2.2 Corrupción Silenciosa de Disco

**Caso de Prueba 4.4** (Bit Rot y Errores de Almacenamiento)**.** *Detectar y manejar la corrupción silenciosa de datos en el almacenamiento persistente (bit rot, degradación de medios) que puede ocurrir entre el momento de escritura y el de lectura del snapshot.*

**Criterio 4.4.** *Aunque el snapshot se haya escrito correctamente, el medio de almacenamiento puede introducir errores posteriormente. El protocolo debe:*

1. *Almacenar metadata de verificación junto al snapshot:*
   - *Hash SHA-256 del contenido*
   - *Timestamp de creación*
   - *Versión del formato de serialización*
   - *Checksum CRC32 de validación rápida (opcional, pre-check antes de SHA-256)*
2. *Durante la carga:*
   - *Verificar CRC32 (si está disponible) como pre-filtro rápido*
   - *Verificar SHA-256 completo*
   - *Si falla verificación: marcar archivo como corrupto, buscar snapshot anterior*
3. *Implementar política de retención con redundancia:*
   - *Mantener los últimos $N = 5$ snapshots válidos*
   - *Permitir recuperación desde snapshot $t_{-k}$ si $t_0$ está corrupto*

*Test de validación:*

1. *Crear snapshot válido en $t_0$*
2. *Introducir corrupción en el archivo binario (modificar bytes aleatorios)*
3. *Intentar cargar el snapshot*
4. *Verificar que el sistema:*
   - *Detecte la corrupción mediante checksum*
   - *Emita `CorruptedSnapshotEvent` con metadata del archivo afectado*
   - *Busque snapshot alternativo en la cola de retención*
   - *Si encuentra snapshot válido anterior: cargarlo (con advertencia de pérdida de estado parcial)*
   - *Si no existe snapshot válido: ejecutar Cold-Start*
   - *NO entre en bucle infinito de verificación-fallo-reinicio*

### 4.2.3 Agotamiento de Espacio en Disco

**Caso de Prueba 4.5** (Manejo de Capacidad Insuficiente)**.** *Validar el comportamiento del sistema cuando el dispositivo de almacenamiento se queda sin espacio durante la escritura de un snapshot.*

**Criterio 4.5.** *Durante la operación de snapshotting, el sistema debe:*

1. *Antes de iniciar escritura: verificar espacio disponible*

$$Espacio\_libre \geq 2 \times Tamaño\_estimado(\Xi_t)$$

*(factor de 2 por seguridad para escritura temporal + renombrado)*

2. *Si no hay espacio suficiente:*
    - *Emitir* `InsufficientStorageEvent`
    - *NO intentar la escritura*
    - *Mantener el snapshot anterior válido*
    - *Continuar operación en memoria (sin persistencia) hasta que se libere espacio*
3. *Si el error de espacio ocurre durante la escritura (fallo de estimación):*
    - *Capturar excepción de I/O*
    - *Eliminar archivo temporal corrupto*
    - *Emitir* `SnapshotWriteFailedEvent`
    - *NO corromper el snapshot anterior válido*

*Test de validación:*

1. *Crear un volumen de almacenamiento con capacidad limitada*
2. *Llenar el volumen artificialmente dejando espacio insuficiente*
3. *Intentar crear snapshot*
4. *Verificar detección preventiva y manejo graceful sin crash*

**Nota 4.2.** *El agotamiento de espacio en disco es un escenario común en sistemas de producción. El diseño debe priorizar: (1) no corromper snapshots existentes válidos, (2) degradar gracefully manteniendo operación en memoria, (3) proveer telemetría clara para intervención del operador, (4) recuperar automáticamente cuando se libere espacio.*

# Capítulo 5

# Pruebas de Paridad y Fidelidad de Hardware (Cross-Platform)

Dado que el sistema contempla implementaciones heterogéneas (CPU, GPU y FPGA), es vital añadir tests de consistencia de bit para garantizar que las diferentes arquitecturas de hardware produzcan resultados matemáticamente equivalentes.

## 5.1 Tests de Consistencia de Bit

**Caso de Prueba 5.1** (Equivalencia Multi-Arquitectura)**.** *Verificar que los resultados numéricos de los algoritmos críticos sean consistentes entre diferentes plataformas de hardware (CPU, GPU, FPGA), dentro de los límites de precisión inherentes a cada arquitectura.*

**Criterio 5.1.** *Para cada componente crítico del sistema (generación de variables aleatorias, cálculo de signaturas, integración SDE), ejecutar el mismo conjunto de datos de entrada en:*

1. *CPU con aritmética de punto flotante IEEE 754 (64 bits)*
2. *GPU con aritmética de punto flotante (32 o 64 bits según disponibilidad)*
3. *FPGA con aritmética de punto fijo (precisión configurable)*

*La diferencia relativa entre implementaciones debe satisfacer:*

$$\frac{\|x^{CPU} - x^{GPU}\|_2}{\|x^{CPU}\|_2} < \epsilon_{GPU}, \quad \frac{\|x^{CPU} - x^{FPGA}\|_2}{\|x^{CPU}\|_2} < \epsilon_{FPGA}$$

*donde* $\epsilon_{GPU} = 10^{-6}$ *para aritmética de 32 bits y* $\epsilon_{FPGA}$ *es el error de cuantización del hardware de menor precisión.*

## 5.2 Test de Deriva Numérica

### 5.2.1 Rama D: Signatures en FPGA vs. CPU

**Caso de Prueba 5.2** (Acumulación de Error en Punto Fijo)**.** *Comparar los resultados de la Rama D (cálculo de signaturas de caminos rugosos) ejecutada en FPGA con aritmética de punto fijo frente a la implementación en CPU con punto flotante de 64 bits.*

**Criterio 5.2** (Criterio de Aceptación de Deriva)**.** *La signatura de un camino* $\gamma : [0, T] \to \mathbb{R}^d$ *se calcula mediante la expansión truncada:*

$$Sig^{(M)}(\gamma) = \left(1, \int_0^T d\gamma_{t_1}, \int_0^T \int_0^{t_1} d\gamma_{t_2} \otimes d\gamma_{t_1}, \dots\right)$$

*Para un test de 10,000 iteraciones (actualizaciones secuenciales de la signatura), se debe cumplir:*

1. *Ejecutar el algoritmo de signatura en CPU (double precision):* $Sig_{10000}^{CPU}$
2. *Ejecutar el mismo algoritmo en FPGA (punto fijo, n bits):* $Sig_{10000}^{FPGA}$
3. *Calcular la divergencia acumulada en norma infinito:*

$$\Delta_{acum} = \|Sig_{10000}^{CPU} - Sig_{10000}^{FPGA}\|_\infty$$

4. ***Criterio Primario (Cota de Error):*** *Verificar que:*

$$\Delta_{acum} \leq N \cdot \epsilon_{quant}$$

*donde* $N = 10000$ *es el número de iteraciones y* $\epsilon_{quant} = 2^{-n+1}$ *es el error de cuantización de la FPGA de n bits de precisión.*

5. ***Criterio Secundario (Preservación de Propiedades Topológicas):*** *Para asegurar que el error de cuantización no invalida la firma topológica, se deben verificar adicionalmente:*

   - ***Norma de la Signatura:*** *El error relativo en la norma debe ser acotado:*

$$\frac{|\|Sig_{10000}^{FPGA}\|_2 - \|Sig_{10000}^{CPU}\|_2|}{\|Sig_{10000}^{CPU}\|_2} < \tau_{norm}$$

   *con* $\tau_{norm} = 0.01$ *(error relativo* $< 1\%$*). Esto garantiza que la magnitud global de la firma se preserva.*

   - ***Preservación de Orden de Componentes:*** *La estructura jerárquica de las componentes tensoriales debe mantenerse. Sea* $s_i^{(k)}$ *la i-ésima componente del nivel k de la signatura. Verificar que:*

$$sgn(s_i^{(k),CPU}) = sgn(s_i^{(k),FPGA}) \quad \forall i, k$$

   *garantizando que no hay inversiones de signo que indiquen corrupción topológica.*

   - ***Distancia Angular (Cosine Similarity):*** *Medir la similitud angular entre las firmas:*

$$\cos(\theta) = \frac{\langle Sig_{10000}^{CPU}, Sig_{10000}^{FPGA} \rangle}{\|Sig_{10000}^{CPU}\|_2 \cdot \|Sig_{10000}^{FPGA}\|_2}$$

   *Se requiere* $\cos(\theta) > 0.9999$ *(desviación angular* $< 0.81°$*), asegurando que las firmas apuntan esencialmente en la misma dirección en el espacio de firmas.*

   - ***Error Relativo por Nivel:*** *Para cada nivel k de la expansión tensorial:*

$$\frac{\|Sig_{10000}^{CPU,(k)} - Sig_{10000}^{FPGA,(k)}\|_2}{\|Sig_{10000}^{CPU,(k)}\|_2} < \tau_k$$

   *donde* $\tau_k = 0.05 \cdot k$ *permite más tolerancia en niveles superiores pero mantiene estricta concordancia en los primeros niveles (que capturan la geometría macroscópica del camino).*

6. ***Criterio de Fallo:*** *El test falla si:*

   - $\Delta_{acum} > N \cdot \epsilon_{quant}$ *(violación del criterio primario), o*
   - *Cualquiera de los criterios de preservación topológica es violado*

*indicando que el error de cuantización acumulado ha degradado la integridad semántica de la firma topológica.*

**Nota 5.1.** *La deriva numérica en sistemas de punto fijo es inevitable debido al truncamiento repetido. El criterio primario establece que la acumulación de error no debe superar el producto del número de iteraciones por el quantum de representación, lo cual representa el peor caso teórico bajo la hipótesis de errores independientes.*

*Sin embargo, para la Rama D, no basta con que el error sea acotado en norma; es esencial que las propiedades cualitativas de la signatura se preserven:*

- *Las firmas de caminos se usan para discriminar trayectorias en espacios de alta dimensión*
- *Pequeñas corrupciones angulares pueden llevar a clasificaciones erróneas*
- *La preservación de signos en las componentes es crítica porque reflejan la orientación de bucles y excursiones del camino*
- *El error relativo por nivel asegura que los niveles bajos (momento, área orientada) que dominan la geometría macroscópica tengan precisión extrema*

**Nota 5.2.** *Para una FPGA con $n = 32$ bits (punto fijo Q16.16), el error de cuantización es $\epsilon_{quant} = 2^{-15} \approx 3.05 \times 10^{-5}$. Tras 10,000 iteraciones, el criterio primario permite $\Delta_{acum} \leq 0.305$, pero los criterios topológicos exigen:*

- *Error relativo en norma $< 1\%$*
- *Desviación angular $< 1°$*
- *Error en primer nivel $< 5\%$*

*Estos criterios adicionales garantizan integridad funcional más allá de la mera acotación numérica.*

### 5.2.2 Reproducibilidad Determinista

**Caso de Prueba 5.3** (Inicialización Controlada de Semillas)**.** *Garantizar que, dada la misma semilla de generador pseudo-aleatorio, todas las plataformas (CPU, GPU, FPGA) produzcan exactamente la misma secuencia de estados.*

**Criterio 5.3.** *Fijar una semilla determinista $s_0$ para el generador de números aleatorios. Ejecutar 1,000 pasos de simulación en cada plataforma y verificar:*

$$\{X_t^{CPU}\}_{t=1}^{1000} = \{X_t^{GPU}\}_{t=1}^{1000} = \{X_t^{FPGA}\}_{t=1}^{1000}$$

*La igualdad debe ser bit-a-bit para las plataformas que comparten el mismo formato de representación (CPU y GPU en punto flotante). Para FPGA, la comparación debe hacerse tras la conversión del formato de punto fijo a punto flotante.*

## 5.3 Validación de Latencia y Throughput

**Caso de Prueba 5.4** (Benchmark de Rendimiento Cross-Platform)**.** *Medir el tiempo de ejecución y el throughput de cada rama del predictor en las tres arquitecturas para identificar cuellos de botella y validar las ventajas de la implementación heterogénea.*

**Criterio 5.4.** *Para un lote de $N = 1000$ predicciones:*

$$T_{CPU} = \textit{tiempo total en CPU}$$
$$T_{GPU} = \textit{tiempo total en GPU}$$
$$T_{FPGA} = \textit{tiempo total en FPGA}$$

*Se espera que:*

- *GPU supere a CPU en operaciones masivamente paralelas (integración Monte Carlo):* $T_{GPU} < 0.3 \cdot T_{CPU}$
- *FPGA supere a CPU en operaciones deterministas de baja latencia (cálculo de signaturas):* $T_{FPGA} < 0.1 \cdot T_{CPU}$

*El throughput se mide como:*

$$Throughput = \frac{N}{T} \quad [predicciones/segundo]$$

# Capítulo 6

# Protocolo de Validación Final (Causalidad)

Este es el test definitivo de rendimiento predictivo que debe aplicarse antes de cualquier despliegue.

## 6.1 Generalización

**Caso de Prueba 6.1** (Rolling Walk-Forward)**.** *Ausencia total de look-ahead bias. El entrenamiento solo utiliza datos estrictamente anteriores al horizonte de test.*

**Criterio 6.1.** *Dividir el conjunto de datos en ventanas móviles:*

$$\mathcal{D} = \{(t_1, y_1), \ldots, (t_N, y_N)\}$$

*Para cada ventana de test* $\mathcal{T}_k = \{t_{n_k}, \ldots, t_{n_k+w}\}$*:*

1. *Entrenar solo con* $\mathcal{D}^k_{train} = \{(t_i, y_i) : t_i < t_{n_k}\}$
2. *Predecir en* $\mathcal{T}_k$
3. *Avanzar la ventana:* $k \to k+1$
4. *Acumular métricas out-of-sample (RMSE, MAE, Sharpe ratio)*

*El sistema debe garantizar que en ningún momento se utilicen datos futuros para entrenar el modelo en tiempo* $t$*.*

## 6.2 Eficiencia de Meta-Optimización

**Caso de Prueba 6.2** (Optimización Bayesiana)**.** *Mejora iterativa del Expected Improvement sobre el error de generalización en comparación con una búsqueda aleatoria.*

**Criterio 6.2.** *Utilizar un proceso Gaussiano (GP) como surrogate model del espacio de hiperparámetros* $\Theta = \{\alpha, \beta, \gamma, \ldots\}$*. Para* $n$ *iteraciones de optimización:*

1. *Búsqueda aleatoria:* $\theta_i \sim$ *Uniforme*$(\Theta)$
2. *Optimización Bayesiana:* $\theta_i = \arg\max_\theta EI(\theta|\mathcal{D}_{1:i-1})$

*Criterio de aceptación:*

$$\min_{i \le n} \mathcal{L}(\theta_i^{BO}) < \min_{i \le n} \mathcal{L}(\theta_i^{random})$$

*donde* $\mathcal{L}$ *es la función de pérdida en el conjunto de validación. La probabilidad de mejora debe ser significativa (p-value* $< 0.05$ *en test de Mann-Whitney).*

## 6.3 Integridad Temporal

**Caso de Prueba 6.3** (Métrica de Abandono (TTL)). *Cancelación de la actualización JKO si el retraso de la señal $y_{target}$ excede $\Delta_{max}$.*

**Criterio 6.3.** *Definir un Time-To-Live (TTL) para las señales objetivo:*

$$TTL(y_t) = t_{current} - t$$

*Si una señal de entrenamiento/actualización satisface:*

$$TTL(y_t) > \Delta_{max}$$

*el sistema debe:*

1. *Descartar la señal*
2. *NO ejecutar la actualización del Orquestador*
3. *Emitir* `StaleDateEvent`
4. *Registrar en log la marca temporal de la señal rechazada*

*Valor típico: $\Delta_{max} = 5$ segundos en sistemas de alta frecuencia.*

### 6.3.1 Test de Inyección de Lag

La política de abandono es crucial para evitar operar con "pesos obsoletos" que reflejan un estado del mercado ya irrelevante. Este test valida el comportamiento del sistema bajo condiciones de latencia extrema.

**Caso de Prueba 6.4** (Retraso Artificial de Señal). *Retrasar artificialmente la señal $y_{target}$ por encima de $\Delta_{max}$ y validar que el sistema active modo de inferencia degradada y suspenda el transporte óptimo.*

**Criterio 6.4.** *Procedimiento de validación:*

1. *Configurar el sistema con $\Delta_{max} = 5$ segundos*
2. *Generar un flujo de datos en tiempo real con timestamps correctos*
3. *Inyectar artificialmente una señal $\tilde{y}_t$ con timestamp retrasado:*

   $$TTL(\tilde{y}_t) = t_{current} - t = \Delta_{max} + \delta$$

   *donde $\delta > 0$ (típicamente $\delta = 1$ segundo para hacer el test inequívoco)*
4. *Verificar que el sistema ejecute la siguiente secuencia:*
   - *Detectar $TTL(\tilde{y}_t) > \Delta_{max}$*
   - *Activar bandera* `DegradedInferenceMode` $=$ `True`
   - *Suspender inmediatamente el algoritmo de transporte JKO:*

     $$JKO\_update(\rho^{n+1}) \to SUSPENDED$$

   - *Congelar pesos del Orquestador en su último valor válido: $\{w_i\}_{frozen}$*
   - *Emitir eventos:*
     - `StaleDataEvent` *con metadata:* $(t, t_{current}, TTL)$
     - `DegradedInferenceModeActivated`

- *Registrar en telemetría el tiempo de lag para análisis post-mortem*
- *Continuar generando predicciones usando solo los pesos congelados (sin actualizaciones dinámicas)*

5. *Verificar condición de recuperación: cuando el flujo de datos se normalice y aparezcan señales con* $TTL(y_t) < 0.8 \cdot \Delta_{max}$ *(umbral de histéresis), el sistema debe:*
   - *Desactivar* `DegradedInferenceMode`
   - *Reanudar transporte JKO*
   - *Emitir* `NormalOperationRestoredEvent`

*Criterios de aceptación:*

- *Tiempo de detección:* $< 100$ *milisegundos desde la recepción de* $\tilde{y}_t$
- *NO se debe ejecutar ni una sola iteración de JKO con datos obsoletos*
- *El sistema NO debe crashear ni entrar en estado indefinido*
- *Las predicciones deben continuar (aunque degradadas) con última configuración válida*

**Nota 6.1.** *El modo de inferencia degradada es un mecanismo de defensa crítico en sistemas de trading de alta frecuencia. Operar con pesos optimizados en base a información obsoleta es matemáticamente equivalente a resolver el problema de optimización incorrecto, lo cual puede llevar a pérdidas significativas. El sistema prioriza correctitud sobre completitud: es preferible operar con pesos estáticos pero consistentes que con pesos "óptimos" calculados sobre datos irrelevantes.*

# Capítulo 7

# Casos Extremos (Edge Cases) y Límites Operacionales

Este capítulo documenta escenarios límite que validan el comportamiento del sistema en condiciones de frontera teóricas y operacionales.

## 7.1 CUSUM: Umbral Dinámico Adaptativo

**Caso de Prueba 7.1** (Adaptación a Regímenes de Volatilidad)**.** *Validar que el umbral del detector CUSUM se adapte correctamente a regímenes de baja y alta volatilidad mediante la formulación dinámica* $h = k \cdot \sigma_{resid}$*.*

**Criterio 7.1.** *El detector CUSUM opera con umbral dinámico:*

$$h_t = k \cdot \sigma_{resid,t}$$

*donde* $k \in [3, 5]$ *es un factor de sensibilidad y* $\sigma_{resid,t}$ *es la desviación estándar móvil del residuo de predicción.*

*Test de validación:*

1. ***Régimen de baja volatilidad:***
   - *Generar una señal con* $\sigma_{true} = 0.01$
   - *Estimar* $\hat{\sigma}_{resid} \approx 0.01$
   - *Verificar* $h_t = k \cdot 0.01$ *(umbral bajo)*
   - *Inyectar un cambio de deriva pequeño:* $\Delta\mu = 0.05$
   - *El detector debe activarse cuando* $G_t^+ > h_t$ *con sensibilidad alta*
2. ***Régimen de alta volatilidad:***
   - *Generar una señal con* $\sigma_{true} = 0.50$ *(volatilidad 50× mayor)*
   - *Estimar* $\hat{\sigma}_{resid} \approx 0.50$
   - *Verificar* $h_t = k \cdot 0.50$ *(umbral alto)*
   - *Inyectar el mismo cambio:* $\Delta\mu = 0.05$
   - *El detector NO debe activarse (el cambio está dentro del ruido)*
   - *Inyectar cambio mayor:* $\Delta\mu = 2.0$
   - *Ahora sí debe detectar el cambio de régimen*
3. ***Transición de regímenes:***

- *Simular una transición de baja a alta volatilidad*
- *Verificar que $h_t$ se actualice suavemente (ventana móvil)*
- *No debe haber activaciones espurias durante la transición*

*Criterio de aceptación:*

$$\frac{h_{\textit{high-vol}}}{h_{\textit{low-vol}}} = \frac{\sigma_{high}}{\sigma_{low}} \pm 0.1$$

*El ratio de umbrales debe seguir el ratio de volatilidades con tolerancia del 10%.*

## 7.2 Orquestador: Convergencia a Máxima Entropía

**Caso de Prueba 7.2** (Pesos Uniformes ante Incertidumbre Total)**.** *Confirmar que el sistema converge a pesos uniformes $w = [0.25, 0.25, 0.25, 0.25]$ cuando el parámetro de regularización de Sinkhorn tiende a infinito ($\varepsilon \to \infty$), representando máxima incertidumbre.*

**Criterio 7.2.** *En el algoritmo de Sinkhorn, el parámetro de entropía $\varepsilon$ controla la regularización:*

$$\min_{\pi \in \Pi(\mu,\nu)} \{\langle C, \pi \rangle - \varepsilon H(\pi)\}$$

*donde $H(\pi) = -\sum_{ij} \pi_{ij} \log \pi_{ij}$ es la entropía de Shannon.*
*Comportamiento límite:*

1. *Cuando $\varepsilon \to 0$: La solución converge al transporte óptimo de Monge-Kantorovich (determinista)*
2. *Cuando $\varepsilon \to \infty$: La entropía domina, forzando máxima dispersión*

*Test de validación:*

1. *Configurar el Orquestador con cuatro ramas activas $(A, B, C, D)$*
2. *Ejecutar optimización con $\varepsilon_k = 10^k$ para $k \in \{0, 1, 2, 3, 4\}$*
3. *Para cada $\varepsilon_k$, registrar los pesos resultantes $\{w_A^k, w_B^k, w_C^k, w_D^k\}$*
4. *Verificar convergencia a uniformidad:*

$$\lim_{\varepsilon \to \infty} \{w_i\} = \left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}$$

5. *Criterio numérico para $\varepsilon = 10^4$:*

$$\max_{i \in \{A,B,C,D\}} |w_i - 0.25| < 0.01$$

*(todos los pesos dentro del 1% del valor uniforme)*

*Interpretación:*

- *$\varepsilon$ alto $\implies$ el sistema no puede distinguir entre ramas $\implies$ pesos uniformes por máxima entropía*
- *Es el principio de máxima entropía de Jaynes: ante ausencia de información, la distribución uniforme maximiza la entropía*

**Nota 7.1.** *Este test valida que el Orquestador respeta principios fundamentales de teoría de la información. En condiciones de incertidumbre total (todas las ramas son indistinguibles), el sistema debe evitar favorecer arbitrariamente una rama sobre otra, distribuyendo uniformemente los pesos.*

## 7.3 Rama D: Invariancia ante Reparametrización Temporal

**Caso de Prueba 7.3** (Invariancia de Signatures a Time-Warping)**.** *Probar el sistema con una señal y su versión "estirada" temporalmente; la signatura del camino rugoso debe ser idéntica bajo reparametrizaciones estrictamente crecientes.*

**Criterio 7.3.** *Una propiedad fundamental de las signaturas de caminos rugosos es su invariancia ante reparametrizaciones:*

$$Sig(\gamma) = Sig(\gamma \circ \phi)$$

*para cualquier función estrictamente creciente* $\phi : [0,1] \to [0,1]$ *con* $\phi(0) = 0$, $\phi(1) = 1$.

*Test de validación:*

1. *Generar una señal de referencia* $\gamma(t)$ *para* $t \in [0,1]$
2. *Calcular su signatura truncada:* $S_0 = Sig^{(M)}(\gamma)$
3. *Aplicar reparametrización no lineal:*

$$\phi_1(t) = t^2 \quad (aceleración)$$

$$\phi_2(t) = \sqrt{t} \quad (desaceleración)$$

$$\phi_3(t) = \frac{1}{2}(1 - \cos(\pi t)) \quad (sigmoidal)$$

4. *Para cada reparametrización, computar:*

$$\gamma_i(t) = \gamma(\phi_i(t))$$

*y su signatura:*

$$S_i = Sig^{(M)}(\gamma_i)$$

5. *Verificar invariancia:*

$$\|S_i - S_0\|_2 < \epsilon_{inv} \quad \forall i \in \{1,2,3\}$$

*con tolerancia* $\epsilon_{inv} = 10^{-8}$

*Test de no-invariancia (control negativo):*

1. *Aplicar una reparametrización NO monótona:* $\psi(t) = t^2 - 0.5t$ *(tiene un mínimo local)*
2. *Verificar que* $Sig(\gamma \circ \psi) \neq Sig(\gamma)$
3. *Este test confirma que el algoritmo no es trivialmente invariante a TODA transformación*

**Nota 7.2.** *La invariancia a reparametrizaciones es la razón por la cual las signaturas son ideales para caracterizar la forma geométrica de trayectorias independientemente de su velocidad de ejecución. En contextos financieros, esto significa que la signatura captura el patrón intrínseco de un movimiento de precio, sin importar si ocurrió en 1 minuto o en 1 hora. Esta propiedad es crucial para la generalización temporal del predictor.*

## 7.4 Tabla Resumen de Edge Cases

Cuadro 7.1: Escenarios Límite y Casos Extremos

| **Módulo** | **Escenario de Test** | **Propósito** |
|---|---|---|
| CUSUM | Umbral Dinámico $h = k \cdot \sigma_{\text{resid}}$ | Validar que el umbral basado en $\sigma_{\text{resid}}$ se adapte correctamente a regímenes de baja/alta volatilidad |
| Orquestador | Máxima Entropía ($\varepsilon \to \infty$) | Confirmar que el sistema converge a pesos uniformes $[0.25, 0.25, 0.25, 0.25]$ ante incertidumbre total |
| Rama D | Invariancia ante Reparametrización | Probar con señal y su versión "estirada" en tiempo; Signature debe ser idéntica ($< 10^{-8}$ error) |

# Capítulo 8

# Resumen de Criterios de Aceptación

## 8.1 Consideraciones Finales

Todos los protocolos aquí descritos son independientes del lenguaje de implementación y se basan exclusivamente en los fundamentos matemáticos y algorítmicos del Predictor Estocástico Universal.

**Nota 8.1.** *La batería completa de pruebas debe ejecutarse antes de cada despliegue a producción. Los resultados deben documentarse en un reporte de validación que incluya:*

- *Resumen ejecutivo de todos los tests ejecutados*
- *Métricas numéricas de cada criterio*
- *Gráficos de convergencia y distribuciones*
- *Análisis de casos de fallo (si los hubiere)*
- *Recomendaciones de calibración de parámetros*

**Nota 8.2.** *Este protocolo es un documento vivo que debe evolucionar junto con el sistema. Cualquier modificación a la arquitectura teórica debe reflejarse en nuevos casos de prueba o actualización de los existentes.*

Cuadro 8.1: Tabla de Validación Global

| **Test** | **Método** | **Criterio de Aceptación** |
|---|---|---|
| Generalización | Rolling Walk-Forward | Ausencia total de look-ahead bias<br>El entrenamiento usa datos $t < t_{\text{test}}$ |
| Eficiencia de Meta-Optimización | Optimización Bayesiana (GP) | Mejora iterativa del Expected Improvement sobre búsqueda aleatoria |
| Integridad Temporal | Métrica de Abandono (TTL básico) | Cancelación de actualización JKO si $\text{TTL}(y) > \Delta_{\max}$ |
| Inyección de Lag (TTL avanzado) | Retraso Artificial $> \Delta_{\max}$ | Activación inmediata de modo degradado y suspensión de transporte JKO |
| Nyquist Soft-Limit | Aliasing Multifractal (SIA) | Congelación de Rama Topológica antes de que error en $H$ supere 10% |
| Estabilidad HJB-DGM (Rama B) | Monitoreo de Gradientes en Alta Volatilidad | Gradient clipping y reducción adaptativa de $\eta$ ante explosiones sostenidas |
| Soluciones de Viscosidad | Principio de Comparación (Crandall-Lions) | Error $< 5\%$ vs. solución de referencia con verificación de subsupersolución |
| Colapso de Modo (DGM Rama B) | Entropía de Entrenamiento | Ratio de varianzas en $[0.3, 1.2]$ y $\text{Var}_x[V_\theta]/\text{Var}[g(\xi)]$ proporcional |
| Atomicidad I/O | Interrupción de Escritura (Snapshot) | Cold-Start seguro sin bucles infinitos y tiempo de recuperación $< 30$ seg |
| Corrupción Silenciosa | Detección de Bit Rot | Verificación SHA-256 con fallback a snapshot anterior válido |
| Umbral CUSUM Dinámico | Adaptación a Volatilidad $h = k \cdot \sigma_{\text{resid}}$ | Ratio de umbrales igual a ratio de volatilidades $\pm 10\%$ |
| Máxima Entropía (Orquestador) | Sinkhorn con $\varepsilon \to \infty$ | Convergencia a pesos uniformes $\max_i \lvert w_i - 0.25 \rvert < 0.01$ |
| Invariancia Temporal (Rama D) | Reparametrización de Caminos Rugosos | Error de signatura $< 10^{-8}$ entre señal original y time-warped |
| Consistencia de Bit | Tests Cross-Platform (CPU/GPU/FPGA) | Equivalencia CPU/GPU/FPGA dentro de márgenes de cuantización |
| Deriva Numérica | Rama D en Punto Fijo vs. Punto Flotante | Divergencia acumulada $\leq N \cdot \epsilon_{quant}$ tras 10,000 iteraciones |