Manual didáctico de usuario de la librería

`hidden-attractors-fo`

Guía explícita para ejecutar, interpretar y defender los workflows de `version_2`

Proyecto: localización de atractores ocultos en sistemas fraccionarios

Versión de trabajo: 20 de mayo de 2026

Índice

1. Propósito de este manual

Este documento explica cómo usar la carpeta `version_2` del repositorio como librería Python. No es una descripción del backend en abstracto. El objetivo es responder, de forma operativa, las siguientes preguntas:

1) ¿Dónde debo pararme en la terminal?

2) ¿Qué debo instalar?

3) ¿Qué comando corro primero?

4) ¿Qué calcula cada comando?

5) ¿Qué archivos de salida debo revisar?

6) ¿Cómo sé si el resultado apoya o descarta la ocultedad?

7) ¿Qué partes son librería pública y qué partes siguen siendo scripts heredados?

Advertencia técnica

La librería no demuestra por sí sola la existencia matemática de atractores ocultos. Produce evidencia numérica bajo contratos explícitos: modelo, orden fraccionario q, paso h, memoria finita L_m, tiempo final, tiempo de descarte, backend y umbrales de clasificación. Una conclusión de ocultedad requiere verificar que la cuenca del atractor candidato no intersecta vecindades de ningún equilibrio bajo las pruebas realizadas.

2. Qué es exactamente `version_2`

La carpeta `version_2` es la versión organizada como paquete Python instalable. Su paquete se llama:

`hidden-attractors-fo`

y el módulo importable se llama:

`hidden_attractors`

La idea central es que el código ya no debe usarse como una colección desordenada de scripts sueltos, sino como una librería con módulos reutilizables:

- modelos dinámicos;
- carga de candidatos finales;
- diagnósticos de trayectorias;
- etiquetas de cuencas;
- gráficas;
- workflows de robustez y verificación;
- envoltorios hacia backends nativos en C/EFORK.

2.1. Límite entre librería y scripts heredados

La regla práctica es:

Carpeta	Cómo debes usarla
`hidden_attractors/`	API principal de la librería. Aquí está lo que se debe importar desde Python.
`examples/`	Ejemplos cortos. Sirven para aprender y verificar instalación.
`tools/cli/`	Comandos mantenidos para ejecutar workflows desde terminal.
`docs/`	Documentación técnica y reportes.
`outputs/`	Salidas de referencia y nuevas salidas generadas por corridas.
`tools/legacy/`	Scripts históricos. Pueden contener lógica útil, pero no deben crecer como API nueva.

Idea operativa

Para uso diario, empieza con `examples/` y `tools/cli/`. Solo después entra a `hidden_attractors/` si necesitas programar tus propios análisis.

3. Instalación desde cero

3.1. Ubicación correcta en la terminal

Primero entra al repositorio y luego a la carpeta `version_2`. En Windows PowerShell:

```
cd C:\Users\moren\Desktop\Hidden-Attractors-Localization\version_2
```

En Linux/macOS:

```
cd ~/Hidden-Attractors-Localization/version_2
```

El archivo `pyproject.toml` está dentro de `version_2`. Por eso, si ejecutas `pip install -e .` desde la raíz equivocada, Python puede no encontrar el paquete.

3.2. Crear entorno virtual recomendado

En PowerShell:

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
```

En Linux/macOS:

```
python -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
```

3.3. Instalar la librería en modo editable

Instalación mínima:

```
python -m pip install -e .
```

Instalación para desarrollo y pruebas:

```
python -m pip install -e ".[dev]"
```

Instalación con análisis externos opcionales:

```
python -m pip install -e ".[analysis]"
```

Instalación para construir documentación con MkDocs:

```
python -m pip install -e ".[docs]"
```

3.4. Dependencias mínimas

La instalación base usa principalmente:

- Python $\geq$ 3.10;
- `numpy`;
- `matplotlib`;
- compilador C si se ejecutan workflows nativos;
- `pytest` solo si instalas `.[dev]`.

En Windows, para workflows nativos, necesitas que `gcc` esté en el `PATH`. En macOS puede ser necesario instalar soporte OpenMP mediante `libomp`.

4. Prueba mínima de instalación

Después de instalar, ejecuta:

```
python -m compileall hidden_attractors examples tests tools/cli
```

Luego:

```
python examples/quickstart_equilibria.py
python examples/list_final_candidates.py
python examples/dynamical_analysis_gallery.py
```

Si instalaste con `.[dev]`, ejecuta:

```
python -m pytest -q
```

Qué debe aparecer si todo salió bien

La prueba de equilibrios debe imprimir tres puntos: `E0`, `E+` y `E-`, con norma residual cercana a cero. La prueba de candidatos debe listar tres candidatos finales. El comando de `pytest` debe pasar las pruebas de humo.

5. Modelo mental de la librería

La librería está organizada alrededor del siguiente flujo:

Etapa	Pregunta que responde
Modelo	¿Qué sistema dinámico se está usando?
Equilibrios	¿Cuáles son los puntos de equilibrio que deben excluirse de la cuenca del atractor candidato?
Candidatos	¿Qué semillas o puntos finales de continuación se van a verificar?
Trayectorias	¿Qué hace el sistema de Caputo desde una condición inicial?
Robustez	¿El comportamiento observado persiste al cambiar h, L_m o el tiempo final?
Controles esféricos	¿La cuenca candidata toca vecindades pequeñas de algún equilibrio?
Cuencas refinadas	¿Qué pasa con puntos previamente marcados como `unknown`?
Reporte	¿Qué archivos justifican numéricamente la conclusión?

5.1. Qué no hace automáticamente

Advertencia técnica

`version_2` no debe interpretarse como una máquina que, al ejecutarse, encuentra y demuestra atractores ocultos. La generación histórica de semillas por Lur'e/Nyquist, función descriptiva clásica o Machado/FDF aparece en scripts heredados o salidas de referencia. La librería activa se concentra en cargar candidatos, analizarlos, simularlos, clasificarlos y generar evidencia reproducible.

6. El sistema de Chua implementado

El modelo base en `hidden_attractors.models.chua` usa la no linealidad por tramos:

$$f(x) = m_1 x + \frac{1}{2}(m_0 - m_1)(|x+1| - |x-1|). \tag{1}$$

El campo vectorial se escribe como:

$$ {}^{C}D_t^q x = \alpha(y - x - f(x)), \tag{2}$$
$$ {}^{C}D_t^q y = x - y + z, \tag{3}$$
$$ {}^{C}D_t^q z = -\beta y - \gamma z. \tag{4}$$

En la librería, la función `rhs_piecewise` no integra el sistema. Solo evalúa el lado derecho:

$$F(X) = \begin{bmatrix} \alpha(y - x - f(x)) \\ x - y + z \\ -\beta y - \gamma z \end{bmatrix}. \tag{5}$$

La integración fraccionaria causal de Caputo se delega al backend nativo EFORK cuando se ejecutan workflows que simulan trayectorias largas.

6.1. Parámetros por defecto

La clase `ChuaParameters` usa por defecto:

Parámetro	Valor
α	8.4562
β	12.0732
γ	0.0052
m_0	-0.1768
m_1	-1.1468

6.2. Equilibrios

La función:

```
from hidden_attractors.models import equilibria_piecewise
```

calcula los tres equilibrios del sistema por tramos:

$$E_0 = (0, 0, 0), \qquad E_+, \qquad E_-. \tag{6}$$

Para los equilibrios exteriores se usa la relación algebraica implementada en la librería. Si

$$\kappa = -\frac{\beta}{\gamma + \beta}, \tag{7}$$

entonces

$$x_+ = -\frac{m_0 - m_1}{m_1 - \kappa}, \qquad x_- = \frac{m_0 - m_1}{m_1 - \kappa}. \tag{8}$$

Después, para cada x, la librería reconstruye:

$$E(x) = (x,\, x + f(x),\, f(x)). \tag{9}$$

Advertencia técnica

Calcular los equilibrios no clasifica estabilidad fraccionaria. Para estabilidad local del sistema de orden q debe revisarse el espectro del Jacobiano y aplicar el criterio de Matignon:

$$|\arg(\lambda_i)| > \frac{q\pi}{2}. \tag{10}$$

La librería, en esta parte, solo da las coordenadas de equilibrio y verifica residuales del campo vectorial.

7. Primer uso: calcular y verificar equilibrios

Ejecuta:

```
python examples/quickstart_equilibria.py
```

Este script hace internamente:

```
from hidden_attractors import chua_piecewise_parameters
from hidden_attractors.models import equilibria_piecewise, rhs_piecewise

params = chua_piecewise_parameters()
for name, point in equilibria_piecewise(params).items():
    residual_norm = norm(rhs_piecewise(point, params))
    print(name, point, residual_norm)
```

7.1. Qué calcula

Calcula las coordenadas de E_0, E_+ y E_- y evalúa:

$$\|F(E_i)\|. \tag{11}$$

Si el residual es casi cero, el punto es efectivamente equilibrio del campo vectorial implementado.

7.2. Cómo interpretar la salida

Una salida esperada tiene la forma:

```
E0: point=[0.0, 0.0, 0.0] residual_norm=0.000e+00
E+: point=[...] residual_norm=...e-16
E-: point=[...] residual_norm=...e-16
```

Si aparece un residual grande, hay una inconsistencia de parámetros, de implementación o de entorno.

8. Segundo uso: listar candidatos finales

Ejecuta:

```
python examples/list_final_candidates.py
```

O, después de instalar la librería:

```
hidden-attractors-list-candidates
```

8.1. Qué calcula

No simula. No integra. No verifica ocultedad. Solo carga registros de candidatos desde salidas de referencia guardadas en `outputs/`. Cada candidato se representa con la clase `CandidateRecord`.

Cada registro contiene, cuando está disponible:

Campo	Significado
candidate_id	Identificador único del candidato.
route	Ruta por la cual se obtuvo: por ejemplo, Machado_FDF o Lure_rank_0001.
q	Orden fraccionario usado en la corrida de referencia.
seed	Semilla inicial teórica o numérica asociada al candidato.
robust_start	Punto desde donde se inician pruebas de robustez. Suele venir del final de una continuación o verificación previa.
mu, theta	Parámetros de la ruta Machado/FDF, si aplica.
A, sigma0, omega	Datos asociados al balance armónico o ruta Lur'e, si existen.
rho_H, residual_abs	Indicadores numéricos de la ruta de búsqueda, si existen.
source	Archivo de salida desde donde se cargó el registro.

8.2. Cómo interpretar la salida

Una línea típica tiene la forma:

```
branch_0_mu_4p00000_theta_0p00000 | route=Machado_FDF | q=0.9998 | start=[...] | seed=[...]
```

Esto significa que la librería encontró un registro de candidato ya existente. No significa que el candidato sea oculto.

Advertencia técnica

Un CandidateRecord es una hipótesis inicial. La ocultedad se evalúa después, con pruebas de cuenca alrededor de todos los equilibrios.

9. Tercer uso: cargar una trayectoria y calcular métricas

La librería usa la convención de trayectoria:

$$\texttt{t,x,y,z}. \tag{12}$$

Es decir, un CSV de trayectoria debe tener columnas:

```
t,x,y,z
0.00,...,...,...
0.01,...,...,...
...
```

9.1. Cargar trayectoria

```python
from hidden_attractors.io import load_trajectory_csv

traj = load_trajectory_csv("outputs/mi_corrida/trajectories/caso.csv")
```

9.2. Calcular métricas

```python
from hidden_attractors.analysis import trajectory_metrics

metrics, payload = trajectory_metrics(
    traj,
    h=0.01,
    t_start=750.0,
    divergence_norm=120.0,
    equilibrium_tol=1e-3,
)

print(metrics)
```

9.3. Qué calcula `trajectory_metrics`

La función calcula diagnósticos de tiempo finito:

Métrica	Interpretación
`bounded`	La trayectoria permanece finita y no supera la norma de divergencia definida.
`diverged`	La trayectoria diverge o contiene valores no finitos.
`equilibrium_like`	El punto final cae cerca de un equilibrio, según `equilibrium_tol`.
`noncollapsed_variance`	La cola de la trayectoria conserva varianza suficiente; evita confundir un colapso numérico con oscilación.
`final_norm`	Norma del estado final.
`max_norm`	Norma máxima observada.
`range_x`, `range_y`, `range_z`	Amplitud observada en la cola temporal.
`var_x_tail`, `var_y_tail`, `var_z_tail`	Varianza en la cola.
`fft_peak`	Frecuencia dominante estimada por FFT para $x(t)$.
`psd_entropy`	Entropía espectral normalizada aproximada.
`section_points`	Número de cruces ascendentes de la sección $x = 0$.
`cloud_median_distance`	Distancia mediana entre nubes de puntos, si se compara contra una referencia.
`section_median_distance`	Distancia entre nubes de sección de Poincaré, si existe referencia.

Advertencia técnica

Estas métricas no sustituyen exponentes de Lyapunov ni demuestran caos. Sirven para comparar trayectorias, detectar divergencia, detectar convergencia a equilibrio y medir similitud geométrica con una referencia.

10. Cuarto uso: generar gráficas didácticas

Para probar el módulo de análisis visual:

```
python examples/dynamical_analysis_gallery.py
```

Con una trayectoria específica:

```
python examples/dynamical_analysis_gallery.py --trajectory-csv outputs/mi_corrida/
    trajectories/caso.csv
```

10.1. Qué produce

El script puede generar:

- retrato de fase 3D;
- proyecciones xy, xz, yz;
- series temporales;
- puntos de bifurcación extraídos de trayectorias existentes;
- `summary.json`;
- `bifurcation_points.csv`.

10.2. Qué no produce

No hace continuación real. No encuentra ramas de bifurcación. Solo postprocesa trayectorias ya generadas.

11. Flujo completo recomendado

El flujo recomendado para usar la librería sin perderse es:

Paso	Acción	Comando principal
1	Instalar librería	`python -m pip install -e .`
2	Probar importación	`python -m compileall hidden_attractors examples tests tools/cli`
3	Verificar equilibrios	`python examples/quickstart_equilibria.py`
4	Listar candidatos	`python examples/list_final_candidates.py`
5	Revisar ayuda de workflows	`hidden-attractors-robustness-overlay -help`
6	Ejecutar robustez geométrica	`hidden-attractors-robustness-overlay`
7	Ejecutar controles alrededor de equilibrios	`hidden-attractors-sphere-controls`
8	Refinar cuencas desconocidas, si existe una cuenca gruesa previa	`hidden-attractors-refined-basin`
9	Revisar CSV/JSON finales	Abrir `summary.json`, `decision.csv` y gráficas.

Idea operativa

No empieces corriendo el workflow más pesado. Primero verifica equilibrios y candidatos. Después ejecuta `-help`. Solo entonces lanza simulaciones largas.

12. Workflow 1: robustness_overlay

12.1. Comando

Después de instalar:

```
hidden-attractors-robustness-overlay
```

O directamente desde el wrapper:

```
python tools/cli/robustness_overlay_c_trajectories.py
```

Para ver opciones sin ejecutar simulaciones:

```
hidden-attractors-robustness-overlay --help
```

12.2. Propósito

Este workflow compara trayectorias iniciadas desde los candidatos finales al cambiar:

- paso de integración h;
- longitud de memoria finita L_m;
- tiempo final de integración;
- tiempo de descarte transitorio.

Su pregunta es:

¿El atractor candidato conserva geometría y rasgos espectrales bajo perturbaciones numéricas razonables?

12.3. Qué no responde

No responde si el atractor es oculto. Para ocultedad se necesitan pruebas alrededor de los equilibrios.

12.4. Casos de robustez por defecto

La librería define casos similares a:

Caso	q	h	L_m	T
R0_base	0.9998	0.01	10	1500
R1_h_finer	0.9998	0.005	10	1500
R2_h_coarser	0.9998	0.02	10	1500
R3_Lm_lower	0.9998	0.01	5	1500
R4_Lm_higher	0.9998	0.01	20	1500
R5_t_longer	0.9998	0.01	10	3000

12.5. Ejecución con carpeta de salida explícita

Recomendado:

```
hidden-attractors-robustness-overlay --output-dir outputs/manual_run/robustness_overlay_test
```

Así no pierdes la corrida en una carpeta con timestamp.

12.6. Archivos que genera

Archivo o carpeta	Significado
robustness_overlay_config.json	Contrato numérico completo: candidatos, casos, parámetros y umbrales.
launch_manifest.json	Procesos lanzados, PID y comandos usados.
logs/	Salidas .out y errores .err de cada proceso.
trajectories/<candidate>/R*.csv	Trayectorias muestreadas para cada candidato y caso de robustez.
metrics_<candidate>.csv	Métricas individuales por candidato.
metrics_<candidate>.done	Marca de que ese candidato terminó.
robustness_overlay_metrics.csv	Tabla agregada de métricas de todos los candidatos.
robustness_overlay_summary.json	Resumen global de la corrida.
plots/overlay_<candidate>.png	Comparación visual de trayectorias.

12.7. Cómo saber si terminó bien

Abre:

```
outputs/manual_run/robustness_overlay_test/robustness_overlay_summary.json
```

Debe contener:

```
"status": "ok"
```

Si dice partial, revisa:

```
outputs/manual_run/robustness_overlay_test/logs/
```

y busca archivos .err no vacíos.

12.8. Cómo interpretar robustness_overlay_metrics.csv

Columnas importantes:

Columna	Lectura práctica
candidate_id	Candidato evaluado.
route	Ruta de origen del candidato.
case_id	Caso de robustez.
bounded	Debe ser True para conservar interés.
diverged	Si es True, ese caso falló por divergencia.
equilibrium_like	Si es True, la trayectoria terminó cerca de un equilibrio.
noncollapsed_variance	Si es False, la trayectoria perdió dinámica relevante.
range_relative_distance	Distancia relativa de rangos respecto al caso base. Menor suele indicar mayor robustez geométrica.
fft_relative_delta	Cambio relativo de frecuencia dominante respecto al caso base.
cloud_median_distance_norm	Distancia normalizada entre nubes de trayectoria.
section_median_distance_norm	Distancia normalizada entre secciones de Poincaré.

Advertencia técnica

Un candidato robusto bajo `robustness_overlay` todavía puede ser autoexcitado si alguna trayectoria iniciada cerca de un equilibrio llega al mismo atractor.

13. Workflow 2: `sphere_controls`

13.1. Comando

```
hidden-attractors-sphere-controls
```

O:

```
python tools/cli/lure_top3_sphere_robustness.py
```

Para revisar opciones:

```
hidden-attractors-sphere-controls --help
```

13.2. Propósito

Este workflow sí está relacionado directamente con la verificación de ocultedad. Su pregunta es:

¿La cuenca del atractor candidato intersecta vecindades pequeñas de algún equilibrio?

Para responder, el workflow genera condiciones iniciales sobre esferas pequeñas alrededor de cada equilibrio:

$$X_0 = E_i + r\,u, \tag{13}$$

donde E_i es un equilibrio, r es un radio pequeño y u es una dirección unitaria.
Después clasifica cada trayectoria con el backend de cuencas.

13.3. Interpretación matemática

La definición operativa usada es:

> Un atractor es oculto si su cuenca de atracción no intersecta ninguna vecindad abierta de los puntos de equilibrio.

Por tanto:

- si desde una esfera alrededor de un equilibrio aparecen trayectorias hacia el atractor candidato, hay evidencia contra la ocultedad;
- si no aparecen impactos hacia el atractor candidato bajo los radios y muestras probados, el resultado es compatible con ocultedad, pero no es una prueba absoluta.

13.4. Ejecución recomendada

Ejemplo con carpeta explícita:

```
hidden-attractors-sphere-controls --output-dir outputs/manual_run/sphere_controls_test
```

Ejemplo aumentando muestras y procesos:

```
hidden-attractors-sphere-controls ^
  --output-dir outputs/manual_run/sphere_controls_deep ^
  --samples-per-radius 1000 ^
  --chunks 8 ^
  --radii "1e-6,3e-6,1e-5,3e-5,1e-4,3e-4,1e-3"
```

En Linux/macOS cambia ^ por
:

```
hidden-attractors-sphere-controls \
  --output-dir outputs/manual_run/sphere_controls_deep \
  --samples-per-radius 1000 \
  --chunks 8 \
  --radii "1e-6,3e-6,1e-5,3e-5,1e-4,3e-4,1e-3"
```

13.5. Parámetros importantes

Parámetro	Significado
`-top-n`	Número de candidatos finales a verificar. Por defecto, 3.
`-radii`	Radios de las esferas alrededor de cada equilibrio.
`-samples-per-radius`	Número de direcciones por radio y equilibrio.
`-chunks`	Número de procesos independientes para dividir las muestras.
`-q`	Orden fraccionario.
`-h`	Paso de integración.
`-memory-length`	Longitud de memoria finita L_m.
`-t-final`	Tiempo final de integración.
`-t-burn`	Tiempo transitorio descartado.
`-divergence-norm`	Umbral de divergencia.
`-equilibrium-tol`	Tolerancia para considerar convergencia a equilibrio.
`-mean-x-gap`	Criterio operativo usado por el clasificador para separar clases objetivo.
`-cap-win`	Ventana usada por el clasificador para estimaciones de cola.

13.6. Archivos que genera

Archivo	Significado
top3_sphere_robustness_config.json	Contrato completo de la prueba: candidatos, equilibrios, radios y parámetros.
sphere_plan.csv	Todas las condiciones iniciales generadas sobre esferas.
sphere_raw_chunk_*.csv	Resultados por proceso.
sphere_raw.csv	Resultados agregados de todas las muestras.
sphere_cumulative_summary.csv	Resumen acumulado por candidato, equilibrio y radio.
sphere_decision.csv	Tabla de decisión principal sobre compatibilidad con ocultedad.
attractor_robustness_raw.csv	Clasificación de puntos finales de candidatos bajo contratos de robustez.
attractor_robustness_summary.csv	Resumen de robustez por candidato.
top3_sphere_robustness_summary.json	Resumen global de la corrida.
logs/	Salida y errores de cada proceso.

13.7. Cómo interpretar sphere_decision.csv

Columnas clave:

Columna	Interpretación
candidate_id	Candidato evaluado.
total_sphere_trajectories	Total de trayectorias lanzadas desde vecindades de equilibrios.
total_target_hits	Número de trayectorias que llegaron a una clase objetivo.
smallest_radius_with_target_hit	Radio más pequeño donde se detectó contacto con la cuenca objetivo.
hiddenness_status	Lectura operativa de la prueba.
notes	Recordatorio de que no es prueba absoluta.

13.8. Estados posibles de hiddenness_status

Estado	Significado
not_supported_by_sphere_equilibrium_test	Se encontró al menos una trayectoria desde vecindades de equilibrio hacia la clase objetivo. Esto contradice la hipótesis de ocultedad bajo el contrato probado.
compatible_with_hiddenness_under_tested_spheres	No se detectaron impactos a la clase objetivo desde las esferas probadas. Es compatible con ocultedad, pero no prueba definitiva.

Advertencia técnica

Si `total_target_hits >0`, no conviene defender ese candidato como oculto con ese contrato numérico. Si `total_target_hits = 0`, se puede decir que el candidato sobrevivió la prueba de esferas bajo los radios, muestras y parámetros usados.

14. Workflow 3: `refined_basin`

14.1. Comando

```
hidden-attractors-refined-basin
```

O:

```
python tools/cli/refine_project_basin_classification.py
```

Para revisar opciones:

```
hidden-attractors-refined-basin --help
```

14.2. Propósito

Este workflow no construye una cuenca desde cero. Revisa una cuenca gruesa previa y toma las celdas marcadas como:

`unknown`

Luego reintegra esas condiciones iniciales y compara sus trayectorias contra referencias positiva y negativa mediante:

- distancia entre nubes de puntos de cola;
- rangos por coordenada;
- frecuencia dominante FFT;
- sección de Poincaré cuando hay suficientes cruces.

14.3. Cuándo se usa

Úsalo solamente si ya existe una corrida previa de cuenca con archivos como:

```
basin_comparison_config.json
project_best_basin_grid.csv
project_best_basin_xy.png
```

Si esos archivos no existen, el workflow no tiene fuente que refinar.

14.4. Ejecución recomendada

```
hidden-attractors-refined-basin ^
  --source-dir outputs/basin_compare_danca_project_h001_grid101_20260517 ^
  --output-dir outputs/manual_run/refined_basin_test ^
  --chunks 8
```

En Linux/macOS:

```
hidden-attractors-refined-basin \
  --source-dir outputs/basin_compare_danca_project_h001_grid101_20260517 \
  --output-dir outputs/manual_run/refined_basin_test \
  --chunks 8
```

14.5. Archivos que genera

Archivo	Significado
`refined_basin_config.json`	Contrato completo de refinamiento.
`unknown_refinement_plan.csv`	Celdas `unknown` que serán reintegradas.
`unknown_refined_chunk_*.csv`	Resultados por proceso.
`unknown_refined_rows.csv`	Todas las celdas refinadas.
`project_best_basin_refined_grid.csv`	Cuenca final con celdas refinadas.
`project_best_basin_refined_grid.npy`	Matriz de clases para uso en Python.
`project_best_basin_refined_xy.png`	Imagen de la cuenca refinada.
`refined_basin_summary.json`	Resumen final.

14.6. Clases refinadas

El workflow conserva las clases base y agrega:

ID	Clase
0	`equilibrium`
1	`target_positive`
2	`target_negative`
3	`infinity`
4	`unknown`
5	`numerical_failure`
6	`bounded_other`

La clase `bounded_other` significa que la trayectoria fue acotada y no colapsada, pero no se pareció lo suficiente a las referencias objetivo bajo los umbrales definidos.

15. Clasificación de cuencas: significado de las etiquetas

La librería usa las siguientes etiquetas operativas:

ID	Etiqueta	Significado práctico
0	`equilibrium`	La trayectoria termina cerca de un equilibrio.
1	`target_positive`	La trayectoria cae en la clase objetivo positiva.
2	`target_negative`	La trayectoria cae en la clase objetivo negativa.
3	`infinity`	La trayectoria diverge o excede umbrales de norma.
4	`unknown`	El clasificador no decide con los criterios actuales.
5	`numerical_failure`	Falló la integración o el cálculo.
6	`bounded_other`	Clase agregada en refinamiento: acotada, no colapsada, pero distinta de referencia objetivo.

Las clases objetivo son:

$$\{1, 2\} = \{\texttt{target\_positive}, \texttt{target\_negative}\}. \tag{14}$$

Advertencia técnica

Una etiqueta `target_positive` o `target_negative` no significa por sí sola "atractor oculto". Solo indica que, bajo el contrato numérico probado, la trayectoria fue asignada a una clase objetivo acotada no trivial.

16. Cómo leer una corrida completa

Supón que ejecutaste:

```
hidden-attractors-sphere-controls --output-dir outputs/manual_run/sphere_controls_deep
```

Revisa en este orden:

16.1. Paso 1: verificar que la corrida terminó

Abre:

```
outputs/manual_run/sphere_controls_deep/top3_sphere_robustness_summary.json
```

Busca:

```
"status": "ok"
```

Si dice `partial`, revisa `logs/`.

16.2. Paso 2: revisar decisión principal

Abre:

```
outputs/manual_run/sphere_controls_deep/sphere_decision.csv
```

La columna central es:

```
hiddenness_status
```

16.3. Paso 3: revisar cuántos contactos hubo

Si:

$$\texttt{total\_target\_hits} > 0, \tag{15}$$

entonces hubo contacto entre una vecindad de equilibrio y la clase objetivo. Bajo esa prueba, no debes afirmar ocultedad.

Si:

$$\texttt{total\_target\_hits} = 0, \tag{16}$$

entonces el candidato sobrevivió los radios y muestras probadas.

16.4. Paso 4: revisar robustez del propio atractor candidato

Abre:

```
outputs/manual_run/sphere_controls_deep/attractor_robustness_summary.csv
```

Busca:

```
robust_attractor
robustness_status
```

Si el candidato no es robusto, aunque no haya contactos desde equilibrios, todavía no es un candidato fuerte.

16.5. Paso 5: revisar crudos si hay dudas

Abre:

```
outputs/manual_run/sphere_controls_deep/sphere_raw.csv
```

Filtra por:

```
target_hit = True
```

Ahí puedes ver:

- equilibrio de origen;
- radio;
- muestra;
- condición inicial exacta;
- clase asignada.

Esto sirve para reproducir casos críticos.

17. Paralelización: qué sí y qué no

17.1. Qué sí se paraleliza

Los workflows paralelizan tareas independientes:

- diferentes candidatos en `robustness_overlay`;
- diferentes fragmentos de muestras esféricas en `sphere_controls`;
- diferentes celdas `unknown` en `refined_basin`.

17.2. Qué no debe paralelizarse ingenuamente

La continuación fraccionaria con memoria no debe romperse como si fuera una colección de condiciones iniciales independientes cuando depende de la historia previa. En sistemas de Caputo, el estado efectivo depende de una ventana de memoria:

$$\{X(t_k - M), \ldots, X(t_k)\}. \tag{17}$$

Por eso, si en el futuro se integra continuación en `version_2`, debe conservarse la ventana de historia entre pasos de continuación.

17.3. Uso práctico de `-chunks`

En `sphere_controls`:

```
hidden-attractors-sphere-controls --chunks 8
```

significa que el plan de condiciones iniciales se divide en 8 subconjuntos independientes. Cada proceso escribe su propio archivo:

```
sphere_raw_chunk_000.csv
sphere_raw_chunk_001.csv
...
```

Después el agregador produce:

```
sphere_raw.csv
sphere_decision.csv
top3_sphere_robustness_summary.json
```

18. Uso desde Python como librería

Además de usar comandos de terminal, puedes escribir tus propios scripts.

18.1. Ejemplo 1: imprimir equilibrios

```python
import numpy as np
from hidden_attractors import chua_piecewise_parameters
from hidden_attractors.models import equilibria_piecewise, rhs_piecewise

params = chua_piecewise_parameters()
for name, eq in equilibria_piecewise(params).items():
    print(name, eq, np.linalg.norm(rhs_piecewise(eq, params)))
```

18.2. Ejemplo 2: cargar candidatos

```
from hidden_attractors import load_final_candidate_records

records = load_final_candidate_records()
for r in records:
    print(r.candidate_id)
    print("route =", r.route)
    print("q =", r.q)
    print("seed =", r.seed)
    print("robust_start =", r.robust_start)
```

18.3. Ejemplo 3: clasificar etiquetas de cuenca

```
from hidden_attractors.basins import class_label, is_target_class

for class_id in range(6):
    print(class_id, class_label(class_id), is_target_class(class_id))
```

18.4. Ejemplo 4: calcular métricas de trayectoria

```
from hidden_attractors.io import load_trajectory_csv
from hidden_attractors.analysis import trajectory_metrics

traj = load_trajectory_csv("outputs/mi_corrida/trayectoria.csv")
metrics, payload = trajectory_metrics(
    traj,
    h=0.01,
    t_start=750.0,
)

for k, v in metrics.items():
    print(k, v)
```

19. Relación con función descriptiva y Machado/FDF

La librería `version_2` carga candidatos que provienen de rutas anteriores, incluyendo ruta Lur'e clásica y ruta Machado/FDF. Operativamente:

- la función descriptiva clásica y la función descriptiva fraccionaria se usan como generadores heurísticos de semillas;
- esas semillas no prueban ciclos exactos en Caputo;
- las semillas deben trasladarse al sistema no lineal completo mediante simulación y verificación;
- `version_2` se concentra en verificar, comparar, clasificar y documentar esos candidatos.

19.1. Puente Weyl–Caputo

Cuando se usan Fourier, balance armónico, Nyquist o función descriptiva, el razonamiento armónico ideal corresponde al régimen estacionario tipo Weyl/Liouville–Weyl. La simulación causal de la librería corresponde a Caputo, con memoria desde t_0. La diferencia aparece como discrepancia de memoria. Por tanto, las semillas armónicas se interpretan como aproximaciones útiles para explorar el sistema de Caputo, no como prueba de ciclos periódicos exactos.

20. Errores comunes

20.1. Error: `ModuleNotFoundError: hidden_attractors`

Causa probable: no instalaste el paquete o no estás en el entorno virtual correcto.
Solución:

```
cd version_2
python -m pip install -e .
```

20.2. Error: no encuentra archivos en `outputs/`

Algunos cargadores esperan salidas de referencia. Verifica que exista la carpeta esperada. Por ejemplo, los candidatos finales pueden depender de archivos en:

```
outputs/extended_search/
outputs/lure_biased_multiparam_q09998_20260515_195444/
```

Si no existen, el cargador no puede inventar candidatos.

20.3. Error de compilador C

Si falla un workflow nativo, verifica que tengas compilador C.
En PowerShell:

```
gcc --version
```

Si no existe, instala MinGW, MSYS2 o un compilador compatible y agrégalo al `PATH`.

20.4. La corrida parece no terminar

Revisa los logs:

```
outputs/tu_corrida/logs/
```

Los archivos `.out` muestran avance. Los archivos `.err` muestran errores.

20.5. El resumen dice `partial`

Significa que no todos los procesos terminaron o no todos escribieron sus archivos `.done`. Revisa:

- archivos `.done`;
- logs de errores;
- si se interrumpió la terminal;
- si hubo error de compilación nativa;
- si la ruta `-source-dir` existe.

21. Checklist para defender una corrida

Antes de usar resultados en un reporte, asegúrate de poder llenar esta lista:

1) ¿Qué modelo se usó? Escribir ecuaciones y parámetros.

2) ¿Qué orden fraccionario q se usó?

3) ¿Qué método de integración se usó? EFORK nativo, ABM u otro.

4) ¿Qué paso h se usó?

5) ¿Qué memoria finita L_m se usó?

6) ¿Cuál fue el tiempo final?

7) ¿Cuál fue el tiempo transitorio descartado?

8) ¿Cuáles son todos los equilibrios?

9) ¿Desde qué candidato o semilla se inició la trayectoria objetivo?

10) ¿La trayectoria candidata fue robusta al cambiar h, L_m y T?

11) ¿Se lanzaron trayectorias desde vecindades de todos los equilibrios?

12) ¿Qué radios se probaron?

13) ¿Cuántas muestras por radio se usaron?

14) ¿Hubo `target_hits` desde algún equilibrio?

15) ¿Qué archivo CSV/JSON respalda cada afirmación?

21.1. Frases correctas para el reporte

Si no hubo impactos desde equilibrios:

> Bajo el contrato numérico especificado, las pruebas esféricas alrededor de los equilibrios no detectaron intersección entre las vecindades probadas y la cuenca objetivo. Por tanto, el candidato es compatible con ocultedad bajo los radios, muestras y parámetros evaluados.

Si sí hubo impactos:

> La prueba esférica detectó trayectorias iniciadas en vecindades de equilibrio que alcanzan la clase objetivo. Bajo este contrato numérico, la hipótesis de atractor oculto no queda respaldada para este candidato.

Si solo se hizo `robustness_overlay`:

> La trayectoria candidata muestra persistencia geométrica y espectral bajo variaciones de h, L_m y tiempo final. Este resultado evalúa robustez numérica, pero no constituye una verificación de ocultedad.

22. Ruta de trabajo sugerida para tus próximas corridas

Para una nueva sesión de trabajo, usa esta secuencia:

Día 1: verificación rápida

```
cd C:\Users\moren\Desktop\Hidden-Attractors-Localization\version_2
.\.venv\Scripts\Activate.ps1
python -m compileall hidden_attractors examples tests tools/cli
python examples/quickstart_equilibria.py
python examples/list_final_candidates.py
```

Día 1: robustez ligera

```
hidden-attractors-robustness-overlay --output-dir outputs/manual_run/robustness_overlay_light
```

Revisar:

```
outputs/manual_run/robustness_overlay_light/robustness_overlay_summary.json
outputs/manual_run/robustness_overlay_light/robustness_overlay_metrics.csv
outputs/manual_run/robustness_overlay_light/plots/
```

Día 2: prueba de ocultedad por esferas

```
hidden-attractors-sphere-controls ^
  --output-dir outputs/manual_run/sphere_controls_main ^
  --samples-per-radius 500 ^
  --chunks 6
```

Revisar:

```
outputs/manual_run/sphere_controls_main/sphere_decision.csv
outputs/manual_run/sphere_controls_main/sphere_cumulative_summary.csv
outputs/manual_run/sphere_controls_main/attractor_robustness_summary.csv
```

Día 3: refinamiento si hay cuenca gruesa

Solo si existe una cuenca previa:

```
hidden-attractors-refined-basin ^
  --source-dir outputs/basin_compare_danca_project_h001_grid101_20260517 ^
  --output-dir outputs/manual_run/refined_basin_main ^
  --chunks 8
```

23. Mapa de módulos principales

Módulo	Para qué sirve
`hidden_attractors.models.chua`	Define parámetros, no linealidad, campo vectorial y equilibrios del Chua no suave.
`hidden_attractors.candidates`	Carga candidatos finales desde salidas de referencia.
`hidden_attractors.analysis.trajectory`	Calcula métricas de trayectoria: rangos, varianzas, FFT, secciones, distancias entre nubes.
`hidden_attractors.basins.classification`	Define etiquetas de clases de cuenca.
`hidden_attractors.io`	Lee y escribe CSV/JSON, carga trayectorias y serializa resultados.
`hidden_attractors.plotting`	Genera figuras de fase, proyecciones, series, bifurcaciones y overlays.
`hidden_attractors.native`	Envoltorios hacia integración EFORK y clasificación nativa en C.
`hidden_attractors.workflows.robustness_overlay`	Workflow de robustez geométrica y espectral.
`hidden_attractors.workflows.sphere_controls`	Workflow de controles alrededor de equilibrios para evaluar ocultedad.
`hidden_attractors.workflows.refined_basin`	Workflow de refinamiento de celdas `unknown` en cuencas.

24. Resumen final

La librería se usa en tres niveles:

1) **Nivel básico:** instalar, verificar equilibrios, listar candidatos y cargar trayectorias.

2) **Nivel operativo:** ejecutar workflows de robustez, controles esféricos y refinamiento de cuencas.

3) **Nivel de investigación:** interpretar los resultados bajo el criterio de atractores ocultos, cuidando que cada afirmación esté respaldada por el contrato numérico y sus archivos de salida.

La ruta mínima para una verificación defendible es:

$$\text{candidato} \longrightarrow \text{robustez} \longrightarrow \text{controles alrededor de equilibrios} \longrightarrow \text{lectura de cuencas.} \tag{18}$$

No debe concluirse ocultedad solo desde una semilla de función descriptiva, ni solo desde una trayectoria atractora, ni solo desde robustez visual. La prueba numérica relevante es la ausencia de contacto detectado entre la cuenca objetivo y vecindades de todos los equilibrios bajo radios, muestras y parámetros explícitos.

A. Comandos de referencia rápida

Instalar

```
cd version_2
python -m pip install -e ".[dev]"
```

Probar

```
python -m compileall hidden_attractors examples tests tools/cli
python examples/quickstart_equilibria.py
python examples/list_final_candidates.py
python -m pytest -q
```

Ayuda de workflows

```
hidden-attractors-robustness-overlay --help
hidden-attractors-sphere-controls --help
hidden-attractors-refined-basin --help
```

Robustez

```
hidden-attractors-robustness-overlay --output-dir outputs/manual_run/robustness_overlay
```

Controles esféricos

```
hidden-attractors-sphere-controls --output-dir outputs/manual_run/sphere_controls --chunks 8
    --samples-per-radius 1000
```

Refinamiento de cuenca

```
hidden-attractors-refined-basin --source-dir outputs/
    basin_compare_danca_project_h001_grid101_20260517 --output-dir outputs/manual_run/
    refined_basin --chunks 8
```

B. Archivos que conviene guardar para el reporte

- `robustness_overlay_config.json`
- `robustness_overlay_metrics.csv`
- `robustness_overlay_summary.json`
- `plots/overlay_*.png`
- `top3_sphere_robustness_config.json`
- `sphere_decision.csv`
- `sphere_cumulative_summary.csv`
- `sphere_raw.csv`
- `attractor_robustness_summary.csv`
- `refined_basin_summary.json`, si se ejecutó refinamiento
- `project_best_basin_refined_xy.png`, si se ejecutó refinamiento