Caso02

November 10, 2025

1 Evaluación del caso “Sitnikov modificado con control de periodicidad”

Este notebook estudia una variante del problema de Sitnikov: dos estrellas de masas similares orbitan en el plano XY y un tercer cuerpo ligero se mueve a lo largo del eje Z. Ajustamos las masas mediante el pipeline híbrido (GA + refinamiento continuo) para buscar órbitas casi periódicas y con menor sensibilidad caótica medida mediante el exponente de Lyapunov.

> **Interpretación** - La penalización `periodicity_weight` fuerza que el estado final se acerque al inicial; un valor pequeño del fitness implica trayectoria casi repetitiva. - Comparar   del centro del rango de masas contra el   optimizado permite medir cuánto se estabiliza la dinámica vertical del tercer cuerpo.

1.1 Preparación del entorno

Normalizamos la ruta raíz del proyecto y la insertamos en `sys.path` para que los imports funcionen sin importar desde qué carpeta se lance el notebook.

[1]:

```python
import sys
from pathlib import Path

PROJECT_ROOT = Path.cwd().resolve()
while PROJECT_ROOT.name != "two_body" and PROJECT_ROOT.parent != PROJECT_ROOT:
    PROJECT_ROOT = PROJECT_ROOT.parent

if PROJECT_ROOT.name != "two_body":
    raise RuntimeError("No se encontró la carpeta two_body")

PARENT = PROJECT_ROOT.parent  # directorio que contiene a two_body
if str(PARENT) not in sys.path:
    sys.path.insert(0, str(PARENT))

print("PYTHONPATH += ", PARENT)
```

```
PYTHONPATH +=
C:\Users\emicr\Documents\CODIGOS_FUENTES\TrabajoTerminal\collision_of_two_bodies
```

1.2 Dependencias y utilidades clave

Cargamos los componentes fundamentales del flujo: - `Config` y utilidades de seeding para reproducibilidad. - El `ContinuousOptimizationController`, visualizadores 2D/3D y el adaptador REBOUND base. - `numpy` y `pathlib` para manejar resultados y artefactos en disco.

[2]:

```python
from two_body import Config, set_global_seeds
from two_body.core.telemetry import setup_logger
from two_body.logic.controller import ContinuousOptimizationController
from two_body.presentation.visualization import Visualizer as PlanarVisualizer
from two_body.presentation.triDTry import Visualizer as Visualizer3D
from two_body.simulation.rebound_adapter import ReboundSim
import numpy as np
from pathlib import Path
```

1.3 Instrumentación de rendimiento

Activamos trazas de tiempo a nivel de bloque (`time_block`) y preparamos utilidades para recuperar los CSV agregados. Esto permite auditar el costo de cada fase del pipeline directamente desde el notebook.

[3]:

```python
import os
os.environ["PERF_TIMINGS_ENABLED"] = "1"
os.environ.setdefault("PERF_TIMINGS_JSONL", "0")

from two_body.perf_timings.timers import time_block
from two_body.perf_timings import latest_timing_csv, read_timings_csv,␣
 ↪parse_sections_arg, filter_rows
```

1.4 Formato de logging para el notebook

Registramos un `logging.Handler` personalizado que acumula los mensajes y los muestra con `display(Markdown(...))`, manteniendo limpio el flujo de salida mientras corren las generaciones del GA.

[4]:

```python
import logging
from IPython.display import display, Markdown

class NotebookHandler(logging.Handler):
    def __init__(self):
        super().__init__()
        self.lines = []

    def emit(self, record):
        msg = self.format(record)
        self.lines.append(msg)
        print(msg)  # aparece en la celda conforme avanza

handler = NotebookHandler()
```

```
handler.setFormatter(logging.Formatter("[%(asctime)s] %(levelname)s -␣
 ↪%(message)s"))

logger = setup_logger(level="DEBUG")
logger.handlers.clear()          # quita otros handlers previos
logger.addHandler(handler)
logger.setLevel(logging.DEBUG)
```

1.5 Configuración del escenario "Sitnikov modificado"

Definimos el diccionario `case` con: - Integración a largo plazo (`t_end_long = 6000`, `dt = 0.08`) usando `whfast`. - Condiciones iniciales simétricas para el binario y un tercer cuerpo con desplazamiento ligero en Y/Z. - Límites estrechos para las masas estelares y un rango reducido para el tercer cuerpo. - Penalización de periodicidad (`periodicity_weight = 0.08`) y presupuesto de evaluación amplio para que el GA explore soluciones estables.

[ ]:
```
case = {
    # Integración (UA, años, masas solares)
    "t_end_short": 0.5,            # ~6 órbitas de la binaria
    "t_end_long": 4.0,             # verificación en un horizonte de 2 años
    "dt": 2.0e-4,                  # ~0.18 días; estable para IAS15
    "integrator": "ias15",

    # Condiciones iniciales (Sitnikov físico)
    "r0": (
        (-0.5, 0.0, 0.0),          # estrellas separadas 1 UA
        (0.5, 0.0, 0.0),
        (0.0, 0.0, 0.04),          # planeta / partícula sobre el eje Z
    ),
    "v0": (
        (0.0, -4.442882938, 0.0),  # binaria circular (omega = √(G·2 / 1^3))
        (0.0, 4.442882938, 0.0),
        (0.0, 0.0, 0.90),          # impulso inicial para disparar el movimiento
    ),

    # Parámetros físicos
    "mass_bounds": (
        (0.8, 1.2),              # estrella 1 ~1 Msun
        (0.8, 1.2),              # estrella 2 ~1 Msun
        (2.5e-6, 5.0e-6),          # cuerpo pequeño tipo Tierra-Júpiter trozo
    ),
    "G": 39.47841760435743,        # 4*pi^2 en unidades astronómicas
    "periodicity_weight": 0.05,    # penaliza deriva moderada (Δr + Δv)

    # GA / búsqueda continua
    "pop_size": 180,
    "n_gen_step": 5,
    "crossover": 0.85,
    "mutation": 0.2,
    "elitism": 2,
    "seed": 1234,

    # Control de ejecución (prueba rápida)
    "max_epochs": 50,
    "top_k_long": 12,
    "stagnation_window": 5,
    "stagnation_tol": 1.25e-4,
    "local_radius": 0.04,
    "radius_decay": 0.85,
    "time_budget_s": 1800.0,
    "eval_budget": 16000,

    # Artefactos / salida
    "artifacts_dir": "artifacts/sitnikov_real",
    "save_plots": True,
    "headless": False,
}
```

[6]:

```
from two_body.logic.controller import ContinuousOptimizationController
from two_body.core.config import Config
from two_body.core.telemetry import setup_logger
from two_body.core.cache import HierarchicalCache

cfg = Config(**case)
logger = setup_logger()
```

1.6 Adaptador especializado de REBOUND

Subclasamos `ReboundSim` para imponer la restricción de Sitnikov: tras cada paso, fijamos `x = vx = 0` del tercer cuerpo de modo que la partícula se mantenga sobre el eje Z. Mediante un monkeypatch temporal logramos que el `FitnessEvaluator` utilice este integrador personalizado durante todo el experimento.

[ ]:

```
import rebound

class SitnikovReboundSim(ReboundSim):
    def setup_simulation(self, *args, **kwargs):
        sim = super().setup_simulation(*args, **kwargs)

        def clamp(_sim_ptr=None):
            if len(sim.particles) > 2:
                particle = sim.particles[2]
                particle.x = 0.0
                particle.vx = 0.0
        sim.post_timestep_modifications = clamp
        return sim

# Patch para que FitnessEvaluator use el adaptador sitnikov
from two_body.simulation import rebound_adapter
rebound_adapter.ReboundSim = SitnikovReboundSim
```

```
c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-
packages\rebound\__init__.py:58: UserWarning: pkg_resources is deprecated as an
API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The
pkg_resources package is slated for removal as early as 2025-11-30. Refrain from
using this package or pin to Setuptools<81.
  import pkg_resources
```

[8]:

```
print(cfg.mass_bounds, cfg.max_epochs, cfg.eval_budget)
```

```
((0.8, 1.2), (0.8, 1.2), (2.5e-06, 5e-06)) 50 16000
```

1.7 Ejecución del controlador híbrido

Creamos la configuración, el `logger` y el `ContinuousOptimizationController`. Usamos `time_block("notebook_run")` para medir la ejecución completa (GA + refinamiento), registramos los eventos relevantes y recopilamos el mejor individuo encontrado.

[9]:

```
with time_block("notebook_run", extra={"source": "Caso01.ipynb"}):
    controller = ContinuousOptimizationController(cfg, logger=logger)
    results = controller.run()
```

```
[2025-11-02 18:03:22,252] INFO - Starting optimization | pop=180 | dims=3 |
time_budget=1800.0s | eval_budget=16000
[2025-11-02 18:03:35,304] INFO - Epoch 0 | new global best (short)
lambda=-12.329815 | fitness=11.034798 | penalty=25.900326 | masses=(0.910586,
1.120749, 5e-06)
[2025-11-02 18:03:42,734] INFO - Epoch 0 complete | lambda_short=-12.329815 |
fitness_short=11.034798 | lambda_best=-12.329815 | fitness_best=11.034798 |
evals short/long=180/12 | total evals=192 | radius=0.0400
[2025-11-02 18:03:56,015] INFO - Epoch 1 | new global best (short)
lambda=-12.831173 | fitness=11.453951 | penalty=27.544443 | masses=(0.801784,
1.161022, 4e-06)
[2025-11-02 18:04:03,206] INFO - Epoch 1 complete | lambda_short=-12.831173 |
fitness_short=11.453951 | lambda_best=-12.831173 | fitness_best=11.453951 |
evals short/long=180/12 | total evals=384 | radius=0.0400
[2025-11-02 18:04:24,257] INFO - Epoch 2 complete | lambda_short=-11.329133 |
fitness_short=10.068536 | lambda_best=-12.831173 | fitness_best=11.453951 |
evals short/long=180/12 | total evals=576 | radius=0.0400
[2025-11-02 18:04:45,245] INFO - Epoch 3 complete | lambda_short=-12.774451 |
fitness_short=11.406101 | lambda_best=-12.831173 | fitness_best=11.453951 |
evals short/long=180/12 | total evals=768 | radius=0.0400
```

```
[2025-11-02 18:05:06,490] INFO - Epoch 4 complete | lambda_short=-7.726118 |
fitness_short=6.345156 | lambda_best=-12.831173 | fitness_best=11.453951 | evals
short/long=180/12 | total evals=960 | radius=0.0400
[2025-11-02 18:05:27,546] INFO - Epoch 5 complete | lambda_short=-9.817725 |
fitness_short=8.501798 | lambda_best=-12.831173 | fitness_best=11.453951 | evals
short/long=180/12 | total evals=1152 | radius=0.0400
[2025-11-02 18:05:48,448] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:05:48,448] INFO - Epoch 6 complete | lambda_short=-11.229010 |
fitness_short=9.848760 | lambda_best=-12.831173 | fitness_best=11.453951 | evals
short/long=180/12 | total evals=1344 | radius=0.0340
[2025-11-02 18:06:01,812] INFO - Epoch 7 | new global best (short)
lambda=-14.124085 | fitness=12.706755 | penalty=28.346593 | masses=(0.8,
1.194372, 5e-06)
[2025-11-02 18:06:08,790] INFO - Epoch 7 complete | lambda_short=-14.124085 |
fitness_short=12.706755 | lambda_best=-14.124085 | fitness_best=12.706755 |
evals short/long=180/12 | total evals=1536 | radius=0.0340
[2025-11-02 18:06:29,393] INFO - Epoch 8 complete | lambda_short=-13.828951 |
fitness_short=12.469175 | lambda_best=-14.124085 | fitness_best=12.706755 |
evals short/long=180/12 | total evals=1728 | radius=0.0340
[2025-11-02 18:06:49,902] INFO - Epoch 9 complete | lambda_short=-11.977930 |
fitness_short=10.626676 | lambda_best=-14.124085 | fitness_best=12.706755 |
evals short/long=180/12 | total evals=1920 | radius=0.0340
[2025-11-02 18:07:10,392] INFO - Epoch 10 complete | lambda_short=-13.281354 |
fitness_short=11.919402 | lambda_best=-14.124085 | fitness_best=12.706755 |
evals short/long=180/12 | total evals=2112 | radius=0.0340
[2025-11-02 18:07:30,796] INFO - Epoch 11 complete | lambda_short=-13.989107 |
fitness_short=12.605036 | lambda_best=-14.124085 | fitness_best=12.706755 |
evals short/long=180/12 | total evals=2304 | radius=0.0340
[2025-11-02 18:07:44,863] INFO - Epoch 12 | new global best (short)
lambda=-14.166024 | fitness=12.784450 | penalty=27.631473 | masses=(0.824681,
1.169188, 3e-06)
[2025-11-02 18:07:52,153] INFO - Epoch 12 complete | lambda_short=-14.166024 |
fitness_short=12.784450 | lambda_best=-14.166024 | fitness_best=12.784450 |
evals short/long=180/12 | total evals=2496 | radius=0.0340
[2025-11-02 18:08:14,692] INFO - Epoch 13 complete | lambda_short=-11.475075 |
fitness_short=10.064643 | lambda_best=-14.166024 | fitness_best=12.784450 |
evals short/long=180/12 | total evals=2688 | radius=0.0340
[2025-11-02 18:08:36,292] INFO - Epoch 14 complete | lambda_short=-13.391028 |
fitness_short=11.975246 | lambda_best=-14.166024 | fitness_best=12.784450 |
evals short/long=180/12 | total evals=2880 | radius=0.0340
[2025-11-02 18:08:58,050] INFO - Epoch 15 complete | lambda_short=-12.830185 |
fitness_short=11.432555 | lambda_best=-14.166024 | fitness_best=12.784450 |
evals short/long=180/12 | total evals=3072 | radius=0.0340
[2025-11-02 18:09:12,674] INFO - Epoch 16 | new global best (short)
lambda=-14.355680 | fitness=12.991546 | penalty=27.282663 | masses=(0.837428,
1.154931, 3e-06)
[2025-11-02 18:09:20,007] INFO - Epoch 16 complete | lambda_short=-14.355680 |
fitness_short=12.991546 | lambda_best=-14.355680 | fitness_best=12.991546 |
evals short/long=180/12 | total evals=3264 | radius=0.0340
[2025-11-02 18:09:34,932] INFO - Epoch 17 | new global best (short)
lambda=-14.963662 | fitness=13.556855 | penalty=28.136140 | masses=(0.841765,
1.194226, 3e-06)
[2025-11-02 18:09:42,712] INFO - Epoch 17 complete | lambda_short=-14.963662 |
fitness_short=13.556855 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=3456 | radius=0.0340
[2025-11-02 18:10:04,310] INFO - Epoch 18 complete | lambda_short=-14.270746 |
fitness_short=12.858681 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=3648 | radius=0.0340
[2025-11-02 18:10:26,225] INFO - Epoch 19 complete | lambda_short=-12.802707 |
fitness_short=11.423131 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=3840 | radius=0.0340
[2025-11-02 18:10:48,670] INFO - Epoch 20 complete | lambda_short=-13.646783 |
fitness_short=12.275005 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=4032 | radius=0.0340
[2025-11-02 18:11:10,673] INFO - Epoch 21 complete | lambda_short=-12.466603 |
fitness_short=11.100381 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=4224 | radius=0.0340
[2025-11-02 18:11:33,143] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:11:33,143] INFO - Epoch 22 complete | lambda_short=-14.246517 |
fitness_short=12.841128 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=4416 | radius=0.0289
[2025-11-02 18:11:55,337] INFO - Epoch 23 complete | lambda_short=-14.837286 |
fitness_short=13.428171 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=4608 | radius=0.0289
[2025-11-02 18:12:18,093] INFO - Epoch 24 complete | lambda_short=-12.804887 |
fitness_short=11.385981 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=4800 | radius=0.0289
[2025-11-02 18:12:39,930] INFO - Epoch 25 complete | lambda_short=-13.968630 |
fitness_short=12.591709 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=4992 | radius=0.0289
[2025-11-02 18:13:02,856] INFO - Epoch 26 complete | lambda_short=-14.284098 |
fitness_short=12.891658 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=5184 | radius=0.0289
[2025-11-02 18:13:25,288] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:13:25,288] INFO - Epoch 27 complete | lambda_short=-12.971980 |
fitness_short=11.554797 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=5376 | radius=0.0246
[2025-11-02 18:13:48,130] INFO - Epoch 28 complete | lambda_short=-12.814210 |
fitness_short=11.444606 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=5568 | radius=0.0246
[2025-11-02 18:14:10,505] INFO - Epoch 29 complete | lambda_short=-12.603794 |
fitness_short=11.183376 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=5760 | radius=0.0246
```

```
[2025-11-02 18:14:33,463] INFO - Epoch 30 complete | lambda_short=-13.018764 |
fitness_short=11.612363 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=5952 | radius=0.0246
[2025-11-02 18:14:55,172] INFO - Epoch 31 complete | lambda_short=-12.941780 |
fitness_short=11.544795 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=6144 | radius=0.0246
[2025-11-02 18:15:17,933] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:15:17,933] INFO - Epoch 32 complete | lambda_short=-12.598179 |
fitness_short=11.214337 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=6336 | radius=0.0209
[2025-11-02 18:15:40,281] INFO - Epoch 33 complete | lambda_short=-13.240984 |
fitness_short=11.867850 | lambda_best=-14.963662 | fitness_best=13.556855 |
evals short/long=180/12 | total evals=6528 | radius=0.0209
[2025-11-02 18:15:55,071] INFO - Epoch 34 | new global best (short)
lambda=-15.703743 | fitness=14.284699 | penalty=28.380877 | masses=(0.832372,
1.2, 5e-06)
[2025-11-02 18:16:03,213] INFO - Epoch 34 complete | lambda_short=-15.703743 |
fitness_short=14.284699 | lambda_best=-15.703743 | fitness_best=14.284699 |
evals short/long=180/12 | total evals=6720 | radius=0.0209
[2025-11-02 18:16:26,167] INFO - Epoch 35 complete | lambda_short=-13.094398 |
fitness_short=11.678969 | lambda_best=-15.703743 | fitness_best=14.284699 |
evals short/long=180/12 | total evals=6912 | radius=0.0209
[2025-11-02 18:16:48,859] INFO - Epoch 36 complete | lambda_short=-14.095560 |
fitness_short=12.687651 | lambda_best=-15.703743 | fitness_best=14.284699 |
evals short/long=180/12 | total evals=7104 | radius=0.0209
[2025-11-02 18:17:11,458] INFO - Epoch 37 complete | lambda_short=-13.386728 |
fitness_short=11.982014 | lambda_best=-15.703743 | fitness_best=14.284699 |
evals short/long=180/12 | total evals=7296 | radius=0.0209
[2025-11-02 18:17:34,660] INFO - Epoch 38 complete | lambda_short=-13.501586 |
fitness_short=12.079813 | lambda_best=-15.703743 | fitness_best=14.284699 |
evals short/long=180/12 | total evals=7488 | radius=0.0209
[2025-11-02 18:17:57,707] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:17:57,707] INFO - Epoch 39 complete | lambda_short=-13.394158 |
fitness_short=11.988888 | lambda_best=-15.703743 | fitness_best=14.284699 |
evals short/long=180/12 | total evals=7680 | radius=0.0177
[2025-11-02 18:18:12,935] INFO - Epoch 40 | new global best (short)
lambda=-15.714886 | fitness=14.292219 | penalty=28.453351 | masses=(0.819164,
1.2, 3e-06)
[2025-11-02 18:18:20,884] INFO - Epoch 40 complete | lambda_short=-15.714886 |
fitness_short=14.292219 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=7872 | radius=0.0177
[2025-11-02 18:18:43,695] INFO - Epoch 41 complete | lambda_short=-14.882700 |
fitness_short=13.482276 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=8064 | radius=0.0177
[2025-11-02 18:19:05,978] INFO - Epoch 42 complete | lambda_short=-15.034034 |
fitness_short=13.612395 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=8256 | radius=0.0177
[2025-11-02 18:19:27,952] INFO - Epoch 43 complete | lambda_short=-13.470992 |
fitness_short=12.055079 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=8448 | radius=0.0177
[2025-11-02 18:19:50,016] INFO - Epoch 44 complete | lambda_short=-14.144508 |
fitness_short=12.745551 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=8640 | radius=0.0177
[2025-11-02 18:20:12,090] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:20:12,091] INFO - Epoch 45 complete | lambda_short=-14.178310 |
fitness_short=12.754140 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=8832 | radius=0.0151
[2025-11-02 18:20:33,916] INFO - Epoch 46 complete | lambda_short=-12.709361 |
fitness_short=11.291695 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=9024 | radius=0.0151
[2025-11-02 18:20:55,690] INFO - Epoch 47 complete | lambda_short=-15.569237 |
fitness_short=14.161830 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=9216 | radius=0.0151
[2025-11-02 18:21:17,141] INFO - Epoch 48 complete | lambda_short=-12.968460 |
fitness_short=11.555418 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=9408 | radius=0.0151
[2025-11-02 18:21:39,102] INFO - Epoch 49 complete | lambda_short=-15.033943 |
fitness_short=13.611259 | lambda_best=-15.714886 | fitness_best=14.292219 |
evals short/long=180/12 | total evals=9600 | radius=0.0151
[2025-11-02 18:21:39,102] INFO - Optimization completed | epochs=50 | evals=9600
| best lambda=-15.714886 | wall=1096.8s
```

1.8 Métricas de referencia y resultado óptimo

Calculamos el fitness del centro de los intervalos (`center`) como línea base y lo comparamos con la solución óptima (`results["best"]`). También recuperamos `metrics` para inspeccionar la evolución de por época y cualquier otra estadística almacenada por el controlador.

[10]:

```
metrics = controller.metrics
results
```

[10]:

```
{'status': 'completed',
 'best': {'masses': [0.8191638121070504, 1.2, 2.5e-06],
  'lambda': -15.714886129963242,
  'fitness': 14.292218582279823,
  'm1': 0.8191638121070504,
  'm2': 1.2,
  'm3': 2.5e-06},
 'evals': 9600,
 'epochs': 50}
```

1.9 Seguimiento de y reconstrucción de trayectorias

Visualizamos: - La secuencia `metrics.best_lambda_per_epoch` y, opcionalmente, un suavizado para detectar tendencias. - Las trayectorias 3D integradas con las masas óptimas (`viz_3d.animate_3d`) y cualquier proyección 2D rápida (`viz_planar.quick_view`) para confirmar la estabilidad obtenida.

[ ]:

```python
from two_body.core.cache import HierarchicalCache
from two_body.logic.fitness import FitnessEvaluator

cache = HierarchicalCache()
evaluator = FitnessEvaluator(cache, cfg)

original_masses = (0.8, 1.2, 2.5e-6)
center = original_masses
#center = tuple((lo + hi) / 2.0 for lo, hi in cfg.mass_bounds)

baseline_fits, baseline_details = evaluator.evaluate_batch(
    [center],
    horizon="long",
    return_details=True,
)
baseline_fit = baseline_fits[0]
baseline_lambda = baseline_details[0].get("lambda")
if baseline_lambda is None or not np.isfinite(baseline_lambda):
    baseline_lambda = -baseline_fit

best_payload = results.get("best", {})
best_fit = best_payload.get("fitness")
best_lambda = best_payload.get("lambda")
if best_lambda is None and best_fit is not None:
    best_lambda = -best_fit

print(
    f"lambda inicial = {baseline_lambda:.6f}, "
    f"lambda optimo = {best_lambda if best_lambda is not None else 'N/A'}"
)
```

```
lambda inicial = -0.984161, lambda optimo = -15.714886129963242
```

[12]:

```python
sim_builder = ReboundSim(G=cfg.G, integrator=cfg.integrator)
best_masses = tuple(results["best"]["masses"])

def _slice_vectors(vectors, count):
    if len(vectors) < count:
        raise ValueError("Config no tiene suficientes vectores iniciales")
    return tuple(tuple(float(coord) for coord in vectors[i]) for i in␣
 ↪range(count))
```

```
r0 = _slice_vectors(cfg.r0, len(best_masses))
v0 = _slice_vectors(cfg.v0, len(best_masses))

sim = sim_builder.setup_simulation(best_masses, r0, v0)
traj = sim_builder.integrate(sim, t_end=cfg.t_end_long, dt=cfg.dt)
print("Trayectoria calculada con masas óptimas.")
print(traj.shape)
print(traj[-1])
xyz_tracks = [traj[:, i, :3] for i in range(traj.shape[1])]
```

```
Trayectoria calculada con masas óptimas.
(20000, 3, 6)
[[-6.67448908e-02  5.85504566e-01 -1.93614735e-07 -5.29717281e+00
  -5.53192498e-01  3.53308421e-08]
 [ 4.55617540e-02 -3.99686140e-01  1.17620941e-07  3.61605193e+00
   3.77617569e-01 -2.88401495e-06]
 [ 3.57753801e-01 -3.13603497e-01  6.98282203e-03 -4.02073893e+00
   5.67702912e+00  1.37275048e+00]]
```

[13]:

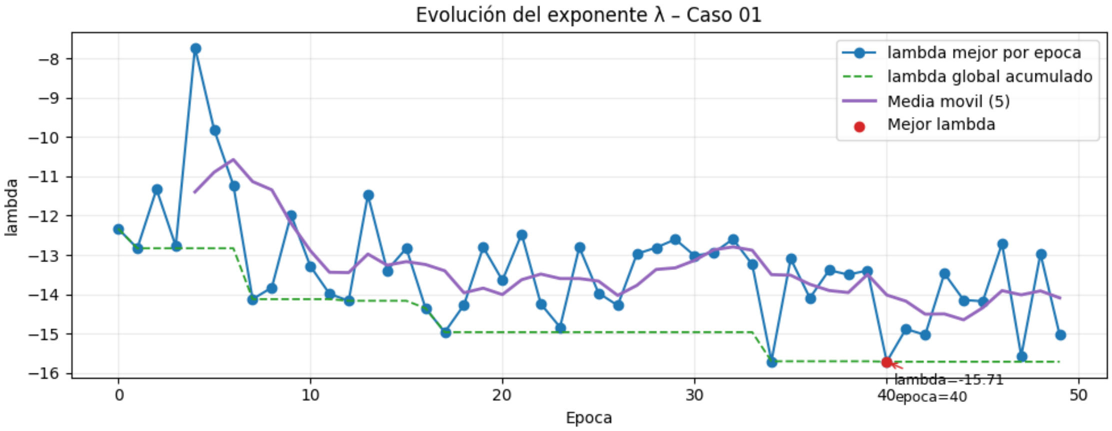
```
from two_body.scripts.demo_tierra import (
    summarize_trajectory,
    compute_total_energy,
    estimate_orbital_period,
)

summarize_trajectory(
    logger=logger,
    traj=traj,
    masses=best_masses,
    cfg=cfg,
)
```

```
[2025-11-02 18:21:42,150] INFO - Resumen de simulacion
[2025-11-02 18:21:42,150] INFO -   pasos=20000 | dt=0.000200 anos | duracion
total=4.000 anos
[2025-11-02 18:21:42,150] INFO -   masas=(0.8191638121070504, 1.2, 2.5e-06)
(M_sun) | G=39.478418
[2025-11-02 18:21:42,152] INFO -   centro de masa: desplazamiento maximo =
1.932e-15 UA
[2025-11-02 18:21:42,173] INFO -   cuerpo 0 -> radio[min, max]=[0.5831, 0.5943]
UA | radio sigma=3.9460e-03 | velocidad media=5.3325 UA/ano
[2025-11-02 18:21:42,186] INFO -   cuerpo 1 -> radio[min, max]=[0.3981, 0.4057]
UA | radio sigma=2.6938e-03 | velocidad media=3.6402 UA/ano
[2025-11-02 18:21:42,193] INFO -   cuerpo 2 -> radio[min, max]=[0.0185, 0.8236]
UA | radio sigma=2.3487e-01 | velocidad media=11.4824 UA/ano
[2025-11-02 18:21:42,234] INFO -   energia total (media)=-1.958813e+01 |
variacion relativa=1.632e-15
```

```
[2025-11-02 18:21:42,249] INFO -   periodo orbital estimado para la Tierra ~=
0.693836 anos
[2025-11-02 18:21:42,249] INFO -   error relativo vs 1 ano ~= 3.062e-01
```

[14]:

```
viz_3d = Visualizer3D(headless=cfg.headless)

_ = viz_3d.plot_lambda_evolution(
    lambda_history=metrics.best_lambda_per_epoch,
    epoch_history=metrics.epoch_history,
    title="Evolución del exponente  - Caso 01",
    moving_average_window=5,   # opcional
)
```

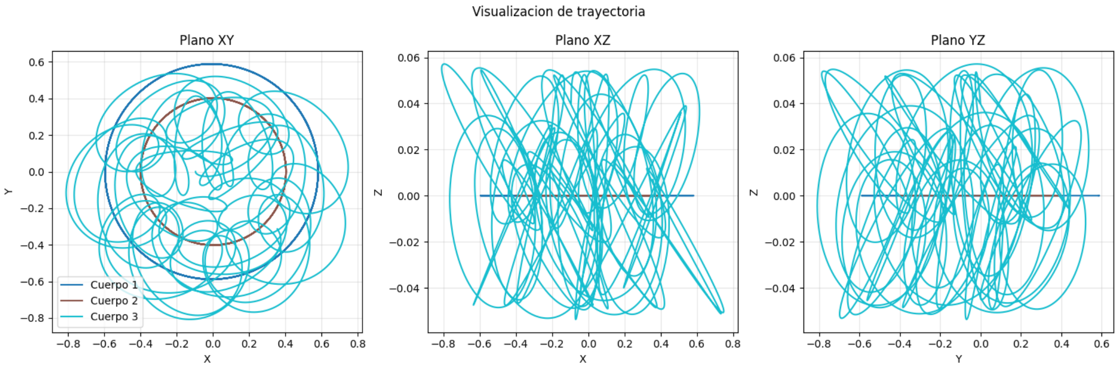


[15]:

```
sim_builder = ReboundSim(G=cfg.G, integrator=cfg.integrator)
best_masses = tuple(results["best"]["masses"])

def _slice_vectors(vectors, count):
    if len(vectors) < count:
        raise ValueError("Config no tiene suficientes vectores iniciales")
    return tuple(tuple(float(coord) for coord in vectors[i]) for i in␣
 ↪range(count))

r0 = _slice_vectors(cfg.r0, len(best_masses))
v0 = _slice_vectors(cfg.v0, len(best_masses))

sim = sim_builder.setup_simulation(best_masses, r0, v0)
traj = sim_builder.integrate(sim, t_end=cfg.t_end_long, dt=cfg.dt)
xyz_tracks = [traj[:, i, :3] for i in range(traj.shape[1])]
```

[16]:

```
viz_planar = PlanarVisualizer(headless=cfg.headless)
_ = viz_planar.quick_view(xyz_tracks)
```

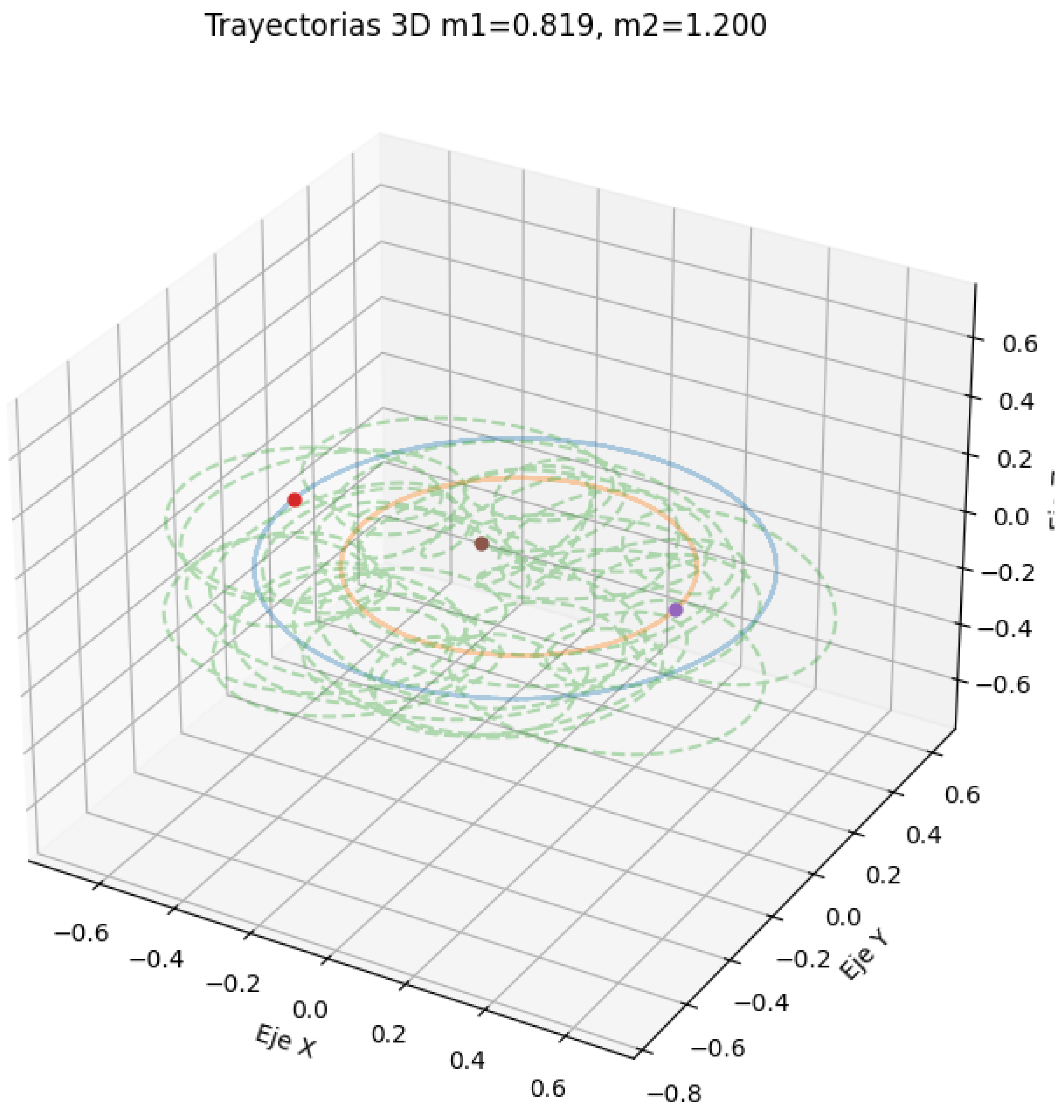


[17]:

```
from IPython.display import HTML

import matplotlib as mpl
mpl.rcParams['animation.embed_limit'] = 1024  # MB
```

[18]:

```
viz_3d = Visualizer3D(headless=False)

anim = viz_3d.animate_3d(
    trajectories=xyz_tracks,
    interval_ms=50,
    title=f"Trayectorias 3D m1={best_masses[0]:.3f}, m2={best_masses[1]:.3f}",
    total_frames=len(xyz_tracks[0]),
)
#HTML(anim.to_jshtml())
```



1.10 Exportación de animaciones

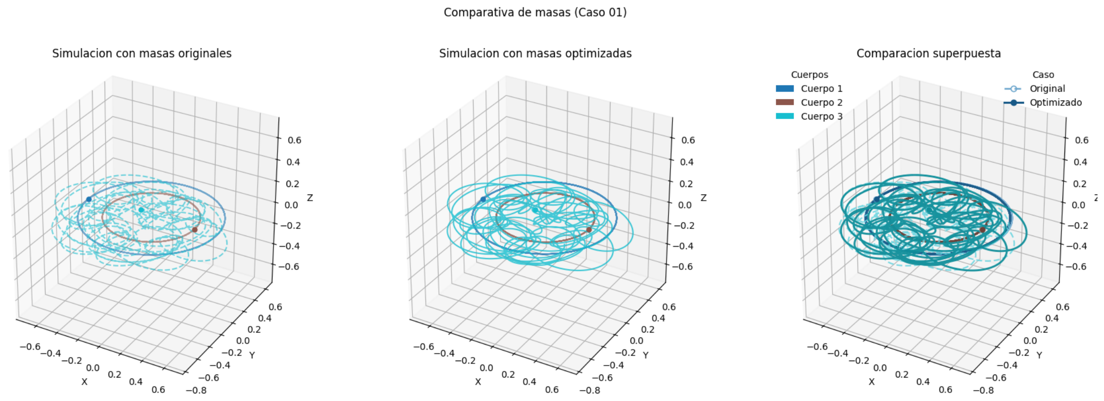
Configuramos un `FFMpegWriter`, nos aseguramos de que el directorio `artifacts/caso02` exista y persistimos los MP4 de la trayectoria final y de la comparación de masas. Ajusta `fps`, `bitrate` o el preset de ffmpeg si necesitas reducir el tiempo de render.

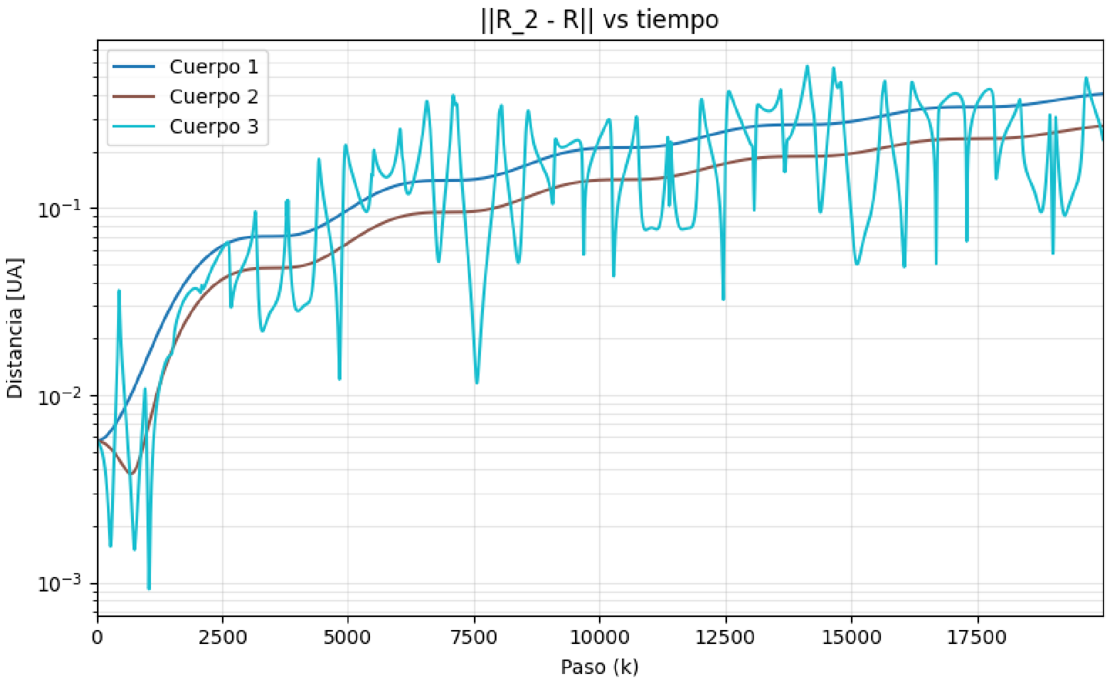
[19]:
```
from matplotlib.animation import FFMpegWriter  # o PillowWriter para GIF

writer = FFMpegWriter(fps=1000 // 50, bitrate=2400)   # fps = 1000/interval_ms
output_path = Path("artifacts/caso02")                # ajusta a tu gusto
output_path.mkdir(parents=True, exist_ok=True)
```

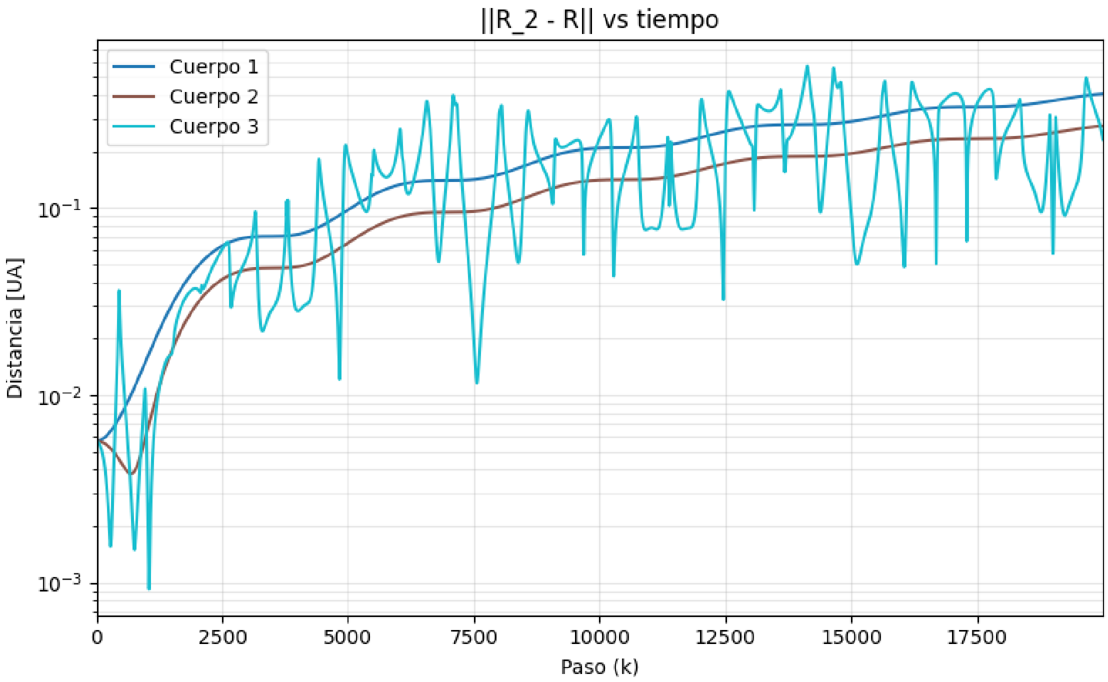
```
anim.save(output_path / "trayectoria_optima.mp4", writer=writer)
```

```
[ ]: # Trayectoria con las masas originales (center)
     sim_orig = sim_builder.setup_simulation(center, r0, v0)
     traj_original = sim_builder.integrate(sim_orig, t_end=cfg.t_end_long, dt=cfg.dt)

     traj_opt = traj
```

```
[ ]: orig_tracks = [traj_original[:, i, :3] for i in range(traj_original.shape[1])]
     opt_tracks = [traj_opt[:, i, :3] for i in range(traj_opt.shape[1])]

     anim_mass = viz_3d.plot_mass_comparison(
         original_tracks=orig_tracks,
         optimized_tracks=opt_tracks,
         original_masses=center,
         optimized_masses=best_masses,
         body_labels=[f"Cuerpo {i+1}" for i in range(len(opt_tracks))],
         dt=cfg.dt,
         title="Comparativa de masas (Caso 01)",
     )

     if anim_mass is not None:
         dist_fig = viz_3d.plot_mass_distance_evolution(
             comparison_data=anim_mass.mass_comparison_data,
             title="||R_2 - R|| vs tiempo",
         )
         if dist_fig is not None:
             display(dist_fig)
         # display(HTML(anim_mass.to_jshtml()))  # descomenta para ver la animación␣
      ↪embebida
```







[22]:
```
anim_mass.save(output_path / "comparativa_masas.mp4", writer=writer)
```

1.11 Reporte de tiempos

Leemos el CSV más reciente generado por la instrumentación, mostramos las primeras filas y podemos agregar estadísticas por sección para identificar cuellos de botella del pipeline en este escenario.

```
[23]: import pandas as pd

csv_path = latest_timing_csv()
display(f"Usando CSV: {csv_path}")

rows = read_timings_csv(csv_path)
df = pd.DataFrame(rows)
display(df.head(10))

# Estadísticas rápidas por sección
section_stats = (
    df.groupby("section")["duration_us"]
    .agg(["count", "mean", "sum"])
    .sort_values("sum", ascending=False)
)
section_stats
```

```
'Usando CSV: C:
 ↪\\Users\\emicr\\Documents\\CODIGOS_FUENTES\\TrabajoTerminal\\collision_of_two_bodies\\two_bo
 ↪csv'

                                 run_id  epoch  batch_id  individual_id  \
0  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
1  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
2  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
3  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
4  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
5  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
6  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
7  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
8  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0
9  ddf0b833-a5c4-4572-b8c0-19a388d2f328      0         0              0

           section        start_ns          end_ns  duration_us  \
0  simulation_step  19529689749600  19529689791500           41
1  simulation_step  19529689823000  19529689841300           18
2  simulation_step  19529689854900  19529689868200           13
3  simulation_step  19529689878500  19529689890900           12
4  simulation_step  19529689898600  19529689910900           12
5  simulation_step  19529689918600  19529689931400           12
6  simulation_step  19529689938900  19529689951000           12
7  simulation_step  19529689958000  19529689971200           13
8  simulation_step  19529689977700  19529689989400           11
```

```
9  simulation_step  19529689996600  19529690008400          11

                                               extra
0     {'step': 0, 'dt': 0.0002, 't_target': 0.0002}
1     {'step': 1, 'dt': 0.0002, 't_target': 0.0004}
2  {'step': 2, 'dt': 0.0002, 't_target': 0.000600…
3     {'step': 3, 'dt': 0.0002, 't_target': 0.0008}
4      {'step': 4, 'dt': 0.0002, 't_target': 0.001}
5  {'step': 5, 'dt': 0.0002, 't_target': 0.001200…
6     {'step': 6, 'dt': 0.0002, 't_target': 0.0014}
7     {'step': 7, 'dt': 0.0002, 't_target': 0.0016}
8  {'step': 8, 'dt': 0.0002, 't_target': 0.001800…
9      {'step': 9, 'dt': 0.0002, 't_target': 0.002}
```

[23]:

```
                         count          mean         sum
section
notebook_run                 1  1.096908e+09  1096907618
batch_eval                 101  1.083186e+07  1094018184
fitness_eval              9601  1.139131e+05  1093680126
lyapunov_compute          9593  1.137711e+05  1091406360
simulation_step       34559992  1.975024e+01   682568261
ga_main                     50  5.628382e+04     2814191
crossover                 1170  6.534650e+02      764554
selection_tournament      1170  2.775957e+02      324787
mutation                  1170  1.874342e+02      219298
```

[ ]:

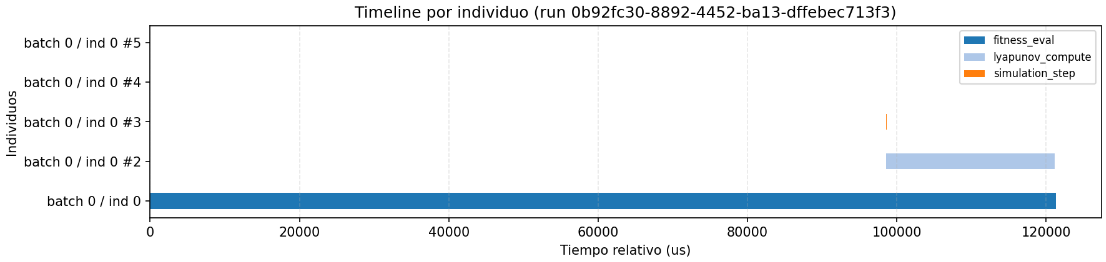
```
import os
import subprocess
from pathlib import Path
from IPython.display import Image, display

PROJECT_ROOT = Path.cwd()
while PROJECT_ROOT.name != "two_body" and PROJECT_ROOT.parent != PROJECT_ROOT:
    PROJECT_ROOT = PROJECT_ROOT.parent

env = os.environ.copy()
env["PYTHONPATH"] = str(PROJECT_ROOT)

run_id = df["run_id"].iloc[0]
cmd = [
    sys.executable,
    "scripts/plot_timings.py",
    "--run-id", str(run_id),
    "--top-n", "5",
]
```

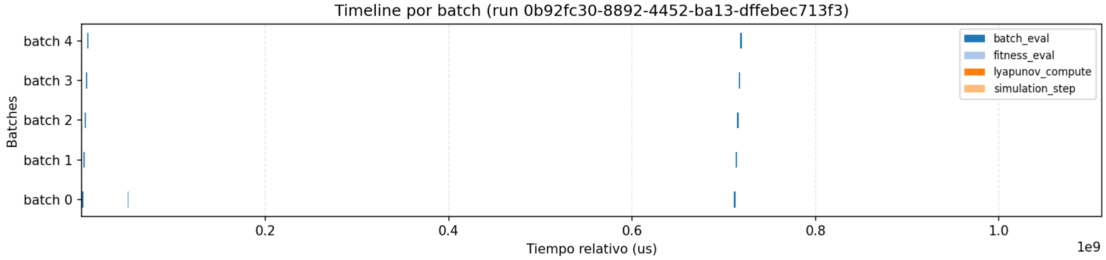
```
print("Ejecutando:", " ".join(cmd))
result = subprocess.run(cmd, cwd=PROJECT_ROOT, env=env, text=True,␣
  ↪capture_output=True)
print(result.stdout)
print(result.stderr)
result.check_returncode()

reports_dir = PROJECT_ROOT / "reports"
```

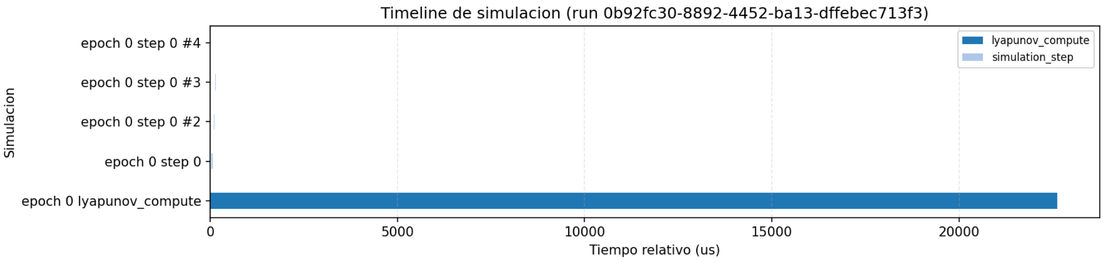
```
Ejecutando: c:\Users\emicr\anaconda3\envs\grav2body\python.exe
scripts/plot_timings.py --run-id ddf0b833-a5c4-4572-b8c0-19a388d2f328 --top-n 5
```

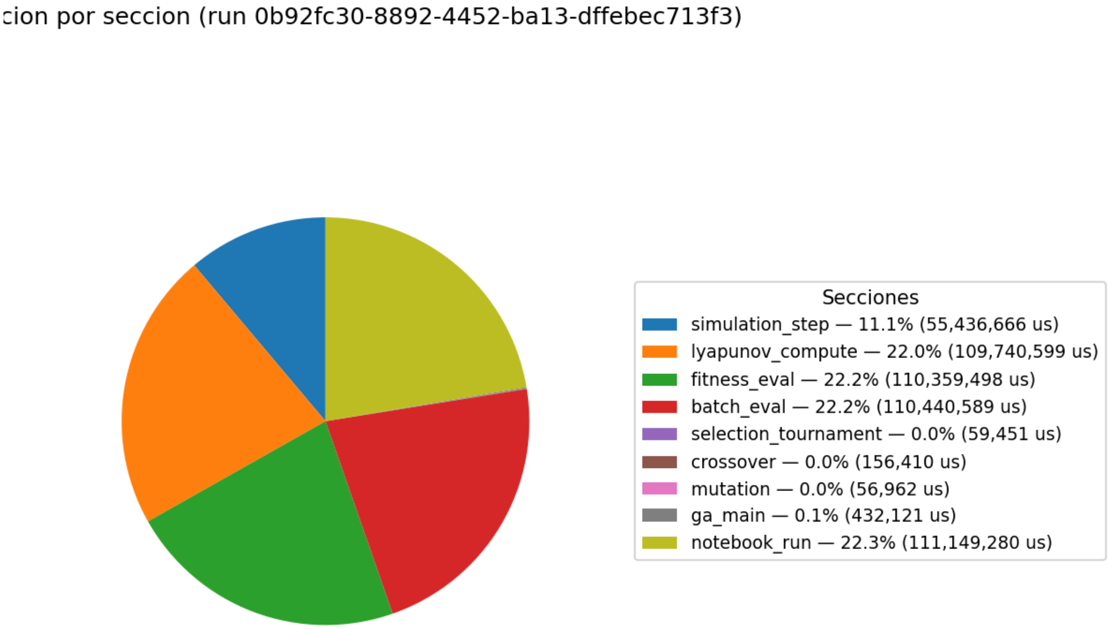
[ ]:

```
display(
    Image(filename=str(reports_dir / f"timeline_by_individual_{run_id}.png")),
    Image(filename=str(reports_dir / f"timeline_by_batch_{run_id}.png")),
    Image(filename=str(reports_dir / f"timeline_simulation_{run_id}.png")),
    Image(filename=str(reports_dir / f"pie_sections_{run_id}.png")),
)
```






Timeline de simulacion (run 0b92fc30-8892-4452-ba13-dffebec713f3)


cion por seccion (run 0b92fc30-8892-4452-ba13-dffebec713f3)