Caso01

November 10, 2025

1 Evaluación del caso “Binaria + planeta cerca del radio de Hill”

En esta sección medimos cómo afecta la optimización al exponente de Lyapunov del sistema. Partimos de un estado base (masas en el centro de sus `mass_bounds`) para obtener un de referencia y lo comparamos con el mejor individuo que devolvió el GA.
Luego integramos la trayectoria con las masas óptimas y visualizamos el comportamiento de los tres cuerpos, poniendo especial atención a los encuentros cercanos del “planeta” con el binario.

> **Interpretación** - Un más pequeño indica una dinámica menos caótica. - El gráfico permite comprobar si el planeta logra mantenerse en órbita o si termina inestable por la proximidad al radio de Hill.

1.1 Preparación del entorno

Aseguramos que el directorio raíz del proyecto esté disponible en `sys.path` para poder importar los módulos internos sin problemas, independientemente de desde dónde se ejecute el notebook.

[1]:
```
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 principales

Importamos los componentes clave del pipeline: - `Config` y utilidades de seeding. - El controlador híbrido (GA + refinamiento). - Herramientas de visualización y simulación REBOUND. - `numpy`

para cualquier análisis adicional.

[2]:
```
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  #
```

[3]:
```
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
```

[ ]:
```
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)

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

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

1.3 Configuración del escenario “Binaria + planeta cerca del radio de Hill”

Definimos la escena de estudio: - Dos estrellas de masas distintas orbitando el baricentro. - Un planeta tipo Júpiter ubicado cerca del radio de Hill. - Hiperparámetros del GA y de la fase continua orientados a detectar combinaciones de masas que mitiguen el caos.

[ ]:
```
# ["Binaria con planeta (Hill) - órbita periódica"]
case = {
    # Simulación (UA, años, masas solares)
    "t_end_short": 0.5,
    "t_end_long": 4.0,
    "dt": 0.0002,
    "integrator": "ias15",
    "r0": (
        (-0.09, 0.0, 0.0),
        (0.11, 0.0, 0.0),
        (0.52, 0.0, 0.0),
    ),
    "v0": (
        (0.0, 8.9411294392, 0.0),
        (0.0, -10.9280470924, 0.0),
        (0.0, 12.3223401880, 0.0),
    ),

    # Parámetros físicos (masas solares)
    "mass_bounds": (
        (0.9, 1.2),
        (0.7, 1.0),
        (8.5e-4, 1.2e-3),
    ),
    "G": 39.47841760435743,
    "x0": (-0.09, 0.0, 0.0, 8.94, 0.11, 0.0, 0.0, -10.93),
    "periodicity_weight": 500.0,

    # Algoritmo genético
    "pop_size": 180,
    "n_gen_step": 5,
    "crossover": 0.9,
    "mutation": 0.2,
    "selection": "tournament",
    "elitism": 2,
    "seed": 9871,

    # Optimización continua
    "max_epochs": 50,
    "top_k_long": 12,
    "stagnation_window": 6,
    "stagnation_tol": 1.25e-4,
    "local_radius": 0.04,
    "radius_decay": 0.85,
    "time_budget_s": 1800.0,
    "eval_budget": 16000,
```

```
    # Backend / cache
    "use_gpu": "false",
    "batch_size": 96,
    "cache_exact_max": 600,
    "cache_approx_max": 1800,

    # I/O
    "artifacts_dir": "artifacts/hill_planet_periodic",
    "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()
```

[7]:
```
from two_body.simulation.rebound_adapter import ReboundSim
from two_body.simulation.lyapunov import LyapunovEstimator

masses = tuple(np.mean(bounds) for bounds in cfg.mass_bounds)
sim = ReboundSim(G=cfg.G, integrator=cfg.integrator).setup_simulation(
    masses, cfg.r0[:len(masses)], cfg.v0[:len(masses)]
)
estimator = LyapunovEstimator()
ret = estimator.mLCE({"sim": sim, "dt": cfg.dt, "t_end": cfg.t_end_short,␣
 ↪"masses": masses})
print(ret)
```

```
{'lambda': 0.9142204128931353, 'series': None, 'meta': {'steps': 2500, 'dt':
0.0002, 'n_bodies': 3, 'masses': (1.05, 0.85, 0.0010249999999999999), 'impl':
'rebound_megno'}}
```

```
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]:
```
import logging
import importlib

from two_body.core.telemetry import setup_logger
import two_body.logic.fitness as fitness_mod
```

```
importlib.reload(fitness_mod)                  # recoge el código nuevo
from two_body.logic.fitness import FitnessEvaluator  # vuelve a importar la␣
 ↪clase

logger = setup_logger(level="INFO")          # asegura nivel INFO
logger.setLevel(logging.INFO)
for handler in logger.handlers:
    handler.setLevel(logging.INFO)
```

1.4 Ejecución del optimizador

Inicializamos el controlador con la configuración anterior, habilitamos el registro de eventos y lanzamos el proceso completo de optimización. Al finalizar, presentamos los logs capturados junto con el resultado agregado (mejor combinación de masas encontrada y métricas básicas).

[9]:
```
print(cfg.mass_bounds, cfg.max_epochs, cfg.eval_budget)
```

```
((0.9, 1.2), (0.7, 1.0), (0.00085, 0.0012)) 50 16000
```

[10]:
```
with time_block("notebook_run", extra={"source": "Caso01.ipynb"}):
    controller = ContinuousOptimizationController(cfg, logger=logger)
    results = controller.run()
```

```
[2025-11-02 18:03:21,228] INFO - Starting optimization | pop=180 | dims=3 |
time_budget=1800.0s | eval_budget=16000
[2025-11-02 18:03:34,262] INFO - Epoch 0 | new global best (short)
lambda=3.001726 | fitness=-1380.676116 | penalty=2.755349 | masses=(1.066355,
0.701437, 0.000897)
[2025-11-02 18:03:40,876] INFO - Epoch 0 complete | lambda_short=3.001726 |
fitness_short=-1380.676116 | lambda_best=3.001726 | fitness_best=-1380.676116 |
evals short/long=180/12 | total evals=192 | radius=0.0400
[2025-11-02 18:03:54,120] INFO - Epoch 1 | new global best (short)
lambda=4.474038 | fitness=-1338.912969 | penalty=2.668878 | masses=(1.066355,
0.701367, 0.000897)
[2025-11-02 18:04:01,092] INFO - Epoch 1 complete | lambda_short=4.474038 |
fitness_short=-1338.912969 | lambda_best=4.474038 | fitness_best=-1338.912969 |
evals short/long=180/12 | total evals=384 | radius=0.0400
[2025-11-02 18:04:21,390] INFO - Epoch 2 complete | lambda_short=-0.623173 |
fitness_short=-2376.415188 | lambda_best=4.474038 | fitness_best=-1338.912969 |
evals short/long=180/12 | total evals=576 | radius=0.0400
[2025-11-02 18:04:41,771] INFO - Epoch 3 complete | lambda_short=-0.832624 |
fitness_short=-4766.814821 | lambda_best=4.474038 | fitness_best=-1338.912969 |
evals short/long=180/12 | total evals=768 | radius=0.0400
[2025-11-02 18:05:02,533] INFO - Epoch 4 complete | lambda_short=3.795571 |
fitness_short=-5236.210198 | lambda_best=4.474038 | fitness_best=-1338.912969 |
evals short/long=180/12 | total evals=960 | radius=0.0400
[2025-11-02 18:05:23,070] INFO - Epoch 5 complete | lambda_short=-0.508520 |
fitness_short=-9918.263572 | lambda_best=4.474038 | fitness_best=-1338.912969 |
```

```
evals short/long=180/12 | total evals=1152 | radius=0.0400
[2025-11-02 18:05:43,686] INFO - Epoch 6 complete | lambda_short=0.919650 |
fitness_short=-10835.874726 | lambda_best=4.474038 | fitness_best=-1338.912969 |
evals short/long=180/12 | total evals=1344 | radius=0.0400
[2025-11-02 18:06:04,456] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:06:04,456] INFO - Epoch 7 complete | lambda_short=2.380386 |
fitness_short=-7889.354443 | lambda_best=4.474038 | fitness_best=-1338.912969 |
evals short/long=180/12 | total evals=1536 | radius=0.0340
[2025-11-02 18:06:18,210] INFO - Epoch 8 | new global best (short)
lambda=1.877384 | fitness=-653.994142 | penalty=1.304234 | masses=(1.065034,
0.7, 0.00085)
[2025-11-02 18:06:25,450] INFO - Epoch 8 complete | lambda_short=1.877384 |
fitness_short=-653.994142 | lambda_best=1.877384 | fitness_best=-653.994142 |
evals short/long=180/12 | total evals=1728 | radius=0.0340
[2025-11-02 18:06:39,300] INFO - Epoch 9 | new global best (short)
lambda=-0.409902 | fitness=-261.166482 | penalty=0.523153 | masses=(1.065934,
0.7, 0.00085)
[2025-11-02 18:06:46,480] INFO - Epoch 9 complete | lambda_short=-0.409902 |
fitness_short=-261.166482 | lambda_best=-0.409902 | fitness_best=-261.166482 |
evals short/long=180/12 | total evals=1920 | radius=0.0340
[2025-11-02 18:07:00,369] INFO - Epoch 10 | new global best (short)
lambda=1.530453 | fitness=-258.068686 | penalty=0.513076 | masses=(1.065924,
0.7, 0.00085)
[2025-11-02 18:07:07,563] INFO - Epoch 10 complete | lambda_short=1.530453 |
fitness_short=-258.068686 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=2112 | radius=0.0340
[2025-11-02 18:07:28,275] INFO - Epoch 11 complete | lambda_short=0.860125 |
fitness_short=-274.754263 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=2304 | radius=0.0340
[2025-11-02 18:07:50,027] INFO - Epoch 12 complete | lambda_short=1.777859 |
fitness_short=-392.949102 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=2496 | radius=0.0340
[2025-11-02 18:08:12,926] INFO - Epoch 13 complete | lambda_short=3.874378 |
fitness_short=-278.551542 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=2688 | radius=0.0340
[2025-11-02 18:08:35,101] INFO - Epoch 14 complete | lambda_short=0.194685 |
fitness_short=-1014.398220 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=2880 | radius=0.0340
[2025-11-02 18:08:57,472] INFO - Epoch 15 complete | lambda_short=2.328787 |
fitness_short=-598.797722 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=3072 | radius=0.0340
[2025-11-02 18:09:19,487] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:09:19,488] INFO - Epoch 16 complete | lambda_short=2.554881 |
fitness_short=-593.437443 | lambda_best=1.530453 | fitness_best=-258.068686 |
evals short/long=180/12 | total evals=3264 | radius=0.0289
[2025-11-02 18:09:34,351] INFO - Epoch 17 | new global best (short)
```

```
lambda=7.085217 | fitness=-151.880869 | penalty=0.289591 | masses=(1.065747,
0.7, 0.0012)
[2025-11-02 18:09:42,288] INFO - Epoch 17 complete | lambda_short=7.085217 |
fitness_short=-151.880869 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=3456 | radius=0.0289
[2025-11-02 18:10:04,774] INFO - Epoch 18 complete | lambda_short=1.624455 |
fitness_short=-176.032388 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=3648 | radius=0.0289
[2025-11-02 18:10:27,617] INFO - Epoch 19 complete | lambda_short=1.392646 |
fitness_short=-230.759749 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=3840 | radius=0.0289
[2025-11-02 18:10:50,146] INFO - Epoch 20 complete | lambda_short=0.856938 |
fitness_short=-516.662691 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=4032 | radius=0.0289
[2025-11-02 18:11:12,260] INFO - Epoch 21 complete | lambda_short=1.045407 |
fitness_short=-759.070054 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=4224 | radius=0.0289
[2025-11-02 18:11:34,940] INFO - Epoch 22 complete | lambda_short=0.368082 |
fitness_short=-240.751613 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=4416 | radius=0.0289
[2025-11-02 18:11:57,090] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:11:57,090] INFO - Epoch 23 complete | lambda_short=1.305923 |
fitness_short=-186.685859 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=4608 | radius=0.0246
[2025-11-02 18:12:20,235] INFO - Epoch 24 complete | lambda_short=0.656055 |
fitness_short=-1408.636160 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=4800 | radius=0.0246
[2025-11-02 18:12:42,372] INFO - Epoch 25 complete | lambda_short=3.041637 |
fitness_short=-357.074151 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=4992 | radius=0.0246
[2025-11-02 18:13:04,859] INFO - Epoch 26 complete | lambda_short=-0.289349 |
fitness_short=-322.929832 | lambda_best=7.085217 | fitness_best=-151.880869 |
evals short/long=180/12 | total evals=5184 | radius=0.0246
[2025-11-02 18:13:19,788] INFO - Epoch 27 | new global best (short)
lambda=-3.239194 | fitness=-149.424272 | penalty=0.305327 | masses=(1.065762,
0.7, 0.0012)
[2025-11-02 18:13:27,839] INFO - Epoch 27 complete | lambda_short=-3.239194 |
fitness_short=-149.424272 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=5376 | radius=0.0246
[2025-11-02 18:13:50,365] INFO - Epoch 28 complete | lambda_short=3.634102 |
fitness_short=-272.771470 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=5568 | radius=0.0246
[2025-11-02 18:14:12,867] INFO - Epoch 29 complete | lambda_short=1.859163 |
fitness_short=-600.789154 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=5760 | radius=0.0246
[2025-11-02 18:14:35,744] INFO - Epoch 30 complete | lambda_short=7.184242 |
fitness_short=-496.945305 | lambda_best=-3.239194 | fitness_best=-149.424272 |
```

```
evals short/long=180/12 | total evals=5952 | radius=0.0246
[2025-11-02 18:14:57,556] INFO - Epoch 31 complete | lambda_short=3.635197 |
fitness_short=-212.210908 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=6144 | radius=0.0246
[2025-11-02 18:15:20,265] INFO - Epoch 32 complete | lambda_short=0.129603 |
fitness_short=-224.723429 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=6336 | radius=0.0246
[2025-11-02 18:15:42,749] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:15:42,749] INFO - Epoch 33 complete | lambda_short=0.903316 |
fitness_short=-430.999801 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=6528 | radius=0.0209
[2025-11-02 18:16:05,521] INFO - Epoch 34 complete | lambda_short=5.653590 |
fitness_short=-215.045638 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=6720 | radius=0.0209
[2025-11-02 18:16:28,667] INFO - Epoch 35 complete | lambda_short=0.486974 |
fitness_short=-416.898875 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=6912 | radius=0.0209
[2025-11-02 18:16:51,012] INFO - Epoch 36 complete | lambda_short=1.386435 |
fitness_short=-160.878281 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=7104 | radius=0.0209
[2025-11-02 18:17:13,793] INFO - Epoch 37 complete | lambda_short=-0.119313 |
fitness_short=-212.074588 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=7296 | radius=0.0209
[2025-11-02 18:17:37,115] INFO - Epoch 38 complete | lambda_short=0.176403 |
fitness_short=-207.823164 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=7488 | radius=0.0209
[2025-11-02 18:18:00,241] INFO - Stagnation detected; reseeding around best
candidate.
[2025-11-02 18:18:00,241] INFO - Epoch 39 complete | lambda_short=2.236577 |
fitness_short=-184.155163 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=7680 | radius=0.0177
[2025-11-02 18:18:23,936] INFO - Epoch 40 complete | lambda_short=1.894693 |
fitness_short=-219.077873 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=7872 | radius=0.0177
[2025-11-02 18:18:46,782] INFO - Epoch 41 complete | lambda_short=0.947936 |
fitness_short=-211.374034 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=8064 | radius=0.0177
[2025-11-02 18:19:09,208] INFO - Epoch 42 complete | lambda_short=2.500270 |
fitness_short=-354.467955 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=8256 | radius=0.0177
[2025-11-02 18:19:31,475] INFO - Epoch 43 complete | lambda_short=2.025432 |
fitness_short=-283.282775 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=8448 | radius=0.0177
[2025-11-02 18:19:53,789] INFO - Epoch 44 complete | lambda_short=4.499353 |
fitness_short=-159.124418 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=8640 | radius=0.0177
[2025-11-02 18:20:16,081] INFO - Stagnation detected; reseeding around best
```

```
candidate.
[2025-11-02 18:20:16,081] INFO - Epoch 45 complete | lambda_short=8.762129 |
fitness_short=-500.794445 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=8832 | radius=0.0151
[2025-11-02 18:20:38,133] INFO - Epoch 46 complete | lambda_short=1.041660 |
fitness_short=-290.826366 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=9024 | radius=0.0151
[2025-11-02 18:21:00,097] INFO - Epoch 47 complete | lambda_short=0.458928 |
fitness_short=-362.751784 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=9216 | radius=0.0151
[2025-11-02 18:21:21,726] INFO - Epoch 48 complete | lambda_short=0.982516 |
fitness_short=-268.095885 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=9408 | radius=0.0151
[2025-11-02 18:22:09,092] INFO - Epoch 49 complete | lambda_short=1.970850 |
fitness_short=-287.636149 | lambda_best=-3.239194 | fitness_best=-149.424272 |
evals short/long=180/12 | total evals=9600 | radius=0.0151
[2025-11-02 18:22:09,092] INFO - Optimization completed | epochs=50 | evals=9600
| best lambda=-3.239194 | wall=1127.9s
```

[11]:
```
metrics = controller.metrics
```

[12]:
```
results
```

[12]:
```
{'status': 'completed',
 'best': {'masses': [1.0657623599327009, 0.7, 0.0012],
  'lambda': -3.239193647667314,
  'fitness': -149.42427240406374,
  'm1': 1.0657623599327009,
  'm2': 0.7,
  'm3': 0.0012},
 'evals': 9600,
 'epochs': 50}
```

1.5 Evaluación comparativa y visualización

Contrastamos el exponente de Lyapunov del estado base (masas en la mitad de sus rangos) contra el obtenido por la solución optimizada. Por último, integramos la dinámica con las masas ganadoras para visualizar la trayectoria de los tres cuerpos y observar el comportamiento cerca del límite de estabilidad.

[ ]:
```
from two_body.core.cache import HierarchicalCache
from two_body.logic.fitness import FitnessEvaluator

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

original_masses = (1.3, 0.8, 1.3e-3)
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 = inf, lambda optimo = -3.239193647667314
```

[ ]:
```
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])]
```

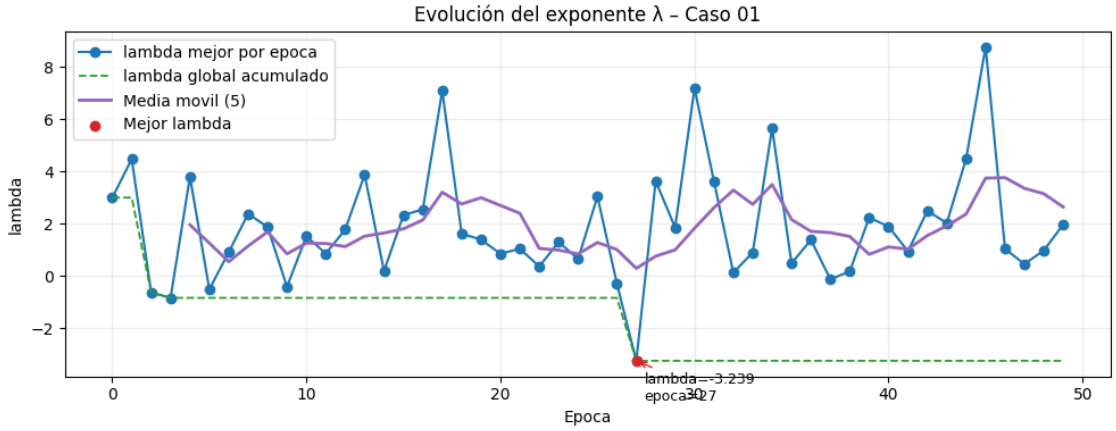
```
Trayectoria calculada con masas óptimas.
(20000, 3, 6)
[[-7.94563536e-02  1.97791068e-03  0.00000000e+00  1.57399352e-01
   7.87035237e+00  0.00000000e+00]
```

```
 [ 1.20484527e-01 -2.27845080e-03  0.00000000e+00 -2.55086583e-01
  -1.19951183e+01  0.00000000e+00]
 [ 2.85351464e-01 -4.27555995e-01  0.00000000e+00  9.00858668e+00
   7.21460124e+00  0.00000000e+00]]
```

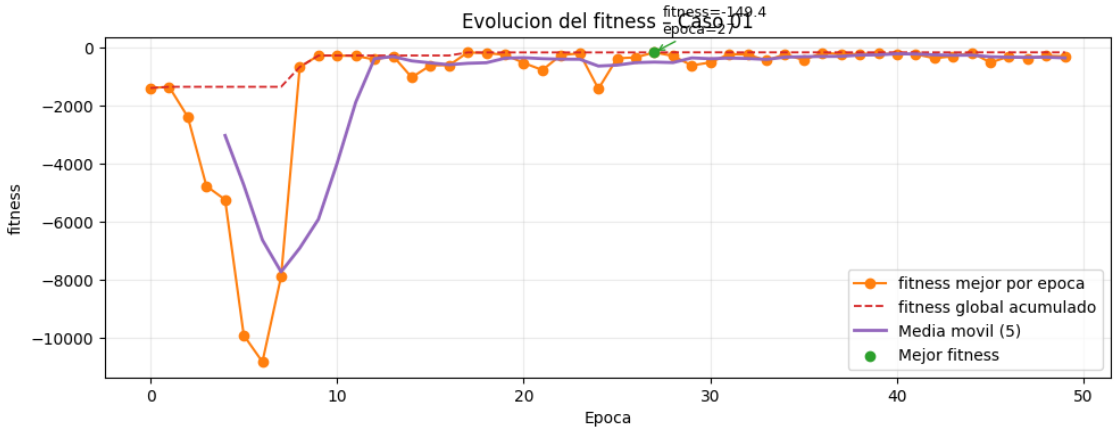
[15]:
```
from two_body.scripts.demo_tierra import (
    summarize_trajectory,
    compute_total_energy,
    estimate_orbital_period,
)

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

```
[2025-11-02 18:22:13,267] INFO - Resumen de simulacion
[2025-11-02 18:22:13,267] INFO -   pasos=20000 | dt=0.000200 anos | duracion
total=4.000 anos
[2025-11-02 18:22:13,267] INFO -   masas=(1.0657623599327009, 0.7, 0.0012)
(M_sun) | G=39.478418
[2025-11-02 18:22:13,286] INFO -   centro de masa: desplazamiento maximo =
3.282e-15 UA
[2025-11-02 18:22:13,301] INFO -   cuerpo 0 -> radio[min, max]=[0.0789, 0.1039]
UA | radio sigma=8.5570e-03 | velocidad media=6.8623 UA/ano
[2025-11-02 18:22:13,314] INFO -   cuerpo 1 -> radio[min, max]=[0.1204, 0.1579]
UA | radio sigma=1.2985e-02 | velocidad media=10.4479 UA/ano
[2025-11-02 18:22:13,314] INFO -   cuerpo 2 -> radio[min, max]=[0.4284, 0.5516]
UA | radio sigma=3.7966e-02 | velocidad media=12.1414 UA/ano
[2025-11-02 18:22:13,381] INFO -   energia total (media)=-6.395075e+01 |
variacion relativa=2.111e-15
[2025-11-02 18:22:13,389] INFO -   periodo orbital estimado para la Tierra ~=
0.083303 anos
[2025-11-02 18:22:13,389] INFO -   error relativo vs 1 ano ~= 9.167e-01
```

[16]:
```
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
)
```



[17]:
```
_ = viz_3d.plot_fitness_evolution(
    fitness_history=metrics.best_fitness_per_epoch,
    epoch_history=metrics.epoch_history,
    title="Evolucion del fitness - Caso 01",
    moving_average_window=5,
)
```

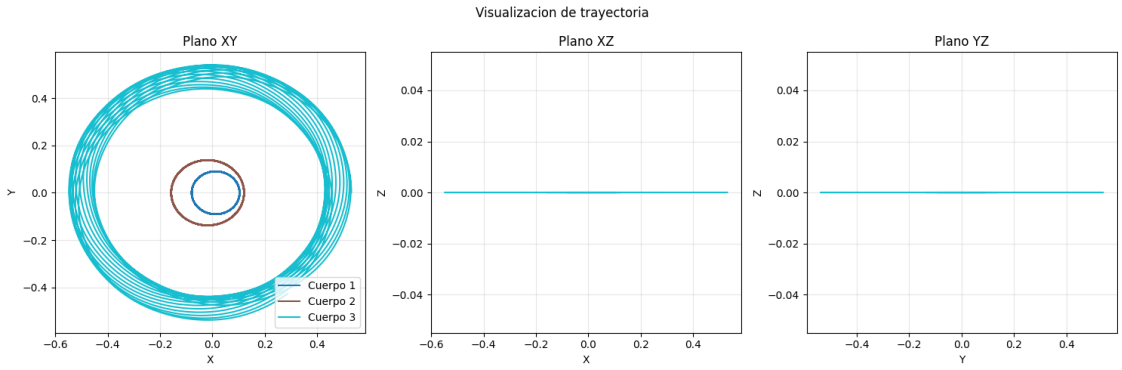


[18]:
```
sim_builder = ReboundSim(G=cfg.G, integrator=cfg.integrator)
best_masses = tuple(results["best"]["masses"])

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])]
```

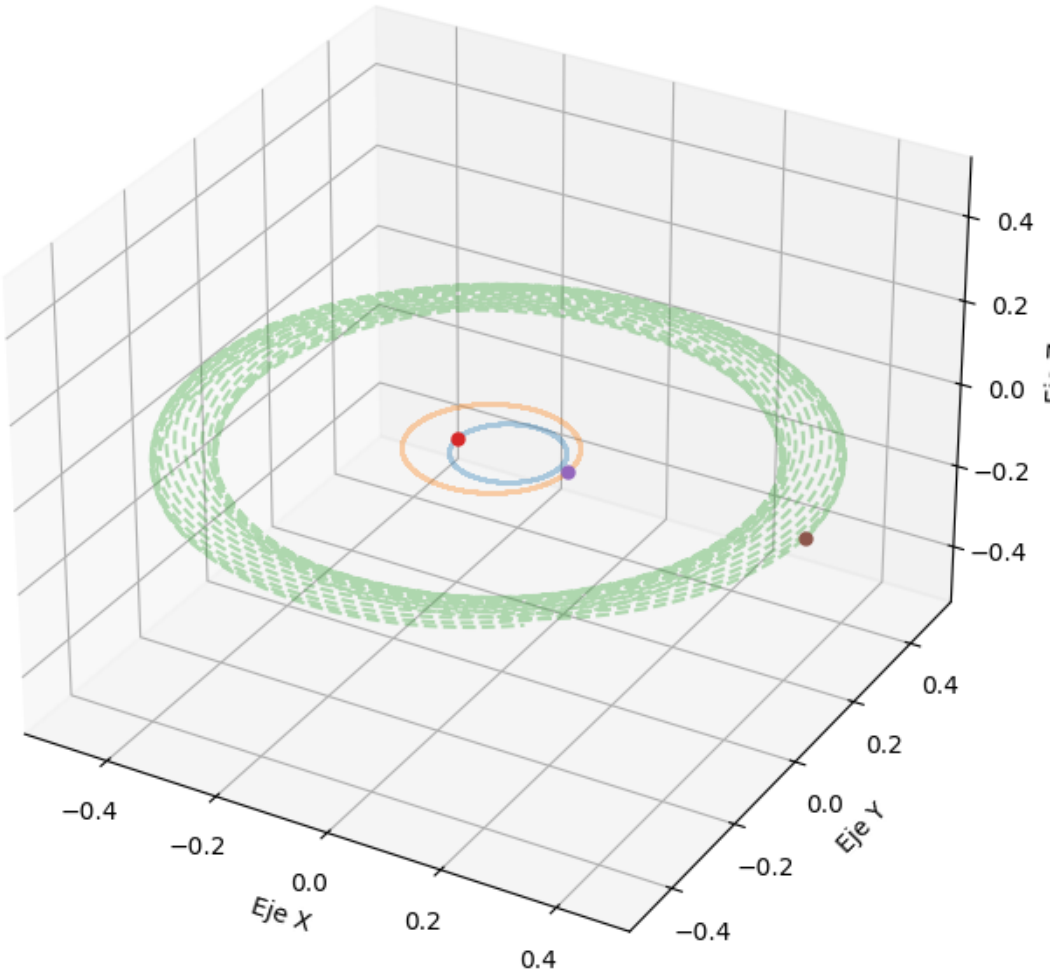
[ ]:
```
sim_orig = sim_builder.setup_simulation(center, r0, v0)
traj_original = sim_builder.integrate(sim_orig, t_end=cfg.t_end_long, dt=cfg.dt)

traj_opt = traj
```

[20]:
```
viz_planar = PlanarVisualizer(headless=cfg.headless)
_ = viz_planar.quick_view(xyz_tracks)  # usa una asignación para que Jupyter no␣
 ↪duplique la figura
```



[21]:
```
from IPython.display import HTML

import matplotlib as mpl
mpl.rcParams['animation.embed_limit'] = 50  # MB
```

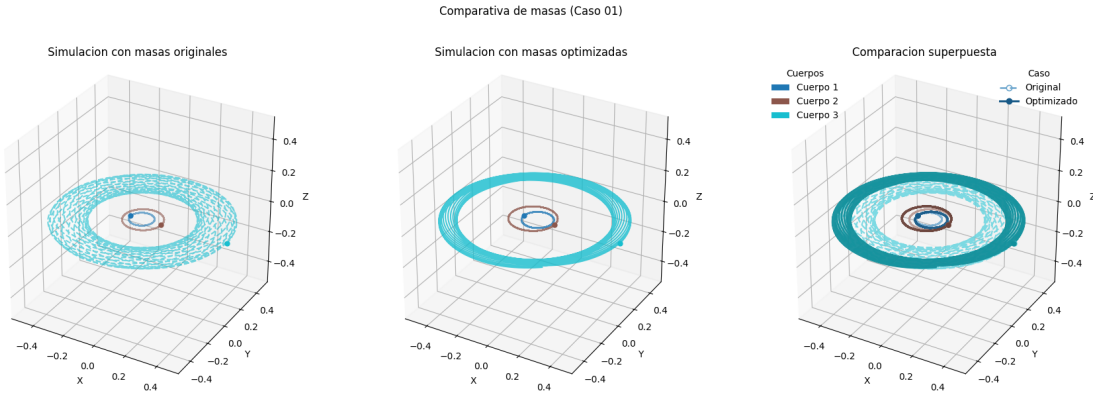
[22]:
```
viz_3d = Visualizer3D(headless=False)

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())
```

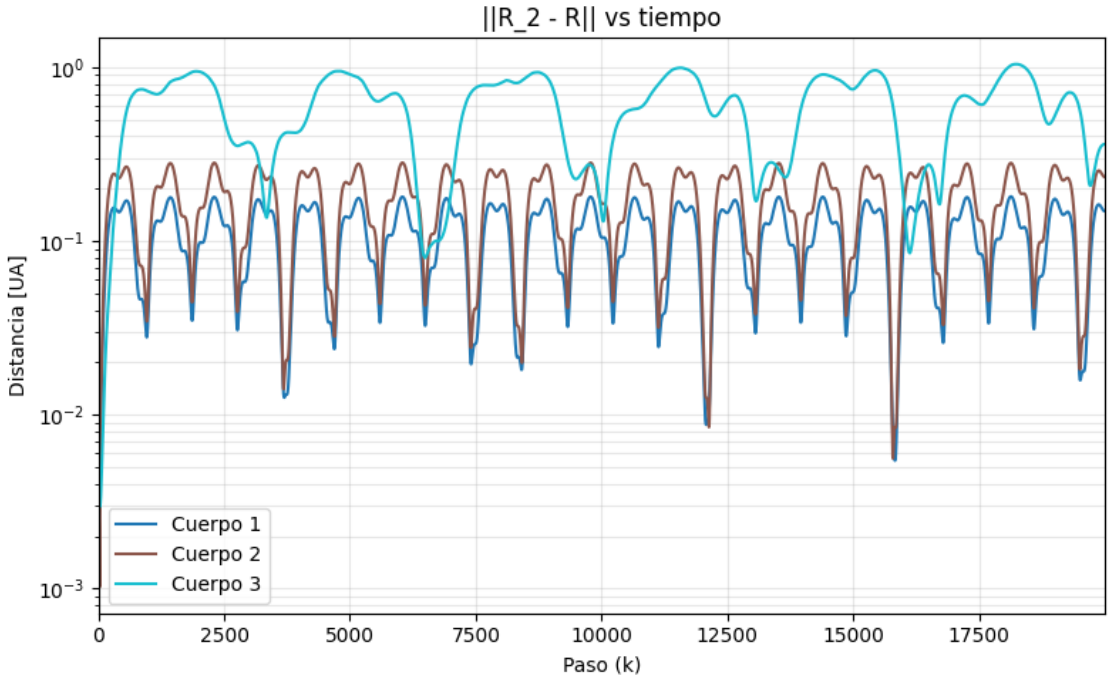


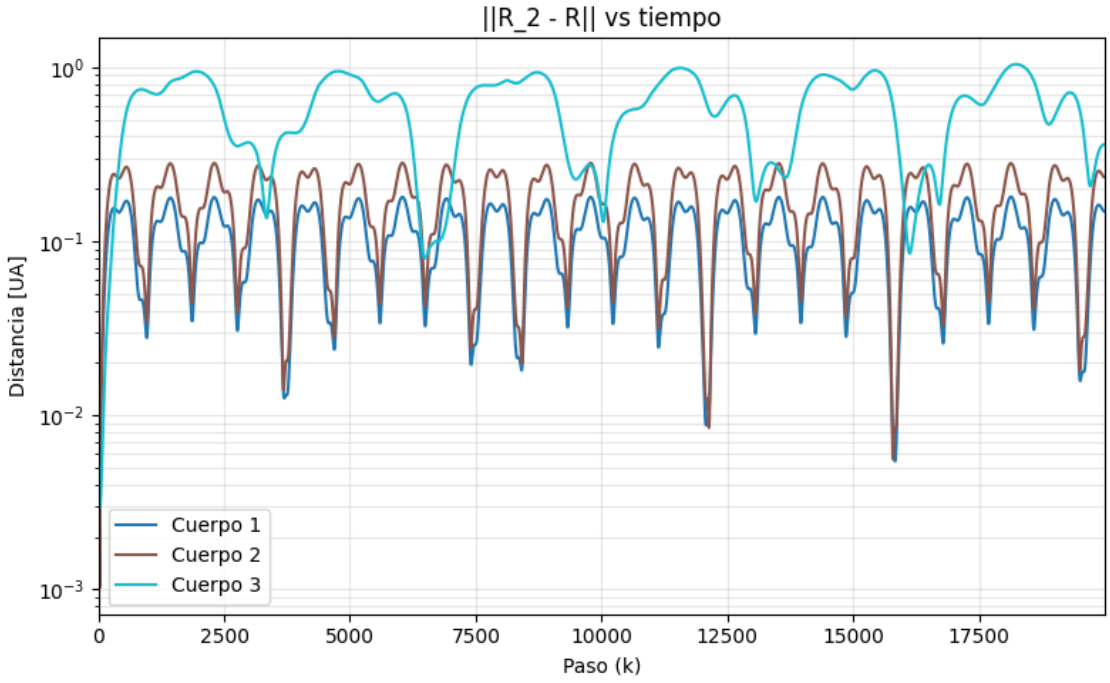
[ ]:
```
from matplotlib.animation import FFMpegWriter

writer = FFMpegWriter(fps=1000 // 50, bitrate=2400)   # fps = 1000/interval_ms
output_path = Path("artifacts/caso01")
output_path.mkdir(parents=True, exist_ok=True)

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

[ ]:
```
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
```







[25]:
```
anim_mass.save(output_path / "comparativa_masas.mp4", writer=writer)
```

[26]:
```
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'
```

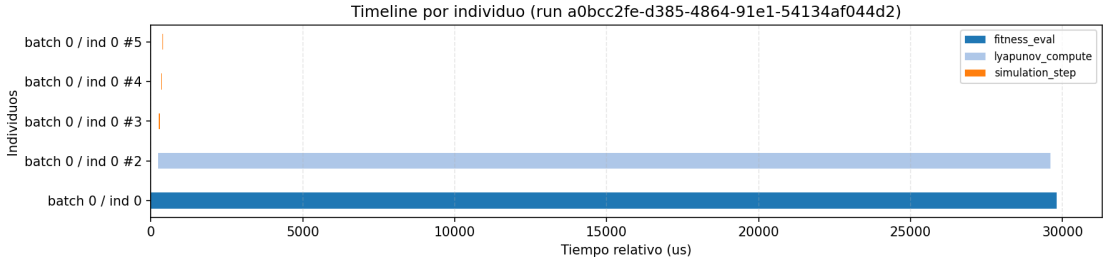
```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
Cell In[26], line 7
      4 display(f"Usando CSV: {csv_path}")
      6 rows = read_timings_csv(csv_path)
----> 7 df = pd.DataFrame(rows)
      8 display(df.head(10))
     10 # Estadísticas rápidas por sección

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\core\frame.py:851,
 ↪in DataFrame.__init__(self, data, index, columns, dtype, copy)
    849     if columns is not None:
    850         columns = ensure_index(columns)
--> 851     arrays, columns, index = nested_data_to_arrays(
    852         # error: Argument 3 to "nested_data_to_arrays" has incompatible
    853         # type "Optional[Collection[Any]]"; expected "Optional[Index]"
    854         data,
    855         columns,
    856         index,  # type: ignore[arg-type]
    857         dtype,
    858     )
    859     mgr = arrays_to_mgr(
    860         arrays,
    861         columns,
   (…)
    864         typ=manager,
    865     )
```

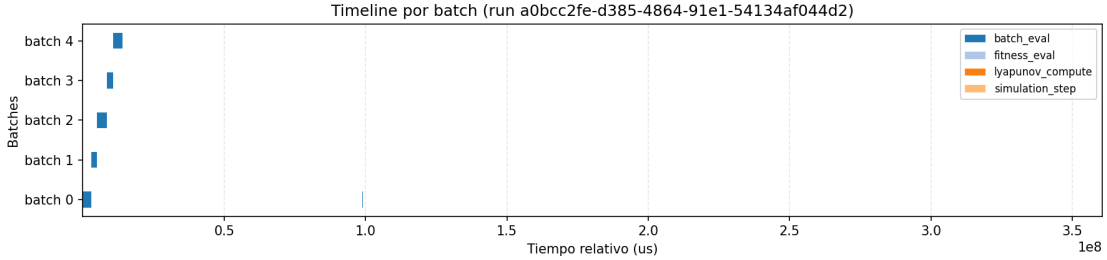
```
    866 else:

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\constr uction.
 ↪py:520, in nested_data_to_arrays(data, columns, index, dtype)
    517 if is_named_tuple(data[0]) and columns is None:
    518     columns = ensure_index(data[0]._fields)
--> 520 arrays, columns = to_arrays(data, columns, dtype=dtype)
    521 columns = ensure_index(columns)
    523 if index is None:

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\constr uction.
 ↪py:845, in to_arrays(data, columns, dtype)
    842     data = [tuple(x) for x in data]
    843     arr = _list_to_arrays(data)
--> 845 content, columns = _finalize_columns_and_data(arr, columns, dtype)
    846 return content, columns

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\constr uction.
 ↪py:945, in _finalize_columns_and_data(content, columns, dtype)
    942     raise ValueError(err) from err
    944 if len(contents) and contents[0].dtype == np.object_:
--> 945     contents = convert_object_array(contents, dtype=dtype)
    947 return contents, columns

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\constr uction.
 ↪py:1070, in convert_object_array(content, dtype, dtype_backend, coerce_float)
   1066             arr = maybe_cast_to_datetime(arr, dtype)
   1068     return arr
-> 1070 arrays = [convert(arr) for arr in content]
   1072 return arrays

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\pandas\core\internals\constr uction.
 ↪py:1030, in convert_object_array.<locals>.convert(arr)
   1028 def convert(arr):
   1029     if dtype != np.dtype("O"):
-> 1030         arr = lib.maybe_convert_objects(
   1031             arr,
   1032             try_float=coerce_float,
   1033             convert_to_nullable_dtype=dtype_backend != "numpy",
   1034         )
   1035         # Notes on cases that get here 2023-02-15
   1036         # 1) we DO get here when arr is all Timestamps and dtype=None
   1037         # 2) disabling this doesn't break the world, so this must be
   1038         #    getting caught at a higher level
```

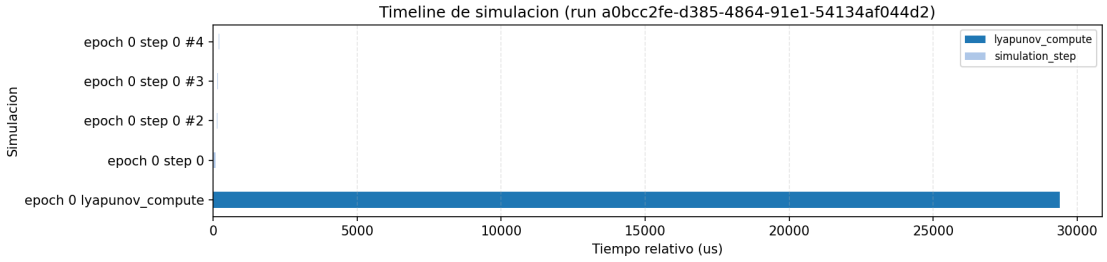
```
   1039         # 3) passing convert_non_numeric to maybe_convert_objects get␣
 ↪this right
   1040         # 4) convert_non_numeric?
   1042         if dtype is None:

File lib.pyx:2541, in pandas._libs.lib.maybe_convert_objects()

MemoryError: Unable to allocate 264. MiB for an array with shape (34582848,) an␣
 ↪data type uint64
```

[ ]:
```
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"

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")),
)
```

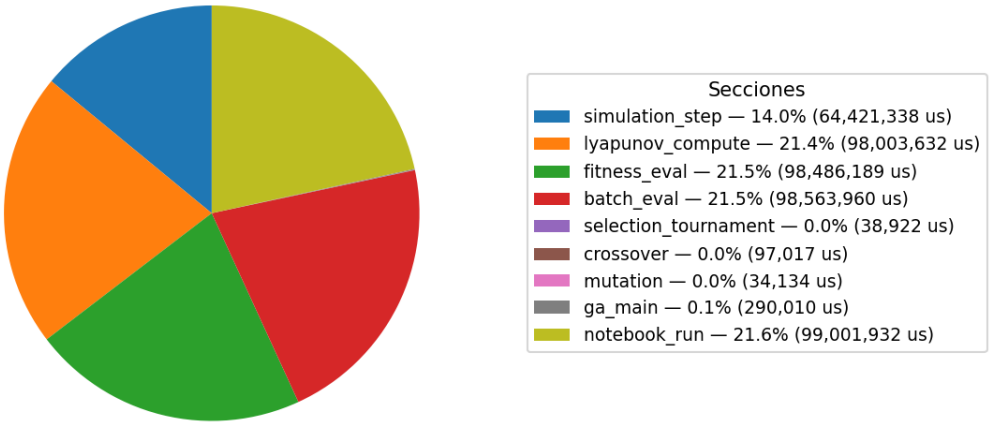
```
Ejecutando: c:\Users\emicr\anaconda3\envs\grav2body\python.exe
scripts/plot_timings.py --run-id a0bcc2fe-d385-4864-91e1-54134af044d2 --top-n 5
Graficas guardadas en C:\Users\emicr\Documents\CODIGOS_FUENTES\TrabajoTerminal\c
ollision_of_two_bodies\two_body\reports
```







cion por seccion (run a0bcc2fe-d385-4864-91e1-54134af044d2)



[ ]:
```
# from pathlib import Path
#
#
# output_path = Path("artifacts/animations/caso01_orbit.gif")
# output_path2 = Path("artifacts/animations/caso01_comparasion.gif")
# output_path.parent.mkdir(parents=True, exist_ok=True)
#
# anim.save(
#     output_path,
#     writer="pillow",
#     fps=20,
#     dpi=100,          # opcional
# )
#
# print(f"Animación 3D guardada en {output_path}")
#
```