Caso01

October 29, 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]:

```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 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  # si quieres guardar animaciones/figuras
```

[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
```

[4]:

```
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)
```

1.3 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.

[5]:

```
# ["Binaria con planeta (Hill) - órbita periódica"]
case = {
    # Simulación
    "t_end_short": 150.0,            # ~5 periodos de la binaria
    "t_end_long": 1500.0,            # verificación en horizonte 10× mayor
    "dt": 0.1,                       # paso fijo fino y estable
    "integrator": "ias15",
    "r0": (
        (-0.5, 0.0, 0.0),
        (0.5, 0.0, 0.0),
        (1.5, 0.0, 0.0),
    ),
    "v0": (
        (0.0, -0.07, 0.0),
        (0.0, 0.07, 0.0),
        (0.0, 0.045, 0.0),
    ),

    # Parámetros físicos
    "mass_bounds": (
        (0.99, 1.01),                # primaria casi fija
        (0.99, 1.01),                # secundaria ~1:2
        (5e-2, 7e-2),                # planeta tipo Neptuno-Saturno
    ),
    "G": 1.0,
    "x0": (-0.5, 0.0, 0.0, -0.7, 0.5, 0.0, 0.0, 0.7),
    "periodicity_weight": 500.0,       # activa el nuevo penalizador Δr+Δv

    # Algoritmo genético
    "pop_size": 96,
    "n_gen_step": 4,
    "crossover": 0.9,
    "mutation": 0.25,
    "selection": "tournament",
    "elitism": 2,
    "seed": 9871,

    # Optimización continua
    "max_epochs": 40,
    "top_k_long": 16,
    "stagnation_window": 5,
    "stagnation_tol": 2.5e-4,
    "local_radius": 0.02,
    "radius_decay": 0.70,
    "time_budget_s": 2400.0,
    "eval_budget": 8000,
    # Backend / cache
    "use_gpu": "false",
    "batch_size": 144,
    "cache_exact_max": 1000,
    "cache_approx_max": 3000,

    # 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)
```

```
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
```

```
{'lambda': 0.0029072693407235983, 'series': None, 'meta': {'steps': 1500, 'dt':
0.1, 'n_bodies': 3, 'masses': (1.0, 1.0, 0.060000000000000005), 'impl':
'rebound_megno'}}
```

[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)
```

```
((0.99, 1.01), (0.99, 1.01), (0.05, 0.07)) 40 8000
```

[10]:

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

```
[2025-10-29 00:08:09,497] INFO - Starting optimization | pop=96 | dims=3 |
time_budget=2400.0s | eval_budget=8000
[2025-10-29 00:08:36,069] INFO - Epoch 0 | new global best (short)
lambda=0.003127 | fitness=-78927.289821 | penalty=157.854573 | masses=(1.00186,
0.990705, 0.068213)
[2025-10-29 00:09:24,330] INFO - Epoch 0 complete | lambda_short=0.003127 |
fitness_short=-78927.289821 | lambda_best=0.003127 | fitness_best=-78927.289821
| evals short/long=96/16 | total evals=112 | radius=0.0200
[2025-10-29 00:09:51,592] INFO - Epoch 1 | new global best (short)
lambda=0.003838 | fitness=-78709.390106 | penalty=157.418773 | masses=(1.009017,
0.993309, 0.067103)
[2025-10-29 00:10:35,806] INFO - Epoch 1 complete | lambda_short=0.003838 |
fitness_short=-78709.390106 | lambda_best=0.003838 | fitness_best=-78709.390106
| evals short/long=96/16 | total evals=224 | radius=0.0200
[2025-10-29 00:11:45,848] INFO - Epoch 2 complete | lambda_short=0.006883 |
fitness_short=-79951.630213 | lambda_best=0.003838 | fitness_best=-78709.390106
| evals short/long=96/16 | total evals=336 | radius=0.0200
[2025-10-29 00:12:55,550] INFO - Epoch 3 complete | lambda_short=0.002968 |
fitness_short=-79748.734954 | lambda_best=0.003838 | fitness_best=-78709.390106
| evals short/long=96/16 | total evals=448 | radius=0.0200
[2025-10-29 00:14:10,902] INFO - Epoch 4 complete | lambda_short=0.005077 |
fitness_short=-80441.300730 | lambda_best=0.003838 | fitness_best=-78709.390106
| evals short/long=96/16 | total evals=560 | radius=0.0200
[2025-10-29 00:15:30,346] INFO - Epoch 5 complete | lambda_short=0.003607 |
fitness_short=-80309.101506 | lambda_best=0.003838 | fitness_best=-78709.390106
```

```
| evals short/long=96/16 | total evals=672 | radius=0.0200
[2025-10-29 00:16:49,469] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:16:49,469] INFO - Epoch 6 complete | lambda_short=0.006210 |
fitness_short=-80944.160262 | lambda_best=0.003838 | fitness_best=-78709.390106
| evals short/long=96/16 | total evals=784 | radius=0.0140
[2025-10-29 00:17:16,394] INFO - Epoch 7 | new global best (short)
lambda=0.006095 | fitness=-78456.353201 | penalty=156.912694 | masses=(1.01,
0.99, 0.07)
[2025-10-29 00:17:43,959] INFO - Epoch 7 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.006095 | fitness_best=-78456.353201
| evals short/long=96/16 | total evals=896 | radius=0.0140
[2025-10-29 00:18:07,928] INFO - Epoch 8 | new global best (short)
lambda=0.005143 | fitness=-78375.779930 | penalty=156.751550 | masses=(1.01,
0.99, 0.067883)
[2025-10-29 00:18:27,990] INFO - Epoch 8 complete | lambda_short=0.005143 |
fitness_short=-78375.779930 | lambda_best=0.005143 | fitness_best=-78375.779930
| evals short/long=96/16 | total evals=1008 | radius=0.0140
[2025-10-29 00:19:14,211] INFO - Epoch 9 complete | lambda_short=0.003227 |
fitness_short=-78425.677767 | lambda_best=0.005143 | fitness_best=-78375.779930
| evals short/long=96/16 | total evals=1120 | radius=0.0140
[2025-10-29 00:19:58,127] INFO - Epoch 10 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.005143 | fitness_best=-78375.779930
| evals short/long=96/16 | total evals=1232 | radius=0.0140
[2025-10-29 00:20:54,115] INFO - Epoch 11 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.005143 | fitness_best=-78375.779930
| evals short/long=96/16 | total evals=1344 | radius=0.0140
[2025-10-29 00:21:26,967] INFO - Epoch 12 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.005143 | fitness_best=-78375.779930
| evals short/long=96/16 | total evals=1456 | radius=0.0140
[2025-10-29 00:21:50,044] INFO - Epoch 13 | new global best (short)
lambda=0.004390 | fitness=-78240.387760 | penalty=156.480767 | masses=(1.01,
0.99, 0.067466)
[2025-10-29 00:22:07,842] INFO - Epoch 13 complete | lambda_short=0.004390 |
fitness_short=-78240.387760 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=1568 | radius=0.0140
[2025-10-29 00:22:54,420] INFO - Epoch 14 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=1680 | radius=0.0140
[2025-10-29 00:23:32,533] INFO - Epoch 15 complete | lambda_short=0.007810 |
fitness_short=-78431.529114 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=1792 | radius=0.0140
[2025-10-29 00:24:32,650] INFO - Epoch 16 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=1904 | radius=0.0140
[2025-10-29 00:25:13,439] INFO - Epoch 17 complete | lambda_short=0.003676 |
fitness_short=-78390.362194 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2016 | radius=0.0140
```

```
[2025-10-29 00:25:55,756] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:25:55,756] INFO - Epoch 18 complete | lambda_short=0.003629 |
fitness_short=-78376.808402 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2128 | radius=0.0098
[2025-10-29 00:26:42,950] INFO - Epoch 19 complete | lambda_short=0.003645 |
fitness_short=-78352.173905 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2240 | radius=0.0098
[2025-10-29 00:27:15,207] INFO - Epoch 20 complete | lambda_short=0.004387 |
fitness_short=-78290.839768 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2352 | radius=0.0098
[2025-10-29 00:27:56,643] INFO - Epoch 21 complete | lambda_short=0.004556 |
fitness_short=-78248.245677 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2464 | radius=0.0098
[2025-10-29 00:28:33,998] INFO - Epoch 22 complete | lambda_short=0.004263 |
fitness_short=-78275.241273 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2576 | radius=0.0098
[2025-10-29 00:29:17,341] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:29:17,341] INFO - Epoch 23 complete | lambda_short=0.005201 |
fitness_short=-78340.292159 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2688 | radius=0.0069
[2025-10-29 00:30:12,501] INFO - Epoch 24 complete | lambda_short=0.005819 |
fitness_short=-78244.830514 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2800 | radius=0.0069
[2025-10-29 00:30:54,380] INFO - Epoch 25 complete | lambda_short=0.003134 |
fitness_short=-78351.079370 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=2912 | radius=0.0069
[2025-10-29 00:31:37,641] INFO - Epoch 26 complete | lambda_short=0.003364 |
fitness_short=-78371.115140 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=3024 | radius=0.0069
[2025-10-29 00:32:22,526] INFO - Epoch 27 complete | lambda_short=0.004014 |
fitness_short=-78292.454643 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=3136 | radius=0.0069
[2025-10-29 00:33:09,648] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:33:09,648] INFO - Epoch 28 complete | lambda_short=0.003891 |
fitness_short=-78253.076689 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=3248 | radius=0.0048
[2025-10-29 00:34:02,712] INFO - Epoch 29 complete | lambda_short=0.003719 |
fitness_short=-78240.829655 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=3360 | radius=0.0048
[2025-10-29 00:34:56,461] INFO - Epoch 30 complete | lambda_short=0.005320 |
fitness_short=-78326.926665 | lambda_best=0.004390 | fitness_best=-78240.387760
| evals short/long=96/16 | total evals=3472 | radius=0.0048
[2025-10-29 00:35:22,689] INFO - Epoch 31 | new global best (short)
lambda=0.004841 | fitness=-78237.903856 | penalty=156.475798 | masses=(1.01,
0.99, 0.067444)
```

```
[2025-10-29 00:35:53,710] INFO - Epoch 31 complete | lambda_short=0.004841 |
fitness_short=-78237.903856 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=3584 | radius=0.0048
[2025-10-29 00:36:36,929] INFO - Epoch 32 complete | lambda_short=0.005818 |
fitness_short=-78346.530504 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=3696 | radius=0.0048
[2025-10-29 00:37:30,872] INFO - Epoch 33 complete | lambda_short=0.003706 |
fitness_short=-78301.016403 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=3808 | radius=0.0048
[2025-10-29 00:38:11,284] INFO - Epoch 34 complete | lambda_short=0.006925 |
fitness_short=-78246.236008 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=3920 | radius=0.0048
[2025-10-29 00:38:53,637] INFO - Epoch 35 complete | lambda_short=0.006095 |
fitness_short=-78456.353201 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=4032 | radius=0.0048
[2025-10-29 00:39:55,501] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:39:55,501] INFO - Epoch 36 complete | lambda_short=0.004014 |
fitness_short=-78261.219023 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=4144 | radius=0.0034
[2025-10-29 00:40:39,354] INFO - Epoch 37 complete | lambda_short=0.003972 |
fitness_short=-78241.856200 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=4256 | radius=0.0034
[2025-10-29 00:41:38,207] INFO - Epoch 38 complete | lambda_short=0.001475 |
fitness_short=-78313.179259 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=4368 | radius=0.0034
[2025-10-29 00:42:44,056] INFO - Epoch 39 complete | lambda_short=0.005657 |
fitness_short=-78275.271603 | lambda_best=0.004841 | fitness_best=-78237.903856
| evals short/long=96/16 | total evals=4480 | radius=0.0034
[2025-10-29 00:42:44,056] INFO - Optimization completed | epochs=40 | evals=4480
| best lambda=0.004841 | wall=2074.6s
```

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

```
[12]: results
```

```
[12]: {'status': 'completed',
       'best': {'masses': [1.01, 0.99, 0.06744385556501262],
        'lambda': 0.004840770420370122,
        'fitness': -78237.90385639518,
        'm1': 1.01,
        'm2': 0.99,
        'm3': 0.06744385556501262},
       'evals': 4480,
       'epochs': 40}
```

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.

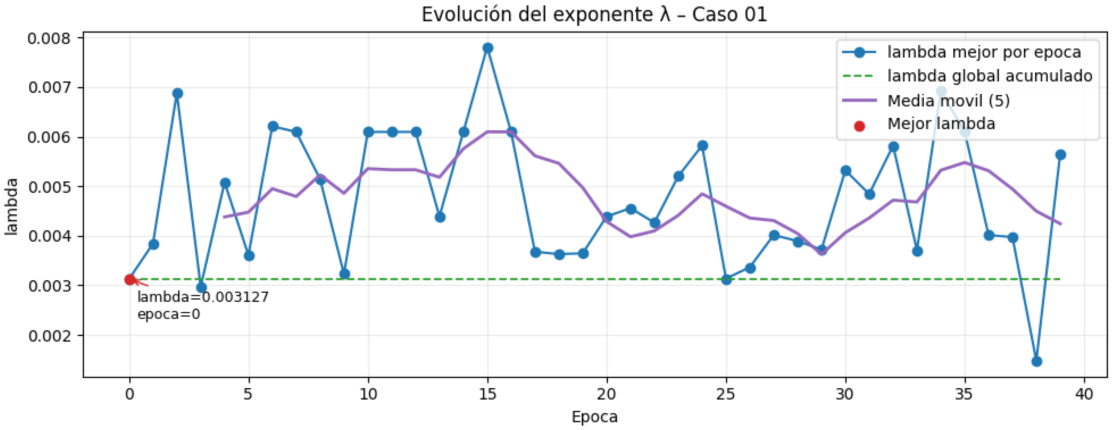
[13]:

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

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

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'}"
)
```

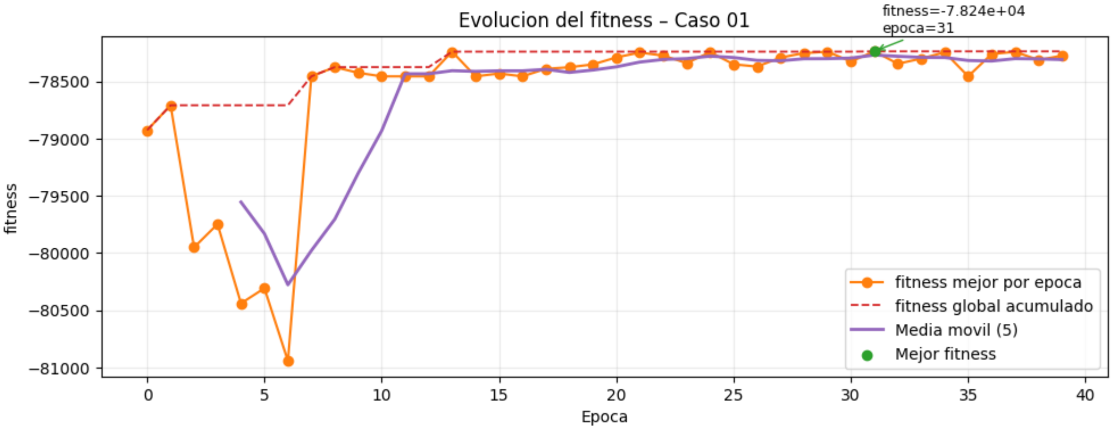
```
lambda inicial = -0.000019, lambda optimo = 0.004840770420370122
```

[14]:

```python
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
)
```



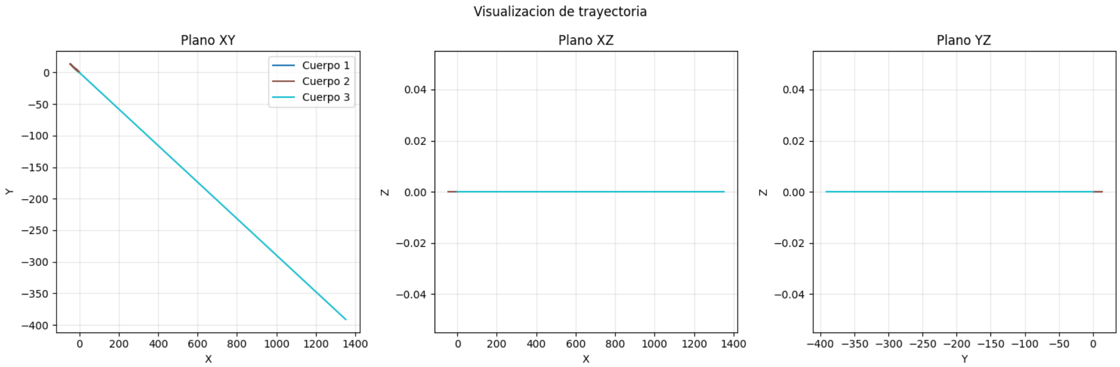
```
[15]: _ = viz_3d.plot_fitness_evolution(
          fitness_history=metrics.best_fitness_per_epoch,
          epoch_history=metrics.epoch_history,
          title="Evolucion del fitness - Caso 01",
          moving_average_window=5,
      )
```



```
[16]: 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))
```

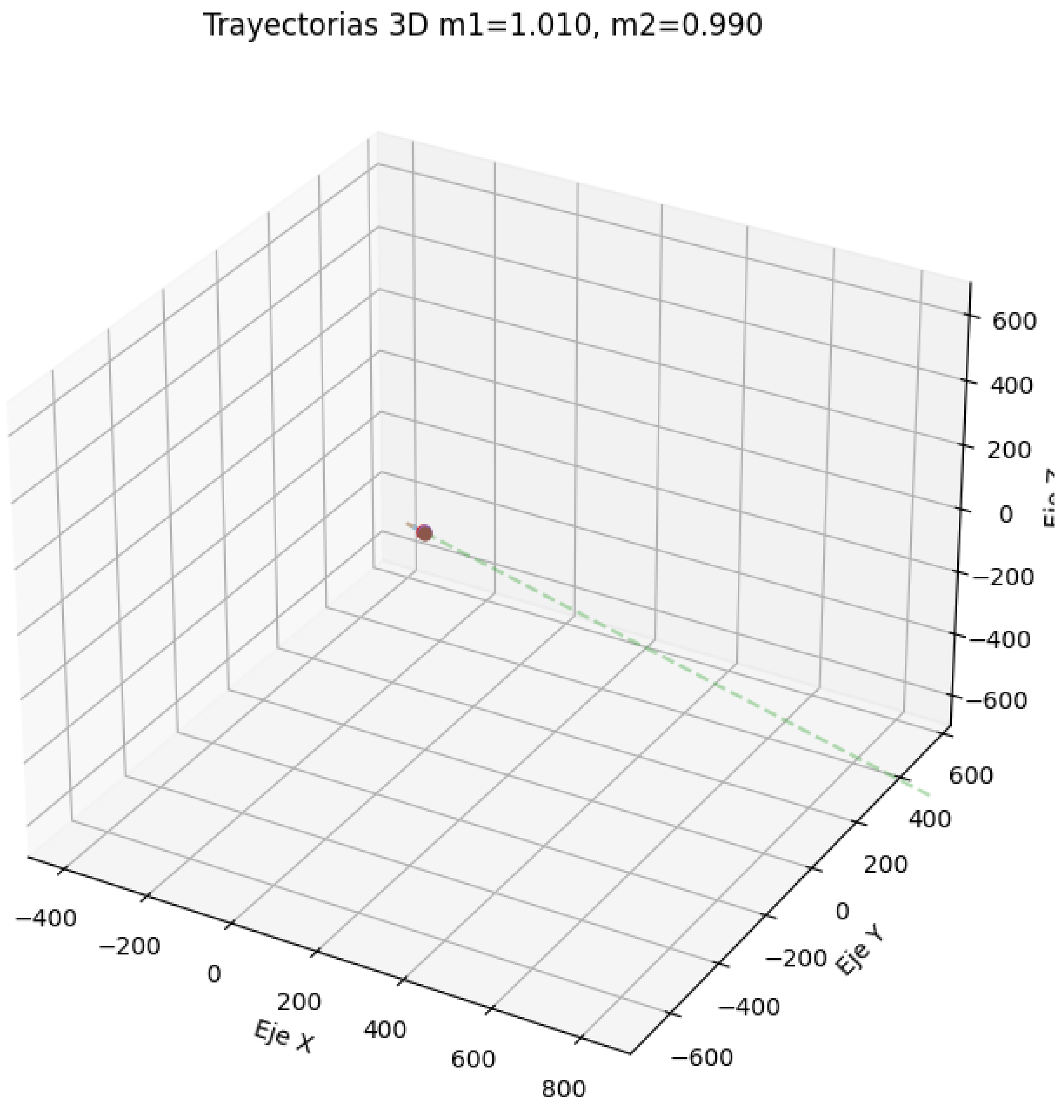
```
r0 = _slice_vectors(cfg.r0, len(best_masses))
v0 = _slice_vectors(cfg.v0, len(best_masses))

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])]
```

[17]:
```
viz_planar = PlanarVisualizer(headless=cfg.headless)
_ = viz_planar.quick_view(xyz_tracks)  # usa una asignación para que Jupyter no␣
 ↪duplique la figura
```



[18]:
```
from IPython.display import HTML

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

[19]:
```
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())
```



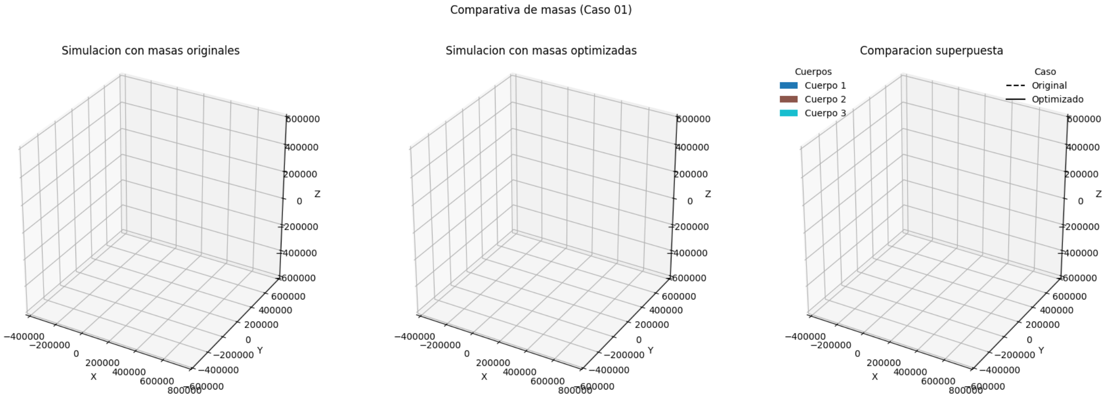
[20]:
```python
from matplotlib.animation import FFMpegWriter  # o PillowWriter para GIF

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

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

[28]:
```python
import importlib
import two_body.presentation.triDTry as triDTry

importlib.reload(triDTry)
```

```
Visualizer3D = triDTry.Visualizer  # o importa el alias que uses
viz_3d = Visualizer3D(headless=False)
```

[29]:

```
anim_mass = viz_3d.plot_mass_comparison(
    original_masses=center,
    optimized_masses=best_masses,
    body_labels=[f"Cuerpo {i+1}" for i in range(len(best_masses))],
    title="Comparativa de masas (Caso 01)",
)
#HTML(anim_mass.to_jshtml())
```



[30]:

```
anim_mass.save(output_path / "comparativa_masas.mp4", writer=writer)
```

[31]:

```
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'
```

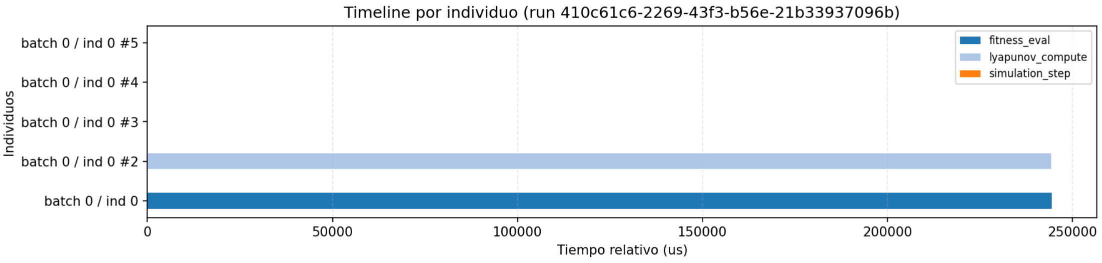
```
                                 run_id  epoch  batch_id  individual_id  \
0  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
1  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
2  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
3  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
4  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
5  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
6  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
7  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
8  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1
9  410c61c6-2269-43f3-b56e-21b33937096b     -1        -1             -1

           section        start_ns          end_ns  duration_us  \
0  simulation_step  23344070622800  23344070691700           68
1  simulation_step  23344070727400  23344070752600           25
2  simulation_step  23344070767900  23344070800400           32
3  simulation_step  23344070812700  23344070835600           22
4  simulation_step  23344070843800  23344070868000           24
5  simulation_step  23344070876300  23344070907100           30
6  simulation_step  23344070914800  23344071017100          102
7  simulation_step  23344071024300  23344072015300          991
8  simulation_step  23344072024300  23344072438300          414
9  simulation_step  23344072446700  23344072541000           94

                                               extra
0            {'step': 0, 'dt': 0.1, 't_target': 0.1}
1            {'step': 1, 'dt': 0.1, 't_target': 0.2}
2  {'step': 2, 'dt': 0.1, 't_target': 0.300000000…
3            {'step': 3, 'dt': 0.1, 't_target': 0.4}
4            {'step': 4, 'dt': 0.1, 't_target': 0.5}
5  {'step': 5, 'dt': 0.1, 't_target': 0.600000000…
6  {'step': 6, 'dt': 0.1, 't_target': 0.700000000…
7            {'step': 7, 'dt': 0.1, 't_target': 0.8}
8            {'step': 8, 'dt': 0.1, 't_target': 0.9}
9            {'step': 9, 'dt': 0.1, 't_target': 1.0}
```

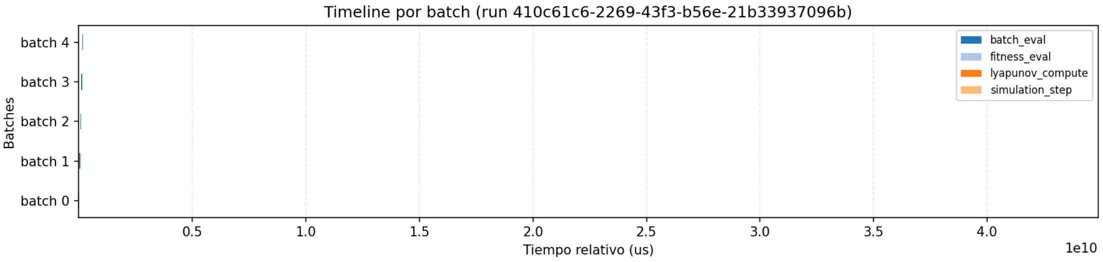
```
[31]:                          count          mean         sum
      section
      batch_eval                  81  2.563310e+07  2076281003
      fitness_eval              4481  4.633180e+05  2076127746
      lyapunov_compute          3833  5.414733e+05  2075466980
      notebook_run                 1  2.074689e+09  2074688739
      simulation_step       10396975  1.846217e+02  1919506905
      ga_main                     40  2.688555e+04     1075422
      crossover                  607  4.878023e+02      296096
      selection_tournament       607  1.949127e+02      118312
      mutation                   607  1.720148e+02      104413
```

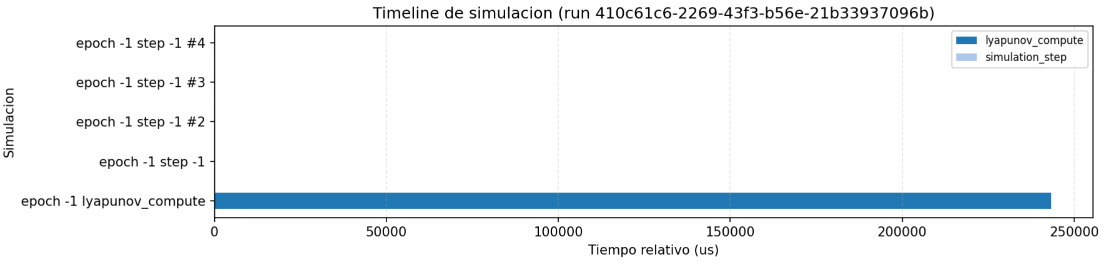
[32]:

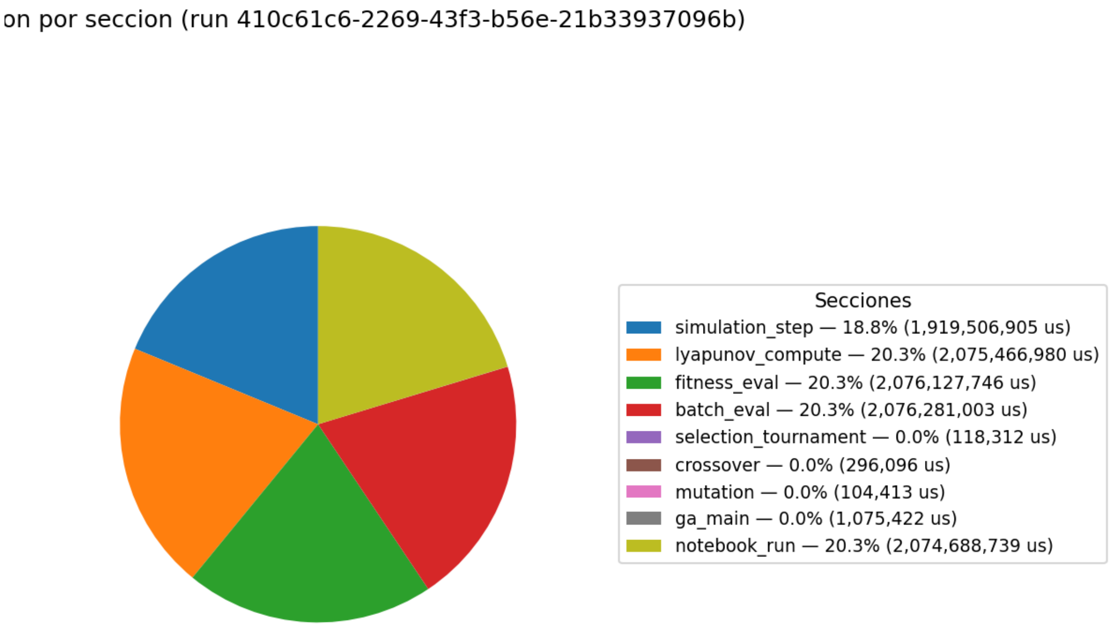
```
import os
import subprocess
from pathlib import Path
from IPython.display import Image, display

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 410c61c6-2269-43f3-b56e-21b33937096b --top-n 5
Graficas guardadas en C:\Users\emicr\Documents\CODIGOS_FUENTES\TrabajoTerminal\c
ollision_of_two_bodies\two_body\reports
```


Timeline por individuo (run 410c61c6-2269-43f3-b56e-21b33937096b)


Timeline por batch (run 410c61c6-2269-43f3-b56e-21b33937096b)


Timeline de simulacion (run 410c61c6-2269-43f3-b56e-21b33937096b)



on por seccion (run 410c61c6-2269-43f3-b56e-21b33937096b)

[ ]:

```
# 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}")
#
```