Caso04

October 29, 2025

1 Evaluación del caso “Trío no colineal con planeta errante”

Exploramos un sistema de tres cuerpos inicialmente no colineales: dos estrellas masivas y un planeta errante que parte con desplazamiento lateral. Buscamos combinaciones de masa que reduzcan la sensibilidad caótica y mantengan la órbita del planeta dentro de un lazo estable durante integraciones largas.

> **Interpretación** - Un reducido indica que el planeta soporta mejor las perturbaciones y permanece ligado al sistema. - La penalización de periodicidad ayuda a mantener la topología orbital tras miles de unidades de tiempo, evitando salidas balísticas.

1.1 Preparación del entorno

Localizamos la carpeta raíz `two_body`, la añadimos a `sys.path` y exponemos `PARENT` para reutilizar módulos comunes del repositorio. Garantizamos que los imports funcionen sin importar la ruta desde la que 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  # si quieres guardar animaciones/figuras
```

1.3 Instrumentación de rendimiento

Activamos las variables de entorno de la suite de timings y cargamos `time_block`, `latest_timing_csv`, etc. Esto nos permitirá auditar qué tan costosa es cada fase (integración short/long, refinamiento continuo, métricas).

[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
```

1.4 Logging amigable para notebooks

Definimos un `NotebookHandler` que acumula los mensajes del optimizador y los muestra en vivo dentro de la celda. Mantiene la salida legible mientras corren las generaciones y el refinamiento continuo.

[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)
```

1.5 Configuración del escenario "Trío no colineal"

El diccionario `case` establece: - Integraciones prolongadas (`t_end_long = 12000`, `dt = 0.06`) con `whfast`. - Condiciones iniciales no colineales: dos estrellas desplazadas y un planeta errante cerca del eje Y. - Rangos de masa relativamente amplios para permitir que el GA redistribuya masa entre los tres cuerpos. - Parámetros evolutivos moderados (`pop_size = 110`, `mutation = 0.08`, etc.) y penalización de periodicidad (`periodicity_weight = 0.1`) para favorecer órbitas reincidentes.

[5]:
```
case = {
    # Integración
    "t_end_short": 800.0,
    "t_end_long": 12000.0,
    "dt": 0.06,
    "integrator": "whfast",           # pasos uniformes  estable + rápido

    # Condiciones iniciales (tres cuerpos no colineales)
    "r0": (
        (-2.5, 0.0, 0.0),             # estrella 1
        (2.3, 0.2, 0.0),              # estrella 2
        (0.2, 1.8, 0.0),              # planeta errante
    ),
    "v0": (
        (0.0, -0.55, 0.0),
        (0.0, 0.45, 0.0),
        (-0.6, 0.0, 0.0),
    ),

    # Rango de masas amplio (caos y asimetría)
    "mass_bounds": (
        (0.3, 2.5),                   # estrella mayor
        (0.15, 1.5),                  # estrella secundaria
        (1e-4, 3e-1),                 # tercer cuerpo pequeño a intermedio
    ),
    "G": 1.0,
    "periodicity_weight": 0.02,       # valorar ligerísima cercanía orbital sin␣
 ↪perder caos

    # GA / búsqueda continua (mucha exploración inicial)
    "pop_size": 220,
    "n_gen_step": 1,
    "crossover": 0.9,
    "mutation": 0.12,
```

```
    "elitism": 2,
    "seed": 98765,

    # Control de ejecución (promueve explotación posterior)
    "max_epochs": 18,
    "top_k_long": 14,
    "stagnation_window": 5,
    "stagnation_tol": 1e-3,
    "local_radius": 0.08,
    "radius_decay": 0.82,
    "time_budget_s": 2400.0,
    "eval_budget": 90000,

    # Artefactos / salida
    "artifacts_dir": "artifacts/caso04_caotico",
    "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 Validación rápida del estimador de Lyapunov

Antes de lanzar el controlador, construimos una simulación con las masas promedio y ejecutamos `LyapunovEstimator.mLCE`. Esto sirve como smoke test para confirmar que el integrador produce trayectorias razonables y que el cálculo de responde al nuevo escenario.

[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': 0.0019689659220968004, 'series': None, 'meta': {'steps': 13334, 'dt':
0.06, 'n_bodies': 3, 'masses': (1.4, 0.825, 0.15005), 'impl': 'rebound_megno'}}
```

1.7 Recarga del evaluador de fitness

Si se modificó `two_body.logic.fitness`, recargamos el módulo con `importlib.reload` y reinstanciamos `setup_logger`. Así garantizamos que el notebook utilice la versión fresca del evaluador antes de iniciar la optimización.

[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)
```

1.8 Ejecución del controlador híbrido

Instanciamos `Config`, el `logger` y `ContinuousOptimizationController`, y envolvemos `controller.run()` dentro de `time_block("notebook_run")` para medir la ejecución total (GA + refinamiento continuo) sobre el sistema no colineal.

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

```
((0.3, 2.5), (0.15, 1.5), (0.0001, 0.3)) 18 90000
```

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

```
[2025-10-29 00:08:29,265] INFO - Starting optimization | pop=220 | dims=3 |
time_budget=2400.0s | eval_budget=90000
[2025-10-29 00:09:18,561] INFO - Epoch 0 | new global best (short)
lambda=0.007577 | fitness=-0.182329 | penalty=8.737577 | masses=(0.543542,
1.26368, 0.252752)
[2025-10-29 00:10:01,990] INFO - Epoch 0 complete | lambda_short=0.007577 |
fitness_short=-0.182329 | lambda_best=0.007577 | fitness_best=-0.182329 | evals
short/long=220/14 | total evals=234 | radius=0.0800
```

```
[2025-10-29 00:11:29,399] INFO - Epoch 1 complete | lambda_short=0.029546 |
fitness_short=-0.182521 | lambda_best=0.007577 | fitness_best=-0.182329 | evals
short/long=220/14 | total evals=468 | radius=0.0800
[2025-10-29 00:12:12,440] INFO - Epoch 2 | new global best (short)
lambda=0.013760 | fitness=-0.179583 | penalty=8.291146 | masses=(1.649407,
1.175987, 0.145679)
[2025-10-29 00:13:00,213] INFO - Epoch 2 complete | lambda_short=0.013760 |
fitness_short=-0.179583 | lambda_best=0.013760 | fitness_best=-0.179583 | evals
short/long=220/14 | total evals=702 | radius=0.0800
[2025-10-29 00:13:46,943] INFO - Epoch 3 | new global best (short)
lambda=0.009473 | fitness=-0.097195 | penalty=4.386075 | masses=(1.630931,
1.249834, 0.16428)
[2025-10-29 00:14:35,845] INFO - Epoch 3 complete | lambda_short=0.009473 |
fitness_short=-0.097195 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=936 | radius=0.0800
[2025-10-29 00:16:15,813] INFO - Epoch 4 complete | lambda_short=0.017168 |
fitness_short=-0.210365 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=1170 | radius=0.0800
[2025-10-29 00:17:54,399] INFO - Epoch 5 complete | lambda_short=0.004007 |
fitness_short=-0.101391 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=1404 | radius=0.0800
[2025-10-29 00:19:22,332] INFO - Epoch 6 complete | lambda_short=0.011205 |
fitness_short=-0.146731 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=1638 | radius=0.0800
[2025-10-29 00:20:52,676] INFO - Epoch 7 complete | lambda_short=0.002490 |
fitness_short=-0.259667 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=1872 | radius=0.0800
[2025-10-29 00:22:23,186] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:22:23,186] INFO - Epoch 8 complete | lambda_short=0.006151 |
fitness_short=-0.325395 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=2106 | radius=0.0656
[2025-10-29 00:23:59,001] INFO - Epoch 9 complete | lambda_short=0.013455 |
fitness_short=-0.219125 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=2340 | radius=0.0656
[2025-10-29 00:25:29,773] INFO - Epoch 10 complete | lambda_short=0.014759 |
fitness_short=-0.186760 | lambda_best=0.009473 | fitness_best=-0.097195 | evals
short/long=220/14 | total evals=2574 | radius=0.0656
[2025-10-29 00:26:16,877] INFO - Epoch 11 | new global best (short)
lambda=0.013013 | fitness=-0.066752 | penalty=2.686954 | masses=(1.70828,
1.28152, 0.109214)
[2025-10-29 00:27:00,161] INFO - Epoch 11 complete | lambda_short=0.013013 |
fitness_short=-0.066752 | lambda_best=0.013013 | fitness_best=-0.066752 | evals
short/long=220/14 | total evals=2808 | radius=0.0656
[2025-10-29 00:28:32,826] INFO - Epoch 12 complete | lambda_short=0.008282 |
fitness_short=-0.100831 | lambda_best=0.013013 | fitness_best=-0.066752 | evals
short/long=220/14 | total evals=3042 | radius=0.0656
[2025-10-29 00:30:13,250] INFO - Epoch 13 complete | lambda_short=0.013322 |
```

```
fitness_short=-0.125193 | lambda_best=0.013013 | fitness_best=-0.066752 | evals
short/long=220/14 | total evals=3276 | radius=0.0656
[2025-10-29 00:31:54,301] INFO - Epoch 14 complete | lambda_short=0.012888 |
fitness_short=-0.164254 | lambda_best=0.013013 | fitness_best=-0.066752 | evals
short/long=220/14 | total evals=3510 | radius=0.0656
[2025-10-29 00:33:35,853] INFO - Epoch 15 complete | lambda_short=0.019222 |
fitness_short=-0.138172 | lambda_best=0.013013 | fitness_best=-0.066752 | evals
short/long=220/14 | total evals=3744 | radius=0.0656
[2025-10-29 00:35:15,578] INFO - Stagnation detected; reseeding around best
candidate.
[2025-10-29 00:35:15,578] INFO - Epoch 16 complete | lambda_short=0.013609 |
fitness_short=-0.144296 | lambda_best=0.013013 | fitness_best=-0.066752 | evals
short/long=220/14 | total evals=3978 | radius=0.0538
[2025-10-29 00:36:05,439] INFO - Epoch 17 | new global best (short)
lambda=-85.509502 | fitness=48.738226 | penalty=1838.563802 | masses=(1.75175,
1.344633, 0.099315)
[2025-10-29 00:36:54,007] INFO - Epoch 17 complete | lambda_short=-85.509502 |
fitness_short=48.738226 | lambda_best=-85.509502 | fitness_best=48.738226 |
evals short/long=220/14 | total evals=4212 | radius=0.0538
[2025-10-29 00:36:54,007] INFO - Optimization completed | epochs=18 | evals=4212
| best lambda=-85.509502 | wall=1704.7s
```

1.9 Métricas base y comparación con la solución óptima

Calculamos el fitness del punto central (`center`) y lo comparamos con `results["best"]`. Guardamos `baseline_details` para estudiar la dinámica inicial y recuperamos `metrics` para posteriores visualizaciones de , fitness y estadísticas del experimento.

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

[12]:
```
results
```

[12]:
```
{'status': 'completed',
 'best': {'masses': [1.7517503842291131,
   1.344633017546932,
   0.0993148822285071],
  'lambda': -85.5095023785923,
  'fitness': 48.7382263393786,
  'm1': 1.7517503842291131,
  'm2': 1.344633017546932,
  'm3': 0.0993148822285071},
 'evals': 4212,
 'epochs': 18}
```

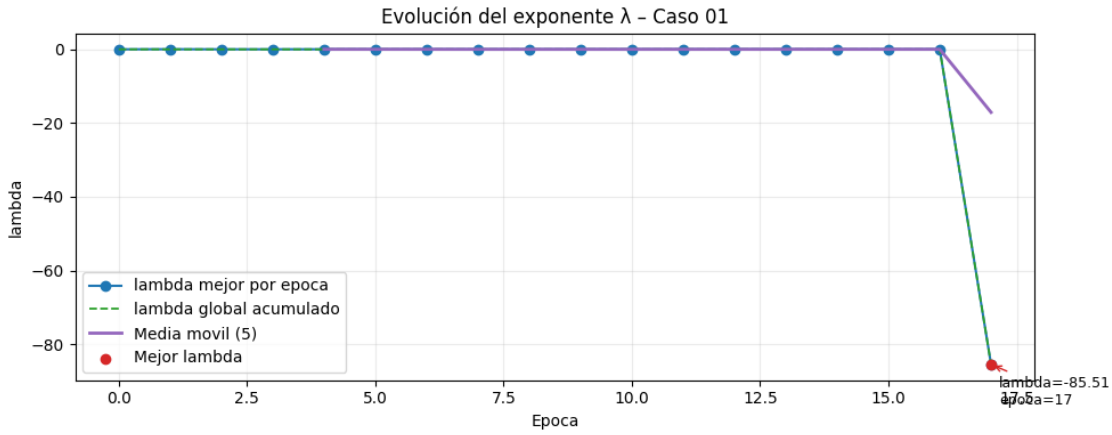
1.10 Visualización de convergencia y órbitas

Generamos: - `plot_lambda_evolution` y `plot_fitness_evolution` para revisar la trayectoria del GA. - Integraciones 2D (`quick_view`) y 3D (`animate_3d`) con las masas óptimas, verificando si el planeta errante permanece confinado pese a la geometría no colineal.

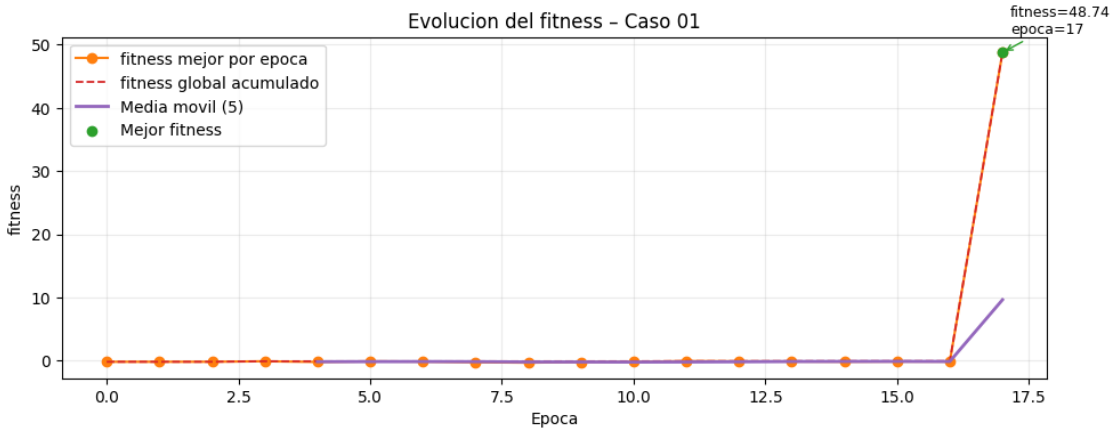
[13]:
```python
from two_body.core.cache import HierarchicalCache
from two_body.logic.fitness import FitnessEvaluator

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 = -0.000117, lambda optimo = -85.5095023785923
```

[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
)
```



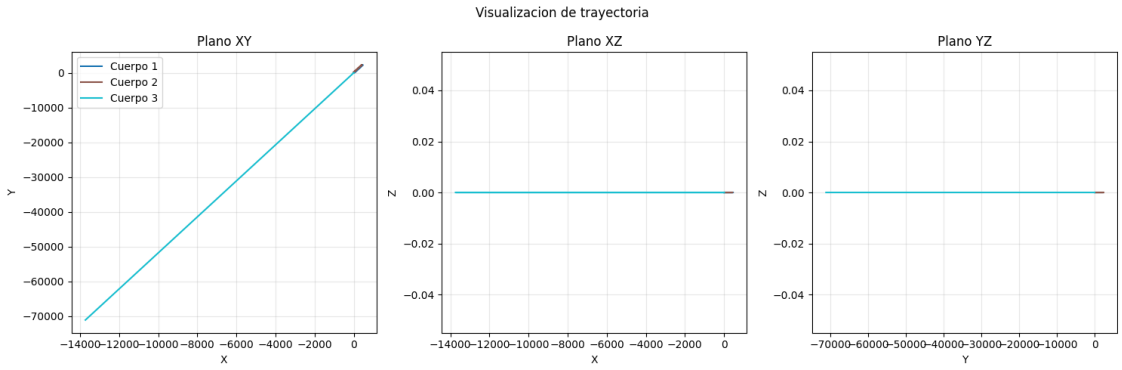
[15]:
```
_ = viz_3d.plot_fitness_evolution(
    fitness_history=metrics.best_fitness_per_epoch,
    epoch_history=metrics.epoch_history,
    title="Evolucion del fitness - Caso 01",
    moving_average_window=5,
)
```



[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))
```

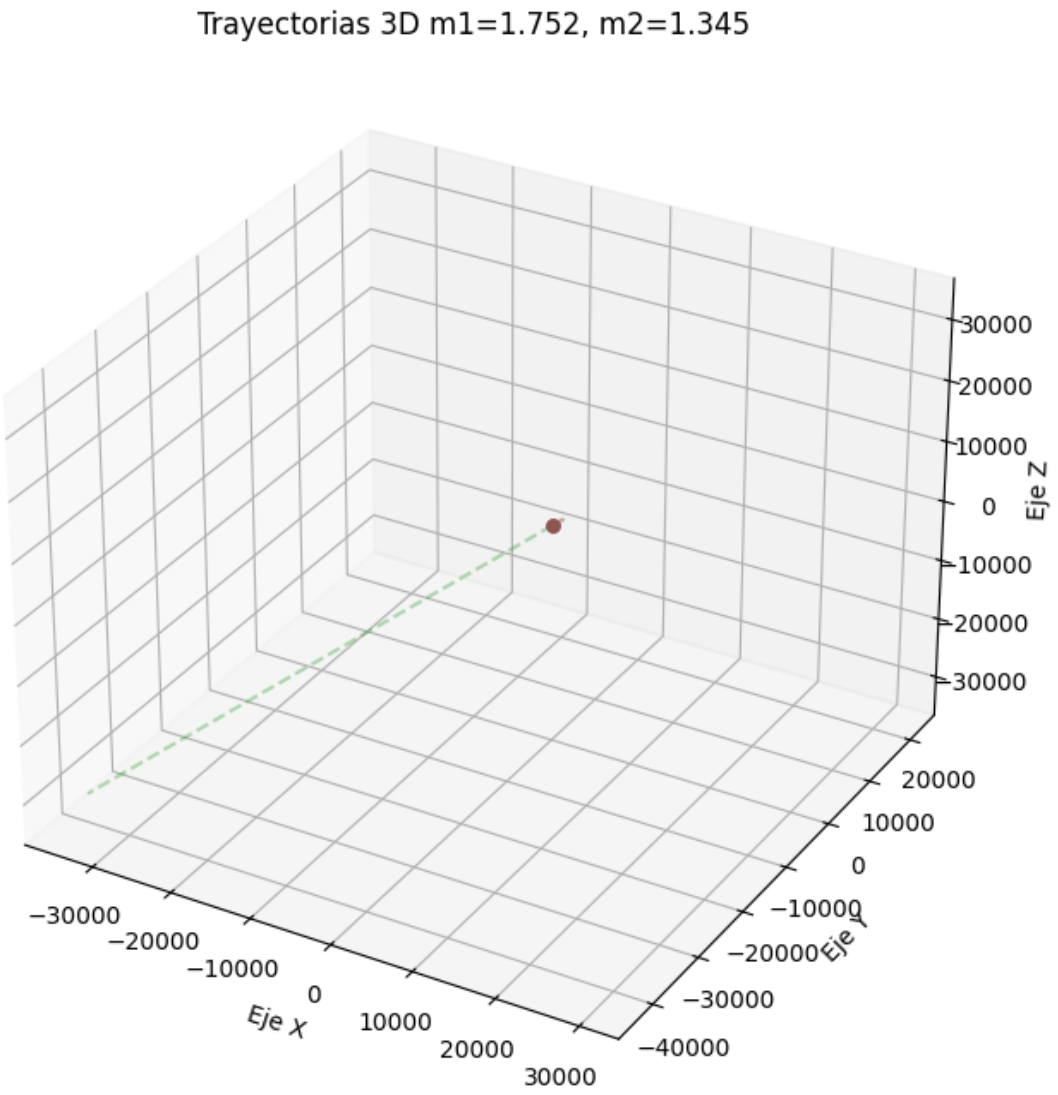
```
r0 = _slice_vectors(cfg.r0, len(best_masses))
v0 = _slice_vectors(cfg.v0, len(best_masses))

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())
```



1.11 Exportación de animaciones

Preparamos un `FFMpegWriter`, creamos `artifacts/caso04` y guardamos los MP4 tanto de la órbita como de la comparación de masas. Ajusta `fps`, `bitrate` o `dpi` si necesitas equilibrar rapidez y calidad de render.

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

writer = FFMpegWriter(fps=1000 // 50, bitrate=2400)   # fps = 1000/interval_ms
output_path = Path("artifacts/caso04")                # ajusta a tu gusto
output_path.mkdir(parents=True, exist_ok=True)

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

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[20], line 7
      4 output_path = Path("artifacts/caso04")                # ajusta a tu gusto
      5 output_path.mkdir(parents=True, exist_ok=True)
----> 7 anim.save(output_path / "trayectoria_optima.mp4", writer=writer)

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\animation.
 ↪py:1122, in Animation.save(self, filename, writer, fps, dpi, codec, bitrate,␣
 ↪extra_args, metadata, extra_anim, savefig_kwargs, progress_callback)
   1119 for data in zip(*[a.new_saved_frame_seq() for a in all_anim]):
   1120     for anim, d in zip(all_anim, data):
   1121         # TODO: See if turning off blit is really necessary
-> 1122         anim._draw_next_frame(d, blit=False)
   1123         if progress_callback is not None:
   1124             progress_callback(frame_number, total_frames)

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\animation.
 ↪py:1158, in Animation._draw_next_frame(self, framedata, blit)
   1156 self._pre_draw(framedata, blit)
   1157 self._draw_frame(framedata)
-> 1158 self._post_draw(framedata, blit)

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\animation.
 ↪py:1183, in Animation._post_draw(self, framedata, blit)
   1181     self._blit_draw(self._drawn_artists)
   1182 else:
-> 1183     self._fig.canvas.draw_idle()

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\backend_ba
 ↪py:1891, in FigureCanvasBase.draw_idle(self, *args, **kwargs)
   1889 if not self._is_idle_drawing:
   1890     with self._idle_draw_cntx():
-> 1891         self.draw(*args, **kwargs)

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\backends\b
 ↪py:382, in FigureCanvasAgg.draw(self)
    379 # Acquire a lock on the shared font cache.
    380 with (self.toolbar._wait_cursor_for_draw_cm() if self.toolbar
    381       else nullcontext()):
--> 382     self.figure.draw(self.renderer)
    383     # A GUI class may be need to update a window using this draw, so
    384     # don't forget to call the superclass.
```

```
    385     super().draw()

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\artis .
 ↪py:94, in _finalize_rasterization.<locals>.draw_wrapper(artist, renderer,␣
 ↪*args, **kwargs)
     92 @wraps(draw)
     93 def draw_wrapper(artist, renderer, *args, **kwargs):
---> 94     result = draw(artist, renderer, *args, **kwargs)
     95     if renderer._rasterizing:
     96         renderer.stop_rasterizing()

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\artis .
 ↪py:71, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     68     if artist.get_agg_filter() is not None:
     69         renderer.start_filter()
---> 71     return draw(artist, renderer)
     72 finally:
     73     if artist.get_agg_filter() is not None:

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\figur .
 ↪py:3257, in Figure.draw(self, renderer)
   3254             # ValueError can occur when resizing a window.
   3256     self.patch.draw(renderer)
-> 3257     mimage._draw_list_compositing_images(
   3258         renderer, self, artists, self.suppressComposite)
   3260     renderer.close_group('figure')
   3261 finally:

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\image
 ↪py:134, in _draw_list_compositing_images(renderer, parent, artists,␣
 ↪suppress_composite)
    132 if not_composite or not has_images:
    133     for a in artists:
--> 134         a.draw(renderer)
    135 else:
    136     # Composite any adjacent images together
    137     image_group = []

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\artis .
 ↪py:71, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     68     if artist.get_agg_filter() is not None:
     69         renderer.start_filter()
---> 71     return draw(artist, renderer)
     72 finally:
     73     if artist.get_agg_filter() is not None:
```

```
File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\mpl_toolkits\mplot3d\axes3d.
 ↪py:467, in Axes3D.draw(self, renderer)
    465     # Then axes, labels, text, and ticks
    466     for axis in self._axis_map.values():
--> 467         axis.draw(renderer)
    469 # Then rest
    470 super().draw(renderer)

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\artist.
 ↪py:71, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     68     if artist.get_agg_filter() is not None:
     69         renderer.start_filter()
---> 71     return draw(artist, renderer)
     72 finally:
     73     if artist.get_agg_filter() is not None:

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\mpl_toolkits\mplot3d\axis3d.
 ↪py:611, in Axis.draw(self, renderer)
    608 self.line.draw(renderer)
    610 # Draw ticks
--> 611 self._draw_ticks(renderer, edgep1, centers, deltas, highs,
    612                  deltas_per_point, pos)
    614 # Draw Offset text
    615 self._draw_offset_text(renderer, edgep1, edgep2, labeldeltas,
    616                        centers, highs, pep, dx, dy)

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\mpl_toolkits\mplot3d\axis3d.
 ↪py:478, in Axis._draw_ticks(self, renderer, edgep1, centers, deltas, highs,␣
 ↪deltas_per_point, pos)
    476 _tick_update_position(tick, (x1, x2), (y1, y2), (lx, ly))
    477 tick.tick1line.set_linewidth(tick_lw[tick._major])
--> 478 tick.draw(renderer)

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\artist.
 ↪py:71, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     68     if artist.get_agg_filter() is not None:
     69         renderer.start_filter()
---> 71     return draw(artist, renderer)
     72 finally:
     73     if artist.get_agg_filter() is not None:

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\axis.
 ↪py:276, in Tick.draw(self, renderer)
    273 renderer.open_group(self.__name__, gid=self.get_gid())
    274 for artist in [self.gridline, self.tick1line, self.tick2line,
    275                self.label1, self.label2]:
```

```
--> 276     artist.draw(renderer)
    277 renderer.close_group(self.__name__)
    278 self.stale = False

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\artis .
 ↪py:71, in allow_rasterization.<locals>.draw_wrapper(artist, renderer)
     68     if artist.get_agg_filter() is not None:
     69         renderer.start_filter()
---> 71     return draw(artist, renderer)
     72 finally:
     73     if artist.get_agg_filter() is not None:

File c:\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\text.
 ↪py:808, in Text.draw(self, renderer)
    804             textrenderer.draw_tex(gc, x, y, clean_line,
    805                                   self._fontproperties, angle,
    806                                   mtext=mtext)
    807         else:
--> 808             textrenderer.draw_text(gc, x, y, clean_line,
    809                                    self._fontproperties, angle,
    810                                    ismath=ismath, mtext=mtext)
    812 gc.restore()
    813 renderer.close_group('text')

File c:
 ↪\Users\emicr\anaconda3\envs\grav2body\Lib\site-packages\matplotlib\backends\b ckend_agg.
 ↪py:192, in RendererAgg.draw_text(self, gc, x, y, s, prop, angle, ismath, mtex )
    189 font = self._prepare_font(prop)
    190 # We pass '0' for angle here, since it will be rotated (in raster
    191 # space) in the following call to draw_text_image).
--> 192 font.set_text(s, 0, flags=get_hinting_flag())
    193 font.draw_glyphs_to_bitmap(
    194     antialiased=gc.get_antialiased())
    195 d = font.get_descent() / 64.0

KeyboardInterrupt:
```

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

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

1.12 Reporte de tiempos

Leemos el CSV más reciente de la instrumentación, mostramos una muestra y agrupamos por sección para identificar los cuellos de botella (por ejemplo, integración larga con tres cuerpos no colineales).

[ ]:
```
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	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
1	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
2	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
3	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
4	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
5	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
6	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
7	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
8	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	
9	95678a96-1f1b-4e50-b1ef-7945fed17930	-1	-1	-1	

	section	start_ns	end_ns	duration_us	\
0	simulation_step	18336739433700	18336739493700	60	
1	simulation_step	18336739547400	18336739560200	12	
2	simulation_step	18336739584500	18336739592900	8	
3	simulation_step	18336739608700	18336739615600	6	
4	simulation_step	18336739628200	18336739635300	7	
5	simulation_step	18336739648600	18336739655800	7	
6	simulation_step	18336739667300	18336739673400	6	
7	simulation_step	18336739684400	18336739690500	6	
8	simulation_step	18336739701800	18336739708100	6	

```
9  simulation_step  18336739720300  18336739727400           7

                                      extra
0  {'step': 0, 'dt': 0.06, 't_target': 0.06}
1  {'step': 1, 'dt': 0.06, 't_target': 0.12}
2  {'step': 2, 'dt': 0.06, 't_target': 0.18}
3  {'step': 3, 'dt': 0.06, 't_target': 0.24}
4   {'step': 4, 'dt': 0.06, 't_target': 0.3}
5  {'step': 5, 'dt': 0.06, 't_target': 0.36}
6  {'step': 6, 'dt': 0.06, 't_target': 0.42}
7  {'step': 7, 'dt': 0.06, 't_target': 0.48}
8  {'step': 8, 'dt': 0.06, 't_target': 0.54}
9   {'step': 9, 'dt': 0.06, 't_target': 0.6}
```

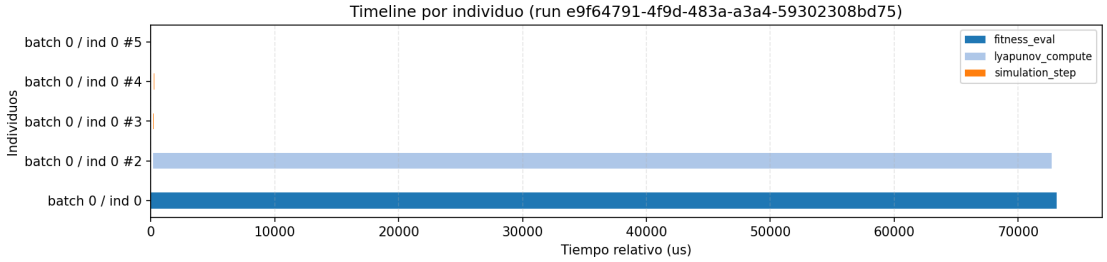
[ ]:

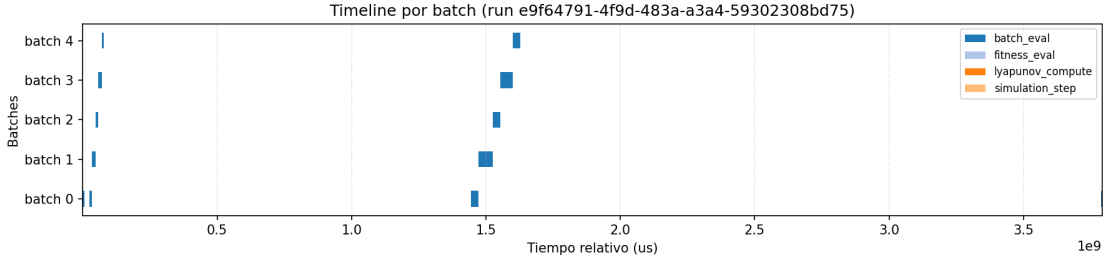
section	count	mean	sum
fitness_eval	2217	9.081621e+05	2013395478
lyapunov_compute	2218	9.074776e+05	2012785257
batch_eval	18	1.065656e+08	1918181606
simulation_step	53102881	1.283903e+01	681789331
ga_main	9	3.466267e+04	311964
crossover	59	1.956780e+03	115450
selection_tournament	59	8.364576e+02	49351
mutation	59	5.801525e+02	34229

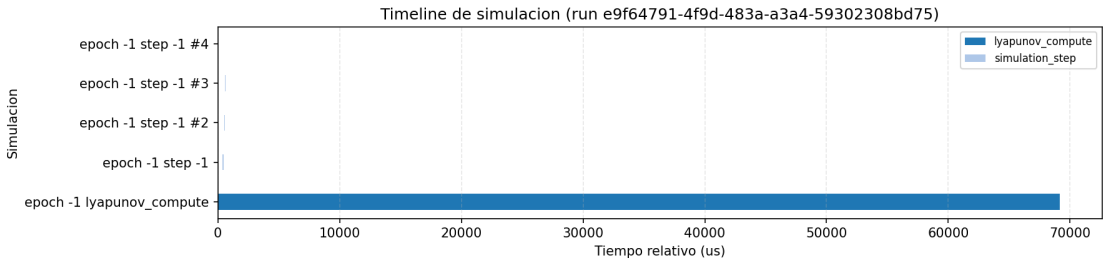
[ ]:
```python
import os
import subprocess
from pathlib import Path
from IPython.display import Image, display

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")),
)
```

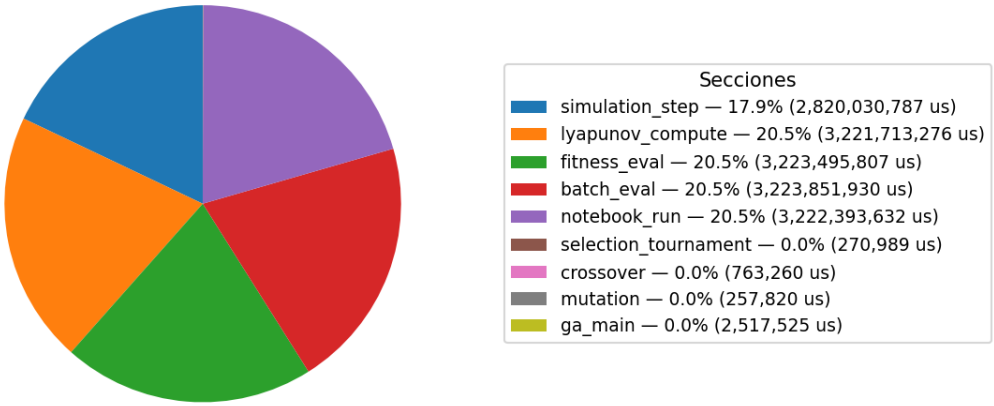
```
Ejecutando: c:\Users\emicr\anaconda3\envs\grav2body\python.exe
scripts/plot_timings.py --run-id e9f64791-4f9d-483a-a3a4-59302308bd75 --top-n 5
Graficas guardadas en C:\Users\emicr\Documents\CODIGOS_FUENTES\TrabajoTerminal\c
ollision_of_two_bodies\two_body\reports
```







on por seccion (run e9f64791-4f9d-483a-a3a4-59302308bd75)



[ ]:
```
# 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}")
#
```