



Instituto Politécnico Nacional

Escuela Superior de Cómputo



Trabajo Terminal No. 2025-B065

Modelo para representar la interacción gravitacional de dos cuerpos

Programa de Ingeniería en Inteligencia Artificial (2020)

Alumnos:

Carrillo Barreiro José Emiliano
Robles Ortero José Ángel

Directores:

Dr. Cesar Hernández Vasquez
Dr. Mauricio Olguín Carbajal

17 de mayo de 2025

Ciudad de México

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

Introducción

Las simulaciones de n^1 -cuerpos, usadas en astrofísica para modelar interacciones galácticas y sistemas planetarios, tienen una restricción: **masas estáticas**. Esto limita la precisión al simular procesos de masa variable, un ejemplo de estos procesos es: la evolución estelar.

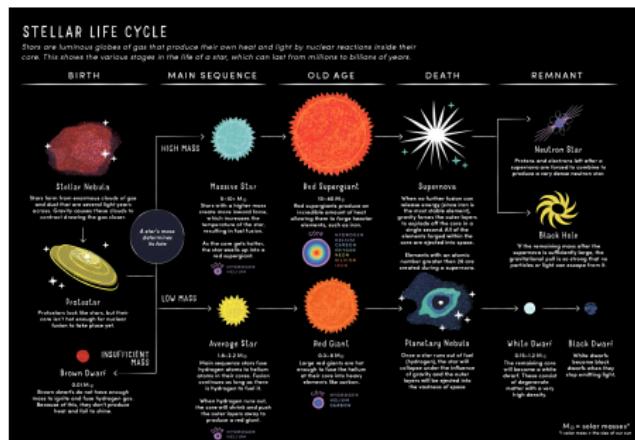


Figura 1: Diagrama del ciclo de evolución de una estrella. Propiedad de: [1]

¹ Durante toda la presentación se usará $n = 2$, dado el acotamiento que se tiene para el presente Trabajo Terminal.

Planteamiento Del Problema

Un problema fundamental en las simulaciones de n -cuerpos es la masa invariable, lo que afecta la estabilidad de los cuerpos involucrados en fenómenos dinámicos como fusiones estelares o acreción. Esta rigidez limita tanto el estado del sistema de n -cuerpos como el potencial para aplicaciones interactivas, requiriendo métodos más flexibles.

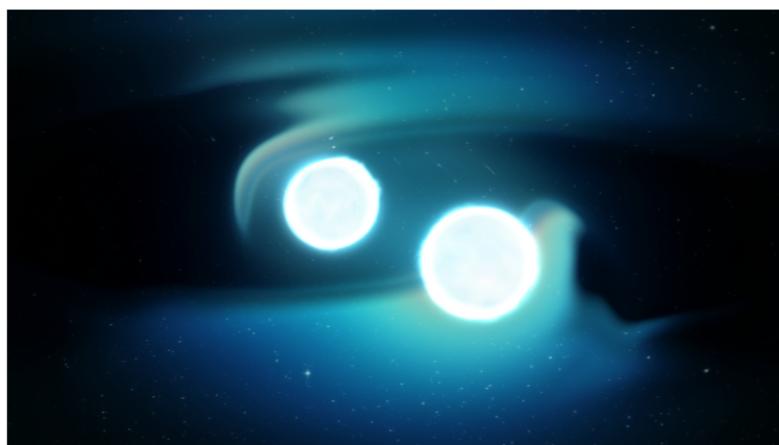


Figura 2: Colisión de dos estrellas de neutrones (Simulación). Extraido de: [2]

Propuesta de Solución

Proponemos un modelo de simulación de n -cuerpos con **modificación dinámica de masas**, de los cuerpos del sistema, en tiempo de ejecución. Se emplearán diversos enfoques de solución al problema de n -cuerpos para **eficiencia computacional**, y algoritmos bio-inspirados para el **ajuste paramétrico adaptativo**. El modelo, **escalable**, se optimizará para hardware accesible, buscando **superar las limitaciones de los modelos actuales**.

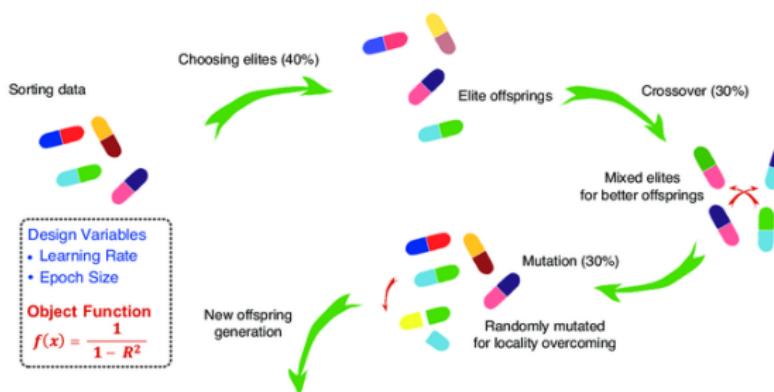


Figura 3: Ejemplo de Aplicación de un Algoritmo Genético (AG) Adaptado de: [3]

Objetivos

Objetivo General

Desarrollar un modelo teórico para la simulación del problema de dos cuerpos que permita la modificación dinámica de la masa, mejorando la estabilidad local del sistema visible en la representación de sus interacciones gravitacionales y eventos asociados.

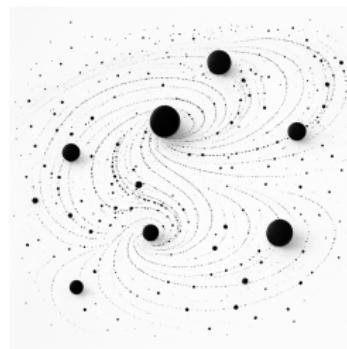


Figura 4: Ejemplo de representación de un sistema de n -cuerpos. *Autoría Propia*

Objetivos I

Objetivo Específicos

Módulo de simulación

Implementar el módulo de simulación encargado de integrar los distintos procedimientos algorítmicos requeridos para obtener la descripción numérica del sistema de interacción de n cuerpos, incluyendo la aplicación de métodos de integración numérica, el cálculo de fuerzas gravitatorias y la detección de colisiones.

Módulo de optimización

Diseñar e implementar el módulo de optimización orientado al ajuste dinámico de las masas de los cuerpos que conforman el sistema de n -cuerpos, mediante el uso de algoritmos bioinspirados, con el fin de identificar el primer conjunto de valores que satisfaga las restricciones impuestas en cuanto a estabilidad y viabilidad del sistema.

Objetivos II

Objetivo Específicos

Módulo de Simulación dinámica

Desarrollar e implementar el modelo computacional de para la simulación dinámica de un sistema de dos cuerpos bajo interacción gravitatoria.

Módulo de visualización

Implementar el módulo de visualización gráfica para la representación dinámica del sistema simulado, mostrando su evolución temporal a lo largo de un conjunto limitado de iteraciones, a fin de apoyar la interpretación de los resultados del modelo

Objetivos III

Objetivo Específicos

Interfaz básica (UI)

Diseñar e implementar una interfaz básica que permita el ingreso estructurado de parámetros asociados a los cuerpos del sistema, diferenciando entre atributos fijos (e.g., radio) y variables susceptibles de optimización (e.g., masa), así como la definición de restricciones obligatorias y optionales que condicionan el espacio de soluciones. Además de permitir visualizar los resultados generados por los módulos de simulación y optimización

Justificación y Relevancia Científico-Tecnológica

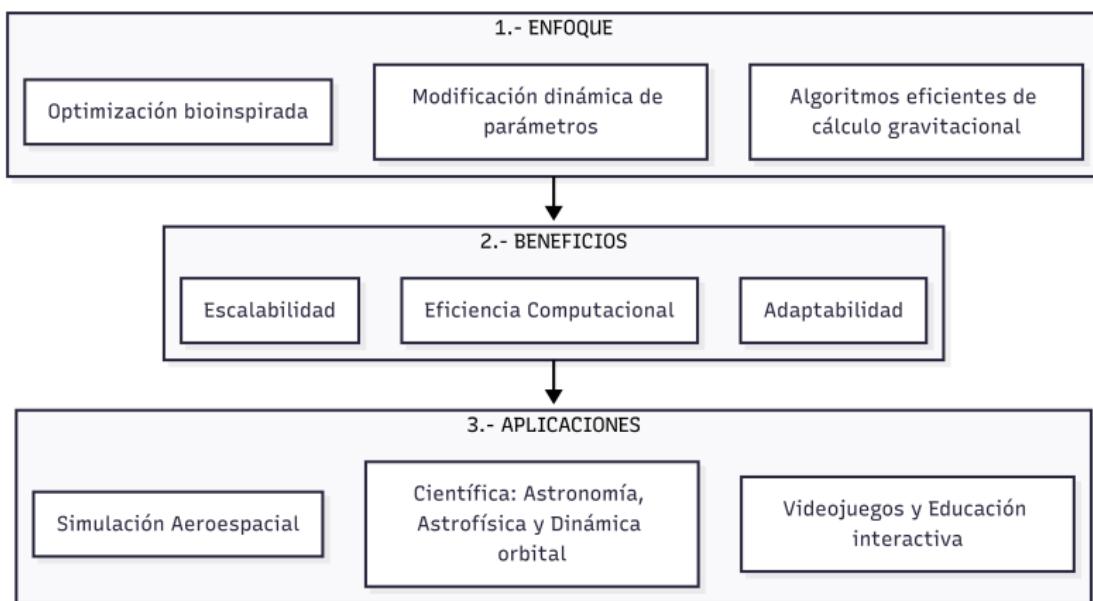


Figura 5: Diagrama representativo de los niveles de Justificación. Autoría Propia

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

1 Introducción

2 Estado del Arte

3 Marco Teórico

Tecnologías: Lenguaje & Librerías.
Métodos & Técnicas a utilizar.

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

1 Introducción

2 Estado del Arte

3 Marco Teórico

Tecnologías: Lenguaje & Librerías.
Métodos & Técnicas a utilizar.

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

Lenguaje de Programación

Python



Figura 6: Logo de Python *Adaptado de: [4]*

- Lenguaje de programación de alto nivel y propósito general
- Sintaxis clara y legible que favorece la productividad
- Multiparadigma: soporta programación orientada a objetos, imperativa y funcional
- Tipado dinámico y administración automática de memoria
- Extensa biblioteca estándar

Biblioteca de Simulación N-Cuerpos

REBOUND

- Implementado en C con interfaz Python muy completa
- Enfocado en dinámica colisional y no colisional, N-cuerpos de propósito general
- Integradores avanzados: Leapfrog, Wisdom-Holman (WHFast), SEI (Simpléctico Epicíclico), IAS15
- Algoritmos de gravedad: Suma directa, Barnes-Hut (Octree)
- Detección y resolución de colisiones físicas



Figura 7: Logo de REBOUND
Adaptado de: [5]

Biblioteca de Optimización

pymoo



Figura 8: Logo de pymoo *Adaptado de: [6]*

- Framework moderno para optimización multiobjetivo (y mono-objetivo)
- Amplia gama de algoritmos: NSGA-II, NSGA-III, MOEA/D, RVEA, CMA-ES, DE, GA, PSO
- Alta flexibilidad para definir problemas, operadores y algoritmos personalizados
- Excelente para análisis multiobjetivo con herramientas específicas
- Soporte para paralelización (Dask, multiprocessing)

Biblioteca de Visualización

Matplotlib

- Biblioteca estándar para visualización en Python
- Gran flexibilidad y personalización
- Excelente para gráficos científicos
- Amplia documentación y comunidad

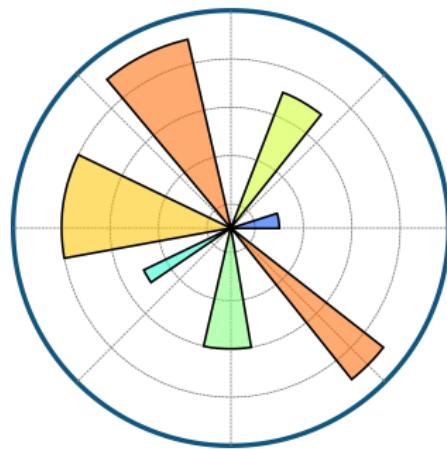


Figura 9: Logo de Matplotlib
Adaptado de: [7]

Biblioteca de GUI para Python

PyQt



Figura 10: Logo de PyQt. *Adaptado de: [8]*

- Basado en Qt (C++), con aspecto profesional y moderno
- Extensa colección de widgets y sistema potente de layouts
- Excelente integración con Matplotlib y otras librerías gráficas
- Buen soporte para multihilo con señales y slots
- Qt Designer para diseño visual de interfaces

1 Introducción

2 Estado del Arte

3 Marco Teórico

Tecnologías: Lenguaje & Librerías.
Métodos & Técnicas a utilizar.

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

Cálculo de Gravedad y Colisiones

Mediante módulos de REBOUND

- **Cálculo de Gravedad:**

- **Suma Directa:** $O(N \cdot N_{\text{active}})$, exacta, ideal para pocos cuerpos
- **Octree (Barnes-Hut):** $O(N \log N)$, aproximada, eficiente para $N > 10^2$

- **Detección de Colisiones:**

- **Búsqueda Directa:** $O(N^2)$, exacta para esferas duras
- **Octree:** $O(N \log N)$, para geometrías 3D generales
- **Barrido Plano:** Eficiente para geometrías quasi-2D

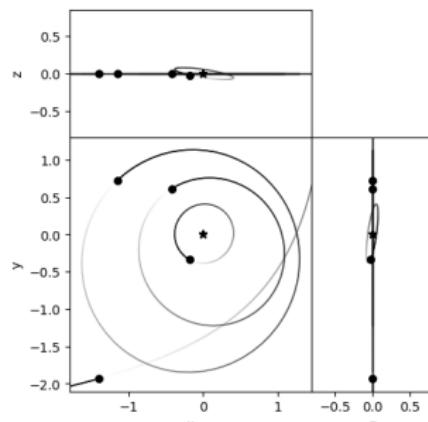


Figura 11: Simulación de órbitas usando REBOUND Adaptado de: [9]

Enfoque del Proyecto

Factibilidad y Optimización

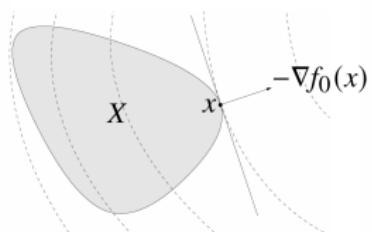


Figura 12: Gráfica 2D que muestre contornos de una función objetivo $f_0(x)$, una región factible sombreada definida por restricciones, y el punto óptimo x^* .
Adaptado de: [10]

- **Problema de Factibilidad (Núcleo):**

- Determinar si existen configuraciones estables
- Criterio clave: Exponente de Lyapunov $\lambda_1 \leq$ umbral
- Restricciones físicas.

- **Marco de Optimización:**

- Función objetivo: minimizar λ_1 directamente
- Algoritmo Genético como herramienta exploratoria
- Permite búsqueda eficiente en el espacio de parámetros

Algoritmo de Exploración

Algoritmo Genético (AG) con pymoo

- **Componentes principales:**

- **Muestreo:**
FloatRandomSampling() para población inicial
- **Selección de Padres:**
TournamentSelection
- **Cruce:** SBX (Simulated Binary Crossover)
- **Mutación:** PolynomialMutation para variables continuas
- **Manejo de Restricciones:**
out["G"] para restricciones físicas

- **Parámetros clave:** Tamaño de población, eliminación de duplicados, semilla aleatoria para reproducibilidad

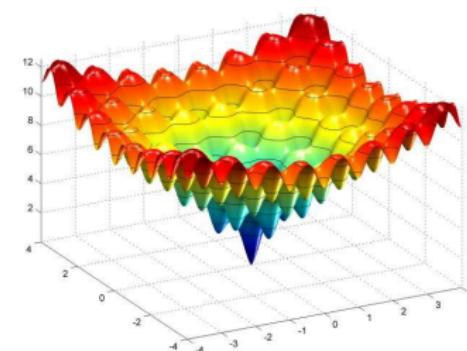


Figura 13: Función tipo benchmark, para la evaluación de AGs *Adaptado de: [11]*

Indicador de Estabilidad

Exponente de Lyapunov

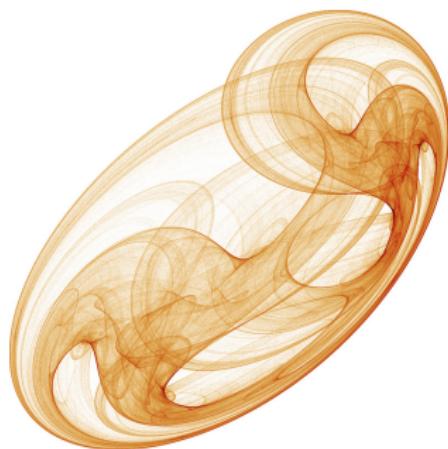


Figura 14: Atractor caótico generado mediante exponentes de Lyapunov. Adaptado de: [12]

- Cuantifica la sensibilidad a condiciones iniciales
- Indicador primario de estabilidad/caos: $\lambda_1 > 0$ implica comportamiento caótico
- Medida objetiva y cuantitativa de la estabilidad dinámica
- Permite predecir comportamiento a largo plazo
- Base matemática rigurosa (Teorema de Oseledec)

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

1 Introducción

2 Estado del Arte

3 Marco Teórico

4 Planeación

5 Análisis

6 Diseño

7 Conclusiones y Trabajo a Futuro

8 Referencias

Referencias I

- [1] Smarty Prints, "Constellation Star Poster: Northern Hemisphere, Stellar Life Cycle, Night Sky Stargazing", Póster educativo que muestra que ilustra el ciclo de vida de una estrella., Smarty Prints. (14 de mayo de 2025), dirección: <https://smartypointsshop.com/products/constellation-star-poster-northern-hemisphere-stellar-life-cycle-night-sky-stargazing-print-16x20-18x24-24x36> (visitado 14-05-2025).

Referencias II

- [2] NASA's Scientific Visualization Studio, "Star Collision", Visualización científica producida por Scott Wiessinger. Visualización por Brian Monroe. Redacción por Francis Reddy. Consultoría científica por Julie McEnery, Brad Cenko y Eleonora Troja., NASA Goddard Space Flight Center. (2 de jul. de 2018), dirección: <https://svs.gsfc.nasa.gov/12949/> (visitado 14-05-2025).
- [3] J. Kim y col., "Prediction of engine NOx for virtual sensor using deep neural network and genetic algorithm", *Oil & Gas Science and Technology - Revue de l' IFP*, vol. 76, pág. 72, nov. de 2021. DOI: 10.2516/ogst/2021054.

Referencias III

- [4] Python Software Foundation, “Python Software Foundation Landing Page”, Python Software Foundation. (14 de mayo de 2025), dirección: <https://www.python.org/psf-landing/> (visitado 14-05-2025).
- [5] H. Rein y col., “REBOUND: an open-source multi-purpose N-body code for collisional dynamics”, *Astronomy & Astrophysics*, vol. 537, pág. 10, 2012, Published online 20 January 2012. DOI: 10.1051/0004-6361/201118085. dirección: <https://doi.org/10.1051/0004-6361/201118085>.
- [6] J. Blank y col., “pymoo: Multi-Objective Optimization in Python”, *IEEE Access*, vol. 8, págs. 89 497-89 509, 2020, Introduces pymoo, a Python framework for multi-objective optimization. DOI: 10.1109/ACCESS.2020.2990567.

Referencias IV

- [7] J. D. Hunter, "Matplotlib: A 2D graphics environment", *Computing in Science & Engineering*, vol. 9, n.º 3, págs. 90-95, 2007. DOI: 10.1109/MCSE.2007.55.
- [8] Qt Group, *Qt Group Frequently Asked Questions*, Qt Group page, last edited May 1, 2025, 2025. dirección: <https://www.qt.io/faq>.
- [9] H. Rein, "Hyperbolic Orbits", Notebook de Jupyter que demuestra la simulación de órbitas hiperbólicas utilizando el código N-body REBOUND., GitHub. (14 de mayo de 2025), dirección: https://github.com/hannorein/rebound/blob/main/ipython_examples/HyperbolicOrbits.ipynb (visitado 14-05-2025).

Referencias V

- [10] S. Boyd y col., *Convex Optimization – Slides*, Presentation slides, Presentation slides accompanying the book "Convex Optimization". Revised by Stephen Boyd, Lieven Vandenberghe, and Parth Nobel. Version date on title page of PDF: 2023., 2023.
- [11] K. Gonçalves-e Silva y col., "Less is more: Simplified Nelder-Mead method for large unconstrained optimization", *Yugoslav Journal of Operations Research*, vol. 28, págs. 14-14, jun. de 2018. DOI: 10.2298/YJOR180120014G.
- [12] P. Bourke, "Random Attractors – Found using Lyapunov Exponents", Página que explora atractores caóticos generados mediante exponentes de Lyapunov en sistemas dinámicos bidimensionales. (oct. de 2001), dirección: <https://paulbourke.net/fractals/lyapunov/> (visitado 14-05-2025).

¡Gracias por su atención!

¿Alguna pregunta o comentario?

Anexos

9 Tablas Comparativas del Marco Teórico

Anexo A : Lenguaje de Programación

Cuadro 1: Comparativa de Lenguaje de Programación

Característica	C++	Python
Velocidad de Ejecución	Máxima. Ideal para cálculos intensivos.	Menor.
Velocidad de Desarrollo	Lento y verboso.	Rápido, claro y flexible para iterar.
Bibliotecas Científicas	Potentes, pero más complejas de integrar.	Vastas y accesibles.
Visualización	Requiere herramientas externas.	Fácil con Matplotlib, Plotly o Bokeh.
Integración con REBOUND	Directa con linking C.	Interfaz oficial en Python lista para usar.
Algoritmos Bioinspirados	Óptimo si se implementan desde cero.	Fáciles de coordinar o conectar con código externo.
Gestión de Memoria	Manual. Mayor control.	Automática. Simplifica el desarrollo.
Curva de Aprendizaje	Alta..	Baja a moderada..
Prototipado / Experimentación	Lento.	Ágil.
Depuración	Complejo.	Sencillo.
Enfoque del Proyecto	Control total, pero puede complicar.	Permite centrarse en la solución y resultados.

Anexo A : Comparativa de Bibliotecas N-Cuerpos

Cuadro 2: Comparativa extensa de bibliotecas para simulación N-cuerpos.

Característica	REBOUND	PKDGRAV3	AMUSE	NBody (Python)	PyGaia
Lenguaje Principal / Interfaz	C (Python API)	C++	Python	Python	Python
Enfoque Principal	Colisional/no colisional, sistemas N-cuerpos	Cosmología a gran escala	Framework multipropósito	Simulaciones educativas	Ánálisis Gaia
Tipos de Problemas	Planetas, cúmulos, anillos	Galaxias, materia oscura	Multifísica astrofísica	Sistemas pequeños	Dinámica galáctica
Algoritmos de Gravedad	Barnes-Hut/Suma directa	TreePM	Hermite/Tree/SPH	Suma directa	Potenciales analíticos
Manejo de Colisiones	Sí (esferas duras)	No	Depende del backend	No	N/A
Integradores Numéricos	WHFast/ IAS15/ Leap-frog	Leapfrog KDK	Hermite/ Symplectic	RK/ Leapfrog	SciPy ODE
Hidrodinámica (SPH/Gas)	No	Sí	Sí	No	No
Paralelización	MPI/OpenMP	MPI	MPI frameworks	Multiprocessing	CPU básica
Flexibilidad/Modularidad	Módulos intercambiables	Enfoque cosmología	Interoperabilidad	Implementación-dependiente	Centrado en Gaia
Facilidad de Uso (Python)	Excelente docs/API	N/A (C++)	Compleja (múltiples backends)	Variable	Astronomer-friendly
Comunidad/Mantenimiento	Activo desarrollo	Cosmología activa	Colaborativo	Individual	Soporte Gaia
Idoneidad para el Proyecto	Óptimo: • Soporta 2 cuerpos • Python • Modular	Inadecuado: Escala/física diferente	Complejidad excesiva para necesidades simples	Muy básico para requisitos	Enfoque observacional no aplicable

Anexo A : Algoritmos de Optimización

Cuadro 3: Comparativa de Bibliotecas de Optimización

Biblioteca	Enfoque Principal	Clave / Fortaleza	Ideal Para (Contexto del Proyecto)	Complejidad / Flexibilidad
pymoo	Framework moderno y completo para optimización multiobjetivo (y mono). Amplia gama de algoritmos.		Problemas complejos, si se requieren múltiples objetivos o algoritmos robustos mono-objetivo.	Moderada / Alta
DEAP	Máxima flexibilidad para construir algoritmos evolutivos desde cero.		Experimentación profunda con la estructura interna de los algoritmos (ej. GA personalizado).	Alta / Muy Alta
Platypus	Optimización multiobjetivo fácil de usar con algoritmos estándar (NSGA-II, SPEA2).		Implementación rápida de MOEAs conocidos, buen punto de partida para multiobjetivo.	Baja-Moderada / Moderada
Inspyred	Framework versátil para varios algoritmos evolutivos y metaheurísticas.		Exploración de diferentes tipos de algoritmos evolutivos si las opciones más nuevas no son prioritarias.	Moderada / Alta
Nevergrad	Optimización sin derivadas (caja negra) ; ideal para funciones costosas/ruidosas.		Cuando la función objetivo (simulación + LE) es compleja y sin gradientes fáciles.	Moderada / Alta
PyGMO/PaGMO	Alto rendimiento y paralelización (backend C++) para problemas globales complejos.		Simulaciones muy costosas donde la paralelización es crítica para la eficiencia.	Moderada-Alta / Alta

Anexo A : Bibliotecas de Visualización

Cuadro 4: Comparativa de Bibliotecas de Visualización

Característica	Matplotlib	Plotly	Bokeh	Seaborn
Uso Principal/Fortaleza	Gráficos científicos.	Interactividad web.	Interactividad web.	Estadísticas.
Animación	Muy capaz. Ideal para simulaciones.	Buena para animaciones web.	Buena para datos en streaming.	Limitada.
Gráficos Estáticos	Excelente.	Buena.	Buena.	Regular.
Interactividad	Básica.	Alta.	Alta.	Básica.
Interfaz	Simple.	Web	Web.	N/A.
Calidad Publicación	Muy alta.	Alta.	Alta.	Alta.
Trayectorias 2D/3D	Directa.	Clara y efectiva.	Buena.	N/A.
Curva Aprendizaje (anim.)	Moderada.	Moderada.	Moderada.	Moderada.
Estética por Defecto	Funcional.	Moderna.	Flexible.	Mejorada estadísticas.
Comunidad/Docs	Muy extensa.	En crecimiento.	Buena.	Buena.
Adecuación Proyecto	Excelente: animación, análisis.	Interactividad web.	Prioridad web.	Ánálisis complementario.

Anexo A : Comparativa de Bibliotecas GUI

Cuadro 5: Comparativa de bibliotecas Python para Interfaces Gráficas.

Característica	PyQt	Tkinter	Streamlit	PySide	Dear PyGui
Toolkit subyacente Estilo	Qt (C++) Profesional, personalizable	Tcl/Tk Anticuado, editable	React (Web) Moderno, limpio	Qt (C++) Igual a PyQt	GPU (ImGui-like) Herramientas/juegos
Complejidad de desarrollo	Media-Alta	Baja-Media	Muy baja	Igual a PyQt	Media
Curva de aprendizaje	Moderada	Baja	Muy baja	Igual a PyQt	Requiere nuevo concepto
Rendimiento	Muy bueno, optimizado	Bueno en GUIs simples	Bueno, depende del navegador	Muy bueno, igual a PyQt	Excelente, acelerado GPU
Widgets disponibles	Extensa colección madura	Básica, útil	Limitada, centrada en datos	Igual a PyQt	Buena para herramientas
Layouts	Extensos, Qt Designer	Pack/grid/place	Automáticos, poco control	Igual a PyQt	Control total programático
Integración gráfica	Excelente con Matplotlib, etc.	Buena con Matplotlib	Excelente para Plotly y Altair	Igual a PyQt	Posible, requiere ajustes
Multihilo	Señales/slots robustos	Limitado, no thread-safe nativo	Abstracción interna	Igual a PyQt	Usuario maneja concurrencia
Multiplataforma	Win/macOS/Linux	Win/macOS/Linux	Navegador (web)	Win/macOS/Linux	Win/macOS/Linux
Licencia	GPL/comercial (LGPL en PyQt5)	Lib. estándar (libre)	Apache 2.0 (libre)	LGPL (comercial viable)	MIT (libre)
Comunidad y documentación	Amplia y activa	Amplia	Activa	Activa y sólida	Creciente y entusiasta
Idoneidad para el proyecto	Excelente: robusto, flexible, GUI + Matplotlib embebido	Adeuada para GUI simple	Menos ideal: mejor para dashboards web	Excelente alternativa LGPL	Paradigma distinto, menos directo