



# 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

19 de mayo de 2025

Ciudad de México

## 1 Introducción

## 2 Estado del Arte

## 3 Marco Teórico

## 4 Planeación

## 5 Análisis

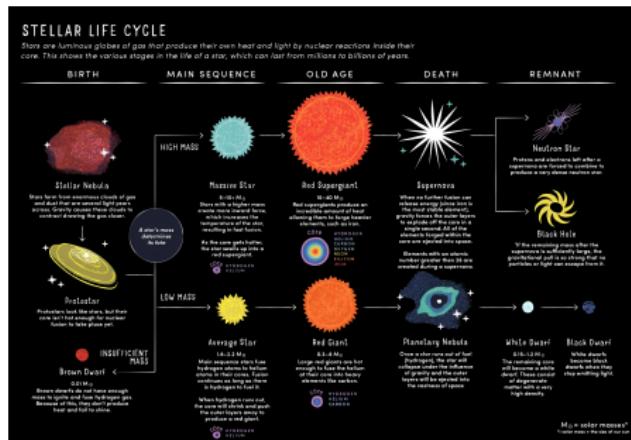
## 6 Diseño

## 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 smartypants constellation poster 2025

<sup>1</sup> Durante toda la presentación se usara  $n = 2$ , 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.

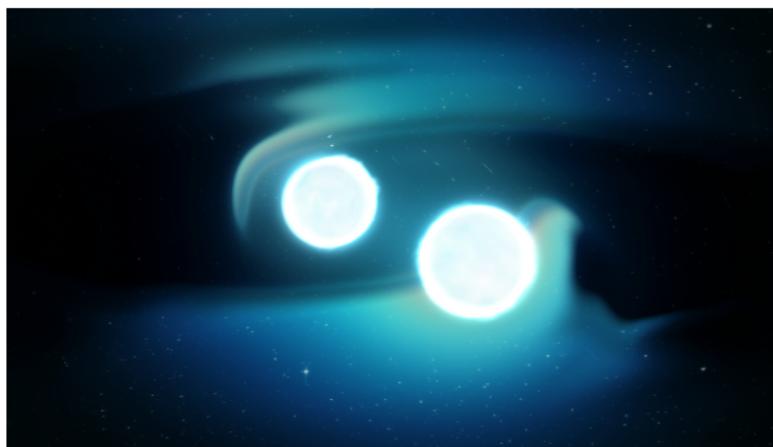


Figura 2: Colisión de dos estrellas de neutrones (Simulación). *Extraido de: nasa\_star\_collision\_2018*

# 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**.

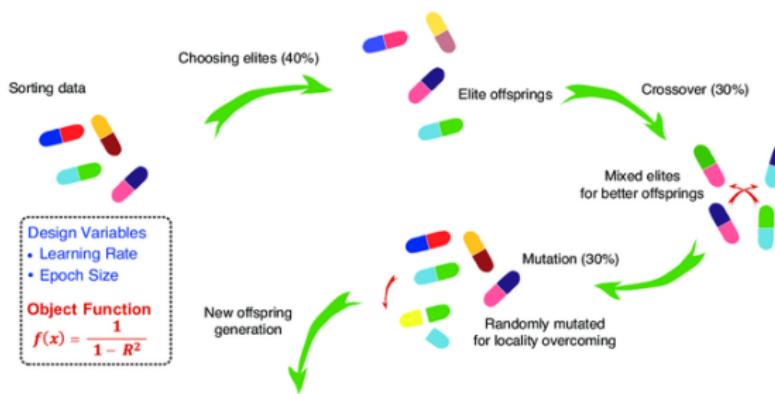
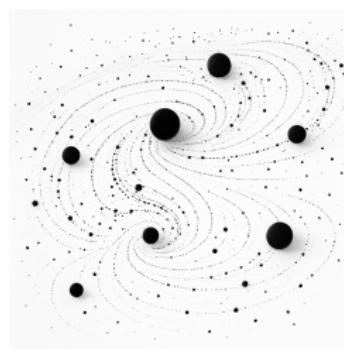


Figura 3: Ejemplo de Aplicación de un Algoritmo Genético (AG) Adaptado de: Kim2021

# 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-Tecnológico

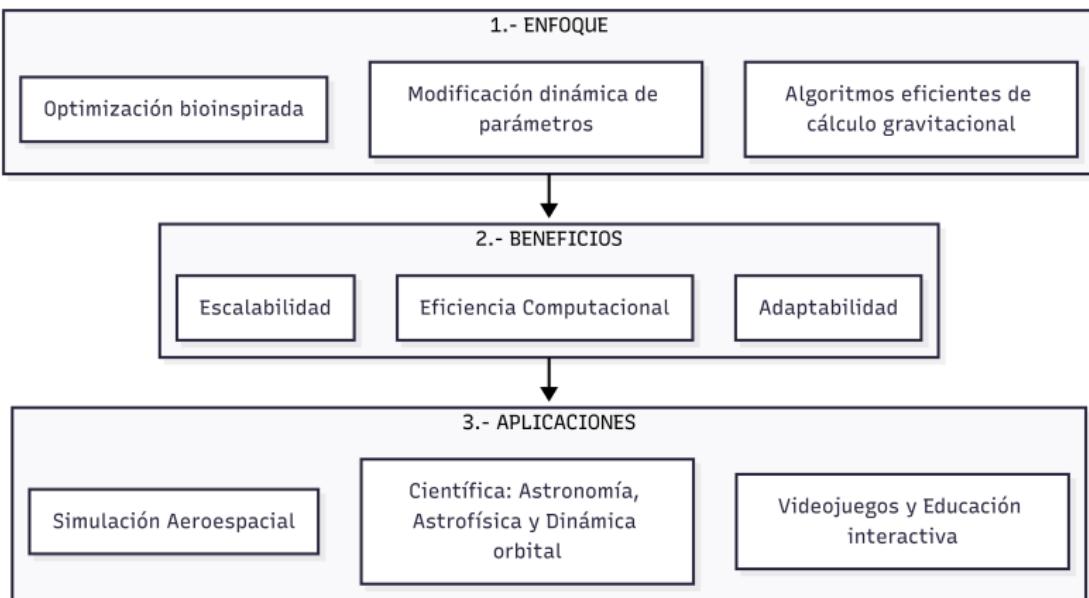


Figura 5: Diagrama representativo de los niveles de Justificación. Autoría Propia

## 1 Introducción

## 2 Estado del Arte

## 3 Marco Teórico

## 4 Planeación

## 5 Análisis

## 6 Diseño

## 7 Conclusiones y Trabajo a Futuro

## 8 Referencias

# Estado del arte

Cuadro 1: Comparativa contra soluciones disponibles

Producto o metodo	Características	Escalabilidad	Usa IA	Cambios dinamicos
ode_num_int	Framework C++11 modular para EDOs; orientado a pruebas de integradores.	Media	No	No
Representación Geométrica	Quadtrees/Octrees para geometría eficiente, no simula dinámica.	Alta	No	No
Método n-NNN	Simulación con n-vecinos y cirugía Hamiltoniana; inspirado en IA.	Alta	Sí	No
PKDGRAV3	Hidrodinámica sin malla (MFM/MFV); vecinos adaptativos.	Alta	No	No

# Estado del arte

Cuadro 2: Comparativa contra soluciones disponibles

Producto o metodo	Características	Escalabilidad	Usa IA	Cambios dinamicos
<b>SPH/N-body Híbrido</b>	Interacciones gas-estrella con árbol Barnes-Hut y pasos bloque.	Alta	No	No
<b>Integrador Simpléctico</b>	Orden 2+, reversible; ideal para colisiones.	Media	No	No
<b>Solver TPM</b>	Combina PM y Tree según densidad local; altamente parallelizado.	Alta	No	No
<b>Algoritmo TPM</b>	Descomposición por densidad; integración multi-escala.	Alta	No	No

# Estado del arte

## Cuadro 3: Comparativa contra soluciones disponibles

Producto o metodo	Características	Escalabilidad	Usa IA	Cambios dinamicos
<b>REBOUND</b>	Modular; incluye varios integradores, colisiones y condiciones frontera.	Alta	No	No
<b>Estabilidad Planetaria</b>	Estudio con REBOUND; Gini vs. inestabilidad. No eventos internos.	Alta	No	No
<b>Solución Propuesta</b>	Combina FMM/Barnes-Hut para cálculo gravitacional eficiente con Algoritmos Bioinspirados para la <i>optimización y ajuste dinámico</i> de parámetros (masa) durante la simulación.	Alta	Sí	Sí

## 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: PythonSoftwareFoundation*

- 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: Rein2012

# Biblioteca de Optimización

pymoo



Figura 8: Logo de pymoo Adaptado de: blank 2020

- 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

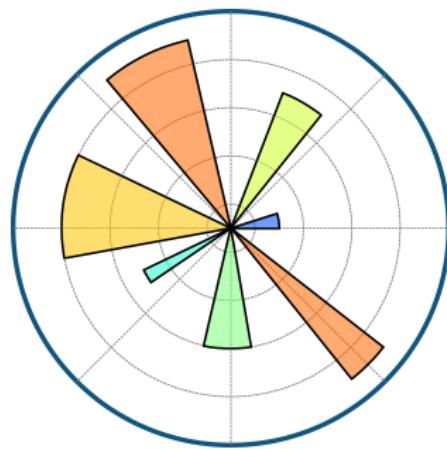


Figura 9: Logo de Matplotlib  
*Adaptado de:* Hunter:2007

# Biblioteca de GUI para Python

## PyQt



Figura 10: Logo de PyQt. *Adaptado de: qt\_wiki*

- 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

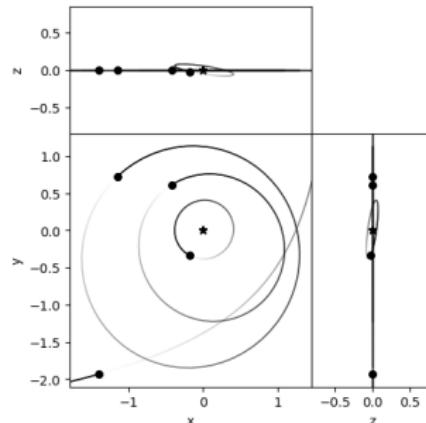
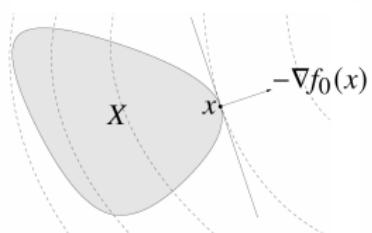


Figura 11: Simulación de órbitas usando REBOUND Adaptado de: rebound\_hyperbolic\_orbits\_2025

# 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: BoydVandenberghe Slides 2023

- **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  $\lambda_1$  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

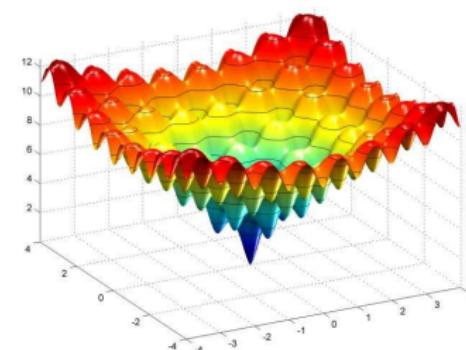
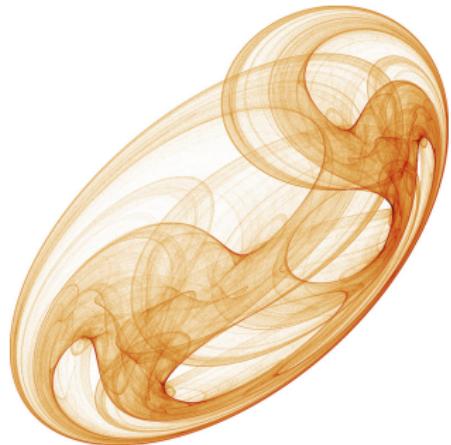


Figura 13: Función tipo benchmark, para la evaluación de AGs Adaptado de: silva2018

# Indicador de Estabilidad

## Exponente de Lyapunov



**Figura 14:** Atractor caótico generado mediante exponentes de Lyapunov. *Adaptado de: bourke\_lyapunov\_attractors\_2001*

- 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

# Juicio de Expertos: M.C. José Alberto Torres León

**Perfil del Experto** Egresado del Centro de Investigación en Cómputo (CIC) del Instituto Politécnico Nacional, sus áreas de especialización incluyen agentes inteligentes para videojuegos (jugadores y dinámicos), optimización evolutiva entre otros.

**Resumen de Opiniones y Observaciones Obtenidas** Tras la revisión del material y la discusión, el M.C. Torres León aportó las siguientes observaciones y valoraciones:

- Modificación de parámetros en tiempo de ejecución
- Simulaciones apegadas a la realidad
- Interacciones dinámicas

# Requerimientos funcionales

- Iniciar simulación dinámica de masa.
- Simular interacciones gravitacionales.
- Usar método multipolar rápido y Barnes-Hut.
- Implementar algoritmos bioinspirados para ajustar dinámicamente los parámetros.
- Incluir módulo de visualización gráfica de evolución.
- Ofrecer interfaz básica para modificar parámetros.

# Requerimientos no funcionales

- Optimizar complejidad de simulaciones.
- Permitir extensión futura a más de dos cuerpos.
- Respetar leyes de la física y mecánica celeste.
- Mantener simulaciones estables al modificar parámetros.
- Ser interfaz intuitiva y accesible.
- Ejecutarse en hardware de gama media.
- Integrarse con entornos virtuales (Unreal Engine).
- Ser modular para facilitar actualizaciones y correcciones.
- Incluir documentación clara.

# Formulación del Problema de Optimización

## Función Objetivo

Minimizar el valor absoluto del exponente de Lyapunov ( $\lambda$ ) en función de las masas  $m_1$  y  $m_2$ :

$$\text{Minimizar } |\lambda(m_1, m_2)|$$

## Variables de Decisión

- $m_1$ : Masa del cuerpo 1
- $m_2$ : Masa del cuerpo 2

# Formulación del Problema de Optimización

## Restricciones

- El exponente de Lyapunov debe ser menor a 0.1:

$$\lambda(m_1, m_2) < 0,1$$

- La masa del cuerpo 2 debe ser menor a 10 veces la masa del cuerpo 1:

$$m_2 < 10m_1$$

- Las masas deben ser positivas:

$$m_1 > 0$$

$$m_2 > 0$$

## 1 Introducción

## 2 Estado del Arte

## 3 Marco Teórico

## 4 Planeación

## 5 Análisis

## 6 Diseño

## 7 Conclusiones y Trabajo a Futuro

## 8 Referencias

# Matriz de procesos

Cuadro 4: Matriz de procesos

Nombre del proceso	Objetivo	Salidas
<b>Captura Parámetros</b>	Recopilar, validar y almacenar parámetros de configuración del usuario.	Estructura ConfigurationData validada, estado de UI actualizado.
<b>Mostrar Resultados</b>	Presentar solución óptima y visualización final al usuario.	Actualización visual de la UI con resultados finales.
<b>Evaluar Fitness</b>	Calcular fitness penalizado ( $F_p$ ) combinando LE y violaciones.	Valor numérico de $F_p(x)$ .
<b>Crear Nueva Simulación</b>	Instanciar un nuevo entorno de simulación vacío en REBOUND.	Referencia a nuevo objeto Simulation.

# Matriz de procesos

Cuadro 5: Matriz de procesos

Nombre del proceso	Objetivo	Salidas
Agregar Cuerpos	Añadir una partícula con propiedades físicas a la simulación.	Instancia sim modificada con nueva partícula.
Iniciar/Ejecutar simulación	Si- Ejecutar la integración numérica paso a paso hasta $T_{\text{máx.}}$ .	Estructura SimulationResult con trayectoria completa.
Recolectar Datos	Extraer estado actual del sistema en instantes de visualización.	Estructura VisualizationState con instantánea del sistema.
Generar Gráficos	Dibujar o actualizar la representación visual en la pantalla.	Representación gráfica actualizada en la UI.

# Diccionario de Datos: Cuerpo celeste

Cuadro 6: Cuerpo celeste.

Nombre del atributo	Descripción	Tipo	Rango	Ejemplo
masa	Masa del cuerpo celeste...	float	> 0 (positivos)...	1.0
a	Semieje mayor de la órbita...	float	> 0 (positivos)	1.0
e	Excentricidad orbital...	float	[0, 1]	0.1
inc_deg	Inclinación orbital...	float	[0°, 180°]	30.0
perturba	Indica si se aplica...	bool	True o False	True

# Diccionario de Datos: Simulación

Cuadro 7: Simulación.

Nombre del atributo	Descripción	Tipo	Rango	Ejemplo
t_max	Tiempo total de simulación...	float	> 0 (positivos)	100.0
N_steps	Número de pasos a almacenar...	entero	> 0 (positivos)	1000
sim.units	Unidades de la simulación...	texto	AU, yr, Msun	AU, yr, Msun
sim.integrator	Indica el integrador numérico...	texto	ias15, whfast, BS, mercurius	ias15
x, y, z	Guarda las posiciones...	array(float) 3x0 (positivos)	[5.0, 20.0]	230.0,

# Diccionario de Datos: Métricas

## Cuadro 8: Métricas.

Nombre del atributo	Descripción	Tipo	Rango	Ejemplo
times	Array que guarda los tiempos...	array(float)	> 0 (positivos)	[0.0, 200.0]
energy	Energía total del sistema...	float	Valor real	-0.5
a_arr, a_pert	Array que guarda el semi-eje...	array(float)	> 0 (positivos)	[1.0, 1.5, 2.0]
e_arr, e_pert	Array que guarda la excentricidad...	array(float)	[0, 1)	[0.1, 0.2, 0.3]
Exponente de Lyapunov ( $\lambda$ )	Indica la tasa de crecimiento...	float	Valor real	0.01

## 1 Introducción

## 2 Estado del Arte

## 3 Marco Teórico

## 4 Planeación

## 5 Análisis

## 6 Diseño

## 7 Conclusiones y Trabajo a Futuro

## 8 Referencias

# Diagrama de Arquitectura

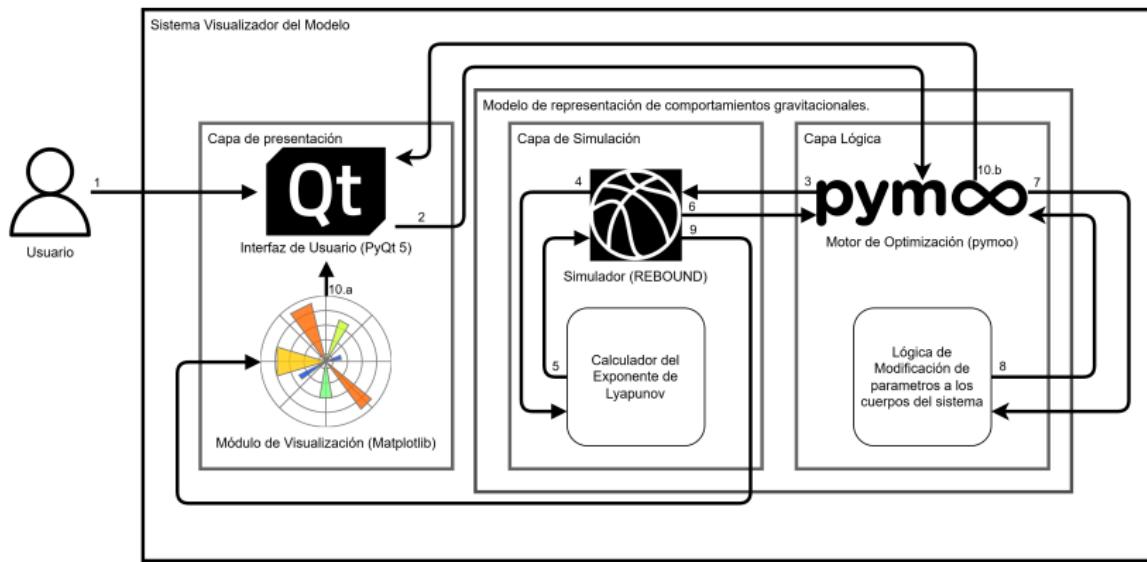


Figura 15: Diagrama de Arquitectura propuesto para el modelo. Autoría Propia

## 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

## 4 Planeación

## 5 Análisis

## 6 Diseño

## 7 Conclusiones y Trabajo a Futuro

## 8 Referencias

# Referencias |

*¡Gracias por su atención!*

¿Alguna pregunta o comentario?

# Anexos

## 9 Tablas Comparativas del Marco Teórico

# Anexo A : Lenguaje de Programación

## Cuadro 9: Comparativa de Lenguaje de Programación

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

## Anexo A : Comparativa de Bibliotecas N-Cuerpos

Cuadro 10: 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	Análisis 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/Trees/SPH	Suma directa	Posiciones analíticas
Manejo de Colisiones	Si (esferas duras)	No	Depende del backend	No	N/A
Integradores Numéricos	WIFast / IAS15 / Leapfrog	Leapfrog KDK	Hermite / Symplectic	RK / Leapfrog	SciPy ODE
Hidrodinámica (SPH/Gas)	No	Si	Si	No	No
Paralelización	MPI/OpenMP	MPI	MPI frameworks	Multiprocessing	CPU básica
Flexibilidad/Modularidad	Módulos intercambiables	Enfoque cosmología	Interoperabilidad	Implementación independiente	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 observational no aplicable

## Anexo A : Algoritmos de Optimización

Cuadro 11: Comparativa de Bibliotecas de Optimización

Biblioteca	Enfoque Principal	Clave / Fortaleza	Ideal Para (Contexto del Proyecto)	Complejidad / Flexibilidad
pymoo	Framework moderno y completo para optimización <b>multiobjetivo</b> (y mono). Amplia gama de algoritmos.		Problemas complejos, si se requieren múltiples objetivos o algoritmos robustos mono-objetivo.	Moderada / Alta
DEAP	Máxima <b>flexibilidad</b> para construir algoritmos evolutivos desde cero.		Experimentación profunda con la estructura interna de los algoritmos (ej. GA personalizado).	Alta / Muy Alta
Platypus	<b>Optimización multiobjetivo fácil de usar</b> 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	<b>Optimización sin derivadas (caja negra)</b> ; ideal para funciones costosas/ruidosas.		Cuando la función objetivo (simulación + LE) es compleja y sin gradientes fáciles.	Moderada / Alta
PyGMO/PaGMO	<b>Alto rendimiento y paralelización</b> (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 12: 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 estéticas.
Comunidad/Docs	Muy extensa.	En crecimiento.	Buena.	Buena.
Adecuación Proyecto	Excelente: animación, análisis.	Interactividad web.	Prioridad web.	Ánalysis complementario.

## Anexo A : Comparativa de Bibliotecas GUI

Cuadro 13: Comparativa de bibliotecas Python para Interfaces Gráficas.

Característica	PyQt	Tkinter	Streamlit	PySide	Dear PyGui
Toolkit subyacente	Qt (C++)	Tcl/Tk	React (Web)	Qt (C++)	GPU (ImGui-like)
Estilo	Profesional, personalizable	Anticuado, editable	Moderno, limpio	Igual a PyQt	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 embeddo	Adeuada para GUI simple	Menos ideal: mejor para dashboards web	Excelente alternativa LGPL	Paradigma distinto, menos directo