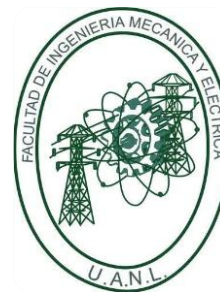




Universidad Autónoma **de Nuevo León**



Facultad de Ingeniería **Mecánica y Eléctrica**

MÉTODOS NUMÉRICOS

Manual Técnico

Día y hora: LMV, V6

Grupo: 005

Periodo: A2025

Catedrático: ORALIA ZAMORA PEQUEÑO

EQUIPO 1

Jorge Aaron Cuellar Fuentes 2007916

Gerardo Ulloa Loredó 2001913

Métodos Numéricos - El Juego

Fecha: 21-11-2025 | **Versión:** 2.0 | **Estado:** Interpolación (Cap. 1) Operativo

1. Resumen del Sistema

Juego educativo en Python (Tkinter) para aprender métodos numéricos. Usa una arquitectura modular donde la **lógica matemática** (methods_engine.py) está separada de la **interfaz gráfica** (game_app.py) y los **datos de las lecciones** (numerical_methods_lessons.py).

2. Estructura de Archivos y Responsabilidades

Archivo	Función Principal
main.py	Punto de entrada. Inicia la app.
game_app.py	Core UI. Maneja ventanas, navegación, usuario, persistencia y renderizado base.
additional_methods.py	Vistas Específicas. Contiene las pantallas personalizadas (tablas, banners) para cada método.
methods_engine.py	Matemáticas. Funciones de cálculo puro (Lagrange, Newton, etc.). Sin interfaz.
methods_mapping.py	Router. Conecta Capítulo/Nivel → Función UI y define colores.
numerical_methods_lessons.py	Datos. Diccionarios con los ejercicios (puntos, respuestas, tiempo) por dificultad.
game_data.py	Estructura base del menú y lecciones teóricas/genéricas.
music_manager.py	Control de audio (pygame) con efectos fade-in/out.
game_progress.json	Base de datos local del usuario (errores, medallas, config).

3. Modelos de Datos Clave

3.1 Lección de Ejercicio (numerical_methods_lessons.py)

```
# Ejemplo de estructura para un ejercicio
'intermedio': {
    'title': 'Nombre del Método',
    'data': [(x1, y1), (x2, y2)], # Puntos para la tabla
    'x_to_find': 3.0,           # Valor a interpolar
    'options': ['Respuesta A', 'Respuesta B'],
    'answer': 'Respuesta A',
    'time': 1200                # Segundos disponibles
}
```

3.2 Mapeo de Métodos (methods_mapping.py)

Este diccionario le dice a la app qué pintar cuando el usuario entra a un nivel difícil:

```
METHODS_MAPPING['Capítulo X']['Nivel Y']['Dificultad'] = {
    'function': 'nombre_funcion_en_additional_methods',
    'banner_color': '#HEXCOLOR'
}
```

4. Guía de Extensión (Cómo agregar un nuevo método)

Para agregar un nuevo tema (ej. "Nuevo Método") con ejercicios personalizados:

1. **Crear Datos:** En numerical_methods_lessons.py, crea un diccionario NUEVO_METODO_LESSONS con claves intermedio, avanzado y final.
2. **Mapear:** En methods_mapping.py, agrega la entrada vinculando el Capítulo/Nivel con una función de renderizado y un color.
3. **Implementar Vista:** En additional_methods.py:
 - Si es una tabla estándar de interpolación, usa la función genérica existente:

```
def show_nuevo_metodo(chapter, level, difficulty, lesson_index):
    # Normaliza dificultad

    diff_key = difficulty.lower().replace(' ', '').replace('pruebafinal', 'final')

    # Carga datos
```

```
data = NUEVO_METODO_LESSONS[diff_key]
```

```
# Renderiza usando la plantilla genérica
```

```
_show_generic_interpolation_exercise(chapter, level, difficulty, lesson_index,  
data, "Título", "#COLOR")
```

- Si requiere una UI única, crea la función desde cero copiando `_show_lagrange_avanzado`.

5. Instalación y Ejecución

Dependencias

Bash

```
pip install pygame Pillow numpy
```

(Nota: tkinter viene incluido con Python).

Ejecución

Bash

```
python main.py
```