**Informe del Programa: Resolución de Problemas Matemáticos Numéricos**

**1. Información General**

Este programa permite al usuario interactuar con un menú para seleccionar diferentes métodos matemáticos y numéricos. Cada método tiene su propia funcionalidad para resolver ecuaciones, aproximar funciones y graficar resultados, utilizando bibliotecas como numpy, matplotlib, math, y scipy.

**2. Estructura General**

* **Menú Principal:** Un sistema de menú permite al usuario navegar entre las diferentes opciones:
  1. Inicio
  2. Método de Taylor
  3. Método de Newton
  4. Método de diferencias finitas
  5. Sistemas de ecuaciones no lineales
  6. Sistemas de ecuaciones lineales
  7. Ecuaciones diferenciales de primer orden
  8. Transformada de Laplace para ecuaciones diferenciales.
* **Bucle Interactivo:** El programa utiliza un bucle while para que el usuario pueda explorar múltiples métodos sin reiniciar el programa.

**3. Descripción de los Métodos Implementados**

**a. Método de Taylor**

* **Funcionalidad:** Este método utiliza la serie de Taylor para aproximar una función en torno a un punto x0x\_0x0​.
* **Detalles Técnicos:**
  + Permite elegir entre funciones sin, cos, o exp.
  + Calcula la serie utilizando la derivada de la función y el factorial, iterando para un número especificado de términos.
  + Gráfica la función original y su aproximación mediante la serie de Taylor.
* **Aplicación:** Se usa en análisis numérico para aproximar funciones alrededor de puntos específicos.

**b. Método de Newton**

* **Funcionalidad:** Resuelve ecuaciones no lineales mediante iteraciones para encontrar raíces.
* **Detalles Técnicos:**
  + Calcula iterativamente nuevas aproximaciones usando xn+1=xn−f(xn)/f′(xn)x\_{n+1} = x\_n - f(x\_n)/f'(x\_n)xn+1​=xn​−f(xn​)/f′(xn​).
  + Previene errores por divisiones entre cero al verificar que f′(xn)≠0f'(x\_n) \neq 0f′(xn​)=0.
  + Gráfica la función y las aproximaciones.
* **Aplicación:** Útil para encontrar raíces de funciones complejas con rapidez.

**c. Método de Diferencias Finitas**

* **Funcionalidad:** Aproxima la derivada de una función en un intervalo dado.
* **Detalles Técnicos:**
  + Utiliza diferencias centradas para calcular f′(x)f'(x)f′(x).
  + Gráfica la función original, su derivada exacta y la aproximada.
* **Aplicación:** Se utiliza en problemas donde no se dispone de una derivada explícita.

**d. Sistemas de Ecuaciones No Lineales**

* **Funcionalidad:** Resuelve sistemas de ecuaciones no lineales.
* **Detalles Técnicos:**
  + Define un sistema con dos ecuaciones y dos incógnitas.
  + Utiliza el método fsolve de scipy para encontrar soluciones aproximadas.
  + Gráfica las curvas de nivel de las ecuaciones y las intersecciones como soluciones.
* **Aplicación:** Resolver problemas complejos en ingeniería y física.

**e. Sistemas de Ecuaciones Lineales**

* **Funcionalidad:** Resuelve sistemas de ecuaciones lineales con matriz de coeficientes y vector de términos independientes.
* **Detalles Técnicos:**
  + Utiliza la función np.linalg.solve para resolver sistemas de tamaño arbitrario.
  + Gráfica la intersección de dos líneas en el caso bidimensional.
* **Aplicación:** Problemas algebraicos en modelado lineal.

**f. Ecuaciones Diferenciales de Primer Orden**

* **Funcionalidad:** Resuelve ecuaciones diferenciales de la forma dydt+ay=b\frac{dy}{dt} + ay = bdtdy​+ay=b.
* **Detalles Técnicos:**
  + Solución general y(t)=ba+Ce−aty(t) = \frac{b}{a} + C e^{-at}y(t)=ab​+Ce−at.
  + Gráfica la solución y muestra los valores inicial y final.
* **Aplicación:** Modelado de sistemas físicos simples, como circuitos eléctricos y disipación de calor.

**g. Transformada de Laplace para Ecuaciones Diferenciales**

* **Funcionalidad:** Resuelve ecuaciones diferenciales utilizando condiciones iniciales y la transformada de Laplace.
* **Detalles Técnicos:**
  + Calcula soluciones en la forma y(t)=C(1−e−2t)+y0e−2ty(t) = C(1 - e^{-2t}) + y\_0 e^{-2t}y(t)=C(1−e−2t)+y0​e−2t.
  + Gráfica la solución para un intervalo dado.
* **Aplicación:** Solución de sistemas dinámicos con entradas impulsivas o escalón.

**métodos implementados**

**menu**  
Muestra las opciones disponibles para el usuario en un menú interactivo.

**main**  
Es el punto de entrada principal del programa y permite seleccionar las diferentes funcionalidades según la entrada del usuario.

1. Serie de Taylor

**tipo\_fun**  
Permite al usuario seleccionar el tipo de función (sin, cos, exp) para calcular su serie de Taylor.

**taylor\_series**  
Calcula la aproximación de la serie de Taylor para una función dada.

2. Método de Newton-Raphson

**newton\_method**  
Implementa el método de Newton para encontrar raíces de funciones no lineales.

**func**  
Define una función (sin(x)) utilizada en el ejemplo del método de Newton.

**dfunc**  
Define la derivada de func (cos(x)) utilizada en el método de Newton.

3. Método de diferencias finitas.

**f**  
Define una función (sin(x)) utilizada en el método de diferencias finitas.

**f\_exact\_prime**  
Calcula la derivada exacta de la función f para comparación en el método de diferencias finitas.

4. Método para ecuaciones no lineales

**system**  
Define un sistema de ecuaciones no lineales.

5. Ecuaciones diferenciales de primer orden

**y (para ecuaciones diferenciales de primer orden)**  
Resuelve una ecuación diferencial de primer orden en forma analítica.

6. Ecuaciones diferenciales transformada de Laplace

**y (para ecuaciones diferenciales con transformada de Laplace)**  
Calcula la solución de una ecuación diferencial utilizando la transformada de Laplace.

**Instalar todas las bibliotecas necesarias para el proyecto en Python**

1. Crear un entorno virtual

Primero, asegúrate de que tienes Python instalado en tu sistema. Luego, abre una terminal y sigue estos pasos:

# Crear un entorno virtual

python -m venv entorno\_numerico

# Activar el entorno virtual

# En Windows:

entorno\_numerico\Scripts\activate

# En Linux o Mac:

source entorno\_numerico/bin/activate

#### Instalar directamente desde la terminal

Ejecuta este comando en la terminal (asegúrate de que el entorno virtual esté activado):

pip install numpy scipy matplotlib sympy pandas

Confirmar las instalaciones

import numpy

import scipy

import matplotlib

import sympy

import pandas

print("Todas las bibliotecas se han instalado correctamente.")

Conclusiones del Proyecto

Consolidación de Conocimientos Matemáticos y Computacionales:

El proyecto permitió profundizar en diversos conceptos matemáticos como las series de Taylor, métodos numéricos para resolución de ecuaciones, diferencias finitas, sistemas de ecuaciones lineales y no lineales, y ecuaciones diferenciales. La implementación práctica de estos conceptos refuerza la comprensión teórica.

Además, se adquirió experiencia en la aplicación de herramientas computacionales como NumPy, Matplotlib, y SciPy, que son fundamentales en el análisis numérico y científico.

Resolución de Problemas Reales con Métodos Numéricos:

Cada módulo del proyecto ofreció una aproximación práctica a problemas que se presentan comúnmente en ingeniería y ciencias aplicadas. Por ejemplo, la aproximación de funciones con series de Taylor y la resolución de ecuaciones diferenciales son herramientas clave en física, economía y muchas otras disciplinas.

Visualización Gráfica como Herramienta de Análisis:

La representación gráfica de las soluciones, como las aproximaciones de Taylor o las derivadas calculadas con diferencias finitas, permitió evaluar visualmente la precisión de los métodos y comprender mejor los comportamientos de las funciones.

Las gráficas también resultaron útiles para identificar errores o problemas, como la falta de convergencia en métodos iterativos.

Importancia de la Precisión y Estabilidad Numérica:

Durante el desarrollo, se destacó cómo factores como el número de iteraciones, el paso en diferencias finitas, o el número de términos en la serie de Taylor afectan la precisión de las soluciones. Este aprendizaje es crucial para optimizar métodos numéricos en problemas reales.

Desarrollo de Habilidades en Programación Científica:

El uso de Python para implementar algoritmos matemáticos complejos ayudó a mejorar habilidades de programación estructurada y orientada a la resolución de problemas. Además, la estructuración del programa mediante funciones y menús interactivos muestra un enfoque organizado y eficiente para resolver problemas numéricos.

Importancia de los Métodos Iterativos:

Métodos como el de Newton y las aproximaciones de diferencias finitas mostraron cómo, mediante iteraciones sucesivas, se puede alcanzar soluciones con gran precisión. Sin embargo, también se identificó la importancia de elegir valores iniciales adecuados para garantizar la convergencia.

Trabajo Colaborativo y Gestión del Proyecto:

La colaboración entre los integrantes del equipo permitió abordar diferentes partes del proyecto de manera eficiente, intercambiar ideas, y resolver problemas en conjunto. Este tipo de trabajo en equipo simula escenarios reales en la resolución de proyectos complejos en la industria o investigación.

Desafíos y Limitaciones:

Algunos métodos presentaron desafíos, como posibles problemas de convergencia en el método de Newton o errores acumulativos en diferencias finitas. Esto destaca la necesidad de seleccionar cuidadosamente el método adecuado según el problema específico y realizar validaciones para asegurar resultados precisos.