Milestone 2:

Justificación de los Apartados:

Después de la clase del jueves 5/10 se han realizado modificaciones al código entregado se realizara el programa milestone2\_conCorreciones.py , el cual es un código único y milestone2\_modulos.py con sus respectivos módulos para separar el código en partes diferenciadas.

Para milestone2\_conCorreciones.py:

1. Condiciones Iniciales y Definición de Función:

Este primer paso es establecer las condiciones iniciales del problema de Kepler, que involucra la posición y velocidad inicial de una partícula en el espacio. Además, se define la función **kepler\_force**, que calcula la fuerza de Kepler en función de las posiciones y velocidades actuales de la partícula. Estas condiciones iniciales y la función de fuerza son fundamentales para llevar a cabo la integración numérica y resolver el problema.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Además, se definirá un vector de tiempos para definir los dt que será el custom\_t, probando valores me di cuenta que el valor mínimo de dt será de 0.01, también intente crear un vector cuyo dt variase. Desde aquí se puede controlar la precion de los esquemas ya sea aumentando el numero de puntos en el intervalo t0-T o reduciendo T para el mismo numero de puntos y perdiendo parte de la solución.

1. Método de Euler:

El método de Euler es una técnica numérica de integración simple pero menos preciso que algunos otros métodos, como veremos más adelante. Permite avanzar en el tiempo en pequeños pasos y calcular la nueva posición y velocidad de la partícula en cada paso.

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

1. Método de Crank-Nicolson:

El método de Crank-Nicolson es un esquema implícito que mejora la precisión de la integración con respecto al método de Euler. Se busca encontrar una función que minimice la diferencia entre dos estados consecutivos en el tiempo.

Texto, Carta

Descripción generada automáticamente

Ademas, me di cuenta que para a = 0 resultaria un esquema euler implícito.

1. Método de RK4 (Runge-Kutta de Cuarto Orden):

El método de RK4 es otro enfoque para la integración numérica que ofrece una mayor precisión en comparación con el método de Euler. Se basa en calcular cuatro pendientes ponderadas en varios puntos en el intervalo de tiempo y combinarlas para obtener una estimación más precisa del siguiente estado.

Texto, Carta

Descripción generada automáticamente

1. Método de Euler Inverso:

El método de Euler Inverso es otro esquema implícito que se utiliza para abordar problemas de integración numérica. Al igual que Crank-Nicolson, se enfoca en encontrar una función que minimice la diferencia entre dos estados en el tiempo.

Texto, Carta

Descripción generada automáticamente

Se añadió un extra del implícito de Euler.

1. Función para Integrar un Problema de Cauchy:

Se crea una función llamada **integrate\_cauchy** que generaliza la integración numérica de los problemas de Cauchy. Esta función permite seleccionar un esquema temporal (cualquiera de los métodos mencionados) y resolver el problema de Kepler a lo largo de un intervalo de tiempo especificado. Almacenará los resultados en matrices para su posterior visualización y análisis.

*Logotipo

Descripción generada automáticamente* Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Diagrama, Forma, Gráfico de burbujas

Descripción generada automáticamente*

*Imagen de la solución para dt=0.01*

*Diagrama, Gráfico de burbujas

Descripción generada automáticamente*

*Imagen de la solución para dt=0.0001*

1. Explicación de los Resultados al Integrar Kepler:

En los gráficos generados por la integración de Kepler con diferentes esquemas temporales (Euler, Crank-Nicolson, RK4 e Inverse Euler), podemos observar las trayectorias orbitales resultantes de cada método:

* Euler: Este esquema muestra una órbita que se aleja gradualmente del camino correcto debido a su naturaleza explícita. La energía total del sistema no se conserva, lo que resulta en una espiral que se aleja del origen.
* Crank-Nicolson: A diferencia de Euler, el método Crank-Nicolson es implícito y conserva mejor la energía total del sistema. La órbita resultante es mucho más precisa y estable en comparación con Euler.
* RK4: El método de Runge-Kutta de cuarto orden es altamente preciso. La órbita es casi idéntica a la esperada y se considera una mejora significativa sobre los métodos anteriores.
* Inverse Euler: Este esquema también es implícito, pero no del todo preciso.

1. Efecto de Variar el Paso de Tiempo:

Cambiar el valor del paso de tiempo (dt) tiene un impacto directo en la precisión y la eficiencia de la integración numérica.

Un paso de tiempo más pequeño produce resultados más precisos, pero con un mayor costo computacional, ya que se requieren más cálculos para avanzar en el tiempo.

Por otro lado, un paso de tiempo más grande puede generar resultados menos precisos, pero es más eficiente en términos de recursos computacionales. En la práctica, encontrar el equilibrio adecuado entre precisión y eficiencia al ajustar el valor de dt es esencial para obtener resultados confiables en problemas de integración numérica.

Resumen/Conclusión sobre la asignación de variables:

La asignación de variables es una técnica aumenta la legibilidad del código y simplifica la comprensión de las operaciones realizadas.

El costo computacional está mayormente determinado por la complejidad del algoritmo utilizado y la cantidad de operaciones ejecutadas en cada intervalo de tiempo. Además, la asignación de variables puede mejorar la eficiencia del código al evitar cálculos redundantes o innecesarios.

Luego el uso de este método es una práctica recomendable para mejorar la claridad del código; por ejemplo, respecto del hito 1.

Para milestone2\_modulos.py:

La estructura de trabajo es la misma pero ahora se han realizados 4 módulos:

El principal: milestone2\_modulos.py

Desde aquí se llama al resto de módulos.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Módulo: cauchy\_problem.py

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Módulo temporal\_schemes.py

Texto

Descripción generada automáticamente

Módulo non\_linear\_systems.py

Texto, Carta

Descripción generada automáticamente

La estructura principal sigue siendo la misma, pero ahora permite una mayor compactación del código y una diferenciación de funciones según su uso.

Quizá una de las posibles mejoras es añadir otro módulo de funciones a integrar, como en este ejercicio solo se ha usado la de Kepler, no me he molestado en poner más o en el módulo de non\_linear\_systems.py se podría añadir más métodos y no solo el de newton, pero como conclusión esta estructura permite una mayor facilidad de entender que estas programando.