Informe Milestone 1:

1. Descripción general

El primer hito tiene como objetivo la integración de órbitas de Kepler mediante los métodos numéricos de Euler, Crank-Nicolson y Runge-Kutta de orden 4. Para ello, se han definido vectores de condiciones iniciales, así como dominios e intervalos de tiempo, para cada uno de los tres modelos. Se ha actuado de esta forma para que, de cara a un uso continuo, no sea necesario seleccionar nada dentro del código y se puedan comparar de manera sencilla las actuaciones de cada uno. Esto conlleva algo más de código al inicio del script para la inicialización de todas estas variables. Asimismo, se ha de comentar que la comparación de resultados se ha basado en las gráficas de las órbitas. A continuación se presenta un análisis detallado de los resultados obtenidos con cada uno, así como los principales problemas que se han encontrado.

2. Método de Euler

El método de Euler es el más sencillo de todos, y al ser explícito no tiene mayor complicación desde el punto de vista del código. Estos son los resultados obtenidos para diferentes valores de intervalo temporal y número de puntos en el dominio del tiempo.

a. Δt = 0.1, N = 100

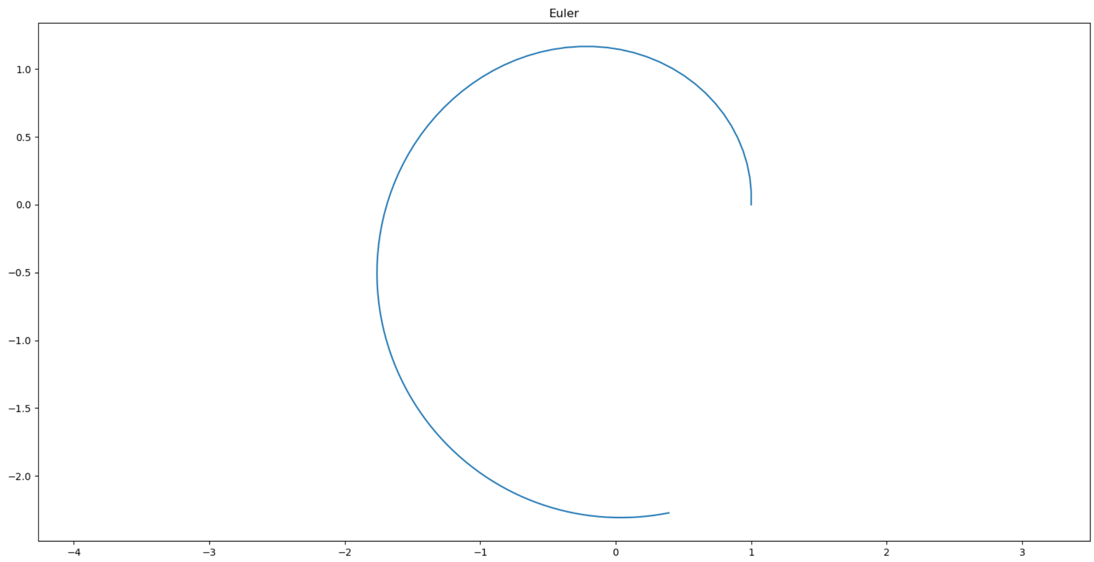

Figura 1: Euler (Δt = 0.1, N = 100)

Se observan malos resultados, la órbita no llega a cerrarse y alcanza valores negativos en torno a -2, cuando realmente se sabe por el desarrollo teórico que con las condiciones iniciales dadas, la trayectoria debería ser una circunferencia.

b. Δt = 0.01, N = 1000

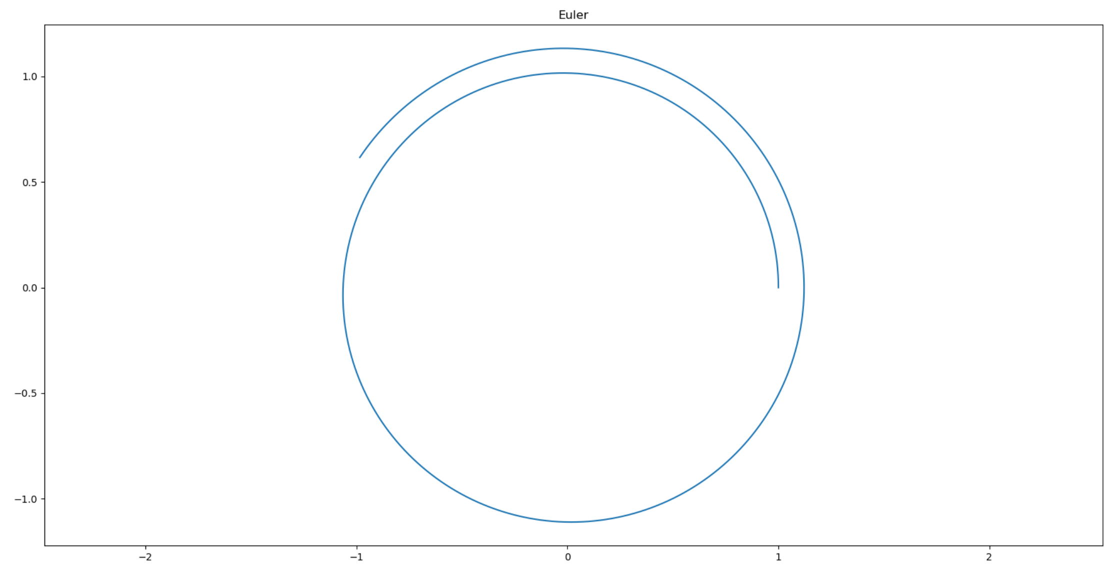

Figura 2: Euler (Δt = 0.01, N = 1000)

Incluso con intervalos de tiempo más pequeños, el método de Euler es incapaz de conseguir un cálculo realista y preciso de la órbita. El poco coste de programación no está justificado por la baja calidad de los resultados.

c. *Δt = 0.001, N = 10000*

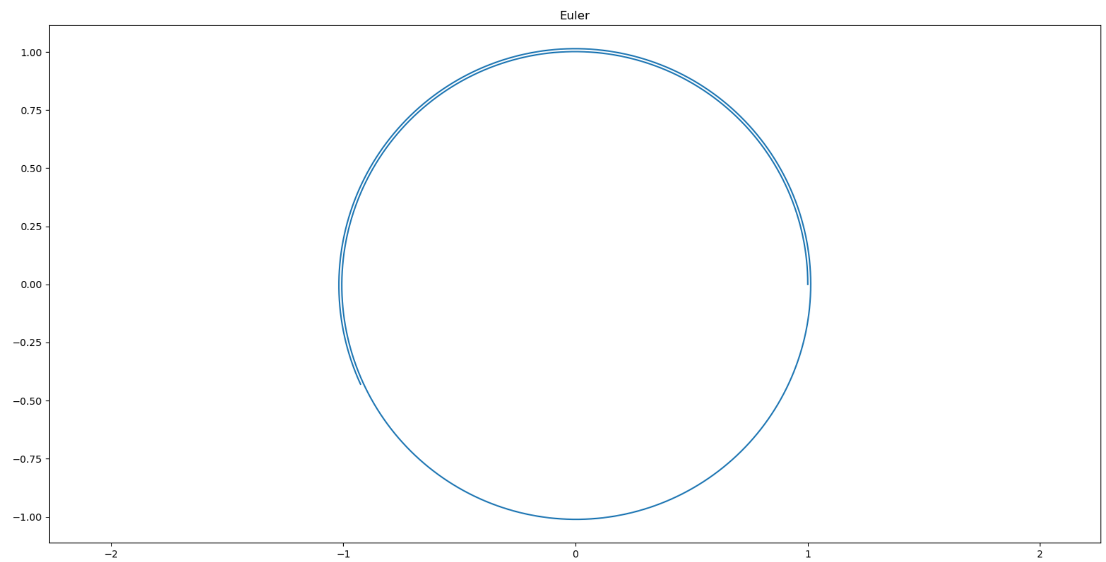

Figura 3: Euler (Δt = 0.001, N = 10000)

Reduciendo dos órdenes de magnitud el valor del intervalo de tiempo el Euler comienza a mostrar resultados con sentido, aunque la órbita no llegue a cerrarse por completo. Reducir más el intervalo de tiempo podría aumentar el coste computacional, lo que podría tener consecuencias negativas al aumentar el tiempo de cálculo.

3. Método de Crank-Nicolson

El método de Crank-Nicolson es el más complejo de los tres ya que se trata de un método implícito, lo que significa que la aproximación numérica de la función en un instante de tiempo N+1 no solo es función de lo que ocurre en el instante de tiempo N, sino también en el N+1. Esto provoca que el código sea bastante más complicado. En un primer momento se planteó una aproximación de la función en el instante N+1 (mediante un método de Euler), que fuera luego utilizada para el cálculo real de la función F y el resultado U en el instante N+1. Sin embargo, los resultados obtenidos con esta hipótesis no fueron positivos. Por ello, se pasó a la resolución del sistema numérico mediante esquemas de Netwon sobre el residuo de las expresiones de este método numérico. Estos son los resultados obtenidos para diferentes valores de intervalo temporal y número de puntos en el dominio del tiempo.

a. $\Delta t = 0.1$, $N = 100$

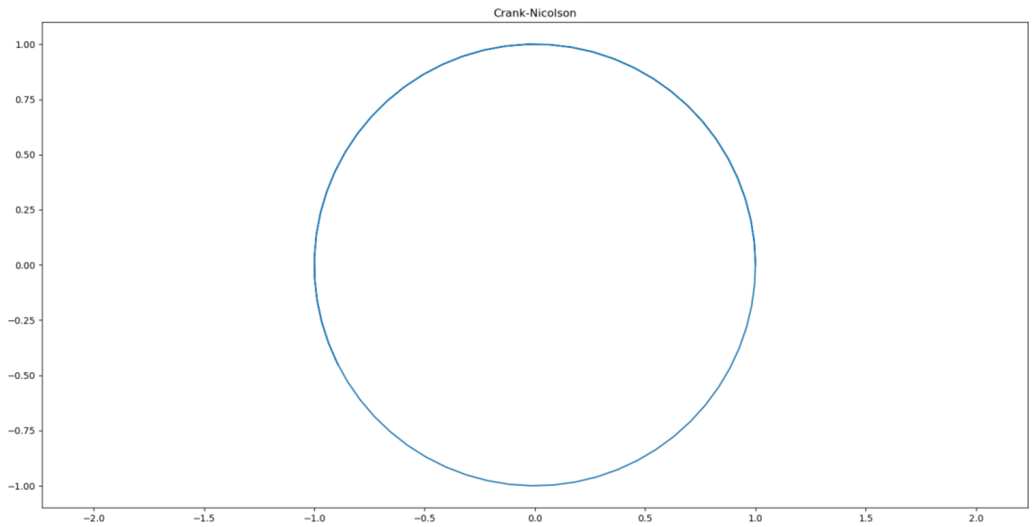

Figura 4: Crank-Nicolson ($\Delta t = 0.1$, $N = 100$)

El método de Crank-Nicolson ofrece mucho mejores resultados, incluso con intervalos de tiempo tan grandes. Esto se debe a que, por norma general, los métodos implícitos son más robustos al tener en cuenta información de varios instantes de tiempo.

b. $\Delta t = 0.01$, $N = 1000$

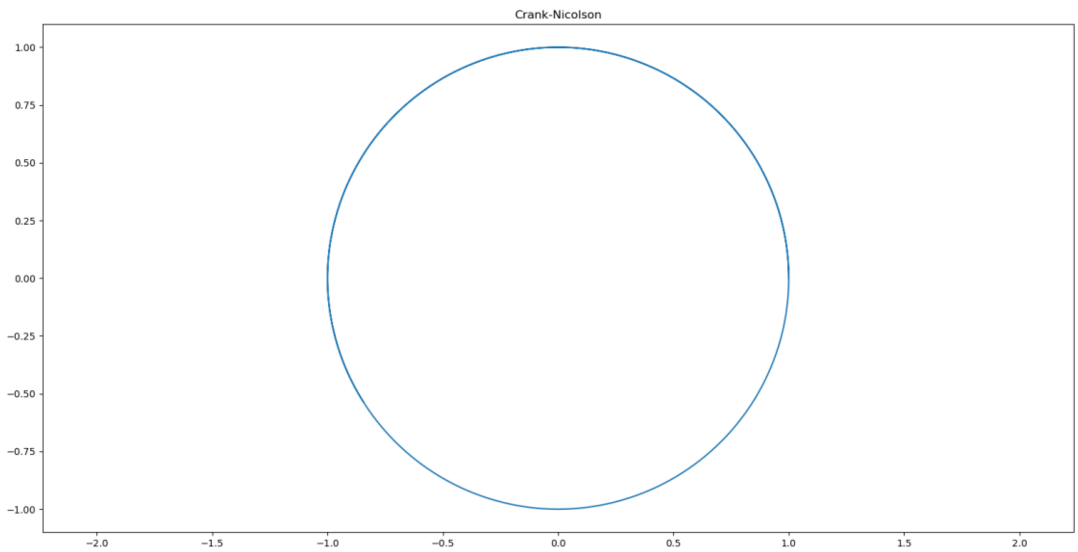

Figura 5: Crank-Nicolson (Δt = 0.01, N = 1000)

El Crank-Nicolson mantiene los buenos resultados, llegando a alcanzar mayores niveles de precisión con órbitas cerradas casi perfectas.

c. *Δt = 0.001, N = 10000*

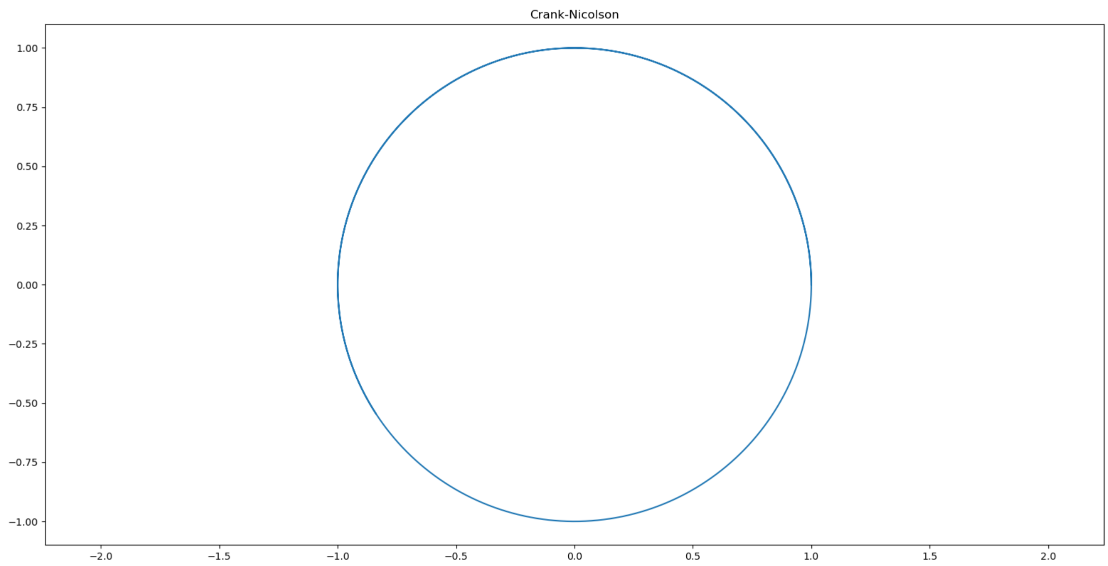

Figura 6: Crank-Nicolson (Δt = 0.001, N = 10000)

Al aumentar aún más la resolución la precisión del Crank-Nicolson mejora. El estudio a pequeñas escalas muestra que se trata de la mejor opción para este problema.

4. Método de Runge-Kutta 4

El método de Runge-Kutta de orden 4 es un método explícito, y por tanto no es excesivamente complejo. Su nombre se debe a que hace un cálculo ponderado de cuatro instantes de tiempo para el cálculo de la solución en el siguiente instante temporal. Al tener en cuenta esa información es más robusto, como el Crank-Nicolson, manteniendo la simplicidad de cálculo y de sus expresiones.

a. $\Delta t = 0.1$, $N = 100$

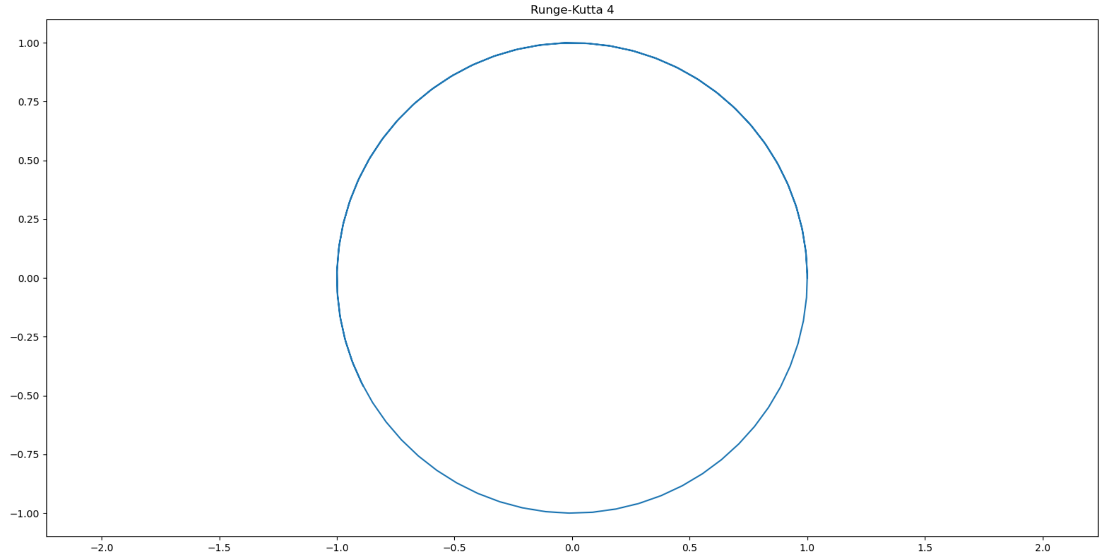

Figura 7: RK4 ($\Delta t = 0.1$, $N = 100$)

Para estos valores de tiempo, se observan resultados muy parecidos al Crank-Nicolson.

b. $\Delta t = 0.01$, $N = 1000$

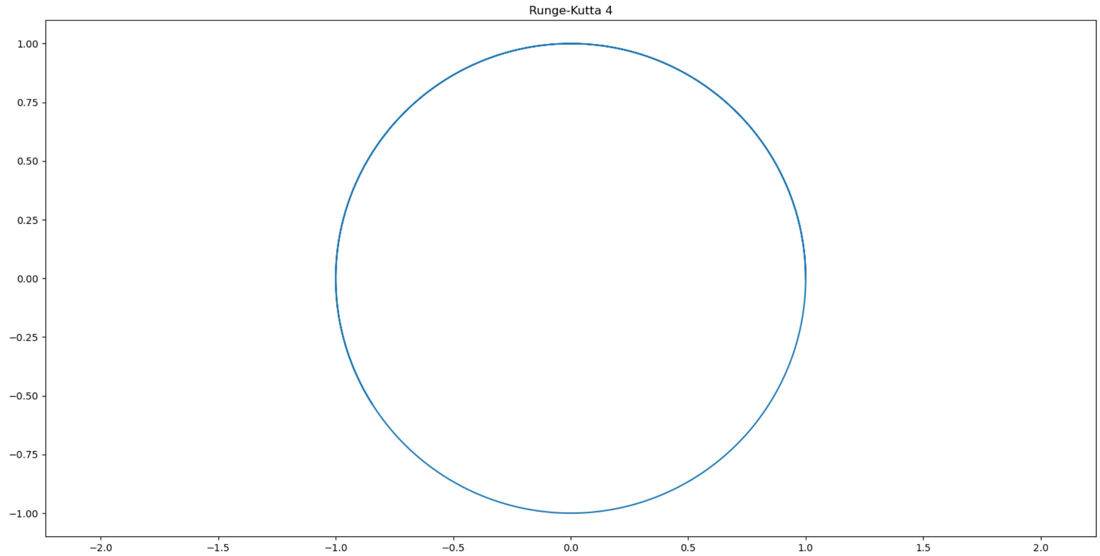

Figura 8: RK4 ($\Delta t = 0.01$, $N = 1000$)

Se observan resultados parecidos a los del Crank-Nicolson, aunque análisis más detallados revelan que este último alcanza mayores niveles de precisión.

c. $\Delta t = 0.001$, $N = 10000$

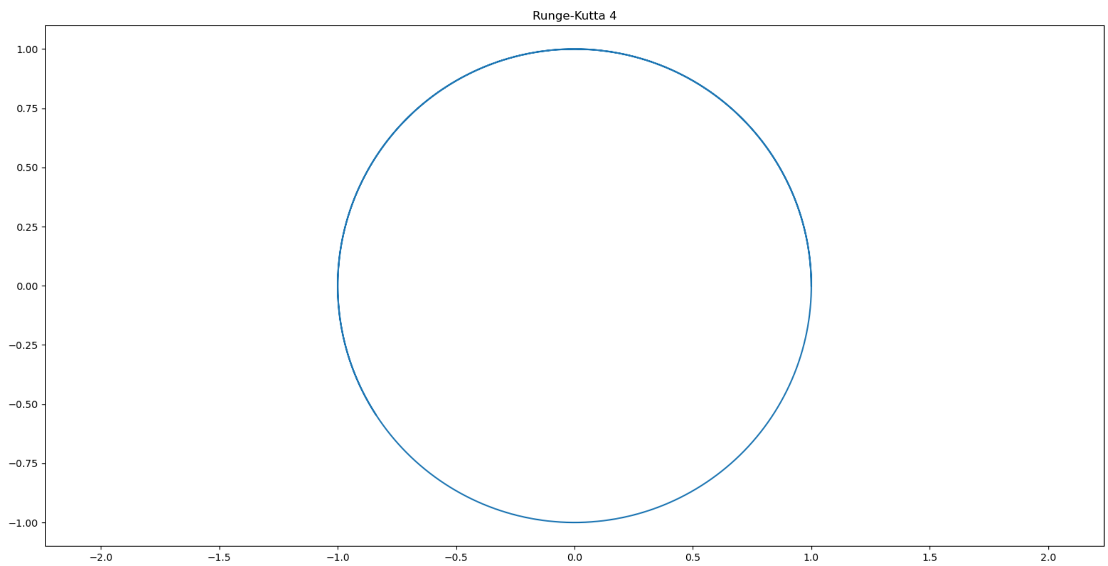

Figura 9: RK4 (Δt = 0.001, N = 10000)

El RK4 se plantea como una solución de compromiso, al producir soluciones de nivel de precisión aceptable sin un coste de programación excesivamente elevado.

5. Consideraciones generales

La programación de este script ha resultado tediosa debido a que debían inicializarse todas las variables de manera separada y no ha sido posible reutilizar ninguna parte de los métodos numéricos con los demás. Además, la programación de los esquemas numéricos por asignación dentro de bucles para calcular la solución en muchos instantes de tiempo ha complicado su ejecución. Especialmente complicado ha sido el Crank-Nicolson, como se ha mencionado antes, al tener que utilizar solvers de Newton para resolver el sistema de ecuaciones. Por último, para ahorrar coste computacional y simplificar la asignación y los cálculos, la función U (solución del problema) se reescribe con cada pasada del bucle del método numérico, lo que obliga a volcar los resultados en otras variables para poder almacenarlos y posteriormente representarlos.