Informe Milestone 2:

1. Descripción general

El segundo hito tiene como objetivo la programación, dentro de un solo script, de una función que, tomando como inputs un esquema temporal, unas condiciones iniciales, un dominio temporal dividido en intervalos de tiempo y la función a la que se iguala la derivada en la EDO, para poder resolver un Problema de Cauchy. También se han definido dentro del script las funciones que integran los diferentes esquemas temporales ofrecidos (Euler, Crank-Nicolson, Runge-Kutta de orden 4 y Euler inverso) para un intervalo de tiempo.

La manera de trabajar, por lo tanto, ha sido completamente diferente a la del hito 1. En primer lugar, en este caso se ha trabajado con matrices de N columnas (siendo N el número de instantes de tiempo considerados), lo que ha permitido que no sea necesario reescribir los resultados tras cada bucle temporal, lo que permite una mayor flexibilidad para el estudio de las soluciones cuando ya se han obtenido para todo el dominio de tiempo. A pesar de esto, para facilitar la definición de las funciones de integración de los esquemas temporales en un intervalo de tiempo, los bucles definidos dentro de la función para resolver el Problema de Cauchy marcan el índice de la columna que se calcula con cada paso del bucle. Estos bucles se activan dependiendo de cuál haya sido el esquema temporal elegido, y van calculando la aproximación numérica de la solución en cada instante mediante llamadas a la función del esquema temporal correspondiente. Las funciones para los esquemas se han basado en los desarrollos del hito 1. El método de Euler Inverso, que no aparecía en el hito anterior, es otro método implícito de mayor sencillez que el Crank-Nicolson, por lo que la manera de plantearlo es idéntica, con el uso de métodos de Newton para resolver la ecuación del residuo.

Una vez que se han programado todas las funciones se plantea un problema ejemplo para comprobar el correcto funcionamiento del script. En este caso se ha utilizado una órbita de Kepler idéntica a la usada en el hito 1, con un Δt de 0.001 y 10000 instantes de tiempo considerados. Se han fijado así los parámetros porque se vio en el hito 1 que eran los que ofrecían mayor nivel de precisión. A continuación se muestran las órbitas calculadas por cada uno de los esquemas temporales con el script.

2. Método de Euler

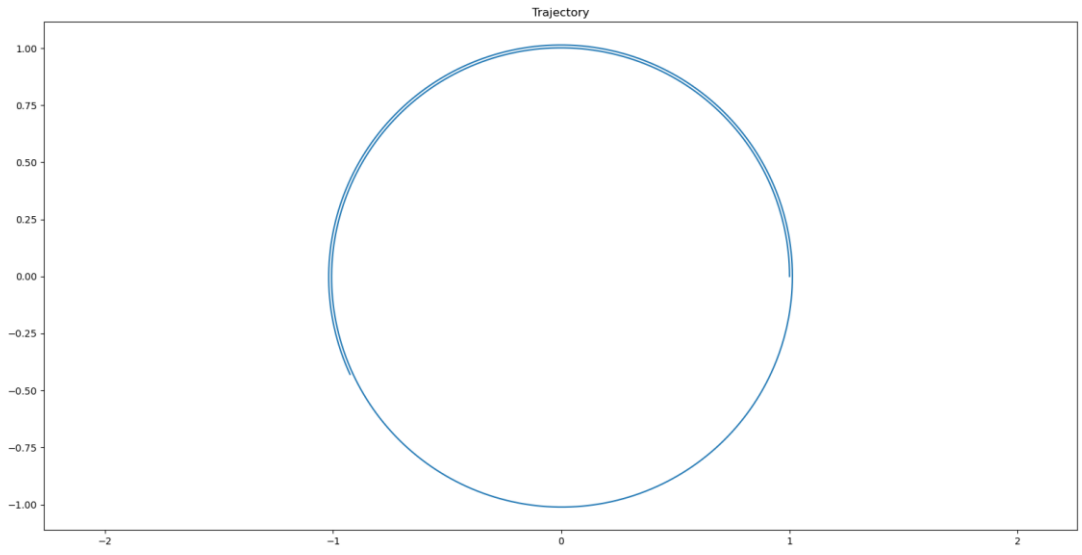

Figura 1: Euler

Al igual que ocurría con el Hito 1, el método de Euler no predice la trayectoria de forma correcta, no llegando a cerrar la órbita y con niveles de precisión muy lejos de los de otros esquemas temporales. Sin embargo, la similitud de los resultados con los obtenidos en el hito 1 revela que las diferentes funciones trabajan de manera correcta de forma individual y con las diferentes llamadas.

3. Método de Crank-Nicolson

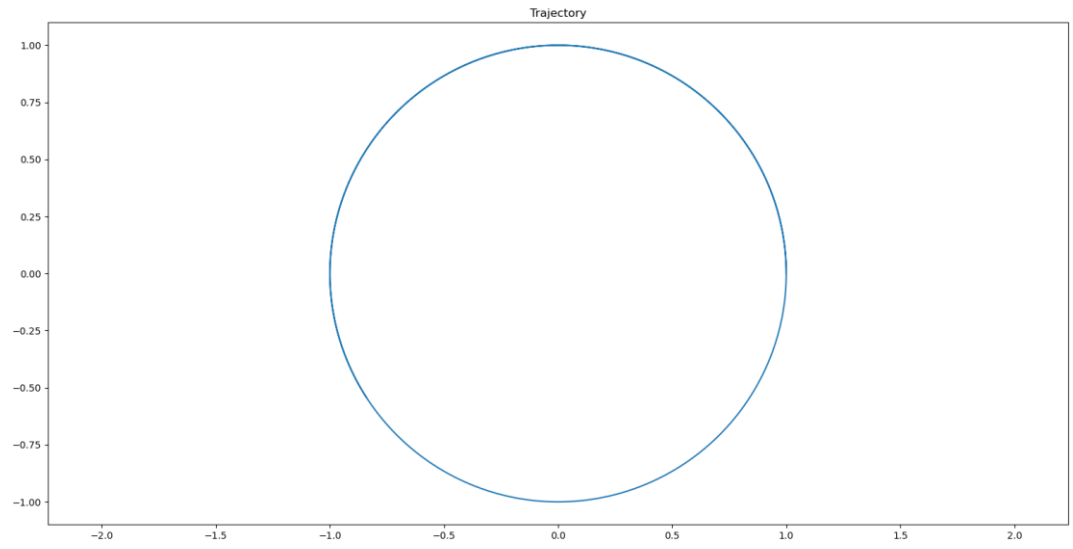

Figura 2: Crank-Nicolson

El método de Crank-Nicolson ofrece mucho mejores resultados, igual que ocurría en el hito 1. En este caso, la mayor sencillez en la programación lo plantea como el mejor candidato para la resolución de este tipo de problemas de condiciones iniciales.

4. Método de Runge-Kutta 4

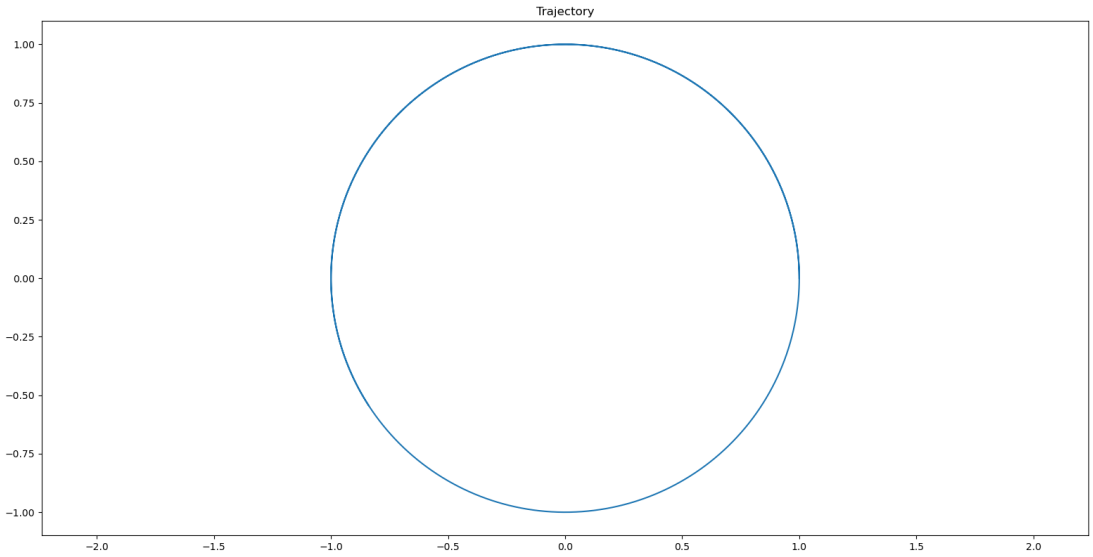

Figura 3: RK4

De nuevo, el script funciona correctamente para el método de Runge-Kutta de orden 4, con resultados positivos con órbita cerrada y circular.

5. Método de Euler Inverso

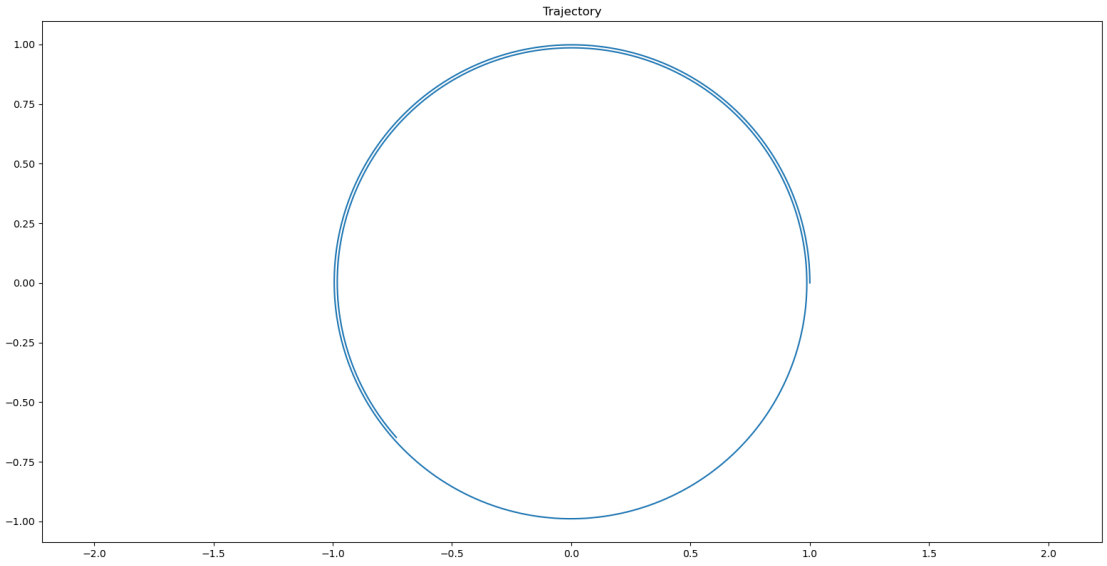

Figura 4: RK4

A pesar de ser un método implícito, y de manera parecida a lo que ocurre con el método de Euler explícito, el Euler Inverso es incapaz de hacer aproximaciones numéricas adecuadas de la órbita con este nivel de resolución temporal. La órbita no llega a cerrarse y parece que disminuye el radio con cada iteración. Esto, unido a la mayor complejidad y el mayor coste computacional al tratarse de un método implícito, llevan a pensar que probablemente no sea una opción adecuada para este tipo de problema.

6. Consideraciones generales

La manera de trabajar del hito 2 es mucho más sencilla y clara. Por un lado, no exige la cantidad de código y de asignaciones que se veían en el hito 1, lo cual reduce además el coste computacional de manera significativa. Asimismo, aporta mucha mayor flexibilidad a la hora de resolver el problema porque no es necesario inicializar un número tan elevado de variables.

Es cierto que requiere de algo más de análisis en las etapas primeras de la programación, sobre todo en el apartado de dimensionado de las variables y de arquitectura del script, para garantizar que las llamadas entre funciones son coherentes y que las variables que aparecen como inputs en unas y outputs en otras no cambian al pasar de una función a otra. También, en mi caso particular debí dedicar bastante tiempo antes de la definición de las funciones para los esquemas a pensar en dónde iban a estar los bucles para integrar a lo largo del tiempo. Esto me ocurrió especialmente en el caso de los métodos implícitos, en los que en un principio pensaba que al tener que considerar varios instantes de tiempo sería importante controlar en qué índice dentro de las matrices ocurrían las operaciones (lo cual está controlado por el instante de tiempo y, por tanto, por el bucle). Al final considero que he resuelto el problema de manera correcta dejando los bucles en la Función de solución del Problema de Cauchy para dejar las funciones de los esquemas temporales únicamente con las expresiones de dichos esquemas.

La compartimentalización del trabajo permite reconocer de manera mucho más directa los errores, por lo que los tiempos de depurado y corrección del código han sido mucho menores que para el hito anterior. Dentro de este ámbito, además, el menor número de asignaciones frente a un mayor número de operaciones que surgen directamente de la matemática (como el caso de los esquemas temporales) implica menor número de instancias que chequear, lo que reduce también las correcciones en el código.

Por todo ello, considero que la manera de trabajar en el hito 2 se adecúa mucho mejor a las necesidades de programación para resolución de este tipo de problemas que la del hito 1.