Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestone 2: Prototipos de integración de órbitas con funciones

Ampliación de Matemáticas I:
Cálculo numérico

08 de octubre de 2022

Autor:

- Daniel Cerón Granda

ÍNDICE

PÁGINA

1.Introducción

El objetivo de este trabajo es la realización de un código que sea capaz de simular de manera básica el recorrido de una órbita. Para ello se aplicarán dos esquemas numéricos implícitos (Euler y Runge-Kutta de orden 4) a un problema de Cauchy o de valor inicial.

Un problema de Cauchy es una ecuación de la forma:

$$\frac{dU}{dt} = F(U,t)$$

Ecuación 1: Expresión general del problema de Cauchy

Además, este problema debe tener una condición inicial en t = 0 tal que: U(0) = U_0. Tanto F como U son vectores (o escalares, si fuera el caso) formados por números reales en sus componentes.

La expresión de una órbita circular es la siguiente:

$$\ddot{\vec{r}} = \frac{-\vec{r}}{|\vec{r}^3|}$$

Ecuación 2: Expresión de la ecuación de una órbita circular

Que a simple vista parece no ajustarse a la expresión del problema de Cauchy. Para transformar esta expresión, expresamos el vector de posición en función de sus componentes (x, y) y también incluimos en la U del problema de Cauchy a las derivadas de las componentes del vector de posición, de tal forma que el vector U queda:

$$U = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix}$$

Ecuación 3: Expresión del vector U del problema de valor inical

Y el problema de valor inicial queda:

$$\frac{dU}{dt} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dfrac{-x}{(x^2+y^2)^{3/2}} \\ \dfrac{-y}{(x^2+y^2)^{3/2}} \end{pmatrix} = F(U,t)$$

Ecuación 4: Expresión del problema de valor inicial en función de x e y

Y expresándolo en función de las coordenadas de U:

$$\frac{dU}{dt} = \begin{pmatrix} U(3) \\ U(4) \\ \frac{-U(1)}{(U(1)^2 + U(2)^2)^{3/2}} \\ \frac{-U(2)}{(U(1)^2 + U(2)^2)^{3/2}} \end{pmatrix} = F(U,t)$$

Ecuación 5: Expresión del problema de valor inicial en función de las componentes de U

Que es la expresión de F que se utilizará en este ejercicio. Ahora el problema reside en cómo obtener el valor de U. De forma matemática, despejando de la ecuación del problema de Cauchy:

$$dU = F(U,t)dt \rightarrow \int_{U(t_1)}^{U(t_2)} dU = \int_{t_1}^{t_2} F(U,t)dt$$

Ecuación 6: Integración del problema de Cauchy

La integral de dU es directa, y la integral de F(U, t) se puede obtener, con mayor o menor dificultad a mano, conocida F(U). Sin embargo, un programa no tiene capacidad de realizar la integración directa de una función arbitraria. Conocida F, se podría obtener su integral a mano e introducirla en el ordenador, pero eso llevaría un tiempo importante y sólo sería válido para una F determinada, variar la función llevaría a tener que repetir todo el proceso. Por tanto, lo que se hace es realizar una aproximación de la integral de la función con esquemas numéricos, que discretizan la función a estudiar y calculan la integral de forma aproximada, a partir de los valores de las funciones para un intervalo de tiempo formado por n (siendo n un número elevado, generalmente) puntos equiespaciados entre ellos por un Δt pequeño (normalmente, es un valor de uno o varios órdenes de magnitud inferiores al segundo).

Si el esquema numérico es preciso, el error cometido es muy bajo. Se denomina orden de un esquema temporal al orden de magnitud del error cometido. Si un esquema numérico es de orden 1, su error local de truncamiento (error cometido en un paso del esquema), su error es del orden del Δt utilizado, si es de orden 2, de ese Δt elevado al cuadrado… Cuanto mayor sea el orden de un esquema, más preciso será respecto a la solución real.

2.Integración de órbitas con el método de Euler

En primer lugar, se deben definir una serie de valores y condiciones iniciales que son válidos para todos los apartados entre el 2 y el 5:

```
#Definición de condiciones iniciales
n = 200 # Número de particiones
deltat = 0.1 #Intervalo de tiempo entre iteraciones
U0 = array([1,0,0,1]) #Condiciones iniciales
matrizU = array(zeros([n,4])) #Matriz que contendrá en sus columnas los valores de las variables, cada fila es un paso de tiempo
F0 = array([0,1,-1,0])#Valor de F en el instante inicial
U = zeros(4)
x = zeros(n)
y = zeros(n)
```

Figura 1: Condiciones iniciales

n (número total de iteraciones) y deltat (intervalo de tiempo entre iteraciones) son los mismos valores utilizados en el código del "Milestone 1". U0 y F0 son las condiciones iniciales de U y F y deben definirse previamente, ya que son inputs de muchas funciones utilizadas. matrizU es una matriz en la que se incluirán todos los valores de U a lo largo de las distintas iteraciones.

La descripción del método de Euler ya se realizó en el informe "Milestone 1". La principal diferencia entre el código de este informe y el ya mencionado "Milestone 1" es que en este código se han utilizado funciones para optimizarlo y reducir la cantidad de líneas de código. Se ha realizado una función para el esquema de Euler de la siguiente manera:

```
18      #Función para integrar un paso de Euler
19      def Euler(U0, deltat, t, F0):
20          U = U0 + deltat*F0
21          return U
```

Figura 2: Función para integrar un paso del esquema de Euler

Se puede observar que sólo se realiza el paso de U^n a U^{n+1}, sin tener en cuenta en qué tiempo estamos o qué iteración sea la n y la n+1. Para tener en cuenta, estos parámetros, esta la función para integrar un problema de Cauchy con cualquier esquema:

```
57      #Función para integrar un problema de Cauchy arbitrario
58      def Cauchy_problem(Temporal_scheme, U0, n, deltat, F0):
59          t=0
60          for i in range (n):
61              t = t+deltat
62              U = Temporal_scheme(U0, deltat, t, F0)
63              matrizU[i,:] = U
64              U0 = U
65              F0 = Kepler(U,t) #Obtengo mi nuevo valor de F(U,t)
66              continue
67          return (matrizU)
```

Figura 3: Función para integrar un problema de Cauchy con un esquema arbitrario

Se puede observar que el esquema temporal utilizado es un input. Si hay una función para el esquema temporal introducido, da igual cuál sea, el programa podrá realizar la integración con ese esquema. Aquí el tiempo t y la partición en la que se esté sí que son importantes. La limitación de

esta función es que integra solamente como F las ecuaciones del movimiento de órbitas keplerianas, definida como:

```
47      #Ecuaciones de la órbita de Kepler
48      def Kepler(U,t):
49          d = ((U[0])**2+(U[1])**2)**1.5
50          x = U[2]
51          y = U[3]
52          vx = -U[0]/d
53          vy = -U[1]/d
54          F = array([x,y,vx,vy])
55          return F
```

Figura 4: Función para integrar un paso de órbitas Keplerianas

Aun así, es una limitación sencilla de solucionar, puesto que simplemente con definir una nueva función para igualarla a F0 ya se podría integrar cualquier función con cualquier método.

En cuanto al resto del programa más allá de las funciones, la integración por este método y su representación gráfica se realizan con unas pocas líneas de código:

```
79      # APARTADO 1: Euler
80      matrizU = Cauchy_problem(Euler, U0, n, deltat, F0)
81
82      #Las dos primeras columnas de la matriz son x e y de la órbita
83      x = matrizU[:,0]
84      y = matrizU[:,1]
85
86      #Representación gráfica
87      plt.plot(x,y)
88      plt.show()
89
```

Figura 5: Integración por el método de Euler y representación gráfica

Se ha decidido extraer la representación gráfica del código de las funciones porque en un futuro puede no interesar hacer la representación gráfica de los resultados, o variar esta según la F nueva introducida. En cuanto la integración en sí, se realiza en una sola línea que llama a funciones definidas previamente, reduciendo mucho la cantidad de líneas de código si se debe realizar la integración varias veces en el mismo código.

Finalmente, los resultados para este método son los siguientes:

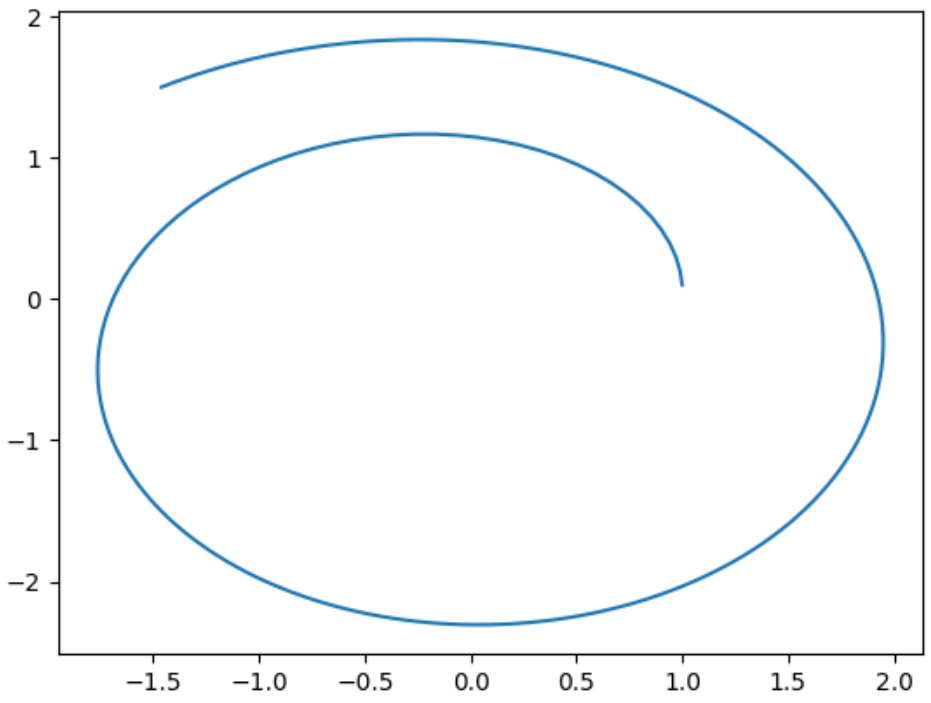

Figura 6: Representación gráfica por el método de Euler

Se observa una representación idéntica a la del método de Euler del informe "Milestone 1", por lo que se puede concluir que se ha realizado una programación de las funciones. Se sigue observando que se desvía rápidamente, siendo lógico al estar aplicando un método de orden 1.

3.Integración de órbitas con el método de Crank-Nicholson

El método de Crank-Nicholson aproxima la integración entre dos iteraciones de la siguiente forma:

$$U^{n+1} = U^n + 0.5 * \Delta t * (F(U^n, t_n) + F(U^{n+1}, t_{n+1}))$$

Ecuación 7: Expresión de U^{n+1} por el método de Crank-Nicholson

Es un método que se puede entender como una media entre el Euler y el Euler inverso (ver apartado 5), y es de orden 2, por lo que debería mostrarse más estable que el Euler, pero menos que el RK4. Se puede observar que es un método implícito, por lo que U no se puede despejar directamente. Para obtenerla, se realiza la siguiente función:

```
23      #Función para integrar un paso de Crank-Nicholson
24      def CN(U0, deltat, t, F0):
25          def Res(X):
26              return X-U0-deltat*0.5*(F0+ Kepler(X,t))
27          return newton(Res, U0)
```

Figura 7: Función para integrar un paso de Crank-Nicholson

También es una función que integra un único paso del esquema. Se define una función llamada Res sobre la que se aplicará un método de Newton-Raphson, ya que es un esquema implícito, y debe resolverse con un método que permita resolver un sistema de ecuaciones diferenciales no lineales.

La integración por este método y su representación gráfica se realiza de la siguiente forma:

```
90      #APARTADO 2: Crank-Nicholson
91      matrizU = Cauchy_problem(CN, U0, n, deltat, F0)
92
93      #Las dos primeras columnas de la matriz son x e y de la órbita
94      x = matrizU[:,0]
95      y = matrizU[:,1]
96
97      #Representación gráfica
98      plt.plot(x,y)
99      plt.show()
```

Figura 8: Integración por el método de Crank-Nicholson y representación gráfica

Evidentemente, se llama a la función del problema de Cauchy para realizar la integración del problema en las n integraciones. La representación gráfica del problema resulta:

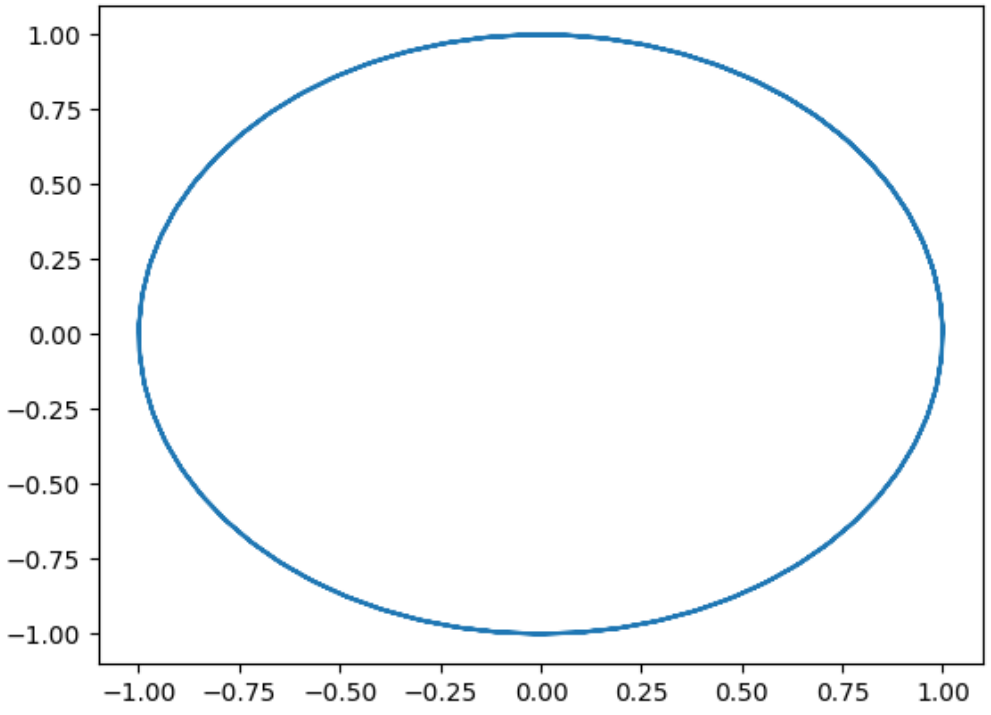

Figura 9: Representación gráfica por el método de Crank-Nicholson

Se observa que es una representación mucho más fiel a la realidad que el método de Euler, lógico al tratarse de un método de orden 2. Para el número de iteraciones elegido, el método apenas se diferencia del RK4, por lo que si el coste computacional es menor (habría que estudiar si la realización

de 4 pasos del RK4 tiene un coste computacional menor que el Newton-Raphson), para este número de iteraciones y Δt se podría estudiar utilizar este método, aunque sea menos preciso (u otro método de orden 2 en su defecto, puesto que ya se ha visto que el orden 1 se ha demostrado demasiado impreciso).

4.Integración de órbitas con el método RK4

La descripción del método RK4 ya se realizó en el informe "Milestone 1". La función para la integración de un paso de este método es la siguiente:

```
35      #Función para integrar un paso de RK4
36      def RK4(U0, deltat, t, F0):
37          k1 = F0 #Primer coeficiente
38          U2 = U0 +  k1*deltat/2
39          k2 = Kepler(U2,t) #Segundo coeficiente
40          U3 = U0 +  k2*deltat/2
41          k3 = Kepler(U3,t) #Tercer coeficiente
42          U4 = U0 +  k3*deltat
43          k4 = Kepler(U4,t) #Cuarto coeficiente
44          U = U0+ deltat*(k1+2*k2+2*k3+k4)/6 #Método RK4
45          return U
46
```

Figura 10: Función para integrar un paso de tiempo por el método RK4

Donde se destaca el hecho de que se realiza una llamada a la función Kepler tres veces, para todos los pasos después del primero. Para realizar la integración en el código y luego la representación gráfica, el código del programa es:

```
101     #APARTADO 3: RK4
102     matrizU = Cauchy_problem(RK4, U0, n, deltat, F0)
103
104     #Las dos primeras columnas de la matriz son x e y de la órbita
105     x = matrizU[:,0]
106     y = matrizU[:,1]
107
108     #Representación gráfica
109     plt.plot(x,y)
110     plt.show()
```

Figura 11: Integración por el método RK4 y representación gráfica

La integración por este método y su representación gráfica se realiza de la siguiente forma:

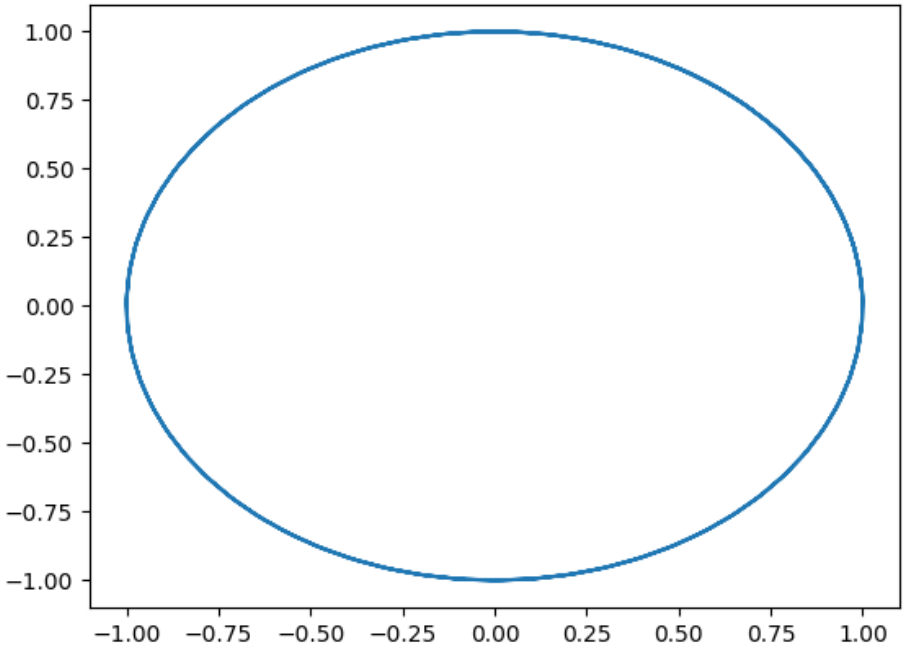

Figura 12: Representación gráfica por el método RK4

La representación es otra vez idéntica al código del informe "Milestone 1", muy similar a la representación real, como corresponde a un método de orden 4, más teniendo en cuenta que el Crank-Nicholson, de orden 2, ya es prácticamente idéntico a la gráfica real.

5.Integración de órbitas con el método Euler inverso

El método de Euler inverso aproxima la integración entre dos iteraciones de la siguiente forma:

$$U^{n+1} = U^n + F(U^{n+1}, t_{n+1}) * \Delta t$$

Ecuación 8: Expresión de U^{n+1} por el método de Euler inverso

Es un método implícito, que para obtener U^{n+1} debe resolver un sistema de ecuaciones diferenciales no lineal. Es de orden 1, por lo que los resultados deben de ser del orden del Euler. La función para la integración de este método es:

```
29      #Función para integrar un paso de Euler inverso
30      def Euler_inverso(U0, deltat, t, F0):
31          def Res2(Y):
32              return Y-U0-deltat*Kepler(Y,t)
33          return newton(Res2, U0,tol = 1e-1)
34
```

Figura 13: Función para integrar un paso de Euler inverso

Que se puede observar que sigue un esquema muy similar al del Crank-Nicholson. En el apartado 6.d se explicará la necesidad de incluir la tolerancia que se ha puesto en el Newton.

Para realizar la integración en el código y luego la representación gráfica, el código del programa es:

```
112     #APARTADO 4: Euler inverso
113     deltat=0.08
114     matrizU = Cauchy_problem(Euler_inverso, U0, n, deltat, F0)
115     #Las dos primeras columnas de la matriz son x e y de la órbita
116     x = matrizU[:,0]
117     y = matrizU[:,1]
118
119     #Representación gráfica
120     plt.plot(x,y)
121     plt.show()
```

Figura 14: Integración por el método Euler inverso y representación gráfica

Donde se ha tenido que introducir un Δt menor porque la solución divergía con el Δt original. Para este Δt la solución no llega a divergir y sigue siendo un valor cercano al del resto de métodos de integración. La representación gráfica del esquema es:

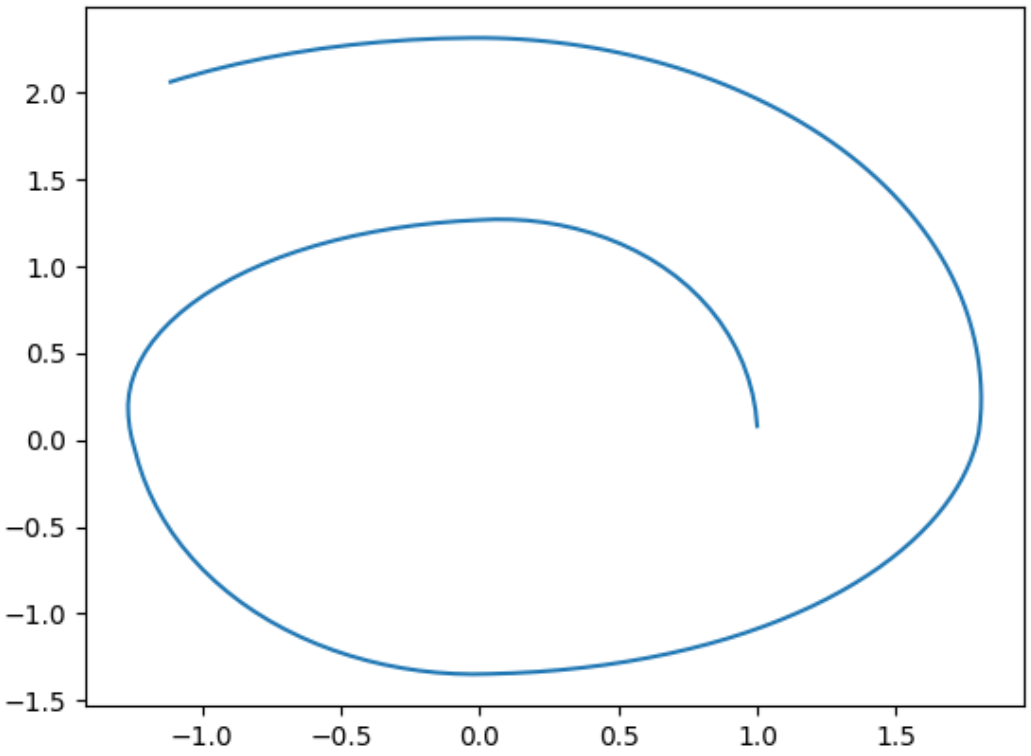

Figura 15: Representación gráfica por el método Euler inverso

Donde se ve que la gráfica es más divergente respecto a la real que en el Euler, pero dentro del mismo orden de magnitud, por lo que es una solución razonable. Habrá que comprobar si aumentando el Δt se puede mitigar este efecto.

6.Efecto de cambios en el Δt requerido

a. Método de Euler

Se van a analizar dos situaciones para todos los métodos. En primer lugar, el aumento del Δt al doble del original (de 0,1 a 0,2 segundos), con la consiguiente reducción de número de iteraciones (de 200 a 100) para acabar en el mismo tiempo final que en la situación inicial. La nueva definición de condiciones iniciales es:

```
123    #APARTADO 5: Aumentar y disminuir delta t
124    #Definición de condiciones iniciales
125    n = 100 # Número de particiones
126    deltat = 0.2 #Intervalo de tiempo entre iteraciones
127    U0 = array([1,0,0,1]) #Condiciones iniciales
128    matrizU = array(zeros([n,4])) #Matriz que contendrá en sus columnas los valores de las variables, cada fila es un paso de tiempo
129    F0 = array([0,1,-1,0])#Valor de F en el instante inicial
130    U = zeros(4)
131    x = zeros(n)
132    y = zeros(n)
```

Figura 16: Condiciones iniciales para el aumento de Δt

Se redefinen las matrices y vectores que tienen dimensión n para adaptarlos al cambio realizado en el valor de n. En esta situación, los esquemas deberían empeorar la precisión. Habrá que prestar atención al efecto sobre los métodos de Euler, ya que su precisión ya no era buena para el Δt elegido, y ahora va a disminuir todavía más. Habrá que observar si en esta situación el método de Crank-Nicholson empeora y muestra diferencias respecto al RK4.

La segunda situación es la contraria, se disminuye el Δt a la mitad del original (de 0,1 a 0,05 segundos), con el consiguiente aumento de número de iteraciones (de 200 a 400) para acabar en el mismo tiempo final que las otras dos situaciones. La nueva definición de constantes iniciales es:

```
147    n = 400 # Número de particiones
148    deltat = 0.05 #Intervalo de tiempo entre iteraciones
149    U0 = array([1,0,0,1]) #Condiciones iniciales
150    matrizU = array(zeros([n,4])) #Matriz que contendrá en sus columnas los valores de las variables, cada fila es un paso de tiempo
151    F0 = array([0,1,-1,0])#Valor de F en el instante inicial
152    U = zeros(4)
153    x = zeros(n)
154    y = zeros(n)
155
```

Figura 17: Condiciones iniciales para la disminución de Δt

En esta situación, los esquemas deberían mejorar su precisión. No se deberían observar efectos apreciables en los esquemas de Crank-Nicholson y RK4 debido a su alta precisión respecto a la órbita real, pero sí que debería verse una mejora considerable en los métodos de Euler.

Para hacer una representación más rápida de los 4 esquemas para las dos situaciones, el código que se realiza es el siguiente:

```
134  Esquemas = [Euler, CN, RK4, Euler_inverso]
135
136  for i in Esquemas:
137      matrizU = Cauchy_problem(i, U0, n, deltat, F0)
138      #Las dos primeras columnas de la matriz son x e y de la órbita
139      x = matrizU[:,0]
140      y = matrizU[:,1]
141
142      #Representación gráfica
143      plt.plot(x,y)
144      plt.show()
145      continue
```

Figura 18: Integración y representación gráfica de los 4 esquemas para diferentes Δt

Este es el código para la integración con Δt = 0,2 segundos, pero para Δt = 0,05 segundos es un código idéntico. Se define una lista con los 4 esquemas utilizados y en un bucle se va iterando cada esquema y haciendo su representación gráfica.

Para el método de Euler, las representaciones obtenidas son las siguientes:

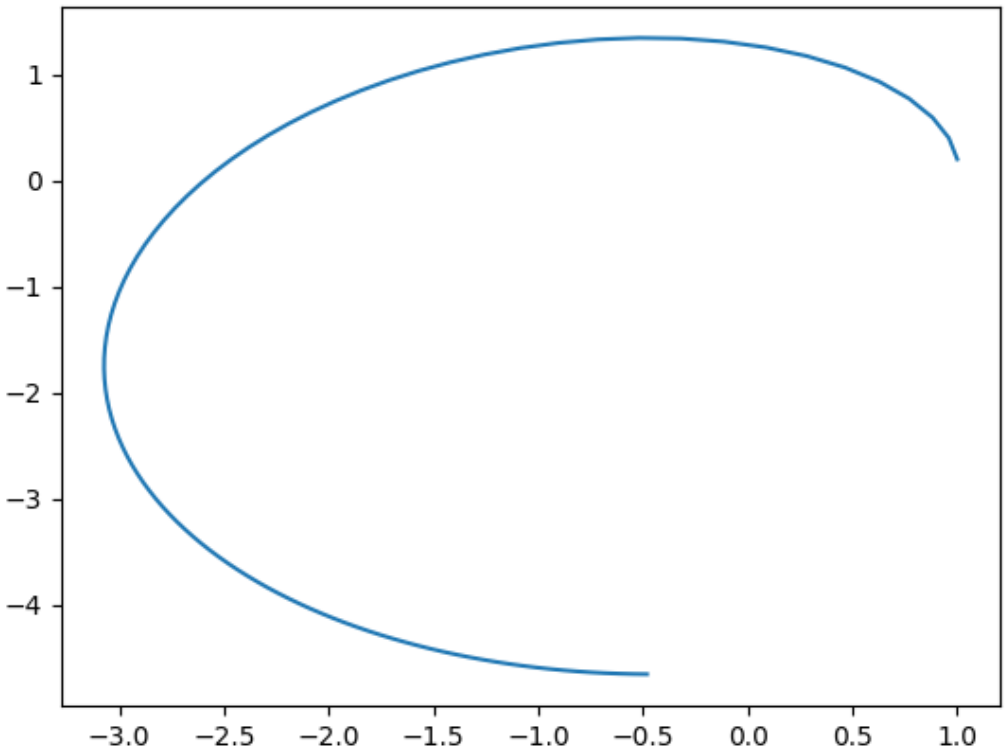

Figura 19: Representación gráfica del método de Euler para Δt = 0,2 segundos

Se observa una desviación mucho más rápida con este Δt, llegando al orden de -4 como valor máximo en la primera vuelta. Evidentemente, esta desviación hace impracticable intentar integrar esta órbita

con este método y este Δt.

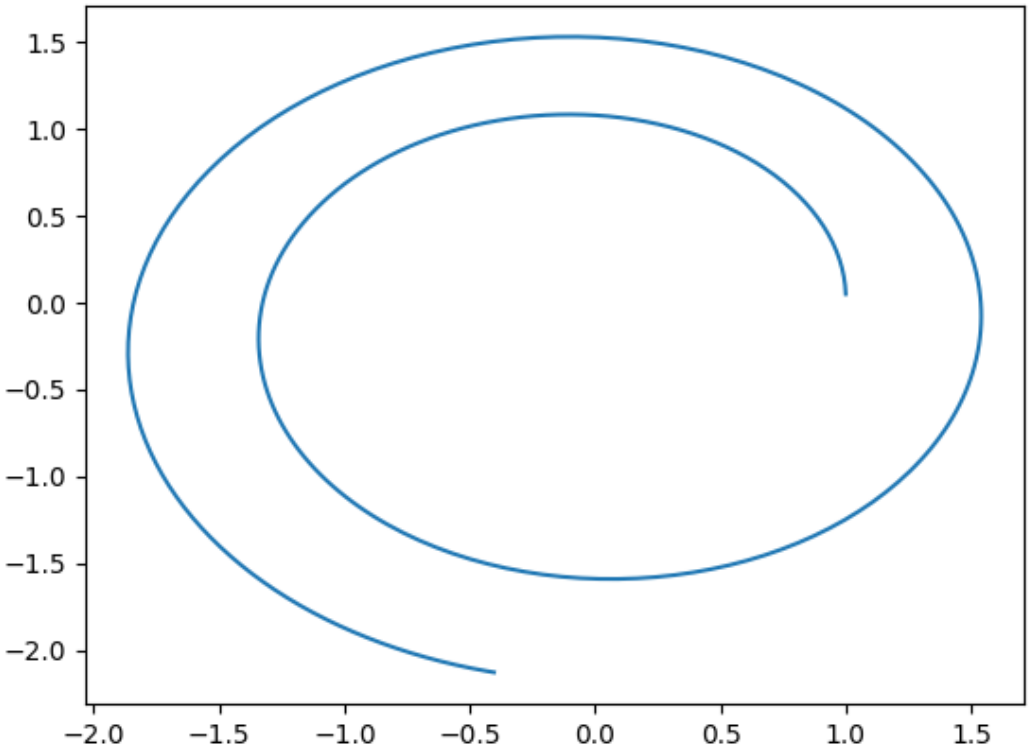

Figura 20: Representación gráfica del método de Euler para Δt = 0,05 segundos

La desviación sigue siendo notable, pero se ha reducido en gran medida en comparación a la inicial y, sobre todo, a la gráfica anterior. Aun así, no es recomendable utilizar este método para integrar la órbita a menos que se realizará una integración con un Δt de un orden mucho menor que los utilizados.

b. Método de Crank-Nicholson

Para el método de Crank-Nicholson, las representaciones obtenidas son las siguientes:

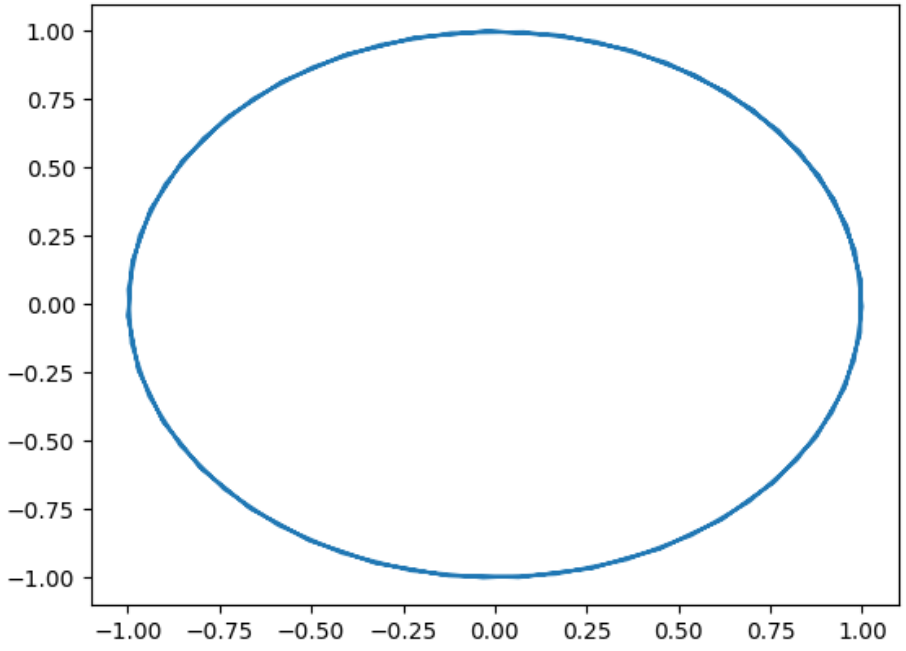

Figura 21: Representación gráfica del método de Crank-Nicholson para Δt = 0,2 segundos

La representación sigue siendo buena, bastante fiel a la realidad, pero se observan unas ligeras desviaciones en la órbita que no se observaban en la representación para el Δt original.

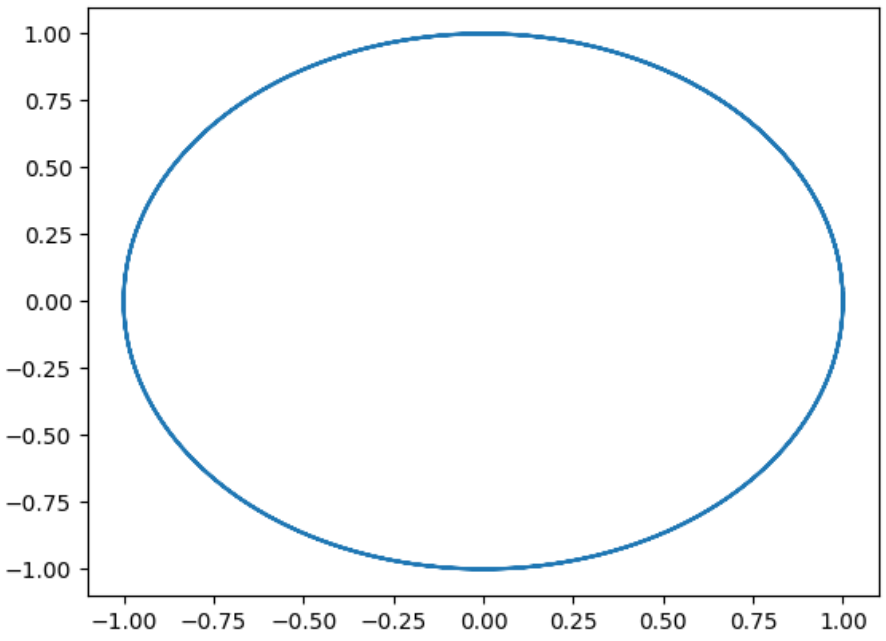

Figura 22: Representación gráfica del método de Crank-Nicholson para Δt = 0,05 segundos

Si para el Δt que se analizó la órbita ya era buena, disminuyéndolo no se van a notar variaciones apreciables en la gráfica.

c. Método RK4

Para el método RK4, las representaciones obtenidas son las siguientes:

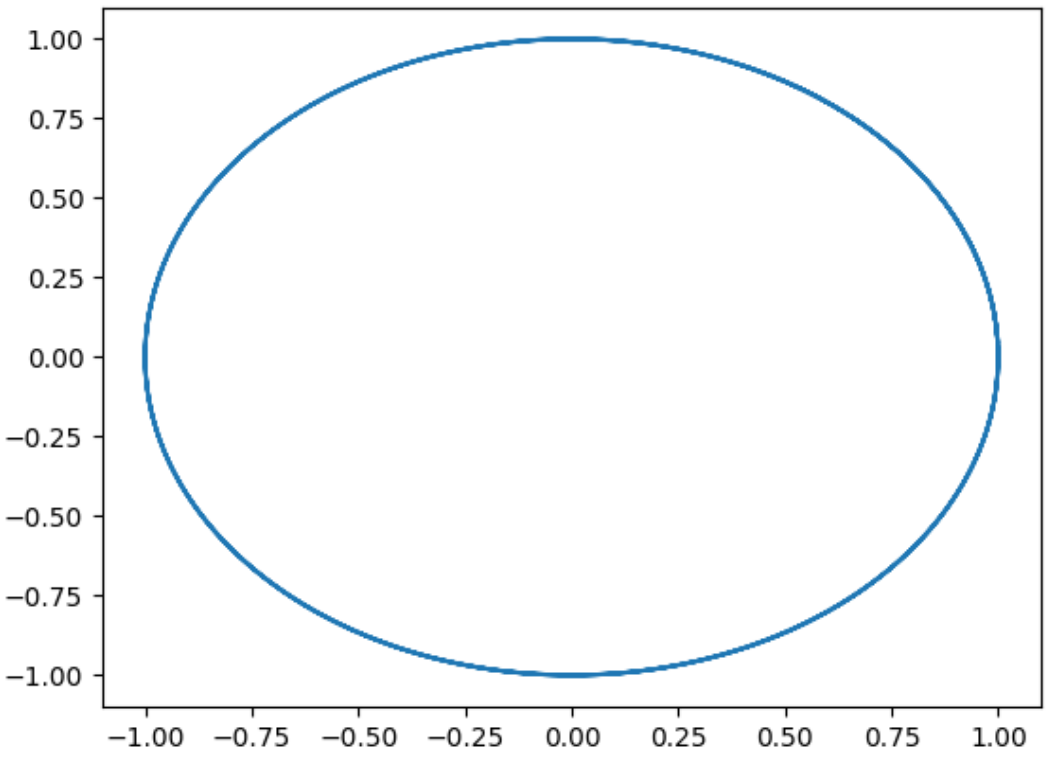

Figura 23: Representación gráfica del método RK4 para Δt = 0,2 segundos

El método RK4, al ser de orden 4, no sufre variaciones apreciables en la órbita dibujada. Para que empezara a sufrir cambios importantes, debería aumentarse el Δt en mucha mayor proporción. Si la órbita ya está bien representada de por sí con Δt = 0,1 segundos, es lógico pensar que disminuyendo este Δt tampoco se van a observar cambios apreciables. Esto se comprueba en la siguiente gráfica:

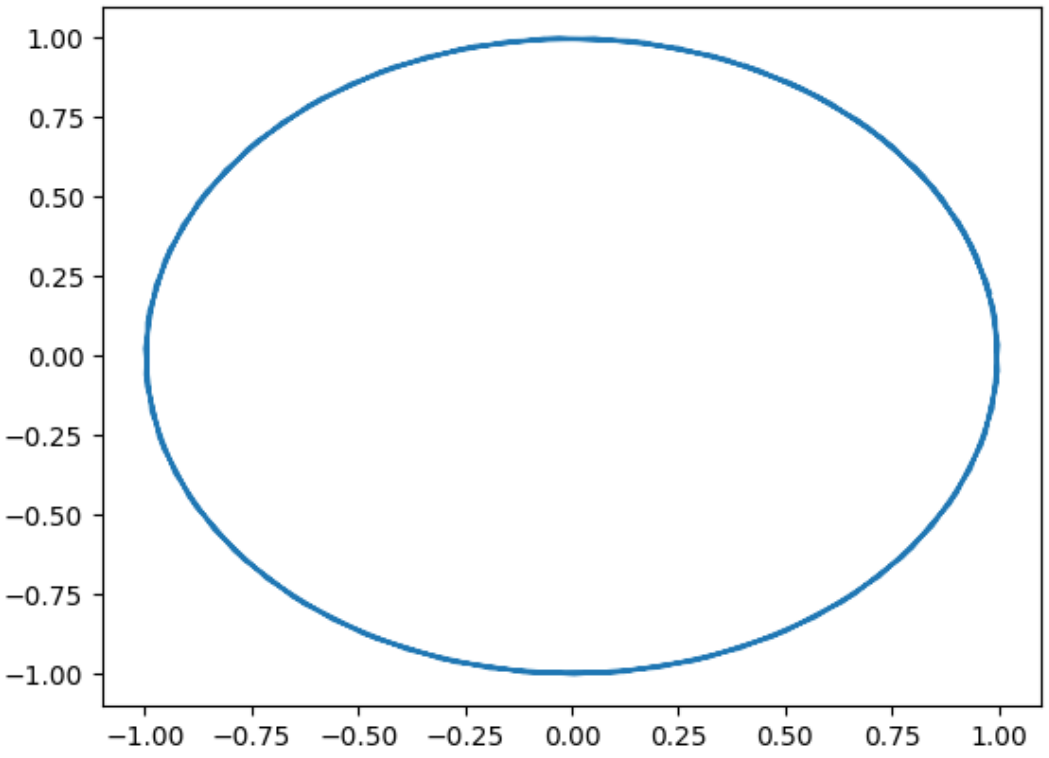

Figura 24: Representación gráfica del método RK4 para Δt = 0,05 segundos

d. Método Euler inverso

Para el método de Euler inverso, las representaciones obtenidas son las siguientes:

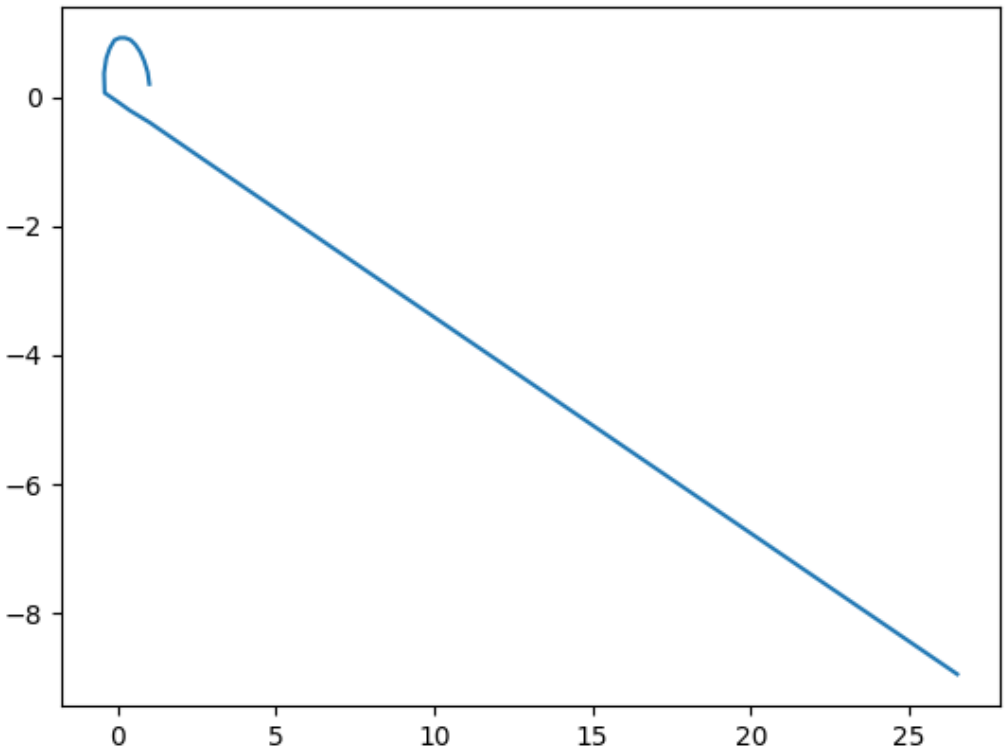

Figura 25: Representación gráfica del método Euler inverso para Δt = 0,2 segundos

Como ya se dijo en el apartado 5, a partir de un Δt algo menor de 0,1 segundos, el método diverge. Esto se comprueba en esta gráfica, donde lo que se constata es que para Δt altos, este método no sirve para integrar las ecuaciones de las órbitas keplerianas.

En cuanto a la representación para un Δt menor:

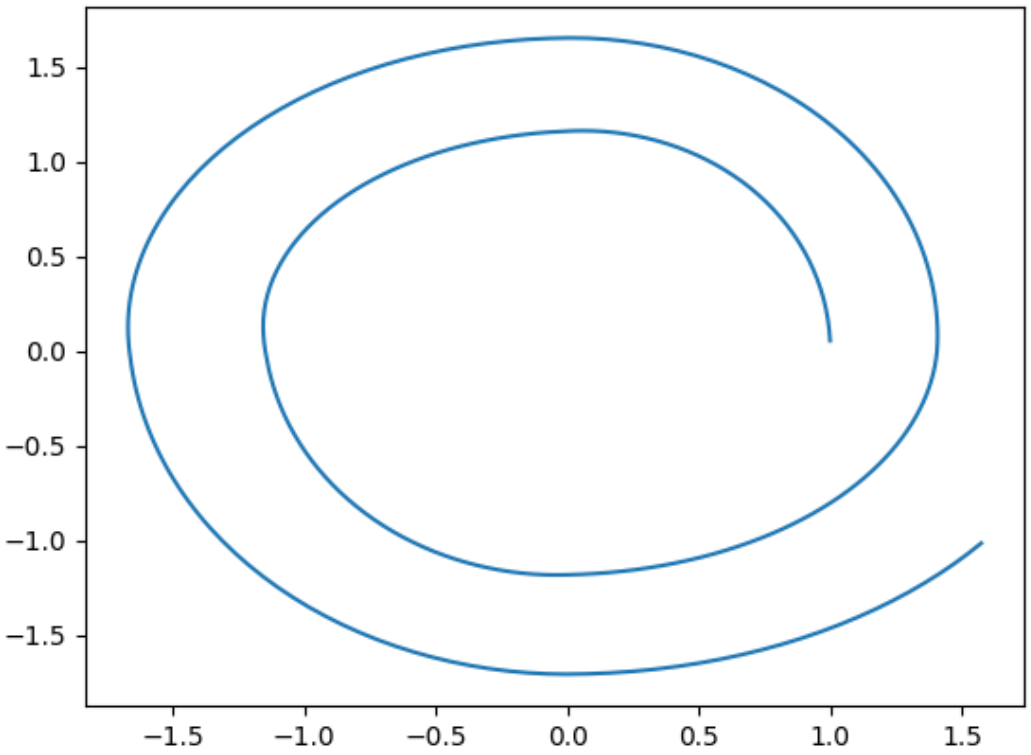

Figura 26: Representación gráfica del método Euler inverso para Δt = 0,05 segundos

Para este Δt, la representación obtenida es bastante mejor que la anterior. Se obtiene una gráfica similar en cuanto a desviación al método de Euler, comprobando que, en efecto, son métodos del mismo orden. Aun así, como en el método de Euler, no es recomendable integrar estas ecuaciones utilizando el Euler inverso, ya que la precisión obtenida es muy baja, incluso para Δt bajos.