FACULTAD DE INGENIERIA
Universidad de Buenos Aires

“ANÁLISIS NUMÉRICO I”

“MÉTODOS MATEMÁTICOS Y NUMÉRICOS”

<75.12> <95.04> <95.13>

DATOS DEL TRABAJO PRÁCTICO

2	2022	Resolución de la Ley de Newton en ecuaciones diferenciales
	AÑO	Misión en búsqueda del objeto interestelar Oumuamua
	2	
TP NRO	CUAT	TEMA

INTEGRANTES DEL GRUPO

	DE LA PLATA FACUNDO	100558
	APELLIDO Y NOMBRE	PADRÓN
GRUPO	APELLIDO Y NOMBRE	PADRÓN

Objetivos

- Discretizar e implementar los métodos de Euler y Runge Kutta 2
- Evaluar el error de los mismos
- Aplicar los métodos en distintos modelos y resolución de problemas

Desarrollo

$$\frac{\mathrm{d}vx}{\mathrm{dt}} = \mathrm{G}\frac{m_s}{{d_s}^2}\cos\propto_{fs} \quad (1)$$

$$\frac{\mathrm{d}vy}{\mathrm{dt}} = \mathrm{G}\frac{m_s}{{d_s}^2}\operatorname{sen}\propto_{fs} \quad (2)$$

$$\frac{\mathrm{d}x}{\mathrm{dt}} = v_x \ (3)$$

$$\frac{\mathrm{d}y}{\mathrm{dt}} = v_y \ (4)$$

Discretización

$$t \longrightarrow t^n$$

$$f(t, v_x) \longrightarrow f(t^n, {v_x}^n) = \mathrm{G}\frac{m_s}{{d_s}^2}\cos\propto_{fs}$$

Euler explícito

$$u^{n+1} = u^n + hf(\mathrm{t}^n, \mathrm{u}^n)$$

$${v_x}^{n+1} = {v_x}^n + h\,\mathrm{G}\frac{m_s}{{d_s}^2}\cos\left(\operatorname{atan}\left(\frac{y_s - y^n}{x_s - x^n}\right)\right) \quad (1)$$

$${v_y}^{n+1} = {v_y}^n + h\,\mathrm{G}\frac{m_s}{{d_s}^2}\operatorname{sen}\left(\operatorname{atan}\left(\frac{y_s - y^n}{x_s - x^n}\right)\right) \quad (2)$$

$$x^{n+1} = \mathrm{x}^n + h{v_x}^n \qquad (3)$$

$$y^{n+1} = \mathrm{y}^n + h{v_y}^n \qquad (4)$$

Runge Kutta 2

$$u^{n+1/2} = u^n + \frac{h}{2} f(\mathrm{t^n}, \mathrm{u^n})$$

$$u^{n+1} = u^n + h f(\mathrm{t^{n+1/2}}, \mathrm{u^{n+1/2}})$$

Paso predictor:

$$v_x{}^{n+1/2} = v_x{}^n + \frac{h}{2}\ \mathrm{G}\frac{m_s}{d_s{}^2}\cos\left(\operatorname{atan}\left(\frac{y_s - y^n}{x_s - x^n}\right)\right)$$

$$v_y{}^{n+1/2} = v_y{}^n + \frac{h}{2}\ \mathrm{G}\frac{m_s}{d_s{}^2}\operatorname{sen}\left(\operatorname{atan}\left(\frac{y_s - y^n}{x_s - x^n}\right)\right)$$

$$x^{n+1/2} = \mathrm{x}^n + \frac{h}{2}\ v_x{}^n$$

$$y^{n+1/2} = \mathrm{y}^n + \frac{h}{2}\ v_y{}^n$$

Paso corrector:

$$v_x{}^{n+1} = v_x{}^n + h\ \mathrm{G}\frac{m_s}{d_s{}^2}\cos\left(\operatorname{atan}\left(\frac{y_s - y^n - \frac{h}{2}v_y{}^n}{x_s - x^n - \frac{h}{2}v_x{}^n}\right)\right) \quad (1)$$

$$v_y{}^{n+1} = v_y{}^n + h\ \mathrm{G}\frac{m_s}{d_s{}^2}\operatorname{sen}\left(\operatorname{atan}\left(\frac{y_s - y^n - \frac{h}{2}v_y{}^n}{x_s - x^n - \frac{h}{2}v_x{}^n}\right)\right) \quad (2)$$

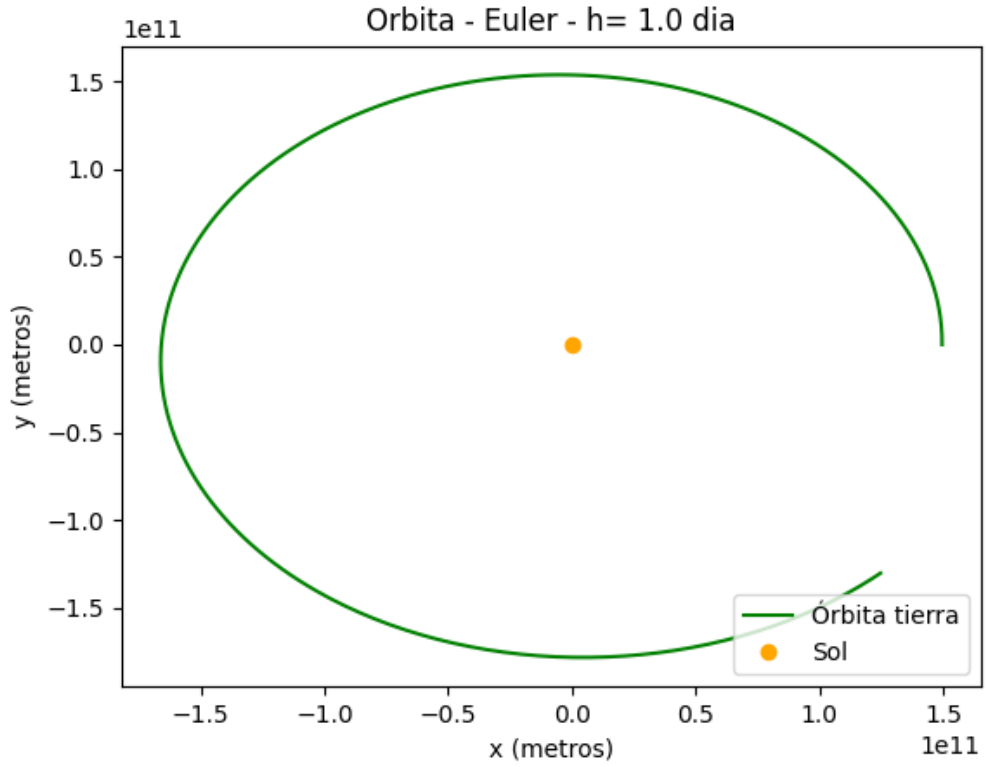
$$x^{n+1} = \mathrm{x}^n + h\left[v_x{}^n + \frac{h}{2}\ \mathrm{G}\frac{m_s}{d_s{}^2}\cos\left(\operatorname{atan}\left(\frac{y_s - y^n}{x_s - x^n}\right)\right)\right] \quad (3)$$

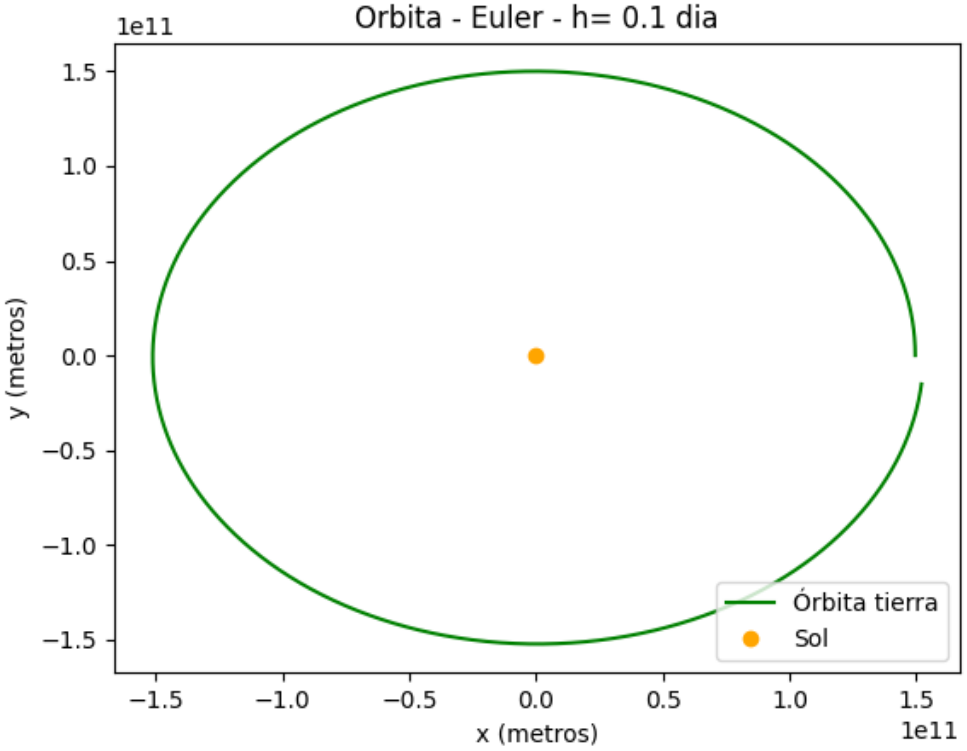
$$y^{n+1} = \mathrm{y}^n + h\left[v_y{}^n + \frac{h}{2}\ \mathrm{G}\frac{m_s}{d_s{}^2}\operatorname{sen}\left(\operatorname{atan}\left(\frac{y_s - y^n}{x_s - x^n}\right)\right)\right] \quad (4)$$

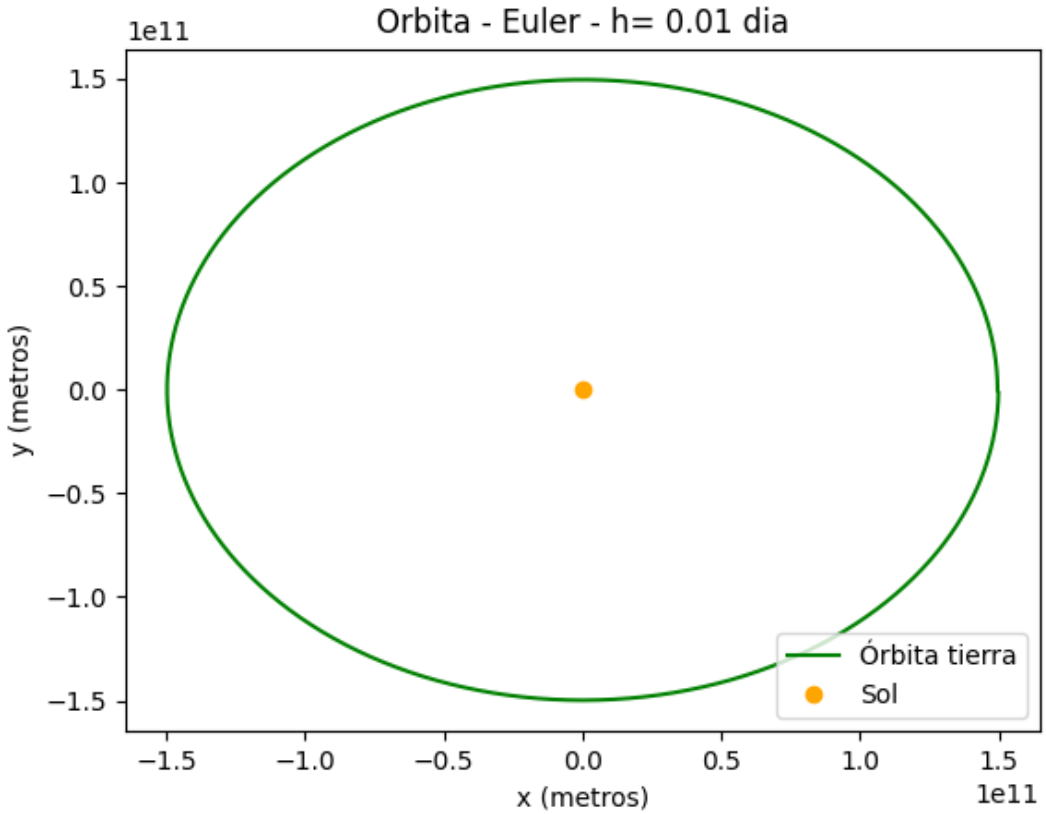
1)

a) Ver implementación del código en ANEXO I

b) <u>Euler</u>

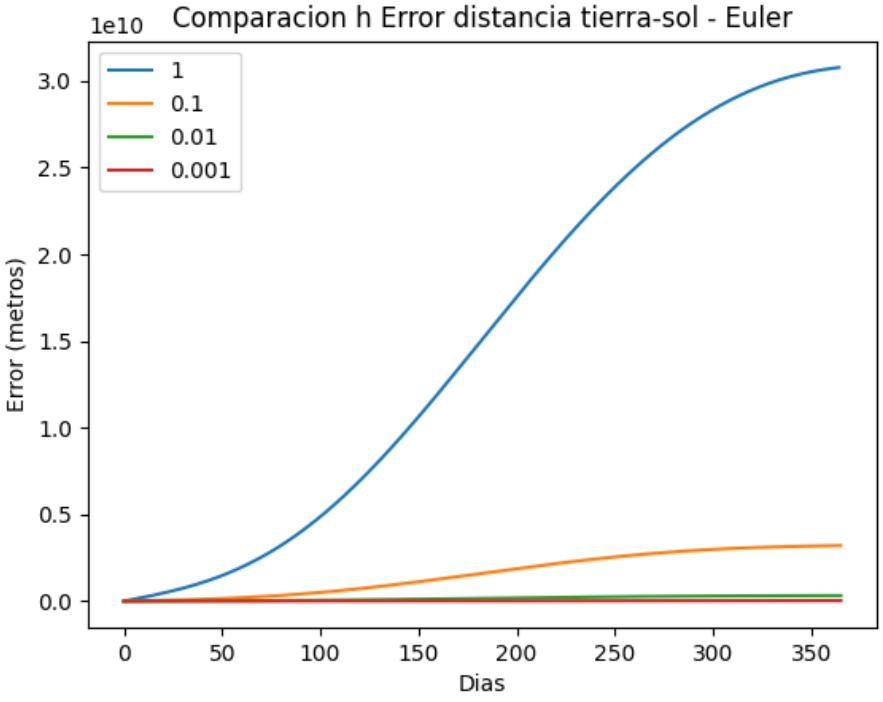






Se tomó h en función del tiempo, donde h es el paso (en días) en que se va a calcular en el método. A menor h, mayor cantidad de pasos.

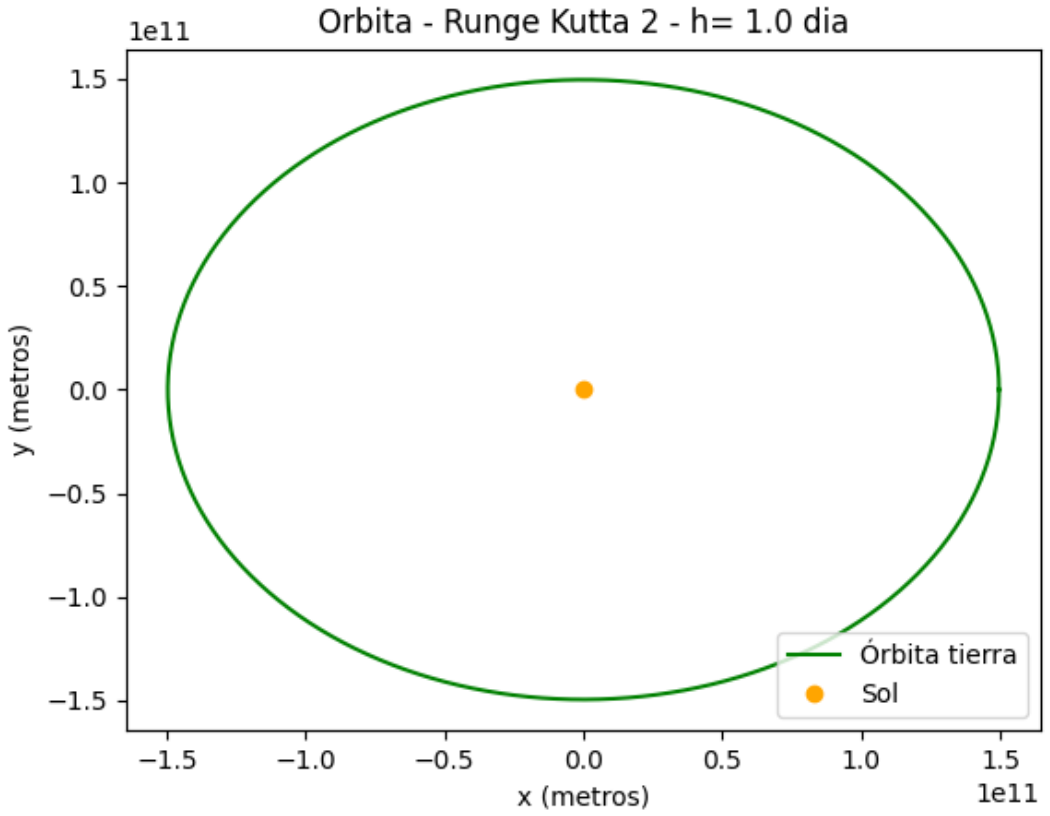
Se observa que existe un error en el cálculo de la órbita, ya que la elipse queda “abierta” al calcularla con un paso h grande.

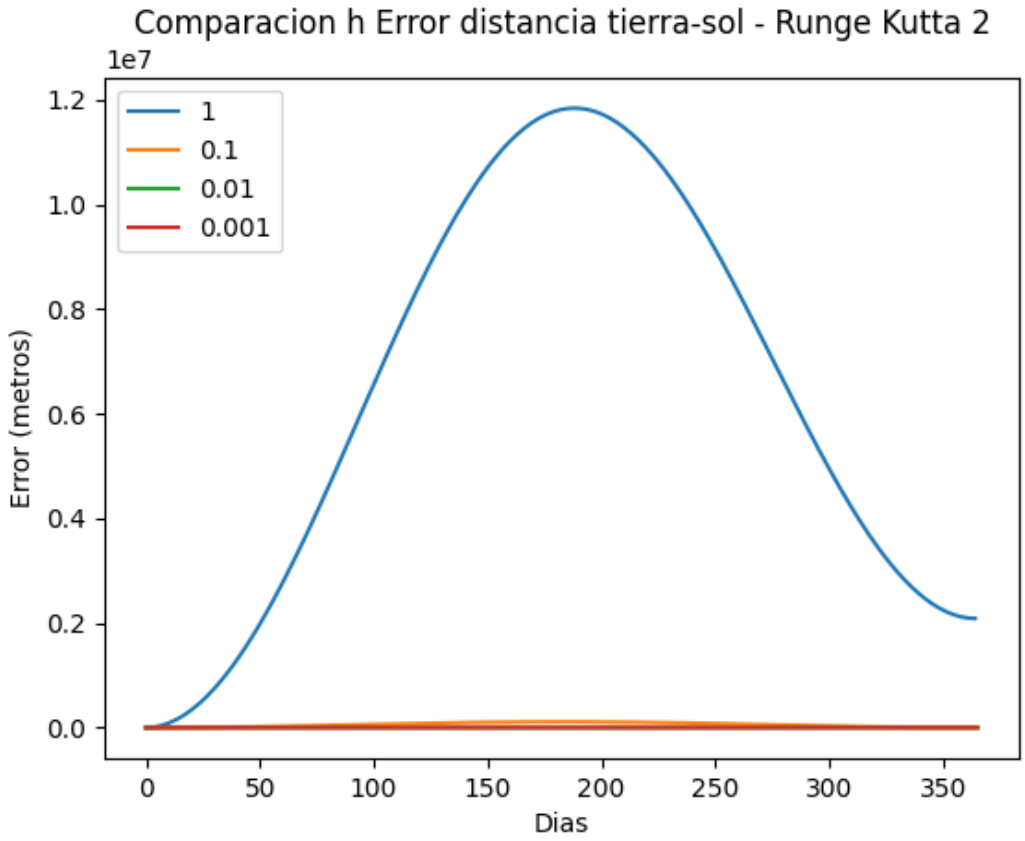


Al comparar el error en la medición de la distancia al sol, se observa que el error se acumula a través del tiempo.

Así mismo, se ve que al reducir h en 1 orden de magnitud, el error se reduce linealmente también en 1 orden de magnitud.

Runge Kutta 2



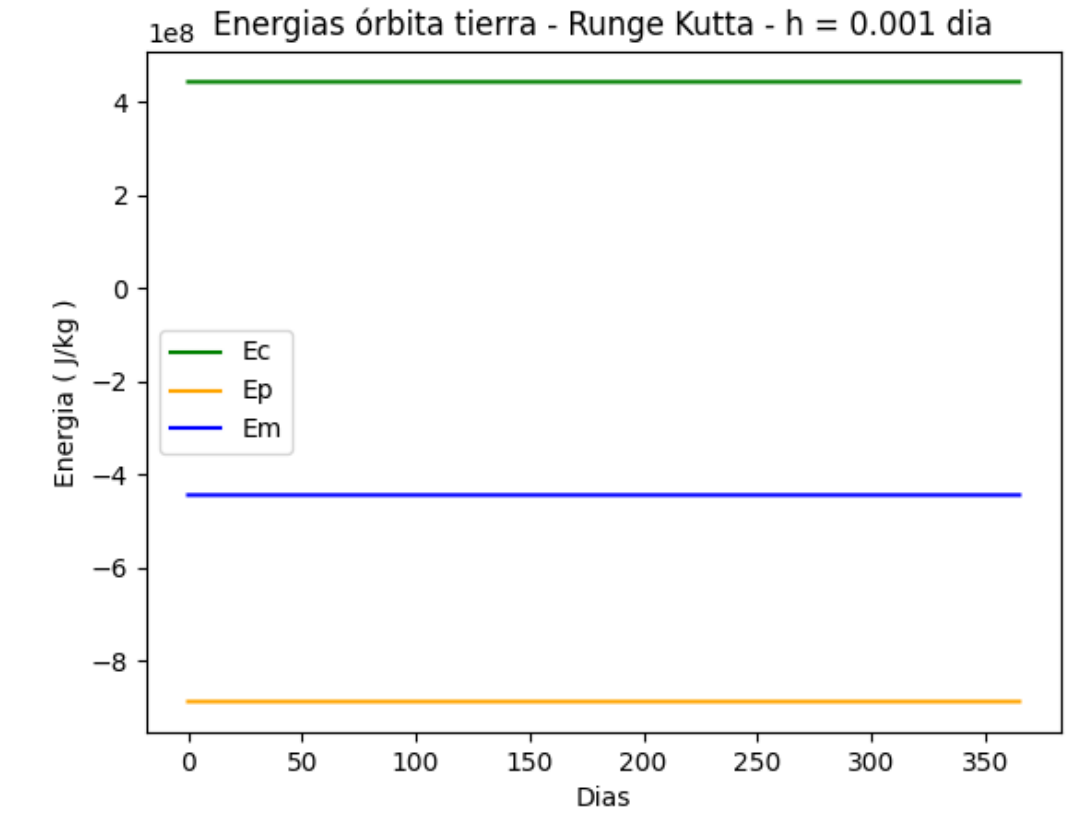


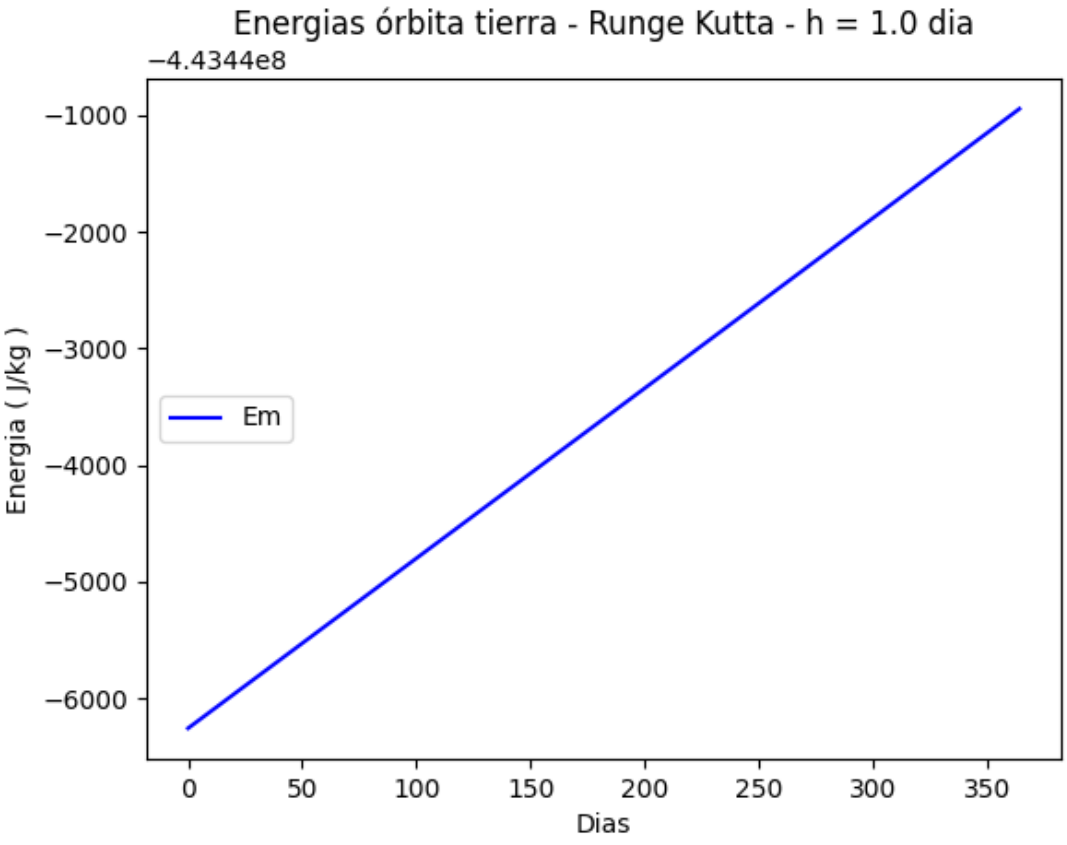
En el caso de Runge Kutta, se aprecia que el error resultante es menor.
Así mismo, al comparar los distintos valores de h, se aprecia que al reducir h en 1 orden de magnitud, el error se reduce en 2 órdenes de magnitud.

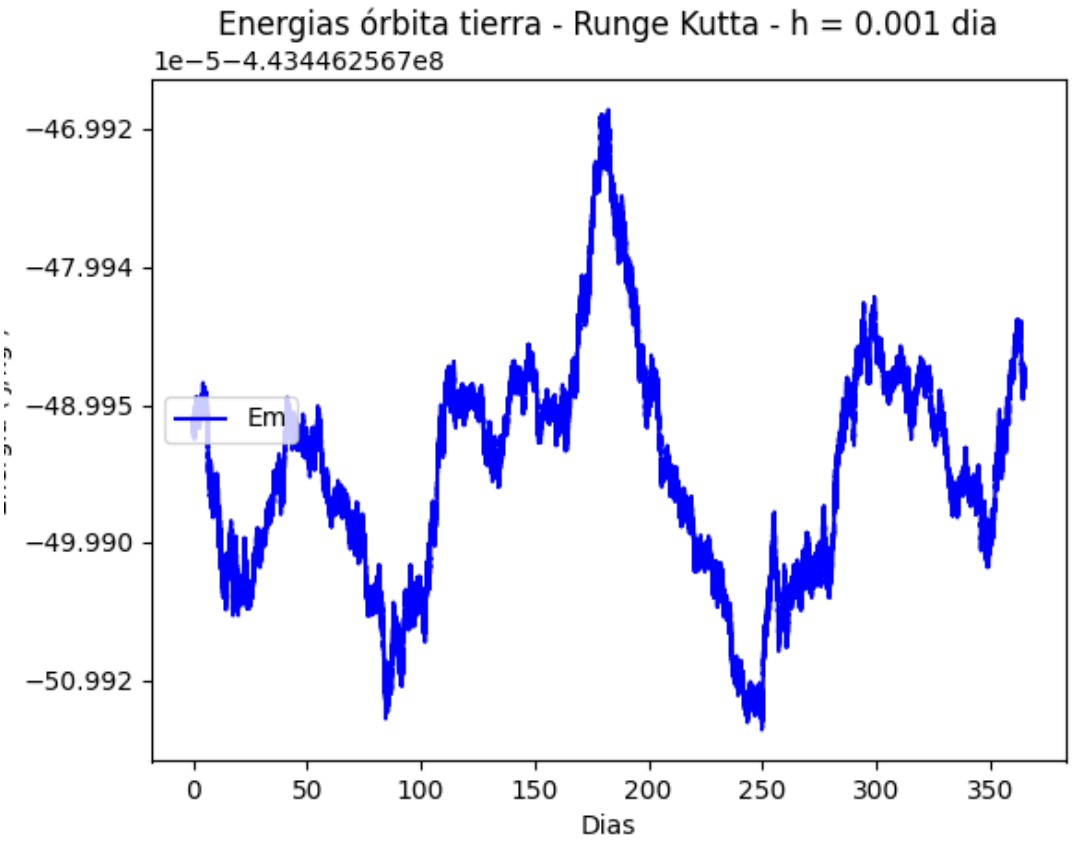
En este caso, al calcular con h=0.01 se obtiene un error de 2 metros al completar la órbita, valor que es nulo en comparación a la distancia entre la tierra y el sol.

Se observa en el gráfico también un pico donde el error aumenta, lo cual se explica con la trayectoria elíptica de la tierra, ya que la comparación se realizó respecto a una distancia al sol fija.

c)


Energias órbita tierra - Runge Kutta - h = 0.001 dia
1e8
4
2
0
−2
−4
−6
−8
Energia (J/kg)
Ec
Ep
Em
0
50
100
150
200
250
300
350
Dias


Energias órbita tierra - Runge Kutta - h = 1.0 dia
−4.4344e8
−1000
−2000
−3000
−4000
−5000
−6000
Energia (J/kg)
Em
0
50
100
150
200
250
300
350
Dias

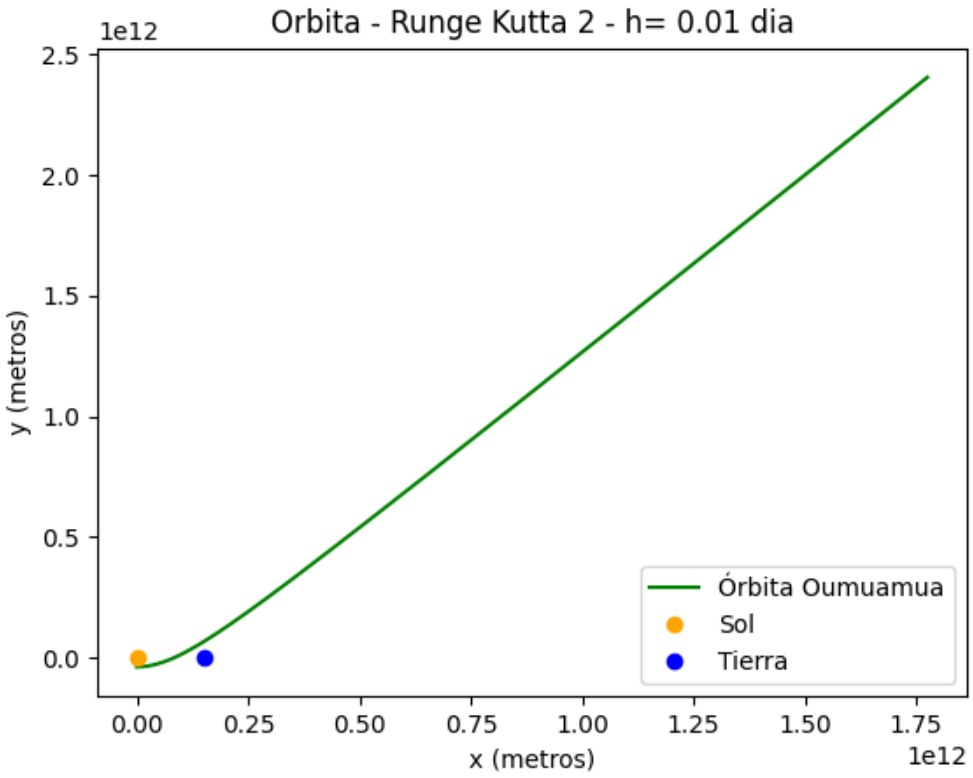


Al calcular la energía mecánica, en principio se observa que la misma no permanece constante. Sin embargo, la variación que tiene no es significante respecto a la magnitud de esta. Al calcularla con una precisión mayor, se aprecia que la misma oscila alrededor de cierto valor.

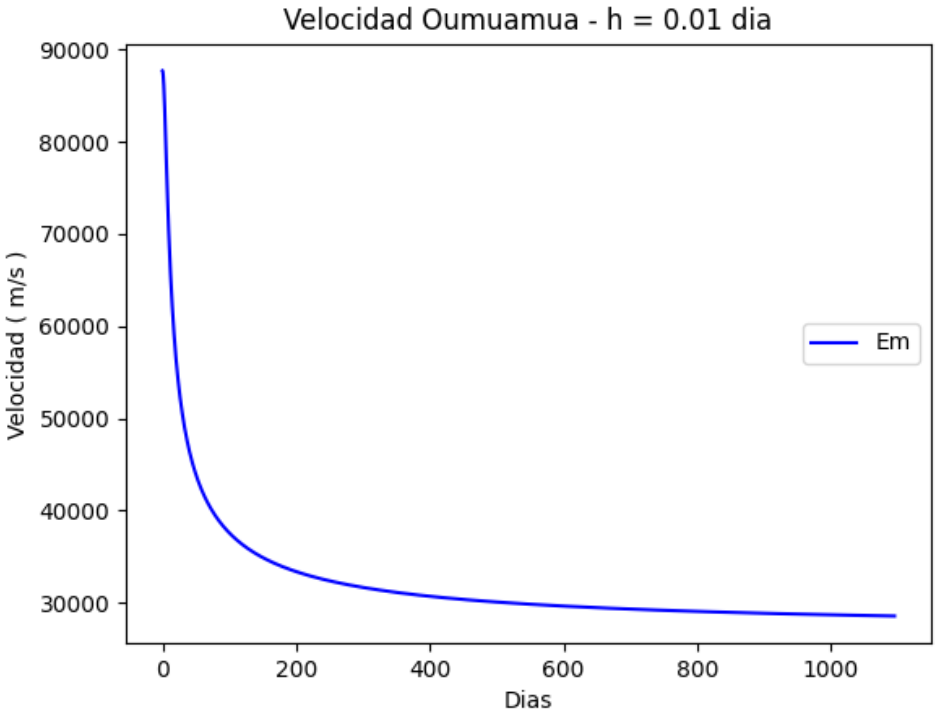
Comparando la Em con la Ep y Ec, se observa en el gráfico que la misma queda representada por una línea horizontal.
Por lo tanto, se puede afirmar que la energía se conserva.

2) Trayectoria de Oumuamua

a)



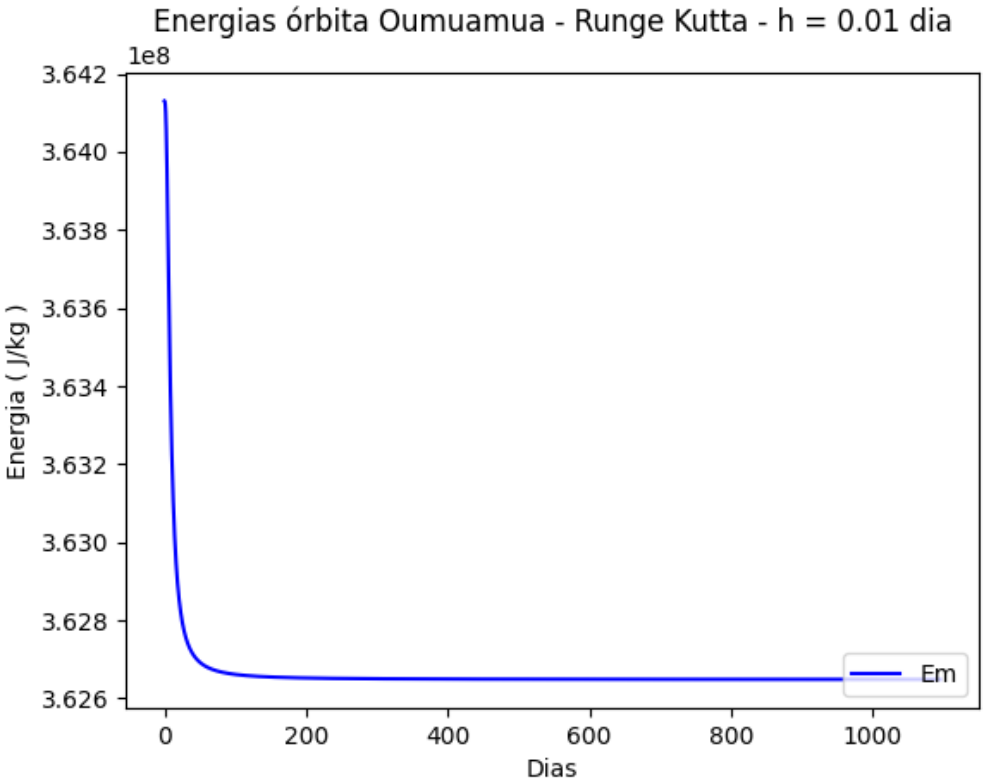
Trayectoria de Oumuamua a lo largo de 3 años

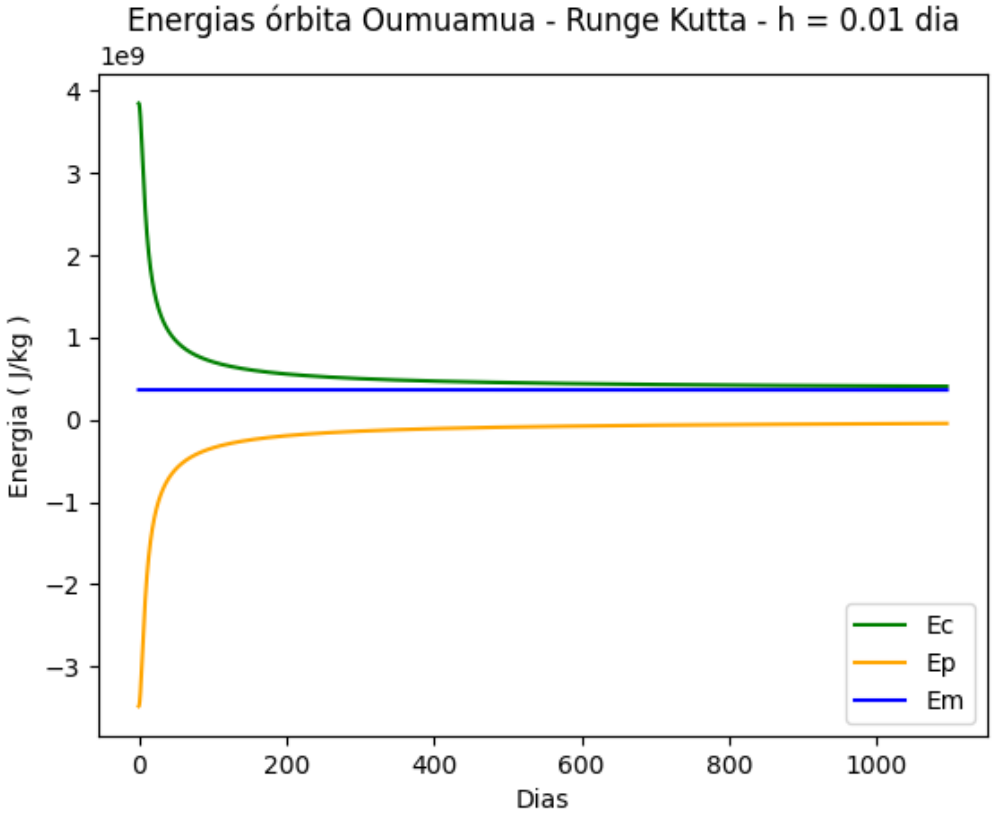


En este tiempo calculado, la velocidad decrece cada vez con menor intensidad, llegando

a una magnitud de 28530 m/s, la cual consideraremos $v_{\infty U}$

b)

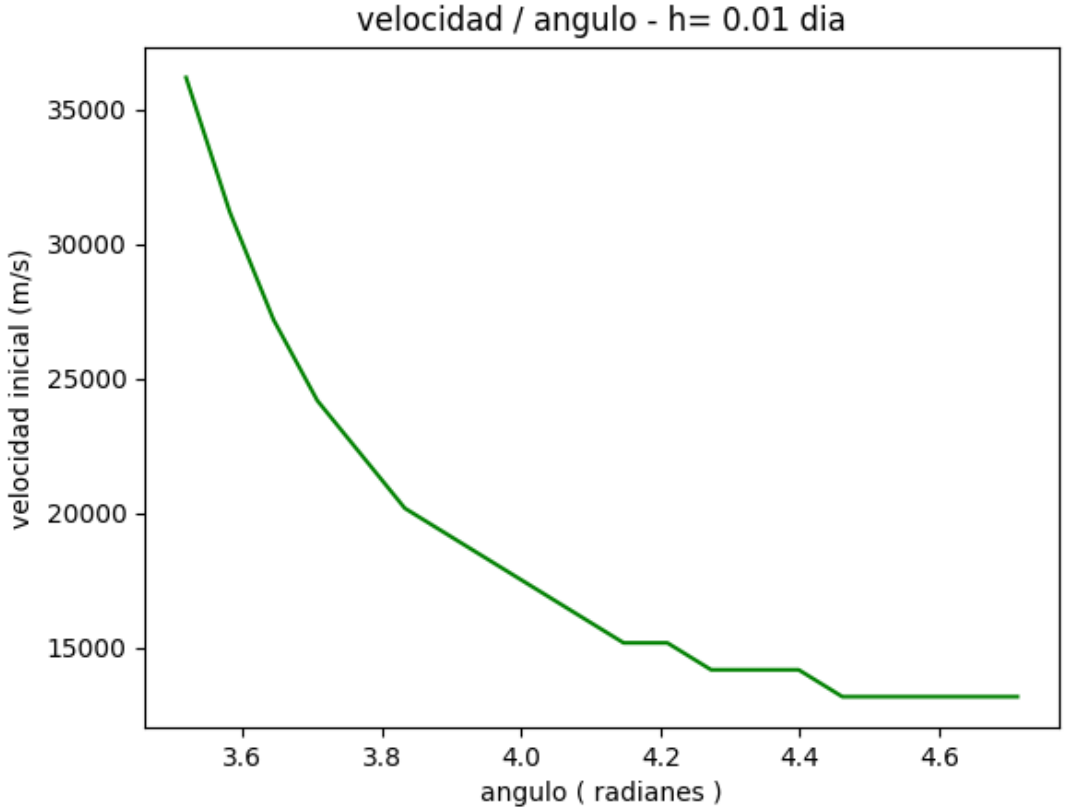




En este caso se puede observar cómo inicialmente la energía mecánica no se conserva en los primeros días, disminuyendo significativamente mientras el sol realiza una fuerte incidencia. Luego la misma se nivela y permanece sin cambios significativos.

3) Misión de búsqueda

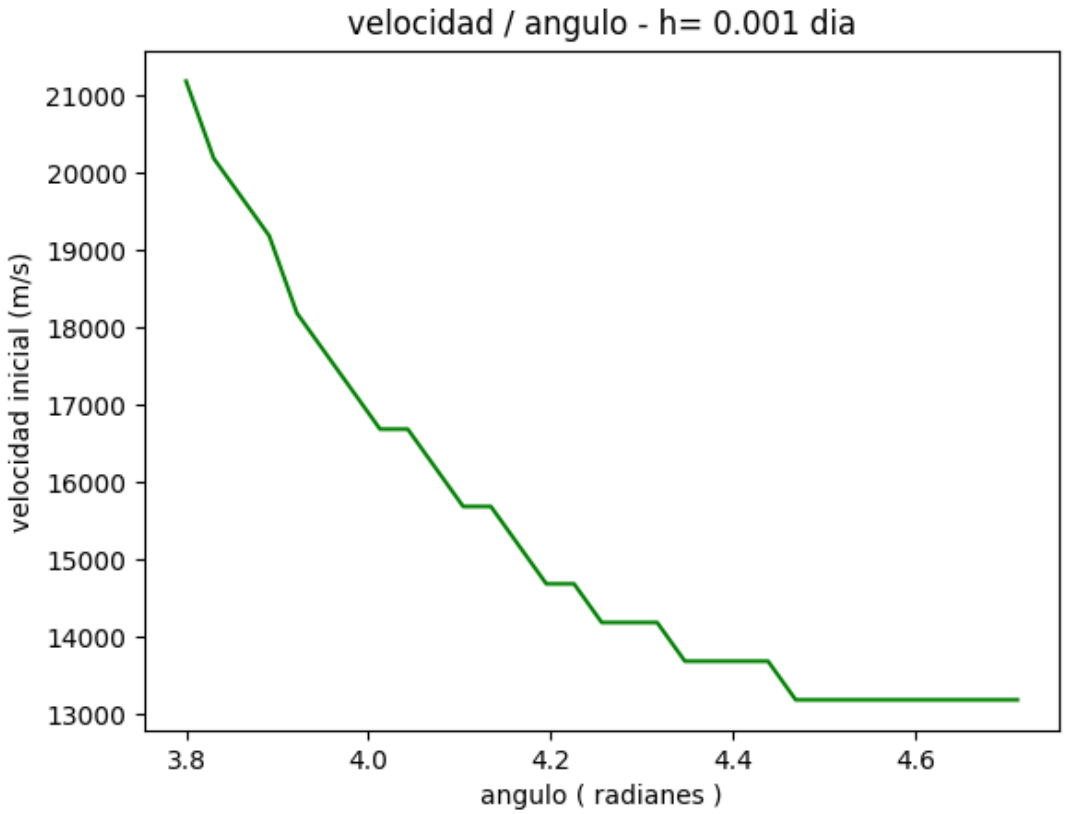
Para evaluar las condiciones iniciales se modeló la trayectoria de la sonda para distintos ángulos y velocidades, variando el primero entre 180° y 270° y el segundo entre 11,18 km/s y 40 km/s.
Para cada par de datos modelado, se evaluó al llegar al perihelio si la sonda se encontraba a una distancia que le permita soportar la carga solar.
El resultado fue el siguiente:



El presente gráfico muestra la velocidad y ángulo mínimos que se requiere para que la misión sea satisfactoria, por lo que todo valor por encima de la curva es válido.

Los valores sobre la curva verde se corresponden a los ángulos que mayor módulo de velocidad desarrollan en el perihelio. Ésta va a permitir optimizar la energía en el impulso de velocidad que se aplicará en dicha posición.

En principio los valores que más nos interesan son los que requieren de una menor velocidad inicial, por lo que volveremos a calcularlos con mayor precisión, para los ángulos mayores a 3,8 radianes.



Impulso en el perihelio

Ahora, con el fin de acotar el resultado hallado, evaluaremos cuáles de los valores de ángulo y velocidad permiten aplicar un Δv_1 en el perihelio, que cumpla las siguientes condiciones:
- Permita escapar de la órbita del sol
- Desarrolle una velocidad mayor a la de Oumuamua
- Tenga una velocidad de costo menor a 40 km/s
- Tiene una orientación similar a la de Oumuamua ($\sim 0{,}96\ rad$, que es el ángulo en el que se encuentra Oumuamua a 5 años) (*)

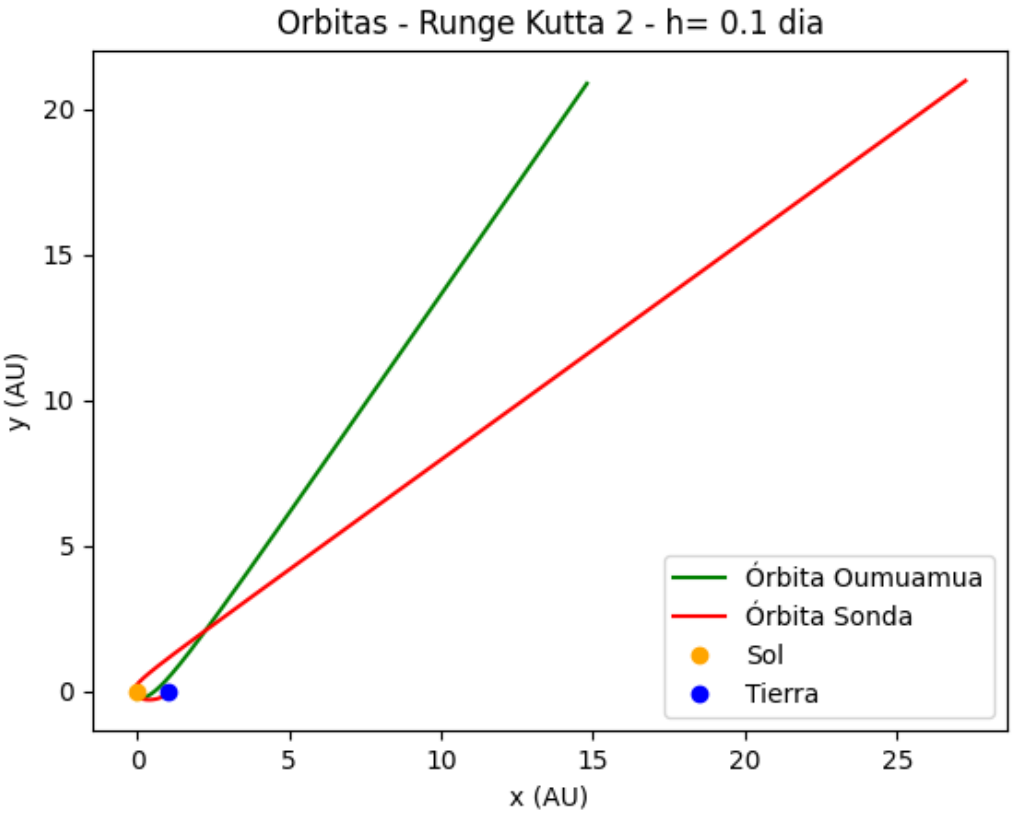
Con los datos hallados, calculando en cada caso para distintos Δv_1, se encontraron 148 configuraciones que llevan a una misión exitosa.

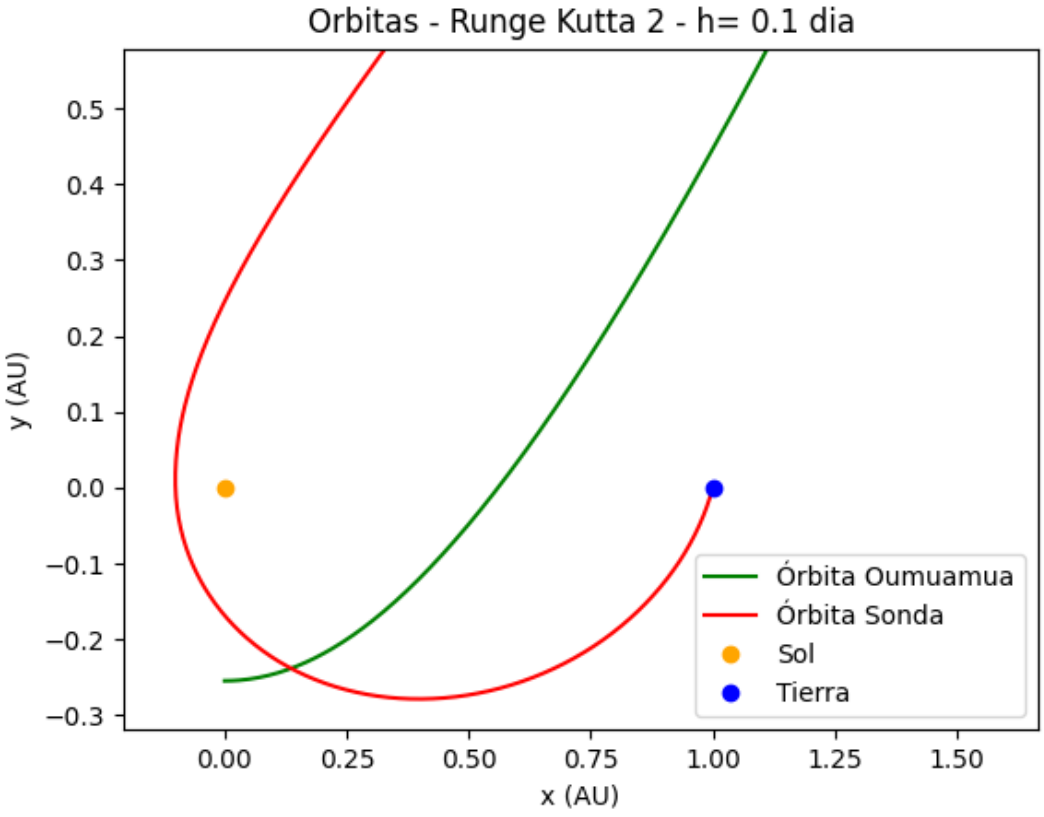
Configuración con menor tiempo

Con el fin de minimizar el tiempo de la misión, se calculó la velocidad infinito v_∞ de la sonda. Aquella que sea mayor resultará ser la que en menor tiempo alcance a Oumuamua.

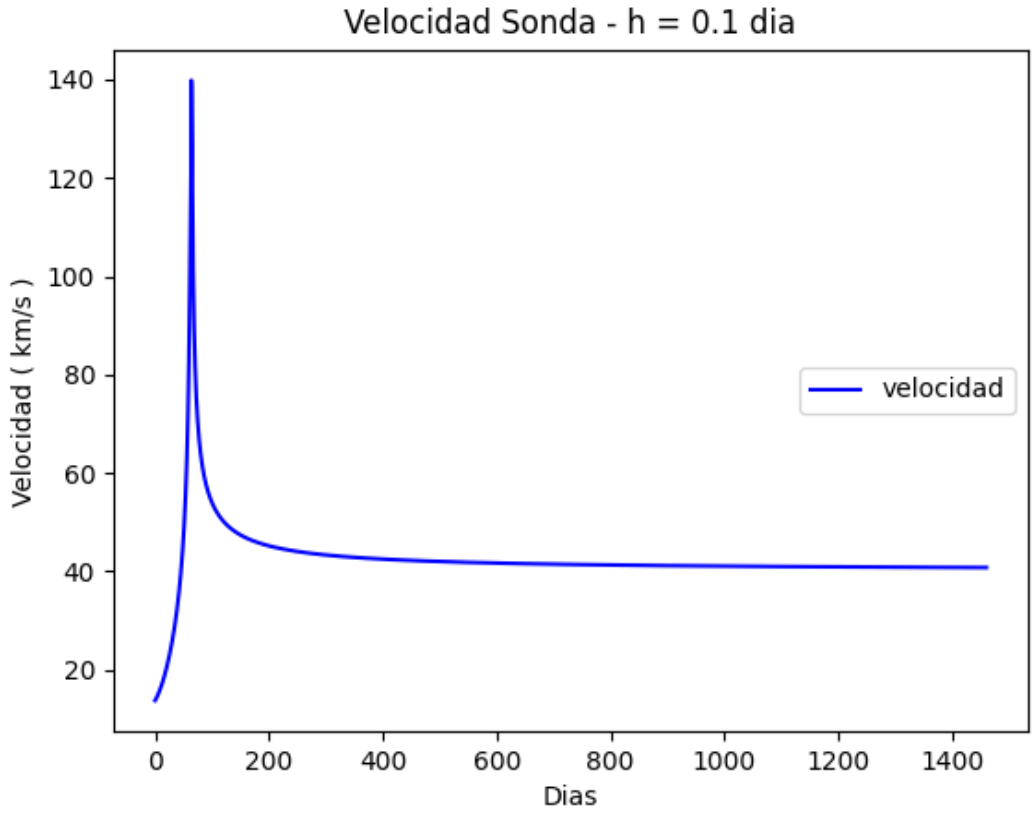
Configuración definitiva misión

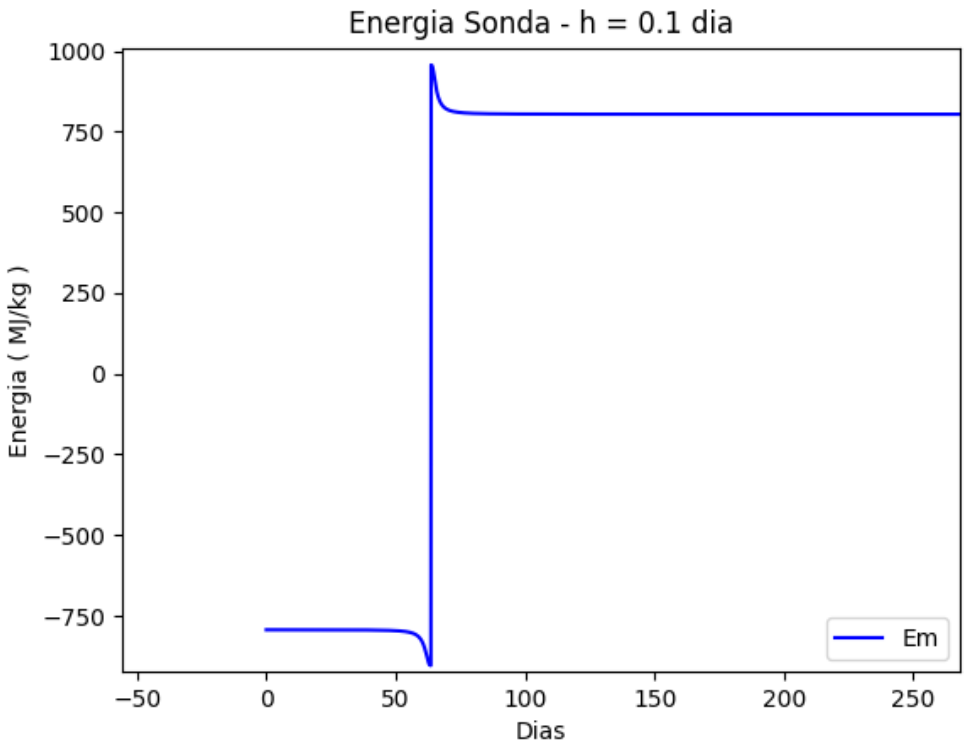
$v_0 = 13{,}7\ km/s$
$\alpha_0 = 4{,}32\ rad \approx 247{,}5\ °$
$\Delta v_1 = 14\ km/s$
$v_\infty = 40{,}7 \frac{km}{s}$
$v_{\text{costo}} = 39{,}9\ km/s$


Orbitas - Runge Kutta 2 - h= 0.1 dia
y (AU)
x (AU)
Órbita Oumuamua
Órbita Sonda
Sol
Tierra

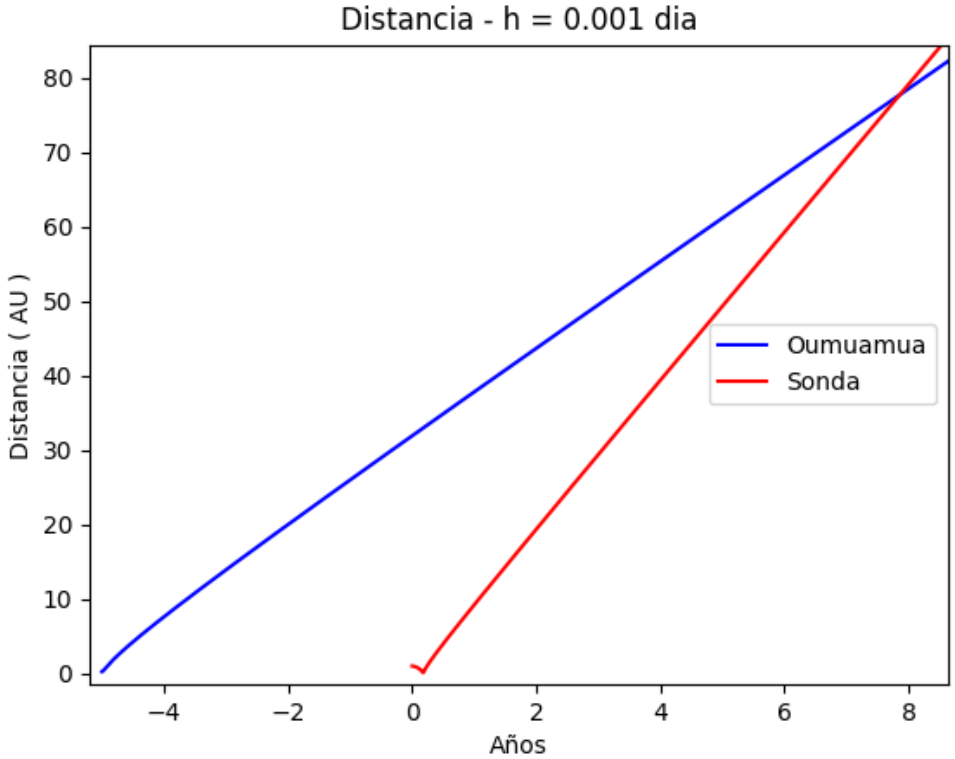

Orbitas - Runge Kutta 2 - h= 0.1 dia
y (AU)
x (AU)
Órbita Oumuamua
Órbita Sonda
Sol
Tierra

(*) No fue posible hallar una configuración dentro del costo, que resulte en una orientación precisa hacia Oumuamua. En los datos hallados se aprecia una correlación entre el α_0, v_0 y la orientación. Los ángulos más cercanos a 180° resultan en una trayectoria más enfocada hacia Oumuamua, pero desarrollan menor velocidad en el perihelio, por lo que requieren mayor impulso Δv_1, mientras que los cercanos a 270° se alejan en su trayectoria, pero desarrollan mayor velocidad.





Se puede apreciar en el gráfico de la energía mecánica de la sonda, que la misma no se conserva. Ésta tuvo un aporte notorio al momento de aportar el impulso de velocidad.



Comparando las distancias al sol de Oumuamua y de la sonda, que fue lanzada 5 años después, hallamos que el encuentro ocurre luego de 7,8 años, a una distancia de 77,5 AU del sol.

CONCLUSIONES

Se pudo modelar exitosamente las ecuaciones que describen las órbitas de los cuerpos, evaluando también su orden y error según el paso.

ANEXO I

```
def euler(x0, y0, vx0, vy0, añosT, T, paso):
    h = T / (365 / paso)
    pasos = int(365 / paso) + 1
    #   print(pasos)
    X, Y, Vx, Vy = inicializarVariables(x0, y0, vx0, vy0, pasos)

    i = 0
    while (i < pasos - 1):
        X[i + 1] = X[i] + h * Vx[i]
        Y[i + 1] = Y[i] + h * Vy[i]
        ang = np.arctan2((YS - Y[i]), (XS - X[i]))

        Vx[i + 1] = Vx[i] + h * ((G * MS / math.pow(obtenerDistanciaAlSol(X[i], Y[i]), 2)) * np.cos(ang))
        Vy[i + 1] = Vy[i] + h * ((G * MS / math.pow(obtenerDistanciaAlSol(X[i], Y[i]), 2)) * np.sin(ang))
        i += 1
    return X, Y, Vx, Vy

def pasoCorrectorRK2(xActual, yActual, vxActual, vyActual, h):
    angPosicion = np.arctan2((YS - yActual), (XS - xActual))
    distanciaAlSolActual = obtenerDistanciaAlSol(xActual, yActual)
    xSiguiente = xActual + h * (
            vxActual + (h / 2) * (((G * MS) / math.pow(distanciaAlSolActual, 2)) *
np.cos(angPosicion)))
    ySiguiente = yActual + h * (
            vyActual + (h / 2) * (((G * MS) / math.pow(distanciaAlSolActual, 2)) * np.sin(angPosicion)))

    angVelocidad = np.arctan2(YS - yActual - (h / 2) * vyActual, XS - xActual - (h / 2) * vxActual)
    vxSiguiente = vxActual + h * (((G * MS) / math.pow(distanciaAlSolActual, 2)) *
np.cos(angVelocidad))
    vySiguiente = vyActual + h * (((G * MS) / math.pow(distanciaAlSolActual, 2)) *
np.sin(angVelocidad))
    return xSiguiente, ySiguiente, vxSiguiente, vySiguiente

def rungeKutta2(x0, y0, vx0, vy0, años, T, paso):
    h = T / (365 / paso)
    pasos = int(365 / paso) * años + 1
    X, Y, Vx, Vy = inicializarVariables(x0, y0, vx0, vy0, pasos)

    i = 0
    while (i < pasos - 1):
        X[i + 1], Y[i + 1], Vx[i + 1], Vy[i + 1] = pasoCorrectorRK2(X[i], Y[i], Vx[i], Vy[i], h)
        i += 1
    return X, Y, Vx, Vy
```

```
def variarCondicionesMision(xM0, yM0, vxM0, vyM0, añosU, T, h):
    anguloInicial = np.pi
    anguloFinal = np.pi * 3 / 2
    velocidadInicial = vET
    velocidadFinal = 30000
    vxM0 = vM0 * np.cos(anguloMision)
    vyM0 = vM0 * np.sin(anguloMision)

    pasoAngular = (anguloFinal - anguloInicial) / 40
    pasoVelocidad = 500
    listaVelocidades = list()
    listaAngulos = list()
    listaDeltaV = list()
    deltaVInicial = 500
    incrementoDeltaV = 500
    costoMaximo = 40000
    X, Y, Vx, Vy = None, None, None, None

    angulo = anguloInicial
    while angulo <= anguloFinal:
        velocidad = velocidadInicial
        aunNoSeHalloLaVelocidad = True
        while (velocidad < velocidadFinal) and aunNoSeHalloLaVelocidad:
            vxM = velocidad * np.cos(angulo)
            vyM = velocidad * np.sin(angulo)
            if rungeKutta2LlegaAlPerihelioConDistanciaSuficiente(xM0, yM0, vxM, vyM, añosU, T, h):
                # Aca pruebo los deltaV
                deltaV = deltaVInicial
                while (velocidad + deltaV) < costoMaximo:
                    if probarMision(xM0, yM0, vxM, vyM, añosU, T, h, deltaV):
                        print("La mision fue Exitosa")
                        listaVelocidades.append(velocidad)
                        listaAngulos.append(angulo)
                        listaDeltaV.append(deltaV)
                    deltaV = deltaV + incrementoDeltaV
                aunNoSeHalloLaVelocidad = False
            velocidad = velocidad + pasoVelocidad
        angulo = angulo + pasoAngular
    return listaAngulos, listaVelocidades, listaDeltaV
```