Trabajo de fin de grado: Movimiento de asteroides en las Zanjas de Kirkwood

Universidad Autónoma de Madrid
Roberto Mazo Fernández

UNIVERSIDAD AUTONOMA DE MADRID

Resumen

- Diseñado un método para caracterizar anchura de las Zanjas
- Simulación movimiento asteroides con Python
- Comparación métodos integración numérica
- Influencia de Marte y Saturno

¿Qué son y como se producen las Zanjas de Kirkwood?

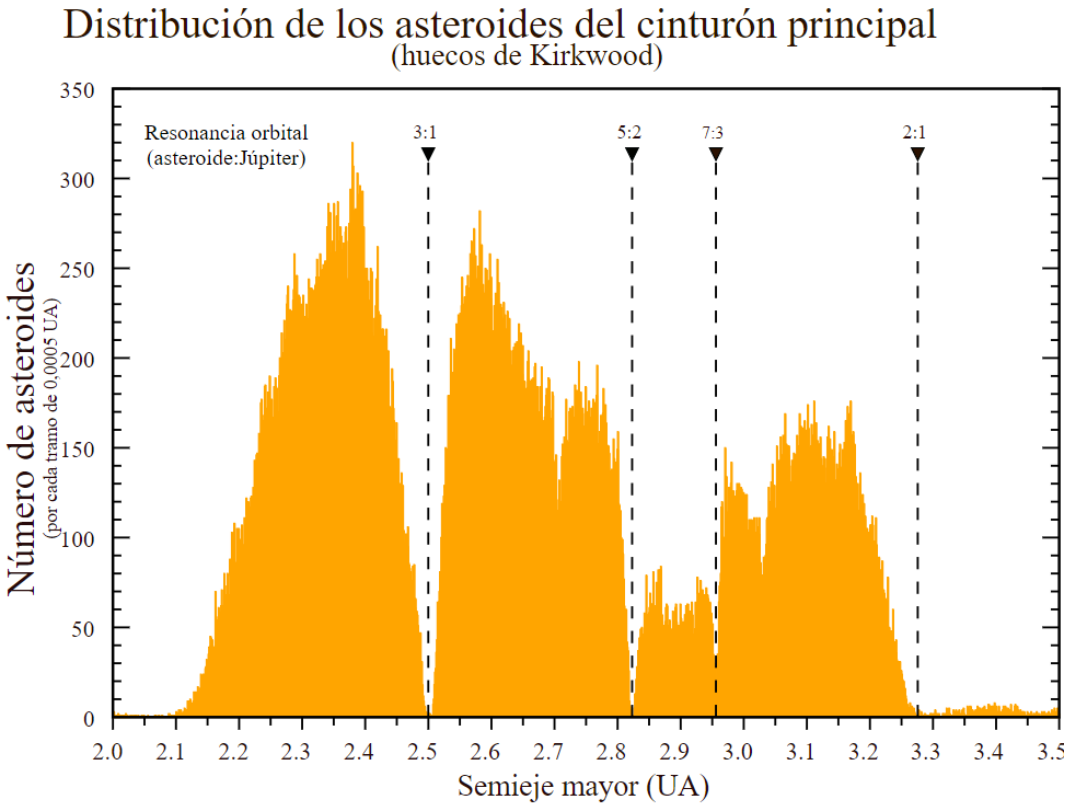
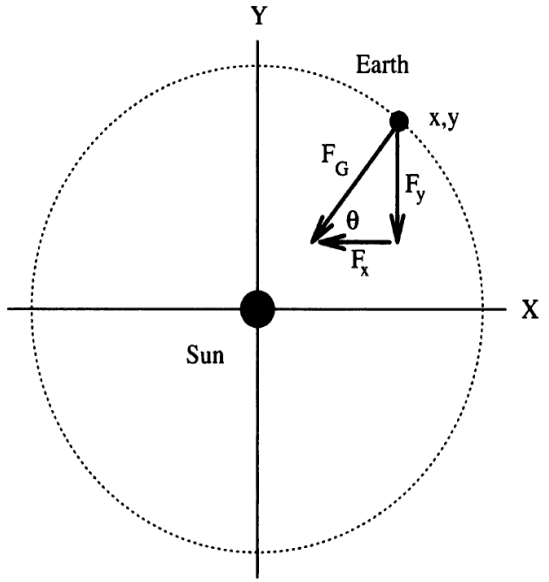


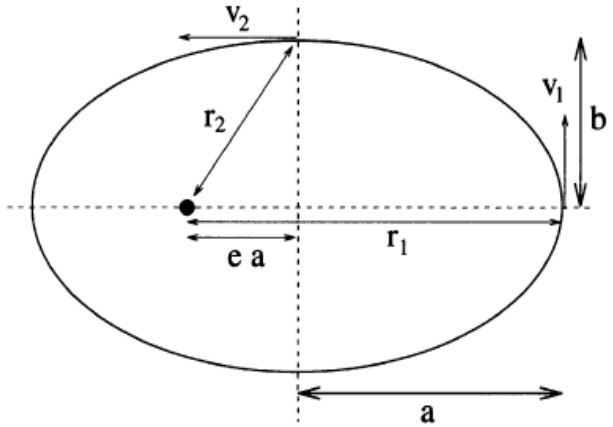
Figura 1: Numero de asteroides en función de la distancia al Sol.

Se **cree** que la acumulación de impulsos que Júpiter tiene sobre el asteroide
→Régimen caótico tal que cualquier mínima perturbación →Expulsa de la orbita [1]

[1] Moons, M. Review of the dynamics in the Kirkwood gaps. *Celestial Mechanics and Dynamical Astronomy*, 1996.

Problema de dos cuerpos





$$F_{Gx} = -\frac{G \cdot M_S \cdot M_T}{r_{ST}^2} \cdot \cos(\theta) = -\frac{G \cdot M_S \cdot M_T}{r_{ST}^3} \cdot \mathrm{x}$$

$$F_{Gy} = -\frac{G \cdot M_S \cdot M_T}{r_{ST}^2} \cdot \sin(\theta) = -\frac{G \cdot M_S \cdot M_T}{r_{ST}^3} \cdot \mathrm{y}$$

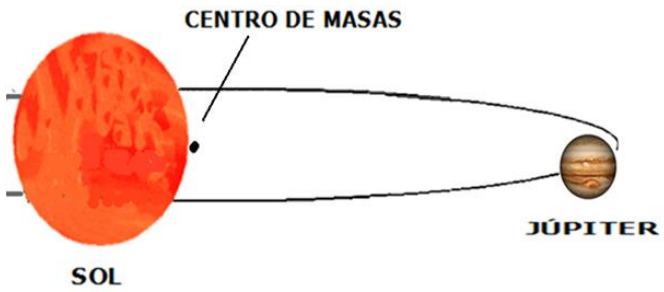
Bajo la acción de fuerza conservativa →Conservación de la energía mecánica y momento angular

Unidades → Tiempo : Años Distancia : Unidades Astronómicas

$$F_{\text{centrípeta}} = F_{\text{gravitatoria}}$$

$$G \cdot M_s = v^2 \cdot r = 4\,\pi^2\ \mathrm{AU}^3/\mathrm{año}^2$$

Problema de tres cuerpos



- Aproximaciones:
  - El Sol debido a su masa permanecerá quieto
  - La fuerza que ejerce el asteroide sobre Júpiter es despreciable

- $F_{Gx} = -\frac{G \cdot M_S \cdot M_a}{r_{aS}^2} \cdot \mathrm{x} - \frac{G \cdot M_J \cdot M_a}{{r_{aJ}}^3} \cdot (x_a - x_J)$
- $F_{Gy} = -\frac{G \cdot M_S \cdot M_a}{r_{aS}^2} \cdot \mathrm{y} - \frac{G \cdot M_J \cdot M_a}{{r_{aJ}}^3} \cdot (y_a - y_J)$

$$GM_J = GM_S \frac{M_J}{M_S} = 4\,\pi^2\,\frac{M_J}{M_S}\ \mathrm{UA^3/año^2}$$

Métodos de integración numérica: Euler, Euler-Cromer, Velocidad Verlet y Runge-Kutta 4

- Ecuaciones a resolver: $m\frac{d^2x}{dt^2} = -\frac{GMm}{r^3}x$ $\quad$ $m\frac{d^2y}{dt^2} = -\frac{GMm}{r^3}y$
- Transformamos en: $\frac{dx}{dt} = f(t,v)$ $\quad$ $\frac{dv}{dt} = g(t,x)$

Análogo para y

```
Funcion que nos vale para calcular la posicion y la velocidad de un objeto acelerado en funcion del tiempo. Metodo de Verlet
    Input:
         a=     aceleracion que sufre el cuerpo
         r0=    vector que contenga posicion inicial
         v0=    vector que contenga la velocidad inicial
         dt=    paso de tiempo
         tend=  tiempo final

    Output:
        x_vector, y_vector, z_vector =      Vectores que contienen las posiciones del cuerpo en funcion del tiempo
        vx_vector, vy_vector, vz_vector =   Vectores que contienen las velocidades del cuerpo en funcion del tiempo
        t_vector =                          Vector que contiene los tiempos

Ejemplo de funcion aceleracion
 def a(x,y,z):
    ax= -G*M*x/(x**2+y**2+z**2)**(3/2)
    ay= -G*M*y/(x**2+y**2+z**2)**(3/2)
    az= -G*M*z/(x**2+y**2+z**2)**(3/2)
    return [ax,ay,az]
```

Euler :

- Aproximación Taylor orden 1: $x(t+\Delta t) = x(t) + \frac{\partial x}{\partial t}\Delta t + \ldots$

$$x_{n+1} = x_n + f(t_n, v_n)\Delta t \qquad v_{n+1} = v_n + g(t_n, x_n)\Delta t$$

Malo en movimientos cíclicos →Aumenta la energía con los ciclos

```
for i in range (0,n):

    [ax, ay, az] = a(x, y,z)

    x_next = x + vx * dt
    y_next = y + vy * dt
    z_next = z + vz * dt

    vx_next = vx + ax * dt
    vy_next = vy + ay * dt
    vz_next = vz + az * dt
```

Euler-Cromer:

- Mejora el método de Euler para movimientos cíclicos

$$v_{n+1} = v_n + g(t_n, x_n)\Delta t \qquad x_{n+1} = x_n + f(t_n, v_{n+1})\Delta t$$

```
for i in range (0,n):

    [ax, ay, az] = a(x, y, z)

    vx_next = vx + ax * dt
    vy_next = vy + ay * dt
    vz_next = vz + az * dt

    x_next = x + vx_next * dt
    y_next = y + vy_next * dt
    z_next = z + vz_next * dt
```

Velocidad Verlet:

- Mejora el método de Verlet ya que podemos calcular velocidad

$$x_{n+1} = x_n + v_{n+1}\,\Delta t + \frac{1}{2}\,a_n\,\Delta t^2 \qquad v_{n+1} = v_n + \frac{a_n + a_{n+1}}{2}\Delta t$$

```
for i in range (0,n):

    [ax, ay, az] = a(x, y, z)
    x_next= x + vx * dt + 0.5 * ax * dt ** 2
    y_next= y + vy * dt + 0.5 * ay * dt ** 2
    z_next= z + vz * dt + 0.5 * az * dt ** 2

    [ax_next, ay_next, az_next] = a(x_next, y_next, z_next)
    vx_next = vx + 0.5 * ( ax + ax_next ) * dt
    vy_next = vy + 0.5 * ( ay + ay_next ) * dt
    vz_next = vz + 0.5 * ( az + az_next ) * dt
```

Órbita de Júpiter

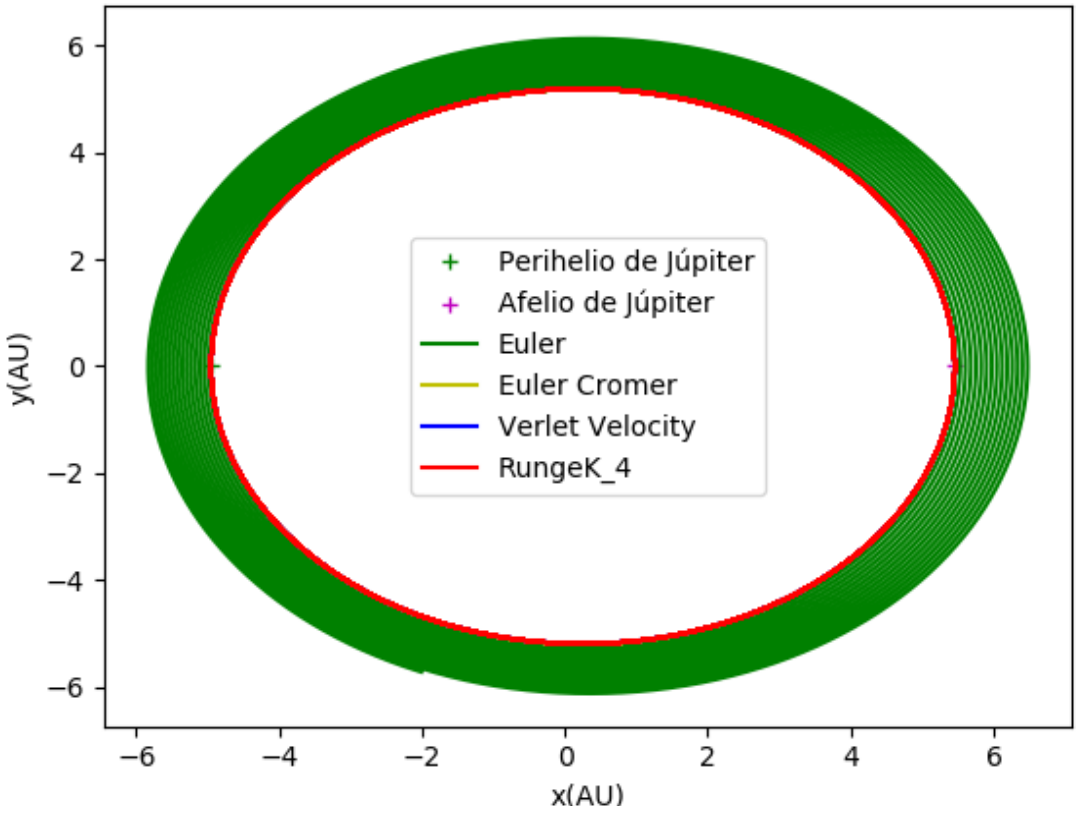


Figura 11: Comparación de la órbita de Júpiter al cabo de 200 años para los distintos métodos de integración.

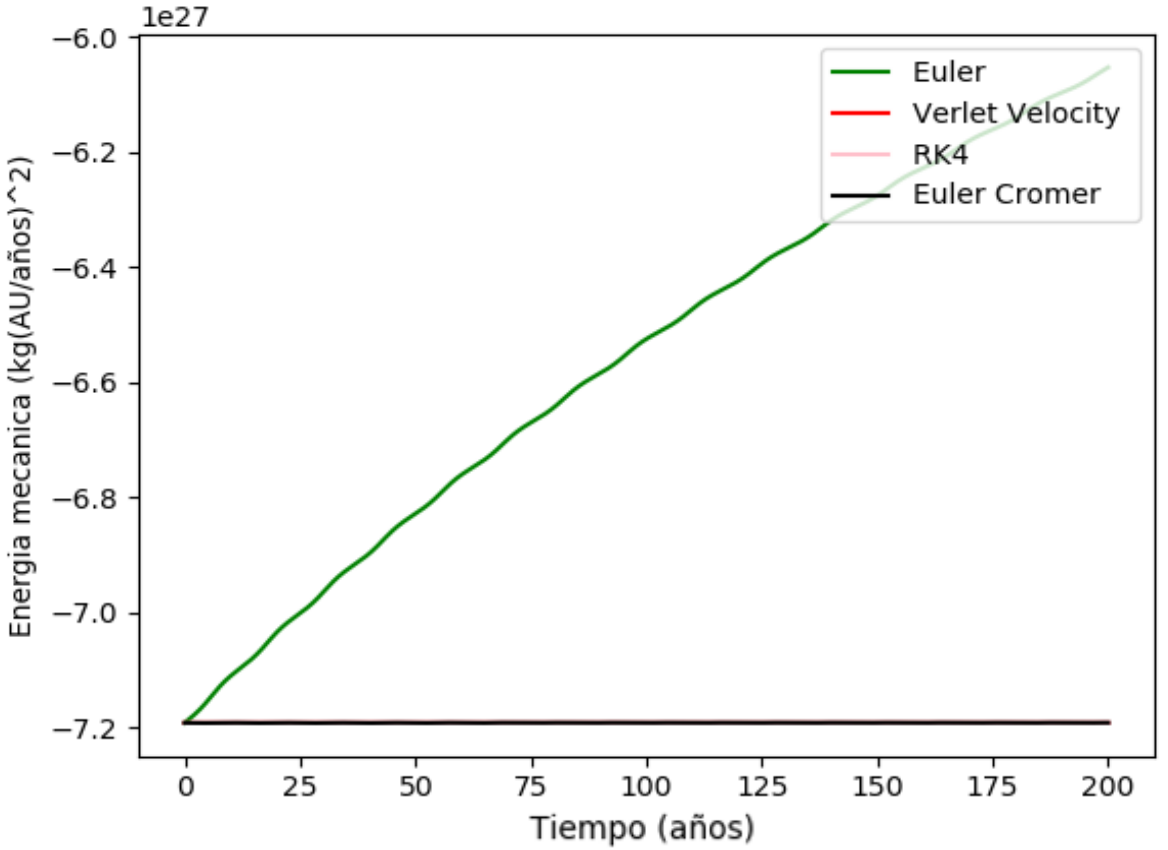


Figura 12: Comparación de la energía mecánica de Júpiter en su órbita con los distintos métodos de integración.

Órbita de Júpiter

Método	Tiempo computacional (s)	Desviación del afelio (UA)
Euler	0.693	No procede
Euler Cromer	0.825	4e-07
Velocidad Verlet	1.129	2e-07
Runge-Kutta 4	1.521	2e-07

Tabla 2: Comparación entre los métodos de integración al realizar el cálculo de la trayectoria y su desviación del afelio al cabo de T=200años

La desviación es menor que el radio de Júpiter (69.911km) → necesitamos hacerlo con un cuerpo mas pequeños

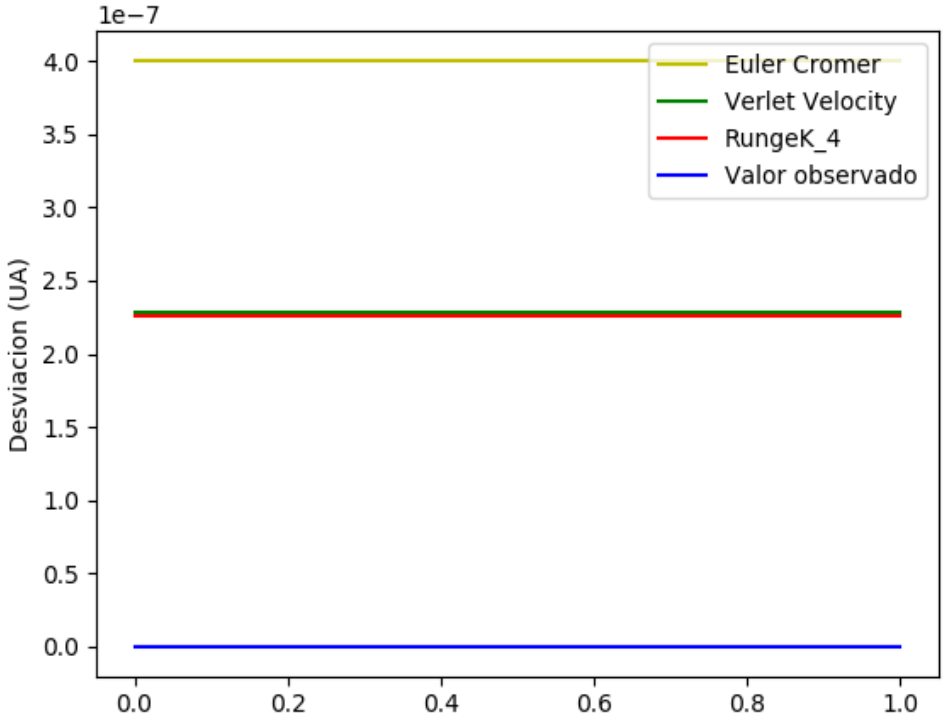


Figura 13: Comparación de los métodos al pasar cerca del afelio de Júpiter.

Período obtenido con el método de Verlet: $T_{experimental}=11.8878 \pm 0.0005$ años $T_{bilbiografico}=11.872 \pm 0.003$ años

Órbita de Juno

Método	Tiempo computacional(s)	Desviación del afelio (UA)
Euler Cromer	32.421	0.006
Velocidad Verlet	37.508	0.006
Rungekutta4	65.548	0.006

Tabla 3: Comparación de los métodos de integración al realizar el cálculo de la trayectoria de Júpiter y Juno, así como la desviación del afelio de Juno al cabo de 2000 años terrestres

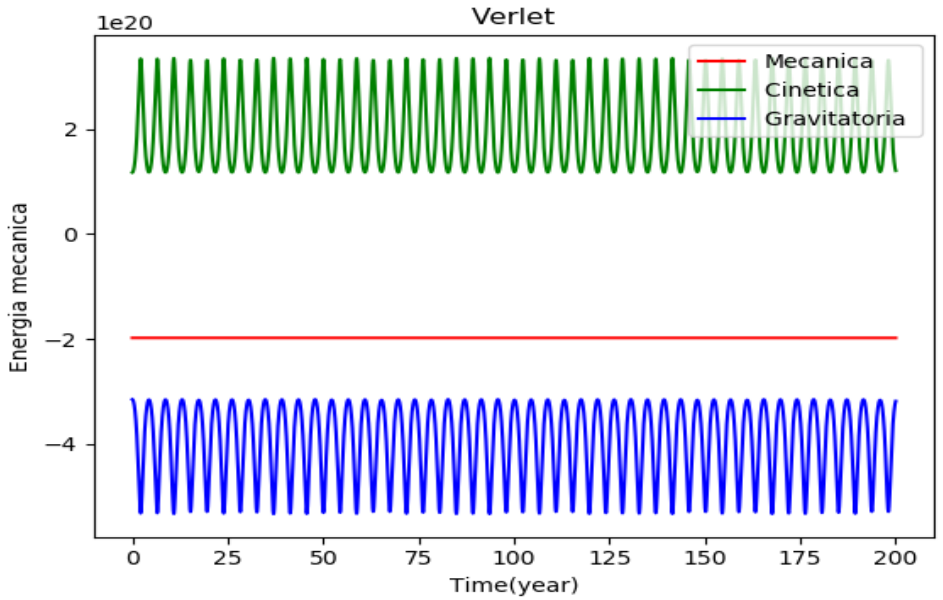


Figura 17: Representación de las energías de Juno con el método de Verlet..

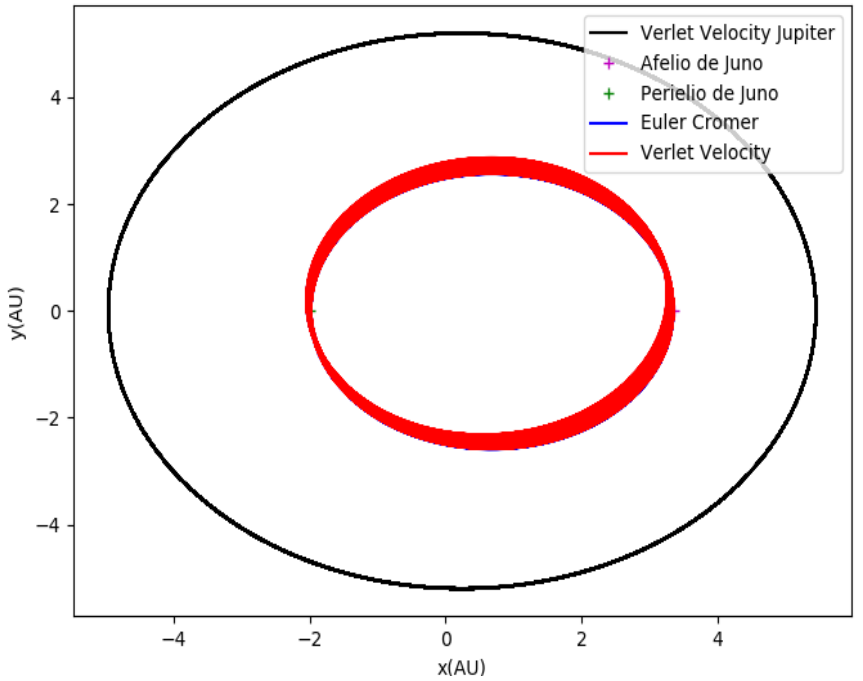


Figura 15: Trayectoria de Júpiter y del asteroide Juno con Verlet y Euler Cromer con T=2000 años terrestres

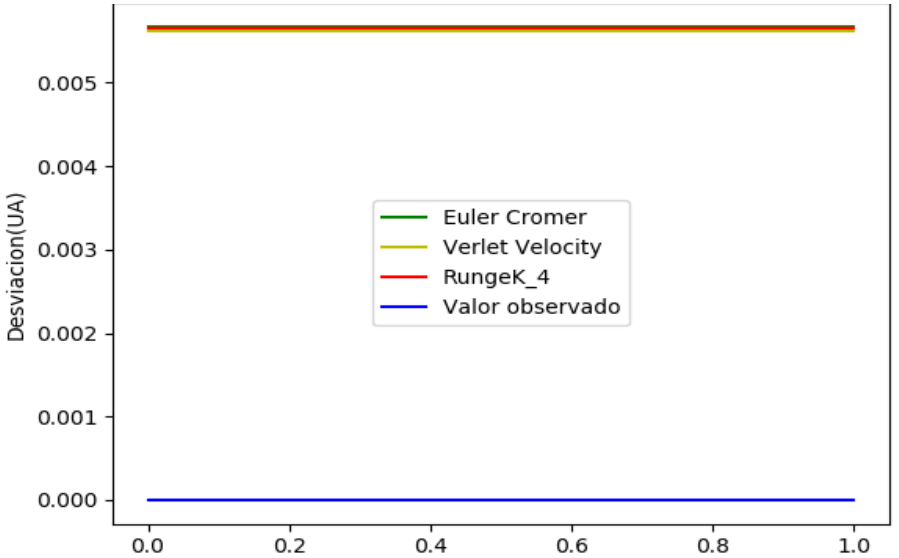


Figura 16: Desviación de los distintos métodos al pasar cerca del afelio de Juno con T=2000 años terrestres

Estudio Zanjas de Kirkwood

Circular

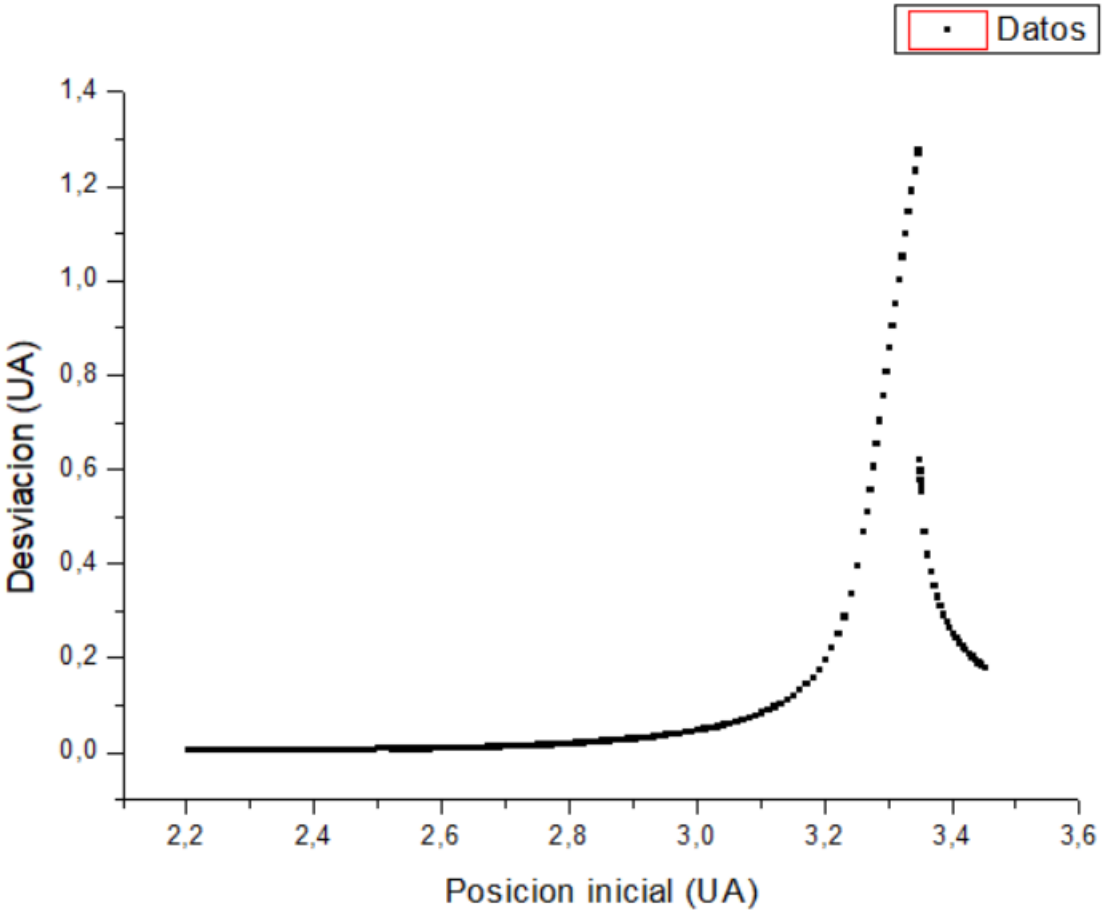


Figura 19: Desviación de un asteroide en función de su posición inicial respecto al Sol al cabo de 1000 años asumiendo órbita de Júpiter circular

Elíptico

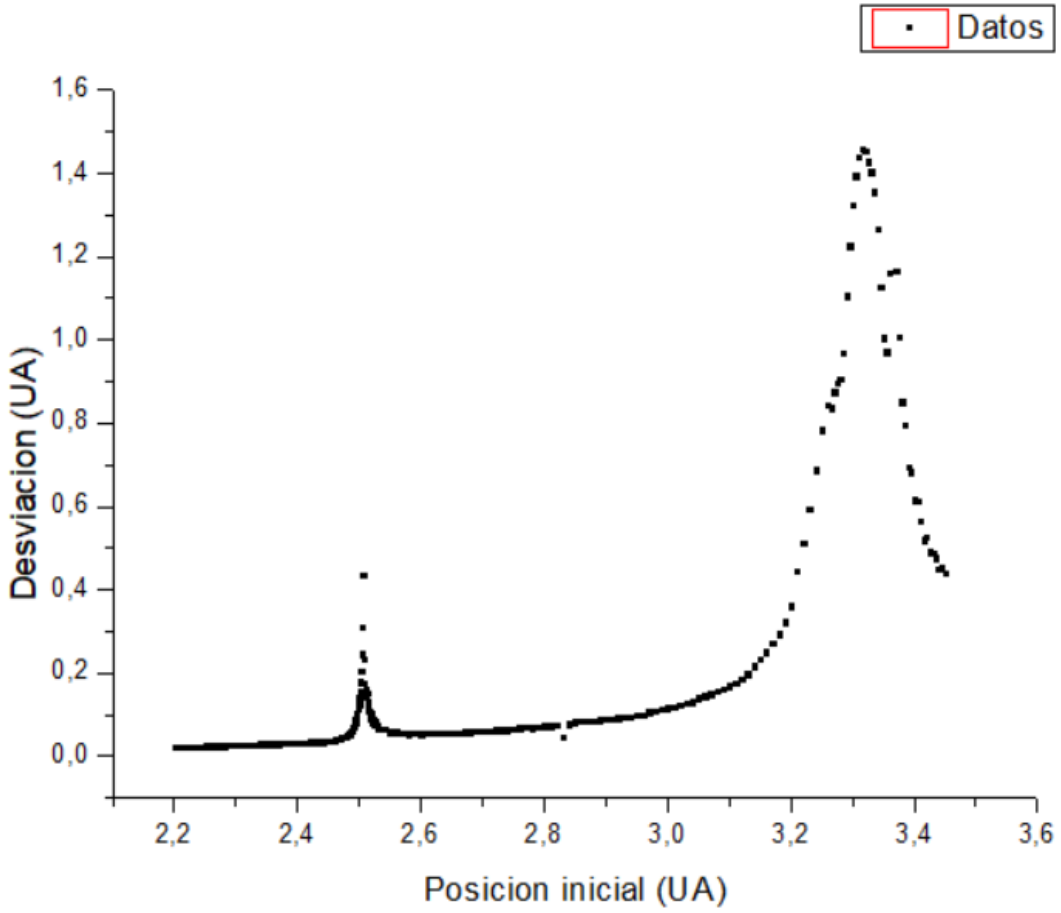


Figura 20: Desviación de un asteroide en función de su posición inicial respecto al Sol al cabo de 1000 años asumiendo órbita de Júpiter elíptica

Comportamiento en las proximidades de las Zanjas de Kirkwood

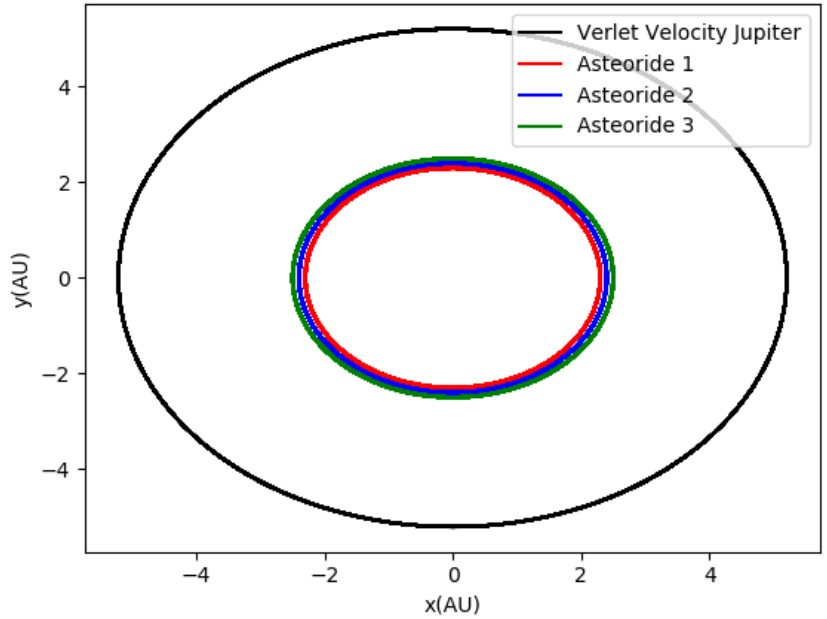
Cuerpo celeste	Distancia (AU)	Velocidad inicial (AU/año)
Asteroide 1	3.000	3.628
Asteroide 2	3.276	3.471
Asteroide 3	3.600	3.312

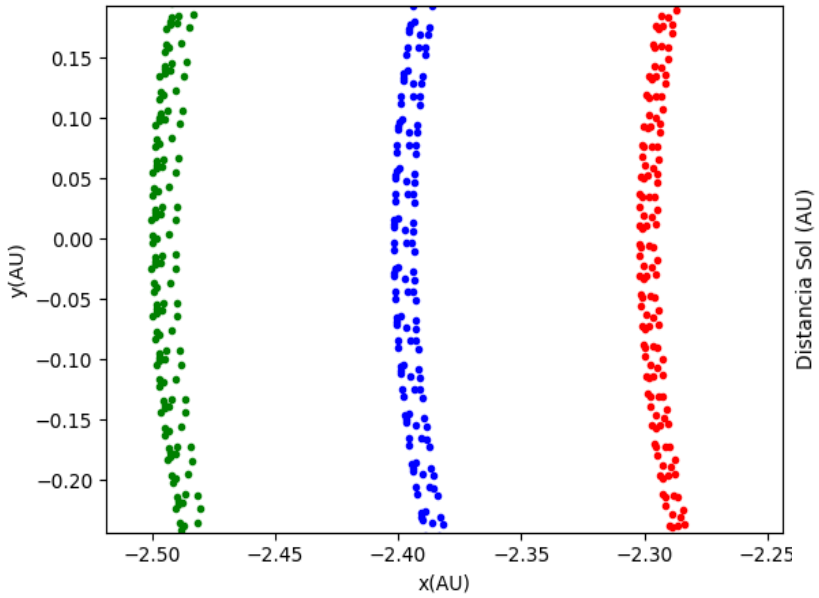
Tabla 4: Posiciones iniciales y velocidades usadas para tres asteroides hipotéticos en la vecindad de la zanja de Kirkwood con resonancia 2/1

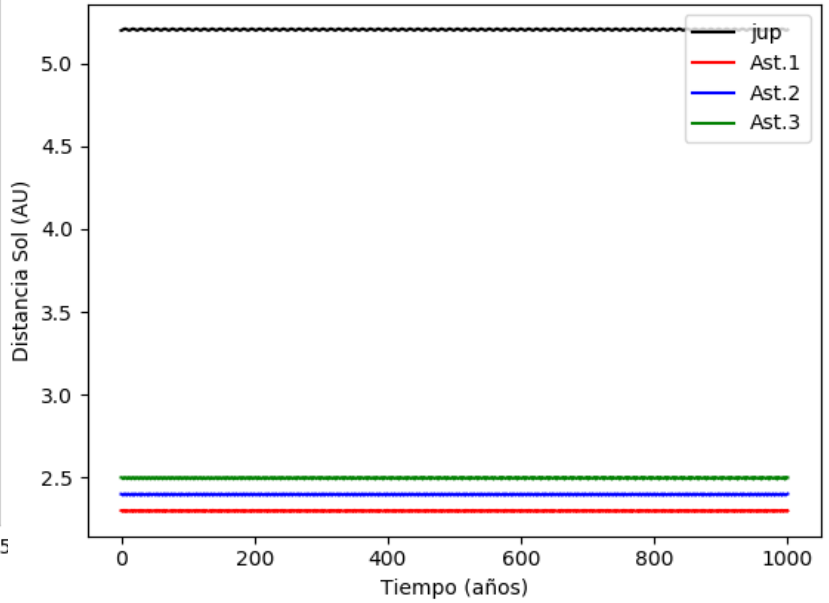
Cuerpo celeste	Distancia (AU)	Velocidad inicial (AU/año)
Asteroide 1	3.000	3.628
Asteroide 2	3.276	3.471
Asteroide 3	3.600	3.312

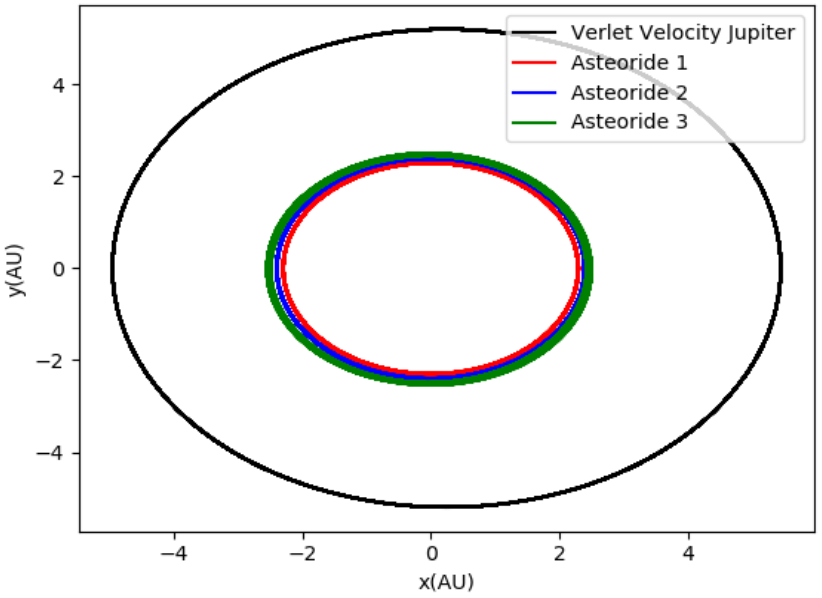
Tabla 5: Posiciones iniciales y velocidades usadas para tres asteroides hipotéticos en la vecindad de la Zanja de Kirkwood con resonancia 3/1

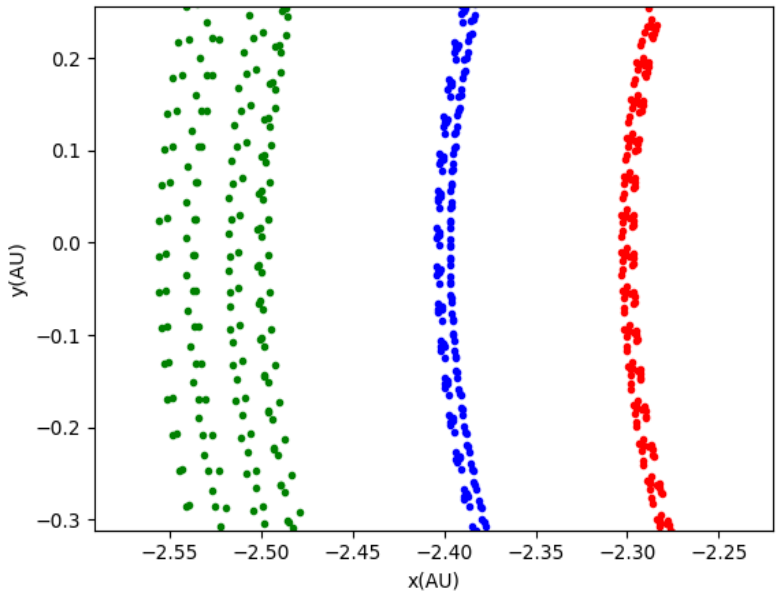
Zanja 3/1

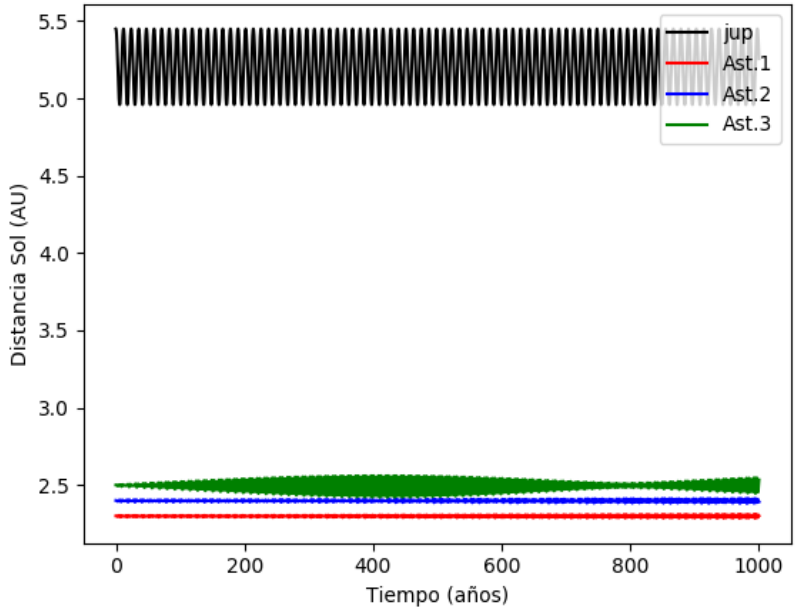

Verlet Velocity Jupiter
Asteoride 1
Asteoride 2
Asteoride 3
y(AU)
x(AU)


y(AU)
x(AU)

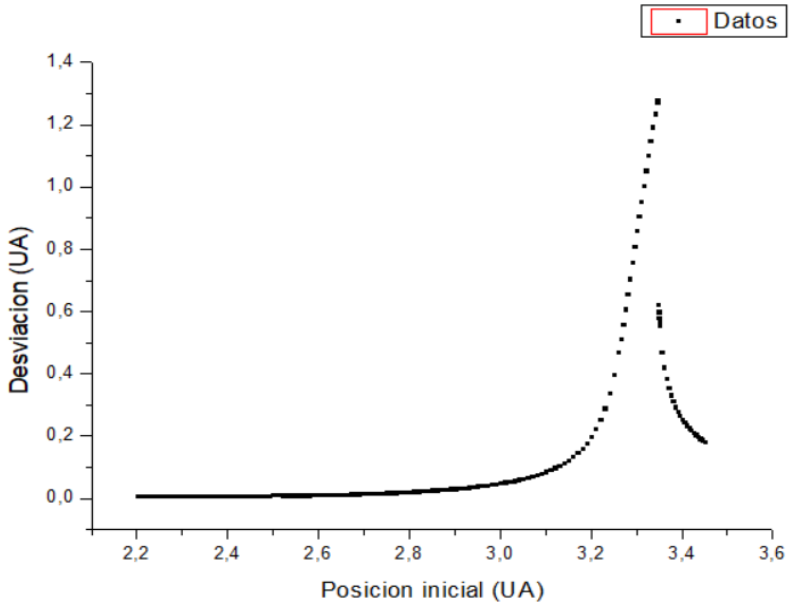

jup
Ast.1
Ast.2
Ast.3
Distancia Sol (AU)
Tiempo (años)

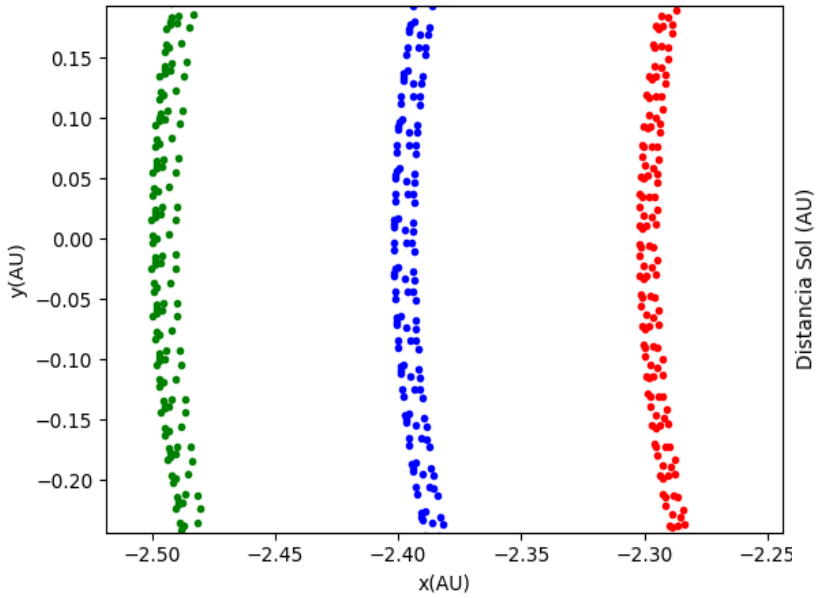

Verlet Velocity Jupiter
Asteoride 1
Asteoride 2
Asteoride 3
y(AU)
x(AU)

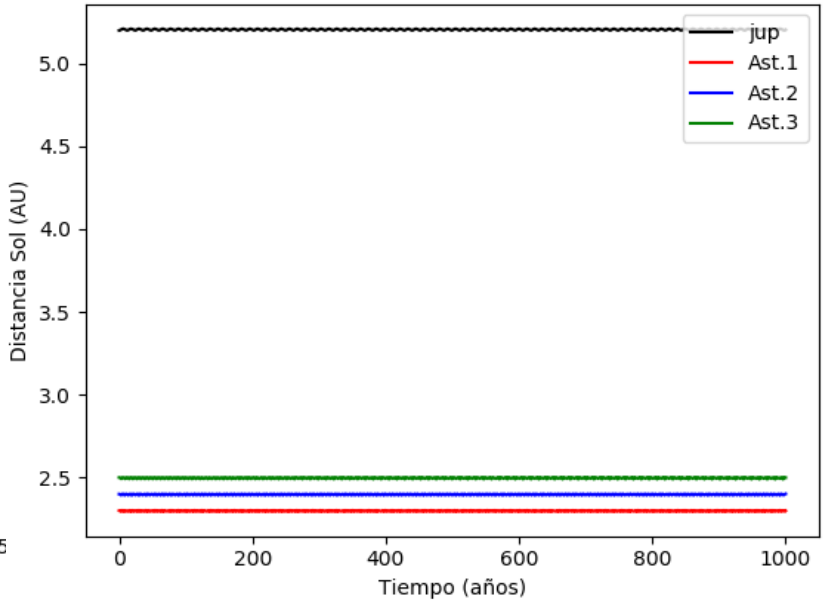

y(AU)
x(AU)

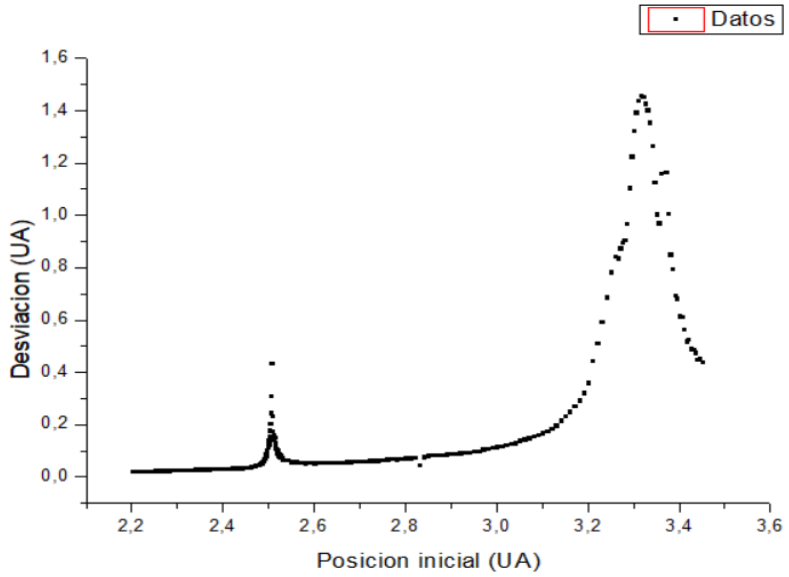

jup
Ast.1
Ast.2
Ast.3
Distancia Sol (AU)
Tiempo (años)

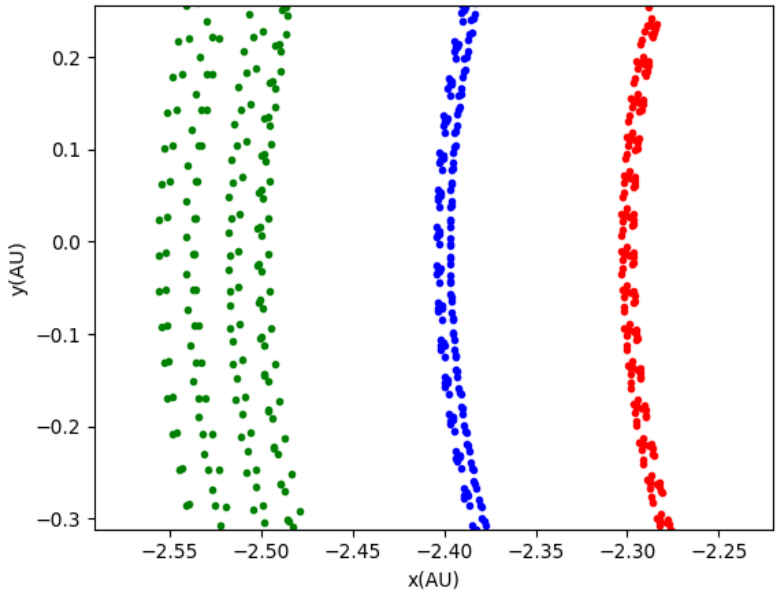
Zanja 3/1

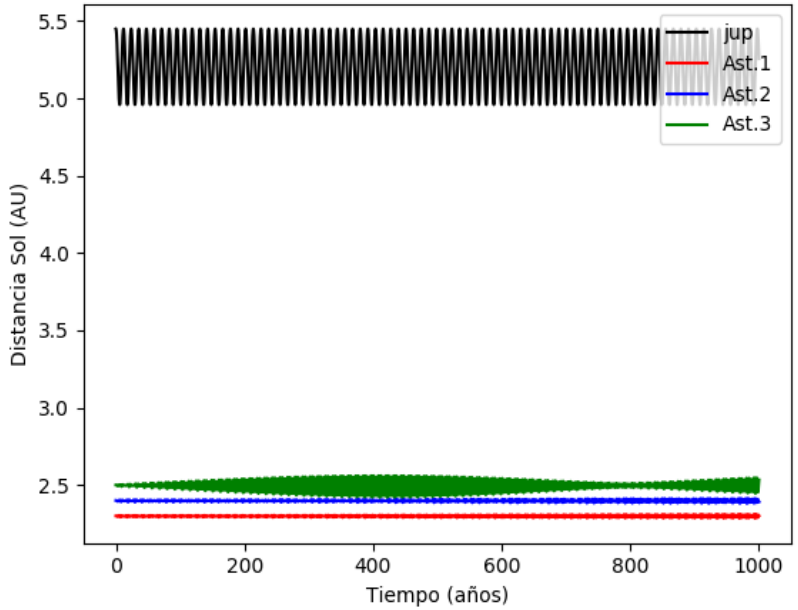

Datos
Desviacion (UA)
Posicion inicial (UA)


y(AU)
x(AU)

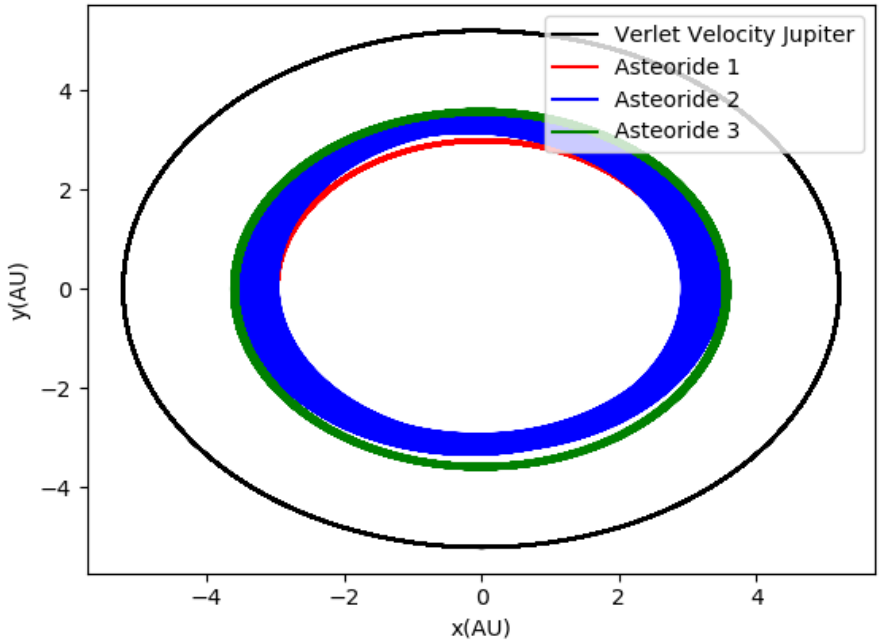

jup
Ast.1
Ast.2
Ast.3
Distancia Sol (AU)
Tiempo (años)


Datos
Desviacion (UA)
Posicion inicial (UA)

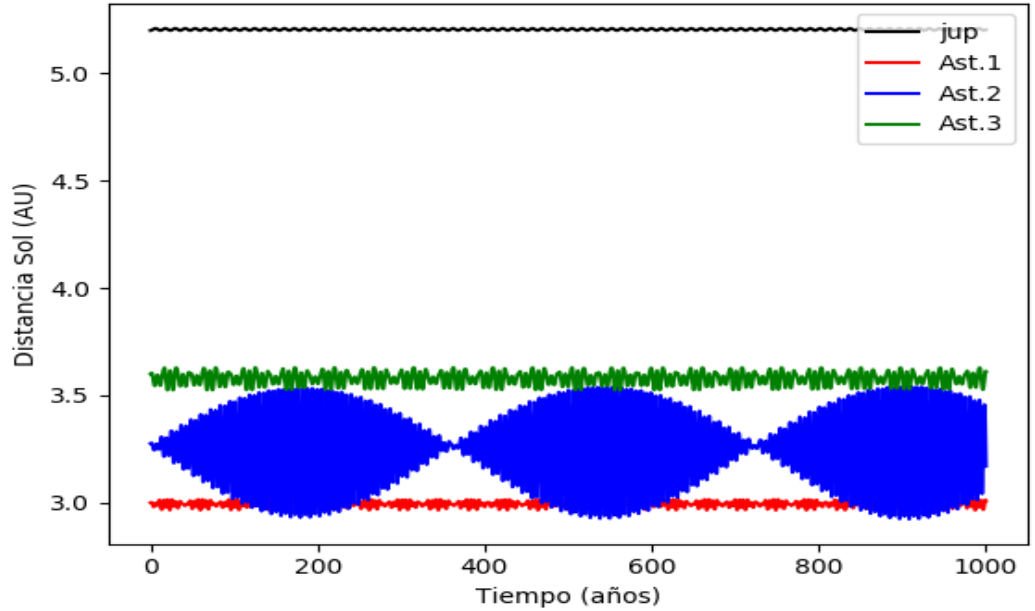

y(AU)
x(AU)

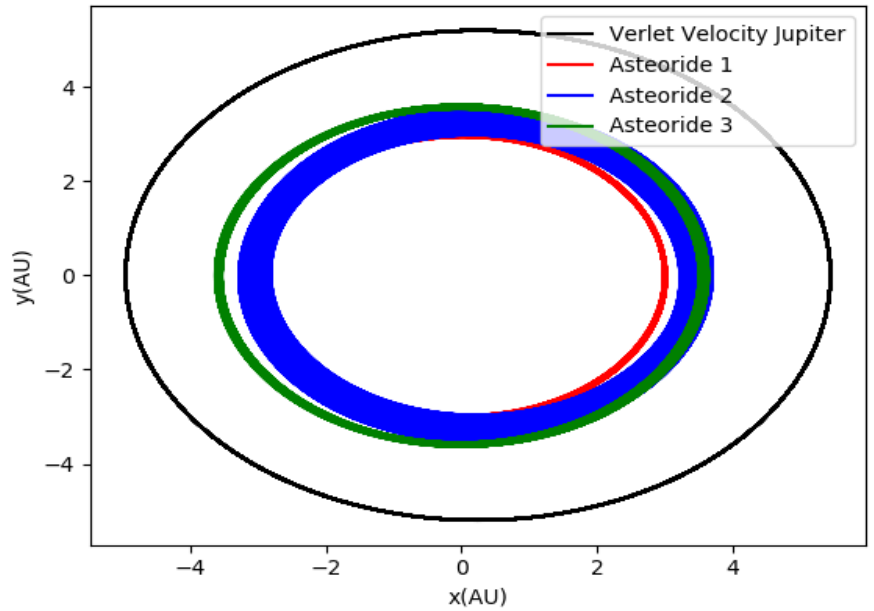

jup
Ast.1
Ast.2
Ast.3
Distancia Sol (AU)
Tiempo (años)

Zanja 2/1

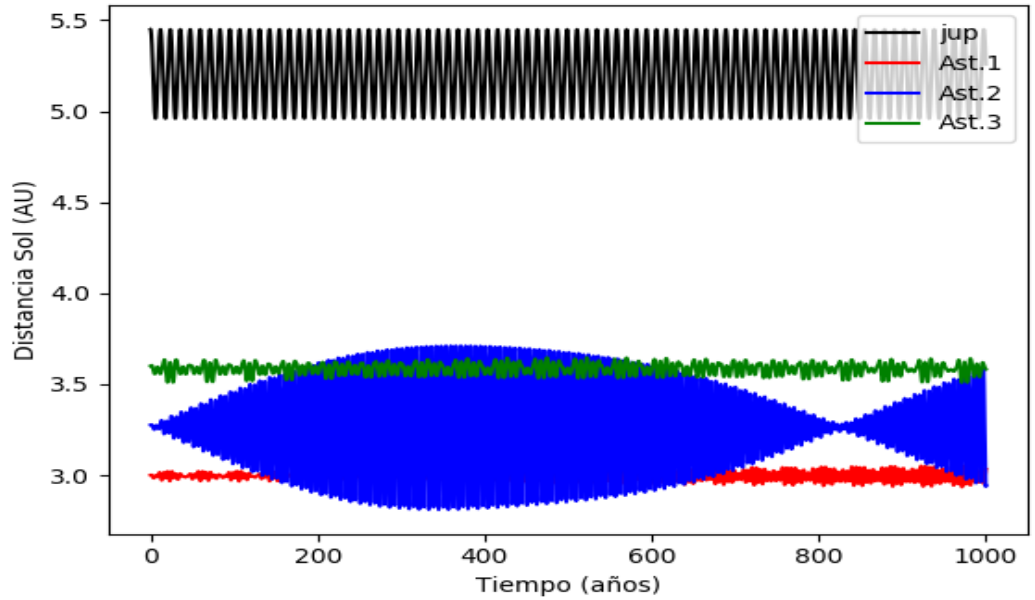


Circular





Elíptico



Anchura de las Zanjas

Zanja 3/1

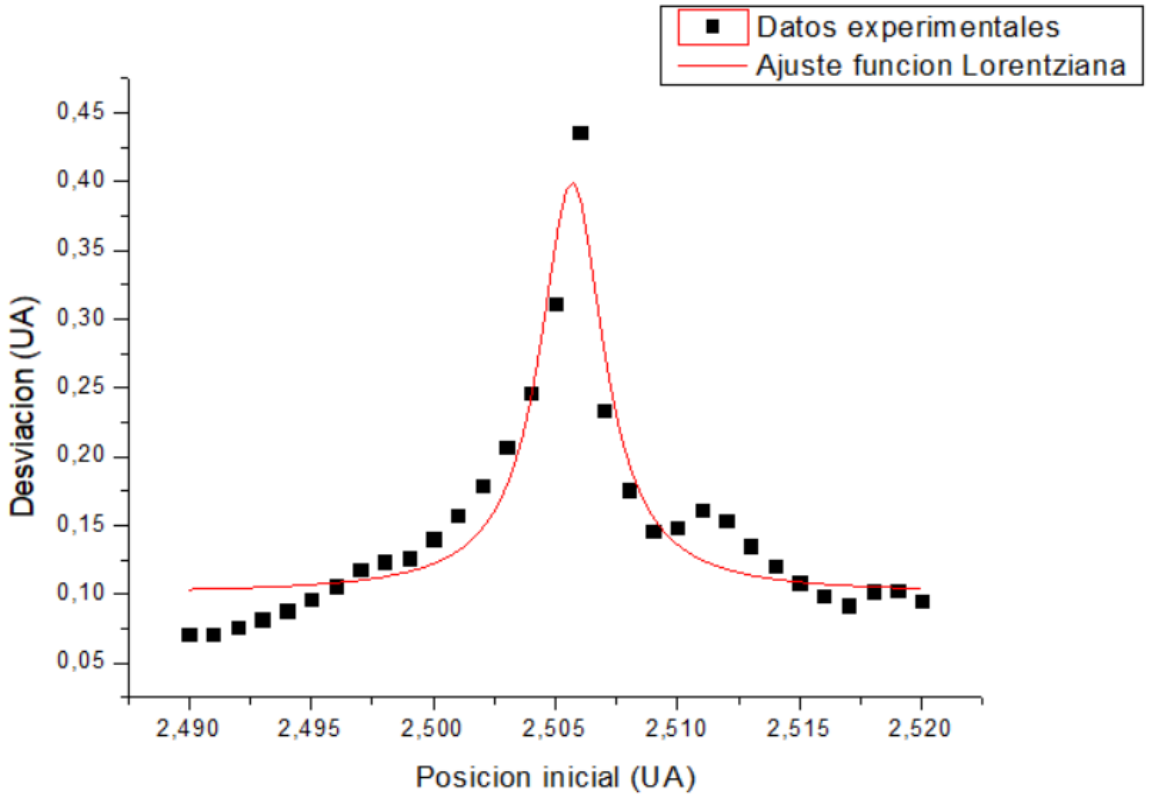


Figura 31: Desviación de un asteroide (en resonancia 3:1 con Júpiter) en función de su posición inicial respecto al Sol, T=1000años. Los datos han sido ajustados a una Lorentziana

Centro de la lorentziana = 2.5056 ± 0.0001 UA

Anchura=0.0032 ± 0.0004 UA

Proximidades: 2.5040 y 2.5072 UA.

Zanja 2/1

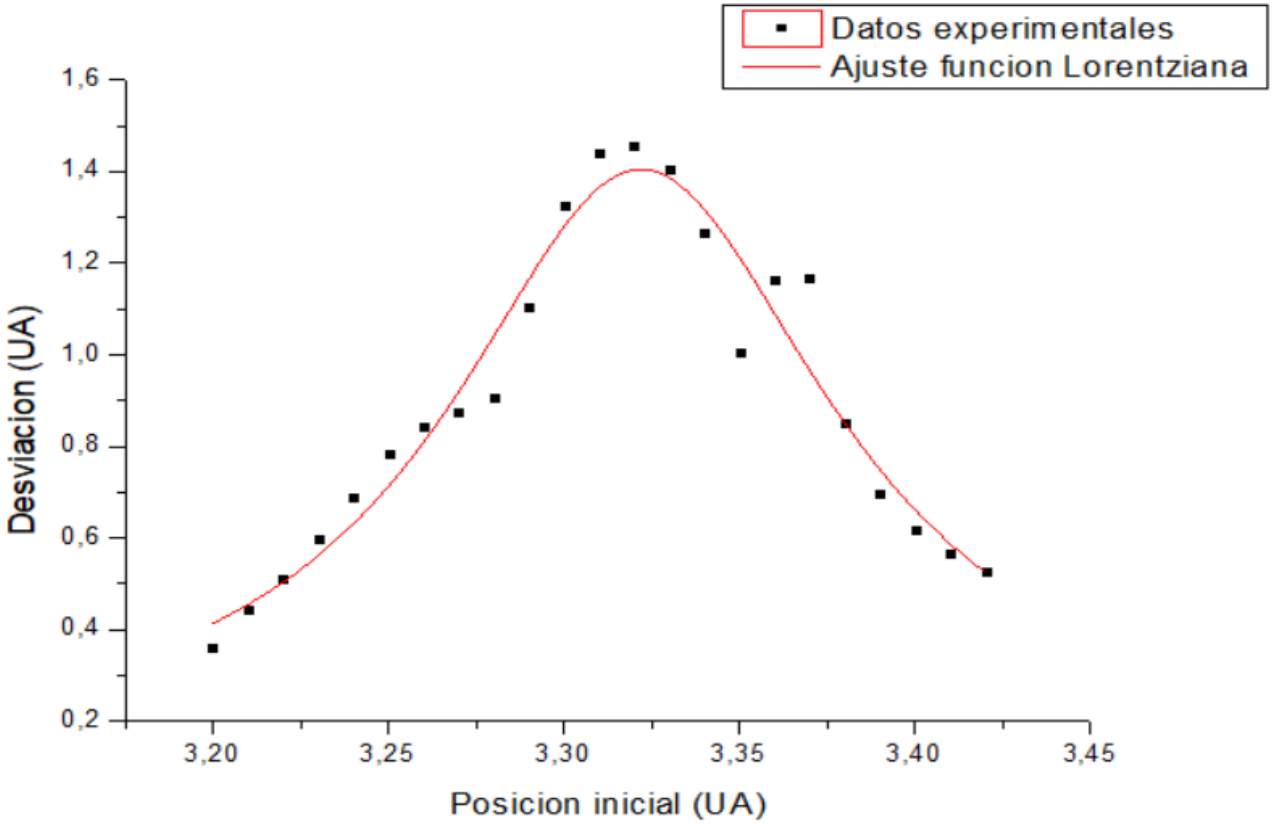


Figura 32: Desviación de un asteroide (en resonancia 2:1 con Júpiter) en función de su posición inicial respecto al Sol, T=1000 años. Los datos han sido ajustados a una Lorentziana

Centro de la lorentziana = 3.322 ± 0.002 UA

Anchura=0.13 ± 0.01 UA

Proximidades: 3.25 y 3.39 UA.

Influencia de Marte

$$F_{Gx} = -\frac{G \cdot M_S \cdot M_a}{r_{aS}^2} \cdot \mathrm{x} - \frac{G \cdot M_J \cdot M_a}{{r_{aJ}}^3} \cdot (x_a - x_J) - \frac{G \cdot M_M \cdot M_a}{{r_{aM}}^3} \cdot (x_a - x_M)$$

$$F_{Gy} = -\frac{G \cdot M_S \cdot M_a}{r_{aS}^2} \cdot \mathrm{y} - \frac{G \cdot M_J \cdot M_a}{{r_{aJ}}^3} \cdot (y_a - y_J) - \frac{G \cdot M_M \cdot M_a}{{r_{aM}}^3} \cdot (y_a - y_M)$$

- Tiempo = 100 000 años
- Masa Marte = 6.41 $\cdot 10^{23}$ kg
- Masa del Sol= 1,98 10^{30} kg
- Masa de Júpiter=1,89 $\cdot 10^{27}$ kg
- Semieje mayor=1,52UA

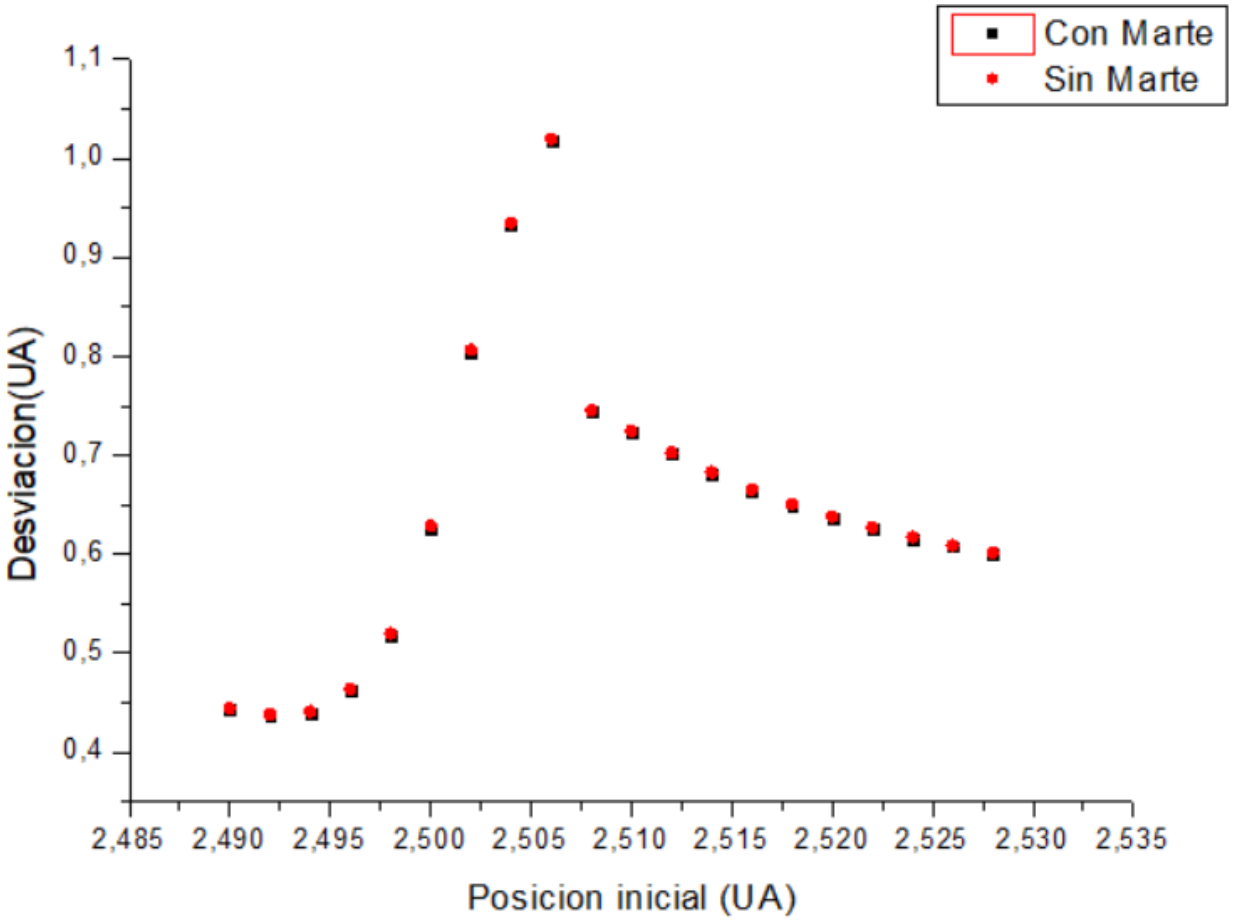


Figura 33: Comparación de la desviación de un asteroide en función de su posición inicial respecto al Sol al cabo de 100 000 años con la presencia de Marte y sin la presencia de Marte.

Influencia de Saturno

- Tiempo = 100 000 años
- Masa Saturno = 5.98 $\cdot 10^{26}$ kg
- Masa del Sol= 1,98 10^{30} kg
- Masa de Júpiter=1,89 $\cdot 10^{27}$ kg

- Semieje mayor= 9,58 UA

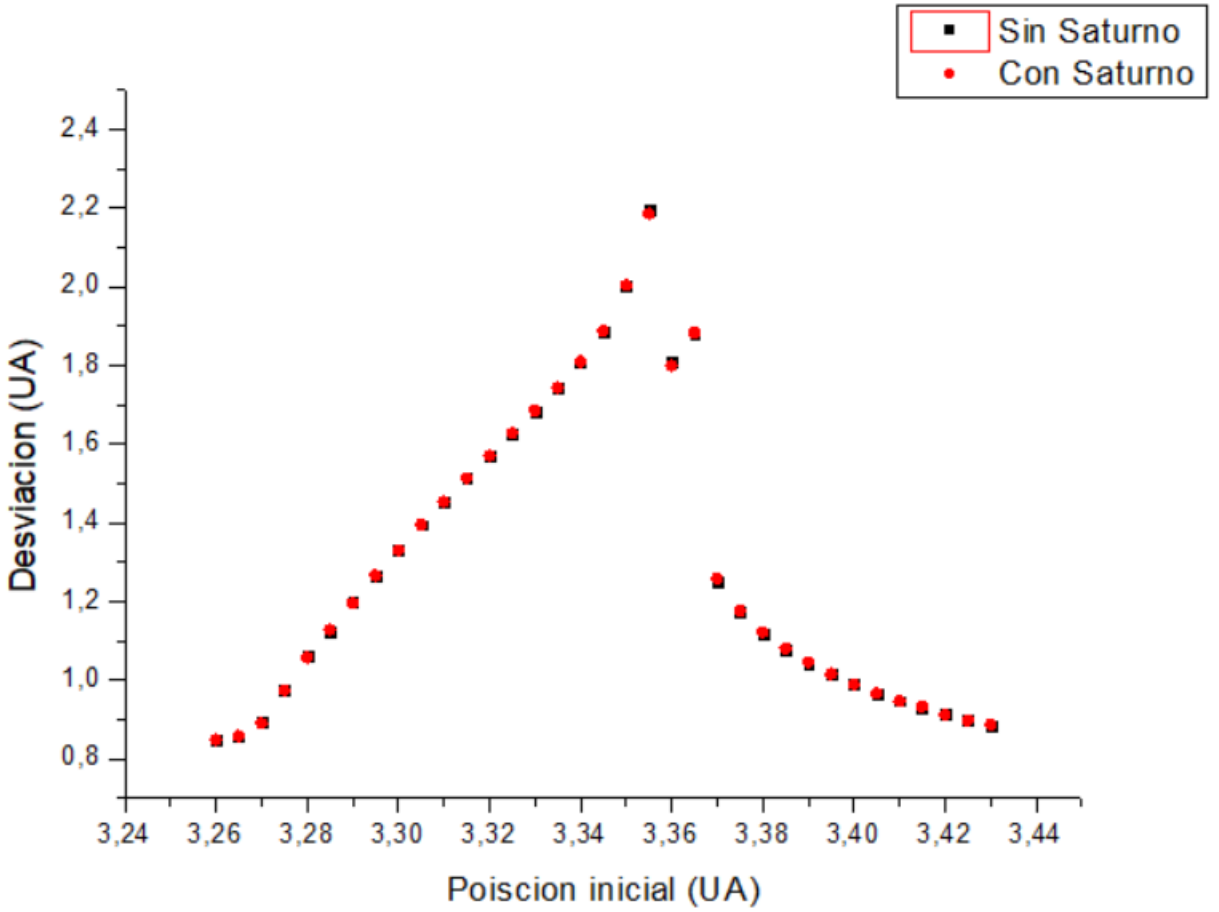


Figura 34: Comparación de la desviación de un asteroide en función de su posición inicial respecto del Sol al cabo de 100 000 años con la presencia de Saturno y sin la presencia de Saturno

Conclusiones

- Existen unas regiones entre Marte y Júpiter en las cuales los asteroides exhiben un comportamiento inusual.
- Con el código y el método desarrollado ha sido posible determinar la anchura aproximada de las Zanjas
- De los métodos estudiados el método de Velocidad Verlet es el más adecuado para el cálculo de órbitas.
- Órbitas elípticas →Más Zanjas de Kirkwood
- La influencia de Marte y Saturno en los asteroides que se encuentran en las Zanjas de Kirkwood es despreciable.

Gracias por su atención

Órbitas elípticas

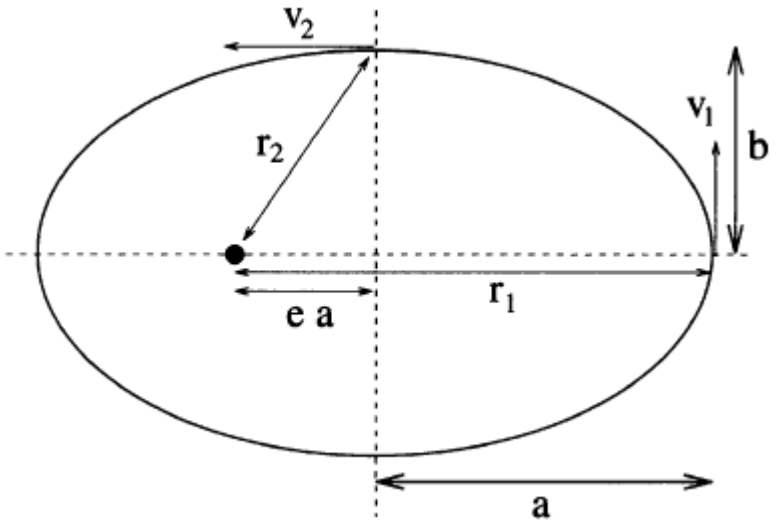


Figura 3: Órbita elíptica alrededor del Sol.

Bajo la acción de fuerza conservativa →Conservación de la energía mecánica y momento angular

$$\frac{1}{2}\,\mathrm{M_p}\,v_1^2 - \frac{\mathrm{GM_sM_p}}{r_1} = -\frac{\mathrm{GM_sM}_p}{r_2} + \frac{1}{2}\,\mathrm{M_p}\,v_2^2$$

$$\mathrm{M}_p v_1 r_1 = \mathrm{M}_p v_2 r_2$$

$$\boldsymbol{V_{max} = \sqrt{\frac{GM(1+e)\cdot(1+\frac{M_{jup}}{Msun})}{a(1-e)}}} \qquad \boldsymbol{V_{min} = \sqrt{\frac{GM(1-e)\cdot(1+\frac{M_{jup}}{Msun})}{a(1+e)}}}$$

Runge-Kutta 4

Para hallar la posición y la velocidad hay que calcular los siguientes coeficientes primero

$$x_{n+1} = x_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Donde:

$$k_1 = f(t_n, v_n)$$

$$k_2 = f\left(t_n + \frac{\Delta t}{2}, v_n + \frac{\Delta t}{2}k_1\right)$$

$$k_3 = f\left(t_n + \frac{\Delta t}{2}, v_n + \frac{\Delta t}{2}k_2\right)$$

$$k_4 = f(t_n + \Delta t, v_n + \Delta t\, k_3)$$

```
134 def RungeK_42(a,r0,v0,dt,tend,x_jup_RK4, y_jup_RK4 ,z_jup_RK4):
135 
136 #Creamos vectores de las posiciones, tiempo y velocidades
137 
138     x_vector = []
139     y_vector = []
140     z_vector = []
141 
142     t_vector = []
143 
144     vx_vector = []
145     vy_vector = []
146     vz_vector = []
147 
148 #Condiciones iniciales
149 
150     x = r0[0]
151     y = r0[1]
152     z = r0[2]
153     t=0          #Tiempo a partir del cual empezamos a contar
154     vx = v0[0]
155     vy = v0[1]
156     vz = v0[2]
157 
158 #Decimos que el primer elemento de nuestros vectores sea:
159 
160     x_vector.append(x)
161     y_vector.append(y)
162     z_vector.append(z)
163 
164     vx_vector.append(vx)
165     vy_vector.append(vy)
166     vz_vector.append(vz)
167 
168     t_vector.append(t)
169 
170     n=int(tend/dt)      #Numero de elementos que contendran nuestros vectores
```

```
    for i in range (0,n):

#Las  k de las velocidades que las denotaremos como k1vx,k1vy,k1vz
#Las k para las posiciones las denotaremos como k1x, k1y,k1z

#Ecuaciones Rungekutta4

        [k1vx, k1vy, k1vz] = a(x, y, z,x_jup_RK4[i], y_jup_RK4[i] ,z_jup_RK4[i])

        k1x = vx
        k1y = vy
        k1z = vz

        [k2vx, k2vy, k2vz] = a(x + k1x * (dt / 2), y + k1y * (dt / 2), z + k1z * (dt / 2),x_jup_RK4[i], y_jup_RK4[i] ,z_jup_RK4[i])

        k2x = vx + k1vx * (dt / 2)
        k2y = vy + k1vy * (dt / 2)
        k2z = vz + k1vz * (dt / 2)

        [k3vx, k3vy, k3vz] = a(x + k2x * (dt / 2), y + k2y * (dt / 2), z + k2z * (dt / 2),x_jup_RK4[i], y_jup_RK4[i] ,z_jup_RK4[i])

        k3x = vx + k2vx * (dt / 2)
        k3y = vy + k2vy * (dt / 2)
        k3z = vz + k2vz * (dt / 2)

        [k4vx, k4vy, k4vz] = a(x + k3x * dt, y + k3y * dt, z + k3z * dt,x_jup_RK4[i], y_jup_RK4[i] ,z_jup_RK4[i])

        k4x = vx + k3vx * dt
        k4y = vy + k3vy * dt
        k4z = vz + k3vz * dt

#Finalmente obtenemos las velocidades y posiciones siguientes

        vx_next = vx + (dt / 6) * (k1vx + 2 * k2vx + 2 * k3vx + k4vx)
        vy_next = vy + (dt / 6) * (k1vy + 2 * k2vy + 2 * k3vy + k4vy)
        vz_next = vz + (dt / 6) * (k1vz + 2 * k2vz + 2 * k3vz + k4vz)

        x_next = x + (dt / 6) * (k1x + 2 * k2x + 2 * k3x + k4x)
        y_next = y + (dt / 6) * (k1y + 2 * k2y + 2 * k3y + k4y)
        z_next = z + (dt / 6) * (k1z + 2 * k2z + 2 * k3z + k4z)
```

```
        x_vector.append(x_next)            # Guardamos los valores en nuestros vectores
        y_vector.append(y_next)
        z_vector.append(z_next)

        vx_vector.append(vx_next)
        vy_vector.append(vy_next)
        vz_vector.append(vz_next)

        t_next = t + dt

        t_vector.append(t_next)

        x = x_next                    # Actualizamos los valores  para realizar de nuevo el bucle
        y = y_next
        z = z_next

        t = t_next

        vx = vx_next
        vy = vy_next
        vz = vz_next

    return t_vector,x_vector, y_vector, z_vector, vx_vector, vy_vector, vz_vector
```

Comparación de los métodos

```
start1=timer()
t_jup_E, x_jup_E, y_jup_E, z_jup_E, vx_jup_E, vy_jup_E, vz_jup_E = Euler (grav_sun , r0_jup, v0_jup, deltat,tend)
end1=timer()
start2=timer()
t_jup_EC, x_jup_EC, y_jup_EC, z_jup_EC, vx_jup_EC, vy_jup_EC, vz_jup_EC = Euler_Cromer (grav_sun , r0_jup, v0_jup, deltat,tend)
end2=timer()
start3=timer()
t_jup_V, x_jup_V, y_jup_V, z_jup_V, vx_jup_V, vy_jup_V, vz_jup_V = Verlet (grav_sun , r0_jup, v0_jup, deltat,tend)
end3=timer()
start4=timer()
t_jup_RK4, x_jup_RK4, y_jup_RK4, z_jup_RK4, vx_jup_RK4, vy_jup_RK4, vz_jup_RK4 =  RungeK_4 (grav_sun , r0_jup, v0_jup, deltat,tend)
end4=timer()
```

```
for i in range(n): #No importan las unidades, queremos ver si varia
    v[i] =sqrt( vx[i] ** 2 + vy[i] ** 2 + vz[i] ** 2)
    Energia_cinet[i]= 0.5*v[i] ** 2 * Mplaneta1
    r[i] =sqrt( x[i] ** 2 + y[i] ** 2 + z[i] ** 2)
    Energia_grav[i] = - 4*pi**2*Mplaneta1/ r[i]
    Energia_mec[i]= Energia_grav[i] + Energia_cinet[i]
```

```
if fabs(x_jup_V[i]-x_jup_V[0])<tol:
    T1_jup=T2_jup
    T2_jup = deltat * i
    T_jup1 =T2_jup-T1_jup
    T_jup.append(T_jup1)
    x_afel.append(fabs(x_jup_V[i]-x_jup_V[0]))
```

Cálculo Afelio

```
if r_afel[i]>r_afel[i-1] and r_afel[i]>r_afel[i+1]:
    rmaxV.append(r_afel[i])
```

Figura 14: Calculo del afelio de Juno

```
#%% Comparacion de los metodos en el afelio

n=int(tend/deltat)

#Convertimos en vectores las listas de Verlet
x1=np.array(x_ast1_V,float)
y1=np.array(y_ast1_V,float)
r_afel=(x1**2+y1**2)**(1/2)

#Convertimos en vectores las listas de Euler Cromer
x2=np.array(x_ast1_EC,float)
y2=np.array(y_ast1_EC,float)
r_afel2=(x2**2+y2**2)**(1/2)
#Convertimos en vectores las listas de RK4
x3=np.array(x_ast1_RK4,float)
y3=np.array(y_ast1_RK4,float)
r_afel3=(x3**2+y3**2)**(1/2)

#Hacemos el bucle para hallar el afelio
rmaxV=[]
rmaxEC=[]
rmaxRK4=[]

for i in range(0,n):
    if r_afel[i]>r_afel[i-1] and r_afel[i]>r_afel[i+1]:
        rmaxV.append(r_afel[i])
    if r_afel2[i]>r_afel2[i-1] and r_afel2[i]>r_afel2[i+1]:
        rmaxEC.append(r_afel2[i])
    if r_afel3[i]>r_afel3[i-1] and r_afel3[i]>r_afel3[i+1]:
        rmaxRK4.append(r_afel3[i])

#Los convertimos en vectores  para poder aplicar las funciones mean y stdev
rV=np.array(rmaxV,float)
rEC=np.array(rmaxEC,float)
rRK4=np.array(rmaxRK4,float)

#Calculamos las desviaciones
print('La media con Verlet es=',rV.mean())
print('La desviacion respecto al valor real',stats.stdev(rV))
print('La media con Euler Cromer es=',rEC.mean())
print('La desviacion respecto al valor real',stats.stdev(rEC))

print('La media con RK4 es=',rRK4.mean())
print('La desviacion respecto al valor real',stats.stdev(rRK4))
```

```
#Pintamos los valores en una recta

a=[stats.stdev(rV),stats.stdev(rV)]
b=[stats.stdev(rEC),stats.stdev(rEC)]
c=[stats.stdev(rRK4),stats.stdev(rRK4)]

d=[0,1]
figure(2)
plt.plot( d,b,'g', label='Euler Cromer ')
plt.plot(d,a,'y', label='Verlet Velocity')
plt.plot( d,c,'r', label='RungeK_4')
plt.ylabel('Desviacion(UA)')
plt.plot(d,[0,0],'b',label='Valor observado')

plt.legend(loc='center')
show()
```

Estudio Zanjas de Kirkwood

```
for radius in np.arange(2.2,3.45,0.001):

    V_i = sqrt(4 * pi ** 2 / (radius))

    Init_pos = [radius, 0, 0]
    V_init = [0, V_i, 0]

    t_ast, x_ast, y_ast, z_ast, vx_ast, vy_ast, vz_ast =\
    Verlet2 (grav_sun_Jupiter , Init_pos , V_init, deltat,tend,x_jup , y_jup , z_jup )

    x_ast = np.array(x_ast,float)
    y_ast = np.array(y_ast,float)
    r_ast = ( x_ast**2 + y_ast**2 ) ** (1/2)
    desviacion=abs(max(r_ast)-min(r_ast))
```

Inclinación→ plano de la eclíptica

- Tierra: 0º
- Marte: 1,85º
- Júpiter: 1,30º
- Saturno 2,48º