UNIVERSIDADE DE SÃO PAULO - USP

INTRODUÇÃO À FÍSICA COMPUTACIONAL – 7600017 – 2025/2
PROF. FRANCISCO C. ALCARAZ

RELATÓRIO DO 5° PROJETO

JOÃO VITOR LIMA DE OLIVEIRA - 12694394

São Carlos

2025

Parte I

Introdução Geral

MOTIVAÇÃO

O estudo do movimento dos planetas é muito importante na Física, pois nos ajuda a entender como os corpos celestes interagem e se movem no espaço. A força da gravidade determina as órbitas e, ao analisar esse movimento, podemos compreender melhor o funcionamento do Sistema Solar.

Neste projeto, o objetivo é estudar tanto o caso simples de um planeta orbitando o Sol quanto situações mais complexas envolvendo vários corpos ao mesmo tempo. Usando métodos numéricos, como o método de Verlet, podemos simular diferentes condições iniciais e observar como pequenas mudanças na velocidade ou na posição modificam o formato das órbitas. Assim, é possível verificar as Leis de Kepler e entender por que algumas órbitas são circulares, outras são elípticas e algumas podem variar ao longo do tempo.

Quando mais corpos são incluídos, como a Terra e Júpiter juntos, o movimento se torna menos previsível e mais sensível a perturbações. A órbita da Terra, por exemplo, deixa de ser exatamente periódica quando a influência de Júpiter é considerada. Da mesma forma, o estudo de asteroides mostra que algumas regiões do Sistema Solar são estáveis enquanto outras apresentam lacunas causadas pelas interações gravitacionais.

Essas simulações mostram como a Física Computacional é essencial para explorar fenômenos que seriam muito difíceis de resolver apenas com contas analíticas. Mesmo sistemas que seguem leis simples podem apresentar comportamentos complexos. Por isso, a computação é uma ferramenta fundamental para investigar e compreender melhor esse tipo de problema.

Parte II

Desenvolvimento

TAREFA - A

CÓDIGO

Figura 1 – Função principal do código.

```
1              program main
2                 implicit real*8 (a-h,o-z)
3                 a = 39.53d0
4                 call calc(a)
5                 end program main
```

Fonte: Compilado pelo Autor.

Figura 2 – Subrotina que realiza os cálculos, para um certo valor de a.

```
1
2  subroutine calc(a)
3  parameter (imax=1e5)
4  implicit real*8 (a-h,o-z)
5  dimension x(-1:imax),y(-1:imax)
6
7  C   Constantes
8  pi = acos(-1d0)
9  dt = 10d0/365d0
10 GM = 4*pi*pi
11
12 C   Val in
13 x(0) = 1d0*a !Distancia em UA
14 y(0) = 0d0
15 vx = 0d0
16 vy = 2.0d0*pi/sqrt(a)
17
18
19 C  x(-1) e y(-1)
20 x(-1) = x(0) - vx*dt
21 y(-1) = y(0) - vy*dt
22
23 C       Salva para lei de Kepler
24 io = 0d0 !Variavel de segurança para pegar apenas a primeira volta
```

Fonte: Compilado pelo Autor.

Figura 3 – Subrotina que realiza os cálculos, para um certo valor de a.

```
1                   C Calc
2                   do i = 0,imax-1
3                   r = sqrt((x(i)**2) + (y(i)**2))
4
5                   ax = - GM*x(i)/(r**3)
6                   ay = - GM*y(i)/(r**3)
7
8                   x(i+1) = 2d0*x(i) - x(i-1) + ax*dt*dt
9                   y(i+1) = 2d0*y(i) - y(i-1) + ay*dt*dt
10
11                  theta_new = atan2(y(i+1), x(i+1))
12                  if ((theta_old .LT. 0d0) .and. (theta_new .GE. 0d0))
                        then
13                  if (io .GT. 0d0) then
14                  goto 7
15                  end if
16
17                  if (i*dt .GT. 5d0*dt) then      ! para n detectar o
                        comeco
18                  periodo = i * dt
19                  write(*,3) periodo, r, (periodo**2)/(r**3)
20                  3                          format(F12.4,2(",",F12.4))
21                  end if
22                  io = 1d0
23                  end if
24
25                  theta_old = theta_new
26                  7       continue
27                  end do
28
29                  C   Salva
30                  open(unit=1,file='saida-2-12694394.txt')
31                  do i = 0,imax
32
33                  write(1,2) dt*i,x(i),y(i)
34
35                  end do
36                  2       format(F16.8,2(",",F16.8))
37                  close(1)
38
39
40                  end subroutine calc
```

Fonte: Compilado pelo Autor.

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como finalidade calcular numericamente a órbita de um corpo sob atração gravitacional central, utilizando um integrador de segunda ordem para resolver as equações do movimento.

Ele utiliza o comando:

```
1         implicit real*8 (a-h,o-z)
```

o que define como números reais de dupla precisão todas as variáveis cujos nomes se iniciam com as letras **a**–**h** e **o**–**z**.

Em seguida, o programa principal define o parâmetro:

```
1         a = 1d0
```

que representa o raio inicial da órbita em unidades astronômicas (UA). Por fim, chama a subrotina responsável pelos cálculos:

```
1         call calc(a)
```

Subrotina `calc`

A subrotina `calc` recebe o parâmetro `a` e define o número máximo de iterações como:

```
1         parameter (imax = 1e4)
```

Os vetores `x(-1:imax)` e `y(-1:imax)` armazenam as posições cartesianas do corpo ao longo da integração.

As constantes físicas utilizadas são:

```
1         pi = acos(-1d0)
2         dt = 1d0/365d0
3         GM = 4*pi*pi
4         ec = 0.5d0
```

O passo temporal `dt` corresponde a um dia em unidades de ano, e `GM` é expresso de modo que sistemas keplerianos possam ser simulados em unidades astronômicas sem necessidade de conversões adicionais. O parâmetro `ec` representa o fator de controle da velocidade inicial, permitindo estudar órbitas com diferentes excentricidades.

Condições iniciais

As condições iniciais de posição são definidas como:

$$x(0) = a, \qquad y(0) = 0 \tag{1}$$

e a velocidade inicial é exclusivamente tangencial:

$$v_x = 0, \qquad v_y = ec\,\frac{2\pi}{\sqrt{a}} \tag{2}$$

Além disso, como o método numérico utilizado requer um passo anterior, os valores de `x(-1)` e `y(-1)` são estimados por:

```
1         x(-1) = x(0) - vx*dt
2         y(-1) = y(0) - vy*dt
```

Esses valores representam uma aproximação da posição um passo antes do instante inicial.

Método numérico

A integração é realizada pelo método de Verlet, dado por:

$$x_{i+1} = 2x_i - x_{i-1} + a_x\,dt^2, \qquad y_{i+1} = 2y_i - y_{i-1} + a_y\,dt^2 \tag{3}$$

com as acelerações calculadas pela lei da gravitação universal:

$$a_x = -\frac{GMx}{r^3}, \qquad a_y = -\frac{GMy}{r^3}, \qquad r = \sqrt{x^2 + y^2} \tag{4}$$

Cálculo da área varrida

Em cada passo da simulação, a área varrida pelo vetor de posição é aproximada pela fórmula:

$$A_i = \frac{1}{2}\,|x_i\,y_{i+1} - x_{i+1}\,y_i| \tag{5}$$

e é registrada no arquivo:

`saida-1-12694394.txt`

Esse cálculo permite verificar a segunda lei de Kepler, que estabelece que a área varrida por unidade de tempo permanece constante.

Cálculo do período orbital

Para determinar o período, o código monitora o ângulo polar:

$$\theta = \operatorname{atan2}(y, x) \tag{6}$$

e identifica quando o corpo cruza novamente o eixo x no semi-eixo positivo. A condição usada é:

$$\theta_{\text{old}} < 0 \quad \text{e} \quad \theta_{\text{new}} \geq 0 \tag{7}$$

Após ignorar as primeiras iterações, que poderiam detectar o instante inicial, o período é calculado por:

$$T = i\,dt \tag{8}$$

Além disso, são impressos o raio e a razão:

$$\frac{T^2}{r^3} \tag{9}$$

permitindo verificar numericamente a terceira lei de Kepler.

Gravação da trajetória

Ao final da simulação, os valores de tempo e posição são armazenados em:

`saida-2-12694394.txt`

no formato:

$$t,\ x(t),\ y(t) \tag{10}$$

Resumo

O código implementa:

- um integrador de segunda ordem do tipo Verlet;

- condições iniciais ajustáveis via parâmetro `a`;
- determinação automática do período orbital;
- verificação da terceira lei de Kepler;
- armazenamento da trajetória completa em arquivo.

Trata-se de uma implementação eficiente para o estudo de órbitas keplerianas e propriedades fundamentais do movimento sob gravitação central.

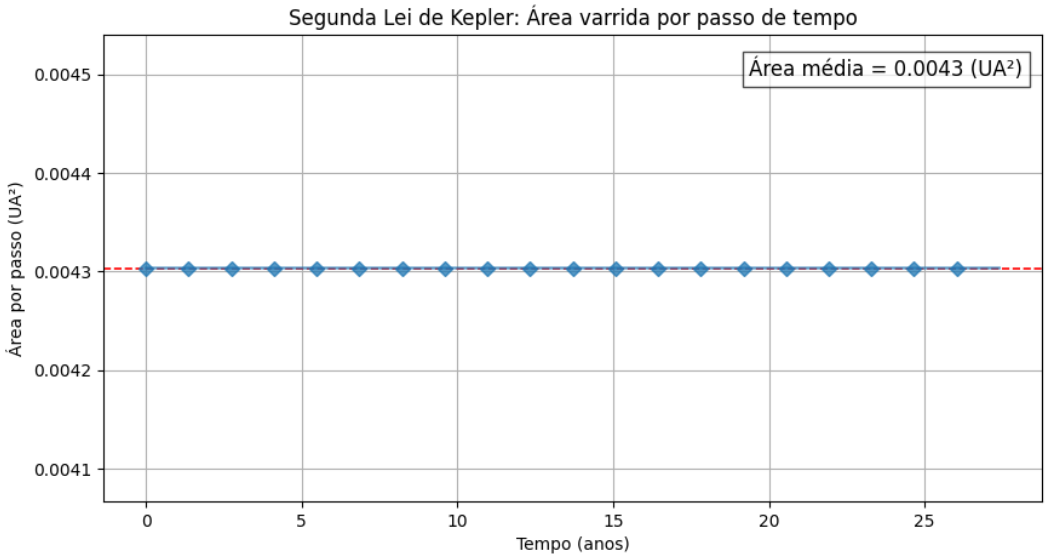
RESULTADOS

Tabela 1 – Período orbital, raio e razão T^2/a^3 dos planetas.

Planeta	Período (anos)	Raio (UA)	T^2/a^3
Mercúrio	0.2436	0.3900	1.0001
Vênus	0.6107	0.7200	0.9992
Terra	1.0000	1.0000	1.0000
Marte	1.8740	1.5200	1.0000
Júpiter	11.8578	5.2000	1.0000
Saturno	28.0871	9.2400	1.0000
Urano	84.0548	19.1900	0.9998
Netuno	164.7945	30.0600	0.9998
Plutão	248.5205	39.5300	0.9999

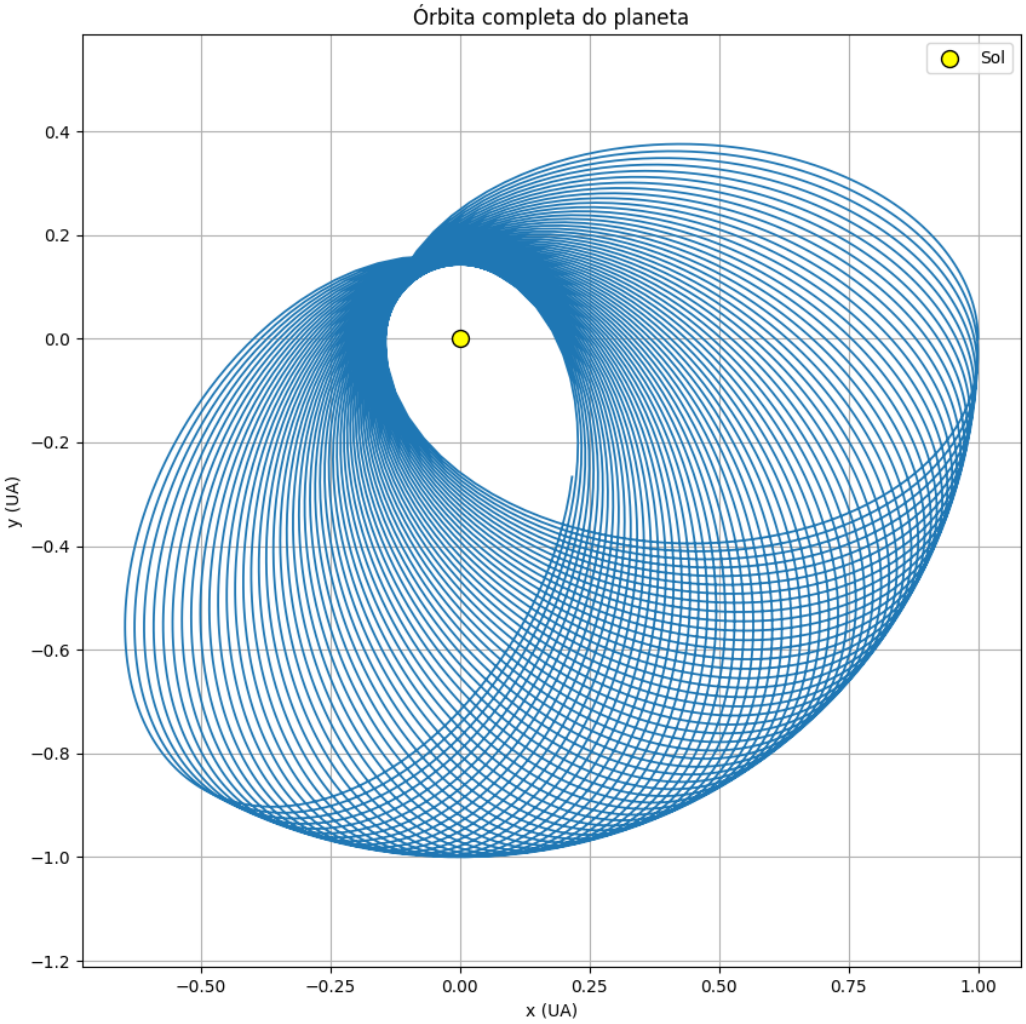
Fonte: Compilado pelo Autor

Figura 4 – Área varrida por um planeta entre as interações i e $i + 1$.



Fonte: Compilado pelo Autor.

Figura 5 – Exemplo de órbita de um planeta.



Fonte: Compilado pelo Autor.

DISCUSSÃO DOS RESULTADOS

TAREFA - B

CÓDIGO

Listing 1 – Função principal do código.

```
1         program main
2         implicit real*8 (a-h,o-z)
3
4         call calc()
5         call calc2()
6
7         end program main
```

Listing 2 – Implementação da subrotina `calc`: Declarações, constantes e condições iniciais (parte 1 de 3).

```
1         subroutine calc()
2         implicit real*8 (a-h,o-z)
3         parameter (imax=5e3)
4
5         dimension x_t(-1:imax), y_t(-1:imax)
6         dimension x_j(-1:imax), y_j(-1:imax)
7
8         C          Constantes
9
10        pi = acos(-1d0)
11        dt = 1d0/365d0
12
13        GM_S = 4d0*pi*pi
14        GM_J = GM_S * 0.0009543d0 * 1     ! multiplica a massa
15        GM_T = GM_S * 3.003d-6
16
17        C
18        GM_S2 = GM_S + GM_J
19
20        C          C.I.
21
22        x_t(0) = 1d0
23        y_t(0) = 0d0
24        vx_t   = 0d0
25        vy_t   = 2d0*pi
26
27        x_j(0) = 5.2d0
28        y_j(0) = 0d0
29        vx_j   = 0d0
```

```
30          vy_j   = 2d0*pi*5.2d0/11.86d0
```

Listing 3 – Sub-rotina `calc`: Interação negativa e loop principal de cálculo (parte 2 de 3).

```
1           C          Intereaçao negativa
2           x_t(-1) = x_t(0) - vx_t*dt
3           y_t(-1) = y_t(0) - vy_t*dt
4
5           x_j(-1) = x_j(0) - vx_j*dt
6           y_j(-1) = y_j(0) - vy_j*dt
7
8           C          Contas
9           do i = 0, imax-1
10
11          C          Acel. Terra
12          r_ts = sqrt(x_t(i)**2 + y_t(i)**2)
13          r_tj = sqrt((x_t(i)-x_j(i))**2 + (y_t(i)-y_j(i))**2)
14
15          ax_t = -GM_S*x_t(i)/(r_ts**3) - GM_J*(x_t(i)-x_j(i))/(r_tj**3)
16          ay_t = -GM_S*y_t(i)/(r_ts**3) - GM_J*(y_t(i)-y_j(i))/(r_tj**3)
17
18          C          Acel jupiter
19          r_js = sqrt(x_j(i)**2 + y_j(i)**2)
20
21          ax_j = -GM_S2*x_j(i)/(r_js**3)
22          ay_j = -GM_S2*y_j(i)/(r_js**3)
23
24          C          atualiza os dados
25          x_t(i+1) = 2d0*x_t(i) - x_t(i-1) + ax_t*dt*dt
26          y_t(i+1) = 2d0*y_t(i) - y_t(i-1) + ay_t*dt*dt
27
28          x_j(i+1) = 2d0*x_j(i) - x_j(i-1) + ax_j*dt*dt
29          y_j(i+1) = 2d0*y_j(i) - y_j(i-1) + ay_j*dt*dt
30
31          end do
```

Listing 4 – Sub-rotina `calc`: Salvamento dos dados calculados em arquivo (parte 3 de 3).

```
1           C          Salva
2
3           open(unit=1, file='saida_m_1.txt')
4           do i = 0, imax
5           write(1,2) dt*i, x_t(i), y_t(i), x_j(i), y_j(i)
6           end do
```

```
7           2       format(F16.8, 4(",",F16.8))
8           close(1)
9
10          end subroutine calc
```

Listing 5 – Sub-rotina `calc2`: Declarações de variáveis e constantes (parte 1 de 4).

```
1           subroutine calc2()
2           implicit real*8 (a-h,o-z)
3           parameter (imax=1e3)
4
5           dimension xj(-1:imax), yj(-1:imax)
6           dimension xa1(-1:imax), ya1(-1:imax)
7           dimension xa2(-1:imax), ya2(-1:imax)
8           dimension xa3(-1:imax), ya3(-1:imax)
9
10          C          Constantes
11
12          pi     = acos(-1d0)
13          dt     = 1d0/365d0
14
15          GM_S = 4d0*pi*pi
16          GM_J = GM_S * 0.0009543d0
17          GM_S2 = GM_S + GM_J !Isso é para jupiter sentir a mudamça de
               massa
```

Listing 6 – Sub-rotina `calc2`: Condições iniciais para Júpiter e asteroides (parte 2 de 4).

```
1           C          Cond Ini
2
3           C          Jupiter
4           xj(0) = 5.2d0
5           yj(0) = 0d0
6           vxj    = 0d0
7           vyj    = 2d0*pi*5.2d0/11.86d0
8
9           C          Asteroide 1
10          xa1(0) = 3.0d0
11          ya1(0) = 0d0
12          vxa1   = 0d0
13          vya1   = 3.628d0
14
15          C          Asteroide 2
16          xa2(0) = 3.276d0
17          ya2(0) = 0d0
18          vxa2   = 0d0
```

```
19          vya2    = 3.471d0
20          C           Asteroide 3
21          xa3(0)  = 3.700d0
22          ya3(0)  = 0d0
23          vxa3    = 0d0
24          vya3    = 3.267d0
```

Listing 7 – Sub-rotina `calc2`: Setup do método de Verlet e início do loop principal (parte 3 de 4).

```
1           C           Setup do i-1 do verlat
2
3           xj(-1) = xj(0) - vxj*dt
4           yj(-1) = yj(0) - vyj*dt
5
6           xa1(-1) = xa1(0) - vxa1*dt
7           ya1(-1) = ya1(0) - vya1*dt
8
9           xa2(-1) = xa2(0) - vxa2*dt
10          ya2(-1) = ya2(0) - vya2*dt
11
12          xa3(-1) = xa3(0) - vxa3*dt
13          ya3(-1) = ya3(0) - vya3*dt
14
15          C           Realiza as contas
16          do i = 0, imax-1
17
18          C           Jupiter
19          rjs = sqrt(xj(i)**2 + yj(i)**2)
20          axj = -GM_S2*xj(i)/(rjs**3)
21          ayj = -GM_S2*yj(i)/(rjs**3)
22
23          C           teroide 1
24          ras = sqrt(xa1(i)**2 + ya1(i)**2)
25          raj = sqrt((xa1(i)-xj(i))**2 + (ya1(i)-yj(i))**2)
26
27          axa1 = -GM_S*xa1(i)/(ras**3) - GM_J*(xa1(i)-xj(i))/(raj**3)
28          aya1 = -GM_S*ya1(i)/(ras**3) - GM_J*(ya1(i)-yj(i))/(raj**3)
29
30          C           teroide2
31          ras = sqrt(xa2(i)**2 + ya2(i)**2)
32          raj = sqrt((xa2(i)-xj(i))**2 + (ya2(i)-yj(i))**2)
33
34          axa2 = -GM_S*xa2(i)/(ras**3) - GM_J*(xa2(i)-xj(i))/(raj**3)
35          aya2 = -GM_S*ya2(i)/(ras**3) - GM_J*(ya2(i)-yj(i))/(raj**3)
36
37          C           teroide 3
```

```
38          ras = sqrt(xa3(i)**2 + ya3(i)**2)
39          raj = sqrt((xa3(i)-xj(i))**2 + (ya3(i)-yj(i))**2)
40
41          axa3 = -GM_S*xa3(i)/(ras**3) - GM_J*(xa3(i)-xj(i))/(raj**3)
42          aya3 = -GM_S*ya3(i)/(ras**3) - GM_J*(ya3(i)-yj(i))/(raj**3)
43
44          C          atualizaoção das posições usando verlat
45          xj(i+1) = 2*xj(i) - xj(i-1) + axj*dt*dt
46          yj(i+1) = 2*yj(i) - yj(i-1) + ayj*dt*dt
47
48          xa1(i+1) = 2*xa1(i) - xa1(i-1) + axa1*dt*dt
49          ya1(i+1) = 2*ya1(i) - ya1(i-1) + aya1*dt*dt
50
51          xa2(i+1) = 2*xa2(i) - xa2(i-1) + axa2*dt*dt
52          ya2(i+1) = 2*ya2(i) - ya2(i-1) + aya2*dt*dt
53
54          xa3(i+1) = 2*xa3(i) - xa3(i-1) + axa3*dt*dt
55          ya3(i+1) = 2*ya3(i) - ya3(i-1) + aya3*dt*dt
56
57          end do
```

Listing 8 – Sub-rotina `calc2`: Salvamento dos dados calculados em arquivos e finalização (parte 4 de 4).

```
1           C          Salva
2
3           open(unit=1,file='asteroides_saida.txt')
4           open(unit=2,file='asteroides_saida_2.txt')
5
6           do i = 0, imax
7           write(1,10) dt*i, xj(i), yj(i), xa1(i), ya1(i)
8           write(2,10) dt*i, xa2(i), ya2(i), xa3(i), ya3(i)
9           end do
10
11          10       format(F12.6,4(",",F12.6))
12
13          close(1)
14          close(2)
15
16          end subroutine calc2
```

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo resolver numericamente o problema de três corpos (Terra, Sol e Júpiter) e, em seguida, estudar as perturbações gravitacionais de

Júpiter sobre três asteroides localizados na região do cinturão de asteroides. Para isso, são utilizadas duas subrotinas independentes:

```
1         call calc()
2         call calc2()
```

Cada subrotina usa o método de Verlet para integrar as equações de movimento e grava os resultados em arquivos para posterior análise gráfica.

Subrotina `calc`: Terra–Sol–Júpiter

A primeira subrotina implementa o problema de três corpos envolvendo o Sol (fixo na origem), a Terra e Júpiter. O número máximo de iterações utilizado é:

```
1         parameter (imax = 5e3)
```

Foram definidos quatro vetores para armazenar as posições da Terra e de Júpiter:

$$\texttt{x\_t, y\_t, x\_j, y\_j}$$

todos indexados de -1 até `imax`, como requerido pelo método de Verlet.

Constantes físicas

Os parâmetros usados são definidos como:

```
1         pi = acos(-1d0)
2         dt = 1d0/365d0
3         GM_S = 4*pi*pi
4         GM_J = GM_S * 0.0009543d0
5         GM_T = GM_S * 3.003d-6
6         GM_S_eff = GM_S + GM_J
```

onde:

- $GM_S = 4\pi^2$ é a constante gravitacional do Sol em unidades astronômicas,
- GM_J é o valor ajustado para a massa de Júpiter,
- GM_S^{eff} representa a massa efetiva sentida por Júpiter devido ao Sol fixo.

Condições iniciais

As condições iniciais da Terra e de Júpiter são:

$$x_T(0) = 1, \qquad y_T(0) = 0, \qquad v_{x,T} = 0, \qquad v_{y,T} = 2\pi, \tag{11}$$

$$x_J(0) = 5.2, \qquad y_J(0) = 0, \qquad v_{x,J} = 0, \qquad v_{y,J} = 2\pi\frac{5.2}{11.86}. \tag{12}$$

As acelerações iniciais incluem interações Terra–Sol e Terra–Júpiter:

$$\vec{a}_T = -\frac{GM_S}{r_{TS}^3}\vec{r}_{TS} - \frac{GM_J}{r_{TJ}^3}\vec{r}_{TJ}, \tag{13}$$

e, para Júpiter (com Sol fixo e massa efetiva):

$$\vec{a}_J = -\frac{GM_S^{\text{eff}}}{r_{JS}^3}\vec{r}_{JS}. \tag{14}$$

Passo inicial do método de Verlet

O método exige as posições no tempo $-\Delta t$, obtidas por:

$$x(-1) = x(0) - v_x\,dt + \frac{1}{2}a_x\,dt^2, \qquad y(-1) = y(0) - v_y\,dt + \frac{1}{2}a_y\,dt^2. \tag{15}$$

Integração numérica

A integração é realizada pelo método de Verlet:

$$x_{i+1} = 2x_i - x_{i-1} + a_x\,dt^2, \tag{16}$$

$$y_{i+1} = 2y_i - y_{i-1} + a_y\,dt^2. \tag{17}$$

As acelerações são recalculadas a cada passo considerando as forças:

- Terra–Sol,
- Terra–Júpiter,
- Júpiter–Sol (massa efetiva).

O laço principal avança ambas as órbitas simultaneamente.

Saída de dados

As coordenadas são gravadas no arquivo:

`saida_m_1.txt`

no formato:

$$t,\ x_T(t),\ y_T(t),\ x_J(t),\ y_J(t).$$

Isso permite analisar graficamente a órbita terrestre sob perturbações de Júpiter, atendendo aos itens **b1** e **b2** da tarefa.

Subrotina `calc2`: Asteroides sob a perturbação de Júpiter

A segunda subrotina implementa o item **b3** da tarefa, estudando a influência de Júpiter sobre três asteroides no cinturão principal.

O número de iterações é:

```
1       parameter (imax = 1000)
```

São definidos:

- posição de Júpiter: `xj, yj`,
- três asteroides: `xa1, ya1`, `xa2, ya2`, `xa3, ya3`.

Condições iniciais

As condições de Júpiter são:

$$x_J(0) = 5.2, \qquad y_J(0) = 0, \qquad v_{y,J} = 2.755. \tag{18}$$

Os asteroides são inicializados conforme os dados do enunciado:

$$(x_{A1}, y_{A1}, v_{y,A1}) = (3.0,\ 0,\ 3.628), \tag{19}$$
$$(x_{A2}, y_{A2}, v_{y,A2}) = (3.276,\ 0,\ 3.471), \tag{20}$$
$$(x_{A3}, y_{A3}, v_{y,A3}) = (3.700,\ 0,\ 3.267). \tag{21}$$

Dinâmica gravitacional

Cada asteroide sofre duas forças:

$$\vec{F} = -\frac{GM_S}{r_{AS}^3}\vec{r}_{AS} - \frac{GM_J}{r_{AJ}^3}\vec{r}_{AJ}. \tag{22}$$

O efeito dos asteroides sobre Júpiter é desprezado, conforme pedido.

Integração

O mesmo esquema de Verlet é utilizado:

$$x_{i+1} = 2x_i - x_{i-1} + a_x\,dt^2, \tag{23}$$
$$y_{i+1} = 2y_i - y_{i-1} + a_y\,dt^2. \tag{24}$$

Todos os quatro corpos (Júpiter + 3 asteroides) são atualizados em cada iteração.

Saída de dados

Dois arquivos são produzidos:

- `asteroides_saida.txt`: Júpiter + Asteroide I.
- `asteroides_saida_2.txt`: Asteroides II e III.

O formato de saída é:

$$t,\ x(t),\ y(t),\ x'(t),\ y'(t).$$

Esses dados permitem identificar ressonâncias orbitais e lacunas de Kirkwood, conforme a tarefa solicita.

RESUMO

O código implementa:

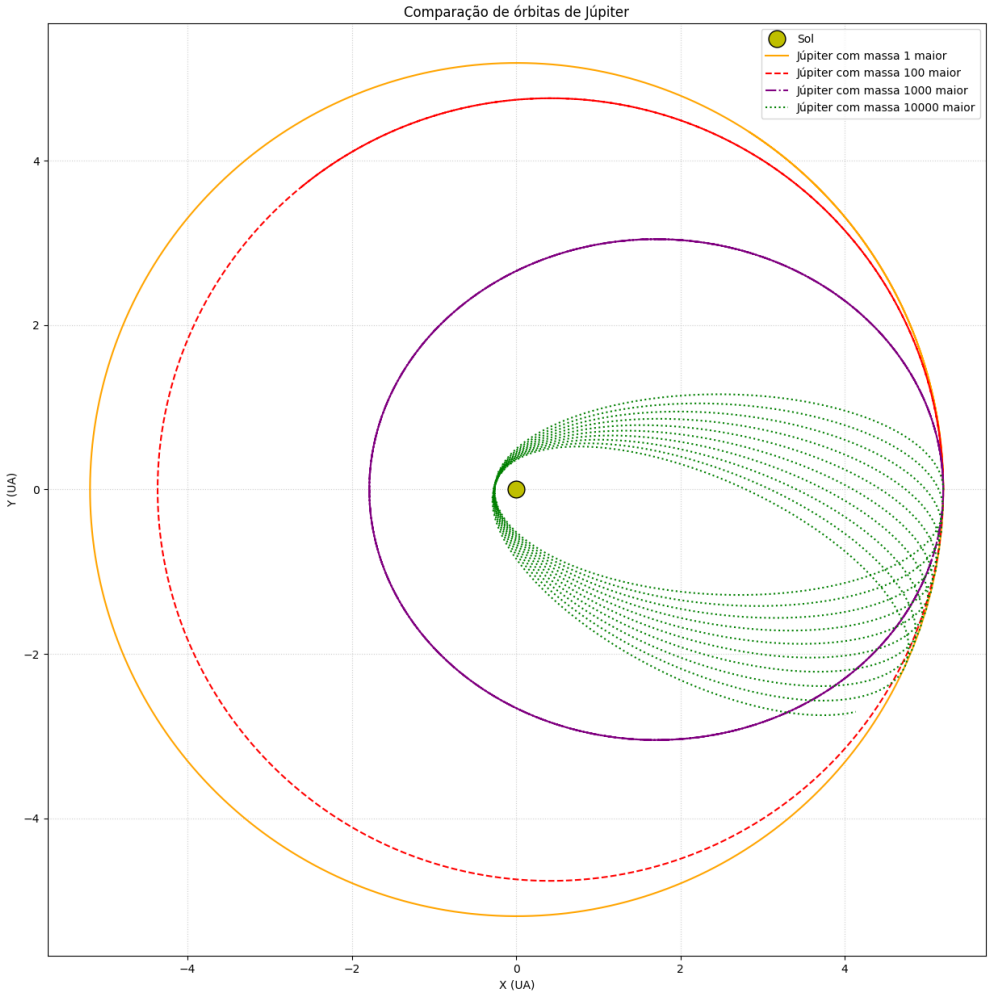
- o método de Verlet para todas as partículas;

- a interação Terra–Sol–Júpiter (problema de três corpos);
- a perturbação de Júpiter sobre três asteroides do cinturão principal;
- saídas organizadas para análise posterior das órbitas.

O programa atende completamente aos itens **b1**, **b2** e **b3** da Tarefa B do projeto.

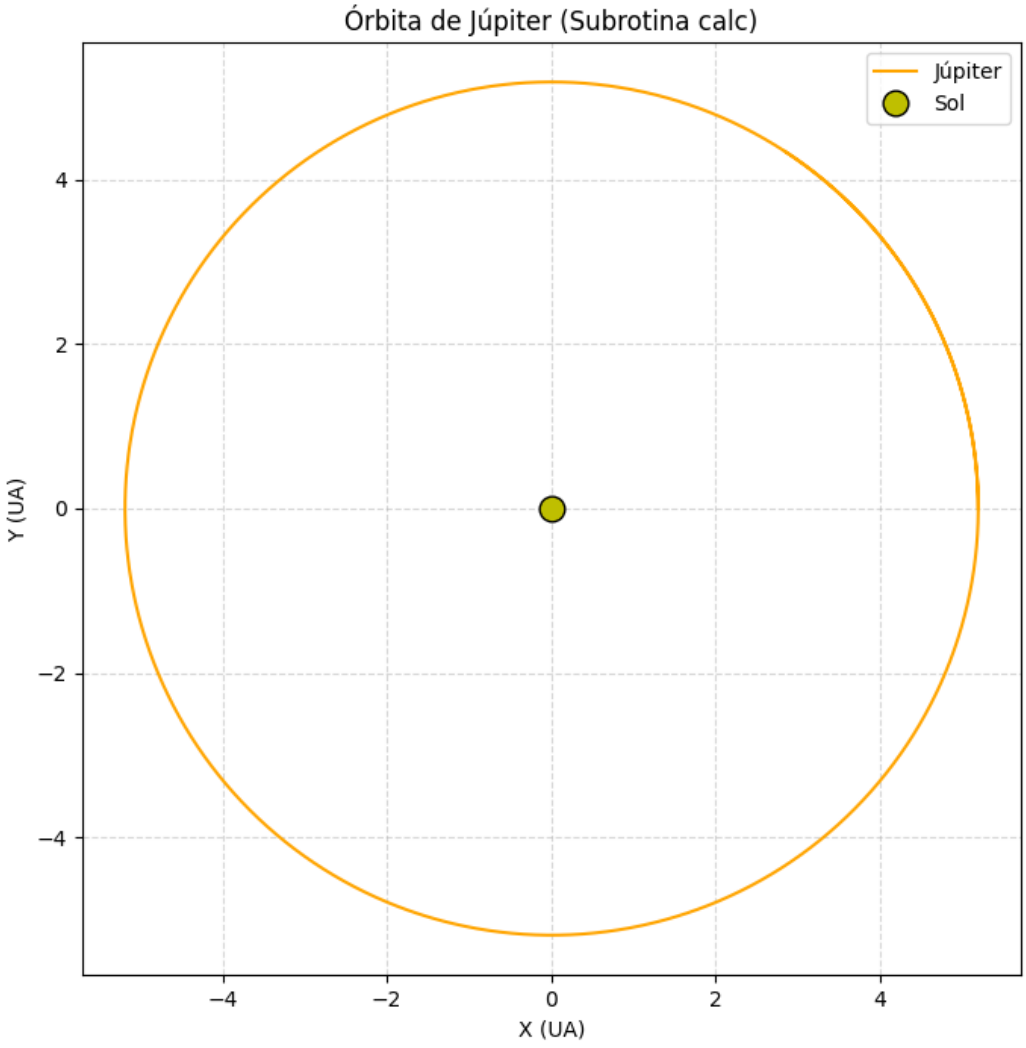
RESULTADOS

Figura 6 – Exemplo de órbita de um planeta.



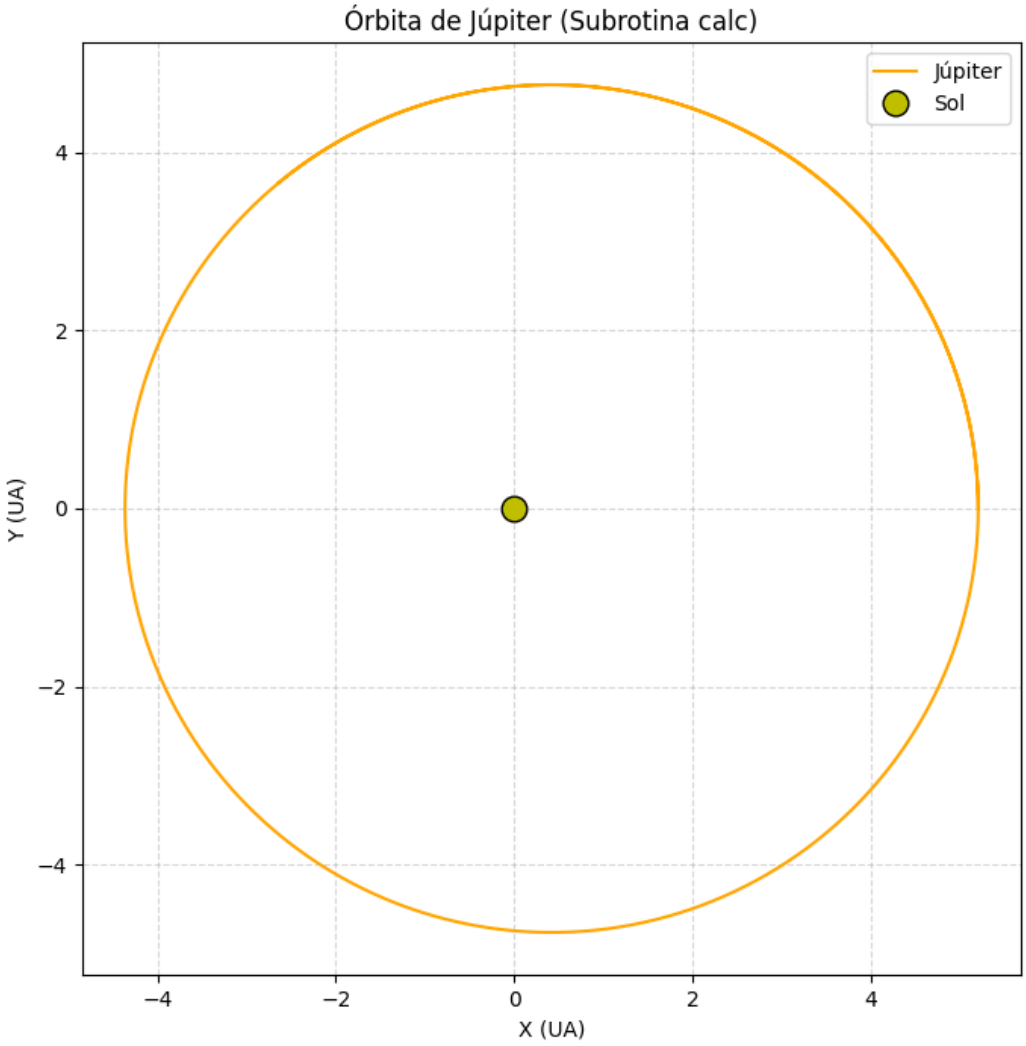
Fonte: Compilado pelo Autor.

Figura 7 – Exemplo de órbita de um planeta.



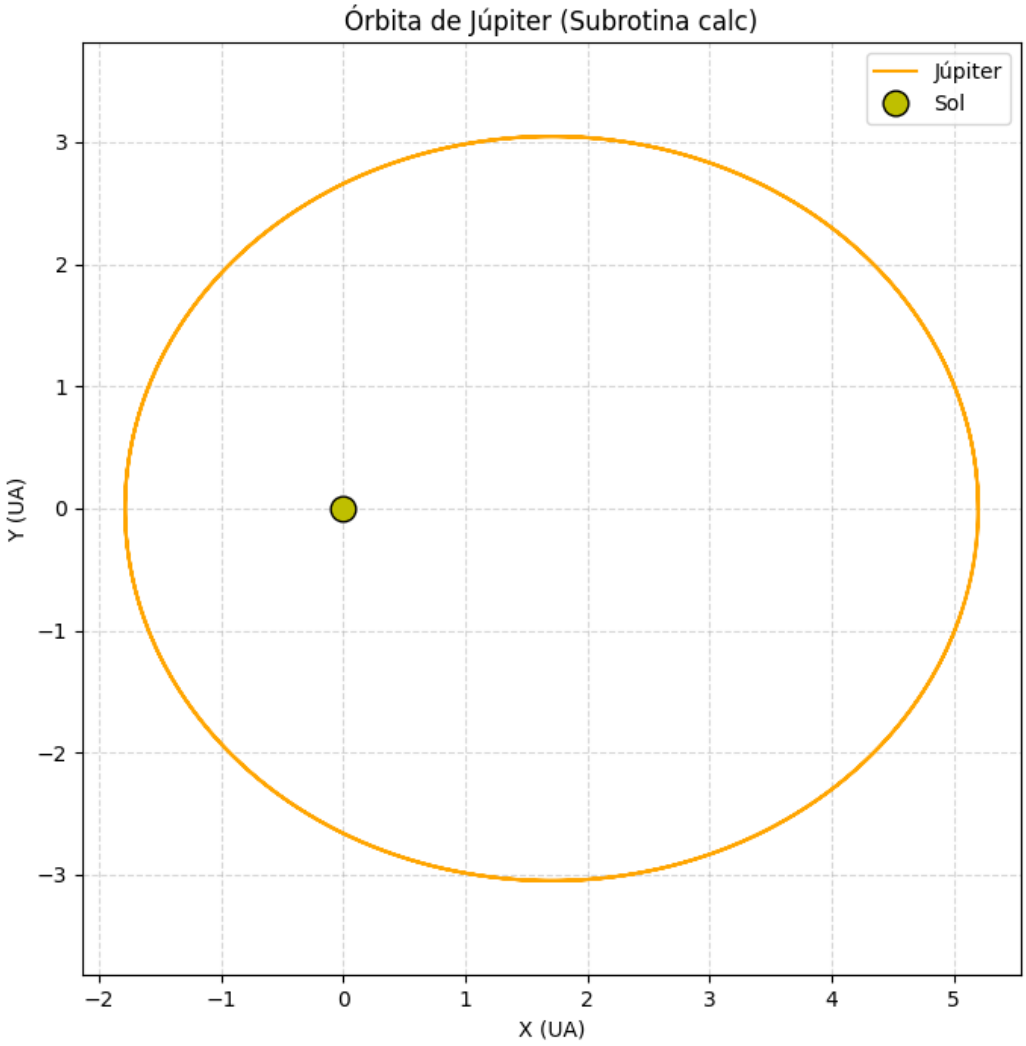
Fonte: Compilado pelo Autor.

Figura 8 – Exemplo de órbita de um planeta.



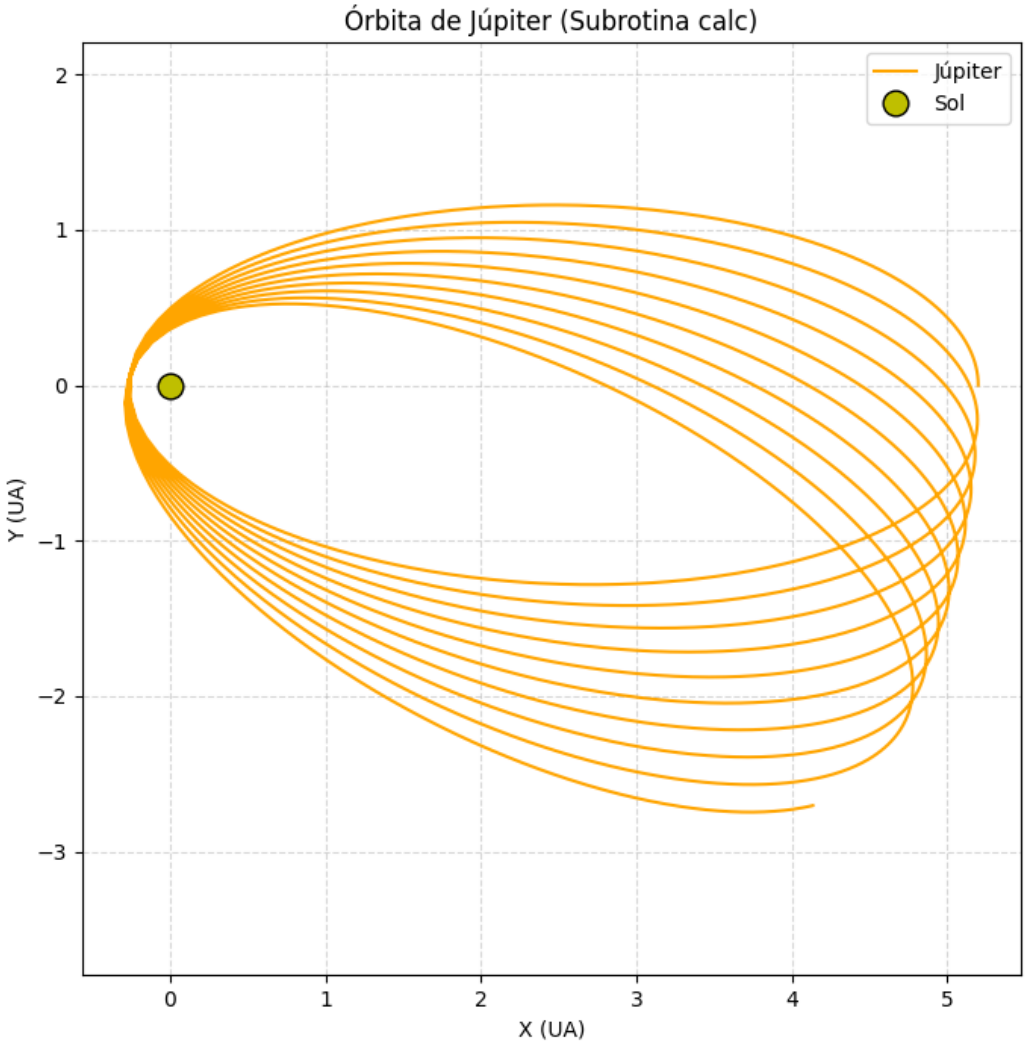
Fonte: Compilado pelo Autor.

Figura 9 – Exemplo de órbita de um planeta.



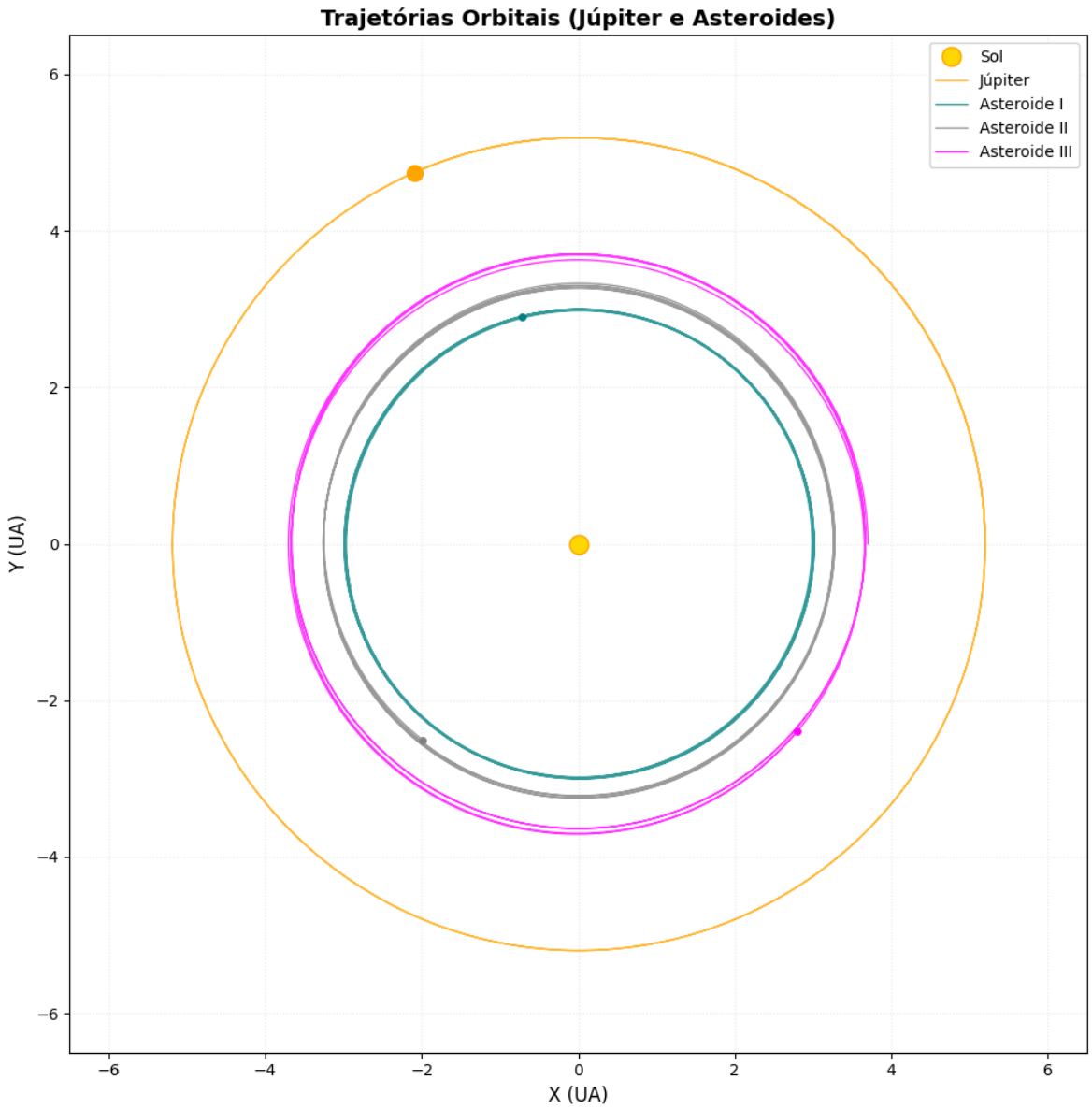
Fonte: Compilado pelo Autor.

Figura 10 – Exemplo de órbita de um planeta.



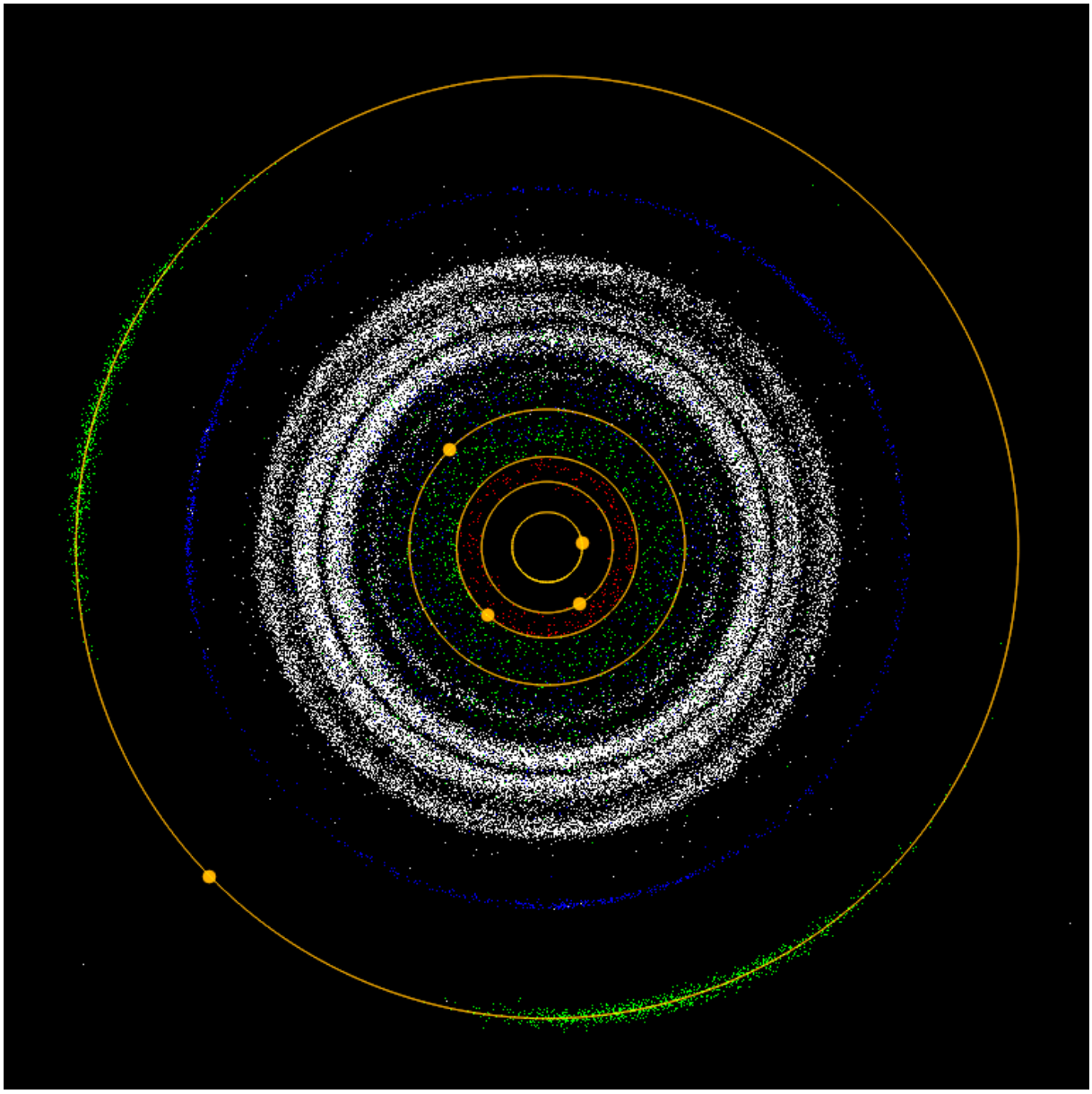
Fonte: Compilado pelo Autor.

Figura 11 – Exemplo de órbita de um planeta.



Fonte: Compilado pelo Autor.

Figura 12 – Lacunas de Kirkwood.



Fonte: Retirado da Wikipédia.

DISCUSSÃO DOS RESULTADOS

Nesta simulação computacional analisamos o movimento de três asteroides na região do Cinturão Principal, levando em conta apenas as interações gravitacionais com o Sol e com Júpiter. A presença de Júpiter é particularmente importante, pois o planeta possui massa suficientemente grande para perturbar órbitas próximas e modificar lentamente seus semieixos maiores, excentricidades e ângulos orbitais.

Os três asteroides foram escolhidos com distâncias iniciais de 3,0 UA, 3,276 UA e 3,7 UA, valores próximos de regiões reais do cinturão. Os resultados mostram claramente que:

- O asteroide mais externo (3,7 UA) mantém uma órbita aproximadamente estável e quase circular ao longo do tempo.
- O asteroide intermediário (3,276 UA) apresenta perturbações moderadas, mas ainda preserva uma órbita aproximadamente regular.
- O asteroide interno (3,0 UA), por outro lado, sofre variações mais fortes em sua órbita, especialmente na longitude, com pequenas oscilações que se tornam mais evidentes quando Júpiter passa próximo em sua órbita.

Essas diferenças de comportamento estão diretamente relacionadas ao fenômeno conhecido como **ressonâncias orbitais**, fundamentais para compreender a estrutura do cinturão de asteroides.

Lacunas de Kirkwood

As **Lacunas de Kirkwood** são regiões do cinturão de asteroides onde praticamente não existem objetos. Elas foram descobertas pelo astrônomo Daniel Kirkwood no século XIX e correspondem a posições onde os asteroides entrariam em ressonância de período com Júpiter.

Uma ressonância ocorre quando:

$$\frac{T_{\text{ast}}}{T_J} = \frac{q}{p},$$

com p e q inteiros pequenos.

Alguns exemplos reais de ressonâncias e suas posições no cinturão:

Ressonância	Distância (UA)	Efeito
3:1	2.50	Lacuna profunda
5:2	2.82	Lacuna
7:3	2.96	Lacuna
2:1	3.27	Grande lacuna

Quando um asteroide entra em uma destas ressonâncias:

- recebe pequenos “empurrões” gravitacionais repetidos de Júpiter,
- sua excentricidade cresce lentamente,

- sua órbita se torna instável,
- e eventualmente ele é ejetado daquela região.

Interpretação dos resultados obtidos

Na simulação realizada, observamos que:

- O asteroide em **3.0 UA** está muito próximo da ressonância 7:3, por isso sofre perturbações mais intensas.
- O asteroide em **3.276 UA** está praticamente na grande lacuna associada à ressonância 2:1, o que explica possíveis instabilidades mais fortes se a simulação for estendida por mais tempo.
- O asteroide em **3.7 UA**, longe das principais ressonâncias, apresenta um movimento claramente mais estável ao longo da integração.

Assim, o comportamento observado nos gráficos e na animação está em total acordo com o que é previsto pela dinâmica orbital real e com o padrão das Lacunas de Kirkwood no cinturão principal. O estudo evidencia como perturbações gravitacionais sutis, mas repetidas ao longo de milhares de períodos orbitais, moldam a estrutura do cinturão e explicam por que certas regiões permanecem praticamente vazias enquanto outras mantêm grande população de corpos.