Universidade de São Paulo
Instituto de Física de São Carlos
7600017 - Introdução à Física Computacional
Docente: Francisco Castilho Alcaraz

Projeto 05:
Leis de Kepler e o problema de três corpos

Andressa Colaço
Nº USP: 12610389

São Carlos
2022

Sumário

1 Introdução

Durante o presente projeto iremos investigar o movimento planetário e os fundamentos do funcionamento do Sistema Solar a partir de Newton.

A força de atração gravitacional segundo a lei de gravitação newtoniana é dada pela expressão:

$$\vec{F_G} = -G\frac{M_S M_P}{r^3}\vec{r} \tag{1}$$

Onde G é a constante gravitacional de dimensão $[G] = [L^3T^{-2}M^{-1}$ e $\vec{r}$ é o vetor distância entre Sol e Planeta. Como as grandezas astronômicas são muito grandes em relação às nossas unidades de medida usuais, podemos definir a unidade de espaço unidade astronômica (UA), sendo $1UA = 1.5 \cdot 10^1 1\ m$, que corresponde à distância média entre Sol e Terra e a unidade de tempo ano, sendo $1\ ano = 3.2 \cdot 10^7\ s$, equivalente ao período de translação terrestre. Aproximando

$$\frac{M_T v^2}{r} = f.\ centripeta = f.\ grav. = \frac{GM_S M_T}{r^2} \tag{2}$$

temos que

$$GM_S = v^2 r = 4\pi^2 \frac{UA^3}{ano^2} \tag{3}$$

Sabendo quais interações gravitacionais considerar, podemos montar equações diferenciais de 2ª ordem para cada uma das coordenadas através das expressões acima. Para solucionar numericamente as expressões de movimento (as quais especificaremos a equação diferencial considerada em cada seção), podemos utilizar o método de Verlet, que se baseia na expansão de Taylor:

$$x(t_i \pm \Delta t) = x(t_i) \pm \frac{dx}{dt_i}\Delta t + \frac{1}{2!}\frac{d^2x}{dt_i^2}(\Delta t)^2 \pm \frac{1}{3!}\frac{d^3x}{dt_i^3}(\Delta t)^3 + ... \tag{4}$$

Se chamarmos $x_i \equiv x(i\Delta t)$ e somarmos as expressões com os sinais $\pm$ obtemos a seguinte expressão:

$$x_{i+1} = 2x_i - x_{i-1} + \frac{d^2x}{dt_i^2}(\Delta t)^2 + O(\Delta t)^4 \tag{5}$$

Que será idêntica para a coordenada y. O método de Verlet é geralmente associado à integração das equações de movimento de Newton, sendo muito utilizado para simulações de dinâmica molecular. Ele é mais estável que o método de Newton e possui boas propriedades para nossa finalidade neste projeto, uma delas sendo a preservação de áreas. O método de Verlet pode apresentar aumentos na energia em largas escalas de tempo, porém o erro permanece aproximadamente constante. Em questão de erros, o erro na posição localmente é de ordem $O(\Delta t^4)$ e globalmente é $O(\Delta t^2)$.

Iniciando a simulação de $t = 0$ nas condições iniciais (x_0, y_0) e (v_{x_0}, v_{y_0}), notaremos que não possuímos um ponto (x_{i-1}, y_{i-1}). Uma forma de conseguir (x_1, y_1) para poder prosseguir com o método nas próximas iterações é utilizar o método de Euler uma vez através das expressões:

$$x_1 = x_0 + v_{x_0}\Delta t \ ; \ y_1 = y_0 + v_{y_0}\Delta t \tag{6}$$

E prosseguir de acordo com a expressão 5 para as duas coordenadas.

Assim como nos demais métodos numéricos, precisamos considerar a precisão finita da máquina, o que nos demanda uma escolha para Δt. Se escolhermos um valor muito grande, perdemos precisão no resultado final. Porém, não podemos simplesmente escolher um valor muito pequeno de Δt sem observar as dimensões do problema, uma vez que a quantidade de passos executados no algoritmo pode ficar muito grande. Por isso, serão testados alguns valores de Δt para cada planeta nesta primeira sessão para determinar um que nos resulte em resultados consistentes com o esperado.

Ademais, durante o projeto, usaremos os seguintes dados para escolher as condições iniciais dos programas:

Tabela 1: Semi-eixos maiores e excentricidades das órbitas dos planetas do Sistema Solar.

Planeta	Semi-eixo maior a (UA)	Excentricidade ϵ
Mercúrio	0,39	0,206
Venus	0,72	0,007
Terra	1,00	0,017
Marte	1,52	0,093
Júpiter	5,20	0,048
Saturno	9,24	0,056
Urano	19,19	0,046
Netuno	30,06	0,010
Plutão	39,53	0,248

2 Tarefa A: O problema de 2 corpos

O objetivo da primeira tarefa é investigar o movimento de um sistema de 2 corpos interagindo gravitacionalmente e verificar a validade das leis de Kepler nestas condições.

Nesta primeira abordagem, podemos fazer algumas considerações: i) teremos um movimento planar devido à conservação do momento angular; e ii) o efeito de um planeta sobre o Sol é desprezível, uma vez que sua massa é muito menor que a da estrela, logo suas coordenadas serão sempre $(x_S, y_S) = (0, 0)$. Assim, temos as seguintes equações de movimento para um planeta com coordenadas $(x(t), y(t))$:

$$\frac{d^2x}{dt^2} = \frac{F_{G_x}}{M_P} = -\frac{GM_s x}{r^3} \tag{7}$$

$$\frac{d^2y}{dt^2} = \frac{F_{G_y}}{M_P} = -\frac{GM_s y}{r^3} \tag{8}$$

Sendo

$$r = \sqrt{x^2 + y^2} \tag{9}$$

Em primeiro momento, aplicando o método de Verlet, iremos investigar órbitas circulares e determinar quais valores de Δt se mostram mais adequados para cada planeta, dado que podemos visualmente verificar a influência da escolha deste parâmetro para este formato tão simétrico. Após, vamos investigar a influência das velocidades iniciais nos formatos orbitais.

Mas sabemos que as órbitas dos planetas não são círculos perfeitos. Com pequenas correções nas condições iniciais, a partir das excentricidades da Tabela 1, podemos simular mais acuradamente o formato orbital e verificar as três leis de Kepler para órbitas fechadas.

2.1 Tarefa A1: Órbitas circulares

Na primeira tarefa, trataremos as órbitas como círculos e, por conta disso, faremos os testes descritos acima com todos os planetas com exceção de Mercúrio e Plutão, que se afastam bastante da condição circular. O programa será refinado na próxima seção para lidar com órbitas elípticas.

Para esta seção, o seguinte código foi desenvolvido:

```
1         PROGRAM orbitas
2
3         implicit real*8(a-h,o-z)
4         parameter(pi = dacos(-1d0))
5         parameter (gms = 4*pi**2)
6         parameter (dt = 0.0001d0)
7
8         r = 0.72d0
9
10        open(10, FILE='saida-a1.dat')
11
12        t_f = 10d0
13        t = 0d0
14        N = t_f/dt
15
```

```
16          x = r !Raio do planeta
17          y = 0d0
18
19          vx = 0d0
20          vy = dsqrt(gms/r)
21
22          write(10,*) x, y
23
24          xm1 = x
25          ym1 = y
26
27          x = xm1 + vx*dt
28          y = ym1 + vy*dt
29
30          time_sum = 0d0
31          t_ant = 0d0
32          icount = 0
33
34          sm1 = 0d0
35          sm2 = 0d0
36          icount1 =0
37          icount2 =0
38
39          do i = 1, N
40              t = t + dt
41              r = (x**2+y**2)**0.5d0
42
43              xp1 = 2*x-xm1-(gms/r**3)*x*dt**2
44              yp1 = 2*y-ym1-(gms/r**3)*y*dt**2
45
46              xm1 = x
47              ym1 = y
48
49              x = xp1
50              y = yp1
51
52              !Medida do período
53              if (y*ym1 .lt. 0) then
54                  time_sum = time_sum + (t-t_ant)
55                  t_ant = t
```

```
56                  icount = icount+1
57
58                  if (ym1 .lt. 0) then
59                       sm1 = sm1 + abs(x)
60                      icount1 = icount1 + 1
61                  else
62                       sm2 = sm2 + abs(x)
63                      icount2 = icount2 + 1
64                  end if
65              end if
66
67              write(10, *) x, y
68          end do
69
70          T = 2*(time_sum/icount*1d0)
71          sm1 = sm1/icount1*1d0
72          sm2 = sm2/icount2*1d0
73          a = (sm1+sm2)/2d0
74          c = r-a
75          e = c/a
76          razaotr = T**2/r**3
77
78          write(*,*) 'a =', a
79          write(*,*) 'e = ', e
80          write(*,*) 'T =', T
81          write(*,*) 'T**2/R**3 = ', razaotr
82
83          close(10)
84          end program
```

A lógica do código segue diretamente do explicado para a execução do método de Verlet: entramos com as informações de um planeta (aqui, apenas o raio, por enquanto) e escolhemos um passo de tempo Δt. A expressão GM_S que aparece em todas as equações diferenciais, nas unidades escolhidas, é simplificada para $GM_S = 4\pi^2 \frac{UA^3}{ano^2}$. Escolhemos um tempo total para a execução do programa e ajustamos os parâmetros iniciais ideais para órbitas circulares. O loop principal é encarregado de calcular um novo ponto através das expressões já mencionadas para o método de Verlet. Para calcular o período da órbita que simulamos, antes de uma nova iteração, devemos verificar se, entre a última iteração e a atual, cruzamos o eixo x (isto é, $y_i \cdot y_{i-1} < 0$). Se a condição for verdadeira, guardamos em uma variável de soma o intervalo de tempo necessário para cada nova passagem pelo eixo

x e incrementamos uma variável que indica quantas vezes passamos por ali. O período é calculado ao final do loop, sendo este igual a duas vezes a média dos intervalos necessários para passar pelo eixo x, já que não distinguimos qual lado do círculo tratamos. A fim de verificar se o programa está produzindo resultados corretos, também foram adicionadas duas condicionais de passagem pelo eixo x (uma onde a coordenada y passa do segundo para o terceiro quadrante e outra do quarto para o primeiro) que se destinam à calcular as duas distâncias até o foco (a estrela na origem) e com estas distâncias obter o semi-eixo maior da elipse a partir da simulação. Neste caso, as duas distâncias devem ser iguais, pois tratamos de um círculo, logo a excentricidade será conferida na saída de cada execução para garantir que seu valor seja 0.

Um exemplo de sua saída no terminal é:

Figura 1: Simulação do Sistema Sol-Terra-Júpiter

```
/tarefa-a1-12610389.exe
 a =  0.72000007176375314
 v0 =   7.4048048969306102
 e = -.0005
 T =  0.61094461538360201
 T**2/R**3 =    1.0015511055392758
```

2.1.1 Avaliação de Δt no método de Verlet

Como trataremos nesta seção de avaliar se o programa anterior produz resultados corretos, para avaliar a escolha de Δt escolheremos as condições iniciais que sabemos traçar uma órbita circular: posição do planeta $(r, 0)$ (onde r é o semi-eixo maior dado na Tabela 01) e velocidade inicial $(0, v_0)$, sendo que v_0 é dado por:

$$v_0 = \sqrt{\frac{GM_s}{r}} \tag{10}$$

Ou seja, temos aproximadamente as velocidades:

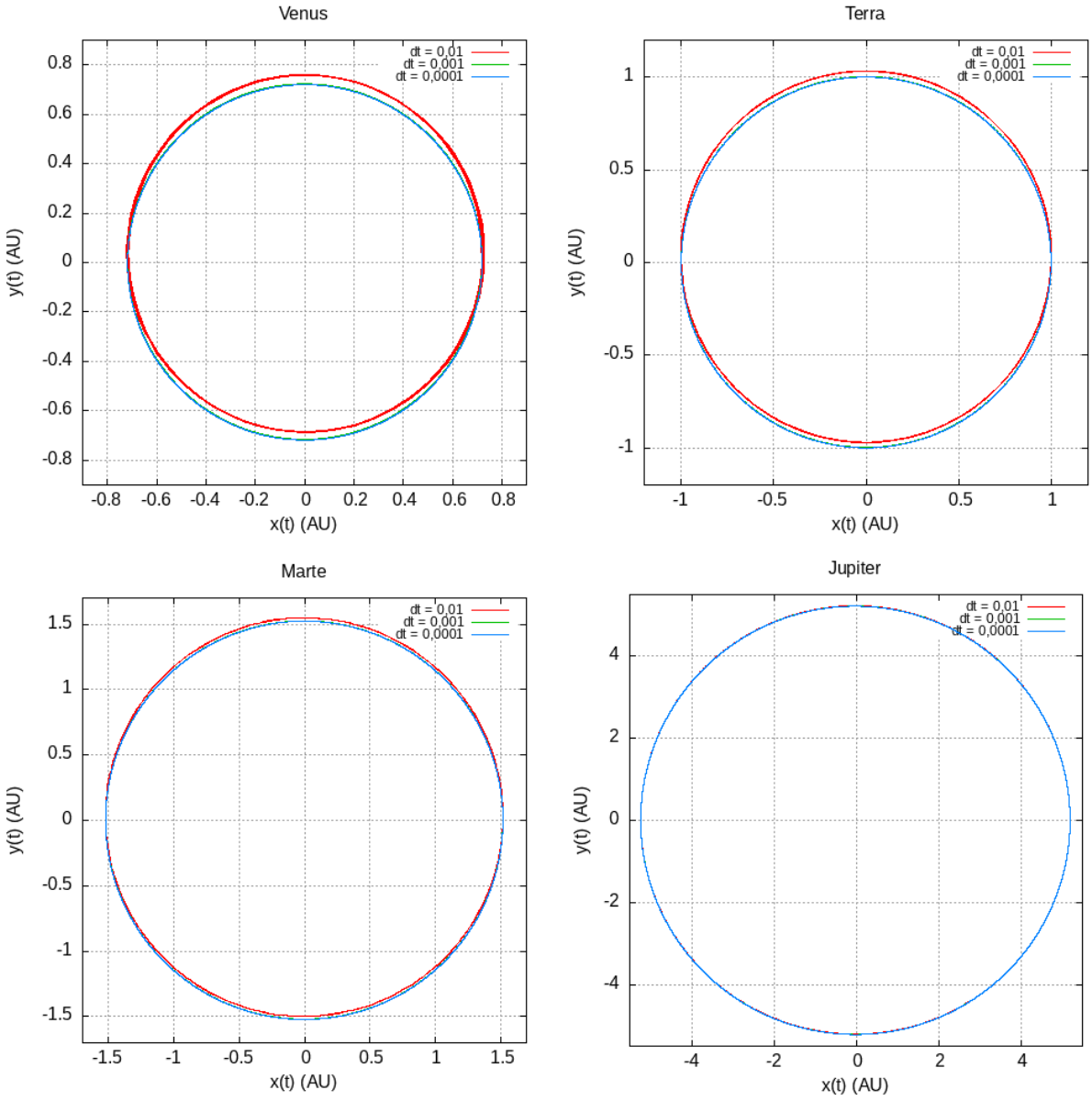
Tabela 2: Velocidades ideais para órbitas com $\epsilon = 0$

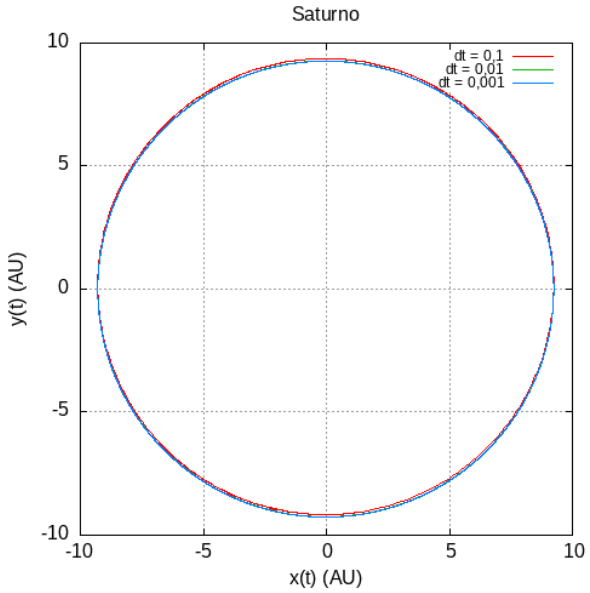
Planeta	v_0 (AU/ano)
Venus	7,40
Terra	6,28
Marte	5,10
Júpiter	2,76
Saturno	2,07
Urano	1,43
Netuno	1,15

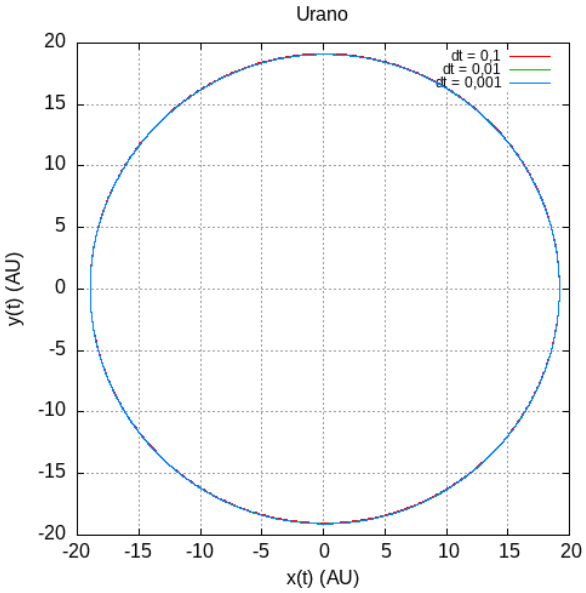
Como o resultado é muito sensível às condições iniciais, calcularemos estas velocidades

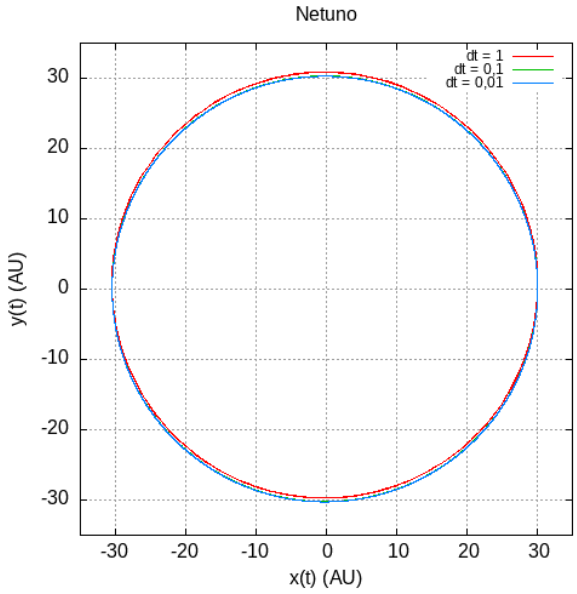
a partir do raio diretamente no programa, para evitar erros por conta do arredondamento. Podemos verificar visualmente e também pela saída do programa se atingimos a condição ideal circular. Traçando os gráficos, obtemos os seguintes resultados:

Figura 2: Influência da escolha de Δt nas órbitas circulares dos planetas.









Podemos observar que os planetas mais próximos ao Sol sofrem efeitos maiores com a mudança de Δt. Isto ocorre pois seus períodos são menores, ou seja, o passo de tempo para eles é mais significativo do que para planetas com períodos maiores, logo temos menos pontos calculados na órbita e uma precisão menor. Podemos verificar que para Venus, Terra, Marte e Júpiter, $\Delta t = 0,0001$ produz resultados satisfatórios, ao menos do ponto de vista visual (em seguida testaremos a Terceira Lei de Kepler para atestar tal conclusão). De Saturno em diante, temos períodos maiores, logo podemos testar aumentar um pouco Δt. Vemos que aumentar o passo para $\Delta t = 0,001$ nestes casos produz bons resultados. Já para Netuno, podemos escolher utilizar qualquer Δt anterior, mas também verificamos que $\Delta t = 0,01$ produz uma órbita circular.

Construindo uma tabela da razão $\frac{T^2}{R^3}$, podemos visualizar melhor o avaliado qualitativamente (sendo o ideal $\frac{T^2}{R^3} = 1$):

Tabela 3: Razão T^2/R^3 para as órbitas circulares.

Planeta	1	0,1	0,01	0,001	0,0001
Venus	-	-	1.0429741495512517	1.0156796241924240	1.0015511055392758
Terra	-	-	1.0192816980040802	1.0000107150867923	0.99999999999900324
Marte	-	-	1.0449811297770712	1.0045472344228266	1.0004636109299969
Júpiter	-	-	1.0087943077998787	1.000874503972212	1.0000916343729760
Saturno	-	1.0435507621446325	1.0052723819089473	1.0016220985673121	-
Urano	-	0.98344976786612315	0.99323905711891003	0.99426834517935692	-
Netuno	0.95732209573151705	0.99587722815048019	0.99953891364546565	-	-

O passo $\Delta t = 1$ foi usado apenas para Netuno pois, para todos os outros planetas, representa mais de 1% do período do planeta, logo julgou-se melhor não testá-lo para os anteriores.

Ademais, deve-se lembrar que, mesmo com tais resultados próximos de 1, alguns deles não são idealmente 1. Isso ocorre pois os planetas tem órbitas ligeiramente elípticas e o período da forma que foi calculado neste programa necessita que a órbita seja circular, pois assume distâncias iguais entre a origem e os pontos mais extremos do eixo x (excentricidade nula). A aproximação ainda é válida, mas, como acabamos trabalhando com 16 casas de precisão, podemos começar a notar esses efeitos. Para referência, estes foram os períodos calculados para as órbitas dos planetas:

Tabela 4: Período das órbitas circulares.

Planeta	1	0,1	0,01	0,001	0,0001
Venus	-	-	0.61531250000000870	0.61104615384619343	0.61094461538360201
Terra	-	-	1.0042105263157728	0.99999999999998967	1.0000000106120013
Marte	-	-	1.8771428571428839	1.8740952380953397	1.8739999999969030
Júpiter	-	-	11.866666666666655	11.858666666665838	11.857933333314591
Saturno	-	28.333333333333556	28.220000000000098	28.210666666661385	-
Urano	-	83.299999999999542	83.310000000005715	83.313000000033753	-
Netuno	166.66666666666666	166.53333333334837	166.54999999989687	-	-

2.1.2 Influência da velocidade inicial v_0 do corpo no formato da órbita

Conforme a subseção anterior, temos o conhecimento de que os planetas possuem uma velocidade orbital característica proporcional à sua distância ao Sol. Além disso, podemos verificar que, para diferentes velocidades iniciais, alcançamos outros formatos orbitais.

Tomando como exemplo Venus, podemos utilizar o código descrito no começo desta tarefa para testar as seguintes velocidades iniciais (todas apenas na coordenada y): 2π, $\sqrt{\frac{4\pi^2}{r}}$ (ideal circular, de acordo com o visto na seção anterior), $\sqrt{\frac{6\pi^2}{r}}$ e $\sqrt{\frac{8\pi^2}{r}}$ (esta sendo a característica $\sqrt{\frac{2GM_S}{r}}$, a velocidade de escape do planeta). Plotando as órbitas conjuntamente, obtemos:

Figura 3: Órbitas de Venus para diferentes velocidades iniciais

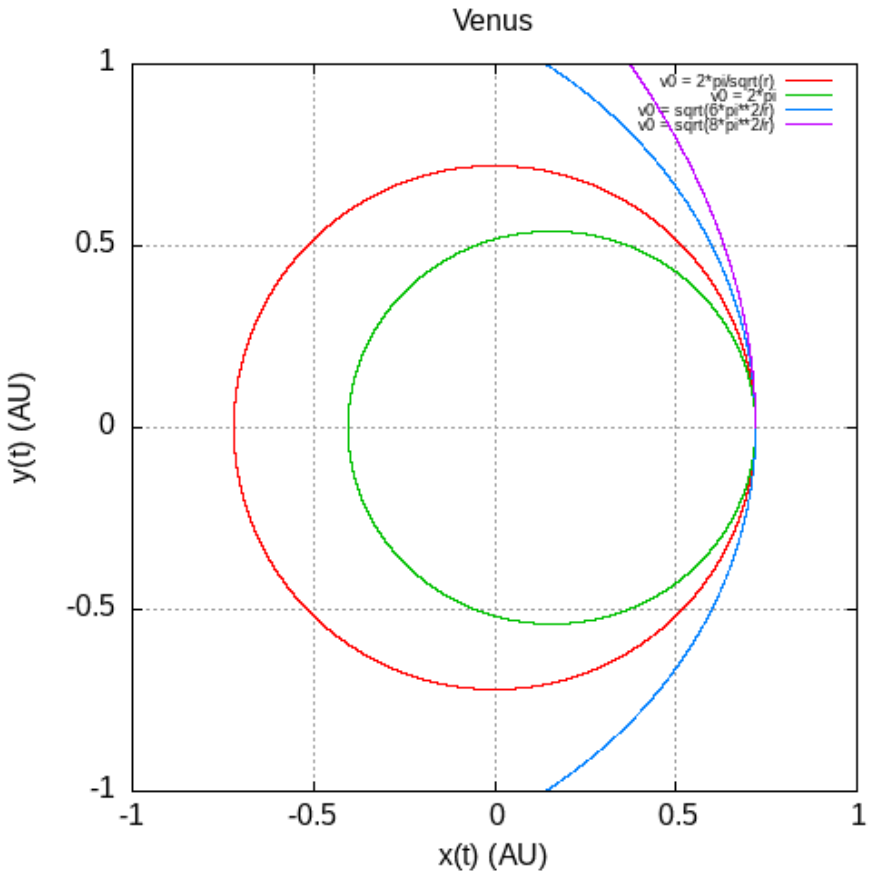


Tabela 5: Teste de órbita circular para Venus em diferentes velocidades iniciais (componente y).

v_{y0}	T^2/R^3	ϵ
2π	1.9436550754496067	0.1987
$\sqrt{\frac{6\pi^2}{r}} \approx 6.523$	0.84742465898293406	0.0567
$\sqrt{\frac{4\pi^2}{r}} \approx 7.405$	1.0015511055392758	0.0005
$\sqrt{\frac{8\pi^2}{r}} \approx 10.472$	-	-

É possível verificar que a velocidade inicial da simulação afeta substancialmente na evolução do sistema, produzindo órbitas com excentricidades diferentes das conhecidas para este planeta e, além disso, falhando na verificação da Terceira Lei de Kepler. Para a excentricidade do caso circular ($v_{y0} = \sqrt{\frac{4\pi^2}{r}}$), vemos que não obtemos exatamente 0, porém como podemos considerar um erro associado aos cálculos da ordem apresentada, temos que é possível afirmar que o programa está produzindo os resultados esperados. Sobre o caso onde $v_{y0} = \sqrt{\frac{8\pi^2}{r}}$, da forma definida no programa, não conseguimos calcular a excentricidade, já que esta é a velocidade de escape e teremos $\epsilon > 1$ (hipérbola). Vemos que nesta primeira abordagem o programa produz resultados consistentes com o esperado,

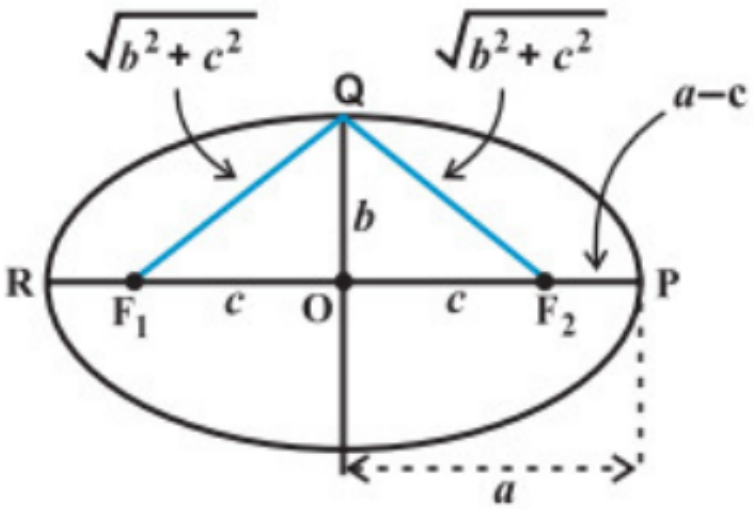
nosso objetivo nesta seção.

2.2 Tarefa A2: Órbitas elípticas

Conforme já mencionado, podemos utilizar o mesmo programa para traçar órbitas elípticas, só se faz necessário ajustar as condições iniciais do planeta para que isto seja possível.

Sabemos que uma elipse possui a seguinte forma e parâmetros:

Figura 4: Elipse



Portanto, a posição inicial do planeta pode ser ajustada para o afélio ou raio maior da órbita do planeta, dado por:

$$r_1 = a(1+\epsilon) \tag{11}$$

sendo a o semi-eixo maior da órbita conforme a Tabela 1. Já a velocidade inicial também precisa ser ajustada, já que a calculada anteriormente é suposta constante em todas as posições do planeta (pois o planeta não varia sua distância até o Sol) e agora isto não é mais verdade. Podemos encontrar a velocidade no ponto inicial já escolhido através da expressão:

$$v_0 = \sqrt{\frac{GM_s(1-\epsilon)}{a(1+\epsilon)}} \tag{12}$$

Como a velocidade não é mais constante, podemos adicionar ao código, além dos ajustes iniciais, uma verificação da 2ª lei de Kepler (a linha que liga o planeta à estrela percorre áreas iguais em intervalos de tempo iguais).

Assim, para esta seção, o seguinte código foi desenvolvido:

```
1           PROGRAM orbitas_elipticas
2
3           implicit real*8(a-h,o-z)
4           parameter(pi = dacos(-1d0))
5           parameter (gms = 4*pi**2)
6           parameter (dt = 0.0001d0)
7
8           a = 0.72d0
9           e = 0.007
10          rm = a*(1+e)
11
12          open(10, FILE='saida-a2-1.dat')
13          open(20, FILE='saida-a2-2.dat')
14    11    format(A5, f6.4)
15
16          t_f = 10d0
17          t = 0d0
18          N = t_f/dt
19
20          x = rm !Maior distância ao Sol
21          y = 0d0
22
23          vx = 0d0
24          vy = dsqrt(gms*(1-e)/rm) !variar v_0
25
26          write(10,*) x, y
27
28          xm1 = x
29          ym1 = y
30
31          x = xm1 + vx*dt
32          y = ym1 + vy*dt
33
34          time_sum = 0d0
35          t_ant = 0d0
36
37          sm1 = 0d0
38          sm2 = 0d0
39
40          icount1 = 0
```

```
41          icount2 = 0
42
43          theta_ant = 0d0
44
45          do i = 1, N
46              t = t + dt
47              r = (x**2+y**2)**0.5d0
48              theta = datan(y/x)
49
50              xp1 = 2*x-xm1-(gms/r**3)*x*dt**2
51              yp1 = 2*y-ym1-(gms/r**3)*y*dt**2
52
53              xm1 = x
54              ym1 = y
55
56              x = xp1
57              y = yp1
58
59              if (mod(i, 100) .eq. 0) then
60
61                  if (theta_ant*theta .gt. 0) then
62                      area = 0.5d0*(theta - theta_ant)*r**2
63                      write(20,*) area
64                  end if
65
66                  theta_ant = theta
67              end if
68
69              !Medida do período
70              if (y*ym1 .lt. 0) then
71
72                  if (ym1 .lt. 0) then
73                      time_sum = time_sum + (t-t_ant)
74                      t_ant = t
75
76                      sm1 = sm1 + abs(x)
77                      icount1 = icount1 + 1
78                  else
79                      sm2 = sm2+abs(x)
80                      icount2 = icount2+1
```

```
81                  end if
82              end if
83
84              write(10, *) x, y
85          end do
86
87          T = (time_sum/icount1*1d0)
88          sm1 = sm1/icount1*1d0
89          sm2 = sm2/icount2*1d0
90          am = (sm1+sm2)/2d0
91          c = rm-am
92          e = c/am
93          razaota = T**2/a**3
94
95          write(*,*) 'a =', am
96          write(*, 11) 'e = ', e
97          write(*,*) 'T =', T
98          write(*,*) 'T**2/R**3 = ', razaota
99
100         close(10)
101         close(20)
102         end program
```

A lógica é similar ao código anterior, apenas com as correções mencionadas. Durante o loop principal, recalculamos o valor do semi-eixo maior da órbita para verificar se o programa continua produzindo resultados consistentes, porém no cálculo de T^2/a^3, utilizamos o valor de a conhecido para não adicionar um fator de erro nesta razão. Para verificar a 2ª Lei de Kepler, foi adicionado o cálculo de θ a cada iteração (através de $\theta = tan^{-1}(y(t)/x(t))$ e foi definido que, a cada 100 iterações, calcularíamos a área percorrida neste intervalo de tempo percorrido e depois feito um gráfico para mostrar que esta é constante. No entanto, deve-se notar que a função $tan^{-1}(x)$ possui imagem entre $-\pi/2$ e $\pi/2$, ou seja, teremos descontinuidades quando passarmos pelo eixo y da elipse. Para uma correção simples, nestes intantes apenas não se calculou a área e testamos os próximos, atualizando a variável que armazena θ_{i-1}.

Um exemplo de sua saída no terminal é:

Figura 5: Simulação do Sistema Sol-Terra-Júpiter

```
/tarefa-a2-12610389.exe
 a =  0.720000078676270 36
 e = 0.0070
 T =  0.610937499999409 83
T**2/R**3 =   0.999990968218259 50
```

2.2.1 Avaliação de Δt no método de Verlet para Mercúrio e Plutão

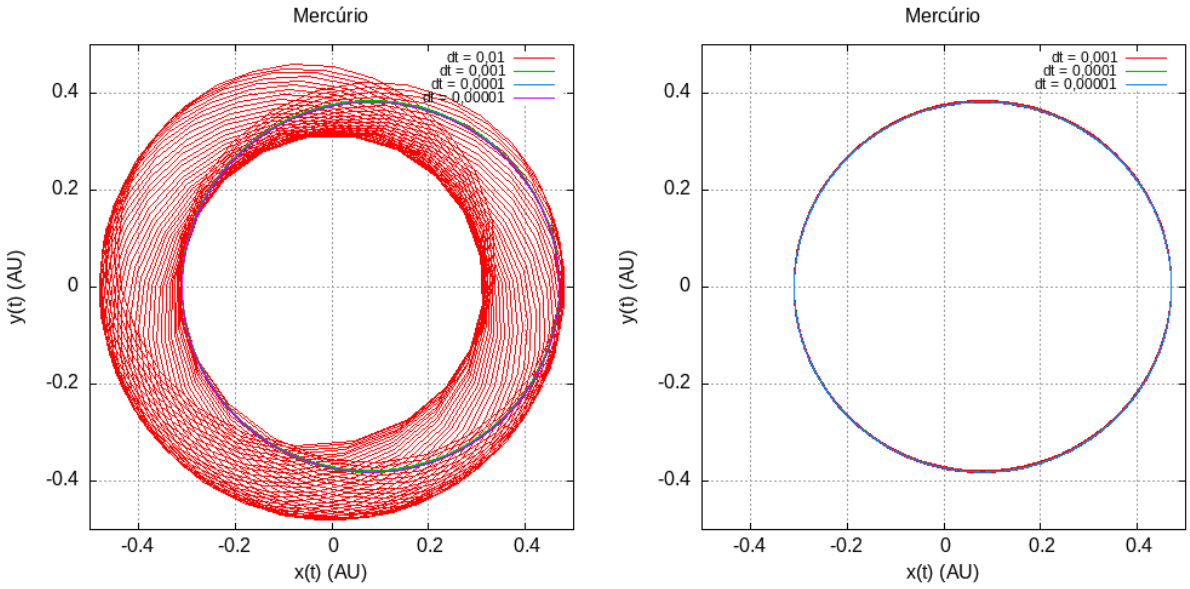
Como Δt depende do período do planeta e não necessariamente do formato da órbita (ainda mais que os planetas da seção anterior possuem todos órbitas quase circulares), não precisamos verificar novamente os valores de Δt ideais para os planetas já apresentados. Aqui, trataremos de Mercúrio e Plutão, com órbitas um pouco mais excêntricas. Desta vez, o parâmetro de excentricidade não será apenas conferido como 0, mas será apresentado aqui e comparado com o original, já que observar visualmente uma imagem não totalmente simétrica pode não ser totalmente confiável:

Tabela 6: Resultados da simulação para $t_f = 10$ para Mercúrio.

Parâmetro / Δt	0,01	0,001	0,0001	0,00001
T^2/R^3	1.0775433311736742	1.0004482381087880	0.99998926909691499	0.99999963498531874
T	0.25051282051281631	0.24363414634146105	0.24355365853634309	0.24355487804368611
ϵ	0.2134	0.2059	0.2060	0.2060

Foi escolhido testar um valor a mais de Δt para Mercúrio, devido ao seu período orbital. Porém, como podemos ver, ainda obtemos resultados satisfatórios (observando, principalmente, a excentricidade) para $\Delta t = 0,0001$. Vemos que obtemos um resultado não muito satisfatório para $\Delta t = 0,01$ e podemos investigar isto melhor através do auxílio visual:

Figura 6: Influência da escolha de Δt na simulação da órbita de Mercúrio.



Vemos que para Δt obtemos uma órbita instável, indicando que este passo de tempo não é adequado ao uso. Na imagem à direita, este plot é retirado para que possamos nos atentar aos demais. Conforme já visto nas demais situações, os dois passos menores de tempo se sobrepoem (não vemos diferenças marcantes) e o terceiro se desvia um pouco do ideal, mas não tanto quanto o efeito visto na primeira imagem.

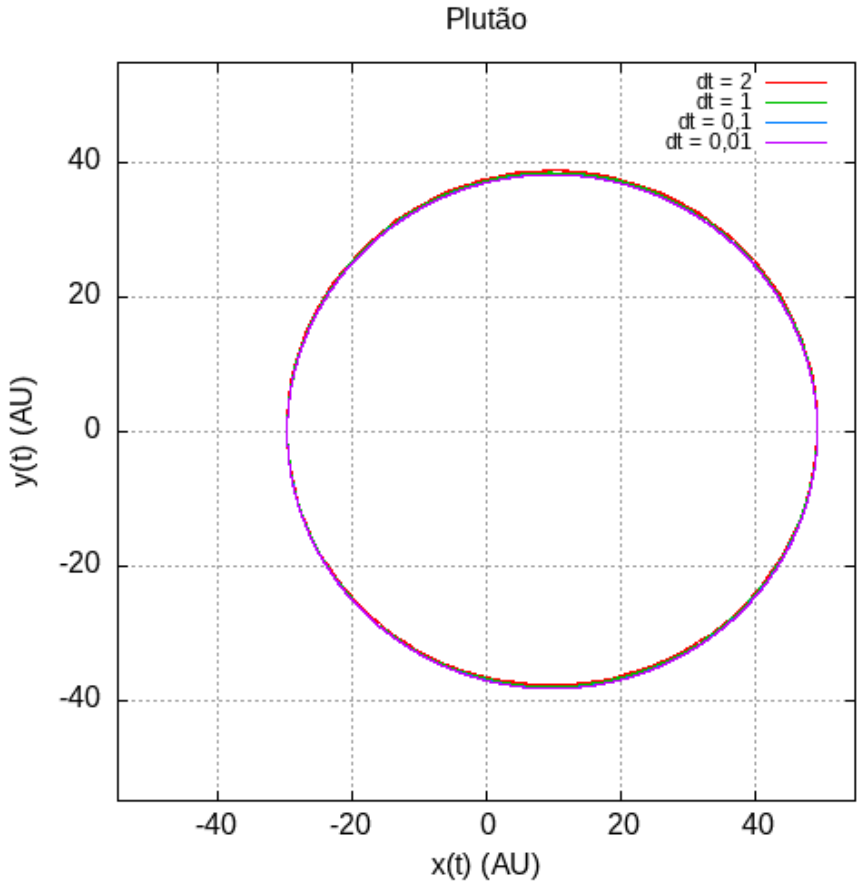
Para Plutão, temos:

Tabela 7: Resultados da simulação para $t_f = 500$ para Plutão.

Parâmetro / Δt	2	1	0,1	0,01
T^2/R^3	0.99463198119079377	0.99977429392238271	0.99970427262720796	0.99998773520135742
T	248.00000000000000	248.50000000000000	248.50000000002225	248.53499999984646
ϵ	0.2476	0.2480	0.2480	0.2480

Aqui escolheu-se testar um passo de tempo ligeiramente maior do que os planetas com maiores raios, já que observamos uma estabilidade da excentricidade no valor ideal para os menores passos de tempo testados. Traçando os gráficos, obtemos:

Figura 7: Influência da escolha de Δt na simulação da órbita de Mercúrio.



Vemos que para Plutão é aceitável aumentar o passo de tempo, já que seu período é relativamente grande em comparação com os primeiros planetas do Sistema Solar.

Com as considerações da sessão anterior e desta, podemos juntar os passos de tempo escolhidos para cada planeta:

Tabela 8: Δt escolhido para a utilização na simulação de cada planeta.

Objeto	Δt
Mercúrio	0,0001
Venus	0,0001
Terra	0,0001
Marte	0,0001
Júpiter	0,0001
Saturno	0,001
Urano	0,001
Netuno	0,01
Plutão	0,01

Apesar de ser aceitável utilizar passos de tempo maiores em alguns casos, escolheu-

se o menor testado que ainda não afeta significativamente a performance do programa. Veremos que em simulações com vários planetas, teremos que manter o menor passo de tempo que não seja pequeno demais a fim de prejudicar a performance sem um aumento significativo de precisão, para que não tenhamos erros com os planetas mais próximos à estrela. Idealmente, usaríamos um diferente para cada planeta, porém isto pode tornar o programa bem mais complicado, logo nos restringimos a fazer um tratamento mais geral do problema.

2.2.2 Influência da velocidade inicial v_0 do corpo no formato da órbita e verificação das Leis de Kepler

Similarmente à seção anterior, vamos testar algumas velocidades iniciais para um planeta e veremos como isso afetará sua órbita, agora com a adição da excentricidade da órbita ao programa. Testaremos as seguintes velocidades iniciais: 2π, $\sqrt{\frac{6\pi^2(1-\epsilon)}{a(1+\epsilon)}}$, $\sqrt{\frac{4\pi^2(1-\epsilon)}{a(1+\epsilon)}}$ e $\sqrt{\frac{8\pi^2(1-\epsilon)}{a(1+\epsilon)}}$. Usaremos Vênus novamente para obter uma comparação com a sessão anterior.

Figura 8: Órbitas de Venus para diferentes velocidades iniciais

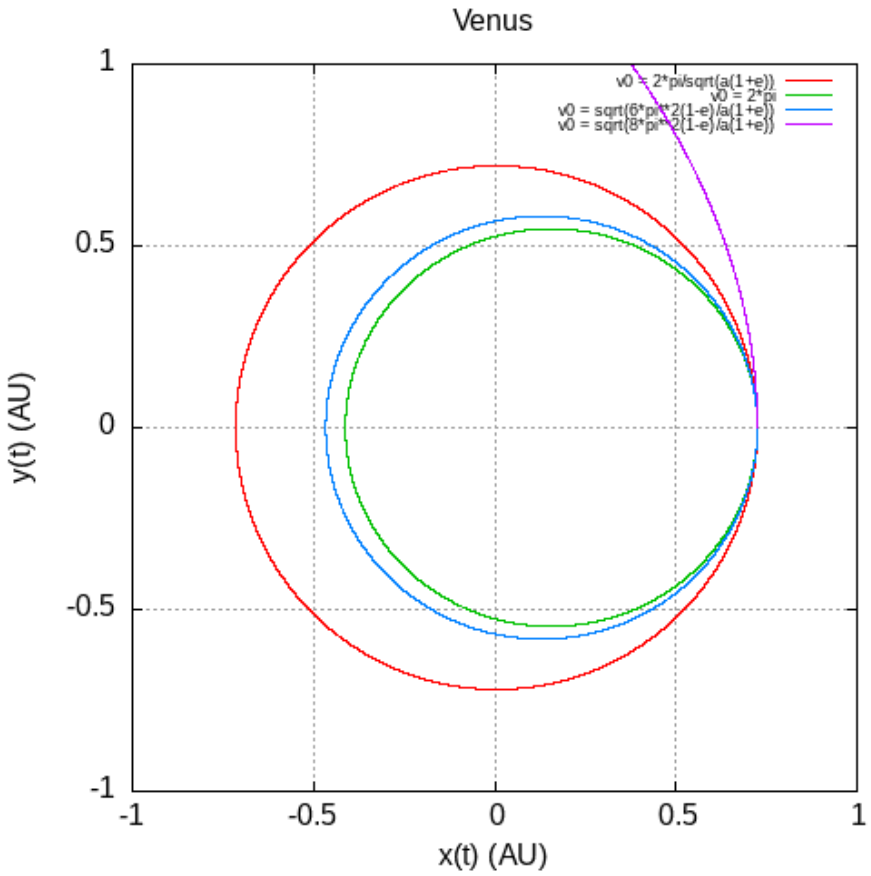


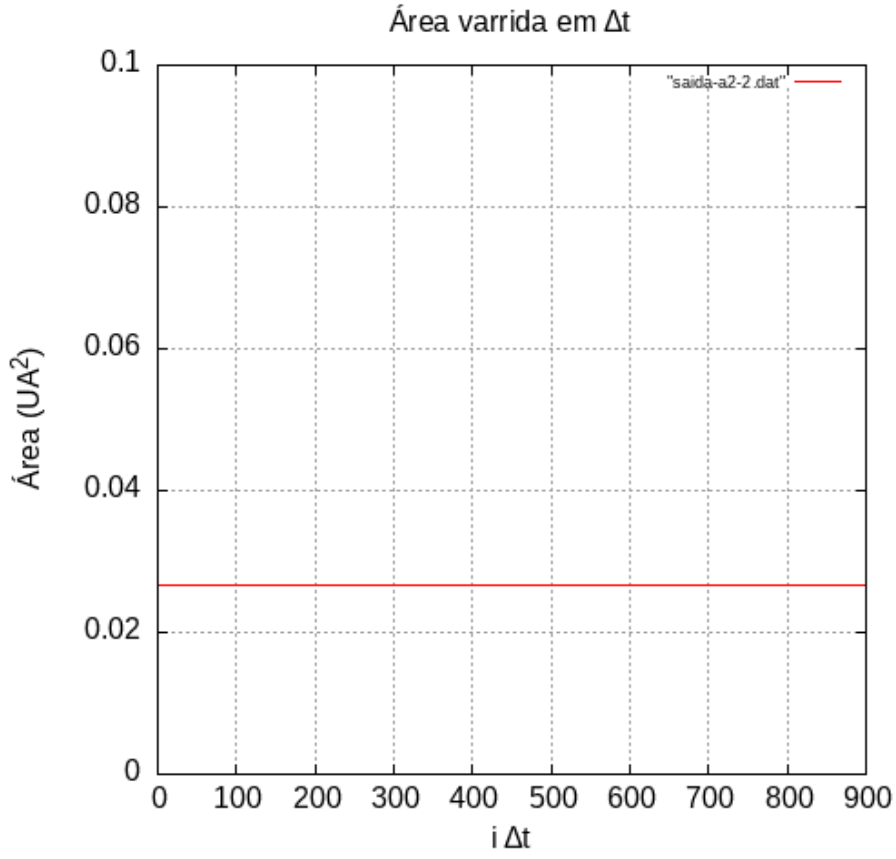
Tabela 9: Teste de órbita circular para Venus em diferentes velocidades iniciais (componente y).

v_{y0}	T^2/R^3	ϵ
2π	0.49270994028554932	0.2750
$\sqrt{\frac{6\pi^2(1-\epsilon)}{a(1+\epsilon)}} \approx 6.529$	0.56652802765160082	0.2170
$\sqrt{\frac{4\pi^2(1-\epsilon)}{a(1+\epsilon)}} \approx 7.353$	0.99999096821825950	0.0070
$\sqrt{\frac{8\pi^2(1-\epsilon)}{a(1+\epsilon)}} \approx 10.400$	-	-

As mesmas considerações utilizadas anteriormente são válidas neste caso. É notável que, desta vez, para a velocidade ideal, atingimos uma razão T^2/R^3 muito mais próxima de 1 do que anteriormente e isto se deve aos ajustes realizados para considerar a excentricidade da órbita.

Para este caso de órbita fechada, nosso teste da 2ª Lei de Kepler produz o seguinte resultado:

Figura 9: Área varrida em um determinado intervalo de tempo $100\Delta t$.



Vemos assim que, sendo constante a área varrida em um determinado intervalo de tempo, mostramos que a 2ª Lei de Kepler é válida nesses casos.

Desta maneira, podemos concluir que nas duas seções da Tarefa A investigamos as questões mais importantes sobre os fundamentos das interações gravitacionais de nosso sistema solar conforme a restrição de dois corpos: verificamos a 1ª Lei de Kepler investigando a excentricidade das órbitas, que devem ser elípticas e concordar com as condições iniciais que definimos para a simulação; verificamos que em órbitas fechadas a 2ª Lei de Kepler, que diz que a área varrida pela linha que liga o planeta à estrela deve ser constante dado um intervalo de tempo fixo (que nos casos circulares é trivial, já que a distância planeta-Sol é sempre a mesma); e verificamos também que a 3ª Lei de Kepler é verdadeira (T^2/a^3 é constante e, em nossas unidades, igual a 1), sendo que considerando a adição das excentricidades, obtemos valores mais próximos de 1. Agora, podemos utilizar os programas criados para simular sistemas com mais corpos, sabendo que no caso mais simples temos concordância com a teoria.

3 Tarefa B: O problema de 3 ou mais corpos

3.1 Tarefa B1: Sistema Sol-Júpiter-Terra

Nesta seção investigaremos o movimento do sistema composto pelo Sol, Júpiter e a Terra. Para isto, além do código criado nas seções anteriores, precisamos incluir as equações de movimento de mais um planeta e sua influência na Terra. Como o Sol representa mais de 99% da massa do sistema solar, nesta primeira abordagem podemos considerar o efeito dos planetas sobre ele desprezível. Desta forma, teremos as seguintes equações de movimento:

Para a Terra:

$$\frac{d^2x_T}{dt^2} = -G\frac{M_S}{r_{TS}^2}(x_T - x_S) - G\frac{M_J}{r_{TJ}^2}(x_T - x_J) \tag{13}$$

$$\frac{d^2y_T}{dt^2} = -G\frac{M_S}{r_{TS}^2}(y_T - y_S) - G\frac{M_J}{r_{TJ}^2}(y_T - y_J) \tag{14}$$

E para Júpiter:

$$\frac{d^2x_J}{dt^2} = -G\frac{M_S}{r_{JS}^2}(x_J - x_S) - G\frac{M_J}{r_{TJ}^2}(x_J - x_T) \tag{15}$$

$$\frac{d^2y_J}{dt^2} = -G\frac{M_S}{r_{JS}^2}(y_J - y_S) - G\frac{M_J}{r_{TJ}^2}(y_J - y_T) \tag{16}$$

Sendo que as distâncias destas equações são dadas por:

$$r_{TS} = \sqrt{(x_T - x_S)^2 + (y_T - y_S)^2} \tag{17}$$

$$r_{JS} = \sqrt{(x_J - x_S)^2 + (y_J - y_S)^2} \tag{18}$$

e

$$r_{TJ} = \sqrt{(x_T - x_J)^2 + (y_T - y_J)^2} \tag{19}$$

As coordenadas do Sol (x_S, y_S) são sempre $(0, 0)$ neste caso. As condições iniciais de Júpiter e da Terra são dadas conforme os dados dispostos nas tabelas das primeiras seções. O código utilizado nesta simulação foi o seguinte:

```
1         PROGRAM orbitas_tres_corpos
2
3         implicit real*8(a-h,o-z)
4         parameter(pi = dacos(-1d0))
5         parameter(aMT = 6d0*10d24)
6         parameter(aMJ = 1.9d0*10d27)
7         parameter(aMS = 2d0*10d30)
8         parameter (gms = 4d0*pi**2)
9         parameter(gmj = gms*(aMJ/aMS))
10
11        parameter (delta_t = 0.0001d0)
12
13        rt = 1d0
14        rj = 5.2d0
15
16        open(10, FILE='saida-b1-terra.dat')
17        open(20, FILE='saida-b1-jupiter.dat')
18
19        t_f = 30d0
20        t = 0d0
21        N = t_f/delta_t
22
23        !Terra
24        xt = rt
25        yt = 0d0
26
27        vxt = 0d0
28        vyt = dsqrt(gms/rt)
29
30        !Jupiter
```

```
31          xj = rj*(1+ej)
32          yj = 0d0
33
34          vxj = 0d0
35          vyj = dsqrt(gms/rj)
36
37          write(10,*) xt, yt
38          write(20,*) xj, yj
39
40          !Terra
41          xm1t = xt
42          ym1t = yt
43
44          xt = xm1t + vxt*delta_t
45          yt = ym1t + vyt*delta_t
46
47          !Jupiter
48          xm1j = xj
49          ym1j = yj
50
51          xj = xm1j + vxj*delta_t
52          yj = ym1j + vyj*delta_t
53
54          write(10,*) xt, yt
55          write(20,*) xj, yj
56
57          t_i= 0d0
58          t_ant = 0d0
59          icount = 0
60
61          do i = 1, N
62              t = t + delta_t
63
64              rts = (xt**2+yt**2)**0.5d0
65              rtj = ((xt-xj)**2+(yt-yj)**2)**0.5d0
66
67              xp1t = 2*xt - xm1t - (delta_t**2)*
68           $     ((gms/rts**3)*xt+(gmj/rtj**3)*(xt-xj))
69              yp1t = 2*yt - ym1t - (delta_t**2)*
70           $     ((gms/rts**3)*yt+(gmj/rtj**3)*(yt-yj))
```

```
71
72              rjs = (xj**2+yj**2)**0.5d0
73
74              xp1j = 2*xj - xm1j - (delta_t**2)*
75          $      ((gms/rjs**3)*xj+(gmj/rtj**3)*(xj-xt))
76              yp1j = 2*yj - ym1j - (delta_t**2)*
77          $      ((gms/rjs**3)*yj+(gmj/rtj**3)*(yj-yt))
78
79
80              xm1t = xt
81              ym1t = yt
82
83              xt = xp1t
84              yt = yp1t
85
86              xm1j = xj
87              ym1j = yj
88
89              xj = xp1j
90              yj = yp1j
91
92              write(10, *) xt, yt
93              write(20, *) xj, yj
94          end do
95
96          close(10)
97          close(20)
98          end program
```

Este código segue a mesma lógica dos anteriores, apenas contando com a adição de um corpo (Júpiter) e dos termos dados pelas equações acima. Foram retirados os contadores de período e excentricidade dada a já verificação das leis de movimento na seção anterior.

Como imagem da órbita, obtemos:

Figura 10: Simulação do Sistema Sol-Terra-Júpiter

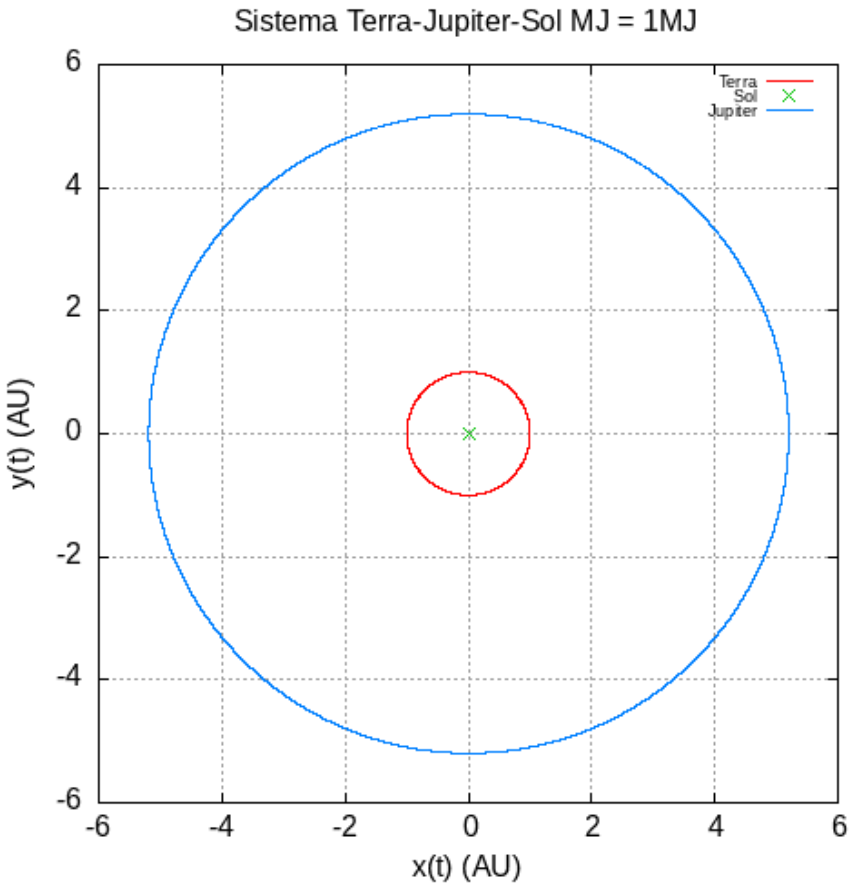
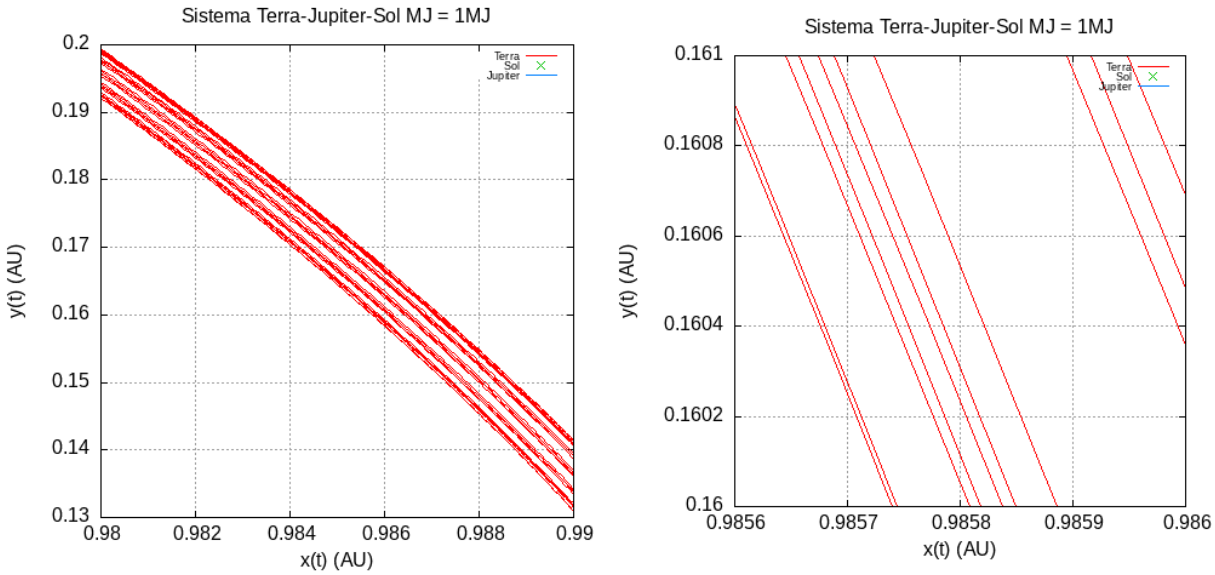


Figura 11: Ampliação da órbita terrestre sob o efeito do Sol e de Júpiter.



É possível notar que agora a órbita da Terra não é mais periódica: ela passa por distâncias ligeiramente diferentes do ponto onde passou no período anterior, efeito cor-

relacionado com a influência gravitacional de Júpiter. Vemos que essas distâncias são da ordem de $10^{-4} - 10^{-5} AU$. Este fato pode explicar por que fizemos, historicamente, tantos ajustes nos calendários para obter boas precisões com datas. Também podemos considerar que, dados determinados períodos de tempo maiores, podemos ter ciclos de variações no clima devido à influência gravitacional de Júpiter (e também de Vênus, que está fora do escopo do observado aqui), ainda que este efeito seja mais complicado do que o investigado nesta seção.

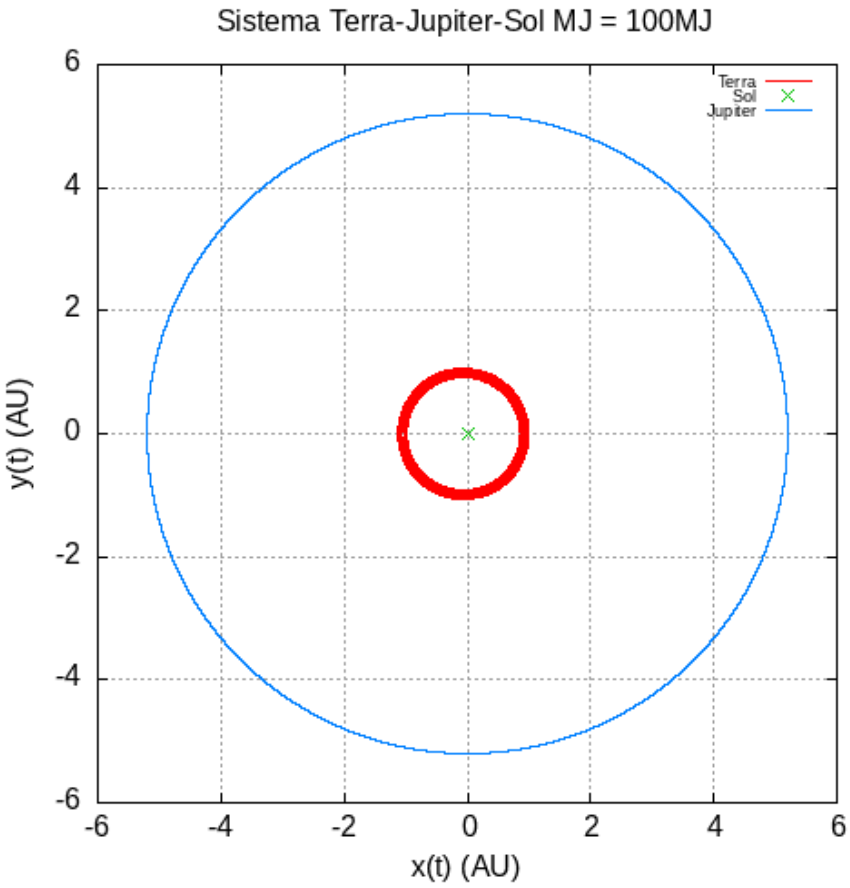
Planetas como Júpiter podem ter uma importância gigantesca em sistemas planetários: eles servem como "protetores"de planetas como órbitas mais internas, provendo estabilidade de órbitas quase circulares para estes, características importantes para o desenvolvimento de vida (já que órbitas muito elíticas significam climas mais instáveis em curtos períodos de tempo). Além disso, estes planetas gigantes podem alterar significantemente órbitas de objetos pequenos que acabam se aproximando (como cometas) para elipses com excentricidades próximas de 1 que fazem com que estes objetos demorem bastante para retornar.

3.2 Tarefa B2: Efeitos do aumento da massa de Júpiter no sistema Sol-Júpiter-Terra

O sistema da seção anterior apresentou órbitas estáveis, ainda que, ao olhar de perto, vêssemos que a Terra mudava ligeiramente de trajetória a cada período. Para investigar maiores efeitos que um terceiro corpo pode causar, podemos aumentar a massa de Júpiter no código anterior e ver o comportamento das órbitas para estes novos valores.

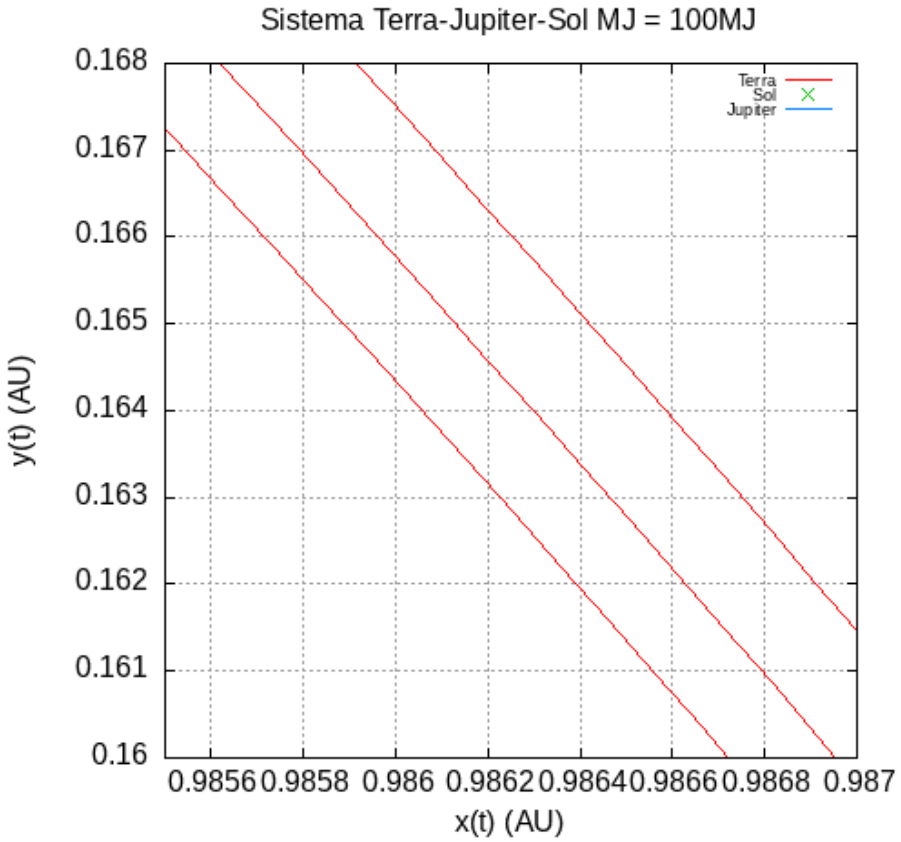
Aumentando-a para 100 vezes a massa de Júpiter original, obtemos o seguinte sistema:

Figura 12: Simulação do Sistema Sol-Terra-Júpiter, M_J aumentada para $100M_J$



Vemos aqui que agora a órbita terrestre começa a a passar por distâncias bem diferentes a cada ano, como observamos através da trajetória no gráfico - a espessura da linha não foi alterada em relação ao gráfico de $1M_J$, mas sim a órbita se tornou pertubada o suficiente por Júpiter para divergir de uma órbita circular perfeita, um pouco mais a cada ano. Ampliando-a, temos:

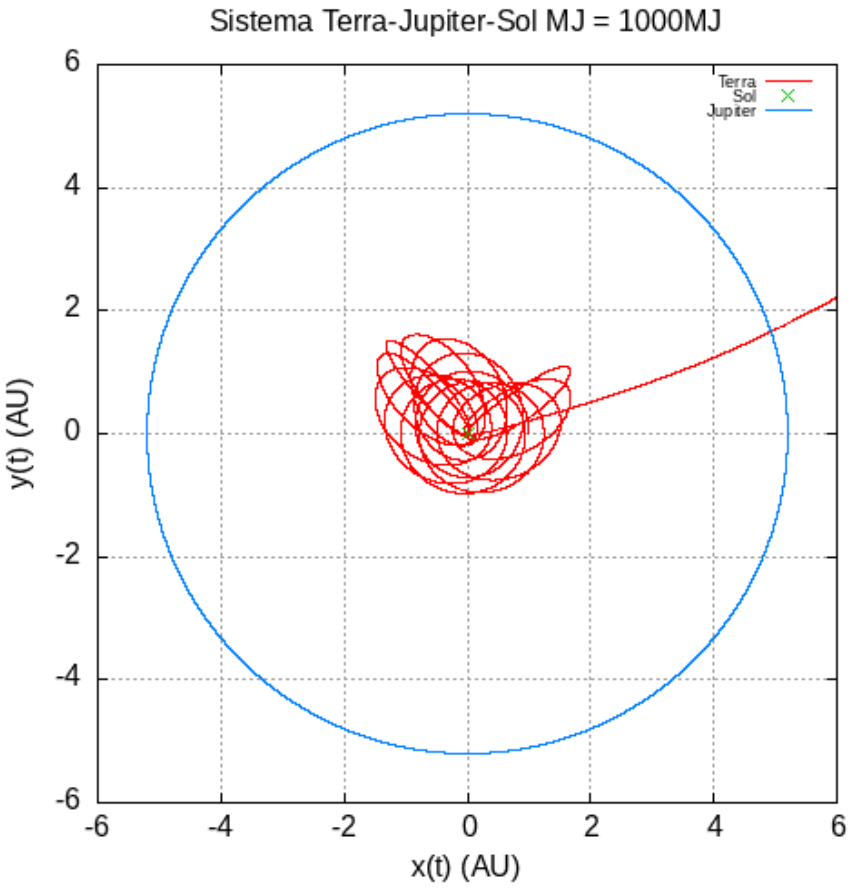
Figura 13: Ampliação da órbita terrestre, M_J aumentada para $100M_J$



Sendo notável que as distâncias típicas que a Terra passa a cada ano de sua última posição são bem maiores neste caso (da ordem de 10^{-4}).

Se aumentarmos ainda mais a massa de Júpiter, desta vez para 1000 vezes a original, começamos a notar que a órbita terrestre se torna bastante instável:

Figura 14: Simulação do Sistema Sol-Terra-Júpiter, massa aumentada para $100M_J$



No entanto, esta representação não está correta, pois, para esta massa de Júpiter, começamos a ver efeitos significativos no movimento do Sol. Portanto, para obtermos uma simulação mais acurada, devemos adicionar as equações de movimento do Sol.

As distâncias e equações da Terra e do Sol continuam as mesmas do código anterior. As do Sol serão:

$$\frac{d^2x_S}{dt^2} = -G\frac{M_T}{r_{TS}^2}(x_S - x_T) - G\frac{M_J}{r_{TJ}^2}(x_S - x_J) \tag{20}$$

$$\frac{d^2y_S}{dt^2} = -G\frac{M_T}{r_{TS}^2}(y_S - y_T) - G\frac{M_J}{r_{TJ}^2}(y_S - y_J) \tag{21}$$

As condições iniciais serão definidas a partir dos raios médios de Júpiter e da Terra e com o Sol na origem. Por generalidade, vamos adicionar um ângulo θ que altera as coordenadas iniciais de Júpiter para que este não seja necessariamente solto alinhado com a Terra. As velocidades iniciais serão dadas por:

$$|\vec{v_T} - \vec{v_S}| = \sqrt{GM_S\frac{1}{r_{TS}}(1 + \frac{M_T}{M_S})} \tag{22}$$

$$|\vec{v_J} - \vec{v_S}| = \sqrt{GM_S \frac{1}{r_{JS}}(1 + \frac{M_J}{M_S})} \tag{23}$$

$$\vec{v_S} = -\frac{M_T}{M_S}\vec{v_T} - \frac{M_J}{M_S}\vec{v_J} \tag{24}$$

Ou seja,

$$v_{xt} = \frac{\frac{M_T}{M_S}\sqrt{\frac{GM_S(1+M_T/M_S)}{r_{TS}}}\sin(\theta)}{\frac{(M_S+M_T+M_J)}{M_S}} \tag{25}$$

$$v_{yt} = \frac{(1+\frac{M_J}{M_S})\sqrt{\frac{GM_S(1+M_T/M_S)}{r_{TS}}} - \frac{M_J}{M_S}\sqrt{\frac{GM_S(1+M_J/M_S)}{r_{JS}}}\cos(\theta)}{\frac{(M_S+M_T+M_J)}{M_S}} \tag{26}$$

$$v_{xJ} = \frac{-(1+\frac{M_T}{M_S})\sqrt{\frac{GM_S(1+M_J/M_S)}{r_{JS}}}\sin(\theta)}{\frac{(M_S+M_T+M_J)}{M_S}} \tag{27}$$

$$v_{yJ} = -\frac{\frac{M_J}{M_S}\sqrt{\frac{GM_S(1+M_T/M_S)}{r_{TS}}} + (1+\frac{M_J}{M_S})\sqrt{\frac{GM_S(1+M_J/M_S)}{r_{JS}}}\cos(\theta)}{\frac{(M_S+M_T+M_J)}{M_S}} \tag{28}$$

$$v_{xS} = \frac{\sqrt{\frac{GM_S(1+M_J/M_S)}{r_{JS}}}\sin(\theta)}{\frac{(M_S+M_T+M_J)}{M_S}} \tag{29}$$

$$v_{yS} = \frac{-\frac{M_T}{M_S}\sqrt{\frac{GM_S(1+M_T/M_S)}{r_{TS}}} + \frac{M_J}{M_S}\sqrt{\frac{GM_S(1+M_J/M_S)}{r_{JS}}}\cos(\theta)}{\frac{(M_S+M_T+M_J)}{M_S}} \tag{30}$$

O centro de massa do sistema, por sua vez, possuirá as seguintes coordenadas:

$$x_{cm} = \frac{x_S + x_T + x_J}{M_S + M_T + M_J} \tag{31}$$

$$y_{cm} = \frac{y_S + y_T + y_J}{M_S + M_T + M_J} \tag{32}$$

O código englobando estas mudanças é o seguinte:

```
1       PROGRAM orbitas_tres_corpos
2
3       implicit real*8(a-h,o-z)
4       parameter(pi = dacos(-1d0))
5
6       parameter(aMT = 6d0*10d24)
7       parameter(aMJ = 1000*1.9d0*10d27)
8       parameter(aMS = 2d0*10d30)
9
```

```
10          parameter(aMTS = aMT/aMS)
11          parameter(aMJS = aMJ/aMS)
12
13          parameter (gms = 4d0*pi**2)
14          parameter(gmt = gms*aMTS)
15          parameter(gmj = gms*aMJS)
16
17          parameter (dt = 0.0001d0)
18
19          dimension r(1:3)
20          dimension x(1:3), y(1:3)
21          dimension vx(1:3), vy(1:3)
22          dimension xm1(1:3), ym1(1:3)
23          dimension xp1(1:3), yp1(1:3)
24          dimension d(1:3) !TS, TJ, JS
25
26          parameter(r = (/1d0, 5.2d0, 0d0/))
27
28          t_f = 15d0
29          t = 0d0
30          N = t_f/dt
31
32          aM = aMT + aMS + aMJ
33          raM = aM/aMS
34
35          theta = 0d0
36
37          open(10, FILE='terra.dat')
38          open(20, FILE='jupiter.dat')
39          open(30, FILE='sol.dat')
40
41          !Terra
42          x(1) = r(1)
43          y(1) = 0d0
44
45          !Jupiter
46          x(2) = r(2)*dcos(theta)
47          y(2) = r(2)*dsin(theta)
48
49          !Sol
```

```
50          x(3) = 0d0
51          y(3) = 0d0
52
53          d(1) = dsqrt((x(1)-x(3))**2+(y(1)-x(3))**2)
54          d(3) = dsqrt((x(2)-x(3))**2+(y(2)-x(3))**2)
55
56          tt = dsqrt((gms+gmt)/d(1))
57          tj = dsqrt((gms+gmj)/d(3))
58
59          vx(1) = aMJS*tj*dsin(theta)/raM
60          vx(2) = -(1+aMTS)*tj*dsin(theta)/raM
61          vx(3) = tj*dsin(theta)/raM
62
63          vy(1) = ((1+aMJS)*tt - aMJS*tj*dcos(theta))/raM
64          vy(2) = -(aMTS*tt +(1+aMTS)*tj*dcos(theta))/raM
65          vy(3) = (-(aMTS*tt)+aMJS*tj*dcos(theta))/raM
66
67          xcm = (x(1) + x(2) + x(3))/aM
68          ycm = (y(1) + y(2) + y(3))/aM
69
70
71          do i = 1, 3
72                x(i) = x(i) - xcm
73                y(i) = y(i) - ycm
74
75                xm1(i) = x(i)
76                ym1(i) = y(i)
77
78                x(i) = xm1(i) + vx(i)*dt
79                y(i) = ym1(i) + vy(i)*dt
80
81                write(i*10,*) x(i), y(i)
82          end do
83
84          t_i= 0d0
85          t_ant = 0d0
86          icount = 0
87
88          do j = 1, N
89              t = t + dt
```

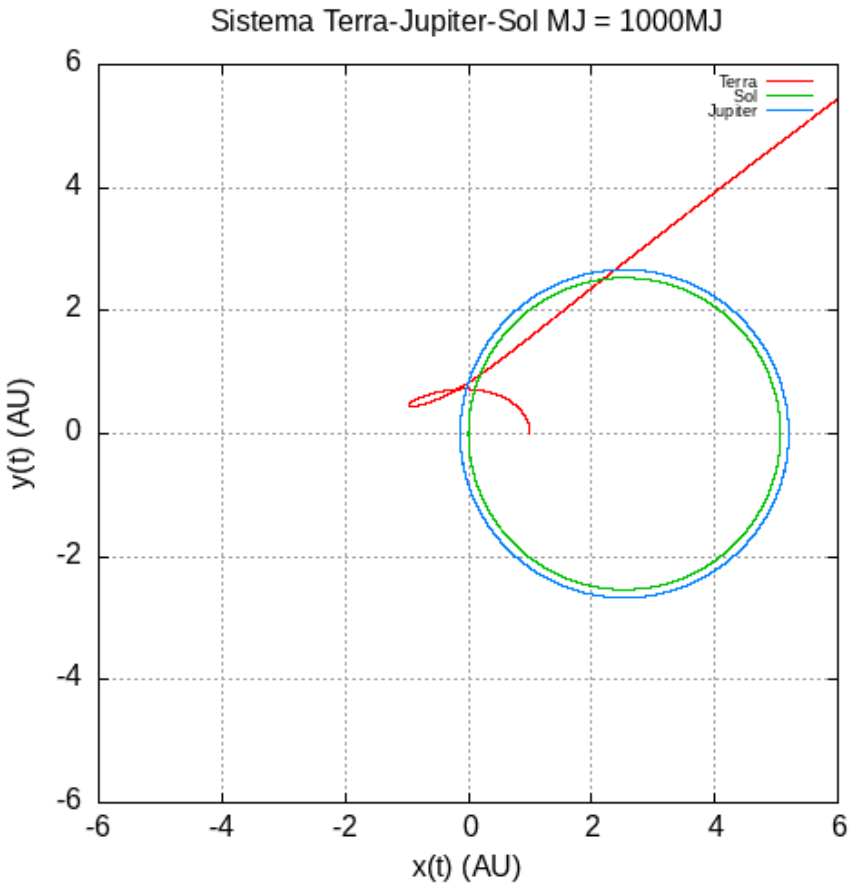
```
90
91          d(1) = dsqrt((x(1)-x(3))**2+(y(1)-x(3))**2)
92          d(2) = dsqrt((x(1)-x(2))**2+(y(1)-y(2))**2)
93
94          xp1(1) = 2*x(1) - xm1(1) - (dt**2)*
95       $  ((gms/d(1)**3)*(x(1)-x(3))+(gmj/d(2)**3)*(x(1)-x(2)))
96          yp1(1) = 2*y(1) - ym1(1) - (dt**2)*
97       $  ((gms/d(1)**3)*(y(1)-y(3))+(gmj/d(2)**3)*(y(1)-y(2)))
98
99          d(3) = dsqrt((x(2)-x(3))**2+(y(2)-y(3))**2)
100
101         xp1(2) = 2*x(2) - xm1(2) - (dt**2)*
102      $  ((gms/d(3)**3)*(x(2)-x(3))+(gmt/d(2)**3)*(x(2)-x(1)))
103         yp1(2) = 2*y(2) - ym1(2) - (dt**2)*
104      $  ((gms/d(3)**3)*(y(2)-y(3))+(gmt/d(2)**3)*(y(2)-y(1)))
105
106         xp1(3) = 2*x(3) - xm1(3) - (dt**2)*
107      $  ((gmt/d(1)**3)*(x(3)-x(1))+(gmj/d(3)**3)*(x(3)-x(2)))
108         yp1(3) = 2*y(3) - ym1(3) - (dt**2)*
109      $  ((gmt/d(1)**3)*(y(3)-y(1))+(gmj/d(3)**3)*(y(3)-y(2)))
110
111         xcm = (xp1(1) + xp1(2) + xp1(3))/aM
112         ycm = (yp1(1) + yp1(2) + yp1(3))/aM
113
114         do i = 1, 3
115                  xm1(i) = x(i)
116                  ym1(i) = y(i)
117
118                  x(i) = xp1(i)
119                  y(i) = yp1(i)
120
121                  x(i) = x(i) - xcm
122                  y(i) = y(i) - ycm
123
124                  write(i*10,*) x(i), y(i)
125            end do
126       end do
127
128       close(10)
129       close(20)
```

```
130         close(30)
131         end program
```

Sua lógica é similar ao da seção anterior, mas desta vez as variáveis antigas são substituídas por vetores. A posição 1 corresponde à Terra, a 2 a Júpiter e a 3 ao Sol. Para simplificar a escrita, vários termos que aparecem repetidos foram calculados uma vez apenas e seu valor numérico foi utilizado nos cálculos.

A saída obtida considerando $\theta = 0$ e $t_f = 15$ *anos* é a seguinte:

Figura 15: Simulação do Sistema Sol-Terra-Júpiter, massa aumentada para $1000 M_J$



Nele vemos que a Terra, dada na posição escolhida, é rapidamente ejetada do sistema solar. No entanto, notamos que Júpiter e o Sol permanecem em órbita, com um formato que lembra muito um sistema binário de estrelas. De fato, quando observamos a razão entre a massa de Júpiter e do Sol, temos que Júpiter é aproximadamente 1000 vezes menos massivo que o Sol. Se aumentamos sua massa 1000 vezes, teríamos um corpo com aproximadamente a massa solar orbitando em nosso sistema (Júpiter não poderia mais ser um planeta) e isto poderia explicar o por quê das órbitas ficarem tão próximas e similares ao observado em sistemas binários de estrelas. Sabemos que existem planetas que orbitam sistemas binários, mas estes precisam ter condições específicas para permanecerem ali, algo que a Terra, neste exemplo, não obteve.

3.3 Tarefa B3: Órbitas de asteroides e as lacunas de Kirkwood

Agora, após investigar o Sistema Sol-Terra-Júpiter, podemos mudar para outro problema que engloba mais corpos: o cinturão de asteroides de Júpiter.

Para isto, vamos considerar três objetos com velocidades e raios orbitais dados na seguinte tabela:

Tabela 10: Tabela com raios e velocidades dos objetos simulados nesta seção.

Objeto	Raio (UA)	Velocidade (UA/ano)
Asteroide I	3.000	3.628
Asteroide II	3.276	3.471
Asteroide III	3.700	3.267
Júpiter	5.200	2.755

Como a massa de um asteroide é insignificante próxima ao Sol e a Júpiter, podemos considerar no programa apenas o efeito destes dois astros sobre o movimento dos asteroides e, novamente, apenas o efeito do Sol em Júpiter, sendo a estrela fixa na origem.

Seguindo as equações já montadas para a Terra nas outras seções, temos o seguinte código:

```
1         PROGRAM orbitas_asteroides
2
3         implicit real*8(a-h,o-z)
4         parameter(pi = dacos(-1d0))
5         parameter(aMJ = 1.9d0*10d27)
6         parameter(aMS = 2d0*10d30)
7         parameter (gms = 4d0*pi**2)
8         parameter(gmj = gms*(aMJ/aMS))
9
10        parameter (dt = 0.0001d0)
11
12        dimension r(1:4)
13        dimension x(1:4), y(1:4)
14        dimension vx(1:4), vy(1:4)
15        dimension xm1(1:4), ym1(1:4)
16        dimension xp1(1:4), yp1(1:4)
17        dimension rs(1:4), rj(1:3)
18
19        parameter(r = (/3d0, 3.276d0, 3.7d0, 5.2d0/))
20        parameter(vx = (/0d0, 0d0, 0d0, 0d0/))
```

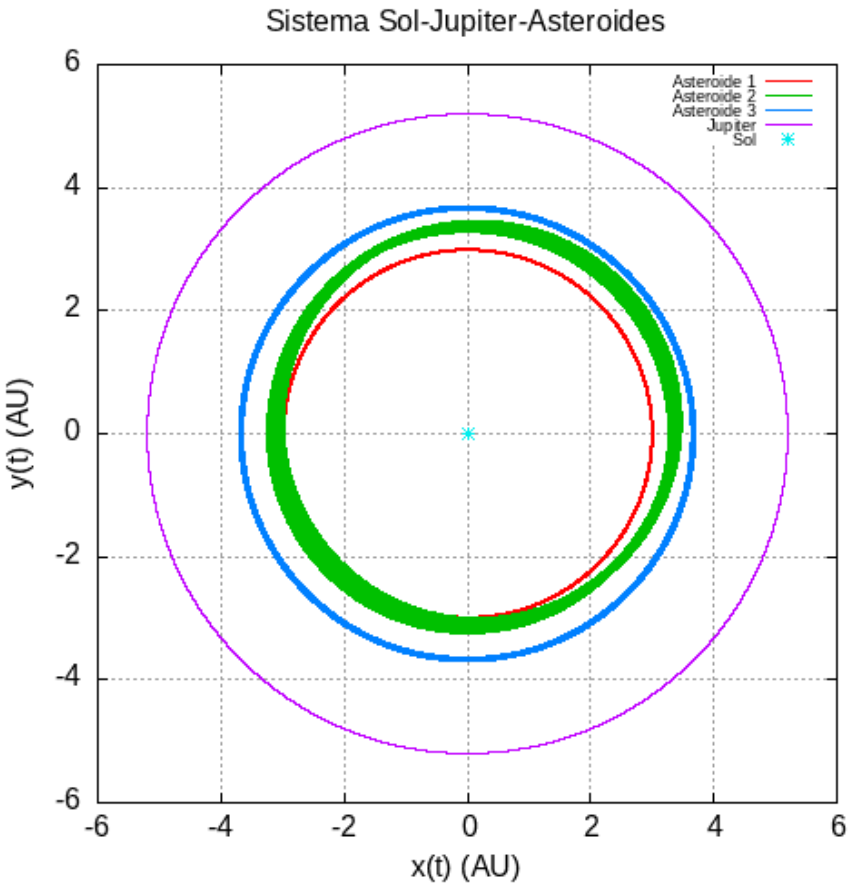
```
21          parameter(vy = (/3.628d0, 3.471d0, 3.267d0, 2.755d0/))
22
23          open(10, FILE='ast1.dat')
24          open(20, FILE='ast2.dat')
25          open(30, FILE='ast3.dat')
26          open(40, FILE='jupiter.dat')
27
28          t_f = 200d0
29          t = 0d0
30          N = t_f/dt
31
32          do i = 1, 4
33                x(i) = r(i)
34                y(i) = 0d0
35
36                write(i*10, *) x(i), y(i)
37
38                xm1(i) = x(i)
39                ym1(i) = y(i)
40
41                x(i) = xm1(i) + vx(i)*dt
42                y(i) = ym1(i) + vy(i)*dt
43
44                write(i*10, *) x(i), y(i)
45          end do
46
47          do j = 1, N
48              t = t + dt
49
50              do i = 1, 3
51                rs(i) = (x(i)**2+y(i)**2)**0.5d0
52                rj(i) = ((x(i)-x(4))**2+(y(i)-y(4))**2)**0.5d0
53
54                xp1(i) = 2*x(i) - xm1(i) - (dt**2)*
55           $    ((gms/rs(i)**3)*x(i)+(gmj/rj(i)**3)*(x(i)-x(4)))
56                yp1(i) = 2*y(i) - ym1(i) - (dt**2)*
57           $    ((gms/rs(i)**3)*y(i)+(gmj/rj(i)**3)*(y(i)-y(4)))
58
59              end do
60
```

```
61              !Jupiter
62              rs(4) = (x(4)**2+y(4)**2)**0.5d0
63
64              xp1(4) = 2d0*x(4) - xm1(4) - (dt**2)*((gms/rs(4)**3)*x(4))
65              yp1(4) = 2d0*y(4) - ym1(4) - (dt**2)*((gms/rs(4)**3)*y(4))
66
67              do i = 1, 4
68                xm1(i) = x(i)
69                ym1(i) = y(i)
70
71                x(i) = xp1(i)
72                y(i) = yp1(i)
73
74                write(i*10,*) x(i), y(i)
75              end do
76
77          end do
78
79          close(10)
80          close(20)
81          close(30)
82          close(40)
83          end program
```

A lógica é similar aos outros códigos, com a adição dos loops que performam o mesmo trabalho para cada objeto com sua respectiva grandeza guardada em um vetor em posição de acordo com a Tabela 10.

Como resultados, para um tempo de simulação $t_f = 200\ anos$ (é necessário aumentar o tempo para observar propriamente os efeitos de Júpiter nas órbitas dos asteroides) temos:

Figura 16: Simulação de três asteroides sob os efeitos gravitacionais de Júpiter e do Sol.



Podemos notar que o primeiro e o terceiro apresentam pequenas perturbações em suas órbitas, mas este efeito se torna muito mais notável para o segundo. Para entender por que isto ocorre, devemos nos perguntar o que o asteroide do meio possui de especial em relação aos outros dois.

A resposta está em sua distância ao Sol e, consequentemente, ao seu período orbital. Historicamente, de acordo com os planetas já conhecidos e suas respectivas distâncias conhecidas, acreditava-se que estes seguiriam um padrão em seus raios orbitais de acordo com a sequência numérica $3, 6, 12, 24, ...$ (somando 4 e dividindo o resultado por 10) esse padrão é conhecido como padrão ou fórmula de Titus-Bode e, se fizermos uma tabela dos planetas em ordem crescente de sua distância ao Sol, observamos que eles seguem bem proximamente esta relação. Porém, entre Marte e Júpiter era estimado haver um planeta: conforme mais observações foram feitas, não foi encontrado um objeto mas vários objetos pouco massivos com raios orbitais de aproximadamente $3UA$ que descobrirmos ser, na verdade, um cinturão de asteroides.

Estudando as novas descobertas, notou-se um padrão nos raios orbitais dos asteroides: Kirkwood, no século XIX, fez um gráfico do número de asteroides em relação à sua distância ao Sol e percebeu que existem lacunas nesta distribuição, determinadas distâncias onde essencialmente não era observado asteroide algum. Essas lacunas ficaram conhecidas

como lacunas de Kirkwood.

Tais espaços vazios na distribuição têm relação direta com o fenômeno de ressonância orbital e com a órbita de Júpiter (como Kirkwood mostrou). A ressonância orbital ocorre quando corpos exercem uma influência periódica e regular sobre outros corpos quando seus períodos estão relacionados por uma razão de números inteiros. Ela é similar às ressonâncias que já estamos acostumados - se exercermos, por exemplo, uma força periódica na frequência natural de um sistema, esta terá um efeito cumulativo no movimento realizado. O caso orbital faz com que o efeito gravitacional entre dois corpos aumente, possivelmente ocasionando uma interação instável (assim como observamos no asteroide II através da imagem) onde os corpos trocam momentum e possuem suas órbitas alteradas até que a ressonância não exista mais - mas cabe notar que em alguns casos o sistema pode ser estável.

A ressonância orbital é o motivo pelo qual existem lacunas nos anéis de Saturno e também um dos efeitos observados para a atual definição de planeta: em ressonâncias 1:1 (raios orbitais similares), corpos mais massivos ejetam menores que compartilham suas órbitas, processo que faz parte de um maior que chamamos de "limpar a vizinhança". É interessante também notar que Netuno e Plutão possuem uma ressonância 2:3. Ademais, o asteroide II de nossa simulação se encontra em ressonância 2:1 com Júpiter, o que explica seu comportamento divergente dos demais. Um asteroide nesta distância pode se manter em ressonância com Júpiter por um bom tempo e, subitamente, mudar sua órbita drasticamente - por exemplo, se aproximando mais de Marte, o que pode alterar permanentemente sua órbita devido ao efeito gravitacional deste outro corpo. Assim, podemos entender a existência destas lacunas. Mais interessante ainda, este comportamento súbito é bem parecido com o comportamento caótico (como o observado para o pêndulo forçado no projeto anterior).

Este é um dos motivos pelo qual o problema de mais de 2 corpos pode se tornar tão complicado: o surgimento de comportamentos caóticos. Isto não é restrito apenas a essa situação dos asteroides e dos aneis de Saturno - acredita-se que o movimento de Plutão também seja caótico em largas escalas de tempo, por exemplo.

3.4 Tarefa B4: Simulação de N-corpos considerando efeitos de todos os planetas

Depois de investigar várias situações de interações gravitacionais, podemos combinar os casos estudados em uma simulação com todos os objetos mencionados previamente, mas desta vez considerando os efeitos gravitacionais de todos eles sobre o movimento de cada um.

Como vimos na seção B2, no caso realístico onde o Sol corresponde à parte majoritária da massa do Sistema Solar, podemos considerar a simplificação de nossa estrela parada na origem do sistema de coordenadas. Este fato será utilizado na presente simulação.

Também optou-se por considerar a característica elíptica das órbitas nas condições iniciais.

Ademais, as equações de movimento são similares às já discutidas nas seções anteriores. O código produzido foi o seguinte:

```
1         PROGRAM sistema_solar
2
3         implicit real*8(a-h,o-z)
4
5         parameter(nb = 9)
6         parameter(pi = dacos(-1d0))
7         parameter(aMS = 2d0*10d30)
8         parameter (gms = 4d0*pi**2)
9
10        parameter (dt = 0.0005d0)
11
12        dimension aM(1:nb)
13        dimension a(1:nb), e(1:nb)
14        dimension x(1:nb), y(1:nb)
15        dimension vx(1:nb), vy(1:nb)
16        dimension xm1(1:nb), ym1(1:nb)
17        dimension xp1(1:nb), yp1(1:nb)
18        dimension r(nb, nb+1)
19
20        parameter(aM = (/2.4d0*10d23, 4.9d0*10d24, 6d0*10d24,
21       $   6.6d0*10d23, 1.9d0*10d27, 5.7d0*10d26, 8.8d0*10d25,
22       $   1.03d0*10d26, 1.3d0*10d22/))
23        parameter(a = (/ 0.39d0, 0.72d0, 1d0, 1.52d0, 5.2d0, 9.24d0,
24       $   19.19d0, 30.06d0, 39.53d0/))
25        parameter(e = (/ 0.206d0, 0.007d0, 0.017d0, 0.093d0, 0.048d0,
26       $   0.056d0, 0.046d0, 0.010d0, 0.248d0/))
27
28        open(10, FILE='mercurio.dat')
29        open(20, FILE='venus.dat')
30        open(30, FILE='terra.dat')
31        open(40, FILE='marte.dat')
32        open(50, FILE='jupiter.dat')
33        open(60, FILE='saturno.dat')
34        open(70, FILE='urano.dat')
35        open(80, FILE='netuno.dat')
36        open(90, FILE='plutao.dat')
```

```
37
38          t_f = 500d0
39          t = 0d0
40          N = t_f/dt
41
42          do i = 1, nb
43                x(i) = a(i)*(1+e(i))
44                y(i) = 0d0
45
46                vx(i) = 0d0
47                vy(i) = dsqrt(gms*(1-e(i))/a(i)*(1+e(i)))
48
49                write(i*10, *) x(i), y(i)
50
51                xm1(i) = x(i)
52                ym1(i) = y(i)
53
54                x(i) = xm1(i) + vx(i)*dt
55                y(i) = ym1(i) + vy(i)*dt
56
57                write(i*10, *) x(i), y(i)
58          end do
59
60          do j = 1, N
61              t = t + dt
62
63              do i = 1, nb
64                r(i, 1) = dsqrt(x(i)**2+y(i)**2)
65
66                xp1(i) = 2*x(i) - xm1(i) - (dt**2)*((gms/r(i, 1)**3)*x(i))
67                yp1(i) = 2*y(i) - ym1(i) - (dt**2)*((gms/r(i, 1)**3)*y(i))
68
69                do k = 1, nb
70
71                  if(k .ne. i) then
72                      r(i, k+1) = dsqrt((x(i)-x(k))**2+(y(i)-y(k))**2)
73                      xp1(i) = xp1(i) - (dt**2)*
74          $            ((((aM(k)/aMs)*gms)/r(i, k+1)**3)*(x(i)-x(k)))
75                      yp1(i) = yp1(i) - (dt**2)*
76          $            ((((aM(k)/aMs)*gms)/r(i, k+1)**3)*(y(i)-y(k)))
```

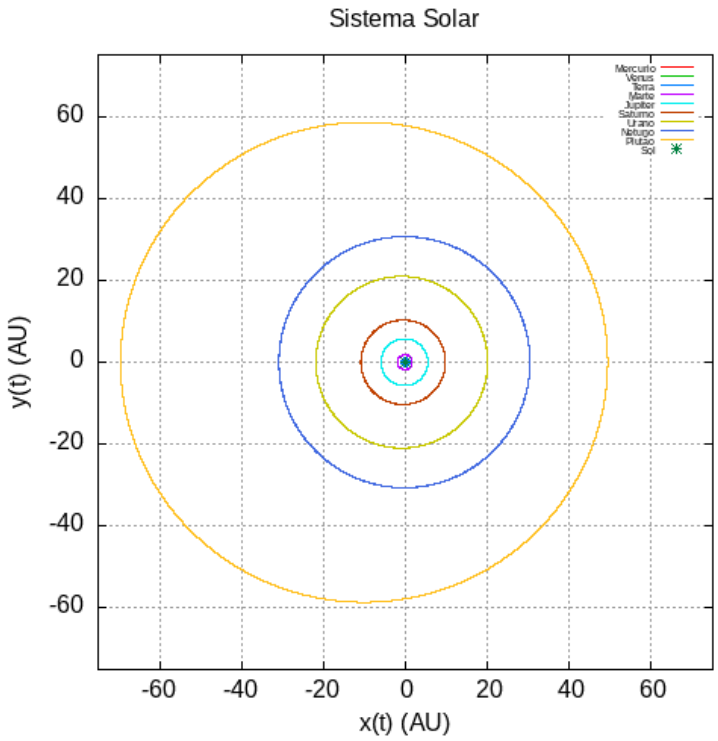
```
77                  end if
78
79                end do
80
81                xm1(i) = x(i)
82                ym1(i) = y(i)
83
84                x(i) = xp1(i)
85                y(i) = yp1(i)
86
87                write(i*10,*) x(i), y(i)
88
89              end do
90            end do
91
92            do i = 1, nb
93                close(i*10)
94            end do
95
96            end program
```

Logicamente, este código é equivalente aos anteriores. Para simplificar a escrita e organização, os dados necessários sobre os planetas (semi-eixo maior, excentricidade e massa) foram dispostos em vetores cujo índice corresponde aos planetas por ordem de distância ao Sol. Todas as outras grandezas também são alocadas em vetores, com exceção da distância entre dois objetos, alocada em uma matriz onde a linha indica o planeta ao qual se está atualizando as coordenadas e a coluna indica o objeto ao qual a distância se refere, sendo que a primeira coluna é reservada para o Sol e as demais correspondem aos planetas na ordem já mencionada.

No loop principal de iterações, realizamos primeiramente o cálculo referente à interação Sol-planeta. Em seguida, um loop secundário percorre os demais planetas adicionando a interação gravitacional entre o primeiro e o atual percorrido à coordenada que se está atualizando. É importante notar que só se deve calcular a iteração caso o planeta seja diferente do que tem as coordenadas atualizadas, caso contrário teremos uma distância nula e uma consequente divisão por 0 não desejada. O restante das atualizações é similar ao código dos asteroides.

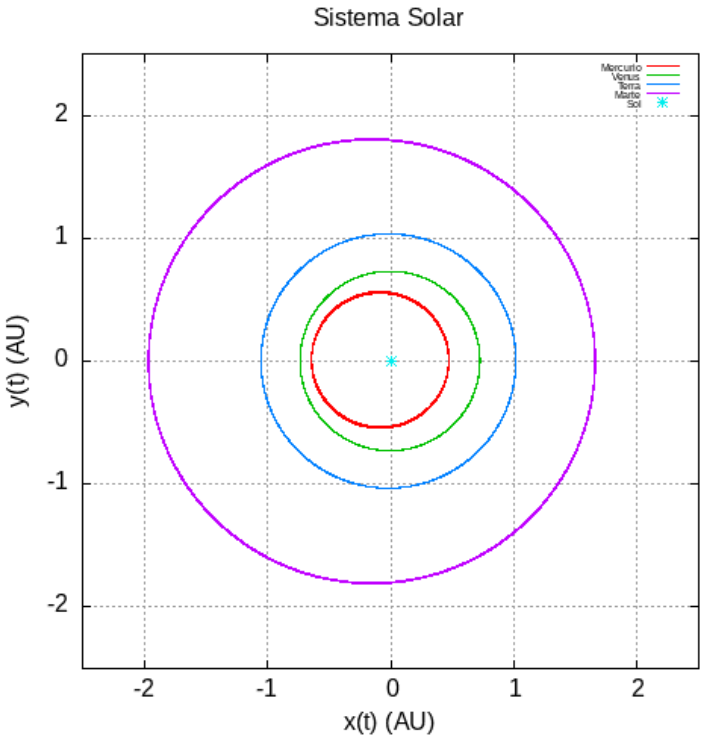
Como resultado, obtemos a seguinte imagem:

Figura 17: Simulação do Sistema Solar



Como os planetas exteriores possuem órbitas grandes, temos também a ampliação deste mesmo gráfico para a visualização das órbitas anteriores a Júpiter:

Figura 18: Simulação do Sistema Solar. Ampliação para a visualização das órbitas menores.



Conforme já esperado, não vemos grandes alterações em relação aos plots já observados

nas simulações mais simples: o Sol ainda é o objeto mais massivo e com maior influência sobre cada órbita, sendo que os efeitos de outros corpos - principalmente Júpiter, o segundo objeto mais massivo - são vistos como pequenos desvios a cada período. Os planetas exercem maior influência sobre sua vizinhança direta, em objetos menos massivos como luas e asteroides.

É importante notar que esta simulação ainda não é totalmente acurada: em primeiro lugar, não estamos considerando os Δt ideais para cada planeta e sim apenas um valor para todos eles, para simplificar o código. Em segundo lugar, foi necessário aumentar o Δt em relação ao estabelecido na primeira sessão para $\Delta t = 0,0005$ por conta de questões computacionais relacionadas à criação do gráfico. Também estamos colocando todos os planetas em um plano, sendo que, na realidade, eles não são perfeitamente alinhados: a órbita entre um deles e o Sol é um plano, mas estes vários planos não são sobrepostos, possuindo leves inclinações em relação uns aos outros. Outro ponto a se considerar é a escolha das condições iniciais: por simplicidade, escolheu-se o valor padrão que utilizamos anteriormente, mas os planetas provavelmente nunca se encontram alinhados em uma linha imaginária na vida real e isto pode afetar ligeiramente a trajetória simulada neste exemplo. Porém, conclui-se que podemos ainda obter características chave de nosso sistema planetário através desta visualização e é possível refinar ainda mais esta simulação a fim de contabilizar estes detalhes e obter figuras ainda mais fieis da realidade.

Como conclusão, temos que no presente projeto foi possível investigar os comportamentos principais do Sistema Solar e as leis que regem seus movimentos. Pudemos investigar como a análise de precisão dos métodos numéricos é importante e também como a escolha de condições iniciais pode influenciar os resultados da simulação (mais ou menos fiéis à realidades) e também, como no problema de 3 corpos, isso pode estar relacionado ao caos. Por fim, estudar os planetas dessa forma começa a explicar diversos fenômenos que já conhecemos e amplia a compreensão de nosso Sistema Solar.