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:

```
x(-1) = x(0) - vx*dt
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 por um método do tipo 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:

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

Esse método é conhecido por sua boa conservação da energia mecânica ao longo do tempo, característica particularmente útil em simulações orbitais.

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:

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

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.

Resumo

Em síntese, o código compara numericamente os métodos de Euler e Euler-Cromer na simulação de um pêndulo simples, permitindo observar diferenças na conservação da energia ao longo do tempo. Cada método gera dois arquivos de saída: um contendo os parâmetros dinâmicos (ω e θ) e outro contendo as energias calculadas em cada instante.

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

TAREFA - B

CÓDIGO

Figura 4 – Função principal do código.

```
1         program main
2         call euler_crommer()
3         end program main
```

Fonte: Compilado pelo Autor.

Figura 5 – Implementação do método de Euler-Cromer.

```
1
2           subroutine euler_crommer()
3           parameter (imax=1e4)
4           implicit real*8(a-h,o-z)
5           dimension th(0:imax),w(0:imax)
6
7   C       Constantes
8           rl = 9.81d0
9           g = 9.81d0
10          m = 1d0
11          y = 0.050d0 !Gamma
12          F0 = 0.50d0 !Força externa
13          ome = 0.75d0 ! Frequência da força externa
14          pi = acos(-1d0)
15          dt = 0.04d0
16
17  C       Valores iniciais
18          w(0) = 0
19          th(0) = pi/3d0
20
21  C       Cálculo
22
23          do i = 0,imax-1
24                  F_ex = -y*w(i) + F0*sin(ome*i*dt)
25                  w(i+1) = w(i)-(g/rl)*sin(th(i))*dt + F_ex*dt
26                  th(i+1) = th(i) + w(i+1)*dt
27          end do
28
29  C       Salva a posição
30          open(unit=1,file='saida-1-12694394.txt')
31          do i =0,imax
32          ! Trem das posições
33          if (th(i+1) .GT. 2*pi) then
34                  th(i+1) = th(i+1) - 2*pi
35          else if (th(i+1) .LT. 0 ) then
36                  th(i+1) = th(i+1) + 2*pi
37          end if
38
39                  write(1,7) i*dt, w(i), th(i)
40          end do
41  7       format(F12.6,F12.6,F12.6)
42          close(1)
43          end subroutine euler_crommer
```

Fonte: Compilado pelo Autor.

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo simular numericamente o movimento de um pêndulo simples sujeito a atrito viscoso e a uma força externa periódica, utilizando o método de integração de *Euler-Cromer*. O cálculo é implementado na subrotina `euler_crommer`, chamada a partir do programa principal.

A diretiva

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

define todas as variáveis cujos nomes iniciam com as letras de **a** a **h** e **o** a **z** como números reais de dupla precisão, garantindo maior precisão nos cálculos numéricos.

Subrotina `euler_crommer`

A subrotina `euler_crommer` executa o cálculo principal da simulação. Primeiramente, são definidos o número máximo de iterações (`imax = 1e4`) e os vetores `w(0:imax)` e `th(0:imax)`, que armazenam respectivamente a velocidade angular (ω) e o ângulo (θ) do pêndulo em cada passo de tempo.

As constantes físicas e parâmetros do sistema são definidos como:

```
rl = 9.81d0       ! Comprimento do pêndulo
g  = 9.81d0       ! Aceleração da gravidade
m  = 1d0          ! Massa
y  = 0.050d0      ! Coeficiente de amortecimento (Gamma)
F0 = 0.50d0       ! Amplitude da força externa
ome = 0.75d0      ! Frequência angular da força externa
dt = 0.04d0       ! Passo temporal
pi = acos(-1d0)   ! Valor de pi
```

As condições iniciais são fixadas como $\omega(0) = 0$ e $\theta(0) = \pi/3$.

O método de Euler-Cromer é então aplicado dentro de um laço de iterações que atualiza a velocidade angular e o ângulo a cada passo de tempo. A força externa e o termo de amortecimento são incluídos na equação de movimento, resultando no seguinte esquema numérico:

```
do i = 0, imax-1
    F_ex = -y*w(i) + F0*sin(ome*i*dt)
    w(i+1) = w(i) - (g/rl)*sin(th(i))*dt + F_ex*dt
    th(i+1) = th(i) + w(i+1)*dt
end do
```

O termo `F_ex` representa a soma da força de amortecimento ($-y\omega$) e da força externa periódica ($F_0 \sin(\omega_{\text{ext}} t)$). O uso do método de Euler-Cromer garante maior estabilidade e melhor conservação de energia do sistema em comparação com o método de Euler tradicional, especialmente em sistemas oscilatórios.

Saída dos resultados

Após o cálculo, os resultados da simulação são gravados no arquivo `saida-1-12694394.txt`. Cada linha do arquivo contém o tempo ($t = i \cdot \Delta t$), a velocidade angular ω e o ângulo θ, conforme o formato:

```
7 format(F12.6,F12.6,F12.6)
```

Antes da escrita, o programa corrige o ângulo θ para mantê-lo dentro do intervalo $[0, 2\pi]$, utilizando a verificação:

```
if (th(i+1) .GT. 2*pi) then
    th(i+1) = th(i+1) - 2*pi
else if (th(i+1) .LT. 0) then
    th(i+1) = th(i+1) + 2*pi
end if
```

Por fim, o arquivo é fechado com o comando `close(1)`.

Resumo

Em resumo, o código realiza a integração numérica das equações de movimento de um pêndulo forçado e amortecido pelo método de Euler-Cromer. O programa permite estudar o comportamento dinâmico do sistema sob diferentes condições de força e amortecimento, sendo útil para análises de regimes oscilatórios, ressonância e comportamento caótico em pêndulos não-lineares.

RESULTADOS E DISCUSSÃO

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.