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.

Figura 6 – Enunciado da Tarefa 1.

Fonte: Compilado pelo Autor.

TAREFA - C

ENUNCIADO

Figura 7 – Enunciado da Tarefa C.

TAREFA C: Para verificarmos a existência ou não do regime caótico vamos considerar, como antes, $\gamma = 0.05, \Omega = \frac{2}{3}, \Delta t = 0.04, F_0 = \frac{1}{2}$ e $F_0 = 1.2$. Consideramos agora o movimento de dois pêndulos soltos com velocidade nula em ângulos iniciais que difiram de $\theta_0^{(2)} - \theta_0^{(1)} = \Delta\theta_0 = 0.001$ radianos. Faça um gráfico de $\Delta\theta(t) = \theta^{(2)}(t) - \theta^{(1)}(t)$ para os casos em que $F_0 = 0.5$ e $F_0 = 1.2$. Repare que no primeiro caso as trajetórias se aproximam exponencialmente (não caótico), enquanto que no segundo caso as mesmas se afastam exponencialmente (caótico), ou seja

$$\Delta\theta(t) \approx \exp \lambda t \tag{0.10}$$

onde

$$\lambda < 0 \rightarrow \text{não caótico}, \quad \lambda > 0 \rightarrow \text{caótico}. \tag{0.11}$$

Faça os gráficos $\Delta(\theta) \times t$ com escala semi-logarítimica e estime o parâmetro λ, chamado de **expoente de Liapunov**.

3

Na realidade o movimento caótico não é tão imprevisível quanto nos parece à primeira vista. Ele possue certa estrutura que podemos visualizar traçando o gráfico $\omega(\theta)$.

Fonte: Compilado pelo Autor.

CÓDIGO

Figura 8 – Função principal do código.

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

Fonte: Compilado pelo Autor.

Figura 9 – Implementação do método de Euler-Cromer.

```
1
2           subroutine euler_crommer()
3               parameter(imax=(3*1e3))
4               dimension w1(0:imax), th1(0:imax)
5               dimension w2(0:imax), th2(0:imax)
6
7   C       Parametros
8           g = 9.81e0
9           rl = 9.81e0
10          rm = 1e0
11          pi = acos(-1e0)
12          dt = 0.04e0
13          ! Parametros da força
14          F0_1 = 0.5e0
15          F0_2 = 0.5e0
16          ome_1 = 0.75e0
17          ome_2 = 0.75e0
18          y_1 = 0.05e0
19          y_2 = 0.05e0
20
21  C       Condições iniciais
22          ! Velocidades angulares
23          w1(0) = 0
24          w2(0) = 0
25          ! Angulo inicial
26          th1(0) = 1e0
27          th2(0) = 1e0 + 0.001e0
28
29  C       Cálculo
30          do i = 0,imax-1
31              F1 = - y_1*w1(i) + F0_1*sin(ome_1*i*dt)
32              F2 = - y_2*w2(i) + F0_2*sin(ome_2*i*dt)
33
34              w1(i+1) = w1(i) - (g/rl)*sin(th1(i))*dt + F1*dt
35              w2(i+1) = w2(i) - (g/rl)*sin(th2(i))*dt + F2*dt
36
37              th1(i+1) = th1(i) + w1(i+1)*dt
38              th2(i+1) = th2(i) + w2(i+1)*dt
39          end do
40
41  C       Salva os resultados
42          open(unit=1,file='saida-2-12694394.txt')
43          write(1,3)
44          do i = 0,imax
45          if (th1(i+1) .GT. 2e0*pi) then
46              th1(i+1) = th1(i+1) - 2e0*pi
47          else if (th1(i+1) .LT. 0e0) then
48              th1(i+1) = th1(i+1) + 2e0*pi
49          end if
50
51          if (th2(i+1) .GT. 2e0*pi) then
52              th2(i+1) = th2(i+1) - 2e0*pi
53          else if (th2(i+1) .LT. 0e0) then
54              th2(i+1) = th2(i+1) + 2e0*pi
55          end if
56
57              r1 = th1(i)
58              r2 = th2(i)
```

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo comparar a evolução temporal de dois pêndulos simples sujeitos a atrito viscoso e a uma força externa periódica, utilizando o método numérico de *Euler-Cromer*. A simulação busca observar como pequenas variações nas condições iniciais podem gerar diferenças significativas no comportamento do sistema, caracterizando um regime caótico.

O cálculo principal é realizado na subrotina `euler_crommer`, chamada diretamente pelo programa principal.

Parâmetros e variáveis

Na subrotina `euler_crommer`, é definido o número máximo de iterações `imax` = $3{\times}10^3$ e são criados quatro vetores: `w1`, `th1`, `w2` e `th2`, que representam respectivamente as velocidades angulares (ω) e ângulos (θ) dos dois pêndulos.

As constantes físicas e os parâmetros da força externa são definidos como:

```
g   = 9.81e0        ! Aceleração da gravidade
rl  = 9.81e0        ! Comprimento do pêndulo
rm  = 1e0           ! Massa
pi  = acos(-1e0)    ! Valor de pi
dt  = 0.04e0        ! Passo temporal

! Parâmetros da força externa
F0_1  = 0.5e0       ! Amplitude da força (pêndulo 1)
F0_2  = 0.5e0       ! Amplitude da força (pêndulo 2)
ome_1 = 0.75e0      ! Frequência da força (pêndulo 1)
ome_2 = 0.75e0      ! Frequência da força (pêndulo 2)
y_1 = 0.05e0        ! Amortecimento (pêndulo 1)
y_2 = 0.05e0        ! Amortecimento (pêndulo 2)
```

Os dois pêndulos são idênticos em todos os parâmetros físicos, diferindo apenas nas condições iniciais dos ângulos. Ambos começam com velocidade angular nula, mas o segundo pêndulo possui um pequeno desvio inicial de $0{,}001$ radiano:

```
w1(0) = 0
w2(0) = 0
th1(0) = 1.0
th2(0) = 1.0 + 0.001
```

Integração pelo método de Euler-Cromer

O laço principal da simulação executa `imax` iterações, aplicando o método de Euler-Cromer para calcular a evolução temporal dos ângulos e velocidades angulares. A cada passo, são calculadas as forças efetivas sobre os dois pêndulos, compostas pelo amortecimento e pela força externa periódica:

```
F1 = - y_1*w1(i) + F0_1*sin(ome_1*i*dt)
F2 = - y_2*w2(i) + F0_2*sin(ome_2*i*dt)
```

Em seguida, atualizam-se as velocidades e ângulos segundo:

```
w1(i+1) = w1(i) - (g/rl)*sin(th1(i))*dt + F1*dt
w2(i+1) = w2(i) - (g/rl)*sin(th2(i))*dt + F2*dt

th1(i+1) = th1(i) + w1(i+1)*dt
th2(i+1) = th2(i) + w2(i+1)*dt
```

O método de Euler-Cromer utiliza a nova velocidade angular (ω_{i+1}) no cálculo da posição angular (θ_{i+1}), o que resulta em maior estabilidade numérica para sistemas oscilatórios.

Armazenamento dos resultados

Após o cálculo, os resultados são gravados no arquivo `saida-2-12694394.txt`. Antes da gravação, é feita a correção do ângulo θ para garantir que permaneça dentro do intervalo $[0, 2\pi]$, evitando o acúmulo de voltas excedentes. Cada linha do arquivo contém:

- o tempo ($t = i \cdot \Delta t$),
- a velocidade angular e o ângulo do primeiro pêndulo,
- a velocidade angular e o ângulo do segundo pêndulo,
- e a diferença angular $\Delta\theta = \theta_2 - \theta_1$.

O formato de escrita utilizado é:

```
7 format(5(F12.6,','),F12.6)
```

e o arquivo é encerrado com o comando `close(1)`.

Resumo

Em síntese, o código realiza a simulação de dois pêndulos forçados e amortecidos com condições iniciais ligeiramente diferentes, permitindo analisar a divergência de trajetórias ao longo do tempo. Essa abordagem é típica em estudos de sistemas dinâmicos caóticos, onde pequenas perturbações nas condições iniciais levam a comportamentos macroscopicamente distintos, mesmo sob as mesmas leis de movimento.

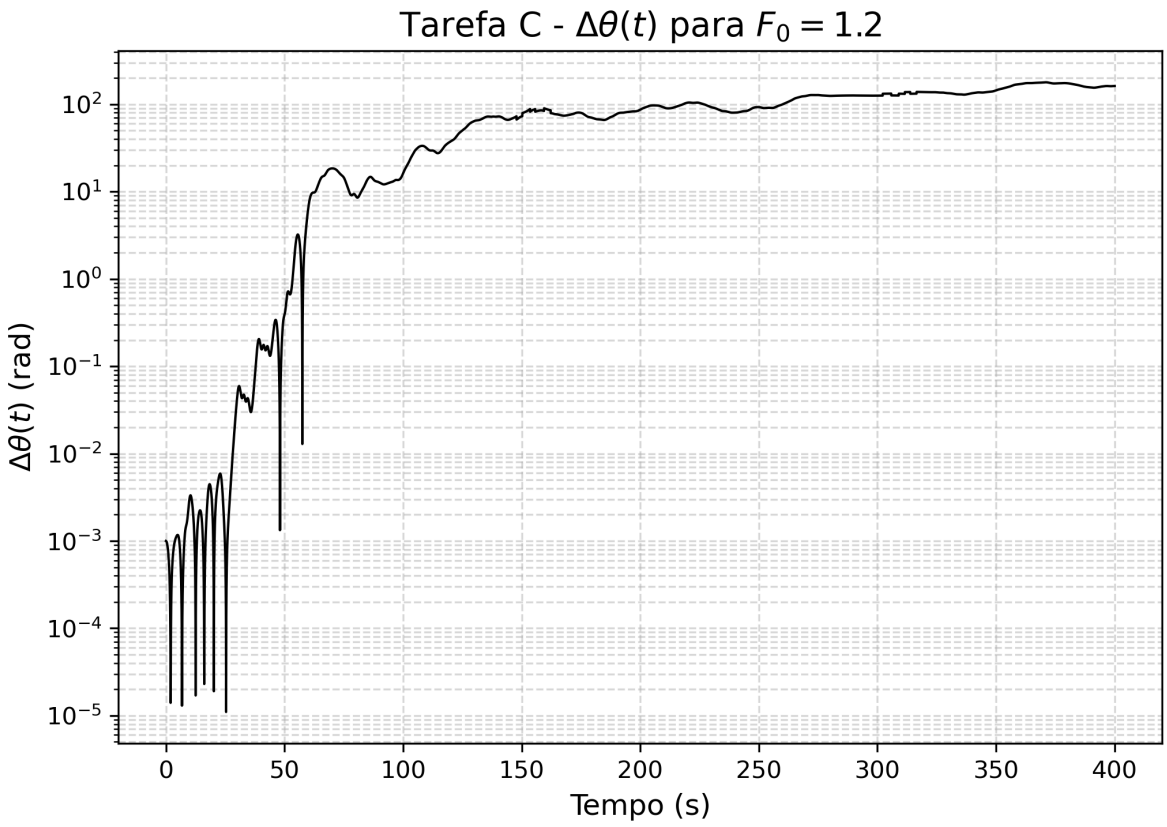
RESULTADOS

Estimativa do coeficiente de Liapunov: 9.2039e-02

Figura 10 – Resultado obtidos.

Fonte: Compilado pelo Autor.

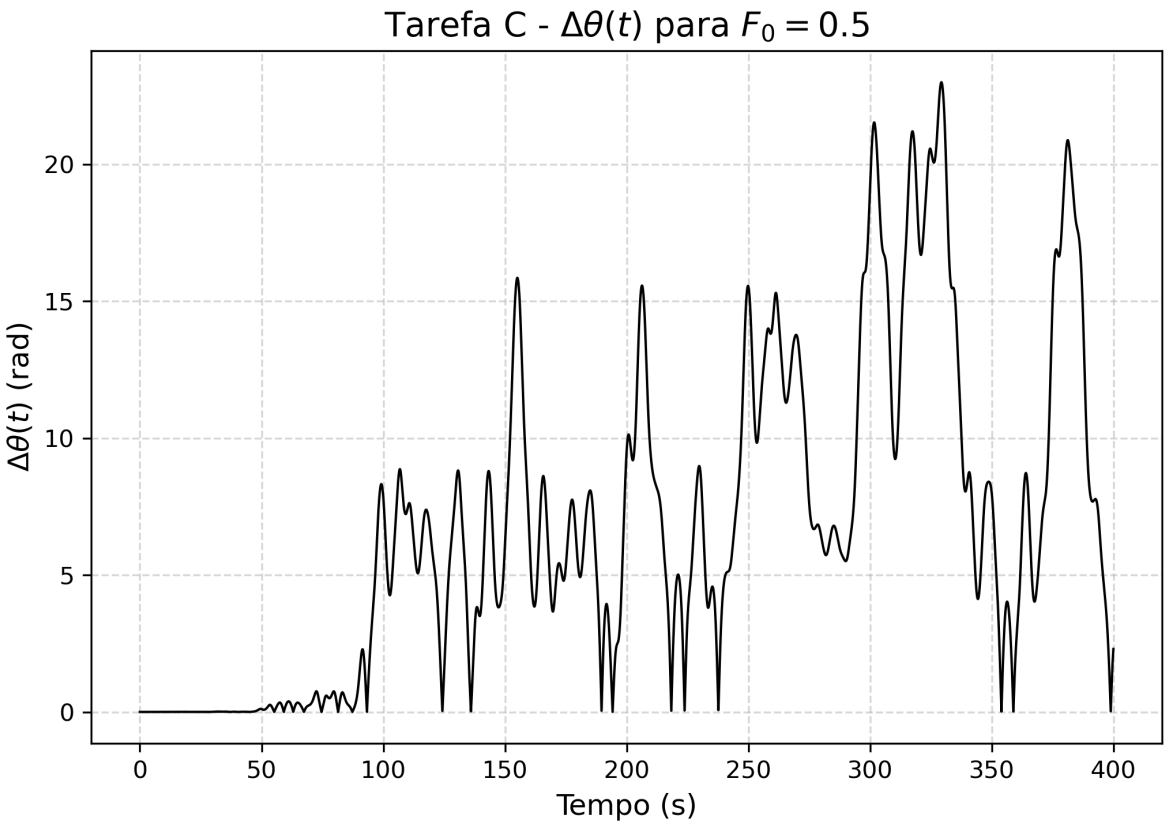
Figura 11 – Resultado obtidos.



Fonte: Compilado pelo Autor.

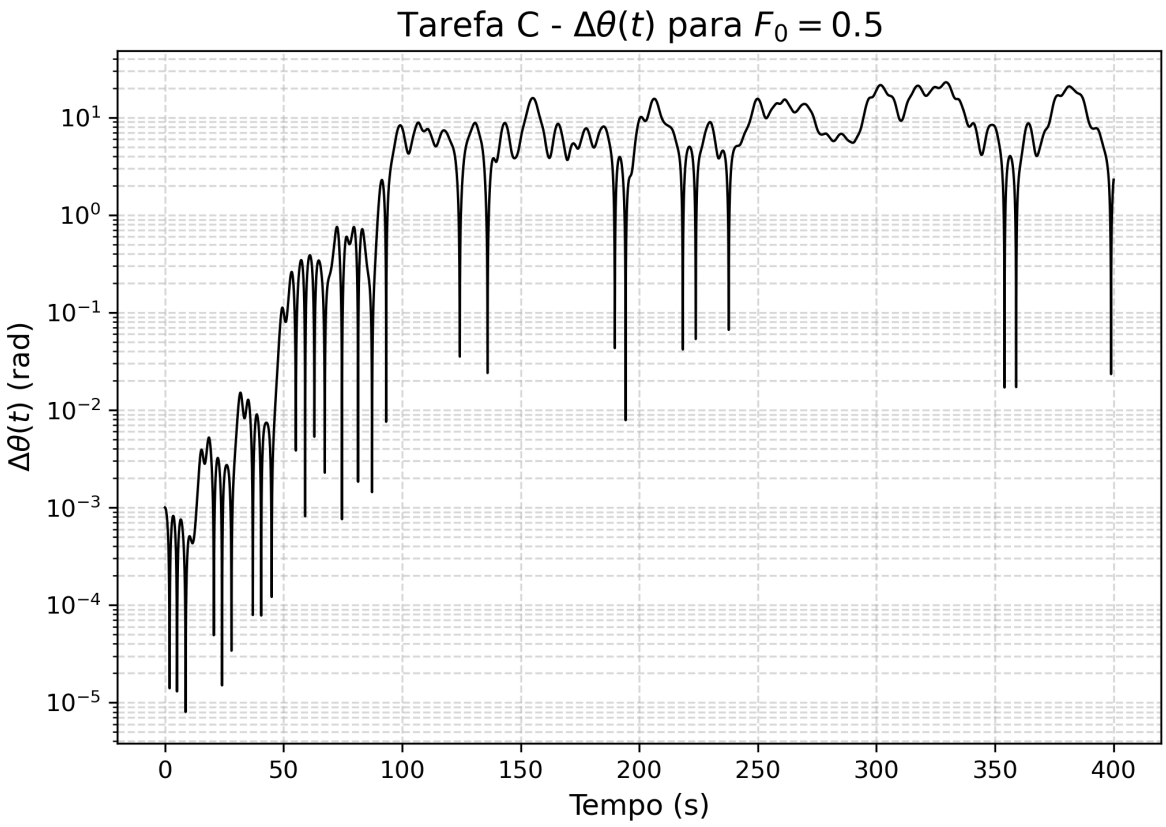
Estimativa do coeficiente de Liapunov: 1.0632e-01

Figura 12 – Resultado obtidos.



Fonte: Compilado pelo Autor.

Figura 13 – Resultado obtidos.



Fonte: Compilado pelo Autor.

TAREFA - D

ENUNCIADO

Figura 14 – Enunciado da Tarefa D.

TAREFA D: Faça os gráficos $\omega(\theta)$ para os casos $F_0 = 0.5$ e $F_0 = 1.2$, para algumas condições iniciais próximas. Compare os resultados obtidos.

Conforme você deve ter observado você não obteve no caso caótico algo tão desordenado. Existe regiões do diagrama que nunca foram visitadas.

Uma maneira mais efetiva de se visualizar a "estrutura" existente no movimento caótico é a realização de uma **secção de Poincaré**. Isto é. só graficamos $\omega(\theta)$ quando $\Omega t = n\pi$ (n inteiro).

Fonte: Compilado pelo Autor.

CÓDIGO

Figura 15 – Função principal do código.

```
1         program main
2             implicit real*8 (a-h,o-z)
3             call poincare()
4         end program main
```

Fonte: Compilado pelo Autor.

Figura 16 – Implementação do método de Euler-Cromer.

```
1
2           subroutine poincare()
3               parameter(imax=1e6)
4               implicit real*8 (a-h,o-z)
5               dimension w(0:imax), th(0:imax)
6               dimension ival(0:imax)
7   C           Constantes
8               pi = acos(-1d0)
9               g = 9.81d0
10              rl = 9.81d0
11              dt = 0.04d0
12              c1 = 0.05d0 ! Fator de amorteciemento (gamma)
13              c2 = 1.2d0 ! F_0
14              c3 = 0.666d0 ! Frequencia angular da força extena
15
16  C           Valores iniciais
17              w(0) = 0
18              th(0) = pi/60d0
19              ival(0) = 0
20  C           Realiza a simulação
21              open(unit=7,file='saida-2-12694394.txt')
22              write(7,9)
23
24              do i = 0,imax-1
25                  rr = c3*i*dt
26                  F = -c1*w(i) + c2*sin(rr)
27                  w(i+1) = w(i) -(g/rl)*sin(th(i))*dt +F*dt
28                  th(i+1) = th(i) + w(i+1)*dt
29
30                  if (abs(mod(rr, pi)) .LE. 1d-3) then
31                      ival(i+1) = 1
32                      write(7,8) w(i+1),th(i+1)
33                  else
34                      ival(i+1) = 0
35                  end if
36
37              end do
38
39              close(7)
40  8           format(F16.8,',',F16.8)
41  9           format('omega,theta')
```

Fonte: Compilado pelo Autor.

Figura 17 – Implementação do método de Euler-Cromer.

```
1
2  C            Salva os dados
3               open(unit=1,file='saida-1-12694394.txt')
4                   write(1,3)
5               do i = 0,imax-1
6               ! Aqui eu tenho que eu tenho que fazer o trem do angulo
7               if (th(i+1).GT. 2*pi) then
8                   th(i+1) = th(i+1) -2*pi
9               else if (th(i+1) .LT. 0d0) then
10                  th(i+1) = th(i+1) + 2*pi
11              end if
12
13
14                  write(1,2) ival(i),dt*i, w(i),th(i)
15              end do
16 2            format(I2,3(',',F16.8))
17 3            format('poincare,temp,omega,theta')
18              close(1)
19
20
21
22         end subroutine poincare
```

Fonte: Compilado pelo Autor.

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como função principal chamar a subrotina `poincare`, responsável por simular o movimento de um pêndulo forçado e amortecido por meio do método numérico de *Euler-Cromer*. O objetivo da simulação é gerar o mapa de Poincaré do sistema, permitindo a análise do comportamento dinâmico e a identificação de regimes periódicos ou caóticos.

Toda a parte computacional é implementada dentro da subrotina `poincare`, que realiza tanto a integração temporal das equações de movimento quanto o registro dos dados em arquivos de saída.

Parâmetros e variáveis

Na subrotina `poincare`, é definido o número máximo de iterações `imax` = 1×10^6, e são criados três vetores principais: `w` (velocidades angulares), `th` (ângulos) e `ival` (indicador lógico que identifica os pontos de Poincaré).

As constantes e parâmetros físicos do sistema são definidos como:

```
1 pi = acos(-1d0)    ! Valor de pi
```

```
g  = 9.81d0        ! Aceleração da gravidade
rl = 9.81d0        ! Comprimento do pêndulo
dt = 0.04d0        ! Passo temporal
c1 = 0.05d0        ! Coeficiente de amortecimento (gamma)
c2 = 1.2d0         ! Amplitude da força externa (F0)
c3 = 0.666d0       ! Frequência angular da força externa
```

As condições iniciais são:

```
w(0)   = 0
th(0)  = pi/60d0
ival(0) = 0
```

Esses valores definem um pêndulo inicialmente quase vertical e sem velocidade angular, sujeito a uma força externa periódica e a um amortecimento viscoso.

Integração pelo método de Euler-Cromer

A simulação é realizada dentro de um laço que percorre todas as iterações de $i = 0$ até `imax - 1`. Em cada passo, calcula-se a força total atuante sobre o pêndulo, composta pelo termo de amortecimento e pela força externa periódica:

```
rr = c3*i*dt
F  = -c1*w(i) + c2*sin(rr)
```

Em seguida, as equações diferenciais do sistema são integradas utilizando o método de Euler-Cromer:

```
w(i+1)  = w(i) - (g/rl)*sin(th(i))*dt + F*dt
th(i+1) = th(i) + w(i+1)*dt
```

Esse método utiliza o valor atualizado da velocidade angular para calcular a posição, conferindo maior estabilidade numérica, especialmente para sistemas oscilatórios.

Construção do mapa de Poincaré

Durante a simulação, o programa verifica se o argumento da força $c_3 i\,dt$ é aproximadamente um múltiplo de π. Essa condição identifica os instantes em que o sistema completa um ciclo da força externa. Quando isso ocorre, o valor da velocidade angular ω e do ângulo θ são registrados no arquivo `saida-2-12694394.txt`, formando o conjunto de pontos do mapa de Poincaré:

```
if (abs(mod(rr, pi)) .LE. 1d-3) then
    ival(i+1) = 1
    write(7,8) w(i+1), th(i+1)
```

```
4  else
5      ival(i+1) = 0
6  end if
```

Esses pontos permitem analisar visualmente a evolução do sistema no espaço de fases sob o regime forçado, revelando comportamentos periódicos, quase-periódicos ou caóticos.

Armazenamento dos resultados completos

Além do mapa de Poincaré, o código também grava os dados completos da simulação no arquivo `saida-1-12694394.txt`. Antes de salvar, o ângulo é ajustado para permanecer dentro do intervalo $[0, 2\pi]$, garantindo a consistência do gráfico de fase. O arquivo contém, para cada iteração:

- o identificador de ponto de Poincaré (`ival(i)`),
- o tempo ($t = i \cdot \Delta t$),
- a velocidade angular (ω),
- e o ângulo (θ).

Os dados são escritos no formato:

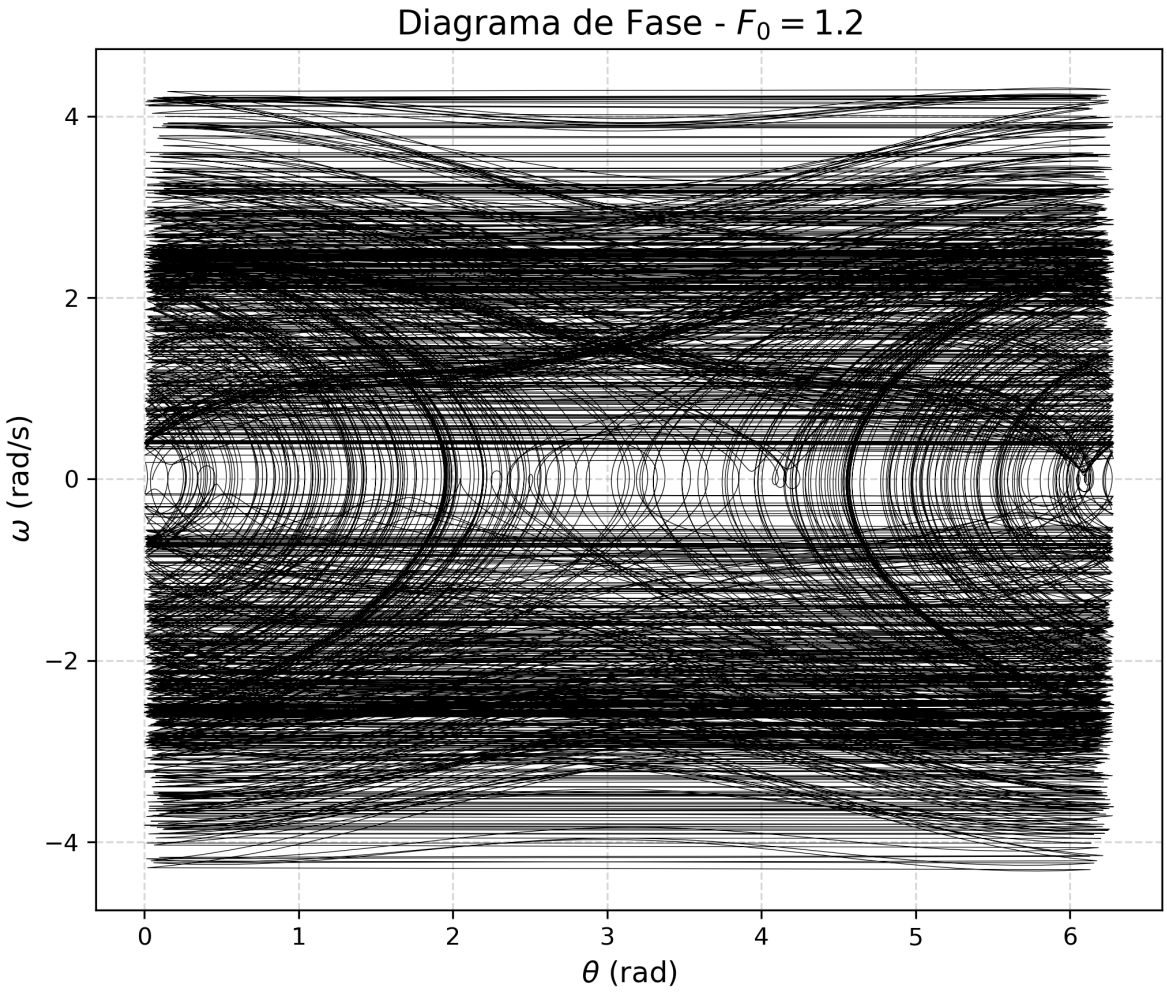
```
1  2 format(I2,3(',',F16.8))
```

e o arquivo é finalizado com o comando `close(1)`.

Resumo

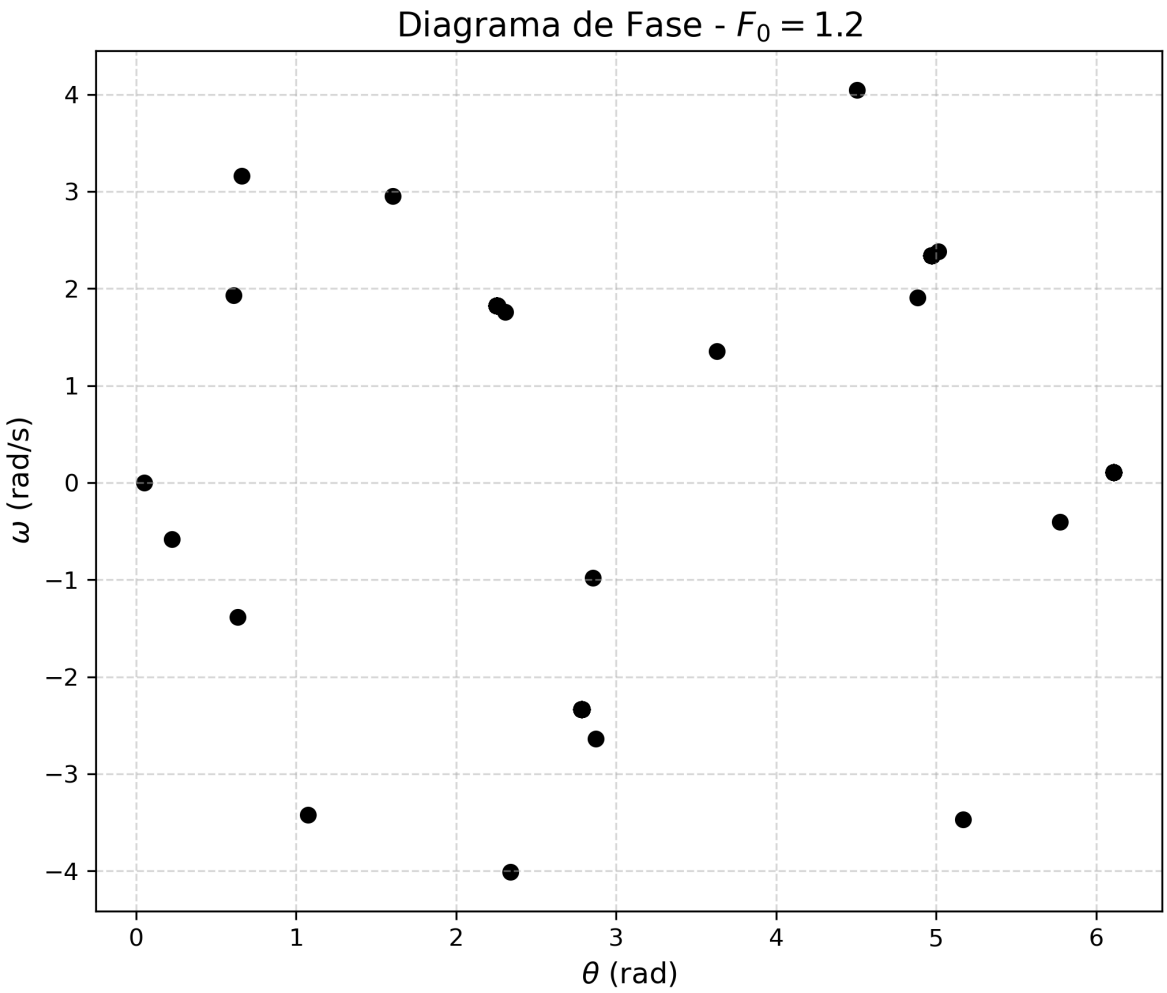
Em resumo, o código implementa a simulação de um pêndulo amortecido e forçado, aplicando o método de Euler-Cromer para integração numérica e gerando o mapa de Poincaré do sistema. A análise dos pontos registrados permite identificar regimes de periodicidade, transições para o caos e comportamento determinístico complexo, característico de sistemas não lineares forçados.

Figura 18 – Resultado obtidos para F_0 = 1.2



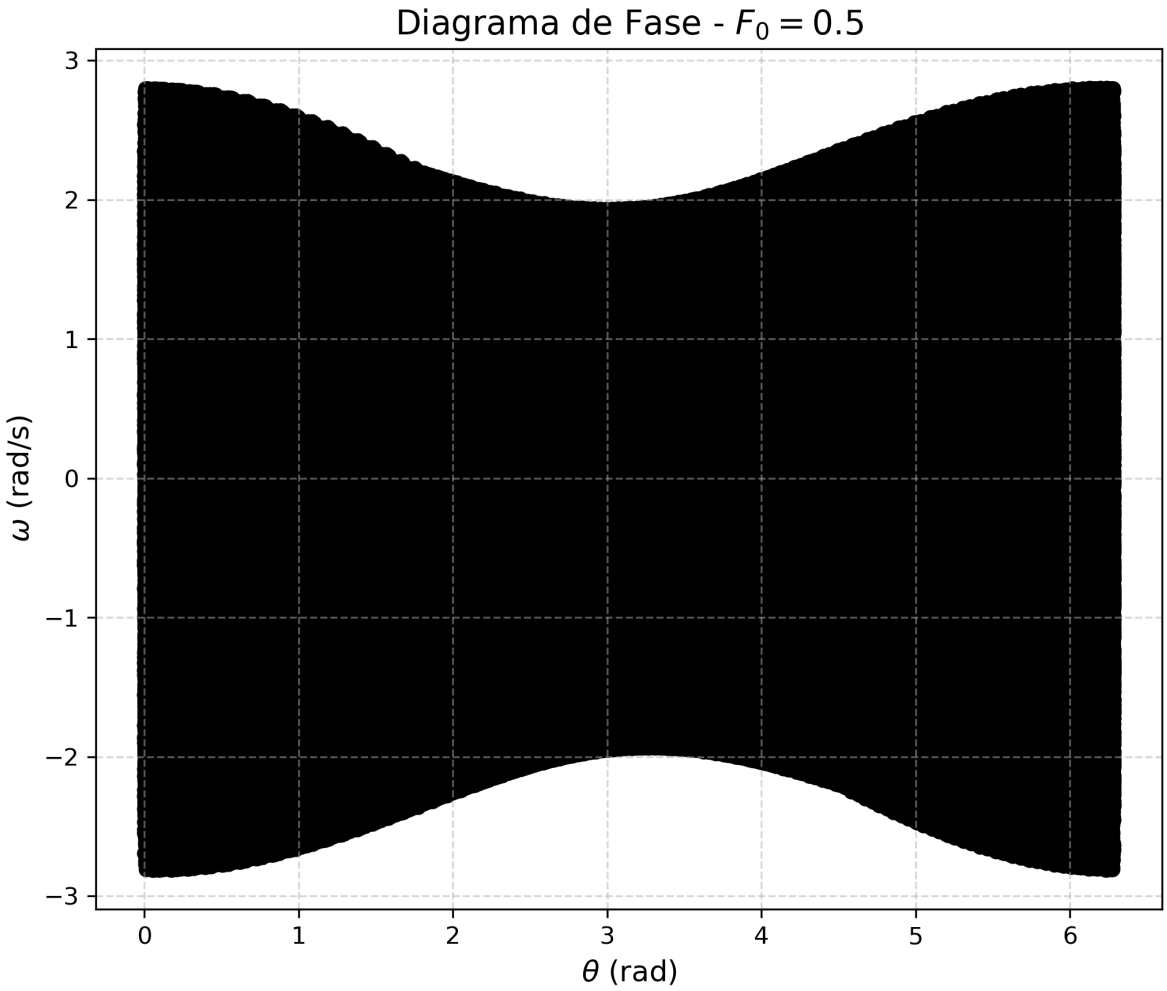
Fonte: Compilado pelo Autor.

Figura 19 – Resultado obtidos para F_0 = 1.2 com filtro de Poincaré



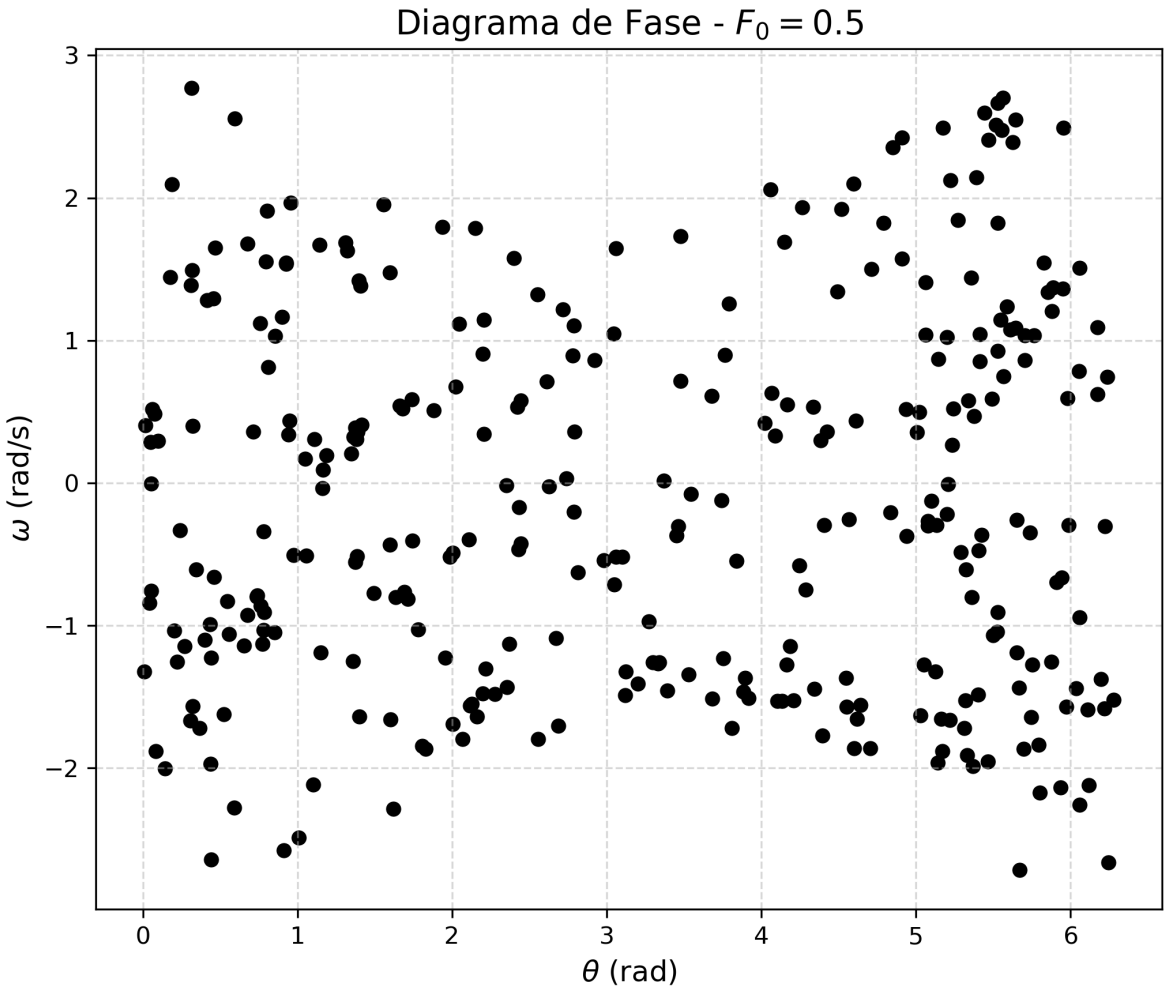
Fonte: Compilado pelo Autor.

Figura 20 – Resultado obtidos para F_0 = 0.5



Fonte: Compilado pelo Autor.

Figura 21 – Resultado obtidos para F_0 = 0.5 com filtro de Poincaré



Fonte: Compilado pelo Autor.

TAREFA - E

ENUNCIADO

Figura 22 – Enunciado da Tarefa E.

TAREFA E: Faça o gráfico de $\omega(\theta)$ na secção de Poincaré $\Omega t = n\pi$, que no presente caso deve ser traduzido numericamente por $|t - n\pi/\Omega| < \Delta t/2$. O gráfico deve ser feito pra o caso $F_0 = 0.5$ e $F_0 = 1.2$. A Figura que você obtem é o "R.G." do movimento caótico em questão. Varie ligeiramente as condições iniciais e verifique que a figura fica inalterada, o que mostra a "universalidade" do seu caos. Na realidade a figura que você obteve não é contínua e define um fractal. O estudo de fractais e caos estará então intimamente ligado. A figura que dá o "R.G." do caos e que você obteve é chamada de "atrator estranho". Repare que no caso determinístico o atrator estranho é um ponto. ‘

Fonte: Compilado pelo Autor.

CÓDIGO

Figura 23 – Função principal do código.

```
1         program main
2             implicit real*8 (a-h,o-z)
3             call poincare_secao()
4         end program main
```

Fonte: Compilado pelo Autor.

Figura 24 – Implementação do método de Euler-Cromer.

```
1
2           subroutine poincare_secao()
3               parameter(imax=1e6)
4               implicit real*8 (a-h,o-z)
5               dimension w(0:imax), th(0:imax)
6
7               pi = acos(-1d0)
8               g  = 9.81d0
9               rl = 9.81d0
10              dt = 0.04d0
11              c1 = 0.05d0
12              c2 = 1.2d0
13              c3 = 0.666d0
14
15              w(0)  = 0.0d0
16              th(0) = pi/60d0
17
18              open(unit=10,file='saida-1-12694394.txt')
19              write(10,9)
20
21              do i = 0, imax-1
22                  t  = i*dt
23                  rr = c3*t
24                  F = -c1*w(i) + c2*sin(rr)
25                  w(i+1)  = w(i) - (g/rl)*sin(th(i))*dt + F*dt
26                  th(i+1) = th(i) + w(i+1)*dt
27
28                  if (th(i+1) .GT. 2*pi) then
29                      th(i+1) = th(i+1) - 2*pi
30                  else if (th(i+1) .LT. 0d0) then
31                      th(i+1) = th(i+1) + 2*pi
32                  end if
33
34                  if (abs(mod(c3*t, pi)) .LT. (c3*dt/2d0)) then
35                      write(10,8) w(i+1), th(i+1)
36                  end if
37              end do
38
39              close(10)
40  8           format(F16.8,',',F16.8)
41  9           format('omega,theta')
42              return
43          end
```

Fonte: Compilado pelo Autor.

DESCRIÇÃO DO CÓDIGO

O programa `main` tem como objetivo chamar a subrotina `poincare_secao`, responsável por calcular a seção de Poincaré de um pêndulo amortecido e forçado,

utilizando o método numérico de *Euler-Cromer*. A simulação permite analisar o comportamento dinâmico do sistema em regimes periódicos e caóticos, registrando apenas os pontos específicos que compõem a seção de Poincaré.

Toda a integração numérica e o armazenamento dos resultados são realizados dentro da subrotina `poincare_secao`.

Parâmetros e variáveis

Na subrotina, define-se o número máximo de iterações `imax = 1×10^6` e são criados dois vetores principais: `w` e `th`, que armazenam, respectivamente, a velocidade angular (ω) e o ângulo (θ) do pêndulo ao longo do tempo.

As constantes físicas e parâmetros da força externa são definidos como:

```
pi = acos(-1d0)    ! Valor de pi
g  = 9.81d0        ! Aceleração da gravidade
rl = 9.81d0        ! Comprimento do pêndulo
dt = 0.04d0        ! Passo temporal
c1 = 0.05d0        ! Coeficiente de amortecimento (gamma)
c2 = 1.2d0         ! Amplitude da força externa (F_0)
c3 = 0.666d0       ! Frequência angular da força externa
```

As condições iniciais do sistema são:

```
w(0)  = 0.0d0
th(0) = pi/60d0
```

Esses valores correspondem a um pêndulo inicialmente em repouso, com um pequeno deslocamento angular inicial, sujeito à ação de uma força externa periódica e a um termo de amortecimento viscoso.

Integração pelo método de Euler-Cromer

A integração numérica é realizada dentro de um laço que percorre de `i = 0` até `imax - 1`. Em cada iteração, calcula-se a força resultante sobre o pêndulo, composta pelos termos de amortecimento e força externa:

```
t  = i*dt
rr = c3*t
F  = -c1*w(i) + c2*sin(rr)
```

As equações diferenciais são integradas pelo método de *Euler-Cromer*, que atualiza primeiro a velocidade angular e, em seguida, o ângulo:

```
w(i+1)  = w(i) - (g/rl)*sin(th(i))*dt + F*dt
th(i+1) = th(i) + w(i+1)*dt
```

Esse método é mais estável para sistemas oscilatórios, pois utiliza o valor atualizado da velocidade para calcular a posição.

Correção angular e critério da seção de Poincaré

Após a atualização, o ângulo é ajustado para permanecer dentro do intervalo $[0, 2\pi]$, evitando que valores excedam esse limite devido à integração acumulada:

```
if (th(i+1) .GT. 2*pi) then
    th(i+1) = th(i+1) - 2*pi
else if (th(i+1) .LT. 0d0) then
    th(i+1) = th(i+1) + 2*pi
end if
```

A cada passo, o código verifica se o instante atual t corresponde aproximadamente a um múltiplo de π/ω_{fora}, ou seja, se o sistema se encontra na mesma fase da força externa. Quando essa condição é satisfeita, o par (ω, θ) é registrado no arquivo de saída, compondo a *seção de Poincaré*:

```
if (abs(mod(c3*t, pi)) .LT. (c3*dt/2d0)) then
    write(10,8) w(i+1), th(i+1)
end if
```

Armazenamento dos resultados

Os pontos da seção de Poincaré são salvos no arquivo `saida-1-12694394.txt`, em formato CSV, com as colunas `omega` e `theta`. O formato de escrita utilizado é:

```
8 format(F16.8,',',F16.8)
```

O arquivo é finalizado com o comando `close(10)` após o término das iterações.

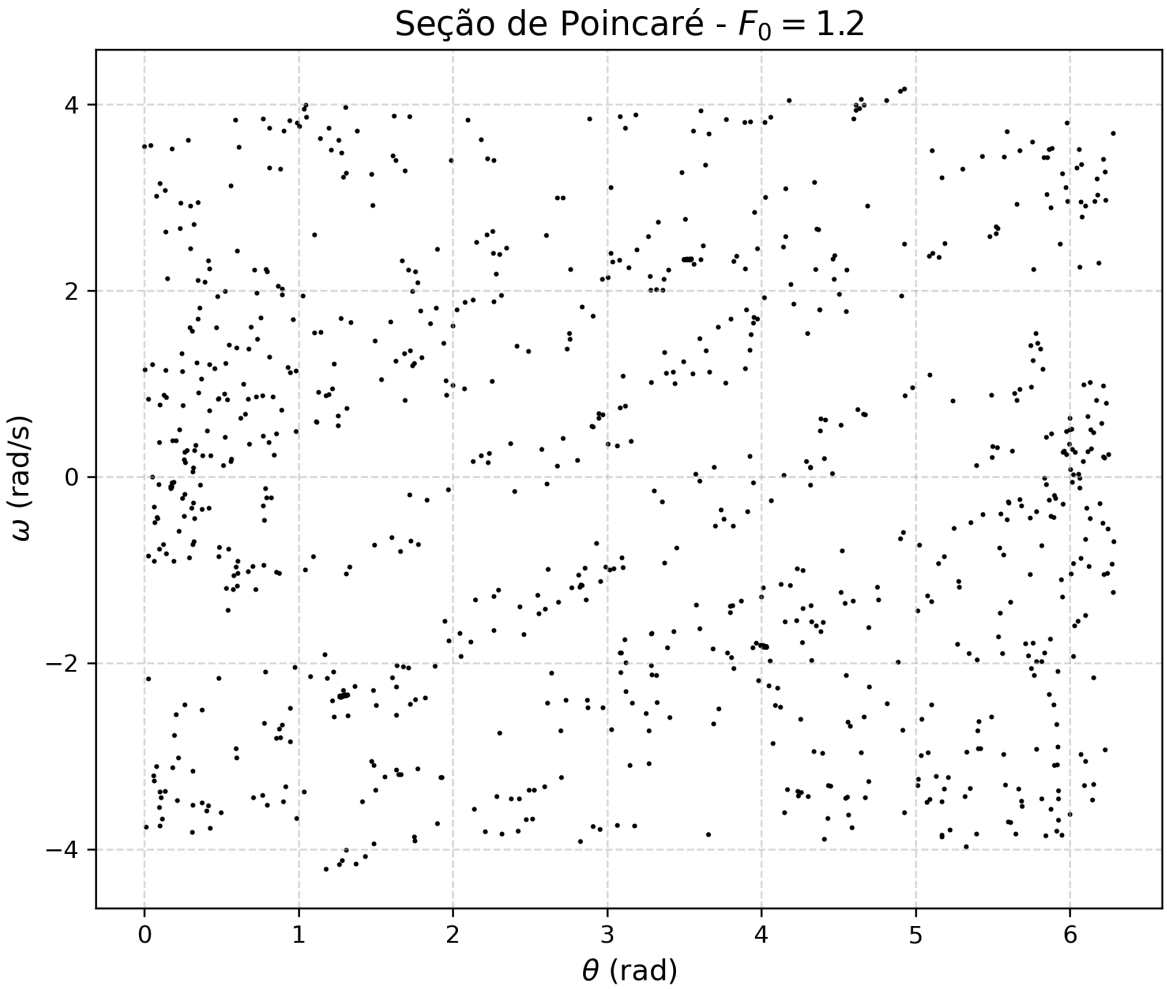
Resumo

Em síntese, o código implementa a simulação de um pêndulo amortecido e forçado, aplicando o método de Euler-Cromer e registrando apenas os pontos pertencentes à seção de Poincaré. Esses pontos representam o estado do sistema em instantes sincronizados com a força externa, permitindo a visualização de órbitas periódicas, quase-periódicas e caóticas no espaço de fases. Esse tipo de análise é

fundamental no estudo de sistemas não lineares e fenômenos caóticos em mecânica clássica.

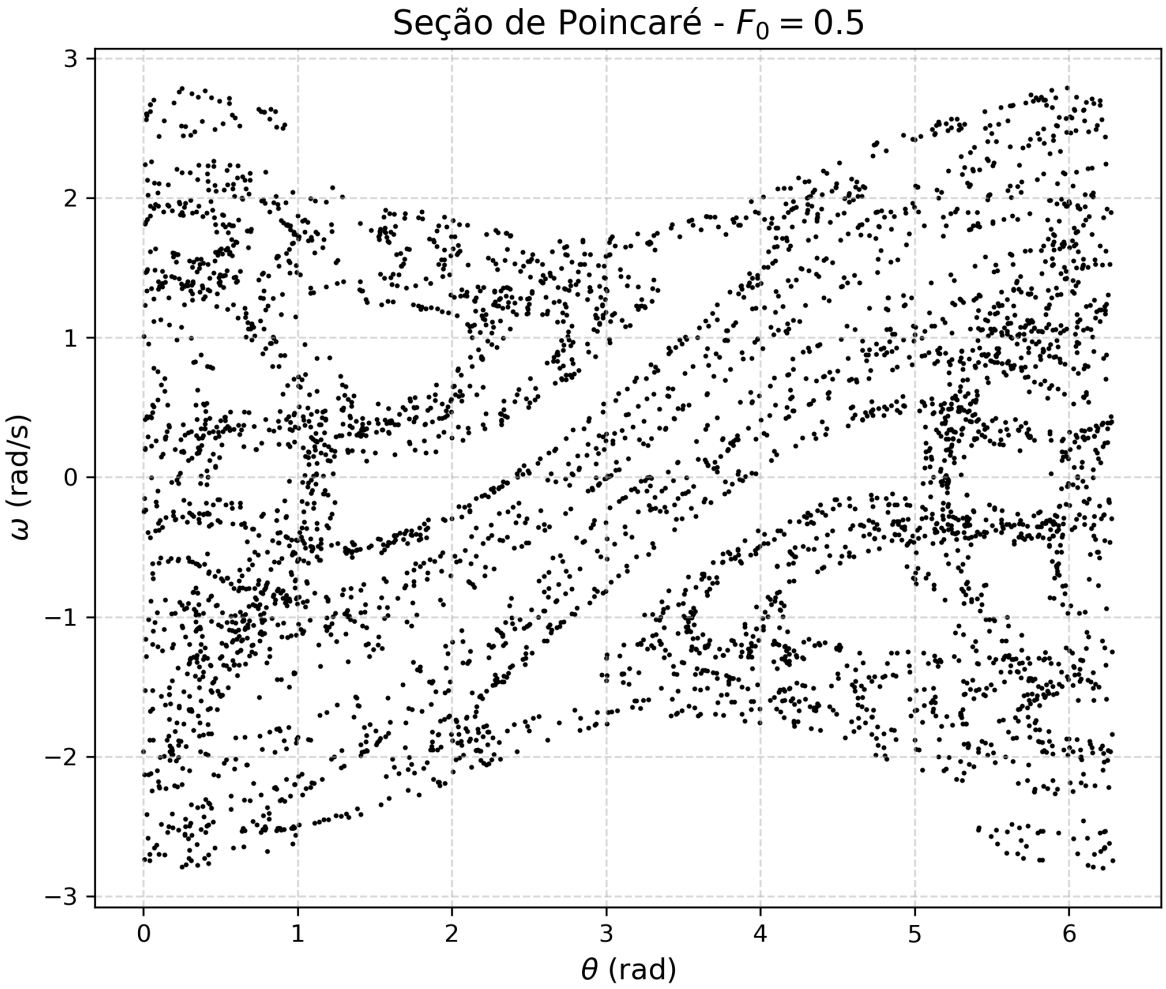
RESULTADOS

Figura 25 – Resultado obtidos para F_0 = 1.2



Fonte: Compilado pelo Autor.

Figura 26 – Resultado obtidos para F_0 = 0.5



Fonte: Compilado pelo Autor.