Simulação da Orbita de Mercúrio

Ana S. Nunes

Simulação e Modelação

Departamento de Física - Universidade de Aveiro

Índice

1. Objetivos

O presente trabalho teve como objetivo principal a aplicação de métodos numéricos na modelação de um fenómeno físico e verificação da eficiência associada. Escolheu-se fazer a simulação e modelação em 2D da órbita de mercúrio desprezando os efeitos relativísticos, o efeito do movimento do sol e dos restantes planetas.

2. Introdução

2.1 Métodos numéricos

Os métodos numéricos utilizados foram os seguintes: Euler, Euler-Cromer, Runge-Kutta de 2ª ordem e Runge-Kutta de 4ª ordem. As equações do movimento expressas, pelas leis de Newton, correspondem a um conjunto de equações diferenciais que relacionam as forças a que um corpo está submetido. Considere-se um corpo de massa constante. Pela segunda lei de Newton,

$$F = m\frac{d^2x}{dt^2} = m\ddot{x}$$

A segunda lei de Newton pode ser escrita como uma equação diferencial,

$$\dot{x} = v \quad \text{e} \quad \dot{v} = \frac{F}{m}$$

Pela definição de derivada,

$$v(t + \Delta t) = v(t) + \frac{f(t)}{m}\Delta t$$
$$v(t_n + \Delta t) = v_{n+1}$$
$$t_n = t_0 + n\Delta$$

O método de Euler trata-se de uma expansão da equação em série de Taylor em que o único termo utilizado é o de ordem um, e é uma aproximação às equações diferenciais de primeira ordem. A escolha de Δt é um compromisso entre o tempo de cálculo gasto. Para fenómenos periódicos como a as órbitas planetárias, uma alternativa ao método de Euler, é o método de Euler-Cromer. Difere do algoritmo anterior por no cálculo de x_{n+1} utilizar a velocidade calculada no princípio do intervalo (v_n) ou no fim deste (v_{n+1}).

O método Runge Kutta de segunda e quarta ordem resultam numa compensação a cada iteração e não a cada período como nos métodos referidos anteriormente.

A implementação do método segue os seguintes passos,

$$r_1 = a(t_n, v_n)h$$

$$r_2 = a\left(t_n + \frac{h}{2}, v_n + \frac{r_1}{2}\right)h$$

$$v_{n+1} = v_n + k_2$$

Para o Runge-Kutta de 4ªordem, ordem superior, segue-se adicionalmente os passos,

$$r_3 = a\left(t_n + \frac{h}{2}, v_n + \frac{r_2}{2}\right)h$$

$$r_4 = a(t_n + h, v_n + r_3)h$$

$$v_{n+1} = v_n + \frac{1}{6}(r_1 + 2r_2 + 2r_3 + r_4)$$

2.2 Órbitas planetárias

As leis de Kepler descrever as órbitas planetárias; um planeta em orbita em torno do Sol, descreve uma elipse, **1ª Lei de Kepler;** e, os planetas movem-se com velocidades relativas diferentes dependendo da distância ao sol, **2ª Lei de Kepler.** O **periélio** corresponde ao ponto mais próximo do sol, e o **afélio** o ponto mais afastado do sol. Desta forma, o planeta atinge a velocidade máxima no periélio, e atinge a velocidade mínima no afélio.

A força a que o planeta está sujeito é dada por,

$$F = -\frac{\sigma Mm}{r^2}\vec{\boldsymbol{r}} = -\frac{\sigma Mm}{r^3}\boldsymbol{r} \quad \textbf{e} \quad \vec{r} = \frac{r}{\|r\|}, \text{ vetor unitário}$$

Segundos as componentes x e y: $F_x = -\frac{\sigma Mm}{r^3}\boldsymbol{x}$ e $F_y = -\frac{\sigma Mm}{r^3}\boldsymbol{y}$, em que m e M correspondem às massas, sol e planeta respetivamente. A força exercida é atrativa pelo que é negativa. Trata-se de um sistema conservativo, logo, $\Delta E_m = 0 \; e \; E_c = \frac{1}{2}mv^2$

2.3 GUI- Graphical User Interface

GUI é uma ferramenta de programação composta por vários objetos, e a cada um estão associados um conjunto de informações. Esta ferramenta permite a visualização e monotorização de variáveis em simultâneo.

3. Implementação do código

Utilizou-se a seguintes condições iniciais **h**=0.001; **t** = 0:h:0.5; e, $GM_s = 4\pi^{2}$(2).

(2) $G = 6.6741 \times 10^{-11}\, m^3 kg^{-1} s^{-2}$(Constante gravitacional)

Para a posição inicial: y(1) = 0; x(1) = 0.47; e, para a velocidade inicial: vx(1) = 0; vy(1) = 8.2

A aceleração segundo as componentes x e y é dada por,

$$\frac{dv_x}{dt} = \frac{GM}{r^3}\boldsymbol{x} \qquad \frac{dv_y}{dt} = \frac{GM}{r^3}\boldsymbol{y}$$

De forma a simplificar os cálculos foram feitos em unidades astronómicas (UA).

$$\frac{M_m v^2}{r} = G\frac{M_m M_s}{r^3} \Leftrightarrow GM_s = v^2 r \Leftrightarrow GM_s = 4\pi^2$$

```
for i = 1:numel(t)-1
r = sqrt(x(i)^2+y(i)^2); %Vetor posição: Raio
vx(i+1) = vx(i)-(Gms*x(i)/r^3)*h;
x(i+1) = x(i) + vx(i)*h;
vy(i+1) = vy(i)-(Gms*y(i)/r^3)*h;
y(i+1) = y(i)+vy(i)*h;
end
```

Excerto da implementação do algoritmo de Euler

E, o método Euler-Cromer difere deste na determinação do vetor posição, em que a velocidade é calculada no final, v(i+1),

x(i+1) = x(i) + **vx(i+1)*h**

y(i+1) = y(i) + **vy(i+1)*h**

A implementação do método Runge-Kutta de ordem 2 (RK2) consiste na decomposição dos passos em componentes x e y, isto é, componente x e y da velocidade r1_x=vx(i) e r1_y=vy(i) procede-se de forma análogo para r2v e r2y. A implementação do algoritmo segue os passos enunciados na introdução.

```
for i=1:numel(t)-1
    r1_x=vx(i);
    r1_y=vy(i);
    r1_vx=-Gms/(norm([x(i) y(i)])^3)*x(i);
    r1_vy=-Gms/(norm([x(i) y(i)])^3)*y(i);
    r2_vx=-Gms/(norm([x(i)+r1_x*h/2
y(i)+r1_y*h/2])^3).*(x(i)+r1_x*h/2);
    r2_vy=-Gms/(norm([x(i)+r1_x*h/2
y(i)+r1_y*h/2])^3).*(y(i)+r1_y*h/2);
    r2_x=vx(i)+r1_vx*h/2;
    r2_y=vy(i)+r1_vy*h/2;
end
```

Excerto da implementação do algoritmo Runge Kutta de 2ª ordem

Neste método escolheu-se fazer o cálculo do ***r*,** pela função norm do matlab, que devolve a norma euclidiana do vetor posição.

```
for i=1:numel(t)-1
...
r3_v=-Gms/(norm([x(i) y(i)]+h/2*r2_r)^3).*([x(i) y(i)]+r2_r*h/2);
r3_r=[vx(i) vy(i)]+r2_v*h/2;

r4_v=-Gms/(norm([x(i) y(i)]+h*r3_r)^3).*([x(i) y(i)]+r3_r*h);
r4_r=[vx(i) vy(i)]+r3_v*h;
%Atualização da trajetória
    A= [x(i) y(i)]+h/6*(r1_r+2*r2_r+2*r3_r+r4_r);
    B= [vx(i) vy(i)]+h/6*(r1_v+2*r2_v+2*r3_v+r4_v);
...
end
```

Excerto da implementação do algoritmo Runge Kutta de 4ª ordem

No cálculo da distância mínima e máxima em cada método utilizou-se a função max e min, em que f corresponde à norma do vetor posição, $\sqrt{x^2 + y^2}$. Para todos os métodos,

```
%Calculo do valor do afelio e do perielio:
f = sqrt(x_x.^2+y_y.^2);
Afelio = max(f)
Perielio = min(f)
af_tabelado=0.466697;
per_tabelado=0.307499;
erro_af_ec=(abs(af_tabelado-Afelio)/af_tabelado)*100;
erro_perielio_ec=(abs(per_tabelado-Perielio)/per_tabelado)*100;
```

4. Análise

Pelo cálculo dos erros relativos, obteve-se um erro associado ao método de Euler igual, a 40% para o afélio e 15% para o periélio, enquanto para os restantes métodos obteve-se 0.7% e 2%, afélio e periélio, respetivamente.

5. Discussão

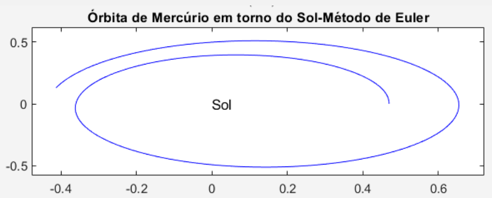

Gráfico 1.1. Este método revela o aumento da instabilidade a cada volta. Aumentando Δt observa-se que a energia do sistema não é constante ao longo do tempo.

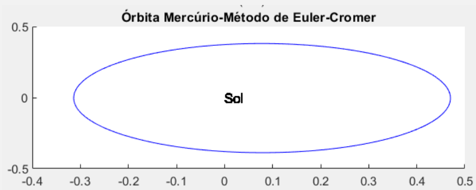

Gráfico 1.2 Para um mesmo h, 0.001, isto é, dt, este método embora não aumente a energia do sistema e seja mais estável, tem um certo erro associado.
O método Euler-Cromer apresenta uma maior estabilidade que o método de Euler.

Pelo cálculo da exatidão dos raios máximos e mínimos em cada método não foi possível determinar qual dos métodos entre os três últimos apresentados seja o mais exato, por tanto, realizou-se gráficos de comparação reativamos à trajetória efetiva. Apenas entre o método de Euler se verificou uma diferença significativa, pelo que o gráfico serve apenas para verificar os resultados anteriormente analisados.

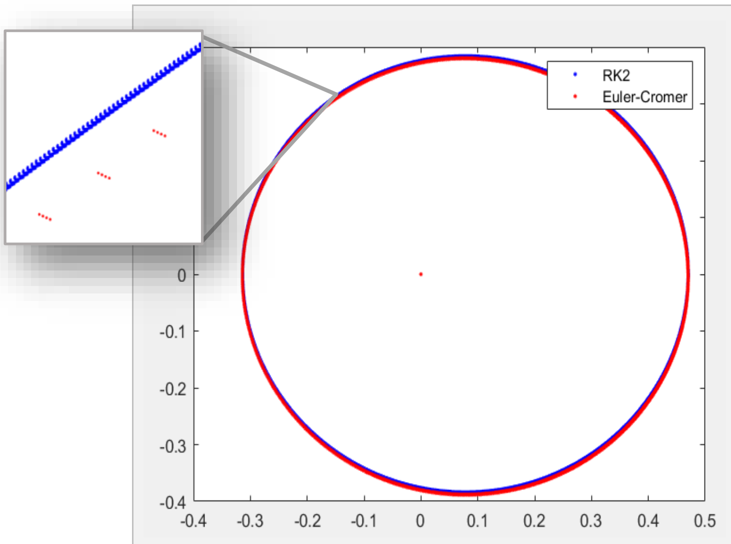

Fig. 1. Runge Kutta 2ª ordem vs Euler Cromer: Verifica-se uma diferença significativa na estabilidade do método na descrição da trajetória.

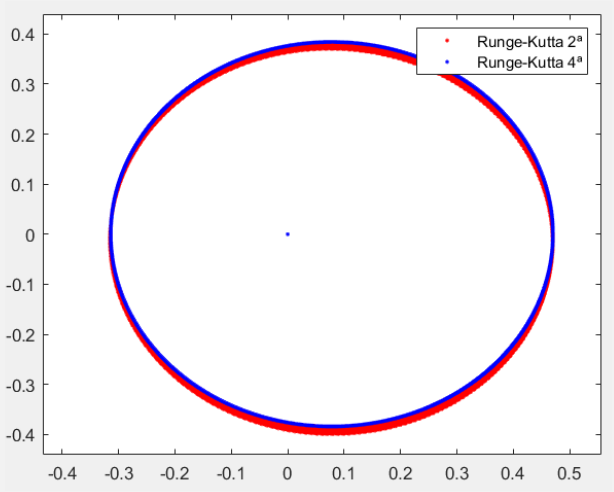

Fig. 2. Runge Kutta 2ª ordem vs Runge Kutta 4ª ordem: Verifica-se uma diferença significativa na descrição da órbita; o método de RK2 diverge significativamente, enquanto RK4 mantem-se praticamente constante em cada volta.

5.7 Guide Graphical User Interface

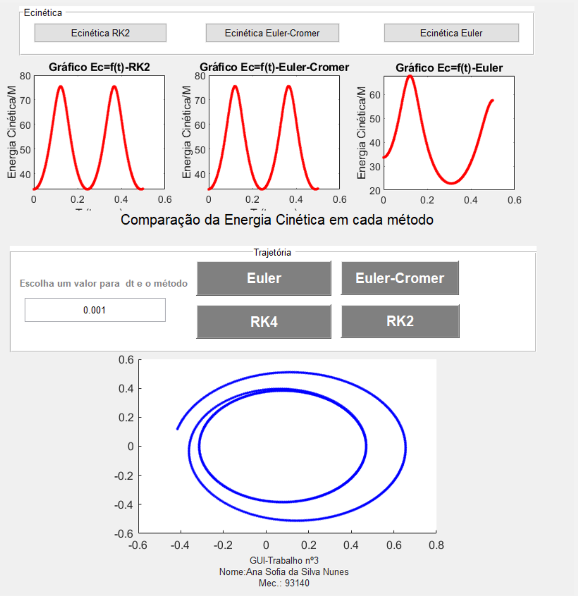

Fig.4 Em auxílio na compreensão da energia envolvida em cada método realizou-se a representação gráfica da energia cinética, $\frac{E_c}{M}$,em função do tempo, $E_c = \frac{1}{2}mv^2$.
Dispôs-se 4 axes, o utilizador pode escolher quais dos gráficos deseja visualizar, e o valor pedido Δt, isto é, h (str2num(get(handles.dt,'string'))-excerto.
Em cada plot, associa-se o método que o utilizador escolhe, por exemplo, plot(handles.axes4,x,y,'.b').

6. Conclusão

Após o estudo dos algoritmos, conclui-se que o método Runge-Kutta de 2 ordem seria o mais adequado para o fenómeno descrito uma vez que há um equilíbrio entre a complexidade e a ordem do método a usar para cada problema dadas as mesmas condições iniciais. Poder-se-á considerar, no entanto, o método Runge-Kutta de 4ª ordem caso se pretenda uma maior exatidão nos resultados e se despreze a complexidade de implementação. Em suma, os métodos RK 2ª ordem e 4ª ordem são mais eficientes.

O estudo da implementação dos algoritmos revelou-se uma mais valia na compreensão dos métodos e sua eficiência. E, o conhecimento das potencialidades dos métodos futurou uma extensa aplicação na área da computação física.

7.Bibliografia

[1] Jorge Dias de Deus, Mário Pimenta e Ana Noronha, *Introdução à Física*, Escolar Editora, 2ª edição, Lisboa, 2014.

[2] *Métodos de Runge-Kutta*, Física Computacional, Departamento de Física, Aveiro, 2015.

[3] *Mercúrio*, 20 de maio, 2019, em https://pt.wikipedia.org/wiki/Merc%C3%BArio_(planeta).

[4] *Applying Kepler's Laws*, *Brilliant.org*, 20 de maio, 2019, em https://brilliant.org/wiki/applying-keplers-laws/laws/.

[5] *Aula 11 Equações diferenciais*, Simulação e Modelação, Ano letivo 2018/2019, em e-leraning.