Trabalho 1 - Oscilador não-linear

Física Computacional

João Inácio, 93039, PL7

06/05/2020

1 Introdução

Este trabalho tem como principal objetivo estudar o comportamento de um oscilador não-linear. Para tal é-nos dada uma equação de movimento para um pendulo caótico e, com recurso a métodos numéricos, temos de resolver a equação diferencial não-linear. Foram usados dois métodos Runge-Kutta, a rotina `ode45()` do MatLab, ou seja, método RK de passo adaptativo, e o método Runge-Kutta de 4ª ordem. Como a rotina `ode45()` é de passo adaptativo podemos esperar resultados mais precisos, comparativamente ao método RK4.
Um objetivo secundário deste trabalho é comparar os métodos de Runge-Kutta de 4ª ordem com a rotina `ode45()` e com o método mais simples de resolução de ODE's, o método de Euler.

2 Métodos

Como já mencionado na introdução, temos de resolver uma equação diferencial que representa o movimento de um pêndulo não-linear. A equação a resolver é

$$m\frac{d^2y}{dt^2} + K(y + \alpha y^3) = \mu \left[\cos\left(\frac{dy}{dt}\right)\right]\frac{dy}{dt} + F_0 cos(\omega_0 t) \tag{1}$$

Como a equação é do segundo grau, para aplicar os métodos numéricos temos dividi-la em duas equações de primeira ordem. Como $\frac{dy}{dt} = v$ e $\frac{dv}{dt} = f(y,t)$, podemos substituir na equação (1). Assim, esta fica

$$\frac{dy}{dt} = v \tag{2}$$

$$\frac{dv}{dt} = \frac{\mu}{m}\left[\cos\left(\frac{dy}{dt}\right)\right]\frac{dy}{dt} - \frac{K}{m}(y + \alpha y^3) + \frac{F_0}{m} cos(\omega_0 t) \tag{3}$$

2.1 Rotina `ode45()`

A rotina `ode45()` é uma função do MatLab, que utiliza métodos de Runge-Kutta para resolver equações diferencias ordinárias com um passo, dt, adaptativo. Este método, como muitos outros para resolver EDO's, apenas precisa das condições de t_{n-1} para calcular as condições de t_n.

Esta função usa o método RK de 4ª ordem ou, no caso de ser necessário uma precisão mais elevada na estimação numérica, utiliza um RK de 5ª ordem.

Para utilizar este método no problema em questão temos de num ficheiro de função colocar as derivadas de primeiro grau, equações (2) e (3). Também temos de especificar o erro máximo que o método pode chegar para ter uma maior precisão. Neste caso foi utilizado um erro da ordem de 10^{-13}.

Assim este método consegue uma aproximação numérica muito exata, já que adapta o passo a cada iteração e utiliza um método RK de ordem superior quando o erro é maior do que o erro mínimo especificado.

2.2 Runge-Kutta 4

O método Runge-Kutta de 4ª ordem é um método numérico usado para resolver EDO's ou sistemas das mesmas, para problemas de valores iniciais.

Sendo a função

$$\frac{dy}{dt} = f(x,t)$$

a equação diferencial a resolver, o método de RK estima a função no ponto y_{i+1} através de uma média ponderada de quatro valores ou estimativas de $f(x,t)$ em pontos diferentes pertencendo ao intervalo $[t_i, t_{i+1}]$. Os diferentes pesos de cada um dos valores da média é ditado por uma tabela de *Butcher*.

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Esta é a tabela usada neste trabalho e, assim, com esta podemos calcular os quatro valores de $f(x,t)$ e a expressão final para o calculo de y_{i+1}. Para a equação (3) as quatro estimativas no intervalo $[t_i, t_{i+1}]$ podem ser calculadas por

$$\begin{aligned} r_{1,y} &= f_v(v_i) \\ r_{2,y} &= f_v\left(v_i + r_{1,v}\frac{dt}{2}\right) \\ r_{3,y} &= f_v\left(v_i + r_{2,v}\frac{dt}{2}\right) \\ r_{4,y} &= f_v(v_i + r_{3,v}dt) \end{aligned}$$

O calculo de $r_{n,v}$(n=1,2,3,...,ordem do RK) é feito de maneira idêntica, apenas temos de ter em conta o incremento de t_i, que os coeficientes são a coluna mais a esquerda da tabela de *Butcher*. Assim podemos encontrar as equações de v_{i+1} e v_{i+1}

$$y_{i+1} = y_i + \frac{dt}{6}(r_{1,y} + 2r_{2,y} + 2r_{3,y} + r_{4,y}) \tag{4}$$

$$v_{i+1} = y_i + \frac{dt}{6}(r_{1,v} + 2r_{2,v} + 2r_{3,v} + r_{4,v}) \tag{5}$$

Nota: dt é o passo do método. Para uma precisão maior podemos diminuir este, ao custo de poder computacional, já que no método RK4 o erro é $\mathcal{O}(dt^4)$, muito próximo de uma aproximação por série de Taylor.

2.3 Método de Euler

Este método é um dos métodos mais conhecidos para resolver EDO's, devido à sua simplicidade. Este método numérico faz uma aproximação da derivada de primeira ordem por a definição de derivada. Ou seja,

$$\lim_{dt \to 0} \frac{y(x, t+dt) - y(x,t)}{dt} = \frac{dy}{dt} = f(x,t) \tag{6}$$

Isto pode ser aproximado rearranjando os termos e obtemos

$$y_{n+1} = y_n + f(x_n, t_n)dt \tag{7}$$

Este método é muito instável e energético, ou seja, para um t_{final} muito grande o sistema começa a ganhar energia. Também é muito impreciso, com erro local $\mathcal{O}(dt)$ e um erro global $\mathcal{O}(dt^2)$. O método de Euler-Cromer é um método que resolve uma derivada de segunda ordem e é idêntico a este. A única diferença é no cálculo da posição, que em vez de usar v_i, usa-se v_{i+1} para melhor conservação de energia.

3 Resultados e Discussão

Nesta secção vão ser apresentados os resultados. Todos os resultados foram obtidos através dos *scripts* em anexo com este relatório.

Em primeiro lugar, usando a rotina `ode45()` do MatLab, calculamos a trajétoria e a velocidade do oscilador ao longo do intervalo de tempo $t \in [0, 150]$, este intervalo é o mesmo para todos os cálculos que se seguem. Nesta primeira parte fizemos variar F_0e ω_0. Começamos com F_0= 0, ou seja, no caso em que não há oscilações forçadas, e seguidamente com F_0= 0.8 para ω_0= 1 e ω_0= 2.

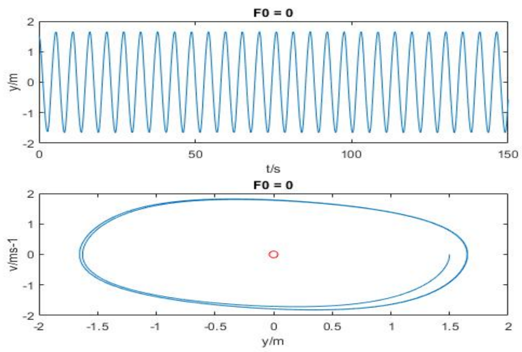

Gráfico 1: F_0= 0. O ponto vermelho no gráfico de baixo indica o centro, $(0, 0)$ do ciclo limite.

Para F_0= 0(gráfico 1), temos, como seria de esperar, oscilações periódicas, já que a equação diferencial que descreve esta oscilação é autónoma. Também podemos verificar isto a partir do diagrama de fases (gráfico de baixo), que para uma EDO autónoma vai ter um atrator, com centro em $(0, 0)$, logo o sistema não perde energia, pois a velocidade não diminui com posição ao longo do tempo.

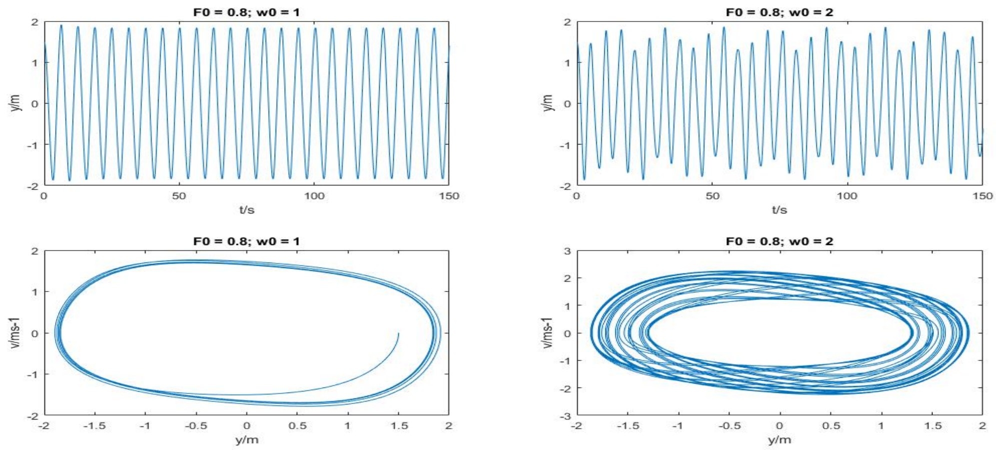

Gráfico 2: F_0= 0.8, com ω_0= 1(esquerda) e ω_0= 2(direita).

No caso de F_0= 0.8(gráfico 2), temos dois resultados diferentes para cada um dos valores de ω_0. Neste caso, a EDO, como já depende de t, é não autónoma, assim é muito mais suscetível a pequenas mudanças nas condições iniciais e nas constantes da expressão.
Quando ω_0= 1 (gráfico 2, esquerda), o oscilador é periódico e tem um atrator com mais voltas em relação ao primeiro caso, com F_0= 0. Assim, podemos dizer que o período da oscilação neste caso é maior do que o período do primeiro caso. Este atrator também tem um centro em $(0, 0)$.
Quando ω_0= 2 (gráfico 2, direita), o oscilador também é periódico como no caso de ω_0= 1, e também tem um ciclo limite, contudo com consideravelmente mais voltas do que no caso de ω_0= 1. Logo, o período deste é maior do que no caso anterior e ainda maior do que o do caso em que F_0= 0.
Assim, em oscilações forçadas, $F_0 \neq 0$, a frequência de oscilação esta ligada ao à frequência das oscilações forçadas. Contudo esta ligação não é proporcional a ω_0, uma vez que o número de voltas em ω_0= 2 teria de ser o dobro das voltas em ω_0= 1, o que não se verifica, pois a partir do gráfico podemos observar que é muito mais do que o dobro.

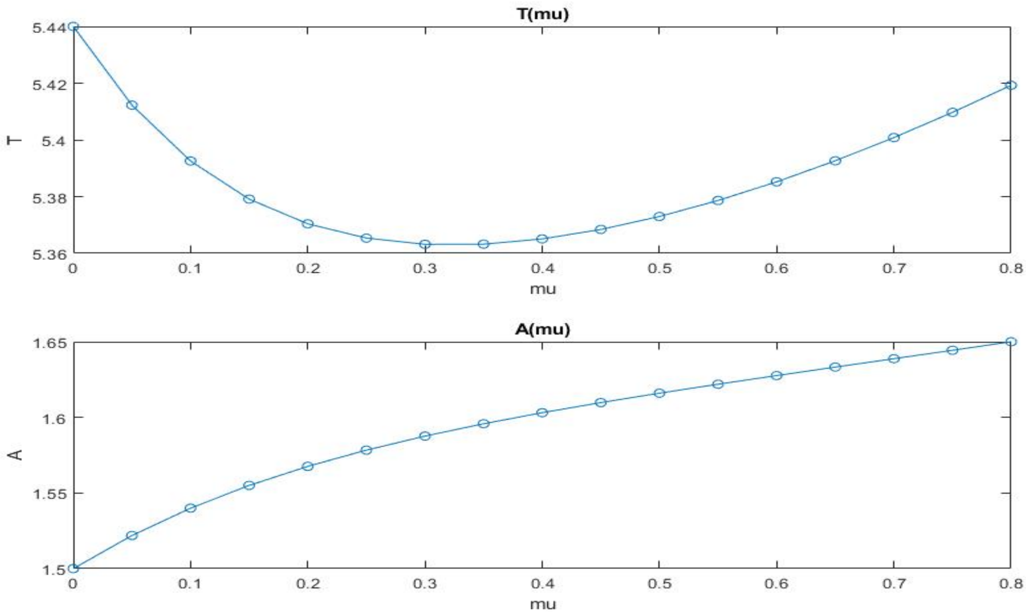

Gráfico 3: Período(cima) e amplitude(baixo) das oscilações em função de μ, com F_0= 0

Em segundo lugar, com F_0= 0, fizemos variar μ, um coeficiente de amortecimento, já que é o coeficiente da primeira derivada da posição em relação ao tempo, de 0 a 0.8 em passos de 0.05 para termos 17 valores para ter uma boa amostra para fazer um gráfico(gráfico 3).

Para encontrar os períodos e amplitudes, tivemos de usar a função `lagr.m`. Esta função toma como argumentos 3 pontos, um ponto antes do máximo da função discreta, o ponto do máximo e um a seguir, para ambos os valores de `t` e `y`. Com estes valores a função encontra os valores máximos e mínimos da função contínua.
Com os todos os valores de `tMax` e `yMax` para cada valor de μ, calculamos os valores de `T` e `A` para cada valor de μ. O resultado disto é o gráfico 3.

Quando fazemos variar μ, o período de oscilação diminui até μ= 0.3, a partir deste valor T aumenta, e para valores do coeficiente de amortecimento maiores que 0.8 tem uma tendência de crescimento. No caso da amplitude, podemos verificar que há um crescimento logarítmico, mesmo para μ> 0.8.

Por fim, tivemos de fazer uma comparação entre os dois métodos descritos na secção 2. O gráfico seguinte contem os gráficos de posição em função do tempo e o gráfico de fases dos dois métodos.

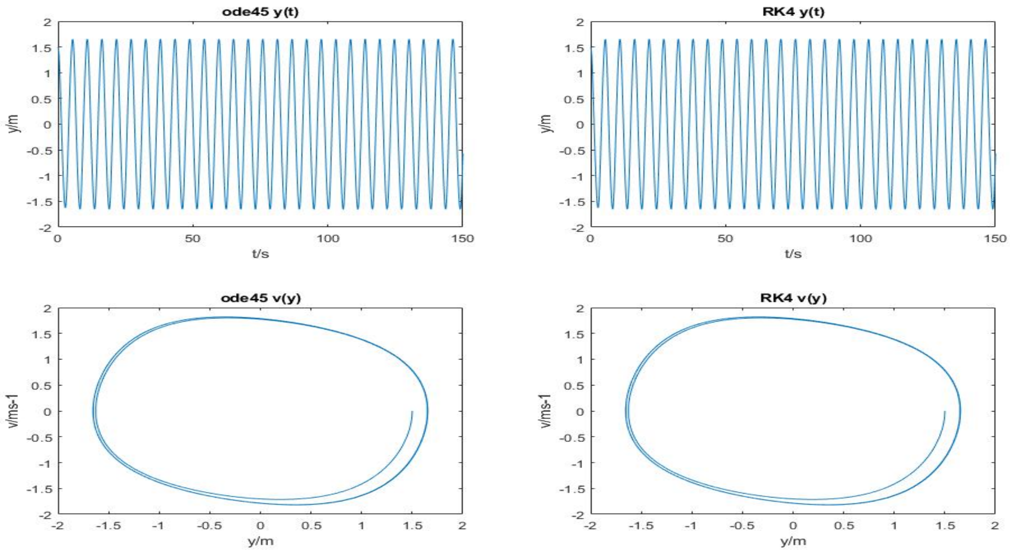

Gráfico 4: Comparação entre os métodos `ode45()` e Runge-Kutta 4ª Ordem

Como podemos ver pela comparação dos gráficos, para a equação de oscilação não forçada, ambos os métodos têm resultados simulares. Em ambos os gráfios $y(t)$ são iguais e o diagrama de fases também seguem a mesma forma em ambos os métodos. Se compararmos o erro de ambos os métodos ($erro = |y_{ode} - y_{rk4}|$) numa escala logarítmica, gráfico 5, podemos observar que no pior dos casos, o erro tem a ordem 1. Esta discrepância dá-se devido ao método `ode45()` ser ter um erro de 10^{-13} e ter um passo, dt, adaptativo, enquanto o método RK4 ter um passo fixo, $dt = 0.001$, e ter um erro máximo $\mathcal{O}(dt^4)$. Para obter um resultado mais conciso entre os dois métodos tínhamos de por um passo extremamente pequeno na execução do RK4.

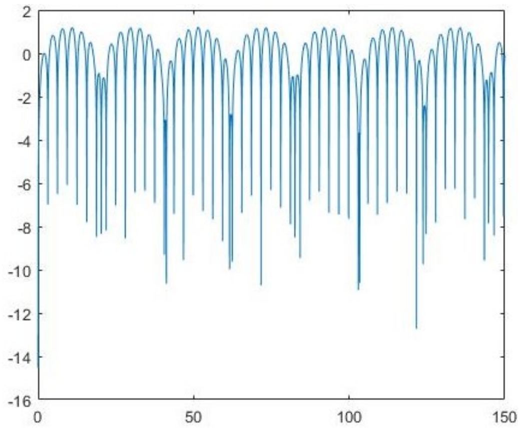

Gráfico 5: Erro entre os dois métodos numa escala logarítmica.

4 Conclusão

Comparando os resultados obtidos na comparação do método RK4 com `ode45()`, podemos concluir que a `ode45()` é uma escolha mais segura, já que é mais simples de implementar no código, pois é uma função própria do MatLab, e é mais precisa, uma vez que usa um passo adaptativo ao erro máximo especificado.

Entre a rotina `ode45()` e o método de Euler-Cromer, apesar de nenhum dos dois ser mais complicado de implementar, o método `ode45()` oferece mais escolhas em relação ao erro, ja que podemos ter um erro de 10^{-13} e não dispender de complexidade computacional, enquanto para o método de Euler-Cromer para ter um erro parecido à `ode45()` temos de ter um dt da ordem de -7, pois o erro global do Euler-Cromer é $\mathcal{O}(dt^2)$.

Em suma, conseguimos verificar que a `ode45()` é mais precisa do que o método RK4, como foi previsto na introdução e também conseguimos estudar o comportamento do oscilador não-linear usando dois métodos para resolver a sua equação de movimento, o método Runge-Kutta de 4ª ordem e a rotina `ode45()` do MatLab.