Universidade de Aveiro
Departamento de Física

**Física Computacional**
**2019/2020**

# Trabalho Prático 3

## Métodos de Runge–Kutta

## Aplicação de métodos de Runge–Kutta de 2ª e 4ª ordens e de passo adaptativo

**Problema 3.1: Oscilador harmónico — Runge–Kutta de 2ª ordem**

a) Sabendo que $x(0) = 1\,\mathrm{m}$, $v_x(0) = 0\,\mathrm{m/s}$, $K = 16\,\mathrm{N/m}$ e $m = 1\,\mathrm{kg}$, encontre a solução numérica do oscilador harmónico simples, usando o método de Runge–Kutta de 2ª ordem com a seguinte tabela de Butcher:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}$$

Pode encontrar o algoritmo (já aplicado a uma ODE de segunda ordem) nos slides 12 a 14 da apresentação 3. Integre até $t_{\text{fin}} = 10\,\mathrm{s}$ e compare com a solução analítica $x(t)$ e com o resultado obtido pelo método de Euler com o mesmo espaçamento $h$.

b) Use um passo $h = 0.10\,\mathrm{s}$ e, tanto para o método de Euler como para o de Runge–Kutta, represente a energia total em função do tempo para os seguintes casos:

   i) $t_{\text{fin}} = 15\,\mathrm{s}$. Deve observar que a energia, que devia ser constante, cresce com o tempo. Além disso, a segunda derivada do gráfico é positiva, ou seja, a taxa de crescimento de energia também aumenta com o tempo, como se pode ver pela curvatura das linhas.

   ii) $t_{\text{fin}} = 500\,\mathrm{s}$.

   iii) $t_{\text{fin}} = 12000\,\mathrm{s}$.

O que podemos concluir dos resultados desta alínea é que nenhum dos métodos é estável para $h = 0.10\,\mathrm{s}$. Na realidade, para o problema do oscilador harmónico, estes métodos não são estáveis para nenhum valor finito de $h$. No caso do método de Runge–Kutta de 2ª ordem isto é mais difícil de observar para valores de $h$ mais pequenos, porque é preciso usar valores bastante maiores de $t_{\text{fin}}$ para obter um comportamento comparável com o caso de $h = 0.10$ s. Para $h = 0.01$ s, por exemplo, a curvatura do gráfico não é muito clara para $t_{\text{fin}} = 1000$ s, mas já é bastante evidente para $t_{\text{fin}} = 10000$ s. Por outro lado, é fácil de observar o aumento da energia com o tempo.

### Problema 3.2: Oscilador harmónico — Runge–Kutta de 4ª ordem

a) Repita a primeira alínea do problema anterior usando deste vez o método de Runge–Kutta de 4ª ordem com a seguinte tabela de Butcher:

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

Pode encontrar o algoritmo (aplicado a uma ODE de primeira ordem) no slide 15 da apresentação 3. Para começar, use um espaçamento $h = 0.10$ s, para o qual o método é estável.

b) Aplicado a este problema, o método de Runge–Kutta de 4ª ordem que estamos a usar é estável para $\omega \cdot h < 2\sqrt{2}$, onde $\omega = \sqrt{K/m}$. Experimente valores de $h$ acima e abaixo do limite imposto por este critério e observe os resultados. Note que não basta que o método seja estável para que um dado valor de $h$ seja uma boa escolha (veja o caso de $h = 0.5$ s, por exemplo).

c) Para vários valores de $h$, sempre abaixo do valor imposto pelo critério de estabilidade, faça um gráfico do logaritmo do módulo do erro numérico em $t = 10$ s em função do logaritmo de $h$, e confirme que se trata de um método de 4ª ordem.

### Informação complementar — Uso de funções anónimas no MATLAB

Para obter $r_{ix}$ e $r_{iv}$, com $i = 1, 2, 3, 4$, tem que escrever quatro vezes a expressão de cada uma das funções calculando-a para diferentes valores dos seus argumentos. Se quiser adaptar o programa para outro tipo de oscilador, vai ter que reescrever essas mesmas linhas. Fazer isto desta maneira dá-nos mais trabalho e mais oportunidades para nos enganarmos.

Nestas situações, torna-se mais fácil usar funções. Uma alternativa seria escrever um ficheiro externo com essa função, mas é mais simples usar o conceito de função anónima disponibilizado pelo MATLAB. Para este exemplo concreto, na parte inicial do programa, depois de atribuídos os valores às constantes, define-se as funções:

```
fx = @(V) V
fv = @(X) -K*X/m;
```

Depois, dentro do ciclo que calcula a solução numérica, chama-se as funções com os argumentos apropriados a este método:

```
r1x = fx(v(k))
r1v = fv(x(k));
r2x = fx(v(k)+r1v*h/2)
r2v = fv(x(k)+r1x*h/2);
(...)
```

**Problema 3.3: Oscilador harmónico simples — Runge–Kutta de passo adaptativo**

Repita a primeira alínea dos problemas anteriores usando a rotina `ode45` do Matlab. Esta rotina usa uma função que tem que ser fornecida pelo utilizador num ficheiro diferente e que deve conter as expressões para as derivadas das variáveis dependentes. Assim, neste caso, ela poderá ter a forma

```
function derivadas = f(t,solucao)
derivadas = zeros(2,1);
% Esta linha é necessária e faz do vetor de saída um vetor coluna.
    …
% O vetor solucao tem os valores de x e v para o tempo t em que a
% função é chamada pela rotina ode45.
derivadas(1) = …
% Escreva acima a expressão da derivada de x em função de solucao(1)
% e de solucao(2).
derivadas(2) = …
% Escreva a acima expressão da derivada de v em função de solucao(1)
% e de solucao(2).
```

Se achar que torna o programa mais claro, pode incluir na função as linhas

```
x = solucao(1);
v = solucao(2);
```

para poder depois escrever `derivadas(1)` e `derivadas(2)` diretamente em função de `x` e `v`.

No programa principal, vai ter que usar os seguintes comandos:

```
options = odeset('RelTol',reltol,'AbsTol',[abstol_1 abstol_2]);
[t,sol] = ode45(@f,[t0 tf],[x0 v0],options);
```

Note que no programa principal, o vetor `sol` tem os valores de `x` e `v`, um em cada coluna, para todos os tempos `t` entre `t0` e `tf`.

`reltol`, `abstol_1` e `abstol_2` são os números que determinam a escolha do passo em cada iteração (leia a documentação do MATLAB). O erro estimado tem que ser menor que o maior dos valores calculados usando os dois critérios: absoluto e relativo. Note, por exemplo, que se só fosse usado o critério da tolerância relativa, para valores de `x` e `v` muito próximos de zero, a ordem de grandeza do último algarismo significativo da estimativa poderia ser excessivamente pequena.

Neste programa, use sempre `reltol = 3E-14` (escolhido porque é praticamente o valor mínimo que o MATLAB aceita) e comece por usar `abstol` iguais a `1E-13`.

05/02/2020