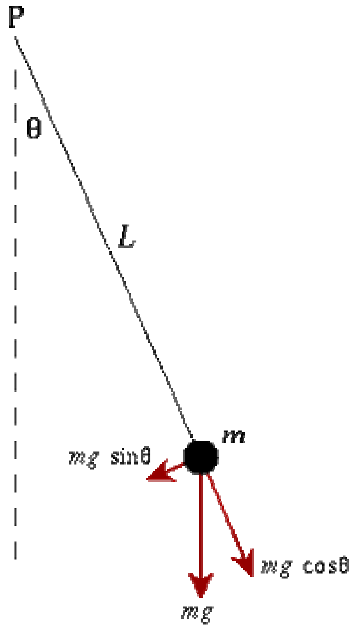

Fig. 1 - Pêndulo simples

Eis a dedução da dinâmica do sistema pêndulo simples:
(Não se considera a massa da haste nem o atrito do ar)

$$\tau + \tau_a = -F_g \cdot L \cdot \sin(\theta)$$

τ : binário resultante do sistema;
τ_a : binário de atrito da junta;
F_g : força gravítica da esfera;
L : comprimento da haste;
θ : orientação angular do pêndulo.

$$J\ddot{\theta} + c\dot{\theta} = -mgL\sin(\theta)$$

J : momento de inércia da esfera;
m : massa da esfera;
c : coeficiente de fricção da junta;
g : aceleração da gravidade (9.81m/s^2).

$$J\ddot{\theta} + c\dot{\theta} + mgL\sin(\theta) = 0$$

Sabendo que o momento de inércia $J = \int r^2 dm$, e que só se resume a uma partícula na extremidade, $J = mL^2$. Por isso…

$$\ddot{\theta} + \frac{c}{mL^2}\dot{\theta} + \frac{g}{L}\sin(\theta) = 0$$

Note que é uma equação diferencial não linear, pelo que métodos especiais são necessários para o seu cálculo. Eventualmente poderia se fazer a aproximação de $\sin(\theta) \approx \theta$ para $\theta \approx 0$, mas perdia-se realismo na representação de θ muito diferente de zero.
Para o cálculo desta equação, usou-se o método de *Runga-Kutta* que resolve equações diferenciais ordinárias de forma numérica. A função *MatLab* que implementa este método é o *ODE45* cujos parâmetros de entrada são os seguintes:

```
[t,θ] = ode45(odefun,tspan,θ0)
```

Em que *odefun*, é uma função que relaciona $\frac{d\theta}{dt} = f(\theta, t)$, *tspan* é a gama temporal no qual se pretende conhecer θ, e θ_0 são as condições iniciais.
Como esta função só lida com equações de 1ª ordem e o que pretendemos resolver é de 2ª, uma técnica, denominada por redução de ordem, terá de ser implementada, em que fragmenta a equação em duas e simultaneamente reduz a sua ordem para um. Eis o procedimento:

$$p_1 = \theta$$

$$p_2 = \dot{\theta}$$

$$\dot{p}_2 = \ddot{\theta} = -\frac{c}{mL^2}\dot{\theta} - \frac{g}{L}\sin(\theta) = -\frac{c}{mL^2}p_2 - \frac{g}{L}\sin(p_1)$$

Por isso…

$$\dot{p} = \frac{d}{dt}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} p_2 \\ -\frac{c}{mL^2}p_2 - \frac{g}{L}\sin(p_1) \end{bmatrix} = \begin{bmatrix} f_1(t,p) \\ f_2(t,p) \end{bmatrix} = f(t,p)$$

As condições iniciais são definidas por…

$$p(0)=\begin{bmatrix} p_1(0) \\ p_2(0) \end{bmatrix}=\begin{bmatrix} \theta(0) \\ \dot{\theta}(0) \end{bmatrix}$$

Por isso, a função *odefun*, seguirá a seguinte estrutura…

```
function dpdt=odefun(t,p)

c=…
m=…
L=…
g=9.81;
dpdt=zeros(size(p));
dpdt(1)=p(2);
dpdt(2)=-c/(m*L^2)*p(2)-g/L*sin(p(1));
```

Quem invoca a função *ode45*, só terá de usar a função desta forma…

```
theta0=…
Dtheta0=…

t_delta=[0:…:…];
p0=[theta0, Dtheta0];
t,theta]=ode45(odefun,t_delta,p0);
```

No fim da execução teremos um *array theta* de 2 colunas, em que a primeira contém os valores de θ e a segunda de $\dot{\theta}$ para a gama temporal indicada no array *t*.

Outra alternativa que evita o cálculo de equações diferenciais é a ferramenta *simulink* (embora o faça igualmente, mas de forma invisível para o utilizador), em que basta construir um modelo que representa o nosso sistema e, pela simulação, observar o comportamento de θ, $\dot{\theta}$ e $\ddot{\theta}$. Modificando o aspecto da equação diferencial atrás deduzida…

$$\ddot{\theta}=-\frac{c}{mL^2}\dot{\theta}-\frac{g}{L}\sin(\theta)$$

Facilmente construímos o modelo que representa o sistema (fig.2):

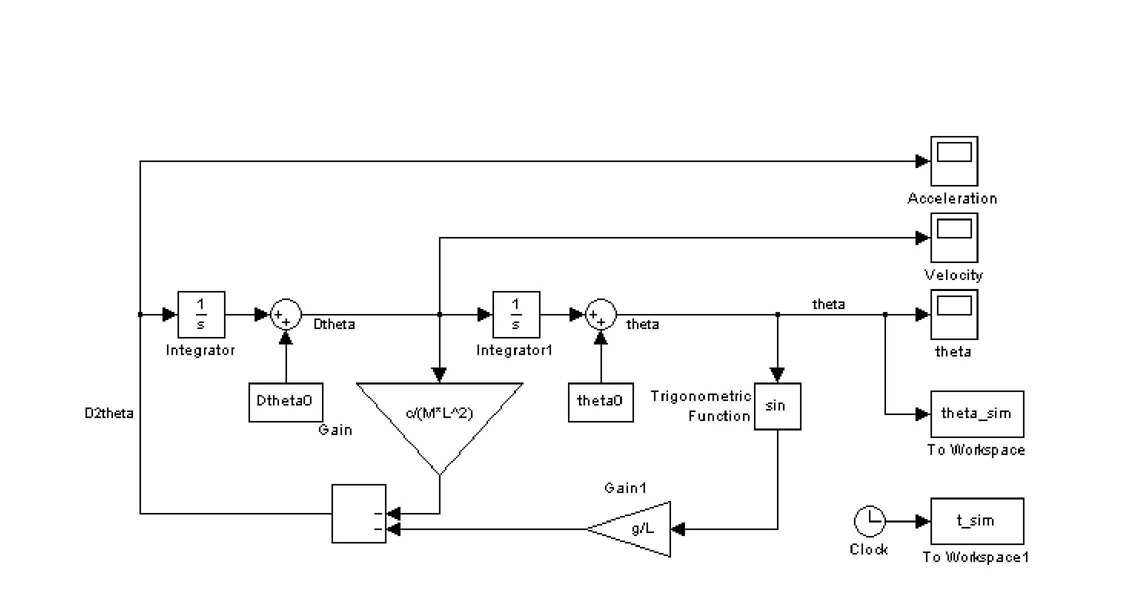

Fig. 2 - Modelo do pêndulo (em simulink)

Note o uso de integradores para a definição das derivadas de θ e a introdução de somadores, a seguir a eles, que introduzem as condições iniciais (θ_0 e $\dot{\theta}_0$).

Resultados:

Simulou-se e registou-se os valores de θ para a gama temporal de 0 a 50 segundos, com um passo de 0.05 segundos. O gráfico da figura 3 apresenta os resultados tanto no MatLab como no Simulink com as seguintes condições iniciais:

$$p(0) = \begin{bmatrix} \theta(0) \\ \dot{\theta}(0) \end{bmatrix} = \begin{bmatrix} \pi\,(rad) \\ 0\,(rad/s) \end{bmatrix}$$

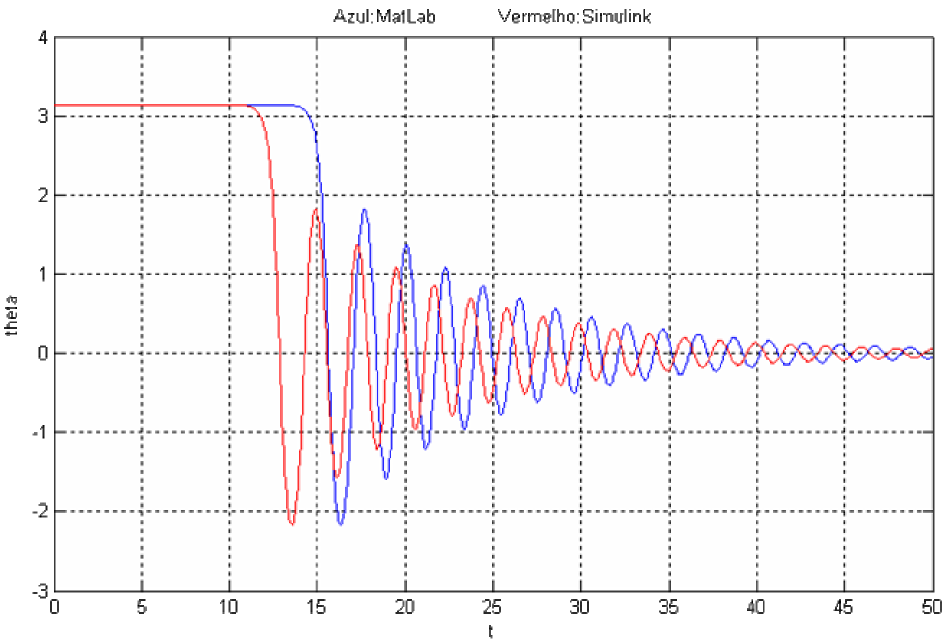

Fig. 3 - Resultados de Theta em MatLab (azul) VS Simulink (vermelho)

Embora o comportamento seja idêntico, os resultados fornecidos pelo Simulink apresentam-se adiantados relativamente aos do MatLab. Tal motivo ainda continua por esclarecer…

Uma nota que vale a pena referir, é o facto de que o nosso modelo matemático desenvolvido não comtempla nenhuma força horizontal, pelo que na posição de 180º e velocidade inicial nula, ele deveria permanecer lá indefinidamente. No entanto tal não acontece…
Este fenómeno deve-se à limitada resolução numérica da máquina de simulação, em que π radianos não é exactamente o valor associado, mas um pouco menos (erro introduzido na última casa binária), por isso, ao fim de algum tempo cai para a posição nula!