Projeto 1 - Física Computacional

Oscilador harmônico

Estevão Pimentel de Barcellos Santos

30 de agosto de 2021

Resumo

Afim de explorar o cálculo numérico e os processos computacionais envolvidos na simulação de um oscilador harmônico foi construído, usando pyton, um simulador interativo de um pêndulo simples amortecido. Para a solução numérica foi experimentado o método de Euler, mas devido à sua perceptível imprecisão, especialmente para um oscilador harmônico em duas dimensões, foi substituído pelo método de Runge-Kutta.

Oscilador harmônico

O que caracteriza um oscilador harmônico é que este pode ser descrito pela lei de Hooke.

$$F(x) = -kx \quad \leftrightarrow \quad m\ddot{x} + kx = 0$$

A solução dessa EDO é conhecida e pode ser expressa de duas formas

$$x(t) = A\sin(\omega_0 t - \delta)$$
$$x(t) = A\cos(\omega_0 t - \phi)$$

em que $\omega_0 = \sqrt{k/m}$. Isso para um oscilador harmônico unidimensional. A solução que nos interessa para o problema em questão é a do oscilador em duas dimensões, que de forma análoga é da da por

$$x(t) = A\sin(\omega_0 t - \alpha)$$
$$y(t) = B\cos(\omega_0 t - \beta)$$

A equação diferencial que descreve um oscilador harmônico amortecido, difere por um termo de força resistiva $-b\dot{x}$.

$$m\ddot{x} + b\dot{x} + kx = 0$$

Cuja solução é dada por

$$x(t) = e^{-\beta t}\left[A_1\exp\left(\sqrt{\beta^2 - \omega_0^2}t\right) + A_2\exp\left(-\sqrt{\beta^2 - \omega_0^2}t\right)\right]$$

Em que $\beta = b/2m$.
Nesta solução, existem três casos gerais de interesse:

Subamortecimento:	$\omega_0^2 > \beta^2$
Amortecimento crítico:	$\omega_0^2 = \beta^2$
Sobreamortecimento:	$\omega_0^2 < \beta^2$

Os quais buscaremos explorar em nosso modelo mais adiante.

Pêndulo simples

Podemos descrever um pêndulo simples a partir das forças que atuam sobre ele, isto é, o peso e a tensão do cabo. A tensão ata na direção radial ao movimento, enquanto o peso atua na vertical. Decompondo a tração em suas componentes horizontal e considerando o ponto mais baixo do arco como formando um ângulo $\theta = 0$ com o eixo do pêndulo temos que

$$T_x = T\cos\theta$$
$$T_y = T\sin\theta$$

As forças resultantes na vertical e na horizontal são, portanto,

$$F_x = T\cos\theta$$
$$F_y = T\sin\theta + P$$

em que P é a força peso exercida sobre a carga do pêndulo. A tração é dada por

$$T = \frac{mv^2}{L} + P\cos\theta$$

em que m é a massa da carga, v é sua velocidade, que é dada por $v = \sqrt{(v_x^2 + v_y^2)}$, e L é o comprimento do cabo. Portanto as forças nos sentidos vertical e horizontal ficam

$$F_x = \left(\frac{mv^2}{L} + P\cos\theta\right)\cos\theta$$
$$F_y = \left(\frac{mv^2}{L} + P\cos\theta\right)\sin\theta + P$$

Adicionando os termos de arraste bv_x e bv_y ficamos com

$$F_x = \left(\frac{mv^2}{L} + P\cos\theta\right)\cos\theta + bv_x$$
$$F_y = \left(\frac{mv^2}{L} + P\cos\theta\right)\sin\theta + P + bv_y$$

Com isso e conhecidas as condições iniciais podemos aproximar o estado deste sistema para qualquer instante de tempo usando calculo numérico iterando com pequenas variações de tempo.

Método de Euler

O método de Euler é um procedimento numérico de pirmeira ordem que consiste na simples discretização da variável independente e na aproximação da função para uma composição de pontos que correspondem aos incrementos calculados dentro dos intervalos de tempo Δt como mostra a figura a seguir.

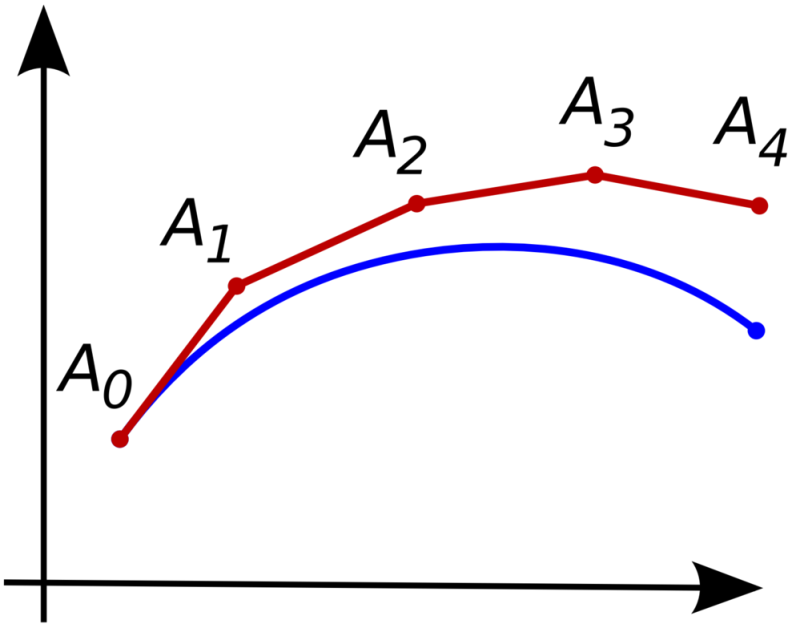

Figura 1: Ilustração do método de Euler. A curva desconhecida está em azul, e sua aproximação polinomial está em vermelho.

A precisão do método de Euler é perfeita para soluções lineares mas, como sugere a imagem é bastante grosseira para qualquer outro cenário. De fato, sempre podemos implementar o método de Euler com bastante precisão, basta reduzir suficientemente o tamanho de Δt e alcançaremos a precisão desejada. O problema é que isso pode acabar tornando-se computacionalmente dispendioso até o ponto de se tornar inviável. Como ilustra a imagem, podemos perceber que os erros se acumulam com o tempo, o que pode tornar longas simulações simplesmente inviáveis, sendo preferível recorrer a outro algoritmo pra encontrar a solução.

Para o problema do pêndulo, o método de Euler pode se mostrar razoável para pequenas oscilações e por um curto período de tempo. É possível estender o tempo e/ou o alcance de sua razoabilidade ao diminuir o tamanho de Δt mas eventualmente o erro se tornará evidente.

A fórmula recursiva do método de Euler para o problema do oscilador harmônico é dada por

$$\begin{aligned} x_{n+1} &= x_n + v_n \Delta t \\ v_{n+1} &= v_n - \frac{k}{m} x_n \Delta t \\ E_{n+1} &= \frac{1}{2} m \left(v_{n+1}\right)^2 + \frac{1}{2} k (x_{n+1})^2 \end{aligned}$$

Método de Runge-Kutta

O método de Runge-Kutta difere do método de Euler por não ser um método de passo único, isto é, a cada iteração são calculados passos intermediários para melhor estimar o próximo ponto.

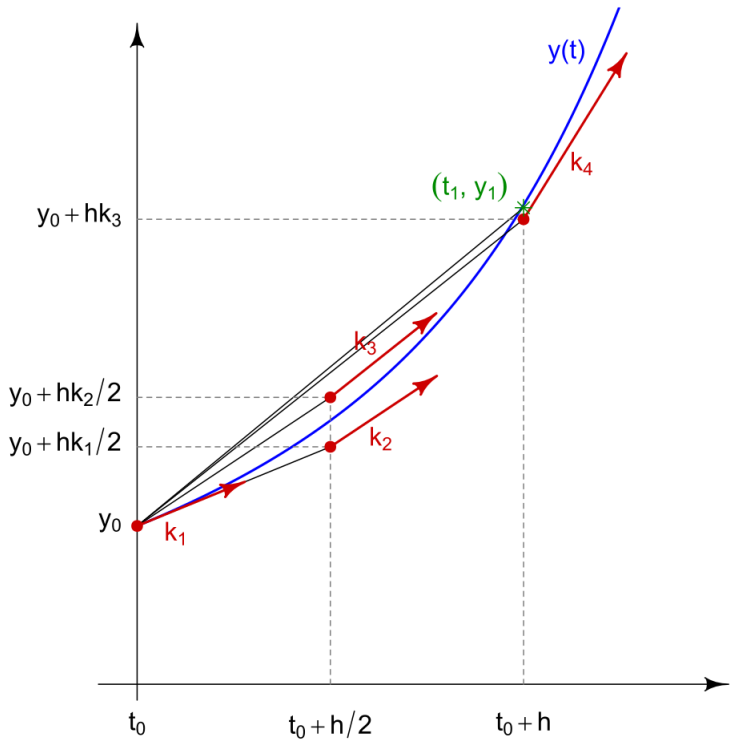

Figura 2: Inclinações utilizadas pelo método de Runge-Kutta de quarta ordem.

Como é possível visualizar na figura acima, este método utiliza das derivadas calculadas nos passos intermediários para estimar a inclinação média do intervalo para então estimar o proximo ponto da curva. Para nosso modelo implementaremos o método de Runge-Kutta de segunda ordem que utiliza apenas um passo intermediário e é exato para soluções quadráticas, o que já o torna uma aproximação muitíssimo melhor que a do método de Euler.

A fórmula recursiva do método de Runge-Kutta de segunda ordem para o problema do oscilador harmônico é dada por

$$
\begin{aligned}
k_1^x &= v_n \Delta t \\
k_1^v &= -(k/m) x_n \Delta t \\
x_{1/2} &= x_n + k_1^x/2 \\
v_{1/2} &= v_n + k_1^v/2 \\
k_2^x &= v_{1/2} \Delta t \\
k_2^v &= -(k/m) x_{1/2} \Delta t
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
x_{n+1} &= x_n + k_2^x \\
v_{n+1} &= v_n + k_2^v \\
E_{n+1} &= \frac{1}{2} m \left(v_{n+1}\right)^2 + \frac{1}{2} k \left(x_{n+1}\right)^2
\end{aligned}
}
$$

Algoritmo

O algoritmo implementado no código reflete a seguinte sequência:

Define-se a posição inicial (x_0, y_0) e a massa da carga. A velocidade inicial é zero;

Calcula-se o seno e o cosseno da atual posição em relação ao eixo do pêndulo: $\sin\theta = x_0/L$, $\cos\theta = y_0/L$;

Calcula-se a força de tração exercida do cabo sobre a carga: $T = (mv^2)/L + mg\cos\theta$;

Decompões-se a força do cabo em componentes vertical e horizontal, soma-se a componente vertical ao peso, soma-se ambas compontentes à sua respectiva força resistiva que será igual a uma constante multiplicada pela velocidade naquele eixo, e então calcula-se a aceleração da carga em ambos os eixos com $F = ma$;

A partir da aceleração, calcula-se a velocidade num incremento de tempo $\Delta t/2$ com $V = V_0 + a\Delta t/2$ e então calcula-se a posição deste ponto com $S = S_0 + v\Delta t/2$;

De forma análoga aos processos anteriores, calcula-se neste novo ponto a velocidade e a aceleração nos eixos vertical e horizontal e estas serão usadas para calcular a posicição e a velocidade da carga no incremento de tempo Δt a partir da posição e da velocidade inicial.

As novas posição e velocidade calculadas serão as condições iniciais da próxima iteração.

Uma réplica do código pode ser vista na próxima página. As variáveis estão nomeadas de forma sugestiva para que seja compreensível.

```
weight_pos = np.array([x0, y0])
delta_t = 1/60
Vx, Vy = 0, 0
mass_value = m
gravity_value = g
resistive_factor_value = r
while True:
    V0x, V0y = Vx, Vy
    sin_theta = (weight_pos[0]) / cable_lenght
    cos_theta = (weight_pos[1]) / cable_lenght
    T = ((mass_value * (Vx ** 2 + Vy ** 2))/(cable_lenght)) +\
        (mass_value * gravity_value * cos_theta)
    Fx = - T * sin_theta - resistive_factor_value * Vx
    Fy = - T * cos_theta + mass_value * gravity_value -\
        resistive_factor_value * Vy
    ax = Fx/mass_value
    ay = Fy/mass_value
    Vx_2 = (ax * (delta_t/2)) + Vx
    Vy_2 = (ay * (delta_t/2)) + Vy
    x_2 = weight_pos[0] + Vx * (delta_t/2)
    y_2 = weight_pos[1] + Vy * (delta_t/2)
    sin_theta_2 = (x_2) / cable_lenght
    cos_theta_2 = (y_2) / cable_lenght
    T_2 = ((mass_value * (Vx_2 ** 2 + Vy_2 ** 2))/\
         (cable_lenght)) + (mass_value * gravity_value *\
                            cos_theta_2)
    Fx_2 = - T_2 * sin_theta_2 - resistive_factor_value * Vx_2
    Fy_2 = - T_2 * cos_theta_2 + mass_value * gravity_value -\
          resistive_factor_value * Vy_2
    ax_2 = Fx_2/mass_value
    ay_2 = Fy_2/mass_value
    Vx = (ax_2 * (delta_t)) + V0x
    Vy = (ay_2 * (delta_t)) + V0y
    x = weight_pos[0] + Vx_2 * (delta_t)
    y = weight_pos[1] + Vy_2 * (delta_t)
    weight_pos = np.array([x, y])
```

As variáveis x0, y0, m, g e r representam valores arbitrários.

Regimes de amortecimento

Para validar nosso modelo convém que avaliemos como ele se comporta no que diz respeito aos regímes de amortecimento. Considerando um pêndulo simples com uma carga de massa igual a 10 kg, um cabo com 10 m de coprimento e uma aceleração gravitacional igual 10 m/s^2 e calculando o valor de β obtemos que $\beta = 4\ rad/s$ o que corresponde a um $b = 2\ kg/s$.

Fazendo testes com o simulador, é fácil perceber que com valores pequenos para o fator de arraste, isto é, para valores entre 0 e 1, obtemos um regime semiamortecido. É característica do regime amortecido reduzir gradativamente a amplitude de oscilação ao longo de vários ciclos de oscilação de forma que com o tempo a amplitude tende a zero. Da mesma forma, para valores maiores ou iguais a 3 já é notório o regime de sobreamortecimento. Este regime é caracterizado por nenhuma oscilação e uma grande redução na velocidade de queda da carga do pêndulo. O amortecimento crítico por sua vez, é caracterizado pelo arreste mínimo em que não existe nenhuma oscilação, e ao fazer testes com o oscilador, variando progressivamente o fator de arraste em 0.1, o valor em que esse comportamento foi encontrado foi para um fator de arraste igual a 2, o que corresponde ao valor de b previsto corroborando com a validade de nosso modelo.

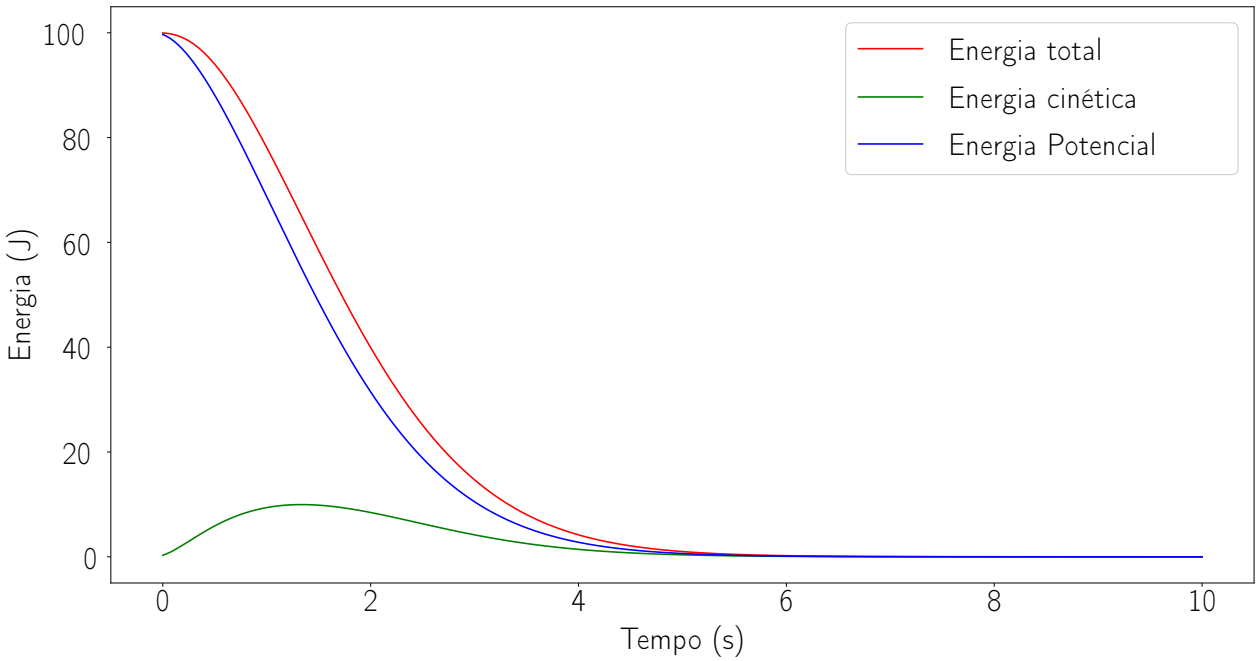

Figura 3: Energias do pêndulo em amortecimento crítico.

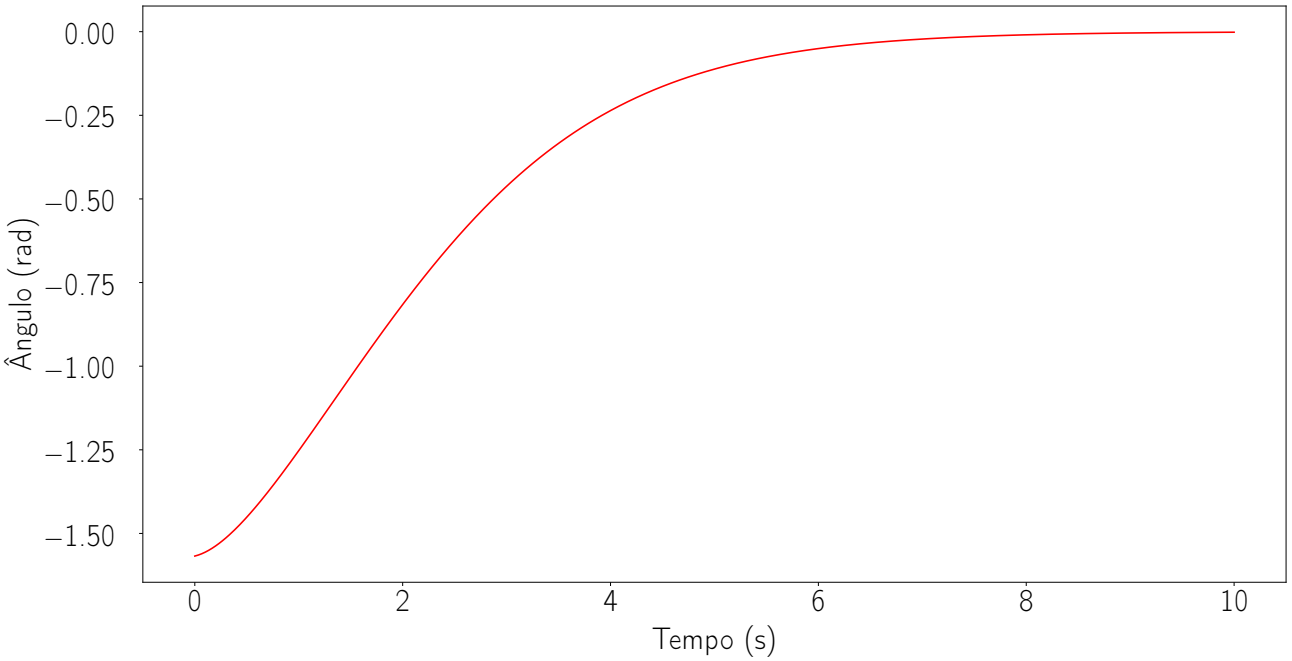

Figura 4: Ângulo de oscilação do pêndulo em amortecimento crítico.

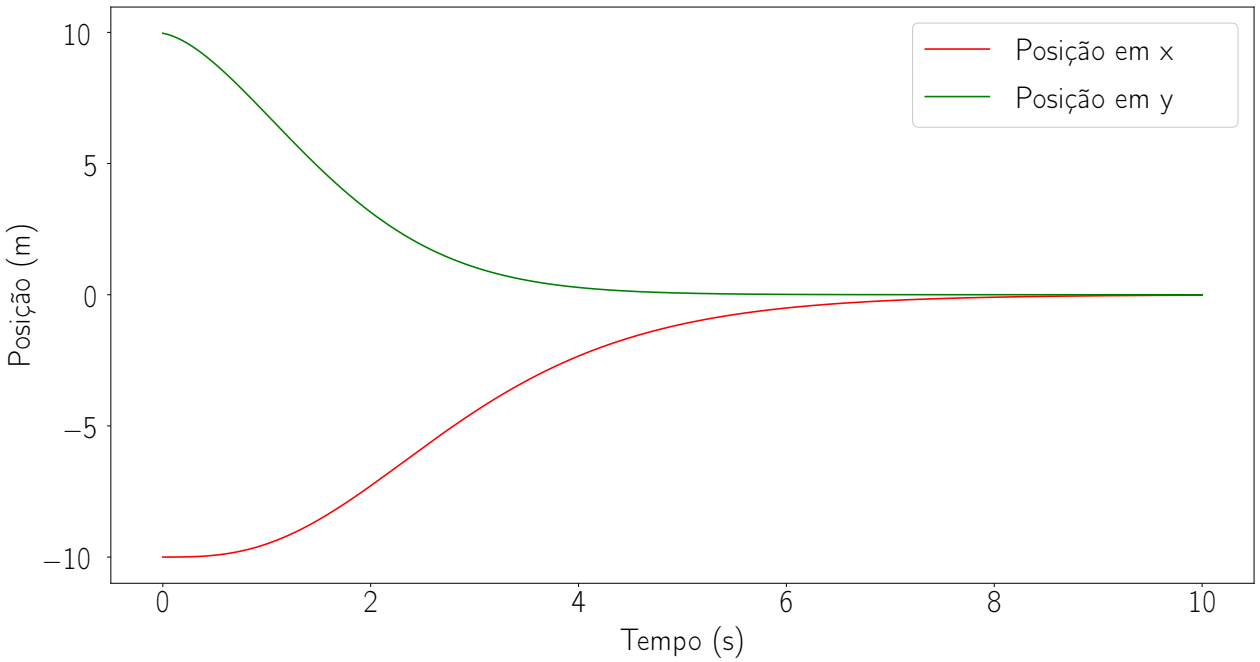

Figura 5: Posição nos eixos x e y da carga do pêndulo em amortecimento crítico.

Manual do usuário

Executando

O programa foi feito de forma a tornar sua utilização fácil e intuitiva, não sendo necessário interagir com o código para alterar os parâmetros de execução. Para sua construção foram utilizadas algumas bibliotecas que não são nativas do python e precisam ser instaladas para que o código possa ser executado. Tais bibliotecas são: pygame, pygame-menu, matplotlib e numpy. Estes dois últimos são velhos conhecidos, o numpy é necessário para realização de cálculos vetoriais eficientes e o matplotlib para exibição de gráficos. As duas primeiras são a princípio bibliotecas destinadas a produção de jogos mas são essencialmente bibliotecas que permitem manipulação de matrizes de pixels e a interação com entradas dos periféricos (mouse, teclado, ...) e por isso foram usadas na construção da interface gráfica. Tais bibliotecas podem ser instaladas pelo pip com os seguintes comandos no terminal:

```
pip install pygame
pip install pygame-menu
pip install numpy
pip install matplotlib
```

Feito isto, basta executar o código.

Iniciando a simulação

O programa inicia uma janela de 1920×1016 pixels que não pode ser redimensionada, e que tentar executá-lo em um monitor de resolução menor pode ser bastante frustrante. Ao iniciar basta selecionar a carga do pêndulo, representada pelo círculo azul-claro, posicionando o mouse sobre ela, pressionando o botão esquedo, movendo o mouse até alguma posição qualquer e largar o botão, e a simulação iniciará. A posição do mouse no momento em que o botão esquerdo for solto, definirá o ângulo inicial de oscilação (note que a posição inicial, bem como o movimento oscilatório é limitado entre os ângulos de $-\pi/2$ e $\pi/2$).

Alterando parâmetros

No menu à esquerda podem ser definidos os parâmetros de comprimento do cabo, massa da carga, aceleração gravitacional e fator de arraste. A alteração de qualquer um desses parâmetros impacta a execução em tempo real. A alteração pode ser feita selecionando o parâmetro desejado com o mouse ou com as teclas direcionais do teclado e digitando a alteração desejada. A alteração do comprimento implica num reinicio da simulação imediatamente definindo a velocidade da carga como zero. Simulações de alta frequência e/ou longos períodos de execução podem tornar evidentes os erros de aproximação, para tais casos basta reiniciar a simulação selecionando a carga com o mouse e dando-lhe uma nova posição inicial. Quando um parâmetro for alterado de forma a ocasionar divisão por zero nos cálculos, ou se a entrada possuir alguma incongruência, as iterações serão interrompidas até que a entrada seja aceitável.

Visualizando os gráficos

Ao executar o programa, pode-se observar um gráfico correspondente às energias cinética, potencial e total do pêndulo. Utilizando o seletor nomeado "gráfico" localizado no menu é possível alternar a exibição entre três gráficos: o gráfico das energias, o gráfico das posições, e o gráfico do ângulo. Para alternar entre os gráficos basta clicar no seletor com o mouse ou selecioná-lo com as teclas direcionais do teclado e usar as teclas direcionais direita e esquerda.

Abaixo do seletor existe um botão nomeado "Exportar gráfico". Ao clicar sobre ele ou seleciná-lo com as teclas direcionais do teclado e pressionar enter, o gráfico que estiver sendo exibido na tela será plotado em uma janela do matplotlib, permitindo que seja redimensionado, visualizado em detalhes e salvo nos formatos disponibilizados pelo matplotlib. Abaixo deste botão existe ainda um controle deslizante de alcance nomeado "Intervalo". Com ele é possível definir o intervalo do gráfico que será exportado quando o referido botão for pressionado. Este controle é composto por dois seletores deslizantes, um para o limite inicial e outro para o limite final do intervalo. Por padrão, o intervalo selecionado vai de -20 s à 0 s, sendo 0 s o instante de execução atual. Para selecionar o intervalo, basta selecionar com o mouse, e arrastar o seletor da esquerda até que exiba o valor de tempo correspondente ao tempo inicial desejado, e fazer o mesmo ao seletor da direita para o tempo final. Os seletores também podem ser manipulados usando as teclas direcionais direita e esquerda e é possível alternar entre os seletores com a tecla "TAB". Ao clicar em "Exportar gráfico", abre-se uma janela do matplotlib e com essa janela aberta é impossível interagir com a janela principal e para tornar a fazê-lo é necessário fechá-la.

É possível pausar e despausar a simulação apertando a tecla "P". É recomendável pausar a execução sempre que desejar observar uma mesma situação em diferentes gráficos, ou para selecionar tranquilamente o intervalo antes de exportar o gráfico, ou mesmo para alterar os parâmetros da simulação com calma. Sempre que a execução estiver pausada aparecerá no canto superior direito da tela o símbolo vermelho composto por duas barras verticais paralelas indicando que a tela está pauada. Sempre que clicar em "Exportar gráfico", a execução será pausada, sendo necessário apertar "P" para retomá-la.

Quando os parâmetros da simulação são alterados, é possível perceber que a escala do gráfico é alterada, mas os valores plotados antes da alteração, por padrão, não são redimensionados para a nova escala. Este é apenas um detalhe visual do gráfico em tempo real, quando exportado, todos os pontos estarão na corretamente dimensionados. Para ativar o redimensionamento em tempo real, permitindo que os pontos sejam readequados sempre que a escala do gráfico for alterada, basta apertar a tecla "R" e aparecerá um grande "R" da cor branca no canto superior direito da janela indicando que esta função está ativa. Para desativar esta função basta apertar a teca "R" novamente.

O gráfico pode ser limpo ao apertar a tecla "C". Ao apertar essa tecla todos os três gráficos serão limpos definindo todos seus pontos como zero.

Por fim, o programa pode ser fechado com o botão de "Sair", ou com as soluções usuais e atalhos disponíveis de acordo com o sistema operacional.

Precisão

Devido ao modelo aqui apresentado se tratar de uma aproximação numérica, existe um limite a ser respeitado para que haja uma precisão razoável. Neste caso o limite diz respeito às frequências de oscilação. A precisão deste modelo cai rapidamente com o aumento da frequência e isso poderia ser corrigido com uma redução do Δt, mas tal opção não foi incluída para que a passagem de tempo na simulação condissesse exatamente à passagem de tempo real numa execução à 60 fps, fora toda complicação implementar técnica que tal opção implicaria. Frequências da ordem de $1,65$ Hz já apresentam erro bastante notável, potanto para que seja possível observar a precisão da simulação, recomenda-se experimentá-la a frequências menores que esta, mas caso deseje explorar os limites da simulação não há nada que o impeça de ir em frente.

Mesmo a frequências mais baixas, com o passar do tempo a imprecisão do modelo tende a vir a tona, para remediar isso, sempre que a simulação é iniciada ao usuário posicionar a carga do pêndulo em seu ponto inicial, são posicionados limites que impedem que o erro se acumule ao longo do tempo ao corrigir a posição e a velocidade sempre que o pêndulo tentar acessar um ângulo que deveria estar inacessível dadas as condições iniciais. Apesar disso este erro pode se acumular em certas situações devido a alteração de parametros durante a execução.

Referências

Medeiros, R. (2016). Equações diferenciais - método de runge-kutta.

Stephen T. Thornton, J. B. M. (2011). *Dinâmica clássica de partículas e sistemas*. Cengage Learning, São Paulo.