UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

Projeto 3 - Movimento Realístico

7600017 - Introdução à Física Computacional
Prof. Dr. Eric de Castro e Andrade

Caio Cesar de Mello Capetti
Nº USP: 10288612

São Carlos
23 de Abril de 2018

Sumário

1 Introdução

O método de Euler é uma ferramenta para a solução de equações diferenciais, as quais são amplamente empregadas para resolver problemas físicos. A possibilidade de implementar o método de Euler por meio de um código abre caminho para a solução de problemas complexos de serem resolvidos manualmente.

Neste projeto são trabalhados três diferentes problemas físicos envolvendo o método de Euler. Os dois primeiros têm, por essência, observar situações que envolvem o efeito da resistência do ar. Considerar a resistência do ar aumenta o grau de complexidade de um problema físico, e por isso, frequentemente é desprezada. No entanto, com o poder computacional à dispôr, podemos enriquecer a abordagem de um problema, tornando-o mais realístico à medida que consideramos esse tipo de força dissipativa.

O último exercício apresenta uma oportunidade em que o método de Euler não é a ferramenta mais adequada para resolver o problema, introduzindo, então, o método de Euler-Cromer, que, sendo um aprimoramento do método de Euler, é capaz de encontrar a resolução do problema.

2 Métodos

2.1 Efeito resistivo do ar em bicicletas

2.1.1 Sem efeito resistivo

Foi escrito um programa FORTRAN que recebe como entrada a massa de um ciclista (m), sua potência ao pedalar uma bicicleta (P), o intervalo de tempo em que o se deseja estudar seu movimento ($deltat$), a partição que se deseja fazer nesse intervalo de tempo (dt) e a velocidade inicial do ciclista ($V0$).

Primeiramente, o programa aloca um vetor ($lista$) de ordem n, sendo n o quociente entre o tamanho do intervalo de tempo e o tamanho da partição desse intervalo, somado de 2 unidades, para incluir os extremos do intervalo. Esse vetor será interpretado como uma lista contendo as velocidades em cada instante. O índice i de cada velocidade corresponde ao número de vezes que se somou a velocidade extra produzida pela aceleração num intervalo de tempo do tamanho da partição realizada, somado de uma unidade, já que o primeiro elemento da lista, a velocidade inicial, tem índice 1.

Em seguida, construiu-se a lista, utilizando a relação 1, iterada para o tempo entre 0 e o tamanho do intervalo solicitado, de partição em partição. Esse método usa como base o **Método de Euler** para resolução de equações diferenciais, por esse motivo, com frequência, a velocidade obtida por esse método será referenciada em associação com o método de Euler. Ao final da montagem da lista, bastou consultar o último elemento dela para obter a velocidade após o intervalo de tempo informado.

$$v_{i+1} = v_i + \frac{P}{mv_i}\Delta t \tag{1}$$

Pôde-se então, comparar a velocidade obtida com a velocidade prevista pelo método tradicional, segundo o qual a velocidade em função do tempo é dada, com os parâmetros do problema, pela igualdade 2.

$$v^2(t) = v_0^2 + \frac{2P}{m}t \tag{2}$$

Por fim, visando encontrar a distância percorrida pelo ciclista durante o movimento, utilizou-se o **Método de Simpson**, descrito no projeto anterior, e que pode ser encontrado no código deste programa. Esse é um método de integração computacional baseado em conceitos de cálculo numérico, cujo uso mostrou-se pertinente no caso, por permitir encontrar o valor da grandeza desejada com boa precisão. Foi necessário fazer uma única alteração significativa no método, para que ele funcionasse no contexto em que estava inserido: A mudança foi solicitar que o programa buscasse por um valor em uma lista, em vez de calcular o valor por uma função. Ademais, foi preciso apenas definir os limites de integração corretamente e verificar o índice da lista para que ele indicasse o valor desejado. O valor obtido foi comparado com aquele fornecido pelo software *Wolfram Mathematica* quando solicitado que ele fizesse a integração da função 2 entre 0 e 300.

O arquivo de saída desse programa mostrava a velocidade do ciclista após o intervalo de tempo pedido, a velocidade esperada e a distância percorrida por ele ao final desse intervalo.

2.1.2 Com efeito resistivo

O programa assemelha-se com o anterior, com algumas modificações, que serão descritas a seguir.

O programa recebe além das variáveis já apresentadas, a densidade do ar (ro), a área de secção transversal do conjunto ciclista-bicicleta (s), e a precisão para se aproximar da velocidade terminal (eps) lidas juntamente com as demais variáveis, no mesmo arquivo. As duas primeiras novas variáveis estão relacionadas com propriedades do ar e do corpo que nele se move, e são relevantes para o cálculo da resistência, que afeta as grandezas cinemáticas envolvidas no problema. A última variável deve existir pelo fato de o ciclista nunca de fato chegar à velocidade terminal, fazendo com que o programa itere infinitamente na tentativa de calcular em quanto tempo o ciclista atinge essa velocidade, dessa forma, deve-se informar quão próximo se deseja chegar da velocidade terminal, limitando o número de iterações.

Podemos calcular a velocidade terminal do ciclista, dada pela relação 3, e construir a lista de velocidades, sendo que a condição que mantém a iteração, nesse caso, é a diferença entre a velocidade do ciclista e a velocidade terminal ser maior que eps, e a velocidade em cada instante é dada pela equação 4.

$$v_{terminal} = \left(\frac{2P}{\rho s}\right)^{1/3} \tag{3}$$

$$v_{i+1} = v_i + \frac{P}{mv_i}\Delta t - \frac{\rho s v_i^2}{2m}\Delta t \tag{4}$$

Com esses cálculos, temos condições de informar ao usuário a velocidade terminal esperada e em qual instante o ciclista atinge tal velocidade com a precisão solicitada, que consiste do último valor de tempo assumido durante a iteração. As informações foram registradas no arquivo de saída.

Em seguida, continuou-se o processo de iteração até que o tempo chegasse ao final do intervalo, a fim de se determinar a distância percorrida nesse movimento. Para esse objetivo, o procedimento foi exatamente o mesmo: aplicar o Método de Simpson de integração, adaptado para utilizar a lista. O resultado da integral é exatamente a distância percorrida, que então foi escrita no arquivo de saída.

Uma análise interessante de ser feita é a influência da área nessas velocidades, ela é fundamental para melhorar a performance de atletas nesse tipo de esporte. Para isso, modificou-se a área e plotou-se o gráfico de velocidade por tempo para diferentes valores de área.

2.2 Lançamento de projéteis

2.2.1 Sem efeito resistivo

Com o intuito de observar o papel da resistência do ar em lançamentos oblíquos, criou-se um programa FORTRAN capaz de receber como entrada a velocidade inicial do projétil ($V0$), o ângulo de lançamento ($theta$), a aceleração da gravidade (g) e a partição do intervalo de tempo (dt) e calcular a trajetória do projétil, para fazer uma descrição mais profunda do movimento.

Primeiramente os dados são lidos de um arquivo e o ângulo de lançamento é transformado para radianos. Determina-se que a posição inicial nos eixo x e y são ambas 0, isto é, o projétil partiu da origem, e decompõe-se a velocidade em cada eixo, utilizando o módulo da velocidade inicial e o ângulo de lançamento. Determinados os parâmetros iniciais, é possível construir uma iteração que atualize os valores das posições e das velocidades de acordo com as equações que caracterizam o movimento, que são as expressões de 5 a 8.

$$x_{i+1} = x_i + v_{x,i}dt \tag{5}$$

$$v_{x,i+1} = v_{x,i} \tag{6}$$

$$y_{i+1} = y_i + v_{y,i}dt \tag{7}$$

$$v_{y,i+1} = v_{y,i} - gdt \tag{8}$$

Essas equações são coerentes com as equações que descrevem um lançamento de projéteis em um caso ideal onde se despreza o efeito resistivo do ar, já que representam um movimento uniforme na direção x e um movimento uniformemente acelerado em y, onde a aceleração é a gravidade. A combinação dessas equações permite-nos encontrar uma função que relaciona o alcance e o tempo, a equação 9, e uma equação que dá o tempo de voo, a equação 10.

$$A = \frac{v_0^2 sen(2\theta)}{g} \tag{9}$$

$$t_{voo} = \frac{2v_0 sen(\theta)}{g} \tag{10}$$

Bastou atualizar os valores de posição e velocidade em cada eixo até que a posição no eixo y valesse 0 novamente. Para evitar que o programa não entrasse no laço iterativo (o que aconteceria naturalmente, visto que a posição inicial em y é 0), fez-se a primeira iteração explicitamente, antes de iniciar o *loop*. Ao final do processo, pôde-se plotar as posições em x e em y que o programa forneceu em cada iteração para várias entradas de ângulo diferentes.

A determinação do ângulo de lançamento que proporciona o maior alcance, junto com o tempo de voo pode ser feita de maneira trivial nesse caso, por cálculos simples. Em outras palavras, não é preciso de um gráfico de alcance em função do ângulo, como será necessário no próximo caso. Para encontrar o ângulo de alcance máximo, derivou-se a equação 9 em relação a θ e igualou-se a zero, finalmente, resolvemos a equação em θ. Para obter o tempo de voo, é preciso apenas utilizar a equação 10 com o ângulo encontrado anteriormente.

2.2.2 Com efeito resistivo

Nesse programa, a mesma ideia do caso sem resistência do ar foi implementada, mas, dessa vez, as equações de posição e de velocidade foram atualizadas de maneira diferente e uma análise mais profunda foi feita, levando em consideração a variação da densidade do ar, por exemplo. Além disso, foi necessário dar como entrada a razão γ_0/m, representada com r_0 para fins práticos.

As posições no eixo x e no eixo y, bem como as velocidades nos mesmo eixos foram atualizadas da seguinte maneira:

$$x_{i+1} = x_i + v_{x,i}dt \tag{11}$$

$$v_{x,i+1} = v_{x,i} - r_0\sqrt{v_{x,i}^2 + v_{y,i}^2}\; v_{x,i}dt \tag{12}$$

$$y_{i+1} = y_i + v_{y,i}dt \tag{13}$$

$$v_{y,i+1} = v_{y,i} - gdt - r_0\sqrt{v_{x,i}^2 + v_{y,i}^2}\; v_{y,i}dt \tag{14}$$

Dessa maneira, podemos obter a trajetória do projétil, levando-se em conta o efeito resistivo do ar. Nosso interesse, agora, é observar qual é o ângulo que proporciona o maior alcance e o tempo de voo desse lançamento. Para isso, depois de calculada a trajetória, uma modificação foi feita no programa: em vez que solicitar um ângulo ao usuário, o programa deveria fazer o cálculo da trajetória com ângulos entre 0 e $\pi/4$ rad, em intervalos de 10^{-3} rad. Então, obteve-se um gráfico do alcance em função do ângulo do qual foi descoberto o valor para o alcance máximo, com ajuda do software *Microsoft Excel*.

O tempo de voo do projétil é o tempo em que a posição no eixo y iguala-se a zero, então, bastou ordenar que o programa fornecesse o valor do tempo quando as iterações eram finalizadas para descobrir o tempo de voo.

Finalmente, consideraremos a diferença na densidade do ar, proveniente da elevada altura que o projétil atinge. Para esse fim, o fator γ_0 foi modificado da seguinte forma:

$$\gamma_1 = \gamma_0 \left(1 - b\frac{y}{T}\right)^{\alpha} \tag{15}$$

O que implica uma modificação em r_0 conforme a seguinte:

$$r_1 = r_0 \left(1 - b\frac{y}{T}\right)^{\alpha} \tag{16}$$

Trocando r_0 por r_1 nas equações 11 - 14, calculou-se, mais uma vez, a trajetória do projétil para diferentes valores de ângulo de lançamento, analisando, também, qual ângulo maximiza o alcance pelo mesmo método com que procedeu-se anteriormente, e qual o tempo de voo para esse ângulo.

Uma última análise feita sobre os resultados foi observar a diferença de alcance que uma pequena variação na velocidade inicial provoca, elucidando a necessidade de precisão em cálculos nesse tipo de problema.

2.3 Pêndulo Simples

Faremos, agora, uma análise do movimento de um pêndulo simples, com objetivo de testar dois métodos diferentes para resolver as equações diferenciais envolvidas nesse tipo de movimento.

2.3.1 O método de Euler

Primeiramente, vamos implementar o método de Euler para descrever a posição angular (θ) e a velocidade angular (ω) do pêndulo. Com essas grandezas, podemos analisar a energia mecânica do sistema. A expectativa é que a energia mecânica mantenha-se constante pelo menos para cada período de oscilação.

O programa recebe a massa, m, do pêndulo, seu comprimento, l, sua posição inicial, θ_0, a aceleração da gravidade, g, o intervalo de tempo, *deltat* e a partição do intervalo de tempo, dt. Estabeleceu-se, então, um processo iterativo em que se calculava, a cada passo de tempo, a posição angular, a velocidade angular e a energia mecânica total do sistema, pelas equações 17, 18 e 19

$$\omega_{i+1} = \omega i - \frac{g}{L} sen(\theta_i) dt \tag{17}$$

$$\theta_{i+1} = \theta i + \omega_i dt \tag{18}$$

$$E_i = \frac{1}{2} m L^2 \omega_i^2 + mgL(1 - cos(\theta_i)) \tag{19}$$

Dessa forma, o programa escreve em um arquivo a posição angular θ_i em cada instante calculado, bem como a energia mecânica do sistema e o tempo. Esses dados permitem plotar um gráfico da posição angular em função do tempo e da energia mecânica em função do tempo. Esperamos obter uma oscilação com amplitude constante, e uma energia mecânica que se conserva, em média.

Vale a pena ressaltar o cuidado que se deve tomar ao atualizar o valor de θ_i e de ω_i, pois, como são dependentes uma da outra, a primeira a ser atualizada impede a atualização da segunda. Por esse motivo, é preciso criar uma variável para armazenar o valor da variável θ_i ou ω_i (a que for atualizada primeiro), chamada de aux, antes que ela seja convertida em θ_{i+1} ou ω_{i+1}.

2.3.2 O método de Euler-Cromer

Uma alternativa ao método de Euler é o método de Euler-Cromer, que também será implementado. O procedimento, aqui, é o mesmo do caso anterior, com uma única alteração: a posição angular no instante $i+1$ depende da velocidade angular no instante $i+1$, diferente do que ocorria anteriormente, quando θ_{i+1} dependia de ω_i. Em forma de equação, isso se expressa como descreve a relação 20. Dessa vez, não precisaremos de uma variável auxiliar, basta atualizar primeiro a velocidade angular, e, depois, a posição angular.

$$\theta_{i+1} = \theta i + \omega_{i+1} dt \tag{20}$$

2.3.3 Período do movimento pendular

Estaremos interessados, também, na análise do comportamento do período de oscilação para diferentes valores de posição angular inicial. Com esse objetivo, foi preciso escrever um programa que aplicasse o método de Euler-Cromer, mas sem a entrada de θ_0, cujo valor será modificado em uma iteração. A posição inicial do pêndulo variará entre 0 e π radianos, a cada 1 milésimo. Além disso, será preciso adaptar o programa para que ele encontre o período do movimento.

Para isso, será utilizado o seguinte fato: quando o pêndulo muda o sentido de sua trajetória, sua velocidade se anula, e, nesse momento, o corpo terá metade do período de seu movimento. De posse dessa informação, foi necessário impor como condição de parada do laço iterativo presente no método de Euler-Cromer a velocidade atingir zero. Basta, então, multiplicar o tempo que o programa encontrou por 2 para obter o período do movimento.

Podemos comparar esse resultado com a previsão teórica, segundo a qual, o período respeita a equação 21

$$T = 4\sqrt{\frac{L}{g}} \int_0^{\pi/2} \frac{du}{\sqrt{1 - k^2 sen^2 u}} \tag{21}$$

E sabemos que:

$$k = sen\frac{\theta_0}{2} \tag{22}$$

Com os diversos pontos de período e posição inicial, podemos obter um gráfico que explicite a relação entre essas duas grandezas, constituindo um meio visual de analisar os resultados. Para efeitos de comparação, será utilizado o software *Wolfram Mathematica* para plotar o gráfico da função 21.

3 Resultados

3.1 Efeito resistivo do ar em bicicletas

Os programas foram executados com as seguintes entradas: a massa do ciclista era de 70 kg ($m = 70$), a potência desenvolvida por ele foi de 400 W ($P = 400$), o tempo analisado foi de 300 s ($deltat = 300$), as partições realizadas foram de 0.1 s ($dt = 0.1$), a densidade do ar foi considerada 1.3 kg/m^3 ($ro = 1.3$) e exigiu-se que a velocidade terminal e a velocidade do ciclista fossem diferentes em 10^{-2} ($eps = 10^{-2}$). Os valores da área serão alterados, e, por isso, informados quando oportuno.

3.1.1 Sem efeito resistivo

O Método de Euler forneceu uma velocidade de aproximadamente 58.7036 m/s, enquanto o esperado era que a velocidade valesse aproximadamente 58.6905 m/s, o que representa uma diferença de cerca de 0.022%. Esses valores mostram que o Método de Euler é eficiente para aproximar a velocidade nesse tipo de problema, dada a pequena diferença percentual entre os valores.

Foi possível construir dois gráficos com os pontos encontrados pelo programa em cada iteração. Um gráfico contempla o valores da velocidade encontrados pelo Método de Euler e o tempo, enquanto o outro contempla os valores esperados de velocidade e o tempo. Os gráficos estão representados na figura 2 e corroboram a conclusão, além de elucidarem que para o caso sem resistência do ar, a velocidade não possui valor máximo, não apresentando comportamento assintótico. Isso é coerente com a teoria, já que a ausência de forças resistivas faz com que a aceleração não se aproxime do zero para nenhum valor de velocidade.

A distância percorrida pelo ciclista ao final dos 300 s foi de aproximadamente 11798.22 m, o que se mostra um valor extremamente alto, indicando que desprezar a resistência do ar nesse problema o simplifica por demasiado, distanciando-se notavelmente da realidade com que os fenômenos ocorrem. Os resultados exatos expressos pelo programa estão ilustrados na figura 1.

Figura 1: Resultados expressos pelo programa.

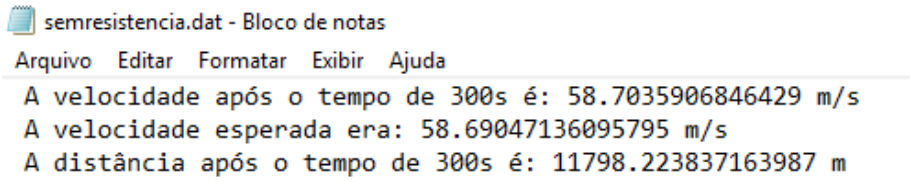
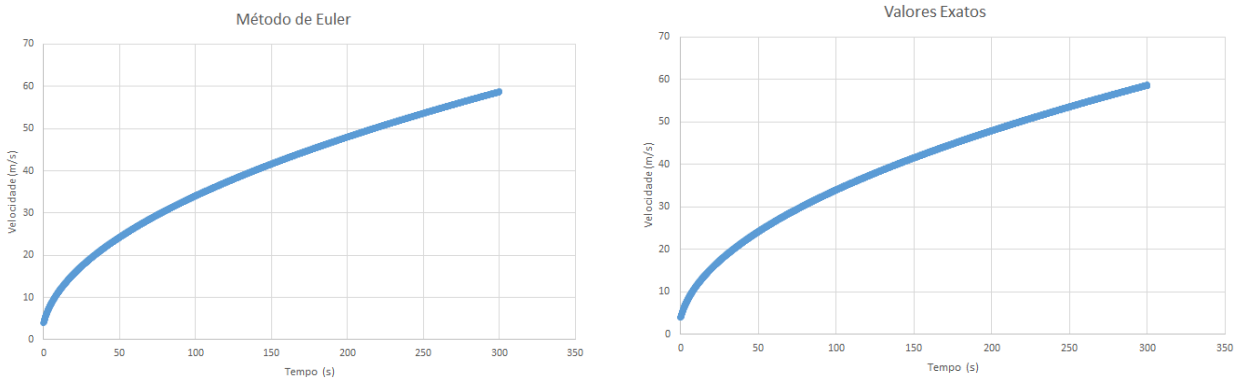


Figura 2: Gráficos para comparação do resultado obtido com o resultado esperado.



(a) Método de Euler

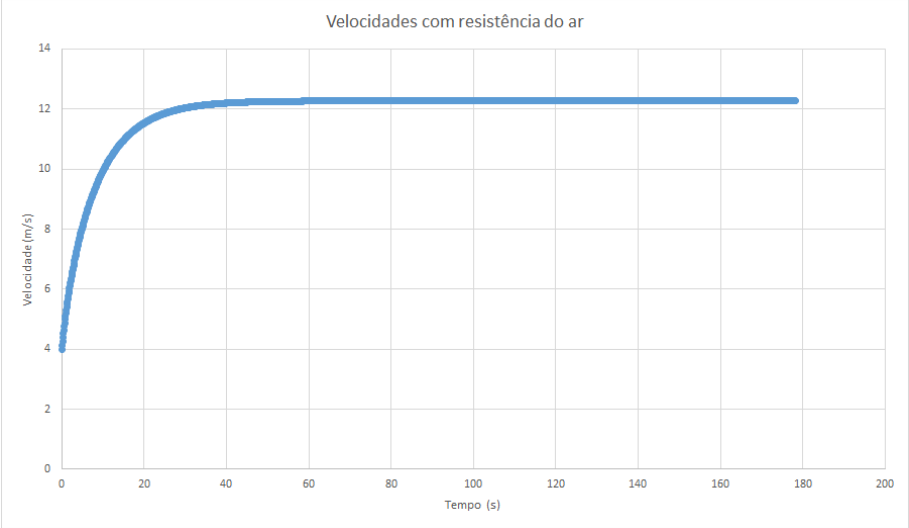
(b) Resultado esperado

3.1.2 Com efeito resistivo

Levando-se em consideração o efeito resistivo que o ar provoca sobre o ciclista, com uma área de secção transversal de 0.333 m^3 ($s = 0.333$), foi possível perceber que a velocidade atinge um valor máximo depois de um certo tempo, essa velocidade é denominada velocidade terminal, e foi calculada como sendo aproximadamente 12.2716 m/s. Podemos perceber esse comportamento pelo gráfico de velocidade por tempo que os dados fornecidos pelo programa permitiram plotar: o gráfico da figura 3.

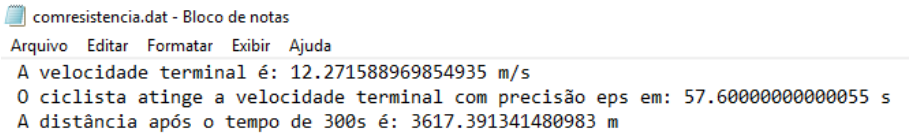
O programa mostrou, junto com o gráfico, que a velocidade terminal, com precisão de 10^{-2}, é atingida em aproximadamente 57.6 s, divergindo apenas 0.08% do valor calculado analiticamente. É possível observar no gráfico que a velocidade fica praticamente constante a partir desse valor, variando de maneira imperceptível para ser representada no gráfico.

Figura 3: Gráfico da velocidade pelo tempo com resistência do ar.



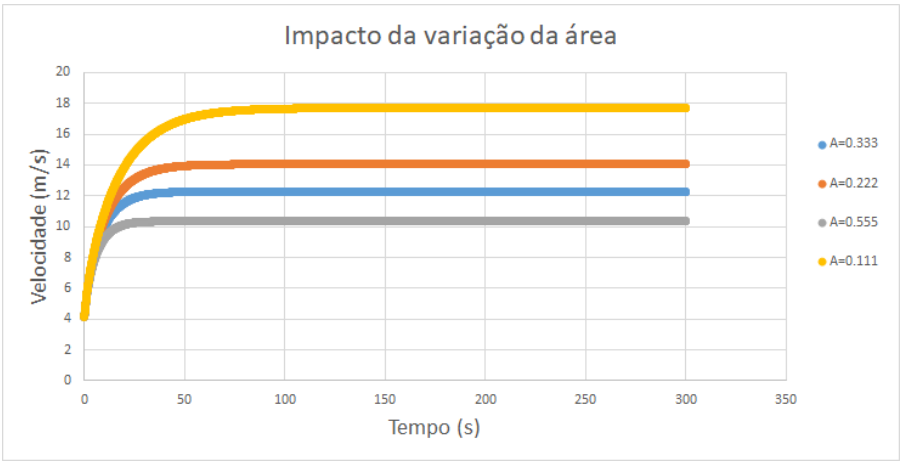
A distância percorrida nos 300 s para esse caso foi de aproximadamente 3617.39 m, o que se mostrou muito inferior à distância percorrida quando desprezamos a resistência do ar, sendo, agora, plausível para um ciclista profissional. A seguir, representa-se a saída do programa, na figura 4.

Figura 4: Resultados expressos pelo programa



Para estudar a interferência da área nos resultados, apresenta-se, na figura 5, os gráficos de velocidade por tempo para os seguintes valores de área: 0.111 m^3, 0.222 m^3, 0.333 m^3 e 0.555 m^3.

Figura 5: Gráfico da velocidade em função do tempo para diversos valores de área.



Pelo gráfico, percebemos que quanto menor a área, maior a velocidade terminal, isto é, maior a velocidade máxima que o ciclista pode atingir. Esse aumento na velocidade máxima é crucial para o aumento do desempenho do ciclista, por isso, uma técnica comum empregada por esses atletas é encolher-se de forma a reduzir ao máximo sua área de secção transversal.

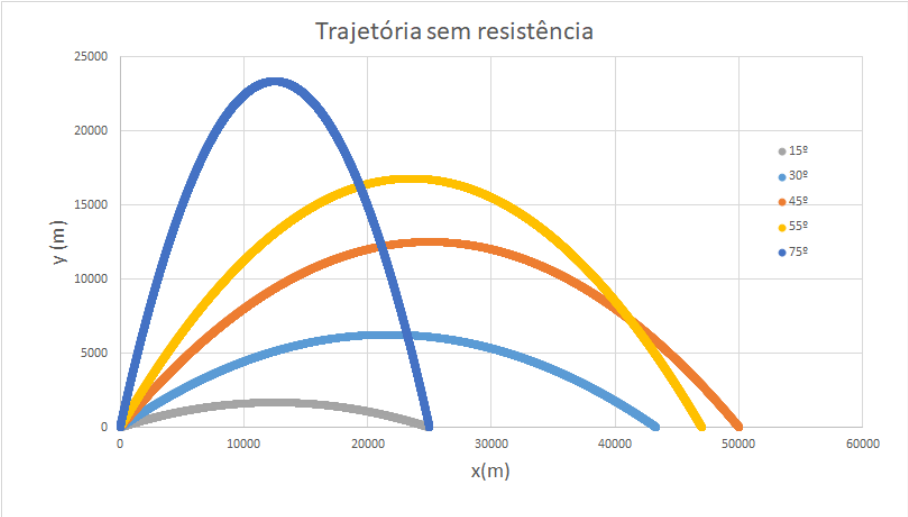
3.2 Lançamento de projéteis

Nos programas desse exercício foram utilizadas como entrada os seguintes valores: a velocidade inicial do projétil foi considerada 700 m/s ($v0 = 700$), a aceleração da gravidade foi considerada 9.8 m/s^2 ($g = 9.8$) e a partição do intervalo de tempo tinha tamanho 0.01 s ($dt = 0.01$). As demais variáveis serão explicitadas quando conveniente.

3.2.1 Sem efeito resistivo

Além das entradas descritas, pediu-se que o programa calculasse a trajetória com os ângulos de 15°, 30°, 45°, 55° e 75°. As trajetórias obtidas para esses ângulos estão ilustradas no gráfico da figura 6.

Figura 6: Gráfico das trajetórias para lançamento sem resistência.



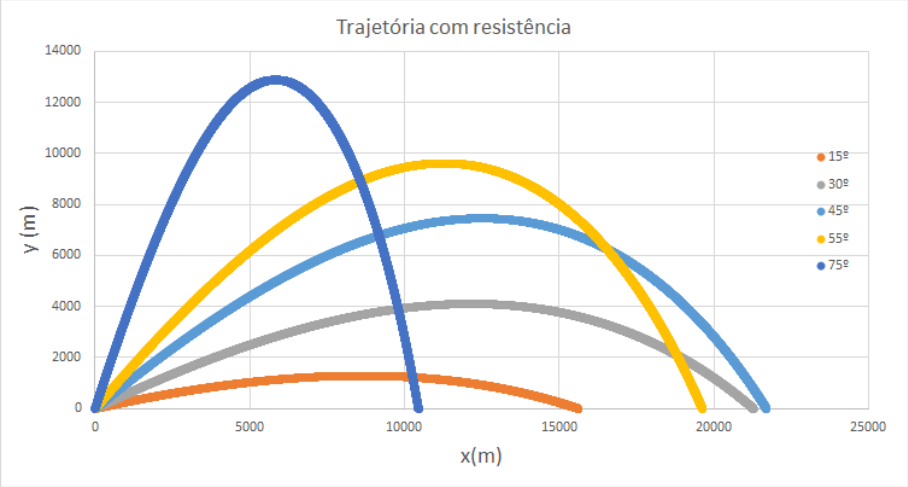
Pelo método de derivar a equação 9 e igualar a zero descrito anteriormente, pôde-se concluir que, para um lançamento oblíquo sem resistência do ar, o ângulo que dá o maior alcance é 45°, sendo o alcance 50002.35 m. O tempo de voo para o ângulo de lançamento que promove um alcance máximo é, pela equação 10, aproximadamente 101.015 s.

3.2.2 Com efeito resistivo

Quando executado o programa, foi obtida uma lista com diversos valores de posição e velocidade, assim como obtido no caso anterior. Esses dados, plotados em um gráfico, fornecem a trajetória do projétil para o caso de densidade constante do ar, e estão ilustrados na figura 7.

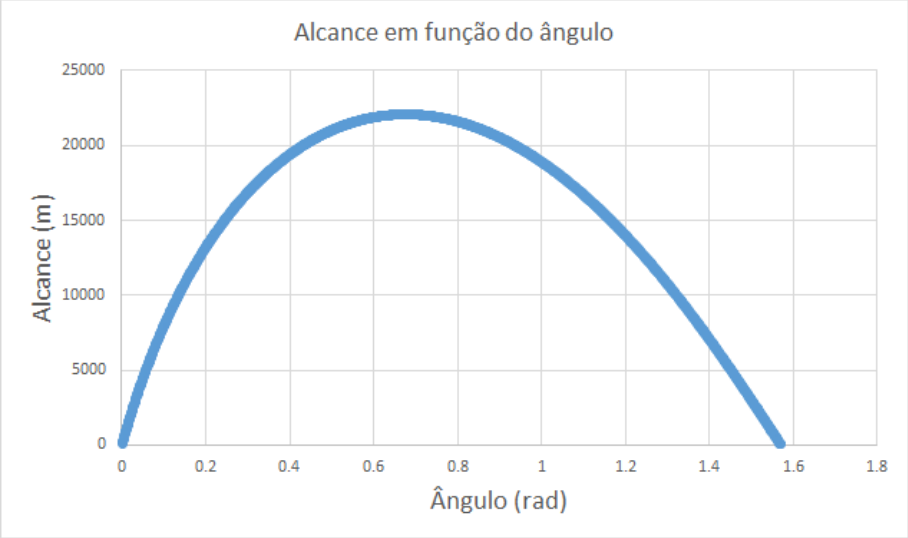
Podemos perceber que esse gráfico difere daquele obtido quando desprezado o efeito resistivo do ar em aspectos fundamentais, como o alcance, notavelmente menor neste caso, a altura máxima atingida, também menor quando considerada a resistência e a própria trajetória, que parece ter uma queda mais acentuada, não sendo simétrica como no caso anterior. A inclusão da resistência associa os movimentos em x e em y, diferente do que ocorria anteriormente.

Figura 7: Trajetórias de lançamento oblíquo com resistência do ar.



Observando os resultados de uma variante do programa que fazia o ângulo de lançamento variar de 0 até $\frac{\pi}{2}$, em intervalos de 10^{-3}, pôde-se perceber que o maior alcance obtido foi de 22071.77 m que corresponde ao ângulo de lançamento de $0.676rad$. Um gráfico com os resultados obtidos está representado na figura 8.

Figura 8: Alcances obtidos para alguns ângulos de lançamento.



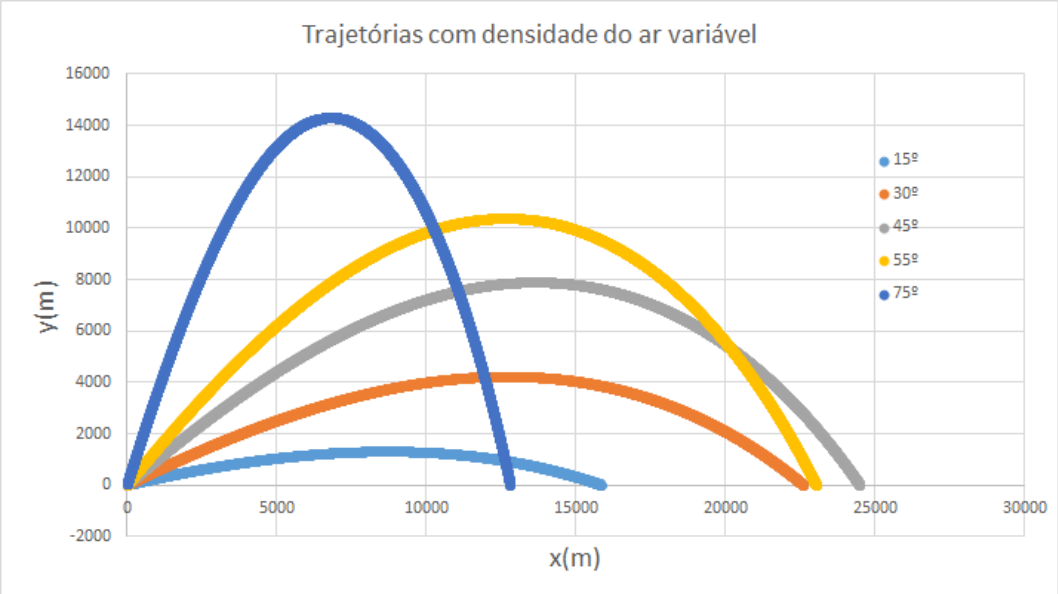
O tempo de voo do projétil quando submetido ao ângulo que proporciona o maior alcance é de, aproximadamente, 69.62 s, no caso em que não se leva em consideração a variação na densidade do ar devida à altura que o projétil atinge.

Observamos, até agora, que ao considerarmos a resistência do ar, sendo o ar um fluido de densidade constante no espaço, obtemos resultados que divergem consideravelmente daqueles onde simplificamos nosso problema a um lançamento oblíquo sem efeito resistivo do ar. Vamos dar mais um passo, considerando, agora, a variação da densidade do ar, fazendo-nos aproximar ainda mais de uma descrição realista do movimento proposto.

Considerando-se, agora, a variação da densidade, fez-se necessária a introdução de novas variáveis, a saber:

os coeficientes α e b e a temperatura $temp$, que valem, respectivamente, 2.5 , $6.5 \cdot 10^{-3}$ K/m e $300K$ ($\alpha = 2.5; b = 6.5 \cdot 10^{-3}; temp = 300$), com isso, plotou-se a trajetória do projétil, conforme representado na figura 9.

Figura 9: Trajetórias considerando a variação de densidade do ar.



É possível observar que ao considerarmos a variação da densidade do ar, o ângulo de lançamento assume um papel ainda mais importante na trajetória: Para pequenos ângulos de lançamento, a variação da densidade do ar é mínima, e a trajetória não se altera. Para ângulos maiores, no entanto, o projétil atinge maior altitude, e, com isso, a variação da densidade do ar passa a ser significativa, de modo que o corpo ganha maior alcance, já que a resistência do ar tem menor impacto em maiores altitudes.

Alterando-se o programa a fim de observar a variação do alcance com a variação do ângulo de lançamento, como procedido anteriormente, foi possível descobrir, aproximadamente, o valor do ângulo que proporciona o maior alcance, o valor desse alcance, e o tempo de voo nesse alcance.

O alcance máximo obtido foi de aproximadamente 24523.68 m para um ângulo de lançamento de 0.762 rad. Nessas condições, o tempo de voo é de 78.09 s. Esses resultados mostram-se intermediários entre os dois obtidos anteriormente, o que é razoável, já que antes, estudamos casos sem resistência alguma e com resistência invariante e, agora, analisamos um caso em que o efeito resistivo existe, mas diminui com a altura.

Finalmente, executamos o programa com 3 velocidades iniciais diferentes, variando 1% a mais e 1% a menos do valor proposto no princípio. Ou seja, o programa foi executado com as seguintes velocidades iniciais: 693 m/s, 700 m/s e 707 m/s. Os alcances para essas velocidades estão na tabela 1

Tabela 1: Alcances para diferentes velocidades iniciais.

Velocidade inicial (m/s)	Alcance (m)
693.0	24216.43599802673
700.0	24523.684730316203
707.0	24829.62951280244

A tabela mostra que para uma variação pequena na velocidade inicial do projétil, mais precisamente, 1%, o alcance muda em cerca de 300 metros, explicitando a dificuldade que se tem de atingir com precisão um alvo nesse tipo de movimento.

3.3 Pêndulo Simples

3.3.1 Trajetória e Energia

Os programas foram executados com as seguintes entradas: a massa do pêndulo foi estabelecida como 1 kg ($m = 1$), o comprimento do pêndulo é de 1 m ($l = 1$), a aceleração da gravidade é de 10 m/s^2, a posição inicial foi de 30°($\theta_0 = 30$), o intervalo de tempo desejado foi de 20 s ($deltat = 20$) e a partição do intervalo de tempo foi de 0.005 s ($dt = 0.005$).

Ao executar o programa e analisar os dados fornecidos por meio do Método de Euler, observamos os gráficos da posição angular e da energia em função do tempo.

Percebemos que a amplitude do movimento aumenta com o tempo, assim como a energia mecânica total do sistema, diferente do que esperávamos. Isso indica uma falha no método de Euler. Vamos, então, verificar quais as respostas do método de Euler-Cromer para esse problema.

Executando o programa do método de Euler-Cromer com as mesmas entradas do caso anterior, foram obtidos os resultados que se expressam na forma de gráficos. Os gráficos das figuras 10 e 11 ilustram as diferenças entre os dois métodos propostos

Pelo método de Euler-Cromer, atingimos nossa expectativa, que era de obter uma amplitude constante e uma energia mecânica que se conserva a cada meio período.

Figura 10: Gráfico da posição angular em função do tempo.

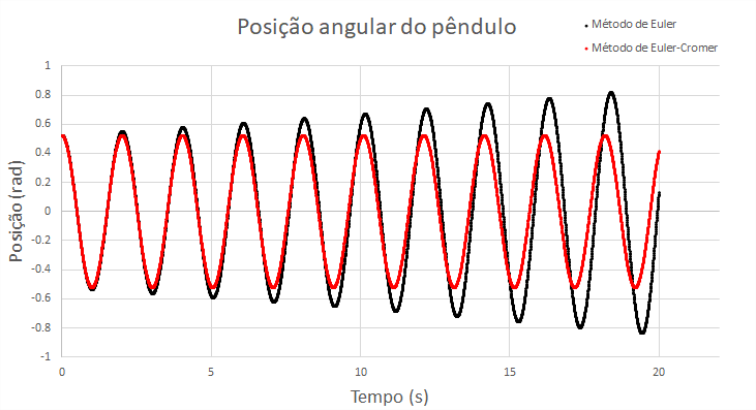
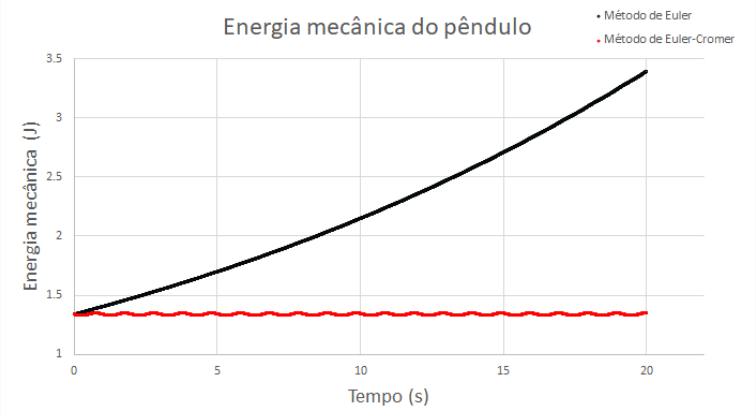


Figura 11: Gráfico da energia em função do tempo.

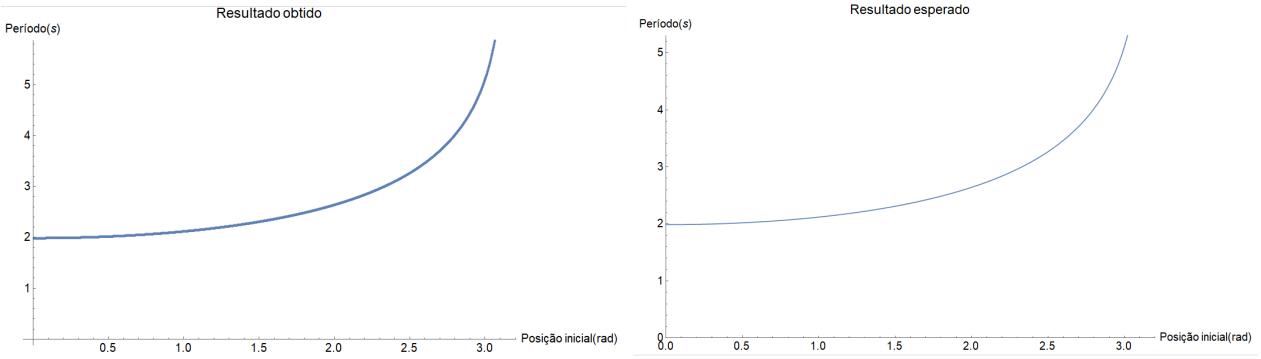


Dessa forma, podemos concluir que para esse problema o método de Euler não é eficiente, perdendo, de maneira rápida, sua capacidade de descrever o problema físico. Precisamos do método de Euler-Cromer para que os resultados sejam coerentes com a expectativa teórica que pressupõe conservação de energia mecânica.

3.3.2 Período do movimento pendular

É possível analisar como o período de movimento do pêndulo varia em função do ângulo θ_0. Para isso, executou-se o programa criado para esse fim, com as mesmas entradas dos itens anteriores, porém sem a posição angular inicial e com uma partição do intervalo de tempo de 10^{-4} s. O arquivo de saída, que continha dados do período de oscilação para cada posição inicial no intervalo $[0, \frac{\pi}{2}]$ permitiu a plotagem de um gráfico relacionando essas duas grandezas. Esse gráfico encontra-se sob rótulo de figura 12.

Figura 12: Gráficos comparando resultado obtido com resultado esperado.



(a) Resultado obtido pelo método de Euler-Cromer

(b) Resultado obtido pela expressão 21

Nesse gráfico, é possível perceber que, para ângulos pequenos, o período de oscilação pouco varia, sendo praticamente equivalente ao período do movimento harmônico correspondente. No entanto, quando empregamos ângulos maiores, o período de oscilação começa a aumentar e se afastar do período harmônico, que vale $T_0 = 1.98692$ s. O resultado é consistente com aquele obtido por meio da expressão 21.

Cabe uma discussão sobre a importância do tamanho da partição do intervalo de tempo. Quando o programa foi executado com uma partição maior,observou-se uma curva similar à da figura 12a, mas com "degraus"sutis que representam saltos repentinos da função. Quanto menor o valor de dt, mais sutis são os saltos, de maneira que, quando pequenos, passa a impressão que a função é uma curva suave.

Isso ocorre porque a variação no período é muito pequena. Como a partição do intervalo de tempo limita as casas decimais do período calculado, quanto menor a partição, maior a precisão do período, que fica sensível à alterações menores. Isso permite que a função dê saltos menores, ficando praticamente contínua.

4 Código dos programas

4.1 Efeito resistivo do ar em bicicletas

4.1.1 Sem efeito resistivo

```
1         program semresistencia
2
3         implicit none
4         integer i,n,a,b
5         real*8 m,P,deltat,dt,v0,vi,t,vi_ext,somasimp1,somasimp2,Isim
```

```
6           real*8, dimension(:), allocatable :: lista
7
8           open(20,file='semresistencia.in')
9           read(20,*)m,P,deltat,dt,v0
10          close(20)
11
12          n=(deltat/dt)+2
13          allocate (lista(n))
14          i=1
15          vi=v0
16          vi_ext=v0
17          t=0
18
19          open(10,file='semresistencia.dat')
20          do while (t<=deltat)
21
22          lista(i)=vi
23          vi=vi+(P*dt/(m*vi))
24          vi_ext=Sqrt((v0**2.0d0)+(2.0d0*P*t/m))
25          t=t+dt
26          i=i+1
27
28          end do
29
30          write(10,*)"A velocidade apos o tempo de 300s e:",lista(n)
31
32          a=1
33          b=deltat
34          somasimp1=0
35          somasimp2=0
36
37          do i=1,n-1,2
38          somasimp1=somasimp1+lista(a+i)
39          end do
40          somasimp1=somasimp1*(4.0d0*dt/3.0d0)
41
42          do i=2,n-1,2
43          somasimp2=somasimp2+lista(a+i)
44          end do
45          somasimp2=somasimp2*(2.0d0*dt/3.0d0)
46
47          Isim=(dt/3.0d0)*(lista(a)+lista(n))+somasimp1+somasimp2
48
49          write(10,*)"A distancia apos o tempo de 300s e:",Isim
50
51          close(10)
52          end program
```

4.1.2 Com efeito resistivo do ar

```
1           program comresistencia
2
3           implicit none
4           integer i,n,a,b
5           real*8 m,P,deltat,dt,v0,ro,s,vi,t,vi_ext,vt,eps,somasimp1,somasimp2,Isim
```

```
6           real*8, dimension(:), allocatable :: lista
7
8           open(20,file='comresistencia.in')
9           read(20,*)m,P,deltat,dt,v0,ro,s,eps
10          close(20)
11
12          n=(deltat/dt)+2
13          allocate (lista(n))
14          i=1
15          vi=v0
16          vi_ext=v0
17          t=0
18          vt=(2.0d0*P/(ro*s))**(1.0d0/3.0d0)
19
20          open(10,file='comresistencia.dat')
21          do while (abs(vi-vt)>eps)
22
23          vi=vi+(P*dt/(m*vi))-(ro*s*(vi**2.0d0)*dt/(2.0d0*m))
24          lista(i)=vi
25          t=t+dt
26          i=i+1
27
28          end do
29
30          write(10,*)"A velocidade terminal e:", vt, "m/s"
31          write(10,*)"O ciclista atinge a velocidade terminal com precisao eps em:",t,
                "s"
32
33          do while (t<=deltat)
34
35          vi=vi+(P*dt/(m*vi))-(ro*s*(vi**2.0d0)*dt/(2.0d0*m))
36          lista(i)=vi
37          t=t+dt
38          i=i+1
39
40          end do
41
42          a=1
43          b=deltat
44          somasimp1=0
45          somasimp2=0
46
47          do i=1,n-1,2
48          somasimp1=somasimp1+lista(a+i)
49          end do
50          somasimp1=somasimp1*(4.0d0*dt/3.0d0)
51
52          do i=2,n-1,2
53          somasimp2=somasimp2+lista(a+i)
54          end do
55          somasimp2=somasimp2*(2.0d0*dt/3.0d0)
56
57          Isim=(dt/3.0d0)*(lista(a)+lista(n))+somasimp1+somasimp2
58
59          write(10,*)"A distancia apos o tempo de 300s e:",Isim, "m"
```

```
60
61          close(10)
62          end program
```

4.2 Lançamento de projéteis

4.2.1 Sem efeito resistivo

```
1           program projeteissr
2
3           implicit none
4           real*8 x0,y0,v0,vx0,vy0,xi,yi,vxi,vyi,t,theta,g,dt
5
6           open(10,file='projeteissr.in')
7
8           read(10,*)v0,theta,g,dt
9
10          close(10)
11
12          theta=theta*Acos(-1.0d0)/180.0d0
13          x0=0.0d0
14          y0=0.0d0
15          vx0=v0*Cos(theta)
16          vy0=v0*Sin(theta)
17
18          xi=x0+vx0*dt
19          yi=y0+vy0*dt
20          vxi=vx0
21          vyi=vy0-g*dt
22
23          t=0.0d0
24          open(20,file='projeteissr.dat')
25
26          do while (yi>=0)
27
28          write(20,*)xi,yi,vxi,vyi
29
30          xi=xi+vxi*dt
31          yi=yi+vyi*dt
32          vyi=vyi-g*dt
33
34
35          end do
36
37          end program
```

4.2.2 Com efeito resistivo

```
1           program projeteiscr
2
3           implicit none
4           real*8 x0,y0,v0,vx0,vy0,xi,yi,vxi,vyi,t,theta,g,dt,r,alfa,temp,b,aux
5
6           open(10,file='projeteiscr.in')
7
8           read(10,*)v0,theta,g,dt,r,alfa,temp,b
```

```
9
10          close(10)
11
12          theta=theta*Acos(-1.0d0)/180.0d0
13          x0=0.0d0
14          y0=0.0d0
15          vx0=v0*Cos(theta)
16          vy0=v0*Sin(theta)
17
18          xi=x0+vx0*dt
19          yi=y0+vy0*dt
20          vxi=vx0-r*Sqrt(vx0**2+vy0**2)*vx0*dt
21          vyi=vy0-g*dt-r*Sqrt(vx0**2+vy0**2)*vy0*dt
22          t=dt
23
24          open(20,file='projeteiscr.dat')
25
26          do while (yi>=0)
27
28          t=t+dt
29          yi=yi+vyi*dt
30          xi=xi+vxi*dt
31          aux=vxi
32          vxi=vxi-r*(1.0d0-b*yi/temp)**alfa*Sqrt(vxi**2+vyi**2)*vxi*dt
33          vyi=vyi-g*dt-r*(1.0d0-b*yi/temp)**alfa*Sqrt(aux**2+vyi**2)*vyi*dt
34
35          write(20,*)xi,yi
36
37          end do
38
39          close(20)
40          end program
```

Para a obtenção dos valores do alcance máximo, ângulo que o proporciona e tempo de voo, utilizou-se o código seguinte, que se assemelha com o anterior, mas possui adaptações:

```
1           program projeteiscr
2
3           implicit none
4           real*8 x0,y0,v0,vx0,vy0,xi,yi,vxi,vyi,t,theta,g,dt,r,alfa,temp,b,aux
5
6           open(10,file='projeteiscr.in')
7
8           read(10,*)v0,theta,g,dt,r,alfa,temp,b
9
10          close(10)
11
12          theta=1.0d-3
13
14          do while (theta<=Acos(-1.0d0)/2.0d0)
15
16          x0=0.0d0
17          y0=0.0d0
18          vx0=v0*Cos(theta)
19          vy0=v0*Sin(theta)
20
```

```
21          open(20,file='projeteiscr.dat')
22
23          xi=x0+vx0*dt
24          yi=y0+vy0*dt
25          vxi=vx0-r*Sqrt(vx0**2+vy0**2)*vx0*dt
26          vyi=vy0-g*dt-r*Sqrt(vx0**2+vy0**2)*vy0*dt
27          t=dt
28
29          do while (yi>=0)
30
31          t=t+dt
32          yi=yi+vyi*dt
33          xi=xi+vxi*dt
34          aux=vxi
35          vxi=vxi-r*(1.0d0-b*yi/temp)**alfa*Sqrt(vxi**2+vyi**2)*vxi*dt
36          vyi=vyi-g*dt-r*(1.0d0-b*yi/temp)**alfa*Sqrt(aux**2+vyi**2)*vyi*dt
37
38          end do
39
40          write(20,*)theta,xi,t
41
42          theta=theta+1.0d-3
43
44          end do
45
46          close(20)
47          end program
```

4.3 Pêndulo Simples

4.3.1 O método de Euler

```
1           program euler
2
3           implicit none
4           real*8 m,l,g,thetai,deltat,dt,t,wi,aux,Ei
5
6           open(10,file='euler.in')
7           read(10,*)m,l,g,thetai,deltat,dt
8           close(10)
9
10          t=0.0d0
11          thetai=thetai*Acos(-1.0d0)/1.8d2
12          wi=0.0d0
13          Ei=((m*(l**2.0d0)*(wi**2))/2.0d0)+m*g*l*(1.0d0-Cos(thetai))
14
15          open(20,file='euler.dat')
16
17          do while(t<=deltat)
18
19          aux=wi
20          wi=wi-(g/l)*Sin(thetai)*dt
21          thetai=thetai+aux*dt
22          Ei=((m*(l**2.0d0)*(wi**2))/2.0d0)+m*g*l*(1.0d0-Cos(thetai))
23          t=t+dt
24
```

```
25          write(20,*)t,thetai,Ei
26
27          end do
28
29          close(20)
30
31          end program
```

4.3.2 O método de Euler-Cromer

```
1           program cromer
2
3           implicit none
4           real*8 m,l,g,thetai,deltat,dt,t,wi,Ei
5
6           open(10,file='euler.in')
7           read(10,*)m,l,g,thetai,deltat,dt
8           close(10)
9
10          t=0.0d0
11          thetai=thetai*Acos(-1.0d0)/1.8d2
12          wi=0.0d0
13          Ei=((m*(l**2.0d0)*(wi**2))/2.0d0)+m*g*l*(1.0d0-Cos(thetai))
14
15          open(20,file='cromer.dat')
16
17          do while(t<=deltat)
18
19          wi=wi-(g/l)*Sin(thetai)*dt
20          thetai=thetai+wi*dt
21          Ei=((m*(l**2.0d0)*(wi**2))/2.0d0)+m*g*l*(1.0d0-Cos(thetai))
22          t=t+dt
23
24          write(20,*)t,thetai,Ei
25
26          end do
27
28          close(20)
29
30          end program
```

4.3.3 Período do movimento pendular

```
1           program periodo
2
3           implicit none
4           real*8 m,l,g,theta0,thetai,deltat,dt,t,wi,Ei
5
6           open(10,file='euler.in')
7           read(10,*)m,l,g,deltat,dt
8           close(10)
9
10          theta0=1.0d-3
11
12          do while(theta0<Acos(-1.0d0))
13
```

```
14          t=0
15          thetai=theta0
16          wi=0.0d0
17          Ei=m*g*l*(1.0d0-Cos(theta0))
18
19          open(20,file='periodo.dat')
20
21          do while(wi<=0)
22
23          t=t+dt
24          wi=wi-(g/l)*Sin(thetai)*dt
25          thetai=thetai+wi*dt
26          Ei=((m*(l**2.0d0)*(wi**2))/2.0d0)+m*g*l*(1.0d0-Cos(thetai))
27
28          end do
29
30          write(20,*)theta0,2.0d0*t
31
32          theta0=theta0+1.0d-3
33
34          end do
35
36          close(20)
37
38          end program
```