

MAC2166 Introdução à Computação para Engenharia

Escola Politécnica – Primeiro Semestre de 2010

Terceiro Exercício-Programa (EP3)
Data máxima para entrega: 15/05/2010
versão 0.5

18 de janeiro de 2011

Jim Lovell: We just put Sir Isaac Newton in the driver's seat.

Apollo 13

1 Introdução



O programa espacial norte americano [Apollo](#) durou de 1961 a 1975 e seu principal objetivo foi levar humanos a pousar pela primeira vez na Lua. O programa foi concebido durante o governo do Presidente Dwight D. Eisenhower e conduzido pela [National Aeronautics and Space Administration](#) (NASA). O programa ganhou um gigantesco impulso depois do discurso do Presidente John F. Kennedy em 25 de maio de 1961 em uma sessão especial para o Congresso dos EUA declarando como objetivo nacional pousar um homem na Lua até o final da década.

Cada uma das naves das missões Apollo contava com uma medida de salvaguarda fazendo parte do plano de voo. Se por alguma razão os foguetes da nave deixassem de funcionar propriamente, a nave seguiria um caminho que havia sido planejado de tal maneira que fosse possível a nave ser “estilingada” (*slingshot*) em volta da Lua em uma trajetória livre de retorno (*free-return trajectory*), voltando à Terra utilizando apenas as forças gravitacionais da Lua e da Terra.

Em 1968 a missão [Apollo 8](#) foi a primeira a escapar da ação gravitacional da Terra, a primeira a ser capturada e escapar da ação gravitacional de outro corpo celeste e a primeira a retornar à Terra vinda de outro corpo celeste. Este outro corpo celeste era a Lua. Na missão Apollo 8 o caminho da nave foi levemente ajustado por três propulsões curtas que gastaram bem pouco combustível. A primeira propulsão da Apollo 8 colocou-a em órbita elíptica em torno da Lua e a segunda colocou-a em órbita circular. Apollo 8 realizou 10 voltas ao redor da Lua e depois realizou uma terceira propulsão que colocou-a no caminho de volta à Terra (*trans earth injection*). Essas propulsões e a trajetória de voo foram planejadas com extrema precisão bem antes da Apollo 8 deixar a Terra.

[Apollo 13](#) foi lançada em 1970. Esperava-se que a sua tripulação fosse a quinta a orbitar e a terceira a pousar na Lua. Após 56 horas do início da missão, uma explosão em um dos tanques de oxigênio desapontou a tripulação pois estes, imediatamente, sabiam que o pouso na Lua teria que ser cancelado. Em poucos minutos, o desapontamento transformou-se em medo já que eles perceberam que as suas vidas estavam em sério risco. A opção de mudar a trajetória dando meia volta e retornando a Terra foi imediatamente descartada pela NASA pois acreditava-se que a integridade dos motores da nave poderia ter sido comprometida devido à explosão do tanque de oxigênio. Eles não tiveram, portanto, outra alternativa além de continuar o vôo até a Lua e voltar a Terra através da trajetória de retorno livre. Devido ao oxigênio que escapou, a trajetória de vôo teve que ser levemente corrigida durante a viagem. Essencialmente, o efeito estilingue ao redor da Lua funcionou como planejado. A sobrevivência da tripulação da Apollo 13 não teria sido possível se não fosse pela medida de salvaguarda do plano de vôo.

A tarefa deste exercício programa¹ será simular e investigar a (versão bidimensional) da trajetória de retorno livre de uma nave sob a ação gravitacional da Terra e da Lua. Essa trajetória é capaz de enviar uma nave à Lua e retorná-la em segurança à Terra sem utilizar os motores depois da propulsão inicial (*trans lunar injection*).

Atenção, neste exercício-programa utilizaremos quilômetros (km) para medir distâncias, quilogramas (kg) para medir massa e horas (hr) para medir tempo.

2 Leis de Newton do movimento



A lei de força mais antiga conhecida é a *lei da gravitação universal* que Isaac Newton² publicou no seu *Principia Mathematica* [2]. Esta lei exprime as forças de interação entre dois corpos, um corpo de massa m_1 em um ponto P_1 e um corpo de massa m_2 em um ponto P_2 , cujo deslocamento relativo é $\vec{r}_{2,1} = \vec{P}_2\vec{P}_1$. Ela diz que:

$$\vec{F}_{2,1} = \frac{Gm_1m_2}{r_{2,1}^2}\hat{r}_{2,1} = -\vec{F}_{1,2} \quad (1)$$

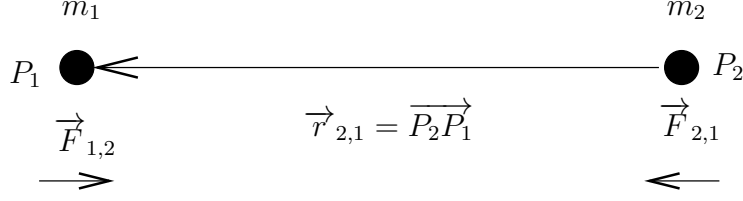
na qual $r_{2,1}$ é a distância entre os centros de massa dos corpos e $\hat{r}_{2,1} = \vec{r}_{2,1}/r_{2,1}$ é o vetor unitário da direção que vai do corpo 2 ao corpo 1. Em palavras, (1) diz que a *magnitude* da força gravitacional é proporcional ao produto das massa dos corpos e inversamente proporcional ao quadrado da distância que os separa. A força está dirigida ao longo da reta que passa pelos centros de massa dos dois corpos e é *atrativa* se a força $\vec{F}_{2,1}$ exercida pelo corpo 1 sobre o corpo 2 está dirigida para o corpo 1, em sentido $\vec{r}_{2,1}$.

A constante proporcional G que aparece em (1) é uma constante universal, ou seja, é a mesma para quaisquer corpos. Essa constante é chamada *constante gravitacional*. Seu valor é

$$G \approx 8.65 \times 10^{-13} \text{ km}^3 \text{ kg}^{-1} \text{ hr}^{-2} \quad (2)$$

¹Este EP é baseado em uma tarefa da *The North Carolina School of Science & Mathematics*.

²As leis de Newton do movimento é um dos tópicos da disciplina Física I.



Devido a segunda lei de Newton da mecânica clássica [3], sabemos que

$$\Sigma \vec{F} = m \vec{a} \quad (3)$$

na qual $\Sigma \vec{F}$ é a força resultante, m é a massa do corpo e \vec{a} é o vetor aceleração do corpo. A aceleração possui a mesma direção e o mesmo sentido da força.

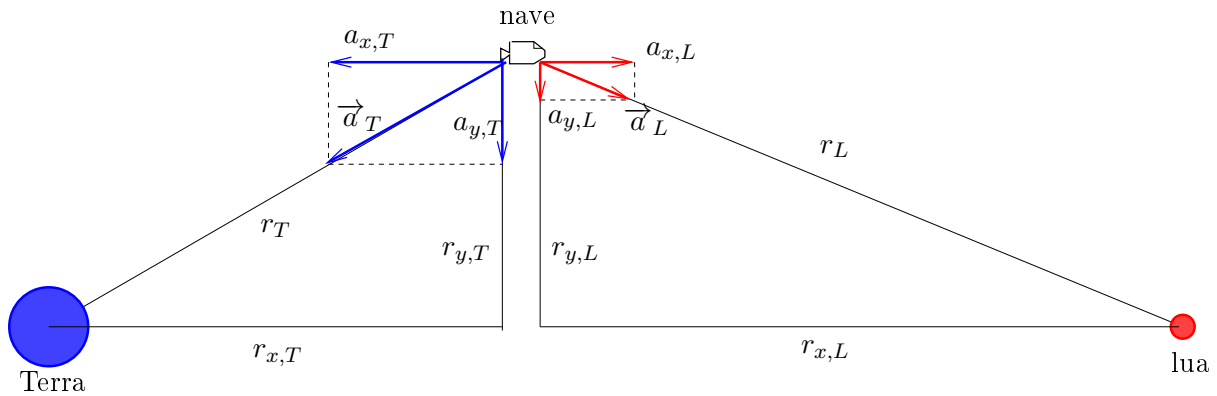
Considerando o corpo no ponto P_1 de (1) como sendo a Terra e o corpo no ponto P_2 de (1) bem como o corpo de (3) como sendo uma nave, digamos a Apollo 8, e combinando (1) e (3) obtemos que

$$\vec{a}_T = \frac{Gm_T}{r_T^2} \hat{r}_T \quad (4)$$

na qual \vec{a}_T é o vetor aceleração da força gravitacional exercida pela Terra sobre a nave, m_T é a massa da Terra, r_T é a distância da nave ao centro da Terra e \hat{r}_T é o vetor unitário da direção que vai da nave à Terra. De maneira similar, temos que

$$\vec{a}_L = \frac{Gm_L}{r_L^2} \hat{r}_L \quad (5)$$

na qual \vec{a}_L é o vetor aceleração da força gravitacional exercida pela Lua sobre a nave, m_L é a massa da Lua, r_L é a distância da nave ao centro da Lua e \hat{r}_L é o vetor unitário da direção que vai da nave à Lua.



Das equações (4) e (5) concluímos que a aceleração da força resultante é

$$\begin{aligned} \vec{a} &= \vec{a}_T + \vec{a}_L \\ &= \frac{Gm_T}{r_T^2} \hat{r}_T + \frac{Gm_L}{r_L^2} \hat{r}_L . \end{aligned} \quad (6)$$

Suponha agora que desejemos determinar para cada instante t o vetor posição $\vec{r}(t) = (x(t), y(t))$ de uma nave que está sob a atração gravitacional da Terra e da Lua. Sabemos que o vetor aceleração instantânea é

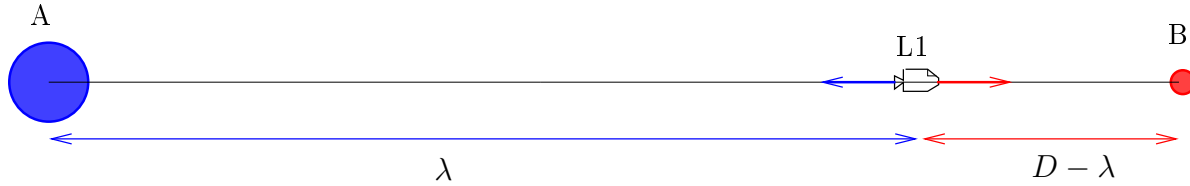
$$\vec{a} = \frac{d\vec{v}}{dt} = \frac{d^2\vec{r}}{dt^2}$$

e que seus componentes (a_x, a_y) são tais que

$$a_x = \frac{dv_x}{dt} = \frac{d^2x}{dt^2} \quad \text{e} \quad a_y = \frac{dv_y}{dt} = \frac{d^2y}{dt^2} . \quad (7)$$

As equações de Newton (7) reduzem o problema de determinar o vetor posição $\vec{r}(t) = (x(t), y(t))$ para subproblemas unidimensionais. Na próxima seção mostramos como as equações de Newton podem ser usadas para determinarmos computacionalmente uma aproximação de $x(t)$ e, similarmente, de $y(t), v_x(t)$ e $v_y(t)$.

Sejam A e B dois corpos celestes. O ponto na linha que liga os centros de massas dos dois corpos onde a aceleração gravitacional resultante das forças gravitacionais de A e B é nula é chamado de [ponto de Lagrange \$L_1\$](#) .



Suponha que o corpo A está na posição de coordenadas cartesianas (x_a, y_a) e tem massa m_A e que o corpo B está na posição de coordenadas cartesianas (x_b, y_b) e tem massa m_B . Se D é a distância entre os centros de massa de A e B , então a distância λ do ponto de Lagrange L_1 ao corpo A é solução da equação

$$\frac{G m_A}{\lambda^2} = \frac{G m_B}{(D - \lambda)^2} .$$

Em outras palavras, λ é raiz da equação

$$(m_A - m_B) \lambda^2 - 2m_A D \lambda + m_A D^2 = 0 . \quad (8)$$

Note que se $m_A = m_B$, então a equação acima é de 1º grau. Se λ é a raiz de (8), $0 < \lambda < D$, então o ponto de Lagrange L_1 em relação aos corpos A e B é o ponto (x, y) onde

$$x = x_a + \lambda(x_b - x_a)/D, \quad \text{e} \quad (9)$$

$$y = y_a + \lambda(y_b - y_a)/D . \quad (10)$$

O cálculo do ponto de Lagrange L_1 em relação a dois corpos será uma parte deste exercício programa.

Finalmente, seja A um corpo de massa m_A e considere um ponto a distância r do centro de massa de A . A [velocidade de escape](#) de A no ponto é a menor velocidade v_e a

partir da qual um objeto ou nave nesse ponto e com velocidade inicial v_e pode viajar sem nunca voltar ao corpo A devido a sua força gravitacional. Por exemplo, a velocidade de escape da Terra na sua superfície é aproximadamente de 40320 km/hr, cerca de 34 vezes a velocidade do som.

Essa velocidade v_e pode ser deduzida a partir da lei de conservação de energia e seu valor é

$$v_e = \sqrt{\frac{2 G m_A}{r}}, \quad (11)$$

onde G é a constante gravitacional.

3 Método de Euler para as equações de Newton



Consideremos a equação da segunda lei de Newton na sua versão unidimensional $F_x = m a_x$ para um objeto de massa m . No nosso caso a força F_x é a gravitacional e pela lei da gravitação universal é função da posição x , de modo que a equação diferencial para a velocidade se escreve:

$$a_x = \frac{dv_x}{dt}.$$

Para obter a posição, recorreremos à relação entre posição e velocidade,

$$v_x = \frac{dx}{dt}.$$

Se conhecermos a posição e a velocidade do objeto num certo instante, estas equações nos permitem determinar $v_x(t)$ e $x(t)$ em qualquer instante t . No entanto, no caso geral, não podemos resolver estas equações analiticamente. Isso significa que temos que recorrer ao computador através de um método numérico. Esse método deverá ser capaz de estimar a velocidade e posição do objeto num certo instante, dadas, por exemplo, a posição e velocidade x_0 e v_0 em um instante inicial t_0 .

O método numérico mais simples para resolver equações diferenciais consiste em substituir as derivadas por razões entre variações da função e da respectiva variável. No caso presente temos que

$$\begin{aligned} \frac{dv_x}{dt} \rightarrow \frac{\Delta v_x}{\Delta t} &\Rightarrow \Delta v_x \approx a_x \Delta t \\ \frac{dx}{dt} \rightarrow \frac{\Delta x}{\Delta t} &\Rightarrow \Delta x \approx v_x \Delta t \end{aligned}$$

A ideia do Δt é substituir a variável contínua t por uma variável *discreta*; ou seja, fazemos o t “andar de soquinho”. Assim, dado um *instante* t , o *instante seguinte* é $t + \Delta t$.

Agora, dados a posição x , a velocidade v e a aceleração a num instante t , podemos calcular seus valores x' , v' , a' no instante seguinte $t + \Delta t$ pelas fórmulas:

$$x' = x + v\Delta t, \quad (12)$$

$$v' = v + a\Delta t. \quad (13)$$

Para fechar as equações, lembre que a aceleração no instante t é dada por

$$a = \frac{Gm_X}{x^2}$$

onde m_X representa a massa da Terra ou da Lua.

As equações (12) e (13) constituem o *método de Euler*³ para o cálculo das posições e velocidades de um corpo sujeito a uma força.

Curiosidade. O método de Euler é dito um método de 1ª ordem. A razão para isto encontra-se no fato de podermos escrever, usando o desenvolvimento em série de Taylor⁴:

$$v' = v + a\Delta t + \frac{da}{dt} \frac{(\Delta t)^2}{2} + \dots$$

$$x' = x + v\Delta t + \frac{dv}{dt} \frac{(\Delta t)^2}{2} + \dots$$

o que significa que, ao aplicar o método de Euler, estamos desprezando termos como potências de Δt iguais ou superiores a 2 — dizemos “termos de 2ª ordem ou superior”.

Esta pequena explicação sobre o método de Euler para resolver as equações de Newton é devida a Pedro Vieira Alberto [1].

4 Tarefa



A primeira coisa que você deve fazer é copiar o arquivo `esqueleto.c` que está disponível na página do EP3. Tendo como base este esqueleto, você deverá escrever um programa em C cujo comportamento é orientado por uma lista de opções. Todas as opções bem como os dados para cada opção serão lidos de um arquivo. O formato desse arquivo é exemplificado na seção 4.2. As opções que serão tratadas pelo programa serão identificadas por caracteres, como veremos. Os números do tipo `float` impressos pelo programa devem conter apenas 2 casas após o ponto decimal como, por exemplo, 3.14 e 2.71.

No programa, as coordenadas cartesianas da Terra e da Lua serão fixas. As coordenadas da Terra serão (X_T, Y_T) e as coordenadas da Lua serão (X_L, Y_L) , onde

```
#define X_T 0
#define Y_T 0
#define X_L 384400
#define Y_L 0
```

³O método de Euler é um dos tópicos de disciplinas de Cálculo Numérico, onde também são considerados métodos mais precisos como o de Runge-Kutta.

⁴Série de Taylor é um dos tópicos da disciplina Cálculo II.

Outras definições que serão usadas pelo programa estão presentes no arquivo [esqueleto.c](#).

Passamos a listar essas opções e descrever o comportamento do programa para cada uma dessas opções:

Opção 'a': o programa deve ler, nessa ordem, a:

- a1) posição de um corpo celeste em coordenadas cartesianas,
- a2) sua massa, e
- a3) posição de uma nave em coordenadas cartesianas.

Nesta opção, o programa deve calcular e imprimir o vetor aceleração (A_X, A_Y) da força gravitacional exercida pelo corpo sobre a nave (essencialmente, equação (4)).

Opção 'A': o programa deve ler, nessa ordem, a:

- A1) posição de um corpo celeste 1 em coordenadas cartesianas;
- A2) massa do corpo celeste 1;
- A3) posição de um corpo celeste 2 em coordenadas cartesianas;
- A4) massa do corpo celeste 2; e
- A5) posição de uma nave em coordenadas cartesianas.

Nesta opção, o programa deve calcular e imprimir o:

1. vetor aceleração ($A1_X, A1_Y$) da força gravitacional exercida pelo corpo 1 sobre a nave (essencialmente, equação (4));
2. vetor aceleração ($A2_X, A2_Y$) da força gravitacional exercida pelo corpo 2 sobre a nave (essencialmente, equação (4));
3. vetor aceleração (A_X, A_Y) da força gravitacional resultante exercida pelos corpos 1 e 2 sobre a nave (essencialmente, equação (6)).

Opção 'e': o programa deve ler, nessa ordem, a:

- e1) posição de um corpo celeste em coordenadas cartesianas;
- e2) massa do corpo celeste;
- e3) posição de uma nave em coordenadas cartesianas.

Nesta opção, o programa deve imprimir a distância da nave ao corpo, e calcular e imprimir a velocidade de escape do corpo na posição da nave (equação (11)).

Opção 'L': o programa deve ler, nessa ordem, a:

- L1) posição de um corpo celeste 1 em coordenadas cartesianas;
- L2) massa do corpo celeste 1;
- L3) posição de um corpo celeste 2 em coordenadas cartesianas;
- L4) massa do corpo celeste 2;

Nesta opção, o programa deve calcular e imprimir as coordenadas cartesianas do ponto Lagrangeano, ou seja, as coordenadas cartesianas (X, Y) do ponto cujo vetor aceleração (A_X, A_Y) da força gravitacional resultante exercida pelos corpos 1 e 2 é nulo (equações (9) e (10)).

Opção 's': o programa deve ler, nessa ordem, a:

- z1) posição inicial de uma nave em coordenadas cartesianas;
- z2) as componentes (V_X, V_Y) do vetor velocidade inicial da nave;
- z3) tempo máximo de simulação;
- z4) o intervalo Δt entre um instante e o instante seguinte da simulação.

Nesta opção, o programa deve calcular a trajetória da nave, sujeita às forças gravitacionais da Terra e da Lua, começando na posição inicial com velocidade (V_X, V_Y). O instante inicial da simulação é 0 e o instante seguinte a um dado instante t é o instante $t + \Delta t$. A simulação da trajetória deverá terminar quando o tempo de simulação ultrapassar o tempo máximo de simulação ou quando a nave chegar na Terra ou na Lua. Um mapa com as posições da Terra, Lua e nave deve ser mostrado na tela no início e durante a simulação. Durante a simulação, o mapa deve ser mostrado somente quando a nave muda de quadrante, conforme explicação na seção 4.1. Juntamente ao mapa, o programa deve mostrar:

1. tempo de viagem;
2. distancia percorrida pela nave;
3. posicao atual da nave em coordenadas cartesianas;
4. vetor velocidade atual da nave;
5. vetor aceleracao atual da nave;

6. distancia da nave à Terra;

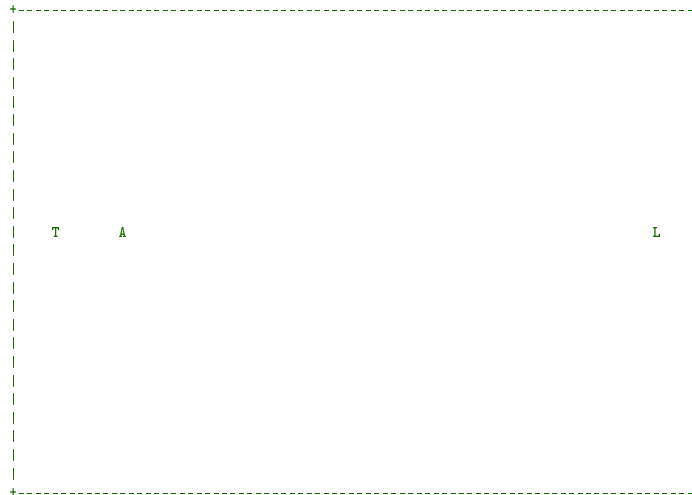
7. distancia da nave à Lua.

O programa deve também imprimir, ao final da simulação, uma mensagem indicando se a nave chegou na Terra, chegou na Lua, ou se a nave ainda se encontra no espaço.

4.1 Impressão do mapa

Um mapa espacial que mostra a Terra, a Lua e a nave é útil para visualizar a posição da nave relativamente às posições da Terra e da Lua. A impressão desses mapas espaciais em instantes apropriados da simulação permite-nos “enxergar” a trajetória da nave.

Tal mapa será construído dividindo-se uma região retangular do plano cartesiano que contém a Terra e a Lua em quadrantes. Neste EP, fixamos o número de quadrantes em 25 no eixo y e 80 no eixo x do plano cartesiano. Um exemplo de tal mapa é exibido a seguir.



Levando em conta que a posição da Terra no plano cartesiano é fixa em $(X_T, Y_T) = (0, 0)$ e a da Lua em $(X_L, Y_L) = (384400, 0)$, consideraremos que cada quadrante corresponde a uma região de $LARGURA = 5414 \text{ km}^2$ e a região retangular do espaço a ser mapeado é o conjunto de pontos de coordenadas cartesianas (x, y) tais que:

$$\begin{aligned} -4.5 * LARGURA &\leq x \leq 75.5 * LARGURA \\ -12.5 * LARGURA &\leq y \leq 12.5 * LARGURA \end{aligned}$$

Desta forma, um ponto do espaço de coordenadas cartesianas (x, y) está no quadrante (i, j) , onde i e j podem ser calculados da seguinte forma

$$\begin{aligned} i &= (y + 13.5 * LARGURA) / LARGURA; \\ j &= (x + 5.5 * LARGURA) / LARGURA; \end{aligned}$$

Logo, o ponto de coordenadas cartesianas $(0, 0)$ da Terra está no quadrante $(13, 5)$, enquanto que o ponto $(X_L, Y_L) = (384400, 0)$ da Lua está no quadrante $(13, 76)$.

4.2 Arquivo de entrada e saída do programa

O arquivo de entrada conterá uma lista de opções a serem tratadas pelo programa. A seguir, um exemplo de um arquivo de entrada no qual cada opção, seguido pelos respectivos dados conforme especificado anteriormente, aparece uma vez.

```
a
0      0      5.97e+24
6563   0
```

```
A
0      0      8e+22
100000 0      5e+21
80000  0
```

```
e
0      0      5.97e+24
6378   0
```

```
L
0      0      8e+22
100000 0      5e+21
```

```
s
6563   0      39300   0
0.1    0.001
```

A saída do programa quando executado usando como entrada o arquivo com o conteúdo acima é mostrada a seguir. Os dois mapas na saída foram reduzidos para caber na página. Os mapas produzidos pelo programa são maiores. Os números do tipo `float` impressos pelo programa devem conter apenas 2 casas após o ponto decimal como, por exemplo, 3.14 e 2.71.

Arquivo de entrada : entrada.txt

```
Opcao [a]
Posicao do corpo      : ( 0.00 , 0.00 )
Massa do corpo       : 5.97e+24
Posicao da nave       : ( 6563.00 , 0.00 )
Aceleracao gravitacional : ( -119890.73 , -0.00 )
```

Tecle <enter> para continuar.

```
Opcao [A]
Posicao do corpo 1 : ( 0.00 , 0.00 )
Massa do corpo 1  : 8.00e+22
```

```
Posicao do corpo 2 : ( 100000.00 , 0.00 )
Massa do corpo 2   : 5.00e+21
Posicao da nave     : ( 80000.00 , 0.00 )

Aceleracao gravitacional (corpo 1) : ( -10.81 , -0.00 )
Aceleracao gravitacional (corpo 2) : ( 10.81 , -0.00 )
Aceleracao gravitacional resultante: ( 0.00 , -0.00 )
```

Tecle <enter> para continuar.

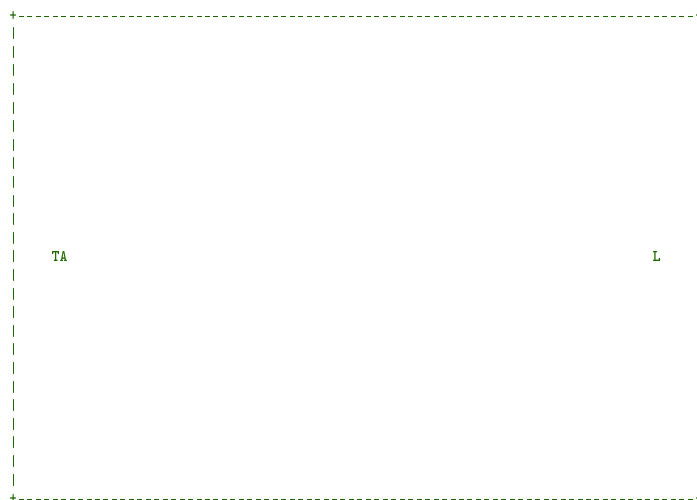
```
Opcao [e]
Posicao do corpo           : ( 0.00 , 0.00 )
Massa do corpo            : 5.97e+24
Posicao da nave            : ( 6378.00 , 0.00 )
Distancia da nave ao corpo : 6378.00
Velocidade de escape      : 40240.93
```

Tecle <enter> para continuar.

```
Opcao [L]
Posicao do corpo 1 : ( 0.00 , 0.00 )
Posicao do corpo 2 : ( 100000.00 , 0.00 )
Massa do corpo 1  : 8.00e+22
Massa do corpo 2  : 5.00e+21
Ponto Lagrangeano : ( 80000.00 , 0.00 )
```

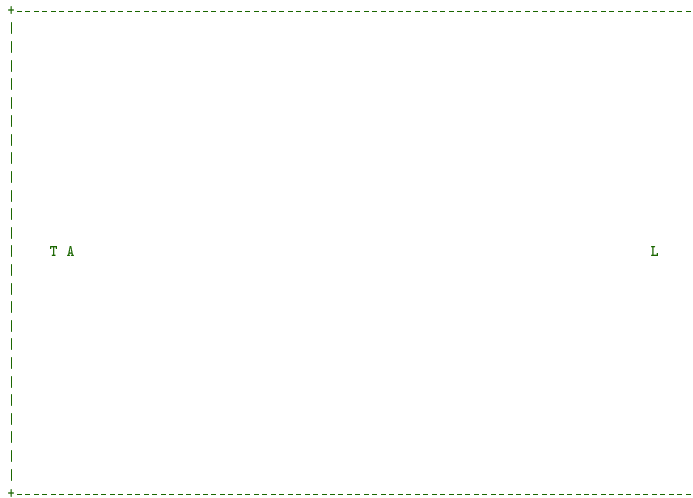
Tecle <enter> para continuar.

Opcao [s]



Tempo de viagem : 0.00
Distancia percorrida : 0.00
Posicao : (6563.00,0.00)
Velocidade : (39300.00,0.00)
Aceleracao : (-119890.29,-0.00)
Distancia da Terra : 6563.00
Distancia da Lua : 377837.00

Tecle <enter> para continuar.



Tempo de viagem : 0.04
Distancia percorrida : 1561.01
Posicao : (8124.01,0.00)
Velocidade : (35227.10,0.00)
Aceleracao : (-78243.26,-0.00)
Distancia da Terra : 8124.01
Distancia da Lua : 376275.99

Tecle <enter> para continuar.

Apos 0.10 horas de voo, a nave ainda esta em orbita!

Tecle <enter> para continuar.

Fim do processamento.

4.3 Funções a serem implementadas

O seu EP deverá implementar e utilizar, obrigatoriamente, as funções especificadas abaixo. O protótipo das funções (tipo de valor de retorno, nome da função, tipo e ordem dos parâmetros) **NÃO** deve ser alterado. Você pode escrever mais funções se achar conveniente e pode usar qualquer função da biblioteca `math.h` se julgar necessário.

f1) `float distancia(float x1, float y1, float x2, float y2) ;`

Recebe as coordenadas (x_1, y_1) e (x_2, y_2) de dois pontos. Retorna a distância entre os pontos.

f2) `float velocidade_esc(float xA, float yA,
float mA,
float x, float y) ;`

Recebe as coordenadas (x_A, y_A) e a massa m_A de um corpo A, e as coordenadas (x, y) de uma nave. Retorna o valor da velocidade de escape do corpo na posição da nave (equação (11)).

f3) `void aceleracao_gravitacional(float xA, float yA, float mA,
float x, float y,
float *ax, float *ay) ;`

Recebe as coordenadas (x_A, y_A) e a massa m_A de um corpo A, e as coordenadas (x, y) de uma nave. Devolve em $(*ax, *ay)$ o vetor aceleração da força gravitacional exercida pelo corpo A sobre a nave (essencialmente, equação (4)).

f4) `void aceleracao_resultante(float xA, float yA, float mA,
float xB, float yB, float mB,
float x, float y, float *ax, float *ay) ;`

Recebe as coordenadas (x_A, y_A) e a massa m_A de um corpo A, as coordenadas (x_B, y_B) e a massa m_B de outro corpo B, e as coordenadas (x, y) de uma nave. Devolve em $(*ax, *ay)$ o vetor aceleração da força gravitacional resultante exercida pelos corpos sobre a nave (essencialmente, equação (6)).

f5) `void ponto_L1(float xA, float yA, float mA,
float xB, float yB, float mB,
float *x, float *y) ;`

Recebe as coordenadas (x_A, y_A) e a massa m_A de um corpo A, as coordenadas (x_B, y_B) e a massa m_B de outro corpo B. Devolve em $(*x, *y)$ as coordenadas cartesianas do ponto Lagrangeano, ou seja, as coordenadas do ponto cujo vetor aceleração da força gravitacional resultante exercida pelos corpos A e B é nula (equações (9) e (10)).

f6) Funções a serem usadas para mostrar a trajetória da nave:

`void quadrante(float x, float y, int *i, int *j) ;`

Recebe as coordenadas (x,y) de uma nave. Devolve em (*i,*j) o quadrante correspondente a (x,y) no mapa (seção 4.1).

```
void mostre_dados(float hora, float distancia_percorrida,
                 float x, float y,
                 float vx, float vy,
                 float ax, float ay) ;
```

Recebe o tempo de simulação `hora`, a distância `distancia_percorrida` percorrida pela nave, as coordenadas (x,y) de uma nave, o vetor velocidade (vx,vy) e o vetor aceleração (ax, ay) da nave. Imprime todas essas informações além das distâncias da nave à Terra e à Lua.

4.4 Funções fornecidas

Além dessas funções, as seguintes funções que podem ser usadas no seu EP estão no arquivo `esqueleto.c` que foi fornecido.

F1) `int sao_iguais(float x_1, float x_2) ;`

Veja descrição na seção 5.

F2) `void espere_enter() ;`

Função cuja execução termina quando o usuário tecla **Enter**.

F3) `int mostre_mapa(int iT, int jT, int iL, int jL,
 float x, float y, int *iN, int *jN)`

Recebe o quadrante (iT,jT) da Terra no mapa, o quadrante (iL,jL) da Lua no mapa, as coordenadas atuais (x,y) da nave, e o quadrante (*iN, *jN) no mapa em que a nave estava no instante anterior da simulação. Se o quadrante em que a nave está atualmente for diferente do quadrante (*iN, *jN), então a função imprime o mapa na tela e devolve em (*iN, *jN) o novo quadrante em que está a nave. Retorna 1 se o mapa é impresso e 0 em caso contrário.

4.5 Constantes

A seguir estão algumas constantes que serão usadas pelo seu programa. Elas já fazem parte do arquivo `esquelo.c` que foi fornecido.

Constante	Notação	Valor
Massa da Terra	MTERRA	5.97×10^{24} kg
Massa de Lua	MLUA	7.35×10^{22} kg
Distância da Terra a Lua	D	384400 km
Constante gravitacional	G	8.65×10^{-13} km ³ kg ⁻¹ hr ⁻²
Raio da Terra	RTERRA	6378 km
Raio da Lua	RLUA	1738 km

5 Erro relativo e comparações entre números reais



Dado um número x e uma aproximação y para x , o *erro* (também chamado de *erro absoluto*) da aproximação y em relação a x é definido como sendo $|y - x|$. Quando a grandeza de x não é próxima da de 1, o erro absoluto pode não ser a maneira mais adequada de medir a qualidade da aproximação y . Por exemplo, os erros absolutos de 1.01 em relação a 1 e de 0.02 em relação a 0.01 são idênticos, mas é claro que a primeira aproximação é muito melhor que a segunda.

Face à limitada avaliação de uma aproximação conferida pelo erro absoluto, tenta-se definir o *erro relativo* de y em relação a x como sendo

$$\left| \frac{y - x}{x} \right|$$

Assim, nos dois exemplos anteriores, os erros relativos são respectivamente de 0.01 (ou 1%) e 1 (ou 100%). Contudo, esta definição ainda é incompleta quando $x = 0$. Nestes casos, a divisão por 0 não pode ser realizada e adotam-se valores arbitrários para o erro relativo. No caso de também ocorrer que $y = 0$, a aproximação certamente é perfeita e adota-se que o erro é 0. No caso de $y \neq 0$, a aproximação é certamente insatisfatória e adota-se o valor arbitrário 1 para o erro relativo. (Na prática, os erros relativos procurados são sempre menores que 1.)

No computador representamos aproximações de números reais, uma vez que estes são representados em uma quantidade finita de *bits*. Portanto, cálculos envolvendo números reais estão sujeitos aos erros de precisão. O valor de uma expressão aritmética que envolve várias operações com números reais pode variar ligeiramente, dependendo da ordem em que as operações são realizadas no computador.

Devido a essas imprecisão, nem sempre dois números que são esperados serem iguais são iguais. No seu programa, para verificar se dois números reais são iguais, utilize a função `sao_iguais` a seguir que devolve 1 se `x` e `y` são considerados iguais e 0 em caso contrário. Essa função já faz parte do arquivo [esqueleto.c](#) que foi fornecido.

```
#define EPS    0.00001

int sao_iguais(float x, float y) {

    return fabs(x-y) < EPS ||
           fabs(x-y) < EPS*fabs(x) || fabs(x-y) < EPS*fabs(y) ;

}
```

6 Executáveis, consistência e entrega do EP

Executáveis deste exercício programa podem ser encontrados em www.ime.usp.br/~mac2166/ep3/executaveis. Caso você tenha dúvidas sobre qual deve ser o comportamento do seu programa em alguma situação, veja como se comporta este executável.



O seu programa não precisa fazer consistência de dados. O programa pode supor que todos os dados lidos (opções, número, ...) estão corretos.

Todo exercício programa deve seguir as observações contidas na página www.ime.usp.br/~mac2166/infoeps, onde estão descritas as diretrizes para forma de entrega do exercício, aspectos importantes na avaliação etc.

Referências

- [1] Pedro Vieira Alberto, *Alguns métodos numéricos para resolver equações de Newton*, http://nautilus.fis.uc.pt/personal/pvalberto/apontamentos/metodos_ode.pdf, Fevereiro 2004.
- [2] Herch Moysés Nussenzveig, *Curso de Física Básica: 1-Mecânica*, Edgard Blücher, 1981.
- [3] Hugh D. Young e Roger A. Freedman, *Sears e Zemansky's Física I: Mecânica*, Addison Wesley, 2003.



7 Errata

Fórmulas (9) e (10) foram corrigidas em 19/ABR (Denis Marques).

Cálculos de i e j na página 9 foram corrigidos em 20/ABR (Adriano Scarso, Denis Marques, Ricardo Hahn).

Faltou um quadrado ($r_{2,1}^2$ em vez de $r_{2,1}$) na equação (1). Corrigido em 21/ABR (Ricardo Hahn).

A função `sao_iguais` não funcionava se um dos dois parâmetros fosse zero e o outro fosse “pequeno”. Foi acrescentada a condição `fabs(x-y) < EPS` no enunciado e no arquivo `esqueleto.c`. Corrigido em 03/MAI (Pedro Augusto Silva).