MAC0115 – Introdução à Computação – - IF Noturno - Bacharelado

EP3 - Problema dos três corpos

Exercício-Programa 3 (EP3)
Datas de entrega:
ep3-aquecimento.py: até 6/novembro/22 - 23h59 (domingo)
ep3.py: até 16/novembro/22 - 23h59 (quarta-feira)

Neste exercício-programa você deverá implementar um programa para simular o *Problema dos Três Corpos*, um problema clássico da Física, que consiste em descrever a trajetória de três corpos celestes que interagem por atração gravitacional. **Na aula do dia 18/outubro será dada uma explicação e dicas sobre este exercício-programa. Não falte!**

1 Simulação de um corpo

1.1 Atração gravitacional

Sejam B_1 e B_2 dois corpos de massas m_1 e m_2, ambos de forma esférica. Seja d a distância entre os centros de B_1 e B_2. A lei de gravitação de Newton diz que há entre B_1 e B_2 uma força de atração gravitacional de magnitude

$$G\,\frac{m_1 m_2}{d^2}, \tag{1}$$

onde G é a constante gravitacional universal, cujo valor é $6.67 \times 10^{-11}\,\mathrm{Nm}^2/\mathrm{kg}^2$.

Suponha agora que haja um terceiro corpo no sistema, digamos B_0, de massa m_0 e também de forma esférica. Há atração gravitacional entre B_0 e B_1 e entre B_0 e B_2, cujas magnitudes podem ser calculadas pela lei de Newton (1). Tendo esses três corpos B_0, B_1 e B_2, para determinar a força de atração gravitacional total que age sobre o corpo B_0, devemos calcular a soma (vetorial) das forças F_1 e F_2 que os corpos B_1 e B_2 exercem sobre B_0.

Observação. Neste exercício-programa trataremos de corpos com dimensões desprezíveis em relação às distâncias envolvidas. Assim, estes podem ser visto como pontos (no espaço em consideração).

1.2 Trajetória de um corpo

Neste exercício, para simplificar, trabalharemos no espaço bidimensional ($\mathbb{R}^2$). Assim, a posição de um corpo B em um instante t_0 é dado por um par $r = (r_x, r_y) \in \mathbb{R}^2$. Seja $v = (v_x, v_y) \in \mathbb{R}^2$ a velocidade de B. Considere que essa velocidade é constante por um intervalo de tempo Δt. Com isso, no instante $t_0 + \Delta t$, o corpo B estará na posição $r + (\Delta t)v$[1]. Agora, considere que B sofre uma aceleração $a = a(t)$ no instante t; então, a velocidade $v(t)$ de B no instante t varia ao longo do tempo e $v(t) = v(t_0) + \int_{t_0}^{t} a(t)\, dt$. Assim, a posição de B no instante $t_0 + \Delta t$ será $r + \int_{t_0}^{t_0+\Delta T} v(t)\, dt$.

Neste exercício (felizmente :-)), você deve ignorar as integrais que ocorrem acima, e deve adotar uma forma aproximada para determinar a posição do corpo B no instante $t_0 + \Delta t$, considerando que a velocidade é constante nesse intervalo. Vamos supor que conhecemos a aceleração $a(t_0)$ no instante t_0. O que fazemos é atualizar o valor atual v da velocidade para o valor $v' = v + (\Delta t)a(t_0)$ e declaramos que B estará na posição $r + (\Delta t)v'$ no instante $t_0 + \Delta t$.

Uma curiosidade interessante sobre o problema dos n-corpos com $n \geq 2$ é que não há uma solução analítica para o mesmo. Logo, soluções numéricas (como solicitadas nesse exercício-programa) são basicamente o melhor que se pode pedir. Quando Δt não é grande, a aproximação é boa. No caso, pode-se tornar Δt cada vez menor para obter um resultado mais refinado; contudo, o programa ficará cada vez mais lento.

1.3 Trajetória sob uma força gravitacional

Seja B o corpo mencionado na seção anterior, e seja m a sua massa. Suponha que B sofra uma atração gravitacional dada por uma força $F = (F_x, F_y) \in \mathbb{R}^2$, devido a outros corpos presentes no sistema. A segunda lei de Newton diz que B sofre então uma aceleração $a = (a_x, a_y) \in \mathbb{R}^2$ dada por $a = (1/m)F$. Note que, conforme B muda de posição, a força gravitacional F que ele sofre pode mudar, dado que F depende da posição relativa de B em relação aos outros corpos do sistema. De qualquer forma, conhecendo a posição r, a velocidade v de B e a força F no instante t_0, podemos usar o método descrito na Seção 1.2 para determinar (aproximadamente) a posição de B no instante $t_0+\Delta t$ (nesse método, ignoramos que F pode mudar entre t_0 e $t_0 + \Delta t$). Fazemos o seguinte:

$$a \leftarrow (1/m)F \tag{2}$$

$$v \leftarrow v + (\Delta t)a \tag{3}$$

$$r \leftarrow r + (\Delta t)v. \tag{4}$$

O valor de r calculado em (4) é a posição de B no instante $t_0 + \Delta t$ (aproximadamente).

Observação. Note que, em (2), (3) e (4), as quantidades a, F, v e r são vetores. Assim, por exemplo, para calcular (2), você precisa fazer $a_x \leftarrow F_x/m$ e $a_y \leftarrow F_y/m$, onde $a = (a_x, a_y)$ e $F = (F_x, F_y)$. O mesmo vale para (3) e (4).

1.4 Exemplo

Suponha que a Terra esteja na origem $(0, 0) \in \mathbb{R}^2$, e que ela não se mova. Suponha que no instante $t_0 = 0$ a Lua esteja na posição

$$(r_x^{(0)}, r_y^{(0)}) = (3.63 \times 10^8, 0) \tag{5}$$

e tenha velocidade

$$(v_x^{(0)}, v_y^{(0)}) = (0.0, 1072). \tag{6}$$

[1] note que isto é uma soma vetorial.

Note que, em (5) e (6), omitimos as unidades. Quando fazemos isso, supomos que a unidade adotada é a do sistema internacional (assim, a unidade em (5) é o metro e em (6) é m/s). Vamos adotar 7.342×10^{22} kg como a massa da Lua e 5.972×10^{24} kg como a massa da Terra.

Com as informações acima, usando (1), temos que

$$F_0 = (-2.21946 \times 10^{20}, 0) \tag{7}$$

é a força que age sobre a Lua no instante $t_0 = 0$ (os números são dados com no máximo 6 dígitos significativos). Executando os passos (2), (3) e (4) com $\Delta t = 3600\,\mathrm{s}$ (uma hora), vemos que a posição da Lua no instante $t_1 = 3600$ é

$$r(t_1) = r(3600) = (3.62961 \times 10^8, 3.8592 \times 10^6). \tag{8}$$

A força que age na Lua no instante $t_1 = 3600$ é

$$F_1 = (-2.21956 \times 10^{20}, -2.35996 \times 10^{18}); \tag{9}$$

e calculando (2), (3) e (4) com $\Delta t = 3600\,\mathrm{s}$ novamente, temos que a posição da Lua no instante $t_2 = 7200$ é

$$r(t_2) = r(7200) = (3.62882 \times 10^8, 7.71798 \times 10^6). \tag{10}$$

Podemos repetir o processo acima para obter a posição da Lua a cada hora. Repetindo o processo até o instante $T = 864000 = 10 \times 24 \times 3600$ (dez dias), obtemos as posições $r(3600)$, $r(2 \times 3600)$, $\ldots$, $r(240 \times 3600)$. O arquivo `lua_10.out` contém a posição original $(3.63 \times 10^8, 0)$ da Lua e as 240 posições acima. O mar de números no arquivo `lua.out` não é muito fácil de se "entender". Plotando os 241 pontos, obtemos a figura 1.

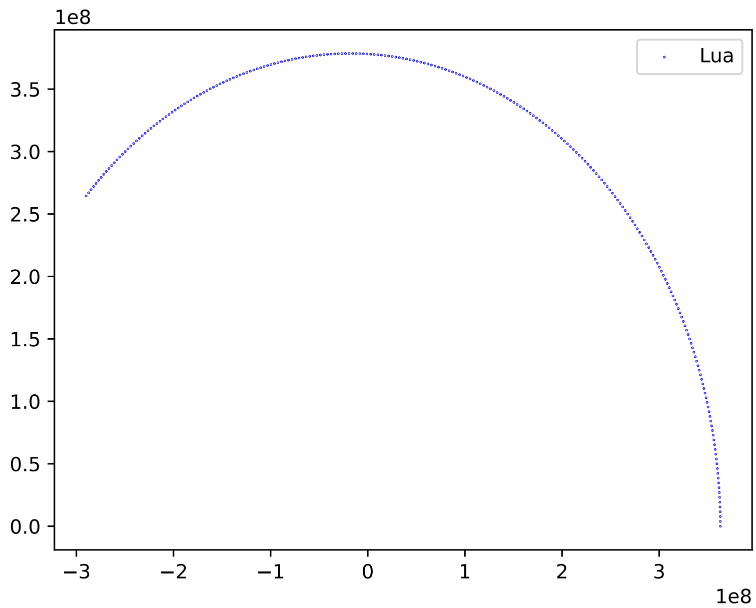

Figura 1: Órbita da Lua ao longo de 10 dias

Executando o processo acima para $T = 2332800$ (27 dias) e $T = 86400000$ (1000 dias), obtemos os arquivos `lua_27.out` (figura 2) e `lua_1000.out` (figura 3).

Se adotamos como velocidade inicial da Lua

$$(v_x^{(0)}, v_y^{(0)}) = (0.0, 1400), \tag{11}$$

e simulamos sua trajetória por 300 dias, obtemos o arquivo `lua14_300.out` (figura 4). Note como a órbita da Lua é diferente com (11). Compare os valores obtidos com velocidades iniciais (6) e (11) para entender de onde vem a diferença, pelo menos intuitivamente.

Veja como seria a trajetória da Lua (300 dias) com velocidade inicial

$$(v_x^{(0)}, v_y^{(0)}) = (0.0, 1500) \tag{12}$$

nos arquivos `lua15_300.out` (figura 5).

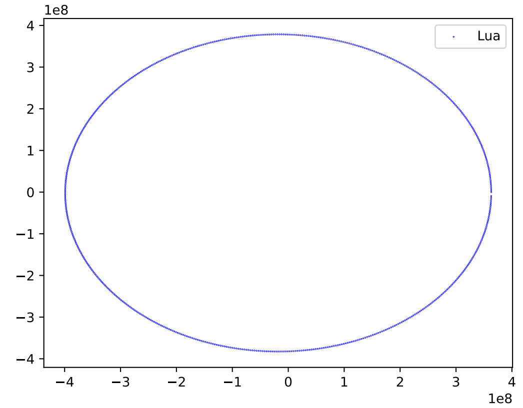

Figura 2: Órbita da Lua ao longo de 27 dias.

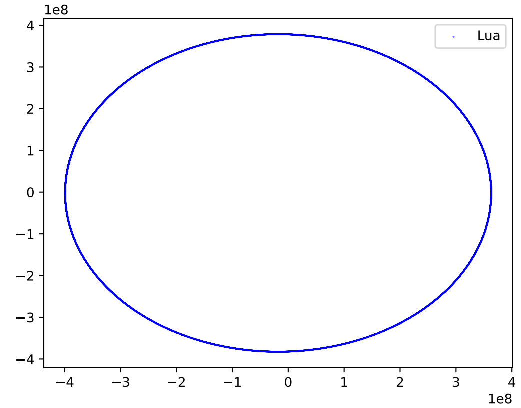

Figura 3: Órbita da Lua ao longo de 1000 dias.

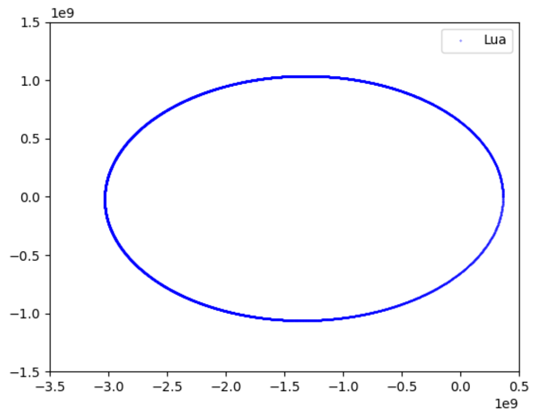

Figura 4: Órbita da Lua com veloc. inicial (11).

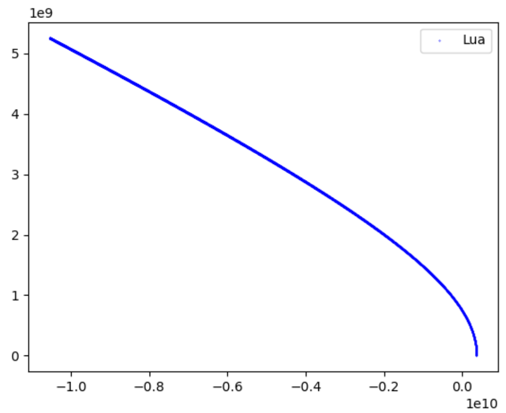

Figura 5: Órbita da Lua com veloc. inicial (12).

Observação. Veremos mais adiante como criar figuras a partir da saída do EP.

2 Sistema de vários corpos

O material discutido até agora é suficiente para simularmos sistemas celestes com vários corpos. Vamos agora considerar o caso em que temos três corpos B_0, B_1 e B_2. Suponha que esses três corpos tenham posições r_0, r_1 e r_2 e velocidades v_0, v_1, e v_2 no instante t_0. Ademais, suponha que esses corpos tenham massa m_0, m_1 e m_2. Suponha também que queremos descobrir a trajetória desses corpos até um dado instante T, determinando suas posições nos instantes t_0, $t_1 = t_0 + \Delta t$, $t_2 = t_0 + 2\Delta t$, ..., onde Δt é dado.

Para determinar a posição do corpo B_0 no instante t_1, determinamos primeiro a força gravitacional F_0 que age sobre B_0 por conta dos corpos B_1 e B_2 (veja a Seção 1.1). Agora executamos os passos (2), (3) e (4) para obter a posição $r_0(t_1)$ de B_0 no instante t_1. Note que devemos executar o procedimento análogo para os corpos B_1 e B_2, para obter suas posições $r_2(t_1)$ e $r_3(t_1)$ no instante t_1. Repetimos esse processo para obter a posições de B_0, B_1 e B_2 nos instantes t_2, t_3, etc.

Exemplo. No que segue, para especificarmos os dados de um corpo no instante $t_0 = 0$, vamos fornecer uma quíntupla, a saber,

$$(r_x, r_y, v_x, v_y, m). \tag{13}$$

Isto é, vamos fornecer as coordenadas da posição do corpo no instante t_0, as duas componentes do vetor velocidade no instante t_0, e sua massa. Neste exemplo, temos três corpos B_0, B_1 e B_2, caracterizados pelas seguintes três quíntuplas:

$$(0.0, 0.0, 0.0, 0.0, 1.9885 \times 10^{30}), \tag{14}$$

$$(1.47095 \times 10^{11}, 0.0, 0.0, 30290, 5.972 \times 10^{27}) \tag{15}$$

e

$$(1.36732 \times 10^{11}, 0.0, 0.0, 35290, 7.342 \times 22). \tag{16}$$

Simulando a evolução de B_0, B_1 e B_2 com o método acima, usando $\Delta t = 3600$ até $T = 31104000$ (360 dias), obtemos o arquivo em `fun.out` (figura 6). Note que a primeira linha desse arquivo especifica as coordenadas de B_0, B_1 e B_2 no instante $t_0 = 0$. A segunda linha especifica as coordenadas desses corpos no instante t_1 e assim por diante.

3 Seu programa

Você deve escrever um programa, digamos `ep3.py`, que recebe na entrada padrão três quíntuplas especificando três corpos B_0, B_1 e B_2, como exemplificamos acima, e os valores de T e Δt. Este programa deve enviar para a saída padrão as coordenadas dos três corpos nos instantes $t_0 = 0$, $t_1 = t_0 + \Delta t$, $t_2 = t_0 + 2\Delta t$, ..., $t_k = t_0 + k\Delta t$, onde k é o maior inteiro tal que $t_k \leq T$.

Exemplo. Com a entrada `fun.in`, seu programa deve produzir a saída `fun.out` ilustrada na figura abaixo. Essa saída foi obtida executando-se o comando a seguir no terminal do linux.

```
$ python3 ep3.py < fun.in > fun.out
```

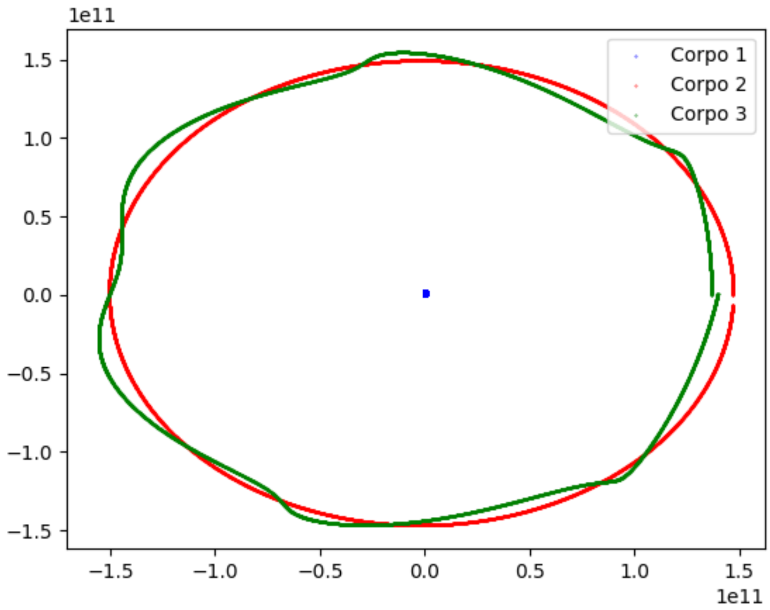

Figura 6: Órbitas obtidas com o arquivo fun.in

Seu programa será executado como acima, para verificar se ele está de acordo com as especificações do enunciado.

Observação. O cômputo de trajetórias como descrevemos acima envolve aritmética de números em ponto flutuante. Assim sendo, pode ocorrer de seu programa produzir, com entrada `fun.in`, uma saída que é levemente diferente da saída `fun.out` fornecida acima, por conta de erros de arredondamento. **Seu EP será considerado correto caso os pontos difiram por um erro absoluto ou relativo de no máximo 0.0001**.

3.1 Entrada do programa

A entrada do seu programa consiste de 5 linhas. Cada uma das primeiras 3 linhas contêm 5 números em ponto flutuante, correspondentes às informações dos 3 corpos separadas por espaços

$$(r_x, r_y, v_x, v_y, m).$$

A quarta linha contém um único inteiro T, o tempo máximo da simulação. A quinta linha contém um único inteiro, correspondente a Δt, o intervalo de tempo entre os cálculos. Veja exemplos de entrada na pasta `inputs`.

3.2 Saída do programa

A saída deve conter k linhas em que $k = \lfloor T/\Delta t \rfloor + 1$. Cada linha deve conter 6 números separados por espaços, correspondentes às posições de cada um dos 3 corpos em cada tempo considerado.

$$(r_x^{(0)}, r_y^{(0)}, r_x^{(1)}, r_y^{(1)}, r_x^{(2)}, r_y^{(3)})$$

A primeira linha da saída deve conter a posição inicial dos corpos. Observe os exemplos de saída na pasta `Output`.

3.3 Representando um corpo em python

Nesse EP você deverá representar um corpo como uma lista de 5 elementos, todos do tipo *float*. Seja *body* um corpo (uma lista com 5 elementos). Vamos convencionar que

$$\begin{aligned}
&body[0] \leftarrow r_x, \text{ ou seja, a posição 0 guarda a posição no eixo } x;\\
&body[1] \leftarrow r_y, \text{ ou seja, a posição 1 guarda a posição no eixo } y;\\
&body[2] \leftarrow v_x, \text{ ou seja, a posição 2 guarda a velocidade no eixo } x;\\
&body[3] \leftarrow v_y, \text{ ou seja, a posição 3 guarda a velocidade no eixo } y;\\
&body[4] \leftarrow m, \text{ ou seja, a posição 4 guarda a massa.}
\end{aligned}$$

Como cada corpo é representado por uma lista, podemos representar todo o sistema como uma lista de listas. Por exemplo, *bodies* pode ser uma lista de listas, onde cada lista representa um dos corpos, ou seja, $bodies[i]$ representa o corpo i. Assim, a velocidade no eixo x do corpo 1 está em $bodies[1][2]$.

3.4 Alguns detalhes de implementação

Sua implementação deve conter, obrigatoriamente, as funções cujos protótipos são dados a seguir, e que **devem estar devidamente comentados no programa**.

1. Função `dist()`:

   ```
   def dist(body_a, body_b):
   ```

   Esta função devolve a distância entre os corpos `body_a` e `body_b`.

2. Função `fg()`:

   ```
   def fg(id, bodies):
   ```

   Esta função recebe um número `id` que é 0, 1 ou 2 indicando o corpo para o qual queremos calcular a força gravitacional, e uma lista, `bodies`, composta de listas de 3 corpos.

   Esta função retorna dois números: as componentes f_x, f_y da força gravitacional f que age no corpo `id`.

3. Função `atualize()`:

   ```
   def atualize(body, fx, fy, dt):
   ```

   Esta função recebe um corpo, `body`, as componentes `fx` e `fy` da força gravitacional f que age no mesmo, e um intervalo de tempo `dt`.

   Esta função atualiza os valores das velocidades e das posições de `body` – nessa ordem – usando (3) e (4) dadas acima, tomando $\Delta t =$ `dt`.

Você poderá usar outras funções, se julgar necessário.

Este enunciado está acompanhado de um arquivo base `main.py` para auxiliá-lo.

4 Gerando as figuras

Para gerar as figuras correspondentes às órbitas que o seu EP vai calcular, você deve usar o programa `plot.py`, que acompanha este enunciado. Para executar este programa (`plot.py`), você precisa fornecer um arquivo (de entrada), digamos `fun.in`, com os pontos. O programa vai gerar uma imagem que será salva no arquivo com o nome do seu arquivo de entrada (no caso, `fun.out`). O arquivo de saída vai ser salvo na mesma pasta em que está `plot.py`. (Se você quiser, por exemplo, gerar o `fun.out` em outra pasta, você pode especificar `Output/fun.out`).

Note que o programa `plot,py` pode gerar figuras para **até** 3 corpos. (OBS: caso a entrada tenha 2 colunas o programa gera a saída para 1 corpo, se tiver 4 colunas gera para 2 corpos, e se tiver 6 colunas gera para 3 corpos.)

Recomendamos que você brinque com o código `plot.py`. O mesmo está comentado para que você possa fazer alterações.

5 Entrega

Sua missão: fazer e entregar seu programa `ep3.py`. Seu programa será considerado correto caso a resposta encontrada difira por um erro absoluto ou relativo de no máximo 10^{-4}.

Você pode entregar também até 3 arquivos de entrada (como o arquivo `fun.in`) e as imagens correspondentes. **Se você conseguir produzir entradas que produzem órbitas interessantes, você poderá receber um bônus (na nota)**.

6 Instruções para entrega do EP

As instruções contidas aqui **devem ser rigorosamente seguidas**, caso contrário, seu programa – mesmo estando correto – não receberá nota integral. As instruções são como segue:

1) O seu programa poderá usar somente os recursos da linguagem Python 3.x vistos até a última aula antes da entrega do EP. Se tiver dúvidas a respeito, pergunte à professora ou ao monitor.

2) A entrada e saída do seu programa devem seguir **exatamente** o formato dos exemplos. O seu programa não deve imprimir nada além da saída e não deve ler nada além da entrada prevista. Comandos como "Digite os dados correspodentes a cada corpo" acarretarão redução da nota.

3) Antes de entregar o seu programa, leia e siga atentamente as observações muito importantes contidas em **Instruções para a entrega de EPs em Python** (veja no e-disciplinas), onde estão descritas as instruções para a entrega dos exercícios-programas, os aspectos importantes na avaliação, a identificação no início do programa, etc.

4) O prazo limite para entrega deve ser obedecido rigorosamente. Programas fora do prazo não serão aceitos.

7 Mais um exemplo

Damos aqui um sistema interessante de três corpos. Simulamos o sistema durante intervalos de tempo variados, a saber, simulamos com valores de T iguais a 1×10^7, 2×10^7, 6.3×10^7, 6.35×10^7 e 6.5×10^7. Os arquivos de entrada que usamos são os arquivos `3body.in`, `3body_200.in`, `3body_630.in`, `3body_635.in` e `3body_650.in`, todos disponíveis na pasta `3body`. A figura 7 corresponde à entrada `3body_630.in`. Produza as demais imagens e veja as diferenças.

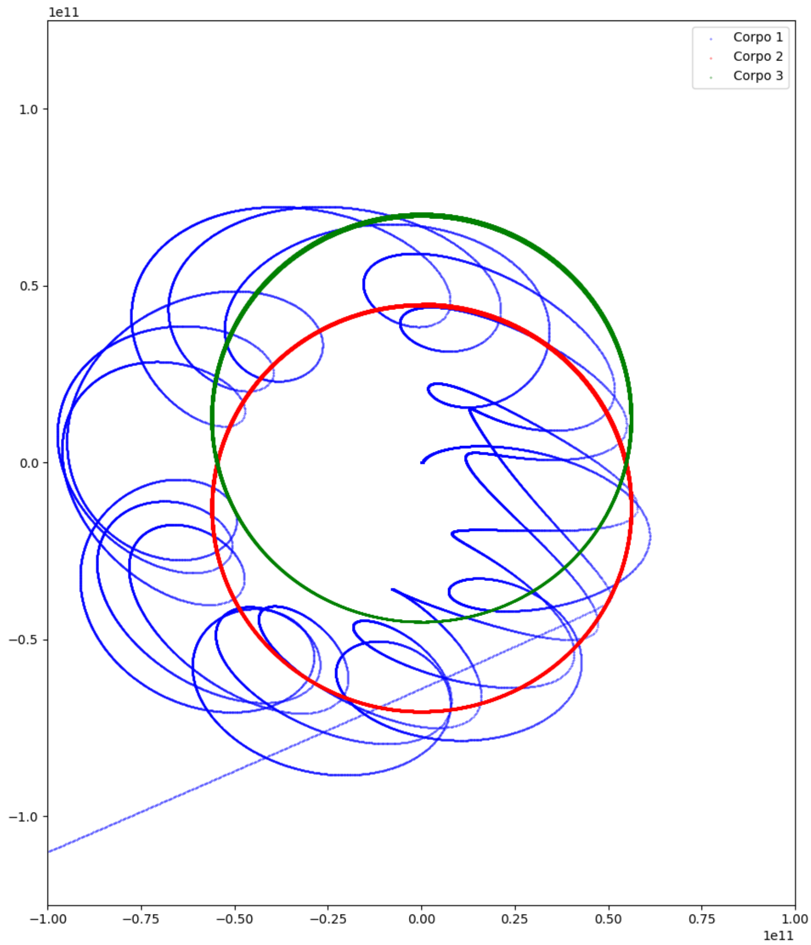

Figura 7: Órbitas obtidas com `3body_630.in`: azul (corpo 1), vermelho (corpo 2) e verde (corpo 3).

8 EP3-aquecimento

Descrevemos nas seções anteriores, como deve ser o seu `ep3.py` (que simula como a força gravitacional interage sobre 3 corpos, e produz as imagens correspondentes às trajetórias desses corpos, conforme foi explicado). No `ep3.py` você deve usar listas de listas (veja a descrição de `bodies` dada na Seção 3.3). Como você ainda não está familiarizado com listas de listas, uso de arquivos de entrada e saída, uso do programa `plot.py`, e também para entender melhor como deve ser o seu programa, você deve fazer um programa mais simples (que não precisa usar listas de listas; mas que você poderá usar, se preferir). O que você deve fazer:

- **Simular apenas a interação da Lua com a Terra, mas supondo que a Terra está na origem** $(0, 0)$ **e não se move**, conforme exemplificamos na Seção 1.4.
- Para isso, você deve fazer e entregar um programa chamado `EP3-aquecimento.py` que deverá produzir as imagens para os seguintes dados:
  (a) a posição e velocidade iniciais da Lua dadas em (5) e (6), mencionados na Seção 1.4, mas produzindo saídas para T correspondentes a 15 dias e 30 dias, e $\Delta t = 3600\,\mathrm{s}$.
  (b) a posição e velocidade iniciais da Lua dadas em (5) e (11), T correspondente a 270 dias e $\Delta t = 3600\,\mathrm{s}$
  (c) a posição e velocidade iniciais da Lua dadas em (5) e (12), T correspondente a 270 dias e $\Delta t = 3600\,\mathrm{s}$
  (d) escolher uma outra posição inicial, outra velocidade inicial para a Lua, um valor para T, $\Delta t = 3600\,\mathrm{s}$, e produzir a imagem correspondente (que seja interessante e diferente das anteriores).

OBS: Para ver se seu programa está correto, você pode simular com os dados da Seção 1.4, para ver se você obtém as mesmas figuras exibidas nessa seção.

OBS: Se você entendeu o uso de listas de listas, e prefere fazer o ep3-aquecimento.py com listas de listas (para adiantar o seu ep3.py), você pode fazer isso. Você pode também usar as funções como foram especificadas (ou adaptar a função fg para o caso desse ep3-aquecimento).