Tarea 3

Física Computacional

Diego Sarceño

201900109

18 de octubre de 2022

Los códigos tanto de *c++* como de *gnuplot*, se pueden encontrar en la carpeta de Github.

Problema 1

Dado el problema de los tres cuerpos, se tienen las ecuaciones, por componentes, de movimiento

$$\begin{aligned}
\ddot{x}_1 &= -\frac{Gm_2}{|\mathbf{r}_1-\mathbf{r}_2|^3}(x_1-x_2) - \frac{Gm_3}{|\mathbf{r}_1-\mathbf{r}_3|^3}(x_1-x_3)\\
\ddot{y}_1 &= -\frac{Gm_2}{|\mathbf{r}_1-\mathbf{r}_2|^3}(y_1-y_2) - \frac{Gm_3}{|\mathbf{r}_1-\mathbf{r}_3|^3}(y_1-y_3)\\
\ddot{x}_2 &= -\frac{Gm_2}{|\mathbf{r}_2-\mathbf{r}_1|^3}(x_2-x_1) - \frac{Gm_3}{|\mathbf{r}_2-\mathbf{r}_3|^3}(x_2-x_3)\\
\ddot{y}_2 &= -\frac{Gm_2}{|\mathbf{r}_2-\mathbf{r}_1|^3}(y_2-y_1) - \frac{Gm_3}{|\mathbf{r}_2-\mathbf{r}_3|^3}(y_2-y_3)\\
\ddot{x}_3 &= -\frac{Gm_2}{|\mathbf{r}_3-\mathbf{r}_1|^3}(x_3-x_1) - \frac{Gm_3}{|\mathbf{r}_3-\mathbf{r}_2|^3}(x_3-x_2)\\
\ddot{y}_3 &= -\frac{Gm_2}{|\mathbf{r}_3-\mathbf{r}_1|^3}(y_3-y_1) - \frac{Gm_3}{|\mathbf{r}_3-\mathbf{r}_2|^3}(y_3-y_2).
\end{aligned}$$

Lo que nos da 12 ecuaciones diferenciales de primer orden. Dadas las ecuaciones diferenciales, definimos las constantes y las respectivas condiciones iniciales:

Masa: Para cada parte del sistema

$$\begin{aligned}
m_1 &= 5.972\times10^{24} \qquad \text{(Tierra)}\\
m_3 &= 7.348\times10^{22} \qquad \text{(Luna)}\\
m_1 &= 1\times10^{3} \qquad \text{(Nave).}
\end{aligned}$$

Condiciones Iniciales: Posición

$$\begin{aligned}
x_1(0) &= 0, & y_1(0) &= 0,\\
x_2(0) &= 3.844\times10^8 m, & y_2(0) &= 0,\\
x_3(0) &= 6.8\times10^3 m, & y_3(0) &= 0.
\end{aligned}$$

Velocidad

$$\begin{aligned}
\dot{x}_1(0) &= 0, & \dot{y}_1(0) &= 0,\\
\dot{x}_2(0) &= 0, & \dot{y}_2(0) &= 1000 m/s,\\
\dot{x}_3(0) &= 7.5\times10^3\cos(\alpha) m/s, & \dot{y}_3(0) &= 7.5\times10^3\sin(\alpha) m/s.
\end{aligned}$$

Con todo esto se utilizó el método Runge-Kutta de 4to orden para resolverlo. Para el primer inciso, se eliminó la masa de la luna y se graficó la orbita de la nave alrededor de la tierra.

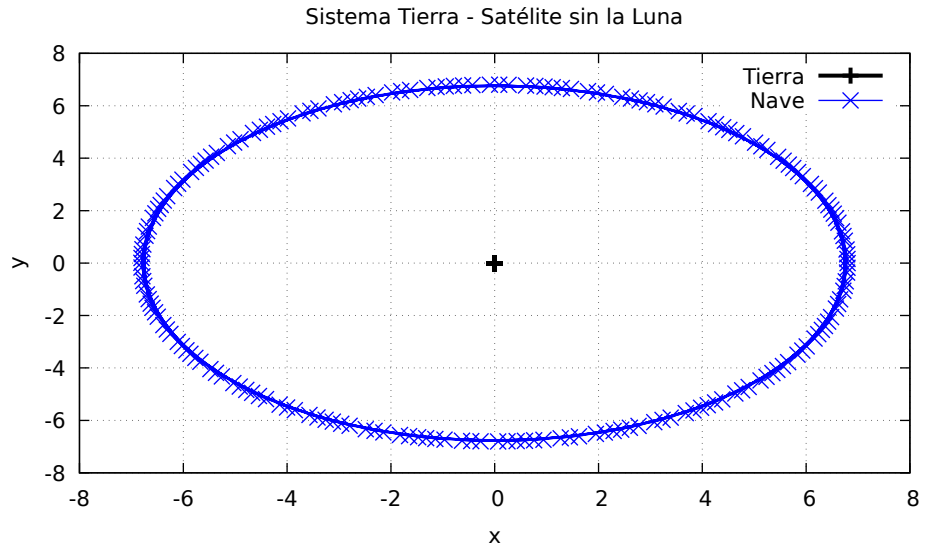

Figura 1: Gráfica de la orbita del satélite alrededor de la tierra. La gráfica no esta bajo la misma escala en ambos ejes por estética, para verla correctamente, agregar "`set size ratio -1`" en el archivo *.gp*.

```
// vie 14 oct 2022 17:25:37 CST
// ej_5-16.cpp
// Diego Sarceno (dsarceno68@gmail.com)

// Resumen

// Codificado del texto: UTF8
// Compiladores probados: g++ (Ubuntu 20.04 Linux) 9.4.0
// Instruciones de Ejecucion: no requiere nada mas
// g++ -Wall -c -o ej_5-16.o ej_5-16.cpp
// g++ -o ej_5-16.x ej_5-16.o


// Librerias
#include <iostream>
#include <cmath>
#include <iomanip>
#include <fstream>

using namespace std;

void RK4(const double *y,
                    const int n_ec,
                    const double t,
                    const double h,
```

```
26                      double *y_imas1,
27                      void (*derivada)( const double *, const double,
                           double *));
28 void salidaSolucion(const double t, const double *y, const int N);
29 void spaceCraft(const double *y, const double t, double *dydt);
30
31
32 int main(){
33   const double t0 = 0.0;
34   const double h = 0.5;
35   const int N = 1000000;
36   const int frec_out = 5000;
37   const int n_ec = 12;
38   const int alpha = 0;
39
40
41   // espacio de y
42   double *y = new double[n_ec];
43   double *y_nueva = new double[n_ec];
44
45   // condiciones iniciales
46   y[0] = 0.0;
47   y[1] = 0.0;
48   y[2] = 3.844e8;
49   y[3] = 0.0;
50   y[4] = (6.8e6)*cos(alpha);
51   y[5] = (6.8e6)*sin(alpha);
52   y[6] = 0.0;
53   y[7] = 0.0;
54   y[8] = 0.0;
55   y[9] = 1000;
56   y[10] = (7.5e3)*cos(M_PI_2 + alpha);
57   y[11] = (7.5e3)*sin(M_PI_2 + alpha);
58
59   // puntero a la 'derivada'
60   void (*derivada)( const double *, const double, double * );
61   derivada = spaceCraft;
62
63
64   // incializar y_nueva
65   for(int i = 0; i < n_ec; i++) y_nueva[i] = 0.0;
66
67   double t = t0;
68
69   // escribir las condiciones iniciales en el programa y la descripcion
        de las columnas de Datos
70   cout << "# Columna 1: Tiempo" << endl;
71   cout << "# Columna 2: Tierra-x" << endl;
```

```
72    cout << "# Columna 3: Tierra-y" << endl;
73    cout << "# Columna 4: Luna-x" << endl;
74    cout << "# Columna 5: Luna-y" << endl;
75    cout << "# Columna 6: Nave-x" << endl;
76    cout << "# Columna 7: Nave-y" << endl;
77    cout << "# Columna 8: Tierra-v-x" << endl;
78    cout << "# Columna 9: Tierra-v-y" << endl;
79    cout << "# Columna 10: Luna-v-x" << endl;
80    cout << "# Columna 11: Luna-v-y" << endl;
81    cout << "# Columna 12: Nave-v-x" << endl;
82    cout << "# Columna 13: Nave-v-y" << endl;
83
84
85    salidaSolucion(t, y, n_ec);
86
87    for(int i = 1; i <= N; i++){
88      RK4(y, n_ec, t, h, y_nueva, derivada);
89
90      y = y_nueva;
91      t += h;
92
93      if (i % frec_out == 0){
94        salidaSolucion(t, y, n_ec);
95      } // END if
96    } // END for
97    return 0;
98  } // END main
99
100
101
102 void RK4(const double *y,
103                       const int n_ec,
104                       const double t,
105                       const double h,
106                       double *y_imas1,
107                       void (*derivada)( const double *, const double,
                             double *)){
108   double *k0 = new double[n_ec];
109   double *k1 = new double[n_ec];
110   double *k2 = new double[n_ec];
111   double *k3 = new double[n_ec];
112   double *z  = new double[n_ec];
113
114   (*derivada)(y, t, k0);
115
116   for(int i=0; i<n_ec; i++)
117     z[i] = y[i] + 0.5*k0[i]*h;
118
```

```
119   (*derivada)(z, t+0.5*h, k1);
120
121   for(int i=0; i<n_ec; i++)
122     z[i] = y[i] + 0.5*k1[i]*h;
123
124   (*derivada)(z, t+0.5*h, k2);
125
126   for(int i=0; i<n_ec; i++)
127     z[i] = y[i] + k2[i]*h;
128
129   (*derivada)(z, t+h, k3);
130
131   for(int i=0; i<n_ec; i++)
132    y_imas1[i] = y[i] + h/6.0 * ( k0[i] + 2*k1[i] + 2*k2[i] + k3[i] );
133
134   delete[] k0;
135   delete[] k1;
136   delete[] k2;
137   delete[] k3;
138   delete[] z;
139 }
140
141
142
143
144
145 void salidaSolucion(const double t, const double *y, const int N){
146   cout << fixed << setprecision(3) << t;
147
148   for( int i=0; i<N; i++ )
149     cout << scientific << setprecision(9) << "\t" << y[i];
150
151   cout << endl;
152 }
153
154
155
156
157 void spaceCraft(const double *y, const double t, double *dydt){
158   const double m1 = 5.972e24; //masa de la tierra
159   const double m2 = 7.348e22; //masa de la luna
160         //const double m2 = 0.0; // masa de la luna, prueba ej 1
161   const double m3 = 1000; //masa de la nave
162   const double G  = 6.66e-11; // Constante de gravitacion universal
163
164   // distancias
165   const double r21_3 = pow( pow(y[2]-y[0],2) + pow(y[3]-y[1],2), 1.5 );
166   const double r31_3 = pow( pow(y[4]-y[0],2) + pow(y[5]-y[1],2), 1.5 );
```

```
167   const double r32_3 = pow( pow(y[4]-y[2],2) + pow(y[5]-y[3],2), 1.5 );
168         ofstream truefalse;
169
170         truefalse.open("truefalse", ios::out);
171         if(r32_3 <= 3.8e6){
172                 truefalse << r32_3 << endl;
173         } // END if
174
175
176
177   dydt[0]  = y[6];
178   dydt[1]  = y[7];
179   dydt[2]  = y[8];
180   dydt[3]  = y[9];
181   dydt[4]  = y[10];
182   dydt[5]  = y[11];
183   dydt[6]  = -G*m2*( y[0]-y[2] ) / r21_3 - G*m3*( y[0]-y[4] ) / r31_3;
184   dydt[7]  = -G*m2*( y[1]-y[3] ) / r21_3 - G*m3*( y[1]-y[5] ) / r31_3;
185   dydt[8]  = -G*m1*( y[2]-y[0] ) / r21_3 - G*m3*( y[2]-y[4] ) / r32_3;
186   dydt[9]  = -G*m1*( y[3]-y[1] ) / r21_3 - G*m3*( y[3]-y[5] ) / r32_3;
187   dydt[10] = -G*m1*( y[4]-y[0] ) / r31_3 - G*m2*( y[4]-y[2] ) / r32_3;
188   dydt[11] = -G*m1*( y[5]-y[1] ) / r31_3 - G*m2*( y[5]-y[3] ) / r32_3;
189 } // END spaceCraft
```

Problema 2

Para este problema únicamente se jugó con las condiciones iniciales, variando parámetros como α y v_o. Se llegó a las siguientes condiciones iniciales para que la nave pasara por la luna.

Posición: $x_3(0) = 6.8 \times 10^6 m$ y $y_3(0) = 0$. Es decir un $\alpha = 0$.

Velocidad: $\dot{x}_3(0) = 2.5 \times 10^4 m/s$ y $\dot{y}_3(0) = 0 m/s$.

Dejando la siguiente gráfica

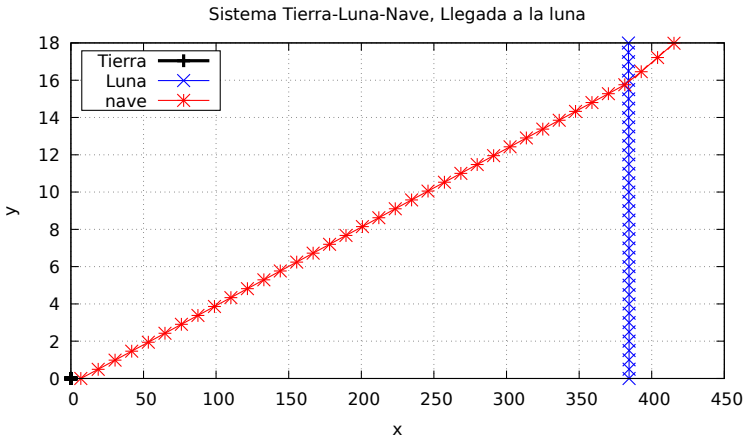

Figura 2: Es claro que la nave pasa lo suficientemente cerca de la luna.

Problema 3

Bajo la misma idea del ejercicio anterior, se econtraron las siguientes condiciones iniciales.

Posición: $x_3(0) = 6.8 \times 10^6 \cos(36.869^o)m$ y $y_3(0) = 6.8 \times 10^6 \sin(36.869^o)m$. Es decir un $\alpha = 36.869^o$.

Velocidad: $\dot{x}_3(0) = 7728m/s$ y $\dot{y}_3(0) = 7420m/s$, se utilizó otro ángulo por pura prueba y error.

Dejando la siguiente gráfica

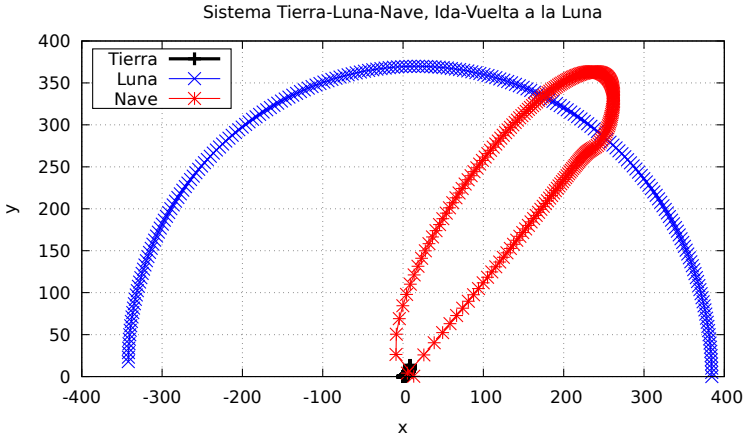

Figura 3: Es claro que la nave pasa lo suficientemente cerca de la luna como para forzar un retorno en su trayectoria.