Simulaciones en el problema restringido circular plano de los tres cuerpos

Fernando Chamizo

6 de junio de 2017

Resumen

Después de algunas consideraciones teóricas básicas sobre los puntos de Lagrange y las curvas de velocidad cero, se estudia experimentalmente el comportamiento de algunas de las órbitas que aparecen en el problema restringido circular plano de los tres cuerpos. El código empleado para generar las figuras y los datos está disponible en:

`http://www.uam.es/fernando.chamizo/physics/files/prog_calc_astr.tar.gz`

Índice

1. El modelo teórico

Dos masas m_1 y m_2 bajo la acción de la atracción gravitatoria admiten órbitas elípticas con foco común en el centro de masas, que supondremos el origen, y ejes mayores contenidos en una misma recta. En el caso especial de excentricidad nula tenemos órbitas circulares recorridas con velocidad angular constante ω_0 dada por la tercera ley de Kepler

$$\omega_0^2 = G\frac{m_1+m_2}{r_{12}^3} \tag{1}$$

con r_{12} la distancia entre ambas masas. Esta situación se ajusta aproximadamente a los movimientos de los planetas con la excepción de Mercurio ya que no presentan grandes excentricidades:

	☿	♀	♁	♂	♃	♄	♅	♆
e	0.2056	0.0068	0.0167	0.0934	0.0484	0.0542	0.0472	0.0086

El *problema restringido circular de los tres cuerpos* consiste en introducir una tercera masa m_3 cuya influencia gravitatoria sobre m_1 y m_2 es despreciable. Por ejemplo, podría corresponder a una sonda espacial suficientemente cerca de Marte para que el sistema Sol-Marte sea predominante.

Si $\mathbf{r}(t)$ describe el movimiento de m_3, las ecuaciones de Newton exigen

$$\ddot{\mathbf{r}} = -\frac{Gm_1\mathbf{r}_{13}}{r_{13}^3} - \frac{Gm_2\mathbf{r}_{23}}{r_{23}^3}. \tag{2}$$

Escribamos $\mathbf{r} = \sum x_i\mathbf{e}_i$ donde $\mathcal{R} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ es un sistema de referencia que gira con velocidad angular $\boldsymbol{\omega}$. Empleando $\dot{\mathbf{e}}_i = \boldsymbol{\omega} \times \mathbf{e}_i$ se deduce

$$\ddot{\mathbf{r}} = \sum \big(\ddot{x}_i\mathbf{e}_i + 2\dot{x}_i\boldsymbol{\omega} \times \mathbf{e}_i + \dot{x}_i\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{e}_i)\big). \tag{3}$$

El segundo sumando es la aceleración de Coriolis y el tercero la aceleración centrífuga. Digamos que el plano orbital de m_1 y m_2 es el plano XY, entonces eligiendo $\mathbf{e}_3 = \widehat{\mathbf{z}}$ y $\boldsymbol{\omega} = \omega_0\mathbf{e}_3$, en el sistema de referencia $\mathcal{R}$ estas masas permanecerán estáticas: sus coordenadas serán constantes.

Escojamos ahora un sistema de unidades de masas y longitudes en el que

$$G(m_1+m_2) = 1 \qquad \text{y} \qquad r_{12} = 1 \tag{4}$$

entonces (1) asegura $\omega_0 = 1$. Si denominamos $(-\mu, 0)$ y $(1-\mu, 0)$ a las posiciones estáticas de m_1 y m_2, se cumple (recordemos que el centro de masas se había supuesto el origen)

$$\mu = \frac{m_2}{m_1+m_2} \qquad \text{y} \qquad 1-\mu = \frac{m_1}{m_1+m_2}. \tag{5}$$

Debido a la simetría, no hay restricción en suponer $0 \leq \mu \leq 1/2$. Con esta notación, las coordenadas en $\mathcal{R}$ del segundo miembro de (2) son

$$(6)\quad \nabla\Big(\frac{1-\mu}{r_1}+\frac{\mu}{r_2}\Big) \quad \text{con} \quad r_1=\sqrt{(x_1+\mu)^2+x_2^2+x_3^2} \quad \text{y} \quad r_2=\sqrt{(x_1-1+\mu)^2+x_2^2+x_3^2}.$$

Separando las coordenadas del primer miembro de (2) con (3), se deducen las ecuaciones de movimiento en $\mathcal{R}$

$$(7)\quad \ddot{x}_1=2\dot{x}_2+\frac{\partial U}{\partial x_1}, \quad \ddot{x}_2=-2\dot{x}_1+\frac{\partial U}{\partial x_2}, \quad \ddot{x}_3=\frac{\partial U}{\partial x_3} \quad \text{con} \quad U=\frac{1}{2}(x_1^2+x_2^2)+\frac{1-\mu}{r_1}+\frac{\mu}{r_2}.$$

Los *puntos de Lagrange* son aquellos en los que la masa m_3 puede permanecer con velocidad relativa nula con respecto a m_1 y m_2. En el sistema no rotado, veríamos que las tres masas avanzan con la misma velocidad angular en sus órbitas circulares. De acuerdo con (7) los puntos de Lagrange vienen determinados por la ecuación $\nabla U=\vec{0}$. La tercera coordenada lleva a $x_3=0$, como es lógico pensando en la dirección de las fuerzas. En consonancia con ello, en este trabajo nos centraremos sobre todo en el *problema plano* en que m_3 se mueve en el plano orbital de m_1 y m_2 identificado con XY. Consecuentemente en lo sucesivo escribiremos x e y en vez de x_1 y x_2. De este modo el *potencial efectivo* U es

$$(8)\qquad U=\frac{1}{2}(x^2+y^2)+\frac{1-\mu}{r_1}+\frac{\mu}{r_2} \quad \text{con} \quad \begin{cases} r_1=\sqrt{(x+\mu)^2+y^2}, \\ r_2=\sqrt{(x-1+\mu)^2+y^2}. \end{cases}$$

Reescribiendo esta fórmula como

$$(9)\qquad U=(1-\mu)\big(r_1^{-1}+\frac{1}{2}r_1^2\big)+\mu\big(r_2^{-1}+\frac{1}{2}r_2^2\big)-\frac{1}{2}\mu(1-\mu),$$

las ecuaciones para los puntos de Lagrange son

$$(10)\qquad \begin{cases} (1-\mu)\big(1-r_1^{-3}\big)(x+\mu)+\mu\big(1-r_2^{-3}\big)(x-1+\mu)=0, \\ (1-\mu)\big(1-r_1^{-3}\big)y+\mu\big(1-r_2^{-3}\big)y=0. \end{cases}$$

Las soluciones que se ven a simple vista son las que son menos naturales desde el punto de vista físico: $r_1=r_2=1$ que dan lugar a los puntos

$$(11)\qquad L_4=\big(\frac{1}{2}-\mu,\frac{\sqrt{3}}{2}\big) \qquad \text{y} \qquad L_5=\big(\frac{1}{2}-\mu,-\frac{\sqrt{3}}{2}\big).$$

Geométricamente corresponden a que las tres masas formen un triángulo equilátero. El resto de las soluciones de (10), tienen $y=0$, es decir, corresponden al caso natural físicamente en que las tres masas están alineadas y las fuerzas sobre m_3 se compensan con la fuerza centrífuga.

La numeración habitual es llamar L_1 a la solución entre m_1 y m_2, L_2 a la que está a la derecha de m_2 y L_3 el que está a la izquierda de m_1. Son soluciones de ecuaciones de quinto grado [Cor98] que en general no admiten soluciones explícitas. Una situación natural se da cuando m_1 es mucho mayor que m_2 (por ejemplo en el citado ejemplo del Sistema Solar), lo que se traduce por (5) en que μ es pequeño. En esa situación, se tienen las aproximaciones

$$L_1 \approx \big(1-(\mu/3)^{1/3},0\big), \qquad L_2 \approx \big(1+(\mu/3)^{1/3},0\big) \quad \text{y} \quad L_3 \approx \big(-1-5\mu/12,0\big). \tag{12}$$

Es fácil dar explicaciones sencillas de estas aproximaciones sin pasar al sistema de referencia $\mathcal{R}$. Por ejemplo, para L_1, una partícula en una órbita circular de radio r con frecuencia angular ω_0 sufre con nuestras unidades una fuerza centrífuga $\omega_0^2 r = r$ y una atracción gravitatoria por m_1 y m_2 dada por $(1-\mu)/(r-\mu)^2$ y $\mu/(r-1+\mu)^2$. El balance de fuerzas para $r = 1-\epsilon$ bajo el *ansatz* $\mu = O(\epsilon^2)$ es

$$\frac{1-\mu}{(r-\mu)^2} = \frac{\mu}{(r-1+\mu)^2} + r \quad \text{que implica} \quad 1+2\epsilon+O(\epsilon^2) = \mu\epsilon^{-2}+1-\epsilon. \tag{13}$$

Por tanto $\epsilon = (\mu/3)^{1/3} + O(\mu^{2/3})$, que es coherente con el *ansatz*.

La siguiente función en `SageMath`[1] genera la lista de los puntos de Lagrange "exactos" resolviendo (10) con $y = 0$ y usando las expresiones (11):

```
def l_lag(mu):
        lag = [0,0,0, ((1-2*mu)/2, sqrt(3)/2), ((1-2*mu)/2, -sqrt(3)/2) ]

        fu = (1-mu)*(x-1+mu)^2*((x+mu)^3-1)+mu*(x+mu)^2*((x-1+mu)^3+1)
        lag[0] = (find_root(fu, -mu,1-mu) ,0)

        fu = (1-mu)*(x-1+mu)^2*((x+mu)^3-1)+mu*(x+mu)^2*((x-1+mu)^3-1)
        lag[1] = (find_root(fu, 1-mu, 2) ,0)

        fu = (1-mu)*(x-1+mu)^2*((x+mu)^3+1)+mu*(x+mu)^2*((x-1+mu)^3+1)
        lag[2] = (find_root(fu, -2, -mu) , 0)

        return lag
```

Parte del código citado después hará referencia a esta función.

2. Curvas de velocidad cero

A partir de (7) se obtiene

$$\dot{x}_1\ddot{x}_1 + \dot{x}_2\ddot{x}_2 + \dot{x}_3\ddot{x}_3 = \dot{x}_1\frac{\partial U}{\partial x_1} + \dot{x}_2\frac{\partial U}{\partial x_2} + \dot{x}_3\frac{\partial U}{\partial x_3} = \frac{dU}{dt}. \tag{14}$$

[1] Es un sistema algebraico computacional que combina varios paquetes contrastados escritos en diferentes lenguajes a través de una sintaxis basada en `Python`.

Lo cual implica que la llamada *constante de Jacobi*

$$c_J = 2U - (\dot{x}_1^2 + \dot{x}_2^2 + \dot{x}_3^2) \tag{15}$$

es una constante de movimiento.

Es posible traducir esta ley de conservación en una formulación Hamiltoniana [Ver09]. Así en el caso plano las ecuaciones de movimiento pueden derivarse de las ecuaciones de Hamilton con

$$H(x, y, p_x, p_y) = \frac{1}{2}\left(p_x^2 + p_y^2\right) + yp_x - xp_y - \frac{1-\mu}{r_1} - \frac{\mu}{r_2} \tag{16}$$

donde $p_x = \dot{x} - y$, $p_y = \dot{y} + x$, son los momentos conjugados a x e y. La conservación de c_J con $x_3 = 0$ equivale a la conservación de la "energía" H.

Según la definición de la constante de Jacobi, las curvas implícitas $c_J = 2U$ están formadas por todos los puntos en los que $\dot{x} = \dot{y} = 0$. Se dice que son *curvas de velocidad cero*. Hay que aclarar que esta velocidad es relativa al sistema de referencia $\mathcal{R}$, por tanto vistas "desde fuera" corresponden a puntos en los que m_3 se mueve con velocidad angular igual a la de m_1 y m_2. Estas curvas no corresponden a órbitas de m_3 porque no satisfacen (7), lo cual es consecuencia indirecta de que solo hay cinco puntos de Lagrange. La importancia de las curvas de velocidad cero radica en que establecen la frontera entre las zonas $c_J > 2U$ que son *zonas prohibidas* al movimiento por corresponder a velocidades imaginarias, y las *zonas permitidas* $c_J > 2U$ que podría alcanzar m_3 en una trayectoria con condiciones iniciales correspondientes a c_J.

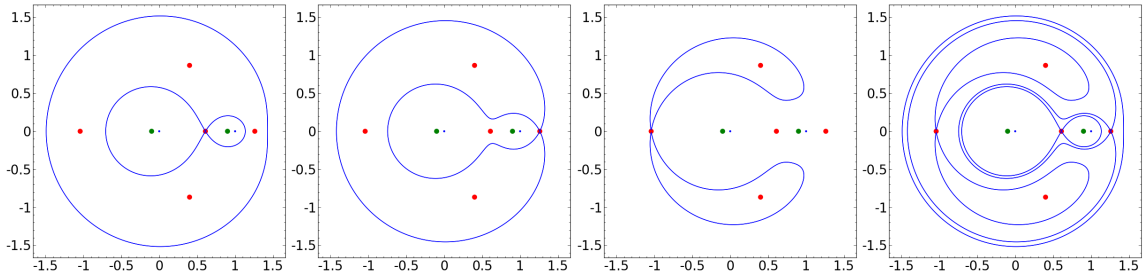
Las curvas de velocidad cero correspondientes a las diferentes constantes de Jacobi no son entonces otra cosa que las curvas de nivel de U. Por otro lado, según vimos tras (7), los puntos de Lagrange son puntos críticos para U. Es fácil ver que en L_4 y L_5 se alcanza un mínimo ya que son puntos críticos y en ellos

$$\frac{\partial^2 U}{\partial x^2} = \frac{3}{4} > 0, \qquad \frac{\partial^2 U}{\partial x^2}\frac{\partial^2 U}{\partial y^2} - \left(\frac{\partial^2 U}{\partial x \partial y}\right)^2 = \frac{3}{4}\cdot\frac{9}{4} - \frac{27}{16}(1-2\mu)^2 = \frac{27}{4}\mu(1-\mu) > 0. \tag{17}$$

Por tanto las curvas de nivel que las contienen se reducen a un punto. Por otra parte, L_1, L_2 y L_3 son puntos de silla de U [MD99, §3.7.1] y entonces existen curvas de velocidad cero críticas que pasan por ellos. La siguiente función en `SageMath` dada una lista de índices `L` dibuja estas curvas críticas donde $i-1$ corresponde a L_i.

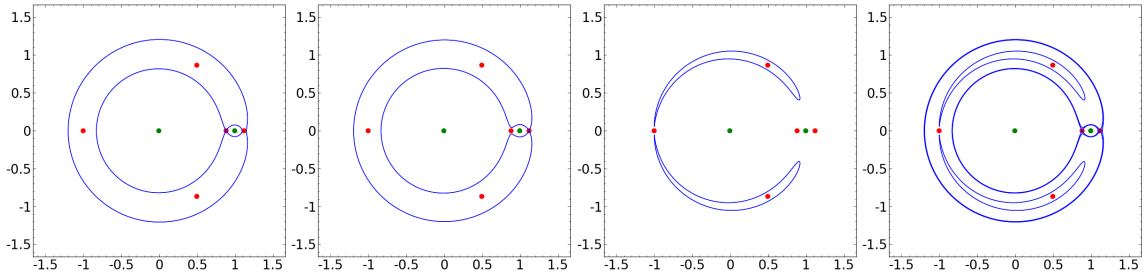
```
def zero_vel_c(a, L):
        lag = l_lag(a)
        f = x^2 + y^2 +2*((1-a)/sqrt((x+a)^2+y^2)+a/sqrt((x-1+a)^2+y^2))
        P = point([(0,0)],size=0)
        for k in L:
                c = f(x=lag[k][0], y=lag[k][1])
                P += implicit_plot(f-c, (x,-2,2), (y,-2,2), plot_points=400)
        return P
```

Las siguientes figuras se obtienen invocando a `zero_vel_c(0.1,L)` con `L=[0]`, `L=[1]`, `L=[2]` y `L=[0,1,2]` (esto es, la superposición de las figuras anteriores).



Para interpretar mejor los resultados se dibujan en rojo los cinco puntos de Lagrange y en verde las dos masas m_1 y m_2. Además se indican en azul en tamaño pequeño el origen y el $(1, 0)$ para que sirvan de referencia.

Es interesante ver la evolución de estas curvas cuando se disminuye el valor de μ, por ejemplo cuando cambiamos $\mu = 0.1$ por $\mu = 0.005$ son

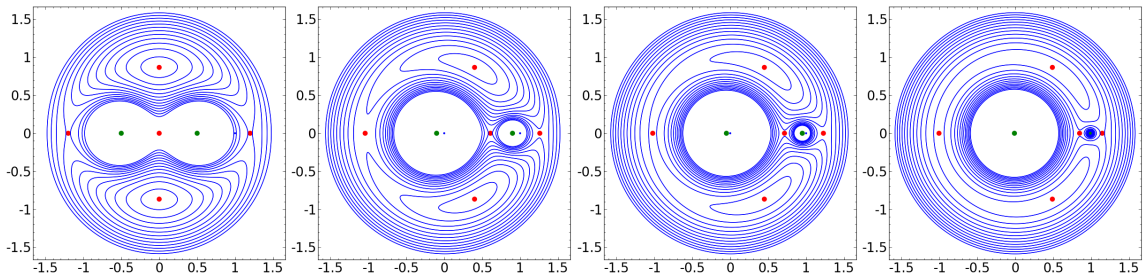


Para dibujar familias de curvas de velocidad cero (no críticas) se emplea la siguiente función que ajusta autmáticamente el rango aprovechando que en L_4 se alcanza un mínimo y buscando un máximo en los bordes del marco de la figura.

```
def zero_vel(mu, l=1.6):
        P = point([(0,0)],size=0)
        f = x^2 + y^2 +2*((1-mu)/sqrt((x+mu)^2+y^2)+mu/sqrt((x-1+mu)^2+y^2))
        cmin = f(x=(1-2*mu)/2, y=sqrt(3)/2)
        cmax = min( f(x=-l, y=0),f(x=0, y=-l) )
        N = 12
        h = (cmax-cmin)/N
        for c in srange(cmin+h/2, cmax, h):
                P += implicit_plot(f-c, (x,-l,l), (y,-l,l), plot_points=400)
        return P
```

El número de curvas viene dado por `N` y está arbitrariamente fijado a 12. el parámetro opcional `l` especifica el tamaño del cuadrado dentro del que se dibujan las curvas.

Aquí se muestran algunos ejemplos para diferentes valores del único argumento obligatorio μ.



$\mu = 0.5$ $\mu = 0.1$ $\mu = 0.05$ $\mu = 0.01$

La última figura muestra un hecho físicamente claro: cuando la masa m_2 es despreciable y no estamos en sus cercanías, el sistema tiende a tener simetría radial y las curvas de velocidad cero son casi circulares.

En [Dan88, p. 259] (ver también [MD99, p. 81]) se puede ver una representación tridimensional de las curvas de velocidad cero como curvas de nivel de $2U$.

3. Esquema del código para el cálculo de las órbitas

La existencia de la formulación Hamiltoniana permitiría en principio el uso de integradores simplécticos pero como el Hamiltoniano (16) no es separable podrían ser costosos numéricamente por ser implícitos (parece que hay alternativas explícitas aumentando artificialmente la dimensión y componiendo soluciones [Tao16]). En aras de la simplicidad, usamos el clásico esquema explícito Runge-Kutta `RK4` que, dicho sea de paso, hemos visto que se emplea en algún artículo de investigación reciente [Sic10]. La implementación de esta parte es en C.

Las coordenadas en el espacio de fases de velocidades (posiciones y velocidades) están representadas por la estructura `ph_sp` que responde a

```
typedef struct _ph_sp{
        double_t x;
        double_t y;
        double_t vx;
        double_t vy;
} ph_sp;
```

Para entender el esquema del código C, examinemos la rutina `main`:

```
1  int main(void){
2          int i;
3          FILE *fp;
4
5          ph_sp *coord_rk4 = (ph_sp*)calloc( N_STEPS+1,  sizeof(ph_sp));
```

```
6
7           runge_kutta4( coord_rk4 );
8
9           fp = fopen("r3b.dat", "w");
10          if (NULL == fp) {
11                  fprintf(stderr, "File failed\n");
12                  return FAILURE;
13          }
14          fprintf(fp,"%.15e\n%.15e\n%d\n", MU, CJ, N_STEPS+1);
15          for(i=0; i<N_STEPS+1; i++){
16                  fprintf(fp,"%e, %e\n", (double)coord_rk4[i].x,
                        ↪ (double)coord_rk4[i].y);
17          }
18          fclose(fp);
19
20          free(coord_rk4);
21
22          return SUCCESS;
23  }
```

La línea 5 reserva el espacio en memoria para almacenar los datos de la órbita y la 7 llama al método numérico de Runge-Kutta. El resultado quedará acumulado en la estructura de datos `coord_rk4`. Las líneas 9–18 extraen las coordenadas x e y y las escribe en el fichero de datos `r3b.dat` que se encabeza con los valores de μ, la constante de Jacobi c_J y el número de pasos más uno. Los dos primeros los utilizará el código que dibuja los datos y el último evitará errores de lectura. Finalmente, en la línea 20 se libera la memoria.

La rutina `runge_kutta4` incorpora el método Runge-Kutta 4 habitual para resolver una ecuación diferencial $y'(x) = f(x, y)$ [SB02, (7.2.1.14)]. La única dificultad en la implementación es que hay tratar el problema vectorial. En relación con ello, se añade una función que halla combinaciones lineales de la forma $\vec{v} + \lambda\vec{w}$ pero nada de esto es suficientemente reseñable para copiar aquí el código concreto. La función f está aquí determinada por la dinámica (7), que en el espacio de fases de velocidades pasa a ser de primer orden. La función que evalúa f se denomina `dynamics` y responde al código

```
ph_sp dynamics(ph_sp ph){
        ph_sp outph;
        double_t r1c = ph.x+MU;
        double_t r2c = ph.x-1.0+MU;
        r1c = sqrt(r1c*r1c+ph.y*ph.y);
        r1c *= r1c*r1c;
        r2c = sqrt(r2c*r2c+ph.y*ph.y);
        r2c *= r2c*r2c;
        /* velocity */
        outph.x = ph.vx;
        outph.y = ph.vy;
        /* Force */
        outph.vx = -(1.0-MU)*(ph.x+MU)/r1c-MU*(ph.x-1.0+MU)/r2c;
        outph.vy = -(1.0-MU)*ph.y/r1c-MU*ph.y/r2c;
```

```
        /* Efective force (Coriolis correction) */
        outph.vx += 2.0*ph.vy + ph.x;
        outph.vy += -2.0*ph.vx + ph.y;
        return outph;
}
```

Los valores que determinan la órbita aparecen como parámetros a través de la directiva `#define` dentro de un fichero de cabecera `.h` que, como veremos, es generado automáticamente por un programa `SageMath`. Los parámetros son μ y las posiciones y velocidades iniciales $\big(x(0), y(0)\big)$ y $\big(\dot{x}(0), \dot{y}(0)\big)$ (indicados a la izquierda para un ejemplo de órbita *horseshoe* que veremos más adelante)

```
#define MU 0.000953875
#define X0 -0.97668
#define Y0 0.0
#define VX0 0.0
#define VY0 -0.06118
```

```
#define N_ORB 30.0
#define DT 0.01
```

Por otra parte, hay dos parámetros relativos al método numérico (columna de la derecha), `N_ORB` indica el tiempo en unidades de órbitas de las masas m_1 y m_2, mientras que `DT` indica la discretización en el tiempo. A partir de ellos se define el número de pasos con

```
#define N_STEPS ((int)(2.0*M_PI*N_ORB/DT))
```

que es conveniente utilizar directamente en la rutina `runge_kutta4`.

Para mayor comodidad, se interactúa con el código C anterior a través de una función, `orbit`, definida en `SageMath` (de hecho en la parte no gráfica es en su mayoría `Python` puro) que tiene tres argumentos, el último de ellos opcional. El primero es una lista con los parámetros orbitales y el segundo con los del método numérico, ordenados como se ha indicado antes. Así, la órbita puesta como ejemplo se invoca con

```
Lorb = [0.000953875, -0.97668, 0.0, 0.0,-0.06118]
Lnum = [30.0, 0.01]
orbit( Lorb, Lnum )
```

El tercer parámetro opcional es una lista de la forma `[xmin,xmax,ymin,ymax,filen]` que indica las dimensiones de la ventana que se va dibujar y el nombre del fichero en el que se guardará la figura. La definición de `orbit` es

```
def orbit( Lor, Lnu, Lpl=[-1.6,1.6,-1.6,1.6,'./r3b.eps']):
        data_to_file(Lor, Lnu)
        run_and_plot(Lpl)
```

donde `data_to_file` crea el fichero de cabecera `.h` para que lo emplee el programa en C y `run_and_plot` llama a dicho programa y lee y representa los resultados almacenados en

el fichero `r3b.dat`. Además de los datos se representa también la curva de velocidad cero correspondiente a la órbita para que sean claras las zonas prohibidas.

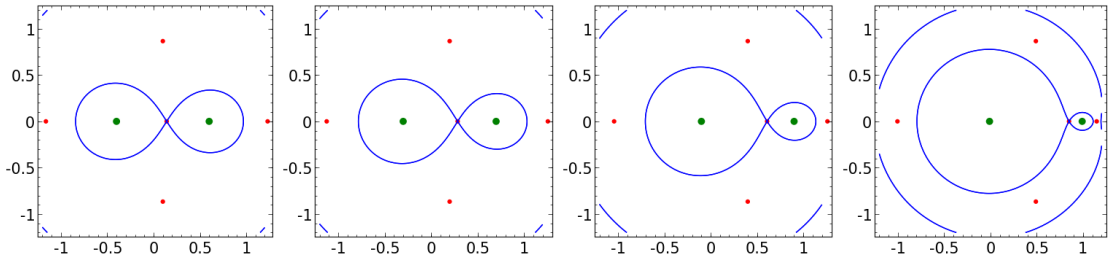
Una pequeña nota técnica es que para evitar imágenes muy "pesadas" que no añaden información, en `run_and_plot` solo se aprovechan los datos de `r3b.dat` que podrían dar lugar a píxeles distintos con resolución razonable. Esto se hace con el `if` de

```
for k in range(nsteps):
        line=f.readline()
        xy = line.split(',')
        xy = (float(xy[0]),float(xy[1]))
        if abs(lxy[0]-xy[0])/res_x + abs(lxy[1]-xy[1])/res_y > 1:
                lxy = xy
                Ldat.append( xy )
```

Aquí `res_x` y `res_y` se calculan automáticamente para que haya a lo más 800 píxeles por coordenada.

4. Confinamiento alrededor de las masas

Como ya hemos mencionado, en $L_1 = (x_1, 0)$ el potencial U alcanza un punto de silla y la presencia a izquierda y derecha, en $(-\mu, 0)$ y $(1-\mu, 0)$, de las singularidades correspondientes a las masas m_1 y m_2 causa que la curva de velocidad cero crítica que pasa por L_1 tenga dos componentes conexas, una exterior con forma redondeada y otra interior con forma de ocho tumbado (aquí [Dan88, Fig. 8.6] puede dar cierta intuición de por qué ocurre esto). Según μ varía entre $1/2$ y 0, la coordenada x_1 se traslada hacia la derecha y consecuentemente el lazo derecho de la componente interior se va reduciendo.



$\mu = 0.4$ $\mu = 0.3$ $\mu = 0.1$ $\mu = 0.01$

El valor de la constante de Jacobi para estas curvas críticas es

$$c_1(\mu) = x_1^2 + \frac{2(1-\mu)}{x_1+\mu} + \frac{2\mu}{1-\mu-x_1}. \tag{18}$$

Cuando $\mu \to 0^+$ se tiene $x_1 \to 1^-$ y por tanto $c_1(0^+) = 3^+$. En el otro extremo, si $\mu \to 0.5^-$ entonces $x_1 \to 0^+$ y $c_1(0.5^-) = 4^-$. Para comprobar que $c_1(\mu)$ crece con μ, llamemos $F(\mu, x_1)$

al segundo miembro de (18). Tenemos que ver que la derivada de $c_1(\mu) = F\big(\mu, x_1(\mu)\big)$ no se anula para $0 < \mu < 1/2$ y así c_1 será monótona. Utilizando que, por definición, en L_1 se cumple $\partial F/\partial x_1 = 0$, la anulación de $c_1'(\mu)$ implicaría

$$c_1'(\mu) = \frac{\partial F}{\partial \mu} + \frac{\partial F}{\partial x_1}\frac{dx_1}{d\mu} = \frac{\partial F}{\partial \mu} = -\frac{2(1+x_1)}{(x_1+\mu)^2} + \frac{2(1-x_1)}{(1-\mu-x_1)^2} = 0, \tag{19}$$

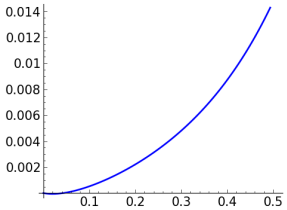
y de aquí

$$\mu = \frac{1}{2} + \frac{1}{2}T - \frac{2T}{1+T^2} \qquad \text{donde se ha escrito} \quad x_1 = \frac{2T}{1+T^2} \quad \text{con } 0 \le T \le 1. \tag{20}$$

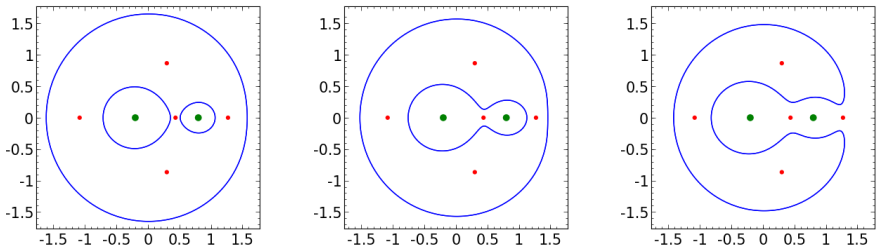
Sustituyendo en la primera ecuación de (10) con $x = x_1$ e $y = 0$ y extrayendo un factor $T(T-1)$, se llega una ecuación de tercer grado que no tiene solución en $T \in (0,1)$.

En [MD99, (3.94)] se menciona la aproximación asintótica $c_1(\mu) \sim 3 + (9\mu)^{2/3} - 10\mu/3$ para $\mu \to 0^+$ pero los siguientes datos muestran que en realidad es un muy buen sustituto de la fórmula no explícita (18) en todo el rango $0 \le \mu \le 1/2$:

$c_1(0.50)$	$= 4.00000$	$c_1(0.20)$	$= 3.80465$
	~ 4.05901		~ 3.81306
$c_1(0.40)$	$= 3.98091$	$c_1(0.10)$	$= 3.59695$
	~ 4.01559		~ 3.59884
$c_1(0.30)$	$= 3.92015$	$c_1(0.01)$	$= 3.16764$
	~ 3.93899		~ 3.16750



Si consideramos una curva de velocidad cero para un valor de la constante de Jacobi c_J mayor que c_1, el ocho se romperá en dos óvalos alrededor en m_1 y m_2. Si por el contrario, c_J es algo menor que c_1 entonces la "cintura" del ocho se ensanchará pero en este caso hay un límite: cuando c_J llegue al valor de c_2 correspondiente a la curva de velocidad cero que contiene a L_2, entonces ya dejará de haber dos componentes conexas.

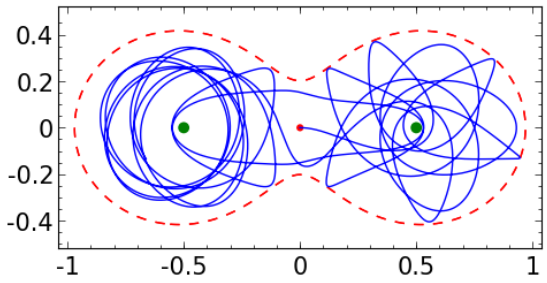


$c_1 < c_J$ $\qquad$ $c_2 < c_J < c_1$ $\qquad$ $c_J < c_2$

Las figuras corresponden a $\mu = 0.2$ que implica $c_1 = 3.8047$ y $c_2 = 3.5524$. Los valores elegidos de c_J son respectivamente 3.9, 3.7 y 3.5.

En los dos primeros casos tenemos órbitas confinadas alrededor de m_1 y m_2 porque, recordemos, las curvas de velocidad cero establecen fronteras para las regiones permitidas.

Ya una situación simétrica da una idea de la complejidad de los movimientos en el problema de los tres cuerpos, incluso en la versión restringida que manejamos aquí. Si tomamos $\mu = 1/2$ con $(x_0, y_0) = (0, 0)$ y $(v_{x_0}, v_{y_0}) = (0.5, 0)$ y esperamos cuatro órbitas de m_1 y m_2 entonces la trayectoria de m_3 es

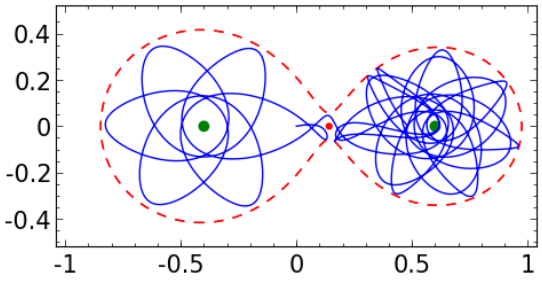


Lorb = [0.5,0.0,0.0, 0.5,0.0]
Lnum = [5.0, 0.0001]

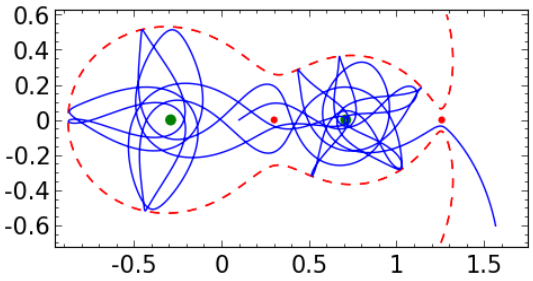
$c_1 = 4$, $c_J = 3.75$, $c_2 = 3.4568$

Si nos fijamos en la salida desde el punto inicial, que es el origen, observamos un desvío en la trayectoria hacia abajo a pesar de la velocidad inicial es horizontal. La explicación es que la simetría $y \leftrightarrow -y$ es solo aparente pues en el sistema inercial centro de masas m_1 y m_2 están rotando. La desviación se debe al término de Coriolis en (3).

Si acercamos c_J a c_1, la región permitida tenderá a estrangularse y la comunicación entre m_1 y m_2 será más inusual. Eso es lo que se trata de ilustrar con la primera de las siguientes figuras, en la que $c_1 - c_J = 0.021975$. Nótese el retroceso inicial debido a que v_{x_0} no es suficientemente grande.



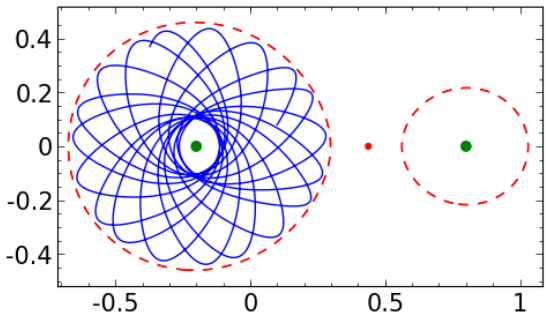
$(x_0, y_0) = (0, 0)$ $(v_{x_0}, v_{y_0}) = (0.6, 0.12)$
$\mu = 0.4$, $c_J = 3.958933$



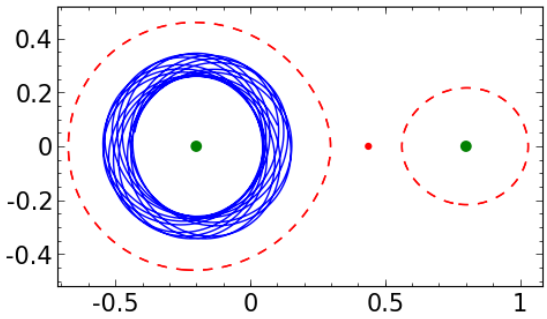
$(x_0, y_0) = (0.1, 0)$ $(v_{x_0}, v_{y_0}) = (0.85, 0.57)$
$\mu = 0.29$, $c_J = 3.558268$

Si, por el contrario, c_J es ligeramente menor que c_2, se abrirá un hueco hacia la región exterior no acotada y m_3 podría escapar de los alrededores de m_1 y m_2, como muestra la segunda figura en la que $c_2 - c_J = 0.003822$.

En el caso $c_1 < c_J$ el movimiento está confinado al óvalo que contiene a una de las masas siempre que la posición inicial esté en él. Tomemos para todos los ejemplos siguientes $\mu = 0.2$ y $c_J = 4.098992$. En este caso los dos óvalos tiene forma casi circular debido a que c_J y c_1 no están próximos. Por (15), la constante de Jacobi depende del módulo de la velocidad pero no de su dirección, sin embargo las órbitas, por supuesto, pueden ser bien distintas. Partiendo del mismo punto $(x_0, y_0) = (0.1, 0.08)$, para el tiempo de tres órbitas de m_1 y m_2 se obtiene, con las velocidades iniciales indicadas,

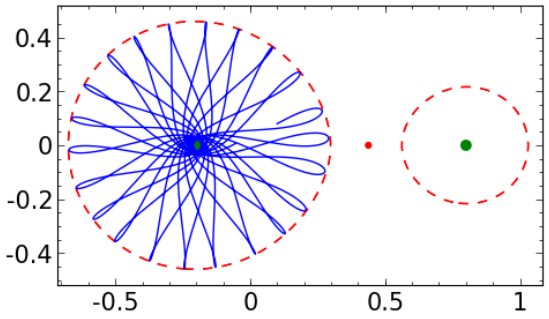


$(v_{x_0}, v_{y_0}) = (0.691587, 1.07708)$

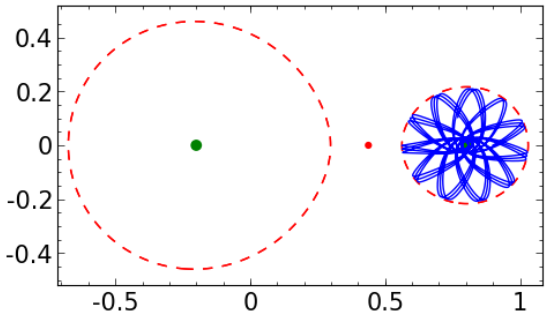


$(v_{x_0}, v_{y_0}) = (-0.532668, 1.16390)$

Para conseguir que m_3 esté confinada alrededor de m_2 podemos tomar un (x_0, y_0) cercano a $(1 - \mu, 0)$, la singularidad de U, y después compensar con una velocidad inicial adecuada en (15) para que c_J sea 4.098992 como antes. Por ejemplo, en la figura de la derecha se ha tomado $(x_0, y_0) = (0.9, 0)$ que habría dado la constante 6.264545 con velocidad inicial nula y que con $v_{x_0}^2 + v_{y_0}^2 = 2.16557$ se convierte en la c_J de más arriba.



$(v_{x_0}, v_{y_0}) = (1.12331, 0.613652)$



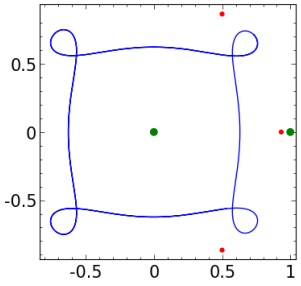
$(v_{x_0}, v_{y_0}) = (1.29144, 0.705515)$

La figura de la izquierda es un ejemplo más correspondiente a $(x_0, y_0) = (0.1, 0.08)$. Nótese que los puntos cuspidales están en el borde del óvalo, esto se debe a que en ellos $\dot{x} = \dot{y} = 0$ y entonces tienen que pertenecer a la curva de velocidad cero.

5. Algunas órbitas periódicas

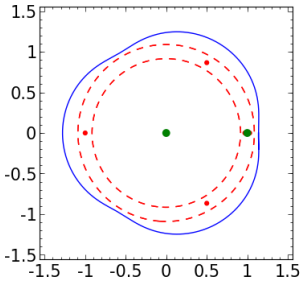
Resulta un poco chocante que en un sistema caótico como el de los tres cuerpos se puedan encontrar soluciones periódicas (y lo es mucho más que en 2000 se hallara una sencilla para el problema sin restricciones [CM00]) pero desde el punto de vista numérico no lo es tanto pensando en métodos de disparo, esto es, intentando por aproximaciones sucesivas lanzar m_3 de modo que la posición y velocidades finales se aproximen a las iniciales. Habitualmente se emplea la simetría $(t, x, y, \dot{x}, \dot{y}) \leftrightarrow (-t, x, -y, -\dot{x}, \dot{y})$ del problema, por ello las condiciones iniciales son a menudo de la forma $y_0 = v_{x_0} = 0$ [BM05].

En [Per17] se prueba una especie de ley de Kepler para las órbitas periódicas y entre los ejemplos que se mencionan allí representamos los dos siguientes:



$x_0 = 0.732035$
$y_0 = 0.713203$
$v_{x_0} = 0.112027$
$v_{y_0} = -0.114985$

$c_J = 2.974994$

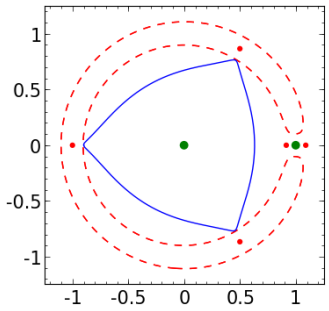


$x_0 = 1.1378$
$y_0 = 0$
$v_{x_0} = 0$
$v_{y_0} = -0.174265$

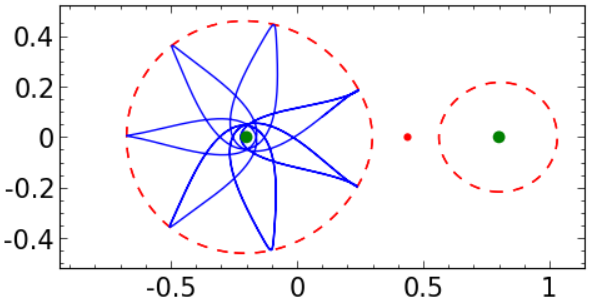
$c_J = 3.022873$

Los valores de μ son $9.53875 \cdot 10^{-4}$ en la primera figura, correspondiente al sistema Sol-Júpiter, y $7.80369 \cdot 10^{-5}$ en la segunda, correspondiente al sistema Júpiter-Ganímedes.

Los siguientes ejemplos, para $\mu = 0.002$ y $\mu = 0.2$, respectivamente, los encontramos al azar experimentando.



$(x_0, y_0) = (-0.9, 0)$
$(v_{x_0}, v_{y_0}) = (0, -0.058)$

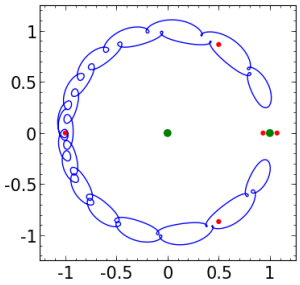


$(x_0, y_0) = (0.1, 0.08)$
$(v_{x_0}, v_{y_0}) = (1.0, 0.8)$

La curiosa estrella de siete puntas apareció mientras se estudiaban las órbitas confinadas.

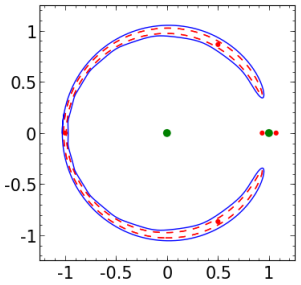
Entre las órbitas menos intuitivas en la variante del problema de los tres cuerpos que nos ocupa, están las llamadas *órbitas de herradura* (*horseshoe orbits*). Nos centramos aquí en el caso periódico. Una definición informal (cf. [BM05]) es que son aquellas con forma de "C" encerrando a L_3, L_4 y L_5 pero no a L_1 ni a L_2. Este fenómeno aparece para valores de μ pequeños y con c_J y c_1 cercanos a 3 (recordemos que $c_1 \geq 3$ es el valor de $2U$ en L_1).

Dos ejemplos, con $\mu = 9.53875 \cdot 10^{-4}$ correspondiente al sistema Sol-Júpiter, que aparecen en [Tay81] y están representadas en [MD99, Fig. 3.17], son:



$x_0 = -0.97668$
$y_0 = 0$
$v_{x_0} = 0$
$v_{y_0} = -0.06118$

$c_J = 2.99892672$



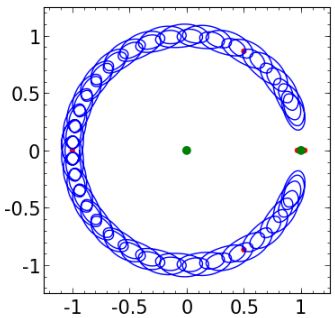
$x_0 = -1.02745$
$y_0 = 0$
$v_{x_0} = 0$
$v_{y_0} = 0.04032$

$c_J = 3.00148629$

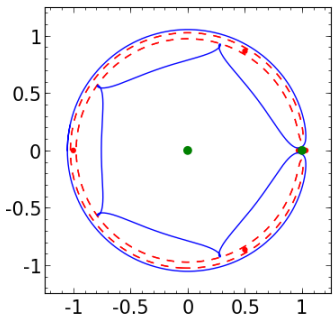
En el primer caso las curvas de velocidad cero se reducen prácticamente a L_4 y L_5, mientras que en el segundo, está próxima a la curva crítica que pasa por L_3, lo cual impide que haya oscilaciones como las del primer caso porque la curva de velocidad cero delimita una zona prohibida.

Estas curiosas órbitas no son una especulación teórica, de hecho el asteroide 2002 AA29 sigue una órbita de herradura en el sistema Sol-Tierra. Lo que vería un observador inercial es que m_3 sigue órbitas más o menos circulares pero que la cabo de un número suficientemente grande de órbitas, su distancia angular a m_2 ha variado ostensiblemente hasta poder estar bastante cerca sin ser capturado. En [MD81] se introducen fórmulas aproximadas cuando μ es muy pequeño para esta distancia mínima y para el grosos de la "herradura".

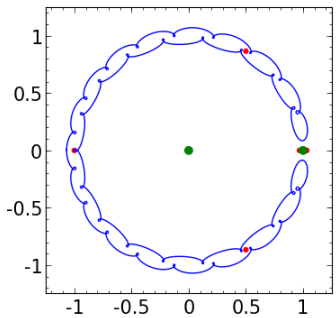
El trabajo [BM05] está dedicado a producir familias de órbitas de herradura. De entre las 27 que se muestran para $\mu = 10^{-4}$, destacamos las tres siguientes:



$(x_0, y_0) = (-0.93449350, 0)$
$(v_{x_0}, v_{y_0}) = (0, -0.14032108)$



$(x_0, y_0) = (-1.05221706, 0)$
$(v_{x_0}, v_{y_0}) = (0, 0.07736088)$



$(x_0, y_0) = (-1.06764576, 0)$
$(v_{x_0}, v_{y_0}) = (0, 0.11629385)$

Los valores de la constante de Jacobi c_J y de c_1 y c_2 (el valor de $2U$ en L_2) están recogidos en la tabla:

c_1	3.00898924
c_J	2.99390329
c_2	3.00885590

c_1	3.00898924
c_J	3.00201256
c_2	3.00885590

c_1	3.00898924
c_J	2.99970858
c_2	3.00885590

Como habíamos anticipado, son valores próximos entre sí y a tres.

6. Animaciones en el sistema inercial centro de masas

Las imágenes que hemos mostrado hasta ahora han sido siempre en el sistema de referencia rotatorio. En esta sección se explica cómo crear fotogramas en el sistema de referencia estático e integrarlos en un fichero GIF animado que permite una visualización cómoda de diversas situaciones relativas al caso particular del problema de los tres cuerpos que estamos estudiando. La función que lleva a cabo el proceso es `make_animation`, definida como

```
def make_animation(Lorb, Lnum, Lplo, nfr, dela=100, f_lag = False):
        Ldat = retrieve_data_frames(Lorb, Lnum, nfr)
        make_frames( Lorb[0], Ldat, Lplo, f_lag)
        filen = os.path.splitext(Lplo[-1])[0]
        subprocess.call('./animate ' + str(dela) + ' ' + filen , shell=True)
```

Los tres primeros argumentos son los datos orbitales, del algoritmo numérico y de la representación que ya han aparecido con anterioridad. Hay un argumento obligatorio más, `nfr` que es el número de fotogramas. Finalmente, están los argumentos `dela` y `f_lag`, el primero es el tiempo en milisegundos entre fotogramas y el segundo tiene la información de si se representan los ejes y los puntos de Lagrange dependiendo respectivamente de si los bits 1 y 0 están activos o no. Es decir, `f_lag=3` pinta ambos y `f_lag=2` pinta los ejes pero no los puntos de Lagrange.

Con `retrieve_data_frames` se llama al programa C que hace los cálculos con el método de Runge-Kutta y del fichero de datos resultante se eligen `nfr` de ellos. La parte relevante del código es:

```
k = 0
#retrieve only nfr values
for nf in range(nfr):
        ind = floor(nf*nsteps/nfr)
        while(k!=ind):
                line = f.readline()
                k +=1
        line = f.readline()
```

La fórmula para `ind` asegura que las `nfr` posiciones que indican los datos recuperados estén uniformemente distribuidos en el tiempo (aquí `nsteps` es el número de líneas del fichero de datos). La salida de `retrieve_data_frames` es una lista de `nfr` pares dados por tiempos y

posiciones en el sistema no rotado. Esta lista la utiliza `make_frames` creando `nfr` imágenes temporales `temp_frame*.png` con un punto azul en la posición de m_3 tomada de `Ldat` y dos puntos verdes en las posiciones de m_1 y m_2. Además, `make_frames` crea en el fichero indicado en `Lplo` una superposición de todos los fotogramas. Finalmente, se llama al *script* de Shell `animate` que a través de la *suite* de código abierto ImageMagick compone el GIF animado con el nombre tomado del último elemento de `L_plo`, salvo que la extensión se modifica a `.gif`. El contenido del *script* es

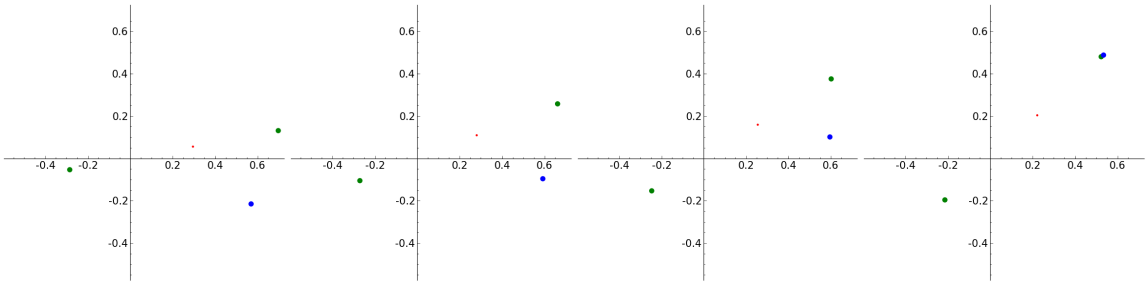
```
#!/bin/bash

convert -delay $1 -loop 1 temp_frame*.png  -deconstruct -layers optimize $2.gif
rm temp_frame*.png
du -h $2.gif
```

Los modificadores `-deconstruct -layers optimize` son importantes para crear ficheros de tamaño reducido.

Como ejemplo, ejecutando

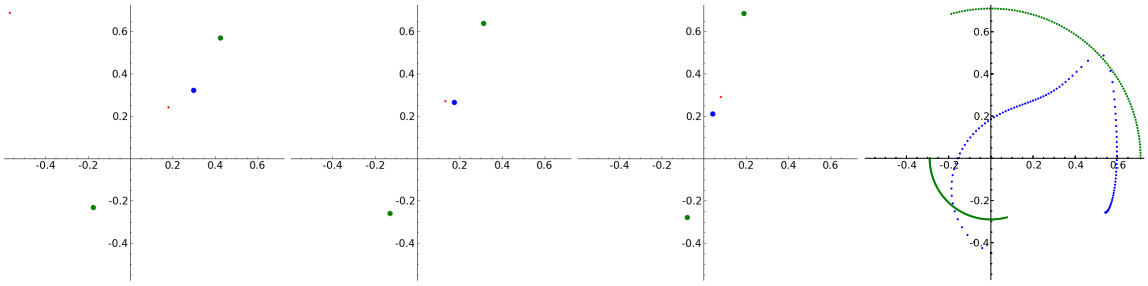
```
Lorb = [0.29,     0.5427008, -0.2574309,   -0.1441290, -0.5512641]
Lnum = [0.2955,  0.0001]
Lplo = [-0.57,0.7,-0.55,0.7,'../../images/frames.eps']
make_animation(Lorb, Lnum, Lplo, 100, 20, 3)
```

tendremos la animación de un *tirón gravitatorio* (*gravitational slingshot*) por parte de m_2 que lanza m_3 hacia m_1. En `http://www.uam.es/fernando.chamizo/oscuro/images/gravsl.gif` está el fichero resultante. Para mostrar algunas imágenes aquí, podemos modificar `animate` de modo que no borre los fotogramas temporales. Para que el tirón gravitatorio sea los suficientemente fuerte como para cambiar totalmente la trayectoria de m_3, tiene que pasar rozando a m_3 y eso es lo que vemos en el último de los siguientes cuatro fotogramas, donde m_3 (azul), casi coincide con m_2 (verde).



Después m_3 se dirige hacia m_1 y la orbitará cierto número de veces. Mostrando tan pocos fotogramas es difícil hacerse una idea del fenómeno, sobre todo teniendo en cuenta la gran

velocidad que adquiere m_3 cuando está cerca de m_2, por ello resulta ilustrativa la última imagen que incluye las posiciones de las tres masas a lo largo de los cien fotogramas.



La separación entre los puntos azules en esta última imagen, da una idea de la velocidad de m_2 que hemos mencionado. Según se acerca al punto del tirón los puntos se van espaciando, tanto que 100 fotogramas se vuelven pocos para ubicarlo bien. Después m_3 sigue bajo la influencia gravitatoria de m_2 que se va alejando por la parte superior, lo que causa en cambio en la curvatura de la trayectoria. En los momentos a medio camino entre la influencia de m_1 y m_2 la velocidad disminuye y después los puntos vuelven a espaciarse cuando m_3 entra claramente bajo la acción de m_1.

También hemos hecho un script para realizar vídeos con el software libre `ffmpeg` a partir de los fotogramas generados por `make_animation`.

```
#!/bin/bash

# Comment 'rm temp_frame*.png' in animate and
# run 'make_animation(Lorb, Lnum, Lplo, 300, 25, 3)'
rm output1.mp4
ffmpeg -framerate 24 -start_number 0 -i temp_frame%03d.png output1.mp4
rm temp_playlist.txt
for i in {1..2}; do printf "file '%s'\n" output1.mp4 >> temp_playlist.txt; done
rm output.mp4
ffmpeg -f concat -i temp_playlist.txt  -c copy output.mp4
rm output1.mp4
echo '  '
du -h output.mp4
```

Un ejemplo correspondiente a una de las órbitas periódicas (la de forma de triángulo) de la sección anterior se puede descargar en `http://www.uam.es/fernando.chamizo/oscuro/images/output.mp4`. El `2` en el bucle `for` indica el número de periodos. A pesar de que $\mu = 0.002$ es pequeño, con un poco de atención se observan pequeños desplazamientos de la masa central.

Referencias

[BM05] E. Barrabés and S. Mikkola. Families of periodic horseshoe orbits in the restricted three-body problem. *Astron. Astrophys.*, 432:1115–1129, mar 2005.

[CM00] A. Chenciner and R. Montgomery. A remarkable periodic solution of the three-body problem in the case of equal masses. *Ann. of Math. (2)*, 152(3):881–901, 2000.

[Cor98] N. J. Cornish. The Lagrange points. 1998. WMAP Education and Outreach https://map.gsfc.nasa.gov/ContentMedia/lagrange.pdf.

[Dan88] J. M. A. Danby. *Fundamentals of celestial mechanics*. Willmann-Bell, Inc., Richmond, VA, second edition, 1988.

[MD81] C. D. Murray and S. F. Dermott. The dynamics of tadpole and horseshoe orbits. I - Theory. *Icarus*, 48:1–22, October 1981.

[MD99] C. D. Murray and S. F. Dermott. *Solar system dynamics*. Cambridge University Press, Cambridge, 1999.

[Per17] O. Perdomo. On the period of the periodic orbits of the restricted three body problem. *Celestial Mech. Dynam. Astronom.*, doi 10.1007/s10569-017-9766-8, 2017.

[SB02] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 2002. Translated from the German by R. Bartels, W. Gautschi and C. Witzgall.

[Sic10] B. Sicardy. Stability of the triangular Lagrange points beyond Gascheau's value. *Celestial Mech. Dynam. Astronom.*, 107(1-2):145–155, 2010.

[Tao16] M. Tao. Explicit symplectic approximation of nonseparable Hamiltonians: Algorithm and long time performance. *Phys. Rev. E (3)*, 94:043303, 2016.

[Tay81] D. B. Taylor. Horseshoe periodic orbits in the restricted problem of three bodies for a sun-Jupiter mass ratio. *Astron. Astrophys.*, 103:288–294, nov 1981.

[Ver09] C. Verzijl. Notes on the r3bp project. https://tinyurl.com/lhezete, 2009.