57-COMPL-planeta-cython-sympl

December 3, 2017

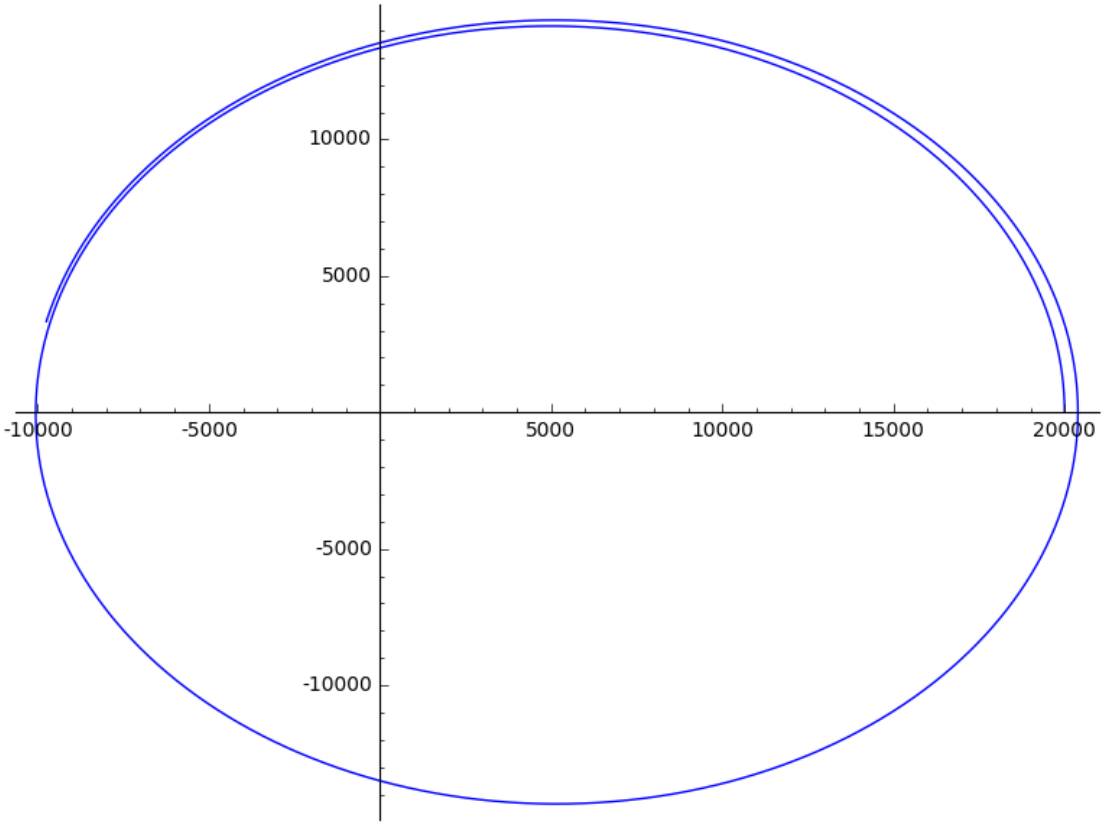
In [1]:
```
from math import *
```

In [2]:
```
var('x y vx vy m G')

        def orbita(x,y,vx,vy,m,G,dt,n_iter):
                L = [(x,y)]
                for j in range(n_iter):
                    distancia = sqrt(x**2+y**2)
                    Fx = -x*G*m/distancia**3    #Ley de Gravitacion de Newton-La constante G i
                    Fy = -y*G*m/distancia**3
                    x += dt * vx              #Actualizamos la posicion usando la definicion de
                    y += dt * vy
                    vx += dt * Fx / m         #Actualizamos la velocidad usando la segunda Ley
                    vy += dt * Fy / m
                    L.append((x,y))
                return L
```

In [3]:
```
%time SOL = orbita(20000.0,0.0,0.0,0.01,1.0,3.0,1000,10000)
        line2d(SOL)
```

```
CPU times: user 204 ms, sys: 28 ms, total: 232 ms
Wall time: 208 ms
```

Out[3]:

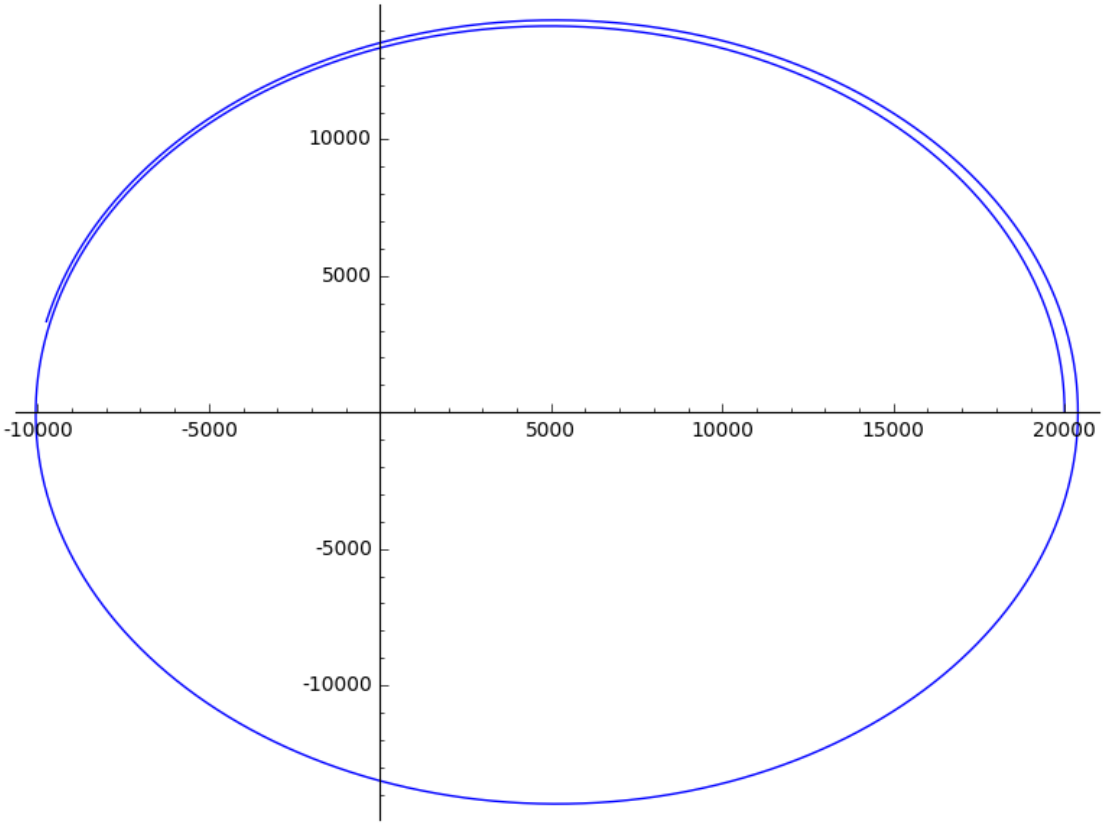


In [5]:
```
%%cython
from libc.math cimport sqrt
def orbita_cy(double x,double y,double vx,double vy,double m,double G,double dt,int n_
        cdef long j
        L = [(x,y)]
        for j in range(n_iter):
            distancia = sqrt(x**2+y**2)
            Fx = -x*G*m/distancia**3
            Fy = -y*G*m/distancia**3
            x += dt*vx
            y += dt*vy
            vx += dt*Fx/m
            vy += dt*Fy/m
            L.append((x,y))
        return L
```

In [6]:
```
%time SOL = orbita_cy(20000.0,0.0,0.0,0.01,1.0,3.0,1000,10000)
line2d(SOL)
```

```
CPU times: user 12 ms, sys: 4 ms, total: 16 ms
Wall time: 14.2 ms
```

Out[6]:



Comentarios:

Estamos resolviendo, en forma numérica, la ecuación diferencial de Newton $F(\mathbf{r}(t)) = m\frac{d^2\mathbf{r}(t)}{d^2t}$, con F el vector de fuerza dado por la ley de gravitación universal de Newton y $\mathbf{r}(t)$ el vector de posición del planeta con respecto al Sol en el origen. Como la ecuación diferencial es una igualdad entre dos vectores, se puede estudiar igualando componente a componente, y eso es lo que se hace en el programa.

La órbita del planeta no se cierra debido a que el método por el que se resuelve la ecuación diferencial es muy primitivo y se producen errores que se acumulan. Uno de los problemas fundamentales del Cálculo Numérico trata de obtener buenos métodos para la resolución numérica de las ecuaciones diferenciales.

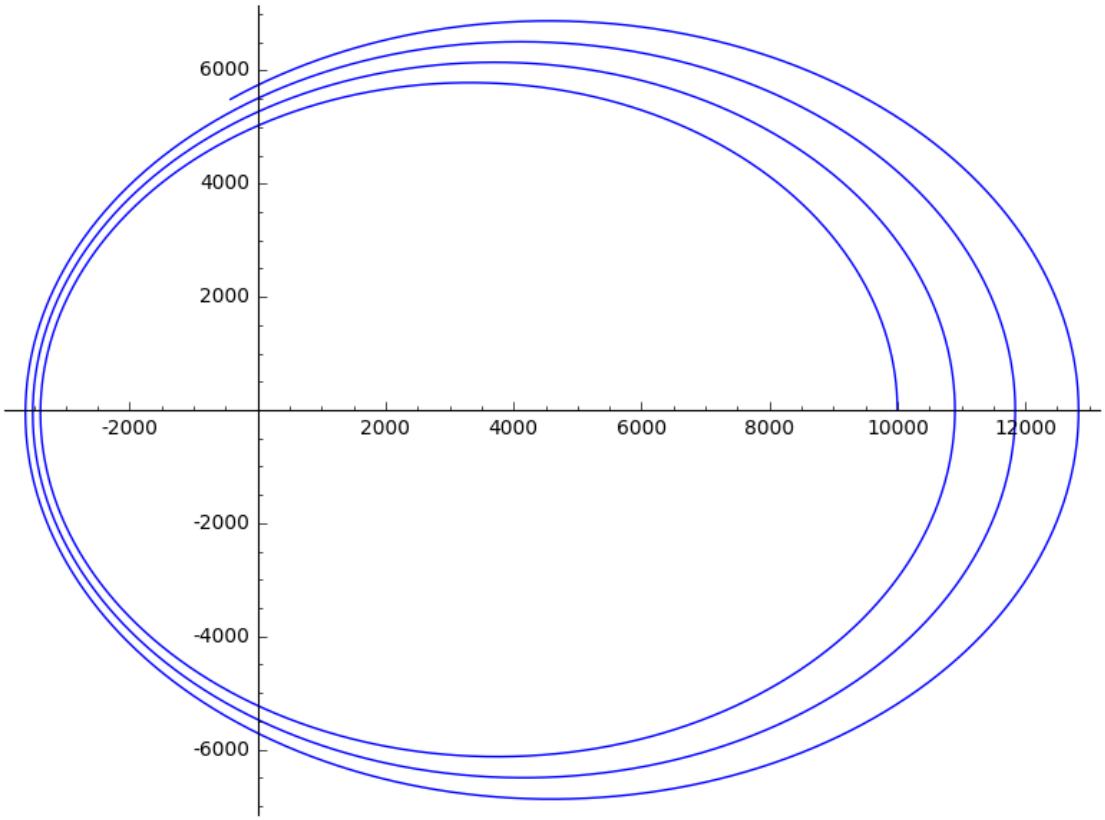
Con estos parámetros, la órbita es prácticamente circular con el Sol en el centro. £Cómo variar alguno de los parámetros para obtener algo más parecido a elipses con el Sol en un foco?

En este ejemplo las mejoras obtenidas con Cython no son muy relevantes porque usando Sage el tiempo es menor que 25 centésimas de segundo. Aún así la mejora es en un factor mayor que 25.

```
In [7]: %time SOL = orbita_cy(10000.0,0.0,0.0,0.01,1.0,2.0,1000,10000)
        line2d(SOL)
```

```
CPU times: user 8 ms, sys: 0 ns, total: 8 ms
Wall time: 9.63 ms
```

Out[7]:



0.1 Integrador simpléctico

In [8]: %%cython

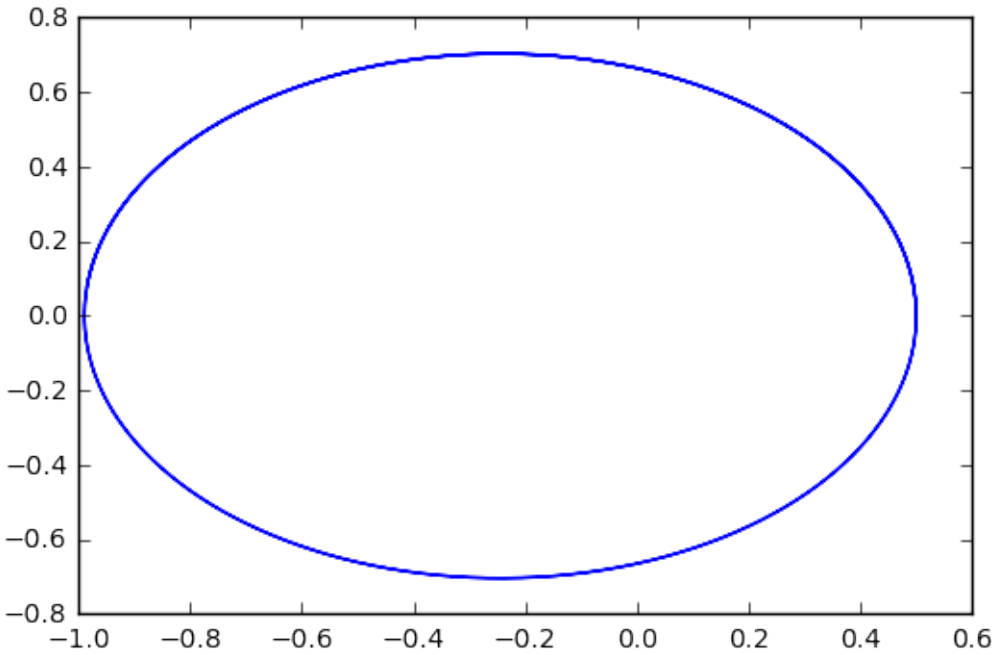
```
import numpy as np
cimport numpy as np

cimport cython

@cython.boundscheck(False)
@cython.wraparound(False)
@cython.cdivision(True)
def symplectic(int runs=10**6,np.float64_t e=0.00001):
    cdef np.ndarray[np.float64_t,ndim=2]r = np.empty((runs,2),dtype=np.float64)
    cdef np.ndarray[np.float64_t,ndim=2] p = np.empty((runs,2),dtype=np.float64)
```

```
    cdef np.float64_t[:,:] rv = r
    cdef np.float64_t[:,:] pv = p

    cdef int i,j

    r0 = [0.5,0]
    p0 = [0,1.63]

    for j in range(r.shape[1]):
        rv[0,j] = r0[j]
        pv[0,j] = p0[j]

    for i in range(1,runs):
        for j in range(r.shape[1]):
            pv[i,j] = pv[i - 1,j] - e * (rv[i - 1,0]**2 + rv[i - 1,1]**2)**(-1.5)*rv[i
            rv[i,j] = rv[i - 1,j] + e * pv[i - 1,j]

    return r,p
```

In [9]:
```
R,P = symplectic()
```

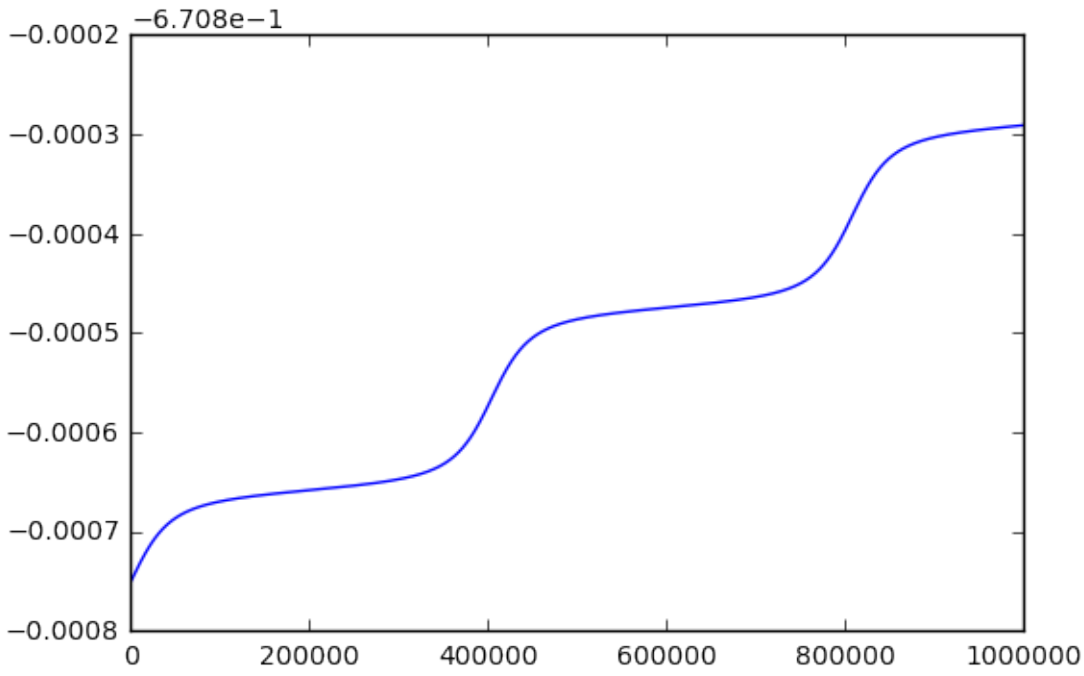
In [10]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(*R.T)
```

Out[10]:
```
[<matplotlib.lines.Line2D object at 0x7f3b98558550>]
```



In [11]:
```
# Energia
plt.plot([(P[i]**2).sum() / 2 - 1/(np.sqrt((R[i]**2).sum())) for i in range(R.shape[0]
```

Out[11]: [<matplotlib.lines.Line2D object at 0x7f3ba4284bd0>]



In [ ]: