Universidad Nacional San Agustin de Arequipa

Facultad de Ingenierias de Produccion y Servicios

Escuela Profesional de Ingenieria de Sistemas

Fisica Computacional

Alumno:

Fuentes Paredes Nelson Alejandro

Mayo 2020

[ ]:

```

```

[1]:

```
%matplotlib inline
#%matplotlib notebook
```

1 Importando Librerias

[2]:

```
import numpy as np
import matplotlib.pyplot as plt
import math
import time
from mpl_toolkits.mplot3d.axes3d import get_test_data
```

2 Movimiento circular

2.1 Dibuje la trayectoria para a = 2 m/s2, r = 8 m, m = 5 kg, h = 0.1, con las condiciones iniciales

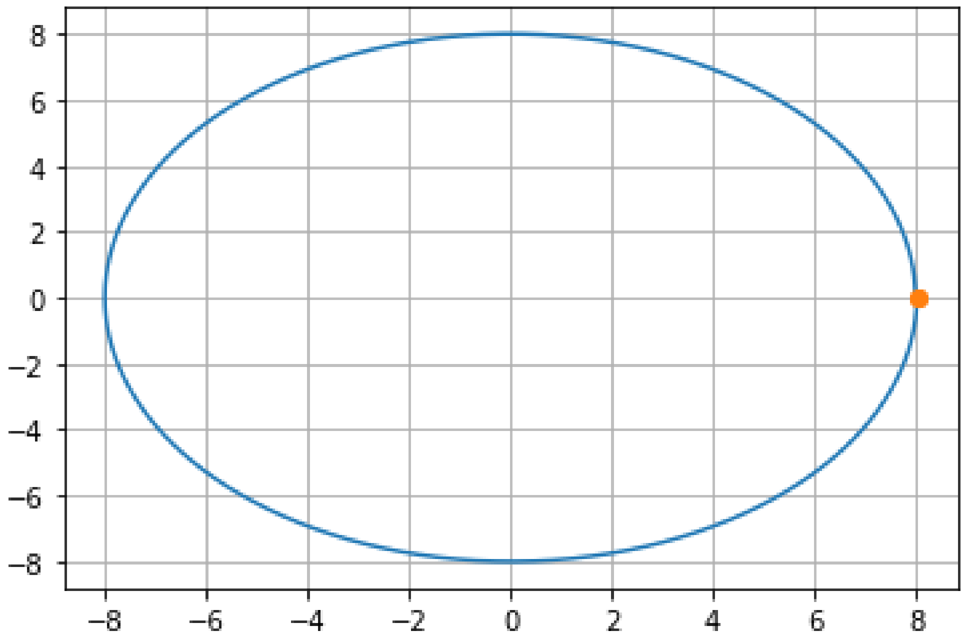
[3]:

```
a = 2
r = 8
m = 5
h = 0.001
```

(a) x0 = r, y0 = 0, vx0 = 0, vy0 = +4 m/s

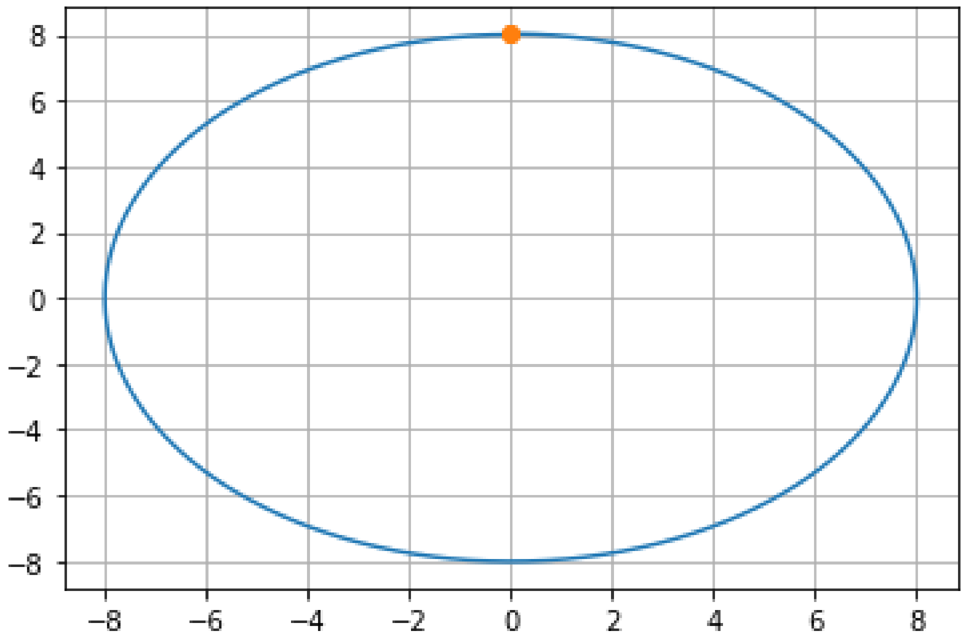
[4]:

```
p0 = np.array([ r , 0 ])
v0 = np.array([ 0 , 4 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
fig = plt.figure()
plt.plot(xs, ys)
plt.plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
#plt.rcParams ['figure.figsize'] = [500, 500]
plt.grid()
```



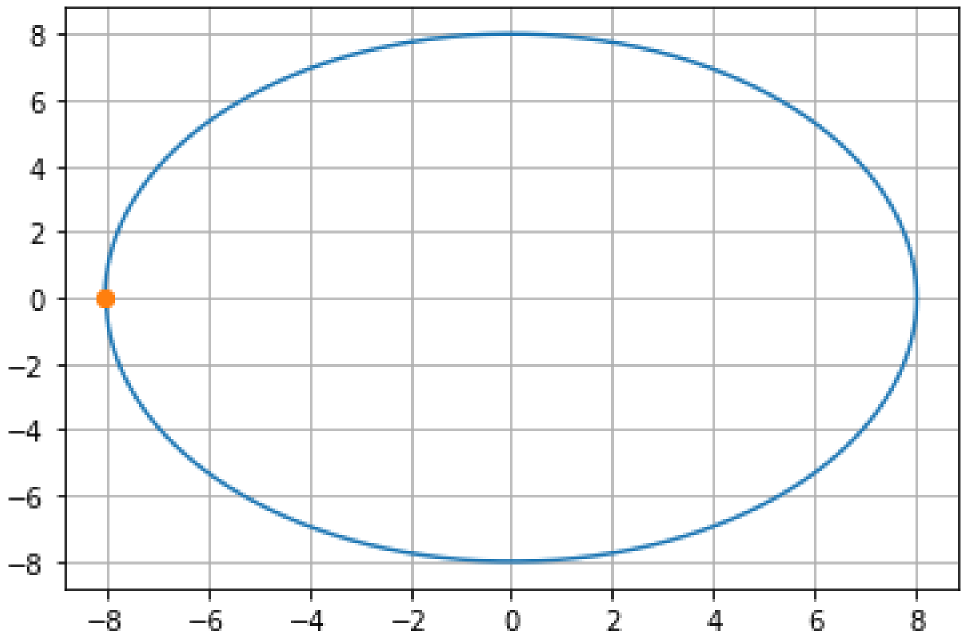
(b) x0 = 0, y0 = r, vx0 = −4 m/s, vy0 = 0

[5]:
```
p0 = np.array([ 0 , r ])
v0 = np.array([ -4 , 0 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
fig = plt.figure()
plt.plot(xs, ys)
plt.plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
plt.grid()
```



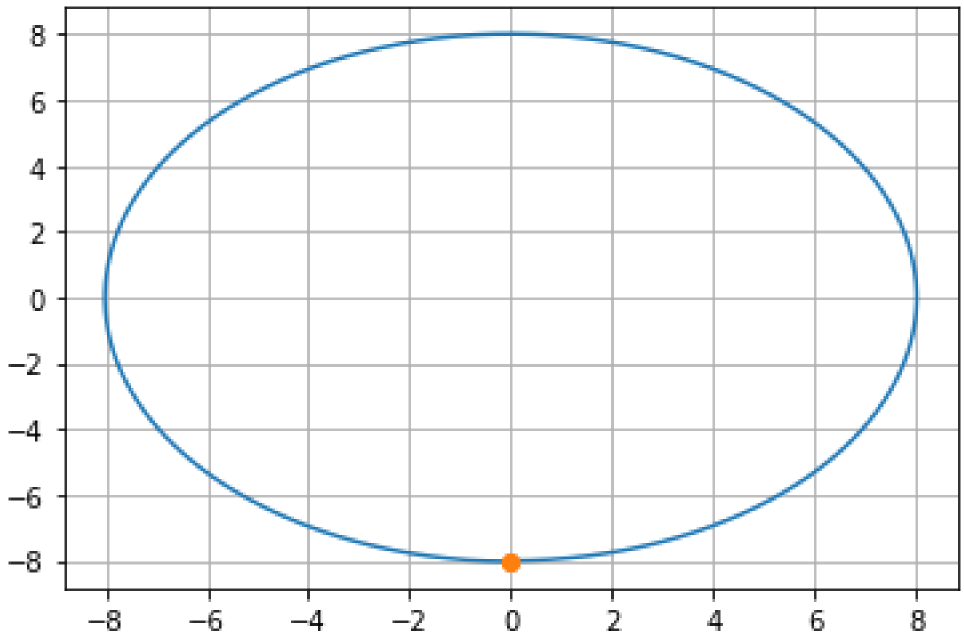
(c) x0 = −r, y0 = 0, vx0 = 0, vy0 = −4 m/s

[6]:
```
p0 = np.array([ -r , 0 ])
v0 = np.array([ 0 , -4 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
fig = plt.figure()
plt.plot(xs, ys)
plt.plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
plt.grid()
```



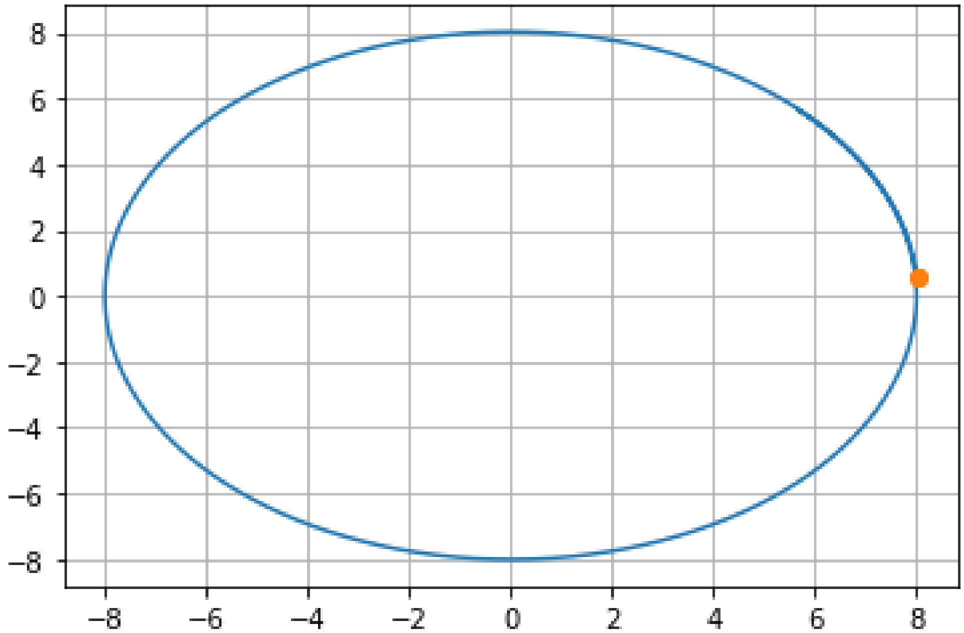
(d)x0 = 0, y0 = −r, vx0 = +4, vy0 = 0 m/s

[7]:
```
p0 = np.array([ 0 , -r ])
v0 = np.array([ 4 , 0 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
fig = plt.figure()
plt.plot(xs, ys)
plt.plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
plt.grid()
```



(e) cuando la partícula inicialmente forma un ángulo de /4

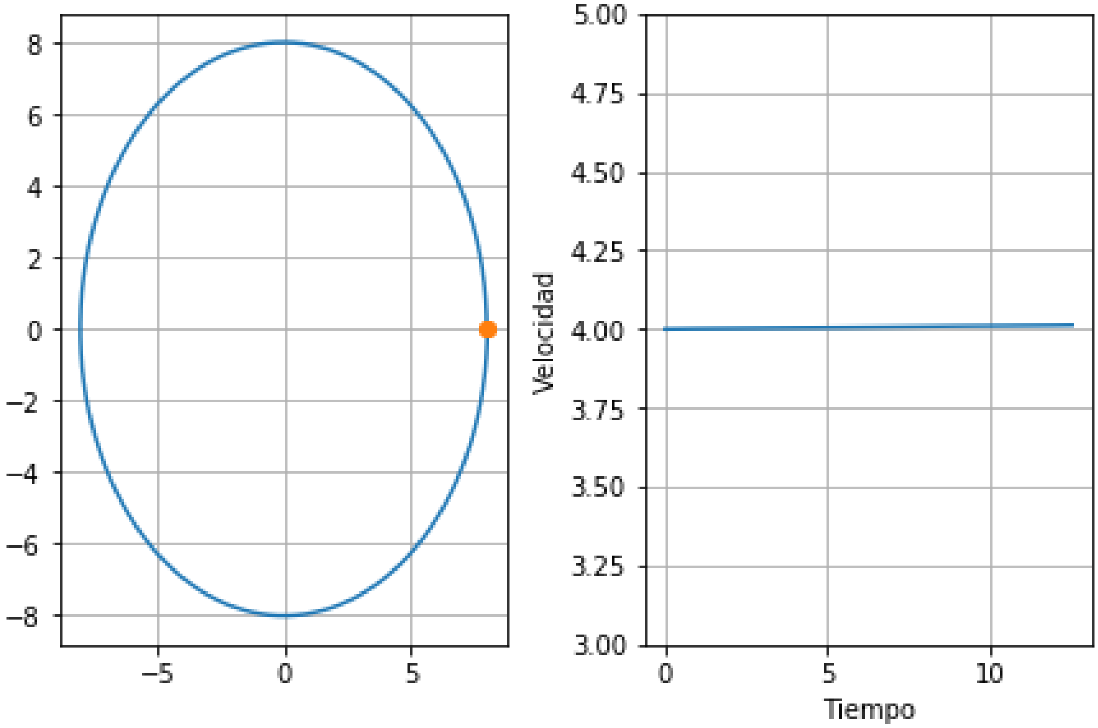
[8]:
```
fig = plt.figure()
p0 = np.array([ math.cos(math.pi/4)*r , math.sin(math.pi/4)*r ])
v0 = np.array([ 4* math.cos(math.pi/4), -4*math.sin(math.pi/4)])
a0 = np.array([ 0 , 0 ])



pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
for i in np.arange(0 , 14, h):
    a0 = p0*(-a/r)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])

plt.plot(xs, ys)
plt.plot([p0[0], p0[0]], [p0[1], p0[1]], 'o')
plt.grid()
```



(f) Pero como at $= 0$, entonces la velocidad es constante v, es decir demuestre que es constante por cada paso del tiempo

[9]:
```
p0 = np.array([ 8 , 0 ])
v0 = np.array([ 0 , 4 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
t = [ 0 ]

vs = [ math.sqrt(v0[0]**2 + v0[1]**2) ]


for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
    t.append(i+h)
    v = math.sqrt(v0[0]**2 + v0[1]**2)
    vs.append(v)
```

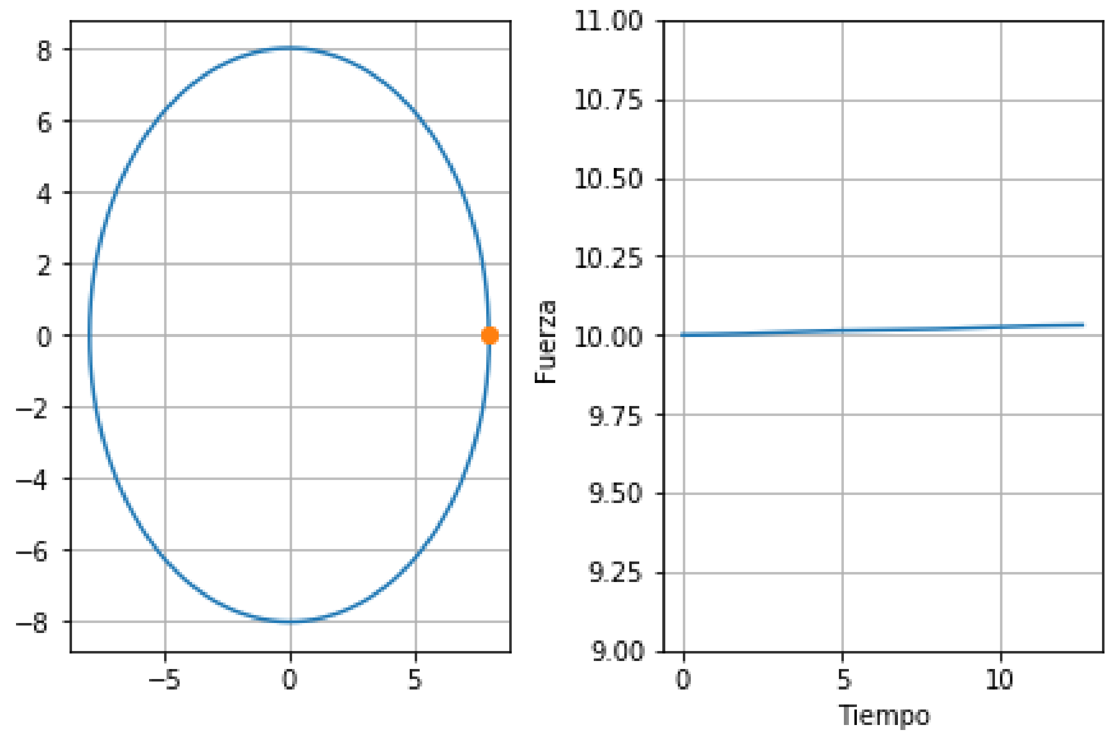
```python
fig, axis = plt.subplots(1, 2, constrained_layout=True)
axis[0].plot(xs, ys)
axis[0].plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
axis[0].grid()

axis[1].plot(t, vs)
axis[1].set(xlabel='Tiempo', ylabel='Velocidad', ylim=(3, 5))
axis[1].grid()
```



(g) Evalúe la fuerza centrípeta Fc por cada paso del tiempo

[10]:
```python
p0 = np.array([ 8 , 0 ])
v0 = np.array([ 0 , 4 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
t = [ 0 ]

fs = []

for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
```

```
    fs.append(math.sqrt(a0[0]**2 + a0[1]**2)*m)
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
    t.append(i+h)

fs.append(math.sqrt(a0[0]**2 + a0[1]**2)*m)
fig, axis = plt.subplots(1, 2, constrained_layout=True)
axis[0].plot(xs, ys)
axis[0].plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
axis[0].grid()

axis[1].plot(t, fs)
axis[1].set(xlabel='Tiempo', ylabel='Fuerza', ylim=(9,11))
axis[1].grid()
```



(h) Utilice análisis dimensional, para saber que ac depende de v y r, tal como indica las condiciones iniciales, es decir

$$\frac{L}{T^2} = L^x * \frac{L^y}{T^y} \tag{1}$$

Donde Lx representa el an´alisis dimensional de r y Ly/Ty representa el an´alisis dimensional de

v. Encuentre x y y.

$$x = R * \cos\theta; \tag{2}$$

$$\Delta x = V_x = -R * \sin\theta * \frac{(\Delta\theta)}{t} \tag{3}$$

$$\Delta V_x = a_x = -R * \cos\theta * (\frac{(\Delta\theta)}{t})^2 \tag{4}$$

$$a_x = -x * (\frac{V_x}{x})^2 \tag{5}$$

$$x = \frac{V_x^2}{a_x} \tag{6}$$

$$y = R * \sin\theta; \tag{7}$$

$$\Delta y = V_y = R * \cos\theta * \frac{(\Delta\theta)}{t} \tag{8}$$

$$\Delta V_y = a_y = -R * \sin\theta * (\frac{(\Delta\theta)}{t})^2 \tag{9}$$

$$a_y = -y * (\frac{V_y}{y})^2 \tag{10}$$

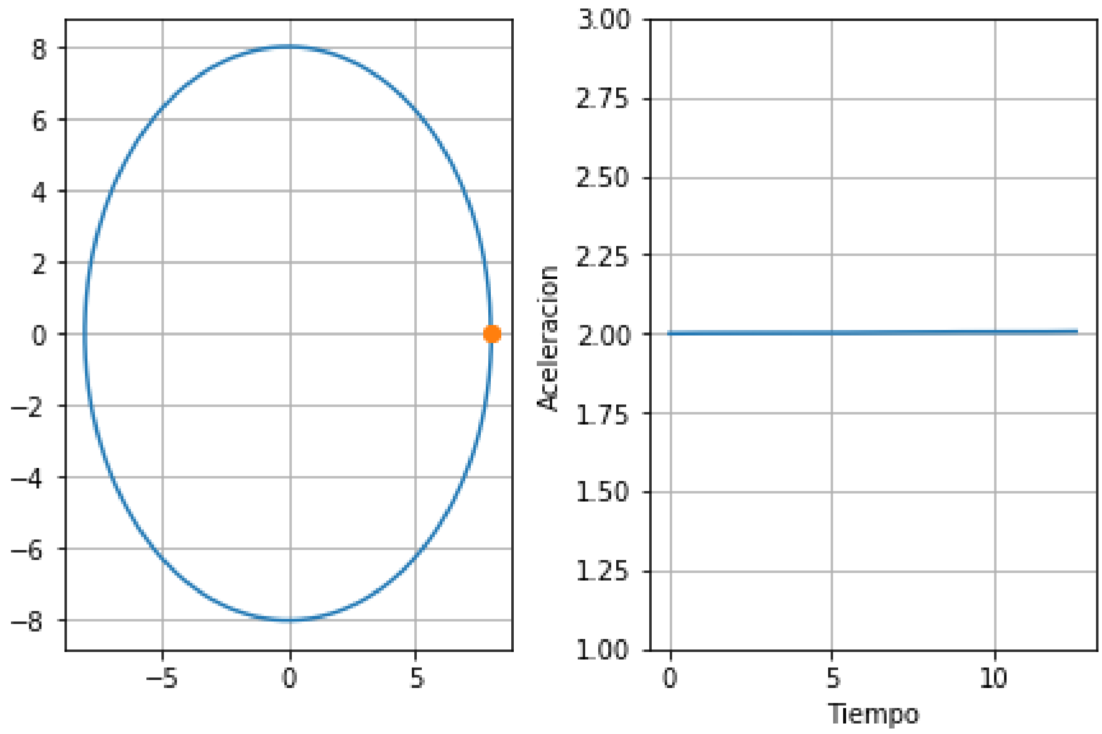
$$y = \frac{V_y^2}{a_y} \tag{11}$$

$$R = \frac{V^2}{a} \tag{12}$$

(i) Con esta condición podria encontrar una fórmula para la ac?. Con esta relación demuestre que ac = 2 m/s2 por cada paso del tiempo es constante, utilizando la fórmula que ha deducido

[11]:
```
p0 = np.array([ 8 , 0 ])
v0 = np.array([ 0 , 4 ])
a0 = np.array([ 0 , 0 ])
pf = p0
xs = [p0[0] ]
ys = [p0[1] ]
t = [ 0 ]

as_ = []
```

```
for i in np.arange(0 , 12.5625, h):
    a0 = p0*(-a/r)
    as_.append((v0[0]**2 + v0[1]**2)/math.sqrt(pf[0]**2 + pf[1]**2))
    p0 = pf
    pf = pf + v0*h
    v0 = v0 + a0*h
    xs.append(pf[0])
    ys.append(pf[1])
    t.append(i+h)

as_.append((v0[0]**2 + v0[1]**2)/math.sqrt(pf[0]**2 + pf[1]**2))
fig, axis = plt.subplots(1, 2, constrained_layout=True)
axis[0].plot(xs, ys)
axis[0].plot([pf[0], pf[0]], [pf[1], pf[1]], 'o')
axis[0].grid()

axis[1].plot(t, as_)
axis[1].set(xlabel='Tiempo', ylabel='Aceleracion', ylim=(1,3))
axis[1].grid()
```



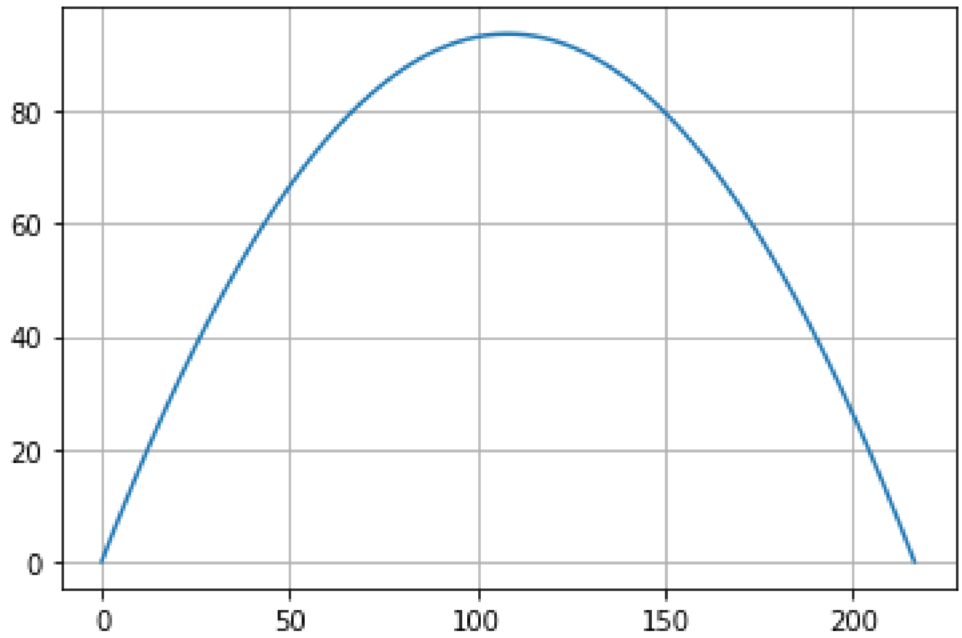
3 Movimiento de proyectiles cuando se presenta la resistencia del aire

3.0.1 Sea un pelota de beisboll con velocidad inicial vi = 50 m/s que se lanza con un ángulo de 60 ∘ con la horizontal. Dicha pelota tiene su coeficiente de arrastre C = 0.5, una masa m = 0.145 kg, un radio r = 0.0367 m y densidad del aire = 1.2 kg/m3.

[12]:
```
v = 50
angle = math.pi*2/3
C = 0.5
m = 0.145
r = 0.0367
A = math.pi * (r ** 2)
p = 1.2
```

(a) Haga la trayectoria de la pelota de beisboll, sin la fuerza de arrastre, hasta que la pelota llegue al suelo.

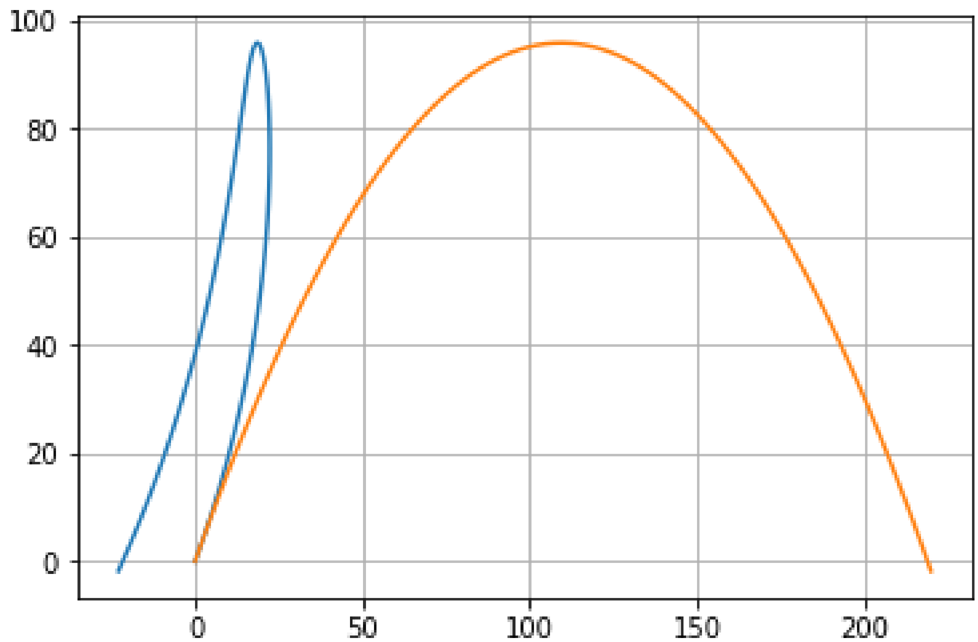
[13]:
```
k = 0#0.5*C*A*p/m
a0 = np.array([0, -10])
v0 = np.array([-math.cos(angle)*v , math.sin(angle)*v])
p0 = np.array([0 , 0])
h=0.00001
x0 = [ p0[0] ]
y0 = [ p0[1] ]

while p0[1] >=0:
    a0 = np.array([- k * math.sqrt( a0[0]**2 + a0[1]**2  ), -10])
    p0 = p0 + v0 * h
    v0 = v0 + a0 * h
    x0.append(p0[0])
    y0.append(p0[1])
fig = plt.figure()
plt.plot(x0, y0)
plt.grid()
```



(b) Haga dos trayectorias simultáneas de la pelota, una sin fuerza de arrastre y la otra con la fuerza de arrastre hasta que la pelota llegue al suelo.

[14]:
```
k0 = 0.5*C*A*p/m
a0 = np.array([0, 0])
v0 = np.array([-math.cos(angle) * v , math.sin(angle)*v])
p0 = np.array([0 , 0])
h=0.1
x0 = [ p0[0] ]
y0 = [ p0[1] ]



k1 = 0
a1 = np.array([0, 0])
v1 = np.array([-math.cos(angle)*v , math.sin(angle)*v])
p1 = np.array([0 , 0])

x1 = [ p1[0] ]
y1 = [ p1[1] ]

while p0[1] >=0:
    v2_0 = (v0[0]**2 + v0[1]**2)
    a0 = np.array([- k0 *  v2_0  , -10])
    p0 = p0 + v0 * h
    v0 = v0 + a0 * h
    x0.append(p0[0])
```

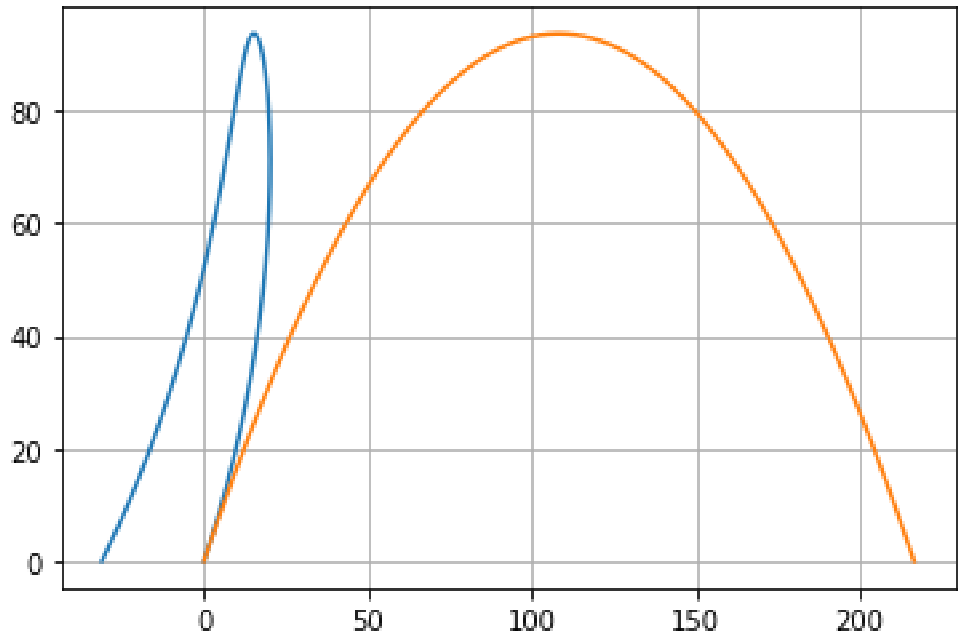
```
    y0.append(p0[1])


    v2_1 = (v1[0]**2 + v1[1]**2)
    a1 = np.array([- k1 *  v2_1 , -10])
    p1 = p1 + v1 * h
    v1 = v1 + a1 * h
    x1.append(p1[0])
    y1.append(p1[1])
fig = plt.figure()
plt.plot(x0, y0)
plt.plot(x1, y1)
plt.grid()
```



(c) Busque en internet la masa y radio de una pelota de golf y repita el mismo procedimiento 2

[15]:
```
v = 50
angle = math.pi*2/3
C = 0.5
m = 0.045935
r = 0.021335
A = math.pi * (r ** 2)
p = 1.2
```

```
k0 = 0.5*C*A*p/m
a0 = np.array([0, 0])
v0 = np.array([-math.cos(angle) * v , math.sin(angle)*v])
p0 = np.array([0 , 0])
h=0.0001
x0 = [ p0[0] ]
y0 = [ p0[1] ]


k1 = 0
a1 = np.array([0, 0])
v1 = np.array([-math.cos(angle)*v , math.sin(angle)*v])
p1 = np.array([0 , 0])

x1 = [ p1[0] ]
y1 = [ p1[1] ]

while p0[1] >=0:
    v2_0 = (v0[0]**2 + v0[1]**2)
    a0 = np.array([- k0 *  v2_0  , -10])
    p0 = p0 + v0 * h
    v0 = v0 + a0 * h
    x0.append(p0[0])
    y0.append(p0[1])


    v2_1 = (v1[0]**2 + v1[1]**2)
    a1 = np.array([- k1 *  v2_1 , -10])
    p1 = p1 + v1 * h
    v1 = v1 + a1 * h
    x1.append(p1[0])
    y1.append(p1[1])
fig = plt.figure()
plt.plot(x0, y0)
plt.plot(x1, y1)
plt.grid()
```


80
60
40
20
0
0
50
100
150
200

```
[ ]:
```