In [2]:
```
import numpy as np
import sympy as sp
from sympy import nroots
from matplotlib import pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython import display
from mpl_toolkits.mplot3d import Axes3D
from IPython.display import HTML
```

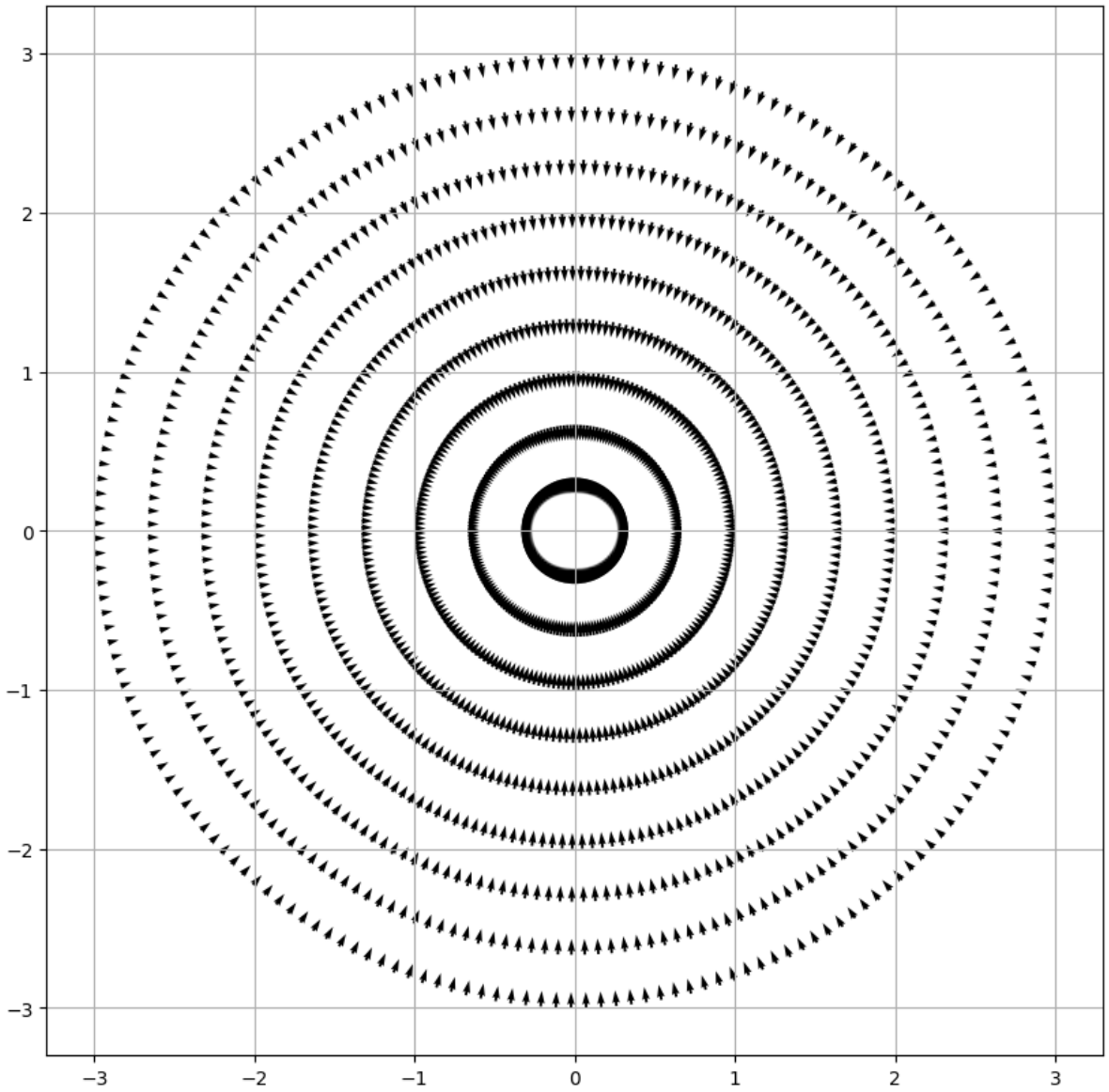
In [5]:
```
x,y,z,r=sp.symbols('x,y,z,r')#
```

In [6]:
```
k=39.478
F1=-k/r**2 #Ley de gravitacion universal
UU=-sp.integrate(F1,r) #Energia potencial del sistema
UU=UU.subs(r,sp.sqrt(x**2+y**2+z**2))
UUU=sp.lambdify([x,y,z],UU)
showc=2
```

Campo vectorial Central

In [7]:
```
fig, ax = plt.subplots(figsize=(10, 10))
ffx=F1*(x/r) ## Fuerza en x
ffy=F1*(y/r) ## Fuerza en y
ffz=F1*(z/r)

Fx=ffx.subs(r,sp.sqrt(x**2+y**2+z**2)) #R-> sqrt(x**2+y**2)
Fy=ffy.subs(r,sp.sqrt(x**2+y**2+z**2)) #R-> sqrt(x**2+y**2)
Fz=ffz.subs(r,sp.sqrt(x**2+y**2+z**2))

phi=np.linspace(0,2*np.pi,200)
rr=np.linspace(0,3,10)

RR,PHI=np.meshgrid(rr,phi)
X=RR*np.cos(PHI)
Y=RR*np.sin(PHI)

fx=sp.lambdify([x,y,z],Fx)
fy=sp.lambdify([x,y,z],Fy)
fz=sp.lambdify([x,y,z],Fz)

FXX=fx(X,Y,X)
FYY=fy(X,Y,X)
FZZ=fz(X,Y,X)
FF=np.sqrt(FXX**2+FYY**2+FZZ**2) ##Magnitud de la fuerza


ax.quiver(X,Y,FXX/FF,FYY/FF) #Campo vectorial

#ax.streamplot(X,Y,FXX/FF,FYY/FF)
#plt.axis('equal')
plt.grid('on')
```

```
<lambdifygenerated-3>:2: RuntimeWarning: invalid value encountered in divide
  return -39.478*x/(x**2 + y**2 + z**2)**(3/2)
<lambdifygenerated-4>:2: RuntimeWarning: invalid value encountered in divide
  return -39.478*y/(x**2 + y**2 + z**2)**(3/2)
<lambdifygenerated-5>:2: RuntimeWarning: invalid value encountered in divide
  return -39.478*z/(x**2 + y**2 + z**2)**(3/2)
```



Runge Kutta 4to orden

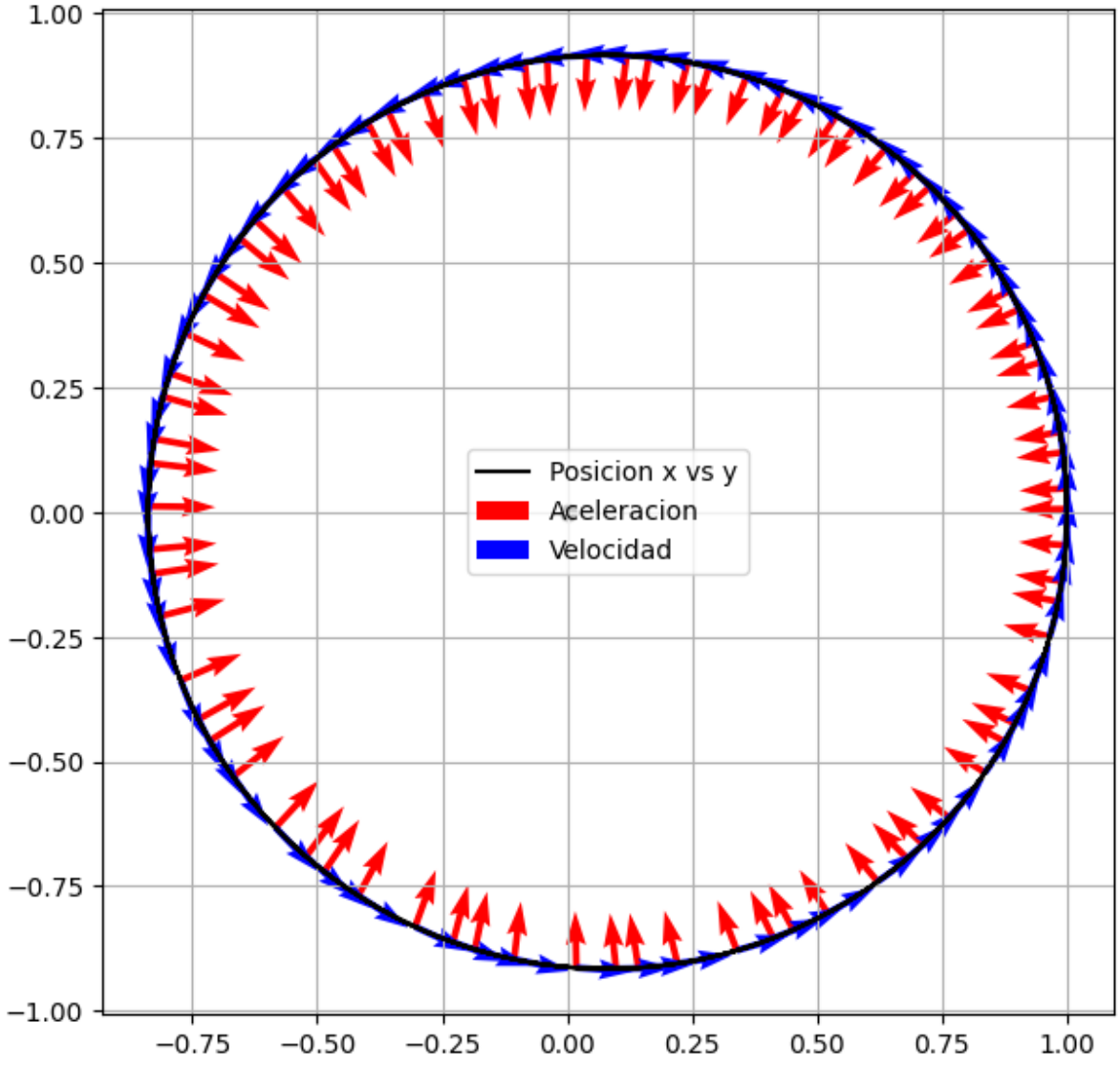
In [8]:
```
t,x,y,z,v_x,v_y,v_z=sp.symbols('t,x,y,z,v_x,v_y,v_z') #Variables simbolicas
```

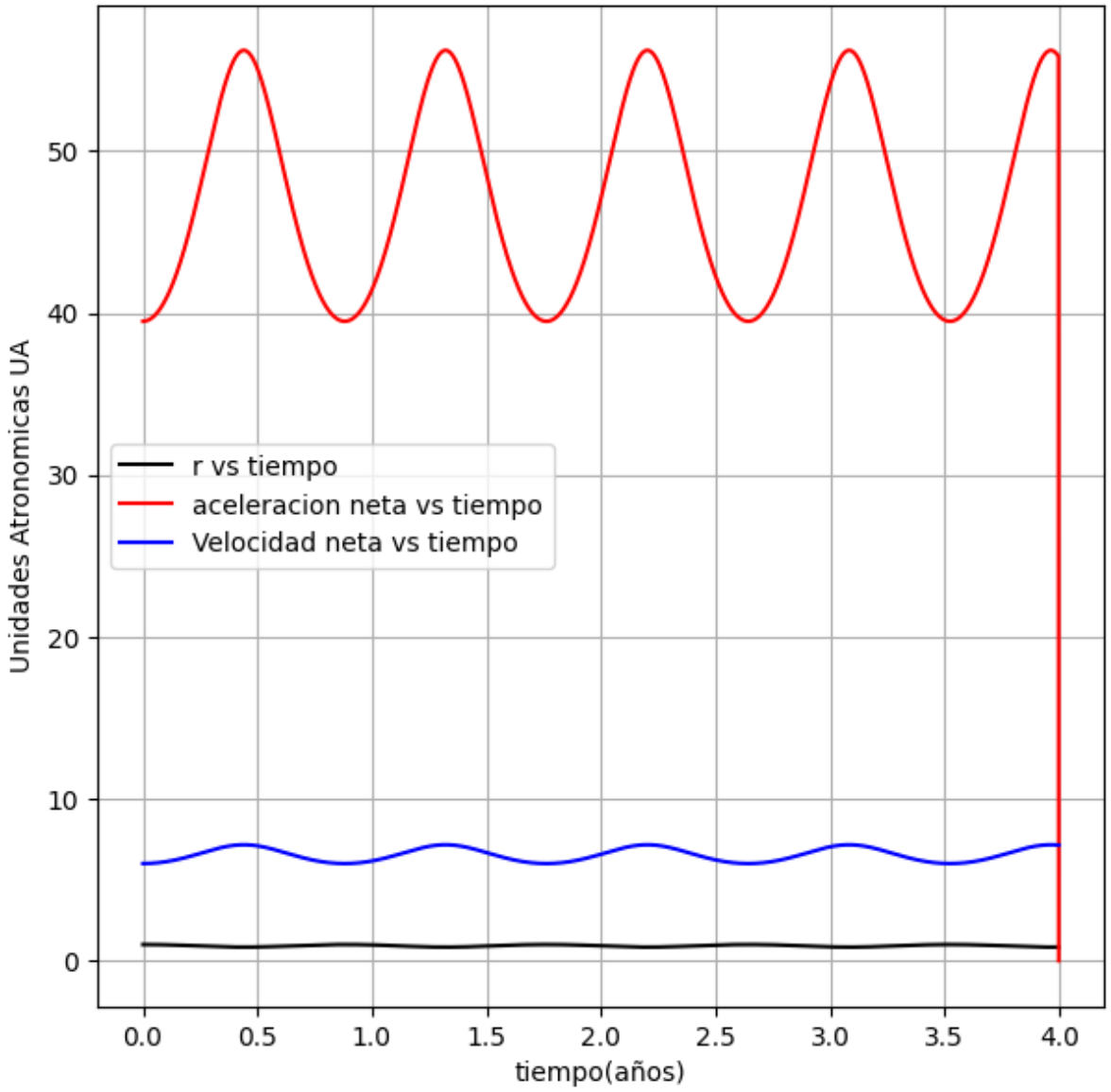
In [9]:
```
def d2ydt2(y,f1,f2,f3):
    return np.array([y[3],y[4],y[5],f1(y[0],y[1],y[2],y[3],y[4],y[5],y[6]),f2(y[0],

def RK42(x0,y0,z0,vx0,vy0,vz0,dt,tf,f,g,h,R,CV,M):
    #Vectorizcion de funciones
    f1=sp.lambdify([x,y,z,v_x,v_y,v_z,t],f)
    f2=sp.lambdify([x,y,z,v_x,v_y,v_z,t],g)
    f3=sp.lambdify([x,y,z,v_x,v_y,v_z,t],h)
```

```
    ts=np.arange(0,tf,dt)
    TT=0#tiempo en completar orbita
    n=len(ts)
    ##Posiciones
    ys=ts*0
    xs=ts*0
    zs=ts*0
    ##Velocidades
    vys=ts*0
    vxs=ts*0
    vzs=ts*0
    ##aceleraciones
    ays=ts*0
    axs=ts*0
    azs=ts*0
    ##Condiciones iniciales
    ys[0]=y0
    xs[0]=x0
    zs[0]=z0###3

    vys[0]=vy0
    vxs[0]=vx0
    vzs[0]=vz0


    for i in range(0,n-1):
        #Runge Kutta
        z0=np.array([xs[i],ys[i],zs[i],vxs[i],vys[i],vzs[i],ts[i]])
        k1=d2ydt2(z0,f1,f2,f3)
        k2=d2ydt2(z0+(dt*k1)/2,f1,f2,f3)
        k3=d2ydt2(z0+(dt*k2)/2,f1,f2,f3)
        k4=d2ydt2(z0+dt*k3,f1,f2,f3)

        #Vectores de x y z vy vx vz ax ay az
        xs[i+1]=xs[i]+(dt/6)*(k1[0]+2*k2[0]+2*k3[0]+k4[0])
        ys[i+1]=ys[i]+(dt/6)*(k1[1]+2*k2[1]+2*k3[1]+k4[1])
        zs[i+1]=zs[i]+(dt/6)*(k1[2]+2*k2[2]+2*k3[2]+k4[2])
        vxs[i+1]=vxs[i]+(dt/6)*(k1[3]+2*k2[3]+2*k3[3]+k4[3])
        vys[i+1]=vys[i]+(dt/6)*(k1[4]+2*k2[4]+2*k3[4]+k4[4])
        vzs[i+1]=vzs[i]+(dt/6)*(k1[5]+2*k2[5]+2*k3[5]+k4[5])

        ays[i]=f2(xs[i],ys[i],zs[i],vxs[i],vys[i],vzs[i],ts[i])
        axs[i]=f1(xs[i],ys[i],zs[i],vxs[i],vys[i],vzs[i],ts[i])
        azs[i]=f3(xs[i],ys[i],zs[i],vxs[i],vys[i],vzs[i],ts[i])

       ########Graficas#####################################

     ##########################vectores relativos#######################################
    if R==True:
        fig2 = plt.figure(figsize=(7,7))
        ax = plt.axes()
        #fig2=plt.add(figsize=(10,10))
        ax.plot(xs,ys,color='black',label='Posicion x vs y')
        ax.scatter(0,0,color='black')
        #plt.plot(vxs,vys,color='blue',label='Velocidad x vs velocidad y')
        #plt.plot(axs,ays,color='red',label='Aceleracion x vs aceleracion y')
```

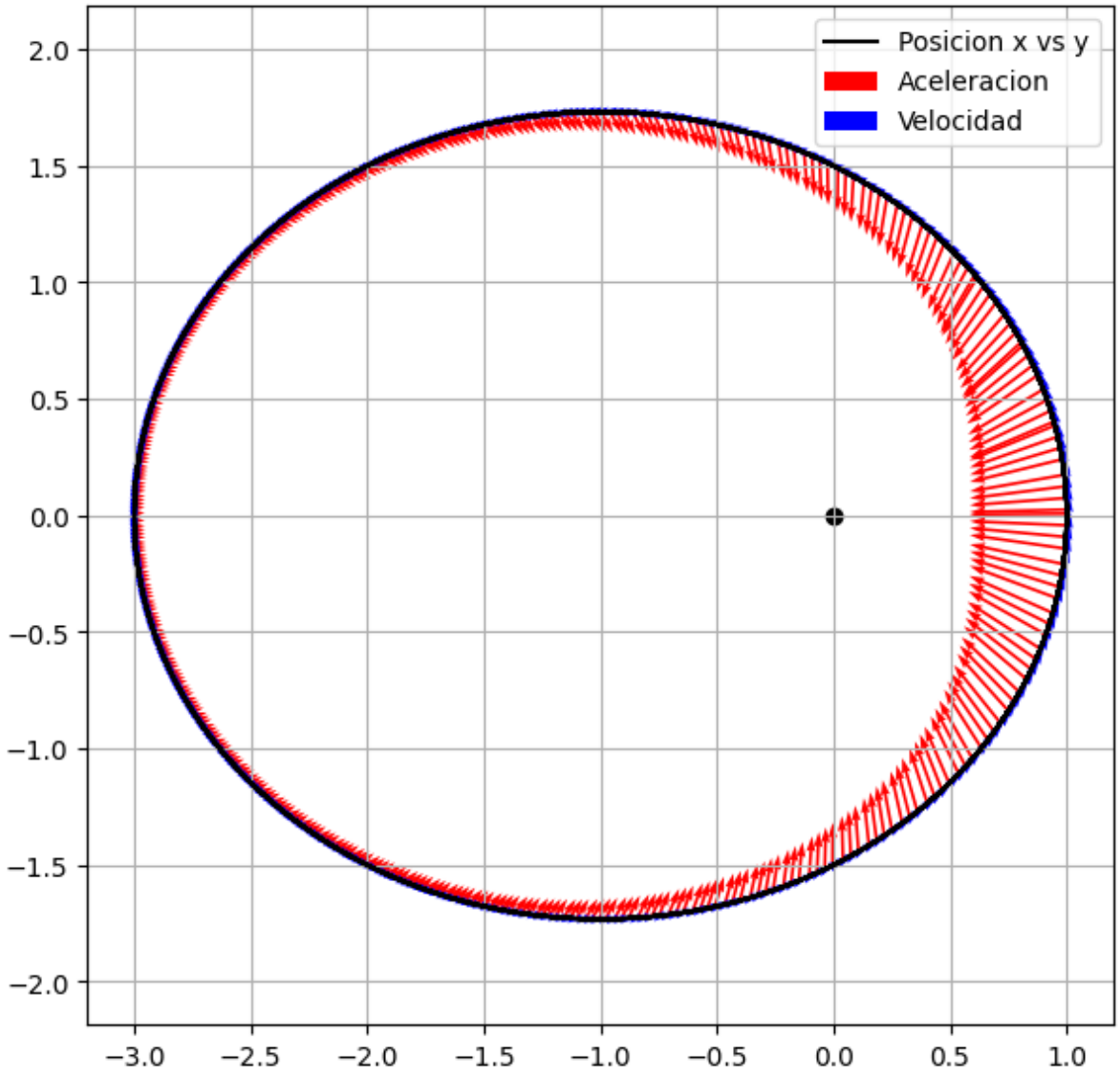
```
        if CV==True:
            ll=50
            ax.quiver(xs[1:-1:ll],ys[1:-1:ll],axs[1:-1:ll],ays[1:-1:ll],color='red'
            ax.quiver(xs[1:-1:ll],ys[1:-1:ll],vxs[1:-1:ll],vys[1:-1:ll],color='blue
        plt.axis('equal')
        plt.grid('on')
        plt.legend()
        plt.show()
    ####################Campos vectoriaels##########################################

    if M==True:
    ##############################Magnitudes#####################################
        fig3=plt.figure(figsize=(7,7))
        plt.plot(ts,np.sqrt(xs**2+ys**2),color='black',label='r vs tiempo')
        plt.plot(ts,np.sqrt(axs**2+ays**2),color='red',label='aceleracion neta vs t
        plt.plot(ts,np.sqrt(vxs**2+vys**2),color='blue',label='Velocidad neta vs ti
        #plt.plot(ts,vxs,color='red',label='Velocidad')
        #plt.plot(ts,axs,color='blue',label='aceleracion')
        plt.ylabel('Unidades Atronomicas UA')
        plt.xlabel('tiempo(años)')
        #plt.axis('equal')
        plt.grid('on')
        plt.legend()
        plt.show()
    return xs,ys,zs,vxs,vys,vzs,axs,ays,azs,ts,TT
```

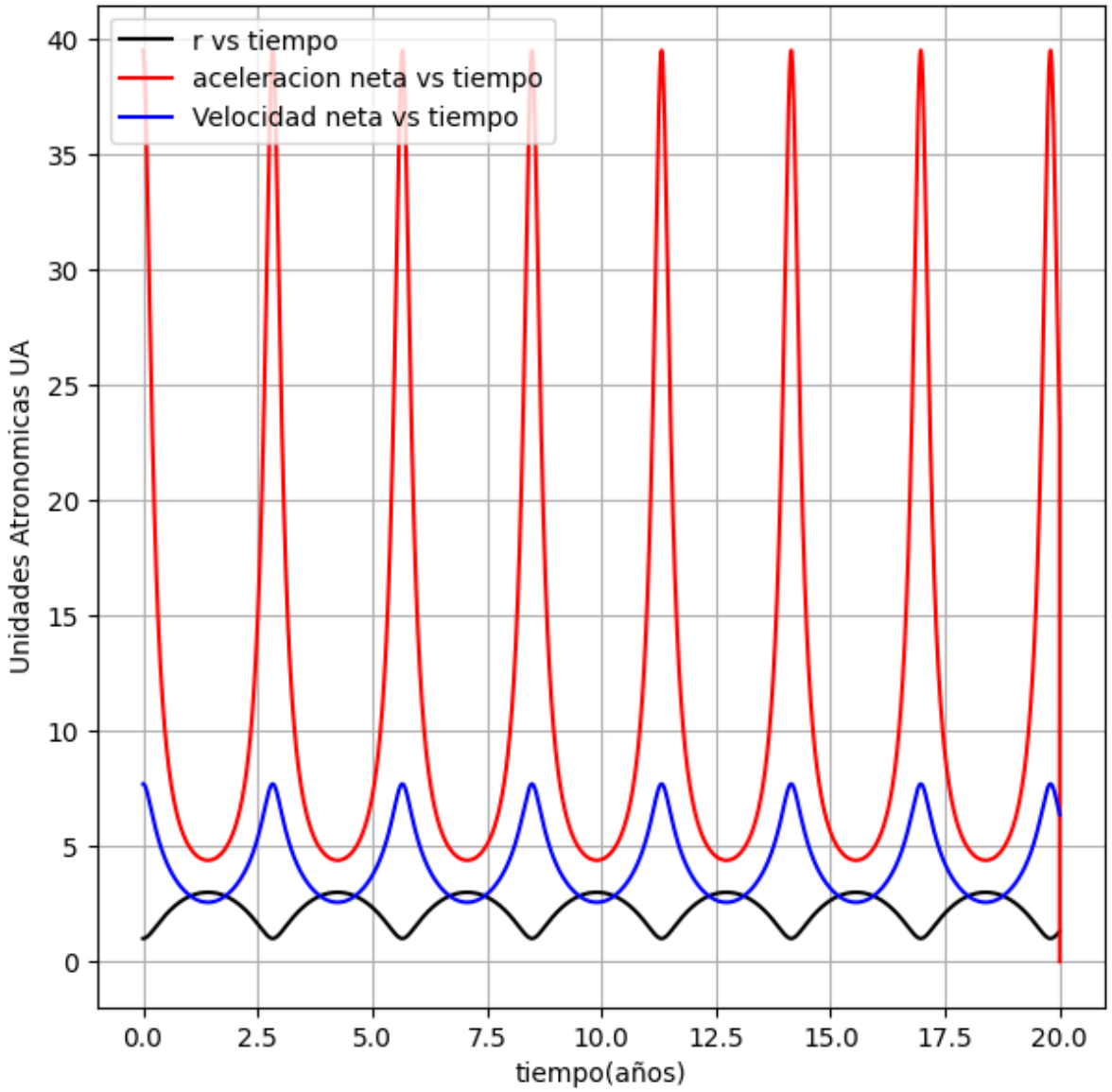
In [10]:
```
xs,ys,zs,vxs,vys,vzs,axs,ays,azs,ts,TT=RK42(1,0,0,0,6,0,0.001,4,Fx,Fy,Fz,True,True,
```


Posicion x vs y
Aceleracion
Velocidad
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



In [11]:
```
xs,ys,zs,vxs,vys,vzs,axs,ays,azs,ts,TT=RK42(1,0,0,0,7.6952,0,0.001,20,Fx,Fy,Fz,True
c=1
a=(max(xs)-min(xs))/2
e=c/(a)
e
```


Posicion x vs y
Aceleracion
Velocidad
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

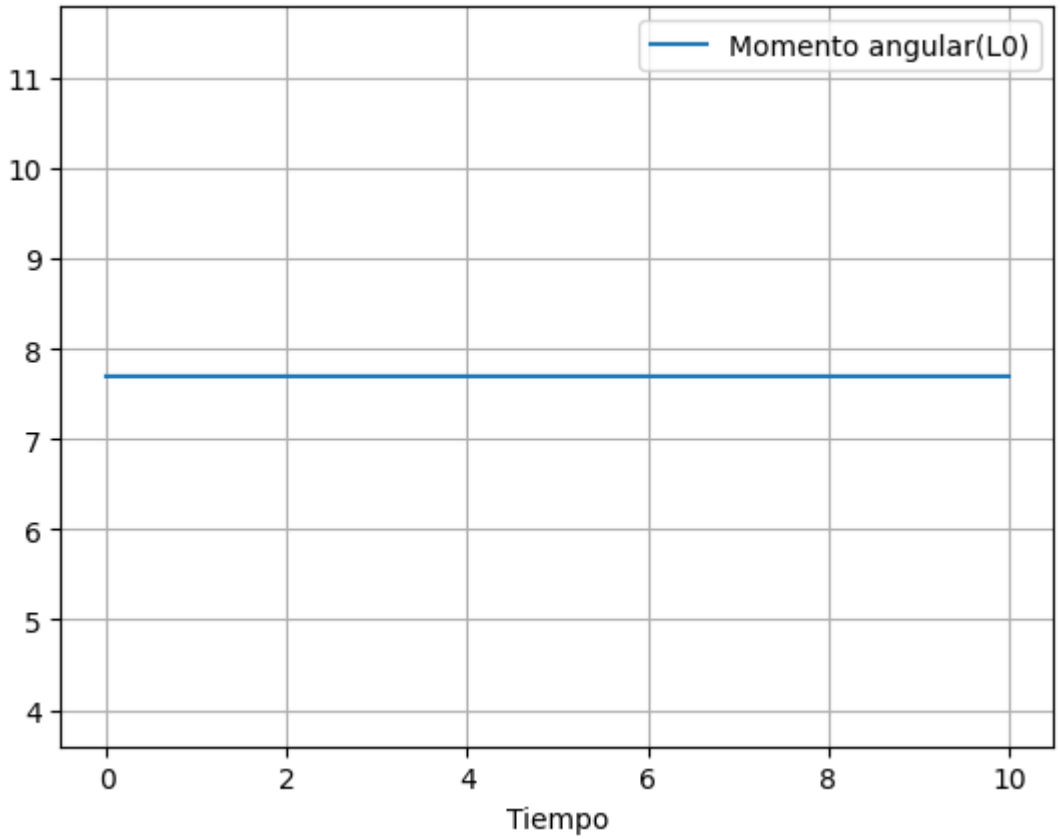


Out[11]: 0.5000227245947085

2da Ley de Kepler

In [65]:
```
m=1
L0=m*(xs*vys-vxs*ys)
plt.plot(ts,L0,label='Momento angular(L0)')

plt.axis('equal')
plt.grid('on')
plt.xlabel('Tiempo')
plt.legend()
```

Out[65]: <matplotlib.legend.Legend at 0x1d802646880>



Areas

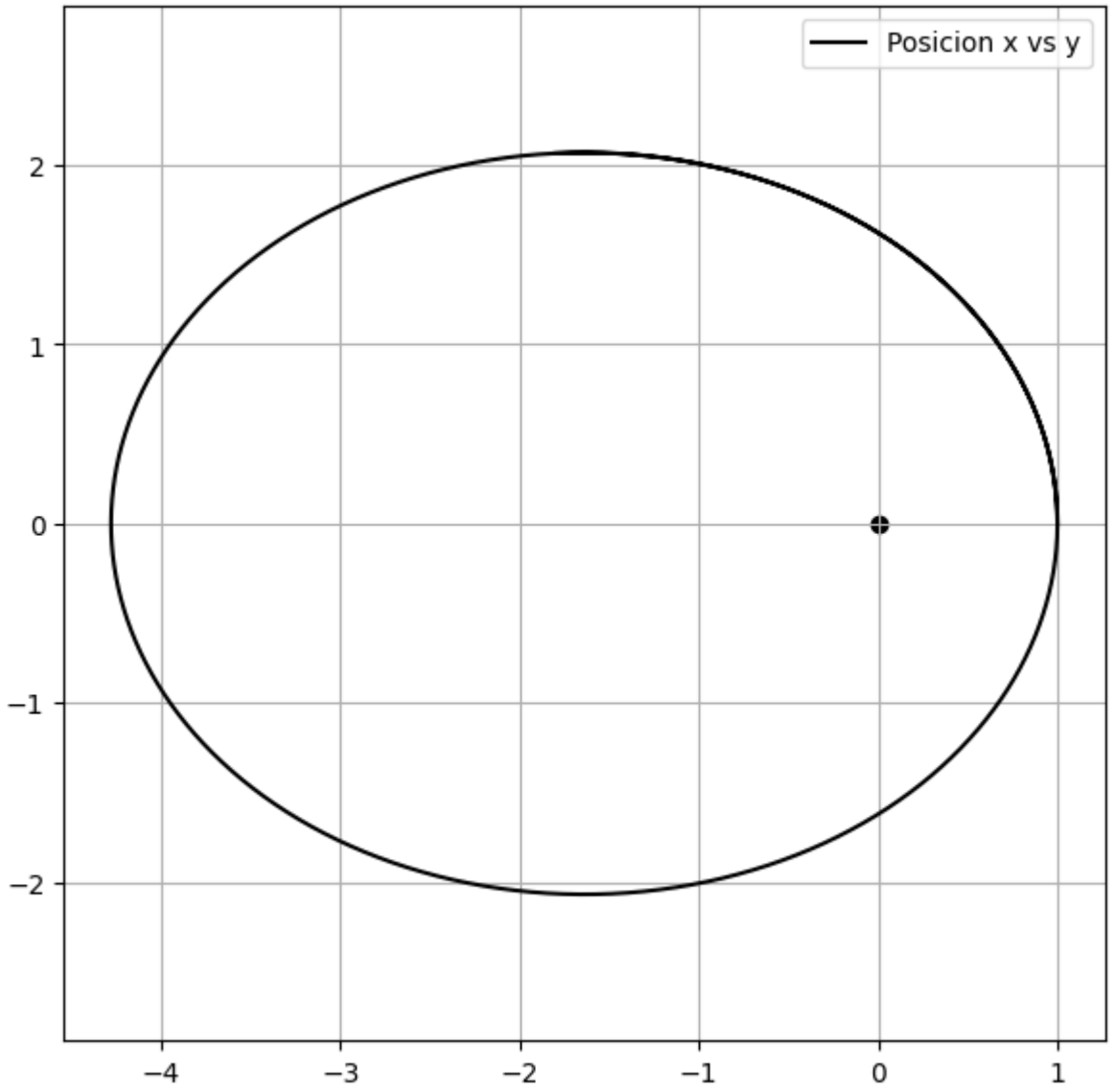
In [66]:
```
A1=abs((L0[0]/(m*2))*(ts[1000]-ts[0]))
A2=abs((L0[0]/(m*2))*(ts[-1]-ts[-1001]))
A2-A1
A1
```

Out[66]: 3.8476

3ra ley de Kepler

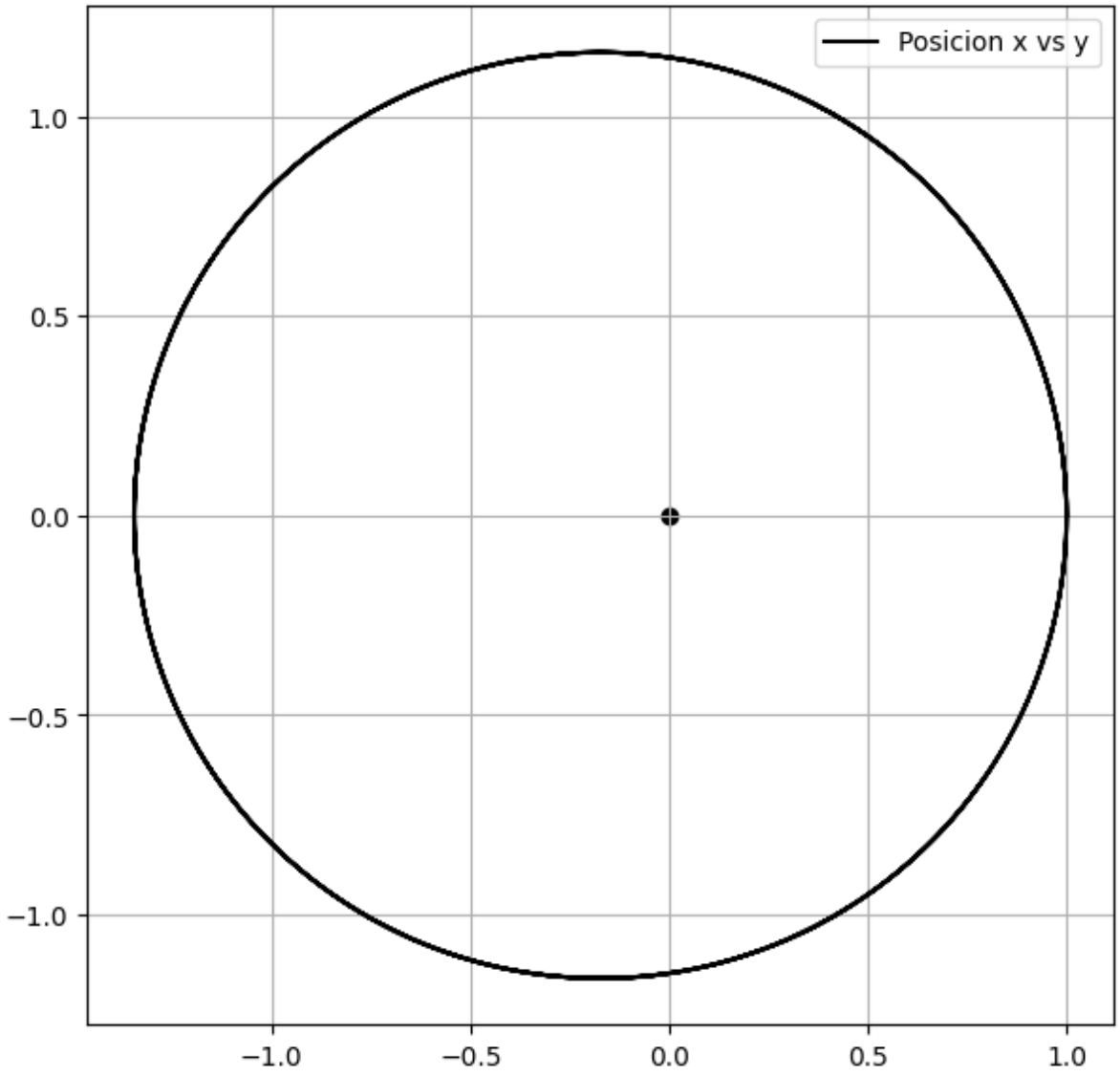
$$\frac{a^3}{T^2} = K$$

In [82]:
```
xs,ys,zs,vxs,vys,vzs,axs,ays,azs,ts,Tf=RK42(1,0,0,0,8,0,0.001,1.66*3,Fx,Fy,Fz,True,
c=1
a=(max(xs)-min(xs))/2

K1=a**2/Tf**2
```




```
C:\Users\Arif\AppData\Local\Temp\ipykernel_8688\1190419523.py:5: RuntimeWarning: d
ivide by zero encountered in double_scalars
  K1=a**2/Tf**2
```

In [85]:
```
xs,ys,zs,vxs,vys,vzs,axs,ays,azs,ts,TT=RK42(1,0,0,0,6.73,0,0.001,1.66*3,Fx,Fy,Fz,Tr
c=1
a=(max(xs)-min(xs))/2
K2=a**3/Tf**2
K2
```




```
C:\Users\Arif\AppData\Local\Temp\ipykernel_8688\325884280.py:4: RuntimeWarning: di
vide by zero encountered in double_scalars
  K2=a**3/Tf**2
```

Out[85]: inf

In [69]:
```
K1-K2
```

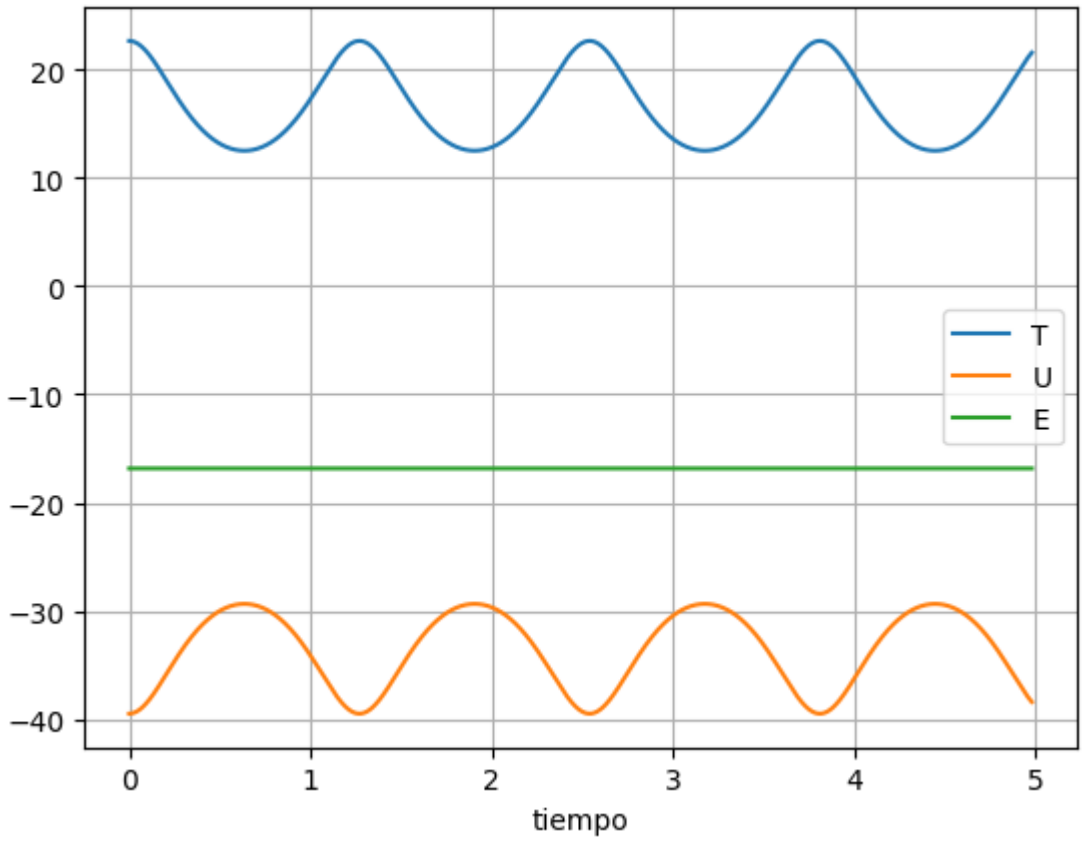
```
C:\Users\Arif\AppData\Local\Temp\ipykernel_8688\4090331271.py:1: RuntimeWarning: i
nvalid value encountered in double_scalars
  K1-K2
```

Out[69]: nan

Conservacion de la Energia

In [86]:
```
U=UUU(xs,ys,zs)
T=1/2*((vxs)**2+vys**2+vzs**2)
plt.plot(ts,T,label='T')
plt.plot(ts,U,label='U')
```

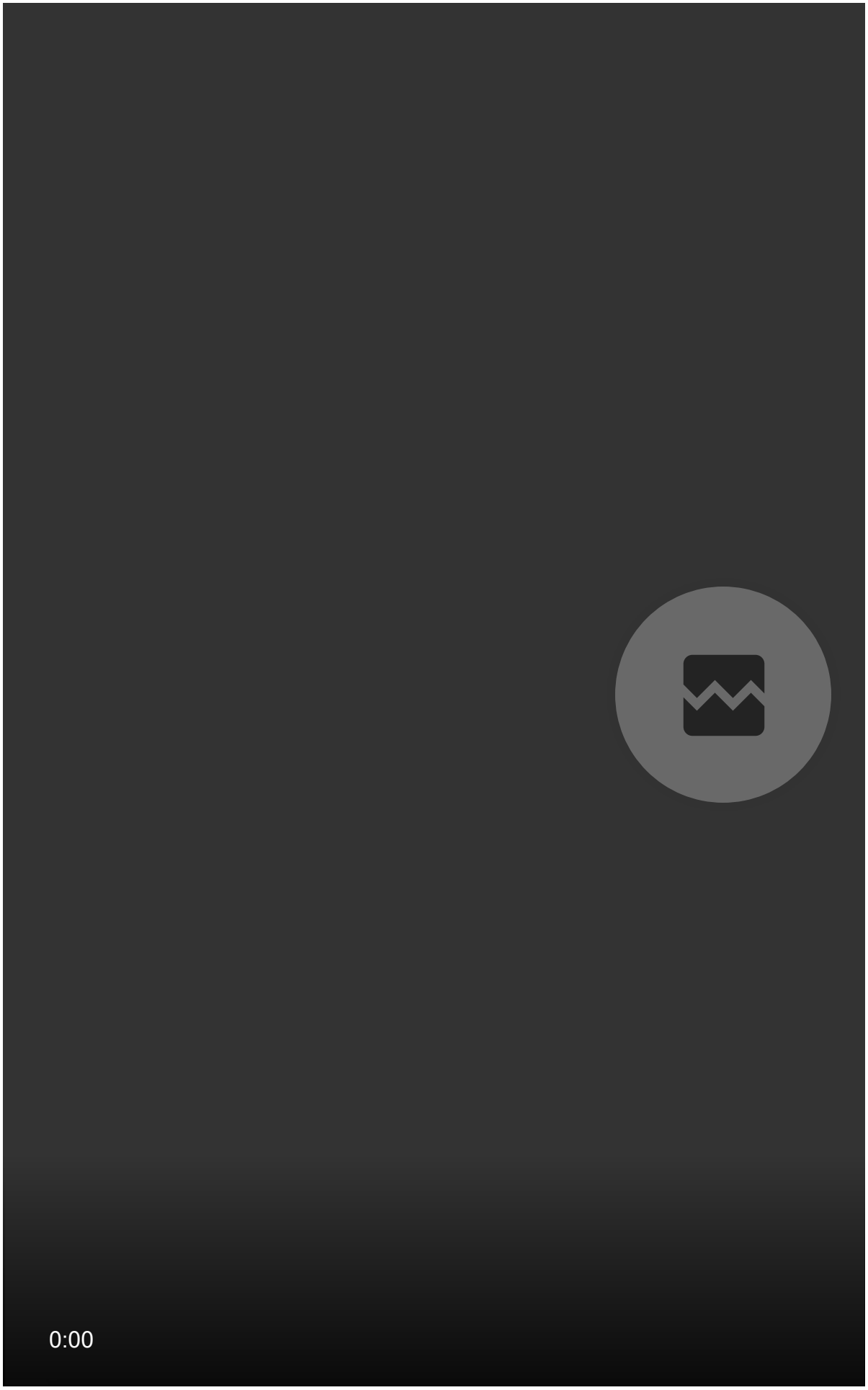
```
plt.plot(ts,T+U,label='E')
plt.xlabel('tiempo')
plt.legend()
plt.grid('on')
```



Animacion 2d

In [12]:
```
fig=plt.figure(figsize=(10, 10))
#plt.plot(0,0,lw=1000,color='red')
plt.grid('on')
plt.axis('equal')
plt.xlim(min(xs)-1,max(xs)+1)
plt.ylim(min(ys)-1,max(ys)+1)

resorte,=plt.plot([],[],color='black', label='trayectoria')
plt.scatter(0,0,color='red')
#Q=plt.quiver(xs,ys,axs,ays)
pp,=plt.plot([],[],'bo',label='Astro')
#plt.axis('equal')
plt.legend()


def animate(i):
    resorte.set_data((xs[0:i],ys[0:i]))
    pp.set_data((xs[i-1],ys[i-1]))
    #Q.set_UVC(axs[i-1],ays[i-1])
    #return Q,
    #plt.quiver(xs[i-1],ys[i-1],axs[i-1],ays[i-1],color='red')
```

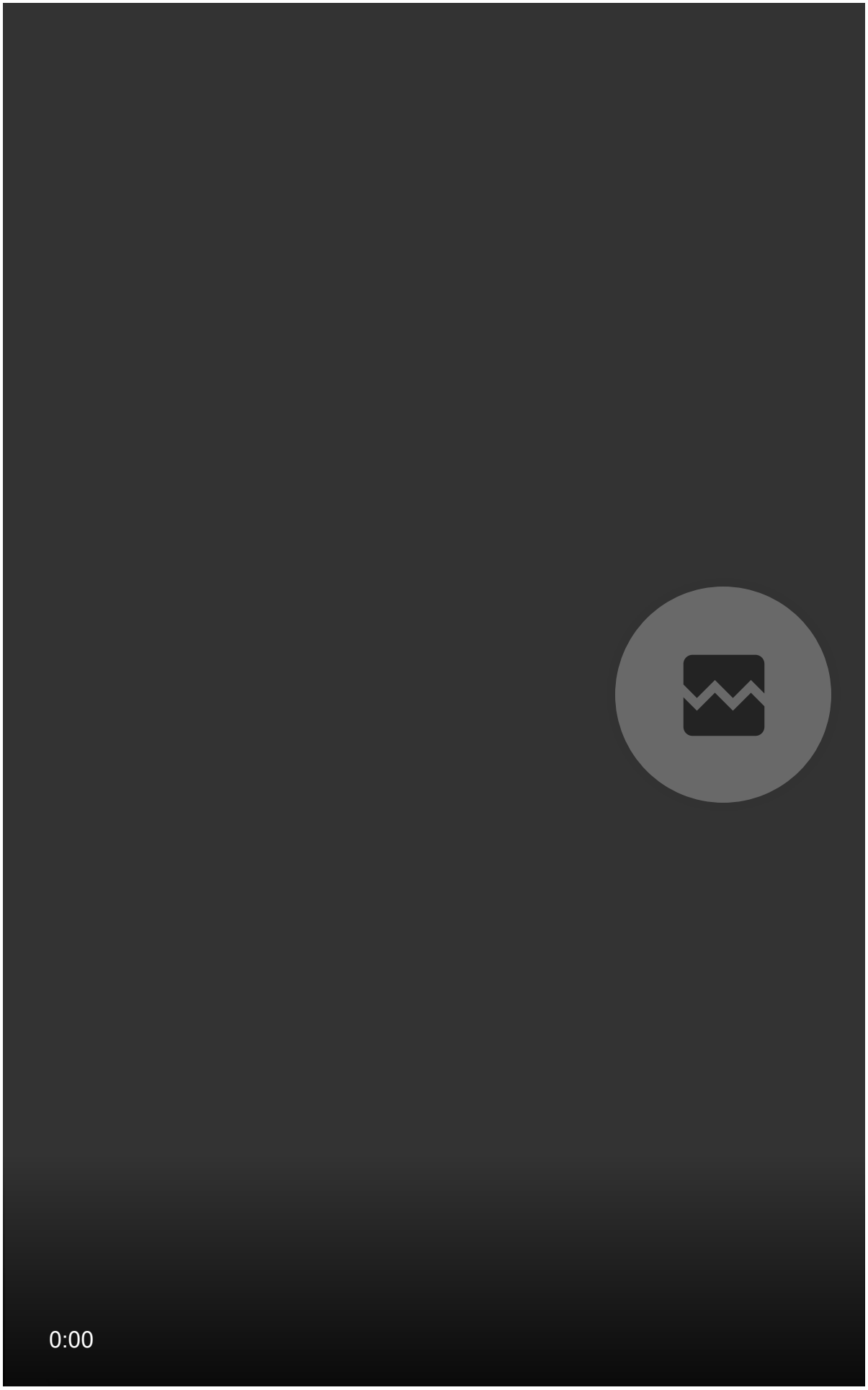
```
    #acc.set_data((xs[i-1],ys[i-1],axs[i-1],ays[i-1]))

n=20
fr=np.arange(0, len(ts)+1,n)
anim=FuncAnimation(fig,animate,frames=fr,interval=20)
video=anim.to_html5_video()
html=display.HTML(video)
display.display(html)
plt.close()
```


0:00

Animacion 3D

In [105…

```
fig4 = plt.figure(figsize=(10,10))
bx = fig4.add_subplot(111, projection='3d')

bx.set_facecolor('black')
plt.axis('equal')
plt.axis('on')
bx.set_xlim3d([min(xs)-1, max(xs)+1])
bx.set_ylim3d([min(ys)-1, max(ys)+1])
bx.set_zlim3d([min(ys)-1, max(ys)+1])
bx.view_init(elev=20, azim=45)


#orbita=ax.plot([],[],[],color='black', label='trayectoria')

#astro,=ax.plot3D([],[],[],'ro',label='Astro')
#astro, = ax.plot([xs],[ys],[ys],'ro',label='Astro')
#orbita, = ax.plot([], [], [], lw=2,color='black', label='trayectoria')

def animate(i):
    bx.clear()
    plt.axis('off')
    bx.set_xlim3d([min(xs)-1, max(xs)+1])
    bx.set_ylim3d([min(ys)-1, max(ys)+1])
    bx.set_zlim3d([min(ys)-1, max(ys)+1])
    bx.view_init(elev=30, azim=45)
    bx.scatter(0,0,0,color='red',s=300)
    #orbita.set_data(xs[:i], ys[:i])
    #orbita.set_3d_properties(ys[:i]*0)
    bx.scatter(xs[i-1],ys[i-1],zs[i-1],color='blue',s=10)
    bx.quiver(xs[i-1],ys[i-1],zs[i-1],axs[i-1],ays[i-1],azs[i-1],color='red',length
    bx.plot(xs[:i],ys[:i],zs[:i],color='white',lw=0.7)
    #astro.set_data(xs[i], ys[i])
    #astro.set_3d_properties(ys[i])
    #astro._offsets3d = ([xs[i-1]], [ys[i-1]], [0])
    return fig4 #orbita, #(astro,)

n=50
fr=np.arange(0, len(ts)+1,n)
anim=FuncAnimation(fig4,animate,frames=fr,interval=20)
video=anim.to_html5_video()
html=display.HTML(video)
display.display(html)
plt.close()
```


0:00

In [ ]: