**Edited by James Amidei**

NOTE: I changed the code so that the values for each of the initial velocities would be displayed simultaneously.

## Central Force and Elliptical Orbit

In this section, elliptical orbit due gravitational force is discussed. The Cartesian and polar coordinates are used.

### Cartesian Coordinates

The gravitational force between two masses $M$ and $m$ at distance $r$ is given by

$$\overrightarrow{F_G} = -G\frac{Mm}{r^2}\hat{r}$$

The $x$ and $y$ components of the gravitational force are given by

$$F_{Gx} = -G\frac{Mm}{r^2}\cos\theta = -G\frac{Mm}{r^2}\frac{x}{r} = -(GMm)\frac{x}{r^3} = -(GMm)\frac{x}{(x^2+y^2)^{3/2}}$$

$$F_{Gy} = -G\frac{Mm}{r^2}\sin\theta = -G\frac{Mm}{r^2}\frac{y}{r} = -GMm\frac{y}{r^3} = -(GMm)\frac{y}{(x^2+y^2)^{3/2}}$$

The Newton's seconds law in $x$ and $y$ direction is given by

$$-(GMm)\frac{x}{(x^2+y^2)^{3/2}} = ma_x$$

$$-(GMm)\frac{y}{(x^2+y^2)^{3/2}} = ma_y$$

The above equations yields

$$a_x = \frac{-GMx}{(x^2+y^2)^{3/2}}$$

$$a_y = \frac{-GMy}{(x^2+y^2)^{3/2}}$$

The Euler-Cromer method $x$ and $y$ is given by

$$v_x(t_{i+1}) = v_x(t_i) + a_x(t_i)\Delta t = v_x(t_i) - \left(\frac{GMx(t_i)}{\left(x^2(t_i)+y^2(t_i)\right)^{3/2}}\right)\Delta t$$

$$x(t_{i+1}) = x(t_i) + v_x(t_{i+1})\Delta t$$

$$v_y(t_{i+1}) = v_y(t_i) + a_y(t_i)\Delta t = v_y(t_i) - \left(\frac{GMy(t_i)}{\left(x^2(t_i)+y^2(t_i)\right)^{3/2}}\right)\Delta t$$

$$y(t_{i+1}) = y(t_i) + v_y(t_{i+1})\Delta t$$

The orbit of is an ellipse and the equation of an ellipse is given by

$$\frac{(x+f)^2}{a^2} + \frac{y^2}{b^2} = 1$$

where $a$ and $b$ are semimajor and semiminor axes and $f$ is the focal length.

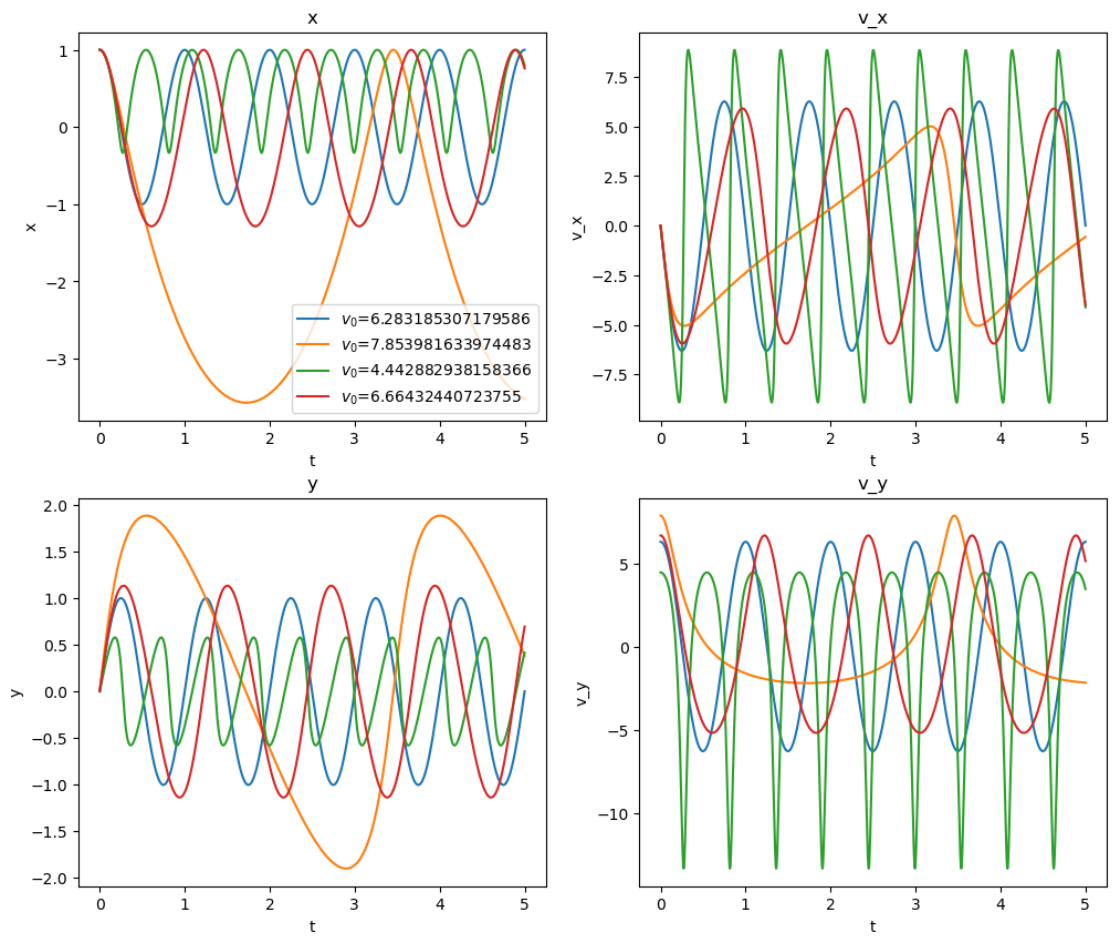
In [22]:

```
import math
import numpy as np
import matplotlib.pyplot as plt

#Gravity, Orbital motion
GM=4*np.pi**2  #AU^3/yr^2
r0=1   #AU
theta0_D=0
theta0 = theta0_D*np.pi/180
#v0=1.5*np.sqrt(2)*np.pi  #2*np.pi
v0_values = [2*np.pi, 2.5*np.pi, np.sqrt(2)*np.pi, 1.5*np.sqrt(2)*np.pi]

del_t=0.001
n=5000
t=np.zeros(n+1)

x_array = np.zeros((len(v0_values), n+1))
vx_array = np.zeros((len(v0_values), n+1))
ax_array = np.zeros((len(v0_values), n+1))

y_array = np.zeros((len(v0_values), n+1))
vy_array = np.zeros((len(v0_values), n+1))
ay_array = np.zeros((len(v0_values), n+1))

Flag_T=1

for index, v0 in enumerate(v0_values):
    x = np.zeros(n+1)
    vx = np.zeros(n+1)
    ax = np.zeros(n+1)

    y = np.zeros(n+1)
    vy = np.zeros(n+1)
    ay = np.zeros(n+1)

    x[0]=r0*np.cos(theta0)
    y[0]=r0*np.sin(theta0)

    vx[0]=v0*np.sin(theta0)
    vy[0]=v0*np.cos(theta0)

    ax[0]=-GM*x[0]/r0**3
    ay[0]=-GM*y[0]/r0**3

    # Euler-Cromer Method Method
    for i in range(0,n):
        t[i+1]=(i+1)*del_t

        #x-direction:
        vx[i+1]=vx[i]+ax[i]*del_t
        x[i+1]=x[i]+vx[i+1]*del_t

        #y-direction:
        vy[i+1]=vy[i]+ay[i]*del_t
        y[i+1]=y[i]+vy[i+1]*del_t

        del_x=x[i+1]-x[i]
```

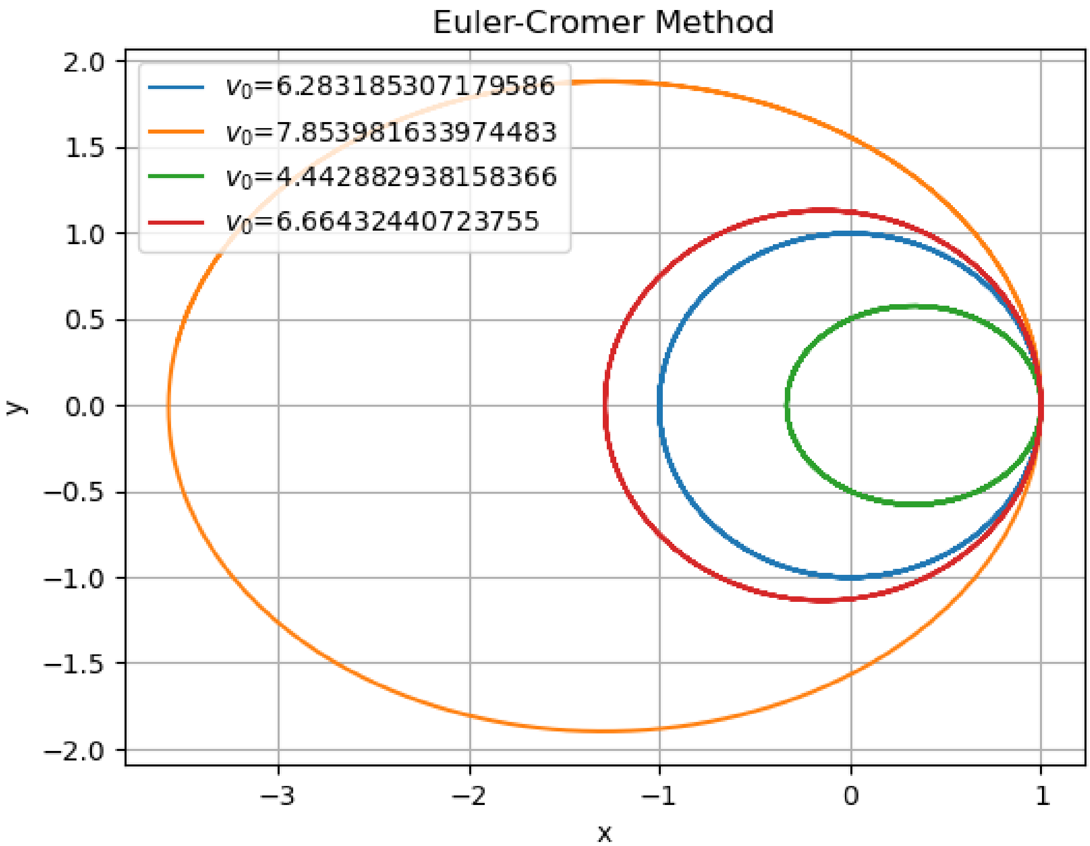
```
        if x[i+1] == 0 or (x[i+1]<= abs(del_x) and x[i+1] >= -abs(del_x)):
            c=y[i+1]

        #if Flag_T==1 and t[i]>0 and np.sign(y[i+1]) != np.sign(y[i]) :
        #    T=t[i+1]+t[i]
        #    Flag_T= 0

        if Flag_T==1 and t[i]>0 and np.sign(y[i+1]) != np.sign(y[i]) and x
            T=t[i+1]#+t[i]
            Flag_T= 0

        r=np.sqrt(x[i+1]**2+y[i+1]**2)

        ax[i+1]=-GM*x[i+1]/r**3
        ay[i+1]=-GM*y[i+1]/r**3

    x_array[index, :] = x
    vx_array[index,:] = vx
    ax_array[index,:] = ax
    y_array[index, :] = y
    vy_array[index,:] = vy
    ay_array[index,:] = ay

plt.figure(figsize=(12,10))

print('Plots for position and velocity using the Euler-Cromer method')

plt.subplot(2,2,1)
for i, v0 in enumerate(v0_values):
    plt.plot(t, x_array[i,:], label=f'$v_0$={v0}')
plt.xlabel('t')
plt.ylabel('x')
plt.title('x')
plt.legend()

plt.subplot(2,2,2)
for i, v0 in enumerate(v0_values):
    plt.plot(t, vx_array[i,:], label=f'$v_0$={v0}')
plt.xlabel('t')
plt.ylabel('v_x')
plt.title('v_x')
#plt.legend()

plt.subplot(2,2,3)
for i, v0 in enumerate(v0_values):
    plt.plot(t, y_array[i,:], label=f'$v_0$={v0}')
plt.xlabel('t')
plt.ylabel('y')
plt.title('y')
#plt.legend()

plt.subplot(2,2,4)
for i, v0 in enumerate(v0_values):
    plt.plot(t, vy_array[i,:], label=f'$v_0$={v0}')
plt.xlabel('t')
plt.ylabel('v_y')
plt.title('v_y')
```

```
#plt.legend()

plt.show()

plt.figure()
for i, v0 in enumerate(v0_values):
    plt.plot(x_array[i,:], y_array[i,:], label=f'$v_0$={v0}')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Euler-Cromer Method')
plt.legend(loc="upper left")
```

Plots for position and velocity using the Euler-Cromer method




Out[22]: <matplotlib.legend.Legend at 0x255ccc20b90>




**The Polar Coordinates**

The gravitational force between two masses $M$ and $m$ at distance $r$ in polar coordinates is given by

$$-G\frac{Mm}{r^2} = \frac{d^2r}{dt^2} - r\left(\frac{d\theta}{dt}\right)^2$$
$$0 = r\frac{d^2\theta}{dt^2} + 2\frac{dr}{dt}\frac{d\theta}{dt}$$

The above equations can be written as following

$$\frac{d^2r}{dt^2} = -G\frac{Mm}{r^2} + r\left(\frac{d\theta}{dt}\right)^2$$
$$\frac{d^2\theta}{dt^2} = -\frac{2}{r}\frac{dr}{dt}\frac{d\theta}{dt}$$

or

$$\frac{dr}{dt} = v_r$$
$$\frac{d\theta}{dt} = \omega$$
$$\frac{dv_r}{dt} = -G\frac{Mm}{r^2} + r\omega^2$$
$$\frac{d\omega}{dt} = -\frac{2}{r}v_r\omega$$

Using Finite-Difference Method, we have

$$v_r(t_{i+1}) = v_r(t_i) + a_r(t_i)\Delta t = v_r(t_i) + \left(-G\frac{Mm}{r^2(t_i)} + r(t_i)\omega^2(t_i)\right)\Delta t$$

$$\omega(t_{i+1}) = \omega(t_i) + \alpha(t_i)\Delta t = \omega(t_i) + \left(-\frac{2}{r(t_i)}v_r(t_i)\omega(t_i)\right)\Delta t$$

$$r(t_{i+1}) = r(t_i) + v_r(t_{i+1})\Delta t$$
$$\theta(t_{i+1}) = \theta(t_i) + \omega(t_{i+1})\Delta t$$

The equation or orbit in polar coordinate is given by

$$r(\theta) = \frac{c}{1 + e\cos\theta}$$

where $e$ is eccentricity and for an ellipse is $0 < e < 1$ and for a circle is $e = 1$ and $c$ is given by

$$r\left(\frac{\pi}{2}\right) = \frac{c}{1 + e\cos\left(\frac{\pi}{2}\right)}$$

The above equation yields

$$c = r\left(\frac{\pi}{2}\right)$$

The minimum and maximum $r$ (i.e. $r_{min} = r(\theta = 0), r_{max} = r(\theta = \pi)$ ) are given by

$$r_{min} = \frac{c}{1+e}$$

$$r_{max} = \frac{c}{1-e}$$

The semimajor and semiminor axes and focal length are given by

$$a = \frac{c}{1-e^2}$$

$$b = \frac{c}{\sqrt{1-e^2}}$$

$$f = ae$$

. From above equations, the ratio of $\frac{a}{b}$ is given by

$$\frac{b}{a} = \sqrt{1-e^2}$$

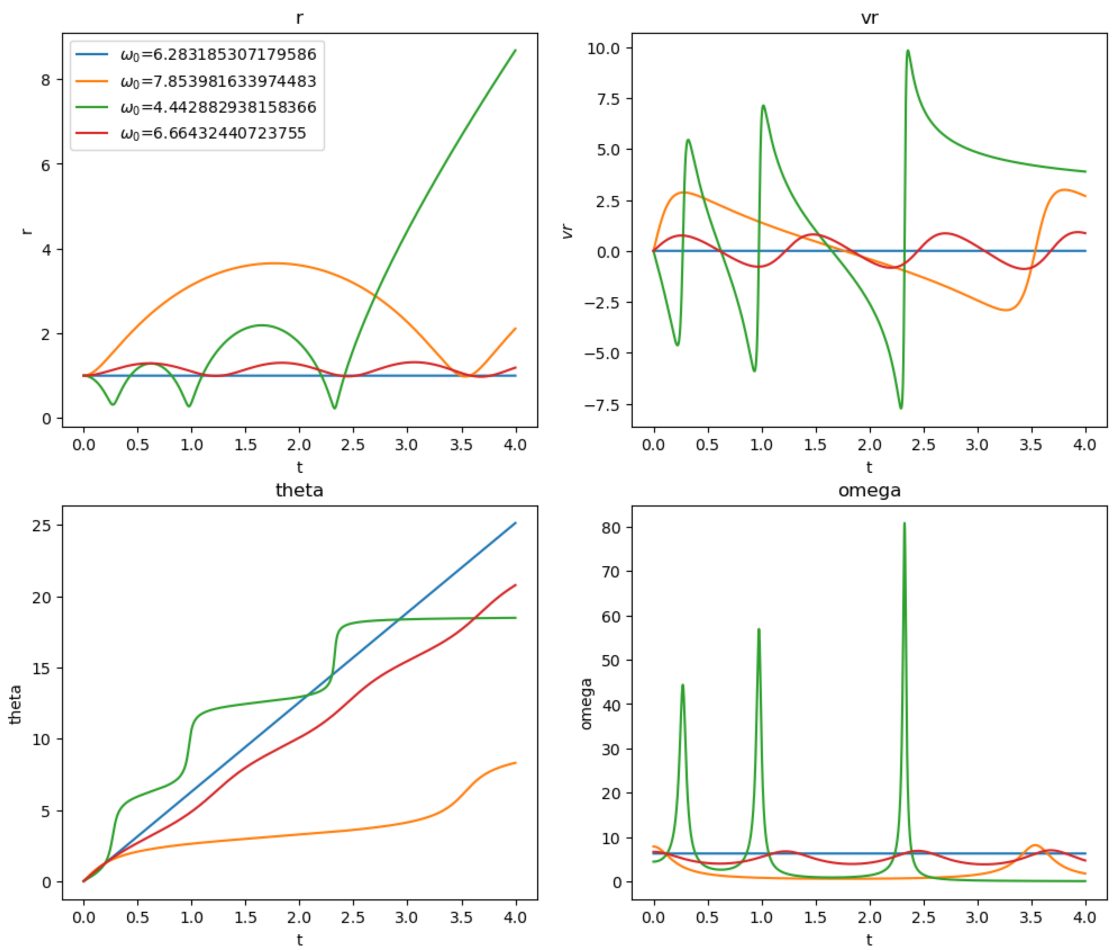
.

In [30]:
```
import math
import numpy as np
import matplotlib.pyplot as plt

#Gravity, Orbital motion
GM=4*np.pi**2  #AU^3/yr^2
r0=1   #AU
theta0_D=0
theta0=theta0_D*np.pi/180

vr0=0
omega0_values = [2*np.pi, 2.5*np.pi, np.sqrt(2)*np.pi, 1.5*np.sqrt(2)*np.p


del_t=0.001
n=4000
t=np.zeros(n+1)

r_array = np.zeros((len(omega0_values),n+1))
vr_array = np.zeros((len(omega0_values),n+1))
ar_array = np.zeros((len(omega0_values),n+1))

theta_array = np.zeros((len(omega0_values),n+1))
omega_array = np.zeros((len(omega0_values),n+1))
alpha_array = np.zeros((len(omega0_values),n+1))

x_array = np.zeros((len(omega0_values),n+1))
y_array = np.zeros((len(omega0_values),n+1))

for index, omega0 in enumerate(omega0_values):
    r = np.zeros(n+1)
    vr = np.zeros(n+1)
    ar = np.zeros(n+1)

    theta = np.zeros(n+1)
    omega = np.zeros(n+1)
    alpha = np.zeros(n+1)

    x = np.zeros(n+1)
    y = np.zeros(n+1)

    r[0]=r0
    theta[0]=theta0

    vr[0]=vr0
    omega[0]=omega0

    ar[0]=-GM/r[0]**2+r[0]*omega[0]**2
    alpha[0]=-2*vr[0]*omega[0]/r[0]

    x[0]=r[0]*np.cos(theta[0])
    y[0]=r[0]*np.sin(theta[0])

    # Euler-Cromer Method Method
    for i in range(0,n):
        t[i+1]=(i+1)*del_t
```

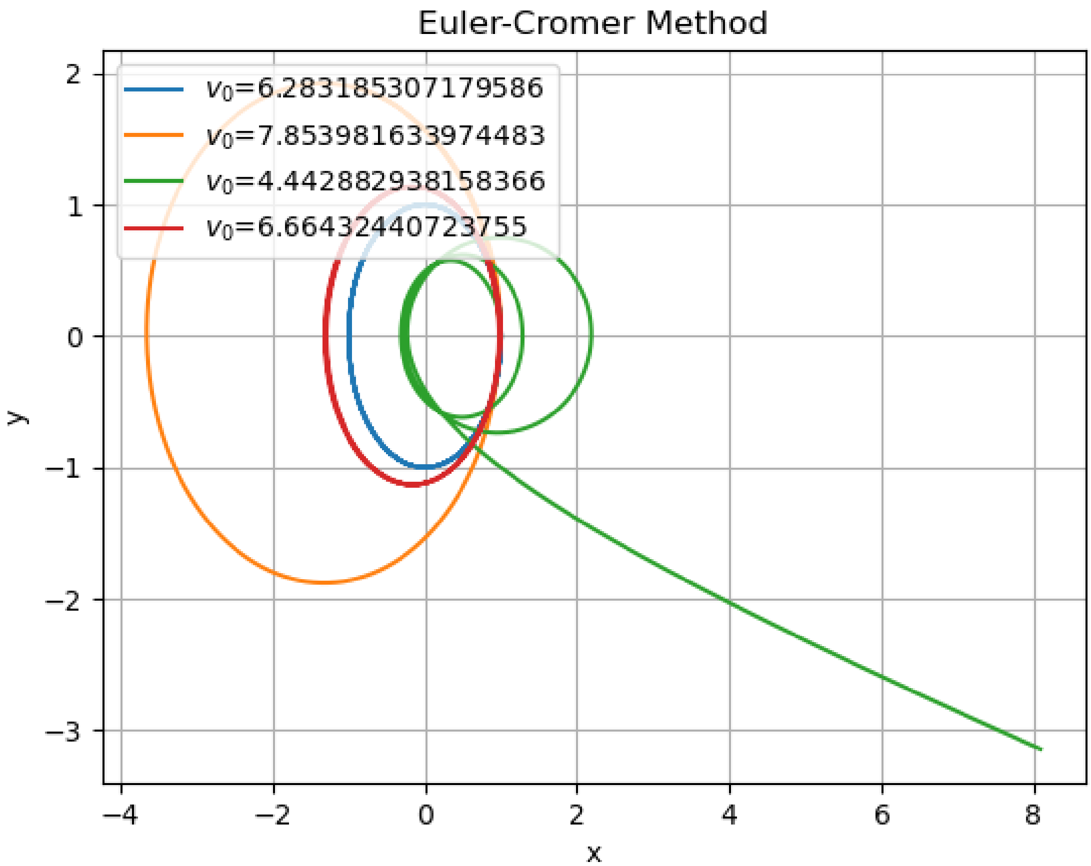
```
        #r:
        vr[i+1]=vr[i]+ar[i]*del_t
        r[i+1]=r[i]+vr[i+1]*del_t

        #theta:
        omega[i+1]=omega[i]+alpha[i]*del_t
        theta[i+1]=theta[i]+omega[i+1]*del_t

        ar[i+1]=-GM/r[i+1]**2+r[i+1]*omega[i+1]**2
        alpha[i+1]=-2*vr[i+1]*omega[i+1]/r[i+1]

        x[i+1]=r[i+1]*np.cos(theta[i+1])
        y[i+1]=r[i+1]*np.sin(theta[i+1])

        del_theta=theta[i+1]-theta[i]
        if theta[i+1] == np.pi / 2 or (theta[i+1] <= np.pi / 2 + abs(del_t
            c = y[i+1]

    r_array[index,:] = r
    vr_array[index,:] = vr
    ar_array[index,:] = ar
    theta_array[index,:] = theta
    omega_array[index,:] = omega
    alpha_array[index,:] = alpha
    x_array[index,:] = x
    y_array[index,:] = y

plt.figure(figsize=(12, 10))

print('Plots for position and velocity using the Euler-Cromer method')

plt.subplot(2, 2, 1)
for i, omega0 in enumerate(omega0_values):
    plt.plot(t, r_array[i,:], label=f'$\omega_0$={omega0}')
plt.xlabel('t')
plt.ylabel('r')
plt.title('r')
plt.legend(loc='upper left')

plt.subplot(2, 2, 2)
for i, omega0 in enumerate(omega0_values):
    plt.plot(t, vr_array[i, :], label=f'$\omega_0$={omega0}')
plt.xlabel('t')
plt.ylabel('$vr$')
plt.title('vr')

plt.subplot(2, 2, 3)
for i, omega0 in enumerate(omega0_values):
    plt.plot(t, theta_array[i, :], label=f'$\omega_0$={omega0}')
plt.xlabel('t')
plt.ylabel('theta')
plt.title('theta')

plt.subplot(2, 2, 4)
for i, omega0 in enumerate(omega0_values):
    plt.plot(t, omega_array[i, :], label=f'$\omega_0$={omega0}')
plt.xlabel('t')
```

```
plt.ylabel('omega')
plt.title('omega')

plt.show()

plt.show()

plt.figure()
for i, v0 in enumerate(v0_values):
    plt.plot(x_array[i,:], y_array[i,:], label=f'$v_0$={v0}')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Euler-Cromer Method')
plt.legend(loc="upper left")
```

```
Plots for position and velocity using the Euler-Cromer method
```




Out[30]: <matplotlib.legend.Legend at 0x255cc267290>


Euler-Cromer Method
$v_0$=6.283185307179586
$v_0$=7.853981633974483
$v_0$=4.442882938158366
$v_0$=6.66432440723755
y
x
2
1
0
−1
−2
−3
−4
−2
0
2
4
6
8

In [67]:

```
r_min=min(r)
r_max=max(r)

y_min=min(y)
y_max=max(y)

a=(r_min+r_max)/2
b=(abs(max(y))+abs(min(y)))/2

print('a=',a,' \nb=',b,'\nf=',f, '\ne=', e, '\nc=',c, '\nT=',T)
print()

f=a-r_min
e=np.sqrt(1-(b/a)**2)

f=a*e
a=c/(1-e**2)
b=c/np.sqrt(1-e**2)

print('Verification: \n f = a*e =', f, '\n a = c/(1-e^2) =', a,'\n b = c/s
print()

# Kepler's third law: T^2=4*pi^2/GM a^3#

T=np.sqrt((4*np.pi**2/GM)*a**3)

print('Kepler third law: \n T = sqrt((4*pi^2)/GM)*a^3) =',T)
```

```
a= 1.1459517844092808
b= 1.1361673024811265
f= 0.14943009390204837
e= 0.13039823833345407
c= 1.1277083297636659
T= 1.2287608286825058

Verification:
 f = a*e = 0.14943009390204837
 a = c/(1-e^2) = 1.1472152340824107
 b = c/sqrt(1-e^2) = 1.1374199644399199

Kepler third law:
 T = sqrt((4*pi^2)/GM)*a^3) = 1.2287608286825058
```

In [ ]: