Circumbinary Planet Individual Sub-Analysis

Kepler-35 System

https://en.wikipedia.org/wiki/Kepler-35 (https://en.wikipedia.org/wiki/Kepler-35)

Srikar Vakkalagadda

Kepler-35 Binary System, two G-type main sequence stars, 6300 ly away from Earth

Kepler-35A: Mass: 0.8877 Msun, Radius: 1.0284 Rsun

Kepler-35B: Mass: 0.8094 Msun, Radius: 0.7861 Rsun

Orbit: Period: 20.73 days, Semi-major axis: 0.176 Rsun

One planet, b, gas giant

b: Mass: 0.127 Mjup, Period: 131.458 days, Semi-major axis: 0.60347 AU

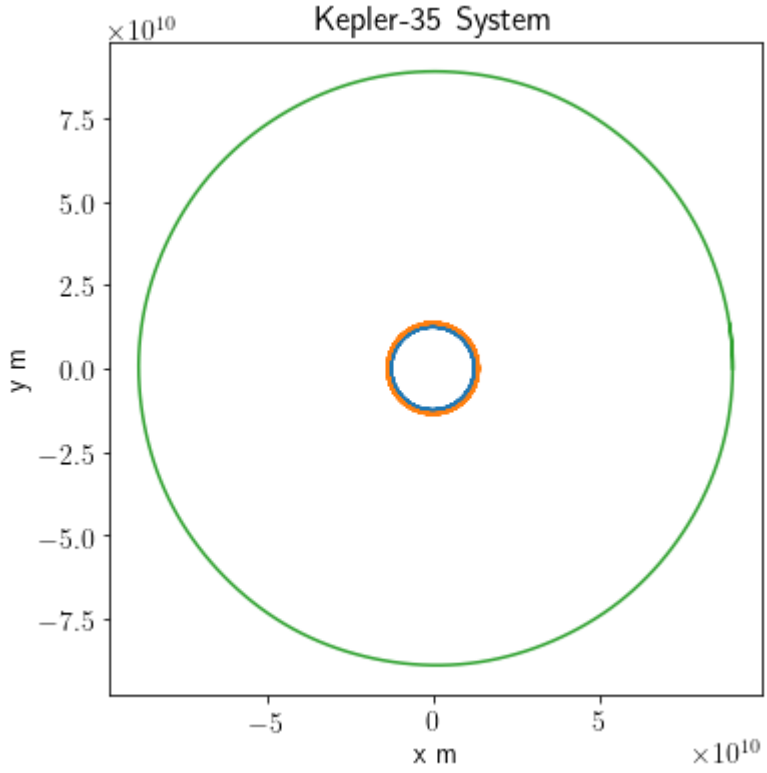
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import integrate
from astropy import constants as const
from astropy import units as u
```

In [2]:
```
plt.rcParams.update({'font.size': 14, 'text.usetex': True})
```

In [3]:
```
#vector input has initial conditions for 3 bodies
def threebodyfunc(vec, t, m3):
    #assigning the initial conditions to vector elements
    x1, y1, z1 = vec[0], vec[1], vec[2]
    vx1, vy1, vz1 = vec[3], vec[4], vec[5]

    x2, y2, z2 = vec[6], vec[7], vec[8]
    vx2, vy2, vz2 = vec[9], vec[10], vec[11]

    x3, y3, z3 = vec[12], vec[13], vec[14]
    vx3, vy3, vz3 = vec[15], vec[16], vec[17]

    #separation vectors
    dx12, dy12, dz12 = x2-x1, y2-y1, z2-z1
    dx13, dy13, dz13 = x3-x1, y3-y1, z3-z1
    dx23, dy23, dz23 = x3-x2, y3-y2, z3-z2

    d12 = np.sqrt(dx12**2+dy12**2+dz12**2)
    d13 = np.sqrt(dx13**2+dy13**2+dz13**2)
    d23 = np.sqrt(dx23**2+dy23**2+dz23**2)

    #acceleration calculations
    ax1 = (G*m2/d12**3)*dx12 + (G*m3/d13**3)*dx13
    ay1 = (G*m2/d12**3)*dy12 + (G*m3/d13**3)*dy13
    az1 = (G*m2/d12**3)*dz12 + (G*m3/d13**3)*dz13
    ax2 = -(G*m1/d12**3)*dx12 + (G*m3/d23**3)*dx23
    ay2 = -(G*m1/d12**3)*dy12 + (G*m3/d23**3)*dy23
    az2 = -(G*m1/d12**3)*dz12 + (G*m3/d23**3)*dz23
    ax3 = -(G*m1/d13**3)*dx13 - (G*m2/d23**3)*dx23
    ay3 = -(G*m1/d13**3)*dy13 - (G*m2/d23**3)*dy23
    az3 = -(G*m1/d13**3)*dz13 - (G*m2/d23**3)*dz23

    #vector with derivatives of initial vector giving velocity and acceleratio
n
    dvec = np.zeros(len(vec))
    dvec[0], dvec[1], dvec[2] = vx1, vy1, vz1
    dvec[3], dvec[4], dvec[5] = ax1, ay1, az1

    dvec[6], dvec[7], dvec[8] = vx2, vy2, vz2
    dvec[9], dvec[10], dvec[11] = ax2, ay2, az2

    dvec[12], dvec[13], dvec[14] = vx3, vy3, vz3
    dvec[15], dvec[16], dvec[17] = ax3, ay3, az3

    return dvec
```

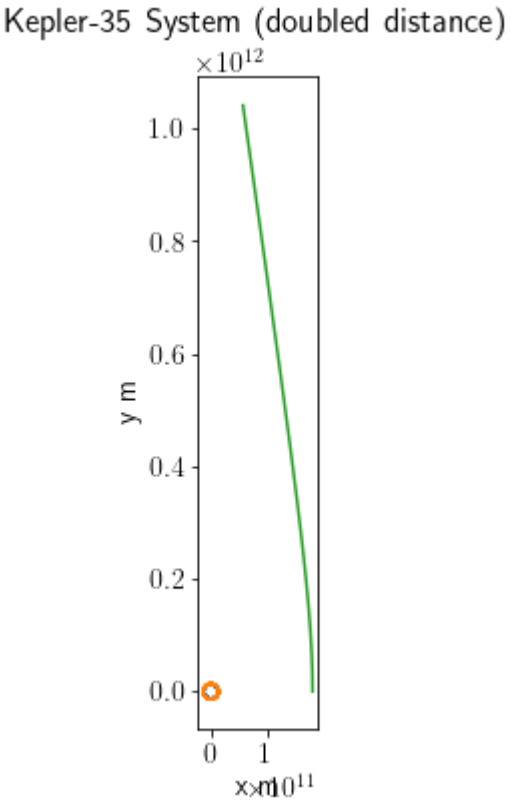
In [4]:
```
#initial conditions for Kepler-35A and B
m1 = (0.8877 * u.Msun).decompose().value
m2 = (0.8094 * u.Msun).decompose().value
mu = (m1*m2)/(m1+m2)
G = const.G.value

a = (0.176 * u.AU).decompose().value

r1 = -a*(mu/m1)
r2 = a*(mu/m2)

P = (20.73 * u.day).decompose().value

v1 = 2*np.pi*r1/P
v2 = 2*np.pi*r2/P
```

In [5]:
```
#including initial conditions for planet b
mb = (0.127 * u.Mjup).decompose().value

ab = (0.60347 * u.AU).decompose().value

rb = ab

#periods
Pb = (131.458 * u.day).decompose().value

#velocities
vb = 2*np.pi*rb/Pb

#initial vectors
vect_b = np.array([r1, 0, 0, 0, v1, 0, r2, 0, 0, 0, v2, 0, rb, 0, 0, 0, vb, 0
])
tarr_b = np.linspace(0,Pb,200)

#integration
ans_b = integrate.odeint(threebodyfunc, vect_b, tarr_b, args=(mb,))

#assigning
x1, y1, z1 = ans_b[:,0], ans_b[:,1], ans_b[:,2]
vx1, vy1, vz1 = ans_b[:,3], ans_b[:,4], ans_b[:,5]
x2, y2, z2 = ans_b[:,6], ans_b[:,7], ans_b[:,8]
vx2, vy2, vz2 = ans_b[:,9], ans_b[:,10], ans_b[:,11]
x3, y3, z3 = ans_b[:,12], ans_b[:,13], ans_b[:,14]
vx3, vy3, vz3 = ans_b[:,15], ans_b[:,16], ans_b[:,17]

plt.figure(figsize=(6,6))
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.xlabel('x m')
plt.ylabel('y m')
plt.title('Kepler-35 System')
plt.gca().set_aspect('equal')
plt.show()
```



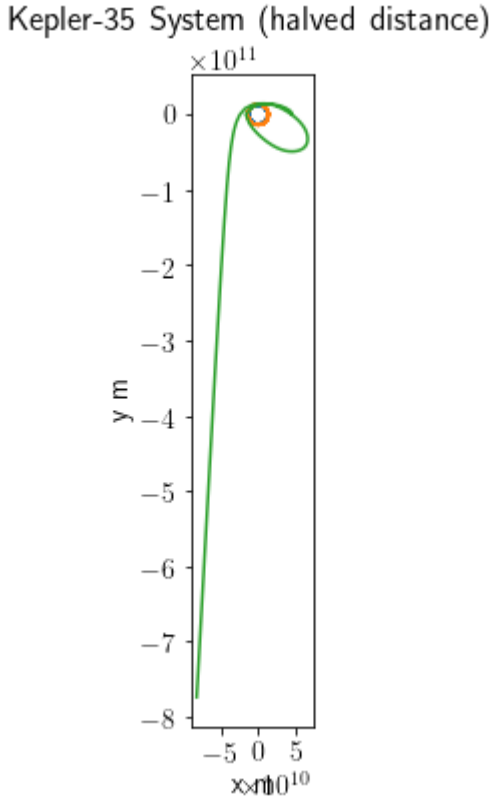
The Kepler-35 system consists of two sun like stars orbiting very close to eachother and a planet that's 10% the size of jupiter.

In [6]:
```
#doubling the distance of the planet
mb = (0.127 * u.Mjup).decompose().value

ab = (2 * 0.60347 * u.AU).decompose().value

rb = ab

#periods
Pb = (131.458 * u.day).decompose().value

#velocities
vb = 2*np.pi*rb/Pb

#initial vectors
vect_b = np.array([r1, 0, 0, 0, v1, 0, r2, 0, 0, 0, v2, 0, rb, 0, 0, 0, vb, 0
])
tarr_b = np.linspace(0,Pb,200)

#integration
ans_b = integrate.odeint(threebodyfunc, vect_b, tarr_b, args=(mb,))

#assigning
x1, y1, z1 = ans_b[:,0], ans_b[:,1], ans_b[:,2]
vx1, vy1, vz1 = ans_b[:,3], ans_b[:,4], ans_b[:,5]
x2, y2, z2 = ans_b[:,6], ans_b[:,7], ans_b[:,8]
vx2, vy2, vz2 = ans_b[:,9], ans_b[:,10], ans_b[:,11]
x3, y3, z3 = ans_b[:,12], ans_b[:,13], ans_b[:,14]
vx3, vy3, vz3 = ans_b[:,15], ans_b[:,16], ans_b[:,17]

plt.figure(figsize=(6,6))
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.xlabel('x m')
plt.ylabel('y m')
plt.title('Kepler-35 System (doubled distance)', y=1.05)
plt.gca().set_aspect('equal')
plt.show()
```



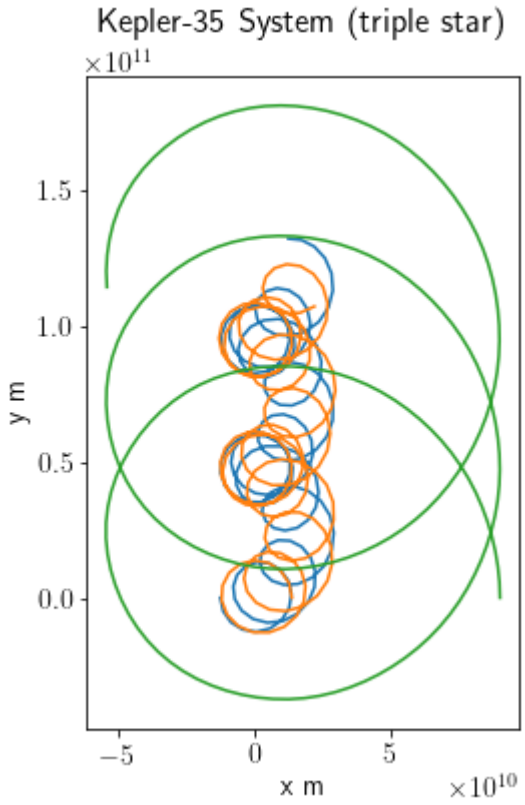
My first change in the initial conditions was to double the distance of the planet from the stars. The result was that the planet simply doesn't revolve around the stars and continues away, effectively becoming a rogue planet. This wasn't completely unexpected as the gravitational pull on the planet by the stars becomes very small at larger distances.

In [7]:
```
#halving the distance of the planet
mb = (0.127 * u.Mjup).decompose().value

ab = (0.5 * 0.60347 * u.AU).decompose().value

rb = ab

#periods
Pb = (131.458 * u.day).decompose().value

#velocities
vb = 2*np.pi*rb/Pb

#initial vectors
vect_b = np.array([r1, 0, 0, 0, v1, 0, r2, 0, 0, 0, v2, 0, rb, 0, 0, 0, vb, 0
])
tarr_b = np.linspace(0,Pb,200)

#integration
ans_b = integrate.odeint(threebodyfunc, vect_b, tarr_b, args=(mb,))

#assigning
x1, y1, z1 = ans_b[:,0], ans_b[:,1], ans_b[:,2]
vx1, vy1, vz1 = ans_b[:,3], ans_b[:,4], ans_b[:,5]
x2, y2, z2 = ans_b[:,6], ans_b[:,7], ans_b[:,8]
vx2, vy2, vz2 = ans_b[:,9], ans_b[:,10], ans_b[:,11]
x3, y3, z3 = ans_b[:,12], ans_b[:,13], ans_b[:,14]
vx3, vy3, vz3 = ans_b[:,15], ans_b[:,16], ans_b[:,17]

plt.figure(figsize=(6,6))
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.xlabel('x m')
plt.ylabel('y m')
plt.title('Kepler-35 System (halved distance)', y=1.05)
plt.gca().set_aspect('equal')
plt.show()
```



My second change to the initial conditions was to halve the distance of the planet from the stars. The orbit starts out with the planet being pulled very close to the stars and then being released, creating an elliptical orbit for a little while. Then the planet is pulled in again and achieves enough velocity to be ejected out of the system, creating a rogue planet. This is also not entirely unexpected as the gravitational forces from the stars are much higher, resulting in higher accelerations.

In [8]:
```
#making the planet a small star
mb = (0.2 * u.Msun).decompose().value

ab = (0.60347 * u.AU).decompose().value

rb = ab

#periods
Pb = (131.458 * u.day).decompose().value

#velocities
vb = 2*np.pi*rb/Pb

#initial vectors
vect_b = np.array([r1, 0, 0, 0, v1, 0, r2, 0, 0, 0, v2, 0, rb, 0, 0, 0, vb, 0
])
tarr_b = np.linspace(0,2*Pb,200)

#integration
ans_b = integrate.odeint(threebodyfunc, vect_b, tarr_b, args=(mb,))

#assigning
x1, y1, z1 = ans_b[:,0], ans_b[:,1], ans_b[:,2]
vx1, vy1, vz1 = ans_b[:,3], ans_b[:,4], ans_b[:,5]
x2, y2, z2 = ans_b[:,6], ans_b[:,7], ans_b[:,8]
vx2, vy2, vz2 = ans_b[:,9], ans_b[:,10], ans_b[:,11]
x3, y3, z3 = ans_b[:,12], ans_b[:,13], ans_b[:,14]
vx3, vy3, vz3 = ans_b[:,15], ans_b[:,16], ans_b[:,17]

plt.figure(figsize=(6,6))
plt.plot(x1,y1)
plt.plot(x2,y2)
plt.plot(x3,y3)
plt.xlabel('x m')
plt.ylabel('y m')
plt.title('Kepler-35 System (triple star)', y=1.05)
plt.gca().set_aspect('equal')
plt.show()
```



My third change to the initial conditions was to replace the planet with a small star of 20% the mass of the sun or 25% the mass of Kepler-35B. The distance from the stars was unchanged, but the time was increased from one revolution of the new star to two to see how the system moves better. The binary system in the center is forced out of place from the original point due to the gravitational force of the new star, but they both continue to orbit each other. This new large mass causes the entire system to shift upwards, but the system itself is "unbroken" in that no body is ejected and the paths are somewhat predictable.