```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

jd = (np.loadtxt("vedit.txt", comments='#'))[:,0]
v_mag = (np.loadtxt("vedit.txt", comments='#'))[:,1]
error = (np.loadtxt("vedit.txt", comments='#'))[:,2]
m_ss = 1.989 * 10**30 #kg
r_ss = 695700 #km

## Locate eclipses around JD2770 the points JD2763 and JD2768
jd_use = jd[np.where((jd>2763) & (jd<2768))]
v_use = v_mag[np.where((jd>2763) & (jd<2768))]

plt.scatter(jd, v_mag,c='r', s=5)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
plt.xlabel("Time in Julian Days", fontsize=16)
plt.ylabel("V_mag of the System", fontsize=16)
plt.title("Visual Magnitude of the binary System", fontsize=20)
plt.show()

plt.scatter(jd_use, v_use,c='r', s=5)
ax = plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
plt.xlabel("Time in Julian Days", fontsize=16)
plt.ylabel("V_mag of the System", fontsize=16)
plt.title("Visual Magnitude of the binary System around JD2770", fontsize=20)
plt.show()

half_period = 2767.7659-2763.71025
period_days = 2 * half_period
period = period_days*86400
print("the period of the system is {:.5} Julian Days or {:.8} seconds".format(period_days,p

jd_phase = jd/period_days
jd_phases = jd_phase % 1
plt.scatter(jd_phases,v_mag, c='b',s=3)
ax=plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
plt.xlabel("Phase in terms of the period", fontsize=16)
plt.ylabel("V_mag of the system", fontsize=16)
plt.title("Visual Magnitude of the binary system", fontsize=20)
plt.show()

#### The times being used for radius calculation
t1 = 0.709945*period
t2 = 0.723673*period
t3 = 0.736974*period
########################
v_p = (np.loadtxt("vel.txt", comments='#'))[:,1] #km/s
v_s = (np.loadtxt("vel.txt", comments='#'))[:,2] #km/s
phases = (np.loadtxt("vel.txt", comments='#'))[:,3] # -0.5 ~ 0.5
testphases = np.linspace(-0.5, 0.5, 1000)

vp_max = np.max(v_p)
vp_min = np.min(v_p)
```

```
vs_max = np.max(v_s)
vs_min = np.min(v_s)

guesses_p = ((vp_max-vp_min)/2,2*np.pi,0.5,((vp_max-vp_min)/2)+vp_min)
guesses_s = ((vs_max-vs_min)/2,2*np.pi,0.5,((vs_max-vs_min)/2)+vs_min)

def fit_function(x, amplitude,frequency,phase,y_offset):
    return amplitude * np.sin(frequency * x + phase ) + y_offset

fit_params_p,fit_cov_p = curve_fit(fit_function,phases,v_p,p0=guesses_p,maxfev=10**5)
fit_params_s,fit_cov_s = curve_fit(fit_function,phases,v_s,p0=guesses_s,maxfev=10**5)

y_guess_p = fit_function(testphases,*fit_params_p)
y_guess_s = fit_function(testphases,*fit_params_s)

plt.scatter(phases, v_p, c='r', label="Primary Star")
plt.scatter(phases, v_s, c='b', label="Secondary Star")
plt.plot(testphases,y_guess_p,linewidth=1, c='r', label="Primary Best fit")
plt.plot(testphases,y_guess_s,linewidth=1, c='b', label="Secondary Best fit")
plt.title("Radial velocities of primary and secondary star", fontsize=20)
plt.xlabel("Arbitrary Phase", fontsize=16)
plt.ylabel("Radial Velocity (km/s)", fontsize=16)
plt.legend(loc="best")
plt.show()
v_pr = -fit_params_p[0]
print("Radial velocity of the primary star is {:.4} km/s".format(v_pr))
v_sr = fit_params_s[0]
print("Radial velocity of the secondary star is {:.4} km/s".format(v_sr))
print("The velocity of the Cente of Mass is {:.4} km/s".format(fit_params_s[3]))
##########Due to the shape of radial velocity and such, the shape is assumed to be circular
a_p = (1000*v_pr*period)/(2*np.pi)
a_s = (1000*v_sr*period)/(2*np.pi)
print("The Semimajor-axis of the primary star is {:.4} km".format(a_p/1000))
print("The Semimajor-axis of the secondary star is {:.4} km".format(a_s/1000))
a_tot = a_p + a_s # in metres
m_ratio = a_s/a_p # mp/ms
m_tot = (4*((np.pi)**2)*((a_tot)**3))/((6.67*10**-11)*(period**2))
m_s = m_tot/(m_ratio+1)
m_p = m_tot - m_s
print("mass of the primary star is {:.4} kg or {:.4} Solar Masses".format(m_p, m_p/m_ss))
print("mass of the secondary star is {:.4} kg or {:.4} Solar Masses".format(m_s, m_s/m_ss))
r_p = (1000*(v_pr+v_sr)*(t3-t2))/2000
r_s = (1000*(v_pr+v_sr)*(t2-t1))/2000
print("The Radius of the primary star is {:.3} km or {:.3} solar radii".format(r_p, r_p/r_s
print("The Radius of the secondary star is {:.3} km or {:.3} solar radii".format(r_s, r_s/r
t4 = 0.208644*period
t5 = 0.223549*period
t6 = 0.237835*period
# Ensuring that the radii are accurate. use the second dip to
r_p2 = (1000*(v_sr+v_pr)*(t5-t4))/2000
r_s2 = (1000*(v_sr+v_pr)*(t6-t5))/2000
print("The Radius of the primary star is {:.3} km or {:.3} solar radii".format(r_p2, r_p2/r
print("The Radius of the secondary star is {:.3} km or {:.3} solar radii".format(r_s2, r_s2
```