Estimating the properties of 51 Pegasi b

November 1, 2020

1 RV curve fitting

```
Shivam Kumaran
SC17B122
```

1.1 Task

To fit the given RV data with a Keplerian orbits, and estimate the best-fit parameters for the exoplanet 51 Pegasi b. using the data given

1.2 Steps followed

Obtain the data from data file

Plot the data with errors

Use LS periodogram to obtain periodicity in the data

Fold radial velocity data for the period calculated in above step

Used Sci-py curve fit module to find best-fit model and thee errors in the parameter

Used χ^2 test to quantify goodness of fit

Used error propagation to calculate errors

1.2.1 Importing required Modules

[35]:
```
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.gridspec as gs
plt.style.use('seaborn-darkgrid')
```

Given values

[203]:
```
m_sun = 1.988e+30
m_pegasi = 1.1*m_sun
m_j = 1.89e+27
G = 6.67259e-11
AU = 1.496e+11
```

[139]:
```
data = np.loadtxt('data.txt' , encoding='ASCII' , delimiter=' ').T
time = data[0]*(24*60*60)
vel = data[1] - np.mean(data[1]) # detrending vel data
vel = data[1]
err = data[2]
t_0 = time[0]
t_off = time - time[0] #removing offset fr
```

1.3 Quick look at the data

[169]:
```
plt.figure(figsize=(8,5))
plt.errorbar(time/86400, vel, fmt='.' , yerr=err,capsize=2, ecolor='k',)
plt.xlabel('TIME [days] (offset by {:.2f})'.format(t_0/86400))
plt.ylabel('Radial velocity (m/s)')
plt.title('Radial velocity vs, time ')
plt.show()
```

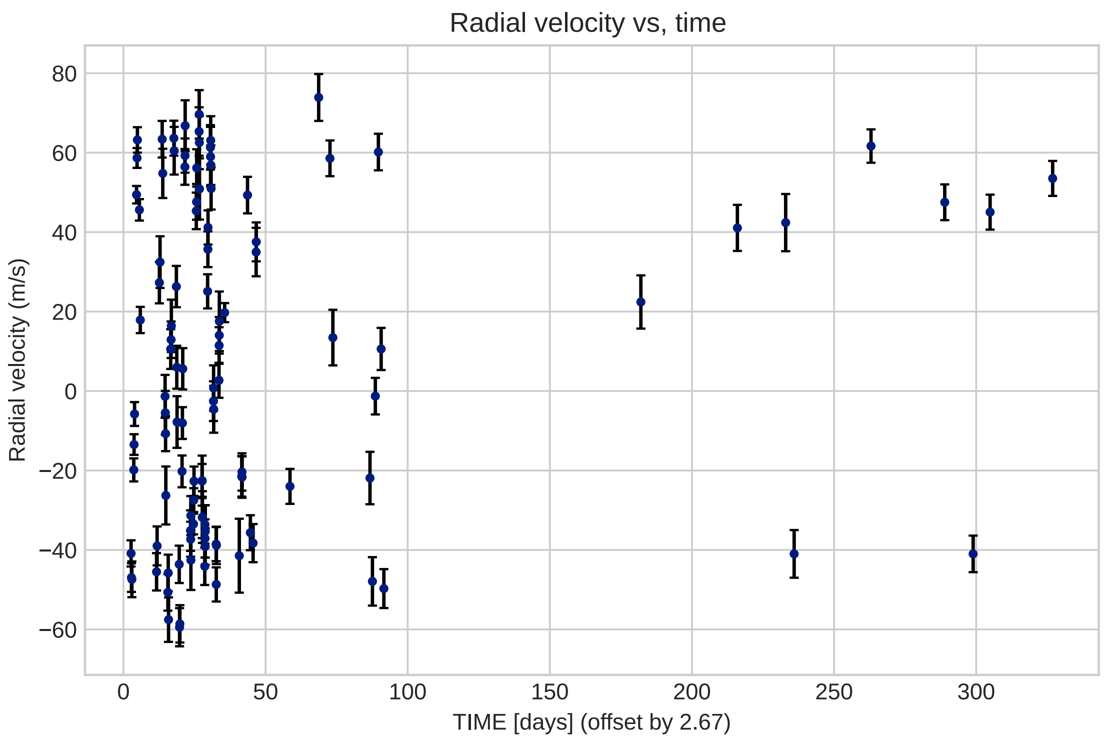

FIgure i : Values of radial velocities (in m/s) vs time (days) with errorbars corresponding to sigma errors in the given data.

1.4 Finding period

Using lombscargle periodogram to find periodicity in the data

```
[166]: from astropy.timeseries import LombScargle
       freq ,  power = LombScargle(time,vel).autopower()
       plt.figure(figsize=(6,4))
       plt.axvline(x=freq[p_max] , color='crimson')
       plt.plot(freq,power)
       p_max = np.argmax(power)
       period = 1/freq[p_max]
       plt.xlabel('frequency (Hz)')
       plt.ylabel('Power')
       plt.legend(['Freq for max power {:.2e} '.format(freq[p_max])])
       plt.show()
       print('period: ' , period , 'sec')
```

```
period:  365225.69804432866 sec
```

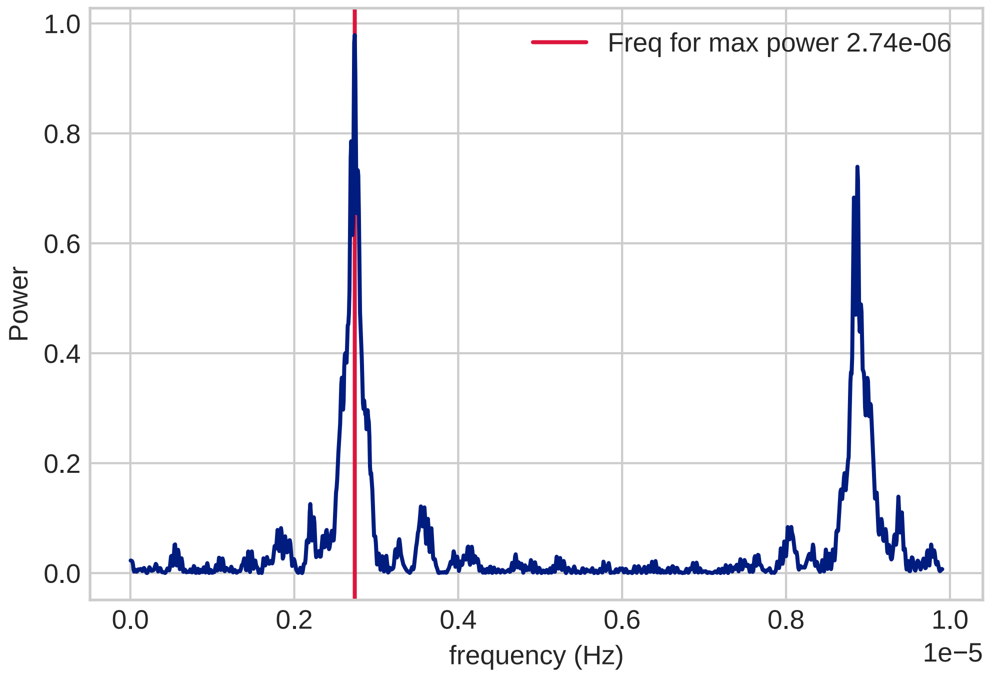

Figure ii : Lombescargle periodogrom for the radial velocity data . Frequency corresponding to maximum power is shown with crimson colour. The other maxima in the plot correspond to harmonics of this frequency.

2 Data Folding

[167]:
```
def fold_data(p , data):
    if(data[0]>p):
        data[0] = data[0] - p
        fold_data(p,data)
    return data
```

[168]:
```
folded_data = []
for t , v, e in zip(t_off , vel , err):
    temp = fold_data(period , [t,v,e])
    folded_data.append(temp)
folded_data = np.asanyarray(folded_data)
folded_data = folded_data[np.argsort(folded_data[:,0])]
#print(s_folded_data)
#print(folded_data[:,0])
```

[171]:
```
plt.figure(figsize=(8,5))
time_f = folded_data[:,0]
vel_f = folded_data[:,1]
err_f = folded_data[:,2]
plt.errorbar(time_f/86400,vel_f,fmt='.' , yerr=err_f ,)
plt.title('Folded radial velocity data')
plt.xlabel('TIME [days] (offset by {:.2f})'.format(t_0/86400))
plt.ylabel('Radial velocity (m/s)')
plt.show()
```

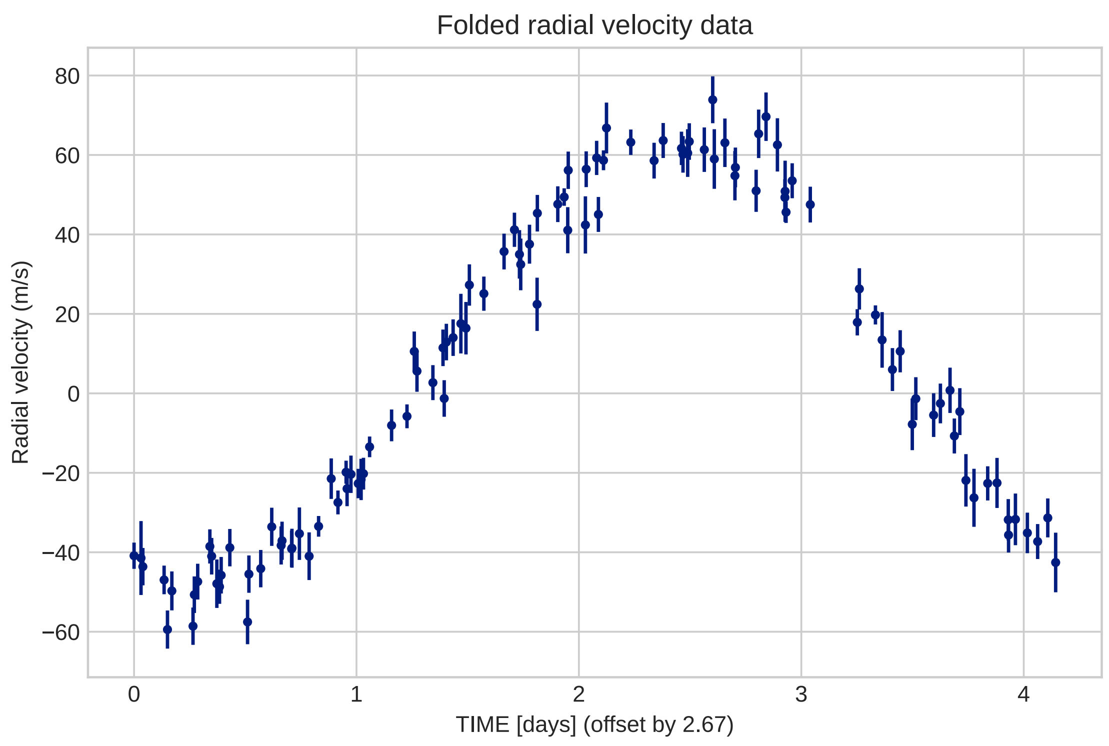

Figure iii : Plot of radial velocity data folded with the time period calculated above.

3 Curve fitting

[172]:
```
def time_to_theta(theta_0 , t , p , e):
    k = (2*np.pi/p)*((1-e**2)**(-3/2))
    def calc(dt):
        theta = theta_0
        n = int(t/dt)
        for i in range(n):
            d_theta = k*((1+e*np.cos(theta))**2)*dt
            theta += d_theta
        return theta
    '''
    chooses the value of dt such that
    convergence is of the order of 1E-5
    increases calculation
    '''
    err  = 1e-5
    dt = 1000
    err_calc = 1
    prev = calc(dt)
    nxt =calc(dt/2)
    while(err_calc > e):
        dt = dt/2
        prev = nxt
        nxt = calc(dt)
        err_calc = abs(prev-nxt)

    return nxt
def calc_vel(t , ks , theta_0, omega, p , e , const ):
    #print('doing for dime :',t)
    t = np.asarray(t)
    theta = np.asarray([time_to_theta(theta_0 , time, p, e) for time in t])
    vel = ks*(e*np.cos(omega)+np.cos(omega+theta))+const
    #print(vel)
    #return np.cos(t)
    return vel
```

Using scipy curve fitting routine.

[174]:
```
from scipy.optimize import curve_fit
```

[175]:
```
params , pcov = curve_fit(calc_vel,time_f, vel_f, sigma=err_f, p0=[50 ,0, np.pi/
 ↪2 , period , 0.01 , 10] , absolute_sigma=True)
```

[176]:
```
k_st = params[0]
theta_0 = params[1]
omega = params[2]
p = params[3]
ecc = params[4]
const = params[5]
print('k*:' , k_st)
print('theta_0 : ' , theta_0)
print('omega:' , omega)
print('Period : ' , p)
print('eccentricity :' , ecc)
print('absolute velocity : ' , const)
```

```
k*: 58.00000011461893
theta_0 :  5.472973210161071e-07
omega: 2.770795804682829
Period :  365225.69924624497
eccentricity : 0.012999996857545902
absolute velocity :  5.99999981720868
```

[215]:
```
#print(pcov)
perr = np.sqrt(np.diag(pcov))
k_st_err = perr[0]
```

```
omega_err = perr[2]
p_err = perr[3]
ecc_err = params[4]
print(perr)
```

```
[6.59225337e-01 1.20773123e-01 1.39586849e-01 6.12816573e+03
 1.72618950e-02 9.72973372e-01]
```

3.1 Calculating Error ,

$V_{model} - v_{observed}$

```
[231]: vel_mod = calc_vel(time_f , k_st , theta_0 , omega , p , ecc , const)
       res = vel_mod - vel_f
```

```
[192]: import matplotlib as mpl
       mpl.rcParams.update(mpl.rcParamsDefault)
       plt.style.use('seaborn-whitegrid')
       plt.style.use('seaborn-dark-palette')
       fig = plt.figure(figsize=(8,6) , constrained_layout=0)
       #plt.title('Model fit curve with observed data and residuals')
       spec = gs.GridSpec(nrows=2,ncols=1, figure=fig ,  height_ratios=[2,1] , hspace=0)
       acc = fig.add_subplot(spec[0,0] )
       acc.plot(time_f/86400 , vel_mod , '-')
       acc.errorbar(time_f/86400,vel_f,fmt='.' , yerr=err_f , capsize=2)
       acc.set_ylabel('Radial velocity (m/s)')
       #acc.xaxis.set_visible(False)
       acc.legend(['Model','observed'])
       loss = fig.add_subplot(spec[1,0], sharex = acc)
       loss.errorbar(time_f/86400,res,fmt='.' , yerr=err_f , capsize=2 , color='k' ,␣
        ↪ecolor='k')
       loss.set_ylabel('Residuals')
       loss.legend(['residuals'])
       plt.xlabel('TIME [days] (offset by {:.2f})'.format(t_0/86400))
       plt.show()
```

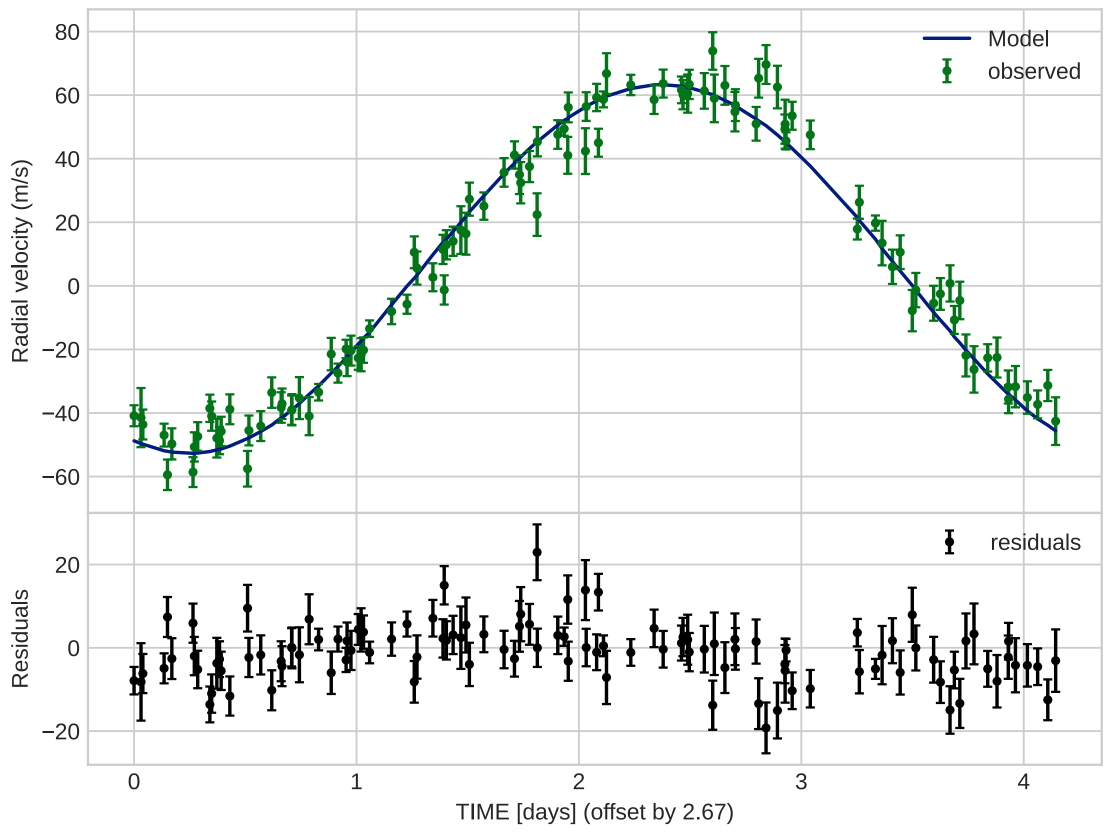

FIgure iv : Top panel - Best-fit Model plotted together with the observed data bottom panel - error model and observed data , errorbars on residuals are sigma error corresponding to observed data

$res = v_{model} - v_{observed}$

$\Delta res = \Delta v_{observed}$

3.1.1 Chi- sq test

```
[194]: chi_sq = sum([((e-o)**2)/err**2 for e,o,err in zip(vel_mod,vel_f,err_f)])
print(r'$\chi{sq}$',chi_sq)
print(r'reduced $\chi_{sq}$ = ',chi_sq/(len(vel_f)-len(params)))
```

3.1.2 Result

1. DOF = No of data points - no of parameters used for fitting = 111 - 6 = 105
2. $\chi^2 = 194.233$
3. Reduced $\chi^2 = \frac{\chi^2}{DOF} = 1.85$

4 Value calculations

[221]:
```
P = p
e = ecc
k = k_st
#m_p sin i in SI units
const_coeff = ((m_pegasi**(2))/(2*np.pi*G))**(1/3)
m_p_sin_i_si = p**(1/3)*const_coeff*k
del_mp_sin_i = const_coeff*(1/3)*p**(-2/3)*p_err*k + const_coeff*p**(1/
 ↪3)*k_st_err

m_p_sini = m_p_sin_i_si/m_j
del_mp_sin_i = del_mp_sin_i / m_j

a_s = (((p**2)*G*(m_p_sin_i_si**3))/(4*np.pi**2*(m_pegasi**2)))**(1/3)
a = (a_s + (m_pegasi/m_p_sin_i_si)*a_s)/AU
```

[230]:
```
print('Radial vel semi-amp : {:.2f} m/s , error({:.2f}m/s)'.format(k_st ,␣
 ↪k_st_err))
print('Orbital Period : {:.3f} days , error({:.5e} days)'.format(p/86400 , ␣
 ↪p_err/86400))
print('Phase angle : {:.2f} , error ({:.2f})'.format(omega*(180/np.pi) ,␣
 ↪omega_err*(180/np.pi)))
print('Eccentricity : {:.4f} , error({:.4f})'.format(ecc ,ecc_err))
print('Orbital Speration : {:.4f} AU'.format(a))
print('Mass of planet: {:.2f} M_j (err : {:.4f})'.format( m_p_sini ,␣
 ↪del_mp_sin_i))
```

```
Radial vel semi-amp : 58.00 m/s , error(0.66m/s)
Orbital Period : 4.227 days , error(7.09278e-02 days)
Phase angle : 158.75 , error (8.00)
Eccentricity : 0.0130 , error(0.0173)
Orbital Speration : 0.0528 AU
Mass of planet: 0.49 M_j (err : 0.0084)
```

[ ]:

5 Result

5.1 Model

: reduced- chi sq = 1.85

5.2 Estimated values

Sl No	**Name**	**Unit**	**Estimated value**	**Published value**
1	V_r **semi-amplitude**	m/s	58.00 (± 0.66)	55.65 (± 0.53)
2	**Orbital Period**	days	4.227 (± 7.09278e-02)	4.2308 (± 3.72905e-05)
3	**Eccentricity**		0.0130 (±0.0173)	0.0069 (-0.0066 +0.0069)
4	**Orbital seperation**	AU	0.0528	0.052
5	**mP sin i**	Jupiter mass Mj	0.49 (±0.0084)	0.46 (-0.06 +0.01)
6	ω	deg	158.75 (±8.00)	54.1 (± 72.3)

Table : 01 Values of parameters calculated from radial velocity curve fitting. Comparison between estimated values and published values