Planetary Sciences Assignment 4
Estimating the properties of 51 Pegasi b

Kiran L
SC17B150

2nd November 2020

1 Introduction

Most of the studies of the planets orbiting stars other than the Sun are possible only through the indirect methods, owing to the limitations of visibility of the planets due to their small size. One of the popular indirect methods is the radial velocity method, in which, various physical characteristics of the planet are determined using the shift in the spectral lines in the spectrum of the star. The wavelength of each spectral line changes due to reflex motion of the star around the planet, which is observed as red and blue shifts in the spectrum of the source. In this assignment, we analyse observational data of time variation of radial velocity. The major tasks performed are:

1. visualizing the time-series data, plotted along with the associated uncertainty values
2. determining the periodicity in the data through periodogram analysis
3. using the determnined period to fold and re-visualize the data to get a better picture of the sampling of the radial velocity values over a period
4. using the previously written codes for synthetically generating radial velocity curves to fit the data to the model and hence, determine the parameters of the source
5. plotting the best fit model over the data for visualizing the goodness of fit

2 Data Visualization

Observations
From the Figure 1 we observe that the radial velocity is unevenly sampled. The observation are conducted at non-uniformly. Further, we observe that the source (51 Pegasi and the associated planet) is observed frequently between 11/10/1995 to 28/11/1995 (since, the data points are densely populated in the above plot)

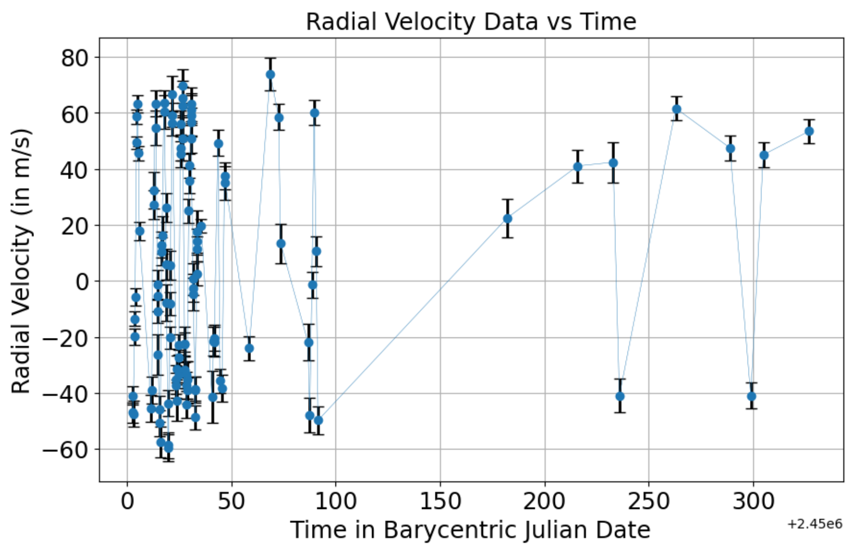

Figure 1: Plot of measured values of radial velocity as a function of time of observation (with line joining the data points)

and the observations are relatively sparse with increasing time (only $\approx$ 20 observations post 28/11/1995 against $\approx$ 80 observations between aforementioned dates).

Inferences
The primary reasons for the non-uniform sampling of the data could be:

1. Weather conditions unfavourable for observations
2. Unavailability of telescope time

Since, the error values in time measurement are not specified, we infer that the uncertainty in the measurement of time is negligible.

3 Periodogram Analysis

The Lomb-Scargle periodogram (after Lomb [1], and Scargle [2]) is a commonly used statistical tool designed to detect periodic signals in unevenly spaced observations. The astropy.timeseries LombScargle class is a unified interface to several implementations of the Lomb-Scargle periodogram, including a fast O[NlogN] implementation following the algorithm presented by Press & Rybicki [3].

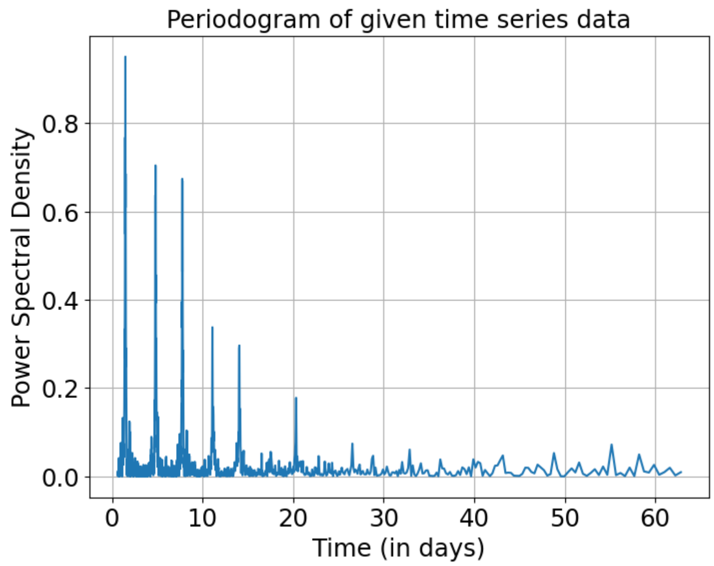

Figure 2: Periodogram of given time series data of radial velocity of 51 Pegasi (produced using scipy.signal's lombscargle function)

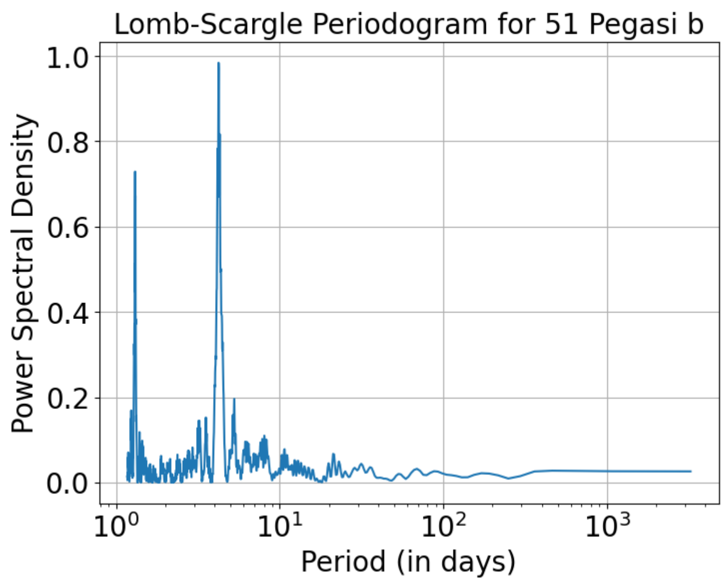

Figure 3: Periodogram of given time series data of radial velocity of 51 Pegasi (produced using astropy.timeseries's LombScargle function)

Observations

We observe peaks at harmonics of fundamental period (4.232 Earth days) ie., peaks at 8.464, and 12.696 Earth days in the periodogram obtained using Scipy module functions. We also observe a spurious spike close to 15 Earth days the source of which is unknown.

Inferences

From the Figure 2 and 3 we infer that the predicted values of the periodicity in the 51 Pegasi radial velocity time-series data is 4.232 ± 0.0005 Earth Days (where, half the step size of the period array is reported as the uncertainty value in the period of the data).

From the Figure 3 we observe and infer that the predicted values of the periodicity in the 51 Pegasi radial velocity time-series data is 4.2271 ± 0.0111 Earth Days (where, the maximum of the step size in either direction around the reported value is reported as the uncertainty value in the period of the data, since the frequencies to search for was taken automatically by the algorithm, not given as user input). No significant peaks are observed at harmonics of fundamental period.

4 Time-Folded Radial Velocity Curve

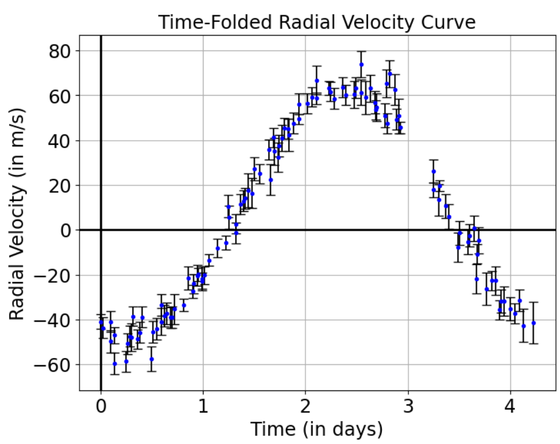

Figure 4: Time-Folded Radial Velocity Curve (with t = 0 sec corresponding to the starting value of the data) plotted with period determined using scipy.signal's lombscargle function

Observations

From the Figure 4 and 5 we observe that the non-unifom sampling of the data has resulted in loss of data points (radial velocity measurement values) corresponding to the time duration between 3 and 3.2 Earth days. The errorbars associated with the radial velocity curves is observed to be $\sim 3-4$ m/s.

Transforming the time to phase, the plot of radial velocity curve as a function of orbital phase is displayed in Figure 7 and 6. We observe that the absence of data samples corresponding to the orbital phase between $\frac{4\pi}{3}$ and $\frac{3\pi}{2}$.

Also, we observe that the small variation in the period values (as determined from astropy and scipy module functions) does not significantly alter the phase-folded or time-folded plot of radial velocity.

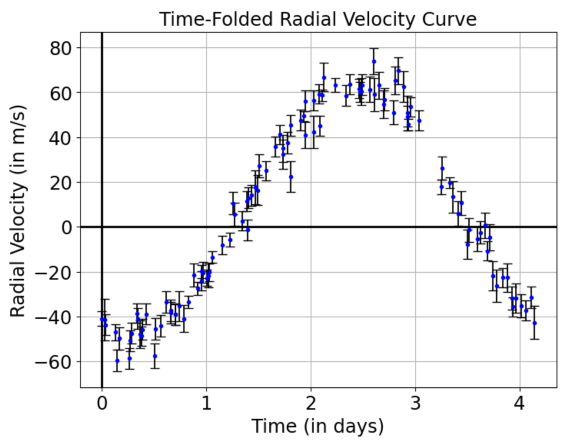

Figure 5: Time-Folded Radial Velocity Curve (with t = 0 sec corresponding to the starting value of the data) plotted with period determined using astropy.timeseries's LombScargle function

Inferences

From the values of errorbars, we infer that the measurement of the radial velocity values are very precise. Also, since, the error values in time measurement are not specified implies high accuracy in time measurement. The poor sampling of the radial velocity values corresponding to the phase between $\frac{4\pi}{3}$ and $\frac{3\pi}{2}$ are is most likely to be coincidence, which can be rectified though careful observations of the source in the future observations. The spread in the radial velocity values in a given time interval can be due to:

1. the observations are perfectly identical (very unlikely) and the system is dynamically evolving
2. the other possibilities include
   (a) different atmospheric conditions on different observing days,
   (b) difference in instrumental setup (i.e., observations taken from different observatories

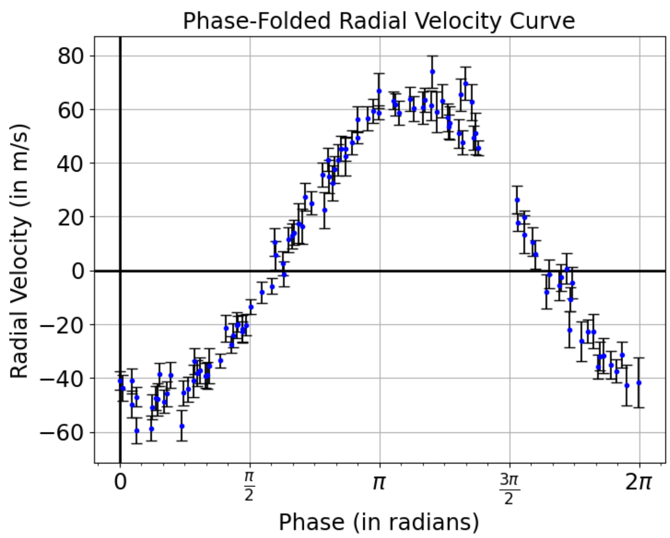

Figure 6: Phase-Folded Radial Velocity Curve (with t = 0 sec corresponding to the starting value of the data) plotted with period determined using scipy.signal's lombscargle function

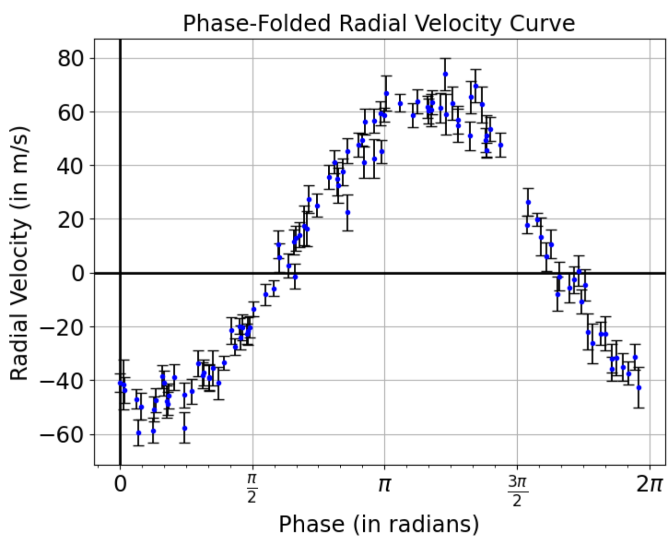

Figure 7: Phase-Folded Radial Velocity Curve (with t = 0 sec corresponding to the starting value of the data) plotted with period determined using astropy.timeseries's LombScargle function

5 Comparison of model fit parameter values with published ones

In this section, we compare and data with the best fit model with four free parameters:

1. Orbital Eccentricity (e)
2. Semi-major amplitude (k)
3. Angle of periastron (ω)
4. Constant radial velocity of system ($v_{r.0}$)

The function used to fit the model radial velocity curve to the data is:

$$v_r = K(ecos(\omega) + cos(\omega + \theta)) + v_{r,0}$$

where, $K = \frac{2\pi a_* sin(i)}{P\sqrt{1-e^2}}$. The offset value is the constant radial velocity of the system (positive if the system is moving away from the observer and negative otherwise).

The period values are frozen at the values corresponding to the value determined from periodogram. Since, the period values determined from astropy and scipy module functions is different, we analyse the best fit model determined separately assuming aforementioned values of period.

5.1 Results assuming period determined using Scipy

The present subsection describes the results obtained upon fitting the model radial velocity curves to data using the period determined using scipy.signal's lombscargle function. The Table 1 displays the comparison of the estimated values of the parameters from grid search to that published. The Figure 8 and 9 displays the Folded RV curve with the best-fit model superimposed on the data, where the fit is performed using scipy's curve_fit function. From the fit we obtained $\chi^2_{107} = 1.02192$ (where, degrees of freedom is 4 (since four parameters are being fit - e, k, ω and $v_{r,0}$). The fit statistics and the parameter values of the best fit model are displayed in Table 2.

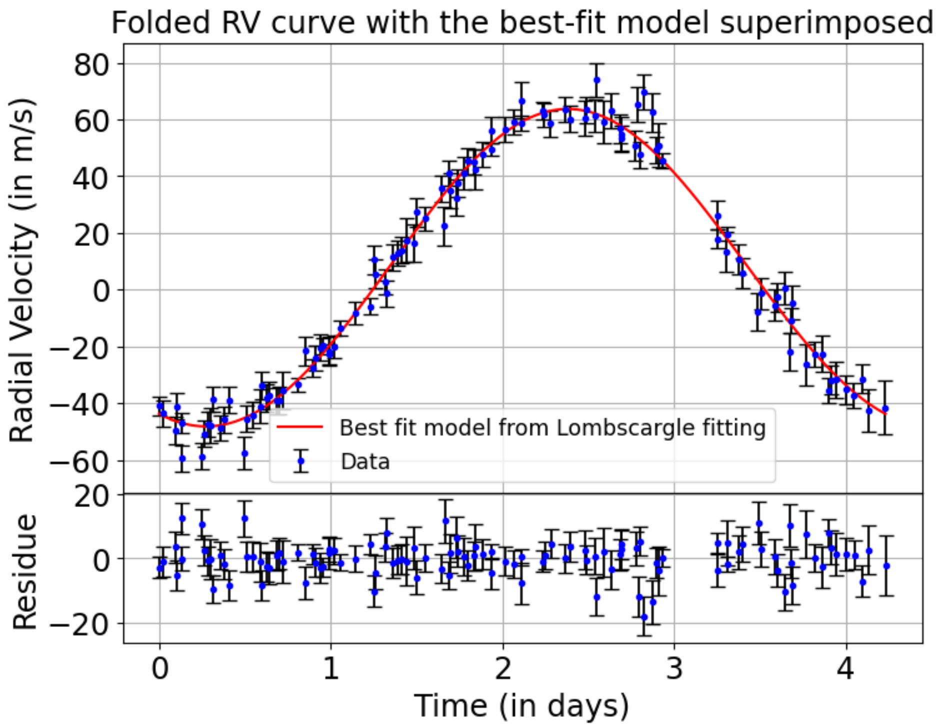

Figure 8: Folded RV curve with the best-fit model superimposed (top, line plot) and residual plot (bottom)

Exoplanet property	**Value estimated from best-fit**	**Value in published works**
$m_p sin(i)$ (in M_J)	0.477 ± 0.017	0.461 ± 0.0164[8]
Orbital Period (P, in Earth days)	4.2307 ± 0.0005	$4.230785 \pm (3.6 \times 10^{-5})$ [8]
Orbital Eccentricity (e)	0.017 ± 0.021	0.013 ± 0.012 [8]
Orbital Separation (a, in AU)	0.053 ± 0.001	0.052 AU [6]
Radial velocity semi-amplitude (in m/s)	55.92 ± 0.6	55.94 ± 0.69 [8]
Orbital semi-major axis of star projected perpendicular to sky plane ($a_* sin(i)$, in m)	$(32.528 \pm 0.351) \times 10^5$	$(34 \pm 2) \times 10^5$ [7]

Table 1: Comparison of parameter values determined by Keplerian Orbit fitting and that published

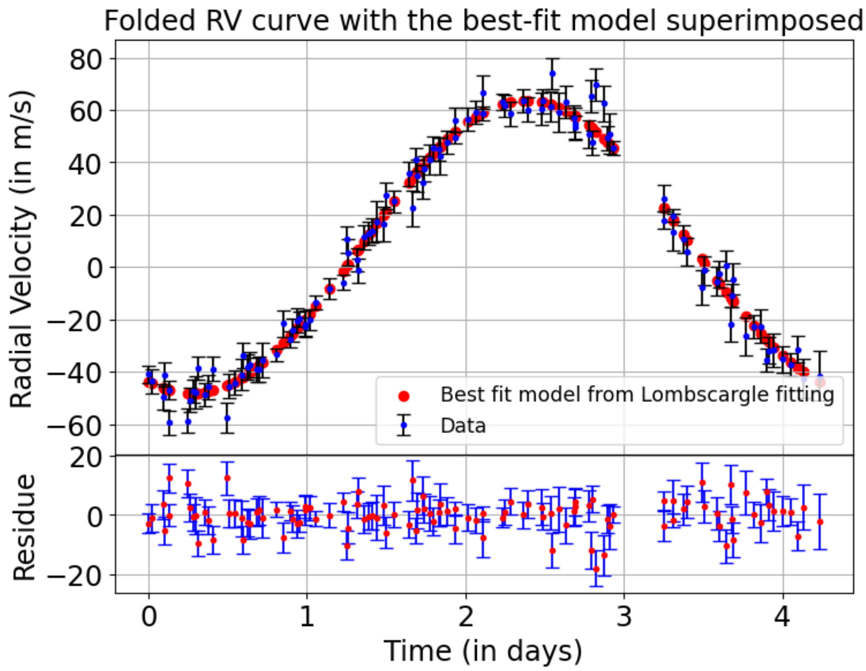

Figure 9: Folded RV curve with the best-fit model superimposed (top, scatter plot) and residual plot (bottom)

Fit parameter	**Value estimated from best-fit**
ω (in ^{o})	-22.93 ± 0.64
Orbital Eccentricity (e)	0.017 ± 0.021
Reduced Chi-square value (χ^2_{107})	1.02192
Number of degrees of freedom	4
Radial velocity semi-amplitude (in m/s)	55.92 ± 0.6
Constant radial velocity ($v_{r,0}$, in m/s)	6.918 ± 0.709

Table 2: Fit statistics and parameter values

5.2 Results assuming period determined using Astropy

The present subsection describes the results obtained upon fitting the model radial velocity curves to data using the period determined using astropy.timeseries's LombScargle function. The Table 3 displays the comparison of the estimated values of the parameters from grid search to that published. The Figure 10 and 11 displays the Folded RV curve with the best-fit model superimposed on the data, where the fit is performed using scipy's curve_fit function. From the fit we obtained $\chi^2_{107} = 1.31229$ (where, degrees of freedom is 4 (since four parameters are being fit - e, k, ω and $v_{r,0}$). The fit statistics and the parameter values of the best fit model are displayed in Table 4.

Exoplanet property	**Value estimated from best-fit**	**Value in published works**
$m_p sin(i)$ (in M_J)	0.477 ± 0.017	0.461 ± 0.0164[8]
Orbital Period (P, in Earth days)	4.2271 ± 0.0111	$4.230785 \pm (3.6 \times 10^{-5})$ [8]
Orbital Eccentricity (e)	0.007 ± 0.021	$0.0069^{+0.0069}_{-0.0066}$ [5]
Orbital Separation (a, in AU)	0.053 ± 0.001	0.052 AU [6]
Radial velocity semi-amplitude (in m/s)	55.92 ± 0.61	55.94 ± 0.69 [8]
Orbital semi-major axis of star projected perpendicular to sky plane ($a_* sin(i)$, in m)	$(32.506 \pm 0.362) \times 10^5$	$(34 \pm 2) \times 10^5$ [7]

Table 3: Comparison of parameter values determined by Keplerian Orbit fitting and that published

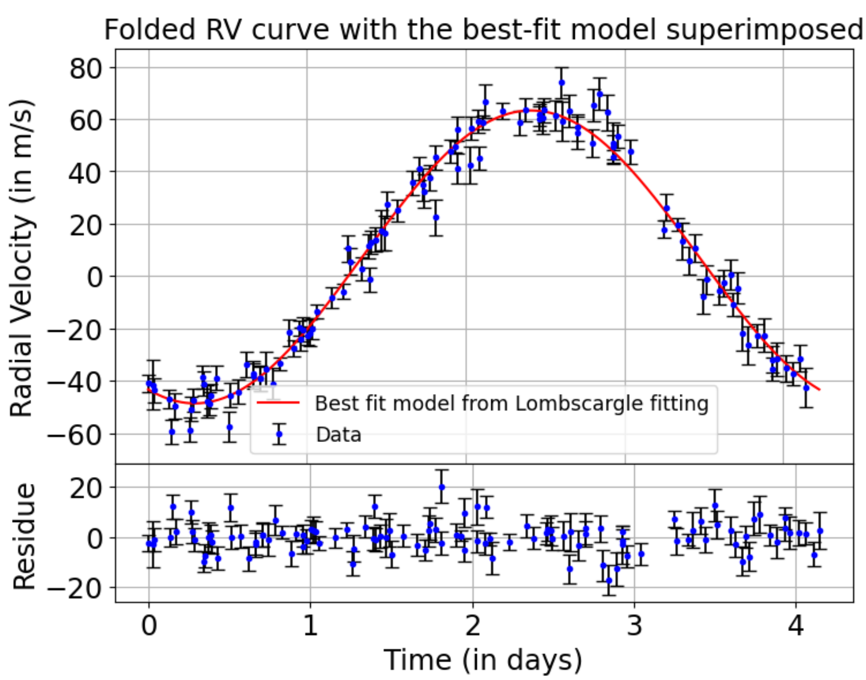

Figure 10: Folded RV curve with the best-fit model superimposed (top, line plot) and residual plot (bottom)

Fit parameter	Value estimated from best-fit
ω (in ^{o})	154.77 ± 0.63
Orbital Eccentricity (e)	0.007 ± 0.021
Reduced Chi-square value (χ^2_{107})	1.31229
Number of degrees of freedom	4
Radial velocity semi-amplitude (in m/s)	55.92 ± 0.61
Constant radial velocity ($v_{r,0}$, in m/s)	7.661 ± 0.692

Table 4: Fit statistics and parameter values

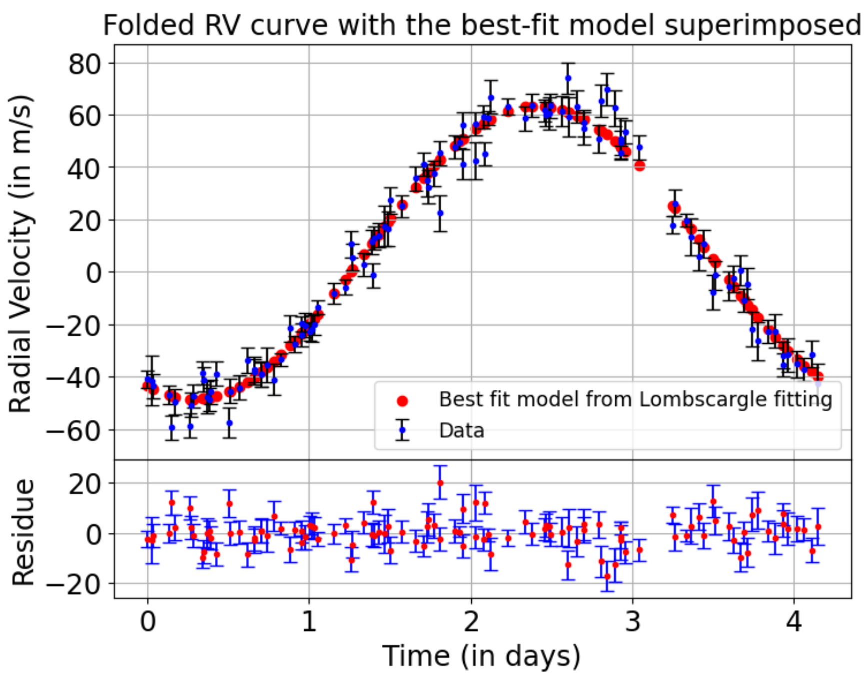

Figure 11: Folded RV curve with the best-fit model superimposed (top, scatter plot) and residual plot (bottom)

6 Conclusion

Through this analysis, we have been able to determine the properties of 51 Pegasi b from the radial velocity measurement data. The fit parameters are: eccentricity (e), semi-amplitude (k), angle of periastron (ω) and constant radial velocity ($v_{r,0}$) under the assumption that the orbit of star (hence, that of planet), is dynamically stable.

From the value of eccentricity, we infer that the orbit of the planet (and hence, the star) around their common center of mass is close to perfect circular orbit. We observe that the values of the parameters obtained matches with that published (as can be seen from the section on comparison of best fit model based estimated parameters and that published. Though the reduced chi-square value of the fit using period determined from astropy.timeseries's LombScargle function is higher than that obtained using that from scipy.signal's lombscargle function, the match between the estimated values of the parameters from best fit model is better in case of astropy than period determined using scipy routines.

Thus, we have fit the model radial velocity curves to the data of 51 Pegasi and 51 Pegasi b and have obtained values of the physical properties of the system that matches with those published by various authors.

7 Program

rvc_astropy.py

```
##################################
###--- Importing Libraries ---####
##################################

import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.signal import lombscargle
from astropy.stats import LombScargle
from astropy import timeseries
from scipy.optimize import curve_fit,fsolve
from astropy import constants as const
from astropy import units as u
from scipy import stats

##################################
####--- Function and Class ---####
####---     Defintion     ---#####
##################################

def multiple_formatter(denominator=2, number=np.pi, latex='\pi'):
    def gcd(a, b):
        while b:
            a, b = b, a%b
        return a
    def _multiple_formatter(x, pos):
        den = denominator
        num = np.int(np.rint(den*x/number))
        com = gcd(num,den)
        (num,den) = (int(num/com),int(den/com))
        if den==1:
            if num==0:
                return r'$0$'
            if num==1:
                return r'$%s$'%latex
            elif num==-1:
                return r'$-%s$'%latex
            else:
                return r'$%s%s$'%(num,latex)
        else:
```

```
            if num==1:
                return r'$\frac{%s}{%s}$'%(latex,den)
            elif num==-1:
                return r'$\frac{-%s}{%s}$'%(latex,den)
            else:
                return r'$\frac{%s%s}{%s}$'%(num,latex,den)
    return _multiple_formatter

class Multiple:
    def __init__(self, denominator=2, number=np.pi, latex='\pi'):
        self.denominator = denominator
        self.number = number
        self.latex = latex

def stopping_criteria(x1,x2,stop_type):
        if(stop_type == "abs"):
                return np.abs(x2 - x1)
        if(stop_type == "rel"):
                return np.abs(x2/x1 - 1.0)

def bisection(f,x1=0.0,x2=2*np.pi,max_iters=1000,xtol=1e-6,ftol=1e-4,*args):
    if(f(x1,*args)*f(x2,*args)>0):
        print("Both f(x1) and f(x2) have same sign! There possibly does not exist a unique root in the given interval. ")
        return
    elif(f(x1,*args) == 0):
        return x1, 0
    elif(f(x2,*args) == 0):
        return x2, 0
    tol = stopping_criteria(x1,x2,"abs")
    x0 = (x1+x2)/2.0;
    ctr = 0
    while(stopping_criteria(x1,x2,"abs")>xtol and f(x0,*args) > ftol and ctr<max_iters):
        x1,x2 = np.where(f(x1,*args)*f(x0,*args) < 0,(x1,x0),(x0,x2))
        ctr += 1
    return x0,ctr

def get_vr(t,K,P,e,w,tau,vr0,return_theta=False):
  E = get_E((t-tau)*2*np.pi/P,e)
  theta = 2*np.arctan2(np.sqrt(1.+e)*np.sin(0.5*E),
                       np.sqrt(1.-e)*np.cos(0.5*E))
  if(not return_theta):
    return K*(np.cos(theta+w)+e*np.cos(w))+vr0
  else:
    return K*(np.cos(theta+w)+e*np.cos(w))+vr0, theta


def get_E(M,e):
  E = []
  for i,mi in enumerate(M):
    x1 = mi - e
    x2 = mi + e
    E_tmp = bisection(lambda E: E-e*np.sin(E)-mi,x1,x2,1000,1e-6,1e-4)[0]
    E.append(E_tmp)
  return np.array(E)

def reduced_chi_square(x,y,s,m): # ddof = v
  v = x.size - m
  chi2 = (np.sum((x-y)**2/s**2))/v
  p = 1 - stats.chi2.cdf(chi2, v)
  return chi2,p

##################################
###---      Data loading    ---####
###---   and Visualization  ---####
##################################

```

```python
data = np.loadtxt("./RV_51Pegasi_Data.txt",dtype = [("time","f8"),("vr","f8")
    ,("dvr","f8")])
time = data["time"].astype(float) # time in days
vr = data["vr"].astype(float) # vr in m/s
dvr = data["dvr"].astype(float) # dvr in m/s
JD_ref = 2450000
time_jd = time+JD_ref
time0 = time-time[0]
ft = 17.5
fig = plt.figure(figsize=(10,6))
plt.errorbar(time+JD_ref,vr,yerr=dvr,fmt='o',ls='-',lw=0.3, capsize=4,ecolor=
    'k',elinewidth=2)
plt.xlabel("Time in Barycentric Julian Date",fontsize=ft)
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.title("Radial Velocity Data vs Time",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.savefig("rvdata_wline.png",bbox_inches="tight")
plt.close()

##################################
###--- Periodogram Analysis ---###
##################################

frequency,power = timeseries.LombScargle(time_jd,vr,dvr).autopower()
ft = 20
fig = plt.figure(figsize=(8,6))
plt.semilogx(1/frequency,power)
plt.title('Lomb-Scargle Periodogram for 51 Pegasi b',fontsize=ft)
plt.xlabel('Period (in days)',fontsize=ft)
plt.ylabel('Power Spectral Density',fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.savefig("periodogram_astropy.png",bbox_inches="tight")
plt.close()

tp = 1/frequency
idx = np.argmax(power)-1
delta = []
for i in range(2):
  delta.append(tp[idx-i]-tp[idx-i+1])
dP = np.amax(delta)
dP = dP*(u.day)

# find period of strongest signal
period = 1/frequency[np.argmax(power)]
P = period*u.day
print("Possible Period of given time-series radial velocity data is : {} +/-
    {} Earth Days".format(np.round(P.value,4),np.round(dP.value,4)))

##################################
###---    Analysis with     ---###
###---     astropy.stats    ---###
###---     LombScargle      ---###
##################################

print("Initial Analysis with astropy.stats's LombScargle \n")
tfit = np.linspace(0,period,1000)
rvfit = LombScargle(time0,vr,dvr).model(tfit,1/period)
semi_amplitude = 0.5*(np.max(rvfit)-np.min(rvfit))
print("The fit semi-amplitude is %10.5f m/s" % semi_amplitude)
phase = (time0 % period)
voffset = np.mean(rvfit)
print("The velocity offset is %10.5f m/s" % voffset)

##################################
```

```python
###--- Plotting Time folded ---###
##--- Radial Velocity Curves ---##
##################################

ft = 17.5
fig = plt.figure(figsize=(8,6))
plt.errorbar((time-time[0])%period,vr,dvr,fmt='.b',capsize=4,ecolor='k',
     elinewidth=1.3)
plt.title(r'Time-Folded Radial Velocity Curve',fontsize=ft)
ax = plt.gca()
ax.grid(True)
ax.axhline(0, color='black', lw=2)
ax.axvline(0, color='black', lw=2)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.xlabel("Time (in days)",fontsize=ft)
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.savefig("time_folded_rv_curve_astropy.png",bbox_inches="tight")
plt.close()

##################################
###---   Synthetic Radial   ---###
##--- Velocity Curve Fitting ---##
##################################

# Using Scipy's Curve Fit

print("RVC fitting using Scipy's Curve Fit \n")

K = semi_amplitude
P = period
e = 0.
w = 0.
tau = time0[0]
vr0 = voffset
guess = (K,e,w,vr0)
modified_get_vr = lambda jd,K,e,w,vr0: get_vr(jd,K,P,e,w,tau,vr0)
rvfit = modified_get_vr(time,K,e,w,vr0)
chisq = np.sum(((vr-rvfit)/dvr)**2)
print("Chi-squared of initial guess is %10.5f" % chisq)

popt,pcov = curve_fit(modified_get_vr,time0,vr,sigma=dvr,absolute_sigma=True,
     p0=guess)

(K,e,w,vr0) = popt
rvfit_discrete = get_vr(time0,K,P,e,w,tau,vr0)
red_chisq,p = reduced_chi_square(rvfit_discrete,vr,dvr,len(popt))
print("Reduced Chi-squared of least-squares fit is %10.5f" % red_chisq)

ft=17.5
fig1 = plt.figure(1,figsize=(8,6))
#Plot Data-model
frame1 = fig1.add_axes((.1,.3,.8,.6))
plt.errorbar(phase,vr,dvr,fmt='.b',capsize=4,label="Data",ecolor='k',
     elinewidth=1.3)
plt.scatter(phase,rvfit_discrete,c='r',label="Best fit model from Lombscargle
      fitting")
frame1.set_xticklabels([]) #Remove x-tic labels for the first frame
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.legend(fontsize=ft-5)
plt.title("Folded RV curve with the best-fit model superimposed",fontsize=ft)
#Residual plot
difference = rvfit_discrete - vr
frame2=fig1.add_axes((.1,.1,.8,.2))
plt.errorbar(phase,difference,yerr=dvr,fmt='r.',capsize=4,ecolor='b',
```

```
        elinewidth=1.3)
plt.ylabel("Residue",fontsize=ft)
plt.xlabel("Time (in days)",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.savefig("astropy_curve_fit_scatter.png",bbox_inches="tight")
plt.close()

tfit = np.linspace(0,P,1000)
rvfit_continuous = get_vr(tfit,K,P,e,w,tau,vr0)

ft=17.5
fig1 = plt.figure(1,figsize=(8,6))
#Plot Data-model
frame1 = fig1.add_axes((.1,.3,.8,.6))
plt.errorbar(phase,vr,dvr,fmt='.b',capsize=4,label="Data",ecolor='k',
        elinewidth=1.3)
plt.plot(tfit,rvfit_continuous,'-r',label="Best fit model from Lombscargle
        fitting")
frame1.set_xticklabels([]) #Remove x-tic labels for the first frame
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.legend(fontsize=ft-5)
plt.title("Folded RV curve with the best-fit model superimposed",fontsize=ft)
#Residual plot
difference = rvfit_discrete - vr
frame2=fig1.add_axes((.1,.1,.8,.2))
plt.errorbar(phase,difference,yerr=dvr,fmt='b.',capsize=4,ecolor='k',
        elinewidth=1.3)
plt.ylabel("Residue",fontsize=ft)
plt.xlabel("Time (in days)",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.savefig("astropy_curve_fit.png",bbox_inches="tight")
plt.close()

if e<0:
  w -= np.pi
  e *= -1

if K<0:
  K *= -1
  w += np.pi

dK   = np.sqrt(pcov[0,0])
de   = np.sqrt(pcov[1,1])
dw   = np.sqrt(pcov[2,2])
dvr0   = np.sqrt(pcov[3,3])
P = period*u.day
P_error = dP.value
M =  1.11*u.M_sun
dM = 0.06*u.M_sun
G = const.G
a = (((P**2)*G*M)/(2*np.pi)**2)**(1/3)
a = a.to(u.au)
da = a*np.sqrt( ((2./3)*(dP/P))**2 + ((1./3)*(dM/M))**2)
da = da.to(u.au)
mpsini = np.cbrt(((M.to(u.kg)**2)*P.to(u.s))/(2*np.pi*G))*K*(u.m/u.s)
mpsini = mpsini.to(u.Mjup)
d_mpsini = (mpsini)*np.sqrt(((2./3)*(dM/M))**2 + ((1./3)*(dP/P))**2)
asini = (K*(u.m/u.s)*P*np.sqrt(1-e**2))/(2*np.pi)
dasini = asini*np.sqrt((np.abs(dK/K))**2 + np.abs(dP/P)**2)
w_deg = w*180/np.pi
dw_deg = dw*180/np.pi
```

```
final_param_values = {"vr0":"{} +/= {} m/s".format(np.round(vr0,3),np.round(
    dvr0,3)), "mpsini": "{} +/- {} M_Jup".format(np.round(mpsini.value,3),np
    .round(d_mpsini.value,3)), "Orbital separation": "{} +/- {} AU".format(
    np.round(a.value,4),np.round(da.value,4)), "Period": "{} +/- {} Earth
    Days".format(np.round(P.value,4),np.round(dP.value,4)), "asini (a =
    a_star)": "{} +/- {} m".format(np.round(asini.to(u.m).value,3),np.round(
    dasini.to(u.m).value,3)), "radial velocity semi-amplitude": "{} +/- {} m
    /s".format(np.round(K,2),np.round(dK,2)), "angle of periastron": "{} +/-
     {} ".format(np.round(w_deg,2),np.round(dw_deg,2)), "eccentricity": "{}
    +/- {}".format(np.round(e,3),np.round(de,3)) }

print(final_param_values)

# phase folded rv curve
E = get_E(phase/((2*np.pi)/P.to(u.day).value),e)
_,theta = get_vr(phase,K,P.value,e,w,tau,vr0,True)
ft = 17.5
fig = plt.figure(figsize=(8,6))
plt.errorbar(theta,vr,yerr=dvr,fmt='.b',capsize=4,ecolor='k',elinewidth=1.3)
plt.title(r'Phase-Folded Radial Velocity Curve',fontsize=ft)
ax = plt.gca()
ax.grid(True)
ax.axhline(0, color='black', lw=2)
ax.axvline(0, color='black', lw=2)
ax.xaxis.set_major_locator(plt.MultipleLocator(np.pi / 2))
ax.xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
ax.xaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.xlabel("Phase (in radians)",fontsize=ft)
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.savefig("phase_folded_rv_curve_astropy.png",bbox_inches="tight")
plt.close()

##################################
#######--- End of Code ---########
##################################
```

rvc_scipy.py

```
##################################
###--- Importing Libraries ---####
##################################

import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.signal import lombscargle
from astropy.stats import LombScargle
from scipy.optimize import curve_fit,fsolve
from astropy import constants as const
from astropy import units as u
from scipy import stats

##################################
####--- Function and Class ---####
####---     Defintion     ---#####
##################################

def multiple_formatter(denominator=2, number=np.pi, latex='\pi'):
    def gcd(a, b):
        while b:
            a, b = b, a%b
        return a
    def _multiple_formatter(x, pos):
```

```
        den = denominator
        num = np.int(np.rint(den*x/number))
        com = gcd(num,den)
        (num,den) = (int(num/com),int(den/com))
        if den==1:
            if num==0:
                return r'$0$'
            if num==1:
                return r'$%s$'%latex
            elif num==-1:
                return r'$-%s$'%latex
            else:
                return r'$%s%s$'%(num,latex)
        else:
            if num==1:
                return r'$\frac{%s}{%s}$'%(latex,den)
            elif num==-1:
                return r'$\frac{-%s}{%s}$'%(latex,den)
            else:
                return r'$\frac{%s%s}{%s}$'%(num,latex,den)
    return _multiple_formatter

class Multiple:
    def __init__(self, denominator=2, number=np.pi, latex='\pi'):
        self.denominator = denominator
        self.number = number
        self.latex = latex

def stopping_criteria(x1,x2,stop_type):
        if(stop_type == "abs"):
                return np.abs(x2 - x1)
        if(stop_type == "rel"):
                return np.abs(x2/x1 - 1.0)

def bisection(f,x1=0.0,x2=2*np.pi,max_iters=1000,xtol=1e-6,ftol=1e-4,*args):
    if(f(x1,*args)*f(x2,*args)>0):
        print("Both f(x1) and f(x2) have same sign! There possibly does not exist a unique root in the given interval. ")
        return
    elif(f(x1,*args) == 0):
        return x1, 0
    elif(f(x2,*args) == 0):
        return x2, 0
    tol = stopping_criteria(x1,x2,"abs")
    x0 = (x1+x2)/2.0;
    ctr = 0
    while(stopping_criteria(x1,x2,"abs")>xtol and f(x0,*args) > ftol and ctr <max_iters):
        x1,x2 = np.where(f(x1,*args)*f(x0,*args) < 0,(x1,x0),(x0,x2))
        ctr += 1
    return x0,ctr

def get_vr(t,K,P,e,w,tau,vr0,return_theta=False):
  E = get_E((t-tau)*2*np.pi/P,e)
  theta = 2*np.arctan2(np.sqrt(1.+e)*np.sin(0.5*E),
                       np.sqrt(1.-e)*np.cos(0.5*E))
  if(not return_theta):
    return K*(np.cos(theta+w)+e*np.cos(w))+vr0
  else:
    return K*(np.cos(theta+w)+e*np.cos(w))+vr0, theta


def get_E(M,e):
  E = []
  for i,mi in enumerate(M):
    x1 = mi - e
    x2 = mi + e
    E_tmp = bisection(lambda E: E-e*np.sin(E)-mi,x1,x2,1000,1e-6,1e-4)[0]
```

```python
    E.append(E_tmp)
  return np.array(E)

def reduced_chi_square(x,y,s,m): # ddof = v
  v = x.size - m
  chi2 = (np.sum((x-y)**2/s**2))/v
  p = 1 - stats.chi2.cdf(chi2, v)
  return chi2,p

##################################
###---     Data loading    ---####
###---  and Visualization  ---####
##################################

data = np.loadtxt("./RV_51Pegasi_Data.txt",dtype = [("time","f8"),("vr","f8")
    ,("dvr","f8")])
time = data["time"].astype(float) # time in days
vr = data["vr"].astype(float) # vr in m/s
dvr = data["dvr"].astype(float) # dvr in m/s
JD_ref = 2450000
time_jd = time+JD_ref
time0 = time-time[0]
ft = 17.5
fig = plt.figure(figsize=(10,6))
plt.errorbar(time+JD_ref,vr,yerr=dvr,fmt='o',ls='-',lw=0.3, capsize=4,ecolor=
    'k',elinewidth=2)
plt.xlabel("Time in Barycentric Julian Date",fontsize=ft)
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.title("Radial Velocity Data vs Time",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.savefig("rvdata_wline.png",bbox_inches="tight")
plt.close()

##################################
###--- Periodogram Analysis ---###
##################################

P = np.linspace(0.1,10,10000,True) # period in days
ang_freq = (2*np.pi)/P
dP = 0.5*(P[1]-P[0])*u.day
pgram = lombscargle(time_jd, vr, ang_freq, normalize=True)
ft = 17.5
fig = plt.figure(figsize=(8,6))
plt.plot(ang_freq, pgram)
plt.xlabel("Time (in days)",fontsize=ft)
plt.ylabel("Power Spectral Density",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.title("Periodogram of given time series data",fontsize=ft)
plt.savefig("periodogram_scipy.png",bbox_inches="tight")
plt.close()
freq = ang_freq[np.argmax(pgram)]
period = (2*np.pi)/freq
print("Possible Period of given time-series radial velocity data is : {} +/-
    {} Earth Days".format(np.round(period,4),np.round(dP,4)))

##################################
###---    Analysis with     ---###
###---     astropy.stats    ---###
###---     LombScargle      ---###
##################################

print("Initial Analysis with astropy.stats's LombScargle \n")
tfit = np.linspace(0,period,1000)
rvfit = LombScargle(time0,vr,dvr).model(tfit,1/period)
```

```
semi_amplitude = 0.5*(np.max(rvfit)-np.min(rvfit))
print("The fit semi-amplitude is %10.5f m/s" % semi_amplitude)
phase = (time0 % period)
voffset = np.mean(rvfit)
print("The velocity offset is %10.5f m/s" % voffset)

##################################
###--- Plotting Time folded ---###
##--- Radial Velocity Curves ---##
##################################

ft = 17.5
fig = plt.figure(figsize=(8,6))
plt.errorbar((time-time[0])%period,vr,dvr,fmt='.b',capsize=4,ecolor='k',
    elinewidth=1.3)
plt.title(r'Time-Folded Radial Velocity Curve',fontsize=ft)
ax = plt.gca()
ax.grid(True)
ax.axhline(0, color='black', lw=2)
ax.axvline(0, color='black', lw=2)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.xlabel("Time (in days)",fontsize=ft)
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.savefig("time_folded_rv_curve_scipy.png",bbox_inches="tight")
plt.close()

##################################
###---   Synthetic Radial   ---###
##--- Velocity Curve Fitting ---##
##################################

# Using Scipy's Curve Fit

print("RVC fitting using Scipy's Curve Fit \n")

K = semi_amplitude
P = period
e = 0.
w = 0.
tau = time0[0]
vr0 = voffset
guess = (K,e,w,vr0)
modified_get_vr = lambda jd,K,e,w,vr0: get_vr(jd,K,P,e,w,tau,vr0)
rvfit = modified_get_vr(time,K,e,w,vr0)
chisq = np.sum(((vr-rvfit)/dvr)**2)
print("Chi-squared of initial guess is %10.5f" % chisq)

popt,pcov = curve_fit(modified_get_vr,time0,vr,sigma=dvr,absolute_sigma=True,
    p0=guess)

(K,e,w,vr0) = popt
rvfit_discrete = get_vr(time0,K,P,e,w,tau,vr0)
red_chisq,p = reduced_chi_square(rvfit_discrete,vr,dvr,len(popt))
print("Reduced Chi-squared of least-squares fit is %10.5f" % red_chisq)

tfit = np.linspace(0,P,1000)
rvfit_continuous = get_vr(tfit,K,P,e,w,tau,vr0)

ft=17.5
fig1 = plt.figure(1,figsize=(8,6))
#Plot Data-model
frame1 = fig1.add_axes((.1,.3,.8,.6))
plt.errorbar(phase,vr,dvr,fmt='.b',capsize=4,label="Data",ecolor='k',
    elinewidth=1.3)
plt.plot(tfit,rvfit_continuous,'-r',label="Best fit model from Lombscargle
    fitting")
frame1.set_xticklabels([]) #Remove x-tic labels for the first frame
```

```
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.legend(fontsize=ft-5)
plt.title("Folded RV curve with the best-fit model superimposed",fontsize=ft)
#Residual plot
difference = rvfit_discrete - vr
frame2=fig1.add_axes((.1,.1,.8,.2))
plt.errorbar(phase,difference,yerr=dvr,fmt='b.',capsize=4,ecolor='k',elinewidth=1.3)
plt.ylabel("Residue",fontsize=ft)
plt.xlabel("Time (in days)",fontsize=ft)
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.grid()
plt.savefig("scipy_curve_fit.png",bbox_inches="tight")
plt.close()

if e<0:
  w -= np.pi
  e *= -1

if K<0:
  K *= -1
  w += np.pi

dK   = np.sqrt(pcov[0,0])
de   = np.sqrt(pcov[1,1])
dw   = np.sqrt(pcov[2,2])
dvr0   = np.sqrt(pcov[3,3])
P = period*u.day
P_error = dP.value
M =  1.11*u.M_sun
dM = 0.06*u.M_sun
G = const.G
a = (((P**2)*G*M)/(2*np.pi)**2)**(1/3)
a = a.to(u.au)
da = a*np.sqrt( ((2./3)*(dP/P))**2 + ((1./3)*(dM/M))**2)
da = da.to(u.au)
mpsini = np.cbrt(((M.to(u.kg)**2)*P.to(u.s))/(2*np.pi*G))*K*(u.m/u.s)
mpsini = mpsini.to(u.Mjup)
d_mpsini = (mpsini)*np.sqrt(((2./3)*(dM/M))**2 + ((1./3)*(dP/P))**2)
asini = (K*(u.m/u.s)*P*np.sqrt(1-e**2))/(2*np.pi)
dasini = asini*np.sqrt((np.abs(dK/K))**2 + np.abs(dP/P)**2)
w_deg = w*180/np.pi
dw_deg = dw*180/np.pi

final_param_values = {"vr0":"{} +/= {} m/s".format(np.round(vr0,3),np.round(dvr0,3)), "mpsini": "{} +/- {} M_Jup".format(np.round(mpsini.value,3),np.round(d_mpsini.value,3)), "Orbital separation": "{} +/- {} AU".format(np.round(a.value,4),np.round(da.value,4)), "Period": "{} +/- {} Earth Days".format(np.round(P.value,4),np.round(dP.value,4)), "asini (a = a_star)": "{} +/- {} m".format(np.round(asini.to(u.m).value,3),np.round(dasini.to(u.m).value,3)), "radial velocity semi-amplitude": "{} +/- {} m/s".format(np.round(K,2),np.round(dK,2)), "angle of periastron": "{} +/- {} ".format(np.round(w_deg,2),np.round(dw_deg,2)), "eccentricity": "{} +/- {}".format(np.round(e,3),np.round(de,3)) }

print(final_param_values)

# phase folded rv curve
E = get_E(phase/((2*np.pi)/P.to(u.day).value),e)
_,theta = get_vr(phase,K,P.value,e,w,tau,vr0,True)
ft = 17.5
fig = plt.figure(figsize=(8,6))
plt.errorbar(theta,vr,yerr=dvr,fmt='.b',capsize=4,ecolor='k',elinewidth=1.3)
plt.title(r'Phase-Folded Radial Velocity Curve',fontsize=ft)
```

```python
ax = plt.gca()
ax.grid(True)
ax.axhline(0, color='black', lw=2)
ax.axvline(0, color='black', lw=2)
ax.xaxis.set_major_locator(plt.MultipleLocator(np.pi / 2))
ax.xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
ax.xaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
plt.xticks(fontsize=ft)
plt.yticks(fontsize=ft)
plt.xlabel("Phase (in radians)",fontsize=ft)
plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
plt.savefig("phase_folded_rv_curve_scipy.png",bbox_inches="tight")
plt.close()

##################################
#######--- End of Code ---########
##################################
```

References

[1] Lomb, N.R. Least-squares frequency analysis of unequally spaced data. ApSS 39 pp. 447-462 (1976)

[2] Scargle, J. D. Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data. ApJ 1:263 pp. 835-853 (1982)

[3] Press W.H. and Rybicki, G.B, Fast algorithm for spectral analysis of unevenly sampled data. ApJ 1:338, p. 277 (1989)

[4] Martins, J. H. C et al., Evidence for a spectroscopic direct detection of reflected light from 51 Pegasi b, 2015, Astronomy & Astrophysics, Volume 576, id.A134, 9 pp., arXiv:1504.05962, 10.1051/0004-6361/201425298

[5] Ji Wang and Eric B. Ford, On the Eccentricity Distribution of Short-Period Single-Planet Systems, 2011, arXiv:1108.1811, 10.1111/j.1365-2966.2011.19600.x

[6] Geoffrey W. Marcy et al., The Planet around 51 Pegasi, 1997, The Astrophysical Journal, Volume 481, Number 2.

[7] Michel Mayor and Didier Queloz, A Jupiter-mass companion to a solar-type star,Nature, Volume 378, issue 6555, 1995, doi 10.1038_378355a0

[8] Butler, R. P. et al., Catalog of Nearby Exoplanets, July 2006, The Astrophysical Journal, Volume 646, Issue 1, pp. 505-522, arXiv:astro-ph/0607493, doi: 10.1086/504701.