

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 determined 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 is 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)

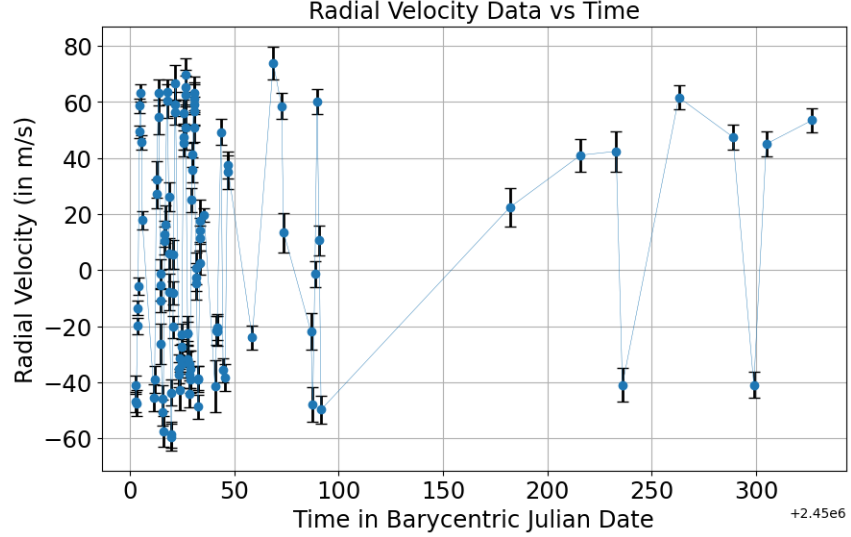


Figure 1: Plot of measured values of radial velocity as a function of time of observation (with line joining the data points)

and the observations are relatively sparse with increasing time (only ≈ 20 observations post 28/11/1995 against ≈ 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 $O[N \log N]$ implementation following the algorithm presented by Press & Rybicki [3].

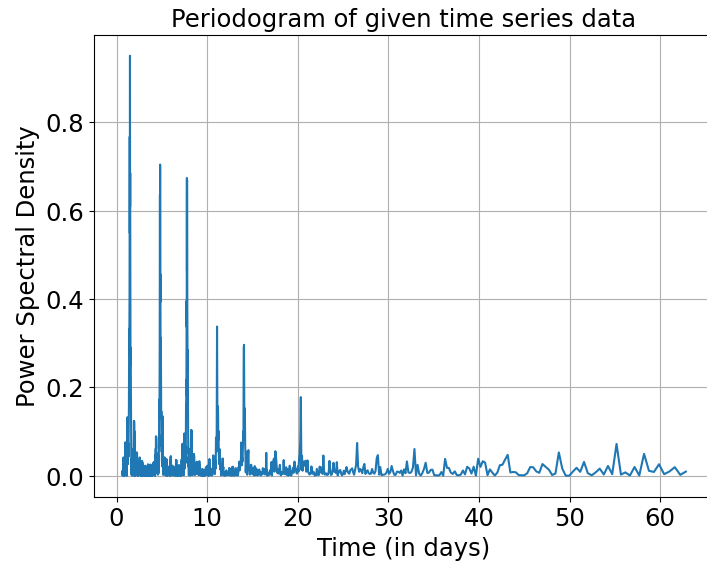


Figure 2: Periodogram of given time series data of radial velocity of 51 Pegasi (produced using `scipy.signal`'s `lombscargle` function)

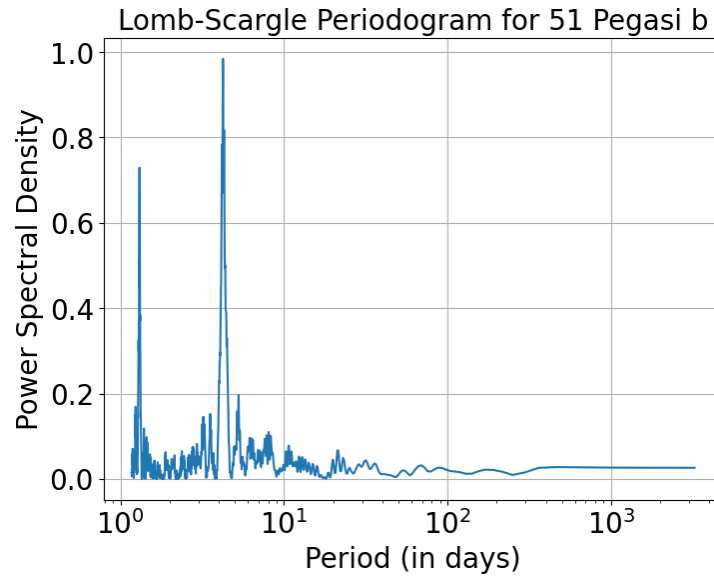


Figure 3: Periodogram of given time series data of radial velocity of 51 Pegasi (produced using `astropy.timeseries`'s `LombScargle` function)

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

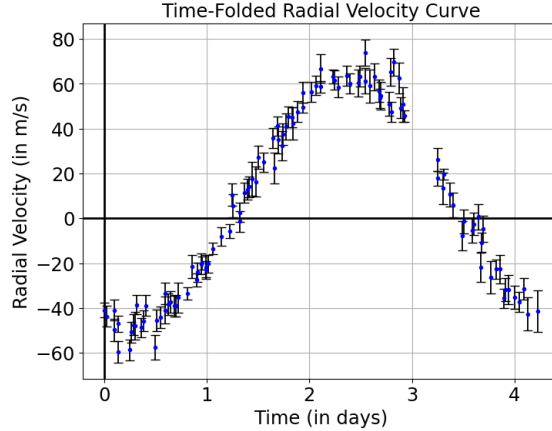


Figure 4: Time-Folded Radial Velocity Curve (with $t = 0$ sec corresponding to the starting value of the data) plotted with period determined using `scipy.signal`'s `lombscargle` function

Observations

From the Figure 4 and 5 we observe that the non-uniform 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.

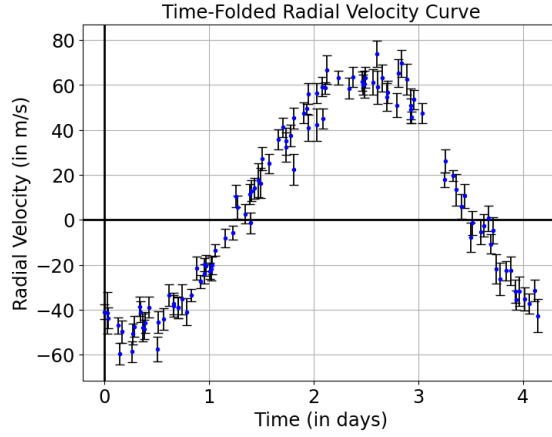


Figure 5: Time-Folded Radial Velocity Curve (with $t = 0$ sec corresponding to the starting value of the data) plotted with period determined using astropy.timeseries's LombScargle function

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)

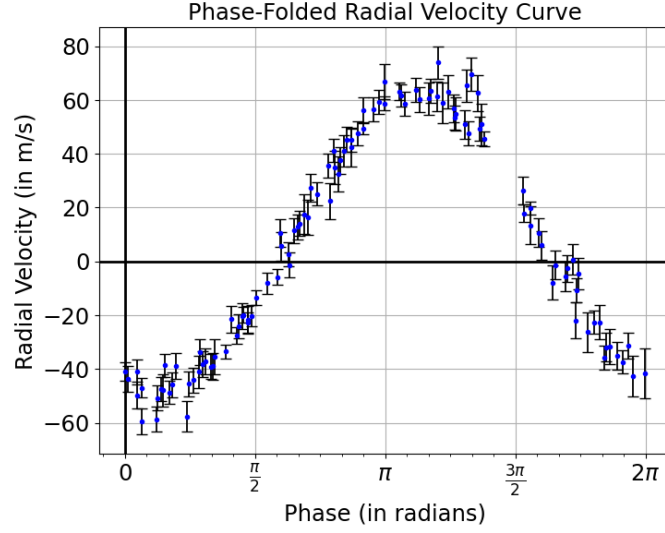


Figure 6: Phase-Folded Radial Velocity Curve (with $t = 0$ sec corresponding to the starting value of the data) plotted with period determined using `scipy.signal`'s `lombscargle` function

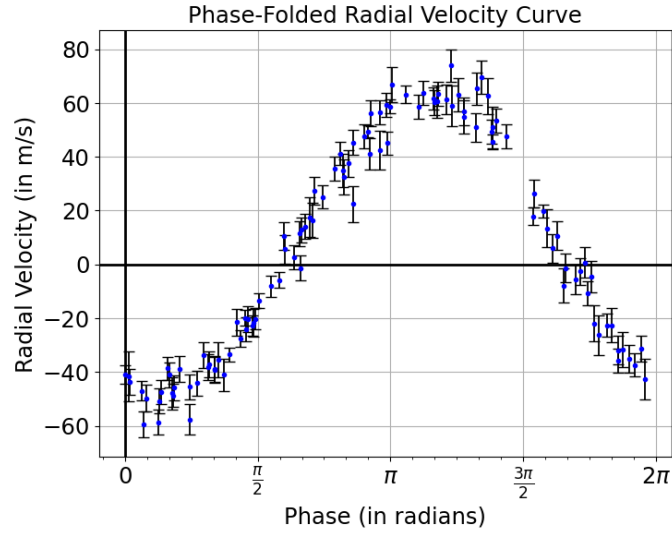


Figure 7: Phase-Folded Radial Velocity Curve (with $t = 0$ sec corresponding to the starting value of the data) plotted with period determined using `astropy.timeseries`'s `LombScargle` function

5 Comparison of model fit parameter values with published ones

In this section, we compare our 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.

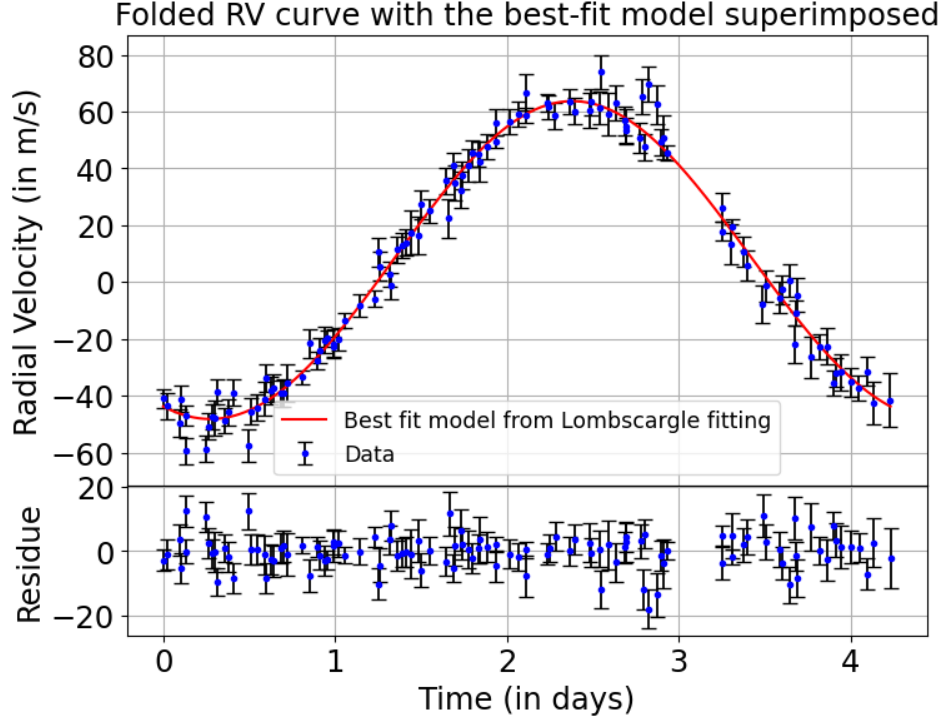


Figure 8: Folded RV curve with the best-fit model superimposed (top, line plot) and residual plot (bottom)

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

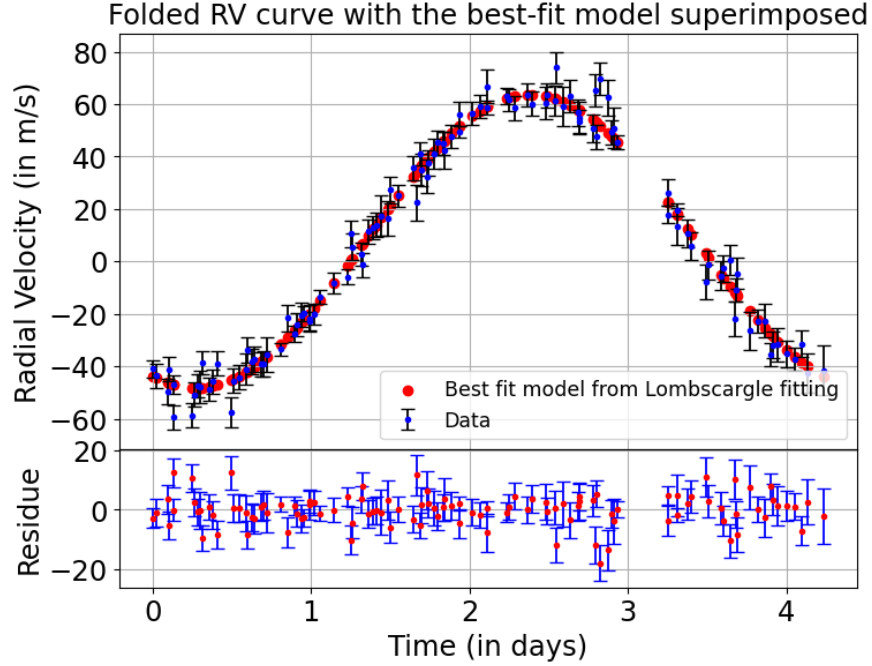


Figure 9: Folded RV curve with the best-fit model superimposed (top, scatter plot) and residual plot (bottom)

Fit parameter	Value estimated from best-fit
ω (in $^{\circ}$)	-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

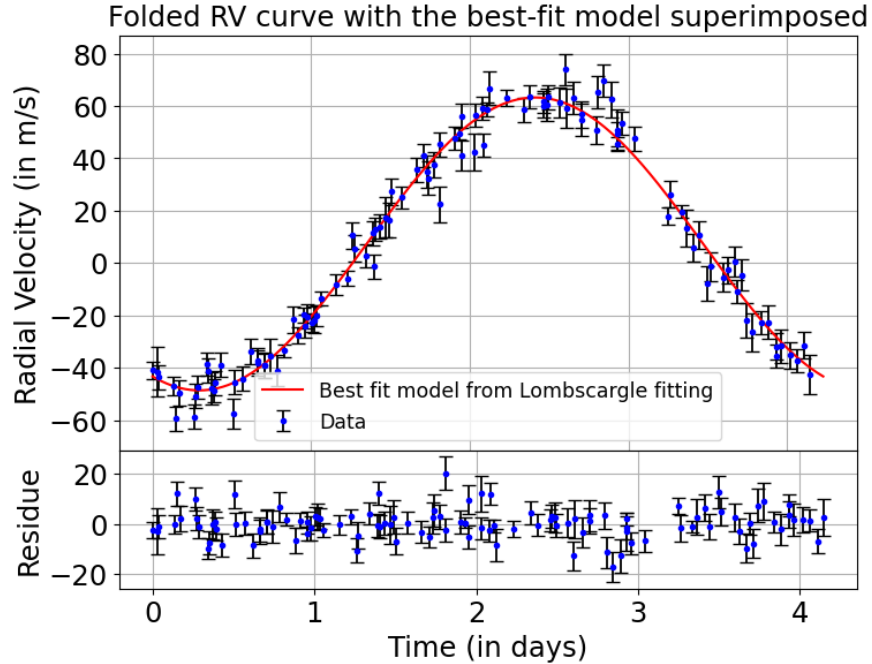


Figure 10: Folded RV curve with the best-fit model superimposed (top, line plot) and residual plot (bottom)

Fit parameter	Value estimated from best-fit
ω (in $^{\circ}$)	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

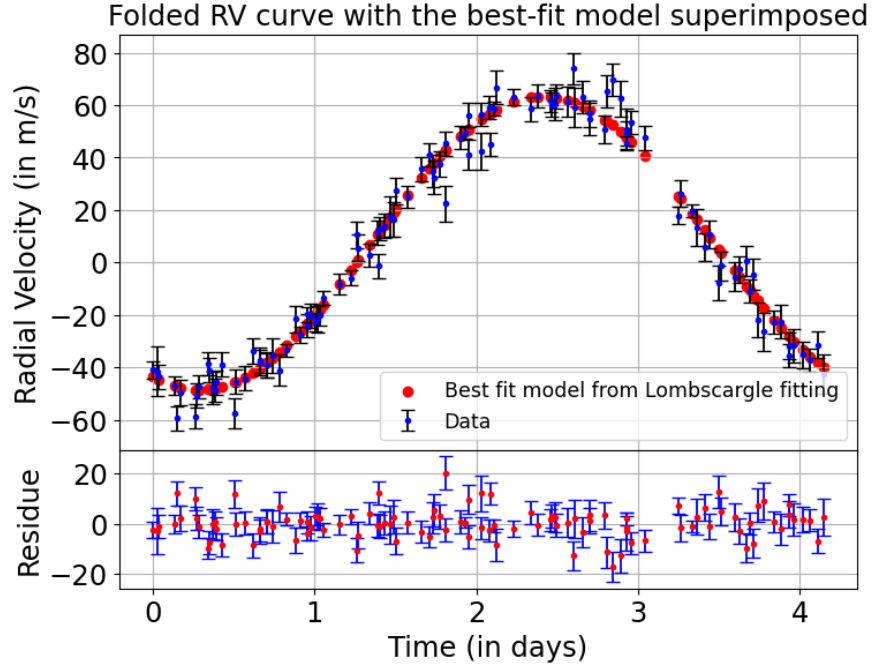


Figure 11: Folded RV curve with the best-fit model superimposed (top, scatter plot) and residual plot (bottom)

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

```

1 #####
2 ###-- Importing Libraries ---###
3 #####
4
5 import numpy as np
6 import matplotlib.pyplot as plt
7 import matplotlib as mpl
8 from scipy.signal import lombscargle
9 from astropy.stats import LombScargle
10 from astropy import timeseries
11 from scipy.optimize import curve_fit, fsolve
12 from astropy import constants as const
13 from astropy import units as u
14 from scipy import stats
15
16 #####
17 ###-- Function and Class ---###
18 ###-- Defintion ---#####
19 #####
20
21 def multiple_formatter(denominator=2, number=np.pi, latex='\pi'):
22     def gcd(a, b):
23         while b:
24             a, b = b, a%b
25         return a
26     def _multiple_formatter(x, pos):
27         den = denominator
28         num = np.int(np rint(den*x/number))
29         com = gcd(num,den)
30         (num,den) = (int(num/com),int(den/com))
31         if den==1:
32             if num==0:
33                 return r'$0$'
34             if num==1:
35                 return r'%s'%latex
36             elif num== -1:
37                 return r'$-%s'%latex
38             else:
39                 return r'%s%s'% (num, latex)
40     else:

```

```

41         if num==1:
42             return r'\frac{%s}{%s}$'%(latex,den)
43         elif num==-1:
44             return r'\frac{-%s}{%s}$'%(latex,den)
45         else:
46             return r'\frac{%s%s}{%s}$'%(num,latex,den)
47     return _multiple_formatter
48
49 class Multiple:
50     def __init__(self, denominator=2, number=np.pi, latex='\pi'):
51         self.denominator = denominator
52         self.number = number
53         self.latex = latex
54
55     def stopping_criteria(x1,x2,stop_type):
56         if(stop_type == "abs"):
57             return np.abs(x2 - x1)
58         if(stop_type == "rel"):
59             return np.abs(x2/x1 - 1.0)
60
61     def bisection(f,x1=0.0,x2=2*np.pi,max_iters=1000,xtol=1e-6,ftol=1e-4,*args):
62         if(f(x1,*args)*f(x2,*args)>0):
63             print("Both f(x1) and f(x2) have same sign! There possibly does not
64               exist a unique root in the given interval. ")
65             return
66         elif(f(x1,*args) == 0):
67             return x1, 0
68         elif(f(x2,*args) == 0):
69             return x2, 0
70         tol = stopping_criteria(x1,x2,"abs")
71         x0 = (x1+x2)/2.0;
72         ctr = 0
73         while(stopping_criteria(x1,x2,"abs")>xtol and f(x0,*args) > ftol and ctr
74             <max_iters):
75             x1,x2 = np.where(f(x1,*args)*f(x0,*args) < 0,(x1,x0),(x0,x2))
76             ctr += 1
77         return x0,ctr
78
79     def get_vr(t,K,P,e,w,tau,vr0,return_theta=False):
80         E = get_E((t-tau)*2*np.pi/P,e)
81         theta = 2*np.arctan2(np.sqrt(1+e)*np.sin(0.5*E),
82                               np.sqrt(1-e)*np.cos(0.5*E))
83         if(not return_theta):
84             return K*(np.cos(theta+w)+e*np.cos(w))+vr0
85         else:
86             return K*(np.cos(theta+w)+e*np.cos(w))+vr0, theta
87
88     def get_E(M,e):
89         E = []
90         for i,mi in enumerate(M):
91             x1 = mi - e
92             x2 = mi + e
93             E_tmp = bisection(lambda E: E-e*np.sin(E)-mi,x1,x2,1000,1e-6,1e-4)[0]
94             E.append(E_tmp)
95         return np.array(E)
96
97     def reduced_chi_square(x,y,s,m): # ddof = v
98         v = x.size - m
99         chi2 = (np.sum((x-y)**2/s**2))/v
100         p = 1 - stats.chi2.cdf(chi2, v)
101         return chi2,p
102
103 #####
104 ###--- Data loading ---###
105 ###--- and Visualization ---###
106 #####

```

```

107 data = np.loadtxt("./RV_51Pegasi_Data.txt",dtype = [("time","f8"),("vr","f8")
108         ,("dvr","f8")])
109 time = data["time"].astype(float) # time in days
110 vr = data["vr"].astype(float) # vr in m/s
111 dvr = data["dvr"].astype(float) # dvr in m/s
112 JD_ref = 2450000
113 time_jd = time+JD_ref
114 time0 = time-time[0]
115 ft = 17.5
116 fig = plt.figure(figsize=(10,6))
117 plt.errorbar(time+JD_ref,vr,yerr=dvr,fmt='o',ls='-',lw=0.3, capsize=4,ecolor=
118             'k',elinewidth=2)
119 plt.xlabel("Time in Barycentric Julian Date",fontsize=ft)
120 plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
121 plt.title("Radial Velocity Data vs Time",fontsize=ft)
122 plt.xticks(fontsize=ft)
123 plt.yticks(fontsize=ft)
124 plt.grid()
125 plt.savefig("rvdata_wline.png",bbox_inches="tight")
126 plt.close()
127
128 #####
129 ###-- Periodogram Analysis --###
130 #####
131 frequency,power = timeseries.LombScargle(time_jd,vr,dvr).autopower()
132 ft = 20
133 fig = plt.figure(figsize=(8,6))
134 plt.semilogx(1/frequency,power)
135 plt.title('Lomb-Scargle Periodogram for 51 Pegasi b',fontsize=ft)
136 plt.xlabel('Period (in days)',fontsize=ft)
137 plt.ylabel('Power Spectral Density',fontsize=ft)
138 plt.xticks(fontsize=ft)
139 plt.yticks(fontsize=ft)
140 plt.grid()
141 plt.savefig("periodogram_astropy.png",bbox_inches="tight")
142 plt.close()
143
144 tp = 1/frequency
145 idx = np.argmax(power)-1
146 delta = []
147 for i in range(2):
148     delta.append(tp[idx-i]-tp[idx-i+1])
149 dP = np.amax(delta)
150 dP = dP*(u.day)
151
152 # find period of strongest signal
153 period = 1/frequency[np.argmax(power)]
154 P = period*u.day
155 print("Possible Period of given time-series radial velocity data is : {} +/-
156       {} Earth Days".format(np.round(P.value,4),np.round(dP.value,4)))
157
158 #####
159 ###-- Analysis with --###
160 ###-- astropy.stats --###
161 ###-- LombScargle --###
162 #####
163
164 print("Initial Analysis with astropy.stats's LombScargle \n")
165 tfit = np.linspace(0,period,1000)
166 rvfit = LombScargle(time0,vr,dvr).model(tfit,1/period)
167 semi_amplitude = 0.5*(np.max(rvfit)-np.min(rvfit))
168 print("The fit semi-amplitude is %10.5f m/s" % semi_amplitude)
169 phase = (time0 % period)
170 voffset = np.mean(rvfit)
171 print("The velocity offset is %10.5f m/s" % voffset)
172
173 #####

```

```

172 ###-- Plotting Time folded ---##
173 ##-- Radial Velocity Curves ---##
174 #####
175
176 ft = 17.5
177 fig = plt.figure(figsize=(8,6))
178 plt.errorbar((time-time[0])%period, vr, dvr, fmt='b', capsize=4, ecolor='k',
179             elinewidth=1.3)
179 plt.title(r'Time-Folded Radial Velocity Curve', fontsize=ft)
180 ax = plt.gca()
181 ax.grid(True)
182 ax.axhline(0, color='black', lw=2)
183 ax.axvline(0, color='black', lw=2)
184 plt.xticks(fontsize=ft)
185 plt.yticks(fontsize=ft)
186 plt.xlabel("Time (in days)", fontsize=ft)
187 plt.ylabel("Radial Velocity (in m/s)", fontsize=ft)
188 plt.savefig("time_folded_rv_curve_astropy.png", bbox_inches="tight")
189 plt.close()
190
191 #####
192 ###-- Synthetic Radial ---##
193 ##-- Velocity Curve Fitting ---##
194 #####
195
196 # Using Scipy's Curve Fit
197
198 print("RVC fitting using Scipy's Curve Fit \n")
199
200 K = semi_amplitude
201 P = period
202 e = 0.
203 w = 0.
204 tau = time0[0]
205 vr0 = voffset
206 guess = (K,e,w,vr0)
207 modified_get_vr = lambda jd,K,e,w,vr0: get_vr(jd,K,P,e,w,tau,vr0)
208 rvfit = modified_get_vr(time,K,e,w,vr0)
209 chisq = np.sum(((vr-rvfit)/dvr)**2)
210 print("Chi-squared of initial guess is %10.5f" % chisq)
211
212 popt, pcov = curve_fit(modified_get_vr, time0, vr, sigma=dvr, absolute_sigma=True,
213                       p0=guess)
214
215 (K,e,w,vr0) = popt
216 rvfit_discrete = get_vr(time0,K,P,e,w,tau,vr0)
217 red_chisq, p = reduced_chi_square(rvfit_discrete, vr, dvr, len(popt))
218 print("Reduced Chi-squared of least-squares fit is %10.5f" % red_chisq)
219
220 ft=17.5
221 fig1 = plt.figure(1, figsize=(8,6))
222 #Plot Data-model
223 frame1 = fig1.add_axes((.1,.3,.8,.6))
224 plt.errorbar(phase, vr, dvr, fmt='b', capsize=4, label="Data", ecolor='k',
225             elinewidth=1.3)
226 plt.scatter(phase, rvfit_discrete, c='r', label="Best fit model from Lombscargle
227             fitting")
228 frame1.set_xticklabels([]) #Remove x-tic labels for the first frame
229 plt.ylabel("Radial Velocity (in m/s)", fontsize=ft)
230 plt.xticks(fontsize=ft)
231 plt.yticks(fontsize=ft)
232 plt.grid()
233 plt.legend(fontsize=ft-5)
234 plt.title("Folded RV curve with the best-fit model superimposed", fontsize=ft)
235 #Residual plot
236 difference = rvfit_discrete - vr
237 frame2=fig1.add_axes((.1,.1,.8,.2))
238 plt.errorbar(phase, difference, yerr=dvr, fmt='r', capsize=4, ecolor='b',

```

```

        elinewidth=1.3)
236 plt.ylabel("Residue",fontsize=ft)
237 plt.xlabel("Time (in days)",fontsize=ft)
238 plt.xticks(fontsize=ft)
239 plt.yticks(fontsize=ft)
240 plt.grid()
241 plt.savefig("astropy_curve_fit_scatter.png",bbox_inches="tight")
242 plt.close()
243
244 tfit = np.linspace(0,P,1000)
245 rvfit_continuous = get_vr(tfit,K,P,e,w,tau,vr0)
246
247 ft=17.5
248 fig1 = plt.figure(1,figsize=(8,6))
249 #Plot Data-model
250 frame1 = fig1.add_axes((.1,.3,.8,.6))
251 plt.errorbar(phase,vr,dvr,fmt='b',capsize=4,label="Data",ecolor='k',
        elinewidth=1.3)
252 plt.plot(tfit,rvfit_continuous,'-r',label="Best fit model from Lombscargle
        fitting")
253 frame1.set_xticklabels([]) #Remove x-tic labels for the first frame
254 plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
255 plt.xticks(fontsize=ft)
256 plt.yticks(fontsize=ft)
257 plt.grid()
258 plt.legend(fontsize=ft-5)
259 plt.title("Folded RV curve with the best-fit model superimposed",fontsize=ft)
260 #Residual plot
261 difference = rvfit_discrete - vr
262 frame2=fig1.add_axes((.1,.1,.8,.2))
263 plt.errorbar(phase,difference,yerr=dvr,fmt='b.',capsize=4,ecolor='k',
        elinewidth=1.3)
264 plt.ylabel("Residue",fontsize=ft)
265 plt.xlabel("Time (in days)",fontsize=ft)
266 plt.xticks(fontsize=ft)
267 plt.yticks(fontsize=ft)
268 plt.grid()
269 plt.savefig("astropy_curve_fit.png",bbox_inches="tight")
270 plt.close()
271
272 if e<0:
273     w -= np.pi
274     e *= -1
275
276 if K<0:
277     K *= -1
278     w += np.pi
279
280 dK = np.sqrt(pcov[0,0])
281 de = np.sqrt(pcov[1,1])
282 dw = np.sqrt(pcov[2,2])
283 dvr0 = np.sqrt(pcov[3,3])
284 P = period*u.day
285 P_error = dP.value
286 M = 1.11*u.M_sun
287 dM = 0.06*u.M_sun
288 G = const.G
289 a = ((P**2)*G*M)/(2*np.pi**2)**(1/3)
290 a = a.to(u.au)
291 da = a*np.sqrt(((2./3)*(dP/P))**2 + ((1./3)*(dM/M))**2)
292 da = da.to(u.au)
293 mpsini = np.cbrt(((M.to(u.kg)**2)*P.to(u.s))/(2*np.pi*G))*K*(u.m/u.s)
294 mpsini = mpsini.to(u.Mjup)
295 d_mpsini = (mpsini)*np.sqrt(((2./3)*(dM/M))**2 + ((1./3)*(dP/P))**2)
296 asini = (K*(u.m/u.s)*P*np.sqrt(1-e**2))/(2*np.pi)
297 dasini = asini*np.sqrt((np.abs(dK/K))**2 + np.abs(dP/P)**2)
298 w_deg = w*180/np.pi
299 dw_deg = dw*180/np.pi

```



```

300
301 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)) }
302
303 print(final_param_values)
304
305 # phase folded rv curve
306 E = get_E(phase/((2*np.pi)/P.to(u.day).value),e)
307 _,theta = get_vr(phase,K,P.value,e,w,tau,vr0,True)
308 ft = 17.5
309 fig = plt.figure(figsize=(8,6))
310 plt.errorbar(theta,vr,yerr=dvr,fmt='b',capsize=4,ecolor='k',elinewidth=1.3)
311 plt.title(r'Phase-Folded Radial Velocity Curve',fontsize=ft)
312 ax = plt.gca()
313 ax.grid(True)
314 ax.axhline(0, color='black', lw=2)
315 ax.axvline(0, color='black', lw=2)
316 ax.xaxis.set_major_locator(plt.MultipleLocator(np.pi / 2))
317 ax.xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
318 ax.xaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
319 plt.xticks(fontsize=ft)
320 plt.yticks(fontsize=ft)
321 plt.xlabel("Phase (in radians)",fontsize=ft)
322 plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
323 plt.savefig("phase_folded_rv_curve_astropy.png",bbox_inches="tight")
324 plt.close()
325
326 #####
327 #####-- End of Code --#####
328 #####

```

rvc_scipy.py

```

1 #####
2 ###-- Importing Libraries --###
3 #####
4
5 import numpy as np
6 import matplotlib.pyplot as plt
7 import matplotlib as mpl
8 from scipy.signal import lombscargle
9 from astropy.stats import LombScargle
10 from scipy.optimize import curve_fit,fsolve
11 from astropy import constants as const
12 from astropy import units as u
13 from scipy import stats
14
15 #####
16 ###-- Function and Class --###
17 ###-- Defintion --#####
18 #####
19
20 def multiple_formatter(denominator=2, number=np.pi, latex='\pi'):
21     def gcd(a, b):
22         while b:
23             a, b = b, a%b
24         return a
25     def _multiple_formatter(x, pos):

```

```

26     den = denominator
27     num = np.int(np rint(den*x/number))
28     com = gcd(num,den)
29     (num,den) = (int(num/com),int(den/com))
30     if den==1:
31         if num==0:
32             return r'$0$'
33         if num==1:
34             return r'%s'%latex
35         elif num== -1:
36             return r'$-s$'%latex
37         else:
38             return r'%s%s'%(num,latex)
39     else:
40         if num==1:
41             return r'$\frac{{s}}{{s}}$'%(latex,den)
42         elif num== -1:
43             return r'$\frac{-s}{{s}}$'%(latex,den)
44         else:
45             return r'$\frac{{s}s}{{s}}$'%(num,latex,den)
46     return _multiple_formatter
47
48 class Multiple:
49     def __init__(self, denominator=2, number=np.pi, latex='\pi'):
50         self.denominator = denominator
51         self.number = number
52         self.latex = latex
53
54     def stopping_criteria(x1,x2,stop_type):
55         if(stop_type == "abs"):
56             return np.abs(x2 - x1)
57         if(stop_type == "rel"):
58             return np.abs(x2/x1 - 1.0)
59
60     def bisection(f,x1=0.0,x2=2*np.pi,max_iters=1000,xtol=1e-6,ftol=1e-4,*args):
61         if(f(x1,*args)*f(x2,*args)>0):
62             print("Both f(x1) and f(x2) have same sign! There possibly does not
63               exist a unique root in the given interval. ")
64             return
65         elif(f(x1,*args) == 0):
66             return x1, 0
67         elif(f(x2,*args) == 0):
68             return x2, 0
69         tol = stopping_criteria(x1,x2,"abs")
70         x0 = (x1+x2)/2.0;
71         ctr = 0
72         while(stopping_criteria(x1,x2,"abs")>xtol and f(x0,*args) > ftol and ctr
73             <max_iters):
74             x1,x2 = np.where(f(x1,*args)*f(x0,*args) < 0,(x1,x0),(x0,x2))
75             ctr += 1
76         return x0,ctr
77
78     def get_vr(t,K,P,e,w,tau,vr0,return_theta=False):
79         E = get_E((t-tau)*2*np.pi/P,e)
80         theta = 2*np.arctan2(np.sqrt(1.+e)*np.sin(0.5*E),
81                             np.sqrt(1.-e)*np.cos(0.5*E))
82         if(not return_theta):
83             return K*(np.cos(theta+w)+e*np.cos(w))+vr0
84         else:
85             return K*(np.cos(theta+w)+e*np.cos(w))+vr0, theta
86
87     def get_E(M,e):
88         E = []
89         for i,mi in enumerate(M):
90             x1 = mi - e
91             x2 = mi + e
92             E_tmp = bisection(lambda E: E-e*np.sin(E)-mi,x1,x2,1000,1e-6,1e-4)[0]

```

```

92     E.append(E_tmp)
93     return np.array(E)
94
95 def reduced_chi_square(x,y,s,m): # ddof = v
96     v = x.size - m
97     chi2 = (np.sum((x-y)**2/s**2))/v
98     p = 1 - stats.chi2.cdf(chi2, v)
99     return chi2,p
100
101 #####
102 ###---      Data loading      ---###
103 ###--- and Visualization ---###
104 #####
105
106 data = np.loadtxt("./RV_51Pegasi_Data.txt",dtype = [("time","f8"),("vr","f8")
107               ,("dvr","f8")])
108 time = data["time"].astype(float) # time in days
109 vr = data["vr"].astype(float) # vr in m/s
110 dvr = data["dvr"].astype(float) # dvr in m/s
111 JD_ref = 2450000
112 time_jd = time+JD_ref
113 time0 = time-time[0]
114 ft = 17.5
115 fig = plt.figure(figsize=(10,6))
116 plt.errorbar(time+JD_ref,vr,yerr=dvr,fmt='o',ls='-',lw=0.3, capsize=4,ecolor=
117             'k',elinewidth=2)
118 plt.xlabel("Time in Barycentric Julian Date",fontsize=ft)
119 plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
120 plt.title("Radial Velocity Data vs Time",fontsize=ft)
121 plt.xticks(fontsize=ft)
122 plt.yticks(fontsize=ft)
123 plt.grid()
124 plt.savefig("rvdata_wline.png",bbox_inches="tight")
125 plt.close()
126
127 #####
128 ###--- Periodogram Analysis ---###
129 #####
130
131 P = np.linspace(0.1,10,10000,True) # period in days
132 ang_freq = (2*np.pi)/P
133 dP = 0.5*(P[1]-P[0])*u.day
134 pgram = lombscargle(time_jd, vr, ang_freq, normalize=True)
135 ft = 17.5
136 fig = plt.figure(figsize=(8,6))
137 plt.plot(ang_freq, pgram)
138 plt.xlabel("Time (in days)",fontsize=ft)
139 plt.ylabel("Power Spectral Density",fontsize=ft)
140 plt.xticks(fontsize=ft)
141 plt.yticks(fontsize=ft)
142 plt.grid()
143 plt.title("Periodogram of given time series data",fontsize=ft)
144 plt.savefig("periodogram_scipy.png",bbox_inches="tight")
145 plt.close()
146 freq = ang_freq[np.argmax(pgram)]
147 period = (2*np.pi)/freq
148 print("Possible Period of given time-series radial velocity data is : {} +/-
149       {} Earth Days".format(np.round(period,4),np.round(dP,4)))
150
151 #####
152 ###---      Analysis with      ---###
153 ###---      astropy.stats      ---###
154 ###---      LombScargle        ---###
155 #####
156
157 print("Initial Analysis with astropy.stats's LombScargle \n")
158 tfit = np.linspace(0,period,1000)
159 rvfit = LombScargle(time0,vr,dvr).model(tfit,1/period)

```

```

157 semi_amplitude = 0.5*(np.max(rvfit)-np.min(rvfit))
158 print("The fit semi-amplitude is %10.5f m/s" % semi_amplitude)
159 phase = (time0 % period)
160 voffset = np.mean(rvfit)
161 print("The velocity offset is %10.5f m/s" % voffset)
162
163 #####
164 ###--- Plotting Time folded ---###
165 ###--- Radial Velocity Curves ---###
166 #####
167
168 ft = 17.5
169 fig = plt.figure(figsize=(8,6))
170 plt.errorbar((time-time[0])%period,vr,dvr,fmt='.b',capsize=4,ecolor='k',
171             elinewidth=1.3)
172 plt.title(r'Time-Folded Radial Velocity Curve',fontsize=ft)
173 ax = plt.gca()
174 ax.grid(True)
175 ax.axhline(0, color='black', lw=2)
176 ax.axvline(0, color='black', lw=2)
177 plt.xticks(fontsize=ft)
178 plt.yticks(fontsize=ft)
179 plt.xlabel("Time (in days)",fontsize=ft)
180 plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
181 plt.savefig("time_folded_rv_curve_scipy.png",bbox_inches="tight")
182 plt.close()
183 #####
184 ###--- Synthetic Radial ---###
185 ###--- Velocity Curve Fitting ---###
186 #####
187
188 # Using Scipy's Curve Fit
189
190 print("RVC fitting using Scipy's Curve Fit \n")
191
192 K = semi_amplitude
193 P = period
194 e = 0.
195 w = 0.
196 tau = time0[0]
197 vr0 = voffset
198 guess = (K,e,w,vr0)
199 modified_get_vr = lambda jd,K,e,w,vr0: get_vr(jd,K,P,e,w,tau,vr0)
200 rvfit = modified_get_vr(time,K,e,w,vr0)
201 chisq = np.sum(((vr-rvfit)/dvr)**2)
202 print("Chi-squared of initial guess is %10.5f" % chisq)
203
204 popt,pcov = curve_fit(modified_get_vr,time0,vr,sigma=dvr,absolute_sigma=True,
205                       p0=guess)
206
207 (K,e,w,vr0) = popt
208 rvfit_discrete = get_vr(time0,K,P,e,w,tau,vr0)
209 red_chisq,p = reduced_chi_square(rvfit_discrete,vr,dvr,len(popt))
210 print("Reduced Chi-squared of least-squares fit is %10.5f" % red_chisq)
211
212 tfit = np.linspace(0,P,1000)
213 rvfit_continuous = get_vr(tfit,K,P,e,w,tau,vr0)
214
215 ft=17.5
216 fig1 = plt.figure(1,figsize=(8,6))
217 #Plot Data-model
218 frame1 = fig1.add_axes((.1,.3,.8,.6))
219 plt.errorbar(phase,vr,dvr,fmt='.b',capsize=4,label="Data",ecolor='k',
220             elinewidth=1.3)
221 plt.plot(tfit,rvfit_continuous,'-r',label="Best fit model from Lombscargle
222         fitting")
223 frame1.set_xticklabels([]) #Remove x-tic labels for the first frame

```

```

221 plt.ylabel("Radial Velocity (in m/s)",fontsize=ft)
222 plt.xticks(fontsize=ft)
223 plt.yticks(fontsize=ft)
224 plt.grid()
225 plt.legend(fontsize=ft-5)
226 plt.title("Folded RV curve with the best-fit model superimposed",fontsize=ft)
227 #Residual plot
228 difference = rvfit_discrete - vr
229 frame2=fig1.add_axes((.1,.1,.8,.2))
230 plt.errorbar(phase,difference,yerr=dvr,fmt='b.',capsize=4,ecolor='k',
231             elinewidth=1.3)
232 plt.ylabel("Residue",fontsize=ft)
233 plt.xlabel("Time (in days)",fontsize=ft)
234 plt.xticks(fontsize=ft)
235 plt.yticks(fontsize=ft)
236 plt.grid()
237 plt.savefig("scipy_curve_fit.png",bbox_inches="tight")
238 plt.close()
239
240 if e<0:
241     w -= np.pi
242     e *= -1
243
244 if K<0:
245     K *= -1
246     w += np.pi
247
248 dK = np.sqrt(pcov[0,0])
249 de = np.sqrt(pcov[1,1])
250 dw = np.sqrt(pcov[2,2])
251 dvr0 = np.sqrt(pcov[3,3])
252 P = period*u.day
253 P_error = dP.value
254 M = 1.11*u.M_sun
255 dM = 0.06*u.M_sun
256 G = const.G
257 a = (((P**2)*G*M)/(2*np.pi)**2)**(1/3)
258 a = a.to(u.au)
259 da = a*np.sqrt(((2./3)*(dP/P))**2 + ((1./3)*(dM/M))**2)
260 da = da.to(u.au)
261 mpsini = np.cbrt(((M.to(u.kg)**2)*P.to(u.s))/(2*np.pi**G))*K*(u.m/u.s)
262 mpsini = mpsini.to(u.Mjup)
263 d_mpsini = (mpsini)*np.sqrt(((2./3)*(dM/M))**2 + ((1./3)*(dP/P))**2)
264 asini = (K*(u.m/u.s)*P*np.sqrt(1-e**2))/(2*np.pi)
265 dasini = asini*np.sqrt((np.abs(dK/K))**2 + np.abs(dP/P)**2)
266 w_deg = w*180/np.pi
267 dw_deg = dw*180/np.pi
268
269 final_param_values = {"vr0": "{0} +/- {0} m/s".format(np.round(vr0,3),np.round(
270     dvr0,3)), "mpsini": "{0} +/- {0} M_Jup".format(np.round(mpsini.value,3),np
271     .round(d_mpsini.value,3)), "Orbital separation": "{0} +/- {0} AU".format(
272     np.round(a.value,4),np.round(da.value,4)), "Period": "{0} +/- {0} Earth
273     Days".format(np.round(P.value,4),np.round(dP.value,4)), "asini (a =
274     a_star)": "{0} +/- {0} m".format(np.round(asini.to(u.m).value,3),np.round(
275     dasini.to(u.m).value,3)), "radial velocity semi-amplitude": "{0} +/- {0} m
276     /s".format(np.round(K,2),np.round(dK,2)), "angle of periastron": "{0} +/-
277     {0} ".format(np.round(w_deg,2),np.round(dw_deg,2)), "eccentricity": "{0}
278     +/- {0} ".format(np.round(e,3),np.round(de,3)) }
279
280 print(final_param_values)
281
282 # phase folded rv curve
283 E = get_E(phase/((2*np.pi)/P.to(u.day).value),e)
284 _,theta = get_vr(phase,K,P.value,e,w,tau,vr0,True)
285 ft = 17.5
286 fig = plt.figure(figsize=(8,6))
287 plt.errorbar(theta,vr,yerr=dvr,fmt='b',capsize=4,ecolor='k',elinewidth=1.3)
288 plt.title(r'Phase-Folded Radial Velocity Curve',fontsize=ft)

```

```

279 ax = plt.gca()
280 ax.grid(True)
281 ax.axhline(0, color='black', lw=2)
282 ax.axvline(0, color='black', lw=2)
283 ax.xaxis.set_major_locator(plt.MultipleLocator(np.pi / 2))
284 ax.xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
285 ax.xaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
286 plt.xticks(fontsize=ft)
287 plt.yticks(fontsize=ft)
288 plt.xlabel("Phase (in radians)", fontsize=ft)
289 plt.ylabel("Radial Velocity (in m/s)", fontsize=ft)
290 plt.savefig("phase_folded_rv_curve_scipy.png", bbox_inches="tight")
291 plt.close()
292
293 #####
294 #####-- End of Code --#####
295 #####

```

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.