Search for Exoplanets Project Report

Nina Selezneva

M1 SPACE Student
University of Aix-Marseille, Marseille, France, 13014

E-mail: nina.selezneva@etu.univ-amu.fr

This document describes Radial Velocity based exoplanets searching techniques learned during the Project work with a special focus on Bayesian statistics. Two techniques: Leastsquare and Particle Swarm Optimization (PSO) are compared. A series of exoplanets searching attempts using PSO on a specially prepared data set are described with efficiency of PSO method calculated for each data set. Results and suggestions for further improvement of aforementioned techniques are inroduced in the summary. Python code used for PSO is present in the Appendix.

Introduction

The discovery of exoplanets is arguably one of the most exciting scientific discoveries in Astronomy since it's beginning with the concept of multiple worlds like our own appearing as far back as 2000 years ago, with Epicurus asserting, ”There are infinite worlds both like and unlike this world of ours... We must believe that in all worlds there are living creatures and plants and other things we see in this world”.[Sara Seager] The search for a planet similar to our own is natural and is much reflected in our cultural heritage : books, movies, and various forms of art, with news of another Earth-like candidate exoplanet found surfacing in mass media almost daily. The focus of this work is on one of the most widely used techniques for exoplanets search – Radiovelocity technique – and Particle Swarm Optimization method for fitting corresponding data.

Radiovelocity Method

One of the often used techniques for exoplanets search is called Radiovelocity method. This method is based on the reflex motion of a star orbited by a planet which can be devided into two components: a radial motion and an orthoradial motion. The first component can be learned by measuring the radial velocity variation of a star using the Doppler–Fizeau effect. [A. Santerne, R-V, 2016]

Using 3 Laws of Kepler we get the following equation for radial velocity of a planet:

$$V_r(t) = \gamma + K[\cos(\nu(t) + \omega) + e\cos(\omega)]$$

where γ is the systemic radial velocity, ω is the argument of periastron, $\nu(t)$ is the true anomaly, K is the radial velocity semi-amplitude and equals to:

$$K = \frac{m\sin(i)}{(M_* + m)^{2/3}} \cdot \frac{(2\pi G)^{1/3}}{P^{1/3}\sqrt{1-e^2}}$$

with the universal gravitational constant G, star mass M_*, planet mass m, inclination i, eccentricity e, and orbital period P. The true anomaly $\nu(t)$ can be deduced from the evaluation of the eccentric anomaly $E(t)$ which is the angle between the periastron and the projected position of the star on the circumscribed circle. The relation between the two is:

$$\nu(t) = 2 \cdot \arctan\left(\sqrt{\frac{1+e}{1-e}} \cdot tan\left(\frac{E(t)}{2}\right)\right)$$

The eccentricity anomaly is linked with the mean anomaly M by the Kepler equation:

$$E(t) = M + e\sin(E(t))$$

$$M(t) = 2\pi \cdot \sqrt{\frac{t - t_p}{P}}$$

with t_p the epoch of periastron.

$E(t)$ can be determined and computed by finding zeros of the function $f(E(t)) = M(t) - E(t) + e\sin(E(t)) = 0$ by the first order Newton method:

$$E_{n+1} = E_n \frac{f(E_n)}{f'(E_n)}$$

$$E_{n+1} = E_n \frac{M(t) + e\sin(E_n) - E_n}{1 - \cos(E_n)}$$

which converges quickly if first step E_0 is set to M.

Data Fitting

Typical search of an exoplanet starts with processing data recorded at a scientific facility by spectrograph. Since it is impossible to deduce parameters analytically as these systems are overdetermined, the only solution is to fit the data according to a chosen model. The easiest model is a one planet model, so it was used in fitting the data sets.

There were 10 data sets overall present for this project, with 8 of them having more than one planet injected in the data, and 2 having only one planet. In our case data files consisted of 3 columns: time of recording, radial velocity value, and error.

Least Squares Fitting

The first fitting method tried on the sets was Least Squares method. The best fit in the least-squares sense minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided by a model. [Internet source] In a situation of no information about the system available, when no reliable initial guess could be made about the period of the planet, the resulting doesn't yield correct results. In all 8 initial files this method returned wrong values for periods, as is shown on the table below.

File name	Period by Least-squares [d]	Real period [d]
BT 1	10	580
BT 2	9.99	0.45
BT 3	10	14.86
BT 4	5	118.38
BT 5	5	14.79
BT 6	10	10.85
BT 7	5	5.90
BT 8	10	22.34

Least squares, however, shows satisfactory results when used with a period more or less resembling the actual period of a planet present in the data set. In real cases the initial period is guessed by creating a periodogram (using tools to directly search periodicity of the signal) for the time-series data and selecting the period of a highest peak, repeated until model matches the data close enough to suggest the model is correct. This method can be employed only in the approximation that the planet has a nearly circular orbit as it relies heavily on the periodicity of the signal. If the orbit is not circular this leads to ambiguity in the definition of initial period as the periodogram will show several peaks at the same time, which in turn leads to wrong fitting results.

Another limitation of the Least Squares method is that the usage of periodogram doesn't allow fitting with models that allow for more than one planet existing in the system. A current workaround of this – by removing the signal of a supposedly already found planet in the dataset – can lead to false positives of finding additional, non-existent in reality, planets in the systems in question.

Particle Swarm Optimization (PSO)

Theory of PSO

The less straightforward way of fitting the data can be found among so-called evolutionary algorithms. This set of algorithms is modeled after the real life systems present in nature, one of them being Particle Swarm Optimization, a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.[Internet source] PSO is initialized with a group of random particles (swarm size) randomly spread over the parameters' volume, defined by lower and upper bounds of possible values, each particle assigned with a random solution. At each step particles get updated by two best values, one being the local best fit (pbest), the other the general best fit (gbest) tracked by the optimizer of the method which checkes values among all particles. After this particles recalculate their positions and velocities, continuing to explore the parameters space for optimal fit until either the number of iterations is reached or minimum change of values, as shown in the pseudo code below[Internet source].

```
For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness
        value (pBest) in history
            set current value as the new pBest
    End

    Choose the particle with the best fitness value of all
    the particles as the gBest
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End
While maximum iterations or minimum error criteria is not attained
```

Since particles are spread across all unknown parameters this method has no limitations inherent to Least Squares as it does not depend on eccentricity or correct initial guess of the period.

Likelihood function

PSO is an algorithm that finds values for the minimum of a function. To find the the parameters required for the Radiovelocity method (6 for one planet model, 11 for 2 planets model) PSO requires a function for minimization.

This function can be found in the Bayesian Statistics and is called Likelihood function. Likelihood gives an estimation of how likely some outcome is for a defined model if its parameters have a certain distribution. Maximum likelihood L corresponds to the most probable parameters of a given model.

If we assume that the parameters in our system have normal distribution, the likelihood function L becomes:

$$L = \prod \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp(-\frac{1}{2}(\frac{x_i - x_m^i}{\sigma_i})^2)$$

where x_i is the measured value, σ_i its error, and x_m^i theoretical/model value.

To make PSO run faster the equation above can be rewritten as:

$$lnL = \sum \left[ln\frac{1}{2\pi\sigma_i} + \left(-\frac{1}{2} \left(\frac{x_i - x_m^i}{\sigma i} \right)^2 \right) \right]$$

Since PSO minimizes a function, the lnL should be passed into PSO as $-lnL$.

Implementation

PSO was implemented by using a Python library called pyswarm. Lower and upper bounds were chosen based on physically sensible results, with the radial semi-amplitude K from 0 to 1 km/s and period from 0.1 m/s as the lowest possible descernible value of equipment (noise level) to the timespan of the dataset.

Since the particles are spread over the parameters space randomly (normal distribution) it is possible to miss the correct convergence values as the particles change their position in steps exceeding the correct value, or get stuck to lower or upper bound.

During first runs of the PSO code this mistake was found in calculation of semi-amplitude K (when K was set to physically meaningless high value of 100 km/s) and in calculation of period, since timespan in the dataset can be relatively big, exceeding other values by several orders of magnitude. Changing upper bound of K to more physically sound value allowed to overcome the first problem. To alleviate the second issue period scale was changed to log, thus reducing the length of period in the parameters space and allowing for more precise convergeance.

Results

The table below shows the periods and semi-amplitudes of the planets injected in the data, results returned by PSO best fit, and its efficiency. Since the model for fitting presumed only one planet existing in the system, the table below shows parameters of the planet with the highest value of semi-amplitude K, as it was the only possible solution for a maximum likelihood. For calculation of efficiency PSO for each file was run 40 times.

File name	P [d], K [m/s]	PSO best fit P [d], K [m/s]	Efficiency [%]
BT 1	580.00, 20.6	578.2, 20.88	30
BT 2	0.45, 0.6	0.453, 0.61	10
BT 3	14.86, 4.3	14.858, 4.7	10
BT 4	118.38, 41.3	118.14, 41.2	50-60
BT 5	14.79, 1.0	14.81, 1.09	10
BT 6	10.85, 4.5	10.86, 4.5	10
BT 7	5.9, 0.8	not found	0
BT 8	22.34, 45.1	22.35, 45.	7.5-10

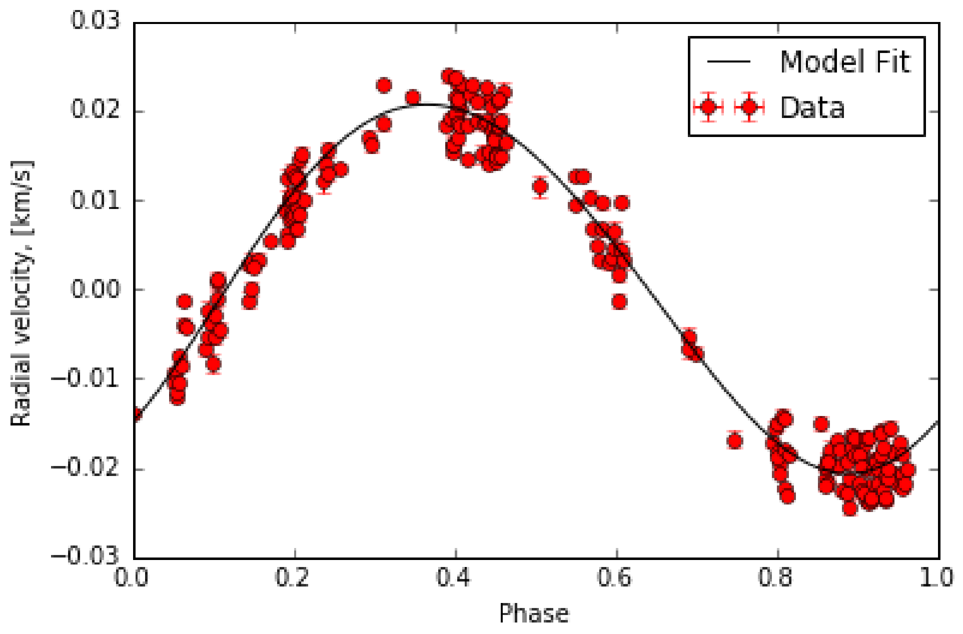
The working hypothesis for such low efficiency of the PSO was in using the wrong model of only one planet present in the dataset. For validation of this hypothesis PSO was also run for 40 times on two new versions of the file BT8, but now holding only one planet each (see table below).

File name	P [d], K [m/s]	PSO best fit P [d], K [m/s]	Efficiency [%]
BT 8 XS	10.95, 2.5	10.96, 2.26	10-20
BT 8 XL	22.34, 45.1	22.328, 44.37	13.3-40

Despite the fact that the efficiency in a model-matching case significantly increased, it still stays low rising a question of the efficiency of the implementation of the method in the search for exoplanets in real life cases.

Figures 1-6 show several best PSO fitted plots of the data files.

Figure 1: BT 1



Summary

As was shown in the document, Bayesian statistics and Particle Swarm Optimization are a promising fitting technique in the search for exoplanets as it doesn't depend on the periodicity of the signal and allows to find correct period and semi-amplitude in a dataset even with a wrong model. However, this technique suffers from relatively low efficiency even in the case of correct model, and finding the right solution is a time consuming process, with the unpredictable amount of time required for one run of fitting.

Possible ways of overcoming these limitations are:

Figure 2: BT 2

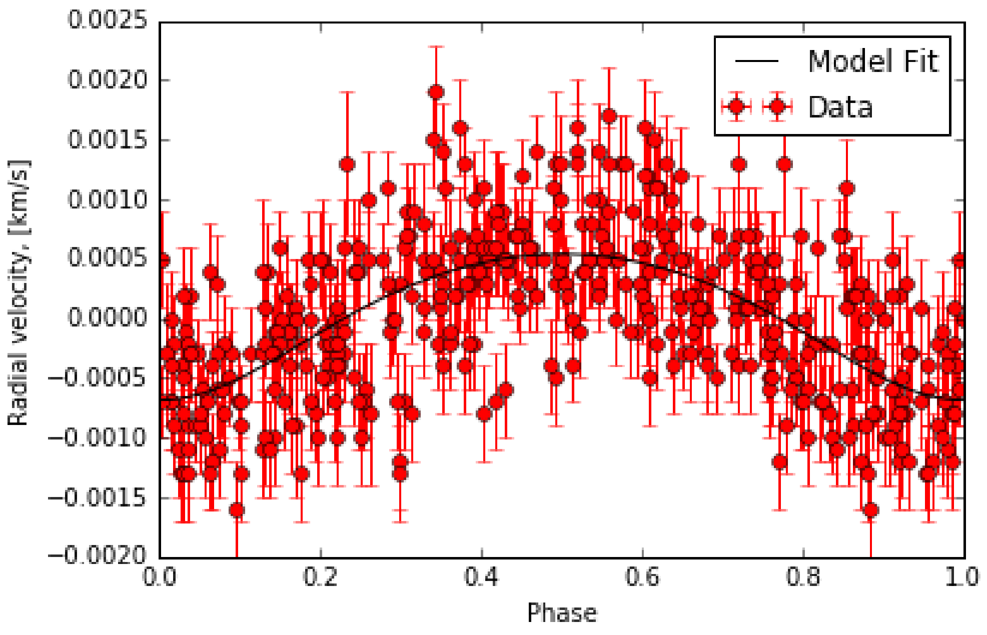


Figure 3: BT 3

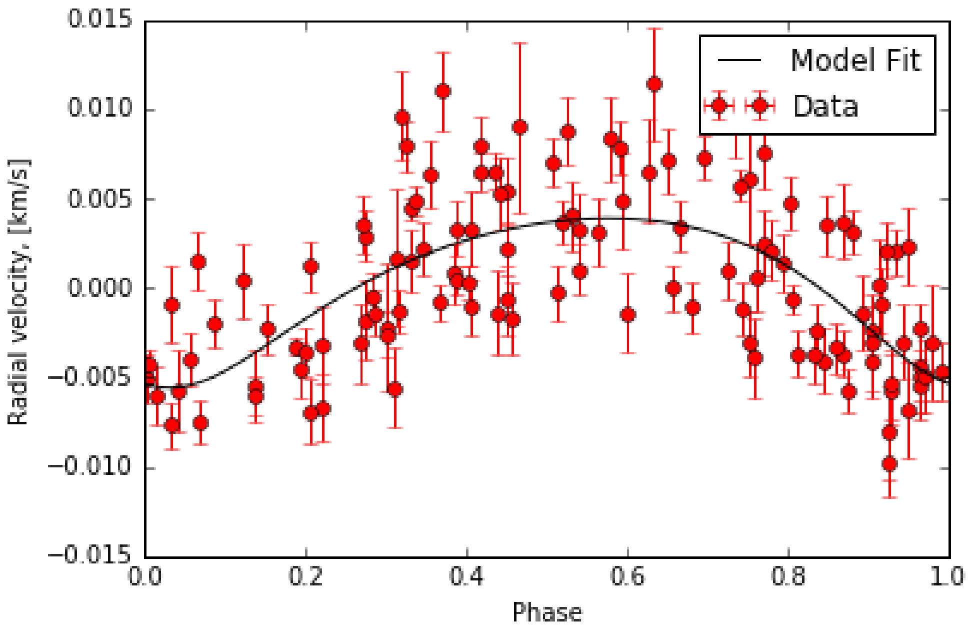


Figure 4: BT 4

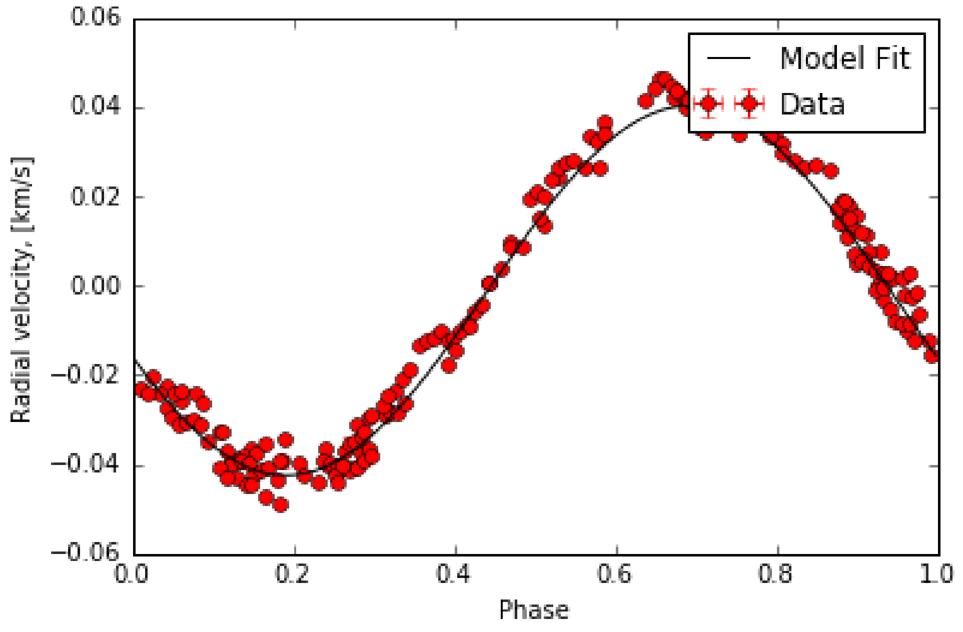


Figure 5: BT 5

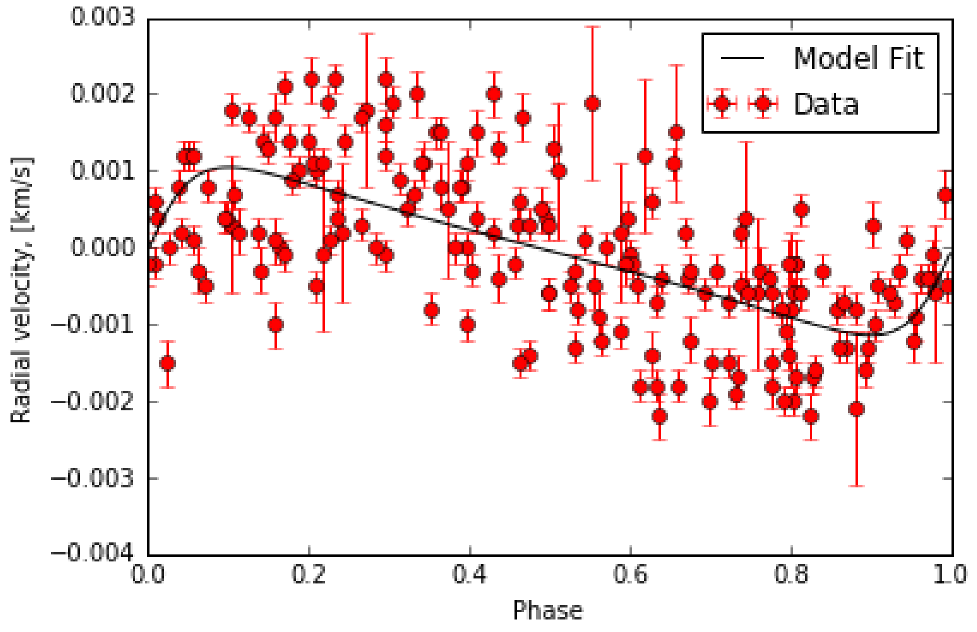
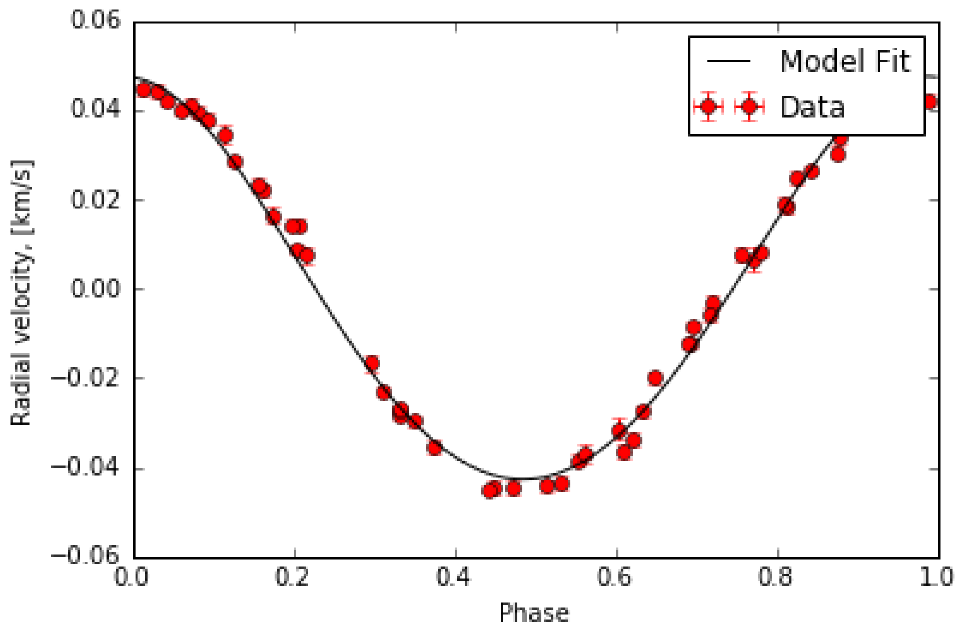


Figure 6: BT 8



- using a more complex model, adjusted to the probable physical parameters and their distributions, of the system in question, with lower and upper bounds adjusted to it accordingly;
- automating pickup of parameters for PSO such as number of iterations and the size of the swarm, initial distribution of parameters;
- automating search of the best fit by calculating χ^2 of the fit or a similar parameter and running PSO fitting until this parameter is not low enough to ensure the best fit;
- running PSO in parallel to reduce the amount of time required for finding the optimal fit.

References and Notes

[A. Santerne, R-V, 2016] Santerne, A. (2016) The Radial velocity technique. Laboratoire d'Astrophysique de Marseille, November, 16, 2016

[Sara Seager] Sara Seager, Massacusetts Institute of Technology, Jack J. Lissauer, NASA Ames Research Center, Introduction to Exoplanets

[Internet source] Wikipedia

[Internet source] PSO Tutorial

Appendix: Implementation Code

Below is the implementation of the algorithm for searching an exoplanet using Radiovelocity method and PSO fitting.

```
#populating space with libraries: numpy (calculations),
#matplotlib (drawing plots),
#pyswarm -- Particle Swarm Optimization
%pylab inline
from pyswarm import pso

data = loadtxt('planetXS.dat') # loading files

time = np.zeros(len(data), dtype=float64)
value = np.zeros(len(data), dtype=float64)
errorBr = np.zeros(len(data), dtype=float64)

for i in range(0, len(data)): #populating numpy arrays
    time[i] = float(data[i][0])
    value[i] = float(data[i][1])
    errorBr[i] = float(data[i][2])

plot(time, value) #plotting data
timespan = time[-1]-time[0] #calculating timespan to be used in
    PSO

def eccNewton(t, tp, P, e): # Newton method
    sEccNewton = np.zeros(len(t))
    Mtime = (2.*pi*(t - tp))/P
    eccNewton = Mtime
    while(norm(eccNewton-sEccNewton)>0.001):
        sEccNewton = eccNewton
        eccNewton = eccNewton + (Mtime + e*sin(eccNewton) - eccNewton)/(1-e*cos(eccNewton))
    nuT = 2.*arctan(sqrt((1.+e)/(1.-e))*tan(eccNewton/2.))

    return nuT

def f(t, gamma, K, e, omega, P, tp): # calculating radial velocity function
    nuT = eccNewton(t, tp, P, e)
    VrT = gamma + K*(cos(nuT+omega) + e*cos(omega))

    return VrT

def forPSO(P0, *args): # calculating -log of likelihood function to pass into PSO function
    t, V, errBr = args
    gamma, K, e, omega, P, tp = P0
    P = exp(P) # since timespan can be big, we use log scale for period in input parameters, thus to get real period we need
    # to take exponent of the input parameter P
    nuT = eccNewton(t, tp, P, e)
    VrT = gamma + K*(cos(nuT+omega) + e*cos(omega))
    residue = (V-VrT)/errBr
    expV = -0.5*(residue**2)
    logL = -1*(sum(log(1./(sqrt(2.*pi)*errBr))) + sum(expV))
    return logL

lb = [-1., 0., 0., 0., log(0.1), 0.] #lower bound for PSO
ub = [1., .1, 0.9, 2*pi, log(timespan), timespan] # upper bound for PSO
args = (time, value, errorBr)

xopt, fopt = pso(forPSO, lb, ub, args=args, swarmsize=150, maxiter=10000, minstep = 1e-15, minfunc=1e-15) #minimizing return
#value of forPSO function, which is -log of our likelihood function
xopt[4] = exp(xopt[4]) # taking exponent of the P value to get true period
```

```

# Drawing plot
phase = (time − xopt[5])%xopt[4]/xopt[4]
xx = linspace(xopt[5], xopt[5] + xopt[4], 1000)
yy = f(xx, xopt[0], xopt[1], xopt[2], xopt[3], xopt[4], xopt[5])
phase2 = (xx − xopt[5])%xopt[4]/xopt[4]
yy = yy[argsort(phase2)]
phase2 = sort(phase2)
errorbar(phase, value, xerr=0., yerr=errorBr, fmt = ’or’, label=”Data”) #plotting data points
ylabel(’Radial velocity, [km/s]’)
xlabel(’Phase’)
plot(phase2, yy, ’−k’, label=”Model Fit”) #plotting fit curve
legend()
print(”gamma: ” + str(xopt[0]) + ” K: ” + str(xopt[1]) + ” e: ” + str(xopt[2]) + ” omega:” + str(xopt[3]) +
      ” Period : ” + str(xopt[4]) + ” tp : ” + str(xopt[5])) # printing corresponding values of the minimized function
```