# Exoplanet Transit Parameter Retrieval using MCMC


## Abstract

This study employs the Markov Chain Monte Carlo (MCMC) technique to ascertain precise transit parameters in exoplanetary systems. Transits, where a planet crosses its host star's disc, yield invaluable information about planetary characteristics. MCMC, a powerful statistical method, efficiently explores parameter spaces to derive the most probable values and uncertainties. By fitting theoretical transit models to observed light curves, this approach extracts critical parameters such as planetary radius, orbital period, and inclination angle. The primary discovery revealed that nearly all transit parameters converged to a singular point, with the exception of $\frac{a}{R_\star}$. Notably, whenever $\frac{a}{R_\star}$ reached convergence, the orbital period $P$ did not. This observation suggests that there is a correlation between the orbital period and the semi-major axis, as expected. The final results are $T_0 = 2,453,989.75 \pm 0$, $\frac{R_\mathrm{p}}{R_\star} = 0.13 \pm 0$, $\frac{a}{R_\star} = 7.60 \pm 0.10$, $i = 83.53 \pm 0.25$ and $P = 2.51 \pm 0.14$.


## 1 INTRODUCTION & THEORETICAL BACKGROUND

Since the advancement of technology and physics, the quest to search for planets outside our solar system has been a goal. These planets are called 'exoplanets'. Today's technology assists us in making certain calculations and deductions through various data retrieval methods. Exoplanets are generally found using indirect methods, that is, they are not seen physically with a telescope but rather inferred from the data of the star they happen to orbit. There are different techniques used to detect exoplanets, such as the transit method (the one used in this experiment), radial velocity method, microlensing, and astrometry methods.

The transit method is based on the fact that when an exoplanet orbits its host star, there is a time period known as the transit period during which the exoplanet is considered to be in transit. This indicates that the planet is between us "the observers" and its host star (European Space Agency, n.d.), it is also known as the inferior conjunction. During this period, the exoplanet blocks some of the light from the star, and by retrieving the relative light flux in contrast to the original flux (no transit occurring), one can make predictions and calculations about some of the exoplanet's parameters, such as planetary radius, semi-major axis distance, periodic time, and so on.

The radial velocity method relies on the fact that, in reality, the star and the exoplanet orbit a common centre of mass, and thus the star orbits around a small circle or elliptical orbit. The size of the orbit of the star depends on the gravitational force between itself and the exoplanet. During its orbit, the star is continuously emitting light, which observers observe. However, when the star is moving towards the observer (in its orbit), the light frequency received will be blue-shifted, while when the star is moving away relative to the observer, the light frequency will be red-shifted. This is known as the Doppler effect (Lovis & Fischer, 2011). From this data, one can infer the radial velocity of the exoplanet and other parameters.

Another method is known as microlensing, essentially, this involves a pair of stars being aligned such that the closer star acts as a lens and bends the light coming from the farthest star to us. The light intensity can be plotted

against time, and one would obtain essentially a normal distribution curve. However, if a planet is present, it would further bend the light waves, resulting in an increase in the light intensity over a short period of time (European Space Agency, n.d.). From such data, one can deduce that there is an exoplanet and further obtain the exoplanet parameters.

Lastly, there is the astrometry method, which is similar to the radial velocity method, but this time the system observed is not moving towards or away from the observer but rather changing its position in a vertical plane. By measuring the position of the star over a period of time, one can determine if there is a secondary mass object affecting the star's orbit. This can be used to find exoplanets. However, one must have accurate positions of the star being observed, which can prove very challenging (European Space Agency, 2019).

Markov Chain Monte Carlo (MCMC) stands as a pivotal algorithm facilitating model fitting to empirical data. Operating within a parameter space vector $\theta$, MCMC manoeuvres 'walkers' through this space, iteratively generating new $\theta$ values. The algorithm assesses likeness by computing a chi-square test ($\chi^2$) to quantify the proximity between observed $y_{data}$ and modelled $y_{model}$ data relative to the initial error $y_{err}$ (Imad Pasha, 2020). This likelihood assessment, described by Equation 1, determines the squared differences between observed and modelled data, informing the walker's steps.

$$\text{likeliness} = \frac{1}{2}\sum_{i=1}^{n}\left(\frac{y_{data} - y_{model}}{y_{err}}\right)^2 \tag{1}$$

The resulting posterior distribution has several sampling values, capturing plausible parameter variations and determining the most likely parameter configurations. This iterative exploration refines parameter estimation, enabling a comprehensive understanding of the parameter space that aligns with the observed data and defined priors (prior function). By iteratively narrowing in on the most likely model parameters, MCMC provides a strong strategy for determining complex systems such as the transit method system. Moreover MCMC is a based on Bayesian statistics which deals with the probability based on the initial conditions.

In order to translate the information coming from far away, generally from stars, scientists use a graph called a light curve. Light curves are curves generally showing flux (brightness per area of the observed instrument) against time (National Aeronautics and Space Administration, 2013). Light curves can be used to describe how the relative flux changes during a transit period. One can generate light curves in Python using the batman package (BAsic Transit Model cAlculatioN ) to simulate transit models (Laura Kreidberg, 2015). Moreover, batman and MCMC can be used in combination to determine the best parameter fit or the most likely values.

The goal of this experiment is to get light curves for Kepler-90 exoplanets with various limb-darkening coefficients and types using Batman, as well as to model a periodic function and determine the likely parameters for $a, b, c, d$ using MCMC. Lastly, calculate the transit parameters for Kepler-1b using batman and MCMC. Kepler-1b transit parameters are expected to be:

$$\frac{a_{\mathrm{q}}}{R_\star} = 7.63 \pm 0.12 \text{ (Holman et al., 2007)} \tag{2}$$

$$\frac{R_{\mathrm{pq}}}{R_\star} = 0.1253 \pm 0.010 \text{ (Holman et al., 2007)} \tag{3}$$

$$i_{\mathrm{q}} = 83.57 \pm 0.14 \text{ (Holman et al., 2007)} \tag{4}$$

$$T_{\mathrm{q}} = 2,453,989.75286 \pm 0.00029 \text{ HJD (Holman et al., 2007)} \tag{5}$$

$$P_{\mathrm{q}} = 2.47063 \pm 0.0010 \text{ days (O'Donovan et al., 2006)} \tag{6}$$

where $\frac{a_q}{R_*}$ is the ratio of the semi-major axis with respect to the stellar radius of Kepler-1. $\frac{R_{pq}}{R_*}$ is the ratio of the radius planet with respect to stellar radius Kepler-1. $i_q$ is the inclination angle in degrees, $T_q$ is the time of the inferior conjunction in HJD (Heliocentric Julian date) and $P_q$ is the orbital period in days.

# 2 METHODOLOGY

## 2.1 Task 1A

Initially, data for Kepler-90 planets was imported into Python from Excel, with each row indexed by the planet name. The Kepler-90 star's radius was collected from scientific sources and used to calculate $\frac{a}{R_\star}$ and $\frac{R_{\mathrm{p}}}{R_\star}$ for each Kepler-90 planet. The values obtained were then entered into the batman package, along with the limb-darkening coefficients and type. Afterwards, for all Kepler-90 planets, a uniform light curve was generated, followed by linear and quadratic light curves Kepler-90h exoplanet.

## 2.2 Task 1B

Data was provided during this task, denoted as $x$ and $y$, which, when plotted, gave a period function. It was defined a trigonometric model function as:

$$y_{model} = a + b\sin(cx + d). \tag{7}$$

The initial estimates of $a$ and $b$ were derived from the analysis of the original data plot, whereas $c$ and $d$ were predefined. The likelihood and probability functions were implemented using these fixed values as priors. As a result, the `emcee` package made it easier to run the Markov Chain Monte Carlo (MCMC) method with pre-defined iterations and walkers. This enabled careful examination of the parameter space. The algorithm converged to posterior distributions for $a$, $b$, $c$, and $d$ by sampling the likelihood function, showing their most likely values.

## 2.3 Task 2

In the second task, similarly to task 1, data was imported and defined. To construct MCMC, first it was defined the vector space $\theta$ (the model function) with parameters $T_0$, $\frac{a}{R_\star}$, $\frac{R_\mathrm{p}}{R_\star}$, $i$, and $P$, as well as the limb darkening coefficients with a quadratic type. Additionally, the domain was defined, and using batman, the ideal light curve was generated relative to the data given, thus deducing the parameters for $\theta$. In addition, a likeliness function was defined as in equation 1, where it determined if the values obtained from the walkers had a lesser error than the given error. If so, the respective new determined values of the parameters were stored; if not, they'd be discarded, and they'd find new likely parameters. Finally, a prior and probability function were defined; the prior function takes the range of parameter values that the model function can take and passes them to the probability function, which returns what the parameters are likely to be; additionally, initial values, the step size, walkers, iterations, and burnout were defined, and MCMC was computed multiple times (similalr to task1B) with different numbers of iterations. The posterior distributions for each parameter were flattened in 2D using the `corner` function.

# 3 RESULTS

## 3.1 Task 1B

Initial estimates were: $a = 32, b = 3, c = 24, d = 7$. Optimal values are:

$$a = 32.00 \text{ with 95\% CI [31.98, 32.03]}$$

$$b = 2.88 \text{ with 95\% CI [2.84, 2.92]}$$

$$c = 24.72 \text{ with 95\% CI [24.71, 24.73]}$$

$$d = 7.24 \text{ with 95\% CI [7.21, 7.27].}$$

Table showing convergence:

| Iterations | $a$ | $\sigma_a$ | $b$ | $\sigma_b$ | $c$ | $\sigma_c$ | $d$ | $\sigma_d$ |
|---|---|---|---|---|---|---|---|---|
| 1,000 | 31.9994 | 0.0148 | 2.8687 | 0.0492 | 24.7289 | 0.0414 | 7.2501 | 0.0463 |
| 5,000 | 31.9993 | 0.0155 | 2.8801 | 0.0221 | 24.7203 | 0.0055 | 7.2427 | 0.0159 |
| 10,000 | 31.9993 | 0.0157 | 2.8800 | 0.0221 | 24.7204 | 0.0055 | 7.2424 | 0.0158 |
| 20,000 | 31.9992 | 0.0156 | 2.8798 | 0.0221 | 24.7203 | 0.0055 | 7.2427 | 0.0159 |
| 50,000 | 31.9991 | 0.1555 | 2.8800 | 0.0220 | 24.7203 | 0.0054 | 7.2427 | 0.0158 |

**Table 1.** Table of MCMC outputs for task 1B at particular sets of iterations such that the convergence within the parameter space can be explored.

## 3.2 Task 2

Initial estimates were: $T_0 = 453989.75, \frac{R_p}{R_\star} = 0.122, \frac{a}{R_\star} = 7.672, i = 83.57, P = 2.471$. Optimal values are:

$$T_0 = 2453989.75 \text{ with 95\% CI [2453989.75, 2453989.75]}$$

$$\frac{R_\mathrm{p}}{R_\star} = 0.13 \text{ with 95\% CI [0.13, 0.13]}$$

$$\frac{a}{R_\star} = 7.60 \text{ with 95\% CI [7.48, 7.72]}$$

$$i = 83.53^\circ \text{ with 95\% CI [83.42, 83.66]}$$

$$P = 2.49 \text{ Days, with 95\% CI [2.41, 2.56].}$$

Table showing convergence:

| Iterations | $T_0$ / HJD | $\sigma_{T_0}$ / HJD | $\frac{R_\mathrm{p}}{R_\star}$ | $\sigma_{\frac{R_\mathrm{p}}{R_\star}}$ | $\frac{a}{R_\star}$ | $\sigma_{\frac{a}{R_\star}}$ | $i$ / $^\circ$ | $\sigma_i$ / $^\circ$ | $P$ / Days | $\sigma_P$ / Days |
|---|---|---|---|---|---|---|---|---|---|---|
| 1,000 | 2453989.752 | 0.0003 | 0.128 | 0.002 | 7.612 | 0.059 | 83.588 | 0.138 | 2.482 | 0.039 |
| 5,000 | 2453989.753 | 0.0001 | 0.128 | 0.001 | 7.600 | 0.058 | 83.537 | 0.060 | 2.485 | 0.037 |
| 10,000 | 2453989.753 | 0.0001 | 0.128 | 0.001 | 7.600 | 0.058 | 83.537 | 0.060 | 2.486 | 0.037 |
| 20,000 | 2453989.753 | 0.0001 | 0.128 | 0.001 | 7.600 | 0.057 | 83.537 | 0.060 | 2.485 | 0.037 |
| 50,000 | 2453989.753 | 0.0001 | 0.128 | 0.001 | 7.600 | 0.057 | 83.537 | 0.060 | 2.485 | 0.037 |

**Table 2.** Table of MCMC outputs for task 2 at particular sets of iterations such that the convergence within the parameter space can be explored.

Graph of Relative flux, $F$, against time from central transit, $t$ / HJD, for Kepler-90 planets with uniform LDP.

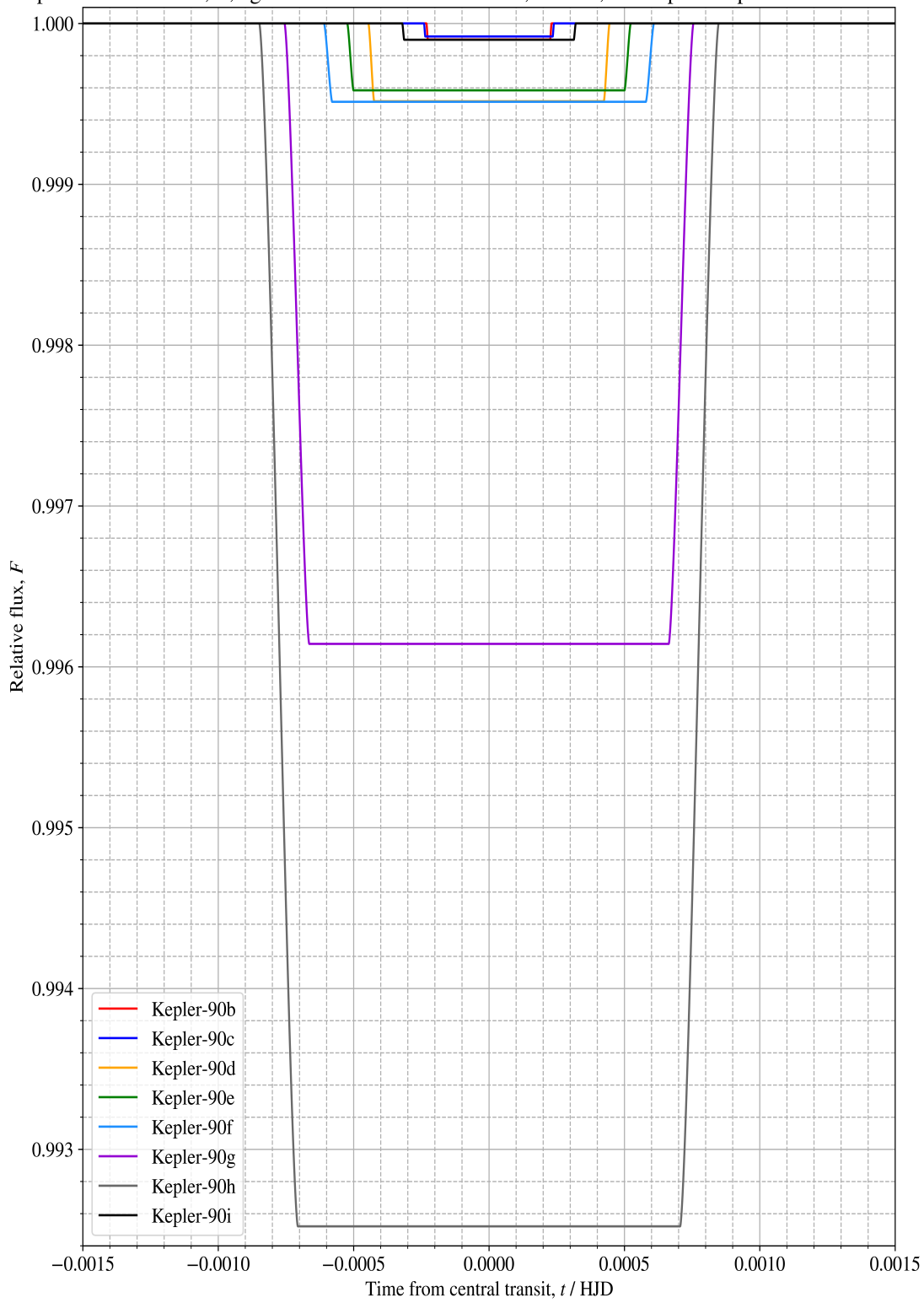


**Fig. 1.** Showing the uniform light curves for the exoplanets of Kepler-90.

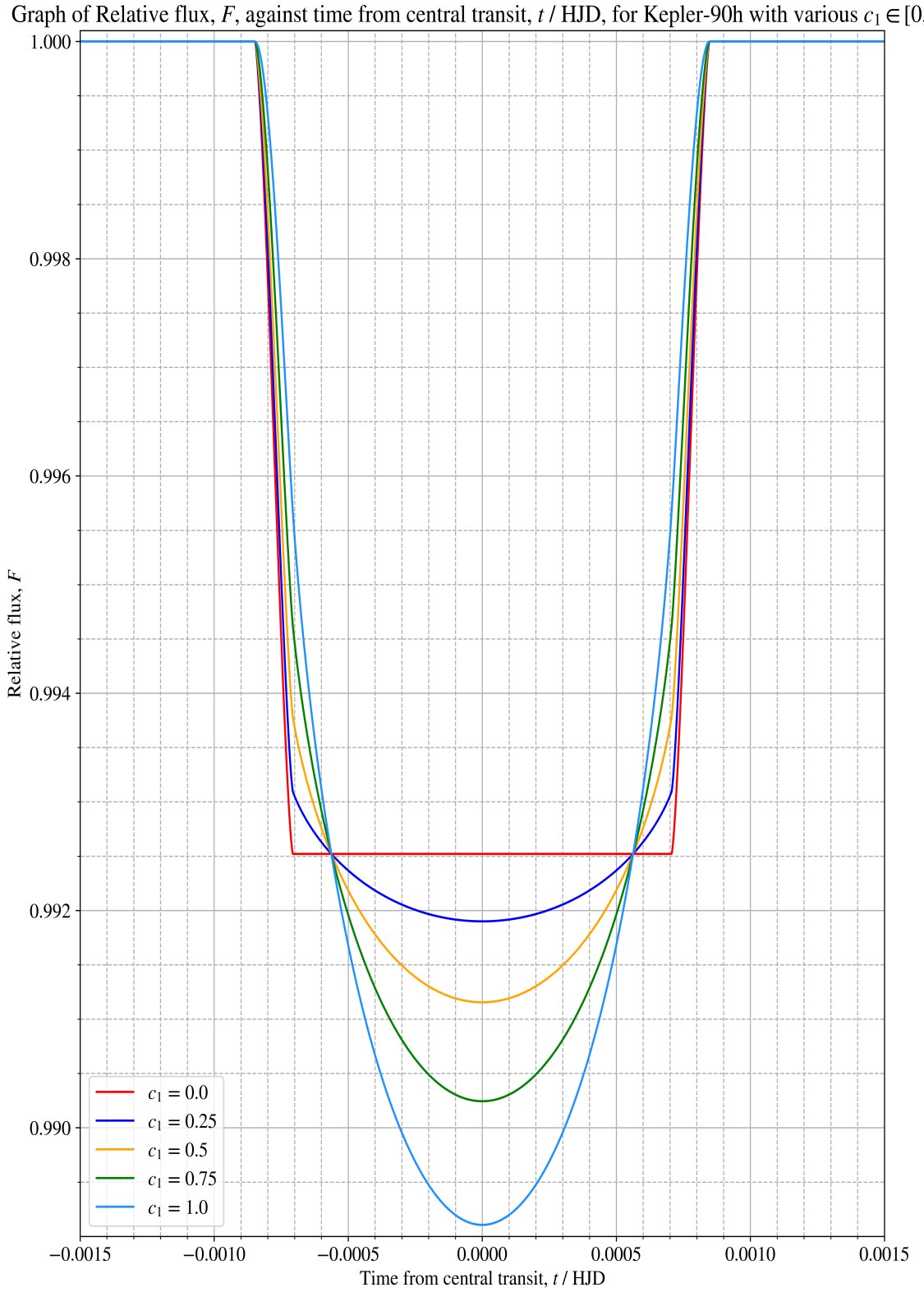


**Fig. 2.** Showing a linear light curve for the exoplanet Kepler-90h.

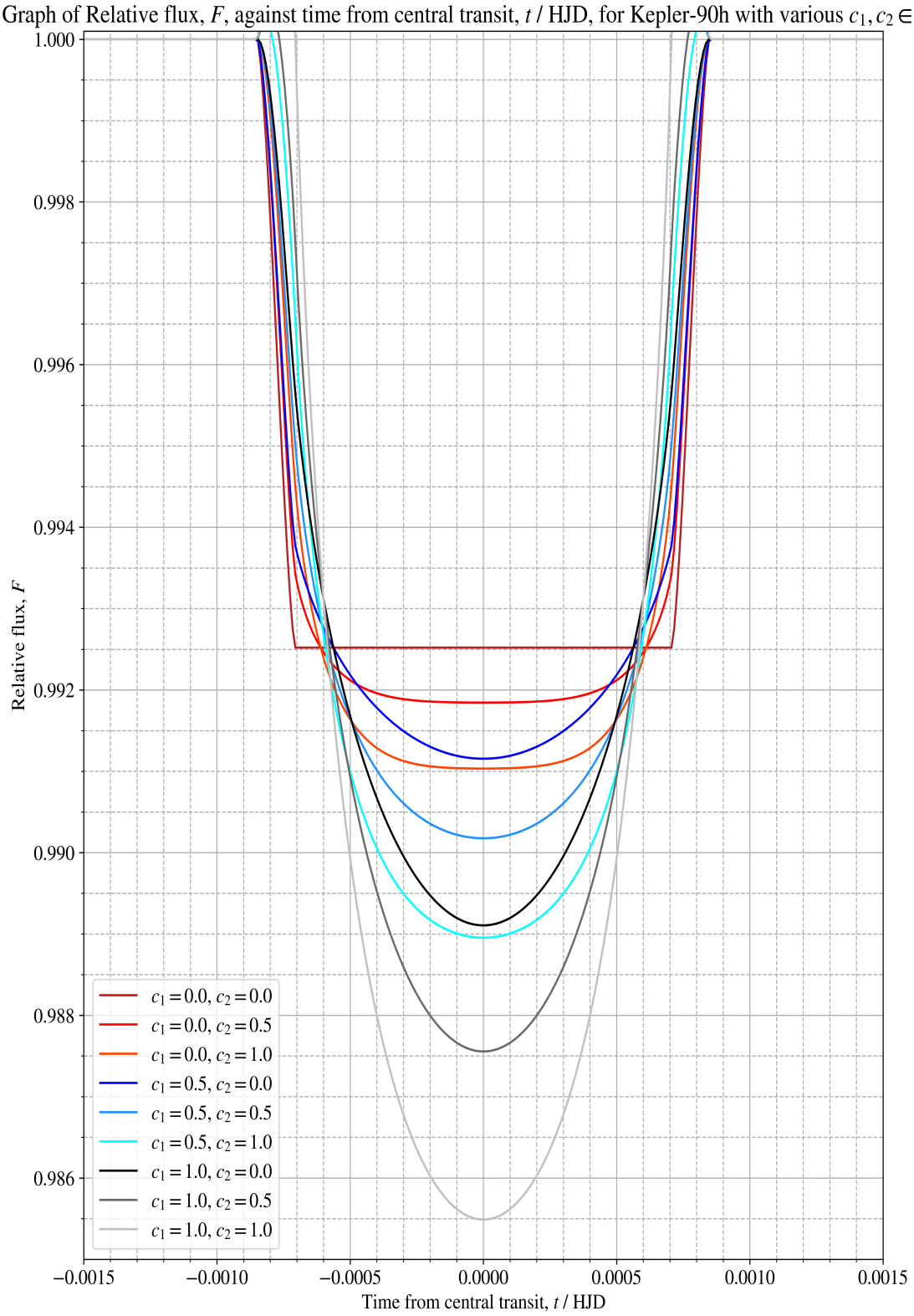


**Fig. 3.** Showing a quadratic light curve for the exoplanet Kepler-90h.

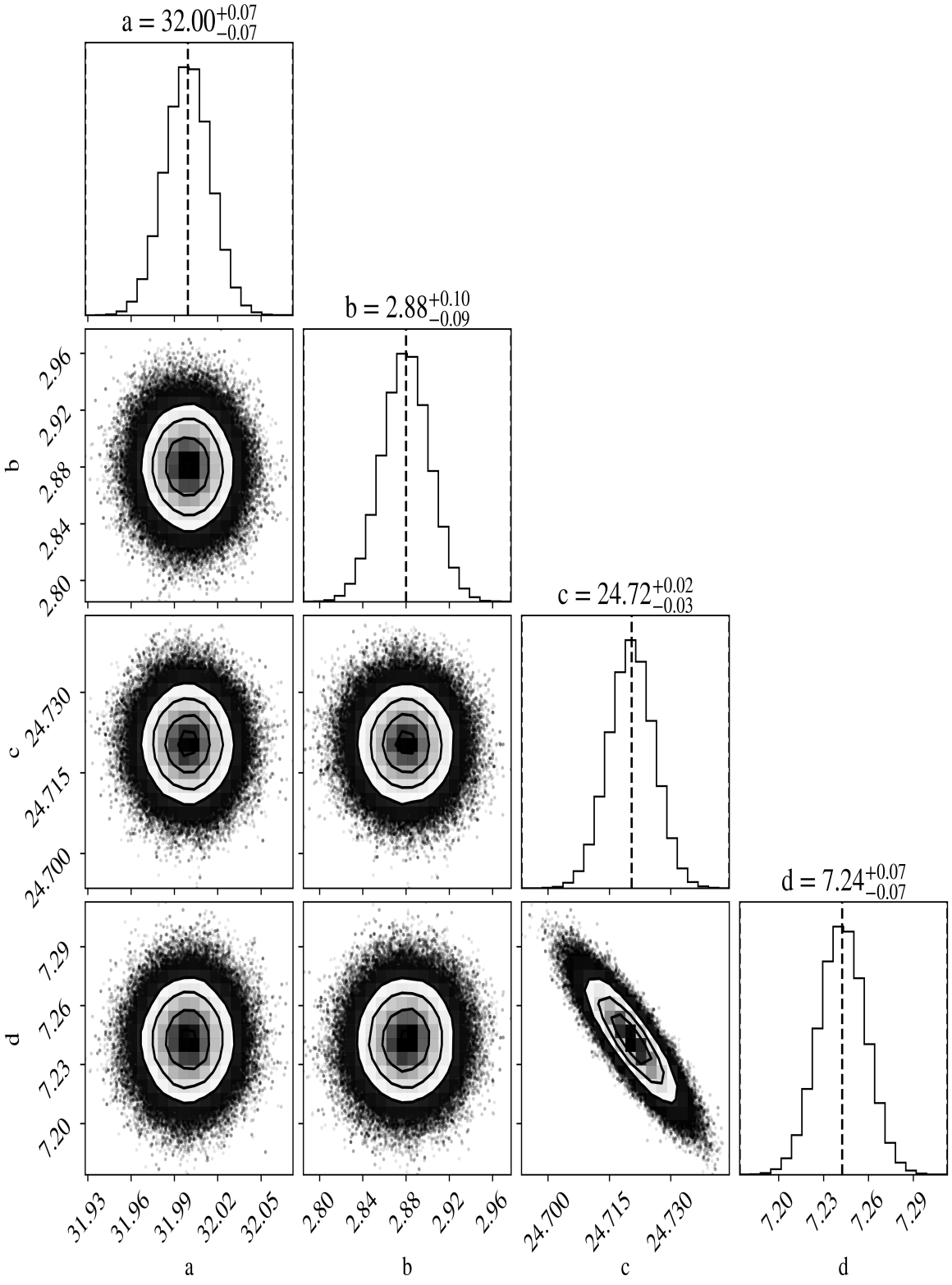


**Fig. 4.** Showing the values and corner plots obtained for $a, b, c$ and $d$. Each parameter follows a binomial distribution.

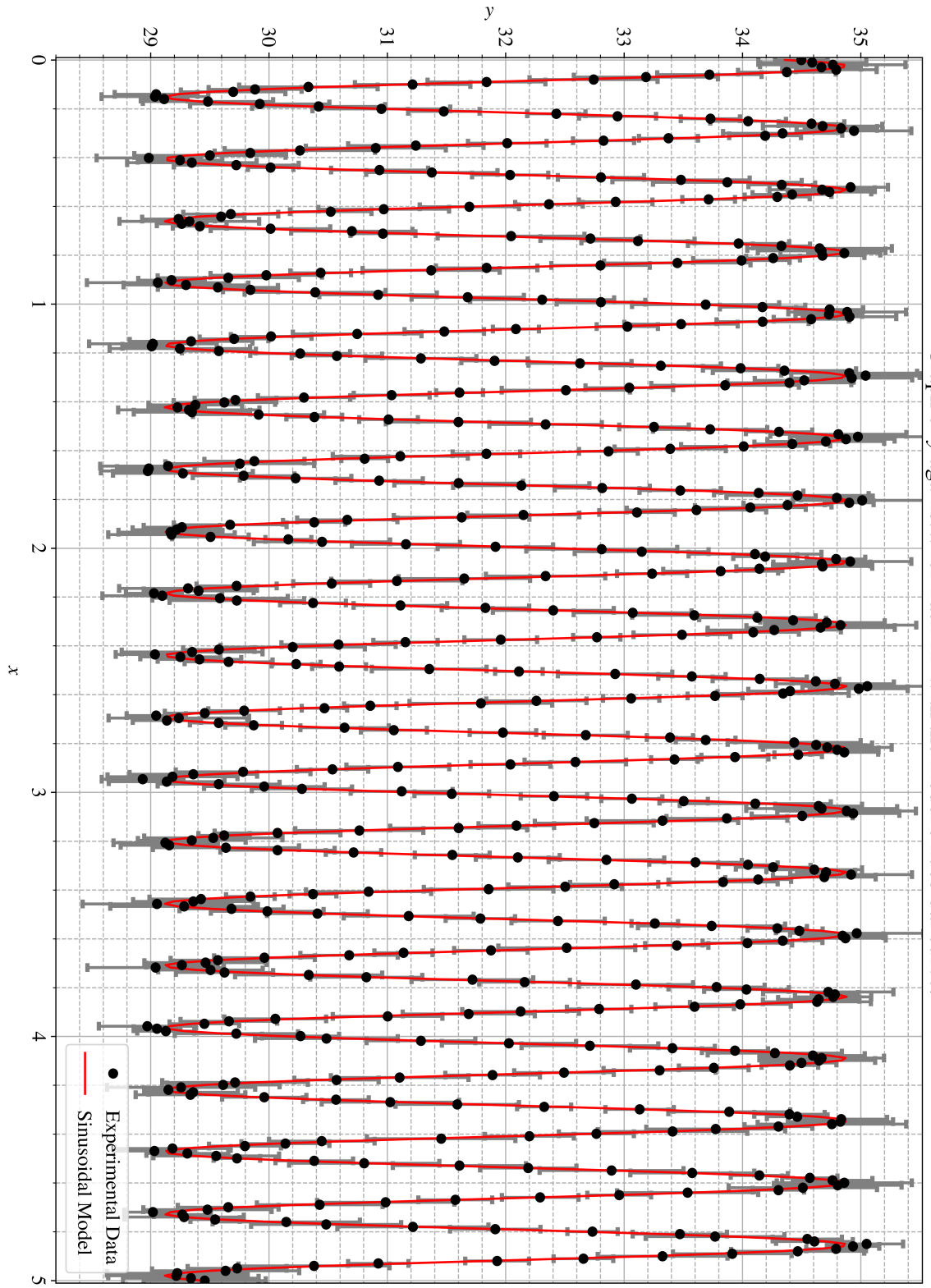


**Fig. 5.** Showing a graph of $y_{model}(a, b, c, d)$ against $x$.

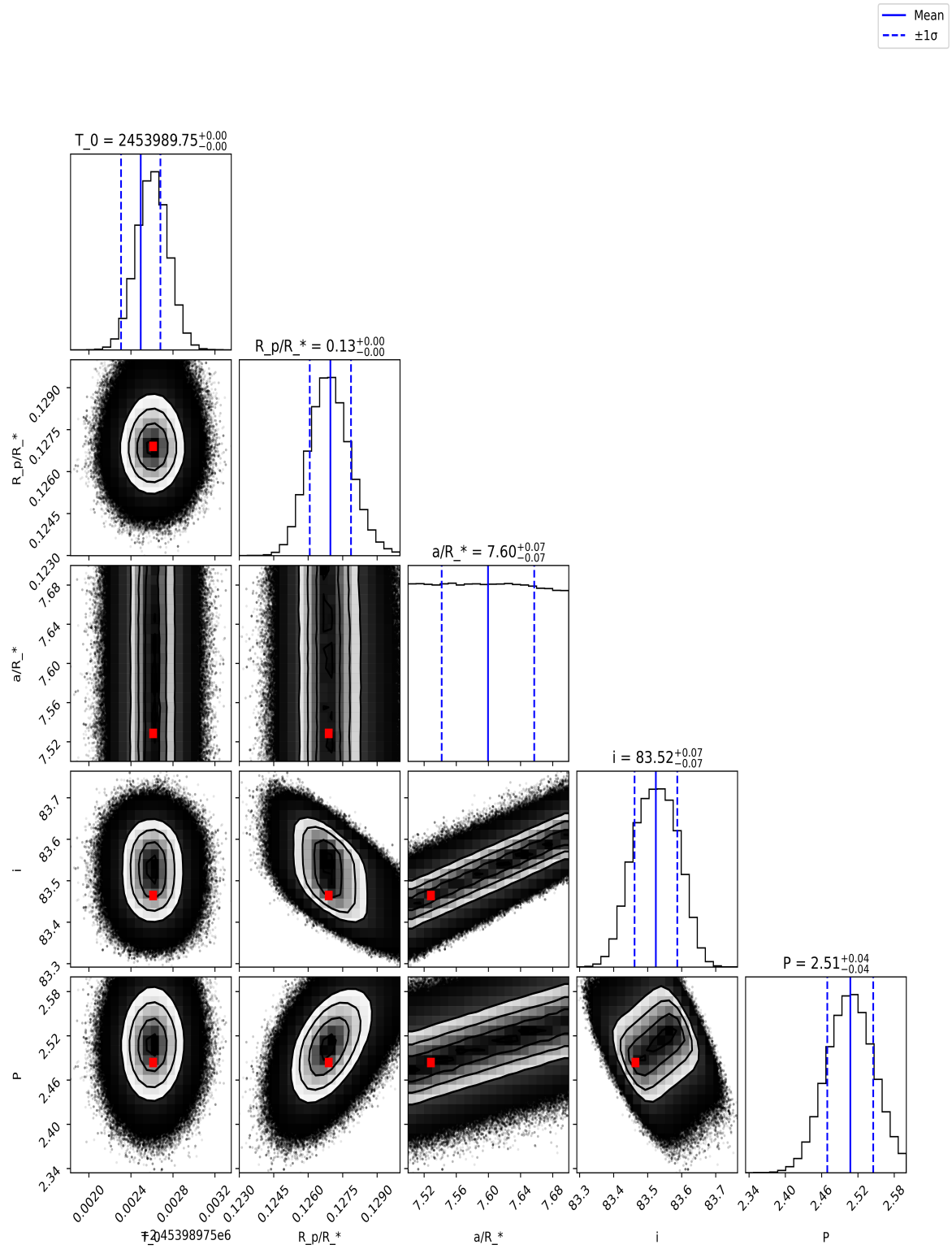


**Fig. 6.** Showing the values and corner plots obtained for $T_0$, $\frac{a}{R_\star}$, $\frac{R_\mathrm{p}}{R_\star}$, $i$, and $P$

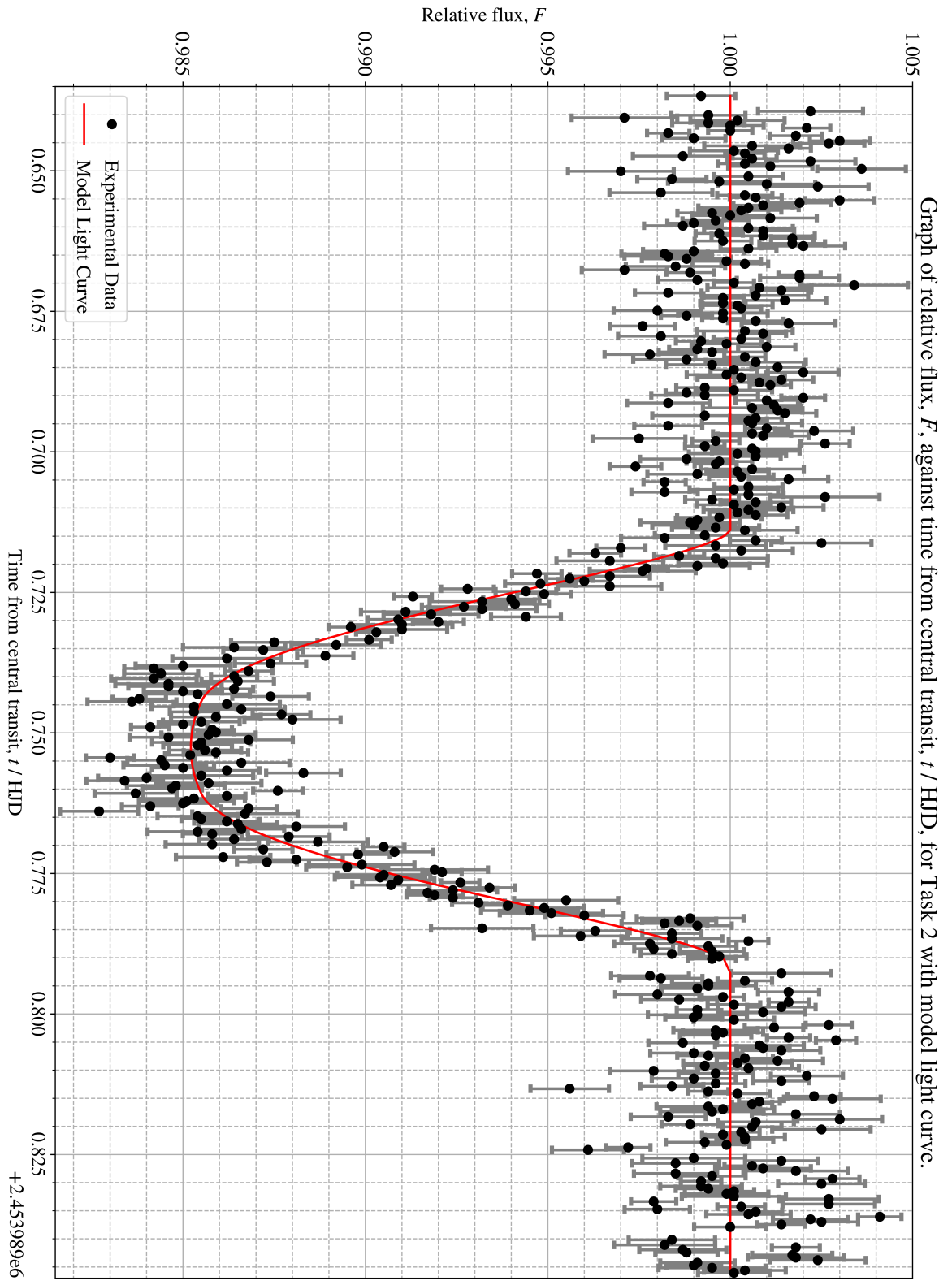


**Fig. 7.** Showing the light curve against the data given for the computed parameters $T_0$, $\frac{a}{R_\star}$, $\frac{R_\mathrm{p}}{R_\star}$, $i$, and $P$

### 3.3 Tools and Equations

The standard deviation used throughout the experiment is calculated by the equation:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{8}$$

where $n$ is the number of readings repeated, $X_i$ is the variable in account and $\bar{X}$ is the average value of those variables. This was used along the equation below to find the uncertainty of a respective value:

$$\Delta\bar{X} = \pm t_{\alpha,n-1}\frac{s}{\sqrt{n}} \tag{9}$$

Where $t_{\alpha,n-1}$ is a constant value, $n$ is the number of values being taken in consideration and $s$ is the standard deviation as shown in equation 8.

However, when the uncertainty of a variable depends on other variables, the uncertainty is then found using the below equation:

$$\Delta Y^2 = \sum_{i=1}^{n}\left(\frac{\partial Y}{\partial x_i} \times \Delta x_i\right)^2 \tag{10}$$

where $\Delta Y$ is the uncertainty of the variable being found and $\Delta x_i$ is the uncertainties of the other variables.

Additionally to find the confidence interval for 95 % ,

$$95\ \%\ \text{confidence interval} = \bar{X} \pm Z\frac{\sigma}{\sqrt{n}} \tag{11}$$

where $\bar{X}$ is the sample mean , $Z$ is a constant, which is 1.96 for 95 % confidence interval , $\sigma$ is the standard deviation and $n$ is the sample size (Lisa Sullivan, n.d.).

## 4 ANALYSIS

### 4.1 Task 1A

For task 1A , the stellar radius of Kepler-90 was found from theoretical values,

$$R_\star = 1.2 \pm 0.1\ R_\odot \text{ (Cabrera et al., 2014)} \tag{12}$$

Where $R_\odot$ is the solar radius. $R_\star$ was used to compute the $\frac{a_i}{R_\star}$ and $\frac{R_{p_i}}{R_\star}$ for each exoplanet of Kepler-90, where i ∈ (b, c, d, e, f, g, h, i). Then using batman it was defined the parameters of the exoplanets. Moreover the limb darkening was set to uniform. The limb darkening for uniform, Linear and Quadratic can determined by;

$$
\begin{aligned}
I(\mu) &= I_0 \text{ (Uniform)} \\
I(\mu) &= I_0[\,1 - c_1(1-\mu)] \text{ (Linear)} \\
I(\mu) &= I_0[\,1 - c_1(1-\mu) - c_2(1-\mu)^2] \text{ (Quadratic)}
\end{aligned} \tag{13}
$$

where $I_0$ is a normalization constant and $\mu$ is a normalised radial coordinate,

$$\mu = \sqrt{1-x} \quad 0 \leq x \leq 1 \text{ (Laura Kreidberg, 2015).} \tag{14}$$

Furthermore, the time array was defined and used alongside the `TransitModel` and `light_curve` functions to generate the light curves for all the exoplanets. As seen in figure 1.

The whole process was repeated for Kepler-90h, with the limb darkening type being linear and then repeated with quadratic. For linear, the coefficient $c_1$ was varied between 0 and 1, similarly, the coefficients of the quadratic $c_1, c_2$ were varied between 0 and 1. The respective graphs for linear and quadratic were plotted as seen in figures 2 and 3.

## 4.2 Task 1B

For task 1B, a periodic model function was defined as,

$$y_{model} = a + b\sin(cx + d) \tag{15}$$

with theta was defined as,

$$\theta = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{16}$$

Then likeliness, prior and probability functions were defined as follows:

```
def model_fn(theta, x):
    a, b, c, d = theta
    return a + b * np.sin(c * x + d)

```

```
def likelihood_fn(theta, x, y, yerr):
    a, b, c, d = theta
    y_model = model_fn(theta, x)

    likelihood = 0
    for i in range(len(x)):
        likelihood += ((y[i] - y_model[i]) / yerr[i]) ** 2
    likelihood *= - 0.5
    return likelihood

def prior_fn(theta):
    a, b, c, d = theta
    if (31.5 < a < 32.5) and (2.5 < b < 3) and (24.5 < c < 25.5) and (7 < d < 7.5):
        return 0
    return - np.inf

def prob_fn(theta, x, y, yerr):
    lp = prior_fn(theta)
    if not np.isfinite(lp):
        return - np.inf
    return lp + likelihood_fn(theta, x, y, yerr)
```

Moreover the initial priors for $c$ and $d$ were given to be,

$$\begin{aligned} 20 < c < 25 \\ 6 < d < 11 \end{aligned} \tag{17}$$

while the values for $a$ and $b$ were easily determined from the data given. By determining the translation on the $y$-axis and the amplitude, These were determined to be:

$$\begin{aligned} 30 < a < 31 \\ 2 < b < 3 \end{aligned} \tag{18}$$

Additionally the number of walkers and iterations were defined:

$$\begin{aligned} n_{\text{walkers}} = 100 \\ n_{iterations} = 1000 \end{aligned} \tag{19}$$

where the $iterations \in \big(1000, 5000, 10,000, 20,000, 50,000\big)$

alongside the step-sizes,

```
def _p0(initial, n_dim, n_walkers):
initial_params = [np.array(initial) + [1e-4, 75e-3, 1e-2, 1e-2] * np.random.randn(n_dim)
 for i in range (n_walkers)]
return initial_params
```

Lastly, MCMC was run to generate the likely values for $\theta$. Then a corner plot and a light curve were plotted to observe the correlation of values, as seen in figures 4 and 5. The obtained values for different number of iterations are found in table 1.

## 4.3 Task 2

Finding the prior values $T_0$, $\frac{a}{R_\star}$, $\frac{R_\mathrm{p}}{R_\star}$, $i$, and $P$:

From the given plotted data, one could easily deduce certain parameters. The inferior conjunction $T_0$ was determined from the minimum of the graph to be,

$$T_0 = 0.75 + 2.453989e6 \text{ HJD} \tag{20}$$

while the $\frac{R_\mathrm{p}}{R_\star}$ can be determined from the change in flux $\Delta F$ and the flux $F$ by,

$$\frac{\Delta F}{F} = \frac{R_\mathrm{p}^2}{R_\star^2} \text{ (Paul Anthony Wilson, 2015)} \tag{21}$$

the ratio of flux $\frac{\Delta F}{F}$ was determine from the given data, thus,

$$0.015 = \frac{R_p^2}{R_*^2} \tag{22}$$

$$0.12 = \frac{R_p}{R_*}. \tag{23}$$

Another important parameter, $\frac{a}{R*}$, can be determined from Kepler's third law. Since the semi-major axis is related to the period by:

$$\left(\frac{P^2}{R_\star^3}\frac{GM}{4\pi^2}\right)^{\frac{1}{3}} = \frac{a}{R_\star} \text{ (Holman et al., 2007).} \tag{24}$$

where $G$ is the gravitational constant and $M$ is the stellar mass. The orbital period $P$ was given to be around 2.5 days, thus using equation 24 one can find the ratio of the semi-major axis with respect to the stellar radius (Holman et al., 2007). This was found to be,

$$7.63 = \frac{a}{R_\star}. \tag{25}$$

Furthermore, the inclination $i$ is determined from theoretical values to be between 80 and 90 degrees (Holman et al., 2007).

After finding the initial (prior) parameters, batman was used to define the parameters of $\theta$ as the parameters of a light curve. The limb darkening type was set to quadratic and the coefficients were obtained from theoretical values as,

$$\begin{aligned} c_1 &= 0.22 \\ c_2 &= 0.32 \text{ (Holman et al., 2007)} \end{aligned} \tag{26}$$

Then, similarly to task 1B, MCMC was run to determine the likely values for $T_0$, $\frac{a}{R_\star}$, $\frac{R_\mathrm{p}}{R_\star}$, $i$, and $P$. Moreover a corner plot and a light curve of the deduced transit parameters were plotted, as seen in figure 6 and 7.

## 4.4 Percentage accuracy & error

Finding the percentage accuracy and error of $T_0$

$$\text{percentage accuracy} = \left(\frac{\text{Experimental Value}}{\text{Quoted Value}} - 1\right) \times 100\ \% \tag{27}$$

$$\text{percentage accuracy} = \left(\frac{2,453,989.75}{2,453,989.75286} - 1\right) \times 100\ \% \tag{28}$$

$$\text{percentage accuracy} = 0\ \% \tag{29}$$

$$\text{percentage error} = \left|\frac{\text{Combined error}}{\text{Experimental value}}\right| \times 100\ \% \tag{30}$$

$$\text{percentage error} = \left|\frac{0.00}{2,453,989.75}\right| \times 100\ \% \tag{31}$$

$$\text{percentage error} = 0\ \%. \tag{32}$$

Finding the percentage accuracy and error of $\frac{R_\mathrm{p}}{R_\star}$,

$$\text{percentage accuracy} = \left(\frac{0.13}{0.1253} - 1\right) \times 100\ \% \tag{33}$$

$$\text{percentage accuracy} = 4\ \% \tag{34}$$

$$\text{percentage error} = 0\ \%. \tag{35}$$

Finding the percentage accuracy and error of $\frac{a}{R_\star}$

$$\text{percentage accuracy} = \left(\frac{7.60}{7.63} - 1\right) \times 100\ \% \tag{36}$$

$$\text{percentage accuracy} = 0\ \% \tag{37}$$

$$\text{percentage error} = \left|\frac{0.10}{7.60}\right| \times 100\ \% \tag{38}$$

$$\text{percentage error} = 1\ \%. \tag{39}$$

finding the percentage accuracy and error of $i$

$$\text{percentage accuracy} = \left(\frac{83.52}{83.57} - 1\right) \times 100\ \% \tag{40}$$

$$\text{percentage accuracy} = 0\ \% \tag{41}$$

$$\text{percentage error} = \left|\frac{0.25}{83.52}\right| \times 100\ \% \tag{42}$$

$$\text{percentage error} = 0\ \%. \tag{43}$$

finding the percentage accuracy and error of $P$

$$\text{percentage accuracy} = \left(\frac{2.51}{2.47} - 1\right) \times 100\ \% \tag{44}$$

$$\text{percentage accuracy} = 2\ \% \tag{45}$$

$$\text{percentage error} = \left|\frac{0.14}{2.51}\right| \times 100\ \% \tag{46}$$

$$\text{percentage error} = 6\ \%. \tag{47}$$

Therefore all transit parameters are accurate and precise.

# 5 DISCUSSION

The experiment conducted demonstrated high accuracy, as the derived outcomes closely aligned with theoretical expectations. The obtained results consistently mirrored anticipated values across multiple parameters, affirming the robustness of the methodology employed. This alignment between the experimental and theoretical data signifies the reliability of the approach in determining the orbital parameters in transit methods. The close resemblance observed between the calculated and expected values across various parameters validates the precision and effectiveness of the experimental procedure (MCMC), suggesting confidence in the reliability of the findings.

The only limitation was the lack of fully convergence of $\frac{a}{R_\star}$, this proved to be challenging as it wouldn't converge regardless of the change it was introduced. Furthermore, when convergence followed for $\frac{a}{R_\star}$, it diverged the value of the orbital period $P$, suggesting a correlation between the semi-major axis and the orbital period. This was expected as it was already known from Kepler's third law, also known as the harmonics law, that the orbital period is related to the semi-major axis.

When the uniform limb darkening was applied in Task 1A, the light curve maintained a uniform shape, assuming that the relative flux decreases equally throughout the transit. Furthermore, In figure 1 when we proceed down to other Kepler-90 exoplanets, the light curve takes on a deeper form and longer time (for a transit), implying that as we travel down each exoplanet, the obstructed light during a transit rises. Additionally the limb darkening of linear and quadratic yields a similar, yet not identical, light curve shape. The quadratic type begins to exhibit a curved light curve before the linear kind as seen in figures 2 and 3.

Hot Jupiters are exoplanets discovered close to stars such as Kepler-1b; they possess similar mass or greater magnitude to that of Jupiter. They are defined as hot since the gas giants tend to reach huge temperatures due to being close to their stars. These pose a challenge to classic planet formation theories. since their close orbits challenge our concept of the formation of the solar system. Thus suggesting a change in the idea of planetary formation (Dawson & Johnson, 2018).

Incorporating complicated models to account for the planet's position in a multi-planetary system would be an improvement to such extensive transit simulations. Holman and Murray emphasise the need of adding complicated models in transit analysis within multi-planetary systems in their 2005 work. They stress how gravitational forces from nearby planets cause transit timing changes that alter measured transit periods. Their

research confirms that ignoring these interactions can lead to inaccurate transit time estimates. Furthermore they emphasise the importance of extensive modelling that accounts for these disturbances in order to appropriately interpret observed transit and find additional planets. Their findings highlight the need of studying multi-planetary interactions and arguing for complicated models to improve transit observation precision.

The experiment yielded highly aligned outcomes with theoretical expectations, validating the methodology's robustness in determining orbital parameters. However, the non-convergence of $\frac{a}{R_\star}$ posed a challenge, leading to a subsequent divergence with the orbital period $P$, suggesting a correlation between the semi-major axis and period, aligning with Kepler's laws. Holman and Murray (2005) advocate incorporating complex models for multi-planetary systems in transit analysis, emphasizing the significant impact of gravitational forces on transit timing variations. Their work stresses the necessity of intricate models to accurately interpret transit observations and detect additional planets, emphasising the importance of studying multi-planetary interactions for improved precision.

## REFERENCES

Cabrera, J Csizmadia, S Lehmann, H Dvorak, R Gandolfi, D Rauer, H Erikson, A Dreyer, C Eigmüller, P & Hatzes, A (2014) The Planetary System to KIC 11442793: A Compact Analogue to the Solar System. *Astrophysical Journal*, *781*(1) Article 18, 18. https://doi.org/10.1088/0004-637X/781/1/18

Dawson, R I & Johnson, J A (2018) Origins of Hot Jupiters. *Annual Review of Astron and Astrophys*, *56*, 175–221. https://doi.org/10.1146/annurev-astro-081817-051853

European Space Agency. (n.d.) *How to find an exoplanet*. https://www.esa.int/Science_Exploration/Space_Science/Exoplanets/How_to_find_an_exoplanet#:~:text=Astrometry%20means%20tracking%20the%20motion,around%20its%20centre%20of%20mass. (accessed: 18.11.2023).

European Space Agency. (2019) *Exoplanet detection methods*. https://sci.esa.int/web/exoplanets/-/60655-detection-methods#:~:text=Astrometry%20is%20the%20method%20that,mass%20of%20the%20planetary%20system. (accessed: 20.11.2023).

Holman, M J Winn, J N Latham, D W O'Donovan, F T Charbonneau, D Torres, G Sozzetti, A Fernandez, J & Everett, M E (2007) The transit light curve (tlc) project. vi. three transits of the exoplanet tres-2. *The Astrophysical Journal*, *664*(2) 1185. https://doi.org/10.1086/519077

Imad Pasha. (2020) *Mcmc: A (very) beginnner's guide*. https://prappleizer.github.io/Tutorials/MCMC/MCMC_Tutorial.html (accessed: 19.11.2023).

Laura Kreidberg. (2015) *Batman: Bad-ass transit model calculation*. https://lkreidberg.github.io/batman/docs/html/index.html (accessed: 17.11.2023).

Lisa Sullivan. (n.d.) *Confidence intervals*. https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_confidence_intervals/bs704_confidence_intervals_print.html (accessed: 25.11.2023).

Lovis, C & Fischer, D (2011) Radial velocity techniques for exoplanets. *Exoplanets, edited by S. Seager. Tucson, AZ: University of Arizona Press, 2011, 526 pp. ISBN 978-0-8165-2945-2., p.27-53*, 2–4.

National Aeronautics and Space Administration. (2013) *Light curves and what they can tell us*. https://imagine.gsfc.nasa.gov/science/toolbox/timing1.html (accessed: 25.11.2023).

O'Donovan, F T Charbonneau, D Mandushev, G Dunham, E W Latham, D W Torres, G Sozzetti, A Brown, T M Trauger, J T Belmonte, J A Rabus, M Almenara, J M Alonso, R Deeg, H J Esquerdo, G A Falco, E E Hillenbrand, L A Roussanova, A Stefanik, R P & Winn, J N (2006) Tres-2: The first transiting planet in the kepler field. *Astrophysical Journal, Letters*, *651*(1) L61–L64. https://doi.org/10.1086/509123

Paul Anthony Wilson. (2015) *The exoplanet transit method*. https://www.paulanthonywilson.com/exoplanets/exoplanet-detection-techniques/the-exoplanet-transit-method/ (accessed: 17.11.2023).

# APPENDIX

## 1 TASK 1A CODE LISTING

```
import pandas as pd
import batman
import numpy as np
import matplotlib.pyplot as plt

def light_curve_model(radius_ratio, distance_ratio, coeffs, profile, incl, ecc, period):
    parameters = batman.TransitParams()
    parameters.t0 = 0
    parameters.w = 0
    parameters.rp = radius_ratio
    parameters.a = distance_ratio
    parameters.u = coeffs
    parameters.limb_dark = profile
    parameters.inc = incl
    parameters.ecc = ecc
    parameters.per = period

    rel_flux = batman.TransitModel(parameters, t).light_curve(parameters)
    return rel_flux


# load data for analysis
data = pd.read_csv(fr"C:\Users\dimit\Downloads\kepler_90_pl.csv")
data = data.set_index('Planet')

#convert and compute required quantities
r_star = 834840000
data['Planetary Radius (Earth Radius)'] = data['Planetary Radius (Earth Radius)']   * 6378e3
ratio_radius = data['Planetary Radius (Earth Radius)']/ r_star
data.insert(1, "Ratio of Radii", ratio_radius.to_list())
ratio_dist = data['Semimajor Axis (AU)'] * 149597870700 / r_star
data.insert(2, "Ratio of A/R_star", ratio_dist.to_list())
data.columns = ['radius', "radius_ratio", "distance_ratio", 'period', 'inclination', '
    distance', 'eccentricity']


# plotting the model light curves for Kepler-90's planets: UNIFORM PROFILE
```

```
t = np.linspace(- 0.005, 0.005, 1000)     #Light curve domain
unif_lc_list = [light_curve_model(data.radius_ratio[n], data.distance_ratio[n], [], 'uniform', data.inclination[n], data.eccentricity[n], data.period[n]) for n in range(0, 8)]

plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams["font.size"] = 14
plt.rcParams["font.weight"] = "normal"

plt.figure(figsize=(11,15))
plt.ylim(0.9924, 1.0001)
plt.xlim(- 0.0015, 0.0015)
plt.minorticks_on()
plt.grid(which='major', linestyle='-')
plt.grid(which='minor', linestyle='--')
plt.title(r"Graph of Relative Flux against Time from Cetral Transit / Days.")
plt.xlabel(r"Time from Cetral Transit / HJD")
plt.ylabel(r"Relative Flux")

letter_list = ['b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']    # list used for labels of each light curve
color_list = ['r', 'b', 'orange', 'g', 'dodgerblue', 'darkviolet', 'dimgray', 'k']    # list used for colour of each light curve
for n in range(0, 8):
    plt.plot(t, unif_lc_list[n], color=color_list[n], label=fr"Kepler-90 {letter_list[n]}")

plt.legend()
plt.savefig('kepler90_unifrom_lightcurves', dpi=600, bbox_inches='tight')
plt.show()


# plotting the model light curves for Kepler-90's planets: LINEAR PROFILE
c1_list = np.linspace(0, 1, 5)
lin_lc_list = [light_curve_model(data.radius_ratio[6], data.distance_ratio[6], [c1], 'linear', data.inclination[6], data.eccentricity[6], data.period[6]) for c1 in c1_list]

plt.figure(figsize=(11,15))
plt.ylim(0.989, 1.0001)
plt.xlim(- 0.0015, 0.0015)
plt.minorticks_on()
plt.grid(which='major', linestyle='-')
```

```
plt.grid(which='minor', linestyle='--')
plt.title(fr"Graph of Relative Flux against Time from Cetral Transit / Days, for Kepler-90 h with various $c_1 \in [0, 1]$ .")
plt.xlabel(r"Time from Cetral Transit / Days")
plt.ylabel(r"Relative Flux")

color_list = ['r', 'b', 'orange', 'g', 'dodgerblue']
for n in range(len(c1_list)):
    plt.plot(t, lin_lc_list[n], color=color_list[n], label=fr"$c_1$ = {round(c1_list[n], 3)}")
plt.legend()
plt.savefig('linear_light_curve', dpi=600, bbox_inches='tight')
plt.show()


# plotting the model light curves for Kepler-90's planets: QUADRATIC PROFILE
c1_c2_list = [[c1, c2] for c1 in const_list for c2 in np.linspace(0, 1, 3)]    # cartesian product to obtain all combinations of constants
light_curve_list = [light_curve_model(data.radius_ratio[6], data.distance_ratio[6], c1_c2, 'quadratic', data.inclination[6], data.eccentricity[6], data.period[6]) for c1_c2 in c1_c2_list]

plt.figure(figsize=(11,15))
plt.ylim(0.985, 1.0001)
plt.xlim(- 0.0015, 0.0015)
plt.minorticks_on()
plt.grid(which='major', linestyle='-')
plt.grid(which='minor', linestyle='--')
plt.title(fr"Graph of Relative Flux against Time from Cetral Transit / Days, for Kepler-90 h with various $c_1 \in [0, 1]$ .")
plt.xlabel(r"Time from Cetral Transit / Days")
plt.ylabel(r"Relative Flux")

color_list = ['firebrick', 'red', 'orangered', 'blue', 'dodgerblue', 'cyan', 'k', 'dimgray', 'silver']    # list used for colour of each light curve
for n in range(len(c1_c2_list)):
    plt.plot(t, light_curve_list[n], color=color_list[n], label=fr"$c_1 = {c1_c2_list[n][0]}$, $c_2 = {c1_c2_list[n][1]}$")
plt.legend()
plt.savefig('quadratic_light_curve', dpi=600, bbox_inches='tight')
```

```
106 plt.show()
```

**Listing 1.** Python code used for Task 1A.

# 2 TASK 1B CODE LISTING

```
import pandas as pd
import batman
import emcee as mc
import numpy as np
import matplotlib.pyplot as plt
import corner

def model_fn(theta, x):
    a, b, c, d = theta
    return a + b * np.sin(c * x + d)

def likelihood_fn(theta, x, y, yerr):
    a, b, c, d = theta
    y_model = model_fn(theta, x)

    likelihood = 0
    for i in range(len(x)):
        likelihood += ((y[i] - y_model[i]) / yerr[i]) ** 2
    likelihood *= - 0.5
    return likelihood

def prior_fn(theta):
    a, b, c, d = theta
    if (31.5 < a < 32.5) and (2.5 < b < 3) and (24.5 < c < 25.5) and (7 < d < 7.5):
        return 0
    return - np.inf

def prob_fn(theta, x, y, yerr):
    lp = prior_fn(theta)
    if not np.isfinite(lp):
        return - np.inf
    return lp + likelihood_fn(theta, x, y, yerr)

def _p0(initial, n_dim, n_walkers):
    initial_params = [np.array(initial) + [1e-4, 75e-3, 1e-2, 1e-2] * np.random.randn(n_dim)
     for i in range (n_walkers)]
    return initial_params


n_walkers = 200
```

```
initial = np.array([32, 2.7, 25, 7.5])
n_dim = len(initial)
p0 = _p0(initial, n_dim, n_walkers)  # initial parameters

data1 = pd.read_csv(fr"C:\Users\dimit\Downloads\mcmc_intro_group_5.csv")
x = np.array(data1.x)
y = np.array(data1.y)
yerr = np.array(data1.y_err)


# mcmc initialisation
sampler = mc.EnsembleSampler(n_walkers, n_dim, prob_fn, args=(x, y, yerr))

p, _, _ = sampler.run_mcmc(p0, 1000, progress=True) # first 1k iterations. Total of 1k so far
samples = sampler.flatchain
print(f"a = {samples.T[0].mean()}, b = {samples.T[1].mean()}, c = {samples.T[2].mean()}, d = {samples.T[3].mean()}")
sampler.reset()

p, _, _ = sampler.run_mcmc(p, 4000, progress=True) # another 4k iterations. Total of 5k so far
samples = sampler.flatchain
print(f"a = {samples.T[0].mean()}, b = {samples.T[1].mean()}, c = {samples.T[2].mean()}, d = {samples.T[3].mean()}")
sampler.reset()

p, _, _ = sampler.run_mcmc(p, 5000, progress=True) # another 5k iterations. Total of 10k so far
samples = sampler.flatchain
print(f"a = {samples.T[0].mean()}, b = {samples.T[1].mean()}, c = {samples.T[2].mean()}, d = {samples.T[3].mean()}")
sampler.reset()

p, _, _ = sampler.run_mcmc(p, 10000, progress=True) # another 10k iterations. Total of 20k so far
samples = sampler.flatchain
print(f"a = {samples.T[0].mean()}, b = {samples.T[1].mean()}, c = {samples.T[2].mean()}, d = {samples.T[3].mean()}")
sampler.reset()

```

```
p, _, _ = sampler.run_mcmc(p, 30000, progress=True) # another 30k iterations. Total of 50k so far
samples = sampler.flatchain
print(f"a = {samples.T[0].mean()}, b = {samples.T[1].mean()}, c = {samples.T[2].mean()}, d = {samples.T[3].mean()}")


# output of corner plot
fig = corner.corner(samples, show_titles=True ,labels=['a', 'b', 'c', 'd'],  plot_datapoints=True, quantiles=[0,0.5,1]) #plot parameters
plt.savefig('task1b_mcmc', dpi=600, bbox_inches='tight')
plt.show()

# plot model function with optimised parameters
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams["font.size"] = 14
plt.rcParams["font.weight"] = "normal"

plt.figure(figsize=(15,11))
plt.ylim(28.2, 35.52)
plt.xlim(- 0.01, 5.01)
plt.minorticks_on()
plt.grid(which='major', linestyle='-')
plt.grid(which='minor', linestyle='--')
plt.title(fr"Graph of $y$ against $x$ for Task 1B.")
plt.xlabel(r"$x$")
plt.ylabel(r"$y$")

plt.errorbar(x, y, xerr=0, yerr=yerr, fmt='o', color='k', elinewidth=2, capsize=3, capthick=3, ecolor='gray')  #errorbars
plt.plot(x, y,  'ko', label='Experimental Data')
plt.plot(x, model_fn([samples.T[0].mean(), samples.T[1].mean(), samples.T[2].mean(), samples.T[3].mean()], x), label='Trendline', color='k')
plt.legend()

plt.savefig('task1b_model', dpi=600, bbox_inches='tight')
plt.show()
```

**Listing 2.** Python code used for Task 1B.

# 3 TASK 2 CODE LISTING

```
import pandas as pd
import batman
import emcee as mc
import numpy as np
import matplotlib.pyplot as plt
import corner

def model_fn(theta, hjd):
    t0, rp, a, inc, per = theta   #deparse list into separate variables
    parameters = batman.TransitParams()
    parameters.ecc = 0
    parameters.w = 0
    parameters.u = [0.22, 0.32] # limb coefficients
    parameters.limb_dark = 'quadratic' # limb darkening model

    parameters.t0 = t0 #mcmc variable
    parameters.rp = rp #mcmc variable
    parameters.a = a #mcmc variable
    parameters.inc = inc #mcmc variable
    parameters.per = per #mcmc variable

    #Generating data points
    return batman.TransitModel(parameters, hjd).light_curve(parameters)

def likelihood_fn(theta, hjd, flux, err):
    t0, rp, a, inc, per = theta   #deparse list into separate variables
    y_model = model_fn(theta, hjd)

    likelihood = 0
    for i in range(len(hjd)):
        likelihood += ((flux[i] - y_model[i]) / err[i]) ** 2
    likelihood *= - 0.5
    return likelihood


def prior_fn(theta):
    t0, rp, a, inc, per = theta
    if (2453989.7 < t0 < 2453989.8) and (2.2 < per < 2.6) and (0.12 < rp < 0.13 ) and (7.5 <
     a < 7.7) and (83 < inc < 85):
        return 0
```

```
    else:
        return - np.inf

def prob_fn(theta, hjd, flux, err):
    lp = prior_fn(theta)
    if not np.isfinite(lp):
        return - np.inf
    return lp + likelihood_fn(theta, hjd, flux, err)


def _p0(initial, n_dim, n_walkers):
    initial_params = [np.array(initial) + [0.001, 0.001, 0.01, 0.001, 0.001]*np.random.randn
    (n_dim) for i in range (n_walkers)]
    return initial_params


data =pd.read_excel(r"C:\Users\omarj\Downloads\kepler_lc_group_4 (2).xlsx")
hjd = np.array(data.HJD)
flux = np.array(data.Rel_Flux)
err = np.array(data.Flux_err)

n_walkers = 500
initial = np.array([2453989.75, 0.13, 7.6, 83, 2.5])
n_dim = len(initial)
p0 = _p0(initial, n_dim, n_walkers)  #initial parameters

#mcmc initialisation
sampler = mc.EnsembleSampler(n_walkers, n_dim, prob_fn, args=(hjd, flux, err))

p, _, _ = sampler.run_mcmc(p0, 1000, progress=True) #initial run, 1000 iterations
samples = sampler.flatchain
print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
    inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
sampler.reset()

p, _, _ = sampler.run_mcmc(p, 4000, progress=True) # another 4k iterations. Total of 5k so
    far
samples = sampler.flatchain
print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
    inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
sampler.reset()
```

```
77
78 p, _, _ = sampler.run_mcmc(p, 5000, progress=True) # another 5k iterations. Total of 10k so
       far
79 samples = sampler.flatchain
80 print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
       inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
81 sampler.reset()
82
83 p, _, _ = sampler.run_mcmc(p, 10000, progress=True) # another 5k iterations. Total of 10k so
        far
84 samples = sampler.flatchain
85 print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
       inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
86 sampler.reset()
87
88 p, _, _ = sampler.run_mcmc(p, 30000, progress=True) # another 5k iterations. Total of 10k so
        far
89 samples = sampler.flatchain
90 print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
       inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
91 sampler.reset()
92
93 p, _, _ = sampler.run_mcmc(p, 50000, progress=True) # another 5k iterations. Total of 10k so
        far
94 samples = sampler.flatchain
95 print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
       inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
96
97 p, _, _ = sampler.run_mcmc(p, 100000, progress=True) # another 5k iterations. Total of 10k
       so far
98 samples = sampler.flatchain
99 print(f"t0 = {samples.T[0].mean()}, rp = {samples.T[1].mean()}, a = {samples.T[2].mean()},
       inc = {samples.T[3].mean()}, per = {samples.T[4].mean()}")
100
101
102 # output of corner plot
103 plt.rcParams['font.family'] = 'STIXGeneral'
104 plt.rcParams['mathtext.fontset'] = 'stix'
105 plt.rcParams["font.size"] = 14
106 plt.rcParams["font.weight"] = "normal"
107
```

```
fig = corner.corner(samples, show_titles=True ,labels=[fr'$t_0$', r'$\frac{r_p}{R_{\star}}$', r"$\frac{a}{R_{\star}}$", fr'$\theta$', fr'$T$'],  plot_datapoints=True, quantiles=[0,0.5,1]) #plot parameters
plt.savefig('task2_mcmc_v2', dpi=600, bbox_inches='tight')
plt.show()

# plot model function with optimised parameters
plt.figure(figsize=(15,11))
plt.minorticks_on()
plt.grid(which='major', linestyle='-')
plt.grid(which='minor', linestyle='--')
plt.title(fr"Relative Flux  against Time from cetral transit / HJD.")
plt.ylabel(r"Relative Flux")
plt.xlabel(r"Time from cetral transit / HJD")

plt.errorbar(hjd, flux, xerr=0, yerr=err, fmt='o', color='k', elinewidth=2, capsize=3, capthick=3, ecolor='gray')  #errorbars
plt.plot(hjd, flux,  'ko', label='Experimental Data')
plt.plot(hjd, model_fn([samples.T[0].mean(), samples.T[1].mean(), samples.T[2].mean(), samples.T[3].mean(), samples.T[4].mean()], hjd), label='Trendline', color='r')
plt.legend()




# computing the standard deviation with the highest likelihood model
samples = sampler.flatchain  # Retrieve MCMC samples

# Compute mean and standard deviation for each parameter from MCMC samples
param_means = np.mean(samples, axis=0)
param_std = np.std(samples, axis=0)

# Find index of highest likelihood model
likelihood_values = sampler.flatlnprobability  # Extract likelihood values ( log probability )
The_highest_likelihood = np.argmax(likelihood_values) # choosing the highest likelihood value
highest_likelihood_params = samples[The_highest_likelihood]

# Plot posterior distribution with corner plot
fig = corner.corner(samples, labels=['T_0', 'R_p/R_*', 'a/R_*', 'i', 'P'], show_titles=True, plot_datapoints=True)
```

```

# Mark the highest likelihood model on the corner plot
ndim = len(initial)
axes = np.array(fig.axes).reshape((ndim, ndim))
for i in range(ndim):
    for j in range(ndim):
        ax = axes[i, j]
        if j < i:
            ax.plot(highest_likelihood_params[j], highest_likelihood_params[i], 'rs')  # Mark the highest likelihood model

# Compute and plot 1$\sigma$ spread on diagonal elements
for i, ax in enumerate(axes.diagonal()):
    mean = param_means[i]
    std = param_std[i]
    ax.axvline(mean, color='b', label='Mean')
    ax.axvline(mean - std, color='b', linestyle='--', label=' $\pm$1$\sigma$')
    ax.axvline(mean + std, color='b', linestyle='--')
    ax.set_yticks([])  # Remove y-axis ticks for clarity

plt.legend()
plt.savefig('posterior_with_likelihood_model.png', dpi=600, bbox_inches='tight')
plt.show()
```

**Listing 3.** Python code used for Task 2.

# 4 APPENDIX B

Listing 1 shows the computed code to generate figures 1, 2, and 3. The main programming language used is Python, along with the packages batman, pandas, numpy, and matplotlib. Batman was used to generate light curves for the provided data. Three different light curves were generated: uniform, linear, and quadratic, which were plotted in figures 1, 2, and 3. Moreover, in listing 2, it was defined a model for the periodic data provided along with the likeliness, prior, and probability functions, which determined the likely values for the model. MCMC was computed with defined iterations and walkers to find such values. Figures 4 and 5 show the corner plot and the light curve with these deduced parameters.

In listing 3, there was a combination of batman and MCMC; these were used in conjunction to determine the likely parameter values of an exoplanet being in transit. Similarly, the model, likeliness, prior, and probability functions were defined. This time the model function embedded the transit parameters to generate the likely light curve using batman after obtaining the likely values of the transit parameters from MCMC.