Observational Astronomy: Exoplanet Transit Parameters

Daniele Caruana

Group 8

PHY2235

B.Sc.(Hons) in Physics

23rd March 2023

Abstract

The transit method describes one of the most effective methods to detect exoplanets using the brightness of stars. In this paper, an overview of the batman library was shown to calculate light curves for different planets. The MCMC method was shown to be able to be effective in obtaining accurate estimates for the transit parameters for Kepler-1b, given observational data.

1 INTRODUCTION & THEORETICAL BACKGROUND

1.1 Exoplanets and the Transit Method

Exoplanets are planets residing in systems outside of our own solar system, orbiting a star or a stellar remnant, or rogue planets orbiting the center of the galaxy. There are several methods to detect the existence of an exoplanet, the most effective being the radial-velocity method, and the Transit method.

The Transit method uses an exoplanet's transit around its observed star in order to discover the exoplanet. When a planet's orbit is at the points at which it is in front of the star, from Earth's perspective, the brightness of the star is observed to dim. This change in brightness (or relative flux) can be measured using photometric instruments, more specifically charged-couple devices, and illustrated using light curves. Several estimates of the characteristics of a planet can be determined by observed light curves, such as the orbital period, the radius of the planet, and the inclination.

1.2 Limb Darkening Effect

Limb darkening is the visually apparent diminished brightness at the edge of a star, when compared to its centre. This effect significantly alters the shape of the light curves of an observed star, hence showing the true depth of the transit, giving a better estimate of the radius of the exoplanet. There are several laws (Espinoza & Jordán, 2015) describing limb darkening, used for different types of stars and light-curves, some of which are given as follows:

$$\begin{aligned}
&\text{Uniform law}: \quad I(\mu) = I_0 \\
&\text{Linear law}: \quad I(\mu) = I_0(1 - u_1[1-\mu]) \\
&\text{Quadratic law}: \quad I(\mu) = I_0(1 - u_1[1-\mu] - u_2[1-\mu]^2) \\
&\text{Logarithmic law}: \quad I(\mu) = I_0(1 - l_1(1-\mu) - l_2\mu \,\ln\mu,
\end{aligned} \tag{1}$$

where $\mu = \cos\theta$, and θ is the angle between the observer's perspective plane, and the normal to the stellar surface.

1.3 Hot Jupiters

Hot Jupiters are a type of exoplanet located outside our solar system which have a similar size to Jupiter, however orbit their star at a very close distance, giving them extraordinarily high temperatures which can exceed a thousand degrees Celsius. Due to this, the interaction between Hot Jupiters' atmosphere and the intense solar radiation is extremely strong, causing strong atmospheric winds that can reach speeds of up to several kilometers per second, and high levels of atmospheric ionization.

Due to Hot Jupiters' massive size and short semi-major axis, their detection and viability of research is much easier when compared to smaller planets. This also facilitates the practice of transit spectroscopy as it allows more light to pass through the planets' atmosphere. Further more, due to their rapid orbit, their transit can be measured more frequent in a set time.

2 MARKOV CHAIN MONTE CARLO (MCMC) METHODS

MCMC describes a type of computation used to obtain accurate samples from a selected distribution. This is used in scenarios where simply obtaining independent samples does not compute results accurately.

Monte Carlo methods are statistical computations which are used to estimate the parameters of a given model. This is done by obtaining random samples from a probability distributions. While this can work for certain simple functions, this will not be sufficient for higher-order functions. Markov Chains are memory-less systems that produce a sequence of states, each one depending only on the active probabilistic properties. MCMC is a Bayesian process which uses a combination of these two methods, meaning that it is dependent on a probability distribution which is based on a set of ranges, called priors, assumed to be true by the analyst. The process works by having a number of walkers which start from an initial set of parameters, θ. Each walker then jumps to a new set of parameters, θ_{trial}, within the previously set priors. The new values of θ_{trial} are then compared to the experimentally observed/measured data using a function which calculates a value for the likeliness of the new trial parameters, such as a χ^2-type check given by:

$$\ln\ell(\theta|x_1, x_2, ..., x_n) = \frac{1}{2}\sum_i^n \left(\frac{y_{obs_i} - y_{model_i}}{y_{err_{obs,i}}}\right)^2 . \tag{2}$$

If the new parameters satisfy the likelihood function, the current state's parameters will be considered as the current parameters, and the probabilistic determination for the next jump will only depend on these current parameters. This process is iterated for a number of times until it is determined that a convergence has been achieved within the distribution.

Besides using the samples to obtain accurate estimates of some parameters, samples can also be used in order to obtain a posterior probability distribution. Posterior distributions describe the uncertainty of our achieved model based on the likelihood and the prior distributions. From the known cumulative distribution function of a normal distribution, one can obtain three values for sigma or standard deviations, 68%, 95%, and 99.7%, in order to compare obtained values with the interval estimate of the obtained normal distribution.

3 DATA ANALYSIS AND RESULTS

This paper displays three different analyses: light curve modelling of Kepler-90 planets using the batman package, fitting a curve to a dataset using MCMC, and fitting a transit curve to an observed dataset of Kepler-1b.

3.1 Exoplanet transit light curve modelling for Kepler-90 planets

Provided data containing transit and physical parameters of all Kepler-90 planets was imported using the pandas library. These parameters consist of the planetary radius (R_p), the orbital period (P), the orbital inclination (i), the semi-major axis (a), and the eccentricity (e). It was made sure that the values for the planetary radius, and the semi-major axis were converted into units of stellar radii, $\frac{R_p}{R_\star}$ and $\frac{a}{R_\star}$ respectively, where $R_\star \approx 1.25$ R$_\odot$ (Stassun et al., 2019).

Planet	R_p / $R_\star$	P /days	a / $R_\star$	i / $^\circ$	e
Kepler-90 b	9.61E-03	7.008	1.273E+01	89.400	0.000
Kepler-90 c	8.65E-03	8.719	1.531E+01	89.680	0.000
Kepler-90 d	2.11E-02	59.737	5.505E+01	89.710	0.000
Kepler-90 e	1.96E-02	91.939	7.225E+01	89.790	0.000
Kepler-90 f	2.12E-02	124.914	8.257E+01	89.770	0.010
Kepler-90 g	5.96E-02	210.607	1.221E+02	89.920	0.049
Kepler-90 h	8.30E-02	331.601	1.737E+02	89.927	0.011
Kepler-90 i	9.68E-03	14.449	1.841E+01	89.200	0.000

The batman (Kreidberg, 2015) package was first used in order to generate light curves for all planets, using a uniform limb darkening profile, in which the time of inferior conjunction (t_0), and the longitude of periastron (w) were set to be $t_0 = w = 0$. Then, the light curves for Kepler-90 e were plotted using both a linear and a quadratic limb darkening profile at different values of the limb darkening coefficients u_1 for the linear profile, and u_1 and u_2 for the quadratic profile.

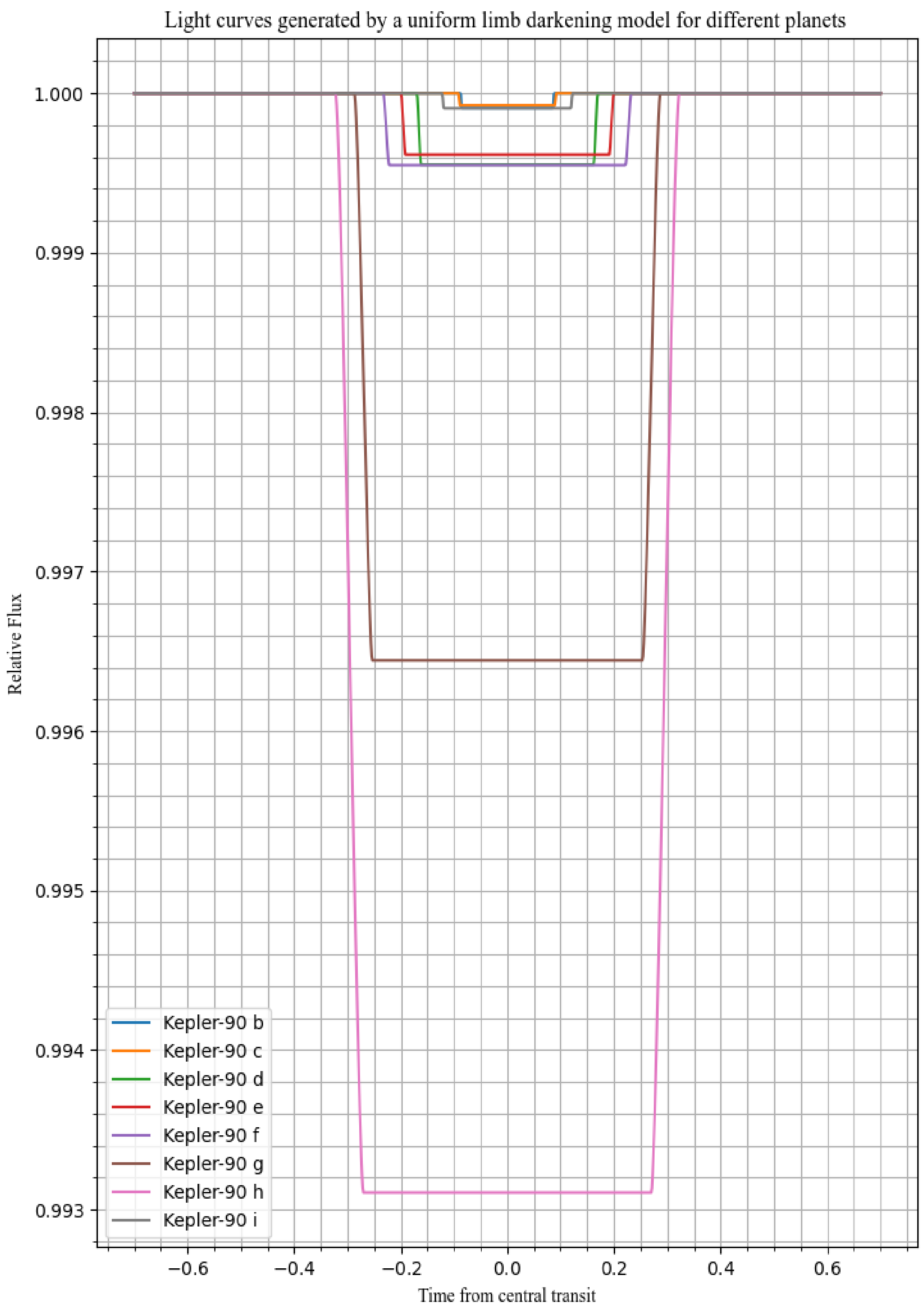

Fig. 1: Light curves of the planets of Kepler-90 using a uniform limb darkening profile

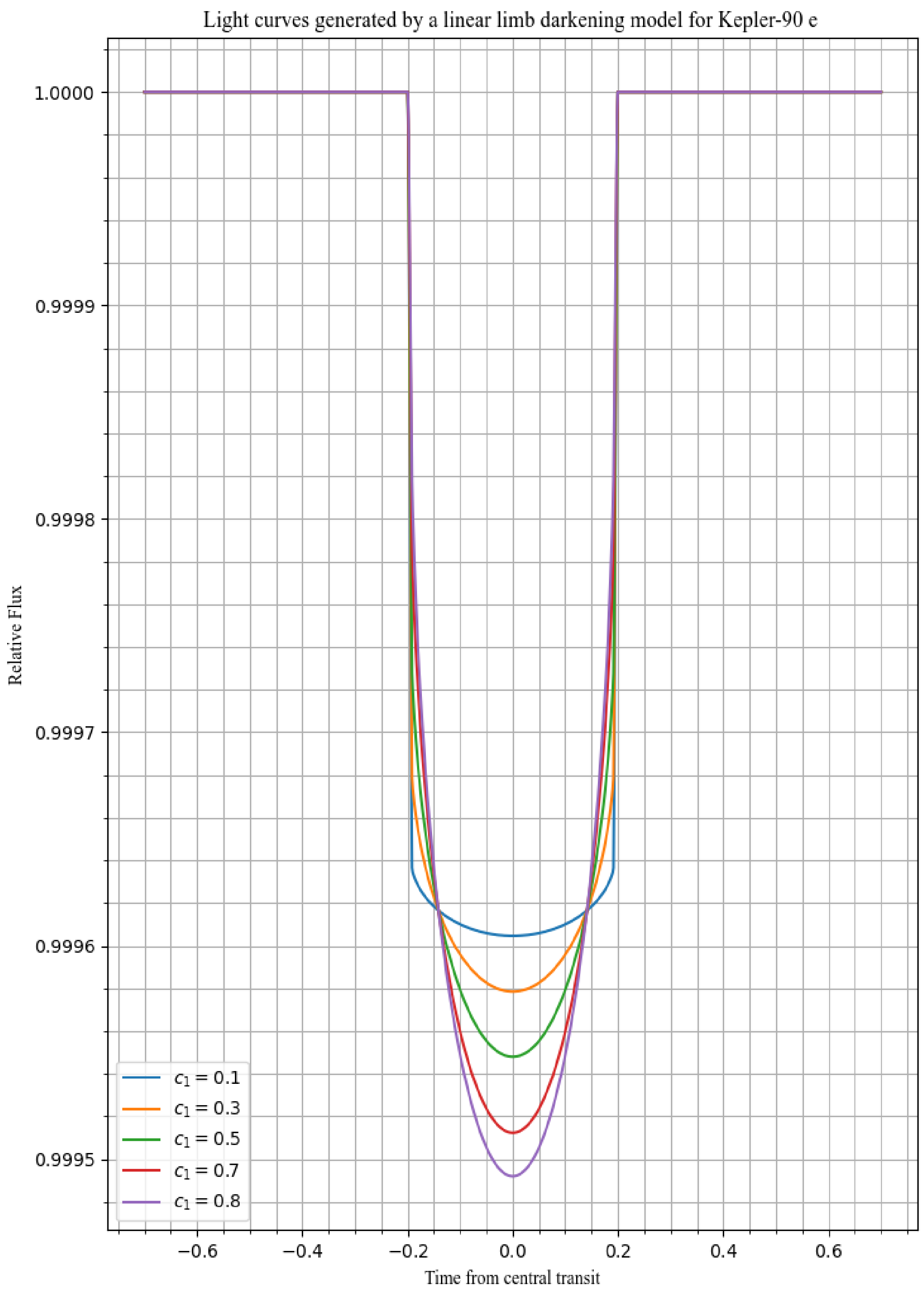

Fig. 2: Light curves of Kepler-90 e using a linear limb darkening profile

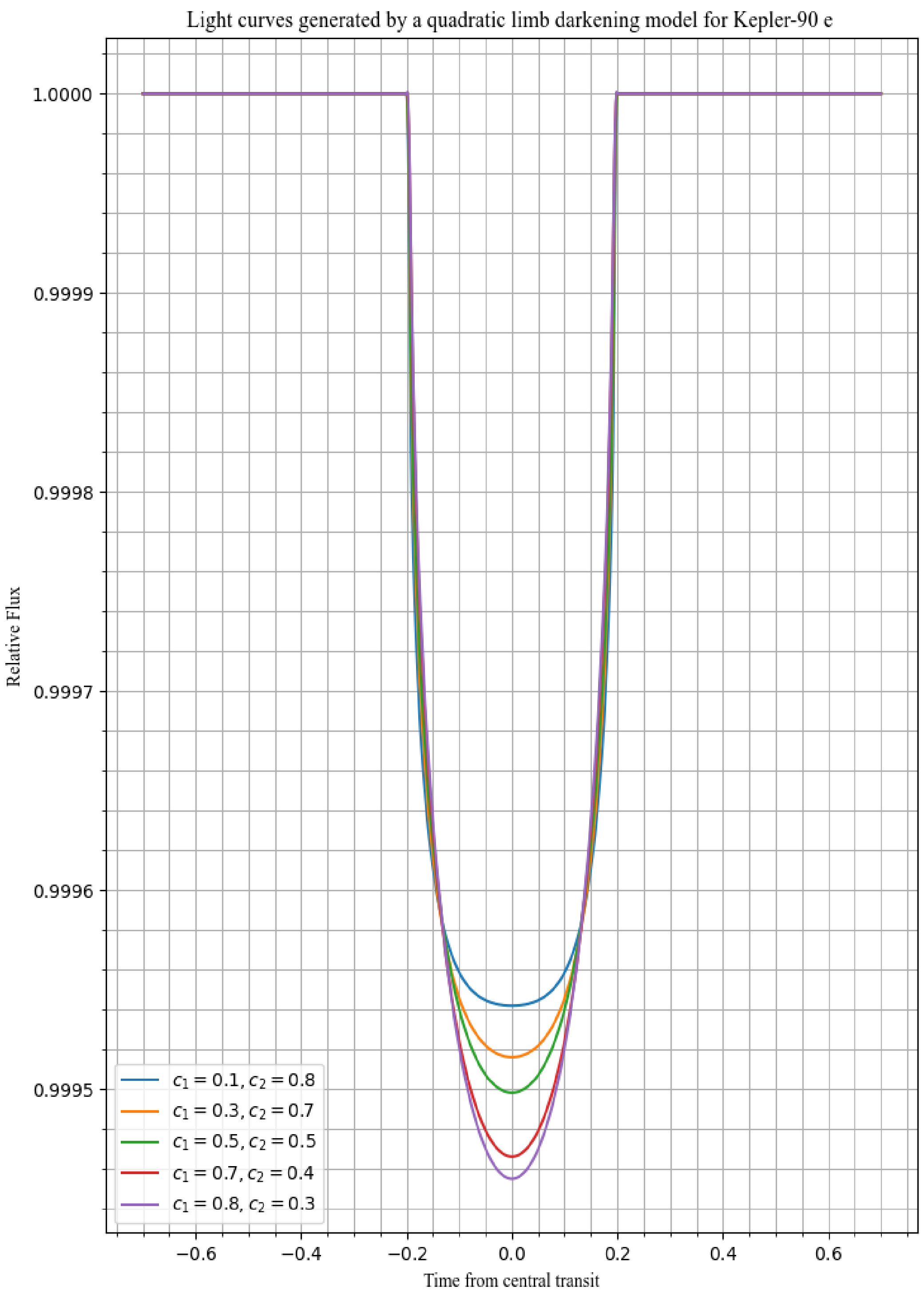

Fig. 3: Light curves of Kepler-90 e using a quadratic limb darkening profile

3.2 Using MCMC to fit a sine curve model to data

The dataset for the X and Y axes was first imported using the pandas library, and a model for the general function of a sine curve,

$$y = a + b\sin(cx + d), \tag{3}$$

was defined. The imported data was then imported in order to obtain a prior range for the y-intercept a, and the amplitude b.

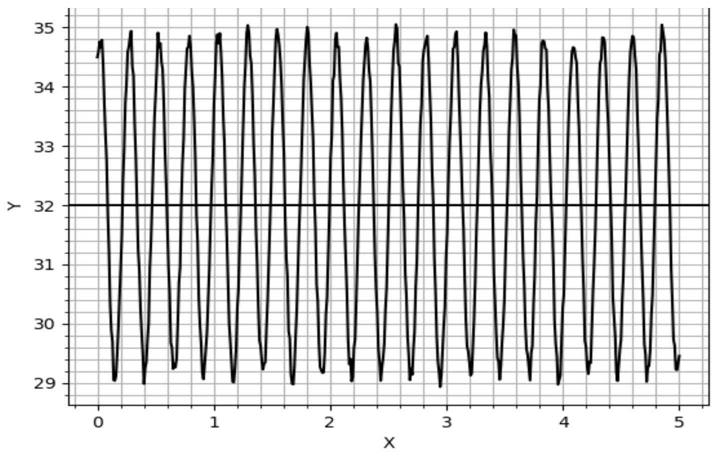

Fig. 4: Plot of the curve data

Using this plot and referencing to the provided values, a function was defined which checks whether the arguments of θ, where $\theta = [a, b, c, d]$, are within their prior ranges. This was done by outputting $-\inf$ if one of the inputted parameters does not satisfy its respective range, and 0 if all parameters are withing their respective ranges. The prior ranges for these parameters were set to be

$$\begin{gathered} 30 \leq a \leq 33.5, \\ 2.5 \leq b \leq 3.5, \\ 23 \leq c \leq 26, \\ 6 \leq d \leq 11. \end{gathered} \tag{4}$$

Another function was defined in order to calculate the likelihood of each parameter using equation 2. This was then referred to another function, called the probability function, which returns the value outputted by the prior function added to the value returned by the likelihood function, such that if all parameters are within their respective prior ranges, the value outputted is the value returned by the likelihood function.

Finally, a function to calculate the step methodology across the parameter space, p_0 was created, followed by the function used to run the MCMC algorithm, using the emcee library, and all the other previously defined functions. This was done using 100 walkers with 3000 iterations.

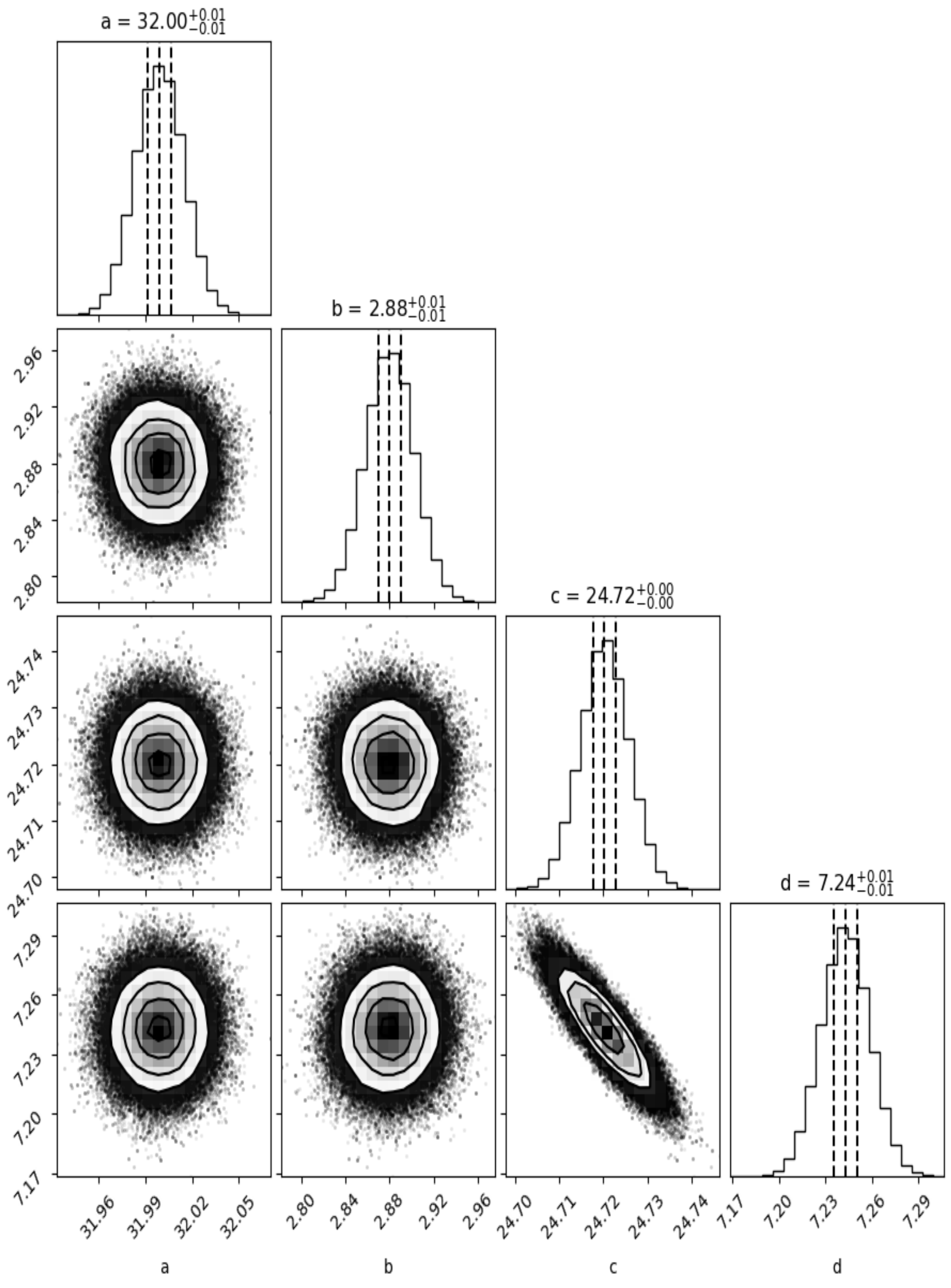

Fig. 5: A corner plot for the parameters of the sine curve

Hence, it was determined that the provided data can be fit using the model in eq. 3, where $a = 32.00 \pm 0.01$, $b = 2.88 \pm 0.01$, $c = 24.72 \pm 0.00$, and $d = 7.24 \pm 0.01$.

3.3 Fitting a transit curve to an observed dataset of Kepler-1b

The data for the relative flux and the respective Heliocentric Julian Date was imported using the pandas library. This was then plotted in order to get an assumption for the time of inferior conjunction, t_0. Moreover, an estimate for the radius of the planet was obtained from the plot and hence,

$$\begin{aligned} \Delta\phi &= \frac{R_p^2}{R_\star} \\ \frac{R_p^2}{R_\star} &\approx 0.12. \end{aligned} \tag{5}$$

As per (Holman et al., 2007), the limb darkening coefficients were taken to be $c_1 = 0.22$, and $c_2 = 0.32$. It was provided that the orbital period of this planet is $P \approx 2.16 \times 10^5$ s, and the mass of the star is $M_\star \approx 1.96 \times 10^{30}$. Hence, using the Kepler's third law, the estimated value of the semi-major axis was found as follows:

$$\begin{aligned} P^2 &= \frac{4\pi^2}{G \cdot (M_\star + M_{planet})} a^3 \\ a &\approx 7.713 \mathrm{R}_\odot, \end{aligned} \tag{6}$$

where the gravitational constant was considered to be $G = 6.67 \times 10^{-11} \mathrm{m}^3\ \mathrm{kg}^-1\ \mathrm{s}^2$. MCMC functions were then defined in a similar manner to those in section 3.2, using the batman parameters as a model as described in section 3.1. Prior ranges were then set with reference to the previously obtained values. The range for the orbital inclination was taken to be $60° \leq 90°$.

```python
def model(theta, x):
    t0, per, rp, a, inc = theta
    params = bat.TransitParams()
    params.w = 0
    params.ecc = 0
    params.limb_dark = 'quadratic'
    params.u = [0.22, 0.32]
    params.t0 = t0
    params.per = per
    params.rp = rp
    params.a = a
    params.inc = inc

    m = batman.TransitModel(params, x)
    rel_flux = m.light_curve(params)
    return rel_flux
```

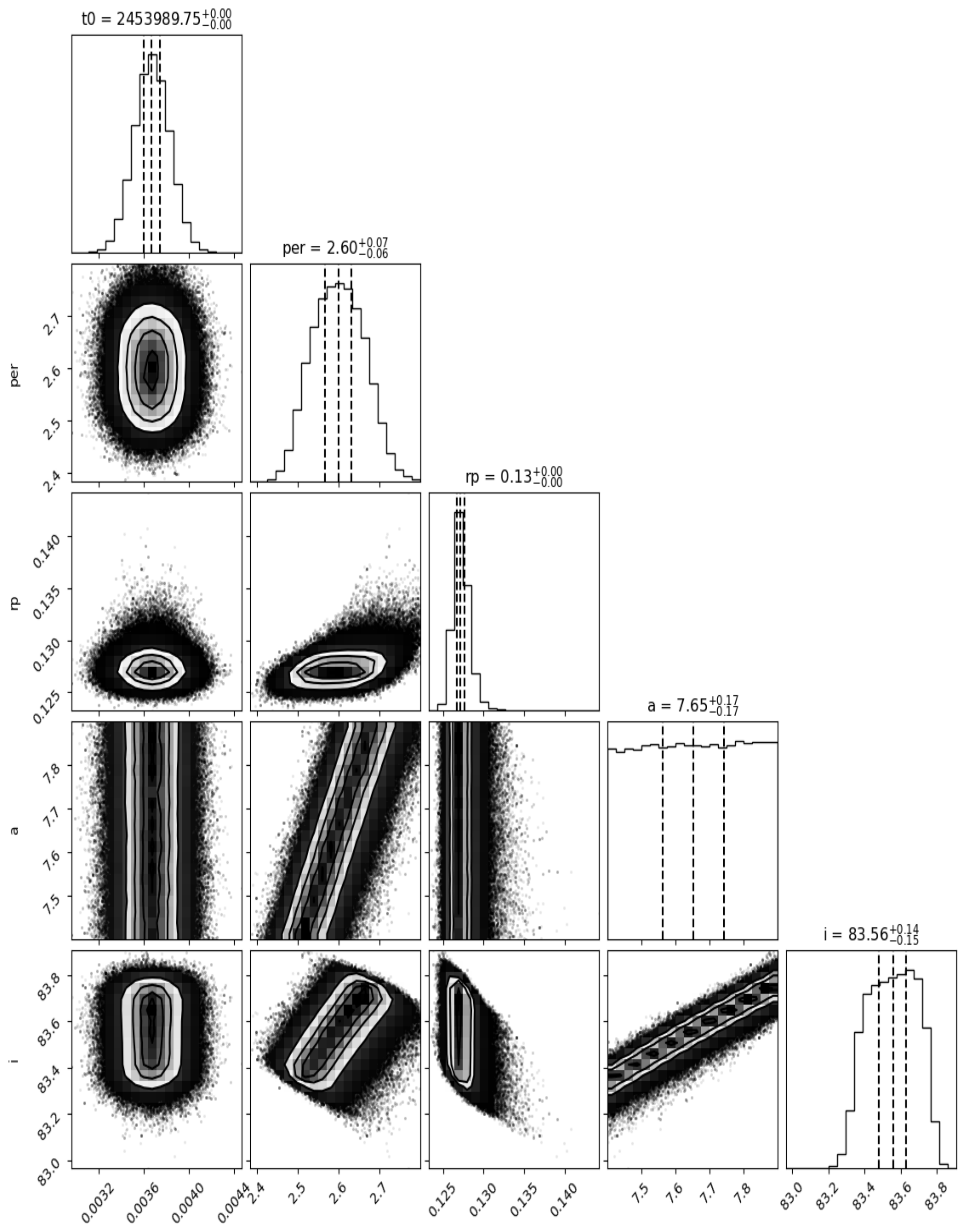

Fig. 6: A corner plot for the parameters of the transit of Kepler-1b

4 DISCUSSION

4.1 Exoplanet transit light curve modelling for Kepler-90 planets

In this section, a light curve was modelled according to different planetary and transit parameters for all the planets in the Kepler-90 system. As can be seen in figure 1, the light curves for the different planets have different shapes, where the dips in relative flux represent the transits of the planet when it is between the observer and the star. Hence it can be said that the planets that are represented to have a wider transit, spend a longer time between the star and the observer. Furthermore, the depth of the dips represent by how much the brightness of the star was observed to decrease, which hints towards the radius of the planet in transit. In the linear and quadratic curves, fig. 2 and fig. 3 respectively, clearly show that that an increase in the values of c_1, increase the depth of the dips in relative flux. Furthermore, the shapes and accuracies of the light curve change significantly according to different values of the limb darkening coefficients, describing better the transit of exoplanets.

4.2 Using MCMC to fit a sine curve model to data

In this section, a model for a sine curve with the general form $y = a + b(sin(cx + d))$, was fitted to provided data for the X and Y axes, using the emcee library to run the MCMC algorithm. A corner plot was then plotted showing the values of the parameters obtained at convergence, as well as the correlation between them. It can be seen by the the correlation plots in 5, that the only correlation present is an inverse relation between c and d, which can be seen by the lack of circularity in the c-d plot.

4.3 Fitting a transit curve to an observed dataset of Kepler-1b

In this section, MCMC and batman were used in order to obtain accurate estimates for different planetary and orbital parameters of Kepler-1b. A corner plot was then plotted showing the values of the different obtained parameters. By referring to fig. 6, it can be seen that the following results were obtained:

$$t_0 \approx 2453989.75 \pm 0.00, \mathrm{inHJD},$$

$$P \approx 2.54 \leq 2.60 \leq 2.67, \mathrm{indays},$$

$$R_p \approx 0.13 \pm 0.00, \mathrm{inR_\star},$$

$$a \approx 7.65 \pm 0.17, \mathrm{inR_\star},$$

$$i \approx 83.42 \leq 83.56 \leq 83.71, in\,^\circ.$$

However, as can be seen in the plot for a, the value didn't converge. This could be due to inaccurate priors, insufficient walkers, or insufficient iterations. Furthermore, the correlation plots show how the different parameters affect each other. While most parameters seem to have little correlation to each other, a positive relation seems to exist between a and P, and a and i. However, due to the lack of convergence for a, this could prove to be inaccurate. A 1σ posterior plot was then plotted to show the deviation of the results.

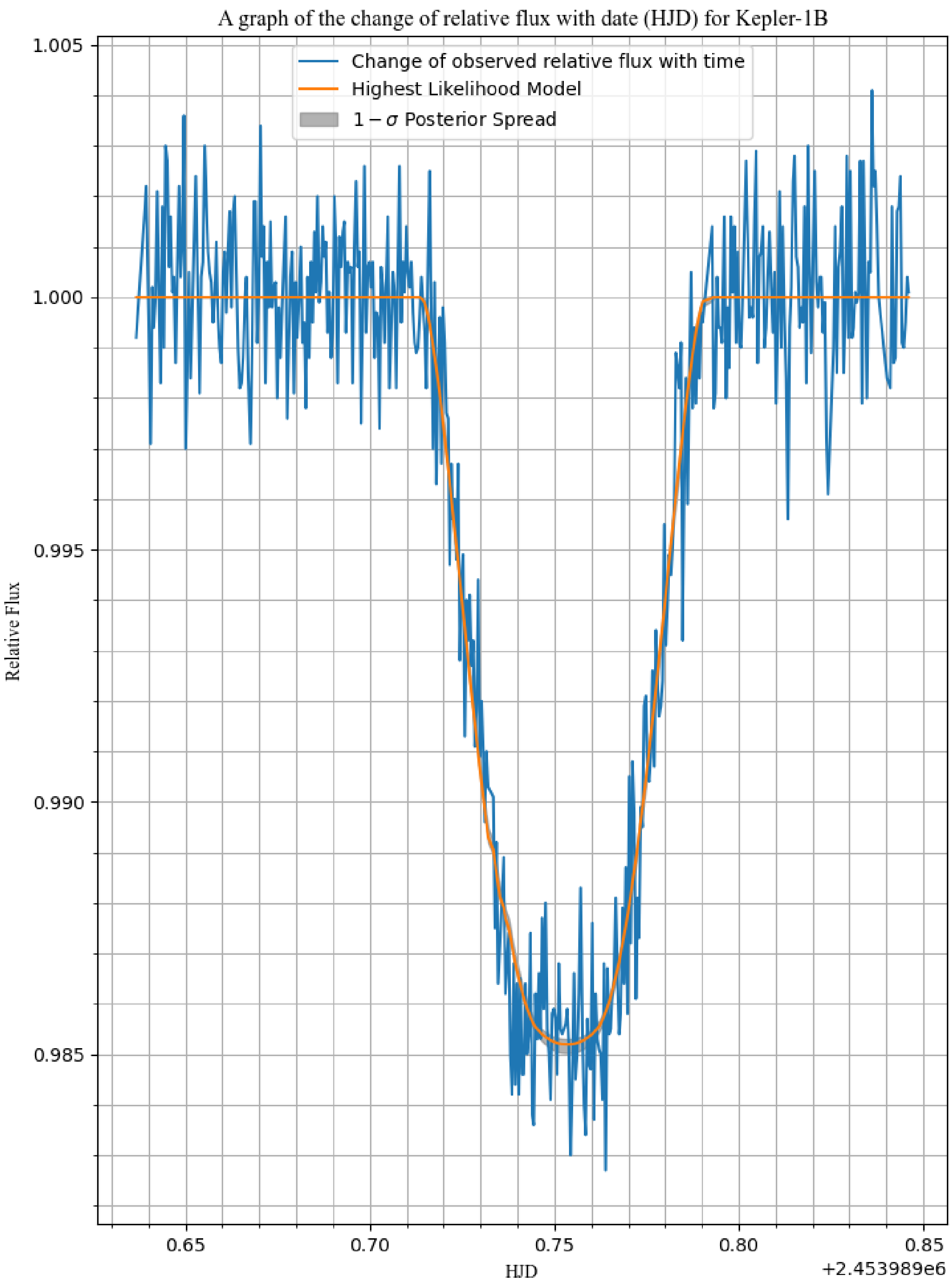

Fig. 7: A 1σ spread with the highest likelihood model

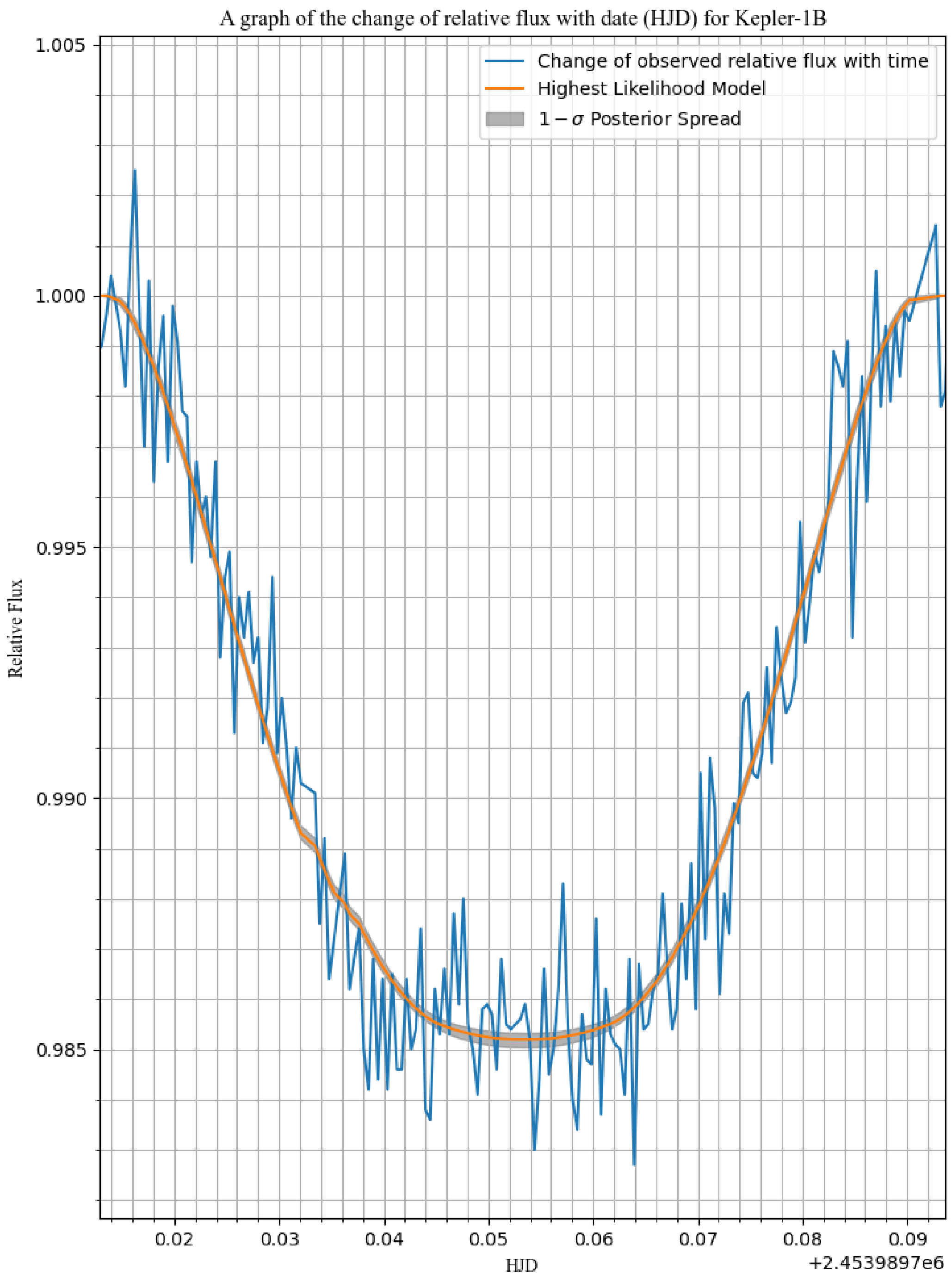

Fig. 8: A partial view of the 1σ spread with the highest likelihood model

5 CONCLUSION

In this experiment, an overview of the batman package was shown by calculating light curves for the planets of the Kepler-90 system and how different parameters affect their light curves. Furthermore a brief overview of MCMC was shown by showing how a model can be fitted to some observed data. Lastly, these two methods were combined in order to calculate estimates of the planetary and orbital parameters of Kepler-1b by fitting a batman model to some observed relative flux data. This is useful as more accurate observed data can be fitted to these models in order to show better properties of far away systems.

REFERENCES

Espinoza, N & Jordán, A (2015) Limb darkening and exoplanets: testing stellar model atmospheres and identifying biases in transit parameters. *Monthly Notices of the Royal Astronomical Society*, *450*(2) 1879–1899. https://doi.org/10.1093/mnras/stv744

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) 1185–1189. https://doi.org/10.1086/519077

Kreidberg, L (2015) batman: BAsic Transit Model cAlculatioN in Python., *127*(957) 1161. https://doi.org/10.1086/683602

Stassun, K G Oelkers, R J Paegert, M Torres, G Pepper, J De Lee, N Collins, K Latham, D W Muirhead, P S Chittidi, J Rojas-Ayala, B Fleming, S W Rose, M E Tenenbaum, P Ting, E B Kane, S R Barclay, T Bean, J L Brassuer, C E . . . Winn, J N (2019) The Revised TESS Input Catalog and Candidate Target List., *158*(4) Article 138, 138. https://doi.org/10.3847/1538-3881/ab3467

APPENDIX

1 CODE LISTING

1.1 Light curve modelling using batman for Kepler-90

```python
import pandas as pd
import numpy as np
import batman as bat
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker


def planet_param(arr):
    planet_dic = {}
    names = arr.index
    for index, param in enumerate(arr):
        planet_dic[names[index]] = param
    return planet_dic


def transit(planet, profile, c1=0.0, c2=0.0):
    parameters = bat.TransitParams()
    parameters.t0 = 0
    parameters.w = 0
    if profile == 'u':
        parameters.limb_dark = 'uniform'
        parameters.u = []
    elif profile == 'l':
        parameters.limb_dark = 'linear'
        parameters.u = [c1]
    elif profile == 'q':
        parameters.limb_dark = 'quadratic'
        parameters.u = [c1, c2]
    parameters.rp = planet['Rp / Rs']
    parameters.a = planet['a / R']
    parameters.inc = planet['Orbital Inclination (deg)']
    parameters.ecc = planet['Eccentricity']
    parameters.per = planet['Orbital Period (yr)'] * 365
    return parameters


```

```python
# Importing data

df = pd.read_csv('kepler_90_pl.csv')
df.set_index('Planet', inplace=True)
k_90_rad = 1.25 * 695700

# Modifying data and units

df['Rp / Rs'] = df['Planetary Radius (Earth Radius)'] * 6378.14 / k_90_rad
df['a / R'] = df['Semimajor Axis (AU)'] * 1.496e8 / k_90_rad

# Initializing arrays
time_arr = np.linspace(-0.7, 0.7, 1000)
planet_arr = np.array(df.index.values.tolist())
c1 = [0.1, 0.3, 0.5, 0.7, 0.8]
c2 = [0.8, 0.7, 0.5, 0.4, 0.3]

# Generating uniform models for all planets

fig, ax = plt.subplots(1)
ax.figure.set_size_inches(8.27, 11.69)
for i in planet_arr:
    m = bat.TransitModel(transit(df.loc[i], 'u'), time_arr)
    rel_flux = m.light_curve(transit(df.loc[i], 'u'))
    plt.plot(time_arr, rel_flux, label=i)

csfont = {'fontname': 'Times New Roman'}
plt.ylabel(r"Relative Flux", **csfont)
plt.xlabel(r"Time from central transit", **csfont)
plt.title(r"Light curves generated by a uniform limb darkening model for different planets "
    , **csfont)
plt.legend()
plt.minorticks_on()
plt.grid(which='both')
formatter = mticker.ScalarFormatter(useMathText=True)
ax.xaxis.set_major_formatter(formatter)
plt.savefig('uniform.png')
plt.show()

#  Generating a linear model

```

```python
fig, ax = plt.subplots(1)
ax.figure.set_size_inches(8.27, 11.69)
for i in c1:
    m = bat.TransitModel(transit(df.loc['Kepler-90 e'], 'l', i), time_arr)
    rel_flux = m.light_curve(transit(df.loc['Kepler-90 e'], 'l', i))
    plt.plot(time_arr, rel_flux, label=r"$c_1 = {}$".format(i))

csfont = {'fontname': 'Times New Roman'}
plt.ylabel(r"Relative Flux", **csfont)
plt.xlabel(r"Time from central transit", **csfont)
plt.title(r"Light curves generated by a linear limb darkening model for Kepler-90 e ", **
    csfont)
plt.legend()
plt.minorticks_on()
plt.grid(which='both')
formatter = mticker.ScalarFormatter(useMathText=True)
ax.xaxis.set_major_formatter(formatter)
ax.get_yaxis().get_major_formatter().set_useOffset(False)
plt.savefig('linear.png')
plt.show()

# Generating a quadratic model

fig, ax = plt.subplots(1)
ax.figure.set_size_inches(8.27, 11.69)
for i in range(len(c1)):
    m = bat.TransitModel(transit(df.loc['Kepler-90 e'], 'q', c1[i], c2[i]), time_arr)
    rel_flux = m.light_curve(transit(df.loc['Kepler-90 e'], 'q', c1[i], c2[i]))
    plt.plot(time_arr, rel_flux, label=r"$c_1 = {}, c_2 = {}$".format(c1[i], c2[i]))

csfont = {'fontname': 'Times New Roman'}
plt.ylabel(r"Relative Flux", **csfont)
plt.xlabel(r"Time from central transit", **csfont)
plt.title(r"Light curves generated by a quadratic limb darkening model for Kepler-90 e ", **
    csfont)
plt.legend()
plt.minorticks_on()
plt.grid(which='both')
formatter = mticker.ScalarFormatter(useMathText=True)
ax.xaxis.set_major_formatter(formatter)
ax.get_yaxis().get_major_formatter().set_useOffset(False)
```

```python
plt.savefig('quadratic.png')
plt.show()

print("Debug")
```

1.2 Using MCMC to fit a sine model to data

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import emcee
import corner
import multiprocessing


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


def ln_like(theta, x, y, y_err):
    ln_like_val = -0.5 * np.sum(np.square((y - model(theta, x)) / y_err))
    return ln_like_val


def ln_prior(theta):
    a, b, c, d = theta
    if 30 <= a <= 33.5 and 2.5 <= b <= 3.5 and 23 <= c <= 26 and 6 <= d <= 11:
        return 0
    else:
        return -np.inf


def ln_prob(theta, x, y, y_err):
    lp = ln_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + ln_like(theta, x, y, y_err)


def _p0(initial_val, n_dim_val, n_walkers_val):
    return [np.array(initial_val) + [1e-4, 75e-3, 1e-2, 1e-2] * np.random.randn(n_dim_val)
     for i in
            range(n_walkers_val)]


def mcmc(p0, n_walkers, n_iter, n_dim, ln_prob, data_tup, pool):
    sampler_val = emcee.EnsembleSampler(n_walkers, n_dim, ln_prob, args=data_tup, pool=pool)
    print("Running burn-in...")
    p0, _, _ = sampler_val.run_mcmc(p0, 800, progress=True)
    sampler_val.reset()

    print("Running production...")
    posteriors_val, prob_val, state_val = sampler_val.run_mcmc(p0, n_iter, progress=True)
    print("Production Done")
    return sampler_val, posteriors_val, prob_val, state_val


df = pd.read_csv('mcmc_intro_group_5.csv')
data = (df['x'], df['y'], df['y_err'])

# Value Checking from plot

plt.plot(df['x'], df['y'])
plt.axhline(y=32)
plt.minorticks_on()
plt.grid(which='both')
plt.xlabel("X")
plt.ylabel("Y")
plt.savefig("Sine")
plt.show()

n_walkers = 100
n_iter = 3000

initial = np.array([32.5, 3, 24, 8])
n_dim = len(initial)

p0 = _p0(initial, n_dim, n_walkers)

if __name__ == '__main__':
    with multiprocessing.Pool(4) as pool:
        sampler, posteriors, prob, state = mcmc(p0, n_walkers, n_iter, n_dim, ln_prob, data,
     pool)
        samples = sampler.flatchain

        # Corner Plot
        labels = ['a', 'b', 'c', 'd']
        print("Done")
        fig, ax = plt.subplots(1)
        ax.figure.set_size_inches(8.27, 11.69)
        fig = corner.corner(samples, show_titles=True, labels=labels, plot_datapoints=True,
    quantiles=[0.32, 0.5, 0.68])
        plt.savefig("corner_sine.png")


print("Debug")
```

1.3 Using MCMC and batman to fit a light curve model to data

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


def model(theta, x):
    t0, per, rp, a, inc = theta
    params = bat.TransitParams()
    params.w = 0
    params.ecc = 0
    params.limb_dark = 'quadratic'
    params.u = [0.22, 0.32]
    params.t0 = t0
    params.per = per
    params.rp = rp
    params.a = a
    params.inc = inc

    m = batman.TransitModel(params, x)
    rel_flux = m.light_curve(params)
    return rel_flux


def ln_like(theta, x, y, y_err):
    ln_like_val = -0.5 * np.sum(np.square((y - model(theta, x)) / y_err))
    return ln_like_val


def ln_prior(theta):
    t0, per, rp, a, inc = theta
    if (0.71 + 2.453989e6 <= t0 <= 0.78 + 2.453989e6) and (2.1 <= per <= 2.8) and (0.09 <=
    rp <= 0.15) and (
            7.4 <= a <= 7.9) and 60 <= inc <= 90:
        return 0
    else:
        return -np.inf


# 0.785, 0.72 = 0.065
# p squared = a cubed
# rp/rs = change in flux root
def ln_prob(theta, x, y, y_err):
    lp = ln_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + ln_like(theta, x, y, y_err)


def _p0(initial_val, n_dim_val, n_walkers_val):
    return [np.array(initial_val) + [1e-3, 1e-4, 1e-3, 1e-4, 0.1] * np.random.randn(
    n_dim_val) for i in
            range(n_walkers_val)]


def mcmc(p0, n_walkers, n_iter, n_dim, ln_prob, data_tup, pool):
    sampler_val = emcee.EnsembleSampler(n_walkers, n_dim, ln_prob, args=data_tup, pool=pool)
    print("Running burn-in...")
    p0, _, _ = sampler_val.run_mcmc(p0, 800, progress=True)
    sampler_val.reset()

    print("Running production...")
    posteriors_val, prob_val, state_val = sampler_val.run_mcmc(p0, n_iter, progress=True)
    print("Production Done")
    return sampler_val, posteriors_val, prob_val, state_val


def posteriors_func(samples_post, chain, x):
    models = []
    drw = np.floor(np.random.uniform(0, len(chain), size=samples_post)).astype(int)
    th_s = chain[drw]
    for i in th_s:
        mdl_post = model(i, x)
        models.append(mdl_post)
    spread = np.std(models, axis=0)
    med_mod = np.median(models, axis=0)
    return med_mod, spread


if __name__ == '__main__':
    with multiprocessing.Pool(4) as pool:
        df = pd.read_csv("kepler_lc_group_4.csv")

        time_arr = np.linspace(np.min(df['HJD'].to_numpy()), np.max(df['HJD'].to_numpy()),
    433)

        data = (time_arr, df['Rel_Flux'], df['Flux_err'])

        n_walkers = 250
        n_iter = 5000

        initial = np.array([0.75 + 2.453989e6, 2.5, 0.12, 7.71, 90])
        n_dim = len(initial)

        p0 = _p0(initial, n_dim, n_walkers)

        sampler, posteriors, prob, state = mcmc(p0, n_walkers, n_iter, n_dim, ln_prob, data,
     pool)
        samples = sampler.flatchain
        print("Done")

        # Corner Plot
```

```python
    labels = ['t0', 'per', 'rp', 'a', 'i']

    fig, ax = plt.subplots(1)
    ax.figure.set_size_inches(8.27, 11.69)
    corner.corner(samples, show_titles=True, labels=labels, plot_datapoints=True, quantiles=[0.32, 0.5, 0.68])
    plt.savefig("cornerbatman.png")
    plt.show()

    # Plotting 1-sigma posterior

    th_max = samples[np.argmax(sampler.flatlnprobability)]
    best_fit = model(th_max, time_arr)
    med_mod_val, spread_val = posteriors_func(600, samples, time_arr)

    fig, ax = plt.subplots(1)
    ax.figure.set_size_inches(8.27, 11.69)
    csfont = {'fontname': 'Times New Roman'}
    plt.plot(df['HJD'], df['Rel_Flux'], label='Change of observed relative flux with time')
    plt.plot(df['HJD'], best_fit, label='Highest Likelihood Model')
    plt.fill_between(df['HJD'], med_mod_val-spread_val, med_mod_val+spread_val, color='grey', alpha=0.6,
                     label=r'$1-\sigma$ Posterior Spread')
    plt.xlabel("HJD", **csfont)
    plt.ylabel("Relative Flux", **csfont)
    plt.title("A graph of the change of relative flux with date (HJD) for Kepler-1B", **csfont)
    plt.legend()
    plt.minorticks_on()
    plt.grid(which='both')
    plt.savefig("PosteriorPlot_full")
    plt.show()
    print('Theta:', th_max)
```