# AST326 Lab #5: Determining Physical Properties of Exoplanet GJ 1214b Through Transit Method

Aaryan Thusoo

March 17th 2024

## ABSTRACT

This study delves into the determination of physical properties of exoplanets detected through the transit method. Focusing on GJ 1214b, we employ observational data and light curve modeling to analyze its orbit, radius, and density. By applying flux normalization and limb darkening modeling techniques, our investigation reveals significant insights into this exoplanetary system. We ascertain that GJ 1214b induces a flux drop of $1.9\% \pm 0.2\%$, corresponding to a radius of $2.617M_\oplus \pm 0.109M_\oplus$. The planet orbits with a semi-major axis of $3.06R_\odot \pm 0.14R_\odot$ and exhibits an inclination of $89.95°\pm0.02°$, indicating a nearly horizontal orbit relative to our frame of reference. Additionally, our model provides limb darkening coefficients of $0.65 \pm 0.10$ and $0.32 \pm 0.04$. Utilizing these findings, we calculate a density of $2.48g/cm^3 \pm 0.23g/cm^3$ and provide interpretations of its implications. This comprehensive analysis sheds light on the physical characteristics of GJ 1214b and contributes to our broader understanding of exoplanetary systems.

## 1. Introduction

In the vast expanse of interstellar space, the possibility of extraterrestrial life captivates the imaginations of many. Scientists dedicate years of research to the pursuit of identifying celestial bodies harboring the conditions conducive to life's existence. Among the arsenal of techniques employed for this study, transit detections stand out as a key method with currently the most exoplanets found. In this report, we implement transit detection to unveil GJ 1214b, an exoplanet nestled within the Ophiuchus constellation (Seager et al. 2007). Orbiting the red dwarf GJ 1214, this celestial body offers a glimpse into the information that simple light curves hold.

Much like a solar eclipse where the moon obscures the sun, causing daylight to dim, the transit method aids in the discovery of potential exoplanets by monitoring the decrease in light emitted by a target star. An exoplanet, akin to Earth, orbits its host star within a consistent time frame known as the orbital period. As observers, we can use this regularity as it means that during periodic cycles, the exoplanet passes between us and the star, resulting in a measurable reduction in the star's brightness. By meticulously observing these fluctuations over typically day-long intervals, discernible patterns of slight flux drops emerge, indicating the presence of a transiting exoplanet. A crucial relationship arises when comparing the change in flux, denoted $\Delta F$, to the standard measured flux. This relationship, depicted in Equation 1, facilitates the determination of the exoplanet's radius, beginning with the well-documented radius of its host star.

$$\frac{\Delta F}{F} = \left(\frac{R_{planet}}{R_{star}}\right)^2 \tag{1}$$

To accurately gauge the flux drop, astronomers construct a light curve of the star by amalgamating numerous observations. Establishing a reliable light curve for GJ 1214 requires the normalization of collected data, as external influences like moonlight and light pollution subtly skew the overall flux recorded in each observation. Consequently, multiple reference stars are observed, and their light curves are utilized to formulate a model depicting the fluctuation in flux throughout the data collection process. By dividing this model out of each individual light curve, we normalize the data effectively isolating the flux discrepancies caused solely by the transit of the exoplanet.

The flux calculations utilize aperture photometry to ensure precise incoming flux data with manageable uncertainties. However, this process encounters slight modifications due to pixel movements between images caused by the Earth's rotation throughout the data collection period. To address this, a movement vector is established by tracking the shifts of a manually selected star, enabling the tracking of any point in the sky across the image series. Subsequently, numerous reference stars are enlisted, and employing the tracking method, an aperture and annulus can be delineated around each star in every image.

Once a light curve is compiled, a model can be derived to ascertain the physical attributes of the observed entity. Notably, limb darkening, which accommodates the variation in intensity of light emitted by the star's edges compared to its center due to differing temperatures, warrants particular attention. The model fitting process necessitates the specification of initial parameters, including the planetary radius, semi-major axis of the orbit, inclination angle, and two limb darkening coefficients. The inclination angle references our perspective of the system, acknowledging that the planet's orbit may appear differently based on our position in space. An angle of $0°$ signifies an orbit where the star is never eclipsed, while $90°$ indicates a perfectly horizontal transit.

To initiate the radius estimation process, we employ Equation 1 and construct a basic top hat model to approximate the flux variation, hence inferring the radius of GJ 1214b. Subsequently, the semi-major axis and inclination values are extracted from the Encyclopedia of Exoplanetary Systems (Cloutier et al. 2021), a comprehensive database containing data on myriad exoplanets. Meanwhile, the limb darkening coefficients are sourced from the Vizier catalog, a repository brimming with celestial object information (F. 2000). However, since the catalog lacks precise data for our target star, we resort to the nearest available values, albeit this may marginally impact the final uncertainties.

With these initial parameters, we proceed to fit a limb darkening curve that accommodates temperature variations across the stellar disc, employing a quadratic model described by Equation 2, where $\mu = \sqrt{\frac{a^2-r^2}{a^2}}$. Here, we ascertain the planet-to-star radius ratio and normalize the separation to GJ 1214 using Equation 3, where $\omega$ signifies the angular velocity of the planet and $i$ represents the inclination angle between the orbital plane and the reference plane. The ratio $\frac{a}{R_s}$ denotes the semi-major axis relative to the star's radius, both expressed in solar radii ($R_\odot$).

$$\frac{I_\mu}{I_{\mu=1}} = 1 - c_1(1-\mu) - c_2(1-\mu)^2 \tag{2}$$

$$z(t) = \frac{a}{R_s}\sqrt{\left[(sin(\omega t))^2 + (cos(i)cos(\omega t))^2\right]} \tag{3}$$

To fit the plot, we determine the coefficients $c_1$ and $c_2$ from the Vizier catalog (F. 2000). Utilizing these parameters, we derive the inclination, planet radius, and limb darkening coefficients through the following equations, enabling us to ascertain various physical characteristics of GJ 1214b.

Collaborating with Maha Macknojia and Umit Uzunboy, this report delves into the determination of physical parameters of GJ 1214b by scrutinizing variations in its host star's light curve. We provide a concise overview of the collected data and its applications in 2, followed by a demonstration of data cleaning and reduction techniques in 3. Subsequently, we delve into the analysis of the collected data in 4, with ensuing discussion of results in 5.

## 2. Data and Observations

Data acquisition was overseen by Ernst de Mooij, utilizing the William Herschel Telescope (WHT) equipped with an Auxiliary-port CAMera (ACAM). Situated on La Palma in the Canary Islands, Spain, the WHT has a 4.20-meter primary mirror, enabling optical and infrared observations. The ACAM captures circular images with an 8.2 arc-minute diameter, each pixel corresponding to approximately 0.25 arc seconds. The resulting images have dimensions of 2071 by 2148 pixels.

The collected data focuses on the Sloan G filter, designed to capture light in the blue-green spectrum centered at 480nm, with a width range of 140nm (Fukugita et al. 1996). The dataset comprises 47 flat files and 26 bias files, pivotal for calibrating the CCD camera and ensuring the fidelity of the collected data. Utilizing these files, master flats and bias frames are generated to facilitate the flat-fielding process, correcting for pixel sensitivity variations and eliminating systematic errors. The observations span a duration of approximately 5 hours, comprising a total of 364 individual observations. This number of observations will help to show the changes in flux very accurately over the span of one transit.

## 3. Data Reduction

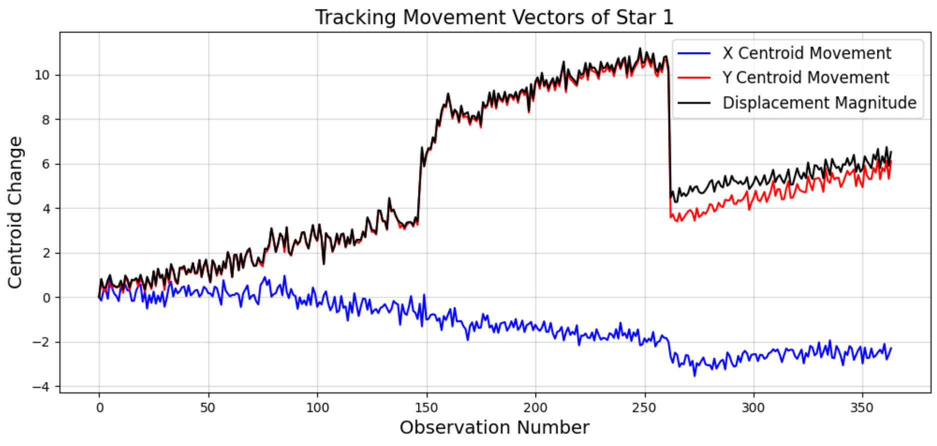
Before delving into data analysis, it's imperative to clean the data to mitigate noise and calibrate the images. Initially, the flat files undergo normalization by determining the median pixel intensity value and dividing all pixels by this value. Consequently, each file's median value becomes 1, and a subsequent median is calculated across the normalized flat arrays. A similar process is applied to the bias files, yielding master bias and master flat files utilized for flat fielding the raw observation data. By subtracting the bias data and dividing the difference by the flats, each file's dataset is calibrated. Additionally, a normalized log scale is applied to the figures to enhance the visuals of star and background intensities.

Considering the Earth's rotation, the telescope's direction gradually deviates over time, despite correction efforts by the data collection team. This shift results in star movements of up to 10 pixels between images, necessitating compensation. To address this, a single star's position change is tracked, and these movement vectors are subsequently applied to other stars. Furthermore, slight variations in overall brightness are observed in the images over time, likely due to changing conditions or new sources of light affecting data collection. To counteract this trend, all light curves are normalized against a master light curve, effectively accommodating these fluctuations.

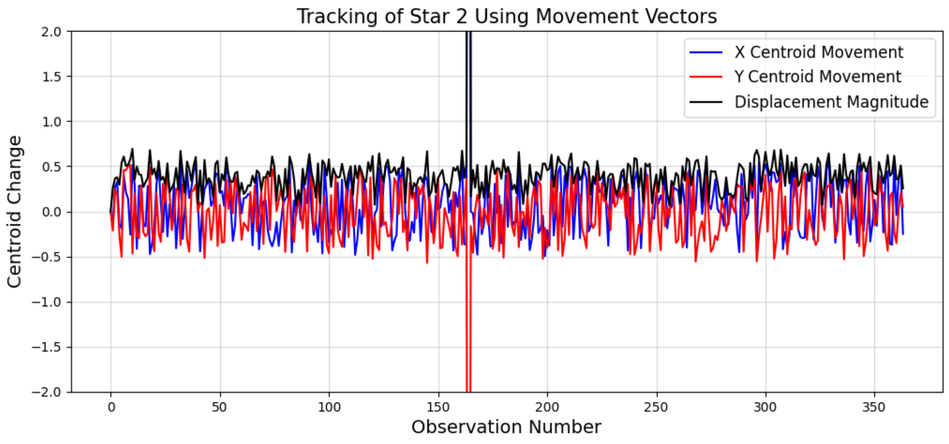
### 3.1. Finding and Tracking Stars

To ensure accurate tracking despite the Earth's rotation-induced shifts, we select a reference star and employ a 2D Gaussian centroid function to pinpoint its precise center. Subsequently, as each image is processed, the centroid of the chosen star is recalculated, and the resulting movements are recorded as movement vectors, illustrating the general image shifts. Figure 1 illustrates this process, showcasing the movements of two stars; 1a demonstrates the overall star movements, while 1b exemplifies the efficacy of the tracking method across different stars.

Given that the background intensity of the images doesn't hover near zero but rather averages around 70, it's crucial for the centroid algorithm to account for this background to prevent significant calculation discrepancies. Failure to correct for this background intensity could skew the calculations considerably. In preliminary attempts using simpler centroiding methods, this background was often overlooked, resulting in poor tracking accuracy and substantial noise accumulation due to an inability to properly center stars for calculations. Having verified the effectiveness of the 2D Gaussian tracking method through successive star tracking, the subsequent step involves identifying additional suitable stars to gather light curves from.



(a) Tracking of Star 1



(b) Tracking of Star 2

Fig. 1.—: Movement vectors are calculated over one star and used to follow every other reference star. (a) Shows the movement of a selected star at pixel coordinate (1130, 470). (b) Shows the movement of a second star at (1277, 229) after applying the calculated movement vectors from Star 1.

To identify stars for analysis, we established a minimum intensity threshold of [insert value] and selected clusters surpassing this threshold. By pinpointing the center pixel of each cluster, we compiled a list of 127 star positions from the initial file. However, some stars exhibited over saturation, hindering accurate calculations. Thus, we imposed an upper intensity limit of 50000 to exclude these from consideration. Furthermore, not every identified star proved suitable for subsequent steps, as stars situated too closely together or near the image edges could yield inaccurate flux approximations and high uncertainties. To mitigate this, we applied a 35-pixel box around each star, examining its surroundings for discrepancies. Stars failing this test were excluded, resulting in a final list comprising 99 stars with satisfactory initial conditions.

### 3.2. Light Curves

Leveraging the tracking vectors, we tracked each star's movement and devised aperture and annulus masks to measure their fluxes. The equations utilized in these calculations are detailed below. To maximize the signal-to-noise ratio, we experimented with various aperture radii ranging from 4 to 7, determining the optimal size to minimize unwanted noise. Signal-to-noise ratio was computed by dividing the flux by its corresponding uncertainty for each chosen radius.

$$I_{bg} = \sum_{annulus} \frac{I(x,y)}{N_{annulus}} \tag{4}$$

$$F_{star} = \sum_{aperture} (I(x,y) - I_{bg}) \tag{5}$$

$$\Delta F = \sqrt{\frac{F}{g} + N_{aperture}\left(1 + \frac{\pi}{2}\frac{N_{aperture}}{N_{annulus}}\right)\sigma_{bg}^2} \tag{6}$$

In our data processing phase, we meticulously address outliers within individual starlight curves, a common occurrence due to imperfect tracking or sub optimal aperture radii. Employing a stringent filtering criterion, we discard points outside the range of 0.97 to 1.02 or with uncertainties surpassing 0.07, ensuring data integrity. Furthermore, to bolster our outlier detection, we calculate the z-scores for each data point, considering any exceeding an absolute value of 3.5 as outliers warranting removal. Following outlier removal, we construct a median model from the refined set of 96 light curves, subsequently evaluating each curve against this model to ascertain standard deviations. Through iterative selection, we identify 13 stars with the least variation, culminating in the creation of a final median model, showcased in Figure 2. This model serves as a robust baseline for normalization, facilitating the removal of most flux variations induced by external factors upon division with individual light curves.

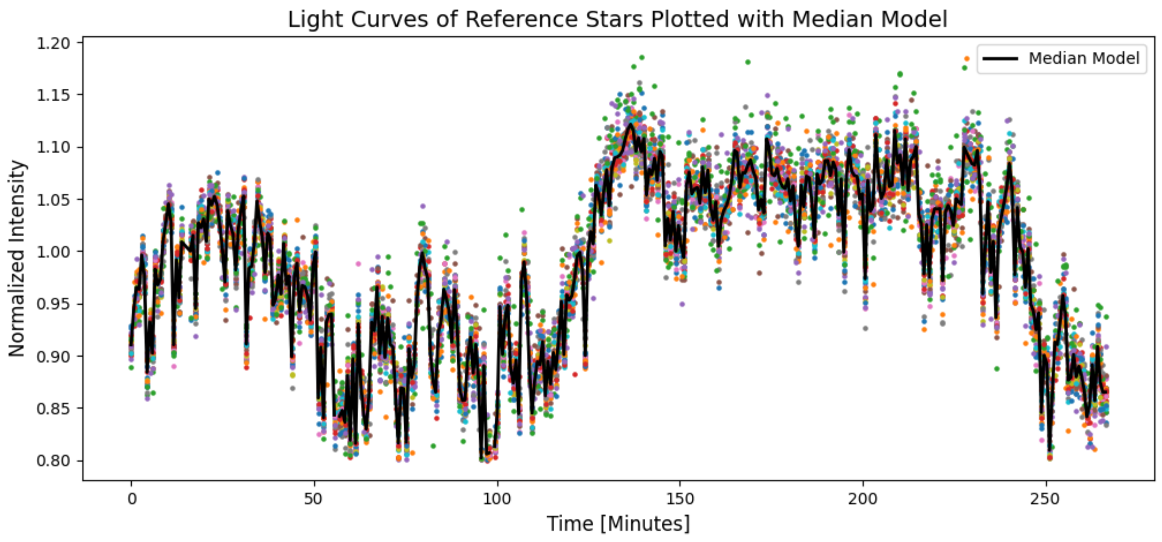


Fig. 2.—: The finalized median light curve is found using 13 reference stars which are represented as coloured scatter points. The black line overlaid on top is the median model curve that is used for normalizing the transit light curve of GJ 1214 in later steps

### 3.3. Detecting GJ 1214b Transit

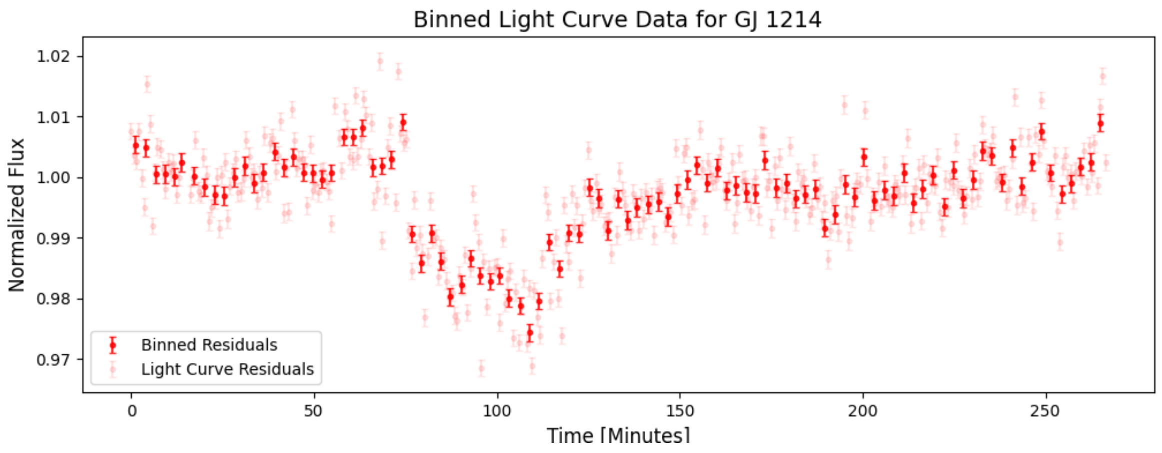


Fig. 3.—: Normalized light curve of GJ 1214 with the binned data set to show cleaner shape in model

Utilizing the ALADIN Sky Atlas, we superimposed the observed data onto sky images centered on the target star GJ 1214, facilitating a cross-correlation analysis to estimate the star's pixel location. Employing the same techniques as with the reference stars, we applied movement vectors and generated a light curve for GJ 1214. Normalizing the curve by the median model yielded the final light curve of the star, illustrated in Figure 3. Slight fluctuations around 1 signify minor changes likely stemming from the flux calculations, while a distinct dip in the star's flux by approximately 0.02 strongly suggests the presence of an exoplanet transiting in front of the star. Binning the light curve aids in visualizing its overall shape.

To determine properties of the exoplanet, two fitting models are employed in this report. Firstly, a simple top hat fit is utilized to provide a preliminary estimate of the planet's radius based on the flux difference. This estimation serves as a foundation for the second model, a limb darkening fit aimed at unveiling the physical characteristics of GJ 1214b. The top hat model delineates the transit depth and provides a rough radius value derived from it. Figure 4 depicts the top hat model overlaid on the data. It's important to note that this model lacks accuracy in representing the flux change as it doesn't incorporate ingress and egress times.

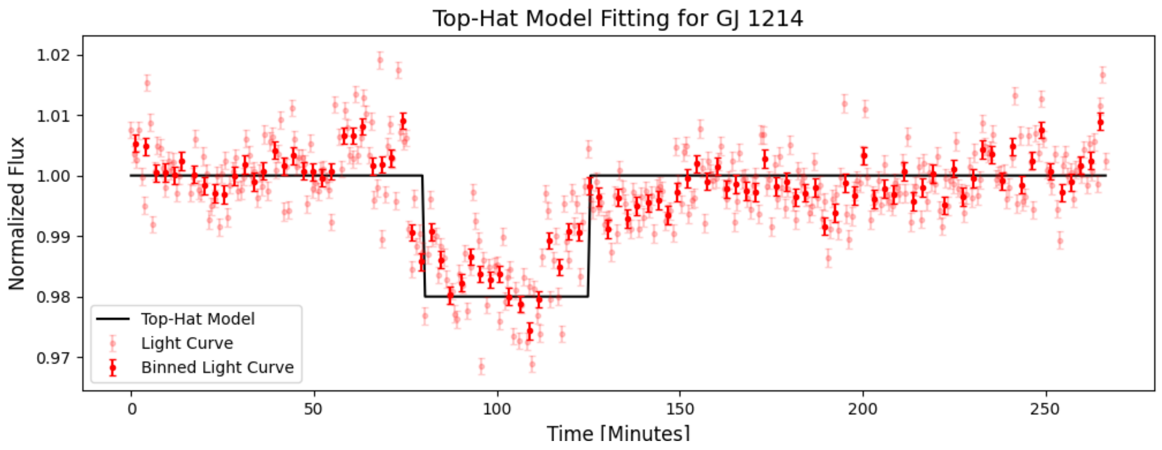


Fig. 4.—: The top hat model is good for determining the maximum change in flux caused by the transit. The returned dip is determined to be about 0.2

Utilizing the parameters obtained, a quadratic limb darkening model fitting is conducted using the program proposed by Mandel & Agol 2002 and Eastman et al. 2012. This program incorporates the inclination, planetary radius, semi-major axis, and orbital period to plot the approximate trajectory followed by the flux dip. Employing a quadratic fit offers enhanced analysis compared to a simple linear fit, although further refinement could be achieved with more robust models.

## 4. Data Analysis

| Parameter | Value | Uncertainty |
|---|---|---|
| Planetary Radius $R_p$ | 0.024 $R_\odot$ | 0.001 $R_\odot$ (5.49%) |
| Semi-Major Axis $a$ | 3.06 $R_\odot$ | 0.14 $R_\odot$ (4.58%) |
| Inclination $i$ | 89.95° | 0.02°(1.44%) |
| Coefficient #1 $c_1$ | 0.65 | 0.09 (15.2%) |
| Coefficient #2 $c_2$ | 0.32 | 0.04 (13.8%) |

Table 1:: The parameters found from the limb darkening model fitting are listed above. Their uncertainties are listed first by value and in brackets by percentage of the found value.

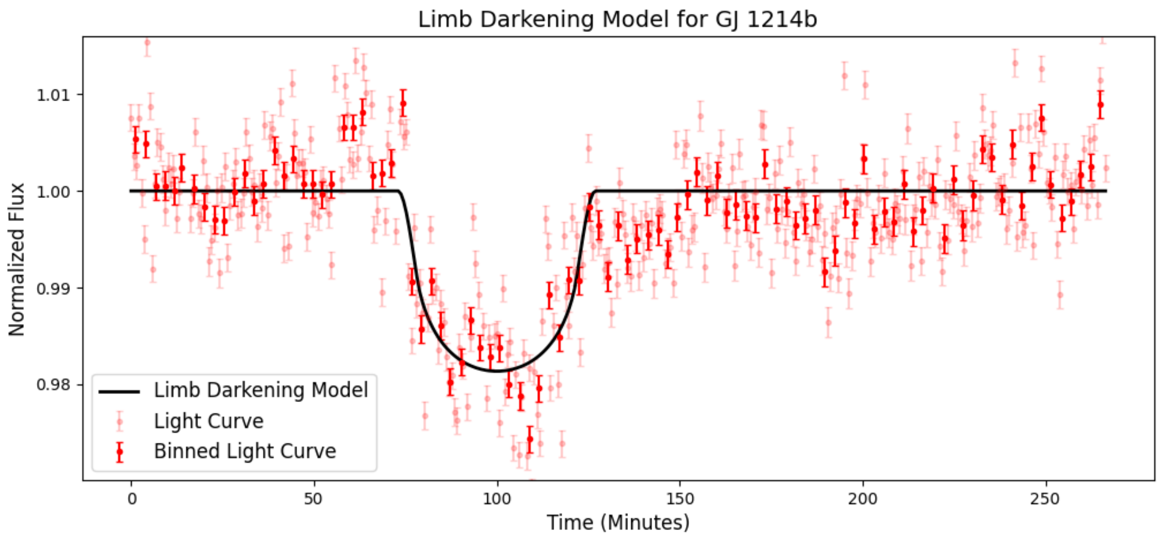


Fig. 5.—: Plot of the model GJ 1214's drop in flux using a limb darkening model with found coefficients of $c_1$ and $c_2$ found from Table 1

The plot depicted in Figure 5 illustrates the limb darkening model utilizing the parameters calculated from Table 1. The line indicates a maximum flux drop of approximately 2% , serving as a reliable estimate for analysis. The radius value corresponds to roughly $0.23M_{Jupiter} \pm 0.01M_{Jupiter}$, meaning it is approximately two thirds the size of Neptune which is about $0.35M_{Jupiter}$.

### 4.1. Physical Properties

Utilizing the obtained parameters, we can ascertain the transit time of GJ 1214b. Firstly, we calculate the impact parameter, $b$, using Equation 7, yielding a value of $1.4*10^{-4} \pm 8*10^{-2}$. This parameter represents the sky-projected distance between the center of the star and the planet.

Subsequently, the total transit time for GJ 1214b to traverse in front of its star is determined. This duration encompasses the period from the initiation of flux drop until the flux returns to its normalized state of 1. The relationship is described by Equation 8, incorporating the total period, semi-major axis, impact parameter, and the radius of both the planet and the star. We calculate a value of 53.28 minutes for the total transit time. There was some difficulty finding the uncertainty of this value due to the arc sine in the equation. Based on the limb darkening plot, it appears that GJ 1214b takes approximately 50 minutes to complete its transit, meaning the calculated value seems to make physical sense.

$$b = a\cos(i) \tag{7}$$

$$t_T = \frac{P}{\pi}\sin^{-1}\left(\frac{\sqrt{(R_s + R_p)^2 + b^2}}{a}\right) \tag{8}$$

The Exoplanet Database provides a mass for GJ 1214b of $8.2M_\oplus \pm 0.4M_\oplus$. Utilizing this information, we calculate the average density of the planet using the well known density formula for a uniform sphere assuming the planet is a perfect shape. This yields a density of $0.11\frac{M_\oplus}{R_\oplus^3} \pm 0.01\frac{M_\oplus}{R_\oplus^3}$, equivalent to $2.48g/cm^3 \pm 0.23g/cm^3$. To assess the validity of our result, we compare it to the Exoplanet Database's reported value of $2.18g/cm^3 \pm 0.12g/cm^3$. Our answer does appear relatively close and simply holds a higher uncertainty.

## 4.2. Quality of Data

In Figure 6, we present the residuals of the light curve and the limb darkening model. Upon examination of the binned data, the plot exhibits satisfactory alignment as the majority of the data points fall within 0.005 of the model, corresponding to flux changes of 0.5%. However, the scattered light points in the background illustrate the entire light curve data against the model, revealing occasional deviations of up to 2%. This significant variability underscores the importance of binning the data, as it mitigates the impact of outlier points on final calculations.

When assessing the uncertainties in the model fittings, a notable discrepancy arises, particularly in the case of the limb darkening coefficients. The high errors observed likely stem from the variability of these coefficients, as the values obtained from the Vizier catalog were not precisely matched to GJ 1214b's system parameters. To address this issue, we considered the guess parameters as true values, as they were sourced from reliable sources. By calculating the relative differences between the model parameters and these true values, we obtained more appropriate uncertainties, as listed in Table 1. However, the limb darkening coefficient still exhibits a substantial error much higher than the other parameters.

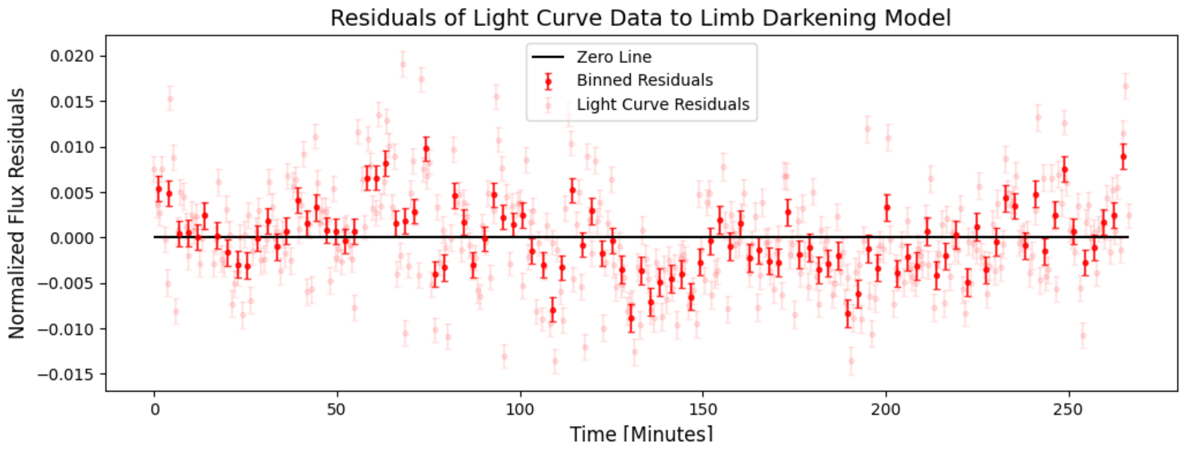


Fig. 6.—: Residual data of the light curve compared to the model fitting. The light red is the original found light curves residuals while the brighter red represents the binned residual data

The Reduced $\chi^2$ value serves as another metric for evaluating the quality of the fit on the data. For the limb darkening model, it returns a value of 6.8, indicating a poor fit. Despite the data appearing relatively close, the persistently high uncertainties in the model parameters contribute to this result. Achieving smaller uncertainties from the model fitting is crucial, and addressing this issue could potentially bring the $\chi^2$ value closer to 1.

## 5. Discussion and Conclusion

One potential source of error to consider lies in the calculation of the median light curve. Our approach involved identifying outliers and determining standard deviations from the model. While this process aims to remove outliers and stars that may have influenced calculations due to background contamination or radius size, the z-score limit of 3.5 may be somewhat lenient. It might be beneficial to tighten this boundary slightly, although caution must be exercised to avoid removing too many points, as this could inflate the significance of remaining points with larger errors. Additionally, reconsidering the number of light curves used in the median model calculation may be prudent. Although we selected 13 as the optimal number, given the initial pool of over 100 stars, conducting one less iterative standard deviation comparison could result in a final model derived from 25 stars, potentially reducing uncertainties during light curve normalization.

It can also help to have a much larger range of data. Having observations spanning multiple transits will help remove uncertainty from final results. Any discrepancies found will have less affect on any parameter estimations and so it will likely help to observe GJ 1214b over several days instead of several hours.

When calculating the density, we assumed the planet to be a perfect sphere, which may not accurately reflect reality. The fast orbital period of 1.58 days suggests the planet may be more ellipsoidal due to tidal forces. Consequently, the assumed volume used for density calculations could skew the actual result. Notably, the density we derived is roughly half that of Earth's, indicating a composition potentially rich in water. One proposed composition suggests a significant water content of 75%, along with 22% silicon and 3% iron (Charbonneau et al. 2009). This suggests that GJ 1214b may predominantly consist of water, unlike Earth, where water primarily exists on the surface.

This report outlines a foundational model for analyzing transiting exoplanets. While useful in determining size and orbital period, it also illustrates how additional parameters can be derived. When combined with other detection methods like radial velocity, it may provide further insights into exoplanets, such as estimating mass and eccentricity. By employing such methods, scientists have detected over 5000 exoplanets to date, some of which may hold potential for life. Continued research holds the promise of discovering more planets with conditions conducive to life as our understanding of exoplanetary systems continues to evolve.

## 6. Bibliography

### REFERENCES

Charbonneau, D., Berta, Z. K., Irwin, J., et al. 2009, Nature, 462, 891–894

Cloutier, R., Charbonneau, D., Deming, D., Bonfils, X., & Astudillo-Defru, N. 2021, The Astronomical Journal, 162, 174

Eastman, J., Gaudi, B. S., & Agol, E. 2012, EXOFAST: Fast transit and/or RV fitter for single exoplanet, Astrophysics Source Code Library, record ascl:1207.001

F., O. 2000, The VizieR database of astronomical catalogues, doi:10.26093/cds/vizier

Fukugita, M., Ichikawa, T., Gunn, J. E., et al. 1996, AJ, 111, 1748

Mandel, K., & Agol, E. 2002, The Astrophysical Journal, 580, L171–L175

Seager, S., Kuchner, M., Hier-Majumder, C. A., & Militzer, B. 2007, The Astrophysical Journal, 669, 1279–1297

## 7. Appendix

Code is provided below for general understanding of the process. Certain parts including the plotting and constant set up is omitted.

### 7.1. tracking.py

This file is finds the movement vectors from one star and applies it to every reference star in the rest of the lab in order to track the movement of the image properly

```python
x, y = [1130,470] #A starting location near an isolated star
boxsize = 20 #Box radius

obs_cube = np.empty([len(target_files), boxsize*2+1, boxsize*2+1])
centroids = np.empty([len(target_files), 2])

for i, f in enumerate(target_files):
    #Open the file
    header, im, cof = read_file(f, sky_image=True, verbose=False)
    #Clean the observation
    im = clean_image(im, median_dark, median_flat)

    #Make a box around the isolated star:
    box = im[y-boxsize: y+boxsize+1,
             x-boxsize: x+boxsize+1].copy()

    #Find the centroid of the star within that box
    centroids[i] =  centroid(box)

    #Save the box
    obs_cube[i] = box
#%%
#Now we can compute the centroid motion vector across the entrie set for this one star
plt.figure(figsize=(14, 5))
movement_vectors = np.array([centroids[i] - centroids[0] for i in range(centroids.shape[0])
    ])
plt.figure(figsize=(12,5))
plt.plot(np.arange(movement_vectors.shape[0]), movement_vectors[:,0],
         label = 'X Centroid Movement',
         color = 'blue')
plt.plot(np.arange(movement_vectors.shape[0]), movement_vectors[:,1],
         label = 'Y Centroid Movement',
         color = 'r')
plt.plot(np.arange(movement_vectors.shape[0]), np.sqrt(movement_vectors[:,0]**2 +
    movement_vectors[:,1]**2),
         label = 'Displacement Magnitude',
```

This preprint was prepared with the AAS LATEX macros v5.0.

```python
        color = 'k')
plt.title("Tracking Movement Vectors of Star 1", size=15)
plt.xlabel("Observation Number", size = 14)
plt.ylabel("Centroid Change", size = 14)
plt.grid(alpha = 0.5)
plt.legend(fontsize=12)
plt.savefig("C:/Users/User/Desktop/University/ASTRO/AST326/Lab 5/Plots/track_1")
plt.show()
```

## 7.2. light_curves.py

This file focuses on using aperture photometry to measure the flux of reference stars to determine a median light curve model

```python
def get_flux(data, aperture, annulus, gain):
  data_annulus = data * annulus
  # background = np.median((data*annulus)[np.nonzero(data * annulus)])      # bg ~ 71,
    tutorial uses average,
  background = np.sum(data_annulus) / np.sum(annulus)                        # bg ~ 72
  # var_background = np.sum((data * annulus - background)**2)
  nonzero_annulus = data_annulus[np.nonzero(data_annulus)]
  var_background = np.mean((nonzero_annulus - background)**2)
  flux = np.sum(data * aperture - background * aperture)
  flux_error = np.sqrt(flux/gain + np.sum(aperture) * (1 + np.sum(aperture)/np.sum(annulus)
    * var_background))    # np.pi/2 for median
  #print(flux, flux_error)
  return flux, flux_error


def get_positions_and_photometry(x, y):
    # Containers
    positions = []
    fluxes = []
    flux_errors = []
    # Loop over each image

    a, b = 0, 364
    for file, vector_x, vector_y in zip(file_list[a:b], movement_x[a:b], movement_y[a:b]):
        # Correct our position with our vector correction

        data, _, _ = fn.read_fits(file)

        tracked_x, tracked_y = int(np.round(x + vector_x,0)), int(np.round(y + vector_y,0))

        # Make a box around the new position
        box = data[tracked_y-boxsize: tracked_y+boxsize+1,
                tracked_x-boxsize: tracked_x+boxsize+1].copy()

        # The tracking in the last lab isn't perfect so recalculate the centroid of the star
     within that box
        new_box_x, new_box_y = fn.get_centroid(box)

        # Convert to the full image coordinates
        new_x = tracked_x + new_box_x - boxsize
        new_y = tracked_y + new_box_y - boxsize
        positions.append((new_x,new_y))

        # Convert to integers for indexing
        new_x, new_y = int(np.round(new_x,0)), int(np.round(new_y,0))

        # Recenter the box
        new_box = data[new_y-boxsize: new_y+boxsize,
                    new_x-boxsize: new_x+boxsize].copy()

        r = fn.find_radius(box)
        print(r)

        # Do the aperture photometry
        aperture = fn.get_aperture(box.shape, (boxsize+new_box_x, boxsize+new_box_y), 5.5)
        annulus = fn.get_annulus(box.shape, (boxsize+new_box_x, boxsize+new_box_y), (15, 18)
    )
        #fn.show_ds9(box)
        #fn.show_ds9(box*(aperture+annulus))
        flux, flux_err = get_flux(box, aperture, annulus, gain=1.16)

        # Store the flux and error
        fluxes.append(flux)
        flux_errors.append(flux_err)

    # Normalize the flux and error
    normalized_flux = fluxes / np.median(fluxes)
    normalized_flux_err = flux_errors / np.median(fluxes)

    # Return normalized flux and error and positions
    return normalized_flux, normalized_flux_err, positions


light_curves = np.empty((len(xs), len(normalized_flux)))
u_light_curves = np.empty_like(light_curves)


for i in range(0, len(xs)):
    try:
        light_curves[i], u_light_curves[i], positions = get_positions_and_photometry(xs[i],
    ys[i])

        zscore = mad(normalized_flux)

        indices_to_replace = zscore > 3.5

        light_curves[i][indices_to_replace] = np.nan
    except ValueError as ve:
        print(f"ValueError occurred at iteration {i}: {ve}")
    except Exception as e:
        print(f"Exception occurred at iteration {i}: {e}")

# Calculate the number of rows and columns for the grid layout
num_light_curves = len(light_curves)
num_rows = int(np.ceil(num_light_curves / 4))  # Adjust the number of columns as needed
num_cols = min(num_light_curves, 2)  # Maximum of 4 columns per row

min_value = 0.8
max_value = 1.2
```

```python
max_uncertainty = 0.07  # TODO: Adjust this when the uncertainties are fixed up

# Function to remove outliers in each light curve
def remove_outliers(light_curve, uncertainty):
    mask_value = np.logical_or(light_curve < min_value, light_curve > max_value)
    mask_uncertainty = uncertainty > max_uncertainty
    mask = np.logical_or(mask_value, mask_uncertainty)
    light_curve[mask] = np.nan  # Mask outliers by setting them to NaN
    uncertainty[mask] = np.nan  # Also mask uncertainties for removed points
    return light_curve, uncertainty

# Apply outlier removal to each light curve
cleaned_light_curves = np.array([remove_outliers(light_curve, uncertainty)
                                  for light_curve, uncertainty in zip(light_curves,
    u_light_curves)])

# Remove light curves with fewer than 200 non-NaN values
valid_light_curves = cleaned_light_curves[:, 0]
valid_uncertainties = cleaned_light_curves[:, 1]
valid_light_curves = valid_light_curves[np.sum(~np.isnan(valid_light_curves), axis=1) >=
    200]
valid_uncertainties = valid_uncertainties[np.sum(~np.isnan(valid_uncertainties), axis=1) >=
    200]

num_light_curves = len(valid_light_curves)
print(num_light_curves)
# Create a figure and subplots

num_rows = int(np.ceil(num_light_curves / 4))  # Adjust the number of columns as needed
num_cols = min(num_light_curves, 2)  # Maximum of 4 columns per row

# Create a figure and subplots
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 4*num_rows))

# Iterate through the light curves and plot each one on a separate subplot
for i, ax in enumerate(axes.flat):
    if i < num_light_curves:
        ax.plot(valid_light_curves[i])
        ax.set_title(f"Light Curve {i+1}")
        ax.set_xlabel("Time")
        ax.set_ylabel("Flux")

# Adjust layout to prevent overlapping
plt.tight_layout()
plt.show()

def calculate_std_dev(light_curve, median_model):
    # Calculate the standard deviation of the flux values of a light curve compared to the
    median model
    std_dev = []

    for lc in light_curve:
        std_dev.append(np.nanstd(lc - median_model))

    return np.array(std_dev)

#%%
num_iterations = 2
```

```python
selected_light_curves = valid_light_curves
for i in range(0, num_iterations):
    median_model = np.nanmedian(selected_light_curves, axis=0)
    std_values = calculate_std_dev(selected_light_curves, median_model)

    sorted_std_array = np.sort(std_values)

# Step 2: Find the median of the sorted array
    median_std = np.median(sorted_std_array)

# Step 3: Keep only the elements with standard deviations less than or equal to the median
    selected_light_curves = [curve for curve, std in zip(valid_light_curves, std_values) if
    std <= median_std]

print(len(selected_light_curves))
final_median_model = np.nanmedian(selected_light_curves, axis=0)
```

## 7.3. fitting.py

This is the main code used in the process of modelling fits onto the transit light curve.

```python
"""Will bin the median data"""
def binning(time_ends, num, x, y, y_err):
    """

    :param time_ends: list, contains the start and end times for a np.linspace array, len(
    time_ends) == 2
    :param num: int, number of steps in the np.linspace array
    :param x: 1D NumPy array, x values for binning
    :param y: 1D NumPy array, y values for binning
    :param y_err: 1D NumPy array, error in y values for binning
    :return: 3 binned 2D NumPy arrays, x binned, y binned, y_err binned
    """

    start, stop = time_ends
    bin_edges = np.linspace(start, stop, num=num)
    inds = np.digitize(x, bin_edges)

    binned_x = np.zeros(len(bin_edges) - 1)
    binned_y_mean = np.zeros(len(bin_edges) - 1)
    binned_y_err = np.zeros(len(bin_edges) - 1)

    for i in range(1, len(bin_edges)):
        bin_mask = (inds == i)
        binned_x[i - 1] = np.mean(x[bin_mask])

        # Calculate weighted mean for y values
        weights = 1.0 / y_err[bin_mask]  # Assuming y_err contains the weights
        weighted_sum = np.sum(y[bin_mask] * weights)
        sum_of_weights = np.sum(weights)
        binned_y_mean[i - 1] = weighted_sum / sum_of_weights
        binned_y_err[i - 1] = np.mean(y_err[bin_mask])

    mask = ~np.isnan(binned_y_mean)
    binned_x = binned_x[mask]
    binned_y_err = binned_y_err[mask]
    binned_y_mean = binned_y_mean[mask]

    return binned_x, binned_y_mean, binned_y_err

"""Top Hat Fitting"""
colour, fs = "#FF0000", 12

def top_hat_model(time, F0, delta):
    return np.where((time >= t_ingress) & (time <= t_egress), F0 - delta, F0)

# TODO: FIX THE INPUT VALUES FOR BETTER FITTING
# Synthetic data (time and flux)
time = julian_dates
F0 = 1.0  # Baseline flux of the star
delta_true = 0.02  # True transit depth
t_ingress = ing  # Ingress time
t_egress = egr  # Egress time
flux = top_hat_model(time, F0, delta_true)

# Fit the top hat model to the synthetic data
top_popt, top_pcov = curve_fit(top_hat_model, time, flux, p0=[F0, delta_true])

# Extract the fitted parameters
F0_fit, delta_fit = top_popt

# Compute the planet-to-star radius ratio
Rp_Rs = np.sqrt(delta_fit)

# Known or estimated radius of the host star
Rs = 0.204

# Determine the radius of the exoplanet
Rp = Rp_Rs * Rs

print("Estimated radius of the exoplanet (Meters):", Rp)

def z_function(t, P, a, i, Rs):

    w = 2*pi/P

    z = (a/Rs)*sqrt((sin(w*t)**2)+(cos(i)**2)*(cos(w*t)**2))
    return z

def lc_model(time, R_p, a, i, u1, u2):
    # Handle any unit conversions

    # Shift the time to be relative to the time of transit
    P = period
    t = time - 100
    # Calculate z, p0, and the light curve
    z = z_function(t, P, a, i, R_s)  # Calculate z using z_function
    p0 = R_p / R_s

    return occultquad(z,u1,u2,p0)[0]


# Set up some initial guesses for the model
guess_planet_radius = R_p # Solar Radii
```

```
guess_semi_major_axis = a # Solar Radii
guess_inclin = inc # degrees
guess_u1 = 0.554   # +/- 0.015
guess_u2 = 0.269    # +/- 0.013
orbit_start_time = 100 # seconds, hours, days, just make sure it's consistent!

guess = np.array([guess_planet_radius, guess_semi_major_axis, guess_inclin, guess_u1,
    guess_u2])
#%%
time = julian_dates

#                   R_p   a    i    c1   c2
limits = np.array([0.1, 3, 1, 0.1, 0.1])

lower_bounds = guess - limits
upper_bounds = guess + limits

# Perfom the curve fitting using the guesses and bounds
popt, pcov = curve_fit(lc_model,
                       xdata = julian_dates,
                       ydata = star_lc,
                       sigma = u_star_lc,
                       p0=[guess_planet_radius,guess_semi_major_axis,guess_inclin,guess_u1,
    guess_u2],
                       bounds=(lower_bounds, upper_bounds),
                       maxfev=20000)
```