課題演習 C3　レポート (2 回目)

0500-34-0042　大平 達也

2025/01/23

1　目的

前回のレポートから, 大きく次の 2 つの進捗があった。

- 参照星のクオリティーチェック
- トランジットモデルとデータの比較

それぞれについて以下で報告する。

2　参照星のクオリティーチェック

前回のレポートで述べた手法で参照星を選んだ。今回は, それらの参照星のクオリティーチェックを行った。

2.1　手法

まず, 参照星のフラックスを平均を用いて各参照星のフラックスを補正した。その後, それぞれの参照星の等級の分散を計算した。最後に, フラックスの変化が 1% を超えるようであれば, フラックスの変化が 1% 程度の, 最低限のビンで移動平均を取った。

コードは (計算しただけであるので) 省略する。

2.2　結果

まず, 参照星の finding chart を図 1 に示す。そして, これら参照星の等級の分散と標準偏差を表 1 に示す。

ここで, フラックス変化が $\Delta F/F$ のときの等級の変化 Δm は,

$$\Delta m = -2.5\log(1+\Delta F/F) \simeq -2.5 \times \frac{1}{\ln 10}\frac{\Delta F}{F} \simeq -1.09\frac{\Delta F}{F}$$

であるため, フラックスの変化が 1% 程度であれば, 等級の変化は約 0.01 等級程度である。

そのため, 観測データの品質は十分であると考えられる。

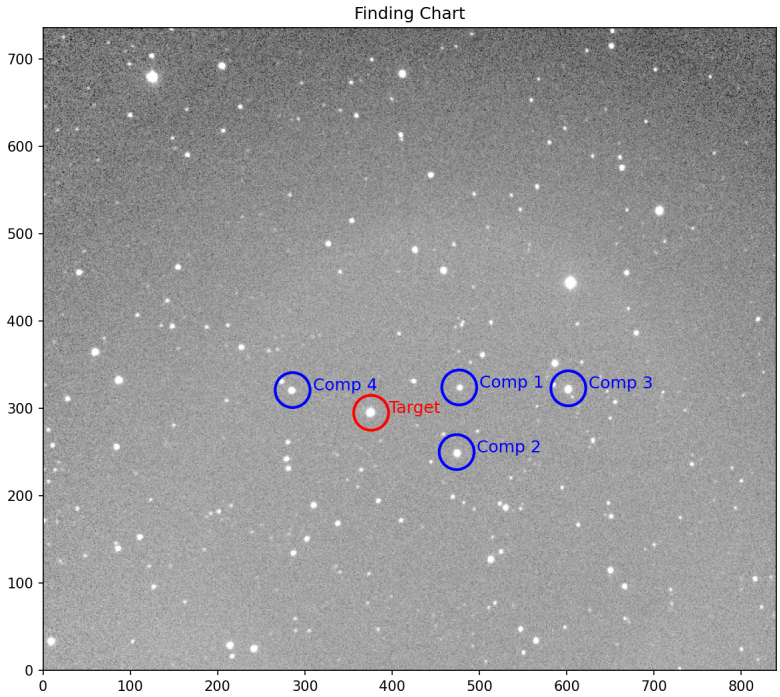

図 1　参照星の finding chart

表 1　参照星の分散と標準偏差

Comp	Variance	StdDev
1	0.00011	0.01028
2	0.00006	0.00782
3	0.00005	0.00693
4	0.00016	0.01249

3　トランジットモデルとデータの比較

前章で品質を確認した参照星を用いて, トランジットの光度曲線を作成する。また, カタログから恒星系のパラメータを取得し, トランジットモデルを作成した。これらを重ねてプロットする。

3.1　コード

Python コードを掲載する。なお, パラメータは, [1] に従った。

実装の方針としては, トランジットモデルの計算の際に, トランジット中心時間をユリウス日で与え, 最後に datetime に変換することで, 観測データとトランジットモデルの時間を合わせた。また, 縦軸は, 観測データから補正したフラックスの, 最後の 50 点の中央値を 1 とする相対値を取った。

ソースコード 1　トランジットモデル計算およびプロット用のコード

```
    dt = datetime.datetime(2024, 11, 11, 21, 22, 00)

    # dt をユリウス日に変換
    t = Time(dt, scale='utc')
    t0_jd = jd_value
```

```
6
7     # Transit parameter 設定
8     params = batman.TransitParams()
9     params.t0 = t0_jd # トランジット中心を JD で設定
10    params.per = 2.056014
11    params.rp = 0.1224
12    params.a = 6.220
13    params.inc = 90.0
14    params.ecc = 0.0
15    params.w = 90.0
16    params.u = []
17    params.limb_dark = "uniform"
18
19    time_jd = np.linspace(params.t0 - 0.07, params.t0 + 0.07, 5000)
20
21    # トランジットモデルの計算
22    m = batman.TransitModel(params, time_jd)
23    flux_model = m.light_curve(params)
24
25    # datetime に変換しておく
26    time_datetime = Time(time_jd, format='jd').to_datetime()
27
28    # ここからはデータのプロット
29    times_jst_naive = [dt.replace(tzinfo=None) if dt is not None and not isinstance(dt,
          float) else dt for dt in times]
30
31    # データをDataFrame に変換
32    df = pd.DataFrame({
33        'Time': times_jst_naive,
34        'Flux_Norm': norm_fluxes
35    })
36
37    # 'Time'をDatetimeIndex に設定
38    df.set_index('Time', inplace=True)
39    time_array = df.index.to_numpy()
40
41    # 光度曲線のプロット
42    fig, ax = plt.subplots(figsize=(10, 6))
43    plt.plot(time_array, df['Flux_Norm'], 'o', markersize=3, alpha=0.3, color='black',
          label="data")
44    plt.plot(time_datetime, flux_model, color='black', lw=3, label="model")
45    ax.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
46    # plt.gca().invert_yaxis()
47    plt.xlabel('Time')
48    plt.ylabel('Flux␣[Rel.]')
49    plt.title('TOI-1516␣Light␣Curve')
50    plt.grid(True)
51    plt.show()
52    fig.savefig('light_curve_fit.png')
```

3.2　結果

図 2 に, 光度曲線を示す。モデルの曲線が概ねデータの範囲に収まっていることが確認された。嬉しい。

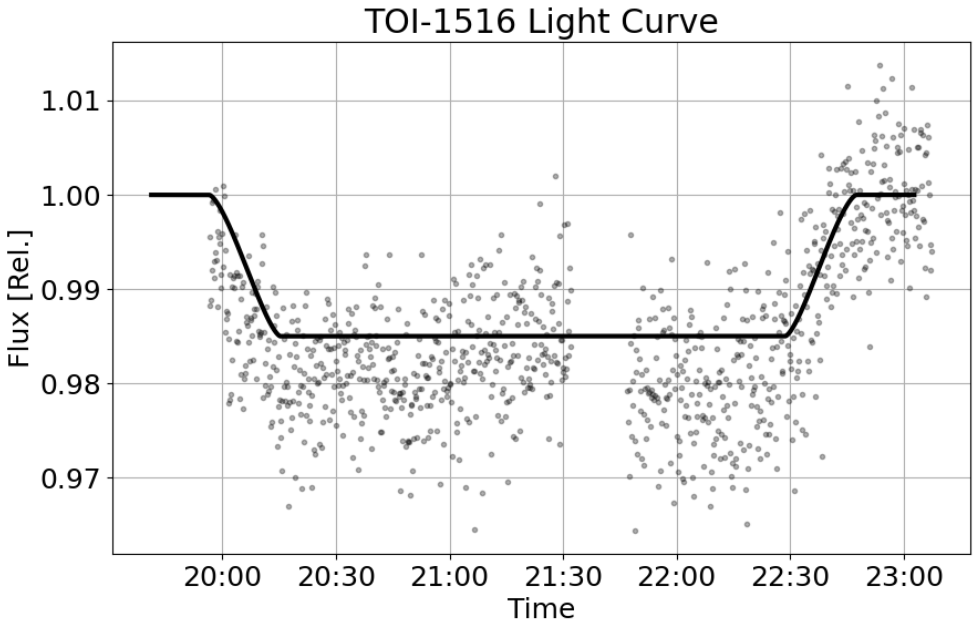

図 2　TOI-1516 の光度曲線

参考文献

[1] Kabáth, P., Chaturvedi, P., et al. 2022, *Monthly Notices of the Royal Astronomical Society*, 513(4), 5955–5972.