

課題演習 C3 レポート (1 回目)

0500-34-0042 大平 達也

2024 年 12 月 23 日

1 目的

系外惑星 TOI1516 b の Transit 検出を目標とする。分かっている、観測日におけるターゲットの情報は以下の通り:

観測日 2024/11/11

ingress 20:00

degrees 22:50

視等級 10.8 mag

減光量 0.016 mag

今回は, ingress の途中から観測を始めた。

2 これまでの進捗状況

2.1 データの一次処理

IRAF を用いて, 11/11 に取得した観測データの一次処理を行った。

2.2 測光用コードの開発

途中まで IRAF のスクリプトを書いていたが, 途中から Python での開発に切り替えた。作成したコードは以下の通り。なおアパーチャーの半径は HWHM の 1.2 倍とした。

ソースコード 1 開口測光用のコード

```
1 import glob
2 from astropy.io import fits
3 from astropy.time import Time
4 from photutils.aperture import CircularAperture, CircularAnnulus, aperture_photometry
5 from photutils.detection import DAOStarFinder
6 from astropy.stats import mad_std
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from zoneinfo import ZoneInfo
10 from datetime import timezone
11 import pandas as pd
12 import matplotlib.dates as mdates
13
14 # 重心座標を検出するための関数を定義
15 def find_star_position(data, initial_pos, search_radius, fwhm=3.0, threshold=5.0):
```

```

16     """
17     初期位置近辺で星を検出し、正確な座標と検出された星の情報を返す関数
18     """
19     # 背景ノイズの推定
20     bkg_sigma = mad_std(data)
21
22     # 星の検出
23     daofind = DAOStarFinder(fwhm=fwhm, threshold=threshold * bkg_sigma)
24     sources = daofind(data - np.median(data))
25
26     if sources is None or len(sources) == 0:
27         print(f"No sources found around position {initial_pos}")
28         return initial_pos, None # 検出できなかった場合は初期位置とNoneを返す
29
30     # 初期位置からの距離を計算
31     distances = np.sqrt((sources['xcentroid'] - initial_pos[0])**2 +
32                        (sources['ycentroid'] - initial_pos[1])**2)
33
34     # 検出範囲内の星をフィルタ
35     within_radius = distances < search_radius
36     if not np.any(within_radius):
37         print(f"No sources within {search_radius} px of position {initial_pos}")
38         return initial_pos, None # 見つからない場合は初期位置とNoneを返す
39
40     # 検出された星の中で最も明るいものを選択
41     brightest = sources[within_radius][sources['flux'][within_radius].argmax()]
42
43     return (brightest['xcentroid'], brightest['ycentroid']), brightest
44
45 # 検索範囲 (pxl)
46 search_radius = 15
47
48 # 結果を保存するリスト
49 times = []
50 rel_mags = []
51 errors = []
52 target_mags = []
53
54 # ターゲット星, 参照星の座標の初期値を入力
55 target_initial = (376.00, 295.00)
56 comp_initials = [
57     (477.00, 324.00),
58     (474.00, 250.00),
59     (310.00, 190.00),
60     (227.00, 371.00)
61 ]
62
63 # 参照星の数を取得
64 num_comps = len(comp_initials)
65
66 # 各参照星のフラックスを保存するリストのリストを作成
67 comp_fluxes_list = [[] for _ in range(num_comps)]
68

```

```

69
70 # FITS ファイルのリストを取得 (日時順にソート)
71 fits_files = sorted(glob.glob('./data/10toi1516a010*.fit'))
72
73 # ターゲット星と参照星の現在の位置
74 target_position = target_initial
75 comp_positions = comp_initials.copy()
76
77 # 各画像での処理
78 for idx, file in enumerate(fits_files):
79     print(f"Processing_{file}_{(idx+1)}/{len(fits_files)}")
80     try:
81         with fits.open(file) as hdu:
82             data = hdu[0].data
83             header = hdu[0].header
84     except Exception as e:
85         print(f"Error_opening_{file}:{e}")
86         for flux_list in comp_fluxes_list:
87             flux_list.append(np.nan)
88         target_mags.append(np.nan)
89         continue
90
91 # 観測時刻を取得
92 if 'DATE-OBS' in header:
93     time_utc = Time(header['DATE-OBS']).to_datetime(timezone=tzutc)
94     time_jst = time_utc.astimezone(ZoneInfo('Asia/Tokyo'))
95 else:
96     print(f"DATE-OBS_not_found_in_{file}.Skipping.")
97     for flux_list in comp_fluxes_list:
98         flux_list.append(np.nan)
99     target_mags.append(np.nan)
100    continue
101    times.append(time_jst)
102
103 # ターゲット星の位置と明るい星の情報を取得
104 target_position, brightest_star = find_star_position(data, target_position,
105     search_radius=search_radius)
106
107 # FWHM の取得
108 if brightest_star is not None and 'fwhm' in brightest_star:
109     measured_fwhm = brightest_star['fwhm']
110     aperture_radius = measured_fwhm * 1.5 # FWHM の 1.5倍をアパーチャ半径とする
111 else:
112     aperture_radius = 6 # デフォルト値
113
114 # 参照星の位置を検索
115 for i, comp_pos in enumerate(comp_positions):
116     comp_positions[i], _ = find_star_position(data, comp_pos, search_radius=
117     search_radius)
118
119 # 測光アパーチャーを定義(動的に設定した半径を使用)
120 positions = [target_position] + comp_positions
121 apertures = CircularAperture(positions, r=aperture_radius)

```

```

120
121 # アパーチャで測光を実行
122 phot_table_apertures = aperture_photometry(data, apertures)
123
124 # 背景補正用アノラスを定義(各アパーチャに対して1つのアノラスを作成)
125 annuli = CircularAnnulus(positions, r_in=aperture_radius + 1, r_out=aperture_radius +
126     4)
127
128 # アノラスで測光を実行
129 phot_table_annuli = aperture_photometry(data, annuli)
130
131 # 各アパーチャに対応するアノラスのフラックスを取得
132 # annuli.area は全アノラスの面積のリストを返す
133 bkg_fluxes = phot_table_annuli['aperture_sum'] / annuli.area * apertures.area
134
135 # アパーチャ内の背景を差し引いたフラックスを計算
136 target_flux = phot_table_apertures['aperture_sum'][0] - bkg_fluxes[0]
137 comp_fluxes = phot_table_apertures['aperture_sum'][1:] - bkg_fluxes[1:]
138
139 # フラックスが負になる場合は np.nan に設定
140 target_flux = target_flux if target_flux > 0 else np.nan
141 comp_fluxes = np.where(comp_fluxes > 0, comp_fluxes, np.nan)
142
143 # 相対等級を計算
144 if not np.isnan(target_flux) and not np.isnan(comp_fluxes).all():
145     comp_flux = np.nanmean(comp_fluxes)
146     if comp_flux > 0:
147         rel_mag = -2.5 * np.log10(target_flux / comp_flux)
148     else:
149         rel_mag = np.nan
150 else:
151     rel_mag = np.nan
152 rel_mags.append(rel_mag)
153
154 # 機械等級を計算
155 if not np.isnan(target_flux) and target_flux > 0:
156     target_mag = -2.5 * np.log10(target_flux)
157 else:
158     target_mag = np.nan
159 target_mags.append(target_mag)
160
161 # 誤差の推定
162 if not np.isnan(comp_flux):
163     error = np.nanstd(comp_fluxes / comp_flux)
164 else:
165     error = np.nan
166 errors.append(error)
167
168 # 各参照星のフラックスをリストに追加
169 for i, flux in enumerate(comp_fluxes):
170     comp_fluxes_list[i].append(flux)
171     print(f'Comparison_{i+1}: {flux}')

```

2.3 測光結果

相対測光により, TOI1516 の等級を各画像で求めた。これを plot したものを図 1 に示す。

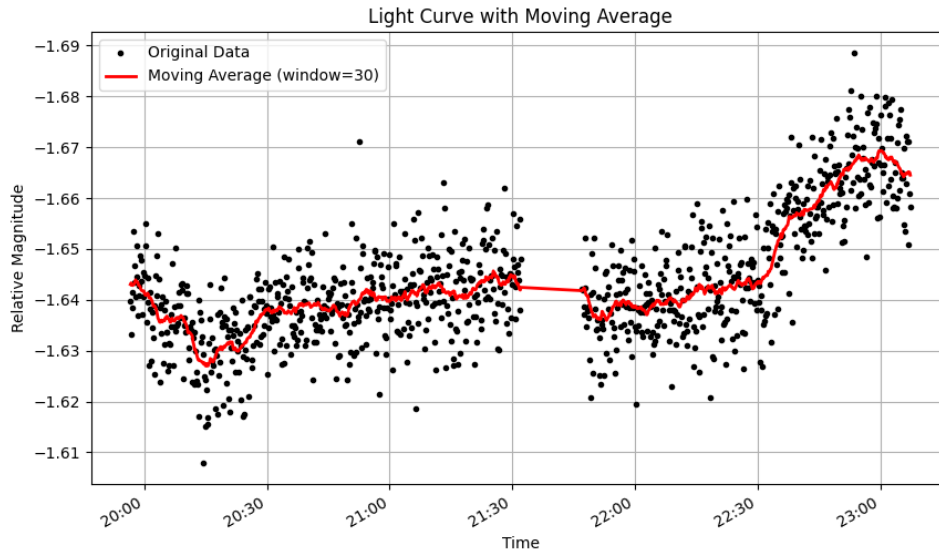


図 1 TOI1516 の光度曲線

2.4 参照星の選び方について

参照星は, 常に視野内に位置し, ターゲット星と同程度の明るさのものを選んだ。また, 測光したのちに, 参照星の等級の時間変化を調べ, 参照星として適切であるか検証した。

たとえば, 最初に選んだ参照星の等級の時間変化は図 2 であった。全体として同じようなトレンドではあるが, Comp 4 (紫) のみノイズが大きいように見える。実際, Comp 4 が参照星の中で最も暗い星であったことから, 参照星として選ぶのは適切でないと判断し, 他の参照星を選んだ。これを繰り返すことで, 最終的な参照星を選んだ。

なお, 参考までに, TOI1516 の高度からエアマスを計算した plot も掲載しておく (図 3)。観測データから, 参照星の等級は時間とともに暗い方へ変化していったためエアマスには反する結果となった。原因は不明。

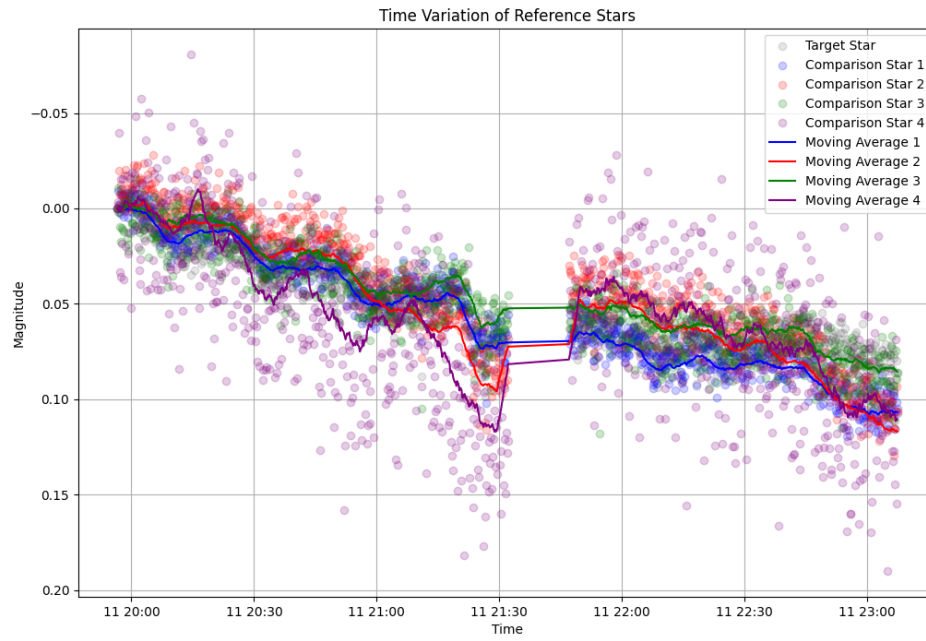


図 2 参照星の光度曲線

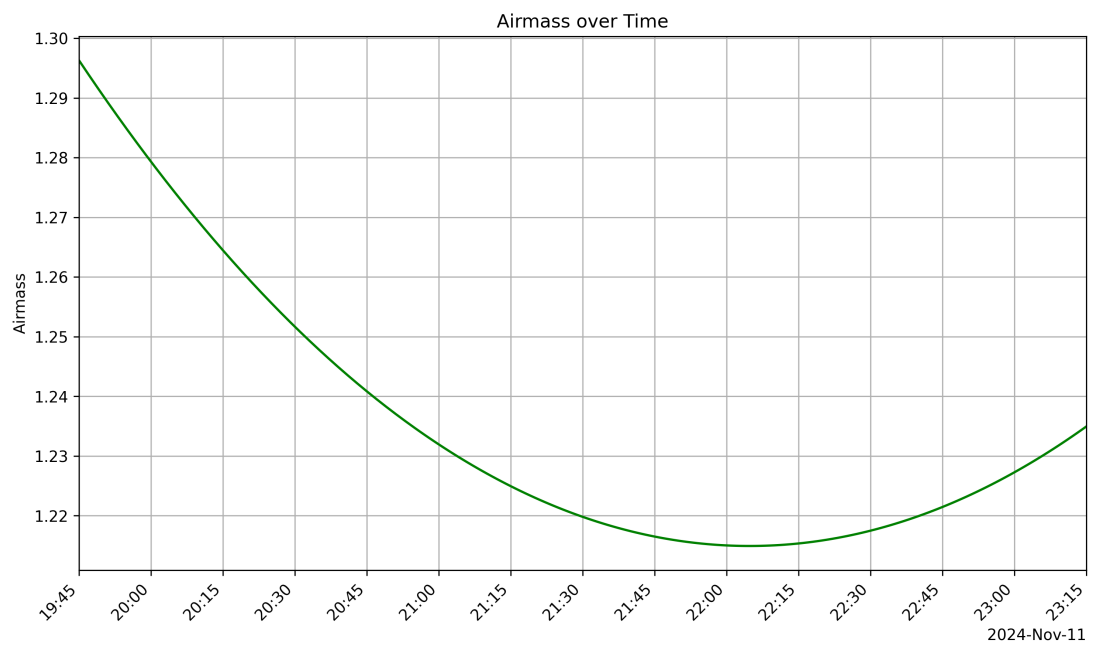


図 3 エアマスの時間変化

3 今後の予定

PSF 測光とかしてみたいです。