課題演習 C3　レポート (1 回目)

0500-34-0042　大平 達也

2024/12/23

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

```python
import glob
from astropy.io import fits
from astropy.time import Time
from photutils.aperture import CircularAperture, CircularAnnulus, aperture_photometry
from photutils.detection import DAOStarFinder
from astropy.stats import mad_std
import numpy as np
import matplotlib.pyplot as plt
from zoneinfo import ZoneInfo
from datetime import timezone
import pandas as pd
import matplotlib.dates as mdates

# 重心座標を検出するための関数を定義
def find_star_position(data, initial_pos, search_radius, fwhm=3.0, threshold=5.0):
```

```
    """
    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
    """
    # 背景ノイズの推定
    bkg_sigma = mad_std(data)

    # 星の検出
    daofind = DAOStarFinder(fwhm=fwhm, threshold=threshold * bkg_sigma)
    sources = daofind(data - np.median(data))

    if sources is None or len(sources) == 0:
        print(f"No sources found around position {initial_pos}")
        return initial_pos, None # 検出できなかった場合は初期位置とNone を返す

    # 初期位置からの距離を計算
    distances = np.sqrt((sources['xcentroid'] - initial_pos[0])**2 +
                        (sources['ycentroid'] - initial_pos[1])**2)

    # 検出範囲内の星をフィルタ
    within_radius = distances < search_radius
    if not np.any(within_radius):
        print(f"No sources within {search_radius}px of position {initial_pos}")
        return initial_pos, None # 見つからない場合は初期位置とNone を返す

    # 検出された星の中で最も明るいものを選択
    brightest = sources[within_radius][sources['flux'][within_radius].argmax()]

    return (brightest['xcentroid'], brightest['ycentroid']), brightest

# 検索範囲 (pxl)
search_radius = 15

# 結果を保存するリスト
times = []
rel_mags = []
errors = []
target_mags = []

# ターゲット星，参照星の座標の初期値を入力
target_initial = (376.00, 295.00)
comp_initials = [
    (477.00, 324.00),
    (474.00, 250.00),
    (310.00, 190.00),
    (227.00, 371.00)
]

# 参照星の数を取得
num_comps = len(comp_initials)

# 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
comp_fluxes_list = [[] for _ in range(num_comps)]

```

```
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=timezone.utc)
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,
           search_radius=search_radius)
105
106    # FWHM の取得
107    if brightest_star is not None and 'fwhm' in brightest_star:
108        measured_fwhm = brightest_star['fwhm']
109        aperture_radius = measured_fwhm * 1.5 # FWHM の 1.5倍をアパーチャ半径とする
110    else:
111        aperture_radius = 6 # デフォルト値
112
113    # 参照星の位置を検索
114    for i, comp_pos in enumerate(comp_positions):
115        comp_positions[i], _ = find_star_position(data, comp_pos, search_radius=
               search_radius)
116
117    # 測光アパーチャーを定義(動的に設定した半径を使用)
118    positions = [target_position] + comp_positions
119    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 +
            4)
126
127     # アノラスで測光を実行
128     phot_table_annuli = aperture_photometry(data, annuli)
129
130     # 各アパーチャに対応するアノラスのフラックスを取得
131     # annuli.area は全アノラスの面積のリストを返す
132     bkg_fluxes = phot_table_annuli['aperture_sum'] / annuli.area * apertures.area
133
134     # アパーチャ内の背景を差し引いたフラックスを計算
135     target_flux = phot_table_apertures['aperture_sum'][0] - bkg_fluxes[0]
136     comp_fluxes = phot_table_apertures['aperture_sum'][1:] - bkg_fluxes[1:]
137
138     # フラックスが負になる場合は np.nan に設定
139     target_flux = target_flux if target_flux > 0 else np.nan
140     comp_fluxes = np.where(comp_fluxes > 0, comp_fluxes, np.nan)
141
142     # 相対等級を計算
143     if not np.isnan(target_flux) and not np.isnan(comp_fluxes).all():
144         comp_flux = np.nanmean(comp_fluxes)
145         if comp_flux > 0:
146             rel_mag = -2.5 * np.log10(target_flux / comp_flux)
147         else:
148             rel_mag = np.nan
149     else:
150         rel_mag = np.nan
151     rel_mags.append(rel_mag)
152
153     # 機械等級を計算
154     if not np.isnan(target_flux) and target_flux > 0:
155         target_mag = -2.5 * np.log10(target_flux)
156     else:
157         target_mag = np.nan
158     target_mags.append(target_mag)
159
160     # 誤差の推定
161     if not np.isnan(comp_flux):
162         error = np.nanstd(comp_fluxes / comp_flux)
163     else:
164         error = np.nan
165     errors.append(error)
166
167     # 各参照星のフラックスをリストに追加
168     for i, flux in enumerate(comp_fluxes):
169         comp_fluxes_list[i].append(flux)
170         print(f'Comparison␣Star␣{i+1}:␣Flux␣=␣{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。

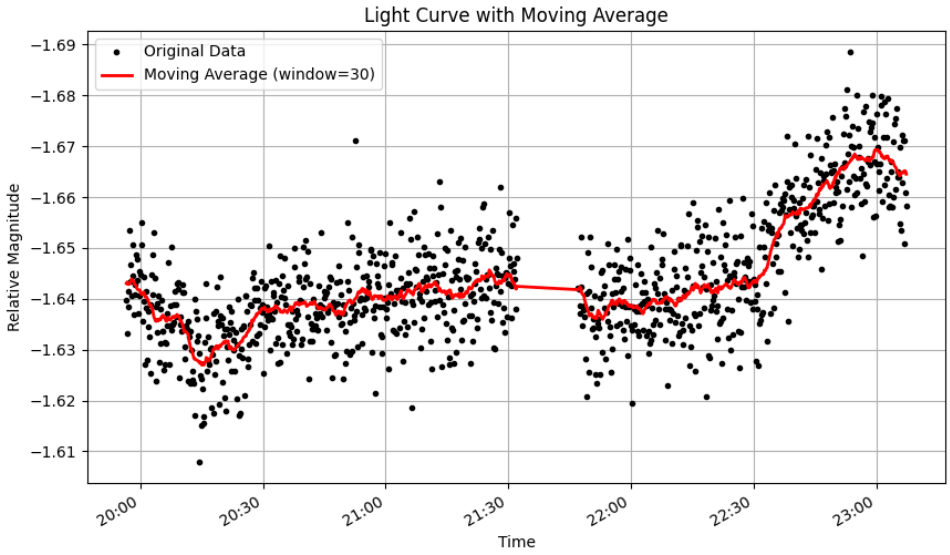

図 1　TOI1516 の光度曲線

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。

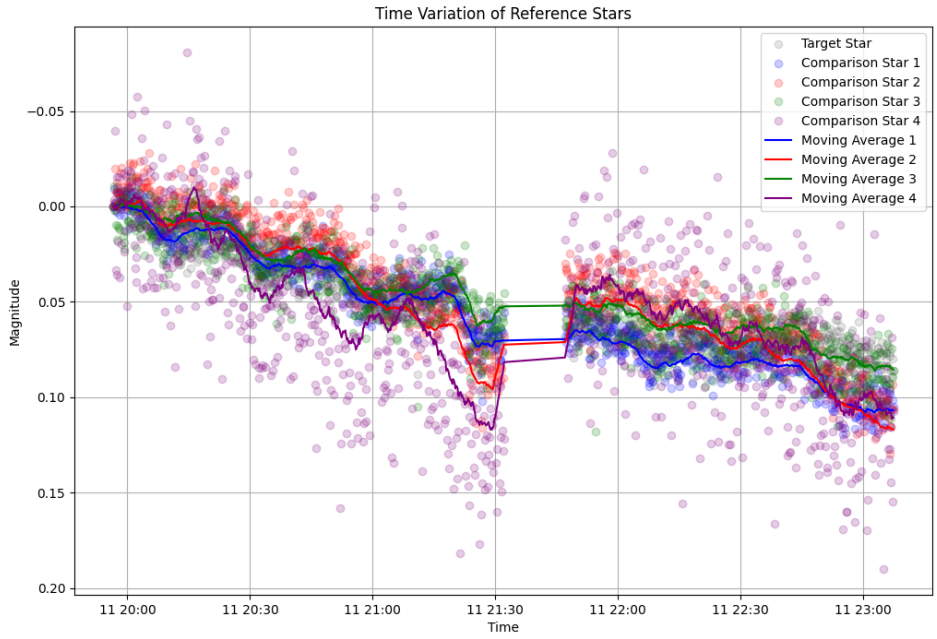

図 2　参照星の光度曲線

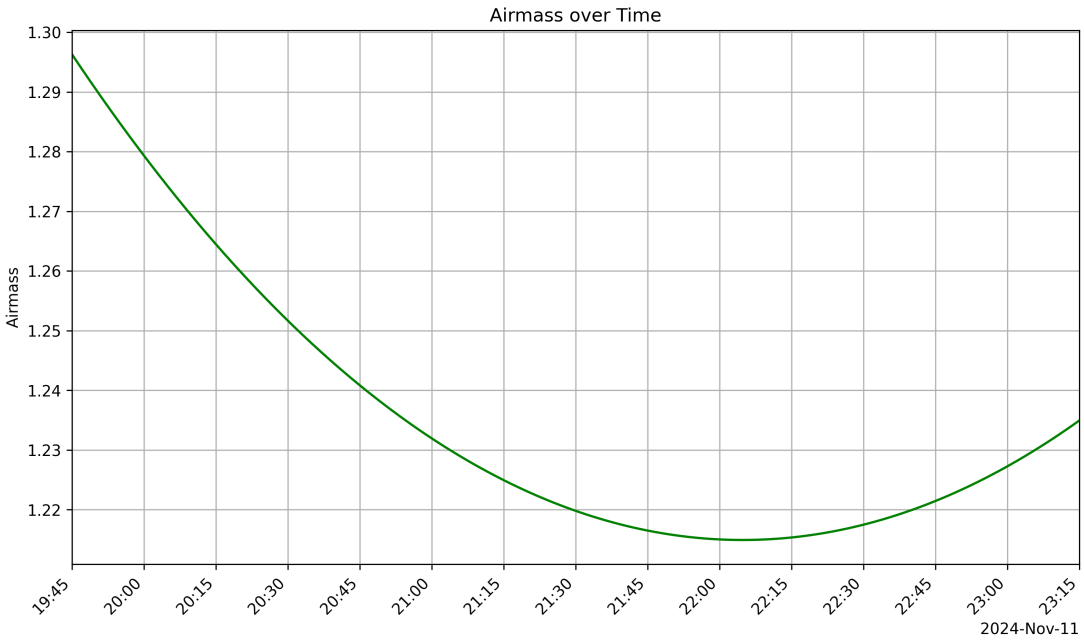

図 3　エアマスの時間変化

3　栗田先生からのフィードバック

- 参照星のペアの quality check
- 分散を取って, 目的として不十分な精度ならば移動平均をとる
  - 移動平均は必要最低限のものを (分散が 1% 程度のものを取る)
  - 移動平均の bin の幅の根拠も
- 予測値を重ねる