ISSN 0367–8466
ИЗВЕСТИЯ КРЫМСКОЙ АСТРОФИЗИЧЕСКОЙ ОБСЕРВАТОРИИ

Изв. Крымск. Астрофиз. Обсерв. 120, № 4, 5–44 (2024)

УДК 520.82

Обработка наблюдений космической обсерватории TESS. Методические рекомендации

М.А. Горбачев

ФГБУН “Крымская астрофизическая обсерватория РАН”, Научный, 298409, Крым
mgorbachev17@gmail.com

Поступила в редакцию 29 ноября 2024 г.

Аннотация. Данная работа является методической инструкцией по использованию данных, полученных орбитальной обсерваторией TESS (Transiting Exoplanet Survey Satellite). В ней описываются особенности наблюдений TESS, а также различные продукты данных, доступных для анализа.

Методическая инструкция в основном представляет собой перевод на русский язык руководства по использованию библиотеки `Lightkurve`, дополненного комментариями и подробными разъяснениями. Библиотека `Lightkurve`, написанная на языке `Python`, позволяет загружать, обрабатывать и визуализировать данные, полученные обсерваторией Kepler/K2 и TESS. Она включает в себя широкий набор функций для поиска, загрузки и анализа кривых блеска различных астрофизических объектов. Представленной информации достаточно для обработки как единичных источников, так и для потокового анализа большого числа объектов, полученных обсерваторией TESS.

Ключевые слова: фотометрия, периодограммный анализ, обработка данных

1 Введение

1.1 TESS

TESS (Transiting Exoplanet Survey Satellite)[1] – миссия NASA, предназначенная для исследования экзопланет, наблюдаемых методом транзита. TESS был запущен в 2018 году. На спутнике установлены четыре идентичные широкоугольные камеры, которые покрывают значительный участок неба (сектор) размером $24° \times 96°$ (рис. 1). Камера 4 центрирована на полюсе эклиптики.

Каждая камера с апертурой 10 см имеет поле зрения $24° \times 24°$ и состоит из четырех ПЗС-матриц с разрешением 4096×4096 пикселей (рис. 2). Изображение строится на площади 2048×2048 пикселей (1 пиксель = $21''$). Остальная область ПЗС-матрицы используется для хранения кадров, чтобы обеспечить быстрое (~ 4 мс) считывание без затвора. Шум считывания не превышает 10 электронов в секунду. Детекторы TESS чувствительны в диапазоне от 600 до 1000 нм (рис. 2). ПЗС-матрицы непрерывно считывают информацию с 2-секундными интервалами. Данные обрабатываются на космической обсерватории блоком обработки данных (DHU). DHU суммирует 2-секундные изображения, чтобы получить 2-минутную или 30-минутную экспозицию. Информация хранится на твердотельном накопителе объемом 192 Гб и передается на Землю, когда обсерватория достигает перигея, каждые 13.7 дня.

Каждый сектор TESS непрерывно наблюдает в течение двух оборотов спутника вокруг Земли или в среднем около 27 дней. В период наблюдения каждого сектора проводятся сеансы связи обсерватории с Землей для планового обслуживания. На время сеансов связи наблюдения останавливаются. Перекрытие секторов вблизи полюса эклиптики означает, что некоторые объекты из этой области будут получать непрерывные данные в течение года. На рис. 3 схематично представлена

[1] https://exoplanets.nasa.gov/tess/

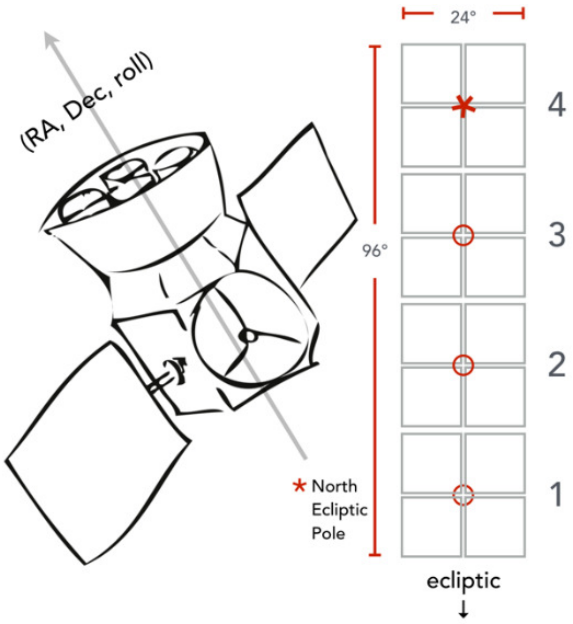

Рис. 1. Схематическое изображение обсерватории TESS и покрываемых участков неба. Изображение взято с официального сайта проекта TESS

карта покрытия небесной сферы после двух лет работы телескопа, где цвет характеризует продолжительность непрерывных наблюдений различных областей. Подробнее о проекте TESS изложено в работе Ricker et al. (2015).

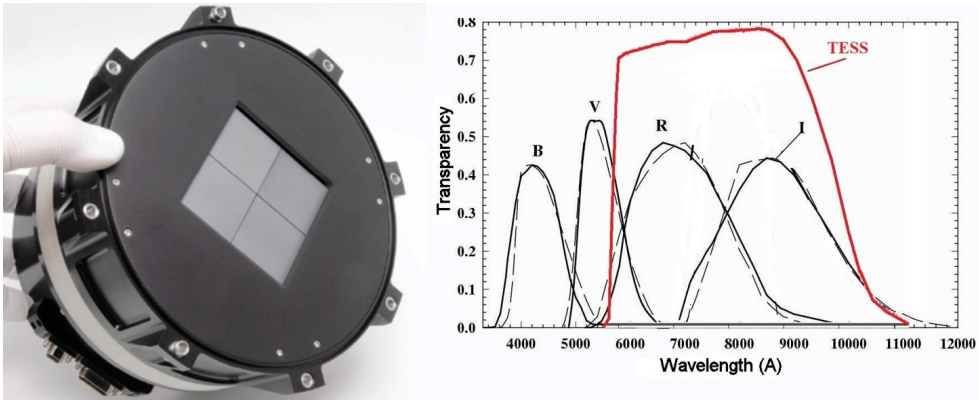

Рис. 2. Изображение одной из четырех камер телескопа TESS (левая панель) и кривой пропускания фильтра TESS (правая панель). Иллюстрация из работы Ricker et al. (2015)

Получаемые TESS данные очень похожи на результаты, предоставляемые миссиями Kepler/K2[2], но, в отличие от последних, TESS покрывает гораздо большую область неба с более низким разрешением ($21''$/pix – TESS, $4''$/pix – Kepler). Кроме того, по сравнению с Kepler, TESS наблюдает в нескольких различных режимах экспозиции, включая 20 и 120 секунд, а также 10 и 30 минут. Существует несколько типов предоставляемой информации:

- Файлы целевых пикселей (Target Pixel File – TPF), представляющие собой определенные группы пикселей с объектом и некоторым полем вокруг. Эти файлы содержат вырезки изображений,

[2] https://www.nasa.gov/mission_pages/kepler/main/index.html

содержащих исследуемый объект и некоторую область фона. TPF доступны с временным разрешением 20 секунд, 2 минуты и 30 минут.

- Полнокадровые изображения (Full Frame Images – FFI) – временные ряды этих файлов получены с временным разрешением 200 секунд, 10 и 30 минут для всего поля зрения (в зависимости от сектора). Доступны как калиброванные (`ffic.fits`), так и некалиброванные (`ffir.fits`) файлы. Сервис `TESScut` (см. раздел 3) позволяет создавать вырезки из FFI, которые представлены как TPF из калиброванных FFI, но без операции вычитания фона.
- CBV-файлы (Cotrending Basis Vectors). Базисные векторы содержат набор систематических трендов, используемых для удаления общих систематических ошибок из данных.
- Кривые блеска (Light curve products). Эти файлы содержат временные ряды измерений, полученные с временным разрешением 20 секунд и 2 минуты. В них содержится несколько наборов данных: простая апертурная фотометрия (SAP_FLUX), значения простой апертурной фотометрии с учетом общих систематических ошибок (CBV) – PDCSAP_FLUX.

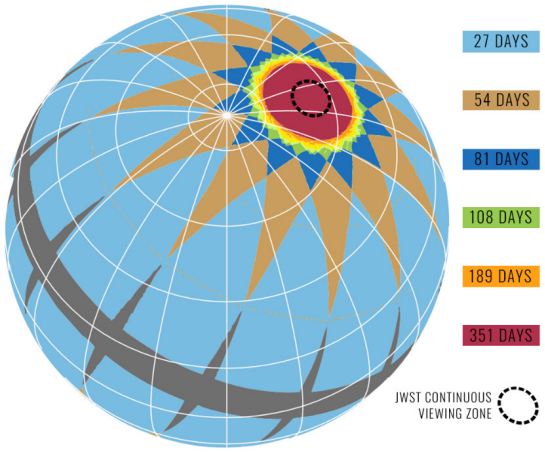

Рис. 3. Карта покрытия небесной сферы за два года работы телескопа TESS. Изображение взято с официального сайта проекта TESS

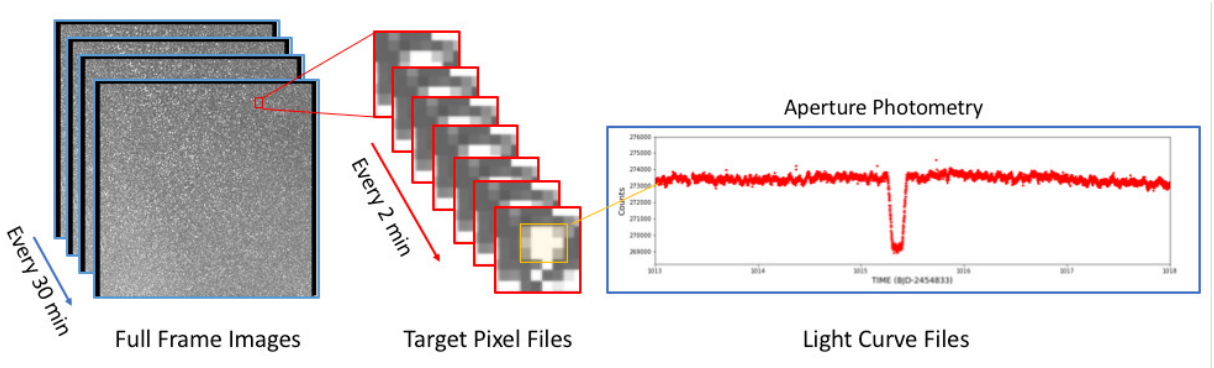

Рис. 4. Основные типы данных, предоставляемых пользователям. Изображение с сайта проекта TESS

Доступ ко всем продуктам данных осуществляется с помощью портала MAST (Mikulski Archive for Space Telescopes)[3]. Схематично все основные типы информации представлены на рис. 4. Использование библиотеки `Lightkurve`[4] позволяет запрашивать и загружать каждый из этих продуктов

[3] https://archive.stsci.edu
[4] https://lightkurve.github.io/lightkurve/

данных. Кроме того, с помощью `Lightkurve` можно запросить и загрузить продукты высокого уровня (High Level Science Products – HLSP) – версии данных, созданные с помощью специальных инструментов обработки, анализа или фотометрии. `Lightkurve` имеет доступ к кривым блеска HLSP[5], полученным с помощью фотометрических инструментов EVEREST, K2SFF и K2SC.

Как было заявлено в информационном сообщении перед запуском проекта, данные TESS становятся общедоступными для всего мирового сообщества через четыре месяца после наблюдений. В течение первого года миссии (июль 2018 – июль 2019) наблюдалось южное полушарие, в течение второго года – северное полушарие. На третий и четвертый год работы обсерватории повторно наблюдается южное и северное полушарие. Стабильность работы инструмента позволила расширить основную миссию и внести ряд изменений в работу телескопа. Начиная с сектора 27 (июль 2020 года) FFI считываются с 10-минутным временным разрешением вместо 30-минутного. Также некоторые выбранные объекты считываются с разрешением в 20 секунд в дополнение к 2-минутному. CBV-файлы не создаются для 20-секундных продуктов. В начале сентября 2022 года (сектор 56) произошли очередные изменения: FFI считываются с 200-секундным интервалом.

Стоит еще раз подчеркнуть, что TESS предоставляет несколько уровней обработки данных. Пользователи должны самостоятельно выбирать нужный тип данных в зависимости от научных задач. Кривые блеска самого высокого уровня наиболее обработаны и подходят для поиска экзопланет, но в таких данных могут быть удалены некоторые астрофизические сигналы. Вместо этого можно использовать простую апертурную фотометрию SAP_FLUX или произвести собственное удаление тренда. В данных PDCSAP_FLUX удаляются инструментальные систематические тренды путем подгонки и удаления тех сигналов, которые являются общими для всех звезд на одной и той же ПЗС. Разработчики отмечают, что иногда этот метод обработки удаляет реальные сигналы, заставляя пользователей вернуться к более раннему уровню данных.

1.2 Портал MAST

MAST – проект, финансируемый NASA, направленный на предоставление мировому сообществу различных архивов астрономических данных с акцентом на космические телескопы в оптическом, ультрафиолетовом и ближнем инфракрасном диапазонах электромагнитного спектра. TESS – одна из миссий, которую поддерживает MAST наряду с другими проектами, такими как Hubble, JWebb, Kepler/K2. MAST является аффилированным центром данных миссии Gaia, архивом проекта PanSTARRS и играет ведущую роль в международной виртуальной обсерватории. Наличие данных TESS, размещенных совместно с более чем 20 другими миссиями, охватывающими десятилетия и широкий диапазон электромагнитного спектра, позволяет проводить уникальный поиск и анализ данных.

2 Определение сектора, камеры и ПЗС-матрицы

Иногда возникает необходимость узнать не только номер сектора, в котором наблюдался исследуемый объект, но и идентификатор камеры, а также ПЗС-матрицы. Например, необходимо получить полнокадровые изображения TESS с исследуемым объектом, используя средство массовой загрузки TESS (см. раздел 2). Для этого необходимо знать номер камеры и ПЗС-матрицы, чтобы не загружать лишние данные.

Получить необходимую информацию можно несколькими способами. Рассмотрим каждый из них:

- Используя сервис `TESScut` (см. раздел 3). При вводе данных объекта пользователю предоставляется информация о том, в каком секторе наблюдался объект, а также номер камеры и ПЗС-матрицы.
- С помощью библиотеки `Lightkurve`. К загруженным объектам `LightCurve`/TPF/`TESScut` необходимо применить свойство `meta`, указав интересующий параметр “CAMERA” и “CCD” (см. раздел 5.2).

[5] Подробная информация о HLSP доступна по адресу: https://archive.stsci.edu/hlsp/

- Кроме того, можно воспользоваться библиотекой `astroquery`. Поиск осуществляется с использованием функции `Tesscut.get_sectors`, в качестве параметров она принимает название объекта или его координаты:

```
from astroquery.mast import Tesscut
from astropy.coordinates import SkyCoord
```

```
sector_table=Tesscut.get_sectors(objectname="TIC 32449963")
coord=SkyCoord(135.1408, −5.1915, unit="deg")
sector_table=Tesscut.get_sectors(coordinates=coord)
```

В результате получим таблицу с нужными параметрами (рис. 5).

sectorName	sector	camera	ccd
str14	int32	int32	int32
tess-s0010-1-4	10	1	4
tess-s0037-1-4	37	1	4
tess-s0046-4-3	46	4	3

Рис. 5. Результат запроса данных посредством `astroquery` для интересующего объекта с информацией об идентификаторе камеры и ПЗС-матрицы

3 TESScut

Сервис[6] позволяет создавать вырезки, центрированные на заданном объекте из полнокадровых изображений телескопа TESS (TESS Full Frame Image). В диалоговом окне (рис. 6) необходимо ввести название/координаты объекта и выполнить поиск данных (`Check for Observations/Resolve Target`).

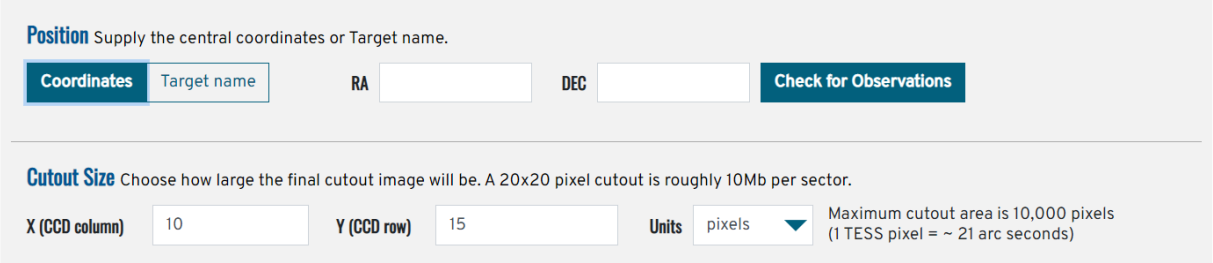

Рис. 6. Диалоговое окно сервиса `TESScut` с областью для ввода координат или названия исследуемого объекта

В подпункте `Cutout Size` можно выбрать размеры вырезаемого фрагмента изображения, указав количество пикселей по осям X и Y. Стоит учитывать угловой размер пикселя TESS (1 TESS pixel = 21 arcsec). В поле `Units` можно выбрать единицы, в которых задается размер вырезки. При нажатии `Refresh Sectors` система отправляет запрос на портал MAST и выдает информацию о секторе, камере и ПЗС-матрице, в которых наблюдался объект (рис. 7).

[6] Получить доступ к `TESScut` можно по адресу: https://mast.stsci.edu/tesscut/.

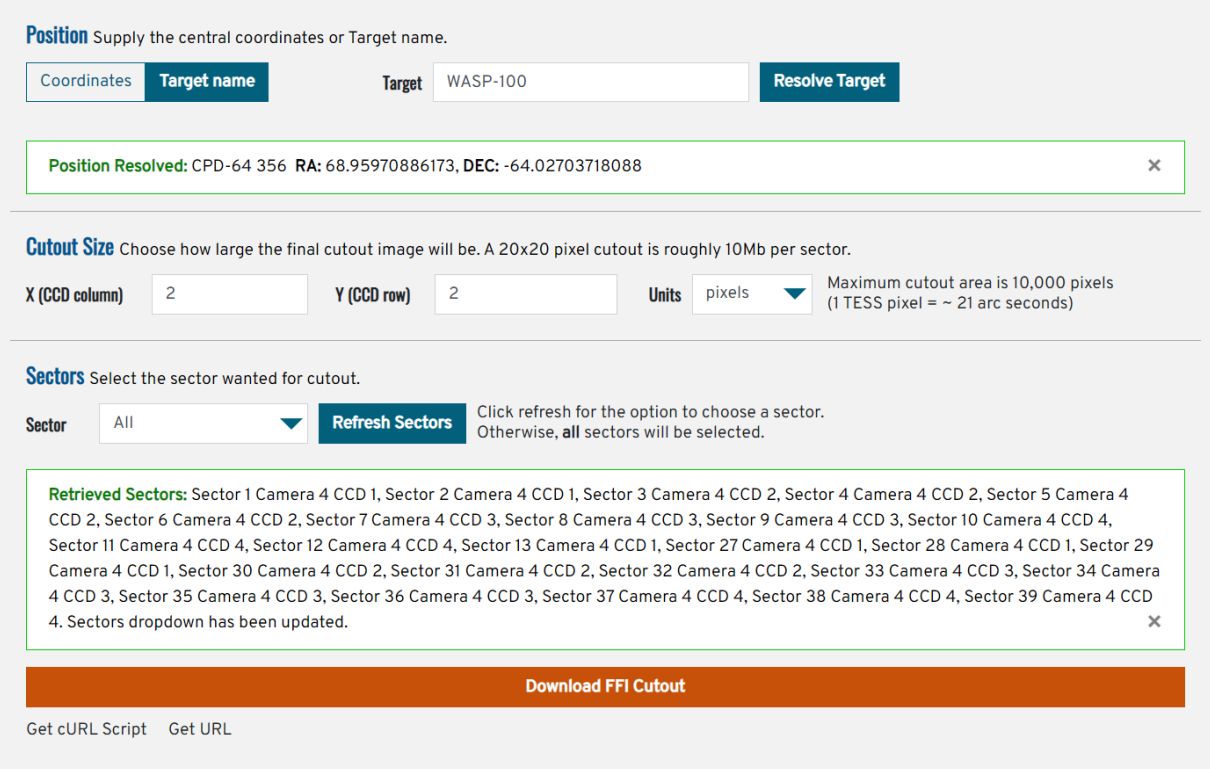

Рис. 7. Страница сервиса `TESScut` с полями для ввода размеров вырезаемой области, а также информации о доступных для загрузки секторах

Выбрав в выпадающем меню нужный сектор (рис. 8), можно загрузить данные, нажав `Download FFI Cutout`.

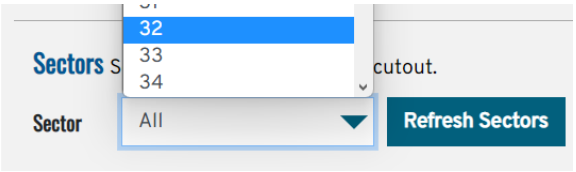

Рис. 8. Пример выпадающего меню со списком доступных для загрузки секторов

В результате будет загружен архив с `fits`-файлом, содержащим значения инструментальных потоков для каждого пикселя из указанного бокса. Полученные данные можно обрабатывать любым удобным способом, а также с помощью библиотеки `Lightkurve`, подробнее об этом в разделе 5. Полезным свойством данного сервиса является удобное получение информации о секторах, в которых наблюдался исследуемый объект.

4 Загрузка полнокадровых изображений

Портал MAST предоставляет возможность загрузки полнокадровых изображений обсерватории TESS. На соответствующей странице проекта[7] указаны ссылки для загрузки данных по каждому сектору. При переходе по ссылке на компьютер загружается скрипт с расширением `.sh` (например, название

[7] https://archive.stsci.edu/tess/bulk_downloads/bulk_downloads_ffi-tp-lc-dv.html

скрипта для сектора 68: `tesscurl_sector_68_ffic.sh`). В нем содержатся URL-адреса для каждого изображения. Для запуска скрипта применяются различные программы, в зависимости от используемой операционной системы.

Как уже говорилось ранее, каждый сектор представлен данными с 16 ПЗС-матриц, по 4 ПЗС-матрицы на каждой из четырех камер. Поле, охватываемое каждой из них, составляет $12^\circ \times 12^\circ$. По умолчанию при загрузке сектора будут последовательно загружаться данные со всех ПЗС-матриц, в связи с этим возникает необходимость наличия большого количества свободного места на диске используемого компьютера. При этом интересующий объект будет находиться только на определенной камере и ПЗС-матрице. Для загрузки данных целого сектора с определенной ПЗС-матрицы желательно наличие около 1 Тб свободного места на жестком диске.

5 Библиотека `Lightkurve`

5.1 Установка и подключаемые модули

Для работы необходимо установить `Python`. Если на ПК уже установлен `Python`, то установить библиотеку `Lightkurve`[8] можно с помощью менеджера пакетов `pip`. В окне терминала необходимо ввести

```
pip install lightkurve
```

Для обновления библиотеки `Lightkurve`:

```
python -m pip install --upgrade lightkurve
```

Перед началом использования `Lightkurve` необходимо подключить библиотеки, которые могут понадобиться в процессе работы:

```
import pandas as pd
import numpy as np
import lightkurve as lk
from astropy.io import fits
import matplotlib.pyplot as plt
from astropy.table import Table
%matplotlib inline
```

5.2 Объекты `LightCurve`

Как уже говорилось во Введении, данные телескопа TESS представляются в нескольких видах. В зависимости от того, какой тип данных необходим, используются те или иные функции. В этом разделе будут рассмотрены объекты типа `LightCurve`. В качестве примера возьмем звезду Kepler-862, у которой обнаружена экзопланета, и посмотрим, какие готовые кривые блеска доступны для нее. Это можно сделать с использованием функции `search_lightcurve` библиотеки `Lightkurve`:

```
search_result=lk.search_lightcurve ('Kepler-862')
search_result
```

Примечание: здесь и далее по тексту приводятся примеры с использованием платформы `Python Jupyter Notebook`, если она не используется, то для вывода информации на печать необходимо применять функцию `print()`.

[8] https://lightkurve.github.io/lightkurve/

Функция `search_lightcurve` (рис. 9, левая панель) возвращает таблицу `search_result`, содержащую информацию о данных, доступных для загрузки. В результате поиска выясняется, что Kepler-862 наблюдался миссией Kepler и TESS. В данной методической инструкции сделан акцент на использование данных телескопа TESS, поэтому работа с данными миссий Kepler и K2 не рассматривается. Можно явным образом указать некоторые критерии поиска, используя ключевые слова. Например, укажем конкретную миссию и сектор, используя ключевое слово `mission` и `sector`:

```
search_result=lk.search_lightcurve ('Kepler-862', mission='TESS', sector=54)
search_result
```

В результате получим таблицу `search_result` (рис. 9, правая панель), содержащую только данные, удовлетворяющие заданным критериям поиска.

SearchResult containing 19 data products.

#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	Kepler Quarter 04	2010	Kepler	1800	kplr009726659	0.0
1	Kepler Quarter 05	2010	Kepler	1800	kplr009726659	0.0
2	Kepler Quarter 06	2010	Kepler	1800	kplr009726659	0.0
3	Kepler Quarter 08	2011	Kepler	1800	kplr009726659	0.0
4	Kepler Quarter 09	2011	Kepler	1800	kplr009726659	0.0
5	Kepler Quarter 10	2011	Kepler	1800	kplr009726659	0.0
6	Kepler Quarter 12	2012	Kepler	1800	kplr009726659	0.0
7	Kepler Quarter 13	2012	Kepler	1800	kplr009726659	0.0
8	Kepler Quarter 14	2012	Kepler	1800	kplr009726659	0.0
9	Kepler Quarter 16	2013	Kepler	1800	kplr009726659	0.0
10	Kepler Quarter 17	2013	Kepler	1800	kplr009726659	0.0
11	TESS Sector 14	2019	CDIPS	1800	273379660	0.0
12	TESS Sector 15	2019	CDIPS	1800	273379660	0.0
13	TESS Sector 41	2021	SPOC	120	273379660	0.0
14	TESS Sector 41	2021	TESS-SPOC	600	273379660	0.0
15	TESS Sector 54	2022	SPOC	120	273379660	0.0
16	TESS Sector 54	2022	TESS-SPOC	600	273379660	0.0
17	TESS Sector 55	2022	SPOC	120	273379660	0.0
18	TESS Sector 55	2022	TESS-SPOC	600	273379660	0.0

SearchResult containing 2 data products.

#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	TESS Sector 54	2022	SPOC	120	273379660	0.0
1	TESS Sector 54	2022	TESS-SPOC	600	273379660	0.0

Рис. 9. Результат поискового запроса для Kepler-862. Общая таблица `SearchResult` (левая панель) и таблица `SearchResult`, полученная с использованием ключевых слов (правая панель)

Ниже приведен полный список принимаемых функцией `search_lightcurve` параметров: `lk.search_lightcurve (target, radius=None, exptime=None, mission=('Kepler', 'K2', 'TESS'), author=None, quarter=None, campaign=None, sector=None)`.

- `target:` общепринятое название объекта; идентификатор TIC/KIC/EPIC (например, 11904151); координаты объекта в градусах: ‘285.67942179 + 50.24130576’, а также в формате часы, минуты, секунды и градусы, минуты, секунды: ‘19:02:43.1 + 50:14:28.7’;
- `radius:` принимает число с плавающей точкой. Предполагается, что оно выражено в угловых секундах. Если указано `None`, то по умолчанию используется 0.0001 угловой секунды;
- `exptime:` можно явным образом указать время экспозиции в секундах (120, 600 и т. п.). Также принимает значения `'long'` (10- и 30-минутные продукты), `'short'` (1- и 2-минутные продукты), `'fast'` (возвращает 20-секундные продукты);
- `mission:` принимает название миссии Kepler/K2 или TESS;

- `author:` принимает название инструмента обработки данных. Официальные инструменты обработки данных для миссий Kepler, K2 и TESS называются “Kepler”, “K2” и “SPOC”. Кроме того, научным сообществом предоставляются альтернативные алгоритмы обработки: “K2SFF”, “EVEREST”. По умолчанию возвращаются все кривые блеска независимо от автора;
- `quarter/campaign/sector:` принимает номер конкретной *четверти/кампании/сектора* для миссий Kepler/K2 и TESS соответственно. По умолчанию возвращаются все доступные данные.

Для получения дополнительной информации о доступных данных в `search_result` необходимо вызвать функцию `search_result.table`. Описание каждого столбца таблицы можно найти на странице портала MAST[9].

Загрузка кривой блеска осуществляется с помощью вызова метода `download()`. При использовании этого метода нужно указать соответствующий индекс строки таблицы `search_result`:

```
lc=search_result[0].download()
```

Результатом является загрузка объекта `LightCurve`, который показан ниже (рис. 10) в виде таблицы `astropy`.

time	flux	flux_err	timecorr	cadenceno	centroid_col	centroid_row	sap_flux	sap
	electron / s	electron / s	d		pix	pix	electron / s	ele
Time	float32	float32	float32	int32	float64	float64	float32	
2769.901308377147	2.3258354e+02	8.3890572e+00	2.0499476e-03	1110560	701.30302	2016.58054	1.4659497e+03	5.0054
2769.9026972947318	1.9983601e+02	8.3581038e+00	2.0499770e-03	1110561	701.30121	2016.58222	1.4439877e+03	4.9869
2769.904086212317	2.0256972e+02	8.3595400e+00	2.0500063e-03	1110562	701.30306	2016.57955	1.4463287e+03	4.9877
2769.9054751296685	2.1221455e+02	8.3691416e+00	2.0500354e-03	1110563	701.30457	2016.58195	1.4501248e+03	4.9935

Рис. 10. Пример представления объекта `LightCurve` в виде таблицы `astropy` для Kepler-862

Загружаемые с портала MAST данные представлены в формате `fits`-файла. Эти файлы содержат множество метаданных о наблюдениях. Все метаданные сохраняются и доступны с помощью вызова свойства `meta` объекта `LightCurve` (рис. 11):

```
lc.meta
```

```
{'INHERIT': True,
 'EXTNAME': 'PRIMARY',
 'EXTVER': 1,
 'SIMDATA': False,
 'TELESCOP': 'TESS',
 'INSTRUME': 'TESS Photometer',
 'OBJECT': 'TIC 273379660',
 'TICID': 273379660,
 'RADESYS': 'ICRS',
 'RA_OBJ': 297.779286442914,
 'DEC_OBJ': 46.4927280242105,
 'EQUINOX': 2000.0,
 'EXPOSURE': 20.774882889667,
 'TIMEREF': 'SOLARSYSTEM',
```

Рис. 11. Фрагмент вывода метаданных

[9] https://mast.stsci.edu/api/v0/_c_a_o_mfields.html

Метаданные включают не только информацию о наблюдениях, но и данные из входного каталога Kepler (KIC) или TESS (TIC), используемые для выбора целей наблюдения, таких как звездные величины, температура и другие параметры. Свойство meta представляет собой словарь Python, обладающий некоторыми удобными функциями. Например, можно получить значение отдельного ключевого слова:

```
lc.meta['SECTOR']
```

```
54
```

Как было показано выше, объект LightCurve представляет собой таблицу. Первые шесть столбцов присутствуют во всех объектах LightCurve и содержат наиболее часто используемую информацию:

- time: значение времени для каждой точки данных в шкале времени BTJD, где BTJD = BJD − 2457000 (BJD – барицентрическая юлианская дата);
- flux: поток рассматриваемого объекта при каждом измерении времени. По умолчанию он содержит значения PDCSAP_FLUX;
- flux_err: статистическая неопределенность в определении потока;
- quality: информация о качестве данных;
- centroid_col & centroid_row: положение целевой звезды на ПЗС-матрице при каждом наблюдении. Этот параметр меняется со временем, например из-за небольших изменений положения/ориентации космического аппарата.

В остальных столбцах представлена более подробная информация о наблюдениях. Некоторые из них дублируются в первых шести столбцах, описанных выше:

- timecorr: значения коррекции, позволяющие пользователям вернуться к не барицентрическим временным меткам;
- cadenceno: это идентификаторы каждой экспозиции для конкретной миссии;
- sap_flux & sap_flux_err: поток SAP и связанная с ним ошибка;
- sap_bkg & sap_bkg_err: расчетный фон, используемый для вычисления потока SAP, и связанная с ним ошибка внутри апертуры;
- pdcsap_flux & pdcsap_flux_err: поток PDCSAP и связанная с ним ошибка. Дублируется по умолчанию в flux и flux_err;
- sap_quality: информация о качестве данных при каждом измерении времени. Дублируется в quality;
- psf_centr1 & psf_centr2 (с ошибками): столбец и строка положения центроида PSF-модели (Point Spread Function), соответствующей целевой звезде;
- mom_centr1 & mom_centr2 (с ошибками): столбец и строка взвешенного по потоку положения центроида объекта. Дублируется в centroid_col и centroid_row, соответственно.

Доступ к этим столбцам можно получить так же, как и к свойствам объекта LightCurve, например:

```
lc.flux
```

```
[232.58354, 199.83601, 202.56972, ..., 175.95938, 161.19321, 194.28177]e^-s
```

Для загрузки нескольких кривых блеска в объекте search_result используется метод download_all(). Он возвращает объект LightCurveCollection в виде списка объектов LightCurve:

```
lc_collection=search_result.download_all()
```

```
LightCurveCollection of 2 objects:
0: <TessLightCurve LABEL = "TIC 273379660" SECTOR = 54 AUTHOR = SPOC FLUX_ORIGIN =
    pdcsap_flux>
1: <TessLightCurve LABEL = "TIC 273379660" SECTOR = 54 AUTHOR = SPOC FLUX_ORIGIN =
    pdcsap_flux>
```

Есть возможность визуализировать все наблюдения из полученной коллекции, используя метод `plot()` (рис. 12):

```
lc_collection.plot()
```

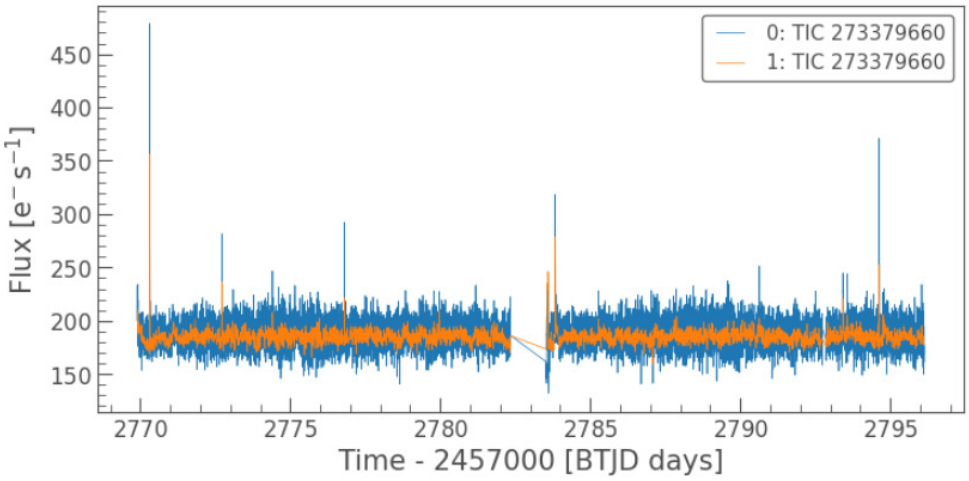

Рис. 12. Кривая блеска Kepler-862. Синим цветом показаны данные с экспозицией в 2 минуты, оранжевым – в 30 минут

5.2.1 Основные функции

В объектах `LightCurve` есть несколько полезных функций, которые можно использовать при работе с данными:

- `flatten()`: удалить долговременные тренды с помощью фильтра Савицкого – Голея;
- `remove_outliers()`: удалить выбросы, используя простое отсечение по сигма;
- `remove_nans()`: удалить бесконечные значения или значения `NaN` (могут образовываться во время включения двигателей), где `NaN` – отсутствие данных;
- `fold()`: свернуть данные с периодом;
- `bin()`: уменьшить временное разрешение массива, взяв среднее значение в каждом бине;
- `normalize()`: нормировать исходную кривую блеска.

Следует подробнее остановиться на указанных функциях, рассмотрев, как они работают и какие основные параметры принимают. Начнем с функции `remove_outliers()`, возвращающей новый объект `LightCurve`, из которого удалены все точки, отклоняющиеся на N · sigma от медианного значения потока (sigma(σ) – стандартное отклонение). Для `remove_outliers()` доступны следующие параметры:

- `sigma`: коэффициент N при сигма, по умолчанию N = 5. Используется как для верхнего, так и для нижнего предела отсечения;
- `sigma_lower/sigma_upper`: коэффициент N при сигма для использования в качестве нижней-/верхней границы предела отсечения. Можно установить значение `float('inf')`, чтобы полностью избежать отсечения выбросов ниже медианы. Если используется `None`, то принимается значение параметра sigma. По умолчанию `None`;
- `return_mask`: принимает значение `True/False`, по умолчанию `False`. Возвращает логический массив, отмечая данные, которые были удалены.

Для начала построим кривую блеска, которую загрузили ранее (рис. 13):

```
lc.plot()
```

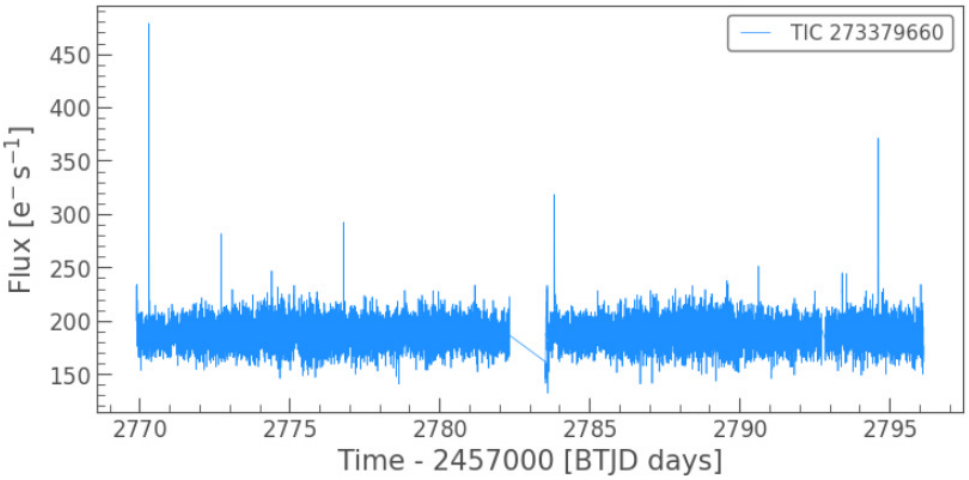

Рис. 13. Кривая блеска Kepler-862 по данным сектора 54 с 2-минутной экспозицией

Видно, что на кривой блеска присутствует достаточно много высокоамплитудных выбросов, которые можно убрать (рис. 14):

```
lc_new, lc_mask=lc.remove_outliers(sigma=3, return_mask=True)
lc_new.plot()
```

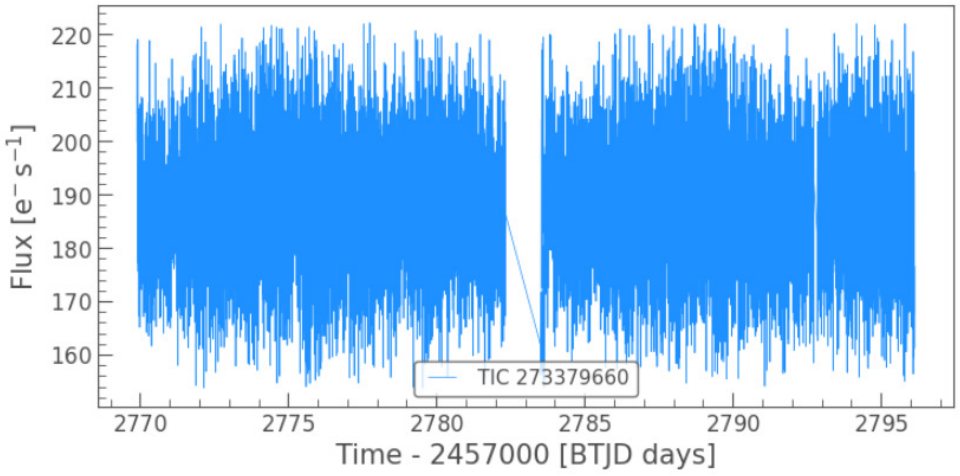

Рис. 14. Пример использования функции `remove_outliers()` с уровнем 3σ для Kepler-862

Мы получили новый объект `LightCurve` (`lc_new`), в котором удалены все выбросы, превышающие 3σ. Теперь посмотрим, что из себя представляет файл `lc_mask`:

```
lc_mask
```

```
array([True, False, False, ..., False, False, False])
```

В `lc_mask` параметром `True` отмечены точки кривой блеска, которые были исключены в результате sigma-отсечения. Файл `lc_mask` имеет такую же длину как `lc_new` или исходный `lc`.

Функция `remove_nans()` позволяет удалить бесконечные значения или значения `NaN`. В качестве единственного параметра эта функция принимает название столбца (`column = 'flux'`), в котором производится поиск значений NaN. По умолчанию значение `column = 'flux'`.

```
lc_rn=lc.remove_nans( column='pdcsap_flux')
```

Визуально эффект от работы `remove_nans()` не всегда заметен. Чтобы проверить его работу, необходимо сравнить размерность исходного и полученного файла. Видно, что размерность файлов отличается на одну единицу:

```
len(lc), len(lc_rn)
```

```
(17899, 17898)
```

Если необходимо уменьшить временное разрешение массива, используется функция `bin`, которая разбивает кривую блеска на равные временные интервалы. Если исходная кривая блеска содержит погрешности потока (`flux_err`), кривая блеска после бинирования будет содержать среднеквадратичную ошибку для каждого бина. Если погрешности отсутствуют, бинированная кривая блеска будет содержать стандартное отклонение данных. Функция `bin` принимает несколько параметров:

- **time_bin_size:** временной интервал (скалярное значение) для временного ряда. Предполагается, что все временные интервалы имеют одинаковую продолжительность. По умолчанию 0.5 дня;
- **n_bins:** количество используемых ячеек (бинов). По умолчанию используется число, необходимое для учета всех точек на кривой блеска. Следует обратить внимание на то, что использование этого параметра создаст указанное количество бинов размерностью `time_bin_size` независимо от длины исходной кривой блеска;
- **bins:** принимает целое число, на которое нужно разделить кривую блеска. В отличие от `n_bins` этот параметр регулирует длину (`time_bin_size`).

Для большей наглядности применения функции `bin`, а также `flatten` и `fold`, загрузим кривую блеска звезды HAT-P-6, у которой также обнаружена экзопланета:

```
search_result=lk.search_lightcurve ('HAT-P 6', mission='TESS', exptime=120)
lc=search_result[0].download()
```

На кривой блеска HAT-P-6 видны выраженные падения блеска/потока (транзиты) (рис. 15):

```
lc.plot()
```

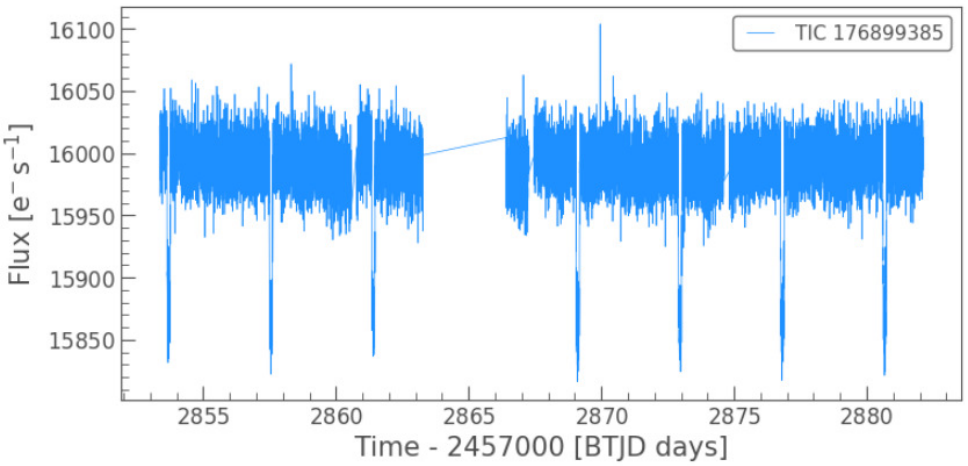

Рис. 15. Кривая блеска HAT-P-6 по данным TESS с 2-минутной экспозицией

Применим описанную ранее функцию `bin` к полученному графику (рис. 16):

```
lc_bin=lc.bin(time_bin_size=0.01)
lc_bin.plot()
```

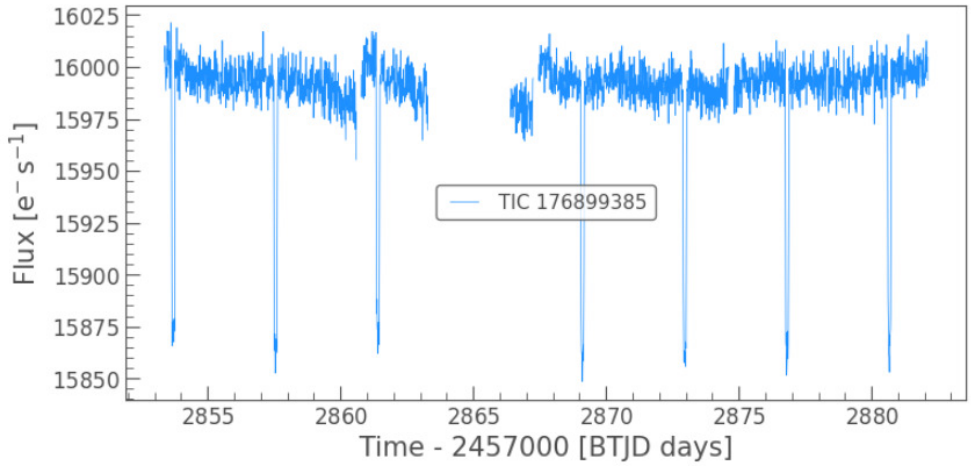

Рис. 16. Кривая блеска HAT-P-6 после бинирования с окном 0.01

Учет долговременного тренда на кривой блеска выполняется с использованием функции `flatten`. Удаление тренда происходит с помощью метода Савицкого – Голея. Для настройки функции `flatten` доступны некоторые параметры:

- `window_length:` длина окна, должна быть целым нечетным числом;
- `polyorder:` степень полинома, должен быть меньше window_length;
- `return_trend:` принимает значение `True/False`. В случае `True` возвращает дополнительный объект `LightCurve`, содержащий удаленный тренд;
- `break_tolerance:` принимает целое число в единицах времени. Если на кривой блеска присутствуют длительные разрывы в данных, то этот параметр позволяет разделить исходную кривую

блеска на несколько частей и к каждой применить метод Савицкого – Голея. Разрыв определяется как период времени, превышающий значение `break_tolerance`, умноженное на среднее значение временного интервала между данными. Для отключения функции устанавливается значение `None`;

- `niters`: принимает целое число. Это количество итераций для многократного `sigma`-отсечения и выравнивания. Если `niters` больше одного, сглаживание будет выполняться несколько раз, каждый раз удаляя выбросы;
- `sigma`: значение `sigma`, выше которого будут удаляться все выбросы;
- `mask`: логический массив для маскирования данных. Значения потока, где маска равна `True`, не будут использоваться для выравнивания данных. Для этих точек будет предоставлен интерполированный результат. Нужно использовать эту маску, чтобы убрать из рассмотрения данные, которые необходимо сохранить, например транзиты.

На кривой блеска HAT-P-6 (рис. 15) видны некоторые низкочастотные изменения блеска. Следует применить функцию `flatten` для их удаления (рис. 17):

```
lc_flatten=lc.flatten(window_length=201, polyorder=2, niters=3)
lc_flatten.plot()
```

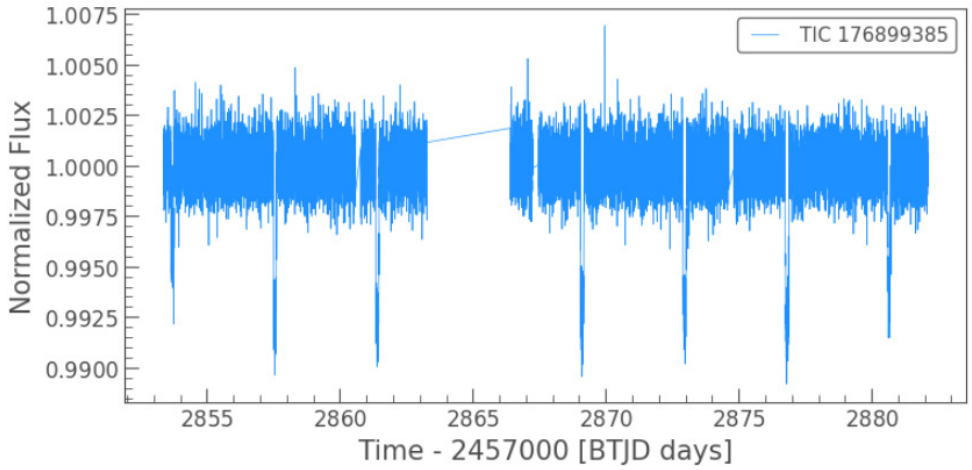

Рис. 17. Результат применения метода `flatten()` для учета тренда кривой блеска HAT P-6

При необходимости можно нормировать кривую блеска, используя функцию `normalize()`. Она возвращает нормированную кривую блеска (рис. 18), полученную путем деления потока на его среднее значение:

```
lc_norm=lc.normalize()
lc_norm.plot()
```

Рассмотрим последнюю из вышеуказанных функций – `fold()`, которая позволяет свернуть исходную кривую блеска с периодом. В библиотеке `Lightkurve` есть возможность периодограммного анализа, но она будет рассмотрена позднее, в разделе 5.2.2. Сейчас при демонстрации этой функции будем считать период известным и равным 3.852980 дня[10]. Рассмотрим параметры, которые принимает функция:

[10] https://exoplanetarchive.ipac.caltech.edu/

- `period`: значение периода, выраженного в днях;
- `epoch_time`: время, используемое в качестве нуль-пункта. По умолчанию используется первая точка во временном ряду;
- `epoch_phase`: значение фазы для `epoch_time`;
- `wrap_phase`: значение фазы, выше которого все значения переносятся на один период назад. Если `normalize_phase` указано `True`, то параметр `wrap_phase` должен указываться в безразмерных единицах, в противном случае – в единицах времени; `normalize_phase` принимает значение `False/True`. Если указано `False`, то значения фазы возвращаются в днях, если `True` – в безразмерных единицах.

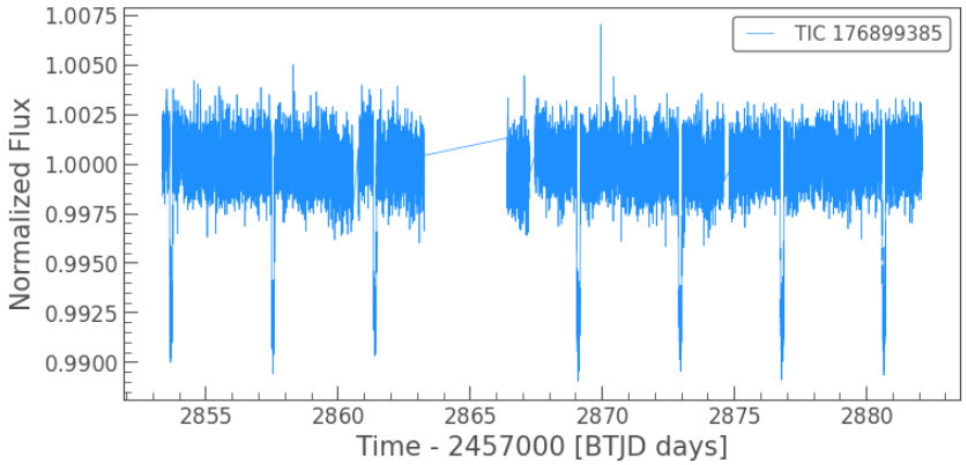

Рис. 18. Нормированная кривая блеска HAT-P-6

Свернем кривую блеска с известным периодом (рис. 19), применив функцию `fold()`:

```
lc_phase=lc.fold(period=3.853003, epoch_time=2869.1, epoch_phase=0, normalize_phase
    =True)
lc_phase.plot()
```

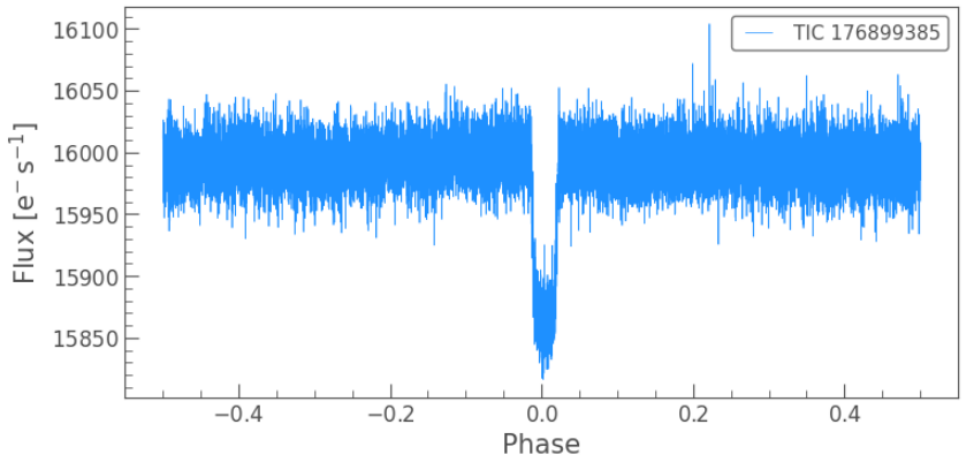

Рис. 19. Фазовая кривая блеска HAT-P-6

Возможно применение сразу всех (по необходимости) рассмотренных выше функций объекта `LightCurve` (рис. 20):

```
lc_total=lc.remove_nans().bin(time_bin_size=0.01).flatten(window_length=201,
    polyorder=2, niters=3).fold(period=3.853003, epoch_time=2869.1, epoch_phase=0,
    normalize_phase=True).plot()
```

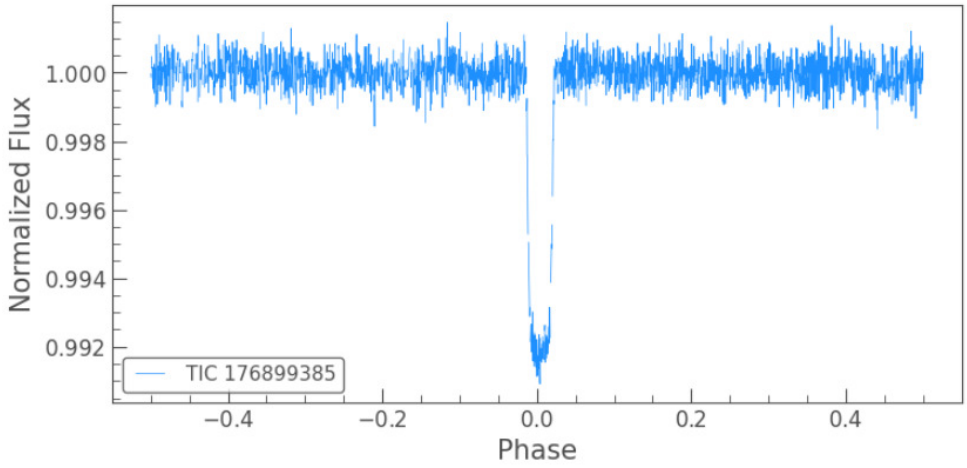

Рис. 20. Фазовая кривая блеска HAT-P-6 после бинирования и учета тренда

У объектов `LightCurve` есть еще несколько функций, которые могут оказаться полезными. Функция `truncate()` позволяет обрезать кривую блеска в указанных временных интервалах. `Truncate()` принимает параметры `before, after` и `column`, в которые передается левая и правая временная граница интервала, а также указывается имя столбца, по которому происходит обрезка. По умолчанию используется столбец `'time'`.

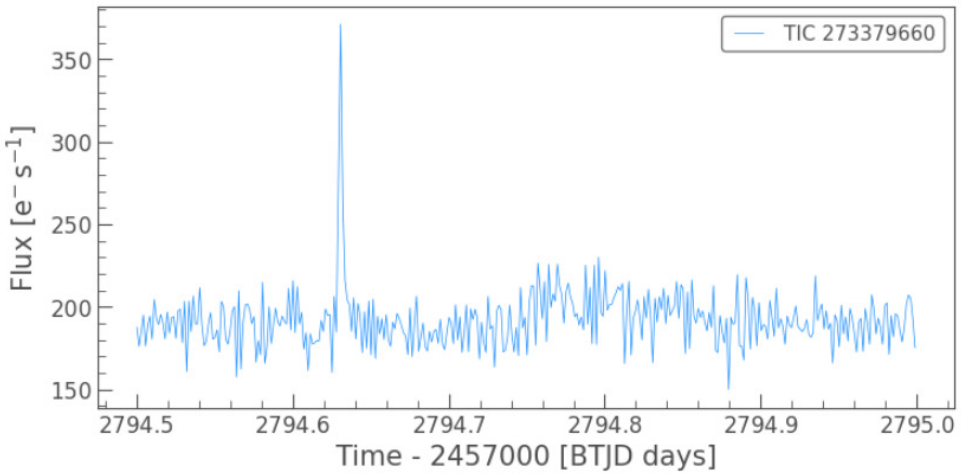

Рис. 21. Фрагмент кривой блеска Kepler-862 с высокоамплитудным событием

Применим эту функцию для выделения некоторых событий на кривой блеска, например, звезды Kepler-862. После загрузки кривой блеска видно, что имеются события, похожие на вспышки (рис. 13). Укажем примерный диапазон со вспышкой: 2794.5–2795. На рис. 21 показан результат применения функции `truncate()`: событие, похожее на вспышку, расположено около 2459794.63 BTJD.

```
lc_flares=lc.truncate(before=2794.5, after=2795.2)
lc_flares.plot()
```

Функция `fill_gaps()` позволяет восполнить пробелы в данных (см., например, рис. 13). По умолчанию промежутки заполняются белым гауссовым шумом ($\mu = \overline{flux}$, $\sigma = CDPP$). В настоящее время другие типы шумов не поддерживаются. Применим данную функцию к кривой блеска Kepler-862 (рис. 22):

```
lc\_gaps=lc.fill_gaps(method='gaussian_noise')
lc\_gaps.plot()
```

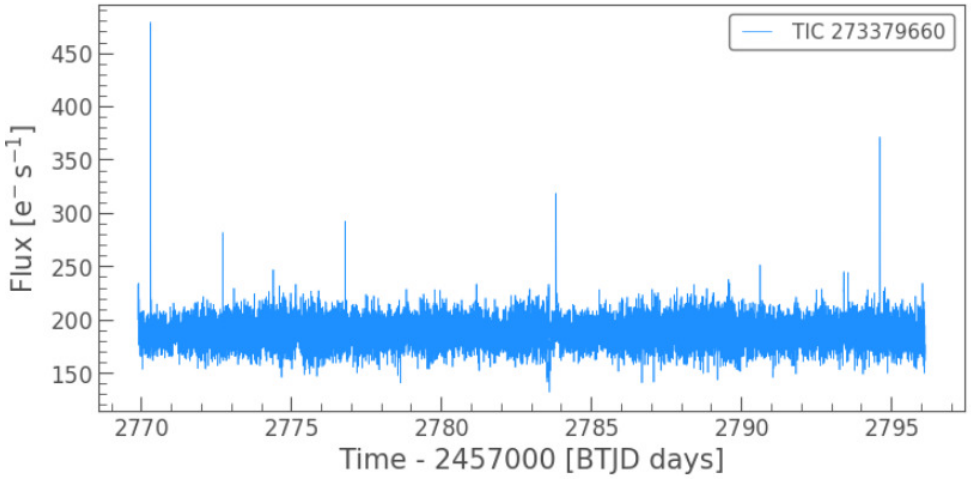

Рис. 22. Результат применения функции `fill_gaps()` к кривой блеска Kepler-862

5.2.2 Периодограммный анализ

В `Lightkurve` есть класс, который специально предназначен для периодограммного анализа временных рядов. Пользователю доступны два метода: `LombScarglePeriodogram` (Lomb, 1976; Scargle, 1982) и `BoxLeastSquaresPeriodogram` (Kovacs, 2002; Hartman, 2016). Начнем с рассмотрения метода `LombScarglePeriodogram` (сокращенно `ls`) и доступных параметров для его настройки:

- `minimum_frequency/maximum_frequency`: указывается минимальная/максимальная частота;
- `minimum_period/maximum_period`: указывается минимальный/максимальный период, выраженный в днях;
- `frequency`: набор частот. Если задана единица измерения, она преобразуется в единицы `freq_unit`. Если нет, то предполагается, что значения выражены в единицах `freq_unit`. Использование этого параметра отменяет любые установленные ограничения частоты;
- `period`: набор периодов. По умолчанию предполагается, что значения выражены в единицах `1/freq_unit`. Использование этого параметра отменяет любые установленные ограничения периода;
- `nyquist_factor`: по умолчанию используется 1. Кратный множитель средней частоты Найквиста. Используется для выбора максимальной частоты, если она не указана;
- `normalization`: по умолчанию `'amplitude'`. Желаемая нормализация спектра может быть либо спектральной плотностью мощности (`'psd'`), либо амплитудой (`'amplitude'`).

ВАЖНО: метод `LombScarglePeriodogram` может не работать с некоторыми версиями библиотеки `astropy`. При возникновении ошибки рекомендуется установить версию `astropy = 5.2.2` путем ввода в командной строке команды `pip install astropy==5.2.2 --user`.

При использовании метода `BoxLeastSquaresPeriodogram` (сокращенно `bls`) доступны следующие параметры:

- `duration`: набор значений продолжительности, которые будут учитываться. По умолчанию принимаются значения [0.05, 0.10, 0.15, 0.20, 0.25, 0.33], выраженные в днях;
- `period`: периоды, для которых должна быть рассчитана периодограмма. Если период не указан, будет создано значение по умолчанию;
- `minimum_period, maximum_period`: если `period` не указан, то алгоритм ищет периоды в пределах минимального/максимального значения, установленного по умолчанию. Значения по умолчанию определяются как

```
maximum period=(max(lc.time) - min(lc.time))/3

minimum period=max[median(diff(lc.time)) · 4,
max(duration) + median(diff(lc.time))]
```

- **frequency_factor:** если значение параметра `period` не указано, то следует учитывать коэффициент для регулирования частотного интервала между периодами.

Рассмотрим применение методов периодограммного анализа на примере звезды HAT-P-6. Ранее (рис. 15) было описано получение кривой блеска этого объекта и его график, поэтому пропустим эту часть в данном разделе. На кривой блеска отчетливо виден сигнал, связанный с транзитом экзопланеты. Для создания периодограммы из объекта `LightCurve` необходимо применить функцию `to_periodogram`. В нее передается название используемого метода (`LombScarglePeriodogram` или `BoxLeastSquaresPeriodogram`), а также соответствующие методу параметры:

```
lc_pg=lc.to_periodogram(method='ls', minimum_period=2, maximum_period=10,
    oversample_factor=1000)
lc\_pg.plot()
```

Поскольку известно, что период, связанный с транзитом экзопланеты, составляет ~ 3.8 дня, то при построении спектра мощности указано минимальное и максимальное значение периода вблизи истинного (рис. 23).

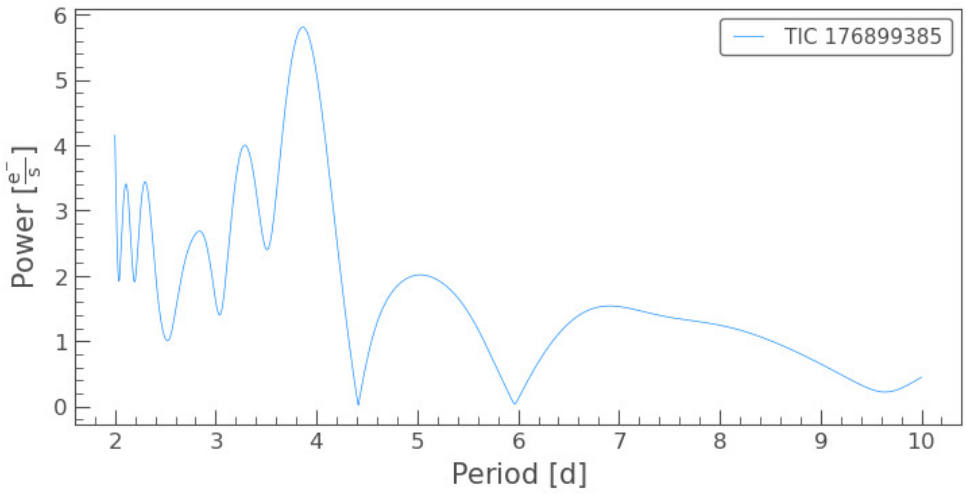

Рис. 23. Спектр мощности с использованием метода `LombScarglePeriodogram` для рассматриваемой кривой блеска HAT-P-6

Важный момент: если в качестве параметров передаются ограничения на период, то по умолчанию строится спектр мощности относительно периода. Если не указывать ограничения на период

либо указать частотные параметры, будет построен спектр мощности относительно частоты. Существует возможность явным образом указать, какой график необходимо построить, указав это в функции `plot()`, например `lc_pg.plot(view = 'frequency', scale = 'log')`.

Доступ к полным данным о периоде и мощности реализуется следующим образом:

```
lc_pg.period()
```

```
[10, 9.9965241, 9.9930506, ..., 2.0003869, 2.0002478, 2.0001086]d
```

```
lc_pg.power()
```

```
[0.4466488, 0.44335964, 0.44008012, ..., 4.1303653, 4.142025, 4.1536879]e^-s
```

В данном случае необходимо знать период, который соответствует самому высокому пику на периодограмме. Чтобы получить прямой доступ к этому значению, используется свойство `period_at_max_power`:

```
lc_pg.period_at_max_power
```

```
3.8640166 d
```

Как видно, найденный период достаточно близок к периоду P = 3.852980 дня, взятому из базы данных NASA по экзопланетам[11]. Теперь необходимо использовать вычисленный период, чтобы создать фазовую кривую блеска (рис. 24), используя функцию `fold()`:

```
lc_phase=lc.fold(period=lc_pg.period_at_max_power, epoch_time=2869.1, epoch_phase
    =0, normalize_phase=True)
lc_phase.plot()
```

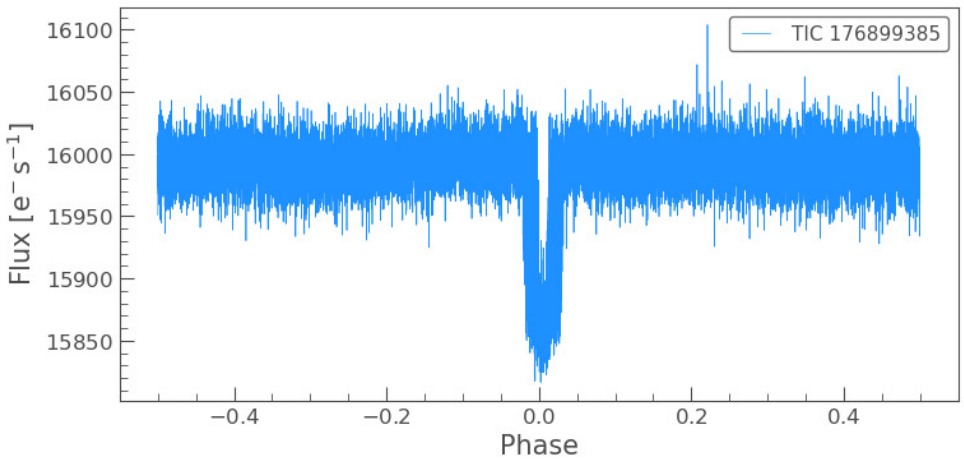

Рис. 24. Фазовая кривая блеска с периодом 3.8640166 дня (метод `LombScarglePeriodogram`)

[11] https://exoplanetarchive.ipac.caltech.edu/

Аналогично предыдущему методу применим к кривой блеска метод `BoxLeastSquaresPeriodogram` (`'bls'`) (рис. 25):

```
lc_pg_bls=lc.to_periodogram(method='bls', minimum_period=2, maximum_period=10)
lc_pg.plot()
```

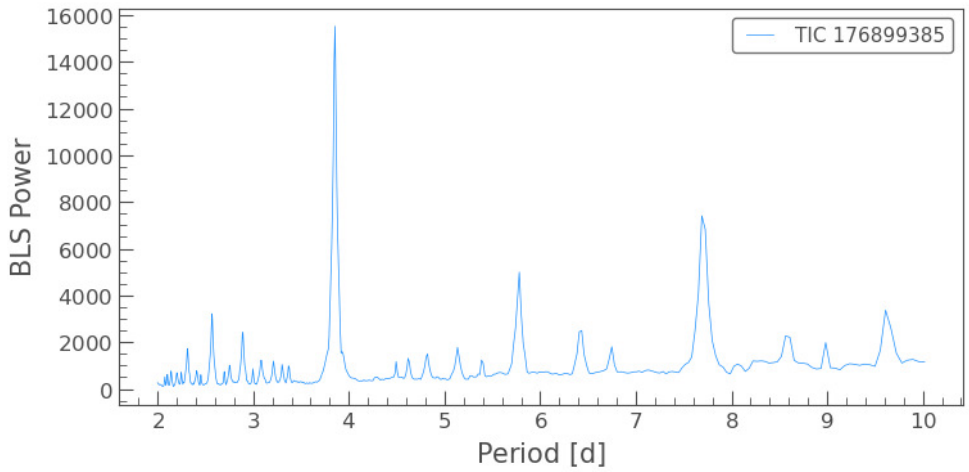

Рис. 25. Спектр мощности с использованием метода `BoxLeastSquaresPeriodogram` для рассматриваемой кривой блеска HAT-P-6

Используя свойство `period_at_max_power`, определяем значение периода, равное 3.855019 дня. Полученное в данном случае с помощью метода `BoxLeastSquaresPeriodogram` значение периода показывает лучшее соответствие с каталогом[12]. Для получения фазовой кривой блеска используется описанный выше алгоритм. В данном случае это объясняется тем, что метод `bls` специально разработан для обнаружения экзопланет по их транзитам, кривая блеска которых имеет ступенчатую форму. В то время как метод `ls` ориентирован на обнаружение синусоидальных периодических сигналов.

5.2.3 Удаление сигнала периода вращения из кривой блеска

Рассмотрим звезды с активностью солнечного типа. Одним из признаков их магнитной активности является наличие звездных пятен, которые проявляются в виде периодических изменений блеска звезды. Из анализа изменения яркости можно определить период вращения звезд. В некоторых задачах возникает необходимость очистить кривую блеска от переменности, вызванной наличием пятен.

После загрузки данных для звезды TAP 26 нормируем кривую блеска, используя функцию `normalize()`, и строим график получившейся кривой блеска (рис. 26):

```
search_result=lk.search_lightcurve('TAP 26', mission='TESS', author='SPOC')
lc=search_result[0].download()
lc_norm=lc.normalize()
lc_norm.plot()
```

Следующим шагом получаем спектр мощности, используя метод `LombScarglePeriodogram` (см. раздел 5.2.2). Ограничим диапазон поиска возможного периода в три дня, поскольку из кривой блеска видно, что период вращения намного короче этого значения. Установим для удобства по

[12] https://exoplanetarchive.ipac.caltech.edu/

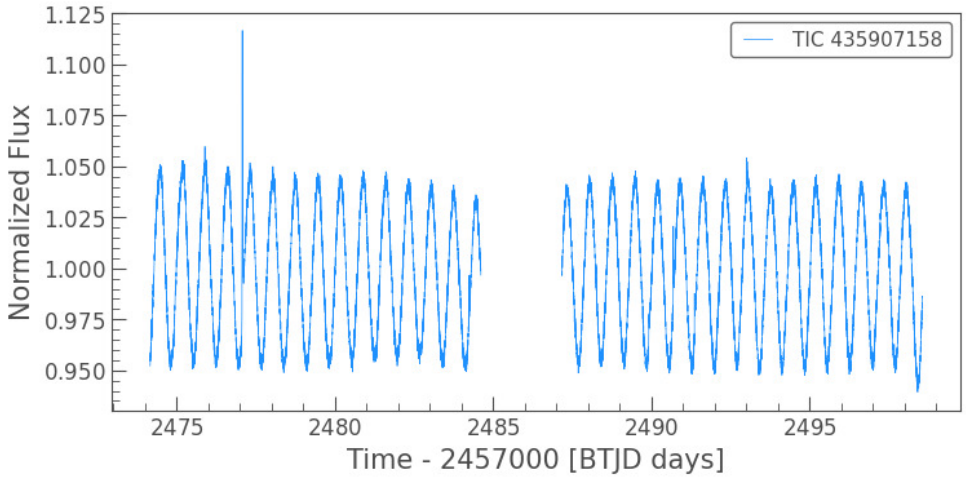

Рис. 26. Нормированная кривая блеска TAP 26 по данным сектора 43 с 120-секундным временным разрешением

оси X единицы измерения периода, используя параметр `view=period` (рис. 27). Кроме этого определим период, который соответствует самому высокому пику на периодограмме, используя свойство `period_at_max_power`:

```
lc_pg=lc_norm.to_periodogram(method='ls', maximum_period=3)
lc_pg.plot(view='period')
lc_pg.period_at_max_power
```

```
0.71485847 d
```

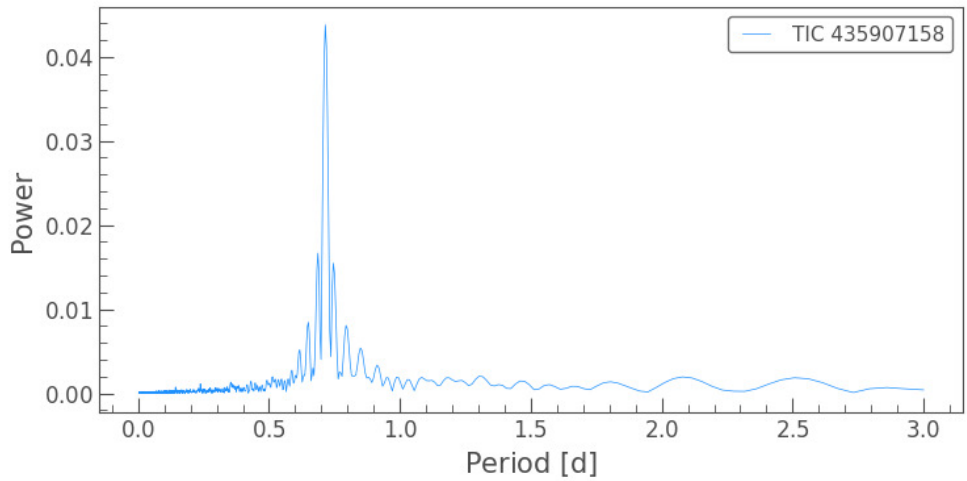

Рис. 27. Спектр мощности с использованием метода `LombScarglePeriodogram` для кривой блеска TAP 26

Как уже отмечалось ранее, в `Lightkurve` используется метод Ломба – Скаргла для периодограммного анализа. Извлечем модель Ломба – Скаргла, соответствующую самому высокому пику периодограммы, и соотнесем с кривой блеска объекта (рис. 28). Для этого используется метод `model()`, в котором необходимо указать значение параметров `time` и `frequency`:

```
lc_model=lc_pg.model(time=lc_norm.time, frequency=lc_pg.frequency_at_max_power)
ax=lc_norm.plot()
lc_model.plot(ax=ax, lw=2, ls='--', c='red')
```

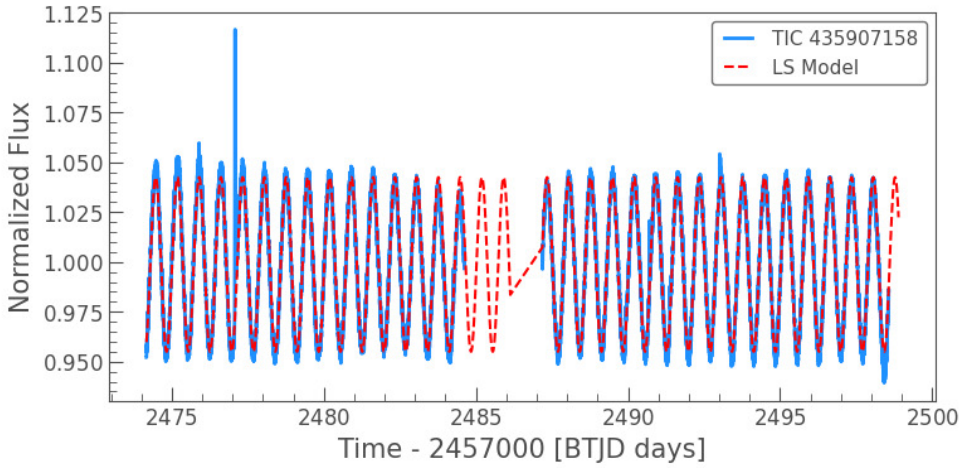

Рис. 28. Синяя линия – нормированная кривая блеска TAP 26. Красная пунктирная линия – модель Ломба – Скаргла, соответствующая самому высокому пику периодограммы

Из рис. 28 видно, что модель Ломба – Скаргла относительно хорошо описывает исходную кривую блеска. Стоит помнить, что такой подход определяет частоту колебаний с некоторой ошибкой. Эта неопределенность отражает тот факт, что сигнал вращения не является идеальной синусоидой, и, кроме этого, в данных измерений существует дополнительный шум от звезды. Наконец, удалим из кривой блеска сигнал, связанный с вращением звезды. Для этого необходимо разделить кривую блеска на модель (рис. 29):

```
lc_new=lc_norm.copy()
lc_new.flux=lc_new.flux / lc_model.flux
lc_new.plot()
```

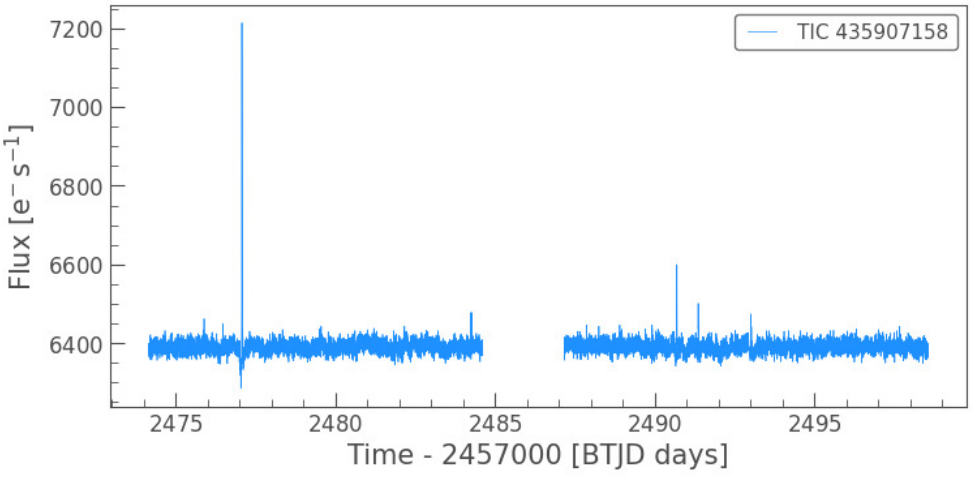

Рис. 29. Кривая блеска TAP 26 после учета сигнала, связанного с вращательной модуляцией блеска

Чаще всего для достижения нужного результата необходимо произвести несколько итераций описанного выше алгоритма. Ниже приведен краткий список последовательности действий и фрагмент кода, повторяющий эту процедуру 10 раз:

- Вычисление периодограммы.
- Построение модели Ломба – Скаргла, соответствующей максимальному пику на спектре мощности.
- Деление данных исходной кривой блеска на модель.
- Повторение действий для новой кривой блеска.

```
lc_new=lc_norm.copy()
for i in range(10):
    lc_pg=lc_new.to_periodogram()
    model=lc_pg.model(time=lc_new.time, frequency=lc_pg.frequency_at_max_power)
    lc_new.flux=newlc.flux / model.flux

ax=lc_norm.plot(alpha=0.5, color='blue', label='Original');
lc_new.plot(ax=ax, color='red', label='New');
```

Здесь важно отметить, что итеративная аппроксимация синусоидальной функцией, используемая в этом разделе, имеет некоторые недостатки. Скорее всего кривая блеска не имеет идеальной синусоидальной формы. Это означает, что с каждой итерацией в кривую блеска могут быть добавлены сложные остаточные структуры и ложные пики в периодограмму. Поэтому следует проявлять внимательность при обработке кривых блеска подобными алгоритмами.

5.2.4 Сохранение и загрузка объекта LightCurve

Для сохранения кривой блеска в библиотеке `Lightkurve` есть возможность конвертировать объект `LightCurve` в различные форматы: `csv`, `fits` и объект библиотеки `pandas (DataFrame)`. По умолчанию данные сохраняются в директорию, где расположен файл с кодом программы. Преобразование кривой блеска в таблицу формата `DataFrame` осуществляется с помощью метода `to_pandas()`:

```
lc=lc.to_pandas()
```

При последующем сохранении файла в формате `csv` используются стандартные функции библиотеки `pandas`:

```
lc.to_csv('lightcurve_name.csv')
```

Примечание: при сохранении файла в формате `csv` не следует использовать функцию удаления индексов (`index = None`), поскольку в качестве столбца индексов выступает столбец `'time'`. Его удаление приведет к тому, что в сохраненном файле не будет привязки данных ко времени.

Можно сразу сохранить кривую блеска в csv-формате, применяя метод `to_csv()`. Он использует один параметр `path_or_buf`, в который передается путь сохранения файла:

```
lc.to_csv(path_or_buf='lightcurve_name.csv')
```

Наконец, возможно сохранить объект `LightCurve` в `fits`-формате с помощью метода `to_fits()`. Этот метод использует несколько параметров:

- `path`: указывается путь сохранения файла;
- `overwrite`: перезапись файла. Принимает значение `True/False`;
- `other`: можно указать дополнительные столбцы для включения в `fits`-файл (например, включить столбец, содержащий `pdcsap_flux` : PDCSAP_FLUX= `lc.pdcsap_flux`, где PDCSAP_FLUX – присваиваемое название столбца).

```
lc.to_fits(path='lightcurve_name.fits', overwrite=True, FLUX=lc.flux, FLUX_ERR=lc.
    flux_err, SAP_FLUX=lc.sap_flux, SAP_FLUX_ERR=lc.sap_flux_err, PDCSAP_FLUX=lc.
    pdcsap_flux, PDCSAP_FLUX_ERR=lc.pdcsap_flux_err)
```

Для сохранения графиков используются стандартные методы графических библиотек `python` (`matplotlib`, `seaborn` и т. п.), например:

```
fg=lc.plot()
fg.figure.savefig('lightcurve.png')
```

Существует возможность вручную загрузить `fits`-файлы кривой блеска из архива, сохранить их на локальном диске компьютера и открыть с помощью функции `lk.read(<filename>)`. Вместо `<filename>` необходимо указать полный путь к загружаемой кривой блеска. Узнать, где библиотека `Lightkurve` сохранила файл кривой блеска, можно с помощью атрибута `lc.filename`:

```
lc.filename
```

```
'C:\\Users\\XXX\\.lightkurve\\cache/mastDownload\\TESS\\tess2022273165103-s0057
    -0000000176899385-0245-s\\tess2022273165103-s0057-0000000176899385-0245-s_lc.
    fits'
```

```
lc_new=lk.read('C:\\Users\\XXX\\.lightkurve\\cache/mastDownload\\TESS\\
    tess2022273165103-s0057-0000000176899385-0245-s\\tess2022273165103-s0057
    -0000000176899385-0245-s_lc.fits')
```

5.3 Объекты TargetPixelFile

Другим основным продуктом данных, используемым в библиотеке `Lightkurve`, является `TargetPixelFile` (TPF). TPF представляет собой массив изображений, содержащих исследуемый объект и некоторую область фона. Подобно описанному выше подходу (см. раздел 5.2), для поиска имеющихся данных наблюдений используется метод `search_targetpixelfile`. Этот метод возвращает таблицу, которая содержит ту же информацию, что и результат поиска кривой блеска. Важно понимать, что метод `search_targetpixelfile` возвращает только те данные TPF, для которых есть кривые блеска, полученные инструментами обработки данных TESS с временным разрешением 1800 секунд. Для более поздних секторов TPF данные доступны с более высоким временным разрешением. Рассмотрим возможности работы с TPF на примере звезды HAT-P-6. Для начала загрузим список доступных данных (рис. 30):

```
search_result=lk.search_targetpixelfile('HAT-P 6', exptime='long')
search_result
```

SearchResult containing 2 data products.

#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	TESS Sector 16	2019	TESS-SPOC	1800	176899385	0.0
1	TESS Sector 17	2019	TESS-SPOC	1800	176899385	0.0

Рис. 30. Результат поискового запроса TPF данных для HAT-P-6

В зависимости от задачи необходимо указать конкретные параметры для поиска информации. Чтобы получить аннотацию к функции `search_targetpixelfile` и узнать, какие параметры она принимает, используется метод `__signature__`:

```
lk.search_targetpixelfile.__signature__
```

```
<Signature (target, radius=None, exptime=None, cadence=None, mission=('Kepler', 'K2
    ', 'TESS'), author=None, quarter=None, month=None, campaign=None, sector=None,
    limit=None)>
```

Когда применяется метод `download()` для результата поиска `search_result`, который содержит более одной записи, будет загружена только первая запись, и `Lightkurve` выдаст предупреждение о том, что для загрузки всех данных необходимо воспользоваться методом `download_all()` (рис. 31):

```
tpf=search_result.download()
```

```
C:\Users\mgorb\anaconda3\lib\site-packages\lightkurve\search.py:414: LightkurveWarning: Warning: 4 files available to download.
Only the first file has been downloaded. Please use `download_all()` or specify additional criteria (e.g. quarter, campaign, or
sector) to limit your search.
  warnings.warn(
```

Рис. 31. Уведомление о неправильно используемой команде для загрузки данных

Аналогично загрузке объектов `LightCurve` можно выбрать конкретный сектор, указав его номер. Для отображения загруженного TPF используется метод `plot()`. По умолчанию возвращается изображение, соответствующее первому кадру в данном секторе. Чтобы получить изображение TPF для другого момента времени, необходимо указать его индекс (пример, `tpf[57].plot()`, рис. 32).

```
tpf=search_result[0].download()
tpf.plot()
```

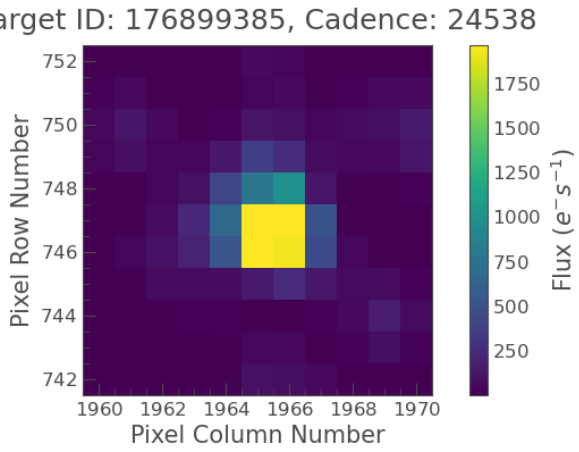

Рис. 32. Пример одного изображения TPF для HAT-P-6 с 30-минутной экспозицией

TPF-файлы также содержат метаданные с подробным описанием процесса наблюдений и информацией о постобработке, такой как предполагаемая интенсивность фона на каждом изображении. Доступ к метаданным осуществляется с помощью вызова свойства `meta` или `get_header()` (рис. 33):

```
tpf.get_header()
```

```
SIMPLE  =                    T / conforms to FITS standards
BITPIX  =                    8 / array data type
NAXIS   =                    0 / number of array dimensions
EXTEND  =                    T / file contains extensions
NEXTEND =                    3 / number of standard extensions
EXTNAME = 'PRIMARY '           / name of extension
EXTVER  =                    1 / extension version number (not format version)
SIMDATA =                    F / file is based on simulated data
ORIGIN  = 'NASA/Ames'          / institution responsible for creating this file
DATE    = '2020-06-22'         / file creation date.
TSTART  =    1738.671032253282 / observation start time in BTJD
TSTOP   =    1763.317287847625 / observation stop time in BTJD
DATE-OBS= '2019-09-12T04:05:08.003' / TSTART as UTC calendar date
DATE-END= '2019-10-06T19:35:44.486' / TSTOP as UTC calendar date
CREATOR = '26042_TargetPixelExporterPipelineModule' / pipeline job and program u
```

Рис. 33. Фрагмент вывода метаданных

Возможно получение доступа к значению потоков конкретного пикселя обрабатываемого TPF-файла. Для каждого кадра TPF имеет ряд свойств данных фотометрии:

- **flux:** поток от звезды после удаления фона;
- **flux_err:** статистическая неопределенность измерений звездного потока после удаления фона;
- **flux_bkg:** фоновый поток изображения;
- **flux_bkg_err:** статистическая неопределенность фонового потока.

Сам TPF-файл представлен в виде массива, содержащего значения указанных выше параметров. Ниже приведен пример вывода значений параметра `flux`.

```
tpf.flux[0].value
```

```
array([[-4.20636463e+00, -1.02202692e+01, -1.23759518e+01, -6.93972063e+00,
    1.58345280e+01,  8.94959793e+01, 7.28748932e+01,  4.53632469e+01,  1.47184741e
    +00, -1.26764393e+01, -9.94957733e+00], ...
```

Построение изображения для конкретного типа данных (рис. 34):

```
tpf.plot(column='flux_err')
```

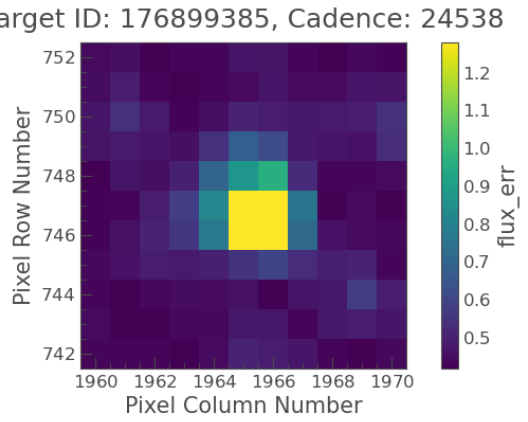

Рис. 34. Пример распределения ошибок при определении потока в виде TPF-кадра

Далее можно преобразовать последовательность TPF-данных в кривую блеска (рис. 35), создав объект `LightCurve`. Для этого необходимо применить метод `to_lightcurve()`. В данном случае для получения кривой блеска используются пиксели, которые алгоритм автоматической обработки данных TESS считает более оптимальными (апертура `'pipeline'`, см. раздел 5.3.1).

```
lc=tpf.to_lightcurve()
lc.plot()
```

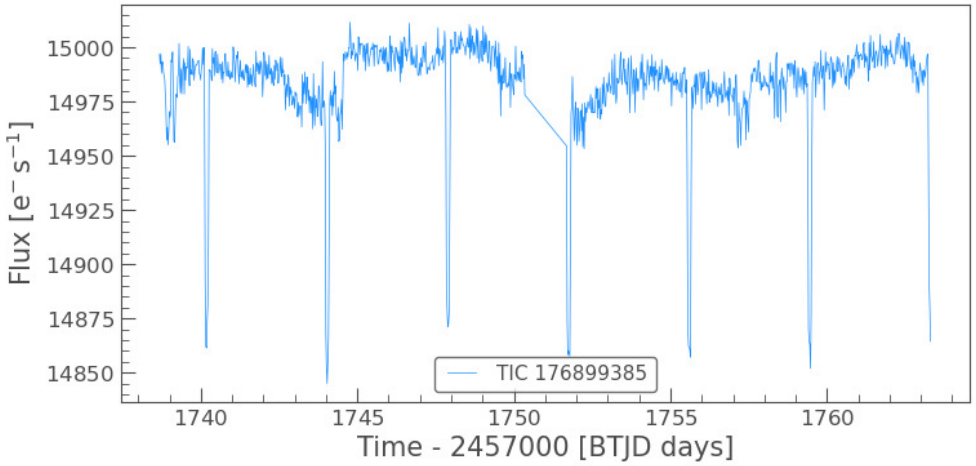

Рис. 35. Кривая блеска HAT-P-6, полученная из TPF-файлов сектора 16

Посмотрим, что из себя представляет таблица `lc` (рис. 36).

```
lc
```

TessLightCurve length=1116 LABEL="TIC 176899385" SECTOR=16

time	flux	flux_err	centroid_col	centroid_row	cadenceno	quality
	electron / s	electron / s	pix	pix		
Time	float32	float32	float64	float64	int32	int32
1738.681449160407	14996.7177734375	3.79941987991333	1965.387080465247	746.8548101731236	24538	0
1738.7022832066036	14992.322265625	3.800065517425537	1965.3873188470964	746.8540124285132	24539	0
1738.7231172500055	14996.99609375	3.799346685409546	1965.387384381143	746.8544783976596	24540	0
1738.7439512910794	14988.1005859375	3.799614191055298	1965.3874843758297	746.853305569779	24541	0
1738.7647853298247	14993.3505859375	3.7996654510498047	1965.3868411540636	746.8536404393678	24542	0
1738.7856193662424	14993.185546875	3.800194263458252	1965.3875532391114	746.8536957760662	24543	0

Рис. 36. Таблица для HAT-P-6, полученная из TPF-файлов сектора 16

Теперь можно работать с данными как с объектом `LightCurve` и применять все функции из раздела 5.2.1. В качестве примера выполним периодограммный анализ и свернем кривую блеска с найденным периодом (рис. 37):

```
lc_bls=lc.to_periodogram(method='bls', minimum_period=2, maximum_period=10)
lc_bls.period_at_max_power
```

```
3.8574389 d
```

```
lc_phase=lc.fold(period=lc_bls.period_at_max_power, epoch_time=1744, epoch_phase=0,
    normalize_phase=True)
lc_phase.plot()
```

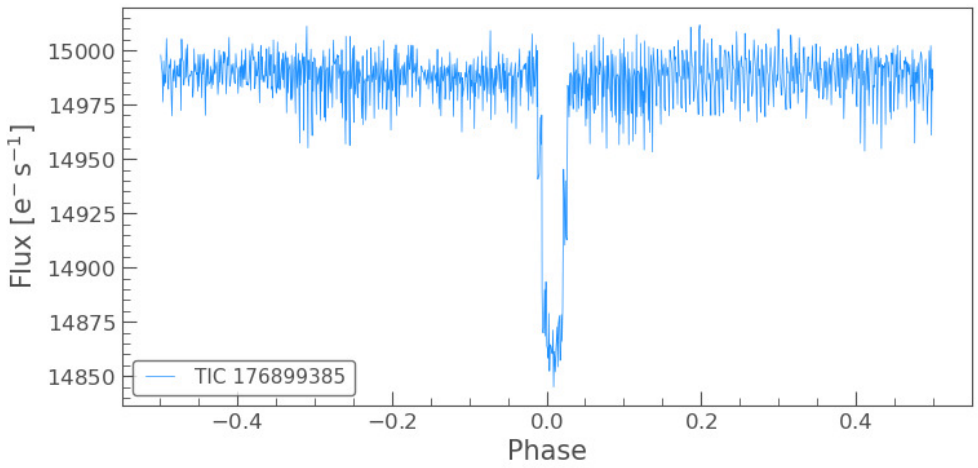

Рис. 37. Фазовая кривая блеска HAT-P-6

Как видно из примера, полученный период близок к периоду, определенному ранее (см. раздел 5.2.2). Несколько большее отклонение от каталожного значения периода обусловлено временным разрешением данных. В этом случае время экспозиции составляет 1800 секунд (по сравнению со 120 сек. в данных из раздела 5.2.2).

5.3.1 Фотометрия с пользовательской апертурой

В рамках алгоритма, который выполняют официальные инструменты обработки данных TESS, каждый TPF содержит рекомендуемую "оптимальную маску апертуры" (`pipeline_mask`). Эта маска оптимизирована таким образом, чтобы сигнал от объекта имел наибольшее соотношение сигнал/шум с минимальным вкладом фона. Есть много причин, по которым может возникнуть потребность в создании собственной кривой блеска с использованием пользовательской апертуры. Например, в плотных полях звезд в апертуру `pipeline` могут попадать несколько объектов, что скажется на кривой блеска. В этом случае есть необходимость построить собственную кривую блеска, используя меньшую апертуру, извлекая сигнал только от исследуемого объекта.

Другой причиной, по которой возникает необходимость создать собственные кривые блеска, является отсутствие доступных данных `SAP/PDCSAP` для некоторых объектов. В поле зрения TESS есть сотни тысяч звезд, которые не имеют готовых кривых блеска. В таком случае фотометрия с пользовательской апертурой идеально подходит для получения кривых блеска конкретных объектов из полнокадровых изображений (см. раздел 5.4).

Выбор наилучшей апертуры для фотометрии является нахождением баланса между уменьшением влияния изменений PSF, а также движения космической обсерватории (большая апертура) и уменьшением "дробового шума" (меньшая апертура). Поэтому данные Kepler/K2 и TESS обрабатываются с "оптимальной апертурой". Яркость, PSF, дробовой шум, плотность звезд в поле и многие другие факторы учитывались при моделировании "оптимальной апертуры" для получения кривой блеска `SAP` или `PDCSAP`. Рассмотрим, как выглядит апертура, используемая инструментами обработки данных TESS. Для этого загрузим TPF для звезды HAT-P-6 и применим апертуру `pipeline` (рис. 38):

```
search_result=lk.search_targetpixelfile('HAT-P 6', exptime='long')
tpf=search_result[1].download()
```

```
tpf.plot(aperture_mask='pipeline')
```

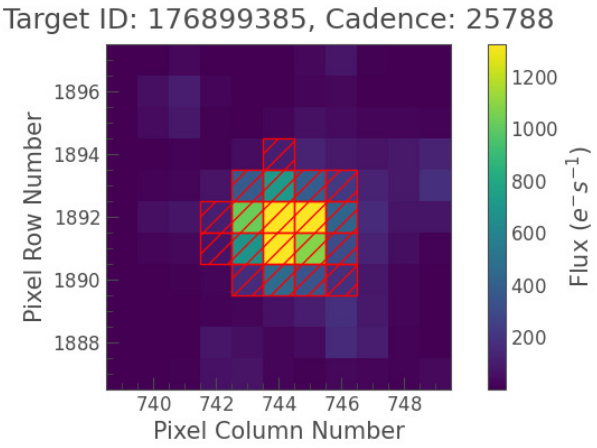

Рис. 38. TPF-изображение для HAT-P-6. Красным цветом выделена апертура `pipeline`, используемая при обработке данных TESS

Создадим кривую блеска, полученную в результате использования этой апертуры (рис. 39):

```
lc=tpf.to_lightcurve(aperture_mask="pipeline")
lc.plot()
```

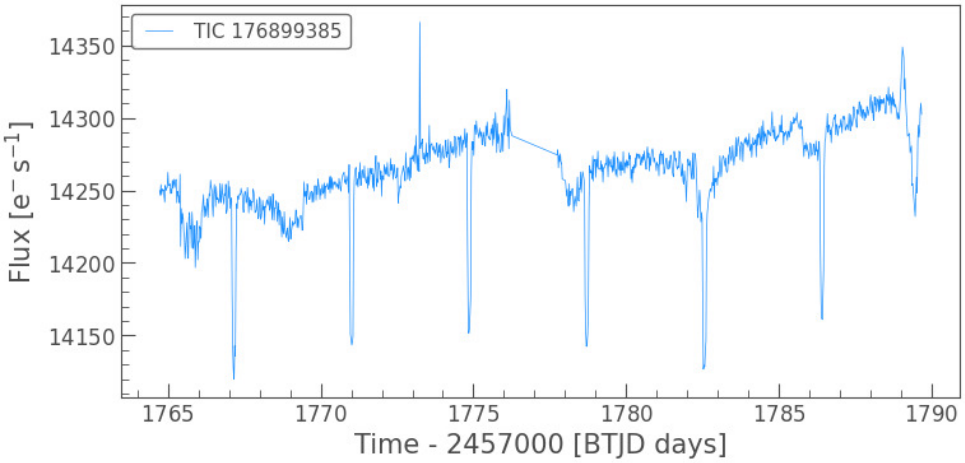

Рис. 39. Кривая блеска HAT-P-6, полученная из TPF-кадров с использованием автоматической апертуры `pipeline`

В функции `to_lightcurve()` ключевое слово `aperture_mask` позволяет изменять апертуру, используемую при выполнении фотометрии. Рассмотрим три стандартных аргумента, которые принимает это ключевое слово:

- **pipeline:** используется по умолчанию. Стандартная апертура, используемая при обработке данных TESS;
- **threshold:** выбирает все пиксели, поток которых превышает среднюю яркость в вырезанной области более чем на 3 стандартных отклонения (3σ);
- **all:** использует каждый пиксель в кадре для фотометрии.

Для примера посмотрим, как будет выглядеть апертурная маска, используя сигма-отсечение (`aperture\_mask = 'threshold'`, рис. 40):

```
tpf.plot(aperture_mask='threshold')
```

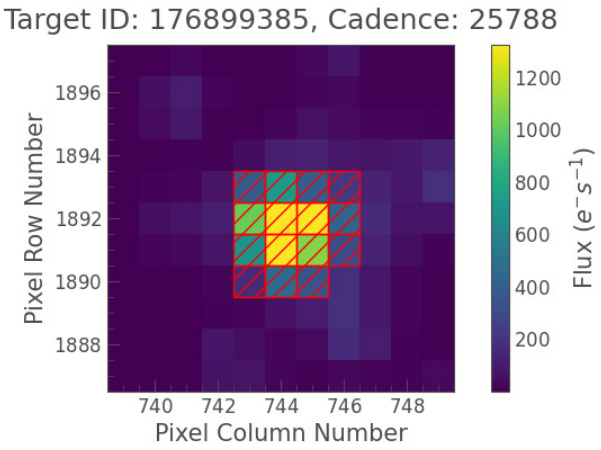

Рис. 40. TPF-изображение для HAT-P-6. Красным цветом выделена апертура, построенная с использованием сигма-отсечения (3σ)

Далее рассмотрим, что собой представляет апертурная маска (рис. 41):

```
tpf. create_threshold_mask()
```

```
array([[False, False, False, False, False, False, False, False, False,
        False, False],
       [False, False, False, False, False, False, False, False, False,
        False, False],
       [False, False, False, False, False, False, False, False, False,
        False, False],
       [False, False, False, False,  True,  True,  True, False, False,
        False, False],
       [False, False, False, False,  True,  True,  True,  True, False,
        False, False],
       [False, False, False, False,  True,  True,  True,  True, False,
        False, False],
       [False, False, False, False,  True,  True,  True,  True, False,
        False, False],
       [False, False, False, False, False, False, False, False, False,
        False, False]])
```

Рис. 41. Апертурная маска `'threshold'` в виде массива

Как можно заметить, маска является массивом, который принимает значения `True` или `False`. Пиксели, отмеченные как `True`, используются для фотометрии. По умолчанию данный метод применяет критерий 3σ. При необходимости возможно использование другого порогового значения (рис. 42):

```
custom_threshold_mask=tpf.create_threshold_mask(threshold=2)
custom_threshold_mask
```

```
array([[[False, False, False, False, False, False, False, False, False,
         False, False],
        [False, False, False, False, False, False, False,  True, False,
         False, False],
        [False, False, False, False, False, False, False,  True, False,
         False, False],
        [False, False, False, False,  True,  True,  True,  True, False,
         False, False],
        [False, False, False, False,  True,  True,  True,  True, False,
         False, False],
        [False, False, False, False,  True,  True,  True,  True,  True,
         False, False],
        [False, False, False, False,  True,  True,  True,  True, False,
         False, False],
        [False, False, False, False, False, False, False, False, False,
         False, False],
        [False, False, False, False, False, False, False, False, False,
         False, False],
        [False, False, False, False, False, False, False, False, False,
```

Рис. 42. Апертурная маска `'threshold'` с порогом 2σ в виде массива

Построим кривую блеска с использованием новой апертурной маски (рис. 43):

```
lc=tpf.to_lightcurve(aperture_mask=custom_threshold_mask)
lc.plot()
```

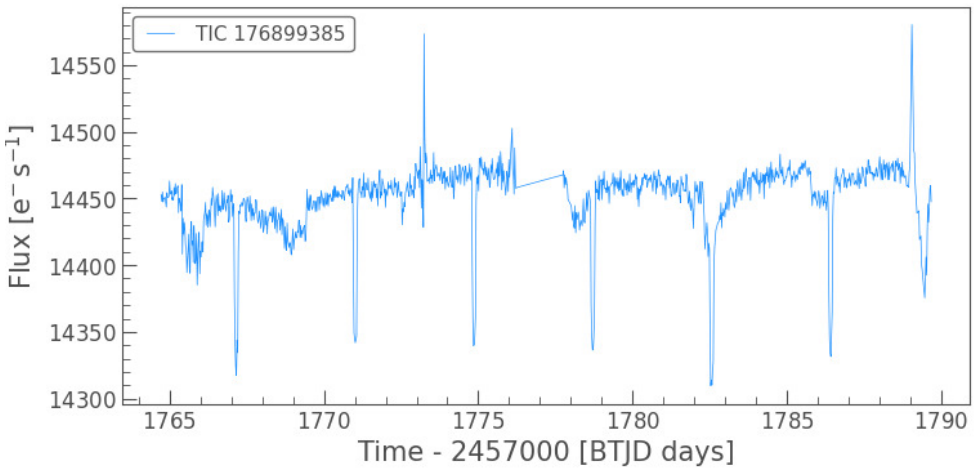

Рис. 43. Кривая блеска HAT-P-6, полученная из TPF-кадров с применением апертуры `threshold` с параметром 2σ

Рассмотрим далее создание собственной апертуры, используя библиотеку `NumPy`. Важно понимать, что при создании апертурной маски сначала указывается индекс по оси Y и отсчет начинается с 0, при этом элемент массива с координатами [0,0] будет располагаться в левом нижнем углу. В качестве примера создадим произвольную маску, выделив наиболее яркие пиксели (рис. 44):

```
cut_mask=np.zeros(tpf[0].shape[1:], dtype='bool')
cut_mask[4,5]=True
cut_mask[5,5]=True
cut_mask[5,6]=True
cut_mask[5,4]=True
cut_mask[4,6]=True
tpf.plot(aperture_mask =cut_mask)
```

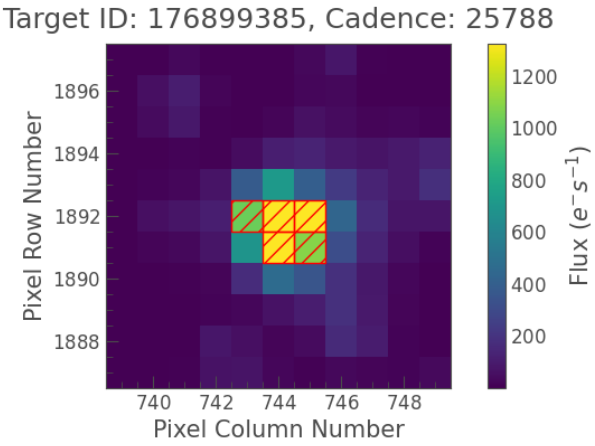

Рис. 44. TPF-изображение для HAT-P-6. Красным цветом выделена пользовательская апертура

Используя описанные выше методы, можно получить кривую блеска для созданной апертуры. Кроме того, в `Lightkurve` есть интерактивные функции для создания собственной маски. Для этого применяется метод `interact()`. На появившейся интерактивной панели необходимо выбирать нужные пиксели, отмечая их кнопкой мыши (подробнее в разделе 5.5).

5.4 Извлечение кривых блеска из FFI

Далеко не для всех объектов в поле зрения TESS доступны готовые кривые блеска и TPF-файлы. Для таких случаев возможно извлечение полезной информации из полнокадровых изображений TESS (FFI). Это делается с помощью метода `search_tesscut` библиотеки `Lightkurve`, в котором применяется инструмент `TESSCut` (Brasseur et al., 2019).

В качестве примера работы с FFI по-прежнему используем звезду HAT-P-6. Для начала выведем список доступных секторов, применяя метод `search_tesscut` (рис. 45):

```
search_result=lk.search_tesscut('HAT-P 6')
search_result
```

SearchResult containing 3 data products.

#	mission	year	author	exptime	target_name	distance
				s		arcsec
0	TESS Sector 16	2019	TESScut	1426	HAT-P 6	0.0
1	TESS Sector 17	2019	TESScut	1426	HAT-P 6	0.0
2	TESS Sector 57	2022	TESScut	158	HAT-P 6	0.0

Рис. 45. Таблица `search_result` со списком доступных для загрузки вырезок из FFI

Аналогично рассмотренным ранее методам при составлении запроса можно использовать ключевые слова. Загрузим данные сектора 16 и применим ключевое слово `cutout_size`, в котором указывается размер вырезаемого изображения (по умолчанию строится изображение 5×5 пикселей, в нашем случае 12×12) (рис. 46):

```
tpf=search_result[0].download(cutout_size=12)
tpf.plot()
```

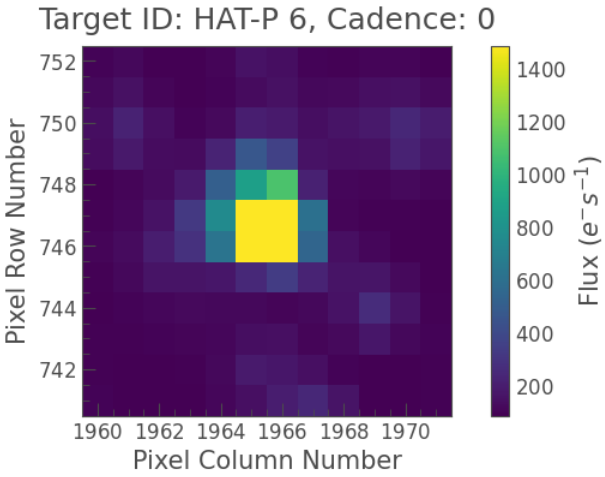

Рис. 46. Вырезанная область для HAT-P-6 по данным сектора 16

Применение автоматической апертуры TESS(`pipeline`) недоступно при работе с вырезками из FFI. В данном случае необходимо создать собственную апертурную маску (см. раздел 5.3.1) или использовать сигма-отсечение (рис. 47):

```
target_mask=tpf.create_threshold_mask(threshold=10)
tpf.plot(aperture_mask=target_mask, mask_color='k')
```

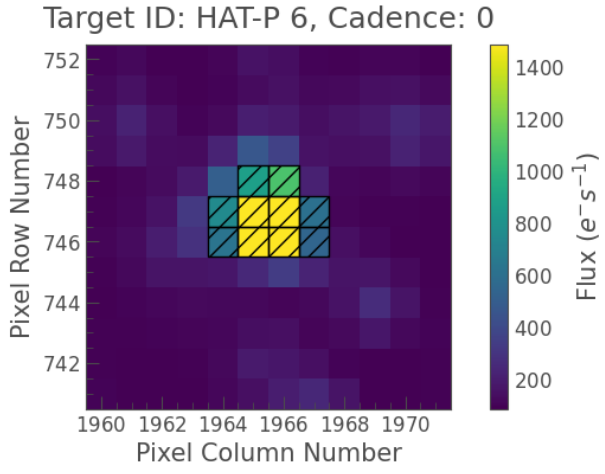

Рис. 47. Вырезанная область для HAT-P-6. Черным цветом выделена апертура, построенная с использованием сигма-отсечения (10σ)

Далее строим график нескорректированной кривой блеска (SAP_FLUX) без учета фона (рис. 48):

```
lc_target=tpf.to_lightcurve(aperture_mask=target_mask)
lc_target.plot(label='Object + background', color='dodgerblue')
```

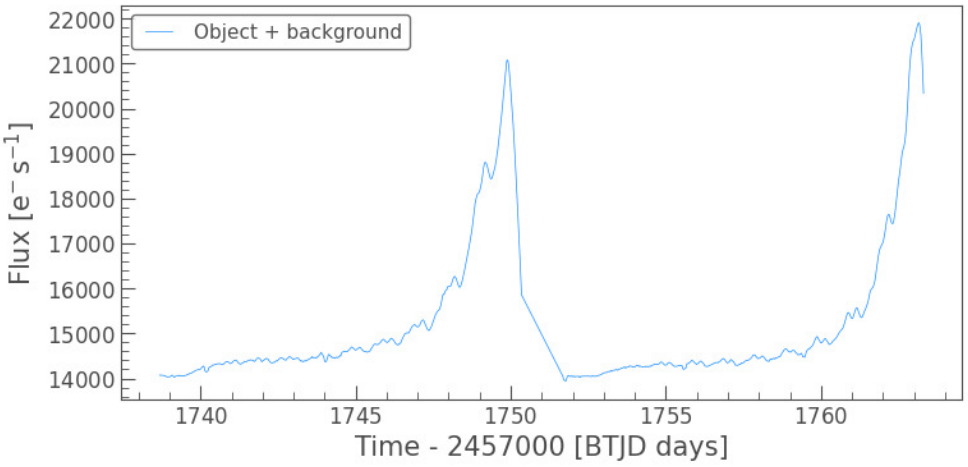

Рис. 48. Нескорректированная кривая блеска HAT-P-6, полученная из TPF-кадров

Следующим этапом необходимо вычесть вклад фона с использованием пороговой маски (рис. 49):

```
background_mask = ~tpf.create_threshold_mask(threshold=0.001)
tpf.plot(aperture_mask=background_mask, mask_color='w')
```

Примечание: использование символа '~' при создании апертурной маски фона обязательно.

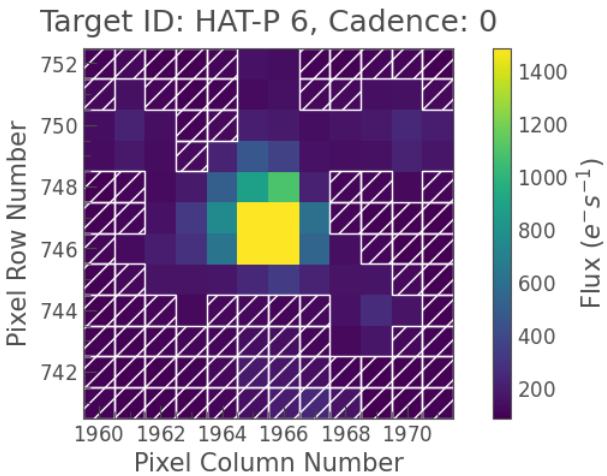

Рис. 49. Вырезанная область для HAT-P-6. Белым цветом выделены пиксели, используемые для оценки фонового потока

Оценка фона является сложной задачей, поскольку в выбранных пикселях может содержаться поток от расположенных поблизости других объектов. Чтобы увидеть объекты из каталога Gaia в анализируемом поле, необходимо воспользоваться интерактивным инструментом `interact_sky()`

(см. раздел 5.5). Для оценки вклада от фона в заданной апертуре (`target_mask`) рассчитывается значение фонового потока на пиксель и умножается на количество пикселей в апертуре (рис. 50):

```
n_target_pixels=target_mask.sum() \# количество пикселей в выбранной апертуре
n_background_pixels=background_mask.sum() \# количество пикселей в маске фона
background_lc_per_pixel=tpf.to_lightcurve(aperture_mask=background_mask) /
    n_background_pixels \# кривая блеска фона в расчете на 1 пиксель
background_lc=background_lc_per_pixel · n_target_pixels \# фон, который необходимо
    вычесть из кривой блеска объекта
corrected_lc=lc_target - background_lc.flux \# кривая блеска объекта после учета
    фона
corrected_lc.plot()
```

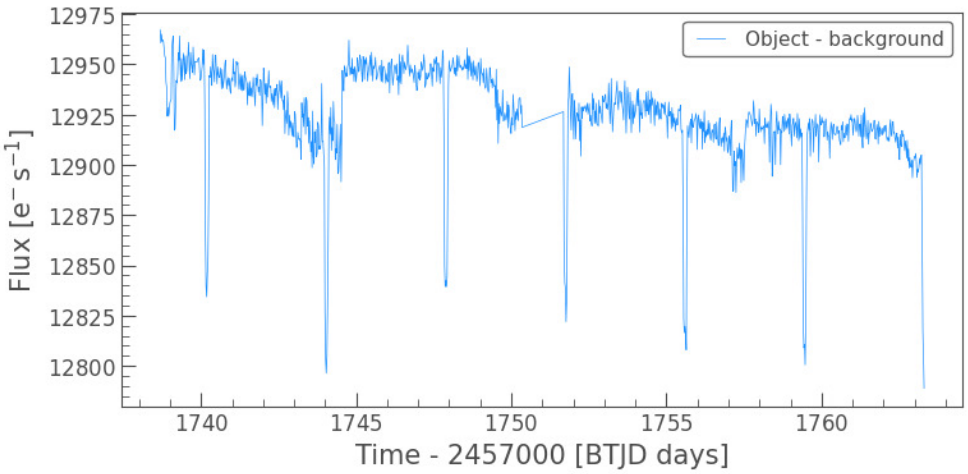

Рис. 50. Кривая блеска HAT-P-6, полученная из TPF-кадров после учета вклада фонового потока

На полученной кривой блеска отчетливо просматриваются транзиты экзопланеты. Теперь можно применять различные методы для работы с объектом `LightCurve`, например периодограммный анализ.

5.5 Интерактивный режим

`Lightkurve` имеет несколько интерактивных инструментов, которые позволяют выполнять быстрый анализ TPF, FFI и кривых блеска. Эти функции используют библиотеку `Bokeh` для создания виджетов и работают только в `Jupyter Notebook`.

Для активации интерактивного режима используется метод `interact()`. Этот метод предоставляет доступ к интерактивному выбору апертуры и мгновенному отображению результирующей кривой блеска. Можно выбрать отдельные произвольные пиксели, и результаты фотометрии автоматически обновятся. Для примера загрузим TPF для звезды HAT-P-6 (см. раздел 5.3) и применим метод `interact()` (рис. 51):

```
search_result=lk.search_targetpixelfile('HAT-P 6', exptime='long')
tpf=search_result[0].download()
```

```
tpf.interact()
```

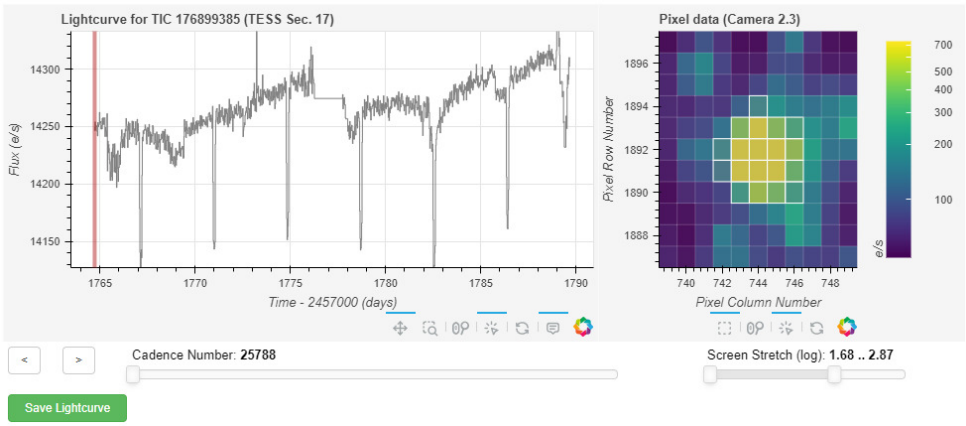

Рис. 51. Скриншот интерактивного режима для HAT-P-6 по данным сектора 17

Режимы взаимодействия:

- инструменты в правой части графиков позволяют масштабировать контраст и выбирать пиксели;
- при нажатии на пиксель отображается кривая блеска только для этого пикселя;
- `shift` + щелчок по нескольким пикселям показывает кривую блеска для выделенных пикселей;
- `shift + ctrl` + щелчок по уже выбранному пикселю отменит выбор этого пикселя (может работать только в Windows);
- удерживая левую кнопку мыши и выделяя нужную область, можно создать прямоугольную апертурную маску (отдельные пиксели из этой маски можно исключить);
- перемещение нижнего левого ползунка позволяет изменить положение вертикальной красной полосы, которая указывает на кадр, отображаемый на изображении TPF в правой области;
- нажатие стрелок влево и вправо равносильно предыдущему пункту;
- щелчок по точке на кривой блеска автоматически выполняет поиск этого TPF и отображает его в правой области диалогового окна.

Кроме того, наведение курсора на отдельные точки данных на кривой блеска отображает всплывающие подсказки, указывающие на дополнительную информацию (рис. 52).

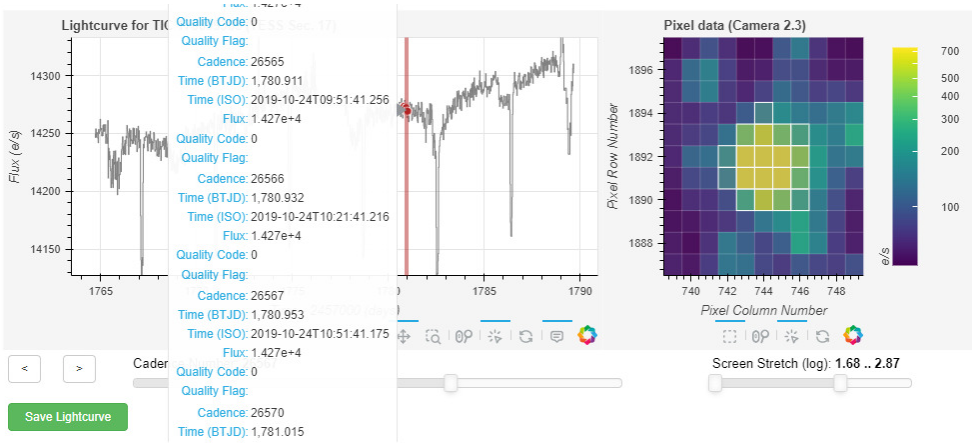

Рис. 52. Скриншот отображения дополнительной информации в интерактивном режиме для выделенного момента наблюдений HAT-P-6

Для объектов с имеющимися кривыми блеска по умолчанию используется апертура `pipeline`, которую можно заменить указанным выше способом. Если рассматриваются вырезки из FFI, то необходимо самостоятельно выделить нужные пиксели. Для сохранения кривой блеска в виде `fits`-файла необходимо нажать на кнопку `Save Lightcurve` в левом нижнем углу.

В `Lightkurve` есть дополнительный инструмент для интерактивной работы – `interact_sky()`. Этот метод отображает один кадр TPF с объектами, идентифицированными по каталогу Gaia (рис. 53, левая панель). При наведении курсора на объекты из каталога Gaia появится информация (идентификатор Gaia, звездная величина в фильтре G, координаты и др.). Как и метод `interact()`, `interact_sky()` позволяет изменять контраст изображения TPF. Для этого необходимо перемещать ползунки в левой нижней части интерактивного интерфейса (рис. 53, правая панель).

```
tpf.interact_sky()
```

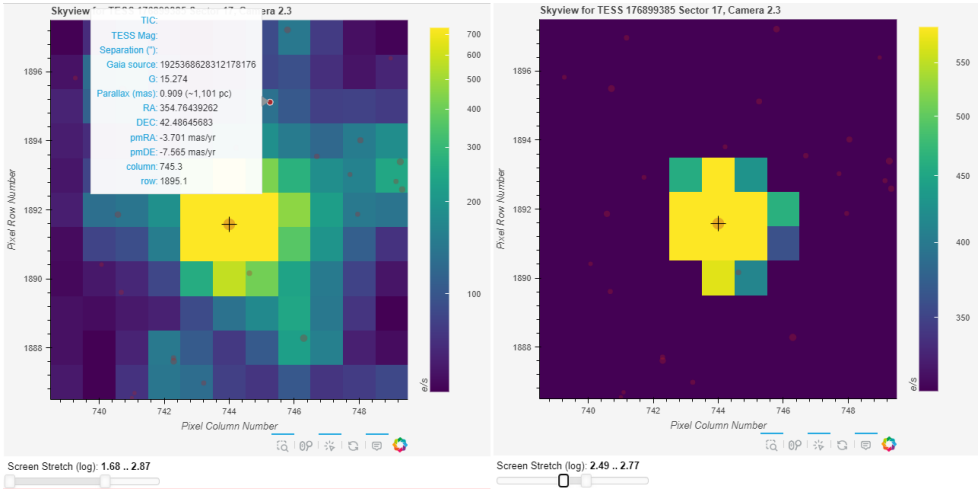

Рис. 53. Левая панель: TPF-кадр, на котором объекты из каталога Gaia обозначены малыми окружностями красного цвета, размер окружностей зависит от звездной величины объекта. Правая панель: аналогичное изображение с измененным контрастом

Выделив необходимую группу пикселей, можно увеличить ее, удерживая левую кнопку мыши (рис. 54).

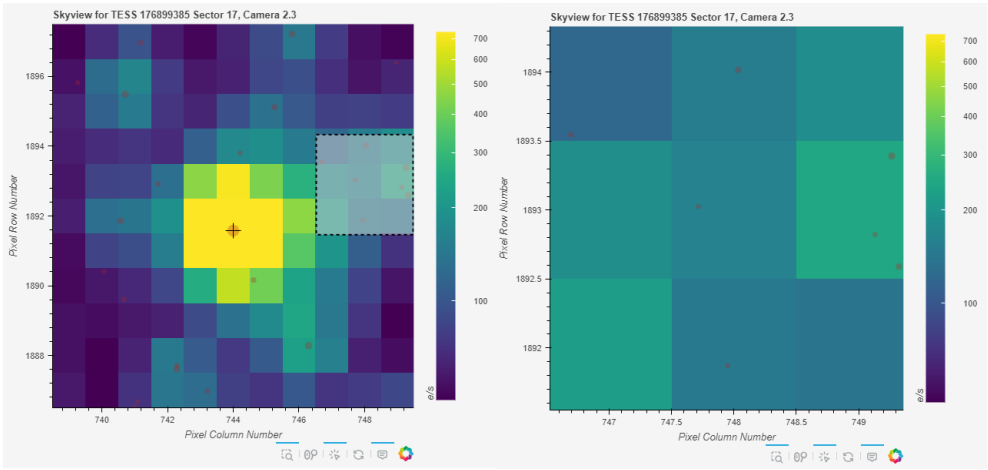

Рис. 54. TPF-кадр, на котором выделена область размером 3×3 пикселя (левая панель). Отображение выделенной области в увеличенном размере (правая панель)

Благодарности. При использовании данных TESS в своих исследованиях необходимо указать об этом одним из следующих способов:

- This paper includes data collected by the TESS mission. Funding for the TESS mission is provided by the NASA's Science Mission Directorate.
- This paper includes data collected by the TESS missions and obtained from the MAST data archive at the Space Telescope Science Institute (STScI). Funding for the TESS mission is provided by the NASA Explorer Program.
- This paper includes data collected by the TESS mission, which are publicly available from the Mikulski Archive for Space Telescopes (MAST).

При использовании библиотеки `Lightkurve` желательно указать об этом в работе: This research made use of Lightkurve, a Python package for Kepler and TESS data analysis (Lightkurve Collaboration et al., 2018).

При использовании сервиса `TESScut` или инструментов `TESScut` в библиотеке `Lightkurve` необходимо ссылаться на работу Brasseur et al. (2019). Кроме того, `Lightkurve` была построена на основе ряда библиотек, включая `NumPy, SciPy, Astropy, Astroquery` и `Matplotlib`. Рекомендуется также ссылаться на эти библиотеки в своих исследованиях.

Литература

Brasseur C.E., Phillip C., Fleming S.W., et al., 2019. Astrophysics Source Code Library, record ascl: 1905.007.

Hartman J.D., Bakos G.Á., 2016. Astronomy and Computing, vol. 17, pp. 1–72.

Kovács G., Zucker S., Mazeh T., 2002. Astron. Astrophys., vol. 391, pp. 369–377.

Lightkurve Collaboration, Cardoso J., Hedges C., et al., 2018. Astrophysics Source Code Library, record ascl: 1812.013.

Lomb N.R., 1976. Astrophys. Space Sci., vol. 39, pp. 447–462.

Ricker G.R., Winn J.N., Vanderspek R., et al., 2015. Journal of Astronomical Telescopes, Instruments, and Systems, vol. 1, id. 014003.

Scargle J.D., 1982. Astrophys. J., vol. 263, pp. 835–853.

ISSN 0367–8466
IZVESTIYA KRYMSKOI ASTROFIZICHESKOI OBSERVATORII

Izv. Krymsk. Astrofiz. Observ. 120, № 4, 5–44 (2024)

Processing observations of the TESS space observatory. Methodological recommendations

M. Gorbachev

Crimean Astrophysical Observatory, Nauchny 298409
mgorbachev17@gmail.com

Abstract. This paper is a methodological guide for using data from the TESS (Transiting Exoplanet Survey Satellite) orbital observatory. It describes features of TESS observations and various data products available for analysis.

The methodological guide is basically a Russian translation of the manual for using the `Lightkurve package`, supplemented with comments and detailed explanations. The `Lightkurve` library, written in `Python`, allows one to download, process, and visualize data from the Kepler/K2 observatory and TESS. It includes a wide range of functions for the search, download, and analysis of light curves of various astrophysical objects. The information presented is sufficient to handle both individual sources and a large number of objects acquired by the TESS observatory.

Key words: photometry, periodogram analysis, data processing