ZTF24aahgqwk in NGC 3443

Light Curve Notebook

This notebook begins with the 36 stacked images produced by our Calibration Notebook, and produces a light curve, consisting of 18 Sloan r' and 18 Sloan g' data points.

In [ ]:
```
import os

# THIS COMMENT IS THE LONGEST A LINE CAN BE AND STILL RENDER COMPLETELY WHEN PRINTING IN LANDSCAPE MODE.

# This notebook needs to be able to find the stacked images.

home_directory = os.path.expanduser('~')
supernova_project_directory = os.path.join(home_directory, 'Projects', 'supernova-observation')
stacked_directory = os.path.join(supernova_project_directory, 'analyses', 'ZTF24aahgqwk', 'stacked')

# The 36 images are in the stacked directory. There were 18 observation sessions with 2 filters each.

# filters

filters = ['r', 'g']
filter_full_names = ["Sloan r'", "Sloan g'"]

# observation dates (UTC)

observation_dates = [
    '2024-03-20',
    '2024-03-21',
    '2024-03-23',
    '2024-03-27',
    '2024-04-02',
    '2024-04-03',
```

```
    '2024-04-04',
    '2024-04-06',
    '2024-04-10',
    '2024-04-11',
    '2024-04-13',
    '2024-04-17',
    '2024-04-21',
    '2024-04-22',
    '2024-04-23',
    '2024-04-29',
    '2024-04-30',
    '2024-05-02'
]
```

Method

We will be using least-squares minimization to fit the image data.

We could write the code from scratch, but it is tricky to find the minimum of a function in a high-dimensional space, so it is better to use well-tested code. The scipy library has an algorithm for fitting.

The first thing we want to do is learn to to use their algorithm. This might seem easy, but remember, that the data has to be passed in, the parameters have to be passed in, and the residuals have to be computed, and in order to do this completely generally, abstractions have to be introduced.

The authors of scipy.optimize.least_squares give an example uing the Rosenbrock function which is considered a challenging test for least squares fitting.

In this example, there is no data. Just two parameters are given, x[0] and x[1], and the function that the user supplies produces two residuals.

In [ ]:
```
import numpy as np
from scipy.optimize import least_squares

def the_rosenbrock_function(parameter_vector, data=None):
    parameter_a = parameter_vector[0]
    parameter_b = parameter_vector[1]
    residual_1 = parameter_b - parameter_a**2
    residual_2 = 1 - parameter_a
    residual_vector = [residual_1, residual_2]
    return np.array(residual_vector)

initial_guess_for_parameter_a = 2
initial_guess_for_parameter_b = 2

initial_parameter_vector = [initial_guess_for_parameter_a, initial_guess_for_parameter_b]


result = least_squares(the_rosenbrock_function, np.array(initial_parameter_vector))

print(result.x)
```

In [ ]:
```
## Problem 1

# Note that in the above example, the data argument wasn't used.

# Here is some data consisting of an array of 13 ordered pairs. You can think of these ordered pairs
# as x-y values, and we want to find the best values for the parameters m and b with y = m * x + b.

some_data = [
    (24, 196),
    (26, 194),
    (28, 195),
    (29, 194),
    (31, 192),
    (32, 191),
    (34, 193),
    (36, 192),
    (38, 191),
    (40, 192),
    (41, 193),
    (42, 190),
    (43, 190)
]

# Use scipy.optimize.least_squares to solve this problem
```

In [ ]:
```
## Problem 2

# This is a linear regression problem. You know how to solve it exactly. Write the code that computes
# the exact valuues of m and b from the data.
```

In [ ]:
```
## Problem 3

# Compare and reconcile the answers to Problems 1 and 2.
```