

NARIT Radio Astronomy Summer School 2023

A brief manual for hands-on activity

Finding the period of maser flares in YSO: G107.298+5.639

using Lomb-Scargle periodogram

(Prepared by Montree Phetra and edited by Kitiyanee Asanok)

This activity will be followed the research of [2020A&A...634A..41O](#) (Olech, M et al. 2020) which observed 4 maser emission lines e.g., 6.7 GHz (CH₃OH), 1.665 GHz (OH), 1.667 GHz (OH), and 22 GHz (H₂O) from the intermediate-mass young stellar object G107.298+5.639. In this activity, we will

- calculate the period of maser flares and discuss the result,
- (optional) plot the VLBI data and discuss the result.

1. Data extraction

Go to the vizier catalog (<http://vizier.cds.unistra.fr/viz-bin/VizieR?-source=J/A+A/634/A41>) and download the integrated flux density as a function of modify Julian day (MJD) (see Figure 1).

J/A+A/634/A41	G107.298+5.639 multi-frequency maser light curves (Olech+, 2020)
1. J/A+A/634/A41/ch3oh	Light curve of 6.7GHz methanol maser [timeSeries] (light curve) (1843 rows)
2. J/A+A/634/A41/oh-1665	Light curve of 1.665GHz hydroxyl maser [timeSeries] (light curve) (144 rows)
3. J/A+A/634/A41/oh-1667	Light curve of 1.667GHz hydroxyl maser [timeSeries] (light curve) (144 rows)
4. J/A+A/634/A41/h2o	Light curve of 22GHz water maser [timeSeries] (light curve) (1128 rows)

Figure 1 An example of Vizier webpage

Click “light curve” of 6.7 GHz CH₃OH maser, it will create a new window which is shown in Figure 2.

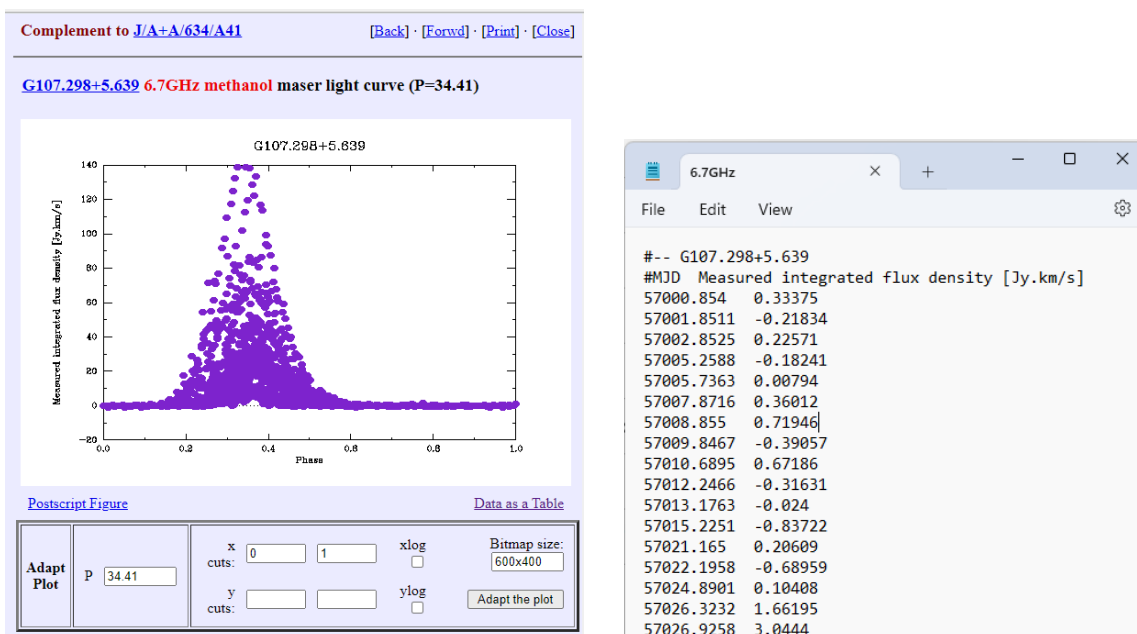


Figure 2 (left) light curve plot from vizier (right) MJD data from text file

At the adapt plot tool, edit P value from 34.41 to 0 (it will convert the x data type from phase to MJD), then click 'Adapt the plot' to change the plot, and click 'Data as a Table' then save the data to the text file with the name of frequency as '6.7GHz.txt'

Do it yourself: save the data for other maser lines 1.665 GHz (OH), 1.667 GHz (OH), and 22 GHz (H₂O).

2. Finding the period of maser lines using Python Libraries on Google colab

**If you have experiences with Python, please try by yourself on your machine/laptop. **

Go to your drive in google account, then create a google colaboratory. Click the folder tool on the left side and upload your data files to the directory. At the main screen, type the import module as follow in the first cell, then hold 'shift+enter' to run the cell

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.signal as signal
import pandas as pd
import re
```

The screen
like in Figure 3.

will shows look

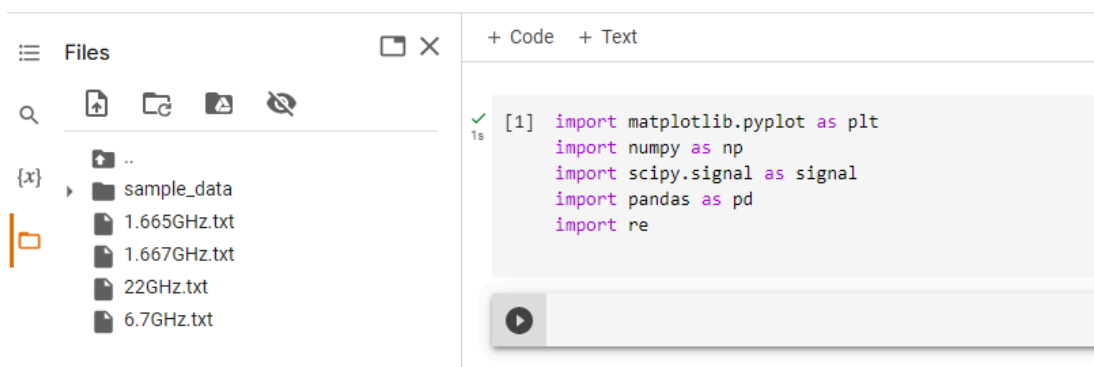


Figure 3 An example of the screen of colab

We need to input the data to our calculation; we define the function to call the data by type the code as follows to the new cell and then run it.

```
def callFlux(masere):
    filename=masere+".txt"
    f=open(filename,"r")
    ls=sum([1 for line in open(filename)]
    dataFlux=[1,1]
    for i in range(ls):
        line=f.readline()
        if i==0:
            masere=line.replace(" ", "")
            second=split_line.replace(" ", "")
            second=split_line.replace(" ", "")
        else:
            seconds=line.split(" ", line)
            dataFlux[0]=seconds[0].replace(" ", "")
            dataFlux[1]=seconds[1].replace(" ", "")
    f.close()
    seconds=line.split(" ", line)
    dataFlux.append(float(seconds[0]))
    dataFlux.append(float(seconds[1]))
    f.close()
    print("dataFlux")
    return dataFlux
```

After run the cell, we need to set the maser name to the function, then it will give you a result of source name, MJD, and Integrated Flux density. Next, try to call the file name of '6.7GHz.txt' with the following code and then run it.

```
maser='6.7GHz'
source, mjd, int_flux=callflux(maser)
```

The data of in parameters 'mjd', and we need to look flux density with

6.7GHz will keep called 'source', 'int_flux'. Now, the integrated the observation

time. We again define the function for plotting as follows code and then run it.

```
def plot(kind, title, x, y, w, h, fig_name):
    fig = plt.figure(figsize=(w,h))
    if kind=='scatter':
        plt.scatter(x[1], y[1])
    if kind=='plot':
        plt.plot(x[1], y[1])
    plt.title(title)
    plt.xlabel(x[0])
    plt.ylabel(y[0])
    plt.show()
    fig.savefig(fig_name,dpi=200)
```

The above function is use to plot the data of 2 parameters x and y in the form of scattering and line. Next, try to plot the integrated flux density as scatter, title='Time series of 6.7GHz', with the figure size of width and high as 12 and 4. Then save the image with the name 'time_series_of_6.7GHz.jpg' by using the code below.

```
title= 'Time series of '+maser
fig_name='time_series_of_'+maser+'.jpg'
plot('scatter', title, mjd, int_flux, 12, 4, fig_name)
```

Then, you look like in

may see the plot Figure 4.

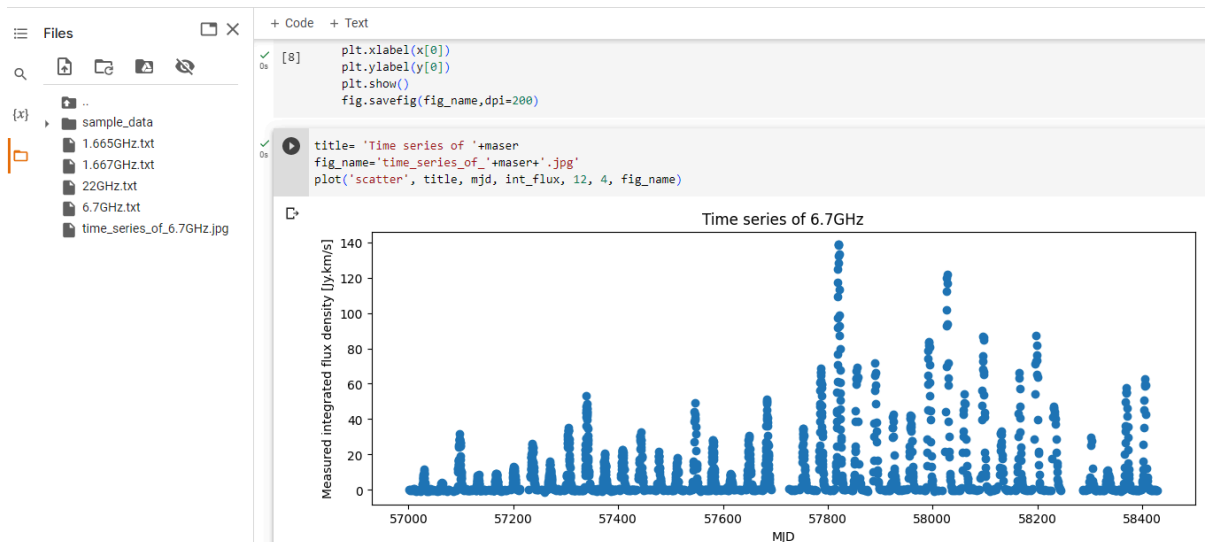


Figure 4 Time series of 6.7 GHz

Next, we will use the Lomb-Scargle periodogram to calculate the period of maser from time series data. This method can be imported from the module name 'scipy.signal'. We again set the function to find a period with the code.

```
def lombscargle(x, y, p_min, p_max, n):
    p = np.linspace(p_min, p_max, n)
    f = 2*np.pi/p
    amp = signal.lombscargle(x[1], y[1], f, normalize=True)
    P=['Period',p]
    Amp=['Amplitude',amp]
    max_a=max(amp)
    for i in range(n):
        if amp[i]==max_a:
            p_peak=p[i]
            break
    return P, Amp, p_peak
```

The above function will use x and y data to calculate the period with the condition by range of p_min and p_max for n bins. The above function will give you the period and the amplitude of each bin, and the period of maximum amplitude which means that the absolute time of flaring period. We try to calculate them, plot a result, and print the flaring period with the following code.

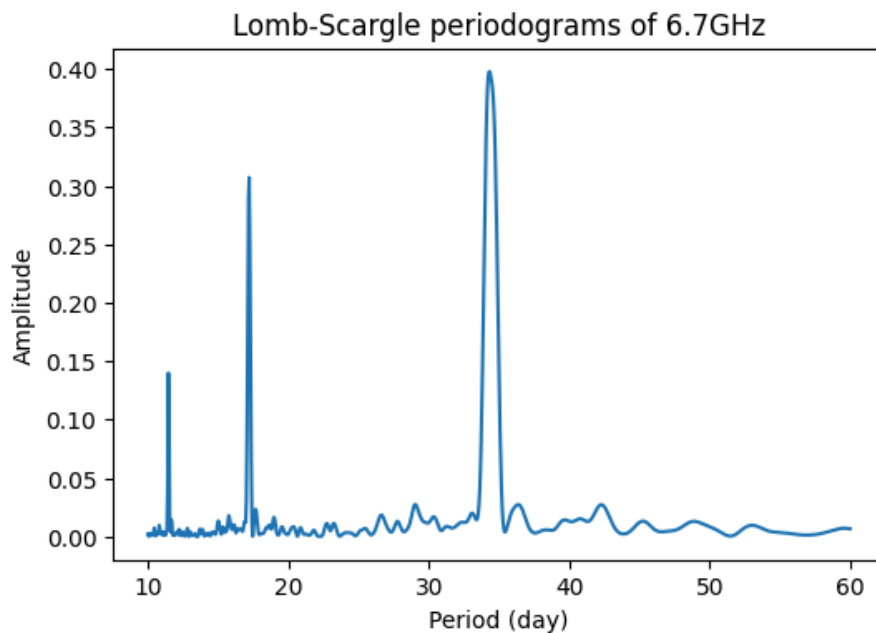
```
p, amp, p_peak=lombscargle(mjd, int_flux, 10, 60, 1000)

title= 'Lomb-Scargle periodograms of '+maser
fig_name='LombScar_of_'+maser+'.jpg'
plot('plot', title, p, amp, 6, 4, fig_name)

print('flraing period is : {:.2f} day'.format(p_peak))
```

Then, you
result look like Figure 5.

may see the



flaring period is : 34.32 day

Figure 5 Lomb-Scargle periodograms of 6.7GHz and text printing from the code.

Do it yourself:

- Call the data of 1.665GHz
- Plot the time-series of 1.665GHz
- Calculate the flaring period of 1.665GHz
- Do it again for 1.667GHz and 22GHz
- Discuss the flaring period from all maser emission lines.

Hint: you may use the function above to do it, but you need to change something like maser name

3. (Optional) Maser distribution from VLBI

In this part, we will use the data from VLBI (Very-long-baseline interferometry) observation. The data of maser from VLBI will be called as 'features', maser feature is calculated by using Gaussian Fitting. It will estimate the peak flux, flux, position offset, frequency and velocity from the spot of maser in particular area.

Unfortunately, the data of maser features is not available in the catalog, but we can take it from the main paper. The maser features data is available in the excel files named 'Features_Olech_2022.xlsx' with the description as follows.

- Sheets : 22GHz, 6.7GHz, and 1.665GHz
- Col.1 : offset position in x-axis (milli-arcsecond)
- Col.2 : offset position in y-axis (milli-arcsecond)
- Col.3 : Doppler velocity at the peak (km/s)
- Col.4 : Maser brightness (Jy/beam)

Note: the center of position offset is RA(J2000) = 22h21m26.7730s and Dec(J2000) = 63°51'37.657''

First, you need to upload the excel file into the google colab directory. Then type the code new code at follow and run it.

```
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, FigureManager
import numpy as np
import pandas as pd
import sys
import os
import glob
import re
import math
import time
import datetime
import random
import string
import itertools
import collections
import statistics
import functools
import operator
import multiprocessing
import concurrent.futures
import threading
import queue
import logging
import warnings
import copy
import pickle
import json
import yaml
import csv
import xml.etree.ElementTree as ET
import sqlite3
import requests
import urllib3
import ssl
import certifi
import urllib.request
import urllib.error
import urllib.parse
import urllib.response
import urllib.robotparser
import urllib.robotfile
import urllib.robotdata
import urllib.robotlog
import urllib.robotlog2
import urllib.robotlog3
import urllib.robotlog4
import urllib.robotlog5
import urllib.robotlog6
import urllib.robotlog7
import urllib.robotlog8
import urllib.robotlog9
import urllib.robotlog10
import urllib.robotlog11
import urllib.robotlog12
import urllib.robotlog13
import urllib.robotlog14
import urllib.robotlog15
import urllib.robotlog16
import urllib.robotlog17
import urllib.robotlog18
import urllib.robotlog19
import urllib.robotlog20
import urllib.robotlog21
import urllib.robotlog22
import urllib.robotlog23
import urllib.robotlog24
import urllib.robotlog25
import urllib.robotlog26
import urllib.robotlog27
import urllib.robotlog28
import urllib.robotlog29
import urllib.robotlog30
import urllib.robotlog31
import urllib.robotlog32
import urllib.robotlog33
import urllib.robotlog34
import urllib.robotlog35
import urllib.robotlog36
import urllib.robotlog37
import urllib.robotlog38
import urllib.robotlog39
import urllib.robotlog40
import urllib.robotlog41
import urllib.robotlog42
import urllib.robotlog43
import urllib.robotlog44
import urllib.robotlog45
import urllib.robotlog46
import urllib.robotlog47
import urllib.robotlog48
import urllib.robotlog49
import urllib.robotlog50
import urllib.robotlog51
import urllib.robotlog52
import urllib.robotlog53
import urllib.robotlog54
import urllib.robotlog55
import urllib.robotlog56
import urllib.robotlog57
import urllib.robotlog58
import urllib.robotlog59
import urllib.robotlog60
import urllib.robotlog61
import urllib.robotlog62
import urllib.robotlog63
import urllib.robotlog64
import urllib.robotlog65
import urllib.robotlog66
import urllib.robotlog67
import urllib.robotlog68
import urllib.robotlog69
import urllib.robotlog70
import urllib.robotlog71
import urllib.robotlog72
import urllib.robotlog73
import urllib.robotlog74
import urllib.robotlog75
import urllib.robotlog76
import urllib.robotlog77
import urllib.robotlog78
import urllib.robotlog79
import urllib.robotlog80
import urllib.robotlog81
import urllib.robotlog82
import urllib.robotlog83
import urllib.robotlog84
import urllib.robotlog85
import urllib.robotlog86
import urllib.robotlog87
import urllib.robotlog88
import urllib.robotlog89
import urllib.robotlog90
import urllib.robotlog91
import urllib.robotlog92
import urllib.robotlog93
import urllib.robotlog94
import urllib.robotlog95
import urllib.robotlog96
import urllib.robotlog97
import urllib.robotlog98
import urllib.robotlog99
import urllib.robotlog100
```

Next, call the features data from excel file and keep it in a different name with the following code.

```
excel_file='Features_Olech_2022.xlsx'
h2o=callfeatures(excel_file,'22GHz')
ch3oh=callfeatures(excel_file,'6.7GHz')
oh=callfeatures(excel_file,'1.665GHz')
```

Now, we will
features from

plot the maser
VLBI data with

[illegible]

the size that refer to maser brightness, color refer to Doppler velocity, and symbol refer to a maser species. Use the following codes.

Finally, the result may look like this.

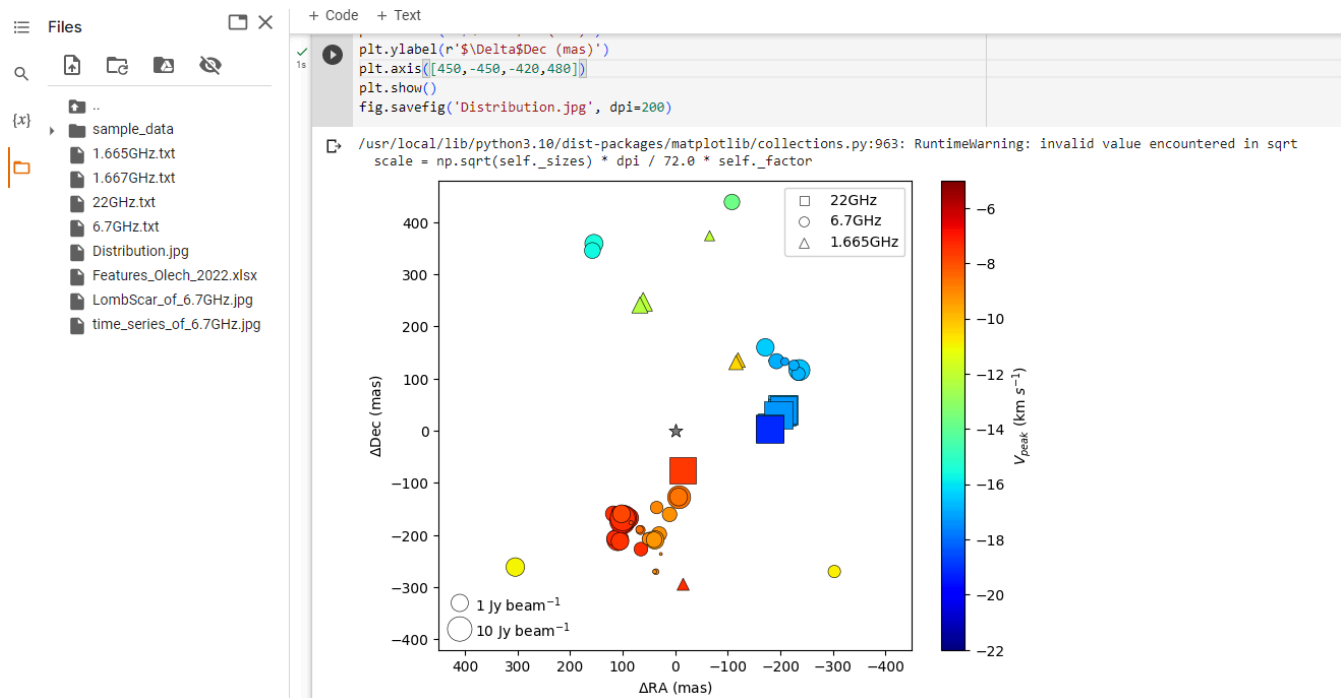


Figure 6 The distribution of various types of masers e.g., 22 GHz, 6.7 GHz and 1.665 GHz.

Do it yourself:

- According to the distribution and Doppler motion, what is the possible structure of this young stellar object? (Outflow, Disk, etc.?) why you think like that?
- Discuss your interpretation with a proper motion of 6.7GHz in the paper
- The distance of this object is about 0.76 kpc, so try to estimate the size of this object.

Note: Blue features mean moving towards the observer and red features mean moving away from the observer.

--- Good Luck ---