# gyre-lc

*Release 0.5*

**Aaron Lopez**

**Jan 03, 2022**

# USER GUIDE

**GYRE-lc** is a Python library for the production of synthetic light curves for pulsating binary systems. It requires at least one MESA stellar model and its corresponding GYRE pulsation model as inputs. A model spectrum is also required– GYRE-lc works best with MSG interpolated spectra for speed, ease of use, accuracy, and reliability, but it also takes SYNSPEC spectra in a *custom HDF5 format*.

**Note:** This project is under active development.

CHAPTER

ONE

# PRELIMINARIES

## 1.1 Obtaining GYRE-lc

The source code for GYRE-lc is hosted on GitHub at github.com/aaronesque/gyre-lc. Like all GYRE expansions, GYRE-lc is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 3.

## 1.2 Development Team

GYRE-lc remains under active development by the following team:

- Aaron Lopez (University of Wisconsin-Madison); project leader
- Rich Townsend (University of Wisconsin-Madison)

## 1.3 Related Links

- The MESA Software Development Kit (SDK), which provides the compilers and supporting libraries needed to build GYRE-lc.
- MESA, which calculates the stellar models compatible with GYRE-lc.
- GYRE, which calculates the pulsation models compatible with GYRE-lc.
- MSG, which rapidly interpolates stellar spectra from a multidimensional grid for GYRE-lc.

## 1.4 Acknowledgments

GYRE-lc has been developed with financial support from the following grants:

**Note:** This project is under active development.

CHAPTER

TWO

# QUICK START

GYRE-lc presumes a basic familiarity with introductory python, which includes the ability to call functions and make simple 1-dimensional plots.

To get started with GYRE-lc, follow these five simple steps:

- download & install the MESA Software Development Kit (SDK);
- download & install MSG;
- download & unpack the GYRE-lc source code;
- set the `GYRELC_DIR` environment variable to point to the newly created source directory;
- implement in Python with `sys.path.insert(0, os.path.join(os.environ['GYRELC_DIR'], 'lib'))` and `import gyrelc`

For a more in-depth installation guide, refer to the Installation chapter. If the package doesn't run properly, consult the troubleshooting chapter. Otherwise, proceed to the next chapter where you'll learn to run your first GYRE-lc calculation.

**Note:** This project is under active development.

CHAPTER

# THREE

# PYTHON WALKTHROUGH

This chapter provides a walkthrough of using the GYRE-lc package to calculate a light curve for the eccentric ellipsoidal variable of $\iota$ Orionis.

## 3.1 Seting up your inputs

There are 3 inputs to consider when producing a GYRE-lc light curve:

- 1-2 stellar models, depending on how many stars contribute to the overall light curve
- 1-2 pulsation models. one per stellar model
- 1 inlist specifying orbital parametersi, paths to stellar and pulsation models, and other context for the problem

### 3.1.1 The iota Orionis Models

The GitHub repository includes the model data necessary to create a light curve and test GYRE-lc's functionality. You will be creating a light curve for the eccentric ellipsoidal variable of $\iota$ Orionis, which we'll refer to as simply $\iota$ Ori for brevity. What follows is a list of input files and descriptions thereof.

**iOri-Aa.mesa & iOri-Ab.mesa** The stellar models for each binary component, $\iota$ Ori Aa & Ab, were created with MESA using stellar parameters listed in Pablo *et al.*[1]. The MESA inlists are included for reproducibility of results.

**iOri-Aa-response.h5 & iOri-Ab-response.h5** The pulsations models and their corresponding GYRE inlists are also included for each component. They are created with GYRE using the parameters listed in Pablo *et al.*[?]. These contain the amplitudes and frequencies for the first 100 normal modes of a star's tidally excited oscillations.

**binary_params.in** A GYRE-lc inlist specifying the binary parameters and synthetic instrument configuration. The `&observer` namelist is optional when run from a Jupyter notebook.

**[*filter*].h5 or tXXXXXgXXX.h5** Lastly, model spectra (produced with SYNSPEC) for each component are also included for testing purposes, but they are entirely optional. GYRE-lc works best with MSG– for that, three MSG-produced photometric grids are included corresponding to the filters BRITE-R, BRITE-B, and Kepler. These grids are included for demonstration purposes, and if you'd like to synthesize light curves for different passbands, you'll have to create those using MSG yourself.

[1] Herbert Pablo, N. D. Richardson, J. Fuller, J. Rowe, A. F. J. Moffat, R. Kuschnig, A. Popowicz, G. Handler, C. Neiner, A. Pigulski, G. A. Wade, W. Weiss, B. Buysschaert, T. Ramiaramanantsoa, A. D. Bratcher, C. J. Gerhartz, J. J. Greco, K. Hardegree-Ullman, L. Lembryk, and W. L. Oswald. The most massive heartbeat: an in-depth analysis of \ensuremath ι Orionis. *\mnras* , 467(2):2494–2503, May 2017. arXiv:1703.02086, doi:10.1093/mnras/stx207.

## 3.2 The GYRE-lc Module

To use GYRE-lc in Python, first make sure the `GYRELC_DIR` environment variable is set (see *Quick Start*). I use a Jupyter notebook for this walkthrough, but you may later choose to write a Python script instead should it better suit your workflow.

First, create a new working directory and copy the `binary_params.in` inlist from `$GYRELC_DIR/test/` into this new directory.

In this same working directory, open a new Jupyter notebok

Copy and past the following imports:

```
# Import standard modules

import numpy as np
import sys
import os

# Import pymsg

sys.path.insert(0, os.path.join(os.environ['MSG_DIR'], 'lib'))
import pymsg

# Import gyrelc modules

sys.path.insert(0, os.path.join(os.environ['GYRELC_DIR'], 'lib'))
import gyrelc as lc
```

Next, create a `Binary` object by feeding it the path to your `binary_params.in` file:

```
# Create Binary object
iori = lc.Binary('./binary_params.in')
```

Now create an `Observer` object:

```
# Creat an Observer object
obs = lc.Observer(iori, 'BRITE-B')
```

The `Binary` object consists of two `Star` objects, an `Irradiation` object, as well as the various attributes and parameters required to provide the `Observer` object sufficient context to synthesize a light curve. The `Observer` object primarily contains functions for light curve synthesis and analysis thereof. The last parameter left to specify, the choice of passband, is left as an argument for the `Observer` class.

Finally, create a light curve:

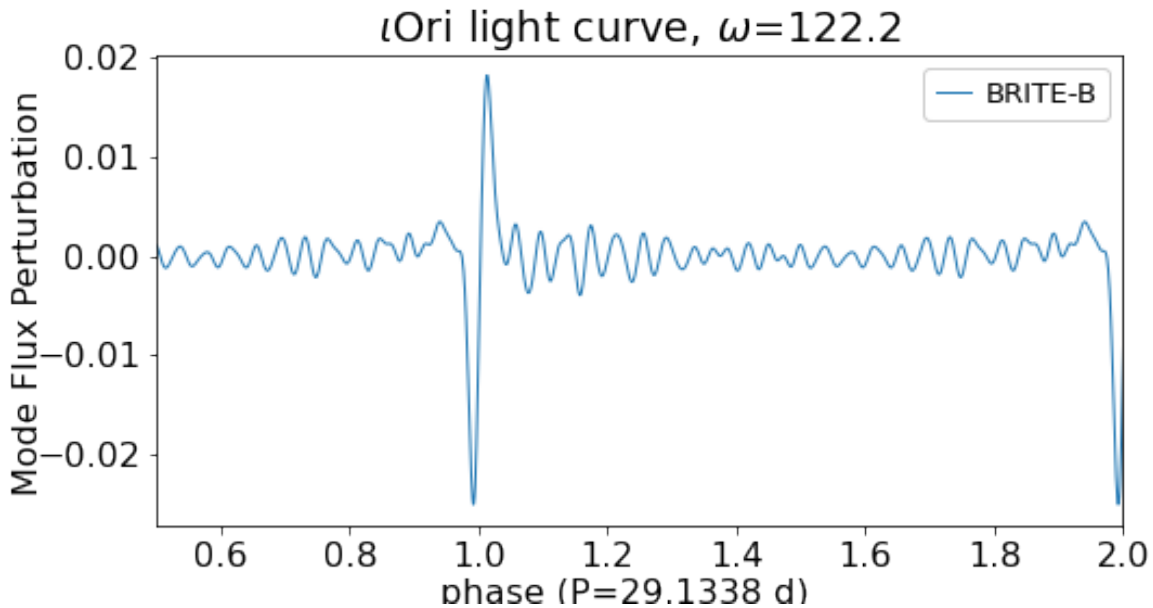
```
# Specify inclination and argument of periastron
inc = 62.86
omega = 122.2

# Duration of 'observation' and number of points
omega_orb = iori.orbit_params['Omega_orb']
t = np.linspace(0.5/omega_orb, 2.5/omega_orb, num=2000, endpoint=False)

flux = obs.find_flux(inc, omega, t)
```

An important subtlety: the `find_flux()` function *requires* the observation time to be in units of the orbital period. Here, I'm simulating a BRITE-B passband observation of $\iota$ iOri that consists of 2000 data points over 2 orbital periods, begining at half a period past periastron.

Using `matplotlib`, you may plot your results:

```
# Plot

fig, ax = plt.subplots(sharex=True, figsize=(8,4))

legend_style = {'framealpha':1.0, 'handlelength':1.2, 'handletextpad':0.5, 'fontsize':
↪'small'}

ax.plot(t*omega_orb, flux, lw=1, label='BRITE-B')
ax.legend(loc=1, **legend_style)

ax.set_xlim(0.5,2.)

ax.set_title(f'$\iota$Ori light curve, $\omega$={omega}')

fig.text(0.01, 0.5, r'Mode Flux Perturbation', va='center', rotation='vertical')
fig.text(0.5, 0.0, f'phase (P={1./omega_orb:4.4f} d)', ha='center')
```

The legend style and labels are entirely a matter of stylistic choice, but a plot with this *xlim* should look something like this:




**Note:** This project is under active development.

CHAPTER

FOUR

# INSTALLATION

This chapter discusses GYRE-lc installation in detail. If you just want to get up and running, have a look at the Quick Start chapter.

## 4.1 Prerequisites

A complete GYRE-lc workflow typically requires the use of additional software to produce the star and pulsation models that go into GYRE-lc as input for light curve synthesis. This includes:

- The MESA Software Development Kit (SDK), which provides the compilers and supporting libraries needed to build GYRE-lc.
- MESA, which calculates the stellar models compatible with GYRE-lc.
- GYRE, which calculates the pulsation models compatible with GYRE-lc.
- MSG, which rapidly interpolates stellar spectra from a multidimensional grid for GYRE-lc.

GYRE and MSG are currently officially compatible with Linux and MacOS platforms only- Windows at your own risk!

Most importantly, GYRE-lc requires Python 3.6+.

To run GYRE-lc, you'll need the following Python libraries installed:

- https://pypi.org/project/h5py/, for HDF5 data management;
- https://pypi.org/project/f90nml/, for namelist handling;
- https://pypi.org/project/scipy/, for special math functions and operations;
- https://pypi.org/project/astropy/, for MESA model handling;

These components can be found via the PIP and Anaconda python package installers.

## 4.2 Setting up GYRE-lc

### 4.2.1 Download

Download the GYRE-lc source code, and unpack it from the command line using the tar utility:

```
tar xf gyre-lc.tar.gz
```

Set the GYRELC_DIR environment variable with the path to the newly created source directory; this can be achieved e.g. using the realpath command1:

```
export GYRELC_DIR=$(realpath gyre-lc)
```

You are ready to test and use GYRE-lc.

### 4.2.2 Test

**Note:** This project is under active development.

CHAPTER

FIVE

# INPUTS

## 5.1 Namelist Input Files

GYRE-lc reads parameters from an *inlist*, which is an input file that defines a number of "namelist" groups. Inlists are an input format designed for Fortran, and GYRE-lc is presently written entirely in Python. However, the Fortran-heavy workflow for GYRE and MESA users makes inlists a naturally convenient way to categorize groups of inputs for GYRE-lc.

### 5.1.1 &orbit

### 5.1.2 &observer

**Note:** This project is under active development.

CHAPTER

SIX

# OUTPUT

**Note:** This project is under active development.

CHAPTER

SEVEN

# TIDAL THEORY

This section provides an overview how GYRE-lc works as well as the underpinning tidal theory.

**Note:** This project is under active development.