pyLIMA: An Open-source Package for Microlensing Modeling. I. Presentation of the Software and Analysis of Single-lens Models

E. Bachelet1, M. Norbury1, V. Bozza2,3, and R. Street1

1 Las Cumbres Observatory, 6740 Cortona Drive, Suite 102, Goleta, CA 93117, USA
2 Dipartimento di Fisica “E.R. Caianiello,” Università di Salerno, Via Giovanni Paolo II 132, I-84084, Fisciano (SA), Italy
3 Istituto Nazionale di Fisica Nucleare, Sezione di Napoli, I-80126 Napoli, Italy

Received 2017 January 20; revised 2017 September 27; accepted 2017 October 2; published 2017 November 3

Abstract

Microlensing is a unique tool, capable of detecting the “cold” planets between ∼1 and 10 au from their host stars and even unbound “free-floating” planets. This regime has been poorly sampled to date owing to the limitations of alternative planet-finding methods, but a watershed in discoveries is anticipated in the near future thanks to the planned microlensing surveys of WFIRST-AFTA and Euclid’s Extended Mission. Of the many challenges inherent in these missions, the modeling of microlensing events will be of primary importance, yet it is often time-consuming, complex, and perceived as a daunting barrier to participation in the field. The large scale of future survey data products will require thorough but efficient modeling software, but, unlike other areas of exoplanet research, microlensing currently lacks a publicly available, well-documented package to conduct this type of analysis. We present version 1.0 of the python Lightcurve Identification and Microlensing Analysis (pyLIMA). This software is written in Python and uses existing packages as much as possible to make it widely accessible. In this paper, we describe the overall architecture of the software and the core modules for modeling single-lens events. To verify the performance of this software, we use it to model both real data sets from events published in the literature and generated test data produced using pyLIMA’s simulation module. The results demonstrate that pyLIMA is an efficient tool for microlensing modeling. We will expand pyLIMA to consider more complex phenomena in the following papers.

Key words: gravitational lensing: micro

1. Introduction

At time of writing, 3413 confirmed planets have been discovered in various planetary systems. The majority of these planets have been discovered through the transit (2693, with a large proportion thanks to the Kepler/K2 missions; Batalha et al. 2013; Coughlin et al. 2015) and the radial velocity (609) methods.4 While other techniques, including astrometry, microlensing, and direct imaging, have contributed relatively few detections so far, they probe complementary regions of parameter space. This large sample of planets has enabled several studies to derive planetary distributions (see, for example, Cassan et al. 2012; Batalha et al. 2013; Clanton & Gaudi 2014) constraining the formation and evolution of planets. The future space missions Transiting Exoplanet Survey Satellite (Ricker et al. 2015) and PLAnetary Transits and Oscillations of stars (Catala et al. 2011) will complete the statistics for this part of the parameter space and conduct studies of these new worlds to an unprecedented level of detail, knowing that the atmospheres of these new planets will be perfectly suited for spectroscopic transit follow-up (Ricker et al. 2014).

However, the transit and radial velocity methods are intrinsically most sensitive to planets orbiting close to their parent stars, while direct imaging cannot survey many targets at separations less than ∼10 au, leaving a gap in our understanding of the distribution of “cold” planets at separations between ∼1 and 10 au where they efficiently form. Fortunately, the sensitivity of the microlensing method peaks around these separations for galactic host stars and is independent of the lens brightness, meaning that it is uniquely capable of detecting objects anywhere along the line of sight from Earth to sources in the Galactic bulge. Microlensing’s capability to complete the planetary census was therefore identified as a priority in the last decadal survey (New Worlds, New Horizons, 2010). In addition, microlensing recently showed its potential to discover “free-floating planets” (Sumi et al. 2011). If these events prove to be caused by truly unbound planets, then the observed distribution of these objects requires that all planetary systems eject about two planets, at least 1 mag higher than predictions (Ma et al. 2016).

The Wide Field Infrared Survey Telescope mission has been predicted to detect around 3000 “cold” planets by surveying about 100 million stars in six observing windows of 70 days (Spiegel et al. 2015), for a total number of ∼37,000 microlensing events. One of the many challenges related to the mission is therefore the need to model microlensing events in a reasonable time.

However, while the modeling of single-lens events is relatively fast and quite easy, the analysis of multiple-lens events, such as planetary or stellar binary systems, is much more difficult. One key problem is encountered when summing the magnification of the background source images due to $N_l$ multiple bodies in a lensing system, when it is necessary to solve an $N_l^2 + 1$ polynomial for each source position. Moreover, the lens mapping presents singularities (called caustics) where the magnification of a point source diverges. The treatment of these singularities, discussed in detail in Wambsganss et al. (1992), Bozza (2010), and Dong et al. (2006), requires methods that are time-consuming and often depend on large cluster computing facilities.

In addition, the large parameter space that must be searched (at least seven parameters describe a binary lens) suffers from

---

4 https://exoplanetarchive.ipac.caltech.edu/index.html
several perfect degeneracies, discussed in Gould (2004), Thomas & Griest (2006), Dominik (2009), and Skowron et al. (2011), and a number of second-order effects (such as parallax and orbital motion) must be taken into account. Robustly identifying the best-fitting model can be very challenging, as recent examples can attest (e.g., OGLE-2013-BLG-0723; see Han et al. 2016).

To date, very few people have developed or used microlensing analysis software. Little systematic testing has been performed of the existing (proprietary) packages, for which little or no documentation is available, exacerbating the perceived “barrier to entry” for newcomers in the field. This stands in marked contrast with other areas of exoplanet research, where public codes for detection and analysis have been developed and systematic data challenges have been conducted to stimulate development (Moutou et al. 2005; Dumusque 2016; Dumusque et al. 2017). This standard of published testing and verification is not only good scientific practice, it also serves to build confidence in the results.

Our aim is therefore to develop a robust, well-tested, and publicly available software package for the modeling of microlensing events that is capable of answering the needs of future large-scale surveys, as well as existing ground-based programs. Our plan is to develop the python Lightcurve Identification and Microlensing Analysis (pyLIMA)5,6 software, which adopts the following philosophy:

1. High performance, capable of analyzing multiple-lens events within reasonable time frames.
2. Well tested.
3. Well documented.
4. Easy to use.
5. Open for community use and participation in development.

The first important step described in this paper is to develop a flexible code architecture that incorporates the fundamental elements of microlensing models and solution-finding modules, which will be built upon as more complex models are introduced in subsequent publications. While the point-source point lens (PSPL) and finite-source point lens (FSPL) models presented here are straightforward, their implementation depends on a number of important elements, including the combination of data for events observed from multiple facilities and initial-guess assumptions on source star colors. These aspects, while rarely discussed, are nontrivial and will form the foundation of the code for modeling more complex events.

In Section 2, we outline examples of how such software might be used to derive the design requirements for it, before introducing a general overview of pyLIMA’s architecture. The microlensing models are presented in Section 3, while the implemented fitting methods are detailed in Section 4. Section 5 presents the results obtained on both simulated and real data sets. We conclude and present future plans in Section 6.

2. pylIMA Description

2.1. Use-case Examples

Here we outline several common scenarios in which the modeling of microlensing events is required and infer the corresponding requirements placed on the design of pyLIMA.

It should be noted that while only single lenses are considered in the current version of the code, binary and higher-order multiple lenses will be incorporated in subsequent versions, which requires the code design to be flexible enough to handle a range of lens types.

Use case 1: “As a ground-based observer, I have time-series photometric measurements from a number of different telescopes of an ongoing microlensing event, and I wish to measure the observed and physical parameters of that event and plot the data overlaid with a model light curve in order to judge whether to continue observations.”

This scenario is typical of ground-based microlensing observations where data from multiple longitudinally separated sites must be combined to fully sample the event light curve. We can deduce several requirements from this case.

1. The software should accept multiple photometric–time-series data sets, potentially taken with different filters, to be combined into a single light curve, implying the need to properly align the data sets taking into account the different degree of blending (overlapping stellar point-spread functions (PSFs) or problems with or lack of absolute flux calibrations) in data from instruments with different pixel scales.
2. Because each data set could have its own format, the user needs to be able to describe this format during data input.
3. Some single-lens models, particularly those including parallax, require information on the location of each observatory, meaning that the user needs a way to specify this for each data set.
4. The software should output both a text summary of all model parameters and a light-curve plot with the model overlaid.

Use case 2: “As a new postgraduate, I have access to time-series photometry from a space-based telescope from a publicly accessible data archive that includes a number of known single-lens microlensing events. I have rudimentary knowledge of both Python and microlensing theory. I would like to determine the observed and physical parameters for all of these events, and I have a single desktop computer with <10 CPUs.”

This use case provides more insight into the circumstances under which the code will be used.

1. Users also have basic but not expert familiarity with microlensing theory, so the documentation should explain the steps of the process and include links to relevant publications.
2. It establishes the use of space-based photometry, telescope resources, and survey cadences.
3. It establishes limits on available computing power: fitting procedures cannot depend on large-scale parallelization or high-end processors to compensate for efficiency.

Use case 3: “As the postdoctorate or faculty-level operator of a survey facility producing time-series photometry, I would like to search for microlensing events within my data, which comprises a database of millions of light curves. I have rudimentary knowledge of Python and expert knowledge of microlensing theory, but I do not have the time to fit each light curve manually.”

We can outline several needs.

1. Establishes different user group that the documentation and user interface should accommodate.

5 https://github.com/cbachelet/pyLIMA
6 https://doi.org/10.5281/zenodo.997468
2. Requires that the code be usable as a library from which users can choose models and fitting algorithms to build software they develop for their own use.
3. Implies that the fitting procedures must be robust in returning sensible output even for light curves that may or may not contain a lensing event.
4. Requires that software be able to fit large numbers of light curves in an automated manner, which implies that it must be able to robustly establish reasonable initial values for the fitted parameters without human intervention.

Use case 4: “As a professor, my team and I would like to conduct a microlensing survey using telescope facilities available to my institution, which may be both ground- or space-based. I would like to maximize the science return of this project by simulating the data produced by different possible observing strategies, in order to optimize my use of the facilities.”

In addition to the requirements above, this implies several additional needs.

1. The user should be able to specify the characteristics of the telescope(s) and observing strategy to be used, including telescope aperture, location, observing cadence, etc.
2. The user should be able to specify the range of microlensing parameters for the events to be simulated
3. The software should be able to generate time-series photometric data sets with realistic noise characteristics and cadence

We note the utility of this simulation module in also providing a means to test the performance of the software itself.

2.2. Architecture

To be applicable in these use cases, pyLIMA’s architecture needs to be efficient and capable of rapidly analyzing large numbers of events in an automated fashion but also highly flexible to enable users to easily conduct detailed analyses. To address this, pyLIMA’s architecture follows three principles:

1. It should be possible to use the software to analyze large data sets in an automatic way.
2. The code should be constructed in a modular manner, so that users can implement analysis functions of their own design simply by adding the desired functions.
3. The code should be open source and structured in such a way that, as new theories and techniques become available, they can be easily integrated within this framework. Community contributions are welcomed.
4. The code should be well documented to enable new users to learn both microlensing theory and the software functions quickly and easily.

Python was adopted as the base programming language because it is free and available, it has been widely adopted in astronomy, and there are a number of excellent libraries already available (e.g., numpy, scipy, and Astropy). It is also trivial for Python to interface with libraries written in other languages, including C and Fortran. In the following, we briefly describe the main modules that are already implemented in pyLIMA. A more complete description can be found in the pyLIMA documentation.7

Events: The fundamental starting block of pyLIMA’s analysis centers around an Event, which is a class with a set of descriptive attributes, including the name, R.A., and decl. (used in the estimation of parallax or extinction along the line of sight). Since an Event may be observed from multiple telescopes and/or with multiple filters, it can have multiple data sets associated with it. Each data set is described as a separate instance of the Telescope class (see below). The principal function of the Event class is fit, which provides the user with a range of options to fit microlensing models to the data.

Telescopes: This module define the class Telescope. This class groups all the characteristics of an observatory that obtains data on a specific event. The user-specified name and filter distinguish different data sets, representing the filter used for the observations. Currently, pyLIMA supports light-curve data, which can be provided in units of flux or magnitude, as this class provides methods to automatically convert between these units. The user also specifies a location (“Space” or “Earth”) for the observatory, accepting altitude, longitude, and latitude in cases where this is needed for parallax estimation. If the telescope location is “Space,” then the spacecraft position is estimated through the JPL Horizon system, https://ssd.jpl.nasa.gov/horizons.cgi, in an automatic manner.

Microlensing: The user can select a number of different microlensing models to fit the data on a given event, described in the MLModel class from this module. The models currently supported are the Paczynski model (PSPL model) and FSPL model. Future versions of the software will provide more complex model options. The user is able to optionally include a range of second-order effects, including microlensing parallax, orbital motion of the lens, etc. The main function of this class is to compute the microlensing model associated with the parameters.

Microlfits: A number of well-documented procedures exist for identifying the best-fit model to a given data set; pyLIMA is structured to provide users with access to commonly used fitting methods as well as an easy way to implement their own, new techniques if they wish by adding to the MLFits class in the microlfits module. The user is able to indicate a preferred fitting method, which produces outputs appropriate to the method applied. The three methods that are already integrated in the package, based on the scipy package (Jones et al. 2001), are detailed in the Appendix. Note that several fits, with different models and/or methods, can be performed on the same Event.

Microloutputs: Upon completing an analysis, there are a number of diagnostic plots commonly used in microlensing. The user is able to produce these using the functions of the microloutputs module, which is based on the matplotlib package (Hunter 2007).

Microlsimulator: In addition to fitting real data on microlensing events, a number of important use cases require the ability to generate simulated data. For this purpose, pyLIMA provides the microlsimulator module, incorporating a series of functions that enable the user to produce realistic simulations of how events of given parameters would be observed from the observatories they specify. This module is based on the astropy package (Astropy Collaboration et al. 2013).

Extensive use of the astropy (Astropy Collaboration et al. 2013) and numpy (van der Walt et al. 2011) packages is made by pyLIMA; both of these are being widely adopted in the community, so the functions, attributes, and behavior of the

7 https://ebachelet.github.io/pyLIMA/
software are as familiar as possible. Figure 1 provides a schematic overview of the architecture, and the code excerpt below gives an example of these modules in use. More sophisticated examples are provided in the pyLIMA documentation.8

### First import the required libraries.
import numpy as np
from pyLIMA import event
from pyLIMA import telescopes
from pyLIMA import microlmodels

### Create an event object.
your_event = event.Event()

### Create two telescopes objects.
data_1 = np.loadtxt('./Survey_1.dat')
telescope_1 = telescopes.Telescope(
    name='Survey',
    camera_filter='I',
    light_curve_magnitude=data_1)

data_2 = np.loadtxt('./Followup_1.dat')
telescope_2 = telescopes.Telescope(
    name='Followup',
    camera_filter='I',
    light_curve_magnitude=data_2)

### Add the telescopes to your event.
your_event.telescopes.append(telescope_1)
your_event.telescopes.append(telescope_2)

### Construct the model.
model_1 = microlmodels.create_model('PSPL', your_event)

### Fit using Levenberg–Marquardt algorithm.
your_event.fit(model_1, 'LM')

### Producing outputs.
your_event.fits[0].produce_outputs()

(Continued)

---

**2.3. Good Coding Practice**

When developing software, it is good practice to maintain a clear structure and adopt consistent naming conventions for variables, functions, etc., to ensure that the code is readable. This pays off in the long term by making it substantially easier to maintain and upgrade. These and other considerations led to the introduction of the PEP8 standards.9 As pyLIMA is intended to be accessible to the entire community, we have adopted this standard for our development while allowing a certain degree of flexibility. We have assigned standard astronomical names for variables where they exist, even if they do not respect the PEP8 standards. An example of this is

---

8 https://github.com/ebachelet/pyLIMA/tree/master/examples

9 https://www.python.org/dev/peps/pep-0008/
the $ra$ variable, which in principle should be written as right_ascension.

The pyLIMA package is made publicly available via GitHub,\footnote{https://github.com/obachelet/pyLIMA} which also facilitates the community contributing to the software development. Extensive documentation is generated automatically based on markup within the code using SPHINX, making it easier to keep it up-to-date. The package also includes some Jupyter notebook-based examples as a guide to users.

However, as with any substantial package, it is always possible to inadvertently introduce bugs in the course of implementing new features. This can be addressed by developing functions that can be run automatically to systematically test all parts of the code and verify that it produces the expected results. Python provides a framework that enables users to develop these unit tests. Thanks to the Travis CI portal, these unit tests are run automatically through GitHub at each new code upgrade.

3. Implementation of PSPL and FSPL Models

3.1. PSPL Model

In microlensing, the lens object is generally a star crossing the line of sight between the observer and a background star. The majority of microlensing events detected are in the region of the Galactic bulge, because this densely populated background field presents the highest event probability. Even there, the microlensing optical depth is still low: $\tau \sim 10^{-6}$ (Udalski et al. 1994; Alcock et al. 2000; Sumi et al. 2011, 2013). If the lens is a single massive object, it deflects the light from the background source star into two images separated by several $q_{E}$, the angular Einstein ring radius (Gould 2000),

$$\theta_E = \sqrt{\kappa M \sigma_{\text{rel}}},$$

(1)

where $M$ is the total lens mass, $\sigma_{\text{rel}}$ is the lens-source relative parallax, and $\kappa$ is a constant. For typical events toward the Galactic bulge, $\theta_E$ is of order of a few milliarcseconds, leading to images that are indistinguishable with current capabilities. However, each image is magnified, and then the source flux increases by the total magnification factor $A(t)$, which for a PSPL model is given by (Paczyński 1986; Gould 2000)

$$A_{\text{PSPL}}(t) = \frac{u(t)^2 + 2}{u(t) \sqrt{u(t)^2 + 4}}; \quad u(t) = \sqrt{u_0^2 + \frac{(t-t_0)^2}{\theta_E^2}},$$

(2)

where $u(t)$ is the source-lens impact parameter and $u_0 = u(t_0)$ is the minimum impact parameter (linked to the maximum amplification $A_\text{max}$) at the time $t_0$. Also, $t_0$ is the Einstein ring crossing time,

$$t_0 = \frac{\theta_E}{\mu},$$

(3)

where $\mu$ is the relative proper motion between the source and lens (i.e., due to the proper motions of Earth, lens, and source).

It is interesting to note the following properties:

$$A_{\text{PSPL}}\begin{cases} \frac{1}{u} & u \to 0 \\ \frac{1}{u} & u \to \infty \end{cases}$$

(4)

The crowded fields where microlensing is typically observed suffer from a high degree of blending, meaning that the source star PSF usually overlaps those of other stars (including the lens itself). In combining data provided by telescopes that may have different pixel scales and seeing characteristics, it is necessary to take into account the different degree of blended flux $f_{i,\text{flux}}$ observed by telescope $i$, as well as the flux from the source, $f_{i,\text{source}}$, to give the total flux as a function of time, $f_i(t)$ (Gaudi 2012):

$$f_i(t) = f_{i,\text{flux}}(t) + f_{i,\text{source}}(t); \quad g_i = \frac{f_{i,\text{source}}}{f_{i,\text{flux}}}.$$  

(5)

Therefore, a PSPL model is described by $3 + 2n_i$ parameters, where $n_i$ is the number of observatories.

3.2. FSPL Model

The PSPL models assume that the source is a point. However, this hypothesis breaks down when the source-lens separation becomes small enough to be comparable to the normalized angular source radius $\rho = \frac{u_0}{\theta_E}$ (Yoo et al. 2004). For a typical source star in the Galactic bulge and a typical lens in the Galactic disk, $\rho \sim 10^{-3}$. This indicates that the effects of finite source size appear when an event becomes highly magnified (i.e., $u$ approaches zero). Following Witt & Mao (1994), Yoo et al. (2004), and Cassan et al. (2006), we use the high-magnification approximation to express the magnification of an extended source with a linear limb-darkening law (Milne 1921; An et al. 2002) for the wavelength $\lambda$,

$$A_{\text{FSPL}}(t, \lambda) = A_{\text{PSPL}}(t)[B_0(z) - B_1(z)]; \quad z = \frac{u}{\rho},$$

(6)

where $\Gamma_\lambda = \frac{2n_i}{1 - m}$ is the microlensing linear limb-darkening coefficient ($u_0$ is the Milne linear limb-darkening coefficient for the wavelength $\lambda$); $B_0(z)$ and $B_1(z)$ are completely defined in Yoo et al. (2004) and Cassan et al. (2006).

We implement a numerical table for $B_0(z)$ and $B_1(z)$ and their derivative terms within pyLIMA and use a linear interpolation to derive appropriate coefficients for each computation. The functions $B_0(z)$ and $B_1(z)$ are defined by incomplete elliptic integrals, which are slow and computation-intensive to calculate on the fly, explaining our choice of a pregenerated table. This infrastructure makes it straightforward to implement a higher-order limb-darkening law if needed in the future. As noted by Lee et al. (2009), this approach breaks when $\rho \geq 0.1$, because the approximation $A \sim \frac{1}{u}$ breaks for $u \geq 0.1$. Fortunately, it is extremely rare to observe $\rho > 0.05$ for events in the Galactic bulge. Lee et al. (2009) proposed a more robust algorithm, but this is more time-consuming for most purposes, since it requires the computation of double integrals. Nevertheless, we plan to make this algorithm available as an option to the user in future versions. Currently, the user can either specify values of $\Gamma_\lambda$ for each telescope manually or allow the code to calculate it automatically from the user-defined telescope filter, source star effective
temperature $T_{\text{eff}}$, and surface gravity $\log g$ using Claret & Bloemen (2011).

4. Fitting Algorithms

To date, we have implemented three main solution-finding techniques to fit events. Note that all methods are applicable to all types of models.

4.1. Implementation of Levenberg–Marquardt Algorithm

This method, called “LM” in pyLIMA, is based on the Levenberg–Marquardt algorithm (Levenberg 1944; Marquardt 1963). In Python, this function is part of the scipy.optimize.leastsq package, which itself is a wrapper of the C library MINPACK. Following Newton’s method, this algorithm uses the gradient to reach the local minimum. The objective function for this method is simply the $\chi^2$,

$$\chi^2 = \sum_i (d_i - m_i)^2 / \sigma_i^2,$$

where $d_i$ is the $i$th data point, $m_i$ is the predicted model value, and $\sigma_i$ is the uncertainty on $d_i$. For PSPL and FSPL, we decided to pass the analytical Jacobian matrix (i.e., the analytical derivatives of each parameter) to speed up the convergence.

In theory, the method should stop when the gradient derivatives reach zero, that is, when the algorithm reaches a critical point. However, the convergence of this algorithm is set differently. The algorithm stops if $f_{\text{red}}$ (the relative objective function improvement), $x_{\text{red}}$ (the parameter absolute difference), and/or $g_{\text{red}}$ (the angle between the Jacobian vectors and residuals) are below given thresholds. Note that all of these parameters can be modified easily by the user if needed.

The main difficulty of this method is the requirement for a good initial guess of the parameters. In pyLIMA, the user can provide these guesses, but we also developed a method for an automatic estimation. While this can be straightforward for our simulated data, where the noise was often negligible relative to the signal, it can be difficult for real data. Here we describe the methods we use to estimate each parameter.

1. $t_o$: This parameter may appear to be the simplest one to find, because a good initial guess should be the brightest point in the light curve. However, this approach can easily fail for noisy data sets, so we choose a different approach. Note that we perform this method for all available light curves present in the event. For each light curve, we first temporarily remove data points with high photometric errors (i.e., $\sigma \leq \min(0.1, \bar{\sigma})$). Next, we construct a smooth light curve using a Savitzky–Golay filter of degree one on the photometry. Then, we perform a loop by selecting points with a flux higher than the actual median and presenting the best photometry as previously described. The loop stops if there are less than 100 points or if the standard deviation of the time is less than 5 days. The $t_o$ for each light curve is then the median in time of the remaining points. The final $t_o$ value is chosen to be the mean, weighted by the magnitude uncertainty, for all of the telescope’s data sets. In fact, weighting the different $t_o$ estimation with the magnitude uncertainty gives more weight to data sets with better photometric precision, where one would expect this algorithm to have greater success.

2. $f_s$: Second, we try to find the baseline flux for the survey telescope. This is done in a loop by selecting points below (or within the error bar of) the median flux. Then, by assuming no blending, the source flux for the survey data set is just equal to the baseline flux.

3. $u_o$: Knowing $f_s$, it is possible to compute $A_{\text{max}} = \max(\text{flux})/f_s$, the maximum magnification in the survey data set at the point selected by the $t_o$ estimation, and then $u_o = \sqrt{\frac{2}{A_{\text{max}}} + 1} - 1$.

4. $t_{\text{fit}}$: The estimation of $t_{\text{fit}}$ is made using three different methods, namely the value being the median of the three. The first method uses the fact that when $A = A_{\text{max}}/2$, then $t_{\text{fit}} = \frac{1}{2} \left( \frac{1}{A_{\text{max}}} + 1 \right)$. The second method uses the fact that $A(t_{\text{fit}}) = \frac{u_o^2}{(u_o^2 + 3)(u_o^2 + 5)}$. The algorithm tries to find the points closest to this value. The last estimation is a very rough approximation that finds the closest point after/before $t_o$ consistent with the baseline flux and uses this as a $t_{\text{fit}}$ approximation.

5. $f_s$ and $g$: These parameters are found for each telescope by using the parameters estimated above and conducting a linear fit. Note that if $f_s < 0$, then the minimum flux value is returned as $f_s$ and $g$ are set to zero.

4.2. The Differential Evolution

This method is a global optimizer originally presented by Storn & Price (1997). This method is really robust for a vast range of problems—see, for example, Vesterstrom & Thomsen (2004)—but is more time-consuming than LM. This method is called “DE” within pyLIMA. Again, we use the $\chi^2$ as the objective function. The convergence condition is set by the parameter $\text{tol} = \chi^2 > 1$, where $\chi^2$ is the mean of the objective function for all population members and $\sigma$ is its standard deviation. We set this parameter to $10^{-5}$. This method is useful to find robust guesses, which are then used in the LM method presented previously. Note that we slightly changed our fitting strategy in this method: $f_s$ and $g$ are not considered standard parameters (i.e., a full differential evolution search), but they are computed for each step as a linear regression of the flux over the magnification. This is due to the fact that if these parameters are set free, the parameter space volume increases dramatically for a small range of potential correct values.

4.3. A Monte Carlo Markov Chain Algorithm

It is useful to generate the posterior distributions for an event and thus be aware of all plausible models. We implemented MCMC based on the Python module emcee (Foreman-Mackey et al. 2013). This method requires good initial parameters, which are produced by the method DE in pyLIMA. The user can select whether $f_s$ and $g$ are used as MCMC parameters or computed using a linear regression of the telescope’s flux versus the magnification. This method can be called the “MCMC” option. Note that this method tries to maximize the log-likelihood, so we set the objective function in this case to be

$$\log L = -\chi^2/2,$$
4.4. Speed Performance

As can be seen from Table 1, the median times for the Ground, Space, and FSPL fits are of order 0.01, 0.02, and 0.1 s, respectively. The fits to the Space data sets are slower owing to them having higher numbers of points on average. The FSPL data set is 10 times slower for two main reasons. First, the magnification computation is slower due to the linear interpolation necessary to take account of finite-source effects. Second, there are two data sets, leading to a more complex model and more data points. Note that the DE method converges in general in about 15 s, which is reasonable given the volume of parameter space it explores. For reference, all of our modeling was conducted using an Intel Core i7 CPU 860 @ 2.80 GHz, 8 processor and 16 GB of memory.

5. pyLIMA Fits on Various Data Sets

As it is crucial to understand how a modeling code performs, we next decide to conduct various tests on both simulated and real data sets. This is a standard method used to test modeling codes on planet detections using the radial velocity and/or transit method (Borsato et al. 2014; Díaz et al. 2014; Dumusque 2016; Dumusque et al. 2017). The details of the simulations can be found in Appendix A, where we also define the quality fits metric.

5.1. The Ground and Space Data Sets

The Σ_u results for the Ground and Space data sets can be seen in Table 1 and Figures 2 and 9. From these metrics, it can be seen that pyLIMA accurately recovered the injected models. However, several trends are observed. First, it appears that the fits to the Space data set are more accurate, though at the expense of requiring extra computation time (pyLIMA is about 1.8 slower for this data set).

The pyLIMA fits on various data sets are intrinsically very difficult to fit, for example, when the observing windows are quite small relative to the event duration. This kind of problem is likely to impact the future space mission. The WFIRST microlensing mission, for instance, will consist of 6 yr of 70 days of observing windows. This is why we limited the observing window to 90 days for the Space data sets. We therefore split the light curves into five distinct categories:

1. **Regular:** The light curve does not present any of the conditions listed below.
2. **No peak:** The light-curve peak occurs outside the observing window.
3. **High blending:** The light curve is highly blended in the model (g > 1).
4. **No baseline:** The light curve never reaches its baseline during the observing windows (t − t_s < τE ∀ t).
5. **Hard:** The light curve presents at least two categories listed above.

An example of “Regular” and “Hard” light curves is visible in the Figure 3 light curves for the Space data set.

We computed the β metric for each subclass and present our results in Table 2 and Figures 4 and 11. Again, the behavior of the fits is similar for both data sets, and we can characterize the software’s performance in each category:

1. **Regular:** For these light curves, pyLIMA accurately recovers the model parameters without any particular trends in the Σ_u distributions and without any failures.
2. **No peak:** For this subset, it is interesting to note that the parameter τ_s is relatively well estimated, without particular trends. However, there are serious trends in the u_o, f_s, and g distributions. A lot of fits for these events converge to very low u_o values (the orange peak around Λ_u = 1 in Figures 4 and 11). In other words, pyLIMA predicts unrealistically high magnification for these events. These results confirm previous work, which indicated that χ^2 minimization algorithms tend to overestimate the magnification if the peak is not observed (see, for example, Albrecht 2004 or Dominik 2009). This may be related to overfitting. In these cases, the magnification is slightly overestimated, leading to an overestimation of the blending g and so to an underestimation of f_s. This occurs because the light-curve baseline flux, f_{baseline} = f_s(1 + g), is in general well constrained (the orange peak around Λ_{g, f} = 1 and the asymmetry on the left for the orange Λ_{f} distribution). We found two fit failures for the Space data set in this category, but considering that Δχ^2 is equal −0.41 and −0.85, respectively, we do not consider these failures as critical.
3. **High blending:** We note that for a large fraction of these light curves, the fitted model indicated no blending (i.e., the green peak around Λ_{g} = 1). This underestimation of the blending is linked to an overestimation of Λ(t), leading to skewed distributions for Λ_u(left) and Λ_o (right). We notice that this category contains a significant number of fit failures for both data sets: 0.9% (Ground) and 2.57% (Space). This is due to light curves with a very low signal-to-noise ratio (i.e., the light curve is nearly flat). These events are particularly difficult to fit, with median blending values of 7.8 and 5.40, respectively. Note that all of these failures come from the LM method.
4. **No baseline:** It is interesting to note that this category does not present particular trends. Intuitively, one might expect that the f_s parameter needs the baseline observations to be well constrained. As shown by Dominik (2009),

[Note: The rest of the text continues with additional analysis and discussions.]
Table 1
pyLIMA Fit Results for the Three Simulated Data Sets

| Data Sets | Computation Time (s) | $\Sigma_{m}$ | $\Sigma_{n}$ | $\Sigma_{il}$ | $\Sigma_{j}$ | $\Sigma_{x}$ | Method DE |
|-----------|----------------------|--------------|--------------|---------------|--------------|-------------|-----------|
| Ground    | 0.013 0.003 3.306    | 62.1% 94.6% 86.6% | 94.6% 65.5% 87.5% | 95.1% 65.4% 88.7% | 96.0% 66.3% 87.0% | 94.2% 63.1% 87.6% | 0         |
| Space     | 0.023 0.005 39.618   | 75.1% 96.2% 99.6% | 96.3% 75.3% 99.5% | 96.3% 74.6% 96.7% | 99.7% 75.8% 96.0% | 99.3% 73.3% 96.0% | 1         |
| FSPL      | 0.103 0.019 348.023  | 50.3% 77.7% 88.6% | 74.6% 51.8% 82.5% | 50.5% 77.1% 88.1% | 50.4% 76.5% 87.4% | 50.1% 76.0% 87.1% | 853       |

Note. Computation time columns indicate the median, minimum, and maximum fit time per light curve. Each $\Sigma_{x}$ column presents the percentage of fits where $[\Sigma_{x}]$ is less than 1, 2, and 3, respectively. The space data set presents the best results, due to the higher number of observations. The FSPL data set, on the contrary, presents the worst results. This is due to an intrinsically more complex model than the PSPL model, as well as difficulties due to the simulations; see text.
Therefore, if this region is well sampled, which is the case for this subset, $t_E$ is relatively well constrained.

5. Hard: As expected, this subset presents the worst fitting results. The ratio of failures is the highest for both data sets, and the fitted parameters are often off from the model (i.e., $\Delta \chi^2 > 3$). We checked the failed-fit events individually and did not find any critical fit (i.e., a fitted model totally in disagreement with the data). The slight disagreement between models and fits (median values of $\Delta \chi^2$ are $-5.9$ and $-3.1$ for the Ground and Space data sets, respectively) comes from an underestimation of the blend flux (median values of injected blend flux are 27.4 and 20.6).

Based on these results, overall, pyLIMA’s fitting procedure is highly reliable. The fitted models and estimated uncertainties accurately represent the data for the majority of events. We carefully analyzed the problematic cases and found that instances of poor model fits are a consequence of the intrinsic nature of the data (i.e., the light-curve category) rather than a software problem.

**Figure 2.** The $\Sigma_x$ distributions for the Ground data set. The $x$-axis represents the model parameters $x$, and the $y$-axis represents $\Sigma_x$ limited to $-5$ to $+5$. The color scales indicate the $\log_{10}(N)$, where $N$ is the total number of events in the corresponding bin. The bottom right plot is the $\chi^2$/dof distribution. A high fraction of fits are consistent with the injected models (i.e., $|\Sigma_x| < 3$). However, several trends can be observed, especially in the $t_o$, $u_o$, and $\chi^2$/dof distributions.

**Figure 3.** Left: example of a “Regular” light curve. The injected model (red) and the fit (orange) are indistinguishable in this case. All metrics indicate a successful fit. Right: example of “Hard” light curve. Various metrics indicate that the fit is nonoptimal. In this case, the event is not constrained enough by the observations.
5.2. The FSPL Data Set

Tables 1 and 3 present the $\Sigma_3$ results for this data set (distributions can be seen in Figures 10 and 12). For this data set overall, pyLIMA performed well, though the fitted model parameters are somewhat less accurately derived than in the previous section. The percentage of events inside the $\Sigma_3$ windows is lower, and pyLIMA required the DE method more frequently. The first trend can be explained by the fact that the fitted parameter errors are ~10 times smaller. For instance, the median values for the $t_o$, $u_o$, and $t_E$ errors are $[10^{-3}, 4 \times 10^{-4}, 9 \times 10^{-2}]$ for the FSPL data set, whereas they are $[2 \times 10^{-2}, 2 \times 10^{-2}, 4 \times 10^{-1}]$ for the Space data set. Another explanation for the relatively low success of the $\Sigma_3$ criterion is that for some events, the finite-source effect is negligible (i.e., when $\rho < u_o$). In these cases, the light curves can be equally well fitted with a PSPL model, causing the parameters $u_o$ and $\rho$ to converge far from the model. The second trend is due to the initial guess for $\rho$ produced by the LM method. It is set to 0.05, which is very naive and can be far from the correct solution. Note that several alternative initial-guess values were tested, such as $\rho = \frac{2}{\sqrt{A_{\text{max}}}}$ (Witt & Mao 1994) or $\rho = u_o$, but this gave similar results due to the rough estimation of $u_o$. It is highly probable that this starting point is too far from the solution, leading to a nonsatisfactory LM fit and causing the software to apply the DE method.

We report the results for the $\Lambda_X$ metric in Table 4. Note that some successful fits have $|\Lambda_X| > 0.1$. This is due to two factors. Some light curves present very weak finite-source effects. In this case, the light curve can be equally fit with a PSPL model. The second case is linked to the following

![Image of FSPL Data Set](image)

**Figure 4.** The $\Lambda_X$ distributions for the Space data set. Each category is represented by a color and a row: first row (blue) is Regular, second row (orange) is No peak, third row (green) is High blending, fourth row (yellow) is No baseline, and fifth row (black) is Hard. It is clear that unsuccessful fits are due to problematic light curves rather than pyLIMA fitting routines.

### Table 2

| Data Set | Category     | $N_{\text{event}}$ | $t_o$ | $u_o$ | $t_E$ | $f_j$ | $g$ | $\forall X$ | $\Delta \chi^2 < 0$ |
|----------|--------------|---------------------|-------|-------|-------|-------|----|---------------|-------------------|
| Ground   | Regular      | 6017                | 99.9% | 79.1% | 88.1% | 69.4% | 24.2% | 23.8%         | 0.0%              |
|          | No peak      | 1315                | 66.7% | 32.6% | 69.7% | 33.1% | 4.5%  | 3.0%          | 0.0%              |
|          | High blending| 2109                | 98.6% | 49.1% | 59.7% | 43.0% | 36.0% | 35.0%         | 0.9%              |
|          | No baseline  | 93                  | 100%  | 75.3% | 76.3% | 72.0% | 36.6% | 36.6%         | 0.0%              |
|          | Hard         | 466                 | 37.3% | 11.2% | 21.5% | 8.6%  | 6.0%  | 5.8%          | 1.93%             |
| Space    | Regular      | 5180                | 100%  | 93.3% | 96.0% | 86.4% | 44.5% | 44.5%         | 0.0%              |
|          | No peak      | 2155                | 86.4% | 52.5% | 86.4% | 48.8% | 8.0%  | 6.4%          | 0.09%             |
|          | High blending| 1827                | 99.8% | 68.1% | 73.7% | 60.8% | 54.6% | 54.5%         | 2.57%             |
|          | No baseline  | 104                 | 100%  | 69.2% | 69.2% | 69.2% | 31.7% | 31.7%         | 0.0%              |
|          | Hard         | 734                 | 41.1% | 12.7% | 21.7% | 11.6% | 7.8%  | 6.3%          | 6.13%             |

Note. The $\forall X$ column indicates the percentage of events satisfying $|\Lambda_X| < 0.1$ for all parameters. The last column indicates the percentage of fit failures (i.e., as a fraction of the total number of events in that category) according to the last metric.
tends to be small, $1/t = \frac{A_N}{\text{failures}}$ more likely to be due to insufficient information inside the light curves that to making any fit problematic. From Equation 6, it is clear that one key fit parameter is $z_o = \frac{u_o}{\rho}$. In fact, it is more relevant to define $\min_z = \min(z)$, the minimum impact parameter sampled by the observations, divided by $\rho$. A distribution of this parameter is presented in Figure 5. We can see that pyLIMA fails when $\min_z > 1$, in the area where finite-source effects tend to be smaller. It is important to note that more than half of the failures (51.6%) occur for an area containing only 17.1% of the total number of events. For failures occurring for $\min_z < 1$, the situation is more complicated and linked to multiple factors, including photometric noise, low sampling, and local minima. To conclude, it is important to emphasize that the DE method is much more reliable (6.5% of failures) than the LM method (18% of failures).

This means that if $z$ tends to be small, $u_o$ is not constrained whenever $\rho$ is (this explains the difference between the $u_o$ and $\rho$ columns in the first line of Table 4). For the fit failures, it is obvious from Table 4 that this comes from an incorrect estimation of $u_o$ and/or $\rho$. However, as we saw previously, some light curves could have very weak finite-source effects, making any fit problematic. From Equation 6, it is clear that $A_{FSPL} \sim \frac{2}{\rho z}; z \to 0$.

\begin{equation}
A_{FSPL} \sim \frac{2}{\rho z}; z \to 0.
\end{equation}

Figure 5. The $\min_z$ normalized distributions for successful fits (blue) and fit failures (green). The significant change for $\min_z \sim 1$ indicates that bad fits are more likely to be due to insufficient information inside the light curves that to fitting routine misbehavior.

### Table 3

| Data Sets | $\Sigma_o$ | $\Sigma_{o2}$ | $\Sigma_r$ | $\Sigma_{r2}$ |
|-----------|------------|-------------|-----------|-------------|
| FSPL      | 52.8%      | 74.8%       | 82.0%     | 48.0%       |
|           | 75.2%      | 85.7%       | 48.8%     | 75.4%       |
|           | 85.7%      |             |           |             |

### Table 4

|$\Lambda_i$| Success Ratio for the FSPL Data Set

| Category | $N_{\text{rest}}$ | $t_o$ | $u_o$ | $t_E$ | $\rho$ | $\sigma_{t}$ | $g_{t_1}$ | $g_{t_2}$ | $g_{\rho}$ | $\forall z$ |
|----------|-------------------|-------|-------|-------|-------|-------------|-----------|-----------|------------|-------------|
| Success  | 7497 (798)        | 99.4% | 73.0% | 98.8% | 89.4% | 98.3%       | 75.7%     | 95.0%     | 41.5%      | 26.8%       |
| Failure  | 1650 (55)         | 98.9% | 6.0%  | 77.6% | 1.5%  | 72.6%       | 36.1%     | 48.7%     | 14.1%      | 0.4%        |

Note. The $\forall z$ column indicates the percentage of events satisfying $|\Lambda_i| < 0.1$ for all parameters. Numbers in parentheses in the first column indicate the number of events where the DE method was used. The low success ratio for $u_o$ and $\rho$ in the failure category is due to the problematic events already seen in Figure 6: these events present weak finite-source effects and/or problematic data. Then, it is possible to find competitive models without finite-source effects.

Figure 6. Maximum amplification $A_o$ (left) and Einstein ring crossing time $t_E$ (right) for the OGLE-II survey microlensing events from various studies vs. pyLIMA$_{LM}$. The dark lines are linear fits where coefficients are visible in Table 5. The gray shaded areas indicate the 1σ fit errors.

5.3. The OGLE-II Survey

To challenge pyLIMA, we decided to model microlensing events from the OGLE-II (Udalski et al. 1994; Udalski 2003; Szymanski 2005) survey. We selected events detected in the Galactic bulge in the three seasons 1998, 1999, and 2000. This led to 41, 46, and 75 light curves, respectively. We performed two runs of PSPL modeling using the LM and MCMC methods.

We discarded light curves that presented ambiguous behavior or physical phenomena not yet incorporated into pyLIMA’s functionality: binary microlensing, falsely classified as microlensing, variable stars, high photometric noise, etc. We rejected 5, 14, and 10 light curves from the respective seasons, which led to a subset of 133 total light curves. We reviewed the literature and found several studies that also examined this data set. Udalski et al. (2000) (U2000) and Wozniak et al. (2001) (W2001) fitted events from the 1998 and 1999 seasons; Tsapras et al. (2003) (T2003) analyzed all three seasons. Albrow et al. (2000) (A2000) modeled the specific event OGLE-1998-BLG-14 in order to estimate its planet sensitivity.
Table 5
Comparison of pyLIMA with Various Studies on the OGLE-II Survey

| Study          | N\textsubscript{event} | 3\Sigma\textsubscript{tu} | SFPL  | \Sigma\textsubscript{tu} | \mathbf{t}_E | \mathbf{t}_E | A\textsubscript{c} | A\textsubscript{c} |
|----------------|--------------------------|-----------------------------|-------|-----------------------------|-------------|-------------|----------------|----------------|
| pyLIMA\textsubscript{MCMC} | 133                      | 100\%                       | 94.7% | 1.019(0.009)                | 0.269(0.194) | 100%        | 1.021(0.010) | 0.222(0.055) |
| T2003          | 132                      | 95.5\%                      | 94.7% | 0.924(0.029)                | 0.458(0.605) | 96.2%       | 1.030(0.039) | -0.412(0.210) |
| U2000          | 52                       | 90.4\%                      | 92.3% | 0.930(0.044)                | 0.448(0.955) | 82.7%       | 0.948(0.066) | -0.309(0.346) |
| W2001          | 43                       | 95.4%                      | 95.4% | 0.949(0.055)                | 1.020(1.358) | 90.7%       | -             | -              |

Note. The 3\Sigma\textsubscript{tu} columns indicate the percentage of events where |\Sigma\textsubscript{tu}| < 3. The \(a\) and \(b\) columns indicate the coefficients (and errors) of linear fits (i.e., \(y = ax + b\)) performed in Figure 6. An asterisk indicates that the linear fit did not converge. Note the excellent agreement between the two independent fitting algorithms used in pyLIMA.

Table 6
ARTEMiS vs. pyLIMA Fit Results for the OGLE 2015 Season

| \(\Sigma\textsubscript{tu}\) | \(\Sigma\textsubscript{tu}\) | \(\Sigma\textsubscript{tu}\) | Anomalous | ARTEMiS Failures | pyLIMA failures | Unknown |
|-----------------------------|-----------------------------|-----------------------------|------------|------------------|----------------|---------|
| 83.7\%, 89.8\%, 92.3\%     | 78.5\%, 89.4\%, 93.1\%     | 82.1\%, 91.4\%, 94.5\%     | 136        | 27               | 22             | 128     |

Note. The percent quantiles are computed on the total of 2145 events. ARTEMiS and pyLIMA agree on a vast majority of events, and only a small fraction (i.e., 150/2145) of events are challenging for pyLIMA, if we consider that all events in the category “unknown” come from a bad pyLIMA fit.

Table 7
The pyLIMA Fit Results for the Three FSPL-like Events.

| Event            | Publication                          | \(t_E - 2450000\) | \(u_E\) | \(t_E\) | \(\rho\) | \(\chi^2\text{(dof)}\) |
|------------------|--------------------------------------|------------------|--------|--------|--------|--------------------------|
| MOA-2007-BLG-400 | Dong et al. (2009)\textsuperscript{a} | 4354.58107       | 2.5 \times 10^{-4} | 14.41  | 0.00326 | 1872.49(773)             |
|                  | pyLIMA\textsubscript{LM}             |                  |        |        |        |                          |
|                  | pyLIMA\textsubscript{TE}             |                  |        |        |        |                          |
| MOA-2008-BLG-310 | Janczak et al. (2010)\textsuperscript{b} | 4656.39975 \pm 5. 10^{-5} | 3.0 \times 10^{-3} | 11.14  | 0.00493 | 2.5 \times 10^{-3}      |
|                  | pyLIMA\textsubscript{LM}             | 4656.39904 \pm 3.9 10^{-5} | 2.84 \times 10^{-5} | 11.50  | 0.00475 | 2.7 \times 10^{-4}      |
|                  | pyLIMA\textsubscript{TE}             | 4656.39904 \pm 3.9 10^{-5} | 2.84 \times 10^{-5} | 11.50  | 0.00475 | 2.7 \times 10^{-4}      |
| OGLE-2013-BLG-0446 | Bachelet et al. (2015)\textsuperscript{c} | 6446.04790 \pm 3. 10^{-5} | -4.21 \times 10^{-4} | 76.9   | 0.000522 | 1.0 \times 10^{-6}      |
|                  | pyLIMA\textsubscript{LM}             | 6446.04660 \pm 1.2 10^{-5} | 4.0 \times 10^{-4} | 80.41  | 0.000495 | 4.6 \times 10^{-5}      |
|                  | pyLIMA\textsubscript{TE}             | 6446.04660 \pm 1.2 10^{-5} | 4.0 \times 10^{-4} | 80.09  | 0.000497 | 4.6 \times 10^{-5}      |

Note. Results are in good agreement with the literature.
\(\textsuperscript{a}\) The parameters are from the binary “Close” model.
\(\textsuperscript{b}\) The parameters are from the binary “Wide” model.
\(\textsuperscript{c}\) The parameters are from the FSPL model in their Table 3.

The results of our analysis are plotted in Figure 6 and shown in Table 5. It is interesting to underline that our studies produced various results for \(A_c\) and \(t_E\), which are the key fit parameters, with the exception of the two methods used by pyLIMA. U2000 and T2003 tended to systematically underestimate \(t_E\) and \(A_c\) in comparison with pyLIMA. This is understandable because they did not include blending flux in their fits. W2001 included blending-flux fitting, leading to better agreement with pyLIMA for \(t_E\). For the special case of OGLE-1998-BLG-14, the results of A2000 and the pyLIMA LM method are in excellent agreement: \(\Sigma t_E = -0.10\), \(\Sigma A_c = 0.10\), and \(\Sigma A_c = 0.10\). Note that we use the results of the last columns of Table 2 in A2000, where the fit was made using OGLE and PLANET follow-up data.

5.4. Comparison with the ARTEMiS System

The ARTEMiS pipeline (Dominik et al. 2008) is one of the best real-time fitter and anomaly detectors. It is used by the MiNDSTEp group to prioritize their follow-up targets. It performs PSPL fits on all data sets available for each event. We collected all microlensing events from the OGLE 2015 season (and all associated data sets available through the ARTEMiS portal) and compared our fits with those of ARTEMiS. A total of 2145 events were fitted, and the results are presented in Table 6. This time, the \(\Sigma t_E\) criterion is computed as follows:

\[
\Sigma t_E = \frac{\chi_{\text{ARTEMiS}}^2 - \chi_{\text{pyLIMA}}^2}{\sigma_E_{\text{pyLIMA}} + \sigma_E_{\text{ARTEMiS}}},
\]

We visually inspected each light curve where ARTEMiS and pyLIMA disagreed (i.e., \(|\Sigma t_E| > 3\)) and found four different possibilities.

1. Anomalous: The event is clearly not a single-lens microlensing event. It could be due to a binary source, binary-lens microlensing event, or cataclysmic variable, for example.
2. **Unknown**: There is no obvious sign as to why ARTEMiS and pyLIMA disagree. This could be due to an under-estimation of errors in both algorithms.

3. **ARTEMiS failures**: The ARTEMiS fit does not agree with the observations.

4. **pyLIMA failures**: The pyLIMA fit does not agree with the observations.

We can see that ARTEMiS and pyLIMA agreed about the vast majority of events. The number of failures was low: 1.3% and 1.0% for ARTEMiS and pyLIMA, respectively. We note that majority of pyLIMA failures came from a bad estimation of \( t_c \). All of these failures were made using the LM method, and despite the sanity check step, pyLIMA treats these fits as acceptable.

### 5.5. Published FSPL Events

As a final test, we wanted to examine pyLIMA’s performance for FSPL events. We found public data for two events: MOA-2007-BLG-400 (Dong et al. 2009) and MOA-2008-BLG-310 (Janczak et al. 2010). These two events present planetary anomalies close to their peaks. However, due to the large source sizes relative to the central caustic sizes (i.e., \( \frac{\rho}{\varrho} < 2 \)), these anomalies have low amplitudes (\( \lesssim 10\% \)), and thus an FSPL model is a reasonable fit for almost the entire light curve. Finally, we also fit OGLE-2013-BLG-0446 (Bachelet et al. 2015), which is a similar event where the authors demonstrated that the hypothetical small planetary signal (\( \lesssim 1\% \)) is probably due to red noise.

For MOA-2007-BLG-400 and MOA-2008-BLG-310, we used the values of \( \Gamma_H \) given by the authors to compute \( \Gamma_H \) using Claret & Bloemen (2011). Note that for these events, the authors used a square-root limb-darkening law, which can explain some fitting differences. For OGLE-2013-BLG-0446, we used \( \Gamma_H \) values as given in the publication. These results can be found in Table 7.

Again, the results are in good agreement with the literature. The \( \chi^2 \) of the pyLIMA fit for OGLE-2013-BLG-0446 is higher, but these calculations are made without any error-bar rescaling. As shown in Bachelet et al. (2015), some telescopes need high error-bar rescaling to obtain a normal distribution of the fit residuals.

### 6. Conclusions and Recommendations

In this paper, we describe the first phase of the development of pyLIMA, an open-source microlensing analysis package. We present the method and tools we used to build a flexible and easy-to-use architecture accessible to all users. We have conducted a series of tests to assess the reliability of pyLIMA fitting of two single-lens models on both simulated and real data sets. To do so, we define three different metrics to assess the quality of the fits.

The results of the simulated PSPL events are satisfying in terms of fit convergence (\( \geq 99\% \)), parameters, and uncertainty estimation. The complete analysis of our simulations reveals several trends already discovered in the literature (Albrow 2004; Thomas & Griest 2006; Dominik 2009). This study also highlights potential issues for future microlensing space missions due to their relatively short observing windows (i.e., \( \leq 100 \) days) every year. We also use pyLIMA to fit the OGLE-II data set and find good agreement with four previous studies.

Finally, we find excellent agreement between pyLIMA and the ARTEMiS system on the OGLE-IV 2015 microlensing season (i.e., \( \geq 90\% \) match).

We also implement the FSPL model and run similar tests. Despite good agreement, we find a higher failure occurrence (\( \sim 18\% \)). This is happens for two main reasons. The first one is the difficulty of finding a “good enough” estimation of \( \rho \) when using the LM method. The second one is the existence of some difficult light curves where the finite-source effects are low (i.e., \( z > 1 \)). The failure rate drops significantly when the DE method is used (6.4% versus 18%).

Based on these results, the authors make some recommendations to users. The fitting of light curves with PSPL models can be made using the LM method with a good expectation of fit convergence. This enables the user to study large data sets in a reasonable amount of time. In case of doubts or the failure of a fit, the DE method should be used. Fitting FSPL models needs a bit more caution. If the light curve is well sampled and exhibits strong finite-source effects, the LM method should converge. In other cases, users are encouraged to use the DE method. Finally, the authors recommend the use of the MCMC method only in two cases. First, the MCMC method is used to derive the event parameter posterior distributions. This give a more complete view to the parameter space than the LM method. Second, if the event is poorly constrained, it is likely that the Fisher matrix inversion will return unrealistic errors. Then, the MCMC approach should return a more comprehensive view of the problem. To conclude, we would like to note that this has a cost: the MCMC method is about 2000 times slower than the LM or DE methods, since it requires the computation of thousands of models.

The authors thank the referee for useful comments. The authors express their gratitude to the microlensing observer teams, especially the OGLE survey, for kindly granting access to their photometry. This research has made use of NASA’s Astrophysics Data System. Work by E.B. and R.A.S. is supported by NASA grant NNX15AC97G.

### Appendix A

#### Description of Simulations

#### A.1. Description of the Data Sets

**A.1.1. The Ground Data Set**

The first data set, called Ground, mimics PSPL observations from a unique terrestrial survey such as OGLE (Ulacinski et al. 2002) or MOA (Bond et al. 2001). The observing window was arbitrarily set to [0, 182] days for each light curve. This approximately corresponds to a 1yr observation of the Galactic bulge from the southern hemisphere. We first implemented a night/day cycle, and we selected the number of exposures per night from a uniform distribution between 1 and 30 to mimic survey observations of different fields with different cadences. To make these simulations more realistic, we implemented potential bad weather. Ten percent of nights were randomly selected to be “bad weather” (no observations). We also implemented a full moon avoidance window (5 days in a row).
However, to ensure that a microlensing event would be detected, we ensured that at least two points were observed around the event peak \( t_e \). We decided to implement Poisson and red noise sources, the latter with a sum of low-amplitude (\( \leq 5\% \)) and low-period (\( \leq 10 \) days) sinusoidal functions. The photometric precision was limited to a minimum of 1%. We selected \( u_0 \) from a uniform distribution between \( 10^{-4} \) and 1, while \( t_E \) was selected from a log-normal distribution (\( \mu = 2.8, \sigma = 0.9 \)), which is a rough approximation of the expected one (see, for example, Sumi et al. 2011). We generated \( t_e \) from a uniform distribution between \(-t_E\) and \( 182 + t_E \) days, and the source and blend magnitudes were produced from a normal distribution of \( (\mu = 18, \sigma = 1.5) \) and \( (\mu = 19.4, \sigma = 1.6) \), respectively.

### A.1.2. The Space Data Set

The second data set, called Space, reproduces PSPL observations from a space-based survey such as WFIRST (Spergel et al. 2015) or EUCLID (Laureijs et al. 2011). The data were simulated in the same way as above, except that we implemented continuous coverage (no night/day cycle), restricted the photometric precision to a minimum of 0.1%, set the observing windows to \([0, 90] \) days, and fixed the observation sampling to 30 minutes. This is an approximation of the expected duration of the annual WFIRST bulge survey. We also removed red noise effects from our simulations.

### A.1.3. The FSPL Data Set

For this data set, we simulated two telescopes (Survey and Follow-up) for each PSPL event. This represents the fact that finite-source effects are detected only in high-magnification events, which are a priority for follow-up teams such as RoboNet, PLANET, MiNDSTEp, or \( \mu \)FUN because they are highly sensitive to planets (Griest & Safizadeh 1998). In these events, the effects of finite angular source size are only seen close to the magnification peak and have a short duration (from hours to a few days). Using two telescopes is also consistent with the design goal that pyLIMA should be generally applicable to all ground-based and space-based data sets. Since until recently a lot of microlensing events have been covered by a combination of surveys and follow-up data, and since this is the most challenging combination of data, this is an excellent test case for the code. The Survey observatory is similar to the Ground simulations presented above. This data set represents a single site survey, which contains a daily gap in the light curve due to the day/night cycle. To ensure that finite-source effects were really present in our simulations, the Follow-up data consist of two days of observations around \( t_E \), in order to catch deviations from a PSPL model. The source magnitude and blending ratio of the Follow-up observations are chosen to be different from those of the Survey data in order to reproduce the fact that these follow-up telescopes have different spatial resolutions, sky conditions, and photometric reduction pipelines. We also chose a random cadence of observations (between 0 and 30) for the Follow-up data set. We decided to implement Poisson noise and red noise for both telescopes. To ensure that finite-source effects were observed, we forced \( t_e \) to be inside the observing window \([0 \text{ to } 183] \) days. We selected \( \rho \) from a uniform distribution from \( \frac{1}{2} u_0 \) to \( \min(10, u_0, 0.05) \). We limited the uniform distribution of \( u_0 \) from \( 10^{-4} \) to 0.025 and the photometric precision to 1%. For the purposes of this paper, we set \( \Gamma_s = 0.5 \) for the simulation and the fitting process. Again, 10,000 events were simulated.

### A.2. Definition of Fit Quality Metrics

To evaluate the accuracy of the fits, we compared the results from pyLIMA to the injected models in the light curves. It is true to say that the best-fit model is never identical to the injected model due to the noise in the data and the discontinuous sampling. However, it is also true that the input model represents the observations accurately. Therefore, a comparison of the injected versus the best-fit model parameters can be used to test the robustness of the fit (i.e., to estimate the accuracy of the parameters derived from the fit). We defined three different metrics to analyze the fits of our simulated events.

We first defined the \( \Sigma_s \) metric,

\[
\Sigma_s = \frac{x_{\text{model}} - x_{\text{pyLIMA}}}{\sigma_{x, \text{pyLIMA}}},
\]

where \( x \) is the model parameter and \( \sigma_{x, \text{pyLIMA}} \) is the error on the parameter returned by pyLIMA. For the LM method, \( \sigma_{x, \text{pyLIMA}} \) is the square root of the parameter’s variance obtained from the fit covariance matrix. In future versions of this software, this approximation will be replaced by more robust methods to estimate the variance, such as the bootstrap technique. It is also informative to judge predictions made by the fit for the future evolution of an event. For example, if an event has not yet reached its peak, we can judge how well pyLIMA estimates the microlensing parameters (i.e., we can define a good prediction if \( |\Sigma_s| < 1 \); \( \forall x \), for example). It also reveals whether the parameter errors are correctly estimated. However, this metric suffers one strong flaw: it vanishes when the parameter error diverges, which can happen with the covariance matrix estimation (see Appendix B).

To counter this problem, we defined a second metric, \( \Lambda_s \):

\[
\Lambda_s = \frac{x_{\text{model}} - x_{\text{pyLIMA}}}{\epsilon_s} = \begin{cases} 
  \epsilon_{t_e} = I_E, \text{model} \\
  \epsilon_{u_0} = I_E, \text{model} \\
  \epsilon_{8} = \text{model} \\
  \epsilon_{f_E} = \text{model} \\
  \epsilon_{g} = \text{model}
\end{cases}
\]

This metric is then a relative error and is insensitive to the estimation of parameter uncertainties. We opted to consider a fitted value acceptable if \( |\Lambda_s| < 0.1 \).

Finally, to judge whether or not a fit is a success, we computed

\[
\Delta x^2 = x^2_{\text{model}} - x^2_{\text{pyLIMA}}.
\]

This is a more robust statistic than \( x^2_{\text{dof}} \) because the latter could be insensitive to a bad fit where the “area of interest” (i.e., the microlensing event) is not significant regarding the total light-curve length. We finally defined a fit as successful if \( \Delta x^2 > 0 \). However, we also computed the \( \frac{\Delta x^2}{\text{dof}} \) for the various studies conducted.
Appendix B
Discussion of the Parameter Error Estimations

In this appendix, we study in more detail the behavior of the parameter error estimations. Error estimations for the LM and DE methods come from the inverse of the Fisher matrix, whereas the error estimations for the MCMC method come from the 1σ confidence interval around the distribution median. The Fisher matrix can be written as (Tsapras et al. 2016)

$$ F_{i,j} = \left( \frac{\log L}{dp_i} \right) \left( \frac{\log L}{dp_j} \right). $$

Figure 7. Distributions of parameter errors from pyLIMA LM (thick line histograms) and MCMC (light plain histograms) for the Space data set. Events are sorted by categories in lines and colors according to Section 5.1. The two methods estimate similar error bars for well-constrained events but present severe differences in the case of problematic light curves. This underlines a data issue rather than a fitting-routine problem.

Figure 8. Distributions of error volume from pyLIMA LM (blue) and MCMC (green) for the OGLE-II data set. Because this data set presents microlensing events with strong signals and good photometry, the two methods converge to equivalent fits and error bars. This illustrates the robustness of the LM and MCMC methods on microlensing events without pathological cases.

The Astronomical Journal, 154:203 (18pp), 2017 November Bachelet et al.
where \( \log L \) is the log-likelihood function. In the case where the noise model is assumed to be independent of the model, uncorrelated and normally distributed (LSST Science Collaboration et al. 2009), the log-likelihood can be written as

\[
\log L = \sum_n \log \frac{2\pi}{\sigma_n^2} - \sum_n \frac{(d_n - m_n)^2}{\sigma_n^2}.
\]

Assuming two parameters \((p_i, p_j)\) of the model \(m\), we can rewrite the Fisher matrix:

\[
F_{i,j} = \left\langle \sum_k \frac{(d_k - m_k)}{\sigma_k} \frac{dm_n}{dp_i} \sum_l \frac{(d_l - m_l)}{\sigma_l} \frac{dm_n}{dp_j} \right\rangle.
\]

Since \((d_k - m_k)(d_l - m_l) = \sigma_k^2 \delta_{k,l}\) (LSST Science Collaboration et al. 2009), with the Kronecker delta function noted, \(\delta_{k,l}\), the Fisher matrix reduces to (Mogavero & Beaulieu 2016)

\[
F_{i,j} = \sum_n \frac{1}{\sigma_n^2} \frac{dm_n}{dp_i} \frac{dm_n}{dp_j}.
\]

The details of each PSPL parameter are listed below.

1. \( \frac{df}{dt_0} = f \left( \frac{dA}{dt} - \frac{t_0}{2} \right) \)
2. \( \frac{df}{dm_n} = f \frac{dA}{dm_n} \frac{t_0}{2} \)
3. \( \frac{df}{dA} = -f \frac{2A}{u} \frac{dA}{du} \)
4. \( \frac{df}{du} = -8 \frac{u^2}{(u^2 + 4)^{3/2}} \)

Figure 9. The \( \Sigma \) distributions for the Space data set. When all parameters present similar results, as Figure 2, the \( \chi^2/\text{dof} \) is significantly different. For this data set, the photometric precision is not limited to 1%, explaining the nonskewed distribution.

Figure 10. The \( \Sigma \) distributions for the FSPL data set. In contrast to the Ground and Space data sets, there is no particular trend in the \( t_o \) and \( u_o \) distributions, because we forced events to peak in the observing windows and the event peak is well constrained by the Follow-up data set.
Since the LM and MCMC methods estimate parameter uncertainties in a different way, it is informative to compare their respective results on the same data sets. First, we refit all the Space light curves defined in Appendix A.1.2 using the MCMC method. Parameter uncertainties for both methods are visible in Figure 7. The general trend to notice is that the MCMC and LM methods agree for small uncertainties (i.e., well-defined events) and disagree for larger uncertainties. The LM method tends to overestimate errors for unconstrained events. This can be explained by the covariance matrix approach of uncertainty estimation, which assumes that the $\chi^2$ landscape is parabolic close to a minimum. This hypothesis breaks in the case of ill-defined events, and therefore a correction factor is needed in the covariance approach. On the other side, uncertainties estimated through the MCMC method look more realistic and stable.
regardless of the nature of the event. This, of course, has a cost: the MCMC fitting time of an event is roughly 2000 times slower than that of the LM method.

We decided to conduct a similar study on real data. We used the analysis made in Section 5 to compare errors coming from the LM and MCMC methods for the OGLE-II fits. However, we decided to change the diagnosis metric and compare the error volume for both data sets (Tsapras et al. 2016):

$$V = \prod \sigma_i. \quad (18)$$

This can be seen in Figure 8. The two distributions present a good agreement ($p$-value of the Kolmogorov–Smirnov test higher than 0.9). However, it seems that the error volume estimated from the MCMC looks slightly smaller (the median values for the $\log_{10} V$ are 3.1 and 1.7 for LM and MCMC, respectively). This is due to the fact that errors estimated from the MCMC method are not divergent when the events are not that well constrained, whereas the errors from the LM method can grow dramatically (see Appendix A).

**References**

Albrow, M. D. 2004, *ApJ*, 607, 821
Albrow, M. D., Beaulieu, J.-P., Caldwell, J. A. R., et al. 2000, *ApJ*, 535, 176
Alcock, C., Allsman, R. A., Alves, D. R., et al. 2000, *ApJ*, 541, 734
An, J. H., Albrow, M. D., Beaulieu, J.-P., et al. 2002, *ApJ*, 572, 521
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, *A&A*, 558, A33
Bachelet, E., Bramich, D. M., Han, C., et al. 2015, *ApJ*, 812, 136
Batalha, N. M., Rowe, J. F., Bryson, S. T., et al. 2013, *ApJS*, 204, 24
Bond, I. A., Abe, F., Dodd, R. J., et al. 2001, *MNRAS*, 327, 868
Borsato, L., Marzari, F., Nascimbeni, V., et al. 2014, *A&A*, 571, A38
Bozza, V. 2010, *MNRAS*, 408, 2188
Cassan, A., Beaulieu, J.-P., Fouqué, P., et al. 2006, *A&A*, 460, 277
Cassan, A., Kubas, D., Beaulieu, J.-P., et al. 2012, *Natur*, 481, 167
Catala, C., Appourchaux, T. & Pluto Mission Consortium 2011, *JPDCS*, 271, 012084
Clanton, C. & Gaudi, B. S. 2014, *ApJ*, 791, 91
Claret, A., & Bloemen, S. 2011, *A&A*, 529, A75
Coughlin, J. L., Mullally, F., Thompson, S. E., et al. 2015, arXiv:1512.06149
Díaz, R. F., Almenara, J. M., Santerne, A., et al. 2014, *MNRAS*, 441, 983
Dominik, M. 2009, *MNRAS*, 393, 816
Dominik, M., Horne, K., Allan, A., et al. 2008, *AN*, 329, 248
Dong, S., DePoy, D. L., Gaudi, B. S., et al. 2006, *ApJ*, 642, 842
Dong, S., Bond, I. A., Gould, A., et al. 2009, *ApJ*, 698, 1826
Dumusque, X. 2016, *A&A*, 593, A5
Dumusque, X., Borsa, F., Damasso, M., et al. 2017, *A&A*, 598, A133
Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, *PASP*, 125, 366
Gaudi, B. S. 2012, *ARA&A*, 50, 411
Gould, A. 2000, *ApJ*, 542, 785
Gould, A. 2004, *ApJ*, 606, 319
Griest, K., & Safizadeh, N. 1998, *ApJ*, 500, 37
Han, C., Bennett, D. P., Udalski, A., & Jung, Y. K. 2016, *ApJ*, 825, 8
Hunter, J. D. 2007, *CSE*, 9, 90
Janczak, J., Fukui, A., Dong, S., et al. 2010, *ApJ*, 711, 731
Jones, E., Oliphant, T., Peterson, P., et al. 2001, *SciPy: Open Source Scientific Tools for Python*, http://www.scipy.org/
LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, arXiv:0912.0201
Laureijs, R., et al. 2011, arXiv:1110.3193
Lee, C.-H., Riffeser, A., Seitz, S., & Bender, R. 2009, *ApJ*, 695, 200
Levenberg, K. 1944, *QApMa*, 2, 164
Ma, S., Mao, S., Ida, S., Zhu, W., & Lin, D. N. C. 2016, arXiv:1605.08556
Marquardt, D. W. 1963, *SIAM Journal on Applied Mathematics*, 11, 431
Milne, E. A. 1921, *MNRAS*, 81, 361
Mogavero, F., & Beaulieu, J. P. 2016, *A&A*, 585, A62
Moutou, C., Font, P., Barge, P., et al. 2005, *A&A*, 437, 355
Paczyński, B. 1986, *ApJ*, 304, 1
Ricker, G. R., Winn, J. N., Vanderspek, R., et al. 2014, *Space Telescopes and Instrumentation 2014: Optical, Infrared, and Millimeter Wave*, 914320
Ricker, G. R., Winn, J. N., Vanderspek, R., et al. 2015, *JATIS*, 1, 014003
Skowron, J., Udalski, A., Gould, A., et al. 2011, *ApJ*, 738, 87
Spergel, D., Gehrels, N., Baltay, C., et al. 2015, arXiv:1503.03757
Storn, R., & Price, K. 1997, *J. of Global Optimization*, 11, 341
Sumi, T., Kamiya, K., Bennett, D. P., et al. 2011, *Natur*, 473, 349
Sumi, T., Bennett, D. P., Bond, I. A., et al. 2013, *ApJ*, 778, 150
Szymanski, M. K. 2005, *AcA*, 55, 43
Thomas, C. L., & Griest, K. 2006, *ApJ*, 640, 299
Tsapras, Y., Horne, K., Kane, S., & Carson, R. 2003, *MNRAS*, 343, 1131
Tsapras, Y., Hundertmark, M., Wyrzykowski, L., et al. 2016, *MNRAS*, 457, 1320
Udalski, A. 2003, *AcA*, 53, 291
Udalski, A., Szymanski, M., Kaluzny, J., et al. 1994, *AcA*, 44, 227
Udalski, A., Zebrun, K., Szymanski, M., et al. 2000, *AcA*, 50, 1
Udalski, A., Paczynski, B., Zebrun, K., et al. 2002, *AcA*, 52, 1
van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, *CSE*, 13, 22
Vesterstrom, J., & Thomsen, R. 2004, in *IEEE Congress on Evolutionary Computation*, Vol. 2 (New York: IEEE), 19801987
Wambsganss, J., Witt, H. J., & Schneider, P. 1992, *A&A*, 258, 591
Witt, H. J., & Mao, S. 1994, *ApJ*, 430, 503
Wozniak, P. R., Udalski, A., Szymanski, M., et al. 2001, *AcA*, 51, 175
Yoo, J., DePoy, D. L., Gal-Yam, A., et al. 2004, *ApJ*, 603, 139