A&A 619, A117 (2018)
https://doi.org/10.1051/0004-6361/201732242

Astronomy & Astrophysics

A dedicated source-position transformation package: pySPT*

Olivier Wertz and Bastian Orthen

Argelander-Institut für Astronomie, Universität Bonn, Auf dem Hügel 71, 53121 Bonn, Germany
e-mail: owertz@alumni.ulg.ac.be

Received 6 November 2017 / Accepted 23 January 2018

ABSTRACT

Modern time-delay cosmography aims to infer the cosmological parameters with a competitive precision from observing a multiply imaged quasar. The success of this technique relies upon a robust modeling of the lens mass distribution. Unfortunately strong degeneracies between density profiles that lead to almost the same lensing observables may bias precise estimates of the Hubble constant. The source position transformation (SPT), which covers the well-known mass-sheet transformation (MST) as a special case, defines a new framework to investigate these degeneracies. In this paper, we present `pySPT`, a `python` package dedicated to the SPT. We describe how it can be used to evaluate the impact of the SPT on lensing observables. We review most of its capabilities and elaborate on key features that we used in a companion paper regarding SPT and time delays. The `pySPT` program also comes with a subpackage dedicated to simple lens modeling. This can be used to generate lensing related quantities for a wide variety of lens models independent of any SPT analysis. As a first practical application, we present a correction to the first estimate of the impact on time delays of the SPT, which has been experimentally found in a previous work between a softened power law and composite (baryons + dark matter) lenses. We find that the large deviations previously predicted have been overestimated because of a minor bug in the public lens modeling code `lensmodel` (v1.99), which is now fixed. We conclude that the predictions for the Hubble constant deviate by ∼7%, first and foremost as a consequence of an MST. The latest version of `pySPT` is available on Github, a software development platform, along with some tutorials to describe in detail how making the best use of `pySPT`.

Key words. cosmological parameters – gravitational lensing: strong

1. Introduction

For about a decade, the modern time-delay cosmography, namely the cosmological parameter inferences from time delay measurements in strong gravitational lensing, have been achieved with an increasingly competitive precision (for a recent review, Treu & Marshall 2016; see and references therein). A crucial step of this technique relies upon a robust characterization of the gravitational potential, which produces the multiply imaged configuration of a background bright quasar (see, e.g., Keeton 2003; Fassnacht et al. 2006; Suyu et al. 2010, 2013; Wong et al. 2011, 2017; Birrer et al. 2016). This gravitational potential is essentially produced by a main deflector but also by any mass distributions lying along the line of sight to the source (Seljak 1994; Bar-Kana 1996). Unfortunately, modeling the main lens mass distribution faces a major hurdle, namely the existence of degeneracies between plausible lens density profiles. In fact, significant freedom exists in choosing lens models that produce the same image configurations but predict different products of time delays and Hubble constant, $H_0\,\Delta t$ (Schneider & Sluse 2013). Thus, these degeneracies translate into systematic errors that propagate to H_0.

Now well-known to the lensing community, the first lensing invariance to have been pointed out is the mass-sheet degeneracy (MSD; Falco et al. 1985). A dimensionless surface mass density $\kappa(\boldsymbol{\theta})$ and all the modified $\kappa_\lambda(\boldsymbol{\theta})$ under the mass-sheet transformation (MST) defined as

$$\kappa_\lambda(\boldsymbol{\theta}) = \lambda\,\kappa(\boldsymbol{\theta}) + (1-\lambda), \tag{1}$$

along with the corresponding unobservable source rescaling $\boldsymbol{\beta} \rightarrow \lambda\boldsymbol{\beta}$, lead to identical lensing observables with the exception of time delays, which transform as in $\Delta t \rightarrow \lambda\,\Delta t$. This pure mathematical degeneracy has nothing to do with the gravitational perturbations caused by external masses along the line of sight. Whereas various solutions have been already proposed to reduce the impact of the MST on time-delay cosmography (see, e.g., Sect. 3 in Treu & Marshall 2016; and references therein), none of them succeed in unambiguously breaking the MSD. The source position transformation (SPT), a more general class of degeneracies which includes the MST as a special case, has been introduced in Schneider & Sluse (2014) and carried forward in Unruh et al. (2017). The SPT defines a new mathematical framework that includes degeneracies that have been neither described nor considered in time-delay cosmography before. The first estimation of its impact on time delays is given in Schneider & Sluse (2014) where the authors show experimentally that predictions for H_0 can deviate by ∼20% when comparing a softened power law (SPL) with a composite (baryons + dark matter) lens models. However, this value has been overestimated owing to a bug in the software used by the author. We address this issue in Sect. 5 and show that the predictions for H_0 can deviate by ∼7% instead.

Recently, a detailed analysis of how the SPT may affect the time-delay cosmography has been presented in Wertz et al. (2018). To address this question, we started by developing a flexible numerical framework that encompasses well-tested and efficient implementations of most of the analytical results published in Schneider & Sluse (2014) and Unruh et al. (2017). Numerous additional features were then added, giving rise to `pySPT`, an easy-to-use and well-documented `python` package dedicated to the SPT. We used `pySPT` to draw the conclusions presented

* https://github.com/owertz/pySPT

in Wertz et al. (2018). Thus, our package is released as open source, making our results easy to reproduce. Beyond that, we also hope that it will be useful to the time-delay cosmography community to quantify the systematic errors that are introduced by the SPT when inferring H_0.

This paper is organized as follows. We outline the basic principles of SPT in Sect. 2. Section 3 gives an overall description of the package design and features, and details are provided in Sect. 4. In Sect. 5, we present the corrected version of some results presented in Schneider & Sluse (2014). We summarize our findings in Sect. 6.

2. Principle of source position transformation

This section focuses on the basic idea underlying the SPT and its mathematical framework. A detailed discussion is provided in Schneider & Sluse (2014) and Unruh et al. (2017). In this paper we adopt the standard gravitational lensing notation (see Schneider 2006).

The relative lensed image positions $\boldsymbol{\theta}_i(\boldsymbol{\theta}_1)$ of a background point-like source located at the unobservable position $\boldsymbol{\beta}$ constitute the lensing observables that we measure with the highest accuracy and precision. As a typical example, just a few mas can be achieved with deep HST observations. When n images are observed, the mapping $\boldsymbol{\theta}_i(\boldsymbol{\theta}_1)$ only provides the constraints

$$\boldsymbol{\theta}_i - \boldsymbol{\alpha}(\boldsymbol{\theta}_i) = \boldsymbol{\theta}_j - \boldsymbol{\alpha}(\boldsymbol{\theta}_j), \qquad \forall\, 1 \le i < j \le n, \tag{2}$$

where $\boldsymbol{\alpha}(\boldsymbol{\theta})$ corresponds to the deflection law caused by a foreground surface mass density $\kappa(\boldsymbol{\theta})$, the so-called lens. The SPT addresses the question of whether we define an alternative deflection law, denoted as $\hat{\boldsymbol{\alpha}}(\boldsymbol{\theta})$, which preserves the mapping $\boldsymbol{\theta}_i(\boldsymbol{\theta}_1)$ for a unique source? If such a deflection law exists, the alternative source position $\hat{\boldsymbol{\beta}}$ differs in general from $\boldsymbol{\beta}$. Furthermore, it defines the new lens mapping $\hat{\boldsymbol{\beta}} = \boldsymbol{\theta} - \hat{\boldsymbol{\alpha}}(\boldsymbol{\theta})$, which leads to

$$\boldsymbol{\theta} = \boldsymbol{\beta} + \boldsymbol{\alpha}(\boldsymbol{\theta}) = \hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\alpha}}(\boldsymbol{\theta}). \tag{3}$$

An SPT consists in a global transformation of the source plane formally defined by a one-to-one mapping $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$, unrelated to any physical contribution such as the external convergence (Schneider & Sluse 2013). To preserve the mapping $\boldsymbol{\theta}_i(\boldsymbol{\theta}_1)$, the alternative deflection law is thus written

$$\hat{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \boldsymbol{\alpha}(\boldsymbol{\theta}) + \boldsymbol{\beta} - \hat{\boldsymbol{\beta}}(\boldsymbol{\beta}) = \boldsymbol{\alpha}(\boldsymbol{\theta}) + \boldsymbol{\beta} - \hat{\boldsymbol{\beta}}(\boldsymbol{\theta} - \boldsymbol{\alpha}(\boldsymbol{\theta})), \tag{4}$$

where in the first step we used Eq. (3) and in the last step we inserted the original lens equation. As defined, the deflection laws $\boldsymbol{\alpha}(\boldsymbol{\theta})$ and $\hat{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ yield exactly the same image positions of the source $\boldsymbol{\beta}$ and $\hat{\boldsymbol{\beta}}$, respectively.

Because $\hat{\boldsymbol{\alpha}}$ is in general not a curl-free field, it cannot be expressed as the gradient of a deflection potential caused by a mass distribution $\hat{\kappa}$. Provided its curl component is sufficiently small, Unruh et al. (2017) have established that one can find a curl-free deflection law $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\nabla}\tilde{\psi}$ that is similar to $\hat{\boldsymbol{\alpha}}$ in a finite region. In other words, $\tilde{\boldsymbol{\alpha}}$ yields the same sets of multiple images up to the astrometric accuracy $\varepsilon_{\rm acc}$ of current observations. Two image configurations are considered indistinguishable when they satisfy

$$|\Delta\boldsymbol{\theta}| := \left|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\right| < \varepsilon_{\rm acc}, \tag{5}$$

for all images $\boldsymbol{\theta}$ of the source $\boldsymbol{\beta}$, and corresponding images $\tilde{\boldsymbol{\theta}}$ of the source $\hat{\boldsymbol{\beta}}$ with $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\beta}} + \tilde{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\theta}})$. In Unruh et al. (2017), the adopted similarity criterion is written

$$|\Delta\boldsymbol{\alpha}(\boldsymbol{\theta})| := |\tilde{\boldsymbol{\alpha}}(\boldsymbol{\theta}) - \hat{\boldsymbol{\alpha}}(\boldsymbol{\theta})| < \varepsilon_{\rm acc}, \tag{6}$$

in a finite region of the lens plane denoted as $\mathcal{U}$ where multiple images occur. Even though this criterion cannot actually guarantee $|\Delta\boldsymbol{\theta}| < \varepsilon_{\rm acc}$ over $\mathcal{U}$, there exists in general a subregion $\mathcal{U}' \subset \mathcal{U}$ that includes image configurations for which Eq. (5) is satisfied (see Sect. 4.1 in Wertz et al. 2018). Thus, an SPT $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$ leads to an alternative lens profile $\hat{\kappa}$, which gives rise to the SPT-transformed deflection law $\hat{\boldsymbol{\alpha}}$, then its curl-free counterpart $\tilde{\boldsymbol{\alpha}}$ is defined based upon the criterion Eq. (6), and may lead to indistinguishable image configurations produced by $\boldsymbol{\alpha}$. To conclude this section, we note that the deflection law $\tilde{\boldsymbol{\alpha}}$ is produced by a surface mass density $\tilde{\kappa} := \boldsymbol{\nabla}\cdot\tilde{\boldsymbol{\alpha}}/2$, which equals $\hat{\kappa}$ by construction.

3. Package overview

The `pySPT` program was developed in `python` (Watters et al. 1996) and only relies on packages included in the `python` standard library[1] and the proven open-source libraries `numpy` (Van Der Walt et al. 2011), `scipy` (Jones et al. 2001), and `matplotlib` (Hunter 2007). The code design and development follow effective practices for scientific computing such as using test cases and a profiler to identify bugs and bottlenecks, keeping an effective collaboration thanks to a version control system (`git`), and providing an extensive documentation (Wilson et al. 2014). This open source code is available on `GitHub`[2] and comes along with a clear description on how to install it. To make the best use out of `pySPT`, a quick start guide and several tutorials are also provided in the form of `Jupyter` notebooks. Benefiting from the `python` object-oriented paradigm, the structure of `pySPT` is highly modular, which avoids code clones and makes it less sensitive to bug propagation.

The code is composed of several modules built from various classes and is organized in a dozen subpackages. Its core is separated into three main subpackages, referred to as `lensing`, `sourcemapping`, and `spt`. The reason for that is simple. The mapping $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$ presented in Sect. 2 defines a one-to-one connection between the source plane and its SPT-transformed counterpart. Through the lens equation $\boldsymbol{\beta} = \boldsymbol{\theta} - \boldsymbol{\alpha}(\boldsymbol{\theta})$, the deflection law (arising from a lens model) characterizes the link between the source plane and the image plane. Thus, only together with $\boldsymbol{\alpha}(\boldsymbol{\theta})$ does $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$ give rise to the alternative deflection law $\hat{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ defined in Eq. (4). As a result, the most significant `pySPT` subpackages are `lensing` to basically deal with $\boldsymbol{\alpha}(\boldsymbol{\theta})$, `sourcemapping` to define $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$, and `spt` to describe $\hat{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ (and all the SPT-transformed lensing quantities). In the remainder of this section, we describe briefly their main functionalities[3].

The subpackage `lensing` shares a lot of functionalities with other lensing-dedicated softwares such as `gravlens` (Keeton 2001b). From its class `Model`, we generated a lens model object that allowed us to perform a wide range of basic lensing calculations. Strictly speaking, this part of the code is not related to the SPT and it can be used independent of any SPT analysis.

[1] https://docs.python.org/2/library/index.html

[2] https://github.com/owertz/pySPT. For `git` users, `pySPT` can be cloned directly from the source code repository via the following bash command

```
git clone --recursive https://github.com/owertz/pySPT
```

[3] We adopt the naming conventions advocated in the Python Enhancement Proposals 8 (PEP-8). In particular, subpackages have all-lowercase names while class names use the so-called CapWords convention. The PEP-8 is accessible at https://www.python.org/dev/peps/pep-0008/

Nevertheless, we decided to implement this part of the code for a simple reason: as a built-in feature of pySPT, the class Model provides to its instances the adequate structure that is required to match the spt requirements. This makes lensing more convenient to use than a third party code. The subpackage sourcemapping is used to define an SPT $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$. Several forms of $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$ are implemented, such as particular radial stretchings presented in Schneider & Sluse (2014). Moreover, the code is designed to accept any user-defined SPT. With sourcemapping also comes functionalities to characterize the mapping $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$, such as testing whether it is one-to-one over a given region. More details are given in Sect. 4.2. The subpackage spt is the heart of pySPT. It is designed to work alongside with sourcemapping and lensing to provide all the basic SPT-transformed quantities, such as $\hat{\kappa}$, $\hat{\boldsymbol{\alpha}}$, $\hat{\psi}$, $\hat{\mathcal{A}}$, $\tilde{\theta}$, $\tilde{\boldsymbol{\alpha}}$, $\tilde{\psi}$. One can also derive the SPT-transformed time delays $\Delta\hat{t}$ and $\Delta\tilde{t}$, and most of the quantities presented in Schneider & Sluse (2014), Unruh et al. (2017), and Wertz et al. (2018). This makes the results presented in these papers straightforward to reproduce. A detailed description of the tools provided by spt is given in Sect. 4.3.

The pySPT code also includes several subpackages dedicated to specific tasks. Based on the package multiprocessing of the standard library, multiproc provides an efficient tool to parallelize functions and methods in a straightforward way. As such, most of the pySPT features support parallel computing to leverage multiple processors fully. The grid feature helps us to create different types of mesh grids. These are of practical interest for generating maps of lensing quantities in a particular region. To calculate $\tilde{\boldsymbol{\alpha}}$ efficiently, we followed Unruh et al. (2017), which suggests using a Riemann mapping to handle a pole numerically. Thus, pySPT contains the subpackage integrals that includes functionalities to deal with and to illustrate conformal mappings.

4. Analyzing the impact of the SPT with pySPT

4.1. Subpackage lensing to deal with lens models

To analyze how the SPT may alter lensing observables in a quantitative way, we needed to select a model that characterizes a mass distribution. A range of lensing observables were then generated and compared against those produced by an SPT-transformed version of the original model. The subpackage lensing relies on the class Model of which one of its instances defines a lens model and provides efficient tools to compute a wide range of lensing quantities. Most of the mass profiles described in Keeton (2001a) are available in the subpackage catalog, together with a complete documentation. We note that, to our knowledge, no analytical expression of the deflection potential for the generalized pseudo-NFW can be found in the literature. Hence, we derived this expression and provide the result in Appendix A. The class Model is also designed to accept a user-defined lens model, as long as some conventions are respected. At least, the deflection potential $\psi(\boldsymbol{\theta})$ must be provided as either a python function or a C shared library. Those that are not defined among $\boldsymbol{\alpha}$, κ, and $\partial\boldsymbol{\alpha}/\partial\boldsymbol{\theta}$ are then computed from numerical approximation at the expense of a more intensive usage of computer resources. Thanks to operator overloading[4], arbitrarily complicated composite models can be obtained by simply adding several Model instances. For example, this feature is used to generate quadrupole models, namely the combination of an axisymmetric matter distribution plus some external shear. To go a step further, the axisymmetric part may itself be composed of different components, whereas additional contributions can be included at arbitrary positions.

The computational methods implemented with the class Model follow the prescription of Keeton (2001b). In particular, the so-called tiling algorithm (with adaptive grid) is used to solve the lens equation and locate the critical curves. Other fundamental lensing quantities are available, such as the caustics, basic image properties (e.g., image type, magnification factor, parity, odd-number, and magnification theorem checks), Fermat potential $\tau(\boldsymbol{\theta})$, and time delays between image pairs $\Delta t(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \equiv \Delta t_{ij}$. In Fig. 1, we illustrate the capabilities of the subpackage lensing by showing several lensing quantities produced by a complex mass distribution. The workflow for generating data used in this figure is as follows[5]:

The subpackage lensing also includes C shared libraries that implement ψ, $\boldsymbol{\alpha}$, κ, $\partial\boldsymbol{\alpha}/\partial\boldsymbol{\theta}$ for all the lens models. As shown in Sect. 4.3, the curl-free deflection angle $\tilde{\boldsymbol{\alpha}}$ is obtained by means of line and surface integrals of functions that involve $\hat{\boldsymbol{\alpha}}$ (see Eq. (4)) and $\hat{\kappa} = \boldsymbol{\nabla} \cdot \hat{\boldsymbol{\alpha}}/2$ (Unruh et al. 2017). Thus, the original deflection angle $\boldsymbol{\alpha}$ is evaluated as many times as the number of iterations required by the solver to converge. Even though this procedure is efficient when only one $\tilde{\boldsymbol{\alpha}}$ is evaluated, high performance becomes critical when $\tilde{\boldsymbol{\alpha}}$ needs to be evaluated on a dense grid. Thanks to the foreign function interface module ctypes[6], the use of the C shared libraries speeds up significantly the execution of pure python code. The pySPT code comes along with a tutorial in the form of a Jupyter notebook that describes in details how to deal with C shared libraries.

4.2. Defining $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$ with the subpackage sourcemapping

After we defined the lens model, we chose an SPT by defining a source mapping $\hat{\boldsymbol{\beta}} \equiv \hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$. To each position $\boldsymbol{\beta}$ of the source plane, this mapping associates a new and unique position $\hat{\boldsymbol{\beta}}$ in the source plane. The so-called radial stretching is simply defined as

$$\hat{\boldsymbol{\beta}}(\boldsymbol{\beta}) = [1 + f(|\boldsymbol{\beta}|)]\boldsymbol{\beta}, \tag{7}$$

where f is called the deformation function. With $\hat{\boldsymbol{\beta}}$ defined this way, $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta}_j)$ always lies on the line passing through $\mathbf{0}$ and $\boldsymbol{\beta}_j$. The most simple case of radial stretching corresponds to a constant deformation function, $f(|\boldsymbol{\beta}|) = \lambda - 1$ with $\lambda \in \mathbb{R}$, which leads to the well-known MST, $\hat{\boldsymbol{\beta}} = \lambda\boldsymbol{\beta}$. In Wertz et al. (2018), we focused most of the work on the radial stretching Eq. (7) where the deformation function $f(|\boldsymbol{\beta}|)$ is defined as the lowest order expansion of more general functions

$$f(|\boldsymbol{\beta}|) = f_0 + \frac{f_2}{2\theta_{\mathrm{E}}^2}|\boldsymbol{\beta}|^2, \tag{8}$$

where $f_0 \coloneqq f(0)$, $f_2 \coloneqq \theta_{\mathrm{E}}^2 f''(0)$, θ_{E} is the Einstein angular radius, and f is an even function of $|\boldsymbol{\beta}|$ to preserve the symmetry (Schneider & Sluse 2014). When $f_2 = 0$, this case simplifies to a pure MST with $\lambda = 1 + f_0$. In Table 1, we provide a list of the radial stretchings implemented in sourcemapping.

[4] Operator overloading is a special case of polymorphism that is well defined in the object-oriented programming (OOP) paradigm. Thus, this feature is not only a python syntactic sugar but exists in any other OOP languages.

[5] A Jupyter notebook dedicated to this figure is available in the pySPT repository on GitHub. In addition to details about the workflow shown above, it includes the code we used to plot Fig. 1.

[6] ctypes is included in the python standard library: https://docs.python.org/2.7/library/ctypes.html

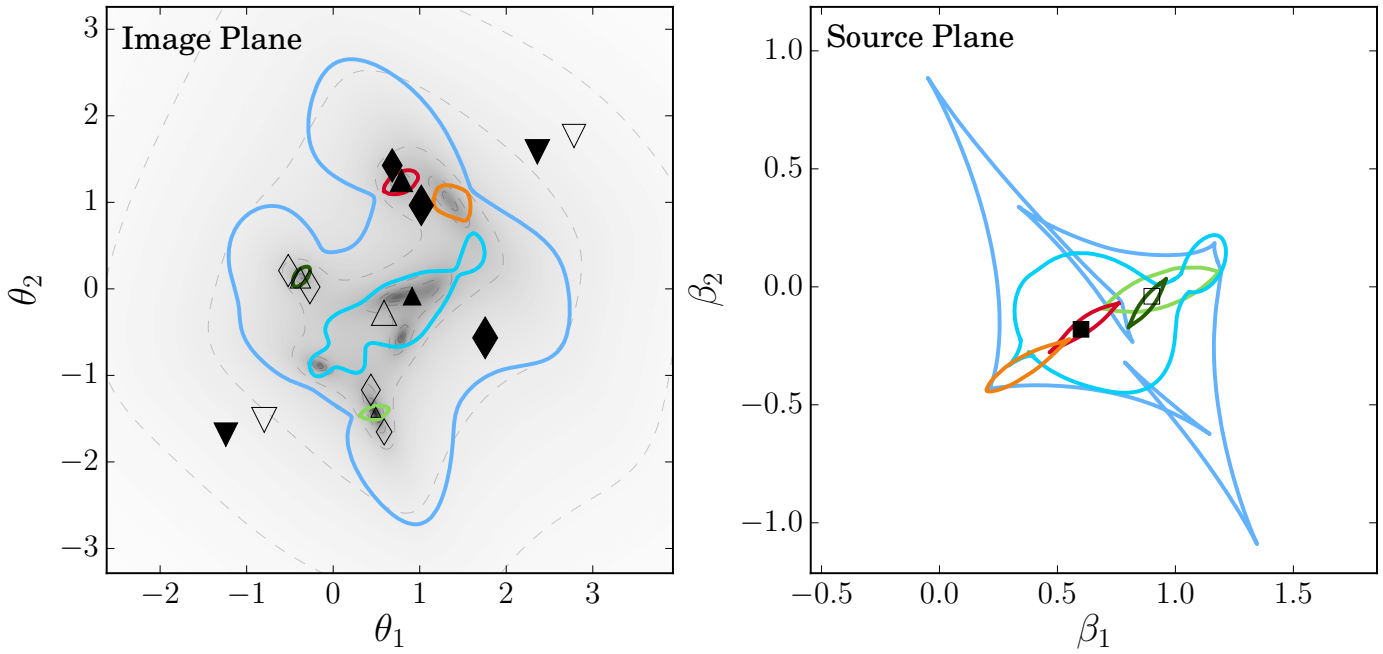

Fig. 1. Example illustrating some capabilities of the subpackage `lensing`. *Left panel*: eight lens profiles with different parameters are combined to generate a complex mass distribution. The total surface mass density is shown in tones of gray and dashed curves highlight few particular iso-density contours. The colored thick curves show the critical curves and the filled and empty markers locate the lensed image positions of two different sources shown in the *right panel*. The inverted triangles correspond to images of type I (maximum of τ), the diamonds to type II (saddle point of τ), and triangles to type III (maximum of τ). The size of the markers is log-proportional to the magnification of the images. *Right panel*: the colored lines show the caustics and the two squares locate the sources. The filled (resp. empty) square has seven (resp. nine) images, all shown in the *left panel*. The axis scale is arcseconds but unit of $\theta_{\rm E}$ can also be used.

The rationale behind the choice of these particular deformation functions are motivated in Schneider & Sluse (2014). The first column refers to the `id` used to identify which source mapping one wants to select. A specific example is given below. Aside from defining an SPT, `sourcemapping` also includes a few functionalities for characterizing the mapping $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$. In particular, one can test whether the source mapping is one to one over a given region, compute the Jacobi matrix $\mathcal{B}(\boldsymbol{\beta}) = \partial\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})/\partial\boldsymbol{\beta}$, and provide its decomposition $\mathcal{B}(\boldsymbol{\beta}) = B_1\mathcal{I} + B_2\Gamma(\eta)$, which is useful to evaluate the amplitude of the curl component of $\hat{\boldsymbol{\alpha}}$ (see Sect. 4.1 in Schneider & Sluse 2014). As an example, the code below illustrates how to define and characterize an MST with $\lambda = 1(\equiv 1 + f_0)$: Similarly, we can first define the deformation function f that characterizes an MST with λ as unique argument, and pass it to `SourceMapping` as an argument. For efficiency, the first derivative f' of the deformation function with respect to $|\boldsymbol{\beta}|$ can also be passed[7]. This process illustrates how we can work with a user-defined SPT.

4.3. Deriving SPT-transformed quantities with `spt`

In the two previous sections, we have shown how to generate lensing observables produced by a given lens model and how to define an SPT. We present the subpackage `spt`, which provides the tools required to determine the SPT-transformed quantities $\hat{\kappa}$, $\hat{\boldsymbol{\alpha}}$, $\hat{\psi}$, $\hat{\mathcal{A}}$, $\Delta\hat{t}$, $\tilde{\boldsymbol{\alpha}}$, $\tilde{\psi}$, $\tilde{\boldsymbol{\theta}}$, and $\Delta\tilde{t}$.

For any SPT and lens model, the alternative deflection law $\hat{\boldsymbol{\alpha}}$ is implemented as defined in Eq. (4), and the Jacobi matrix of the alternative lens equation given in Eq. (3) as $\hat{\mathcal{A}}(\boldsymbol{\theta}) = (\partial\hat{\boldsymbol{\beta}}/\partial\boldsymbol{\beta})\,(\partial\boldsymbol{\alpha}/\partial\boldsymbol{\theta}) \equiv \mathcal{B}(\boldsymbol{\beta}(\boldsymbol{\theta}))\,\mathcal{A}(\boldsymbol{\theta})$. The determination of $\hat{\mathcal{A}}$ perfectly illustrates how `spt` works along with `lensing` and `sourcemapping`: $\mathcal{B}(\boldsymbol{\beta})$ is obtained with `sourcemapping`, $\boldsymbol{\beta}(\boldsymbol{\theta})$ and $\mathcal{A}(\boldsymbol{\theta})$ are obtained with `lensing`, and `spt` combines the results to derive $\hat{\mathcal{A}}(\boldsymbol{\theta})$. In the axisymmetric case, $\hat{\boldsymbol{\alpha}}$ is a curl-free field and there exists a deflection potential $\hat{\psi}$ such that $\mathrm{d}\hat{\psi}(\theta)/\mathrm{d}\theta = \hat{\alpha}(\theta) = \theta - \hat{\beta}(\theta - \alpha(\theta))$, where $\theta = |\boldsymbol{\theta}|$. Thus, $\hat{\psi}$ is obtained as follows:

$$\hat{\psi}(\theta) = \int_0^{\theta} \left[\vartheta - \hat{\beta}(\vartheta - \alpha(\vartheta))\right] \mathrm{d}\vartheta, \tag{9}$$

up to a constant independent of θ. Because the integrand in Eq. (9) depends on the SPT and the lens model choices, no general analytical solution can be derived. The integral is therefore computed numerically using the `pythonmodule`

Table 1. List of radial stretchings implemented in `lensmapping`.

Id	$f(\lvert\boldsymbol{\beta}\rvert)$	Arguments
1	$f_0 + f_2\lvert\boldsymbol{\beta}\rvert^2/2$	f_0, f_2
2	$f_0 + \beta_0^2 f_2 \lvert\boldsymbol{\beta}\rvert^2 \left[2\left(\beta_0^2 + \lvert\boldsymbol{\beta}\rvert^2\right)\right]$	f_0, f_2, β_0
3	$2f_0/\cosh(\lvert\boldsymbol{\beta}\rvert/\beta_0) - f_0$, with $\beta_0 = \theta_{\rm E}\sqrt{3(1-f_0)/(1+f_0)}$	$f_0, \theta_{\rm E}$
4	$a\left[1 - \cos(c\,\lvert\boldsymbol{\beta}\rvert)\right]$	a, c

Notes. The first column `id` is used to select the source mapping when we instantiate the class `SourceMapping` (see in text for more details).

[7] For axisymmetric profile $\kappa(\theta) = \kappa(|\boldsymbol{\theta}|)$, the analytical expression for $\hat{\kappa}$ involves f' (see Eq. (16) in Schneider & Sluse 2014).

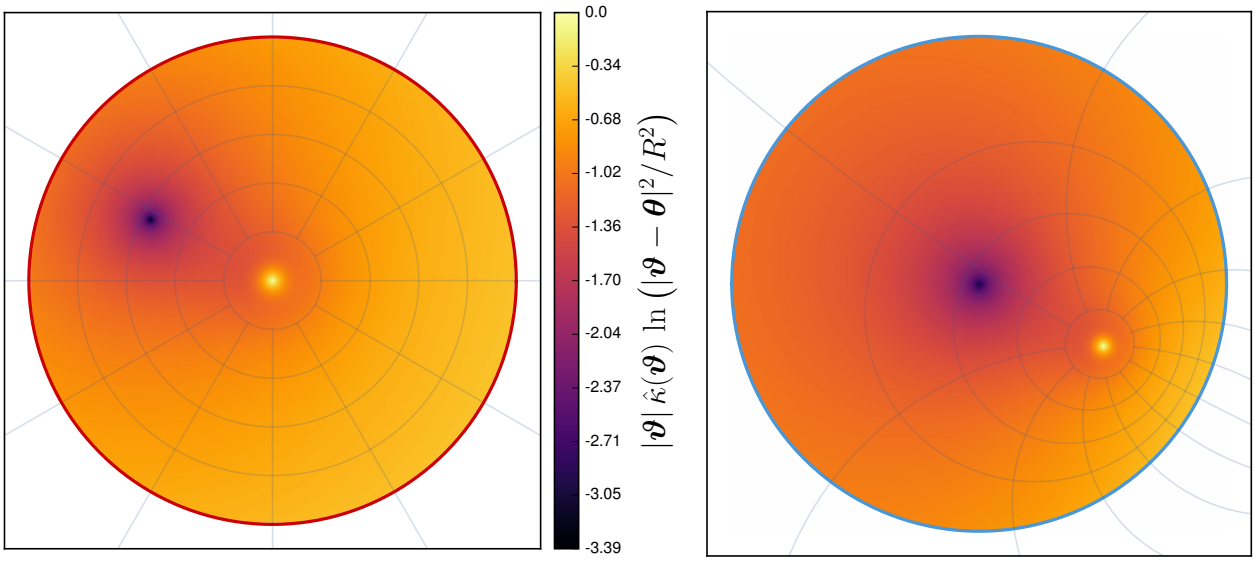

Fig. 2. Representative example that illustrates how the integration domain $\mathcal{U}$ is mapped onto the unit disk in the complex plane under a Riemann mapping. *Left panel*: the color coding depicts the integrand $|\boldsymbol{\vartheta}|\,\hat{\kappa}(\boldsymbol{\vartheta})\,\ln(|\boldsymbol{\vartheta}-\boldsymbol{\theta}|^2/R^2)$ for all $\boldsymbol{\vartheta}\in\mathcal{U}$. It shows a pole at $\boldsymbol{\vartheta}=\boldsymbol{\theta}$ with $\boldsymbol{\theta}=(-0.5,0.25)\,\theta_\mathrm{E}$ and a secondary peak at the origin caused by $\kappa(|\boldsymbol{\vartheta}|=0)$. The lens model is an NIS ($\theta_\mathrm{c}=0.1\theta_\mathrm{E}$) plus external shear ($\gamma_\mathrm{p}=0.1$) transformed by the radial stretching $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})=f_0+f_2|\boldsymbol{\beta}|^2/(2\theta_\mathrm{E}^2)$ with $(f_0,f_2)=(0,0.55)$. The red circle delimits $\mathcal{U}$ with radius R. *Right panel*: integrand after applying the Riemann mapping described in Appendix A in Unruh et al. (2017). The pole $\boldsymbol{\vartheta}=\boldsymbol{\theta}$ now lies at the origin of the unit (blue) circle of the complex plane. The polar grid (gray lines) allows us to visualize how the Riemann mapping acts on $\mathcal{U}$. For this figure, we used the subpackage `integrals` that addresses both the pole and second peak.

`integrate.quad`[8] from `scipy` (Jones et al. 2001). Under a radial stretching, the axisymmetric mass profile $\kappa(\theta)$ transforms into $\hat{\kappa}(\theta)=\kappa(\theta)-[1-\kappa(\theta)]\,f(\beta(\theta))-\theta\det\mathcal{A}(\theta)\,f'(\beta(\theta))/2$, where $\beta=|\boldsymbol{\beta}|$ (Schneider & Sluse 2014). Otherwise, the more general form $\hat{\kappa}(\boldsymbol{\theta})=1-\mathrm{Tr}(\hat{\mathcal{A}})/2$ is valid regardless of the lens model symmetry.

When the axisymmetry assumption for the original lens model is dropped, `spt` also provides the physically meaningful $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\psi}$. The analytical expressions implemented in `spt` are slightly simplified versions of those first presented in Unruh et al. (2017). The deflection potential $\tilde{\psi}$ evaluated at the position $\boldsymbol{\theta}$ in the lens plane is explicitly written

$$\tilde{\psi}(\boldsymbol{\theta})=\left\langle\tilde{\psi}\right\rangle+2\int_{\mathcal{U}}H_1(\boldsymbol{\theta};\boldsymbol{\vartheta})\,\hat{\kappa}(\boldsymbol{\vartheta})\,\mathrm{d}^2\vartheta-\int_{\partial\mathcal{U}}H_2(\boldsymbol{\theta};\boldsymbol{\vartheta})\,\hat{\boldsymbol{\alpha}}\cdot\boldsymbol{n}\,\mathrm{d}s\,, \quad (10)$$

where the region $\mathcal{U}$ is a disk of radius R, $\left\langle\tilde{\psi}\right\rangle$ is the average of $\tilde{\psi}$ on $\mathcal{U}$, $\mathrm{d}s$ the line element of the boundary curve $\partial\mathcal{U}$,

$$H_1(\boldsymbol{\theta};\boldsymbol{\vartheta})=\frac{1}{4\pi}\left[\ln\left(\frac{|\boldsymbol{\vartheta}-\boldsymbol{\theta}|^2}{R^2}\right)+\ln\left(1-\frac{2\boldsymbol{\vartheta}\cdot\boldsymbol{\theta}}{R^2}+\frac{|\boldsymbol{\vartheta}|^2|\boldsymbol{\theta}|^2}{R^4}\right)-\frac{|\boldsymbol{\vartheta}|^2}{R^2}\right], \quad (11)$$

and

$$H_2(\boldsymbol{\theta};\boldsymbol{\vartheta})=\frac{1}{4\pi}\left[2\ln\left(\frac{|\boldsymbol{\vartheta}-\boldsymbol{\theta}|^2}{R^2}\right)-1\right]. \quad (12)$$

In Appendix B, we explicitly show that both versions are fully equivalent. The corresponding simplified version of the deflection angle $\tilde{\boldsymbol{\alpha}}$ can be derived by obtaining the gradient of H_1 and H_2 with respect to $\boldsymbol{\theta}$, which is written

$$\tilde{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \frac{1}{\pi}\int_{\mathcal{U}}\left(\frac{\boldsymbol{\theta}-\boldsymbol{\vartheta}}{|\boldsymbol{\theta}-\boldsymbol{\vartheta}|^2}+\frac{|\boldsymbol{\vartheta}|^2\boldsymbol{\theta}-R^2\boldsymbol{\vartheta}}{R^4-2R^2\boldsymbol{\vartheta}\cdot\boldsymbol{\theta}+|\boldsymbol{\vartheta}|^2|\boldsymbol{\theta}|^2}\right)\hat{\kappa}(\boldsymbol{\vartheta})\,\mathrm{d}^2\vartheta - \frac{1}{\pi}\int_{\partial\mathcal{U}}\frac{\boldsymbol{\theta}-\boldsymbol{\vartheta}}{|\boldsymbol{\theta}-\boldsymbol{\vartheta}|^2}\,\hat{\boldsymbol{\alpha}}\cdot\boldsymbol{n}\,\mathrm{d}s. \quad (13)$$

To deal with the pole $\boldsymbol{\vartheta}=\boldsymbol{\theta}$ in the first term of Eqs. (11) and (13), we used a Riemann mapping as described in Appendix A in Unruh et al. (2017) and implemented in the subpackage `integrals`. This mathematical trick makes the previous integrals easier to solve by mapping $\mathcal{U}$ onto the unit disk in the complex plane, such as the pole is moved at the origin[9] $z=0$. Additional care is however needed in the vicinity of $\boldsymbol{\theta}=\mathbf{0}$, where the gradient of $\kappa(\boldsymbol{\theta})$ may vary significantly and produce a second sharp peak of the integrand. This peak may even be a new pole when κ is singular at the origin. Thus, a physically meaningful lens model should always be favored. In the left panel in Fig. 2, we illustrate the integrand[10] $|\boldsymbol{\vartheta}|\,\hat{\kappa}(\boldsymbol{\vartheta})\,\ln\left(|\boldsymbol{\vartheta}-\boldsymbol{\theta}|^2/R^2\right)$ of the first term of the integral over $\mathcal{U}$ defined in Eq. (10). For this illustrative example, we choose an non-singular isothermal sphere (NIS, $\theta_\mathrm{c}=0.1\theta_\mathrm{E}$) lens model plus external shear ($\gamma_\mathrm{p}=0.1$) transformed by the radial stretching $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})=f_0+f_2|\boldsymbol{\beta}|^2/(2\theta_\mathrm{E}^2)$ with $(f_0,f_2)=(0,0.55)$ and $\boldsymbol{\theta}=(-0.5,0.25)\,\theta_\mathrm{E}$. The negative peak comes from the pole $\boldsymbol{\vartheta}=\boldsymbol{\theta}$ while the central peak is caused by $\kappa(|\boldsymbol{\vartheta}|=0)$. The right panel in Fig. 2 illustrates the integrand after applying the Riemann mapping. The peaks have moved and the pole $\boldsymbol{\vartheta}=\boldsymbol{\theta}$ now lies at the origin $z=0$, as expected. To deal with the second peak, the subpackage `integrals` splits the integration domain to place it on a boundary and to ensure the gradient of the integrand is smooth in the sub\domains.

[8] This package provides an interface to QUADPACK (Piessens et al. 1983) whose routines use the adaptive quadrature method to approximate integrals.

[9] The quantity z represents a complex number $z=x+\mathrm{i}y$ with $(x,y)\in\mathbb{R}^2$.

[10] The term $|\boldsymbol{\vartheta}|$ corresponds to the Jacobian of polar coordinates.

With $\hat{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\alpha}}$, we can evaluate the SPT validity criterion adopted in Unruh et al. (2017) and recalled in Eq. (6). With the same lens model and SPT adopted for Figs. 2 and 3 shows the map $|\Delta\boldsymbol{\alpha}(\boldsymbol{\theta})|$ over a circular grid $|\boldsymbol{\theta}| \leq 2\,\theta_{\rm E}$ in the lens plane. It is worth noting that this figure is similar to the map $|\Delta\boldsymbol{\alpha}(\boldsymbol{\theta})|$ illustrated in the Fig. 7 in Unruh et al. (2017) while $\tilde{\boldsymbol{\alpha}}$ was obtained from two different and independent approaches. Unruh et al. (2017) first calculated $\tilde{\psi}$ by solving numerically a Neumann problem thanks to a successive over-relaxation method (Press et al. 1992) on a square grid of width $4\,\theta_{\rm E}$; then they derived $\tilde{\boldsymbol{\alpha}}$ from $\tilde{\psi}$ using a second-order accurate finite differencing scheme (see their Sect. 3.2 for a detailed overview). This iterative process necessarily requires systematically calculating $\tilde{\psi}$ over the whole square grid. Conversely, in Fig. 3, $\tilde{\boldsymbol{\alpha}}$ is obtained directly from the explicit Eq. (13) for each position on a circular sampling grid[11]. Thus, although the similarity between the two figures confirms the consistency of the two approaches, the semi-analytical approach implemented in `pySPT` yields $\tilde{\boldsymbol{\alpha}}$ at a particular position.

In Wertz et al. (2018), we analyzed the impact of the SPT on time delays in detail. To achieve this, we compared, for a given lens model, the time delays Δt_{ij} between image pairs $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ of a source $\boldsymbol{\beta}$ with the time delays $\Delta\tilde{t}_{ij}$ between the image pairs $(\tilde{\boldsymbol{\theta}}_i, \tilde{\boldsymbol{\theta}}_j)$ of the modified source $\hat{\boldsymbol{\beta}}$ under an SPT. The images $\tilde{\boldsymbol{\theta}}$ satisfy the lens equation $\hat{\boldsymbol{\beta}} = \tilde{\boldsymbol{\theta}} - \tilde{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\theta}}) = \tilde{\boldsymbol{\theta}} - \boldsymbol{\nabla}\tilde{\psi}(\tilde{\boldsymbol{\theta}})$. By construction of $\tilde{\boldsymbol{\alpha}}$, we expect $\tilde{\boldsymbol{\theta}}_i$ to be close to $\boldsymbol{\theta}$, at least in a subregion $\mathcal{U}' \subset \mathcal{U}$ (see Sect. 4 in Wertz et al. 2018). These images $\tilde{\boldsymbol{\theta}}$ can be obtained easily thanks to the subpackage `lensing`. Because its main class `Model` is designed to accept a user-defined lens model, we benefited from all the tools provided by `lensing` to characterize the SPT-transformed lens model (see Sect. 4.1). The code below illustrates how to generate the SPT-transformed lensing quantities thanks to the use of the subpackages `lensing`, `sourcemapping`, and `spt`.

5. Impact of the SPT on time delays: Empirical estimation

The first empirical estimation of the impact of the SPT on time delays was presented in Schneider & Sluse (2013, 2014). These authors showed that a quadrupole model composed of a SPL profile predicts the same lensed image positions (with a 0.004 arcsec accuracy) as a composite fiducial model (Hernquist profile + generalized NFW + external shear) for a set of sources $\hat{\boldsymbol{\beta}}$ and $\boldsymbol{\beta}$, respectively[12]. We want to stress that the SPL model is not an SPT-generated model from the composite fiducial model but they represent two different models for which the nature of the degeneracy can be approximated by an SPT. Thus, the set of sources $\hat{\boldsymbol{\beta}}$ was obtained independent of $\boldsymbol{\beta}$ by fitting the lensed image positions produced by the fiducial model. The

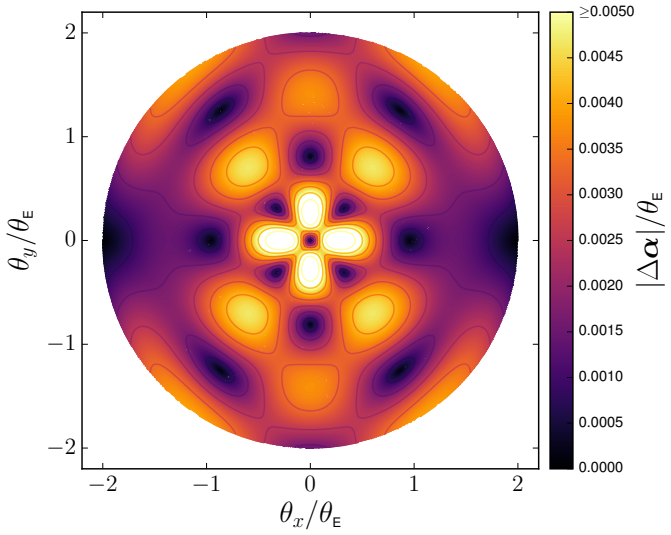

Fig. 3. Map of $|\Delta\boldsymbol{\alpha}(\boldsymbol{\theta})|$ over a circular grid $|\boldsymbol{\theta}| \leq 2\,\theta_{\rm E}$ for $f_2 = 0.55$, $\theta_{\rm c} = 0.1\,\theta_{\rm E}$ and $\gamma_{\rm p} = 0.1$. We set the radius R of the circular region $\mathcal{U}$ in such a way that the area of $\mathcal{U}$ is equal to the area of the square grid used in the pure numerical approach, i.e., $R = 4\,\theta_{\rm E}/\sqrt{\pi} \approx 2.257\,\theta_{\rm E}$. This figure is similar to the Fig. 7 in Unruh et al. (2017) even though it is based on a different approach (see the text for more details). This figure has been obtained with the subpackage `spt` in less than five minutes for a grid of about 2×10^4 positions.

top panel of the Fig. 4 in Schneider & Sluse (2014) represents $|\hat{\boldsymbol{\beta}}|/|\boldsymbol{\beta}|$ as a function of $|\boldsymbol{\beta}|$. For sources located in a disk of radius 0.7 arcsec, the connection between $\hat{\boldsymbol{\beta}}$ and $\boldsymbol{\beta}$ is slightly anisotropic and roughly resembles a radial stretching of the form Eq. (8) with $f_0 = -0.068$ and $f_2 \approx 0.012$. The bottom panel of the Fig. 4 in Schneider & Sluse (2014) shows that $(\Delta\hat{t}/\Delta t)/(|\hat{\boldsymbol{\beta}}|/|\boldsymbol{\beta}|)$ were almost constant in the double image regime and not conserved in the quadruple image regime. These authors also found that $(\Delta\hat{t}/\Delta t)/(|\hat{\boldsymbol{\beta}}|/|\boldsymbol{\beta}|)$ was never smaller than 1.2, reaching a mean of 1.45 for the quadruple image configuration of a source located at $|\boldsymbol{\beta}| \approx 0.18$ arcsec. Thus, this figure shows that the degeneracy between the SPT and fiducial models can affect the inferred value of H_0 by an average of 20% to up to 45% for particular image configurations.

As a first application of `pySPT`, we compare these time delay ratios with those obtained when we transform the fiducial model under a radial stretching Eq. (8) with $(f_0, f_2) = (-0.068, 0.012)$. As shown in Fig. 4, we find that the impact of the SPT on time delays is much smaller than predicted in Fig. 4 in Schneider & Sluse (2014). For instance, they found that a source located at $|\boldsymbol{\beta}| = 0.7$ arcsec leads to $\Delta\hat{t}/\Delta t \approx 1.169$ (between the two outer images), whereas we find $\Delta\hat{t}/\Delta t \approx 0.935$, knowing that the major contribution comes from an MST with $\lambda = 1 + f_0 = 0.932$. Moreover, the small anisotropic feature of the empirical source mapping $\hat{\boldsymbol{\beta}}(\boldsymbol{\beta})$ alone cannot explain this tension.

The discrepancy between our prediction and the results obtained in Schneider & Sluse (2014) are triggered by a minor bug in the public lens modeling code `lensmodel` (v1.99; Keeton 2001a) that the authors used to compute the time delays. We spotted this bug when we compared the outputs produced with our package `pySPT` and `lensmodel` for the deflection angle and deflection potential for the SPL model. Denoting the logarithm slope of the SPL model as a, we found that $2\,\hat{\kappa}_{\texttt{lensmodel}} = a\,\boldsymbol{\nabla}^2\psi_{\texttt{lensmodel}}$ and $\hat{\boldsymbol{\alpha}}_{\texttt{lensmodel}} = a\,\boldsymbol{\nabla}\psi_{\texttt{lensmodel}}$, showing a

[11] We may have chosen a square or another shape for the sampling grid. The choice of a disk is motivated by the fact that the region in which multiple images occur is typically $|\boldsymbol{\theta}| \leq 2\,\theta_{\rm E}$. Furthermore, the radius of the circular sampling grid over which we evaluate $|\Delta\boldsymbol{\alpha}|$, which is depicted in Fig. 3, is not the radius R of $\mathcal{U}$. For each position on the grid, solving the Eq. (13) requires to define R. For consistency, we must adopt the same R for each evaluation of $\tilde{\boldsymbol{\alpha}}$, which implies that R must be chosen (at least equal or) larger than the radius of the circular sampling grid.

[12] We follow the same notation as in Schneider & Sluse (2014), i.e., we denote the lensing quantities associated with the SPL model with a *hat*, e.g., $\Delta\hat{t}$ for the time delay, whereas no *hat* is used for the composite fiducial model.

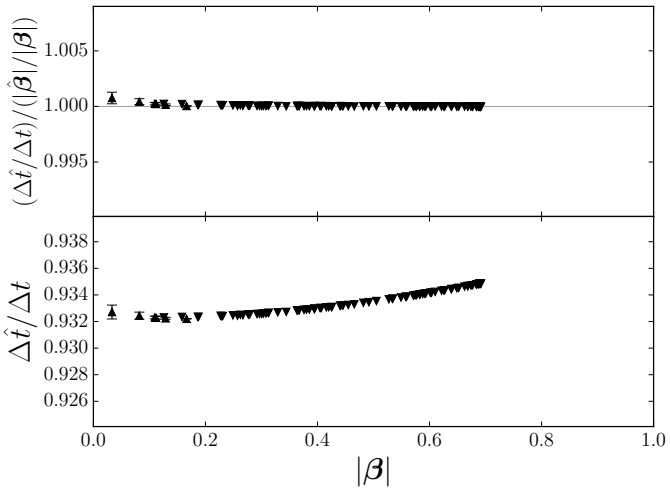

Fig. 4. Time delay ratios of image pairs between the composite fiducial model and its SPT-transformed counterpart under a radial stretching with $(f_0, f_2) = (-0.068, 0.012)$. *Top panel*: $\Delta\hat{t}/\Delta t$ normalized by the ratio $|\hat{\boldsymbol{\beta}}|/|\boldsymbol{\beta}|$ are very close to 1, even in the quadruple image regime, which disagree with the results obtained in Schneider & Sluse (2014). *Bottom panel*: The impact of the SPT (cleaned from the pure MST with $\lambda = 1 + f_0 = 0.932$) is around only a few tenths of percent. The error bars illustrates that the time delay ratios are not conserved in the quadruple image regime, as first suggested in Schneider & Sluse (2014).

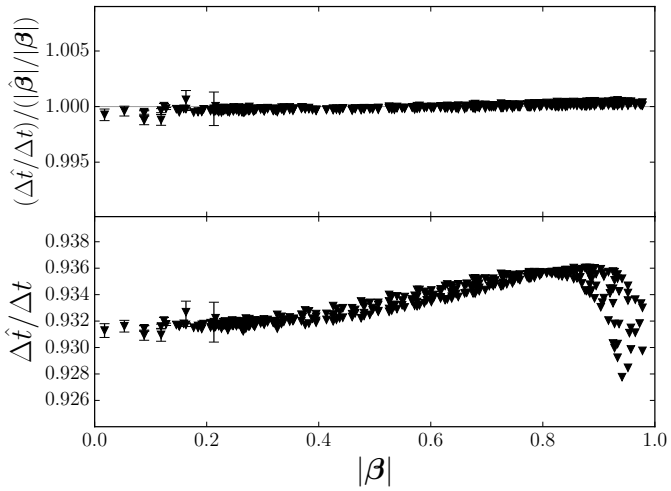

Fig. 5. Time delay ratios of image pairs between the composite fiducial model and quadrupole composed of a SPL. This figure constitutes the corrected version of the Fig. 4 published in Sect. 4.3 in SS14. The hat lensing quantities correspond to the SPL model while standard notation is used for the composite fiducial model.

different normalization factor between $\hat{\kappa}$, $\hat{\boldsymbol{\alpha}}$, and $\hat{\psi}$. This extra normalization factor propagates into the code, leading to a biased value of the time delay $\Delta\hat{t}_{\texttt{lensmodel}}$. It is worth mentioning that, for isothermal profiles ($a = 1$), this extra normalization factor a has no impact on the lensing quantities computed with `lensmodel`. This might explain why this minor bug has remained unnoticed so far. Nevertheless, the latter has been fixed and a corrected version of `lensmodel` has been immediately released by Chuck Keeton. To obtain the Fig. 4 in Schneider & Sluse (2014), the SPL model was characterized by the logarithmic three-dimensional slope $\gamma' = 2.24$, which is linked to a by the relation $a = 3 - \gamma'$, hence $a = 0.76 \neq 1$, which finally explains the discrepancy mentioned before.

In Fig. 5, we show the corrected version of the Fig. 4 that we produced with our package pySPT. We confirm that the exact same result can now be obtained from the corrected version of `lensmodel`. The normalized time delay ratios $(\Delta\hat{t}/\Delta t)/(|\hat{\boldsymbol{\beta}}|/|\boldsymbol{\beta}|)$ plotted against $|\boldsymbol{\beta}|$ (top panel) are very close to 1. Thus, the time delay ratios $(\Delta\hat{t}/\Delta t)$ (bottom panel) closely resembles the source ratios $|\hat{\boldsymbol{\beta}}|/|\boldsymbol{\beta}|$ represented in top panel in Fig. 4. This behavior is well understood and is described in detail in the companion paper Wertz et al. (2018). The impact of the SPT (separated from the MST with $1 + f_0 = 0.932$) on time delays now reaches only around 0.32% for $|\boldsymbol{\beta}| = 0.7$ arcsec, which is obviously much smaller than the 20% previously found in Schneider & Sluse (2014). The corrected version now fully agrees with the conclusions drawn from Fig. 4. In addition, this confirms that the degeneracy between the SPT and the fiducial models mimics an SPT, as first established in Unruh et al. (2017).

6. Conclusions

In this paper we have presented the pySPT package for analyzing the SPT. The pySPT program relies on several subpackages, the most important of which are `lensing` to deal with lens model, `sourcemapping` to define an SPT, and `spt` to provide the SPT-transformed lensing quantities defined in previous papers. The subpackage `lensing` is particular in a sense that it can be used independent of any SPT analysis. To some extent, it somewhat offers a `python` alternative to the public lens modeling code `lensmodel` with which it shares a lot of functionalities. The pySPT program implements functionalities for generating lensing quantities produced by SPT-transformed lens models. Thanks to its modularity, pySPT is also designed to accept both user-defined lens model and SPT. In such a case, pySPT constitutes a user friendly interface to deal with the SPT.

As a first application, we used pySPT to explore how a radial stretching may affect the time delay measurements for a fiducial model composed of a Hernquist profile + generalized NFW + external shear. We found that the impact of the SPT is much smaller than first suggested in Schneider & Sluse (2014). We addressed the tension between these results by spotting a minor bug in the public lens modeling code `lensmodel` that was used by the authors. This bug resulted in biased values of the deflection potential for the nonisothermal SPL model, leading to an overestimated impact of the SPT on the time delays. Using the subpackage `lensing`, we produced a corrected version of the Fig. 4 published in Schneider & Sluse (2014), which now fully agrees with the results presented in this paper. As a result, the impact of the SPT on time-delay cosmography might not be as crucial as initially suspected. We address this question in details in the companion paper Wertz et al. (2018).

With the next version of pySPT, we plan to include state-of-the-art lens modeling capabilities. For example, combining stellar dynamics data obtained from spectroscopy of the lens galaxy with lensing measurements has become a standard practice within the strong lensing community. Furthermore, we still do not have a clear answer to the question of how the kinematic information of a mass distribution is affected under an SPT. Schneider & Sluse (2013) showed that the fiducial and SPL models discussed in Sect. 5 could not be satisfactorily distinguished thanks to the measurement of the stellar velocity dispersion σP with a typical 10% uncertainty. This thus suggests that the use of σP may be of limited help for breaking the SPT. In a future work, we aim to address this open question with the use of pySPT. In this context, we plan to update the software with

a new subpackage fully dedicated to the determination of the stellar velocity dispersion associated with an SPT-modified mass profile. Finally, pySPT is provided not only as an ultimate tool for the lens modeling community, but primarily as an attractive choice to identify the possible degeneracies from which time-delay cosmography may suffer. Nonetheless we hope that its permanent development will attract more users and will extend its purpose to more than just dealing with the SPT.

Acknowledgements. We would like to thank Dominique Sluse and Chuck Keeton for valuable discussions that allowed us to spot and fix the small bug in `lensmodel`. We are very grateful to the anonymous referee for his comments and suggestions that contributed to improving the quality of pySPT and this paper. This work was supported by the Humboldt Research Fellowship for Postdoctoral Researchers.

References

Bar-Kana, R. 1996, ApJ, 468, 17
Birrer, S., Amara, A., & Refregier, A. 2016, J. Cosmol. Astropart. Phys., 8, 020
Falco, E. E., Gorenstein, M. V., & Shapiro, I. I. 1985, ApJ, 289, L1
Fassnacht, C. D., Gal, R. R., Lubin, L. M., et al. 2006, ApJ, 642, 30
Hunter, J. D. 2007, Comput. Sci. Eng., 9, 90
Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open Source Scientific Tools for Python, Online; accessed 2017-08-07
Keeton, C. R. 2001a, ArXiv e-prints [arXiv: astro-ph/0102341]
Keeton, C. R. 2001b, ArXiv e-prints [arXiv: astro-ph/0102340]
Keeton, C. R. 2003, ApJ, 584, 664
Muñoz, J. A., Kochanek, C. S., & Keeton, C. R. 2001, ApJ, 558, 657
Piessens, R., de Doncker-Kapenga, E., Überhuber, C., & Kahaner, D. 1983, QUADPACK: A Subroutine Package for Automatic Integration (Berlin, Heidelberg: Springer-Verlag), 0179
Press, W. H., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in C 2nd edn.: The Art of Scientific Computing (Cambridge, UK: Cambridge University Press)
Schneider, P. 2006, in Saas-Fee Advanced Course 33: Gravitational Lensing: Strong, Weak and Micro, eds. G. Meylan, P. Jetzer, P. North, C. S. Kochanek, &, J. Wambsganss, 1
Schneider, P., & Sluse, D. 2013, A&A, 559, A37
Schneider, P., & Sluse, D. 2014, A&A, 564, A103
Seljak, U. 1994, ApJ, 436, 509
Suyu, S. H., Marshall, P. J., Auger, M. W., et al. 2010, ApJ, 711, 201
Suyu, S. H., Auger, M. W., Hilbert, S., et al. 2013, ApJ, 766, 70
Treu, T., & Marshall, P. J. 2016, A&ARv, 24, 11
Unruh, S., Schneider, P., & Sluse, D. 2017, A&A, 601, A77
Van Der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, Comput. Sci. Eng., 13, 22
Watters, A., Ahlstrom, J. C., & Rossum, G. V. 1996, Internet Programming with Python (New York: Henry Holt and Co., Inc.)
Wertz, O., Orthen, B., & Schneider, P. 2018, A&A, 617, A140
Wilson, G., Aruliah, D. A., Titus Brown, C., et al. 2014, PLOS Biol., 12, e1001745
Wong, K. C., Keeton, C. R., Williams, K. A., Momcheva, I. G., & Zabludoff, A. I. 2011, ApJ, 726, 84
Wong, K. C., Suyu, S. H., Auger, M. W., et al. 2017, MNRAS, 465, 4895

Appendix A: Generalized pseudo-NFW

To our knowledge, no analytical expression of the deflection potential $\psi(\boldsymbol{\theta})$ for the generalized pseudo-NFW model has ever been published in the literature. For practical purposes, we present such an analytical expression, which has been implemented into pySPT. We first recall that the spherical density distribution $\rho(r)$ of the generalized pseudo-NFW model is defined as (see the Eq. (1) in Muñoz et al. 2001, with $n = 3$)

$$\rho(r) = \frac{\rho_\mathrm{s}}{(r/r_\mathrm{s})^\gamma[1 + (r/r_\mathrm{s})^2]^{(3-\gamma)/2}}, \tag{A.1}$$

where ρ_s is a characteristic density, r_s the scale radius, and γ the logarithmic slope of the density profile at small radius. Up to an additive constant, the axisymmetric deflection potential $\psi(\boldsymbol{\theta})$ is given by

$$\psi(\boldsymbol{\theta}) = r_\mathrm{s}\,\kappa_\mathrm{s}\left[\frac{K\left(0, \frac{3-\gamma}{2}; \gamma; \frac{|\boldsymbol{\theta}|}{r_\mathrm{s}}\right) - K\left(1, 0; \gamma; \frac{|\boldsymbol{\theta}|}{r_\mathrm{s}}\right)}{\Gamma^*(\gamma/2)} - \mathrm{Li}_2\left(-\frac{|\boldsymbol{\theta}|^2}{r_\mathrm{s}^2}\right)\right], \tag{A.2}$$

where $\kappa_\mathrm{s} = \rho_\mathrm{s}\, r_\mathrm{s}/\Sigma_\mathrm{cr}$, the term $\Gamma^*(\upsilon) = \Gamma(\upsilon)/\Gamma(\upsilon - 1/2)$ is a particular combination of the gamma function Γ,

$$K(k_0, l; m; z) = \sum_{k=k_0}^{+\infty} \frac{z^{2(k+l)}}{(k+l)^2}\Gamma^*\left[k + l + m/2\right]\, {}_2F_1^*\left[k + l, z\right], \tag{A.3}$$

where ${}_2F_1^*\left[a, z\right] = {}_2F_1\left[a, a, a+1, -z^2\right]$ is a particular Gauss hypergeometric function, and the dilogarithm $\mathrm{Li}_2(z)$ can be defined by the series

$$\mathrm{Li}_2(z) = \sum_{k=1}^{+\infty} \frac{z^k}{k^2}. \tag{A.4}$$

Appendix B: Proof of the relation (10)

As a preamble, the notation adopted here differs from that used in Unruh et al. (2017). We use $\boldsymbol{\theta}$ as a position in the lens plane and $\boldsymbol{\vartheta}$ as the corresponding integration variable for $\boldsymbol{\theta}$. The inverse is partially used in Unruh et al. (2017), in particular in the Sect. 3.3 where $\tilde{\psi}$ and $\tilde{\boldsymbol{\alpha}}$ are derived.

Starting with Eq. (18) in Unruh et al. (2017), the deflection potential $\tilde{\psi}$ evaluated at the position $\boldsymbol{\theta}$ is given by

$$\tilde{\psi}(\boldsymbol{\theta}) = \langle\tilde{\psi}\rangle + 2\int_{\mathcal{U}} H(\boldsymbol{\theta};\boldsymbol{\vartheta})\,\hat{\kappa}(\boldsymbol{\vartheta})\,\mathrm{d}^2\vartheta - \int_{\partial\mathcal{U}} H(\boldsymbol{\theta};\boldsymbol{\vartheta})\,\hat{\boldsymbol{\alpha}}\cdot\boldsymbol{n}\,\mathrm{d}s, \tag{B.1}$$

where a solution for the Green function H is analytically known when $\mathcal{U}$ is a disk of radius R

$$H(\boldsymbol{\theta};\boldsymbol{\vartheta}) = \frac{1}{4\pi}\left[\ln\left(\frac{|\boldsymbol{\vartheta}-\boldsymbol{\theta}|^2}{R^2}\right) + \ln\left(1 - \frac{2\,\boldsymbol{\vartheta}\cdot\boldsymbol{\theta}}{R^2} + \frac{|\boldsymbol{\vartheta}|^2\,|\boldsymbol{\theta}|^2}{R^4}\right)\right] - \frac{|\boldsymbol{\vartheta}|^2 + |\boldsymbol{\theta}|^2}{4\,\pi\,R^2}. \tag{B.2}$$

Firstly, we note that $|\boldsymbol{\vartheta}| = R$ for all $\boldsymbol{\vartheta}$ on the boundary $\partial\mathcal{U}$, which implies

$$1 - \frac{2\,\boldsymbol{\vartheta}\cdot\boldsymbol{\theta}}{R^2} + \frac{|\boldsymbol{\vartheta}|^2\,|\boldsymbol{\theta}|^2}{R^4} = \frac{|\boldsymbol{\theta}|^2}{R^2} - \frac{2\,\boldsymbol{\vartheta}\cdot\boldsymbol{\theta}}{R^2} + \frac{|\boldsymbol{\vartheta}|^2}{R^2} = \frac{|\boldsymbol{\vartheta}-\boldsymbol{\theta}|^2}{R^2}. \tag{B.3}$$

Thus, the two logarithm-terms in $H(\boldsymbol{\theta};\boldsymbol{\vartheta})$ are equal when we consider the line integral.

Secondly, the term $-|\boldsymbol{\theta}|^2/(4\pi R^2)$ in $H(\boldsymbol{\theta};\boldsymbol{\vartheta})$ does not depend on $\boldsymbol{\vartheta}$, and therefore contributes neither to the integral over $\mathcal{U}$ nor to the line integral. Therefore, Eq. (B.1) contains the term

$$-\frac{|\boldsymbol{\theta}|^2}{4\pi R^2}\left(2\int_{\mathcal{U}}\hat{\kappa}(\boldsymbol{\vartheta})\,\mathrm{d}^2\vartheta - \int_{\partial\mathcal{U}}\hat{\boldsymbol{\alpha}}\cdot\boldsymbol{n}\,\mathrm{d}s\right) = 0, \tag{B.4}$$

where the equality holds because of $2\,\hat{\kappa} = \boldsymbol{\nabla}\cdot\hat{\boldsymbol{\alpha}}$ and we made use of Gauß divergence theorem. As a result, the term $-|\boldsymbol{\theta}|^2/(4\pi R^2)$ in $H(\boldsymbol{\theta};\boldsymbol{\vartheta})$ does not contribute to $\tilde{\psi}$.

Finally, combining Eqs. (B.1), (B.3), and (B.4) leads to the definition of $\tilde{\psi}$ given in Eq. (10). We note that the same reasoning holds for $\tilde{\boldsymbol{\alpha}}$, leading to the Eq. (13).