Astronomical Data Analysis Software and Systems XXV
ASP Conference Series, Vol. 512
Nuria P. F. Lorente, Keith Shortridge, and Randall Wayth, eds.

SimCADO - An End-to-end Instrument Data Simulator for MICADO on the E-ELT

Kieran Leschinski,[1] Oliver Czoske,[1] Rainer Köhler,[2,1] Werner Zeilinger,[1] Wolfgang Kausch,[1,2] Norbert Przybilla,[2] João Alves,[1] Michael Mach,[1] Gijs Verdoes Kleijn,[3] Richard Davies,[4] Ronny Ramlau,[5,6] Roland Wagner,[6] Thorsten Ratzka,[7] Robert Greimel,[7] Martin Leitzinger,[7] Veronika Schaffenroth,[2] Manuel Güde,l[1] and Bodo Ziegler[1]

[1]*Department of Astrophysics, University of Vienna, Austria;*
`kieran.leschinski@univie.ac.at`

[2]*Institute for Astro- and Particle Physics, University of Innsbruck, Austria*

[3]*Kapteyn Astronomical Institute, University of Groningen, Netherlands*

[4]*Max Planck Institute for Extraterrestrial Physics, Garching, Germany*

[5]*Industrial Mathematics Institute, Johannes Kepler University, Linz, Austria*

[6]*Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Linz, Austria*

[7]*Institute for Physics/IGAM NAWI, Graz, University of Graz, Austria*

Abstract. MICADO will be the first-light wide-field imager for the European Extremely Large Telescope (E-ELT) and will provide diffraction limited imaging (7 mas at 1.2 μm) over a $\sim 53''$ field of view. As part of the development of this instrument we are building an end-to-end instrument data simulator. SimCADO generates a full model of the optical train from source to detector readout. SimCADO is thus a tool to provide simulated detector array images to both the science and instrument development teams. Here we present an overview of the inner workings of SimCADO and outline our plan for its further development.

1. Motivation for developing SimCADO

Building a new instrument is an iterative process between the science team and the design team. The design team requires a list of science drivers in order to tailor the instrument design and fulfil the science requirements. The science team on the other hand requires information on the instrument design and its expected performance in order to generate a list of realistic science drivers. This mutual dependency therefore requires a high degree of communication between both teams if an efficient and timely design of the instrument is to happen. A quick and easy-to-use instrument data simulator is one way of providing a platform to facilitate the transfer of information between both teams. The goals of SimCADO are thus fourfold:

1. collect and use the most recent results of detailed subpackage simulations to facilitate continual science case analysis,
2. estimate the performance of MICADO to aid design decisions,
3. generate realistic mock detector frames for developing and testing the pipelines and
4. provide a tool to the astronomical community to assist in the evaluation and planning of observation strategies.

2. What is SimCADO?

SimCADO is a python package that will enable the construction of a virtual model of the optical path through the atmosphere, the E-ELT (Gilmozzi & Spyromilio 2007) and MICADO (Davies et al. 2010) and then visualise how astronomical objects will appear on the detector array. For the development of SimCADO, we have chosen to use an array-based approach instead of a photon-following approach. Photon-following approaches like the LSST PhoSim tool (Peterson et al. 2015), offer a high degree of accuracy, however they require access to massive computing facilities. Array-based approaches, like those used in the METIS (Schmalzl et al. 2012) and HARMONI (Zieleniewski et al. 2014) simulators, are fast and are designed to run on personal computers. The accuracy of these array-based approaches is expected to be sufficient for scientific feasibility studies and the current trade-off analyses.

Although SimCADO has been constructed with MICADO in mind, in practice any telescope can be modelled as long as the user has the relevant technical data for the desired telescope configuration. The SimCADO python package consists of a series of class objects which represent different aspects of the optical elements. In general these class objects can be split into three major categories:

- objects which affect the spatial distribution of incoming photons,
- objects which introduce wavelength-dependent changes to the incoming photons,
- objects which represent photon sources (e.g. science target, sky background, detector noise, etc)

The user is free to choose the combination of these class objects which best describes the elements in the optical system. For example, the point spread function (PSF) of the E-ELT's system of five mirrors may be represented by one 3-dimensional (x,y,λ) wavelength-dependent image cube combined with a 1-dimensional (λ) spectral curve for the mirror reflectivity.

As MICADO has just entered the preliminary design phase, the initial versions of SimCADO will be shipped with the technical data generated during the Phase A study. Hence SimCADO will, at first, only be able to simulate images based on this information. As more and more details about the individual subsystem components become available, these will be incorporated into SimCADO updates, allowing SimCADO to more faithfully represent the real detector array output.

3. How to generate an optical train

Due to the modular way in which we have designed SimCADO, generating a simple optical path model is a matter of $\sim 20 - 30$ lines of python code. Figure 1 shows the steps needed to generate a rudimentary telescope model. In this section we give a brief run down of the python commands needed to achieve this.

This (non-exhaustive) list of classes and functions is purely to illustrate the main steps involved in generating an optical path model. SimCADO provides pre-configured high-level functions for simulating the MICADO optical train, however the user is free to use the basic building blocks to test other optical configurations.

`PsfCube()` and `SystemPsf()`: PSF cubes are used to represent a series of wavelength dependent PSFs in order to model the major spatial effects - e.g. atmospheric dispersion, telescope diffraction, atmospheric seeing - not removed by the adaptive optics (AO) system, etc. The class `PsfCube()` can currently generate PSFs with a Gaussian, Airy, Moffat, or shifted point kernel for a list of wavelengths at a specific pixel scale. Alternatively the user may import a FITS cube with PSFs. The `SystemPsf()` object takes a list of `PsfCube` instances and produces a single PSF cube which describes the whole optical train. It is this combined PSF cube that the object will "see" when the optical train is applied to the input source cube.

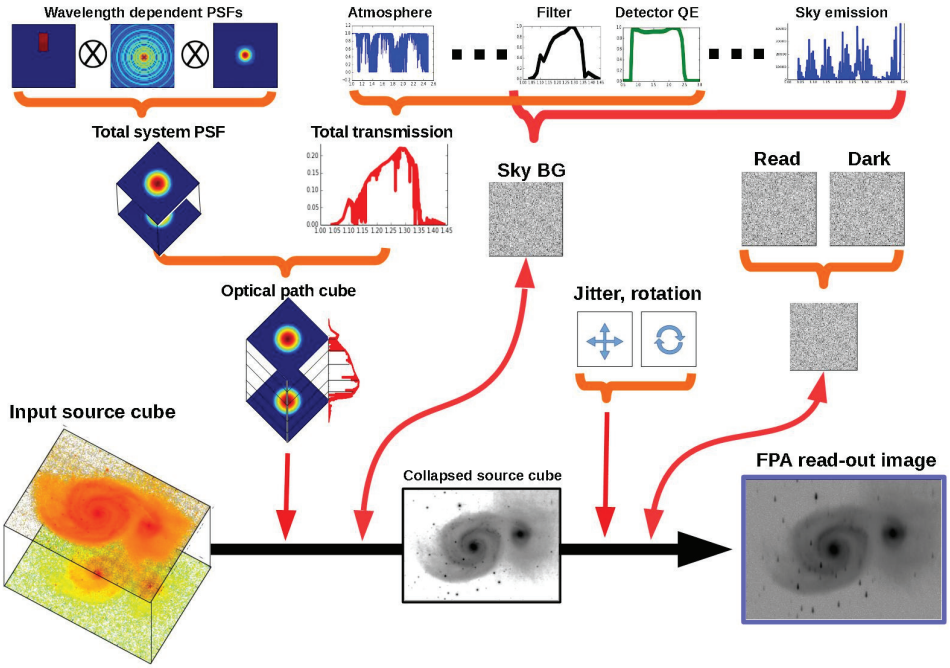

Figure 1. SimCADO builds a model of the optical path out of a combination of basic building blocks. The PSF cube objects represent 2D spatial effects caused by the optical elements (either wavelength independent or dependent). The spectral curve objects contain wavelength dependent information, e.g. transmission curves.

`SpectralCurve()` and `SystemCurve()`: Reflectivity curves, transmission curves and emission curves all use this class. Once read, the curve is resampled to the specified resolution. Multiple `SpectralCurve` instances can be combined in a single `SpectralCurve()` object where all spectral curves are resampled to the coarsest resolution before being combined along a new common wavelength range. The resulting `SystemCurve` can then be linked straight to the `SystemPsf`.

`InputCube()` and `OutputCube()`: The "science" data cube contains the spatial and wavelength information about the target to be "observed" with MICADO. The data are imported along with information in the FITS header as an instance of the `InputCube`. The `OutputCube` represents the science data cube after it has passed through the optical path model. It is as if all the photons had (figuratively) stopped still just in front of the focal plane array (FPA). Spectral and spatial information is still preserved up until this point. The `OutputCube` is also converted to photon counts per second so that the conversion to electrons in the focal plane array is straight forward.

`Fpa()`: The final Focal Plane Array (FPA) readout is generated by creating an `Fpa` instance. It produces realistic photon noise estimates by simulating the full detector sampling scheme as appropriate for IR detectors. The FPA class also simulates various sources of detector noise for the HAWAII 4RG, following the HxRG Noise Generator package developed for the JWST (Rauscher 2015).

Although most of the core features have been implemented, SimCADO is still in an early phase of development. We plan to release an alpha version of the code with the core features to the MICADO science team in January 2016 for testing purposes, with the dual goals of evaluating how the code performs as well as the user experience.

Acknowledgments. This publication is supported by the Austrian Ministry of Science, Research and Economy (bmwfw) through project IS538003 (Hochschulraumstrukturmittel). GVK acknowledges support by the Netherlands Research School for Astronomy (NOVA) and Target. Target is supported by Samenwerkingsverband Noord Nederland, European fund for regional development, Dutch Ministry of economic affairs, Pieken in de Delta, Provinces of Groningen and Drenthe.

References

Davies, R., et al. 2010, in SPIE Conf. Series, vol. 7735 of SPIE Conf. Series, 2
Gilmozzi, R., & Spyromilio, J. 2007, The Messenger, 127, 11
Peterson, J. R., et al. 2015, ApJS, 218, 14
Rauscher, B. J. 2015, PASP, 127, 1144
Schmalzl, E., et al. 2012, in SPIE Conf. Series, vol. 8449, 1
Zieleniewski, S., et al. 2014, in SPIE Conf. Series, vol. 9147, 93