MAISIE: a Multipurpose Astronomical Instrument SImulator Environment

Alan O'Brien[a], Steven Beard[a], Vincent Geers[a], and Pamela Klaassen[a]

[a]The UK Astronomy Technology Centre, Blackford Hill, Edinburgh, EH9 3HJ, UK

ABSTRACT

Astronomical instruments often need simulators to preview their data products and test their data reduction pipelines. Instrument simulators have tended to be purpose-built with a single instrument in mind, and attempting to reuse one of these simulators for a different purpose is often a slow and difficult task. MAISIE is a simulator framework designed for reuse on different instruments. An object-oriented design encourages reuse of functionality and structure, while offering the flexibility to create new classes with new functionality. MAISIE is a set of Python classes, interfaces and tools to help build instrument simulators. MAISIE can just as easily build simulators for single and multi-channel instruments, imagers and spectrometers, ground and space based instruments. To remain easy to use and to facilitate the sharing of simulators across teams, MAISIE is written in Python, a freely available and open-source language. New functionality can be created for MAISIE by creating new classes that represent optical elements. This approach allows new and novel instruments to add functionality and take advantage of the existing MAISIE classes. MAISIE has recently been used successfully to develop the simulator for the JWST/MIRI- Medium Resolution Spectrometer.

Keywords: Instrument Simulation, MAISIE, MIRI, MRS, Python, Software

1. INTRODUCTION

Simulated data for astronomical instruments and telescopes provides the communities with valuable opportunities to study approximate data-products before operation. The simulated data products can inform the development and design of data reduction and calibration pipelines before the instrument or telescope is operational. By quickly producing data in the correct format simulators can provide an understanding of the produced data and give access to large sets of test data. Known sky scenes can be passed through a simulator to produce simulated data frames which can be then be used to test the data reduction pipelines ability to recover data about the original scene.

Generally simulators are custom built for one instrument, this is often done as each instrument and telescope has some aspects which are unique to it. Specsim[1] is an example of one simulator custom built for a single purpose, simulating James Webb Space Telescope (JWST) Mid-Infrared Instrument (MIRI)- Medium Resolution Spectrograph (MRS) data. Modifying one of these proven systems to work on a similar instrument or telescope is possible and has been done with success[2] however changing the requirements of purpose built software can be difficult. Some of the key unique features of the simulator may be hard coded requiring large areas of the code base to be refactored. SCASim[3] was developed to improve on the very basic detector simulation within SpecSim. SCASim is an object orientated, Python based simulator. By creating a separate simulator for the detector, the code can be shared between the different MIRI simulators of different operating modes, reducing the MIRI simulators code base. SCASim was designed as a general purpose simulator for an integrating detector with non-destructive readout, which, by providing new parameters can be use to simulate different detectors. By coding in a language which can be freely distributed, user friendly and Object Orientated, the Python based SCASim is easily distributed amongst teams and collaborators, requiring only a freely available Python interpreter. Following on from the success and reusability of Specsim and SCASim,there is a need for a a general purpose, python based simulator for producing illumination information.

Send correspondence to A.OB.,
A.OB.: E-mail: alan.obrien@stfc.ac.uk, Telephone: +44 131 668 8441

MAISIE (Multipurpose Astronomical Instrument SImulator Environment) is designed to be an open source, general purpose, portable tool to help build instrument simulators.

MAISIE consists of a set of general purpose Python classes and interfaces designed to help build instrument simulators. Combinations of the general purpose classes are used in construction of basic simulators which can then be used to produce simulated data. More advanced simulators that simulate more complex effects will require custom classes that follow the interfaces. These new classes can be used alongside the included classes to produce a bespoke simulator. MAISIE is focused towards building light path simulations. The standard MAISIE-based simulator accepts data cubes of the sky scene and produces detector illumination maps.

2. SIMULATOR COMPONENTS

MAISIE simulators are built with two major components, a setupData object, which contains all the data about the instrument required to be simulated by the simulator, and the simulator object. This is made of python objects and performs the simulation based on the values contained within the setupData.

When a MAISIE simulator carries out a simulation, some information is inherently related to to a component of the instrument or the instrument itself. This is abstracted out into the setupData and simulator structure. Information related to the instrument should be contained within a setupData object. This object only has attributes which relate to different parts of the simulation. The structure of this object should be closely related to the structure of the simulations as discussed below. Different information is more directly related to the light passing through the simulation. MAISIE simulations have a class that represents this information, called a SkyCube. Contained within the skyCube is a dataCube which is a three dimensional array, two spatial dimensions and one spectral dimension, contains the flux per spaxel (spectral pixel), a list of wavelengths corresponding to spaxel edges, the size of the field of view in both spatial directions and other information more related to the photons passing through the instrument than the instrument itself.

The basic building block of a MAISIE simulation is called an Effector: an Effector simulates one or more optical effects. An Effector is an object that encapsulates a particular (singular) aspect of the instrument. Effectors follow a simple interface, see Figure 1, and act as the individual building blocks of a simulation. Each Effector is required to have a constructor, which takes a single setup data argument, and a method called "applyDataTo" which takes single argument: a list of skyCubes. This simple interface allows for the easy creation of a few key independent components which can then be inserted into simulator to extend the functionality provided by MAISIE.

```
class Effector(Object)
    def __init__(setupData object)
        ...
    def applyDataTo(List of skyCubes)
        ...
```

Figure 1. Pseudo code of the Effector Interface

3. BUILT IN FUNCTIONALITY

MAISIE provides some built in classes and methods to help assist the creation of simulators. Not all of provided classes will be appropriate for all simulators, nor are they designed to provide enough fine control for a full detailed simulator; the provided modules should act as a framework and structure for detailed simulators to be built around.

3.1 Effectors

To build a simulator, Effector objects should be created that closely resemble domain objects. That is, an Effector should be created for each abstract component (e.g. Atmosphere, filters, thermal emission). Each Effector should have a single responsibility, that is if there are two effects to simulate there should be two effectors. If a single component causes multiple effects on the skyCube then a single Effector should be created to represent the component which should contain an Effector for each. A simulator is built as a reverse pipeline

where the output of each Effector is connected to the input of the next. Effectors are applied sequentially so order can be significant. They are usually arranged in the order of the light path although exceptions do exist. Transformations can remove spectral and spatial information and so should be the final steps in a simulator. Using this method, complex simulators can be constructed. For example, a simulator may contain many top-level Effectors representing the atmosphere, the telescope and the instrument. Each of these could contain Effectors for layers in the atmosphere, transmission of gratings and mirrors or simulate the thermal emission profile of physical components. Figure 2 gives an example of how to structure a simulator.

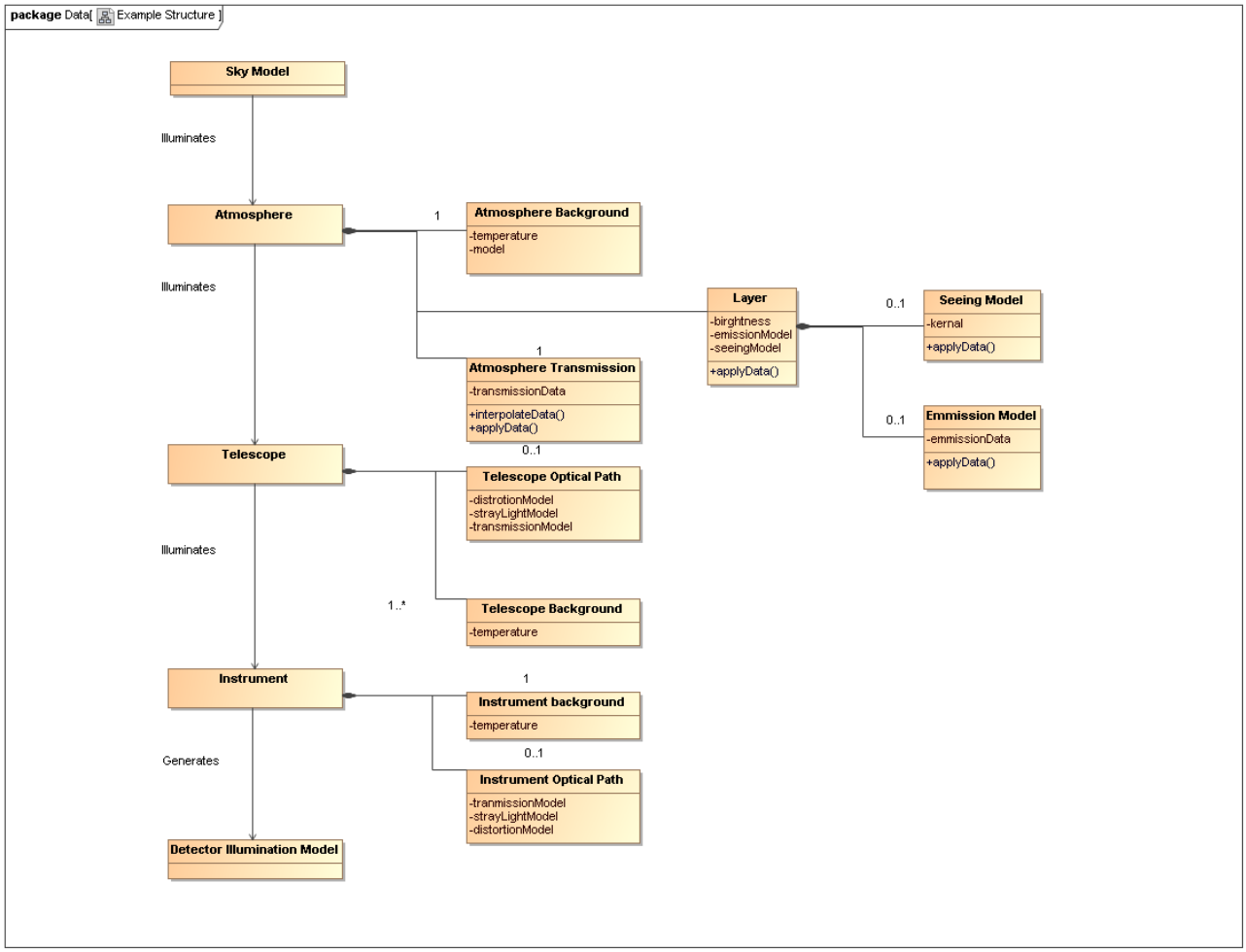

Figure 2. An example of how to structure a simple simulator. For simplicity the optical paths have not been expanded.

MAISIE contains several pre-built effectors for simple use-cases. They range from simple classes that add two data sets, to more complete classes like the transmission Effector which can interpolate data sets and apply a transmission function to a skyCube. If new functionality is required the pre-built Effectors can be extended to simulate new effects, or new Effectors that implement the Effector interface can be created. Basic Effectors are included so non-expert python users can create simulations.

The following sections list some of the built-in Effectors and describes they function.

3.1.1 Example Effector

An example Effector to provide a bare bones example of how an Effector should be structured. New Effectors should extend an existing similar Effector, if no similar effector exists a new Effector should be created following

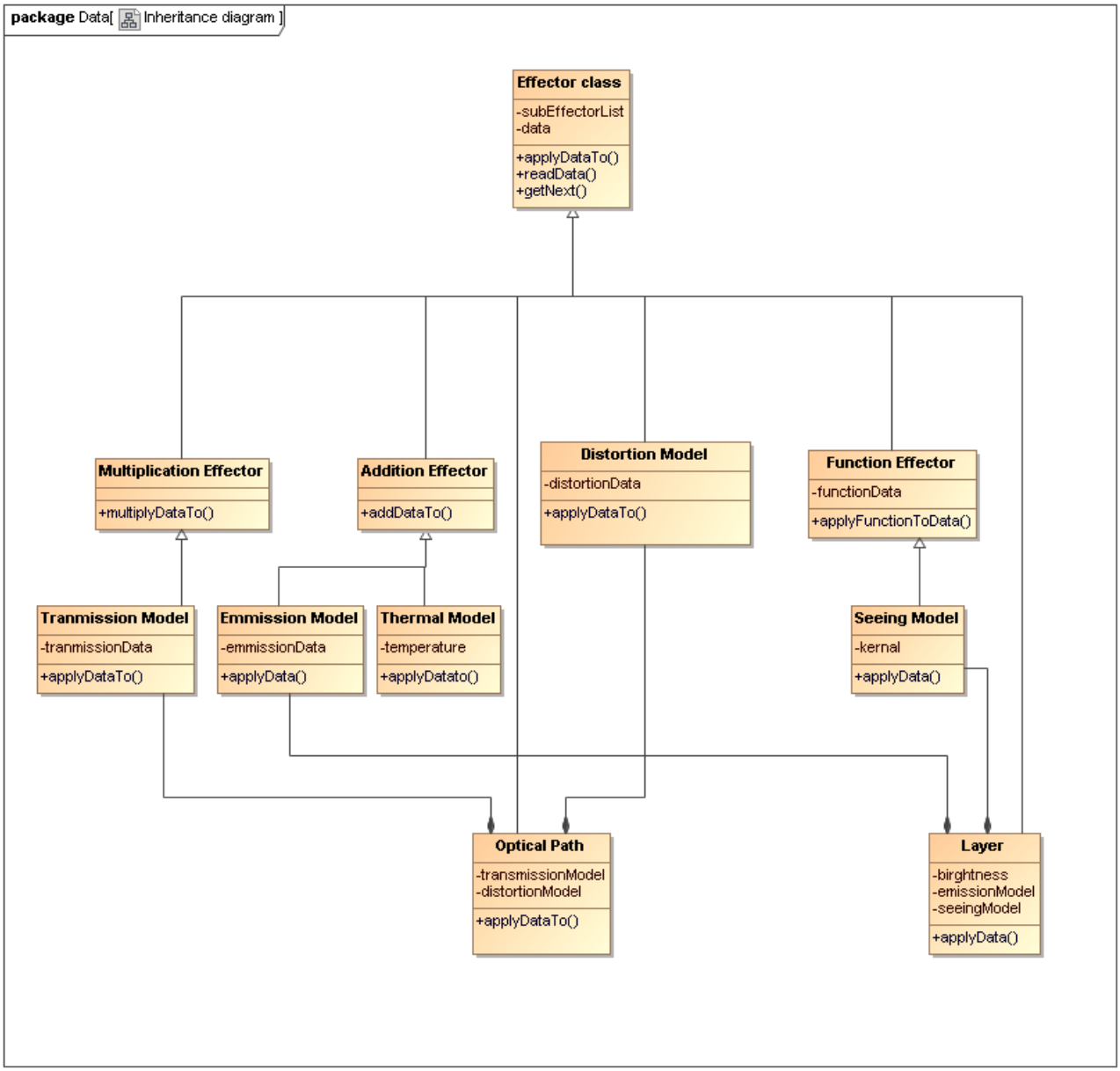

Figure 3. Inheritance diagram showing the relationship between classes. Some classes have been omitted for clarity.

the interface.

3.1.2 Abstract Effectors

To aid in the creation of new Effectors several abstract Effectors are provided. These effectors contain functions that apply data to a data cube in different ways,such as the Multiplication Effector and Addition Effector which multiply and add a dataset to the data cube broadcasting the data as needed. The Function Effector applies a user-defined function to the sky cube and the List Effector apply a series of Effectors to the sky cube.

3.1.3 Transmission Effector

The Transmission effector will take a data file with wavelength and transmission values, interpolate the data to the correct wavelength values and then multiply the transmission values with the data cube.

3.1.4 Emission Effector

Similar to the Transmission Effector, the Emission Effector will take a data set, interpolate it to the correct wavelengths values and add a certain flux to the data cube.

3.1.5 Thermal Emission Effector

The Thermal Emission Effector simulates black-body radiation from within the instrument.

3.1.6 Seeing Effector

The Seeing Effector convolves each two dimensional wavelength slice of the data cube with a kernel (This can also be used to apply simple point spread functions).

3.1.7 Distortion Effector

The Distortion Effector remaps the data cube onto a two dimensional detector surface using polynomial coordinate transformations and the included distortion mapping tools (see Sec.3.2.3).

3.1.8 Optical Path Effector

The Optical Path Effector creates other Effectors using the Object Factory (see Sec.3.2.2), this allows the quick creation of Effectors from a list of files names. The created Effectors are applied sequentially to the data cube.

3.2 Tools

Contained within the tools package of MAISIE are modules to assist in the creation of simulators. Modules within the Tools package do not follow the Effector interface and are intended to help keep Effectors conceptual, while collecting common code

3.2.1 Reader factory

The readerFactory module contains a series of objects which can be used to get data and a readerFactory class to quickly find the right object to access data. The reader object given should extend or implement the same methods as the provided ExampleReader. Reader objects can be used to access data from a local file or get data from a source. The provided Readers allow the reading of a few basic files types (e.g. CSV, JSON, FITS).

New Reader objects can be registered to the readerFactory if new data types or formats need to be read. Registered Reader objects can shadow built in Readers and replace the functionality.

3.2.2 Object factory

To aid the quick construction of simulators MAISIE provides a factory which can be quickly used to generate Effectors from a file, based on the file names and type. More developed, later life simulators should not use this option and instead fully create their own simulators.

3.2.3 Distortion mapping tools

To help assist with re-sampling distortion mapping, MAISIE contains a few simple algorithms for determining the proportional of a quadrilateral is covered by a subsection of a detectors pixels.

4. CASE STUDY

The medium resolution spectrograph (MRS) simulator for the Mid Infrared Instrument (MIRI) of the JWST[4] is the first instrument simulator to be developed using MAISIE. This simulator, which we are calling MIRI-MAISIE, is part of a new unified suite of MIRI simulators called MIRISim. The reusable nature of MAISIE has allowed MIRI-MAISIE to incorporate more easily new effects seen in the MIRI test data, or adapt to changes in the data format. The output of MIRISim can be used with the MIRI MRS Calibration Pipeline.[5]

MIRI-MAISIE can simulate all four Integrated Field Unit (IFU) channels, with each channel having three sub-bands. The MIRI MRS simulation accepts a skyCube object (generated by another component of MIRISim) and a data object read from a file. It then applies the relevant combined transmission values and a general transmission value. The main component of MIRI-MAISIE is contained within the IFU detector transformation Effector. The resulting illumination map is combined with data from the paired channel and, finally, a MIRI Illumination Model is created, Figure 4 show the pipeline within MIRI-MAISIE. MIRI-MAISIE treats MIRI as if it had 12 distinct light paths and is mostly constructed from the built in classes from MAISIE. The new classes include a setupData object and a simulation file, which all new simulators need. A new reader class to allow access to the latest MIRI Calibration Data Products and new detector transformation class.

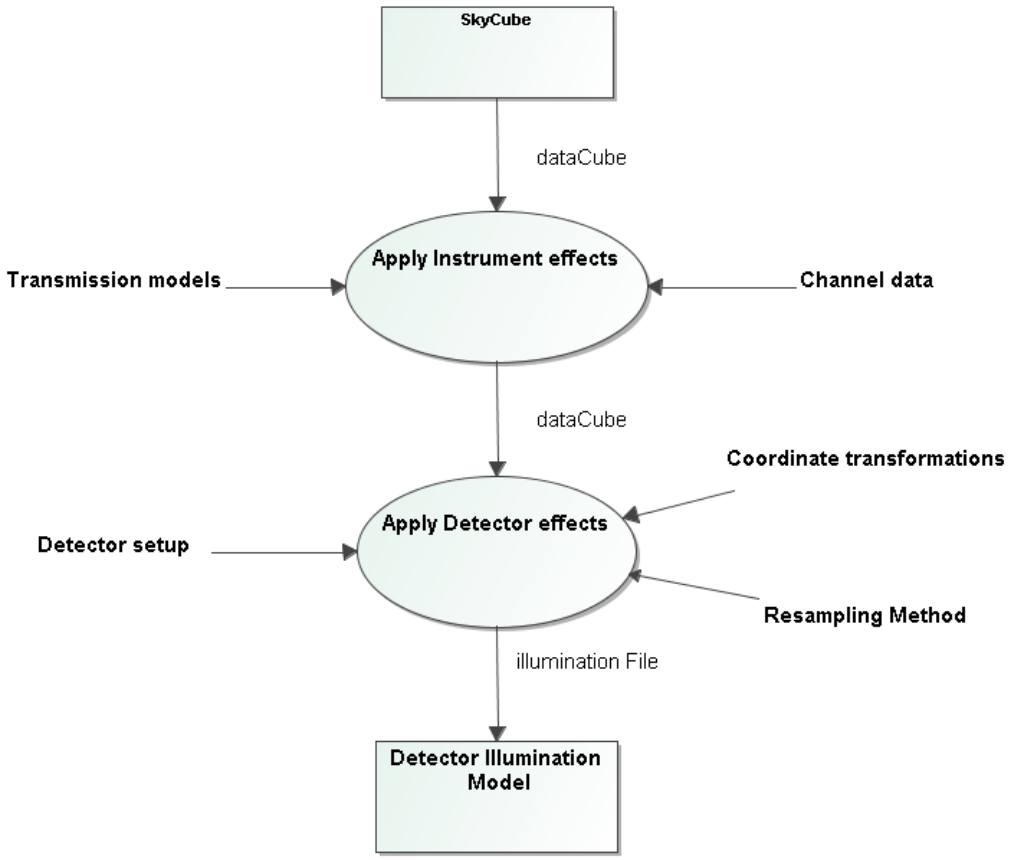

Figure 4. Description of the pipeline within MIRI-MAISIE

5. CONCLUSION AND AVAILABILITY

MAISIE is a simulator builder, consisting of a collection of Python classes, interfaces and tools designed to help assist in the creation of an instrument simulator. It has been successfully use to create MIRI-MAISIE, the MIRI-MRS simulator. MAISIE reduces the effort needed to build a new instrument simulator by providing adaptable and reusable building blocks. The building blocks accept a populated sky cube and produce a detector

illumination model. The three common use cases for MAISIE are: constructing a simple simulator from existing classes; extending the existing Effectors and classes; and creating new Effectors to simulate new effects. MAISIE is freely available with an open license from https://github.com/ukatc/maisie.

ACKNOWLEDGMENTS

MAISIE was originally funded by the Science and Technology Facilities Council (STFC) Centre For Instrumentation and MIRI-MAISIE was constructed with funding, advice and input from the MIRI European Consortium.

We would like to thank and acknowledge Edinburgh-based children's author Aileen Paterson for providing the inspiration for the name MAISIE, by writing many stories about a local cat, amongst them "Maisie goes to Morningside".

REFERENCES

[1] Lorente, N. P., Glasse, A. C., Wright, G. S., and García-Marín, M., "Specsim: the MIRI medium resolution spectrometer simulator," in [*SPIE Astronomical Telescopes+ Instrumentation*], 62741F–62741F, International Society for Optics and Photonics (2006).

[2] Lorente, N., Glasse, A., Wright, G., Ramsay, S., and Evans, C., "Specsim: A Software Simulator for Integral Field Unit Spectrometers," in [*The 2007 ESO Instrument Calibration Workshop*], 295–300, Springer (2008).

[3] Beard, S., Morin, J., Gastaud, R., Azzollini, R., Bouchet, P., Chaintreuil, S., Lahuis, F., Littlejohns, O., Nehme, C., and Pye, J., "SCASim: A Flexible and Reusable Detector Simulator for the MIRI instrument of the JWST," in [*Astronomical Data Analysis Software and Systems XXI*], Ballester, P., Egret, D., and Lorente, N. P. F., eds., *Astronomical Society of the Pacific Conference Series* **461**, 169 (Sept. 2012).

[4] Wells, M., Lee, D., Oudenhuysen, A., Hastings, P., Pel, J.-W., and Glasse, A., "The MIRI medium resolution spectrometer for the James Webb Space Telescope," in [*SPIE Astronomical Telescopes+ Instrumentation*], 626514–626514, International Society for Optics and Photonics (2006).

[5] Labiano, A., Allollini, R., Bailey, J., Beard, S., Dicken, D., García-Marín, M., Geers, V., Glasse, A., Glauser, A., Gordon, K., et al., "The MIRI Medium Resolution Spectrometer Calibration Pipeline," in [*SPIE Astronomical Telescopes+ Instrumentation*], International Society for Optics and Photonics (2016).