PyMGal: A Python Package for Generating Optical Mock Observations from Hydrodynamic Simulations

Patrick Janulewicz,[1,2,3,4]★ Weiguang Cui,[5,6]†
[1]*Department of Physics, McGill University, Montreal, Canada*
[2]*Trottier Space Insititute, McGill University, Montreal, Canada*
[3]*Ciela Institute, Montreal, Canada*
[4] *Mila - Quebec Artificial Intelligence Institute, Montreal, Canada*
[5] *Departamento de Física Teórica, Módulo 15, Facultad de Ciencias, Universidad Autónoma de Madrid, 28049 Madrid, Spain*
[6] *Centro de Investigación Avanzada en Física Fundamental (CIAFF), Facultad de Ciencias, Universidad Autónoma de Madrid, 28049 Madrid, Spain*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

We introduce PyMGal, a Python package for generating optical mock observations of galaxies from cosmological simulations. PyMGal obtains the positions and physical properties of star particles within the simulation and is compatible with different simulation snapshot formats. It then generates spectral energy distributions (SEDs) for these particles based on a variety of stellar population models which can be set and customized on the user's choice for their applications. Given these SEDs, the program can calculate the brightness of particles through a given filter in the user's choice of output units. The particles can then be projected to a 2D plane mimicking a telescope observation. This package provides a great deal of flexibility, allowing the user to select between different models, filters, axes of projection, and output units. The package supports additional features including dust attenuation, particle smoothing, and the option to output maps of age, mass, and metallicity. These synthetic observations can be used to directly compare the simulated objectives to reality for modelling galaxy evolution, studying different theoretical models and investigating different observation effects.

Key words: cosmological simulations – galaxy evolution – mock observations

1 INTRODUCTION

Cosmological simulations provide key insights to enhance our understanding of astrophysics. By solving the equations of gravity and hydrodynamics within a discretized box, they can be used to trace the distribution of matter and the properties of astrophysical systems from early times to the modern era. These simulations have greatly evolved since they were first introduced, with many different codes and formats being available.

A widely used set of simulation codes is the GADGET family. First introduced with GADGET-1 (Springel et al. 2001), this code computes gravitational forces with a hierarchical tree algorithm and represents fluids using smoothed-particle hydrodynamics (SPH). Since its introduction, many improvements have been made to the original code, with the most recent being the GADGET-4 format (Springel et al. 2021). In addition to the mainline versions, others have written modifications to these codes while maintaining a similar format, such as GADGET-MUSIC or GADGET-X (Cui et al. 2018). Another popular format is GIZMO (Hopkins 2015), which builds off GADGET and introduces new, flexible methods for solving fluid equations.

As these codes track the evolution of a system, the data can be saved in discrete timesteps. These periods, commonly referred to as snapshots, will then contain a wealth of information regarding the physics of the system at that point in time. Different codes may have different formatting for the snapshot data, with each one potentially requiring unique handling when being read. These snapshot files will contain positions and properties of various particle types, including gas, stars, and black holes.

These files, however, may not contain direct measurements of the brightness of a given star particle. To obtain this information, one often needs to infer the spectrum based on different physical properties of the particles. The way in which the light in these particles is distributed at different wavelengths can be modelled by a spectral energy distribution (SED).

Stellar population synthesis (SPS) can be used to model the way these SEDs evolve as a function of time. However, SPS modelling is a challenging task, as many unknowns exist within the field. Different design choices must be made to handle these unknowns, including the initial mass function (IMF), the star formation history (SFH), and the treatment of stellar evolution models. Many different attempts have been made to model these processes, each with their own assumptions about the underlying physics of the system. As a result, the correct choice of model is not necessarily clear, and the ability to compare different models is extremely important.

A simple solution to this problem is proposed by EzGal (Mancone

★ E-mail: patrick.janulewicz@mail.mcgill.ca
† E-mail: weiguang.cui@uam.es

& Gonzalez 2012), which provides tools to compare magnitudes and other observables for different choices of simple stellar population (SSP) models. One such set is the commonly-used BC03 model introduced by Bruzual & Charlot (2003). The M05 model set from Maraston (2005) is also included, which thoroughly treated thermally pulsating asymptotic giant branch (TP-AGB) stars due to their importance in the infrared emission of young stellar populations. EzGal also includes the CB07 set described in Bruzual (2007), which updates the BC03 models to treat these TP-AGB stars. The α-enhanced BaSTI models by Percival et al. (2009) and the popular FSPS models (Conroy et al. 2009; Conroy & Gunn 2010b) are also included, which we henceforth refer to as P09 and C09, respectively. After its publication, the PEGASE set from Fioc & Rocca-Volmerange (1997) was also added to this list. These models can be used in combination with an initial mass function such as Salpeter (1955), Kroupa (2001), or Chabrier (2003) to model SEDs of different stellar populations. EzGal also provides a framework for constructing complex stellar population (CSP) models from these SSP models, which may involve more sophisticated star formation histories.

In addition to parameters from stellar population synthesis, the appearance of the SED will vary as a function of its age and metallicity. With cosmological simulations typically containing age and metallicity data, SPS can be leveraged to model SEDs for simulated star particles given some assumed model. By combining this with the flexible approach described above, we are able to predict SEDs in cosmological simulations for any arbitrary choice of stellar population model. Using these SEDs and some assumed filter response curve, we can calculate the magnitudes of simulated star particles as would be seen through different photometric bands. Furthermore, the positions of these particles within the simulation box are known. With magnitudes calculated and positions provided, the information can be combined in order to visualize the inside of the simulation box.

In this paper, we present PyMGal, a Python program to generate mock observations of cosmological simulations in the optical regime. PyMGal introduces a flexible way of mimicking telescope images for different choices of SPS models. These projections can be made in various different output units along any arbitrary axis, with over one hundred filters included in the package. It is compatible with both GADGET and GIZMO simulation codes. In addition to brightness, maps of mass, age, and metallicity can be generated for any axis of projection. PyMGal supports multiple features including variable output units, flexible particle smoothing, and various projection effects.

In the following sections, we will perform a variety of different tests to demonstrate the capabilities and features of the software. To avoid unnecessary repetition of the parameters of these tests, we will define a default configuration. For the simulation data, we use simulated galaxy clusters from The Three Hundred project. We select the 166th cluster from the Gadget-X run. We select the 128$^{\text{th}}$ snapshot, which corresponds to a redshift of $z = 0$. We set the centre of these projections to be the cluster's halo centre as defined by the Amiga Halo Finder (Knollmann & Knebe 2009). As a default model, we will select a BC03 model with a Chabrier IMF. We calculate brightness in the rest frame and set a default filter to be the r-band from the Sloan Digital Sky Survey (SDSS). We place the object at a distance equivalent to $z = 0.25$ for observing, and we set the smoothing length to be the distance to a particle's 100$^{\text{th}}$ nearest neighbour.

The paper is organized as follows. In Section 2, we describe the approach taken to model stellar populations and generate SEDs for simulated particles. In Section 3, we explain the methods used to calculate magnitudes in a given photometric band given a filter response curve. In Section 4, we describe the techniques used to generate projections given these particles and their computed magnitudes. We also describe optional modifiable parameters such as particle smoothing, the option to produce maps of mass, age, and metallicity, and more. Finally, in Section 5, we conclude the paper and discuss the applications of this software to modern problems.

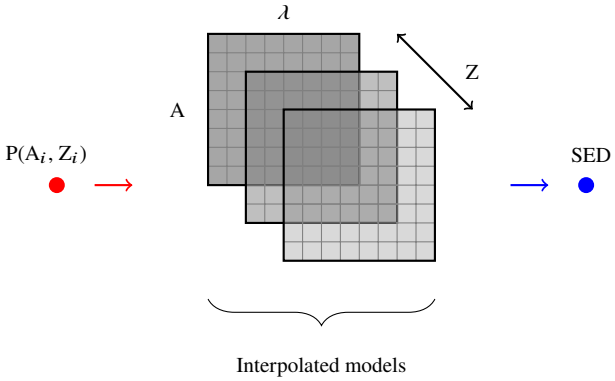

Figure 1. Visualization of how SEDs can be inferred given the age and metallicity of a particle. Each square represents a 2D array indexed by λ and age. The particle P with age A_i and metallicity Z_i is assigned to the closest metallicity value, and its SED is then inferred given its age.

2 MODELLING STELLAR POPULATIONS

2.1 Reading data and generating SEDs

The first data product required by the software is the model. Though the appearance of this distribution is dependent upon the choice of model, all share a common structure. Each one can be represented by a 2D array containing ages on one axis and wavelengths or frequencies on the other. The value at any given age-wavelength pair will describe the energy output of the object being studied. However, these SEDs vary as a function of metallicity as well. To address this, the metallicity range can be divided into different metallicity levels, with each level being assigned its own file containing its SED. The result is that for any choice of model, the SED can be described using arrays of age and wavelengths, with each metallicity having its own array.

However, particles within the simulation may have any range of such values. To resolve this, the SEDs can be interpolated, allowing the software to assign SEDs to particles given their age. To address the dependence on metallicity, different particles will be assigned a different interpolated SED depending on their metallicity. We illustrate this process in Figure 1

With the model files read and interpolated, PyMGal can then begin to assign SEDs to particles within the simulation. To do this, data must be read from the snapshot file, which provides the coordinates and physical properties of particles. The program reads a centre and radius representing a spherical region of the box. A sphere is then computed given this information, and all star particles within the it are then handled by the program. Any particles falling outside this region will not be considered. Evidently, the age and metallicities of the particles are of particular interest for the task of assigning SEDs. PyMGal also reads the cosmology of the simulation, which is required to relate values including age, redshift, and distance. Given

Category	IMF	CSP	Metallicities
P09	Kroupa		3
BC03	Chabrier Salpeter	Constant Exponential Burst	3
C09	Chabrier Salpeter Kroupa	Exponential	6
CB07	Chabrier Salpeter	Constant Exponential Burst	3
M05	Kroupa Salpeter	Exponential	1
P2	Salpeter		5

Table 1. Available models. The first column indicates the work from which the model was obtained. The second column indicates the list of available initial mass functions for the given model, while the third shows the complex stellar population models that are also available in addition to the simple stellar population models.

this information, particle SEDs can be inferred, and the program can move on to the next step.

2.2 Model selection

PyMGal must be provided with a model on which it bases its SEDs. The software supports different formats for these model files.

To begin, the software is compatible with all EzGal SSP and CSP models. It comes pre-installed with all these files, though more can easily be added if needed. The SSP model files are defined by the model type, the IMF, and the metallicity. These model types include BC03, M05, CB07, P09, C09, and P2. Choices of IMFs include Salpeter, Kroupa, and Chabrier, though not all models support all IMFs. Each set will have its own list of available metallicities, which will vary according to the model. Each of model file will contain the SEDs evolution at various ages. In addition to these SSP models, complex stellar population (CSP) from EzGal are also supported. While the SSP models feature a delta-burst model, the CSP models may feature exponentially decaying burst of star formation, a short burst of constant star formation, or a constant star formation.

Among the EzGal models, another compatible file type is the binary ised format in which the BC03 and CB07 are distributed.

To add maximum flexibility, PyMGal can also read files in the ASCII format. In theory, this format allows the user to implement any arbitrary model with any arbitrary set of parameters. The user may define such a custom file and add it in the relevant directory. This file, or set of files, will then be read by the program and used in interpolation. A full list of models can be found in Table 1.

2.3 Dust functions

The effect of dust attenuation can optionally be included when calculating SEDs, provided it has not already been taken into account. Though some stellar population synthesis software, such as recent versions of FSPS, provides the option to apply dust attenuation immediately upon calculating the SEDs, this may not be the case every time. As a result, we provide simple two dust models to add the effect of dust attenuation.

A typical example of dust attenuation is described by Charlot & Fall (2000). This relationship can be described mathematically by $\Gamma(\lambda) = e^{-\tau(t)(\lambda/5500)^{-0.7}}$, where $\tau(t) = 1.0$ for $t \leq t_{\rm break}$ and $\tau(t) = 0.5$ for $t > t_{\rm break}$. In this expression, $t_{\rm break}$ is a cutoff time in years, often assigned a value of 10^7 years.

Another common example is the Calzetti dust attenuation function from Calzetti et al. (2000) . This curve is expressed as $\Gamma(\lambda) = 10^{0.4E_s(B-V)k'(\lambda)}$, where $k'(\lambda)$ is the starburst reddening curve and $E_s(B-V)$ is the color excess of the stellar continuum.

Users may also define custom a dust function that suits the specific needs of their research goals. While PyMGal provides built-in support for these two functions, the user may also add custom ones if required.

3 FILTERS AND MAGNITUDES

3.1 Calculating magnitudes

With the SEDs computed, the next step is to calculate magnitudes. The software uses a similar approach to EzGal when calculating magnitudes. This approach calculates magnitudes in the observer's frame as a function of redshift z and formation redshift z_f. The age can then be denoted $t(z, z_f)$, and the redshifted SED at this age can then be written $F\nu[\nu(1+z), t(z, z_f)]$. The magnitude can then be defined as the projection of the redshifted SED through a filter response curve $R(\nu)$, compared to a zero mag AB source. This process is illustrated in Equation 1.

$$M_{\rm AB}[z, t(z, z_f)] = \qquad (1)$$
$$-2.5\log\left[\frac{\int_{-\infty}^{\infty}\nu^{-1}(z+1)F_\nu[\nu(1+z), t(z, z_f)]R(\nu)d\nu}{\int_{-\infty}^{\infty}\nu^{-1}R(\nu)d\nu}\right] - 48.60$$

The rest-frame magnitude uses the same approach, but calculates the magnitude as $M_{\rm AB}[0, t(z, z_f)]$.

PyMGal supports conversions to other units and magnitude systems. It can convert to Vega or solar magnitude, as well as different units of flux, flux density, and luminosity. More details on output units and conversions can be found in Section 3.3.

We provide a demonstration of computed magnitude in Figure 2 for various choices of models. To keep all else constant, we fix the metallicity and IMF for each model. We choose a solar metallicity and a Salpeter IMF, and we calculate the luminosity of all particles satisfying stasifying $Z = Z_\odot$ given a reasonable error threshold of roughly 5% or $\delta Z = 0.001$. We linearly interpolate these values and plot the result as a function of particle age. While the luminosities show slight variations for each model, all are within reasonable agreement. We also find the luminosities to be reasonable given the mass resolution of the simulation.

3.2 Set of filters

The program contains many filters across different systems, instruments, and surveys. The transmission through the filter is defined by a filter response curve $R(\nu)$ or $R(\lambda)$, where ν and λ are the frequency and wavelength, respectively. This response curve is read and is used to calculate brightness as using Equation (1).

Many of these are obtained from the Flexible Stellar Population Synthesis (FSPS) software by Conroy & Gunn (2010a). These include the Johnson-Cousins system described in (Bessell 1990), the Bessell filters from (Bessell & Brett 1988), and the Strömgren filters from (Bessell 2011). It also includes the Steidel set from (Steidel et al. 2003), Buser, and the idealized bandpass from Dickinson.

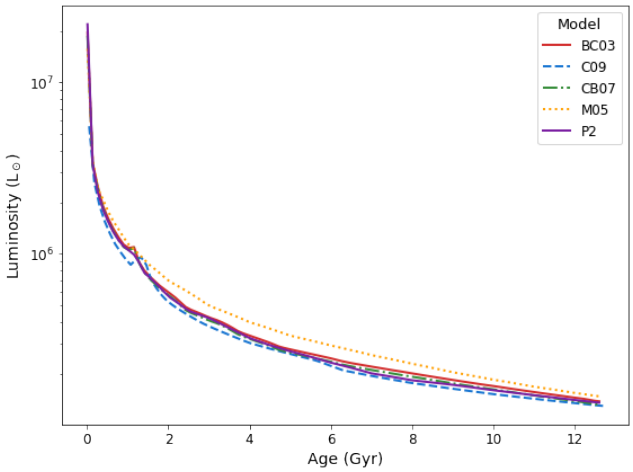

Figure 2. Calculated luminosities of galaxy cluster particles for different model choices. All models assume a Salpeter IMF with a solar metallicity. Simulated star particles are selected if they have metallicities within 0.001 of the solar benchmark. All models roughly agree, with luminosities being sensible for the mass resolution of the simulation.

Filters from various telescopes and surveys are also included. This includes the Two Micron All-Sky Survey (2MASS), the Sloan Digital Sky Survey (SDSS), the Galaxy Evolution Explorer (GALEX) telescope, the Dark Energy Survey (DES), the Wide Field Camera (WFCAM) from the United Kingdom Infrared Telescope (UKIRT), the Wide-field Infrared Survey Explorer (WISE), the Swift Ultraviolet and Optical Telescope (UVOT), the Submillimetre Common-User Bolometer Array (SCUBA), and the Infrared Astronomical Satellite (IRAS). It also contains the National Optical Astronomy Observatory's Extremely Wide Field Infrared Imager (NEWFIRM), the Visible and Infrared Survey Telescope for Astronomy (VISTA), the Subaru telescope, the Panoramic Survey Telescope and Rapid Response System 1 (Pan-STARRS1), the Vera C. Rubin Observatory's Large Synoptic Survey Telescope (LSST), as well as the Euclid and Roman space telescopes.

Some of these telescopes or surveys contain multiple instruments. The Hubble Space Telescope (HST) is an example of this, as it features the Wide Field and Planetary Camera 2 (WFPC2), the Advanced Camera for Surveys (ACS), the Near Infrared Camera and Multi-Object Spectrometer (NICMOS), and the Wide Field Camera 3 (WFC3), which contains channels for ultraviolet and visible light (UVIS) as well as infrared (IR). This list also encompasses: the Near Infrared Camera (NIRCam) from the James Webb Space Telescope (JWST); the Infrared Array Camera (IRAC) and Multiband Imaging Photometer (MIPS) from Spitzer; the Infrared Spectrometer And Array Camera (ISAAC) and FOcal Reducer and low dispersion Spectrograph (FORS) from the Very Large Telescope (VLT); the Photoconductor Array Camera and Spectrometer (PACS) and the Spectral and Photometric Imaging Receiver (SPIRE) from the Herschel Space Observatory; and the Canada-France-Hawaii 12K camera (CFH12K) and MegaCam from the Canada-France-Hawaii Telescope (CFHT).

In addition to those provided by FSPS, we also add filters from the Chinese Space Station Telescope (CSST), and and bands from the Beijing-Arizona Sky Survey (BASS) and the Mayall z-band Legacy Survey (MzLS), which are used in the Dark Energy Spectroscopic Instrument (DESI) survey.

The full set of filters can be found in Table A1, with an updated version being maintained at the documentation website. In addition to the currently existing set, the user may add any custom filter as well.

Category	Units	Symbol
Luminosity	erg s^{-1}	-
	$L_\odot$	-
Flux/Flux Density	erg s^{-1} cm^{-2}	F
	erg s^{-1} cm^{-2} Hz^{-1}	F_ν
	erg s^{-1} cm^{-2} $Å^{-1}$	F_λ
	Jansky	Jy
Magnitude	AB	-
	Vega	-
	Solar	-

Table 2. Available output units. Note that magnitudes in any system can be set to either absolute or apparent.

3.3 Output units

PyMGal provides the option to convert between various output units to facilitate the comparison with observed data. We divide this into three main classes: luminosity, flux or flux density, and magnitude. A summary of available units can be found in Table 2.

The first option is luminosity. This may be useful if the user wishes to study the intrinsic energy output of the object, with no dependence on the distance at which the projection is held. This can provide a direct measurement of the total energy output of stars within the simulation. Typical units in this category are erg s^{-1} or solar luminosities.

The second category is flux or flux density. We denote the flux by the symbol F. Since flux is the luminosity per unit area, it is related to luminosity given $F = \frac{L}{4\pi d^2}$.

An alternative to flux is the spectral flux density, which can measure either the flux per unit wavelength of the flux per unit frequency. We denote the former F_λ and the latter F_ν. Another common unit for flux density is the jansky, typically denoted Jy. The jansky is equivalent to F_ν within a multiplicative factor of 10^{-23}. F_ν and F_λ at a given wavelength are related by $F_\lambda = \frac{c}{\lambda^2} F_\nu$.

The final category is the magnitude system. These magnitude systems are logarithmically scaled and defined by some choice of reference brightness. The brightness of an object is then compared with that of the reference and can either be apparent or absolute. The absolute magnitude can be obtained from the formula $m = -2.5 \log \frac{F}{F_{\text{ref}}}$.

A magnitude system can then be expressed in either absolute or apparent terms. The absolute magnitude is defined as the brightness that an object would have if it were placed 10 kpc away from the observer. By doing so, it eliminates the effect of distance when calculating magnitudes. For apparent, the distance of the object affects its magnitude. The apparent magnitude can be obtained by calculating the absolute magnitude and adding a distance term.

$$m_{\text{apparent}} = m_{\text{absolute}} + 5 \log \left(\frac{d}{10 \text{ kpc}} \right)^2 \quad (2)$$

We include three common magnitude systems. The default is AB magnitude, which uses a reference point of 3631 Jy that is independent of filter choice. The Vega magnitude system is also supported. In this system, the zero-point is equal to the brightness of the star Vega

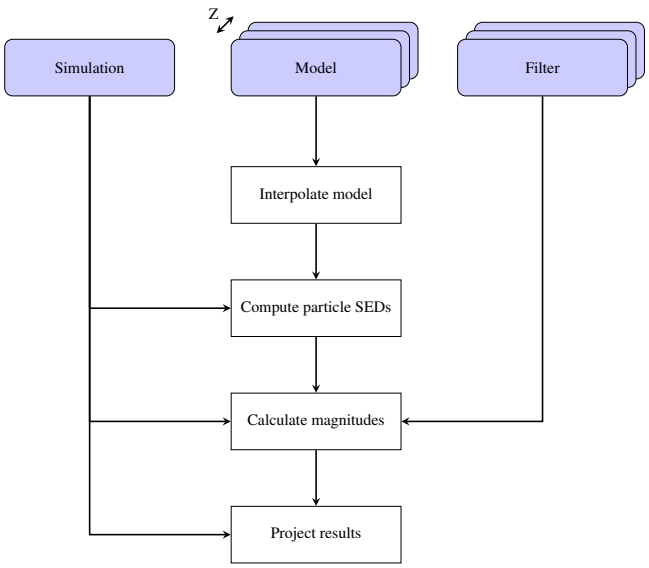

Figure 3. A flowchart demonstrating the way PyMGal processes its input. This example shows the process for models containing multiple metallicities with magnitudes being calculated over several filters. However, the program can also be run with a single model file and a single filter.

in the given filter. The third is the solar magnitude system, which uses the magnitude of the Sun in a given filter to serve as a reference.

Once the magnitudes have been calculated, the final step is to project the particles to two dimensions. Though we discuss the projection process in greater detail, we close this section by summarizing the way in which data is read and handled by the program. A flowchart demonstrating these processes can be seen in Figure 3.

4 PROJECTIONS

4.1 Projecting along an axis

Once the brightness of particles in the appropriate units are known, mock observations can be generated by projecting the three-dimensional region to two dimensions. To do this, an axis of projection must be specified. The principal axes x, y, and z are an intuitive choice that may be selected. Additionally, PyMGal supports arbitrary projection directions via a rotation matrix, allowing for the simulation data to be observed through any possible angle.

With an axis specified, the positions of particles are projected to two dimensions. Mock observations are produced by binning these particle brightness values to a 2D histogram.

The centre of this histogram will coincide with the centre of the projection region used to extract star particles. The number of bins in the histogram, which is equivalent to the number of pixels in the output image, can be specified by the user. The angular resolution can also be specified by the user. Conversion between coordinates in simulation space and pixel space are directly connected to the angular resolution and the pixel dimensions. Furthermore, the distance between the object and the hypothetical observer will also affect the binning process, as it will affect the conversion between angular and physical sizes. Further details on the pixel size and the angular resolution of the image can be found in Section 4.2, while details on the distance between the observer and the object can be found in Section 4.6.

Each particle is assigned to a bin in the histogram given its coordinates from the simulation file. The brightness of the particle is then added to the histogram bin, and this process is repeated for all particles.

Should the user opt for smoothing, the particles will be replaced with a 2D Gaussian kernel. This method spreads the light in a single pixel to its neighbouring pixels. By using a Gaussian kernel, the image can be smoothed while also preserving the total flux. Optionally, images of the mass, age, and metallicity for a given projection can also be provided. In addition to these options, several other parameters be modified to change the resulting mock observation. We include a more detailed discussion of these parameters in the remainder of this section.

We provide a first example of these projections in Figure 4. We demonstrate a projection of the simulated data along the three principal axes and using filter response curves from a multitude of different projects and instruments. To model the brightness of the simulation across multiple wavelengths, we select filters ranging from the ultraviolet to the infrared. This decreasing frequency trends can be seen from left to right in the figure. Furthermore, to display the flexibility of the software to mimic different surveys, each frequency uses a filter from a different system, survey, or instrument. From left to right, we show the u-band from the Vera C. Rubin Observatory, the g-band from the Sloan Digital Sky Survey, the V-band from the Johnson system, the VIS band from Euclid, the F110W band from Hubble Space Telescope's Wide Field Camera 3, and the F444W band from the James Webb Space Telescope's Near Infrared Camera.

4.2 Dimensions and angular resolution

The user may select a custom angular resolution and pixel dimension for the output image. The angular resolution defines the smallest angular separation between two objects that can be distinguished in the resulting mock observation. A finer angular resolution results in a finer grid of pixels when binning the particles to a 2D histogram. The angular resolution may be set by the user to match that of the instrument they wish to mimic.

Another parameter is the number of pixels in the image. For a fixed angular resolution increasing the number of pixels will increase the field of view of the observation. As a result, the angular size of the image will be dependent on the angular resolution and the number of pixels. In addition to this, the distance of the simulated region to the observer described in Section 4.6 will also play a role.

4.3 Smoothing

PyMGal performs Gaussian smoothing on particles using a k-nearest neighbours (kNN) approach. For each particle, the distance to its kth nearest neighbour in the simulation box is calculated and is converted to a pixel value. This distance is then used as 1σ for a 2D Gaussian distribution. The brightness of the pixel is then multiplied by the Gaussian kernel. This ensures that the total flux is conserved, as the sum of all pixel values will be equal to the initial brightness.

The user may define the value of k, which will affect how smoothed the particles become. The user may also opt to forgo smoothing altogether. In this case, the particles are binned to 2D projections and are left as individual pixel values.

For mass, metallicity, and age maps, it may not make physical sense to spread the particles as though they are light, since these properties do not travel in the same way as light. In this case, we add another parameter for the user to modify. The maximum size of

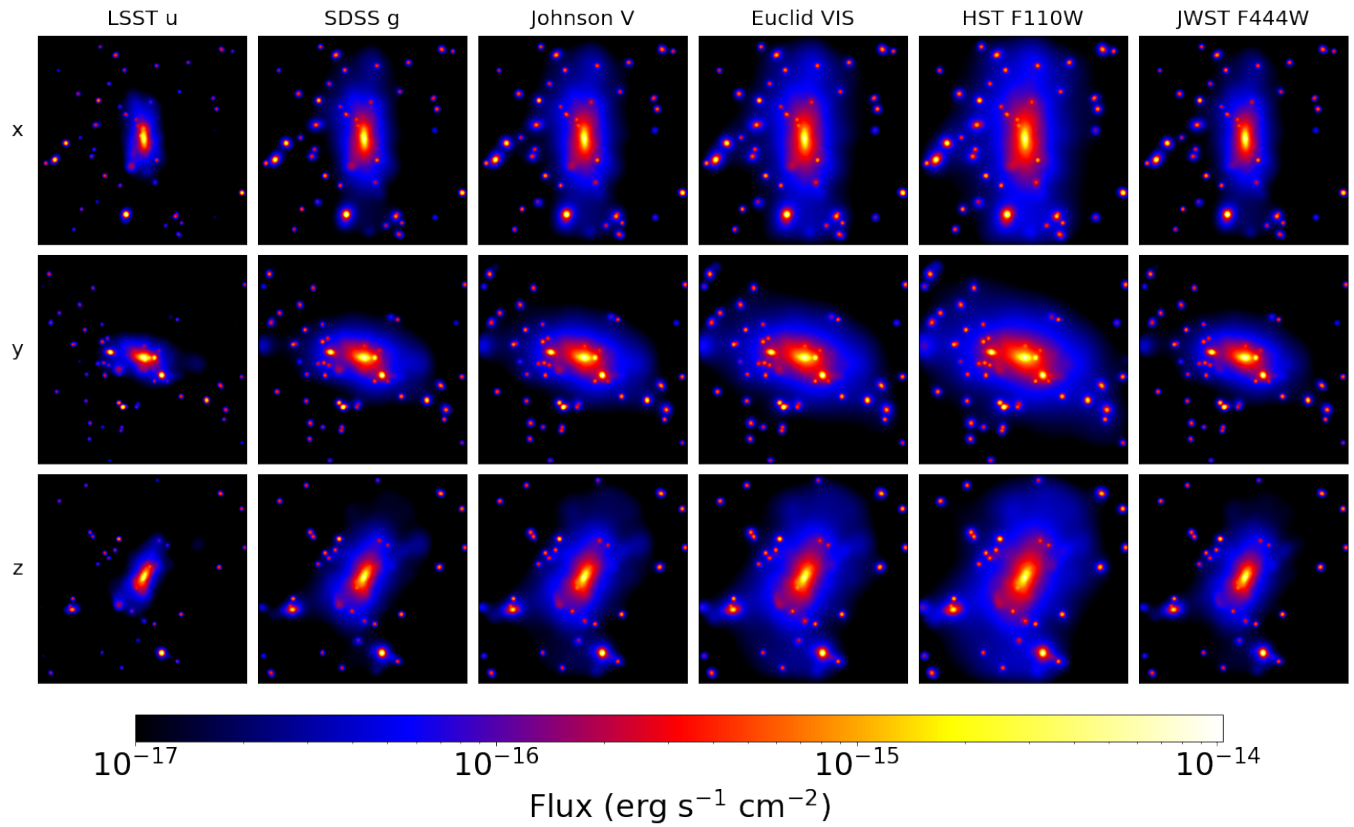

Figure 4. A sample image of a galaxy cluster along different axes and different filters. The three rows show projections along the three principal axes, while the columns indicate different filters from different surveys and instruments. From left to right, the pivot wavelength of the filter becomes larger.

kernels for these can be limited by the gravitational softening length. The gravitational softening length, chosen by the user to be a distance in the simulation box, is converted to a pixel value during projection. This pixel value is set to be the maximum 1σ for the Gaussian kernel used to spread mass, age, and metallicity.

4.4 Mapping mass, age, and metallicity

In addition to projections of luminosity, magnitdues, or flux through a given filter, PyMGal also provides the option to output the maps mass, age, and metallicity. This is an optional output that is calculated after the position of the particles have been calculated in a 2D projection. In other words, each unique projection has a corresponding map for mass, age, and metallicity.

To calculate the mass inside a pixel, the mass of all particles within the pixel are added together. In the case of age and metallicity maps, values are mass-weighted. Within a pixel, the age or metallicity of a particle is multiplied by its mass. All of these values are then summed together and then divided my the total mass. Mathematically, we can describe this as follows, where N is the number of particles assigned to a pixel.

$$m_{\rm px} = \sum_{i=1}^{N} m_i, \quad A_{\rm px} = \frac{\sum_{i=1}^{N} A_i * m_i}{\sum_{i=1}^{N} m_i}, \quad Z_{\rm px} = \frac{\sum_{i=1}^{N} Z_i * m_i}{\sum_{i=1}^{N} m_i} \tag{3}$$

These maps can also be smoothed using the techniques described in Section 4.3. For mass maps, the mass pixels are replaced by Gaussian kernels and are smoothed similar to light. For age maps, the product of the age and the mass $A_i * m_i$ is computed and is then replaced by a Gaussian kernel. It is then divided by the Gaussian kernel of m_i, which has the same size as its numerator. An analogous process is used for metallicity maps.

The user can define a gravitational softening length to limit the smoothing on these particles. For example, setting a gravitational softening length equivalent to the size of a single pixel will ensure that no Gaussian kernel has 1σ greater than one pixel. An example of these maps and their smoothing can be seen in Figure 5.

4.5 Generating spectra

The spectra of the observations can also be output. In this case, rather than outputting a mock observation through a single filter, the projection can be seen at arbitrary wavelength.

4.6 Observed redshift

The observer's redshift is an useful parameter that can be modified to move the observation nearer or further away. It is important to note that this does not affect any evolutionary traits of the simulation. Rather, it affects the distance of the projection from the frame. If z_{obs} is specified by the user, the projection will be placed at this distance. If it is not, the distance will be set by the redshift of the simulation file, unless the simulation redshift is near $z = 0$, in which case a minimum distance is set. The user may adjust this value to customize the apparent distance of the observation.

We demonstrate this effect in the columns of Figure 6. We show

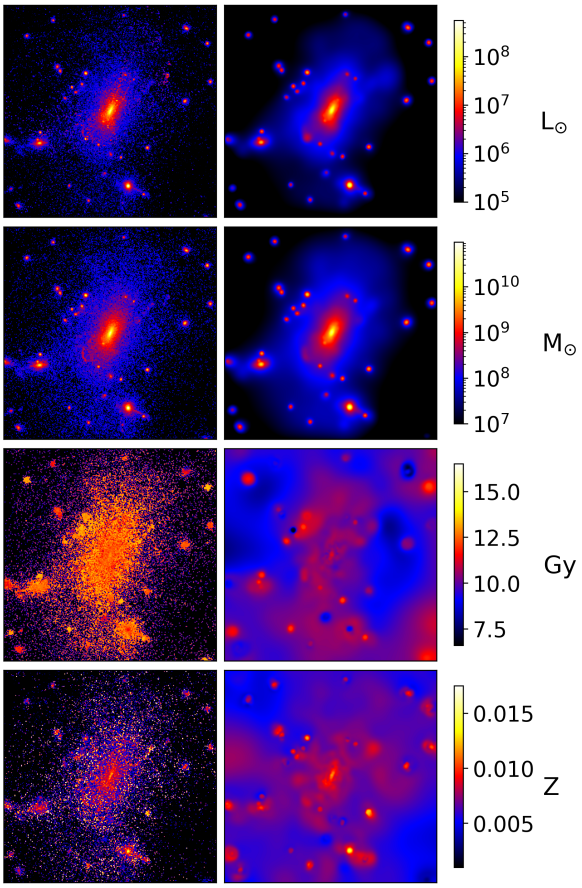

Figure 5. A sample projection of luminosity, mass, age, and metallicity. The left column shows an image without smoothing, and the right column contains the same image smoothed with a length equal to the distance between a particle and its 100$^{\text{th}}$ nearest neighbour. The first row shows the luminosity in units of $L_\odot$, while the second row shows the mass in $M_\odot$. The third and fourth row show the age and metallicity, respectively. Restrictions on the smoothing length are not applied to the mass, age, and metallicity maps.

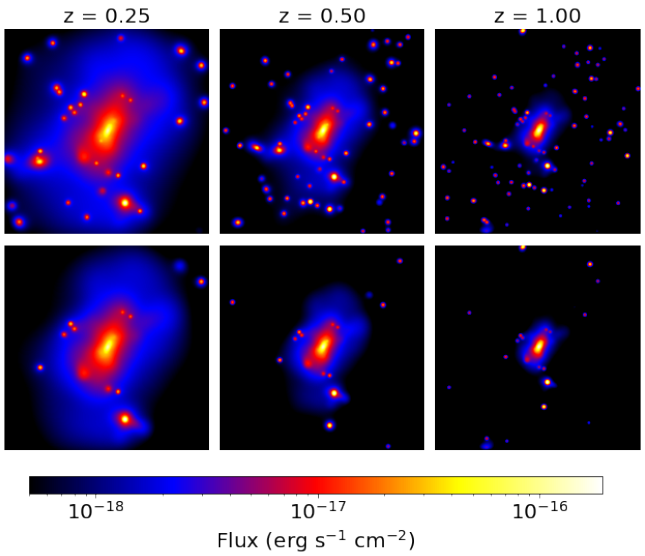

Figure 6. A demonstration of selecting various redshift values for observation. In the top row, the entire thickness of the region is selected. In the bottom row, a cut of 100 kpc is taken in the projection direction.

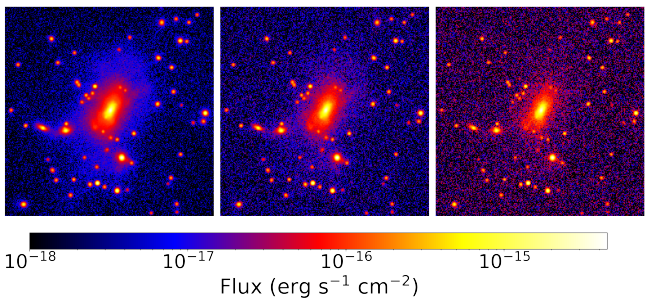

Figure 7. A demonstration of different noise levels added to the image. From left to right, noise is increased by one AB magnitude per square arcsecond.

a projection of the same simulation region in units of flux for three different redshift values. The rows of this figures display different choices of thicknesses along the axis of projection. We will describe this effect further in Section 4.7.

4.7 Thickness

The user can optionally elect to project only a slice of the data along the projection axis. By default, are particles in the specified projection region are shown in the resulting maps. However, the thickness in the projection direction can be selected to limit this. If a user selects a thickness, only particles within [centre−thickness, centre−thickness] along the projection axis will be selected. All particles within this range will be included, while all particles beyond this range will be omitted.

This feature allows the user to analyze the 3D structure of the simulation. By omitting certain regions, the position of the particles in 3D space can better be inferred.

This effect is illustrated in the rows of Figure 6. In the top row, we project the entire simulated region for different redshift values as described in Section 4.6, with no restriction placed on thickness. In the bottom row, we restrict the thickness to be 100 kpc. While the two settings produce similar images, we find that some particles are omitted when the thickness is limited.

4.8 Adding noise

Gaussian noise can be added to the observations. The noise value is provided by the user as a surface density. This noise can be adjusted with a continuous scale, offering the flexibility to generate different signal-to-noise ratios (SNR) depending on the observational scenario being modelled.

We show this effect in Figure 7, where we show projections for three gradually increasing noise levels. In each case, we amplify the noise level by the equivalent of one AB magnitude per squared arcsecond. As expected, the signal-to-noise ratio visibly decreases as the level of Gaussian noise is increased.

5 CONCLUSIONS

PyMGal provides a quick, simple, and flexible way of generating mock observations from cosmological simulations. The software can

be broadly split into three main parts, which each correspond to a section in this paper. The first part, highlighted in Section 2, involves modelling stellar populations and leveraging this to generate SEDs. The software infers the SEDs of star particles in simulation snapshot files by reading a user-defined SPS model and fitting particle properties to the interpolated model. The second part, shown in Section 3, describes the way PyMGal calculates magnitudes using a projection of the SEDs through a filter response curve and converts to the appropriate units. Finally, once these magnitudes have been computed, mock observations can be created by projecting the data two two diemensions as described in Section 4. This can be done by mapping the physical positions of the particles to a pixel position in a 2D image.

The package supports both GAGDET and GIZMO snapshot formats, and it supports high flexibility for the choice of SPS model and filter response curves. In addition to observations of light, maps of the mass, age, and metallicity can also be output. The mock observations can be modified in a variety of ways, from variable smoothing to an adjustable distance between the object and the observer.

In summary, PyMGal is a versatile tool that bridges cosmological simulations with observational data, allowing researchers to create optical mock observations of galaxies with ease. Its adaptability to different simulation formats, output units, SPS models, and filter response curves makes it a valuable resource for a wide range of astrophysical research applications.

ACKNOWLEDGEMENTS

We acknowledge access to the theoretically modelled galaxy cluster data via The Three Hundred[1] collaboration. The simulations used in this paper have been performed in the MareNostrum Supercomputer at the Barcelona Supercomputing Center, thanks to CPU time granted by the Red Española de Supercomputación. As part of The Three Hundred project, this work has received financial support from the European Union's Horizon 2020 Research and Innovation programme under the Marie Skłodowska-Curie grant agreement number 734374, the LACEGAL project.

DATA AVAILABILITY

PyMGal is registered with the Python Package Index at https://pypi.org/project/pymgal/, which contains all the relevant information, documentation, and links, with documentation available at https://pymgal.readthedocs.io.

The simulations used in this work have been provided by The Three Hundred collaboration. The data may be shared upon request to the collaboration.

REFERENCES

Bessell, M. S., 1990. UBVRI passbands., *PASP*, **102**, 1181–1199.
Bessell, M. S., 2011. Photonic Passbands and Zero Points for the Strömgren uvby System, *PASP*, **123**(910), 1442.
Bessell, M. S. & Brett, J. M., 1988. JHKLM Photometry: Standard Systems, Passbands, and Intrinsic Colors, *PASP*, **100**, 1134.
Bruzual, A. G., 2007. On TP-AGB stars and the mass of galaxies, in *Stellar Populations as Building Blocks of Galaxies*, vol. 241 of **IAU Symposium**, pp. 125–132.
Bruzual, G. & Charlot, S., 2003. Stellar population synthesis at the resolution of 2003, *MNRAS*, **344**(4), 1000–1028.
Calzetti, D., Armus, L., Bohlin, R. C., Kinney, A. L., Koornneef, J., & Storchi-Bergmann, T., 2000. The Dust Content and Opacity of Actively Star-forming Galaxies, *ApJ*, **533**(2), 682–695.
Chabrier, G., 2003. Galactic Stellar and Substellar Initial Mass Function, *PASP*, **115**(809), 763–795.
Charlot, S. & Fall, S. M., 2000. A Simple Model for the Absorption of Starlight by Dust in Galaxies, *ApJ*, **539**(2), 718–731.
Conroy, C. & Gunn, J. E., 2010a. FSPS: Flexible Stellar Population Synthesis, Astrophysics Source Code Library, record ascl:1010.043.
Conroy, C. & Gunn, J. E., 2010b. The Propagation of Uncertainties in Stellar Population Synthesis Modeling. III. Model Calibration, Comparison, and Evaluation, *ApJ*, **712**(2), 833–857.
Conroy, C., Gunn, J. E., & White, M., 2009. The Propagation of Uncertainties in Stellar Population Synthesis Modeling. I. The Relevance of Uncertain Aspects of Stellar Evolution and the Initial Mass Function to the Derived Physical Properties of Galaxies, *ApJ*, **699**(1), 486–506.
Cui, W., Knebe, A., Yepes, G., Pearce, F., Power, C., Dave, R., Arth, A., Borgani, S., Dolag, K., Elahi, P., Mostoghiu, R., Murante, G., Rasia, E., Stoppacher, D., Vega-Ferrero, J., Wang, Y., Yang, X., Benson, A., Cora, S. A., Croton, D. J., Sinha, M., Stevens, A. R. H., Vega-Martínez, C. A., Arthur, J., Baldi, A. S., Cañas, R., Cialone, G., Cunnama, D., De Petris, M., Durando, G., Ettori, S., Gottlöber, S., Nuza, S. E., Old, L. J., Pilipenko, S., Sorce, J. G., & Welker, C., 2018. The Three Hundred project: a large catalogue of theoretically modelled galaxy clusters for cosmological and astrophysical applications, *MNRAS*, **480**(3), 2898–2915.
Fioc, M. & Rocca-Volmerange, B., 1997. PEGASE: a UV to NIR spectral evolution model of galaxies. Application to the calibration of bright galaxy counts., *A&A*, **326**, 950–962.
Hopkins, P. F., 2015. A new class of accurate, mesh-free hydrodynamic simulation methods, *Monthly Notices of the Royal Astronomical Society*, **450**(1), 53–110.
Knollmann, S. R. & Knebe, A., 2009. Ahf: Amiga's halo finder, *The Astrophysical Journal Supplement Series*, **182**(2), 608–624.
Kroupa, P., 2001. On the variation of the initial mass function, *MNRAS*, **322**(2), 231–246.
Mancone, C. L. & Gonzalez, A. H., 2012. EzGal: A Flexible Interface for Stellar Population Synthesis Models, *PASP*, **124**(916), 606.
Maraston, C., 2005. Evolutionary population synthesis: models, analysis of the ingredients and application to high-z galaxies, *MNRAS*, **362**(3), 799–825.
Percival, S. M., Salaris, M., Cassisi, S., & Pietrinferni, A., 2009. A Large Stellar Evolution Database for Population Synthesis Studies. IV. Integrated Properties and Spectra, *ApJ*, **690**(1), 427–439.
Salpeter, E. E., 1955. The Luminosity Function and Stellar Evolution., *ApJ*, **121**, 161.
Springel, V., Yoshida, N., & White, S. D. M., 2001. GADGET: a code for collisionless and gasdynamical cosmological simulations, *New Astron.*, **6**(2), 79–117.
Springel, V., Pakmor, R., Zier, O., & Reinecke, M., 2021. Simulating cosmic structure formation with the gadget-4 code, *Monthly Notices of the Royal Astronomical Society*, **506**(2), 2871–2949.
Steidel, C. C., Adelberger, K. L., Shapley, A. E., Pettini, M., Dickinson, M., & Giavalisco, M., 2003. Lyman Break Galaxies at Redshift z ~3: Survey Description and Full Data Set, *ApJ*, **592**(2), 728–754.

APPENDIX A: TABLE OF AVAILABLE FILTERS

We include a table of all filters included in the package upon download. An up-to-date version of this list can be found at the documentation website. The user may also add custom filters if the desired one is not present.

[1] https://www.the300-project.org

This paper has been typeset from a TEX/LATEX file prepared by the author.

System	Instrument	Filters
Johnson-Cousins		U, V, B, R, I
Buser		B2
Bessell		L, LP, M
Strömgren		u, v, b, y
Steidel		U_n, G, Rs, I
Idealized bandpass		1500, 2300, 2800
2MASS		J, H, Ks
SDSS		u, g, r, i, z
GALEX		FUV, NUV
DES		g, r, i, z, Y
UKIRT	WFCAM	Z, Y, J, H, K
WISE		W1, W2, W3, W4
Swift UVOT		W2, M2, W1
SCUBA		450WB, 850WB
IRAS		12, 25, 60, 100
NEWFIRM		J1, J2, J3, H1, H2, K
VISTA	VIRCAM	Y, J, H, K
Subaru	Suprime-Cam	B, g, V, r, i, z
Pan-STARRS1		g, r, i, z, y
LSST		u, g, r, i, z, y
Euclid		VIS, Y, J, H
Roman		F062, F087, F106, F129, F158, F184
HST	WFPC2	F255W, F300W, F336W, F439W, F450W, F555W, F606W, F814W, F850LP
	ACS	F435W, F475W, F555W, F606W, F625W, F775W, F814W, F814W, F850LP
	WFC3 UVIS	F218W, F225W, F275W, F336W, F390W, F438W, F475W, F555W, F606W, F775W, F814W, F850LP
	WFC3 IR	F098M, F105M, F110M, F125M, F140M, F160M
	NICMOS	F110W, F160W
JWST	NIRCam	F070W, F090W, F115W, F150W, F200W, F277W, F356W, F444W
Spitzer	IRAC	1, 2, 3, 4
	MIPS	24, 70, 160
VLT	ISAAC	Ks
	FORS	V, R
CFHT	CFH12K	B, R, I
	MegaCam	u, g, r, i, z
Herschel	PACS	70, 100, 160
	SPIRE	250, 350, 500
DESI	BASS	g, r
	MzLS	z
CSST		nuv, u, g, r, i, z, y

Table A1. Available filters. An updated version is maintained at the documentation website https://pymgal.readthedocs.io.