Publications of the Astronomical Society of Australia (2023), **40**, e048, 20 pages
doi:10.1017/pasa.2023.47

CAMBRIDGE UNIVERSITY PRESS

Research Article

SimSpin v2.6.0—constructing synthetic spectral IFU cubes for comparison with observational surveys

K. E. Harborne[1,2], A. Serene[1], E. J. A. Davies[3,4], C. Derkenne[2,5], S. Vaughan[2,5,6], A. I. Burdon[5], C. del P Lagos[1,2], R. McDermid[2,5], S. O'Toole[2,3,4], C. Power[1,2], A. S. G. Robotham[1,2], G. Santucci[1,2] and R. Tobar[1]

[1]International Centre for Radio Astronomy (ICRAR), M468, The University of Western Australia, Crawley, WA, Australia, [2]ARC Centre of Excellence for All Sky Astrophysics in 3 Dimensions (ASTRO 3D), [3]Australian Astronomical Optics, Macquarie University, Sydney, NSW, Australia, [4]Astrophysics and Space Technologies Research Centre, Macquarie University, Sydney, NSW, Australia, [5]Research Centre for Astronomy, Astrophysics, and Astrophotonics, School of Mathematical and Physical Sciences, Macquarie University, Sydney, NSW, Australia and [6]Centre for Astrophysics and Supercomputing, School of Science, Swinburne University of Technology, Hawthorn, VIC, Australia

Abstract

In this work, we present a methodology and a corresponding code-base for constructing mock integral field spectrograph (IFS) observations of simulated galaxies in a consistent and reproducible way. Such methods are necessary to improve the collaboration and comparison of observation and theory results, and accelerate our understanding of how the kinematics of galaxies evolve over time. This code, SIMSPIN, is an open-source package written in R, but also with an API interface such that the code can be interacted with in any coding language. Documentation and individual examples can be found at the open-source website connected to the online repository. SIMSPIN is already being utilised by international IFS collaborations, including SAMI and MAGPI, for generating comparable data sets from a diverse suite of cosmological hydrodynamical simulations.

Keywords: Virtual observatory tools; galaxies: evolution; galaxy: kinematics; methods: numerical

Abbreviations: IFS; integral field spectroscopy; SSP; simple stellar populations

(Received 6 July 2023; revised 14 August 2023; accepted 30 August 2023)

1. Introduction

Astronomy is divided. Observers are collecting increasingly exquisite data using telescopes focused on the Universe around us. Theorists, meanwhile, are attempting to explain and predict the observable Universe from first principles using fundamental physics and progressively more complex computational models. The discussion between these parties is most commonly separated by paper preparation and publication cadence, while further data is collected and new simulations' features are implemented and tested.

To accelerate the conversation between these parties, and our understanding of galaxy evolution as a result, it is imperative that like-for-like comparisons between observational data and theory results are easy to produce in a consistent and reproducible manner. This is particularly important given ongoing advances in both observational and theoretical astrophysics.

We have seen a revolution in spatially resolved kinematic studies of stars and gas with the development of the integral field spectrograph (IFS). Based on the principles developed for the TIGER and OASIS instruments (Bacon et al. 1995; Bacon & Monnet 2017), which used lens-let arrays to collect spectra in a grid across the surface of galactic nuclei, further instruments such as SAURON (Bacon et al. 2001) paved the way for studying the stellar motions of entire galaxy structures. Following the final data releases of SAMI (Croomet al. 2021) and MaNGA (Bundy et al. 2015; Abdurro'uf et al. 2022), instruments with multi-object apertures that allow the collection of many galaxies during a single observation, astronomers now have access to spatially resolved, kinematic observations of over 10000 galaxies. These products give us the required statistics to examine the kinematic variety within the nearby Universe at a scale only imagined at the turn of the century. Availability of such data is due only to increase in resolution and scale with the commissioning of the Hector instrument in 2022 July (Bryant et al. 2020).

Alongside these developments, only the most recent of the large-scale cosmological hydrodynamical simulations have sufficient resolution to explore individual galaxies on a case-by-case basis within a representative cosmological volume. Cosmological simulations such as EAGLE (Schaye et al. 2015; Crain et al. 2015), Magneticum Pathfinder (Teklu et al. 2015; Schulze et al. 2018), HorizonAGN(Dubois et al. 2014) and IllustrisTNG (Pillepich et al. 2018; Springel et al. 2018; Nelson et al. 2019) have baryonic particles that represent of order 10^6–10^7 solar masses such that an individual resolved galaxy can be composed of 10^3–10^5 individual stellar particles. In comparison to the early cosmological models of Metzler & Evrard (1994) and Katz, Weinberg, & Hernquist (1996), in which galaxies were represented by single particles

Corresponding author: K. E. Harborne; Email: katherine.harborne@uwa.edu.au

Cite this article: Harborne KE, Serene A, Davies EJA, Derkenne C, Vaughan S, Burdon AI, Lagos C. del P, McDermid R, O'Toole S, Power C, Robotham ASG, Santucci G and Tobar R. (2023) SimSpin v2.6.0—constructing synthetic spectral IFU cubes for comparison with observational surveys. *Publications of the Astronomical Society of Australia* **40**, e048, 1–20. https://doi.org/10.1017/pasa.2023.47

or tens of stellar particles respectively, the structural parameters of individual galaxies can now be examined in a cosmological context.

The numerical convergence, and hence the kinematics, of these galaxies will be affected by the smoothness of the underlying potential, specifically the number of dark matter particles within the simulation in question. In modern simulations, this number is generally minimised to reduce the computational cost of large volume codes which results in numerical disk heating (e.g. Ludlow et al. 2019, Ludlow et al. 2023, Wilkinson et al. 2023). Never-theless, these simulations are an important test-bed for experimental models of galaxy evolution. They enable us to uncover the key ingredients necessary for recovering observed distributions. It remains important that our comparisons between observation and simulation are made consistently such that the impact of any changes to sub-grid physics and numerical methods can be properly contextualised.

In recent years, we have seen a number of direct comparisons made between cosmological models and integral field spectroscopic observations -

- Bendo & Barnes (2000) demonstrated the first example of post-processing idealised galaxy merger simulations into projected line-of-sight (LOS) velocity and dispersion maps. These were used for direct comparison with observations made around this time using long-slit spectra, in an effort to explore the possible formation paths of different kinematic morphologies.
- The concept of utilising theoretical simulations to explore formation scenarios was further utilised by the results of the SAURON survey (Bacon et al. 2001; de Zeeuw et al. 2002; Emsellem et al. 2004). Jesseit et al. (2007, 2009) produced 2D kinematic maps with the aim of exploring the formation mechanisms driving the range of kinematic morphologies discovered by the survey, e.g. counter-rotating cores and slow rotating ellipticals. Subsequently, as part of the ATLAS3D survey (Cappellari et al. 2011), Naab et al. (2014) demonstrated the first example of comparison with cosmological simulations from Oser et al. (2010) to explore the cosmological origin of variety in kinematic morphology.
- A thorough study systematically comparing results from modern cosmological simulations and observational surveys was presented in van de Sande et al. (2019). The key purpose of this study was to demonstrate key areas of success and tension between various hydrodynamical simulations and IFS observational surveys. Although every attempt was made to ensure consistency, each simulation's data was compiled by the respective team and methodological differences exist between the samples as a result. For example, (1) the method of determining the projected ellipticity of a galaxy is done iteratively using the observational method of Cappellari et al. (2007) at 1.5 times the effective radius (R_e) for the Magneticum simulation, while EAGLE and HorizonAGN were measured using the eigenvalues of the moment-of-inertia tensor within 1 R_e. (2) Various particle-per-pixel choices are made per simulation; HorizonAGN has a lower particle limit of 10 per pixel, while Magenticum uses Voronoi bins to increase this resolution to at least 100 particles per 'pixel' (Schulze et al. 2018).

Then in Foster et al. (2021), we saw the first example of a survey incorporating comparisons with theoretical simulations from the project conception. Since this time, the number of examples have increased exponentially, with Bottrell & Hani (2022), Nanni et al. (2022) and Sarmiento et al. (2023) the most recent examples of mock observations produced for either simulation suites, or individual surveys. Other works, such as Poci et al. (2021) and Zhu et al. (2022), have used such mocks as independent tests to explore the success of Schwarzschild models in reconstructing the full orbital distributions of galaxies.

As the popularity of these comparisons increases, it is important that concrete *methods* of constructing our comparative data sets are established. Differences in constructing these data may introduce errors that carry through to later inference. It is important that methods are: (1) applicable to different simulations and telescopes, (2) that their operation is well-documented and tested, and (3) that this operation is open to extension and modification by the wider community, i.e. that the code is *open source*.

In this paper, we present an updated version of the software SIMSPIN. This code is open-source and fully documented with function descriptions and examples. SIMSPIN is designed to be agnostic to the input simulation, with various cosmological hydrodynamical simulations supported including EAGLE, MAGNETICUM Pathfinder, HORIZONAGN and ILLUSTRISTNG.

It is worth noting that, especially for open-source code, it is difficult to provide a static reference for the current capabilities of a given code-base. For that reason, this paper is just one form of reference for SIMSPIN. When using this code, we advise you visit the website www.github.io/kateharborne/SimSpin for the most recent updates and code examples. If you use this code for your research, we ask that you cite this paper, as described in the citation file contained in the repository.

Aim of this paper

The code presented in this paper is a substantial body of work, extending the capabilities of the original code presented in Harborne, Power, & Robotham (2020a). A new publication is warranted to record the new methodologies involved. In summary, new features of the code include:

- the addition of spectral data cube generation, such that mock data-products can be run through analysis pipelines in the same way as real IFS observations;
- the analysis and incorporation of gas particles, requiring smoothing techniques, within mock data-products;
- the addition of higher-order kinematic measurements in both gas and stellar mock-kinematic data cubes;
- and the incorporation of multi-threading capabilities to aid speed-up of processing large numbers of galaxies from a cosmological simulation.

In this paper, we present the new methodology behind each of these added features. For further documentation details, go to https://kateharborne.github.io/SimSpin/. This website contains a series of walk-throughs and examples, as well as the full documentation for each SIMSPIN function. Here, you will also find details about the web application and API, via which those uncomfortable with the R-package can still produce SimSpin mock observations in FITS format for processing in any language of choice. The information at these locations will continue to evolve with development time (the date at the end of each page will reflect the last time that document was modified). You can also check out the NEWS on

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

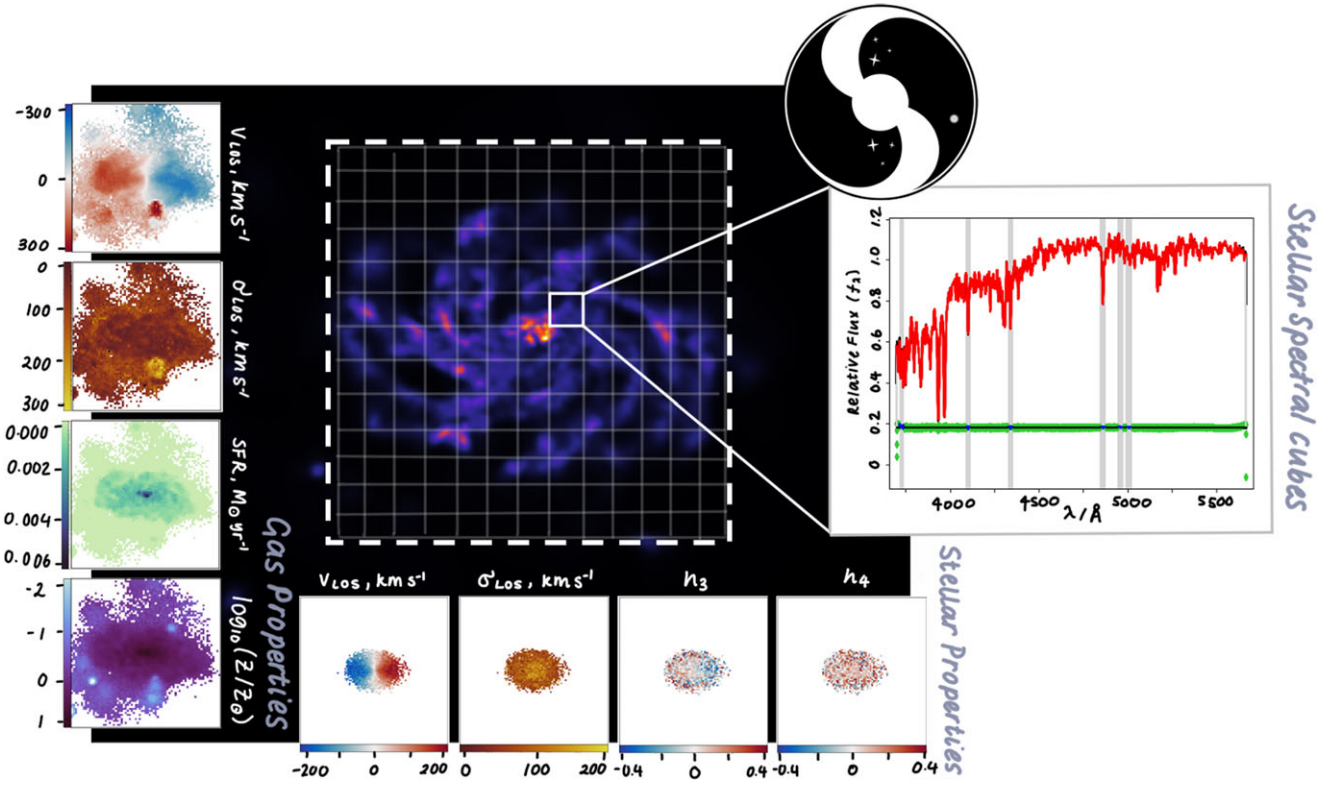

Figure 1. Demonstrating some of the possible outputs of a SimSpin observation using a MUSE-like telescope on an inclined Eagle disk galaxy from the $z = 0.271$ snapshot of the `RefL0100N1504` box. To the left, we show the possible outputs from a kinematic data cube measured for the smoothed gas component. In this scenario, the code has been run with `method = 'gas'`, meaning that all gas particles within the subhalo have been used to compute the observations shown. From top to bottom, we show the line-of-sight velocity, dispersion, star formation rate (SFR) and logged total metallicity of the gas in the model. Along the bottom, we show a similar range of possible outputs for a kinematic mode observation of the stellar component where the code has been run with `method = 'velocity'`. Here we show the kinematics as light-weighted values, while setting `mass_flag = TRUE` would result in mass-weighted kinematics being produced. From left to right, we show the line-of-sight velocity, dispersion and higher-order kinematics h_3 and h_4. On the right, we demonstrate a spectrum from a central spaxel, as fit with pPXF. An associated spectrum per pixel will be produced by the code when `method = 'spectral'`. These can be run through software tools such as pPXF to recover the underlying kinematics. The central image of Eagle GalaxyID 16382120 has been made using the code SPLASH (Price 2007). This shows the smoothed gas and stellar distribution of particles in the galaxy. The white boundary illustrates the size of the MUSE field-of-view relative to this galaxy model.

GitHub[a] to see the latest updates to the code since the publication of this paper.

As this code is continuously improving and extending to tackle new science questions, we have chosen to use traditional semantic versioning standards. This paper presents the methodology behind the code at the time of writing, with SimSpin v2.6.0. For further information about the current version of the code, please visit the website for the live documentation.

2. Methodology

The key function performed by SimSpin is the construction of a mock IFS data cube from a galaxy simulation input, as shown in Fig. 1. In this section, we describe the methodology used for constructing such an observation. The process is broken into three steps: (1) preparing the input simulation; (2) preparing the mock observation settings (i.e. telescope and object projection); (3) building the mock data cube. This section does not aim to act as documentation for each function, rather to highlight the key methodological principles incorporated at each step. For specific documentation and examples, we refer the reader to the live and continuously-updated documentation website https://kateharborne.github.io/SimSpin/.

The aim is for this code to be agnostic to the type of simulation supplied: smoothed particle hydrodynamics, adaptive mesh refinement, or N-body. In all cases, you should receive a consistent and comparable output cube with metadata such that the whole product can be reconstructed with the information contained within the file itself and the input simulation.

2.1. Creating an input file

We begin with a function, `make_simspin_file()`, whose purpose is to prepare the simulation into a consistent format. This first step allows all other processes to occur in the same way for any type of input simulation or requested telescope. The function accepts an output simulation file (in either HDF5 or Gadget2 Binary format) and returns a binary (`.RData`) file in a universal format that SimSpin can process.

```
make_simspin_file(

    filename, cores = 1,
    disk_age = 5,
    bulge_age = 10,
```

[a] https://github.com/kateharborne/SimSpin/blob/master/NEWS.md.

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

```
disk_Z = 0.024,
bulge_Z = 0.001,
write_to_file = TRUE,
output, overwrite = F,
template = "BC03lr",                    # 1
centre = NA, half_mass = NA,            # 2
sph_spawn_n = 1                         # 3

)
```

Currently, SIMSPIN directly supports simulation inputs cut-out from a range of cosmological models including EAGLE, MAGNETICUM Pathfinder, HORIZONAGN and ILLUSTRISTNG. However, the expected format is fully described within the documentation such that any HDF5 file with the required parameters and units can be read and processed by the code. Further details about how to format your simulation data for ingestion into SIMSPIN can be found through the documentation website.[b] We summarise the main important features here.

Each particle will have a number of existing tagged properties used to track their progress throughout the simulation. In order to make a SIMSPIN observation, the key elements we require include positions (x, y, z), velocities (v_z, v_y, v_z), and masses for the stellar and/or gas components. In order to assign a spectrum to a given 'stellar' particle, we also require ages, metallicities [M/H] and the initial mass of that star. In the case of hydrodynamical simulations, these properties will be tracked throughout the evolution of the system and can be used directly from the output. For isolated N-body models, in which we are just tracing gravitational effects on the motions of bulge and disk stars, we specify these age and metallicity parameters for the stellar particles within the `make_simspin_file()` function to assign these values arbitrarily. A summary of these necessary particle properties will be tabulated and stored as list elements within the SIMSPIN file for later data cube processing. The stellar and gas particle properties will be split into two separate data tables, in the case that gas is present in the input model.

This formatting of the SIMSPIN input allows the code specific metadata to be summarised in an efficient way. In the output of this file, we summarise the properties of the input simulation (e.g. the simulation type and location of the input file from which this product has been made); the parameter choices (e.g. the name and properties of the chosen spectral templates); as well as a record of the code version used to build the file and the date on which it was constructed. This aids the user in inspecting the status of a given file in a human-readable way. It also enables the user to re-create the same file with the same methodology in the future without needing to retain the code used to generate the file.

Besides the universal formatting procedure and metadata addition, the main justification for creating a 'SIMSPIN'-formatted input file is to pre-compute the computationally expensive steps—(1) associating a spectrum with each particle, (2) to align the object within the field-of-view such that our observations are clearly defined and (3) smoothing gas particle or cell properties across their kernel. The galaxy within the output file can be observed multiple times once a single SIMSPIN file has been constructed. However, there are some choices made at this stage that may depend on the type of observations you wish to make, as highlighted in the code snippet above. Further information about these choices will be discussed in this section. We present the necessary considerations for hydrodynamical models, including the treatment of gas (Section 2.1.1) and stars (Section 2.1.2), and for N-body models (Section 2.1.3) below.

[b] https://kateharborne.github.io/SimSpin/examples/generating_hdf5.html.

2.1.1. *Hydrodynamic models: gas components*

In the case of galaxies extracted from hydrodynamical simulations, a population of particles or cells trace the underlying distribution of a fluid (such as the gas in a galaxy). Properties of the fluid are computed across a volume, described by the smoothing 'kernel' or cell size, centred at the given location. In order to ensure that we reproduce this smoothing in our data cubes and recover the underlying fluid properties appropriately within our images, we use an over-sampling method to visualise this kernel volume.

This means that, when we generate a mock observation of the gas component, we must project particles with adaptive sizes onto a fixed grid of pixels. As discussed in Borrow & Kelly (2021), there are many methods of doing this. However, many of the simple methods result in inaccuracies and artefacts due to the projection of spherical kernels onto a rectangular grid.

The smoothed particle hydrodynamic (SPH) kernel projection method is outlined in Borrow&Kelly (2021) (a flavor of which is used in Dolag et al. 2005). We have taken the sub-sampling regime described in these papers and redesigned them for use in SIMSPIN. Particles are treated as Monte Carlo tracers of the field. The basic features of this algorithm are stated below:

1. Each SPH particle read in contains information about its 'smoothing length', h, across which hydrodynamical equations have been computed for the fluid represented at that particle position. In the case of AMR codes, the equivalent information about the 'cell size' is computed from the mass and density of the cell, as described below.
2. We randomly sample `sph_spawn_n` tracer particles within a sphere centred on the true SPH particle position.
3. Each tracer particle is associated with a numerical weight as described by the relevant SPH kernel. All weights for an individual SPH particle will sum to one in order to conserve mass within the system.
4. These new tracer particles replace the original SPH particle. They gain all the properties of the original particle, but a weighted fraction of the total mass according to the weight assigned using the kernel.

This results in a new table of particle properties. The new table will contain '`sph_spawn_n`' times as many rows as the original component of SPH particles. The number of particles it is necessary to spawn will be dependent on the underlying number density of the input model, the desired telescope properties and projected distance to the object. We recommend adjusting this value until each pixel in the mock images contains a minimum of 200 particles to avoid statistical particle noise from dominating the resulting kinematics. Once this table of over-sampled gas particles has been computed, the processing of these observations with `build_datacube()` will be very quick due to the $\mathcal{O}(n)$ computation used to grid particles into pixels. For this reason, we perform the smoothing at the point of making the SIMSPIN input file, rather than at the `build_datacube()` step.

When using this option, we attempt to ensure that the projection kernel corresponds to the kernel used for the SPH calculations within the simulation. For supported hydrodynamic simulations, we provide a smoothing kernel to best match the one used in

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

the original model. These are selected automatically based on the metadata contained within the input file.

Most SPH simulations use a flavour of the Wendland kernel outlined in Wendland (1995). The C^2 Wendland kernel, used in EAGLE (Schaller et al. 2015), is a spherically symmetric kernel, $W(r,h)$, which has the form:

$$W(r,h) = \begin{cases} \frac{21}{2\pi}(1-r/h)^4(4r/h+1), & \text{if } 0 \le r/h < 1 \\ 0, & \text{if } r/h \ge 1 \end{cases} \quad (1)$$

Here, r denotes the distance from the particle to another position at which the weight is calculated and h denotes the smoothing length of a particle. For each simulation, this smoothing length, h, is a value given by requiring that the weighted number of nearest neighbouring particles, N_{neigh}, is a pre-defined constant:

$$N_{neigh} = \frac{4\pi h_i^3}{3} \sum_j W\left(|x_i - x_j|, h_i\right). \quad (2)$$

For the EAGLE simulation, $N_{neigh} = 48$, but this will vary for each SPH simulation. This smoothing length is computed for each particle throughout the simulation, as this value will obviously be dependent on the local number density of particles. The smoothing length, h, is commonly stored as a parameter within the output files. We can use this parameter to then determine the radius across which each individual gas particle should be over-sampled.

The C^6 Wendland kernel used in MAGNETICUM (Teklu et al. 2015) has the form:

$$W(r,h) = \begin{cases} \frac{1365}{64\pi}(1-r/h)^8 \times \\ (1+8r/h+25(r/h)^2+32(r/h)^3), & \text{if } 0 \le r/h < 1 \\ 0, & \text{if } r/h \ge 1 \end{cases} \quad (3)$$

In MAGNETICUM, the smoothing lengths have been computed with $N_{neigh} = 64$ (Beck et al. 2016), but again, the raw h for each particle is given in the output for this simulation.

Finally, in the case of GADGET2 SPH simulations, and for visualisation of AMR/cell model implementations, we use the M4 cubic spline kernel to smooth gas distributions across our image grid. In particular, for mesh-based codes, we do not have a smoothing length for a given cell. As an approximation, we use the quoted cell density and mass to compute an 'effective' smoothing length at a position at the centre of the cell (at the position where cell properties are given).

$$h_i = 2\frac{3}{4\pi}\left(M_i/\rho_i\right)^{1/3} \quad (4)$$

where the effective smoothing length, h, for a given cell, i, is the mass within that cell, M_i, divided by the density of the cell, ρ_i. A spherical distribution is assumed so that the system can be observed fairly from any angle without observing discontinuities at low density locations.

We then use a simplest appropriate kernel, the M4 cubic spline kernel, as an approximation of the behaviour of the gas within a given cell:

$$W(r,h) = \begin{cases} \frac{1}{4\pi}\left((2-r/h)^3 - (1-r/h)^3\right), & \text{if } 0 \le r/h < 1 \\ 0, & \text{if } r/h \ge 1 \end{cases} \quad (5)$$

This approximation is used for visualisation of HORIZONAGN and ILLUSTRISTNG simulations.

It is also important to remember that within true observations of these systems, only gas within specific phases would be observable by an integral field unit. At the point of constructing the SimSpin file, we record the basic attributes of all gas particles within the simulation in a list element `'gas_part'`. At the point of building a mock observation from this information, cold, dense gas can be filtered to produce a comparable set of kinematic maps to observables.

2.1.2. Hydrodynamical simulations: stellar components

Within a hydrodynamical model, stars from in dense, cold gas and their age, metallicity and birth mass are tracked as they evolve through time in the simulation. We use these parameters to tag each stellar particle with a spectral template. These can later be adjusted to reflect the line-of-sight velocity within a given pixel, the distance to the projected object through cosmological dimming, and wavelength resolution of a given telescope.

There are currently three options to choose from for spectral templates used to associate spectra with individual stellar particles, which are listed in Table 1. These prepared templates have been taken from PROSPECT, a generative spectral energy distribution code (Robotham et al. 2020), for which these templates have been prepared using a Chabrier initial mass function (Chabrier 2003). We give this selection of options as a user may wish to focus on a different science question with one set of templates better suited than the other, e.g. for observations using higher spectral resolution instruments, the high-resolution template options will be necessary, but these may be avoided in other cases due to the increased memory requirements and computation. This suite of templates is also a reflection of those commonly used within the literature for exploring stellar kinematics. As this changes with time, we intend to update and expand this selection of template libraries in future versions of the code.

When selected, the spectral templates within the chosen library are used to tag each stellar particle with an associated spectrum. Here, the requirement to select the correct template for the science in question is made clear. E-MILES templates are higher spectral resolution ($\Delta\lambda = 0.9$ Å with $\sigma_{\rm LSF} = 2.51$ Å) in comparison to the variable spectral resolution $\Delta\lambda = 1$ to 50 Å with $\sigma_{\rm LSF} = 3$ Å for the BC03 templates. However, the grid of possible age and metallicity combinations is more sparse in the E-MILES template set, with 636 combinations in comparison to the 1326 available for the BC03 templates. Depending on the science in question, you may value higher spectral resolution, or higher age-metallicity resolution.

The assigned spectrum for a given age-metallicity stellar particle is computed as a weighted interpolation of the four template spectra that surround that particle within the age and metallicity grid of the chosen template. An index for the age and metallicity, as well as the assigned weights based on the location of the particle relative to the template bins, are then stored against the `spectral_weights` list element in the file. Given the template, recorded in the header of the file, these weights can then be used to construct a unique spectrum for each age-metallicity combination during the build of the mock data cube.

In the current version of SimSpin, v2.6.0, we do not modify the spectra of young stars to reflect attenuation due to birth clouds. However, we note that this is an important extension to the code

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

Table 1. Demonstrating the resolution properties of the variety of spectral templates available in SimSpin, including the GalexEV (Bruzual & Charlot 2003) (hereafter BC03) and E-MILES (Vazdekis et al. 2016) as prepared for the ProSpect code (Robotham et al. 2020). It is useful to be aware that the resolution of mock data is built upon templates with finite resolution themselves.

Name	Age steps	Age range (Gyr)	*Z* steps	*Z* range ($Z_\odot$)	λ steps	λ range (Å)	LSF FWHM (Å)
'BC03lr'	221	0–20	6	0.0001–0.05	1221	91–1.6×10^6	3
'BC03hr'	221	0–20	6	0.0001–0.05	6990	91–1.6×10^6	3
'E-MILES'	53	0.03–14	12	0.0001–0.04	53689	1680.2–49999.4	2.51

that will be added in the future and should be noted if your input model is dominated by younger stellar populations.

2.1.3. N-body models

Within isolated N-body models, stellar particles are treated as collisionless and move only under the force of gravity. Unlike their hydrodynamic partners, these stars are initialised with a given 6D distribution. In particular, for Gadget models, particles can be separated into two different distributions to reflect the bulge and disk of a galaxy.

Within SimSpin, we can generate mock data cubes of these kinds of models using these bulge and disk tracer particles as stars. For Gadget-like formatted files, disk particles are listed under `PartType2` and bulge particles under `PartType3`. We assume that these populations all represent stellar material and summarise their attributes within the `star_part` list of the SimSpin file.

Of course, because these stars are not evolved hydrodynamically throughout the course of the simulation, these stellar particles will not contain age, metallicity, or birth mass information. For this reason, we give the user the choice of specifying the star formation history of the bulge and disk stars within the code (i.e. via the `bulge_age`, `bulge_Z`, `disk_age`, and `disk_Z` input parameters). Currently, this will allocate a single age/metallicity value for all bulge stars and all disk stars respectfully. It would be trivial to modify this to assign a more physically motivated range for each population. The age and metallicity information is then used to assign spectra to these particles as described for hydrodynamical systems in Section 2.1.2.

The type of the input simulation is recorded within the metadata of the file, but otherwise should result in a file that can be processed in an identical fashion to the hydrodynamical examples.

2.1.4. Alignment choice

By default, the input galaxy simulation will be aligned in this function such that the semi-major axis of an ellipsoid fit to the stellar component is oriented with the *x*-axis of the reference frame, and the minor axis of the fitted ellipsoid is oriented with the *z*-axis. This provides consistency for multiple observations made at a variety of inclination angles. However, in the case that a full cluster, galaxy group, or a particularly 'clumpy' galaxy with lots of substructure is requested for observation, this alignment will be strongly affected by this non-axisymmetric structure.

Hence, SimSpin gives the user the option to define a single location around which to centre the system (`centre`) and define a half-mass value in solar masses at which the shape of the galaxy will be measured (`half_mass`). If unspecified, the code will evaluate the alignment about the median stellar particle position with an iteratively fit ellipsoid that has grown to contain half the total stellar mass in the input file.

This alignment is done using the method described in the work of Bassett & Foster (2019), which in turn is based on the work from Li et al. (2018) and Allgood et al. (2006). We first assume that the initial distribution of stellar particles is an ellipsoid with axis ratios $p = q$ (i.e. a sphere, where $p = b/a$ and $q = c/a$, with a, b and c representing the axes lengths in decreasing size such that $a > b > c$ and $p > q$, by necessity). This ellipsoid is grown from the median position of all stellar particles within the file (or from the position specified by `centre`) until it contains half the total stellar mass within the file (or the threshold mass described by the specified `half_mass` parameter input). Once this limit is reached, we use the stellar particles within the region to measure the reduced inertia tensor.

The reduced inertia tensor, I, is computed:

$$I_{i,j} = \sum_n \frac{M_n x_{i,n} x_{j,n}}{r_n^2}, \tag{6}$$

where we perform this sum for n stellar particles within the ellipsoid with given positions, x_n, weighted by individual stellar particle masses, M_n, which may vary within the simulation and r_n, the 3D radius of that particle from the centre as described by,

$$r_n = \sqrt{x_n^2 + y_n^2/p^2 + z_n^2/q^2}. \tag{7}$$

The eigenvalues and eigenvectors of this tensor, I_{ij} give the orientation and distribution of matter within the ellipsoid. Specifically, p and q are given by the square-root of the ratios between the intermediate and largest eigenvalues (b and a) and the smallest and largest eigenvalues (c and a) respectively.

The ellipsoid is then deformed to match the distribution of stellar particles. The whole system is reoriented such that the major axis of the distribution identified is now aligned with the major axis of the ellipsoid. We then begin the procedure again, this time growing an ellipsoid with new a, b, and c reflecting the matter distribution of the stellar particles contained. This is repeated until the axis ratios p and q stabilise over ten iterations.

All particles within the input simulation file are aligned with the major axis, a, along the *x*-axis of the volume and the minor axis, c, aligned with the *z*-axis using this method. In the majority of cases, we find that this is suitable for finding the underlying shape of the galaxy in question and aligning the object within the frame. In a data set of 1835 galaxies taken from the IllustrisTNG50-1 simulation (Nelson et al. 2019; Pillepich et al. 2019), a comparison between the alignment found by the `mgefit.find_galaxy` function (Cappellari 2002) and SimSpin revealed that 92.3% (1693/1835) of the alignments agreed within ten degrees. Caution is advised when making mock observations of ellipticals or galaxies undergoing merger interactions, as a visual analysis of the farthest outliers found most fell under these categories.

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

Table 2. A list of predefined parameters for each `telescope()` 'type' available in v2.6.0. A number of these parameters are variables that the user can further specify, which have been emphasised in bold below.

Telescope parameter	Units	SAMI (Croom et al. 2012)	MaNGA (Bundy et al. 2015)	CALIFA	MUSE	Hector
`fov`	arcsec	15	$n = 12, 17, 22, 27$ or 32	74	$n < 60$	30
`aperture_shape`		'circular'	'hexagonal'	'hexagonal'	'square'	'hexagonal'
`wave_range`	Å	3750–5750	3600–6350	3700–4750	4700.15–9351.4	3720–5910
`wave_centre`	Å	4800	4700	4225	6975	4815
`wave_res`	Å	1.04	1.04	2.7	1.25	1.60
`spatial_res`	arcsec/pixel	0.5	0.5	1	0.2 (WFM) or 0.025 (NFM)	0.2
`lsf_fwhm`	Å	2.65	2.85	2.7	2.51	1.3

The steps laid out in this subsection allow us to correctly re-orient the galaxy to the user specified inclination and twist projection at the stage of building the mock data cubes. In cases where the semi-major axis is not well defined, this can be adjusted for purpose with some experimentation of the alignment parameters, `centre` and `half_mass`.

Once this SimSpin file is created for one simulated object, it can be used many times for observations. This file contains all of the multi-dimensional information from the simulation file, with an additional set of tagged properties for SimSpin to construct each cube.

2.2. Initialising the telescope and observing strategy

SimSpin acts as a virtual telescope wrapper. You can choose to observe your galaxy model in a variety of different ways with any integral field unit (IFU) instrument. This requires you to set two distinct groups of properties: (1) the properties of the instrument used to take the observation, i.e. the `telescope()`, and (2) the properties of the object under scrutiny, i.e. the `observing_strategy()`.

The properties are split in this way to enable a suite of observations to be generated in a straightforward manner. It is common that an observer will wish to observe a suite of galaxies using the same telescope, but may want to iterate over a number of projected viewing angles, distances or seeing conditions. Hence, we have split the description classes for the observing telescope and observed object properties into two. We describe the mathematics behind the functions in the sections below, but direct the reader to the specific documentation pages[c] for up-to-date, detailed examples of running each function.

2.2.1. Telescope choice

```
telescope(

    type="IFU",
    fov=15,
    aperture_shape="circular",
    wave_range=c(3700,5700),
    wave_centre, wave_res=1.04,
    spatial_res=0.5,
    filter="g", lsf_fwhm=2.65,
    signal_to_noise = NA

    )
```

[c] https://kateharborne.github.io/SimSpin/docs/documentation.

SimSpin has a number of predefined IFU telescopes, for which the required field-of-view, spectral and spatial resolutions have been taken from the available literature. In Table 2, we describe the values associated with these defaults and their appropriate references.

For a number of these choices, there are further selections that can be made. For example, the 'MaNGA' telescope has a variable field-of-view size that the user can select. If a specific telescope 'type' is not covered by the available options, the parameters can be fully specified by using the `type = 'IFU'`. This requires the user to describe the remaining parameter options of the telescope, including the field-of-view size in arcseconds, the shape of the field-of-view, the wavelength range and central wavelength in Å, the wavelength resolution in Å, the spatial resolution in arcseconds, and the associated line spread function (LSF) of the instrument in Å. Two parameters can be further altered by the user when using the predefine telescope types: the filter, and the minimum level of signal-to-noise.

The available filters in SimSpin v2.6.0 include the SDSS u, g, r, i and z filters (Fukugita et al. 1996; Doi et al. 2010). Each of these data tables are stored as an `rda` file optimally compressed using `xz` compression such that they are lazy loaded with the package. The associated documentation gives the location from which these data have been collected. As with the predefined telescope types, the list of available filters may grow in time. Any updates will be listed on the live documentation website. These filters are then ready to be used in the `build_datacube()` function.

The signal-to-noise specified will be implemented in spectral and kinematic data cubes when a minimum signal-to-noise value is specified. Following the mathematical implementation of noise to cubes given in Nanni et al. (2022) we similarly scale the level of Gaussian perturbation added to each spectrum based on the total flux measured within an integrated spectrum:

$$\frac{dF_i}{F_i} = \frac{\sqrt{\tilde{F}}}{S/N \times \sqrt{F_i}}, \tag{8}$$

where dF/F is the fractional perturbation of flux within a given spaxel i, S/N is the requested parameter given in the `telescope()` function, and $\tilde{F}$ is the median pixel flux from the observation. At each spaxel, we draw a random number from a Gaussian distribution, scaled by this dF/F, and add this perturbation as a function of wavelength to each spectrum. For kinematic cubes, this perturbation is applied to the observed fluxes alone.

With the `telescope()` elements defined, parameters can be precomputed. The number of spatial pixels, `sbin`, required to fill the diameter of the field-of-view (FOV) is computed and stored

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

for gridding purposes. When combined with the coordinate information for a simulation at the `build_datacube()` stage, we can use the following simple equation to label each particle with a corresponding pixel in the FOV of the telescope. We bin the particle data along the x- and y-axes respectively ($x_{\rm bin}$ and $y_{\rm bin}$) labelling each bin with an integer value from 1 to `sbin` and then combine these using,

$$\texttt{pixel\_pos} = x_{\rm bin} + (\texttt{sbin} \times y_{\rm bin}) - \texttt{sbin}, \quad (9)$$

such that every pixel within the FOV has a unique identifier which can be associated with each particle within the model at the `build_datacube()` stage.

Checks are also performed at this stage such that a user does not waste the time loading in a large simulation file only to have the code fail due to a filter mismatch. We ensure that the requested filter will overlap with the telescope wavelength range coverage and the centre of this wavelength range, if not provided, is computed as the centre of the given range. A further check is made for the variable parameters such as MANGA field-of-view, that the requested value is one of the available bundle sizes (i.e. 12", 17", 22", 27" or 32"). If not, the closest value larger than the requested parameter will be taken by default and a warning will be issued. Similarly, if a user asks for a MUSE cube with greater than 60" field-of-view, the value will be reduced to a value of 60". Users will also be able to specify wide-field mode (WFM) in which spaxels are 0.2" or near-field mode (NFM) where spaxels are 0.025" for MUSE. If another value is suggested, the function will default to WFM (as this is the most computationally efficient due to the smaller number of spaxels per arcsecond) and issue a warning to the user that this has occurred.

Further default telescope types will be added in the future to keep up with ongoing developments. The live documentation will reflect any changes made.

2.2.2. Observation strategy choice

```
observing_strategy(

    dist_z       = 0.05,
    inc_deg      = 70,
    twist_deg    = 0,
    pointing_kpc = c(0,0),
    blur         = T,
    fwhm         = 1,
    psf          = "Gaussian"

    )
```

Another necessary ingredient for specifying a mock observation is the description of the conditions in which the model galaxy is observed. How far away is the object? How is it projected on the sky? How severe are the seeing conditions? These properties are specified using the `observing_strategy()` function.

It is expected that a user may wish to observe the same galaxy at a range of distances, inclinations, and seeing conditions, while the overall properties of the `telescope()` are more likely to remain fixed.[d]

To describe the distance to the observed galaxy model, the user may specify a redshift distance (`dist_z`), a physical luminosity distance in Mpc (`dist_Mpc`) or an angular scale distance in kpc per arcsecond (`dist_kpc_per_arcsec`). When any one of these parameters are specified, the other two are calculated through the S4 `Distance` class using the Hogg (1999) methods implemented in the R-package `celestial`.[e]

The inclination and twist parameters define how the model is projected onto the sky. Following the `make_simspin_file()` function, the system is aligned such that the major axis of the ellipsoid (a) is aligned with the x-axis, while the minor axis (c) is aligned with the z-axis. With this knowledge, we can then use basic trigonometry to incline the ellipsoid to a requested inclination and twist.

The inclination of the object describes the level of rotation about the x-axis defined in degrees. We use the definition that $\texttt{inc\_deg} = 0$ is a face on system, while $\texttt{inc\_deg} = 90$ is edge-on. The following mathematics then gives us the coordinates at which the particles would be observed in the y- and z-axis frames.

$$y_i^{obs} = -y_i \sin\left(\frac{\pi}{180}\texttt{inc\_deg}\right) + z_i \cos\left(\frac{\pi}{180}\texttt{inc\_deg}\right), \quad (10)$$

$$z_i^{obs} = y_i \cos\left(\frac{\pi}{180}\texttt{inc\_deg}\right) + z_i \sin\left(\frac{\pi}{180}\texttt{inc\_deg}\right), \quad (11)$$

where the y_i^{obs} and z_i^{obs} denote the observed y and z coordinates of particle i in the rotated frame, and y_i and z_i are the y and z coordinates in the original, fixed ellipsoid frame. The same projections are used for the velocities observed along the rotated y- and z-axis.

Similarly, the 'twist' of the object is described as the rotation about the z-axis of the ellipsoid, i.e. the azimuthal projected rotation on the sky, defined also in degrees. Here, $\texttt{twist\_deg} = 0$ is an object viewed with the major axis, a, parallel to the $x-$axis of the projection, while $\texttt{twist\_deg} = 90$ would be the ellipsoid viewed from the side, such that a is now aligned with the y axis instead. This is computed using similar trigonometric projections as above,

$$x_i^{obs} = x_i \cos\left(\frac{\pi}{180}\texttt{twist\_deg}\right) - y_i \sin\left(\frac{\pi}{180}\texttt{twist\_deg}\right), \quad (12)$$

$$y_i^{obs} = x_i \sin\left(\frac{\pi}{180}\texttt{twist\_deg}\right) + y_i \cos\left(\frac{\pi}{180}\texttt{twist\_deg}\right). \quad (13)$$

where, as above, the x_i^{obs} denotes the observed x coordinates of particle i in the rotated frame, and x_i is the x coordinate in the original, fixed ellipsoid frame. The same equations are used to project the particle velocities.

These projections are performed in the order discussed, i.e. the galaxy ellipsoid is inclined on the sky using Equations (10)–(11) and then twisted using Equations (12)–(13), such that the object can be observed from any angle across the surface of a sphere. This is useful for exploring the effects of inclination and projection on the recovery of galaxy kinematics (Harborne et al. 2019).

The final specification of `observing_strategy()` describes the level of atmospheric seeing via the parameters `psf` and `fwhm`, describing the shape and full-width half-maximum (FWHM) size of the point-spread function (PSF) smoothing kernel respectively. We compute and store the kernel shape here, for each image plane of the observed cube to be convolved at a later stage. Currently, this PSF is not wavelength dependent, but the implementation of

[d] It is also possible to iterate over a range of these parameters to produce a series of observations using `lapply` within R, an example of which can be found at https://kateharborne.github.io/SimSpin/docs/observing_strategy.html.

[e] https://github.com/asgr/celestial.

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

such a convolution kernel would be trivial in future iterations of the code.

Two options are currently available to the user, where the `psf` may be described by a 'Gaussian' kernel, or a 'Moffat' kernel (Moffat 1969) which has a Normal-like distribution at the centre with more extended wings. These are taken from the parameterisation in the R-package `ProFit` (Robotham et al. 2017). A Gaussian kernel is parameterised:

$$I(R) = I_0 \exp\left(-\frac{R^2}{2\sigma^2}\right), \tag{14}$$

where,

$$\sigma = \frac{\texttt{FWHM}}{2\sqrt{2\ln(2)}} \tag{15}$$

and I_0 is the peak intensity at the centre and the `FWHM` is the value specified in the function. A Moffat kernel is parameterised:

$$I(R) = I_0 \left[1 + \left(\frac{R}{R_d}\right)^2\right], \tag{16}$$

where,

$$R_d = \frac{\texttt{FWHM}}{2\sqrt{2^{1/c} - 1}} \tag{17}$$

and $c = 5$, in line with the common defaults. We ensure that the kernel is normalised to 1 such that convolution with the kernel results in suitable flux conservation. These kernels are then stored for use in the blurring step later on.

Having specified the nature of the observation, these functions (`telescope()` and `observing_strategy()`) are combined to summarise the properties of the resulting observation. This is stored as metadata in the final data cube produced. Storing the data in this way ensures that the same file can be produced at a later time using the information stored in the output cube alone. With these parameters specified, we can now go about building our mock observation.

2.3. Building a data cube

Once the observing telescope and properties of the underlying galaxy have been specified, we can go about building a mock observation. Within SimSpin, we present the user with an option at this stage. Either, a series of kinematic maps can be generated from the line-of-sight velocity distributions at each spaxel using the 3D velocity information present for stars, gas or just star forming, cold gas in the simulation; or, you can choose to create a spectral cube using the stellar spectra themselves, shifted in wavelength space to reflect those underlying velocities and projected redshift distance. The resulting spectral cube needs to be run through observational software to generate kinematic maps, and as such is useful for exploring the reliability of reduction pipelines. This choice is specified in the input parameters by the key word `method`:

```
build_datacube(

    simspin_file,
    telescope,
    observing_strategy,
    method = "spectral",
    verbose = F, write_fits = F

    )
```

The behaviour of the code will be different depending on the method chosen, though the outputs of the `method = 'spectral'` and `'velocity'` are equivalent once run through an observational fitting code such as pPXF (Cappellari & Emsellem 2004; Cappellari 2017). We demonstrate this equivalence in the results section.

Whether we are building a kinematic data cube, or a spectral one, the process of re-projecting the model galaxy to a given orientation (using the information provided in the `observing_strategy()`) and gridding particles into the necessary pixel locations is done in the same way (using the `telescope()` specific information) before splitting off into method specific functions.

The output of `build_datacube()` will always include five list elements containing (1) the observed data cube, (2) the metadata table recording the details of the observation, (3) the raw particle property images for reference against, (4), the observed kinematic property images and (5) the observed inverse variance cube ($1/noise^2$). The final two image elements ('raw' and 'observed') will vary in length depending on the type of observation requested. These are summarised in each `method` description below.

2.3.1. Spectral data cubes

If `method = 'spectral'`, the `build_datacube()` function will return a data cube in which each spatial coordinate, $x - y$, holds a spectral energy distribution in gridded wavelength bins along the z-axis. As particles have been allocated to individual pixels within the FOV, we can parallelise over each pixel and perform the mathematics at each pixel in turn, as demonstrated in Fig. 2.

Each stellar particle has been assigned a spectrum using the template described within the `make_simspin_file()` function in Section 2.1. These spectra are at the resolution of the templates from which they have been drawn, e.g. with E-MILES templates, these spectra will have a wavelength resolution of $\Delta\lambda = 0.9$ Å and a spectral resolution of 2.51 Å. The template spectrum is weighted by the particle's stellar initial mass to give the luminosity as a function of wavelength.

We shift the wavelength labels to $\lambda_{\text{obs-z}} = \lambda(1 + z)$ to account for the input redshift of the system. Within each pixel, we then further shift the wavelength labels according to the LOS velocity of each individual particle, $\lambda_{\text{obs}} = \lambda_{\text{obs-z}} \exp(v_{\text{LOS}}/c)$. At this stage we are still just modifying the raw spectral templates.

Once each template is both z-shifted and v_{LOS}-shifted, we then interpolate these spectra onto the wavelengths observed by the requested `telescope()`. This is done using a spline function in which an exact cubic is fitted using the method described by Forsythe, Malcolm, & Moler (1977). Next, the individual particle spectra are summed column-wise to produce the observed spectrum at that pixel. This procedure is repeated for every pixel within the FOV and then the spaxels are combined into a volume to construct a 3D data cube, with spatial dimensions in the x- and y-axes and wavelength information in the z-axes.

If a PSF (i.e. atmospheric blurring) has been specified in the `observing_strategy()`, we then perform a spatial 2D convolution across each $x - y$ plane in the cube. The convolution kernel will have a shape and width as described by the `observing_strategy()` in Section 2.2.2:

$$F_{obs}(\lambda) = F(\lambda) \circledast PSF. \tag{18}$$

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

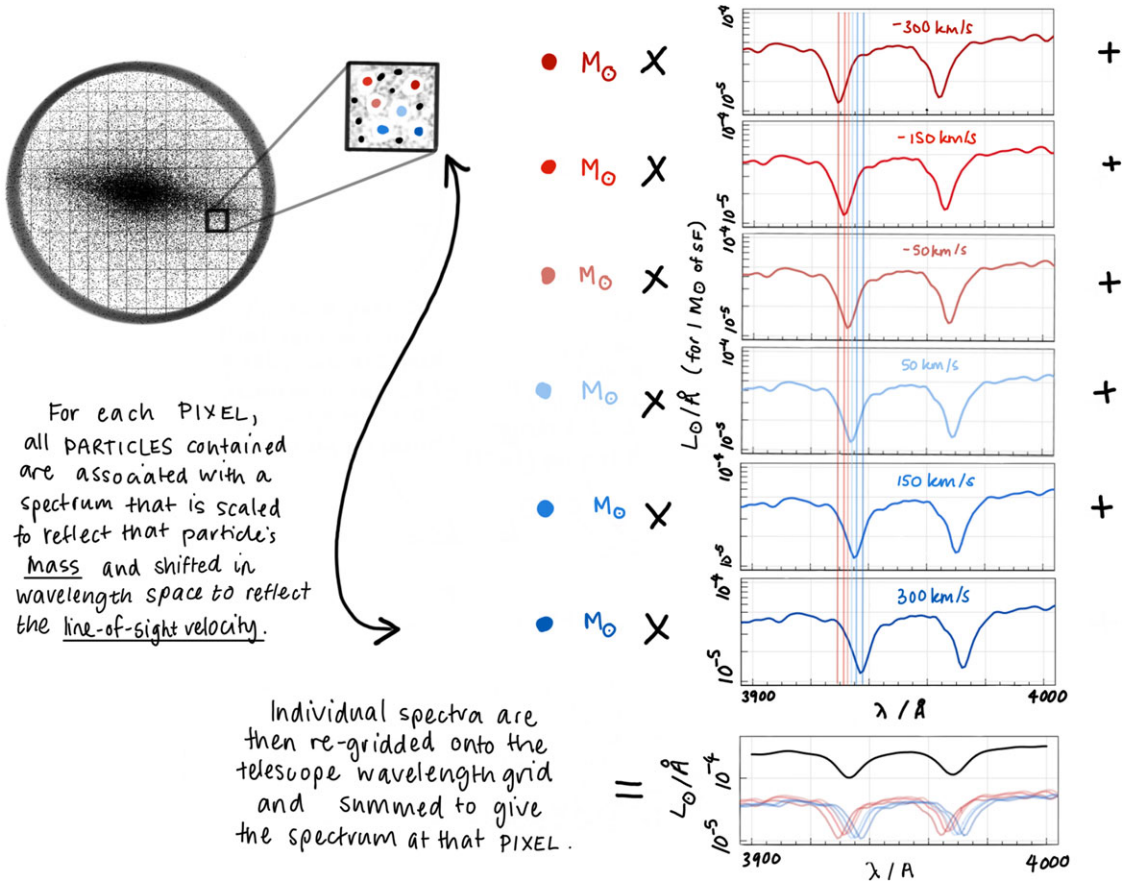

Figure 2. Method for constructing spectral data cubes. For the set of particles associated with each spaxel position across the field-of-view, we take the spectrum associated with each stellar particle, weight it by the initial mass of that star particle and then shift the spectrum in wavelength space to reflect that particle's line-of-sight velocity. Each spectrum in the pixel is then interpolated onto the wavelength range of the observing telescope and summed to give the overall spectrum at that spaxel position. The summed spectrum is finally convolved with the $\lambda_{\rm LSF}$ of the observing telescope.

Following the spatial convolution, we also need to convolve the summed spectrum with a Gaussian kernel, with width $\Delta\sigma_{\rm LSF}$, mimicking the effects of the spectral resolution of the instrument, where $\Delta\sigma_{\rm LSF}$ is the root-square of the difference between the telescope and the redshift-ed templates.

The template spectra associated with a single particle have an intrinsic spectral resolution, $\lambda_{\rm LSF}^{template}$. Of the templates included within this package, these resolutions range from 2.51–3 Å in the rest-frame. This spectral resolution represents a 'minimum dispersion' due to the instrument with which the template was observed or modelled. When the template spectrum is moved to greater redshift, the spectrum is stretched in wavelength space. When we wish to model a galaxy at redshift, z, the intrinsic spectral resolution of the templates must also be adjusted to this new redshift-ed spectrum. At higher redshift, the minimum dispersion we can detect with these templates becomes larger, as the wavelength space is broadened.

Hence, we must account for this when mimicking the effect of using our 'mock' telescope with its spectral resolution, $\lambda_{\rm LSF}^{telescope}$. The value of this resolution is fixed by the telescope and is assumed constant with redshift. However, the templates which we have redshift-ed to some distance, z, will now have some intrinsic spectral resolution, $\lambda_{\rm LSF@z}^{template} = \lambda_{\rm LSF}^{template}(1+z)$. To match the spectral resolution of the observing telescope then, we only need convolve our templates with a Gaussian the root-square of the difference between the telescope and the redshift-ed templates, i.e.

$$\Delta\lambda_{\rm LSF} = \sqrt{\left(\lambda_{\rm LSF}^{telescope}\right)^2 - \left(\lambda_{\rm LSF@z}^{template}\right)^2}. \quad (19)$$

This is computed using the metadata information contained in the input SimSpin file. The user simply needs to specify the resolution of the observing telescope.

Finally, we add the level of noise requested to each spectrum, as described in the `telescope()` function. The inverse variance of this noise ($1/noise^2$) is also returned to the user under the '`variance_cube`' list element. If no noise is requested, this list element will be returned the user with NULL.

The resulting 'observed' spectral cube is returned under the '`spectral_cube`' list element. A summary of the run observation details are tabulated and returned under the list element '`observation`'. At each pixel, we also measure a number of particle properties, including the total number of particles in each location, the total particle flux, the mean and standard deviation LOS velocity, the mean stellar age and mean stellar metallicity. This information is stored as an image returned to user under the list element '`raw_images`'. All of these details can optionally be saved to a FITS file that contains each of these elements in subsequent HDU extensions for later processing with observational

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

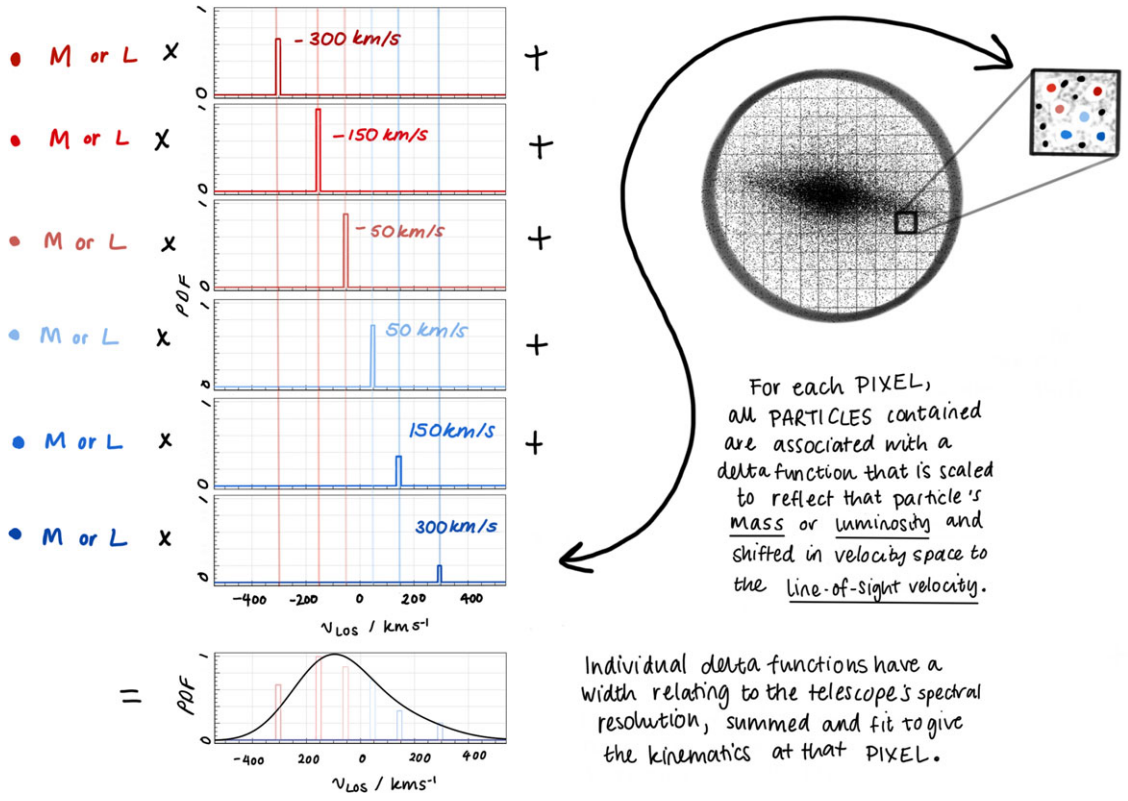

Figure 3. Method for constructing kinematic data cubes. At each pixel, the velocities of the contained particles are binned into velocity channels that map back to the underlying spectral resolution of the observing telescope. These LOSVD's can be weighted by the underlying particle mass or luminosity depending on the settings selected at the `build_datacube()` stage.

pipelines. Examples of this can be found at the documentation website.[f]

2.3.2. Kinematic data cubes

If `method = 'velocity'`, the `build_datacube()` function will return a data cube in which each spatial coordinate, $x-y$, holds a line-of-sight velocity distribution in gridded velocity bins along the z-axis. A visual representation of this process is outlined in Fig. 3.

Given the wavelength and spectral resolution of the underlying telescope, we can compute the effective velocity sampling rate of a given instrument as:

$$\Delta v = c\,\Delta\log(\lambda), \tag{20}$$

were λ is the wavelength resolution of the given instrument, $\Delta\log(\lambda)$ represents the smallest wavelength gap in log space and c is the speed of light.

As in the previous methodology, we can use the gridded FOV to perform the required mathematics on a pixel-by-pixel basis. For each pixel, we take each contained stellar particle. Each stellar particle has been assigned a spectrum using the template described within the `make_simspin_file()` function in Section 2.1. This spectrum is multiplied by the initial mass of the stellar particle and re-gridded on the wavelength scale of a given telescope to give the luminosity at all wavelengths measured. From this spectrum, the luminosity of that particle can be computed. Each particle also has a mass, which can be used to weight the kinematics in place of the particle luminosity when `mass_flag = T`.

Each particle's velocity is binned along the velocity axis dependent on the wavelength (and associated velocity) resolution as specified in Equation (20). This distribution is weighted either the particle's luminosity in a given band (given by passing the observed spectrum through the specified band pass filter) or the mass of the particle. This leaves us with a line-of-sight velocity distribution (LOSVD), weighted by luminosity or mass, for each spatial pixel at the resolution of the respective `telescope()`. This process is repeated for every spatial pixel.

At each pixel, as in the spectral mode case, we also measure a number of the raw particle properties including the total number of particles, the mean and standard deviation of the population of particle velocities, the mean stellar age and mean stellar metallicity. These are returned to the user as 2D named arrays embedded within the list element, '`raw_images`'.

If an atmospheric blurring is specified, convolution of the kernel selected and described in Section 2.2.2 is performed across each spatial plane of the kinematic data cube following its construction, this time as a function of the velocity channels rather than wavelength channels:

$$F_{obs}(v) = F(v) \circledast PSF. \tag{21}$$

If requested, noise is added per spaxel as described in the `telescope()` function, applying dF/F as a function of velocity, rather than wavelength. We save a volume of the added noise as an inverse variance velocity cube ($1/noise^2$) and return this to the user under the list element '`variance_cube`'. The final, 'observed' 3D

[f] https://kateharborne.github.io/SimSpin/examples/examples.

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

array structure, containing spatial planes of the data in the $x-y$ at subsequent velocity channels along the $z-$axis, is returned to the user under the '`velocity_cube`' list element.

From this kinematic data cube, we also compute a number of '`observed_images`'. At each pixel in the cube, we now have a LOSVD sampled at the same resolution as the wavelength resolution of the telescope. This distribution is fit with a Gauss-Hermite function of the form:

$$L(\omega, h_3, h_4) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\omega^2}{2}\right)\left[1 + h_3 H_3 + h4 H_4\right], \quad (22)$$

where,

$$\omega = \frac{v_i - V}{\sigma}, \quad (23)$$

$$H_3 = \frac{1}{\sqrt{6}}\left(2\sqrt{2}\omega^3 - 3\sqrt{2}\omega\right), \quad (24)$$

$$H_4 = \frac{1}{\sqrt{24}}\left(4\omega^4 - 12\omega^2 + 3\right), \quad (25)$$

where v_i is the observed velocity channels, V and σ are the first and second order moments of the LOSVD, h_3 and h_4 represent the expanded third and fourth moments of the Hermite polynomial (van der Marel & Franx 1993; Cappellari 2017). This fit is performed using the quasi-Newton method published simultaneously by Broyden, Fletcher, Goldfarb and Shanno in 1970 (known as BFGS) (Broyden 1970; Fletcher 1970; Goldfarb 1970; Shanno 1970) using the `optim` minimisation provided in base R.

We compute the observed LOS velocity, dispersion and higher-order kinematics h_3 and h_4 on a pixel-by-pixel basis through this fit. If a PSF has been specified in the `observing_strategy()`, this fit is performed on the spatially blurred cubes and the resulting images will have this blurring effect incorporated (unlike the raw particle properties, which will be returned as a summary of the underlying simulation). Each parameter is stored as a 2D array and returned to the user under the list element '`observed_images`'. The residual of this fit to the LOSVD is also returned to give an understanding of how well these returned parameters describe the true underlying distribution. This is also output as a 2D array under the same list element as the observed images.

As in the spectral mode case, the returned observation can be written to a FITS file for later processing. Each of the arrays in the list elements are saved to subsequent HDU extensions with explanatory names so that the raw and observed images can be distinguished (e.g. `OBS_VEL` and `RAW_VEL` for the observed LOSVD and the raw particle mean velocity images respectively). These will be presented in a consistent format to the spectral FITS files, but with the velocity cube output under the `DATA` extension, with the necessary axes labels given the header information.

2.3.3. Gas data cubes

If `method = 'gas'` or `'sf gas'`, the `build_datacube()` function will follow the kinematic data cube methodology, but only for the gas component (or gas classed as star forming, in the later case) of the input model. As in Section 2.3.2, this results in a data cube containing the spatial information about the gas distribution along the x-y axes, with velocity information along the z-axis.

To distinguish between all gas and gas particles that are classed as star forming, we filter by the instantaneous star-formation rate. These properties are commonly reported against each gas particle within the model and allow us to filter gas that has met the threshold for star formation.

Beyond the focus on the gas component, rather than the stellar component, the process by which this cube is constructed is almost identical to above. The gas kinematics are weighted by the observed gas mass per pixel, rather than using a luminosity. This is equivalent to forcing `mass_flag = T` in the stellar kinematic cube construction. Each gas particle also has some intrinsic dispersion related to their thermal motions. We compute the thermal contribution to the dispersion of each particle as:

$$\sigma^2_{\rm thermal} = P/\rho = u(1-\gamma), \quad (26)$$

where P is the gas pressure, ρ is the gas density, u is the internal energy of the gas and $\gamma = 5/3$ is the adiabatic index. Of course, due to the effective equation of state employed by many cosmological simulations, this approximation for thermal motions is no longer valid once gas cools below the star forming threshold. At this stage, the temperature and internal energies become effective measures. In this regime, we assume a thermal value which reflects the sound speed of gas at the temperature floor in each hydrodynamical simulation (Pillepich et al. 2019; Jiménez et al. 2023).

The mock observed kinematic images are constructed as above and we return the observed mass, velocity, dispersion, h_3 and h_4 images under the '`observed_images`' However, a number of additional raw particle properties are also included in the gas output.

In addition to the raw gas mass per pixel, we record the mean mass-weighted instantaneous star formation rate, the mean gas metallicity, and the mean oxygen over hydrogen abundance ratio. The raw mass-weighted mean velocity and standard deviation are also returned in this '`raw_images`' list under clearly named 2D arrays. A number of these images are shown for our example galaxy from the EAGLE simulation in Fig. 1 in the gas property maps on the right hand side.

In the future, we aim to incorporate the gas information at each pixel position within the spectral cube, through the addition of emission lines of appropriate ratio and kinematics. This is currently beyond the scope of the code, due to the necessity to incorporate other features of realism such as the attenuation and re-emission due to dust which is currently beyond the resolution limits of the majority of simulations. We direct the user to codes such as SKIRT (Camps & Baes 2020) and the work of Barrientos Acevedo et al. (2023) for the proper radiative transfer treatment through an assumed dust distribution.

3. Results

3.1. Comparison of spectral and kinematic cubes

A kinematic data cube should mimic the kinematic information included within a full spectral cube. Here, we present a series of tests to ensure the similarity of these products using two high-resolution galaxy models. One model represents a disk galaxy with highly coherent rotation. The other represents an elliptical galaxy with highly dispersive support. At these extremes, we hope to identify any systematic offsets between the kinematic cubes and spectral cubes as a function of the underlying model.

These high-resolution N-body models have been constructed using the initial conditions code `GalIC` (Yurin & Springel 2014)

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

and evolved in a smooth analytic potential using a modified version of Gadget2 (Springel 2005). Each galaxy contains 6.5×10^6 particles, each of mass 1×10^4 $M_\odot$.

The elliptical system is modelled as a spherically symmetric model with density profile described by:

$$\rho(r) = \frac{M}{2\pi} \frac{a}{r(r+a)^3}, \tag{27}$$

where a is the scale factor given by,

$$a = \frac{r_{200}}{c} \sqrt{2 \left[\ln(1+c) - \frac{c}{(1+c)} \right]}, \tag{28}$$

where r_{200} and c is the concentration of the distribution.

The disk model has been initialised with an exponential radial profile and sech^2-profile in the vertical direction, described by:

$$\rho(R,z) = \frac{M_*}{4\pi z_0 h^2} \mathrm{sech}^2 \left(\frac{z}{z_0} \right) \exp \left(-\frac{R}{h} \right). \tag{29}$$

The velocity profiles of these structures are initialised using the optimisation procedure outlined in Yurin & Springel (2014). We allow these systems to evolve in an analytic potential for 10 Gyr using Gadget2. We outline the procedure for each test below.

We begin by generating three SimSpin files. As these are N-body models, we must assign each particle a stellar age and metallicity (such that an appropriate stellar template can be assigned). We produce two SimSpin files with identical particle ages and metallicities in order to examine both the variation of the underlying kinematic model (i.e. bulge vs. disk) with consistent underlying spectra:

```
make_simspin_file(
    filename = "disk_model.hdf5",
    disk_age=5, disk_Z=0.024,
    template = "E-MILES",
    output = "disk_age05_Z024.Rdata"
    )

make_simspin_file(
    filename = "bulge_model.hdf5",
    bulge_age=5,
    bulge_Z=0.024,
    template = "E-MILES",
    output = "bulge_age05_Z024.Rdata"
    )
```

The final SimSpin file contains more realistic stellar ages and metallicities for their component, with older, more metal poor stars present in the elliptical system and younger, more metal rich stars in the disk galaxy. This allows us to examine the effect of varied stellar templates on the comparison between spectral and kinematic data cubes.

```
make_simspin_file(
    filename = "bulge_model.hdf5",
    bulge_age=10, bulge_Z=0.001,
    template = "E-MILES",
    output = "bulge_age10_Z001.Rdata"
    )
```

We build two versions of each of these files. One is prepared using the E-MILES templates (Vazdekis et al. 2016), which have both higher wavelength and spectral resolution parameters than the alternative BC03 models (Bruzual&Charlot 2003) from which we prepare the second SimSpin file.

These SimSpin files are used for the following tests in an effort to evaluate the consistency between our two mock observing methods. In each case, we generate a kinematic data cube and a spectral data cube. These cubes have identical observing conditions, i.e. projected distance, observed projection angle on the sky, field-of-view, etc. The spectral cube is then fit using pPXF, with the E-MILES spectra used as fitting templates. We then compare the kinematic maps produced through the penalised pixel fitting method and our kinematic cubes. With the two versions of SimSpin file (E-MILES and BC03), we can examine consistency across a variety of spectral qualities. Within each test, we can further turn the dials of the `telescope()` and `observing_strategy()` functions to explore the reliability of the results with respect to the LSF, spatial and spectral resolution and seeing.

Selected properties are described in each of the case studies below. We have comparison figures for each test contained in the Supplementary Material at the end of the paper, though a summary of these is also presented at the end of the results section. A walk-through of code used to generate these examples can also be found at the SimSpin website.

3.1.1. Test 1: Intrinsic template spectral resolution at low redshift

We would like to ensure that, in the simplest regime where there is no LSF convolution and the object is projected to a small redshift, a kinematic data cube and a spectral data cube fit with pPXF should return consistent answers. In essence, this tests that the velocity-shift added to each particle's spectrum is working appropriately. This is done for all three examples (the young disk, young bulge and old bulge). We use different `telescope()` configurations for the E-MILES and BC03 cubes to suit the different resolution constraints for these spectra, and to explore the robustness of the comparison to different configurations of the telescope.

This is done using the following telescope parameters for the E-MILES and BC03hr cubes respectively:

```
telescope(
    type = "IFU", signal_to_noise=30,
    lsf_fwhm=0,
    wave_res=1.04, aperture_shape = "circular",
    fov=15,
    spatial_res=0.5
    )

telescope(
    type = "IFU",
    signal_to_noise=30,
    lsf_fwhm=0, wave_res=3,
    aperture_shape = "hexagonal",
    fov=17, spatial_res=0.7
    )
```

The same observing strategy is used for all observations:

```
observing_strategy(
    dist_kpc_per_arcsec=0.3,
    inc_deg=60,
    blur = F)
```

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

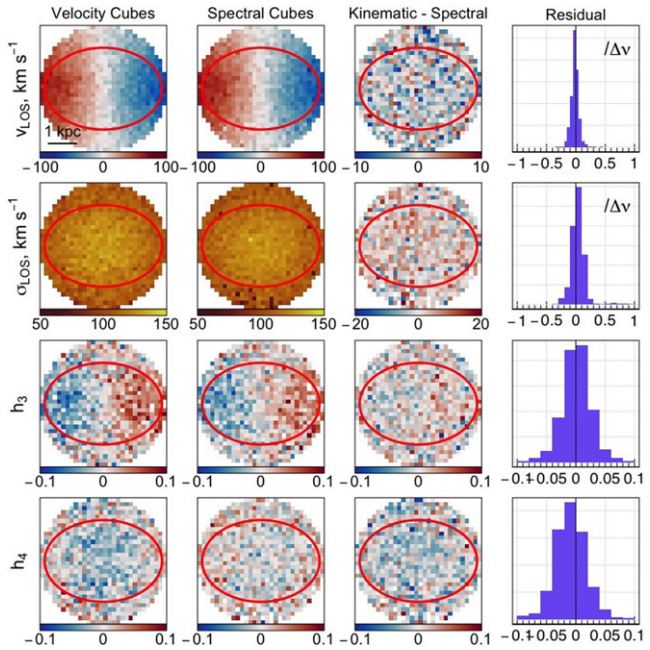

Figure 4. Case Study 1: The disk model built with E-MILES templates observed with an intrinsic telescope resolution of $\lambda_{\rm LSF}^{telescope} = 0$ Å at a low redshift distance of $z = 0.0144$. Here we compare the output kinematic cubes to the kinematics fit with pPXF, where the average pixel spectral fit has $\chi^2/DOF = 0.95$. The red ellipse demonstrates 1 R_e for this model.

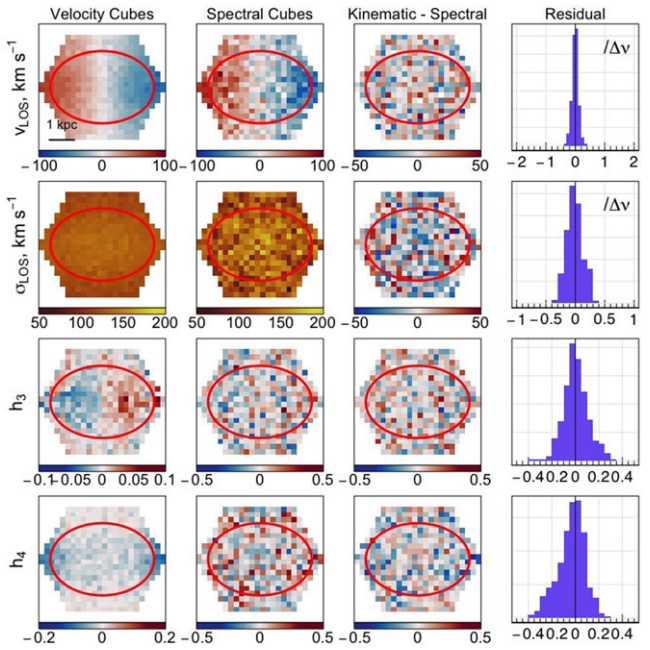

Figure 5. Case Study 1: The disk model built with BC03 templates observed with an intrinsic telescope resolution of $\lambda_{\rm LSF}^{telescope} = 0$ Å at a low redshift distance of $z = 0.0144$. Here we compare the output velocity cubes to the kinematics fit with pPXF, where the average pixel fit $\chi^2/DOF = 3.46$. The final column demonstrates these residuals as histograms

In this test, we force SimSpin to generate a spectral cube at the intrinsic template spectral resolution by requesting a telescope with a $\lambda_{\rm LSF}^{telescope} = 0$ Å. This will cause the code to issue a warning that the templates used have insufficient resolution to construct such an observation, but will produce the output spectral cube never-the-less.

It is important to remember that the underlying templates used to construct the observed galaxy do have some intrinsic LSF, as shown in Table 1. Hence, when using spectral templates to fit the SimSpin spectral cubes with pPXF, it is important that we do match the fitting templates to the true underlying LSF, which is dependent on the templates from which the cube has been made $\left(\lambda_{\rm LSF}^{templates} = 2.51\,\text{Å}\right.$ in the case of E-MILES SimSpin cubes and $\left.\lambda_{\rm LSF}^{templates} = 3\,\text{Å}\right)$ in the case of BC03 SimSpin cubes. When performing the pPXF fit using the E-MILES templates to fit the model spectra, we do convolve the fitting templates with the root-square difference between the BC03 and E-MILES LSF (i.e. $\sqrt{3^2 - 2.51^2} = 1.64$ Å and $\sqrt{2.51^2 - 2.51^2} = 0$ Å), due to the fact that the intrinsic templates from which the mock observation has been built have a greater LSF than the templates used to perform the fit.

We compare the output of the pPXF run in this case to a kinematic data cube run using the same parameters, but this time with `method = 'velocity'`. We expect that the observed kinematics will be consistent within the noise and the resolution of the telescope. The resulting comparison can be seen visually for our disk model in Figs. 4 and 5. Similar plots for each of the models built for these tests can be found in Appendix 1.1 (see Supplementary Material). Visually, it is clear that the E-MILES spectral cube comparison in Fig. 4 is much more consistent than the BC03 spectral cube comparison in Fig. 5. However, in both cases the residual distributions are centred around zero. In the recovery of the kinematics in the BC03 example, we struggle to find a sufficiently good fit, with the χ^2/DOF averaging $\simeq$4, as opposed to the E-MILES comparison value of $\simeq$1. We believe this is due to mismatch between the templates used for cube generation and for fitting with pPXF. As discussed in Nanni et al. (2023), the biases introduced through the adoption of different spectral models are important to consider. Here, we demonstrate the impact of these biases. It is important to note, of course, that the spread of residuals recorded in both cases are within the velocity resolution of the telescope used.

In Fig. 6, we show the residual differences between the kinematic and spectral cubes as a histogram for each model and spectral template set. This allows us to directly compare the differences between spectral cubes built with the E-MILES and BC03 templates. At low redshift and with no additional LSF effects, we see that the two methods (`'spectral'` and `'velocity'`) compare quite nicely, with all resulting residuals centred around zero. As noted visually from the kinematic maps, there is a broader difference between the returned kinematics for the BC03 SimSpin cubes fit with the E-MILES templates through pPXF.

3.1.2. *Test 2: Intrinsic template spectral resolution at high redshift*

Following the success at low redshift, where we tested that spectra are shifted in wavelength space effectively, we next consider the effect of projecting the galaxies to larger distances. In this study, we use the same telescope definitions as in Test 1, keeping the templates from which the cubes are built at their intrinsic resolution using $\lambda_{\rm LSF}^{telescope} = 0$ Å, but modifying the observing strategy as follows:

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

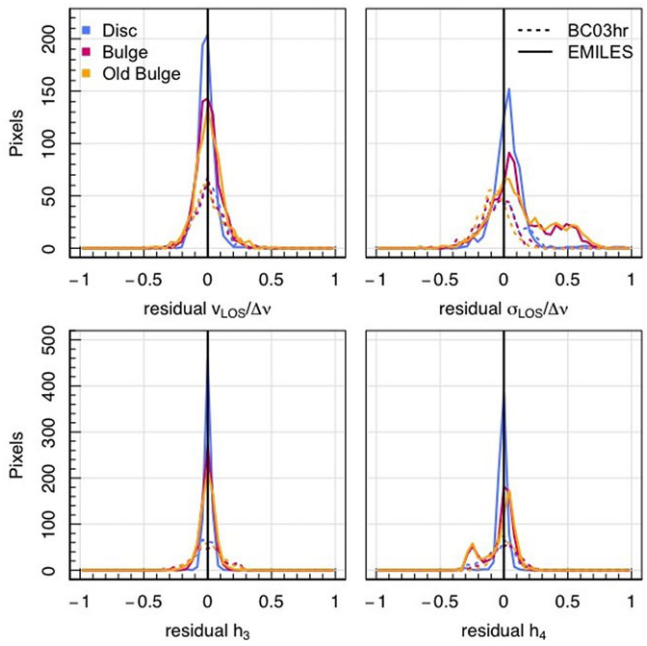

Figure 6. Case Study 1: The residual differences between the kinematic observations and the spectral fits in case study 1 for the v_{LOS} and σ_{LOS} each with respect to the velocity resolution of the telescope, and h_3 and h_4, for each of the models (disc, bulge, and old bulge in blue, pink and yellow respectively). The solid lines show the residual relationship for the E-MILES cubes, while the dotted lines demonstrate the residuals for the BC03hr cubes. All are nicely centred around zero as we would expect, though we do see broader distributions for the BC03hr models in comparison to the E-MILES models.

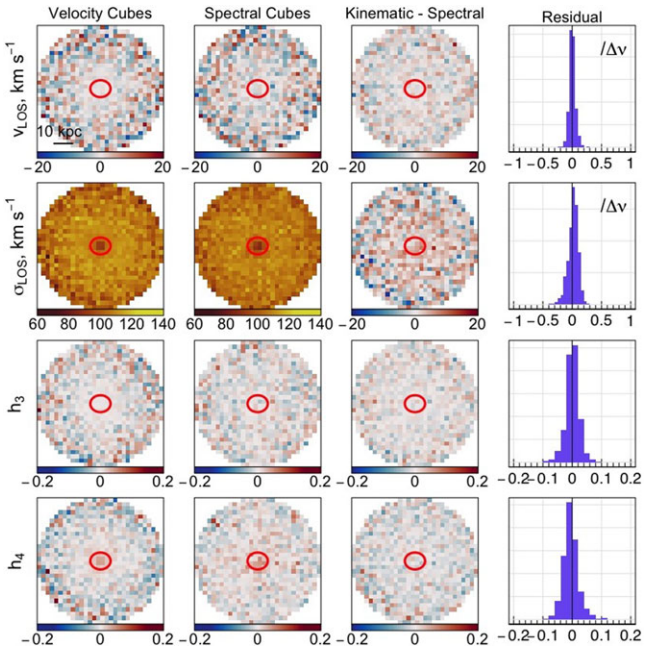

Figure 7. Case Study 2: The bulge model built with E-MILES templates observed with an intrinsic telescope resolution of $\lambda_{LSF}^{telescope} = 0$ Å at a high redshift distance of $z = 0.3$. Here we compare the output kinematic cubes to the kinematics fit with pPXF, where the average pixel fit $\chi^2/DOF = 0.88$.

```
observing_strategy(
    dist_z = 0.3,
    inc_deg = 60,
    blur = F
    )
```

We note here that the median signal-to-noise is set to 30, as in the previous test. It is important to remember that, with objects projected to further distances, we do not perform an exposure time calculation and as such these may not be realistic of the noise expected from such an observation.

Here, we examine whether the red-shifting module is working effectively in both methods and still produces equivalent results between the `spectral` and `velocity` cubes. We build both a spectral and velocity cube of each of the simulations with these specifications. The resulting spectral cubes are fit using pPXF to find the observed spectral kinematics and the maps are compared with their `method = 'velocity'` counterparts.

As in the previous test, we compare the kinematic maps for each model, as shown in Figs. 7 and 8. This time, we demonstrate using the bulge model, but provide the images for every model tested in Appendix 1.2 (see Supplementary Material). We successfully recover kinematic details in the E-MILES built images in Fig. 7. However, we find that it is much more difficult to get a successful fit for the BC03 spectral cubes through pPXF. The direct comparison between the two is clearly demonstrated in the histograms in Fig. 9. At the wavelength resolution of 3 Å, as is run for the BC03 SimSpin cubes, we find that it is especially difficult to recover the higher-order kinematics, as would be expected for higher redshift observations for a telescope with poorer spectral resolution.

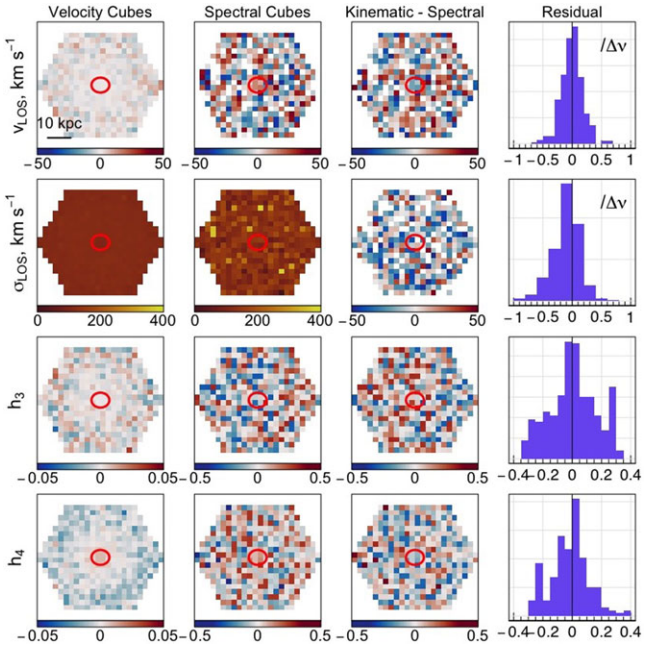

Figure 8. Case Study 2: The bulge model built with BC03 templates observed with an intrinsic telescope resolution of $\lambda_{LSF}^{telescope} = 0$ Å at a high redshift distance of $z = 0.3$. Here we compare the output kinematic cubes to the kinematics fit with pPXF, where the average pixel fit $\chi^2/DOF = 51$.

3.1.3. Test 3: `telescope()` spectral resolution at low & high redshift

The next test is designed to evaluate the module of the code that varies the spectral resolution. In this case study, we take the disk simulation built with each the E-MILES and BC03 templates and

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

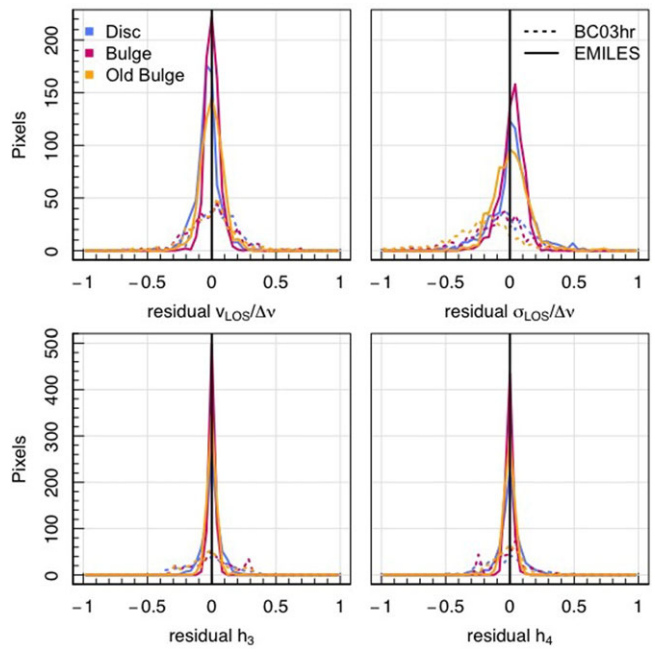

Figure 9. Case Study 2: The residual differences between the kinematic observations and the spectral fits as in Fig. 3, but for case study 2. Again, we find that all distributions are nicely centred around zero as we would expect, though we do see significantly broader distributions for the BC03hr models in comparison to the E-MILES models.

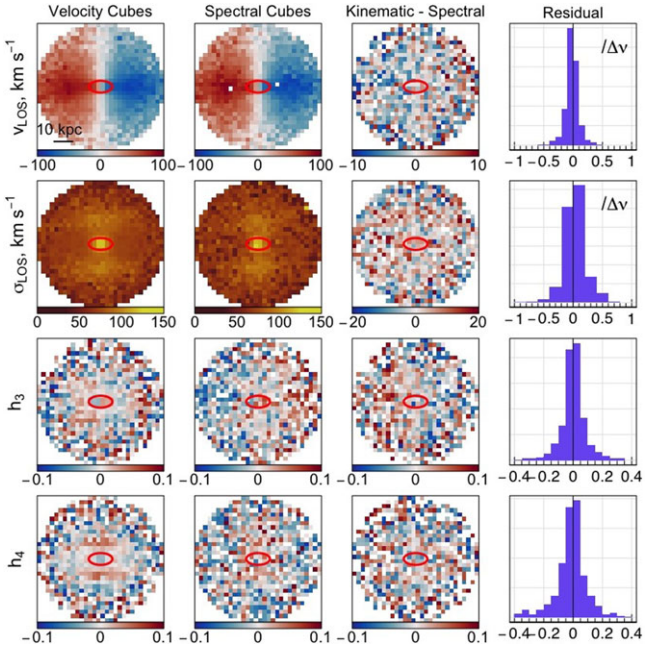

Figure 10. Case Study 3: The disk model built with E-MILES templates observed with an intrinsic telescope resolution of $\lambda_{LSF}^{telescope} = 3.61$ Å at a high redshift distance of $z = 0.3$. Here we compare the output kinematic cubes to the kinematics fit with pPXF.

observe them using telescopes with LSFs greater than the underlying templates. We do this with the disc projected at both low and high redshift, as the convolution kernel used for the LSF will change as a function of z as demonstrated by Equation (19).

We broaden each set of templates by different amounts as shown in the `telescope()` specifications below for the E-MILES and BC03 SimSpin files respectively:

```
telescope(
    type = "IFU",
    signal_to_noise=30,
    lsf_fwhm=3.61,
    wave_res=1.04,
    aperture_shape = "circular",
    fov=15, spatial_res=0.5
    )

telescope(
     type = "IFU",
    signal_to_noise=30,
    lsf_fwhm=4.56,
    wave_res=3,
    aperture_shape = "hexagonal",
    fov=17, spatial_res=0.7
    )
```

We then run each model twice, once at low and once at high z, using the following `observing_strategy()` functions:

```
observing_strategy(
    dist_kpc_per_arcsec=0.3,
    inc_deg=60,
    blur = F
    )
```

```
observing_strategy(
    dist_z=0.3,
    inc_deg=60,
    blur = F
    )
```

As before, we produce a spectral and kinematic SimSpin cube for each iteration and run the spectral cubes through pPXF to recover the observable kinematics. For this set of pPXF fits, when using the E-MILES templates to fit the model spectra, we only need to convolve the fitting templates with the root-square difference between `telescope()` LSF for each observation and the fitting templates (i.e. $\sqrt{3.61^2 - 2.51^2} = 2.59$ Å and $\sqrt{4.56^2 - 2.51^2} = 3.81$ Å for the E-MILES and BC03 examples respectively).

The results of these fits are demonstrated visually in Figs. 10 and 11. These examples show the high redshift examples, with the low z fits shown in Appendix 1.3 (see Supplementary Material).

In Fig. 10, we can see that the structure of the LOS dispersion has been well captured in the resulting spectral fit. However, we see that the higher-order kinematics, h_3 and h_4 become quite difficult to explore as you go out in radius where noise begins to dominate.

We provide a direct comparison between the BC03 and E-MILES residuals in Fig. 12, built at both high and low redshift. It is quite clear from this comparison that there is no significant difference between the high and low redshift behaviour, except in the case of the cube built with BC03 templates. In this example, we see that the lower redshift model appears to under-estimate the true dispersion, as shown by the positive dispersion residuals.

Given the difficulty we have had fitting the BC03 spectral models for kinematics, it is unclear whether these discrepancies are the fault of the SimSpin code, or the fitting methodology. As the fits are quite consistent for the E-MILES spectral cubes, we will proceed with the final test to check for consistency when atmospheric blurring conditions are incorporated.

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

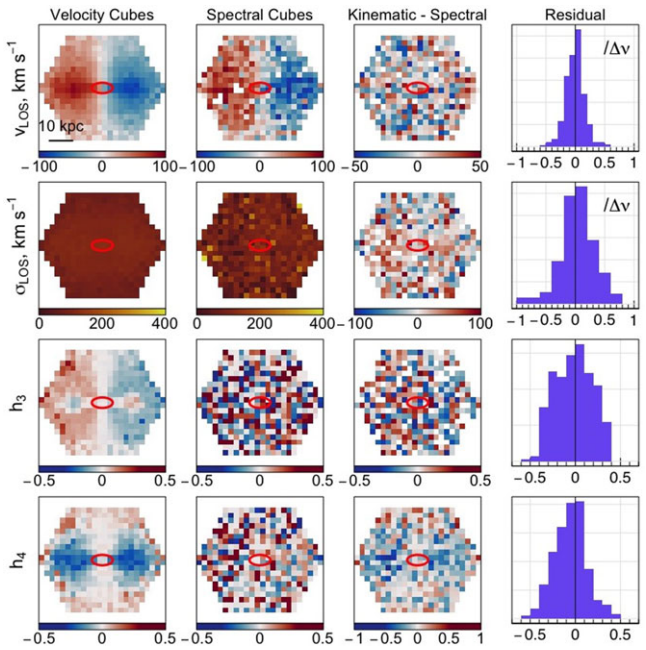

Figure 11. Case Study 3: The disk model built with BC03 templates observed with an intrinsic telescope resolution of $\lambda_{LSF}^{telescope} = 4.56$ Å at a high redshift distance of $z = 0.3$. Here we compare the output kinematic cubes to the kinematics fit with pPXF.

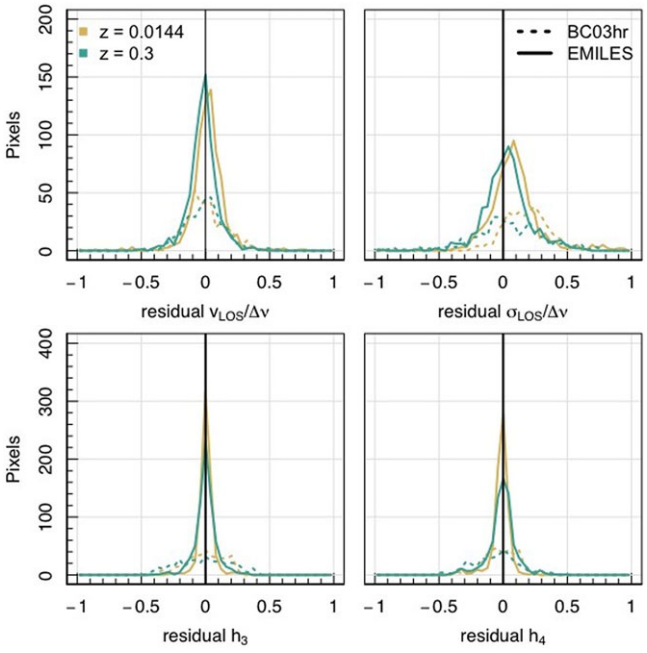

Figure 12. Case Study 3: The residual differences between the kinematic observations and the spectral fits for the low and high redshift disc models (in yellow and green respectively) built with E-MILES and BC03 templates. All distributions are nicely centred around zero as we would expect, though we do see significantly broader distributions for the BC03hr models in comparison to the E-MILES models.

3.1.4. Test 4: `telescope()` spectral resolution with atmospheric seeing

The final test involves taking the previous mock observations, and introducing seeing conditions. As described in Section 2.2.2, we convolve each spatial plane of our spectral or velocity data cube

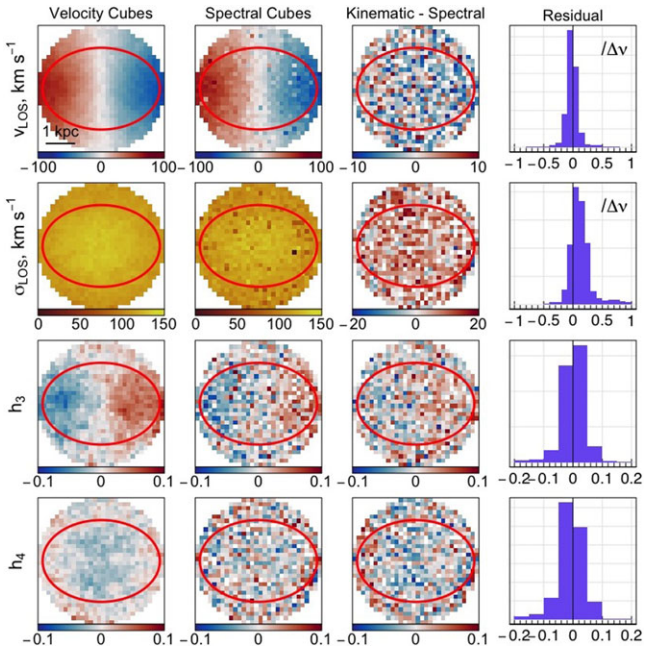

Figure 13. Case Study 4: The disk model built with E-MILES templates observed with an intrinsic telescope resolution of $\lambda_{LSF}^{telescope} = 3.61$ Å at a low redshift distance of $z = 0.0144$ with an added seeing condition of a Gaussian kernel with FWHM of 1 arcsec. Here we compare the output kinematic cubes to the kinematics fit with pPXF.

with a kernel to imitate the blurring effects of the atmosphere. This is done by specifying the `blur = T` parameter below, indicating that we would like the image to be blurred, as well as the size and shape of the convolution kernel. This is all done in the `observing_strategy()` function. For the following study, we use the following specification for the E-MILES and BC03 models, projecting each to both near and far distances with the varied seeing conditions:

```
observing_strategy(
    dist_kpc_per_arcsec = 0.3,
    inc_deg = 60, blur = T,
    fwhm = 1,
    psf="Gaussian"
    )

observing_strategy(
    dist_z = 0.3,
    inc_deg = 60,
    blur = T,
    fwhm = 2.8,
    psf="Moffat"
    )
```

The rest of the `telescope()` parameters remain consistent with the previous case study. As in the previous case, we test these observations at both high and low redshift distances for the young disc model with the two flavours of spectral templates. The results of these fits are demonstrated in Figs. 13 and 14, where we show the fitting results for the low redshift E-MILES galaxy and the high redshift BC03 galaxy. The remaining images are included in the final Appendix 1.4 (see Supplementary Material).

Even in the blurred images, as shown in Fig. 13, we can see that the kinematics between the `method = 'velocity'` and

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

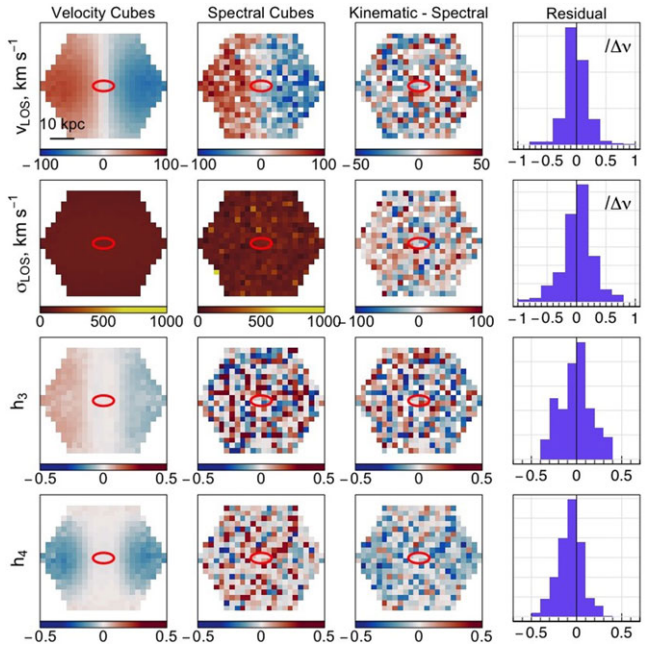

Figure 14. Case Study 4: The disk model built with BC03 templates observed with an intrinsic telescope resolution of $\lambda_{LSF}^{telescope} = 4.56$ Å at a high redshift distance of $z = 0.3$ with an added seeing conditions of a Moffat kernel with FWHM of 2.8 arcsec. Here we compare the output kinematic cubes to the kinematics fit with pPXF.

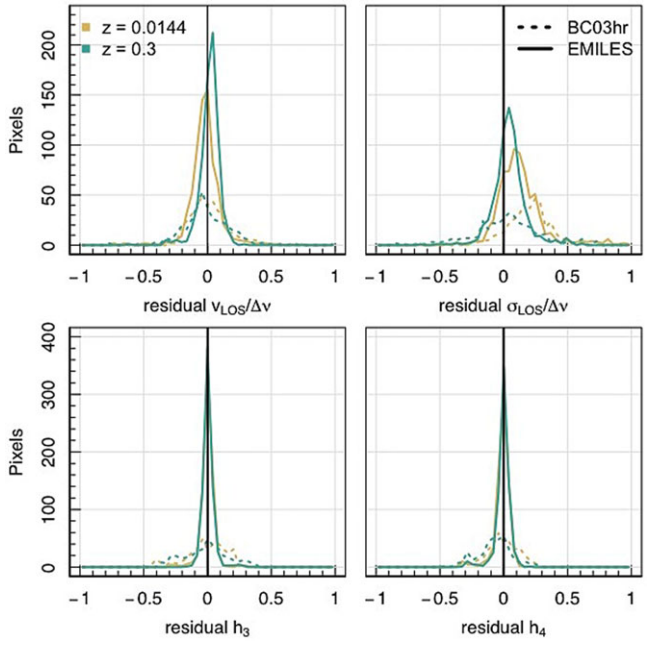

Figure 15. Case Study 4: The residual differences between the kinematic observations and the spectral fits for the low and high redshift disc models (in yellow and green respectively) built with E-MILES and BC03 templates. Most distributions are nicely centred around zero as we would expect, though we do see significantly broader distributions for the BC03hr models as well as some offset between the low redshift dispersion measured.

'spectral' cubes are closely comparable, with residuals nicely balanced around the zero point. With the BC03 system, we see a poorer recovery. Comparing the two directly using the histograms in Fig. 15, a hollow yellow bump is visible towards the positive residuals showing that the kinematic cubes provide an overestimate of the dispersion in comparison to the spectral cube fit with pPXF.

These are important tests to run in order to evaluate the success and flexibility of the code. Here we have taken each feature in turn and assessed how its addition affects the resulting kinematic image. We note that, within the extra-galactic community, the use of E-MILES templates for kinematic fitting is commonplace and as such it is good to see the consistency between input and output in the simulations that have been built and kinematics fit using the same set of stellar population synthesis models. Concern is raised with regards to the poorer fits found between the BC03 models fit using E-MILES templates, though we note that these differences are within the spectral resolution of the respective instruments used. Furthermore, he BC03 templates are evolutionary stellar population synthesis codes that are commonly used within the theory community for semi-analytic models and for stellar population fitting.

3.2. Web application

SimSpin is a flexible and modular code, as demonstrated in this article and the numerous examples available online. As the number of applications for mock simulation data grows with ever more resolved models of galaxy formation and evolution, it is important that access to the code is accessible and usable by a wide range of users, theorists and observers alike. In order to remove some of the barriers we perceive preventing users working with this code (including working with R, handling simulation data, or running large memory jobs locally), we have built a web application of SimSpin.[g]

The SimSpin web application has the same range of functionality as the R-package, without the necessity to download and install the package yourself. It is a performant React Single Page App communicating asynchronously with a RESTful API, hosted by Data Central. The application allows for instant data exploration via a dedicated viewer, where authenticated users can re-visit previous queries and share results with others. Generated FITS files can be directly downloaded for further exploration and quantification. All services are containerised and managed by docker compose, such that the project is easily re-deployable. The API is fully documented, and comes with an API Schema (adhering to the OpenAPI Specification) to aid users in calling the API from other services.

The SimSpin app removes the barrier of entry for novice astronomers, providing an accessible and time-saving tool for simulated galaxy visualisations. The API further removes a code language barrier as individuals can generate SimSpin queries using whichever language they choose. An example of this can be found within the documentation.[h]

4. Conclusion

In conclusion, we have presented a significant update to the mock observation code, SimSpin. We have demonstrated a number of new features available in the code v2.6.0, including the measurement of higher-order kinematics, the construction of spectral data cubes and the inclusion of gas component analysis. The code

[g] https://simspin.datacentral.org.au/app/.
[h] https://kateharborne.github.io/SimSpin/examples/query_the_API.html.

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press

now supports a wide number of different cosmological, hydrodynamical simulations, including EAGLE, ILLUSTRISTNG, MAGNETICUM, and HORIZONAGN. We further have containerised the code into a web application such that anyone can work with mock data, regardless of their coding language or computer specifications.

All of these features have been tested using unit testing, as well as the longer case study explorations that are presented in the results of this paper. In line with standard continuous integration procedures, we run all unit tests and require them to pass before any changes can be merged into the main branch of the code. We also require the code coverage (as measured by the number of lines within the code hit by the unit tests) to remain at approximately 90% for tests to pass. In the future, as more developers aim to expand the capabilities of the code, we may further implement another set of checks by core developers using the review system in place through GitHub.

The range of applications for this code is already beginning to be demonstrated within the literature for applications from designing corrections for the effects of seeing conditions (Harborne et al. 2020b), exploring the observational signatures of slow rotating systems formed in different ways (Lagos et al. 2022), or building machine learning models to explore the connection between intrinsic 3D shape and observable kinematics (Yong et al. in preparation). Of particular interest, with the ready incorporation of theory data with observational surveys, we hope to see similar data releases of simulated galaxies for comparison alongside observations (such as is being prepared for theMAGPI survey as described in Foster et al. 2021). With tools like SIMSPIN, we are enabling these comparisons to be made consistently, both between simulations and observations, but also consistently between the different simulations themselves.

Simulations provide us with the ability to explore the far reaches of space and time, while SIMSPIN now enables us to compare these simulations to our exquisite observations. The benefit of this is that, we our models we know the ground truth—projection effects can be modified by simply moving our observer, the atmosphere can be turned 'on' or 'off', and we can fast-forward through time to examine how a given system may change over the course of its life. Such information is undoubtedly useful for contextualising the results we find in observations, as well as to improve existing sub-grid recipes within simulations in line with this. The future of mock observables is bright.

Acknowledgements. We would like to thank the anonymous reviewer for the thoughtful comments on this manuscript that led to significant memory saving functionality in the code. Further thanks is due to Adriano Poci for helpful discussions about the measurement of higher-order kinematics. KH acknowledges funding from CL's Discovery Project, DP210101945, funded by the Australian Government. AS acknowledges support through the summer internship programme from the International Centre for Radio Astronomy Research (ICRAR) and the Pawsey Supercomputing Centre.

This work has been made possible through the Astronomy Data and Computing Services (ADACS). This research was conducted under the Australian Research Council Centre of Excellence for All Sky Astrophysics in 3 Dimensions (ASTRO 3D), through project number CE170100013.

Parts of this research, including the construction of N-body models for the case study analysis, were undertaken on Magnus at the Pawsey Supercomputing Centre in Perth, Australia.

Supplementary material. The supplementary material for this article can be found at http://doi.org/10.1017/pasa.2023.47

Data Availability. The data that support the findings of this study are openly available in the public repository `kateharborne/project_simspin` at https://github.com/kateharborne/project_simspin. This includes the raw data and code used to generate the plots for the paper, as well as the source code for the paper. The simulations, from which this data has been consolidated, are also available in this repository as HDF5 files. Simulation models have been built using GalIC (Yurin & Springel 2014) and Gadget2 (Springel 2005). Several public, open-source codes have been used in this work to generate the data. These include SimSpin (https://github.com/kateharborne/SimSpin), magicaxis (https://github.com/asgr/magicaxis) and pPXF (https://pypi.org/project/ppxf/). For pPXF fits, we also make use of the EMILES_PADOVA_CH templates, documented in Vazdekis et al. (2016) and available online at http://research.iac.es/proyecto/miles/pages/spectral-energy-distributions-seds/e-miles.php.

References

Abdurro'uf, et al. 2022, ApJS, 259, 35
Allgood, B., et al. 2006, MNRAS, 367, 1781
Bacon, R., et al. 1995, A&AS, 113, 347
Bacon, R., et al. 2001, MNRAS, 326, 23
Bacon, R., & Monnet, G. 2017, Optical 3D-Spectroscopy for Astronomy (Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA), doi: 10.1002/9783527674824
Barrientos Acevedo, D., et al. 2023, MNRAS, 524, 907
Bassett, R., & Foster, C. 2019, MNRAS, 487, 2354
Beck, A. M., et al. 2016, MNRAS, 455, 2110
Bendo, G. J., & Barnes, J. E. 2000, MNRAS, 316, 315
Borrow, J., & Kelly, A. J. 2021, arXiv, arXiv:2106.05281
Bottrell, C., & Hani, M. H. 2022, MNRAS, 514, 2821
Broyden, C. G. 1970, IMA JAM, 6, 76
Bruzual, G., & Charlot, S. 2003, MNRAS, 344, 1000
Bryant, J. J., et al. 2020, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 11447, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 1144715
Bundy, K., et al. 2015, ApJ, 798, doi: 10.1088/0004-637X/798/1/7
Camps, P., & Baes, M. 2020, A&C, 31, 100381
Cappellari, M. 2002, MNRAS, 333, 400
Cappellari, M., et al. 2007, MNRAS, 379, 418
Cappellari, M., et al. 2011, MNRAS, 413, 813
Cappellari, M. 2017, MNRAS, 466, 798
Cappellari, M., & Emsellem, E. 2004, PASP, 116, 138
Chabrier, G. 2003, PASP, 115, 763
Crain, R. A., et al. 2015, MNRAS, 450, 1937
Croom, S. M., et al. 2012, MNRAS, 421, 872
Croom, S. M., et al. 2021, MNRAS, 505, 2247
de Zeeuw, P. T., et al. 2002, MNRAS, 329, 513
Doi, M., et al. 2010, AJ, 139, 1628
Dolag, K., Hansen, F. K., Roncarelli, M., & Moscardini, L. 2005, MNRAS, 363, 29
Dubois, Y., et al. 2014, MNRAS, 444, 1453
Emsellem, E., et al. 2004, MNRAS, 352, 721
Fletcher, R. 1970, CJ, 13, 317
Forsythe, MALCOLM, M. A., & Moler, M. A. 1977, Computer Methods for Mathematical Computations (Wiley)
Foster, C., et al. 2021, PASA, 38, e031
Fukugita, M., et al. 1996, AJ, 111, 1748
Goldfarb, D. 1970, MC, 24, 23
Harborne, K. E., et al. 2020 b, MNRAS, 497, 2018
Harborne, K. E., Power, C., & Robotham, A. S. G. 2020 a, PASA, 37, doi: 10.1017/pasa.2020.8
Harborne, K. E., Power, C., Robotham, A. S. G., Cortese, L., & Taranu, D. S. 2019, MNRAS, 483, 249
Hogg, D. W. 1999, arXiv e-prints, astro
Jesseit, R., Cappellari, M., Naab, T., Emsellem, E., & Burkert, A. 2009, MNRAS, 397, 1202
Jesseit, R., Naab, T., Peletier, R. F., & Burkert, A. 2007, MNRAS, 376, 997

Jiménez, E., Lagos, C. d. P., Ludlow, A. D., & Wisnioski, E. 2023, MNRAS, 524, 4346
Katz, N., Weinberg, D. H., & Hernquist, L. 1996, ApJS, 105, 19
Lagos, C. d. P., et al. 2022, MNRAS, 509, 4372
Li, H., et al. 2018, MNRAS, 473, 1489
Ludlow, A. D., et al. 2019, MNRAS, 488, 3663
Ludlow, A. D., Fall, S. M., Wilkinson, M. J., Schaye, J., & Obreschkow, D. 2023, arXiv e-prints, arXiv:2306.05753
Metzler, C. A., & Evrard, A. E. 1994, ApJ, 437, 564
Moffat, A. F. J. 1969, A&A, 3, 455
Naab, T., et al. 2014, MNRAS, 444, 3357
Nanni, L., et al. 2022, MNRAS, 515, 320
Nanni, L., et al. 2023, MNRAS, 522, 5479
Nelson, D., et al. 2019, MNRAS, 490, 3234
Oser, L., Ostriker, J. P., Naab, T., Johansson, P. H., & Burkert, A. 2010, ApJ, 725, 2312
Pillepich, A., et al. 2018, MNRAS, 473, 4077
Pillepich, A., et al. 2019, MNRAS, 490, 3196
Poci, A., et al. 2021, A&A, 647, A145
Price, D. J. 2007, PSSA, 24, 159
Robotham, A. S. G., et al. 2020, ProSpect: Generating Rapid Spectral Energy Distributions with Complex Star Formation and Metallicity Histories
Robotham, A. S. G., Taranu, D. S., Tobar, R., Moffett, A., & Driver, S. P. 2017, MNRAS, 466, 1513
Sarmiento, R., et al. 2023, A&A, 673, A23
Schaller, M., et al. 2015, MNRAS, 454, 2277
Schaye, J., et al. 2015, MNRAS, 446, 521
Schulze, F., et al. 2018, MNRAS, 480, 4636
Shanno, D. F. 1970, MC, 24, 647
Springel, V. 2005, MNRAS, 364, 1105
Springel, V., et al. 2018, MNRAS, 475, 1, 676, 475, 676
Teklu, A. F., et al. 2015, ApJ, 812, 29
van de Sande, J., et al. 2019, MNRAS, 484, 869
van der Marel, R. P., & Franx, M. 1993, ApJ, 407, 525
Vazdekis, A., Koleva, M., Ricciardelli, E., Röck, B., & Falcón-Barroso, J. 2016, MNRAS, 463, 3409
Wendland, H. 1995, ACM, 4, 389
Wilkinson, M. J., et al. 2023, MNRAS, 519, 5942
Yurin, D., & Springel, V. 2014, MNRAS, 444, 62
Zhu, L., et al. 2022, A&A, 660, A20

https://doi.org/10.1017/pasa.2023.47 Published online by Cambridge University Press