Fast GPU-Powered and Auto-Differentiable Forward Modeling of IFU Data Cubes

Ufuk Çakır*
Interdisciplinary Center
for Scientific Computing,
University of Heidelberg,
Im Neuenheimer Feld 205,
D-69120 Heidelberg
mail@cakir-ufuk.de

Anna Lena Schaible
Interdisciplinary Center
for Scientific Computing,
University of Heidelberg,
Im Neuenheimer Feld 205,
D-69120 Heidelberg
annalena.schaible@iwr.uni-heidelberg.de

Tobias Buck
Interdisciplinary Center
for Scientific Computing,
University of Heidelberg,
Im Neuenheimer Feld 205,
D-69120 Heidelberg
tobias.buck@uni-heidelberg.de

Abstract

We present RUBIX, a fully tested, well-documented, and modular Open Source tool developed in JAX, designed to forward model IFU cubes of galaxies from cosmological hydrodynamical simulations. The code automatically parallelizes computations across multiple GPUs, demonstrating performance improvements over state-of-the-art codes by a factor of 600. This optimization reduces compute times from hours to only seconds. RUBIX leverages JAX's auto-differentiation capabilities to enable not only forward modeling but also gradient computations through the entire pipeline paving the way for new methodological approaches such as e.g. gradient-based optimization of astrophysics model parameters. RUBIX is open-source and available on GitHub[2].

1 Motivation

In the field of astrophysics, researchers are divided into two main groups: observers and theorists. Observers build and operate advanced instruments and telescopes, such as the James Webb Space Telescope (JWST) and the Very Large Telescope (VLT), to collect empirical data from distant galaxies and stars by counting photons. Integral Field Unit (IFU) spectroscopy is one key observational technique that produces datacubes with spatially resolved spectra. Theorists, on the other hand, develop and refine physical equations to model the Universe's behavior. They use high-end supercomputers to run cosmological simulations, to replicate the conditions of the early universe. These simulations help test the implications of various physical theories and require advanced computational techniques and statistical analysis. One significant challenge in astrophysics is bridging the gap between observational data and theoretical models. Forward modeling techniques, which translate simulation outputs into observable data, are crucial for effective collaboration. The advances of Machine Learning

*Corresponding Author: Now at Intelligent Earth UKRI Centre for Doctoral Training in AI for the Environment, University of Oxford

[2] https://github.com/ufuk-cakir/rubix

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

models are hugely influenced by the improvements of hardware that has an architecture that works well with the calculations performed in ML applications: the GPU. They are extremely well suited to perform calculations in parallel, hence implementing a IFU forward model code that works on GPUs is a major advantage to current state-of-the-art codes and will enable us to produce a sufficient number of samples required for statistical analysis, which was so far the bottleneck for Machine Learning (ML) applications. Additionally, with our JAX implementation we are able to compute gradients needed to perform optimization in the context of ML and Simulation Based Inference. This paper aims to bridge the gap between observers and theorists by introducing a forward modelling of mock IFUs: `RUBIX` is written in JAX, runs natively parallel on multiple GPUs and leverages performance improvements from just-in-time compilation using XLA.

2 Related Work

There are several codes that can forward model IFU data that are commonly used in astrophysics research. One of the most popular codes is SimSpin [3], which is written in R and is a CPU only package that takes in a simulation of a galaxy and produces mock IFU observations. The user can freely choose any instrument configuration and spectral library to create mock observations. There is extensive documentation and examples available, which makes the usage of the package very user-friendly. Another code called GalCraft [10] generates mock IFU data cubes of the Milky Way (MW). GalCraft uses the mock stellar catalog that is based on the analytical chemodynamical model of [8]. The analytical model predicts the joint distribution of position, velocity, age, extinction, photometric magnitude and the chemical abundances of stars in the MW, which is then used to produce mock IFU data cubes. In [7], the authors emulated observation data from the MaNGA survey [2] using IllustrisTNG data [6] to generate mock observations. Previously, [1] emulated 893 MaNGA observations using the *RealSimIFS* code from data of the TNG50 simulation [6, 5]. A similar project of [4] produced a catalog of around 1000 unique mock IFU observation to mimic the MaNGA primary sample – again using data from the TNG50 simulation. However, all current codes are CPU only use and have no option of calculating the gradient of the forward modelling process with respect to the input parameters - hence limiting their applicability within the context of ML.

3 RUBIX Codebase

The `RUBIX` pipeline is a modular and efficient framework for forward modeling IFU data from cosmological simulations, leveraging the power of JAX for high-performance computing. `RUBIX` is implemented as a linear pipeline, where each function sequentially transforms the input data, ensuring that the framework remains extremely modular and easily extensible for future developments. `RUBIX` utilizes multiple GPUs for parallel computation and significantly reduces processing time. The Input Handler extracts and transforms relevant star and gas particle information from cosmological simulation data into a unique data file, which is the input for the `RUBIX` pipeline. The first step in the pipeline is to orientate the galaxy in the field of view, following the specifications provided in the configuration file. Next, the particles are assigned to the IFU spaxels, accounting for telescope-specific configurations. The key part is the spectra calculation. Each star spectrum is calculated as a lookup from a simple stellar population (SSP) library. Then the spectra are Doppler shifted based on the galaxy distance and line-of-sight velocity of each stellar particle. Additional resampling is performed to match the wavelength grid of the observed telescope. Afterwards, the stellar spectra in each spaxel are summed up. To simulate observational effects, we apply a point-spread function (PSF) and line-spread function (LSF) convolution and add realistic noise. The entire pipeline is configurable using JSON. Each pipeline run starts with a JSON or Python dictionary, where the user chooses all the hyperparameters, i.e SSP library, galaxy distance and orientation, telescope, etc.

4 Results

Qualitative Analysis To verify the output, `RUBIX` is executed for different Subhalos from the IllustrisTNG simulation using the IllustrisAPI. For a set of galaxies from the TNG50-1 simulation, snapshot 99, mock observations are created with a MUSE instrument configured with `fov=20` and a Gaussian PSF and LSF. The `Mastar_CB19_SLOG_1_5` SSP template [9] is employed to compute the stellar spectra, and `RUBIX` is executed on eight NVIDIA A100 GPUs. The mock MUSE observations

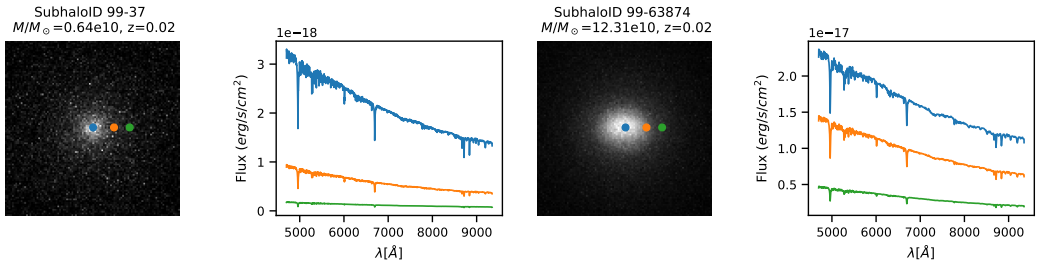


Figure 1: ***MUSE mock observations*** – for different Subhalos, the total flux in each pixel is shown as an image representation on the left. On the right, the spectra of three different spaxels are plotted.

are illustrated in Figure 1. The galaxies were chosen to have an increasing mass, with a dwarf galaxy on the left and a massive spherical galaxies on the right. For each galaxy, an image representation is provided on the left, where the total flux in each spaxel is summed to produce a two-dimensional array. The right column presents the spectra (in units of erg/s/cm²) for three different spaxels with increasing distance from the galactic center. Spectra from the center of the galaxy exhibit higher flux compared to those from the outskirts, which is expected due to the higher density of stars in the galaxy's center. The shapes of the spectra differ significantly; spectra from the outskirts tend to be flatter. In general we can observe that `RUBIX` can reproduce the trends that we expect.

Speed comparison The primary objective of this paper is to highlight the methodological improvements `RUBIX` provides for the forward modeling process. In Figure 2, the compute times of different codes are compared. According to [10], the authors state that "for a typical MUSE FoV containing $6 \cdot 10^6$ particles, the execution time spent with a 24-core CPU (2.50GHz) is 1.4 hours." This result serves as a benchmark to contextualize the execution time of `RUBIX`. A galaxy with a comparable number of particles ($6 \cdot 10^6$) is forward modeled both on the CPU and GPU using `RUBIX`. Running `RUBIX` on a 24-core CPU (AMD Epyc 7452, 2.35 GHz) takes 123.57 seconds, representing a 43.7-fold improvement over the Galcraft code. When the mock observation is computed on a single NVIDIA A-100 GPU, the execution time is reduced to 8.58 seconds, which is 600 times faster than Galcraft and 14.4 times faster than the same `RUBIX` code executed on the CPU. Despite the clear performance improvements, we should take the benchmark comparisons with caution, because we use different hardware configurations. GPUs and CPUs have different architecture and GalCraft does not share the exact same methodology. Despite these differences, the comparison still offers a valuable general trend. One significant reason for `RUBIX`'s superior speed is its efficient implementation. In `RUBIX`, instead of naively looping over the particles, every function is vectorized using `vmap`. This approach leverages XLA to fuse operations together, resulting in substantial speed improvements.

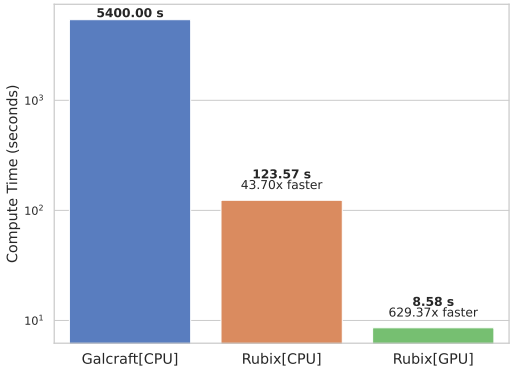
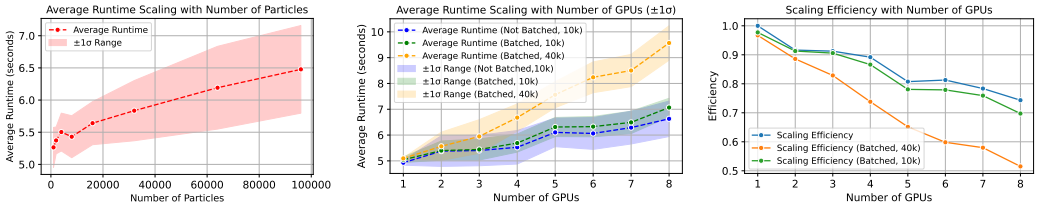


Figure 2: ***Speed comparison*** – the execution time of different codes are compared. Note that the y-axis is logarithmic.

Strong Scaling In Figure 3(a), the average runtime is plotted against the number of particles. At each number of particles, the runtime is measured five times to get some statistics. The red shaded area represents the $\pm 1\sigma$ range, indicating the variability in the runtime measurements. One can observe that as the number of particles increases, the average runtime also increases, but not linearly. This indicates that `RUBIX` does not have perfect strong scaling, which may be caused by the communication overhead between the GPUs.

Weak Scaling To evaluate how compute time scales with the number of GPUs, `RUBIX` is initially run with 10,000 particles on a single GPU, and the runtime is measured. Next, the number of particles is doubled, and the code is executed on two GPUs, continuing this process until the maximum number



(a) ***Strong Scaling*** (b) ***Weak Scaling*** (c) ***Scaling Efficiency***

Figure 3: ***Scaling plots*** – (a) Strong Scaling: Increasing particle size, while keeping number of GPUs fixed (8 NVIDIA A100 GPUs). (b) Average runtime of different `RUBIX` runs, where we proportionally increase particle size and number of available GPUs, such that the workload per GPU remains constant. (c) Scaling efficiency calculated as the ratio of the runtime with one GPU to the runtime with multiple GPUs.

of GPUs is reached, which in this case is eight NVIDIA A100 GPUs. In Figure 3(b), the average runtime of different `RUBIX` runs is measured. Each run is repeated 5 times to get some statistics, and the 1σ area is shaded in the background (blue: starts with 10,000 particles; green: split data into four batches on each GPU; orange: starts with 40,000 particles and batching). Ideally, in a best-case scenario, the compute time should remain constant as both the workload (number of particles) and the compute resources (number of GPUs) increase proportionally. This would demonstrate perfect scaling. From Figure 3(b) we can clearly see that the scaling is not perfect. The runtime increases slowly with the number of GPUs, indicating that the computational work is not distributed equally between the GPUs or that communication overheads are still large in `RUBIX`.

Scaling Efficiency To make this more quantitatively, we can measure the scaling efficiency as: Scaling Efficiency $= \frac{T_1}{T_N}$ where T_1 is the runtime with one GPU and T_N is the runtime with N GPUs. Ideally, this scaling efficiency should be close to one. In Figure 3(c) we clearly see that the scaling efficiency decreases with increasing number of GPUs, which means that the scaling is not optimal. One major factor is the communication overhead between GPUs, which can become significant as more GPUs are added. Additionally, the efficiency of load balancing can decrease with more GPUs. Furthermore, the complexity of managing more GPUs can introduce inefficiencies in the parallelization process, such as increased latency in coordinating tasks and distributing data evenly among the GPUs. This indicates that `RUBIX` is not yet fully optimized and requires further improvements. One significant bottleneck might be the current implementation of `pmap` and `jit`. In the current version of `RUBIX`, only the datacube calculation inside the pipeline is parallelized across the GPUs using `pmap`. However, during the pipeline assembly, all the functions are concatenated, and the final function is just-in-time compiled using `jit`. There is a known issue in JAX warning users that using `jit` on a `pmap`-function can lead to inefficient data movement, as it essentially collects all data onto a single device. This issue is discussed in detail on the official JAX GitHub page[3].

5 Conclusions and Limitations

`RUBIX` represents a significant leap forward in computational efficiency and flexibility for modeling IFU observations from cosmological hydrodynamical simulations. Its ability to rapidly process large-scale simulations and its potential for future enhancements makes it a powerful tool for astrophysical research. The combination of high performance and open-source accessibility underscores the contribution of `RUBIX` to the field, facilitating innovation and collaboration within the scientific community. Despite its impressive performance, there remains potential for further optimization, i.e. further profiling is required. Apart from speed improvements, there are additional features that will be implemented into `RUBIX`. Some of those include:

- **Gas Modeling** – Incorporating detailed models for interstellar gas will allow `RUBIX` to simulate and analyze gas emission lines.

[3] `https://github.com/google/jax/issues/2926`

- **Dust Modeling** – Adding support for dust attenuation models will provide more realistic mock observations, that should closer relate to real observations.
- **Radiative Transfer** – Implementing advanced radiative transfer models will enhance the precision of the `RUBIX` simulations. This will allow for a more realistic representation of how light propagates through various media. However, this needs to be implemented in pure JAX, which can be a quite challenging task.

With these additional features `RUBIX` will be ideally suited to tackle key scientific machine learning tasks in astrophysics, such as performing SBI inference of fundamental galaxy parameters with high-dimensional complex observational data, perform Bayesian model comparison, do gradient based optimization tasks on the forward modelling pipeline and incorporate the differentiable forward model `RUBIX` into machine learning architectures to train them end-to-end, e.g. build hybrid NN encoder-physics-based-decoder architectures. As such, we think that `RUBIX` provides the astrophysical community with a unique, versatile and new methodological approach to perform downstream scientific tasks.

Broader impact statement

The authors are not aware of any immediate ethical or societal implications of this work. This work purely aims to aid scientific research and proposes a method of using a pipeline of forward modelling IFU data cubes to learn about galaxy formation and evolution.

Acknowledgments and Disclosure of Funding

The authors thank the Scientific Software Center at Heidelberg University for the support. This work is funded by the Carl-Zeiss-Stiftung through the NEXUS programm.

References

[1] Connor Bottrell and Maan H Hani. Realistic synthetic integral field spectroscopy with realsim-ifs. *Monthly Notices of the Royal Astronomical Society*, 514(2):2821–2838, June 2022.

[2] Kevin Bundy, Matthew A. Bershady, David R. Law, Renbin Yan, Niv Drory, Nicholas MacDonald, David A. Wake, Brian Cherinka, José R. Sánchez-Gallego, Anne-Marie Weijmans, Daniel Thomas, Christy Tremonti, Karen Masters, Lodovico Coccato, Aleksandar M. Diamond-Stanic, Alfonso Aragón-Salamanca, Vladimir Avila-Reese, Carles Badenes, Jésus Falcón-Barroso, Francesco Belfiore, Dmitry Bizyaev, Guillermo A. Blanc, Joss Bland-Hawthorn, Michael R. Blanton, Joel R. Brownstein, Nell Byler, Michele Cappellari, Charlie Conroy, Aaron A. Dutton, Eric Emsellem, James Etherington, Peter M. Frinchaboy, Hai Fu, James E. Gunn, Paul Harding, Evelyn J. Johnston, Guinevere Kauffmann, Karen Kinemuchi, Mark A. Klaene, Johan H. Knapen, Alexie Leauthaud, Cheng Li, Lihwai Lin, Roberto Maiolino, Viktor Malanushenko, Elena Malanushenko, Shude Mao, Claudia Maraston, Richard M. McDermid, Michael R. Merrifield, Robert C. Nichol, Daniel Oravetz, Kaike Pan, John K. Parejko, Sebastian F. Sanchez, David Schlegel, Audrey Simmons, Oliver Steele, Matthias Steinmetz, Karun Thanjavur, Benjamin A. Thompson, Jeremy L. Tinker, Remco C. E. van den Bosch, Kyle B. Westfall, David Wilkinson, Shelley Wright, Ting Xiao, and Kai Zhang. Overview of the SDSS-IV MaNGA Survey: Mapping nearby Galaxies at Apache Point Observatory. , 798(1):7, January 2015.

[3] Katherine Harborne. Simspin: Kinematic analysis of simulated galaxies, 2023. Publications of the Astronomical Society of Australia, Volume 40, article id. e048, Oct 2023.

[4] Lorenza Nanni, Daniel Thomas, James Trayford, Claudia Maraston, Justus Neumann, David R Law, Lewis Hill, Annalisa Pillepich, Renbin Yan, Yanping Chen, and Dan Lazarz. iMaNGA: mock MaNGA galaxies based on IllustrisTNG and MaStar SSPs – I. Construction and analysis of the mock data cubes. *Monthly Notices of the Royal Astronomical Society*, 515(1):320–338, 06 2022.

[5] Dylan Nelson, Annalisa Pillepich, Volker Springel, Rüdiger Pakmor, Rainer Weinberger, Shy Genel, Paul Torrey, Mark Vogelsberger, Federico Marinacci, and Lars Hernquist. First results

from the tng50 simulation: galactic outflows driven by supernovae and black hole feedback. *Monthly Notices of the Royal Astronomical Society*, 490(3):3234–3261, August 2019.

[6] Annalisa Pillepich, Dylan Nelson, Volker Springel, Rüdiger Pakmor, Paul Torrey, Rainer Weinberger, Mark Vogelsberger, Federico Marinacci, Shy Genel, Arjen van der Wel, and Lars Hernquist. First results from the tng50 simulation: the evolution of stellar and gaseous discs across cosmic time. *Monthly Notices of the Royal Astronomical Society*, 490(3):3196–3233, September 2019.

[7] Regina Sarmiento, Marc Huertas-Company, Johan H. Knapen, Héctor Ibarra-Medel, Annalisa Pillepich, Sebastián F. Sánchez, and Alina Boecker. Mangia: 10 000 mock galaxies for stellar population analysis. *Astronomy amp; Astrophysics*, 673:A23, April 2023.

[8] Sanjib Sharma, Michael R Hayden, and Joss Bland-Hawthorn. Chemical enrichment and radial migration in the Galactic disc – the origin of the [Fe] double sequence. *Monthly Notices of the Royal Astronomical Society*, 507(4):5882–5901, 07 2021.

[9] S. F. Sánchez, J. K. Barrera-Ballesteros, E. Lacerda, A. Mejía-Narvaez, A. Camps-Fariña, Gustavo Bruzual, C. Espinosa-Ponce, A. Rodríguez-Puebla, A. R. Calette, H. Ibarra-Medel, V. Avila-Reese, H. Hernandez-Toledo, M. A. Bershady, M. Cano-Diaz, and A. M. Munguia-Cordova. Sdss-iv manga: pypipe3d analysis release for 10,000 galaxies. *The Astrophysical Journal Supplement Series*, 262(2):36, sep 2022.

[10] Zixian Wang, Michael R. Hayden, Sanjib Sharma, Jesse van de Sande, Joss Bland-Hawthorn, Sam Vaughan, Marie Martig, and Francesca Pinna. The milky way in context: Building an integral-field spectrograph data cube of the galaxy, 2023.

A Appendix / supplemental material