The Journal of Open Source Software

Starmatrix: Modelling nucleosynthesis of galactic chemical elements

Juanjo Bazán [1] **and Mercedes Mollá** [1]

1 Departamento de Investigación Básica, CIEMAT, Avda. Complutense 40, E-28040, Madrid, Spain

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 01 June 2022
Published: unpublished

Summary

`Starmatrix` is a Python package for computing the chemical contribution to the interstellar medium ejected by simple stellar populations (SSPs).

One of the key ingredients of galactic chemical evolution (GCE) models is the nucleosynthetic contribution returned to the interstellar medium by the evolving stellar populations and supernovae. These yields vary depending on the age and metallicity of the stars and are combined following the mass distribution of stars to represent a SSP, stars formed at the same time and having the same initial element composition. Repeating this process, adjusting for the the star formation at each time step, is one of the main mechanisms of GCE models.

`Starmatrix` reads a single configuration file and calculates the combined contributions of chemical elements ejected to the interstellar medium during the lifetime of stars in the provided mass range. For each mass step an ejections matrix is computed for these fifteen elements: H, D, ^{3}He, ^{4}He, ^{12}C, ^{16}O, ^{14}N, ^{13}C, ^{20}Ne, ^{24}Mg, ^{28}Si, ^{32}S, ^{40}Ca, ^{56}Fe, and also all neutron-rich CNO isotopes as only one group.

Using explicit (and configurable) values for *solar abundances*, *metallicity (z)*, *ejection rates* and *Initial Mass Function (IMF)*, `Starmatrix` calculates matrices Q_{ij} of masses of elements `j` converted to element `i` and ejected to the galactic medium, integrating for a given range of stellar masses, and accounting for Supernovae (SNe) of types *I* and *II*. Based on the *Matrices Q formalism* (Ferrini et al., 1992; Portinari L., 1998), an output file is generated containing matrices linking any ejected species to all its different nucleosynthetic sources.

Statement of need

Calculating the contributions of chemical elements from different stars in several mass ranges is a necessary step for galactic chemical evolution models (Mollá et al., 2017). Usually this is done internally and the chemical yields are convoluted with a star formation history (SFH) inside the model, making it difficult to find reusable open-sourced code for calculating the yield for a simple stellar population. The central module in Chempy (Rybizki, Jan et al., 2017) and the SYGMA module from the NUPYCEE framework (Ritter et al., 2018) are exceptions, but being part of bigger codebases, they are not available as standalone libraries, not listed in the official Python Package Index, and learning how to install and use them is not simple.

`Starmatrix` isolates the SSP yield calculation step as an open-source permissively-licensed Python implementation, in a modular and flexible way that can be used to provide detailed datasets to be used as input for galactic chemical evolution models, to compare the validity of different yield sets from the literature or to assess different nucleosynthesis modeling assumptions (Bazán et al., 2003).

Features

In order to calculate Q matrices for the whole range of selected stellar masses, the code estimates the stellar lifetime for the minimum and maximum masses and obtains a time interval that is then divided into a fixed number of steps to integrate. The total time steps can be configured but also the integration step can be forced via settings to be constant in t or in $log(t)$. For each time interval of the integration, the supernova rates are obtained using the corresponding IMF for core-collapse supernovae and a configurable delay time distribution for type Ia supernovae. Then, the time step is converted back into stellar mass intervals and a Q matrix is calculated for that mass interval weighing it by the selected IMF to create the final output data.

All quantities in the Q matrix are calculated on the basis of the ejected mass from stars, using for it the dataset of yields per stellar mass that is entered as input for the code. The reference unit used for stellar mass is the solar mass. The code is prepared to include any sets of stellar yields from the literature but if none is declared, a default dataset for solar metallicity and with low and intermediate mass star yields from Gavilán, M. et al. (2006) and yields of massive stars from Chieffi & Limongi (2004) will be used.

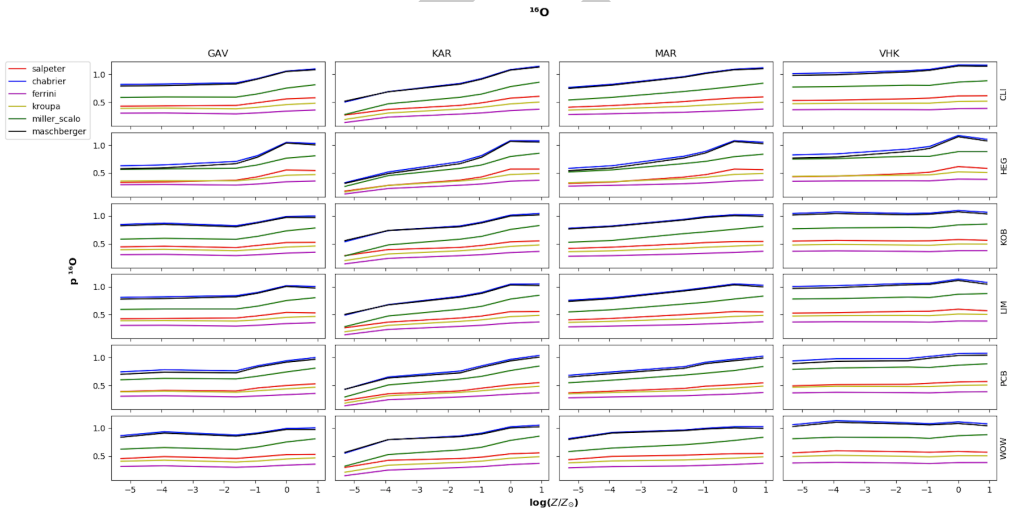

Figure 1: A sample plot using the output data from several `Starmatrix` runs. Lines show true yields p of oxygen using different datasets for low and intermediate mass yields (columns) and for massive stars yields (rows), for a range of metallicity values (bottom horizontal axis) and different IMFs available as Starmatrix' settings.

Configurable parameters include metallicity, mass range for Q matrices, fraction of binary systems, lower and upper mass limits for the IMF, total time steps for integration, and yield correction factors. The code also includes a variety of options for IMFs, `solar abundances` data, Supernovae `yields`, and `Delay Time Distributions` from the literature to choose from. A complete list of available options can be found in the documentation. To make the code more reusable and to allow it to be integrated in or called by other codebases, Starmatrix defines a base class for each of these parameters to be subclassed if needed so that custom options can be easily added programmatically. The code also includes extensive test coverage.

`Starmatrix` depends on `numpy` (Harris et al., 2020) and `scipy` (Virtanen et al., 2020). It is released with an open-source licence (MIT), available as a public git repository, distributed through the Python Package Index, and is easily installable using the standard `pip` package manager.

References

Bazán, J. J., Mollá, M., & Cerviño, M. (2003). Dispersion in modeled abundances. In J. Gallego, J. Zamorano, & N. Cardiel (Eds.), *Highlights of Spanish Astrophysics III* (pp. 35–38). Springer Netherlands. https://doi.org/10.1007/978-94-017-1778-6_6

Chieffi, A., & Limongi, M. (2004). Explosive yields of massive stars from Z=0 to Z=Zsun. *The Astrophysical Journal*, *608*(1), 405–410. https://doi.org/10.1086/392523

Ferrini, F., Matteucci, F., Pardi, C., & Penco, U. (1992). Evolution of spiral galaxies. I. Halo-disk connection for the evolution of the Solar neighborhood. *Astrophysical Journal*, *387*, 138. https://doi.org/10.1086/171066

Gavilán, M., Mollá, M., & Buell, J. F. (2006). Low and intermediate mass star yields - II. The evolution of nitrogen abundances. *Astronomy & Astrophysics*, *450*(2), 509–521. https://doi.org/10.1051/0004-6361:20053590

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Mollá, M., Díaz, Á. I., Ascasibar, Y., & Gibson, B. K. (2017). Galaxy chemical evolution models: the role of molecular gas formation. *Monthly Notices of the Royal Astronomical Society*, *468*(1), 305–318. https://doi.org/10.1093/mnras/stx419

Portinari L., B. A., Chiosi C. (1998). Galactic chemical enrichment with new metallicity dependent stellar yields. *Astronomy & Astrophysics*, *334*, 505–539.

Ritter, C., Côté, B., Herwig, F., Navarro, J. F., & Fryer, C. L. (2018). SYGMA: Stellar yields for galactic modeling applications. *The Astrophysical Journal Supplement Series*, *237*(2), 42. https://doi.org/10.3847/1538-4365/aad691

Rybizki, Jan, Just, Andreas, & Rix, Hans-Walter. (2017). Chempy: A flexible chemical evolution model for abundance fitting - do the Sun's abundances alone constrain chemical evolution models? *Astronomy & Astrophysics*, *605*, A59. https://doi.org/10.1051/0004-6361/201730522

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1. 0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2