Visualizing Nuclear Reaction Rates and Constructing Networks with pynucastro

Donald E. Willcox (LBNL), Adam Jacobs (MSU), Xinlong Li (SBU), and Michael Zingale (SBU)

Lawrence Berkeley National Laboratory (LBNL), Michigan State University (MSU), Stony Brook University (SBU)

Abstract

pynucastro is developed to meet both pedagogical and research needs in the field of nuclear astrophysics by providing a Python interface to nuclear reaction rate databases (including the JINA Reaclib nuclear reaction rate database). It is meant for both **interactive exploration** of reaction rates (through Jupyter notebooks) and for creating **reaction networks for simulation codes**.

Getting pynucastro

- pynucastro is freely available under the **BSD 3-Clause open source** license.
- For the companion **software paper**, see [1].
- Visit us on GitHub to **download** pynucastro or report **issues** and **request features**:

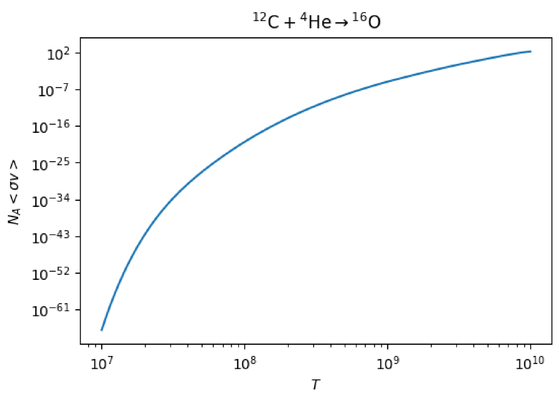
`https://github.com/pynucastro/pynucastro`

- For pynucastro **documentation** and examples, visit the pynucastro website:

`https://pynucastro.github.io/pynucastro`

Working with Reaclib

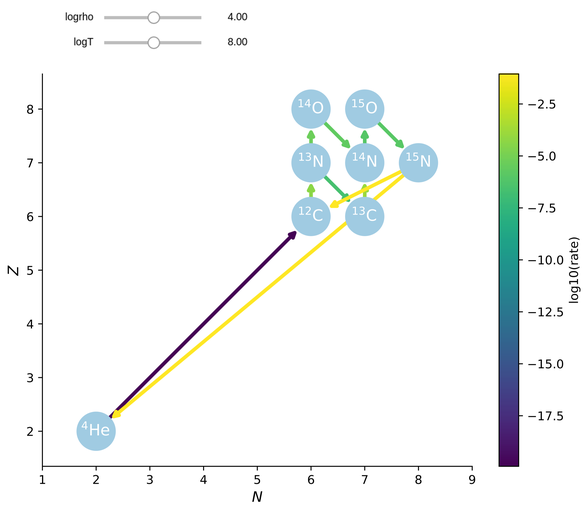
pynucastro can interpret reaction rates parameterized in the standard Reaclib formats, storing each rate internally using the **Rate** class. **Rate** interprets the Reaclib format and knows how to evaluate the rate as a function of temperature.



Creating a Network

The **Library** class allows pynucastro to work easily with files containing multiple rates, including Reaclib snapshots. We use **Library** to obtain the CNO rates linking our desired nuclei in just a few lines of Python in the following Jupyter notebook:

```
%matplotlib inline
import pynucastro as pyna

mylibrary = pyna.rates.Library('20180201ReaclibV2.22')

all_nuclei = ["p", "he4", "c12", "n13", "c13", "o14", "n14", "o15", "n15"]

cno_library = mylibrary.linking_nuclei(all_nuclei, with_reverse=False)

cno_network = pyna.networks.PythonNetwork(libraries=cno_library)

cno_network.write_network('network_module.py')

comp = pyna.Composition(cno_network.get_nuclei())
comp.set_solar_like()

re = pyna.Explorer(cno_network, comp)
re.explore()
```



Using the **Composition** and **Explorer** classes, we also construct an interactive graph of the network with the value of each rate indicated by its arrow color with variable density and temperature.

Python Code Generation

The **PythonNetwork** class generates Python code to calculate each rate as a function of temperature and the ODE right hand side. pynucastro includes a sample Python integrator using SciPy and supports JIT-compilation using **numba** for RHS evaluation.

StarKiller Microphysics

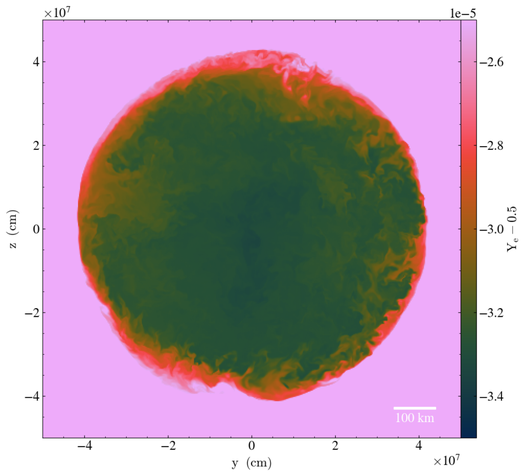
The **StarKillerNetwork** class generates Fortran reaction network code for astrophysical simulations that use the StarKiller Microphysics repository [2].

- Right hand side and Jacobian with SymPy [3]
- Couples to a stellar equation of state
- Incorporates screening and thermal neutrinos
- Supports CUDA Fortran

This short Python script builds a ^{12}C burning network with tabulated weak A = 23 rates [4]:

```
# C-burning with A=23 URCA rate module generator

from pynucastro.networks import StarKillerNetwork

files = ["c12-c12a-ne20-cf88",
         "c12-c12n-mg23-cf88",
         "c12-c12p-na23-cf88",
         "c12-ag-o16-nac2",
         "na23--ne23-toki",
         "ne23--na23-toki",
         "n--p-wc12"]

urca_net = StarKillerNetwork(files)
urca_net.write_network()
```

Electron fraction in a Maestro low-Mach hydrodynamics simulation of a 3-D convecting white dwarf core using the A = 23 Urca network:

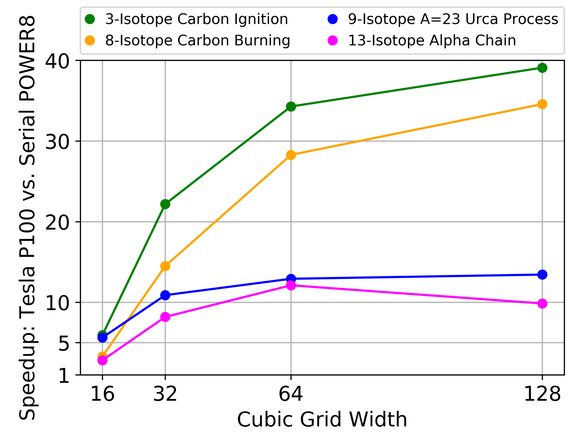


Ongoing Development

- Implementing nuclear partition functions
- Expanding support for weak rate tabulations

GPU Acceleration

pynucastro Fortran networks are compatible with PGI's CUDA Fortran compiler and we are also developing GPU accelerated ODE integrators. Below is speedup (serial time/GPU time) for our CUDA Fortran port of the VODE [5] integrator.



All except the 13-isotope approximate alpha chain network are pynucastro networks, and the 9-isotope Urca network includes tabulated weak rates.

Acknowledgements

This work was supported by DOE/Office of Nuclear Physics grant DE-FG02-87ER40317. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. We thank Lyra Cao for contributing numba integration for Python networks.

References

[1] D. E. Willcox and M. Zingale.
pynucastro: an interface to nuclear reaction rates and code generator for reaction network equations.
Journal of Open Source Software, 3(23):588, 2018.

[2] M. Zingale, A. S. Almgren, M. G. Barrios Sazo, V. E. Beckner, J. B. Bell, et al.
Meeting the Challenges of Modeling Astrophysical Thermonuclear Explosions: Castro, Maestro, and the AMReX Astrophysics Suite.
Journal of Physics: Conference Series, 1031(1):012024, 2018.

[3] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, et al.
SymPy: symbolic computing in Python.
PeerJ Computer Science, 3:e103, January 2017.

[4] Toshio Suzuki, Hiroshi Toki, and Ken'ichi Nomoto.
Electron-capture and β-decay Rates for sd-Shell Nuclei in Stellar Environments Relevant to High-density O-Ne-Mg Cores.
The Astrophysical Journal, 817(2):163, 2016.

[5] Peter N. Brown, George D. Byrne, and Alan C. Hindmarsh.
VODE: A Variable-Coefficient ODE Solver.
SIAM Journal on Scientific and Statistical Computing, 10(5):1038–1051, 1989.