nucleardatapy

Release 0.1

Jérôme Margueron, IRL NPA, USA

Nov 15, 2024

CONTENTS

1 Contents **3**
- 1.1 Usage . . . 3
- 1.2 API . . . 4
- 1.3 Miscelaneous . . . 4

2 Complement **7**
- 2.1 SetupEOSFFG . . . 7
- 2.2 SetupEOSMicro . . . 9
- 2.3 SetupEOSMicroBand . . . 14
- 2.4 SetupEOSMicroLP . . . 15
- 2.5 SetupEOSPheno . . . 17
- 2.6 SetupEOSAM . . . 24
- 2.7 SetupEOSBeta . . . 24
- 2.8 SetupEOSHIC . . . 26
- 2.9 SetupEOSEsym . . . 28
- 2.10 SetupNucBEExp . . . 29
- 2.11 SetupNucBETheo . . . 33
- 2.12 SetupNucRchExp . . . 35
- 2.13 SetupNucISGMRExp . . . 37
- 2.14 SetupCrust . . . 39
- 2.15 SetupAstroMasses . . . 40
- 2.16 SetupAstroMup . . . 43
- 2.17 SetupAstroMtov . . . 44
- 2.18 SetupAstroMR . . . 45
- 2.19 SetupAstroGW . . . 47
- 2.20 SetupCorEsymLsym . . . 48

3 Indices and tables **51**

Python Module Index **53**

Index **55**

nucleardatapy (/in short nuda/) is a Python library for nuclear physicists facilitating the access to theoretical or experimental nuclear data. It is specificaly designed for equation of state practitionners interested in the modeling of neutron stars, and it offers *simple* and *intuitive* APIs.

All data are provided with their reference, so when using these data in a scientific paper, reference to data should be provided explicitely. The reference to this toolkit could be given, but it should not mask the reference to data.

This python toolkit is designed to provide: 1) microscopic calculations in nuclear matter, 2) phenomenological predictions in nuclear matter, 3) experimental data for finite nuclei.

Check out the *Usage* section for further information, including how to *install* the project.

Note

This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

To use nucleardatapy, first download the .zip file from the git repository, or clone it in your local computer:

```
$ git clone https://github.com/jeromemargueron/nucleardatapy
```

If you have downloaded the .zip file, you can unzip it anywhere in your local computer:

```
$ unzip nucleardatapy.zip
```

Then, in all cases, you shall enter into the new folder */nucleardatapy*:

```
$ cd nucleardatapy
```

and launch the install script:

```
$ bash install.sh
```

This will copy the Python toolkit into $HOME/mylib/ as well as a few samples. It will also give you the content of the global variable NUCLEARDATAPY_TK. If you edit install.sh, you can change the version (by default it is set to the latest one) as well as the destination folder (by default it is $HOME/mylib).

Finally, you will have to create the global variable NUCLEARDATAPY_TK with its right content. If you do not want to create it each time you open a new terminal, then you can define it in your .profile or .zprofil or .bash file as:

```
export NUCLEARDATAPY_TK=$HOME/mylib/nucleardatapy
```

> **Note**
>
> The exact path to write above is given at the end of the installation.

1.1.2 Use nucleardatapy

Go to the folder *mylib/nucleardatapy/samples/nucleardatapy_samples/* and try that:

```
$ python3 sample_SetupMicroMatter.py
```

1.1.3 Test nucleardatapy

A set of tests can be easily performed. They are stored in tests/ folder.

```
$ bash run_tests.sh
```

1.1.4 Get started

How to obtain microscopic results for APR equation of state:

```
import os
nucleardatapy_tk = os.getenv('NUCLEARDATAPY_TK')
sys.path.insert(0, nucleardatapy_tk)

import nucleardatapy as nuda

mic = nuda.SetupMicroMatter( model = '1998-VAR-AM-APR' )

mic.print_outputs( )
```

1.2 API

nucleardatapy	This module provides microscopic, phenomenological and experimental data constraints.

1.2.1 nucleardatapy

This module provides microscopic, phenomenological and experimental data constraints.

1.3 Miscelaneous

1.3.1 Contributing

For the moment, contributions are based on co-optation among the team.

To make contribution easy, we all work in the *main* branch and we shall therefore remember to pull before working and pulling after, with a running version. For long developments, you can work in a local folder (in *mylib* for instance) and copy your contribution to the GitHub folder once you are sure it is functionning. So the final step should last less than 5 minutes, and can be safely done between a pull and before a push. Since we are not numerous, we hope that no one will work in the same part of the code at the same time (i.e. between a pull and a push). It is probably the simpler way to proceed.

Once the toolkit is released, the rules to contribute will be changing. A team of developpers should be defined and a generic email to contact them should be created. Here is a suggestion to contribute after the release.

This file describes how new contributors to the project can start contributing.

Two ways:

You can provide your data and interacting with one of our developer.

You can also join the developing team and extend the functionality of this toolkit.

Provide your data:

Please contact the developer team directly by shooting an email to TBC.

Then you can interact directly with one of our developer and provide your data. You will not be able to push your data to the repository, but an updated version of the toolkit will contain your new data.

Join the team:

Please contact the developer team directly by shooting an email to TBC. Explain the reason why you wish to join the team and if you have ideas about extending the functionality of the toolkit.

Once in the team, a branch will be dedicated to your contribution. You could show it during our virtual meetings, and your contribution will be merged to the new version of the toolkit.

1.3.2 License

TBC.

1.3.3 Report issues

For the current version, we report issues chatting among us. Once this toolkit is released, we should setup a way that users could contact us and report issues or difficulties in installing or using the toolkit.

1.3.4 Thanks

A special thanks to all contributors who accepted to share their results in this toolkit.

CHAPTER

TWO

COMPLEMENT

2.1 SetupEOSFFG

class nucleardatapy.setup_eos_ffg.**SetupEOSFFG**(*den*, *delta*)

Instantiate the object with free Fermi gas (FFG) quantities.

Parameters

- **den** (*float or numpy vector of floats.*) – density or densities for which the FFG quantities are calculated.
- **delta** (*float or numpy vector of floats.*) – isospin density or densities for which the FFG quantities are calculated.

Attributes:

delta

Attribute isospin parameter

den

Attribute isoscalar density

den_n

Attribute neutron density

den_p

Attribute proton density

e2a_int

Attribute FFG energy per particle

e2v_int

Attribute FFG energy per unit volum

eF_n

Attribute neutron Fermi energy

eF_p

Attribute proton Fermi energy

esym

Attribute FFG symmetry energy

esym2

Attribute FFG quadratic contribution to the symmetry energy

esym4

Attribute FFG quartic contribution to the symmetry energy

kf

Attribute Fermi momentum

kf_n

Attribute neutron Fermi momentum

kf_p

Attribute proton Fermi momentum

label

Attribute providing the label the data is references for figures.

note

Attribute providing additional notes about the data.

pre

Attribute FFG pressure

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_eos_ffg.**den**(*kf*)

Density as a function of the Fermi momentum.

Parameters
kf_n (*float or numpy vector of real numbers.*) – Fermi momentum.

nucleardatapy.setup_eos_ffg.**den_n**(*kf_n*)

Neutron density as a function of the neutron Fermi momentum.

Parameters
kf_n (*float or numpy vector of real numbers.*) – neutron Fermi momentum.

nucleardatapy.setup_eos_ffg.**eF_n**(*kf_n*)

Neutron Fermi energy as a function of the neutron Fermi momentum.

Parameters
kf_n (*float or numpy vector of real numbers.*) – neutron Fermi momentum.

nucleardatapy.setup_eos_ffg.**effg**(*kf*)

Free Fermi gas energy as a function of the Fermi momentum.

Parameters
kf (*float or numpy vector of real numbers.*) – Fermi momentum.

nucleardatapy.setup_eos_ffg.**effg_NM**(*kf_n*)

Free Fermi gas energy as a function of the neutron Fermi momentum.

Parameters
kf_n (*float or numpy vector of real numbers.*) – neutron Fermi momentum.

nucleardatapy.setup_eos_ffg.**effg_SM**(*kf*)

Free Fermi gas energy as a function of the Fermi momentum in SM.

Parameters
kf (*float or numpy vector of real numbers.*) – neutron Fermi momentum.

nucleardatapy.setup_eos_ffg.**esymffg**(*kf*)

Free Fermi gas symmetry energy as a function of the Fermi momentum.

Parameters
kf (*float or numpy vector of real numbers.*) – Fermi momentum.

nucleardatapy.setup_eos_ffg.**kf**(*den*)

Fermi momentum as a function of the density.

Parameters
den (*float or numpy vector of real numbers.*) – density.

nucleardatapy.setup_eos_ffg.**kf_n**(*den_n*)

Neutron Fermi momentum as a function of the neutron density.

Parameters
den_n (*float or numpy vector of real numbers.*) – neutron density.

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupEOSMicro.py

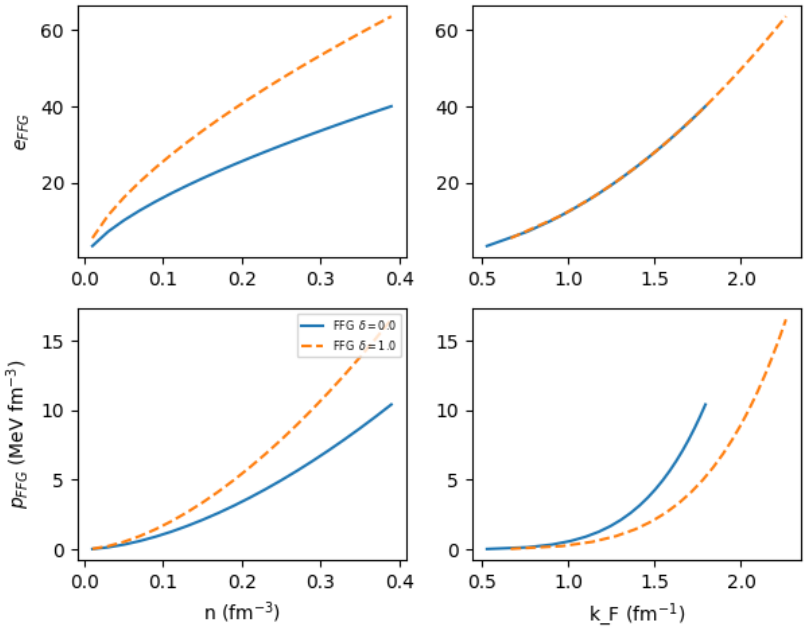

Fig. 1: This figure shows the free Fermi gas energy (top) and pressure (bottom) in symmetric matter (SM) (Blue solid line) and neutron matter (NM) (orange dashed line) as function of the particle density (left) and Fermi momentum (right).

2.2 SetupEOSMicro

class nucleardatapy.setup_eos_micro.**SetupEOSMicro**(*model='1998-VAR-AM-APR'*, *var1=array([0.01, 0.01393939, 0.01787879, 0.02181818, 0.02575758, 0.02969697, 0.03363636, 0.03757576, 0.04151515, 0.04545455, 0.04939394, 0.05333333, 0.05727273, 0.06121212, 0.06515152, 0.06909091, 0.0730303, 0.0769697, 0.08090909, 0.08484848, 0.08878788, 0.09272727, 0.09666667, 0.10060606, 0.10454545, 0.10848485, 0.11242424, 0.11636364, 0.12030303, 0.12424242, 0.12818182, 0.13212121, 0.13606061, 0.14, 0.14393939, 0.14787879, 0.15181818, 0.15575758, 0.15969697, 0.16363636, 0.16757576, 0.17151515, 0.17545455, 0.17939394, 0.18333333, 0.18727273, 0.19121212, 0.19515152, 0.19909091, 0.2030303, 0.2069697, 0.21090909, 0.21484848, 0.21878788, 0.22272727, 0.22666667, 0.23060606, 0.23454545, 0.23848485, 0.24242424, 0.24636364, 0.25030303, 0.25424242, 0.25818182, 0.26212121, 0.26606061, 0.27, 0.27393939, 0.27787879, 0.28181818, 0.28575758, 0.28969697, 0.29363636, 0.29757576, 0.30151515, 0.30545455, 0.30939394, 0.31333333, 0.31727273, 0.32121212, 0.32515152, 0.32909091, 0.3330303, 0.3369697, 0.34090909, 0.34484848, 0.34878788, 0.35272727, 0.35666667, 0.36060606, 0.36454545, 0.36848485, 0.37242424, 0.37636364, 0.38030303, 0.38424242, 0.38818182, 0.39212121, 0.39606061, 0.4])*, *var2=0.0*)

Instantiate the object with microscopic results choosen by the toolkit practitioner.

This choice is defined in *model*, which can chosen among the following choices: '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-iAPR', '2006-BHF-AM*', '2008-BCS-NM', '2008-AFDMC-NM', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2012-AFDMC-NM-1', '2012-AFDMC-NM-2', '2012-AFDMC-NM-3', '2012-AFDMC-NM-4', '2012-AFDMC-NM-5', '2012-AFDMC-NM-6', '2012-AFDMC-NM-7', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '2022-AFDMC-NM', '2024-NLEFT-AM', '2024-BHF-AM-2BF-Av8p', '2024-BHF-AM-2BF-Av18', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-CDBONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97c', '2024-BHF-AM-2BF-NSC97d', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-2BF-SSCV14', '2024-BHF-AM-23BF-Av8p', '2024-BHF-AM-23BF-Av18', '2024-BHF-AM-23BF-BONN', '2024-BHF-AM-23BF-CDBONN', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97d', '2024-BHF-AM-23BF-NSC97e', '2024-BHF-AM-23BF-NSC97f', '2024-BHF-AM-23BF-SSCV14'

Parameters

model (*str, optional.*) – Fix the name of model. Default value: '1998-VAR-AM-APR'.

Attributes:

init_self()

Initialize variables in self.

model

Attribute model.

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_eos_micro.**eos_micro_models**()

Return a list with the name of the models available in this toolkit and print them all on the prompt. These models are the following ones: '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-APRfit', '2006-BHF-AM*', '2008-BCS-NM', '2008-AFDMC-NM', '2012-AFDMC-NM-1', '2012-AFDMC-NM-2', '2012-AFDMC-NM-3', '2012-AFDMC-NM-4', '2012-AFDMC-NM-5', '2012-AFDMC-NM-6', '2012-AFDMC-NM-7', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '2022-AFDMC-NM', '2024-NLEFT-AM', '2024-BHF-AM-2BF-Av8p', '2024-BHF-AM-2BF-Av18', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-CDBONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97c', '2024-BHF-AM-2BF-NSC97d', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-2BF-SSCV14', '2024-BHF-AM-23BF-Av8p', '2024-BHF-AM-23BF-Av18', '2024-BHF-AM-23BF-BONN', '2024-BHF-AM-23BF-CDBONN', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97d', '2024-BHF-AM-23BF-NSC97e', '2024-BHF-AM-23BF-NSC97f', '2024-BHF-AM-23BF-SSCV14', '2024-BHF-AM-23BFmicro-Av18', '2024-BHF-AM-23BFmicro-BONNB', '2024-BHF-AM-23BFmicro-NSC93' :return: The list of models. :rtype: list[str].

nucleardatapy.setup_eos_micro.**eos_micro_models_group_NM**(*group*)

nucleardatapy.setup_eos_micro.**eos_micro_models_group_SM**(*group*)

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupEOSMicro.py

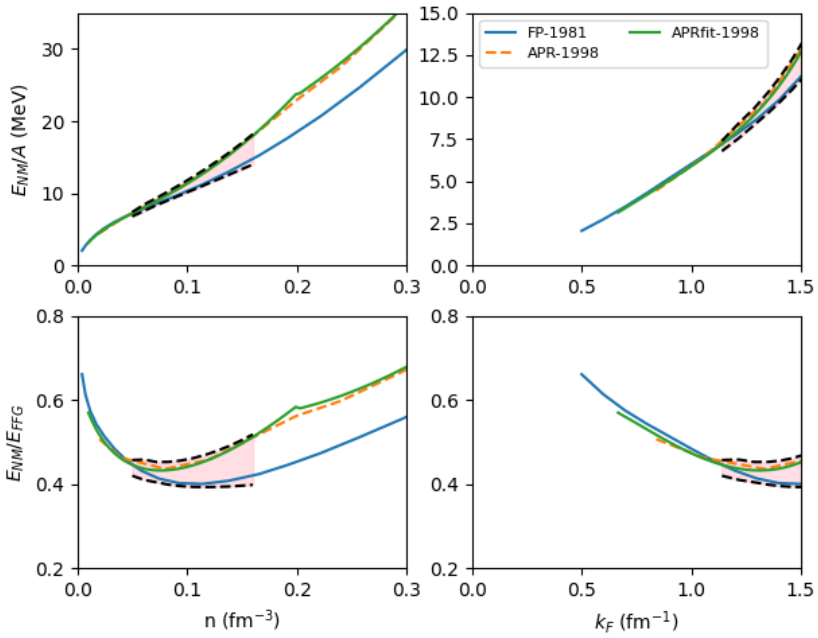

Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the variational models available in the nucleardatapy toolkit.

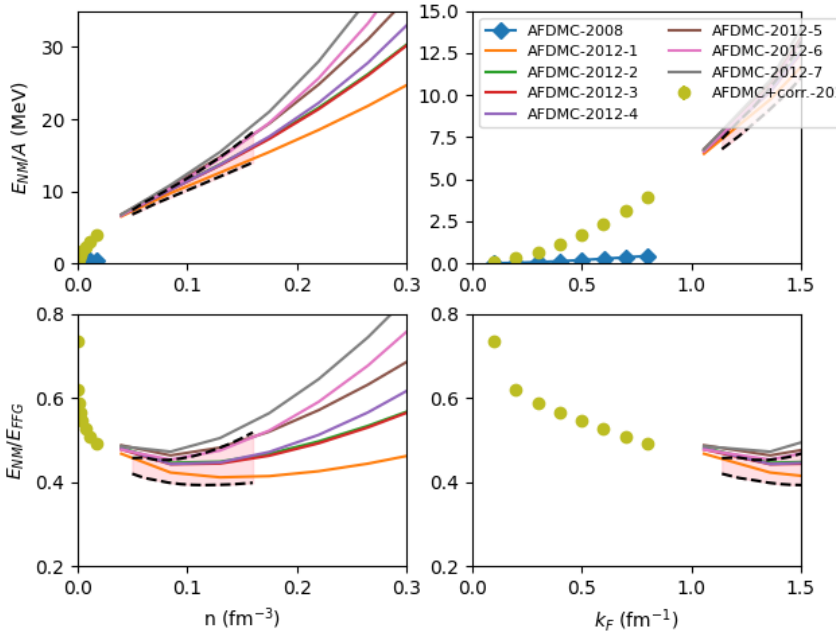

Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

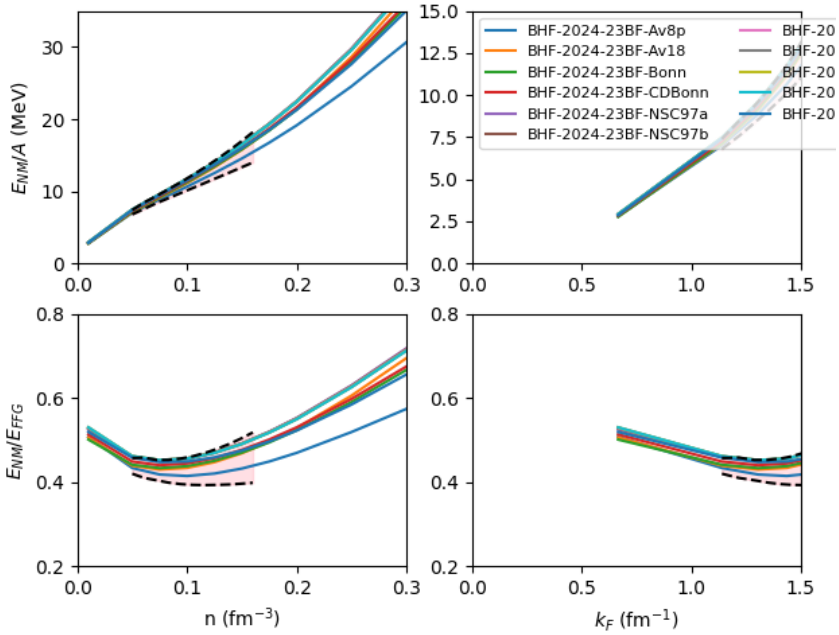

Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

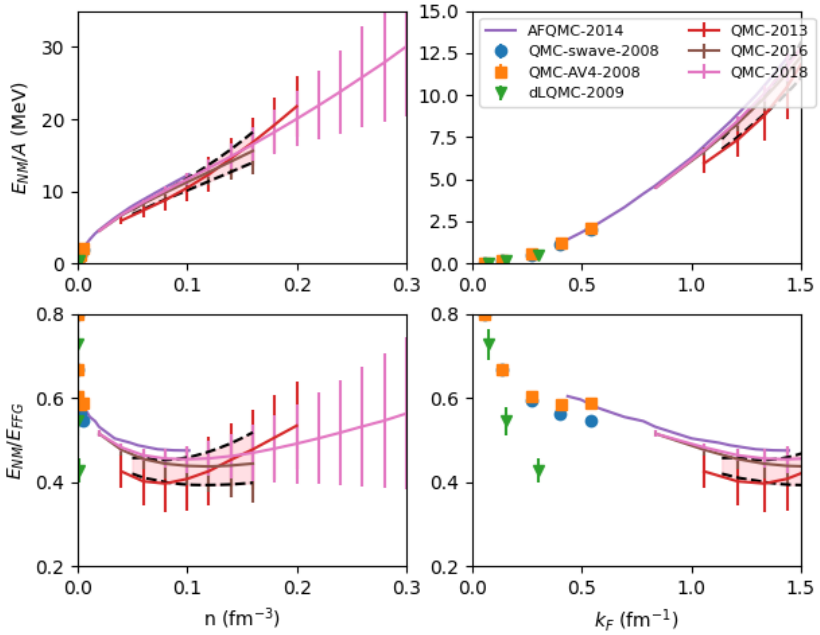

Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

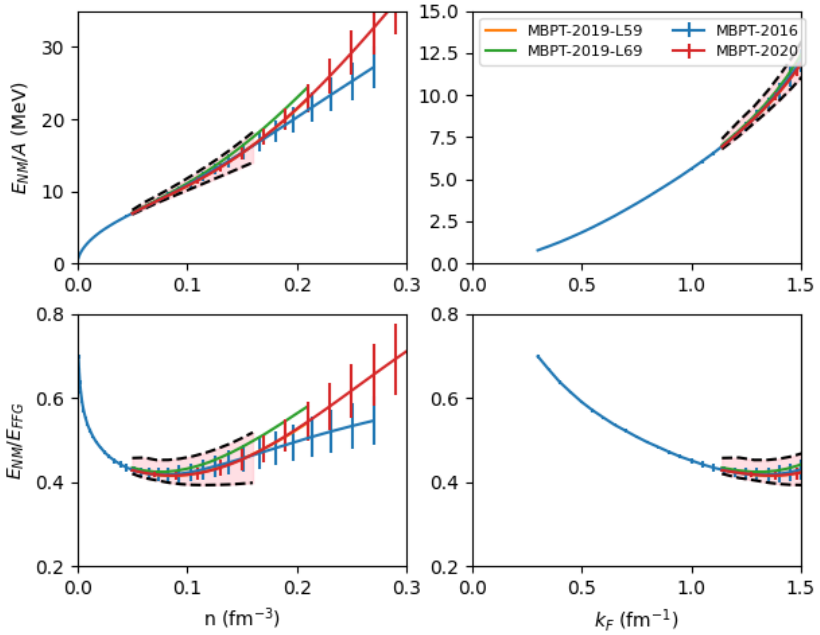

Fig. 6: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.

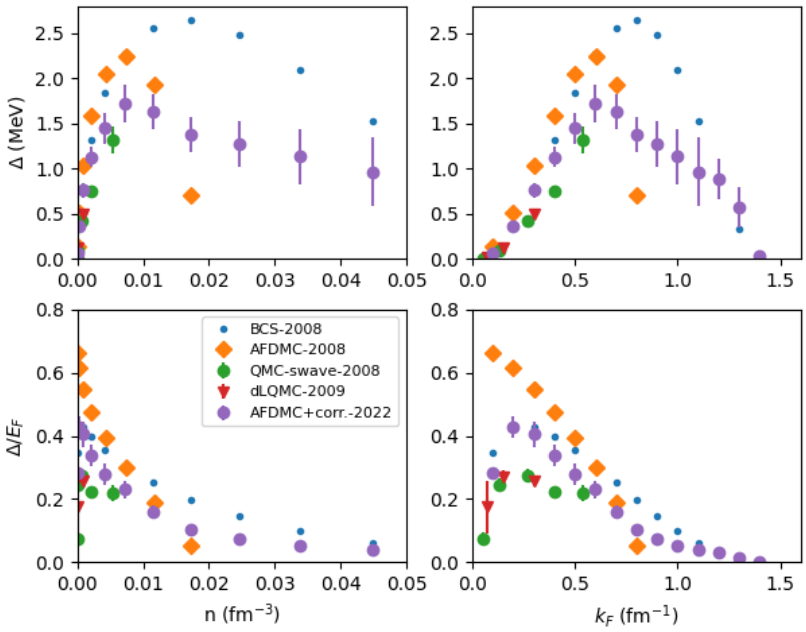

Fig. 7: This figure shows the pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

2.3 SetupEOSMicroBand

class nucleardatapy.setup_eos_micro_band.**SetupEOSMicroBand**(*models=['2016-MBPT-AM']*, *nden=10*, *ne=200*, *den=None*, *matter='NM'*, *e2a_min=-20.0*, *e2a_max=50.0*)

Instantiate the object with statistical distributions averaging over the models given as inputs and in NM.

Parameters

- **models** (*list.*) – The models given as inputs.
- **nden** (*int, optional.*) – number of density points.
- **ne** (*int, optional.*) – number of points along the energy axis.
- **den** (*None or numpy array, optional.*) – if not None (default), impose the densities.
- **matter** (*str, optional.*) – can be 'NM' (default), 'SM' or 'ESYM'.

Attributes:

den

Attribute a set of density points.

init_self()

Initialize variables in self.

matter

Attribute matter str.

models

Attribute model.

nden

Attribute number of points in density.

print_outputs()

Method which print outputs on terminal's screen.

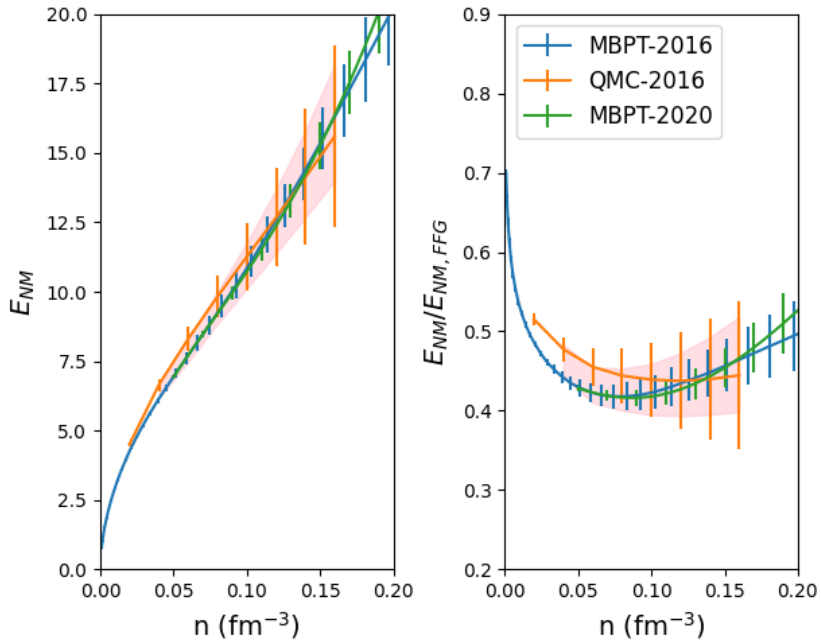

Fig. 8: Uncertainty band in NM obtained from the analysis of different predictions: MBPT-2016, QMC-2016 and MBPT-2020.

2.4 SetupEOSMicroLP

class nucleardatapy.setup_eos_micro_lp.**SetupEOSMicroLP**(*model='1994-BHF-SM-LP-AV14-GAP'*)

Instantiate the object with Landau parameters from microscopic calculations choosen by the toolkit practitioner.

This choice is defined in *model*, which can chosen among the following choices: '1994-BHF-SM-LP-AV14-GAP', '1994-BHF-SM-LP-AV14-CONT', '1994-BHF-SM-LP-REID-GAP', '1994-BHF-SM-LP-REID-CONT', '1994-BHF-SM-LP-AV14-CONT-0.7'.

Parameters

model (*str, optional.*) – Fix the name of model. Default value: '1994-BHF-LP'.

Attributes:

init_self()

Initialize variables in self.

model

Attribute model.

print_outputs()

Method which print outputs on terminal's screen.

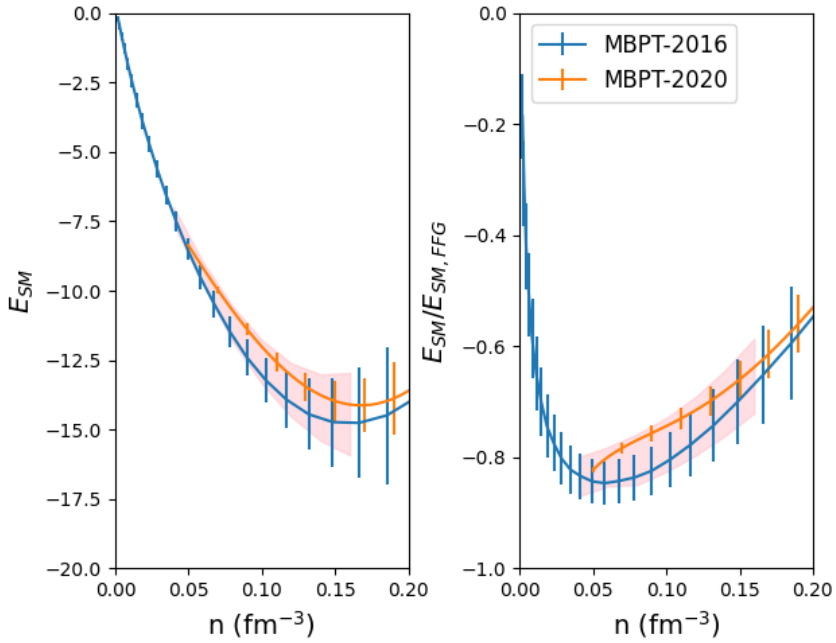

Fig. 9: Uncertainty band in SM obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

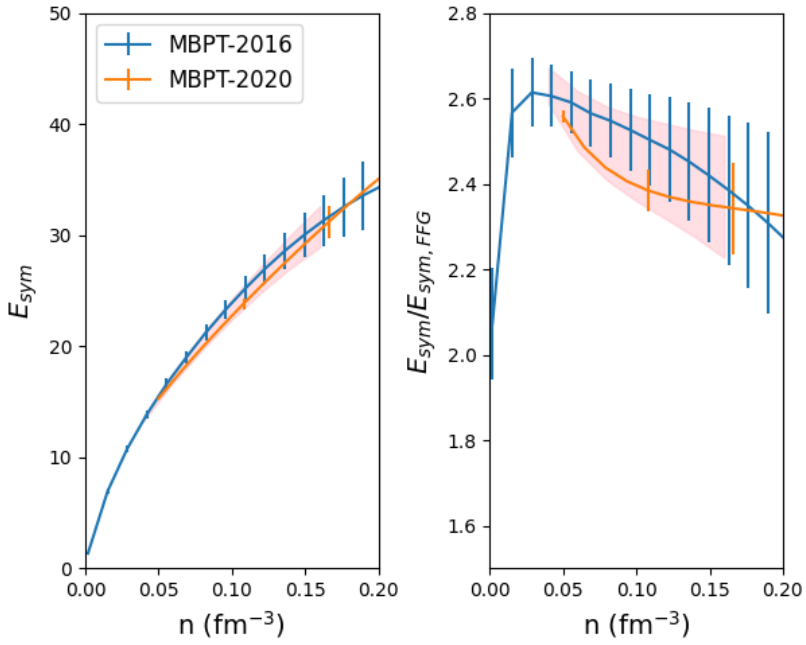

Fig. 10: Uncertainty band for the symmetry energy obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

nucleardatapy.setup_eos_micro_lp.**eos_micro_LP_models**()

Return a list with the name of the models available in this toolkit and print them all on the prompt. These models are the following ones: '1994-BHF-SM-LP-AV14-GAP', '1994-BHF-SM-LP-AV14-CONT', '1994-BHF-SM-LP-REID-GAP', '1994-BHF-SM-LP-REID-CONT', '1994-BHF-SM-LP-AV14-CONT-0.7'.

Returns

The list of models.

Return type

list[str].

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupeosmicroLP.py

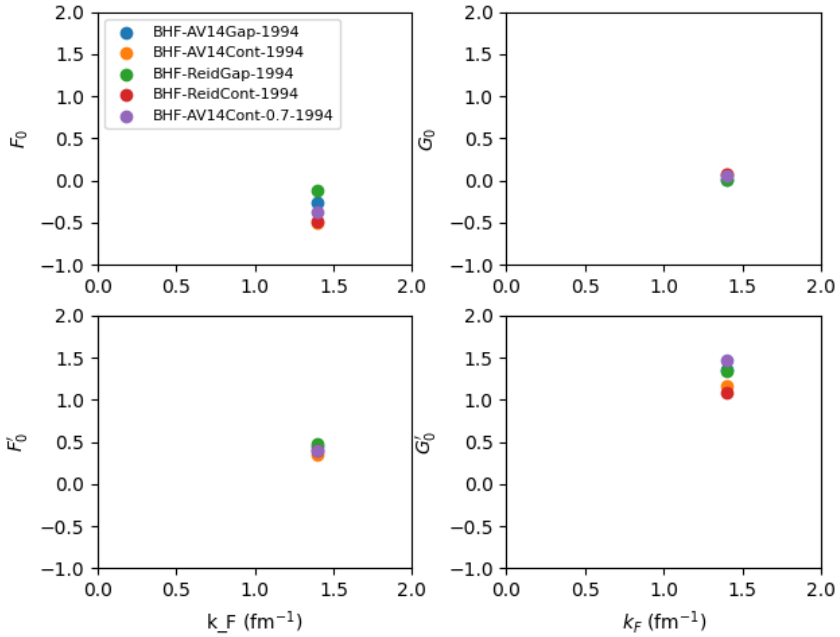

Fig. 11: This figure shows the L=0 Landau parameters in SM for different NN interactions obtained from BHF calculations.

2.5 SetupEOSPheno

class nucleardatapy.setup_eos_pheno.**SetupEOSPheno**(*model='Skyrme'*, *param='SLY5'*)

Instantiate the object with results based on phenomenological interactions and choosen by the toolkit practitioner. This choice is defined in the variables *model* and *param*.

If *models* == 'skyrme', *param* can be: 'BSK14', 'BSK16', 'BSK17', 'BSK27', 'F-', 'F+', 'F0', 'FPL', 'LNS', 'LNS1', 'LNS5', 'NRAPR', 'RATP', 'SAMI', 'SGII', 'SIII', 'SKGSIGMA', 'SKI2', 'SKI4', 'SKMP', 'SKMS', 'SKO', 'SKOP', 'SKP', 'SKRSIGMA', 'SKX', 'Skz2', 'SLY4', 'SLY5', 'SLY230A', 'SLY230B', 'SV', 'T6', 'T44', 'UNEDF0', 'UNEDF1'.

If *models* == 'NLRH', *param* can be: 'NL-SH', 'NL3', 'NL3II', 'PK1', 'PK1R', 'TM1'.

If *models* == 'DDRH', *param* can be: 'DDME1', 'DDME2', 'DDMEd', 'PKDD', 'TW99'.

If *models* == 'DDRHF', *param* can be: 'PKA1', 'PKO1', 'PKO2', 'PKO3'.

Parameters

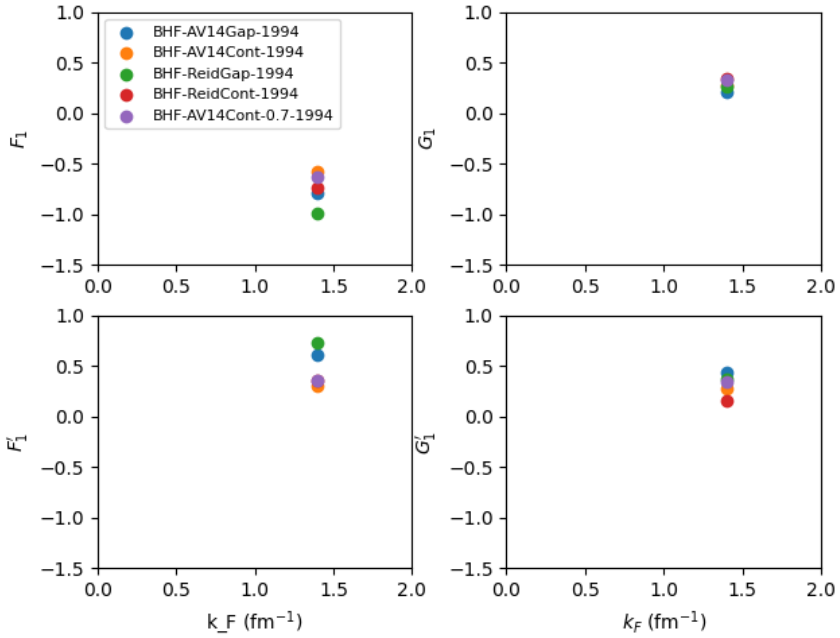

Fig. 12: This figure shows the L=1 Landau parameters in SM for different NN interactions obtained from BHF calculations.

- **model** (*str, optional.*) – Fix the name of model: 'Skyrme', 'NLRH', 'DDRH', 'DDRHF'. Default value: 'Skyrme'.
- **param** (*str, optional.*) – Fix the parameterization associated to model. Default value: 'SLY5'.

Attributes:

init_self()

Initialize variables in self.

label

Attribute providing the label the data is references for figures.

model

Attribute model.

note

Attribute providing additional notes about the data.

param

Attribute param.

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_eos_pheno.**eos_pheno_models**()

Return a list of models available in this toolkit and print them all on the prompt.

Returns

The list of models with can be 'Skyrme', 'NLRH', 'DDRH', 'DDRHF'.

Return type
list[str].

nucleardatapy.setup_eos_pheno.**eos_pheno_params**(*model*)

Return a list with the parameterizations available in this toolkit for a given model and print them all on the prompt.

Parameters
model (*str.*) – The type of model for which there are parametrizations. They should be chosen among the following options: 'Skyrme', 'NLRH', 'DDRH', 'DDRHF'.

Returns
The list of parametrizations. If *models* == 'skyrme': 'BSK14', 'BSK16', 'BSK17', 'BSK27', 'F-', 'F+', 'F0', 'FPL', 'LNS', 'LNS1', 'LNS5', 'NRAPR', 'RATP', 'SAMI', 'SGII', 'SIII', 'SKGSIGMA', 'SKI2', 'SKI4', 'SKMP', 'SKMS', 'SKO', 'SKOP', 'SKP', 'SKRSIGMA', 'SKX', 'Skz2', 'SLY4', 'SLY5', 'SLY230A', 'SLY230B', 'SV', 'T6', 'T44', 'UNEDF0', 'UNEDF1'. If *models* == 'NLRH': 'NL-SH', 'NL3', 'NL3II', 'PK1', 'PK1R', 'TM1'. If *models* == 'DDRH': 'DDME1', 'DDME2', 'DDMEd', 'PKDD', 'TW99'. If *models* == 'DDRHF': 'PKA1', 'PKO1', 'PKO2', 'PKO3'.

Return type
list[str].

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupEOSPheno.py

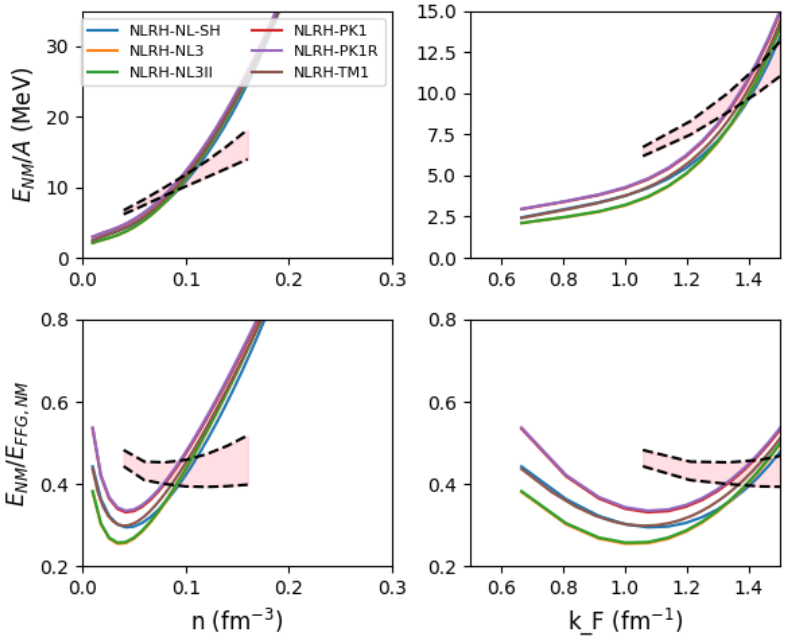

Fig. 13: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.

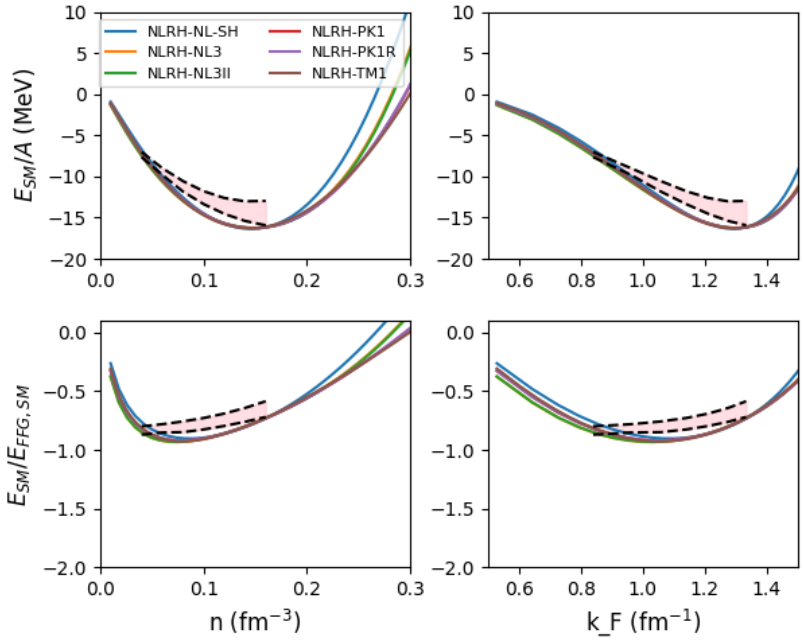

Fig. 14: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.

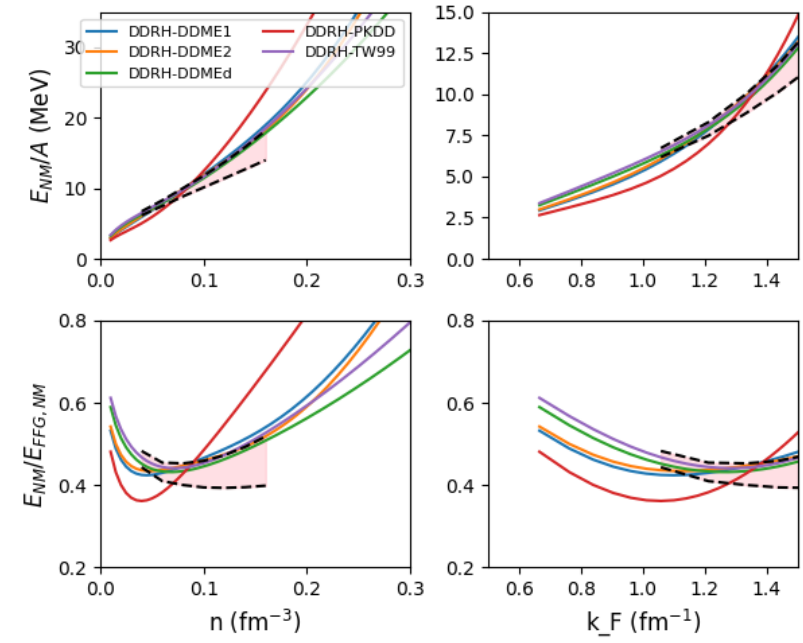

Fig. 15: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree (DDRH) approach available in the nucleardatapy toolkit.

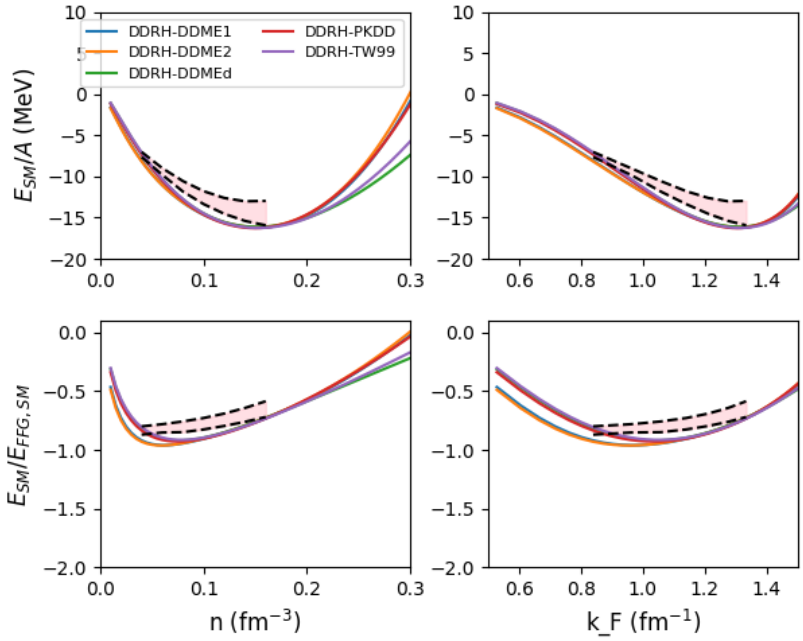

Fig. 16: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree (DDRH) approach available in the nucleardatapy toolkit.

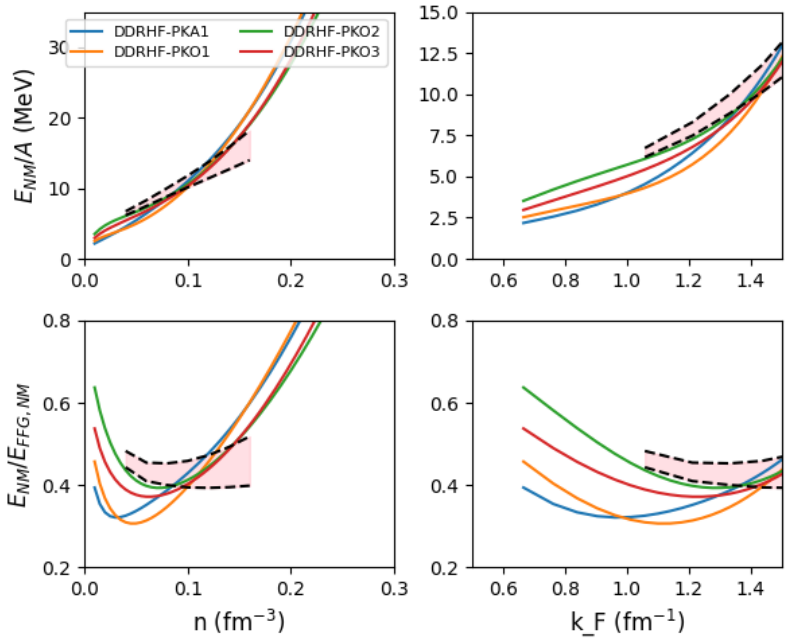

Fig. 17: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock (DDRHF) approach available in the nucleardatapy toolkit.

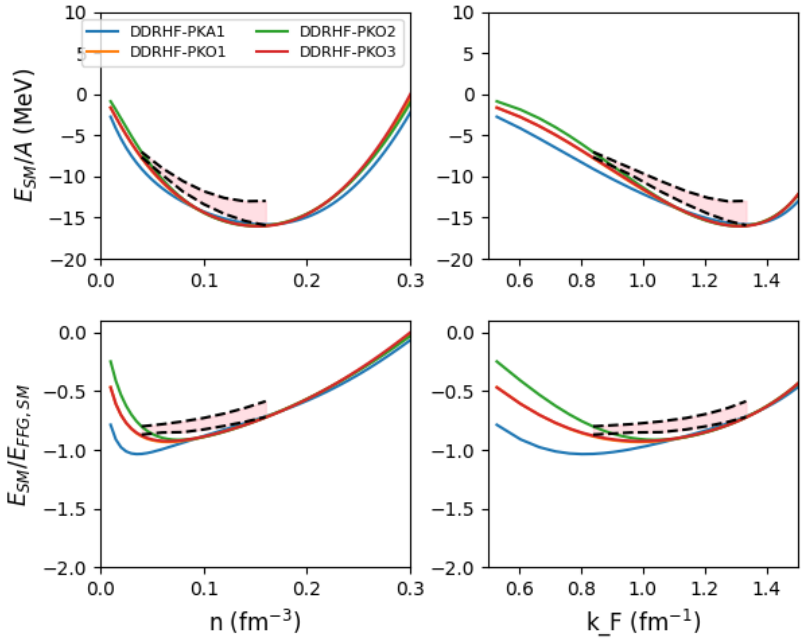

Fig. 18: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock (DDRHF) approach available in the nucleardatapy toolkit.

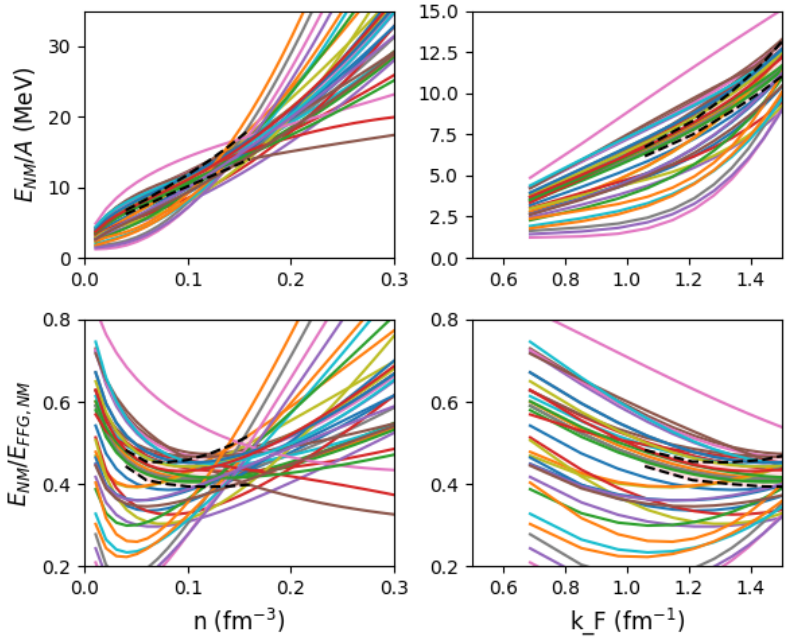

Fig. 19: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available in the nucleardatapy toolkit.

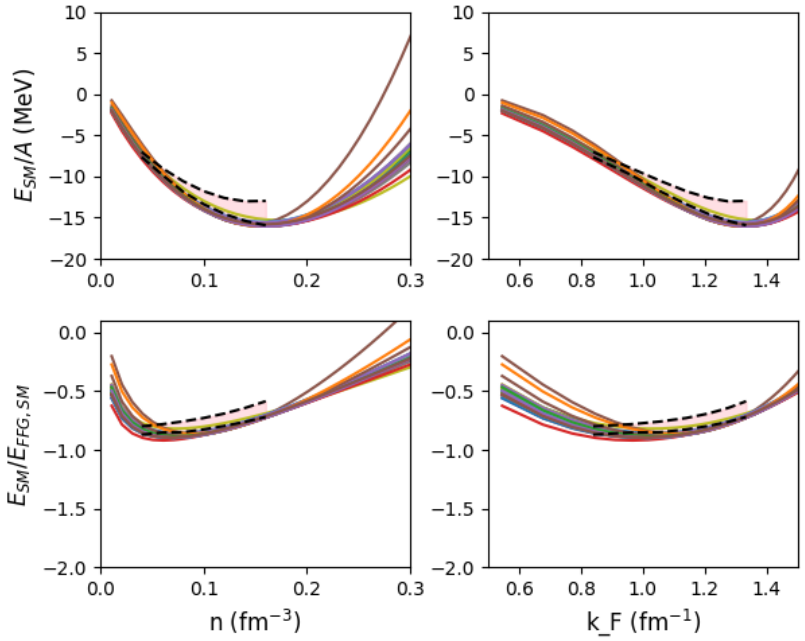

Fig. 20: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available in the nucleardatapy toolkit.

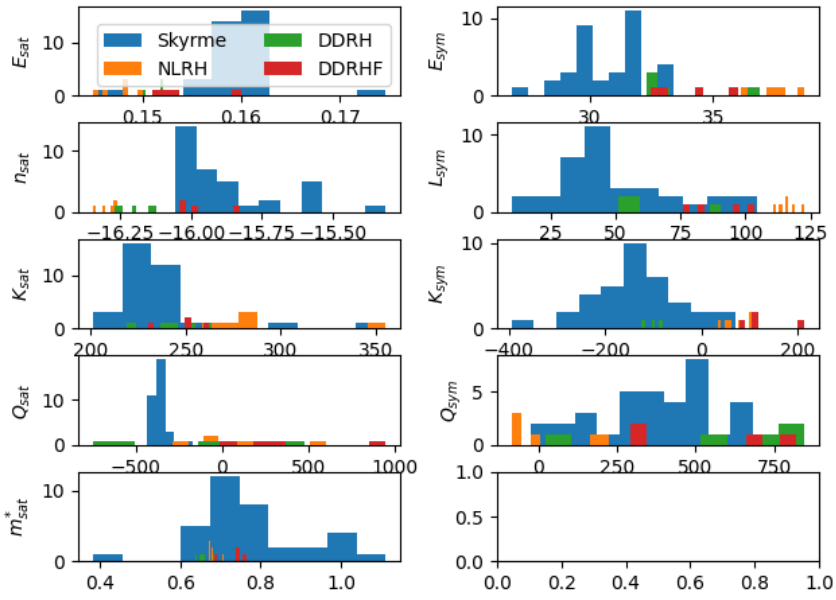

Fig. 21: Distribution of NEP for phenomenological models available in the nucleardatapy toolkit.

2.6 SetupEOSAM

class nucleardatapy.setup_eos_am.**SetupEOSAM**(*model='1998-VAR-AM-APR'*, *param=None*, *kind='micro'*, *asy=0.0*, *var1=array([0.01, 0.01393939, 0.01787879, 0.02181818, 0.02575758, 0.02969697, 0.03363636, 0.03757576, 0.04151515, 0.04545455, 0.04939394, 0.05333333, 0.05727273, 0.06121212, 0.06515152, 0.06909091, 0.0730303, 0.0769697, 0.08090909, 0.08484848, 0.08878788, 0.09272727, 0.09666667, 0.10060606, 0.10454545, 0.10848485, 0.11242424, 0.11636364, 0.12030303, 0.12424242, 0.12818182, 0.13212121, 0.13606061, 0.14, 0.14393939, 0.14787879, 0.15181818, 0.15575758, 0.15969697, 0.16363636, 0.16757576, 0.17151515, 0.17545455, 0.17939394, 0.18333333, 0.18727273, 0.19121212, 0.19515152, 0.19909091, 0.2030303, 0.2069697, 0.21090909, 0.21484848, 0.21878788, 0.22272727, 0.22666667, 0.23060606, 0.23454545, 0.23848485, 0.24242424, 0.24636364, 0.25030303, 0.25424242, 0.25818182, 0.26212121, 0.26606061, 0.27, 0.27393939, 0.27787879, 0.28181818, 0.28575758, 0.28969697, 0.29363636, 0.29757576, 0.30151515, 0.30545455, 0.30939394, 0.31333333, 0.31727273, 0.32121212, 0.32515152, 0.32909091, 0.3330303, 0.3369697, 0.34090909, 0.34484848, 0.34878788, 0.35272727, 0.35666667, 0.36060606, 0.36454545, 0.36848485, 0.37242424, 0.37636364, 0.38030303, 0.38424242, 0.38818182, 0.39212121, 0.39606061, 0.4])*)

Instantiate the object with microscopic results choosen by the toolkit practitioner.

Parameters

- **model** (*str, optional.*) – Fix the name of model. Default value: '1998-VAR-AM-APR'.
- **kind** (*str, optional.*) – chose between 'micro' and 'pheno'.

Attributes:

init_self()

Initialize variables in self.

model

Attribute model.

print_outputs()

Method which print outputs on terminal's screen.

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupEOSAM.py

2.7 SetupEOSBeta

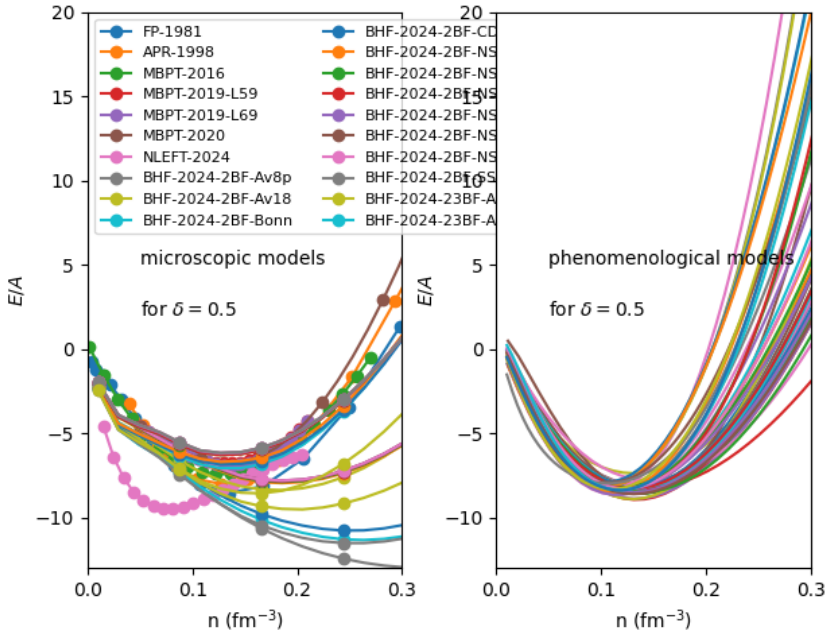

Fig. 22: This figure shows the energy per particle in asymmetric matter (AM) with d=0.5. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

class nucleardatapy.setup_eos_beta.**SetupEOSBeta**(*model='1998-VAR-AM-APR'*, *param=None*, *kind='micro'*, *var1=array([0.01, 0.01393939, 0.01787879, 0.02181818, 0.02575758, 0.02969697, 0.03363636, 0.03757576, 0.04151515, 0.04545455, 0.04939394, 0.05333333, 0.05727273, 0.06121212, 0.06515152, 0.06909091, 0.0730303, 0.0769697, 0.08090909, 0.08484848, 0.08878788, 0.09272727, 0.09666667, 0.10060606, 0.10454545, 0.10848485, 0.11242424, 0.11636364, 0.12030303, 0.12424242, 0.12818182, 0.13212121, 0.13606061, 0.14, 0.14393939, 0.14787879, 0.15181818, 0.15575758, 0.15969697, 0.16363636, 0.16757576, 0.17151515, 0.17545455, 0.17939394, 0.18333333, 0.18727273, 0.19121212, 0.19515152, 0.19909091, 0.2030303, 0.2069697, 0.21090909, 0.21484848, 0.21878788, 0.22272727, 0.22666667, 0.23060606, 0.23454545, 0.23848485, 0.24242424, 0.24636364, 0.25030303, 0.25424242, 0.25818182, 0.26212121, 0.26606061, 0.27, 0.27393939, 0.27787879, 0.28181818, 0.28575758, 0.28969697, 0.29363636, 0.29757576, 0.30151515, 0.30545455, 0.30939394, 0.31333333, 0.31727273, 0.32121212, 0.32515152, 0.32909091, 0.3330303, 0.3369697, 0.34090909, 0.34484848, 0.34878788, 0.35272727, 0.35666667, 0.36060606, 0.36454545, 0.36848485, 0.37242424, 0.37636364, 0.38030303, 0.38424242, 0.38818182, 0.39212121, 0.39606061, 0.4])*)

Instantiate the object with microscopic results choosen by the toolkit practitioner.

Parameters

- **model** (*str, optional.*) – Fix the name of model. Default value: '1998-VAR-AM-APR'.
- **kind** (*str, optional.*) – chose between 'micro' and 'pheno'.

Attributes:

init_self()

Initialize variables in self.

model

Attribute model.

print_outputs()

Method which print outputs on terminal's screen.

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupEOSBeta.py

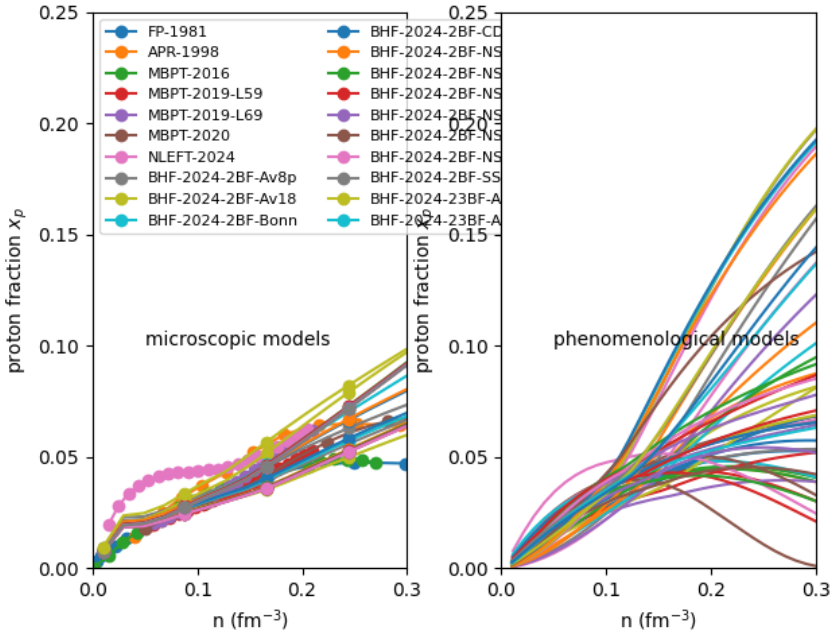

Fig. 23: This figure shows the proton fraction at beta-equilibrium. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

2.8 SetupEOSHIC

class nucleardatapy.setup_eos_hic.**SetupEOSHIC**(*constraint='DLL-2002'*)

Instantiate the constraints on the EOS from HIC.

This choice is defined in the variable *constraint*.

constraint can chosen among the following ones: ['DLL-2002', 'FOPI-2016'].

Parameters

constraint (*str, optional.*) – Fix the name of *constraint*. Default value: 'DLL-2002'.

Attributes:

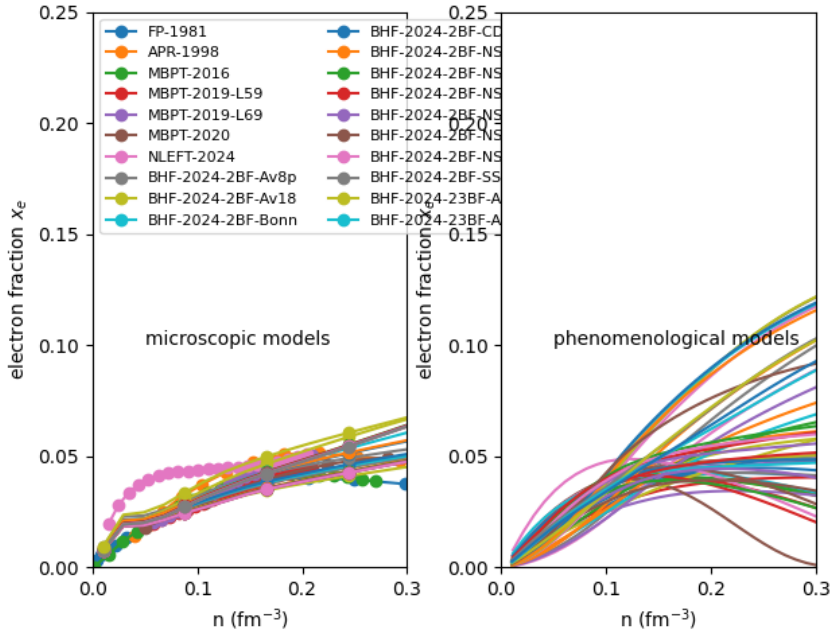

Fig. 24: This figure shows the electron fraction at beta-equilibrium. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

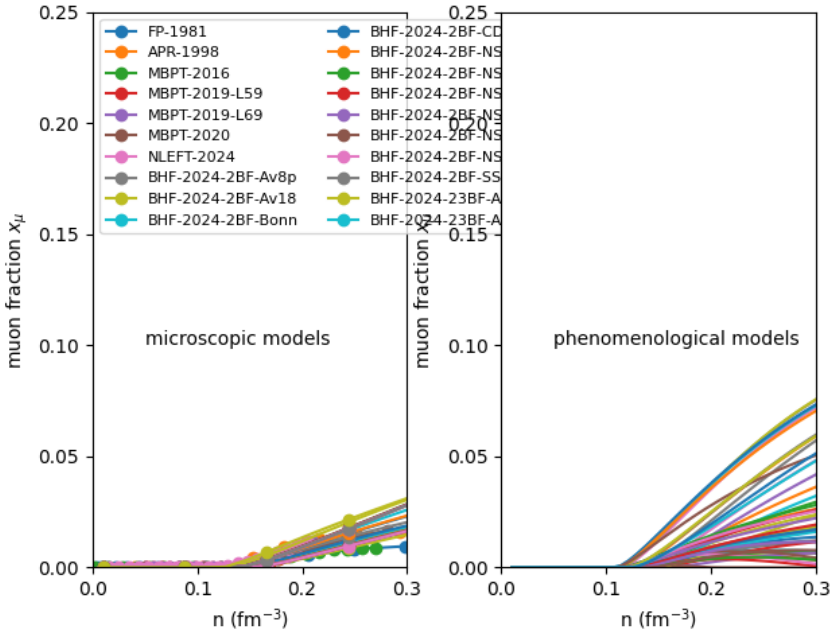

Fig. 25: This figure shows the muon fraction at beta-equilibrium. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

init_self()

Initialize variables in self.

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_eos_hic.**eos_hic_constraints**()

Return a list of the HIC constraints available in this toolkit for the equation of state in SM and NM and print them all on the prompt. These constraints are the following ones: ['DLL-2002', 'FOPI-2016'].

Returns

The list of constraints.

Return type

list[str].

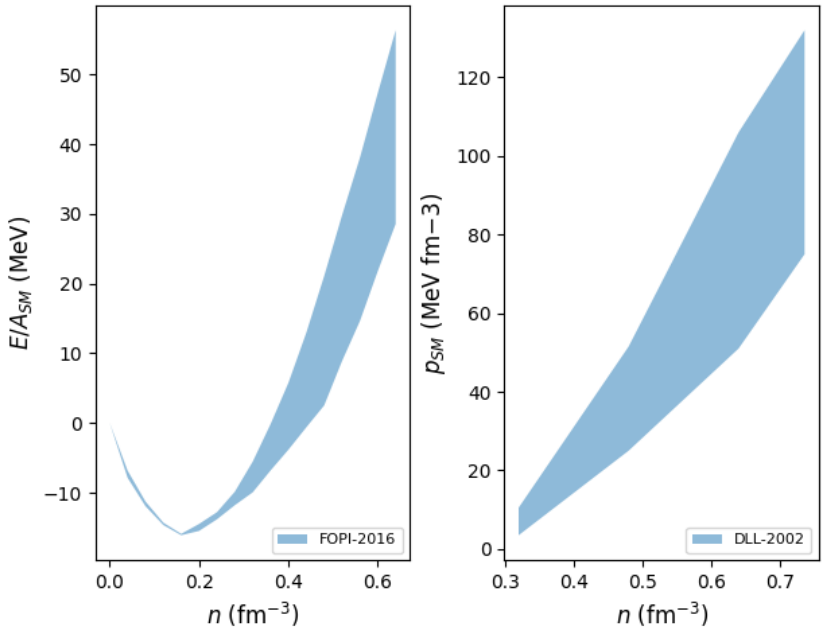

Fig. 26: HIC Experimental constraints for the energy per particle (left) and pressure (right) in SM as a function of the particle density for different analyses available in the *nuda* toolkit.

2.9 SetupEOSEsym

class nucleardatapy.setup_eos_esym.**SetupEOSEsym**(*constraint='2014-IAS'*, *Ksym=0.0*)

Instantiate the values of Esym and Lsym from the constraint.

Parameters

constraint (*str.*) – name of the model: '2014-IAS', ...

Returns

constraint, ref, label, note, Esym, Lsym.

constraint

Attribute the constraint

esym_e2a_max

Attribute the maximal symmetry energy

esym_e2a_min

Attribute the minimal symmetry energy

print_outputs()

Method which print outputs on terminal's screen.

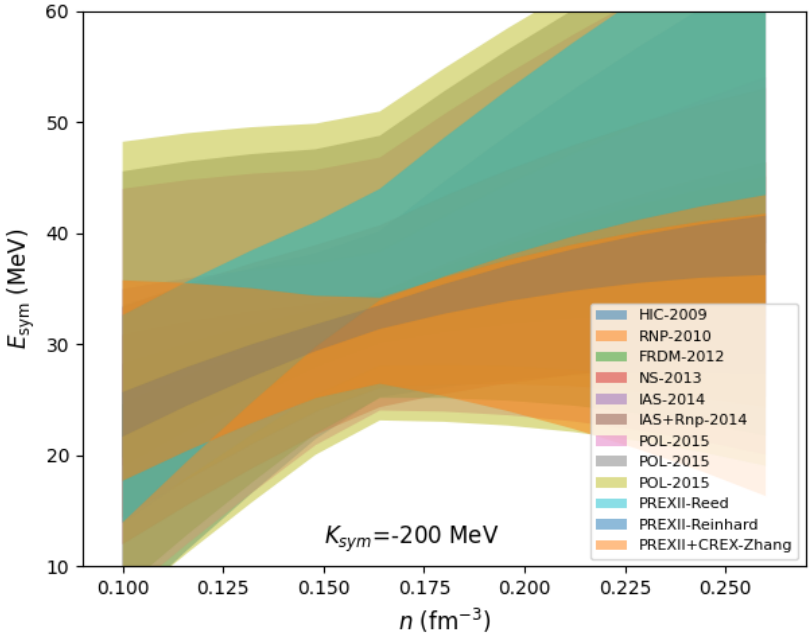

Fig. 27: Uncertainty band for Esym as a function of the density for Ksym=-200 MeV.

2.10 SetupNucBEExp

class nucleardatapy.setup_nuc_be_exp.**SetupNucBEExp**(*table='AME'*, *version='2020'*)

Instantiate the experimental nuclear masses from AME mass table.

This choice is defined in the variables *table* and *version*.

table can chosen among the following ones: 'AME'.

version can be chosen among the following choices: '2020', '2016', '2012'.

Parameters

- **table** (*str, optional.*) – Fix the name of *table*. Default value: 'AME'.
- **version** (*str, optional.*) – Fix the name of *version*. Default value: 2020'.

Attributes:

Zmax

maximum charge of nuclei present in the table.

Type

Attribute Zmax

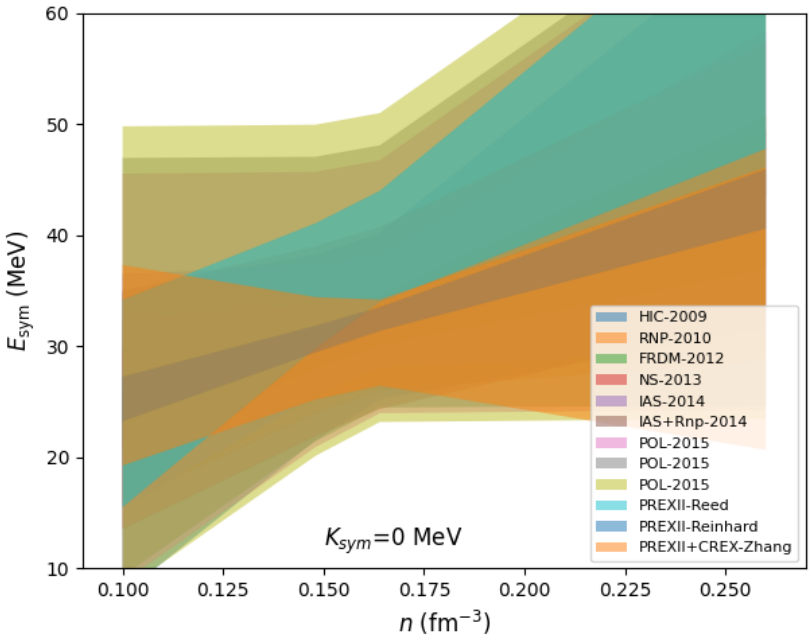

Fig. 28: Uncertainty band for Esym as a function of the density for Ksym=0 MeV.

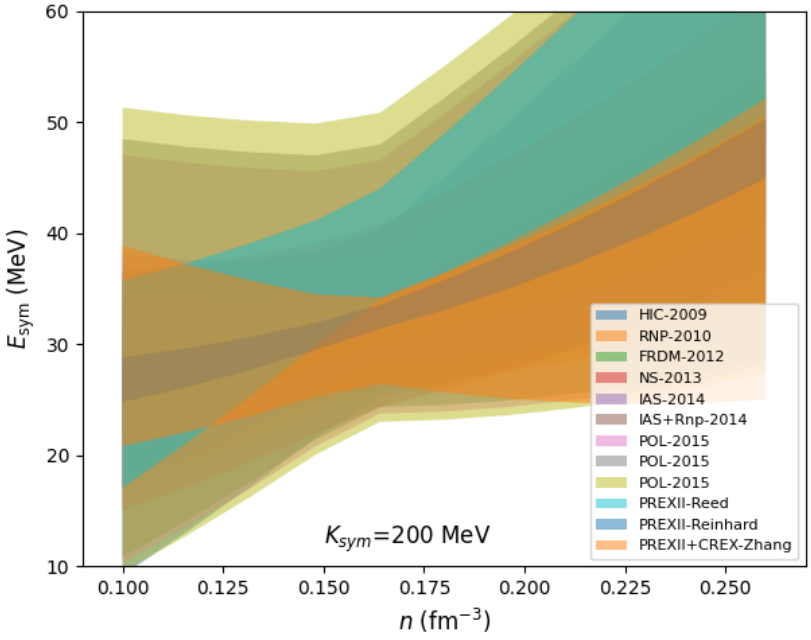

Fig. 29: Uncertainty band for Esym as a function of the density for Ksym=200 MeV.

dist_nbNuc

attribute number of nuclei discovered per year

dist_year

attribute distribution of years

drip(*Zmax=95*)

Method which find the drip-line nuclei (on the two sides).

Parameters

Zmax (*int, optional. Default: 95.*) – Fix the maximum charge for the search of the drip line.

Attributes:

flagI

Attribute I.

flagInterp

Attribute Interp (interpolation). Interp='y' is the nucleushas not been measured but is in the table based on interpolation expressions.otherwise Interp = 'n' for nuclei produced in laboratory and measured.

label

Attribute providing the label the data is references for figures.

nbLine

Attribute with the number of line in the file.

nbNuc

Attribute with the number of nuclei read in the file.

note

Attribute providing additional notes about the data.

nucA

Attribute A (mass of the nucleus).

nucBE

Attribute BE (Binding Energy) of the nucleus.

nucBE_err

Attribute uncertainty in the BE (Binding Energy) of the nucleus.

nucHT

Attribute HT (half-Time) of the nucleus.

nucN

Attribute N (number of neutrons of the nucleus).

nucStbl

Attribute stbl. stbl='y' if the nucleus is stable (according to the table). Otherwise stbl = 'n'.

nucSymb

Attribute symb (symbol) of the element, e.g., Fe.

nucYear

Attribute year of the discovery of the nucleus.

nucZ

Attribute Z (charge of the nucleus).

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

select(*Amin=0*, *Zmin=0*, *interp='n'*, *state='gs'*, *nucleus='unstable'*, *every=1*)

Method which select some nuclei from the table according to some criteria.

Parameters

- **interp** (*str, optional. Default = 'n'.*) – If interp='n', exclude the interpolated nuclei from the selected ones. If interp='y' consider them in the table, in addition to the others.
- **state** (*str, optional. Default 'gs'.*) – select the kind of state. If state='gs', select nuclei measured in their ground state.
- **nucleus** (*str, optional. Default 'unstable'.*) – 'unstable'.

It can be set to 'stable', 'longlive' (with LT>10 min), 'shortlive' (with 10min>LT>1 ns), 'veryshortlive' (with LT< 1ns) :param every: consider only 1 out of *every* nuclei in the table. :type every: int, optional. Default every = 1.

Attributes:

select_year(*year_min=1940*, *year_max=1960*, *state='gs'*)

Method which select some nuclei from the table according to the discovery year.

Parameters

- **year_min**
- **year_max**
- **state** (*str, optional. Default 'gs'.*) – select the kind of state. If state='gs', select nuclei measured in their ground state.

Attributes:

nucleardatapy.setup_nuc_be_exp.**nuc_be_exp_tables**()

Return a list of the tables available in this toolkit for the experimental masses and print them all on the prompt. These tables are the following ones: 'AME'.

Returns
The list of tables.

Return type
list[str].

nucleardatapy.setup_nuc_be_exp.**nuc_be_exp_versions**(*table*)

Return a list of versions of tables available in this toolkit for a given model and print them all on the prompt.

Parameters
table (*str.*) – The table for which there are different versions.

Returns
The list of versions. If table == 'AME': '2020', '2016', '2012'.

Return type
list[str].

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupNucBEExp.py

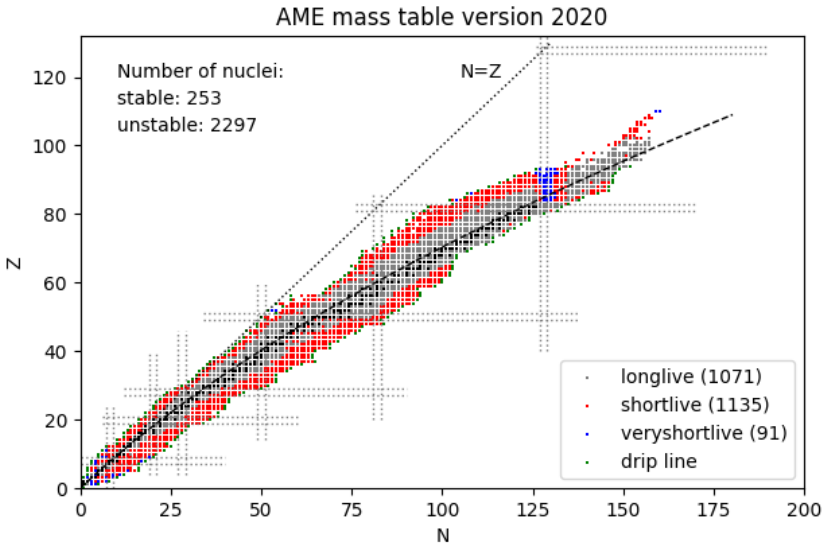

Fig. 30: The nuclear chart based on AME 2020 table. The different colors correspond to the different measured half-times of nuclei.

2.11 SetupNucBETheo

class nucleardatapy.setup_nuc_be_theo.**SetupNucBETheo**(*table='1995-DZ'*)

Instantiate the theory nuclear masses.

This choice is defined in the variable *table*.

table can chosen among the following ones: ['1988-MJ', '1995-DZ', '1995-ETFSI', '1995-FRDM', '2005-KTUY', '2007-HFB14', '2010-WS3', '2010-HFB21','2011-WS3', '2013-HFB26']

Parameters
table (*str, optional.*) – Fix the name of *table*. Default value: '1995-DZ'.

Attributes:

diff(*table*, *Zref=50*)

Method calculates the difference between a given mass model and table_ref.

Parameters

- **table** (*str.*) – Fix the table to analyze.
- **Zref** (*int, optional. Default: 50.*) – Fix the isotopic chain to study.

Attributes:

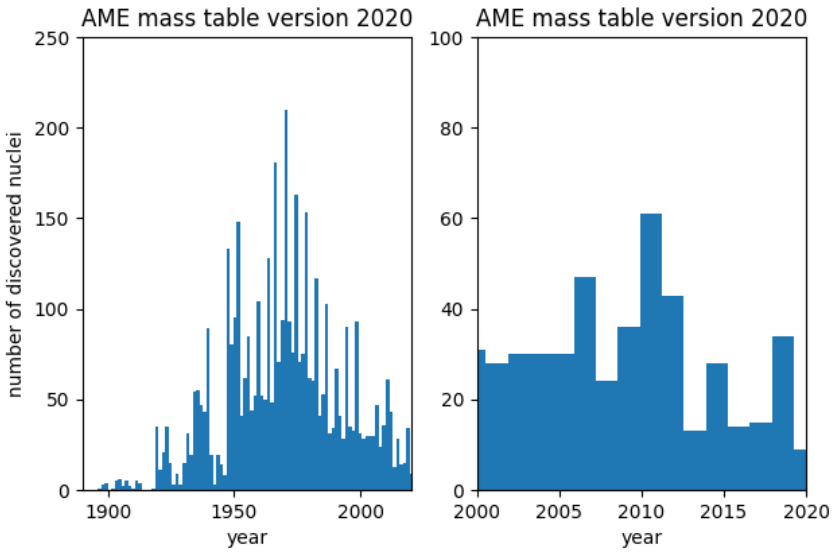

Fig. 31: Histogram showing the distribution of nuclei per discovery year, since the first one discovered in 1851.

diff_exp(*table_exp*, *version_exp*, *Zref=50*)

Method calculates the difference between a given experimental mass (identified by *table_exp* and *version_exp*) and table_ref.

Parameters

- **table** (*str.*) – Fix the table to analyze.
- **Zref** (*int, optional. Default: 50.*) – Fix the isotopic chain to study.

Attributes:

drip(*Zmax=95*)

Method which find the drip-line nuclei (on the two sides).

Parameters

Zmax (*int, optional. Default: 95.*) – Fix the maximum charge for the search of the drip line.

Attributes:

init_self()

Initialize variables in self.

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_nuc_be_theo.**nuc_be_theo_tables**()

Return a list of the tables available in this toolkit for the masses predicted by theoretical approaches and print them all on the prompt. These tables are the following ones: ['1988-MJ', '1995-DZ', '1995-ETFSI', '1995-FRDM', '2005-KTUY', '2007-HFB14', '2010-WS3', '2010-HFB21', '2011-WS3', '2013-HFB22', '2013-HFB23', '2013-HFB24', '2013-HFB25', '2013-HFB26', '2021-BSkG1', '2022-BSkG2', '2023-BSkG3']

Returns

The list of tables.

Return type
list[str].

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupNucBETheo.py

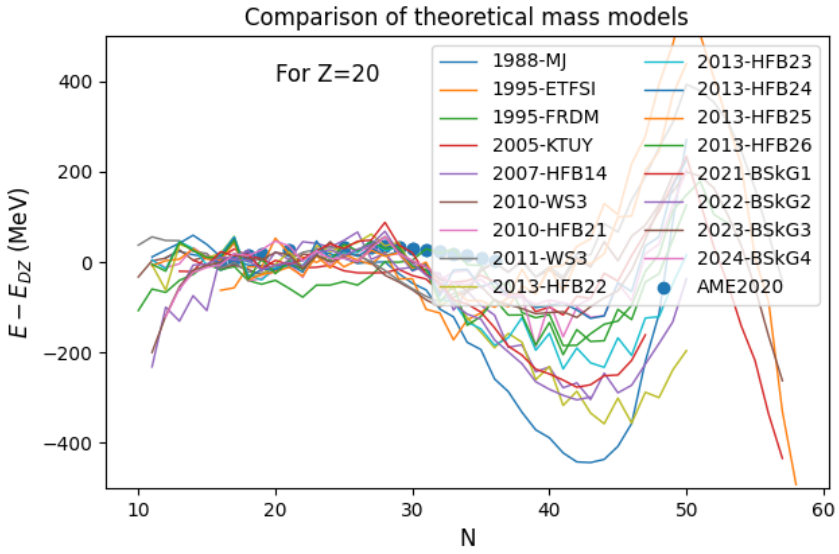

Fig. 32: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 20.

2.12 SetupNucRchExp

class nucleardatapy.setup_nuc_rch_exp.**SetupNucRchExp**(*table='2013-Angeli'*)

Instantiate the object with charge radii choosen from a table.

This choice is defined in the variable *table*.

The tables can chosen among the following ones: '2013-Angeli'.

Parameters
table (*str, optional.*) – Fix the name of *table*. Default value: '2013-Angeli'.

Attributes:

R_unit

Attribute radius unit.

Rch_isotopes(*Zref=50*)

This method provide a list if radii for an isotopic chain defined by Zref.

label

Attribute providing the label the data is references for figures.

note

Attribute providing additional notes about the data.

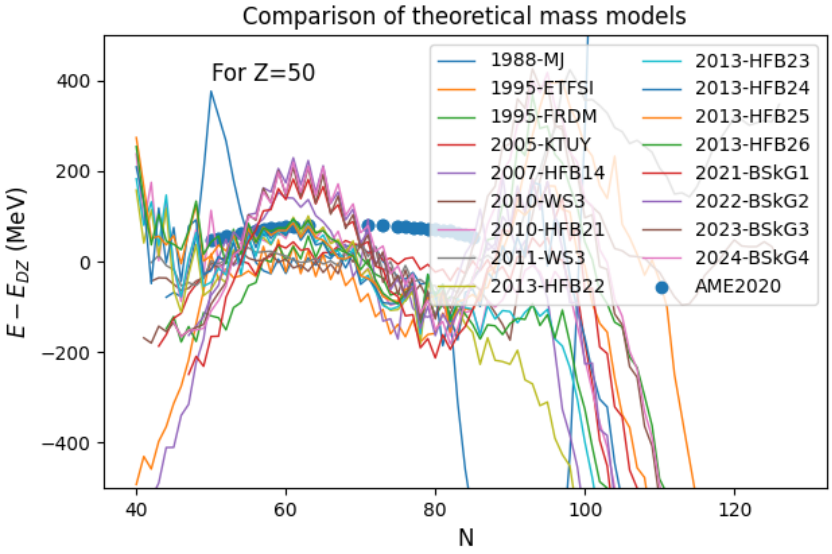

Fig. 33: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 50.

nucA

Attribute A (mass of the nucleus).

nucN

Attribute N (number of neutrons of the nucleus).

nucRch

Attribue R_ch (charge radius) in fm.

nucRch_err

Attribue uncertainty in R_ch (charge radius) in fm.

nucSymb

Attribute symb (symbol) of the element, e.g., Fe.

nucZ

Attribute Z (charge of the nucleus).

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

nucleardatapy.setup_nuc_rch_exp.**nuc_rch_exp_tables**()

Return a list of the tables available in this toolkit for the charge radiuus and print them all on the prompt. These tables are the following ones: '2013-Angeli'.

Returns

The list of tables.

Return type

list[str].

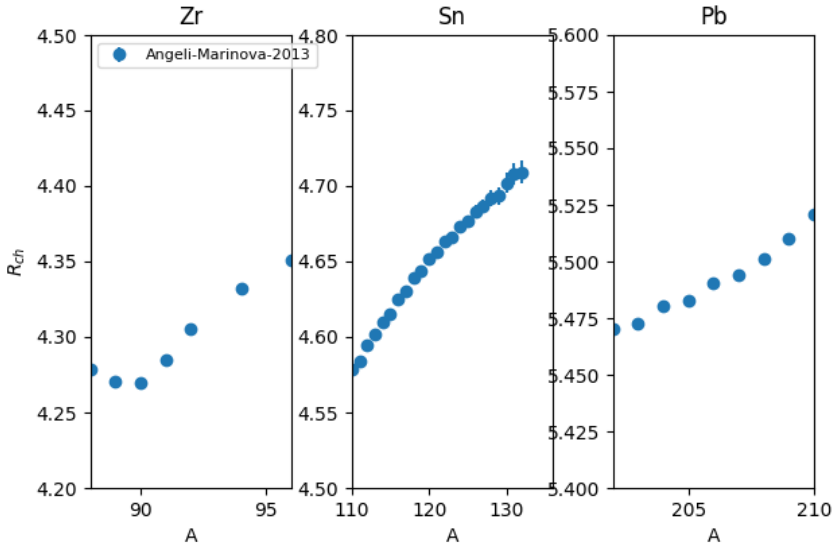

Fig. 34: Charge radii for Zn, Sn, and Pb isotopes and for the models available in the nuda toolkit.

2.13 SetupNuclSGMRExp

class nucleardatapy.setup_nuc_isgmr_exp.**SetupNucISGMRExp**(*table='2018-ISGMR-GARG'*)

Instantiate the object with microscopic results choosen by the toolkit practitioner. This choice is defined in the variable *table*.

The *table* can chosen among the following ones: '2010-ISGMR-LI', '2018-ISGMR-GARG'.

Parameters

table (*str, optional.*) – Fix the name of *table*. Default value: '2018-ISGMR-GARG', '2018-ISGMR-GARG-LATEX'.

Attributes:

E_unit

Attribute energy unit.

label

Attribute providing the label the data is references for figures.

note

Attribute providing additional notes about the data.

nucA

Attribute A (mass of the nucleus).

nucM12Mm1_cent

Attribute energy centroid.

nucM12Mm1_errm

Attribute (-) uncertainty in the energy centroid.

nucM12Mm1_errp

Attribute (+) uncertainty in the energy centroid.

nucSymbol

Attribute the symbol of the element.

nucZ

Attribute Z (charge of the nucleus).

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

table

Attribute table.

nucleardatapy.setup_nuc_isgmr_exp.**nuc_isgmr_exp_tables**()

Return a list of tables available in this toolkit for the ISGMR energy and print them all on the prompt. These tables are the following ones: '2010-ISGMR-LI', '2018-ISGMR-GARG', '2018-ISGMR-GARG-LATEX'.

Returns

The list of tables.

Return type

list[str].

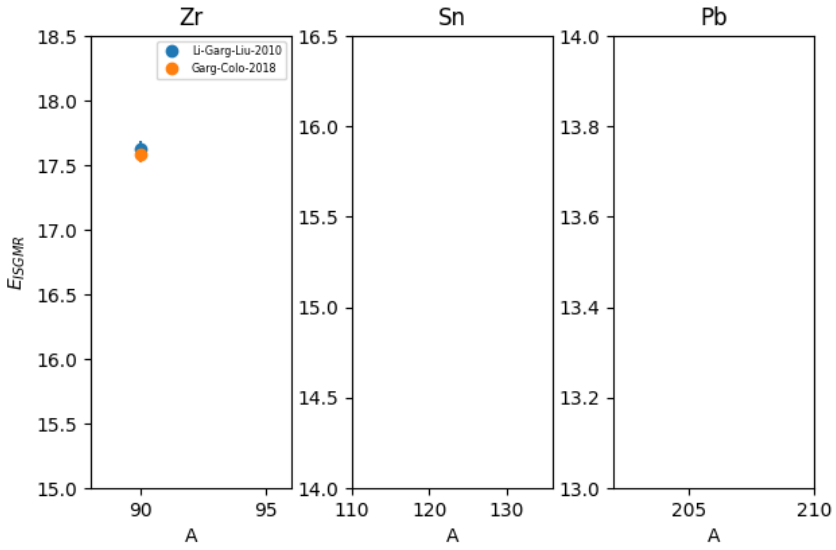

Fig. 35: Experimental ISGMR energies available in the nucleardatapy toolkit.

2.14 SetupCrust

class nucleardatapy.setup_crust.**SetupCrust**(*modcrust='1973-Negele-Vautherin'*)

Instantiate the properties of the crust for the existing models.

This choice is defined in the variable *crust*.

crust can chosen among the following ones: 'Negele-Vautherin-1973'.

Parameters
crust (*str, optional.*) – Fix the name of *crust*. Default value: 'Negele-Vautherin-1973'.

Attributes:

A

Attribute A (total number of nucleons of the WS cell).

A_bound

Attribute A_bound (mass of the cluster).

I_bound

Attribute the asymmetry parameter for bound particles.

N

Attribute N (total number of neutrons of the WS cell).

N_bound

Attribute N_bound (number of neutrons in the cluster).

N_g

Attribute N_g (number of neutrons in the gas).

RWS

Attribute the radius of the WS cell (in fm).

Rcl

Attribute the radius of the cluster (in fm).

Z

Attribute Z (total number of protons of the WS cell).

Z_bound

Attribute Z_bound (charge of the cluster).

den

Attribute the density of the system (in fm^-3).

den_cgs

Attribute the density of the system (in cm^-3).

den_g

Attribute the approximate density of neutron in the gas (in fm-3).

e2a_int

Attribute the internal energy (in MeV).

e2a_int2

Attribute the energy minus the neutron mass (in MeV).

e2a_int_g

Attribute the internal energy of the gas component (in MeV).

e2a_rm

Attribute the rest mass energy (in MeV).

mu_n

Attribute the neutron chemical potential (in MeV).

mu_p

Attribute the proton chemical potential (in MeV).

n_g

Attribute n_g (neutron density in the gas).

print_outputs()

Method which print outputs on terminal's screen.

xn

Attribute the fraction of neutrons.

xn_bound

Attribute the fraction of bound neutrons.

xp

Attribute the fraction of protons.

xp_bound

Attribute the fraction of bound protons.

xpn_bound

Attribute the approximate ratio of proton to neutron in the nucleus.

nucleardatapy.setup_crust.**models_crust**()

Return a list of the tables available in this toolkit for the experimental masses and print them all on the prompt. These tables are the following ones: 'Negele-Vautheron-1973'.

Returns
The list of tables.

Return type
list[str].

2.15 SetupAstroMasses

class nucleardatapy.setup_astro_masses.**SetupAstroMasses**(*source='J1614–2230'*, *obs=1*)

Instantiate the observational mass for a given source and obs.

This choice is defined in the variables *source* and *obs*.

source can chosen among the following ones: 'J1614–2230'.

obs depends on the chosen source.

Parameters

- **source** (*str, optional.*) – Fix the name of *source*. Default value: 'J1614–2230'.
- **obs** (*str, optional.*) – Fix the *obs*. Default value: 1.

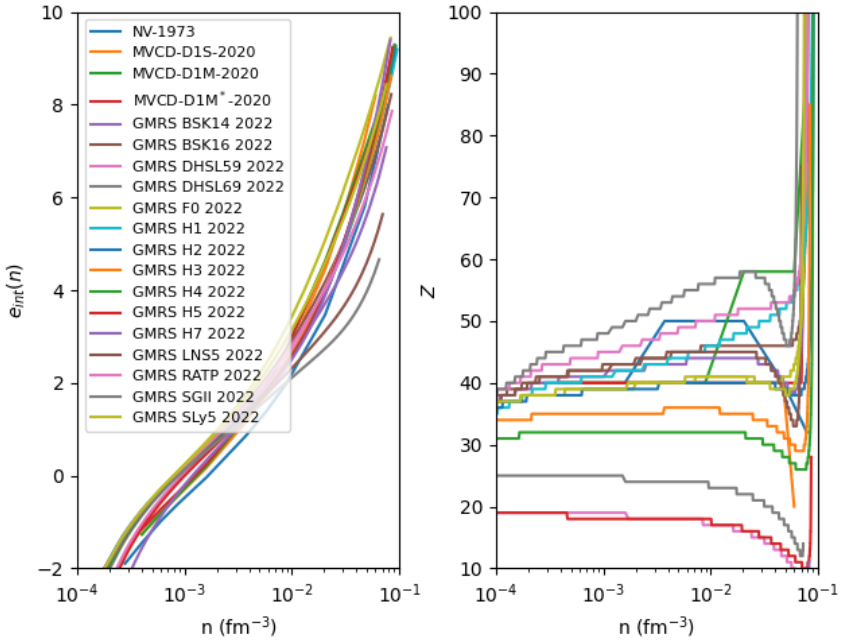

Fig. 36: Properties of the crust as given by the models available in the nuda toolkit.

Attributes:

label

Attribute providing the label the data is references for figures.

latexCite

Attribute latexCite.

mass

Attribute the observational mass of the source.

note

Attribute providing additional notes about the observation.

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

sig_do

Attribute the negative uncertainty.

sig_up

Attribute the positive uncertainty.

class nucleardatapy.setup_astro_masses.**SetupAstroMassesAverage**(*source='J1614–2230'*)

Instantiate the observational mass for a given source and averaged over obs.

This choice is defined in the variable *source*.

source can chosen among the following ones: 'J1614–2230'.

Parameters
source (*str, optional.*) – Fix the name of *source*. Default value: 'J1614–2230'.

Attributes:

print_outputs()
Method which print outputs on terminal's screen.

nucleardatapy.setup_astro_masses.**astro_masses**()
Return a list of the astrophysical sources for which a mass is given

Returns
The list of sources.

Return type
list[str].

nucleardatapy.setup_astro_masses.**astro_masses_source**(*source*)
Return a list of observations for a given source and print them all on the prompt.

Parameters
source (*str.*) – The source for which there are different observations.

Returns
The list of observations. If source == 'J1614–2230': 1, 2, 3, 4, 5.

Return type
list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy_plots/plot_setupAstroMasses.py

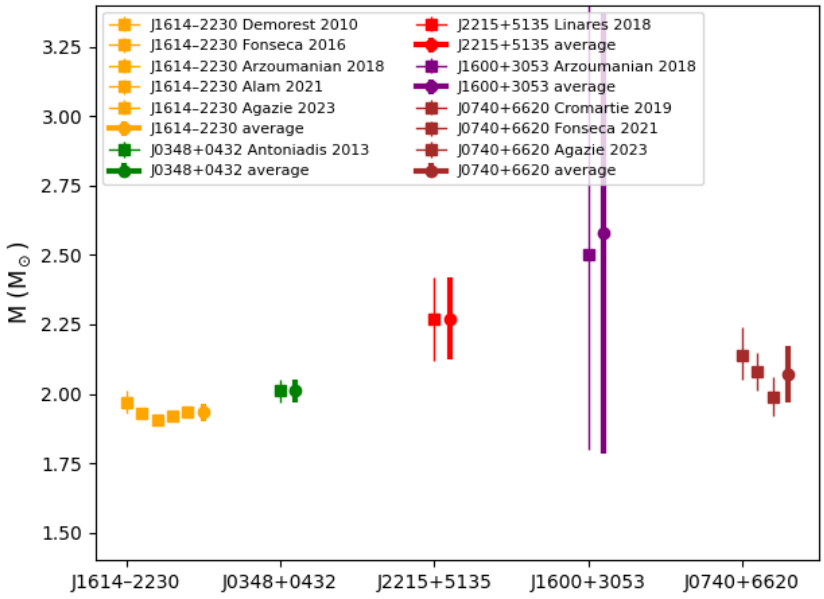

Fig. 37: The masses measured for massive neutron stars is radio-astronomy. The different colors correspond to the different sources.

2.16 SetupAstroMup

class nucleardatapy.setup_astro_mup.**SetupAstroMup**(*source='GW170817'*, *hyp=1*)

Instantiate the upper mass for a given source and hyptheses.

This choice is defined in the variables *source* and *hyp*.

source can chosen among the following ones: 'GW170817'.

hyp depends on the chosen hypotheses.

Parameters

- **source** (*str, optional.*) – Fix the name of *source*. Default value: 'GW170817'.
- **hyp** (*str, optional.*) – Fix the *hyp*. Default value: 'low-spin+TaylorF2'.

Attributes:

label

Attribute providing the label the data is references for figures.

latexCite

Attribute latexCite.

mup

Attribute the observational mass of the source.

note

Attribute providing additional notes about the observation.

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

sig_do

Attribute the negative uncertainty.

sig_up

Attribute the positive uncertainty.

class nucleardatapy.setup_astro_mup.**SetupAstroMupAverage**(*source='GW170817'*)

Instantiate the upper mass for a given source and averaged over hypotheses.

This choice is defined in the variable *source*.

source can chosen among the following ones: 'GW170817'.

Parameters

source (*str, optional.*) – Fix the name of *source*. Default value: 'GW170817'.

Attributes:

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_astro_mup.**astro_mup**()

Return a list of the astrophysical sources for which a mass is given

Returns

The list of sources.

Return type
list[str].

nucleardatapy.setup_astro_mup.**astro_mup_source**(*source*)

Return a list of observations for a given source and print them all on the prompt.

Parameters
source (*str.*) – The source for which there are different observations.

Returns
The list of observations. If source == 'J1614–2230': 1, 2, 3, 4, 5.

Return type
list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy_plots/plot_setupAstroMup.py

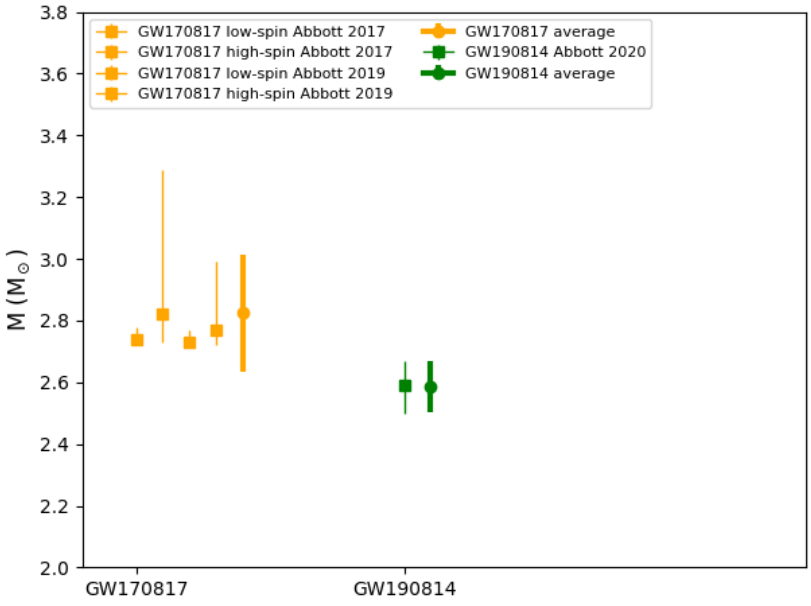

Fig. 38: The upper masses measured from GW observations. The different colors correspond to the different sources.

2.17 SetupAstroMtov

class nucleardatapy.setup_astro_mtov.**SetupAstroMtov**(*sources_do=array(['J1614–2230'], dtype='<U10'), sources_up=array(['GW170817'], dtype='<U8')*)

Instantiate the observational mass for a given source and obs.

This choice is defined in the variable *source*.

source can chosen among the following ones: 'J1614–2230'.

Parameters
source (*str, optional.*) – Fix the name of *source*. Default value: 'J1614–2230'.

Attributes:

print_outputs()

Method which print outputs on terminal's screen.

Here is a figure which is produced with the Python sample: /sample/nucleardatapy_plots/plot_setupAstroMtov.py

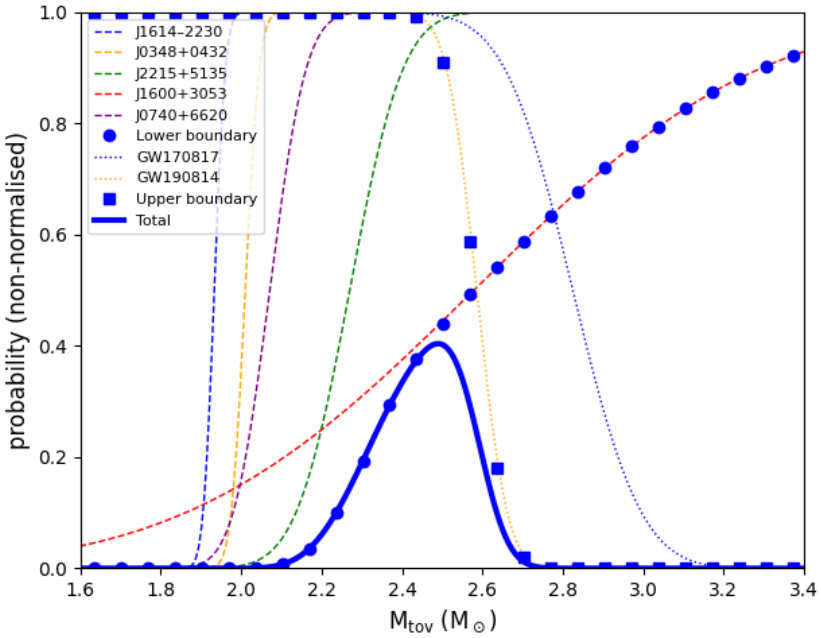

Fig. 39: The probability distribution function for the TOV mass constructed from radio and gravitational-wave observations. The different colors correspond to the different sources.

2.18 SetupAstroMR

class nucleardatapy.setup_astro_mr.**SetupAstroMR**(*source='J0030+0451'*, *obs=1*)

Instantiate the observational mass for a given source and obs.

This choice is defined in the variables *source* and *obs*.

source can chosen among the following ones: 'J1614–2230'.

obs depends on the chosen source.

Parameters

- **source** (*str, optional.*) – Fix the name of *source*. Default value: 'J1614–2230'.
- **obs** (*str, optional.*) – Fix the *obs*. Default value: 1.

Attributes:

label

Attribute providing the label the data is references for figures.

latexCite

Attribute latexCite.

mass

Attribute the observational mass of the source.

mass_sig_do

Attribute the negative uncertainty.

mass_sig_up

Attribute the positive uncertainty.

note

Attribute providing additional notes about the observation.

print_outputs()

Method which print outputs on terminal's screen.

rad

Attribute the observational mass of the source.

rad_sig_do

Attribute the negative uncertainty.

rad_sig_up

Attribute the positive uncertainty.

ref

Attribute providing the full reference to the paper to be citted.

class nucleardatapy.setup_astro_mr.**SetupAstroMRAverage**(*source='J1614–2230'*)

Instantiate the observational mass for a given source and averaged over obs.

This choice is defined in the variable *source*.

source can chosen among the following ones: 'J1614–2230'.

Parameters
source (*str, optional.*) – Fix the name of *source*. Default value: 'J1614–2230'.

Attributes:

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_astro_mr.**astro_mr**()

Return a list of the astrophysical sources for which a mass is given

Returns
The list of sources.

Return type
list[str].

nucleardatapy.setup_astro_mr.**astro_mr_source**(*source*)

Return a list of observations for a given source and print them all on the prompt.

Parameters
source (*str.*) – The source for which there are different observations.

Returns
The list of observations. If source == 'J1614–2230': 1, 2, 3, 4, 5.

Return type
list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy_plots/plot_setupAstroMR.py

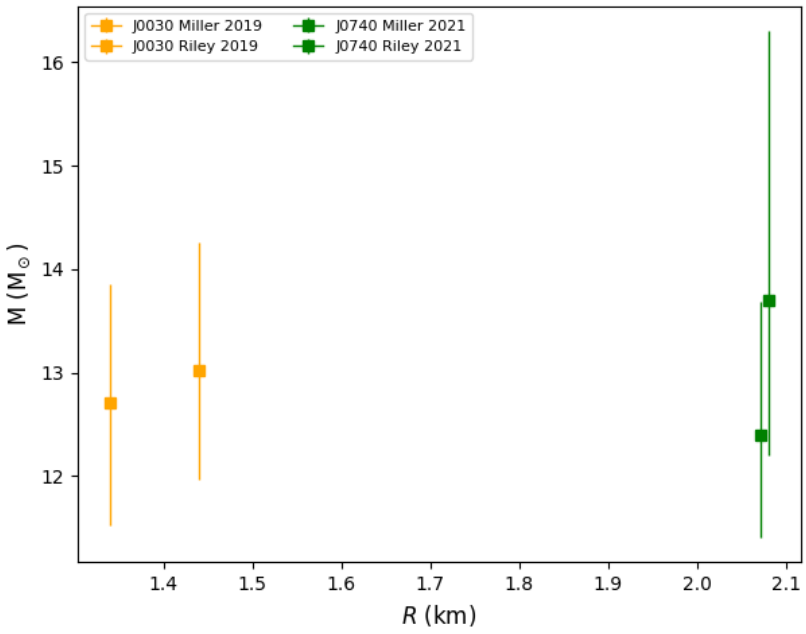

Fig. 40: The masses and radii measured by NICER. The different colors correspond to the different sources.

2.19 SetupAstroGW

class nucleardatapy.setup_astro_gw.**SetupAstroGW**(*source='GW170817'*, *hyp=1*)

Instantiate the tidal deformability for a given source and obs.

This choice is defined in the variables *source* and *obs*.

source can chosen among the following ones: 'GW170817'.

obs depends on the chosen source.

Parameters

- **source** (*str, optional.*) – Fix the name of *source*. Default value: 'GW170817'.
- **obs** (*str, optional.*) – Fix the *obs*. Default value: 1.

Attributes:

label

Attribute providing the label the data is references for figures.

lambda_sig_do

Attribute the upper bound of the tidal deformability for the source.

lambda_sig_up

Attribute the lower bound of the tidal deformability for the source.

latexCite

Attribute latexCite.

note

Attribute providing additional notes about the data.

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

class nucleardatapy.setup_astro_gw.**SetupAstroGWAverage**(*source='GW170817'*)

Instantiate the total mass for a given source and averaged over hypotheses.

This choice is defined in the variable *source*.

source can chosen among the following ones: 'GW170817'.

Parameters
source (*str, optional.*) – Fix the name of *source*. Default value: 'GW170817'.

Attributes:

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_astro_gw.**astro_gw**()

Return a list of the astrophysical sources for which a mass is given

Returns
The list of sources.

Return type
list[str].

nucleardatapy.setup_astro_gw.**astro_gw_source**(*source*)

Return a list of observations for a given source and print them all on the prompt.

Parameters
source (*str.*) – The source for which there are different hypotheses.

Returns
The list of hypotheses. If source == 'GW170817': 1, 2, 3, 4, 5.

Return type
list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy_plots/plot_setupAstroGW.py

2.20 SetupCorEsymLsym

nucleardatapy.setup_CorEsymLsym.**CorEsymLsym_constraints**()

Return a list of constraints available in this toolkit in the following list: '2009-HIC', '2010-RNP', '2012-FRDM', '2013-NS', '2014-IAS', '2014-IAS+RNP', '2015-POL-208PB', '2015-POL-120SN', '2015-POL-68NI', '2017-UG', '2021-PREXII-Reed', '2021-PREXII-Reinhard', '2023-PREXII+CREX-Zhang'; and print them all on the prompt.

Returns
The list of constraints.

Return type
list[str].

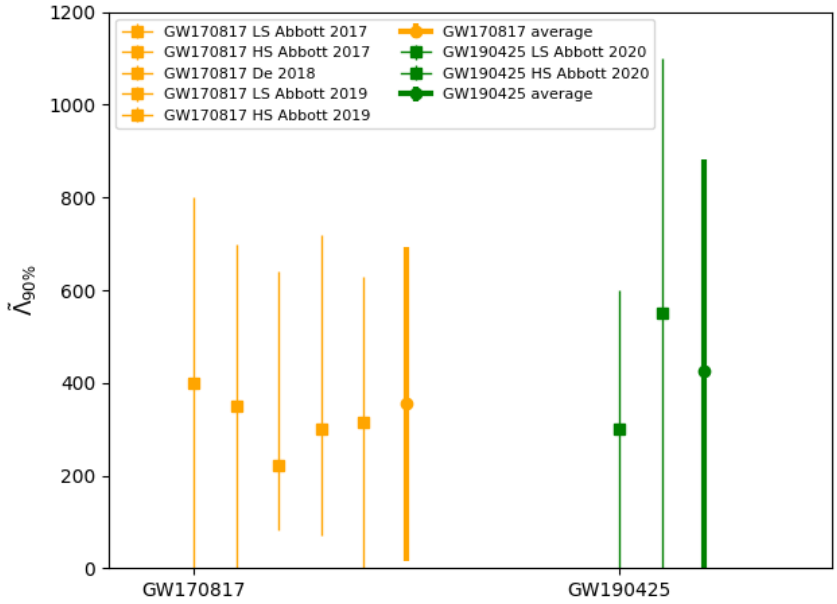

Fig. 41: Estimation of the effective tidal deformability from difference sources (different colors) and different hypotheses on the pulsar spin and the waveform employed for match filtering.

class nucleardatapy.setup_CorEsymLsym.**SetupCorEsymLsym**(*constraint='2014-IAS'*)

Instantiate the values of Esym and Lsym from the constraint.

The name of the constraint to be chosen in the following list: '2009-HIC', '2010-RNP', '2012-FRDM', '2013-NS', '2014-IAS', '2014-IAS+RNP', '2015-POL-208PB', '2015-POL-120SN', '2015-POL-68NI', '2017-UG', '2021-PREXII-Reed', '2021-PREXII-Reinhard', '2021-PREXII+CREX-Zhang'.

Parameters

constraint (*str, optional.*) – Fix the name of *constraint*. Default value: '2014-IAS'.

Attributes:

Esym

Attribute Esym.

Esym_err

Attribute with uncertainty in Esym.

Esym_max

Attribute max of Esym.

Esym_min

Attribute min of Esym.

Lsym

Attribute Lsym.

Lsym_err

Attribute with uncertainty in Lsym.

Lsym_max

Attribute max of Lsym.

Lsym_min

Attribute min of Lsym.

alpha

Attribute the plot alpha

constraint

Attribute constraint.

label

Attribute providing the label the data is references for figures.

note

Attribute providing additional notes about the constraint.

print_outputs()

Method which print outputs on terminal's screen.

ref

Attribute providing the full reference to the paper to be citted.

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupCorEsymLsym.py

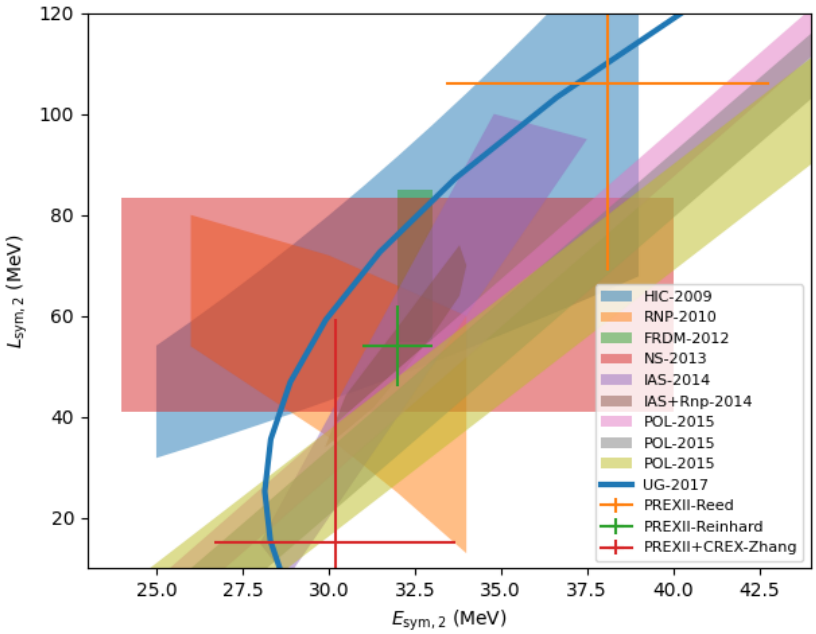

Fig. 42: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nucleardatapy toolkit.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

nucleardatapy, 4
nucleardatapy.setup_astro_gw, 47
nucleardatapy.setup_astro_masses, 40
nucleardatapy.setup_astro_mr, 45
nucleardatapy.setup_astro_mtov, 44
nucleardatapy.setup_astro_mup, 43
nucleardatapy.setup_CorEsymLsym, 48
nucleardatapy.setup_crust, 39
nucleardatapy.setup_eos_am, 24
nucleardatapy.setup_eos_beta, 24
nucleardatapy.setup_eos_esym, 28
nucleardatapy.setup_eos_ffg, 7
nucleardatapy.setup_eos_hic, 26
nucleardatapy.setup_eos_micro, 9
nucleardatapy.setup_eos_micro_band, 14
nucleardatapy.setup_eos_micro_lp, 15
nucleardatapy.setup_eos_pheno, 17
nucleardatapy.setup_nuc_be_exp, 29
nucleardatapy.setup_nuc_be_theo, 33
nucleardatapy.setup_nuc_isgmr_exp, 37
nucleardatapy.setup_nuc_rch_exp, 35

INDEX

A

A (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
A_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
alpha (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 50
astro_gw() (*in module nucleardatapy.setup_astro_gw*), 48
astro_gw_source() (*in module nucleardatapy.setup_astro_gw*), 48
astro_masses() (*in module nucleardatapy.setup_astro_masses*), 42
astro_masses_source() (*in module nucleardatapy.setup_astro_masses*), 42
astro_mr() (*in module nucleardatapy.setup_astro_mr*), 46
astro_mr_source() (*in module nucleardatapy.setup_astro_mr*), 46
astro_mup() (*in module nucleardatapy.setup_astro_mup*), 43
astro_mup_source() (*in module nucleardatapy.setup_astro_mup*), 44

C

constraint (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 50
constraint (*nucleardatapy.setup_eos_esym.SetupEOSEsym attribute*), 28
CorEsymLsym_constraints() (*in module nucleardatapy.setup_CorEsymLsym*), 48

D

delta (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
den (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
den (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
den (*nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand attribute*), 14
den() (*in module nucleardatapy.setup_eos_ffg*), 8
den_cgs (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
den_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
den_n (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
den_n() (*in module nucleardatapy.setup_eos_ffg*), 8
den_p (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
diff() (*nucleardatapy.setup_nuc_be_theo.SetupNucBETheo method*), 33
diff_exp() (*nucleardatapy.setup_nuc_be_theo.SetupNucBETheo method*), 33
dist_nbNuc (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 29
dist_year (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
drip() (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp method*), 31
drip() (*nucleardatapy.setup_nuc_be_theo.SetupNucBETheo method*), 34

E

e2a_int (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
e2a_int (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
e2a_int2 (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
e2a_int_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
e2a_rm (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
e2v_int (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
E_unit (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
ef_n (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7

eF_n() (*in module nucleardatapy.setup_eos_ffg*), 8
eF_p (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
effg() (*in module nucleardatapy.setup_eos_ffg*), 8
effg_NM() (*in module nucleardatapy.setup_eos_ffg*), 8
effg_SM() (*in module nucleardatapy.setup_eos_ffg*), 8
eos_hic_constraints() (*in module nucleardatapy.setup_eos_hic*), 28
eos_micro_LP_models() (*in module nucleardatapy.setup_eos_micro_lp*), 15
eos_micro_models() (*in module nucleardatapy.setup_eos_micro*), 10
eos_micro_models_group_NM() (*in module nucleardatapy.setup_eos_micro*), 11
eos_micro_models_group_SM() (*in module nucleardatapy.setup_eos_micro*), 11
eos_pheno_models() (*in module nucleardatapy.setup_eos_pheno*), 18
eos_pheno_params() (*in module nucleardatapy.setup_eos_pheno*), 19
Esym (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
esym (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
esym2 (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
esym4 (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 7
esym_e2a_max (*nucleardatapy.setup_eos_esym.SetupEOSEsym attribute*), 28
esym_e2a_min (*nucleardatapy.setup_eos_esym.SetupEOSEsym attribute*), 29
Esym_err (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
Esym_max (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
Esym_min (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
esymffg() (*in module nucleardatapy.setup_eos_ffg*), 8

F

flagI (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
flagInterp (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31

I

I_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
init_self() (*nucleardatapy.setup_eos_am.SetupEOSAM method*), 24
init_self() (*nucleardatapy.setup_eos_beta.SetupEOSBeta method*), 26
init_self() (*nucleardatapy.setup_eos_hic.SetupEOSHIC method*), 26
init_self() (*nucleardatapy.setup_eos_micro.SetupEOSMicro method*), 10
init_self() (*nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand method*), 14
init_self() (*nucleardatapy.setup_eos_micro_lp.SetupEOSMicroLP method*), 15
init_self() (*nucleardatapy.setup_eos_pheno.SetupEOSPheno method*), 18
init_self() (*nucleardatapy.setup_nuc_be_theo.SetupNucBETheo method*), 34

K

kf (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 8
kf() (*in module nucleardatapy.setup_eos_ffg*), 9
kf_n (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 8
kf_n() (*in module nucleardatapy.setup_eos_ffg*), 9
kf_p (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 8

L

label (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 47
label (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41
label (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 45
label (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43
label (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 50
label (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 8
label (*nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute*), 18
label (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31

label (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
label (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 35
lambda_sig_do (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 47
lambda_sig_up (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 47
latexCite (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 47
latexCite (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41
latexCite (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 45
latexCite (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43
Lsym (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
Lsym_err (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
Lsym_max (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49
Lsym_min (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 49

M

mass (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41
mass (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 45
mass_sig_do (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 45
mass_sig_up (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 46
matter (*nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand attribute*), 14
model (*nucleardatapy.setup_eos_am.SetupEOSAM attribute*), 24
model (*nucleardatapy.setup_eos_beta.SetupEOSBeta attribute*), 26
model (*nucleardatapy.setup_eos_micro.SetupEOSMicro attribute*), 10
model (*nucleardatapy.setup_eos_micro_lp.SetupEOSMicroLP attribute*), 15
model (*nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute*), 18
models (*nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand attribute*), 14
models_crust() (*in module nucleardatapy.setup_crust*), 40
module
    nucleardatapy, 4
    nucleardatapy.setup_astro_gw, 47
    nucleardatapy.setup_astro_masses, 40
    nucleardatapy.setup_astro_mr, 45
    nucleardatapy.setup_astro_mtov, 44
    nucleardatapy.setup_astro_mup, 43
    nucleardatapy.setup_CorEsymLsym, 48
    nucleardatapy.setup_crust, 39
    nucleardatapy.setup_eos_am, 24
    nucleardatapy.setup_eos_beta, 24
    nucleardatapy.setup_eos_esym, 28
    nucleardatapy.setup_eos_ffg, 7
    nucleardatapy.setup_eos_hic, 26
    nucleardatapy.setup_eos_micro, 9
    nucleardatapy.setup_eos_micro_band, 14
    nucleardatapy.setup_eos_micro_lp, 15
    nucleardatapy.setup_eos_pheno, 17
    nucleardatapy.setup_nuc_be_exp, 29
    nucleardatapy.setup_nuc_be_theo, 33
    nucleardatapy.setup_nuc_isgmr_exp, 37
    nucleardatapy.setup_nuc_rch_exp, 35
mu_n (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
mu_p (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
mup (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43

N

N (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
N_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
N_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
n_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
nbLine (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nbNuc (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nden (*nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand attribute*), 14
note (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 47
note (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41

note (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 46
note (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43
note (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 50
note (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 8
note (*nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute*), 18
note (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
note (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
note (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 35
nuc_be_exp_tables() (*in module nucleardatapy.setup_nuc_be_exp*), 32
nuc_be_exp_versions() (*in module nucleardatapy.setup_nuc_be_exp*), 32
nuc_be_theo_tables() (*in module nucleardatapy.setup_nuc_be_theo*), 34
nuc_isgmr_exp_tables() (*in module nucleardatapy.setup_nuc_isgmr_exp*), 38
nuc_rch_exp_tables() (*in module nucleardatapy.setup_nuc_rch_exp*), 36
nucA (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucA (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
nucA (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 35
nucBE (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucBE_err (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucHT (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucleardatapy
    module, 4
nucleardatapy.setup_astro_gw
    module, 47
nucleardatapy.setup_astro_masses
    module, 40
nucleardatapy.setup_astro_mr
    module, 45
nucleardatapy.setup_astro_mtov
    module, 44
nucleardatapy.setup_astro_mup
    module, 43
nucleardatapy.setup_CorEsymLsym
    module, 48
nucleardatapy.setup_crust
    module, 39
nucleardatapy.setup_eos_am
    module, 24
nucleardatapy.setup_eos_beta
    module, 24
nucleardatapy.setup_eos_esym
    module, 28
nucleardatapy.setup_eos_ffg
    module, 7
nucleardatapy.setup_eos_hic
    module, 26
nucleardatapy.setup_eos_micro
    module, 9
nucleardatapy.setup_eos_micro_band
    module, 14
nucleardatapy.setup_eos_micro_lp
    module, 15
nucleardatapy.setup_eos_pheno
    module, 17
nucleardatapy.setup_nuc_be_exp
    module, 29
nucleardatapy.setup_nuc_be_theo
    module, 33
nucleardatapy.setup_nuc_isgmr_exp
    module, 37
nucleardatapy.setup_nuc_rch_exp
    module, 35
nucM12Mm1_cent (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
nucM12Mm1_errm (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
nucM12Mm1_errp (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 37
nucN (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucN (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 36
nucRch (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 36
nucRch_err (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 36
nucStbl (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucSymb (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucSymb (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 36
nucSymbol (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 38

nucYear (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucZ (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 31
nucZ (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 38
nucZ (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 36

P

param (*nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute*), 18
pre (*nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute*), 8
print_outputs() (*nucleardatapy.setup_astro_gw.SetupAstroGW method*), 47
print_outputs() (*nucleardatapy.setup_astro_gw.SetupAstroGWAverage method*), 48
print_outputs() (*nucleardatapy.setup_astro_masses.SetupAstroMasses method*), 41
print_outputs() (*nucleardatapy.setup_astro_masses.SetupAstroMassesAverage method*), 42
print_outputs() (*nucleardatapy.setup_astro_mr.SetupAstroMR method*), 46
print_outputs() (*nucleardatapy.setup_astro_mr.SetupAstroMRAverage method*), 46
print_outputs() (*nucleardatapy.setup_astro_mtov.SetupAstroMtov method*), 44
print_outputs() (*nucleardatapy.setup_astro_mup.SetupAstroMup method*), 43
print_outputs() (*nucleardatapy.setup_astro_mup.SetupAstroMupAverage method*), 43
print_outputs() (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym method*), 50
print_outputs() (*nucleardatapy.setup_crust.SetupCrust method*), 40
print_outputs() (*nucleardatapy.setup_eos_am.SetupEOSAM method*), 24
print_outputs() (*nucleardatapy.setup_eos_beta.SetupEOSBeta method*), 26
print_outputs() (*nucleardatapy.setup_eos_esym.SetupEOSEsym method*), 29
print_outputs() (*nucleardatapy.setup_eos_ffg.SetupEOSFFG method*), 8
print_outputs() (*nucleardatapy.setup_eos_hic.SetupEOSHIC method*), 28
print_outputs() (*nucleardatapy.setup_eos_micro.SetupEOSMicro method*), 10
print_outputs() (*nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand method*), 15
print_outputs() (*nucleardatapy.setup_eos_micro_lp.SetupEOSMicroLP method*), 15
print_outputs() (*nucleardatapy.setup_eos_pheno.SetupEOSPheno method*), 18
print_outputs() (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp method*), 32
print_outputs() (*nucleardatapy.setup_nuc_be_theo.SetupNucBETheo method*), 34
print_outputs() (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp method*), 38
print_outputs() (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp method*), 36

R

R_unit (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 35
rad (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 46
rad_sig_do (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 46
rad_sig_up (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 46
Rch_isotopes() (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp method*), 35
Rcl (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
ref (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 48
ref (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41
ref (*nucleardatapy.setup_astro_mr.SetupAstroMR attribute*), 46

ref (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43
ref (*nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym attribute*), 50
ref (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 32
ref (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 38
ref (*nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute*), 36
RWS (*nucleardatapy.setup_crust.SetupCrust attribute*), 39

S

select() (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp method*), 32
select_year() (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp method*), 32
SetupAstroGW (*class in nucleardatapy.setup_astro_gw*), 47
SetupAstroGWAverage (*class in nucleardatapy.setup_astro_gw*), 48
SetupAstroMasses (*class in nucleardatapy.setup_astro_masses*), 40
SetupAstroMassesAverage (*class in nucleardatapy.setup_astro_masses*), 41
SetupAstroMR (*class in nucleardatapy.setup_astro_mr*), 45
SetupAstroMRAverage (*class in nucleardatapy.setup_astro_mr*), 46
SetupAstroMtov (*class in nucleardatapy.setup_astro_mtov*), 44
SetupAstroMup (*class in nucleardatapy.setup_astro_mup*), 43
SetupAstroMupAverage (*class in nucleardatapy.setup_astro_mup*), 43
SetupCorEsymLsym (*class in nucleardatapy.setup_CorEsymLsym*), 48
SetupCrust (*class in nucleardatapy.setup_crust*), 39
SetupEOSAM (*class in nucleardatapy.setup_eos_am*), 24
SetupEOSBeta (*class in nucleardatapy.setup_eos_beta*), 24
SetupEOSEsym (*class in nucleardatapy.setup_eos_esym*), 28
SetupEOSFFG (*class in nucleardatapy.setup_eos_ffg*), 7
SetupEOSHIC (*class in nucleardatapy.setup_eos_hic*), 26
SetupEOSMicro (*class in nucleardatapy.setup_eos_micro*), 9
SetupEOSMicroBand (*class in nucleardatapy.setup_eos_micro_band*), 14
SetupEOSMicroLP (*class in nucleardatapy.setup_eos_micro_lp*), 15
SetupEOSPheno (*class in nucleardatapy.setup_eos_pheno*), 17
SetupNucBEExp (*class in nucleardatapy.setup_nuc_be_exp*), 29
SetupNucBETheo (*class in nucleardatapy.setup_nuc_be_theo*), 33
SetupNucISGMRExp (*class in nucleardatapy.setup_nuc_isgmr_exp*), 37
SetupNucRchExp (*class in nucleardatapy.setup_nuc_rch_exp*), 35
sig_do (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41
sig_do (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43
sig_up (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 41
sig_up (*nucleardatapy.setup_astro_mup.SetupAstroMup attribute*), 43

T

table (*nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute*), 38

X

xn (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
xn_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
xp (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
xp_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 40
xpn_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 40

Z

Z (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
Z_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 39
Zmax (*nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute*), 29