nucleardatapy

Release 0.2.0

Jérôme Margueron, IRL NPA, USA

Mar 12, 2025

CONTENTS

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 Matter

2.1.1 matter.setupFFG

nucleardatapy.matter.setup_ffg.**den**(*kf*)

Density as a function of the Fermi momentum.

Parameters

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

nucleardatapy.matter.setup_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.matter.setup_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.matter.setup_ffg.**eF_n_nr**(*kf_n*)

Non-relativistic 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.matter.setup_ffg.**effg_NM_nr**(*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.matter.setup_ffg.**effg_SM_nr**(*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.matter.setup_ffg.**effg_nr**(*kf*)

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

Parameters

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

`nucleardatapy.matter.setup_ffg.`**`esymffg_nr`**`(`*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.matter.setup_ffg.`**`kf`**`(`*den*`)`

Fermi momentum as a function of the density.

Parameters

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

`nucleardatapy.matter.setup_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.

`class nucleardatapy.matter.setup_ffg.`**`setupFFGLep`**`(`*den_el*, *den_mu*`)`

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:

Parameters

- **den_e** (*float or numpy array of floats.*)
- **component.** (*Density or densities for the muon*)
- **den_mu** (*float or numpy array of floats.*)
- **component.**

`den_el`

Attribute electron density

`den_lep`

Attribute lepton density

`den_mu`

Attribute muon density

`e2a_el`

Attribute FFG energy per particle (degeneracy = 2)

`e2v_el`

Attribute FFG energy per particle (degeneracy = 2)

`eF_el`

Attribute electon Fermi energy (degeneracy = 2)

`eF_mu`

Attribute muon Fermi energy (degeneracy = 2)

`h2v_el`
: Attribute enthalpy

`kf_el`
: Attribute electron Fermi momentum (degeneracy = 2)

`kf_mu`
: Attribute muon Fermi momentum (degeneracy = 2)

`label`
: Attribute providing the label the data is references for figures.

`note`
: Attribute providing additional notes about the data.

`pre_el`
: Attribute FFG pressure (degeneracy = 2)

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

`x_el`
: Attribute electron fraction

`x_mu`
: Attribute muon fraction

class `nucleardatapy.matter.setup_ffg.setupFFGNuc`(*den*, *delta*, *ms=1.0*)

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:

Parameters

- **den** (*float or numpy array of floats.*)
- **calculated.** (*Isospin density or densities for which the FFG quantities are*)
- **delta** (*float or numpy array of floats.*)
- **calculated.**
- **ms** (*effective mass in unit of mass.*)

`delta`
: Attribute isospin parameter

`den`
: Attribute isoscalar density

`den_n`
: Attribute neutron density

`den_p`
: Attribute proton density

`e2a_rm`
: Attribute rest mass energy per particle (degeneracy = 2)

`e2v_int`
: Attribute FFG energy per unit volum (degeneracy = 2)

`eF_n`
: Attribute neutron Fermi energy (degeneracy = 2)

`eF_p`
: Attribute proton Fermi energy (degeneracy = 2)

`esym2_nr`
: Attribute FFG quadratic contribution to the symmetry energy

`esym4_nr`
: Attribute FFG quartic contribution to the symmetry energy

`esym_nr`
: Attribute FFG symmetry energy (degeneracy = 2)

`h2a`
: Attribute enthalpy

`kf`
: Attribute Fermi momentum for a Fermi system with degeneracy = 4

`kf_n`
: Attribute neutron Fermi momentum (degeneracy = 2)

`kf_p`
: Attribute proton Fermi momentum (degeneracy = 2)

`label`
: Attribute providing the label the data is references for figures.

`ms`
: Attribute the effective mass in unit of mass.

`note`
: Attribute providing additional notes about the data.

`pre`
: Attribute FFG pressure (degeneracy = 2)

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

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_matter_setupFFGNuc.py

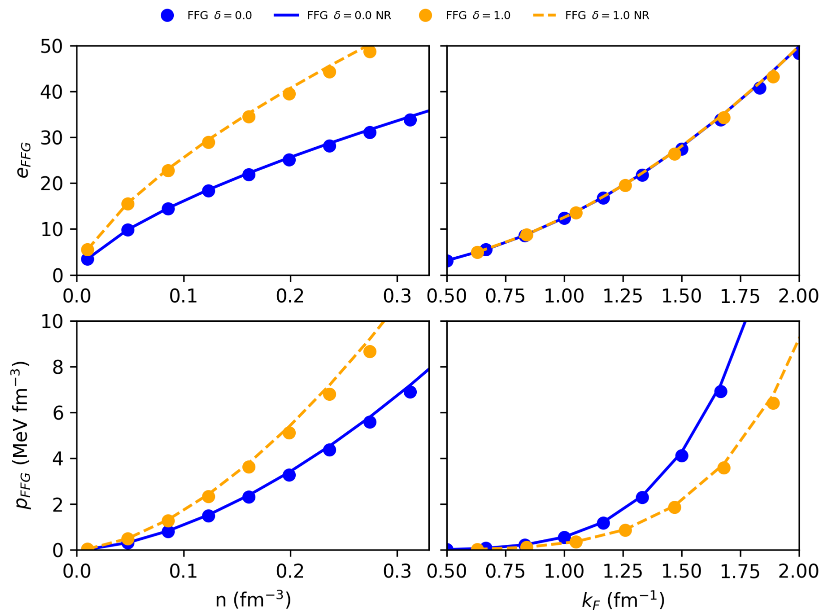

Fig. 1: This figure shows the free Fermi gas energy (top) and pressure (bottom) in symmetric matter (SM) (Blue solid line) and neutron matter (NM) (orange dashed line) as function of the particle density (left) and Fermi momentum (right).

2.1.2 matter.setupMicro

`nucleardatapy.matter.setup_micro.`**`micro_mbs`**`()`

Return a list of many-bodys (mbs) approaches available in this toolkit and print them all on the prompt.

Returns
The list of models with can be 'VAR', 'AFDMC', 'BHF', 'QMC', 'MBPT', 'NLEFT'.

Return type
list[str].

`nucleardatapy.matter.setup_micro.`**`micro_models_mb`**`(`*mb*`)`

Return a list with the name of the models available in this toolkit for a given mb appoach and print them all on the prompt.

Parameters
mb (*str.*) – The mb approach for which there are parametrizations. They should be chosen among the following options: 'VAR', 'AFDMC', 'BHF', 'QMC', 'MBPT', 'NLEFT'.

Returns
The list of parametrizations.

These models are the following ones: If *mb* == 'VAR': '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-APR-fit', If *mb* == 'AFDMC': '2012-AFDMC-NM-RES-1', '2012-AFDMC-NM-RES-2', '2012-AFDMC-NM-RES-3', '2012-AFDMC-NM-RES-4', '2012-AFDMC-NM-RES-5', '2012-AFDMC-NM-RES-6', '2012-AFDMC-NM-RES-7', '2012-AFDMC-NM-FIT-1', '2012-AFDMC-NM-FIT-2', '2012-AFDMC-NM-FIT-3', '2012-AFDMC-NM-FIT-4', '2012-AFDMC-NM-FIT-5', '2012-AFDMC-NM-FIT-6', '2012-AFDMC-NM-FIT-7', '2022-AFDMC-NM', If *mb* == 'BHF': '2006-BHF-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', If *mb* == 'QMC': '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2024-QMC-NM', If *mb* == 'MBPT': '2010-MBPT-NM', '2020-MBPT-AM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69' If *mb* == 'NLEFT': '2024-NLEFT-AM',

`nucleardatapy.matter.setup_micro.`**`micro_models_mb_NM`**(*mb*)

`nucleardatapy.matter.setup_micro.`**`micro_models_mb_SM`**(*mb*)

`nucleardatapy.matter.setup_micro.`**`micro_models_mb_matter`**(*mb*, *matter*)

matter can be 'sm', 'SM' or 'nm', 'NM'

`nucleardatapy.matter.setup_micro.`**`micro_models_old`**()

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-APR-fit', '2006-BHF-AM*', '2012-AFDMC-NM-RES-1', '2012-AFDMC-NM-RES-2', '2012-AFDMC-NM-RES-3', '2012-AFDMC-NM-RES-4', '2012-AFDMC-NM-RES-5', '2012-AFDMC-NM-RES-6', '2012-AFDMC-NM-RES-7', '2012-AFDMC-NM-FIT-1', '2012-AFDMC-NM-FIT-2', '2012-AFDMC-NM-FIT-3', '2012-AFDMC-NM-FIT-4', '2012-AFDMC-NM-FIT-5', '2012-AFDMC-NM-FIT-6', '2012-AFDMC-NM-FIT-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', '2006-BHF-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', '2024-QMC-NM'

Returns
The list of models.

Return type
list[str].

`class nucleardatapy.matter.setup_micro.`**`setupMicro`**(*model='1998-VAR-AM-APR'*, *var1=array([0.01, 0.03052632, 0.05105263, 0.07157895, 0.09210526, 0.11263158, 0.13315789, 0.15368421, 0.17421053, 0.19473684, 0.21526316, 0.23578947, 0.25631579, 0.27684211, 0.29736842, 0.31789474, 0.33842105, 0.35894737, 0.37947368, 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-APR-fit', '2006-BHF-AM*', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2012-AFDMC-NM-RES-1', '2012-AFDMC-NM-RES-2', '2012-AFDMC-NM-RES-3', '2012-AFDMC-NM-RES-4', '2012-AFDMC-NM-RES-5', '2012-AFDMC-NM-RES-6', '2012-AFDMC-NM-RES-7', '2012-AFDMC-NM-FIT-1', '2012-AFDMC-NM-FIT-2', '2012-AFDMC-NM-FIT-3', '2012-AFDMC-NM-FIT-4', '2012-AFDMC-NM-FIT-5', '2012-AFDMC-NM-FIT-6', '2012-AFDMC-NM-FIT-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', '2024-QMC-NM'

Parameters

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

Attributes:

Parameters

- **model** (*str, optional*)
- **between** (*The model to consider. Choose*)
- **var2** (*var1 and*)
- **np.array([0.1** (*var1 =*)
- **0.15**
- **0.16**
- **0.17**
- **0.2**
- **0.25])**

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: /nucleardatapy_sample/plots/plot_matter_setupMicro.py

2.1.3 matter.setupMicroBand

class nucleardatapy.matter.setup_micro_band.**setupMicroBand**(*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.

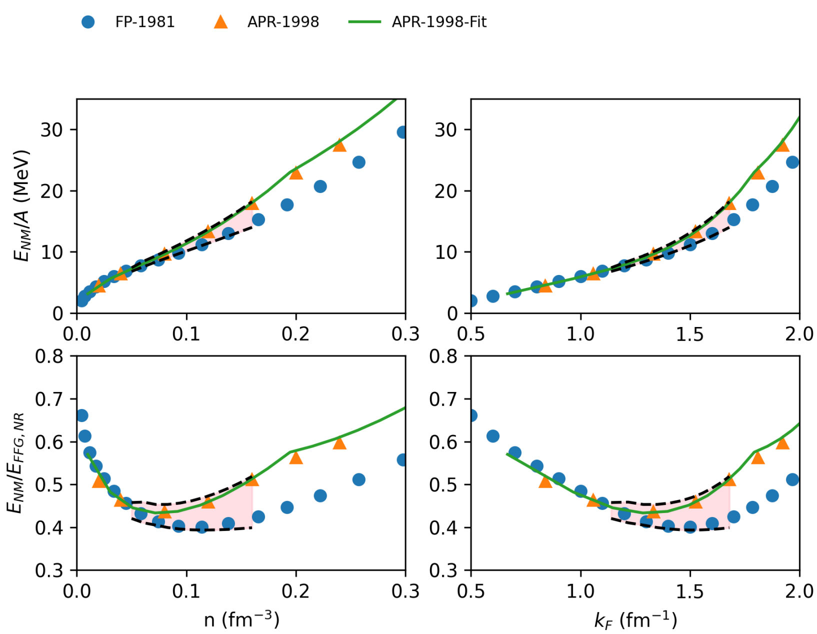

Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the variational models available in the nucleardatapy toolkit.

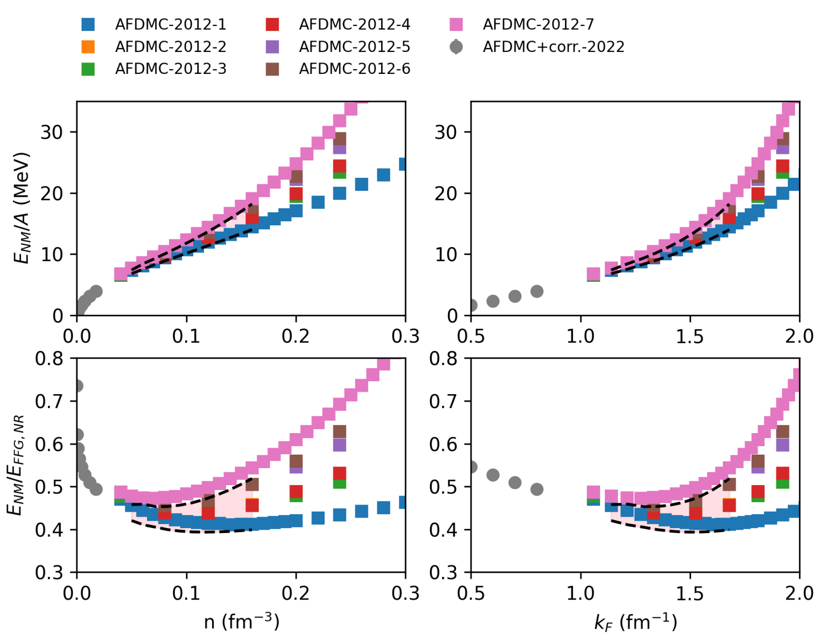

Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

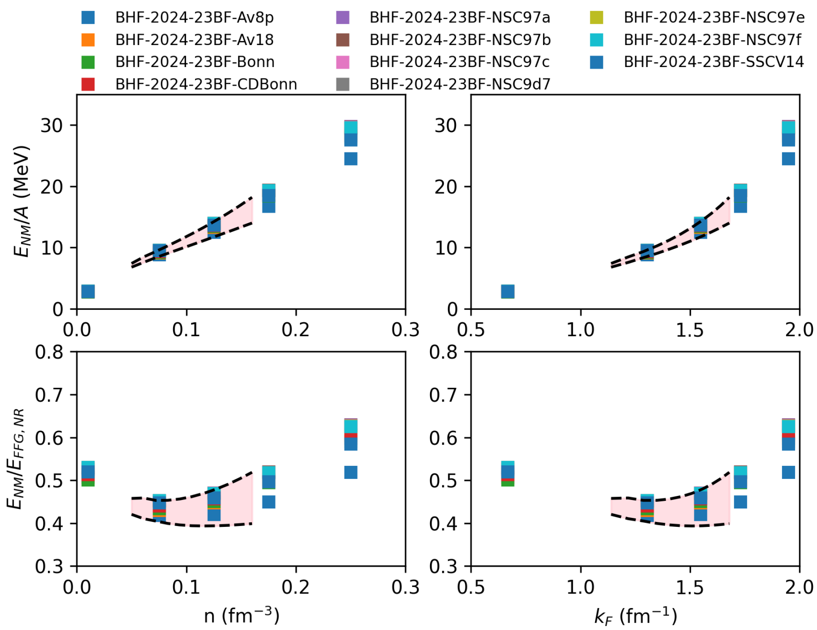

Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

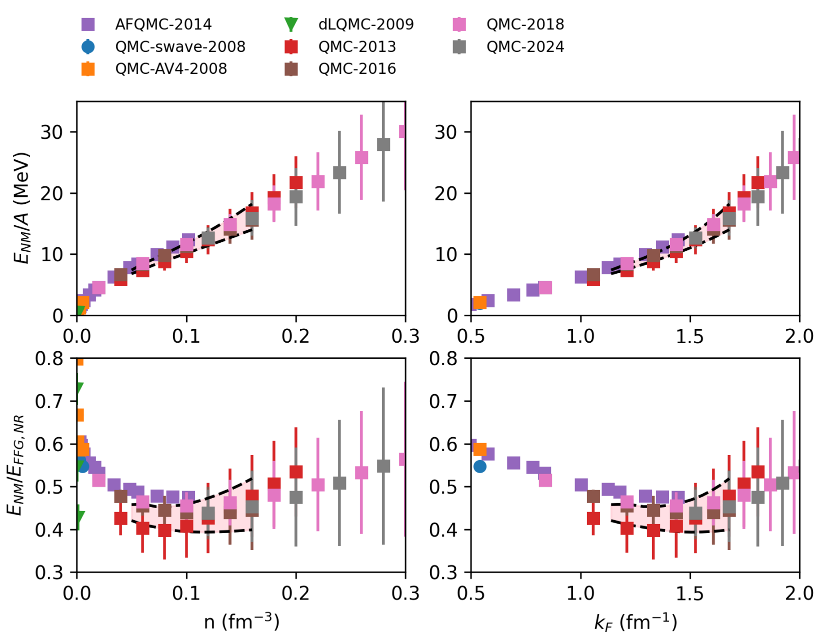

Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

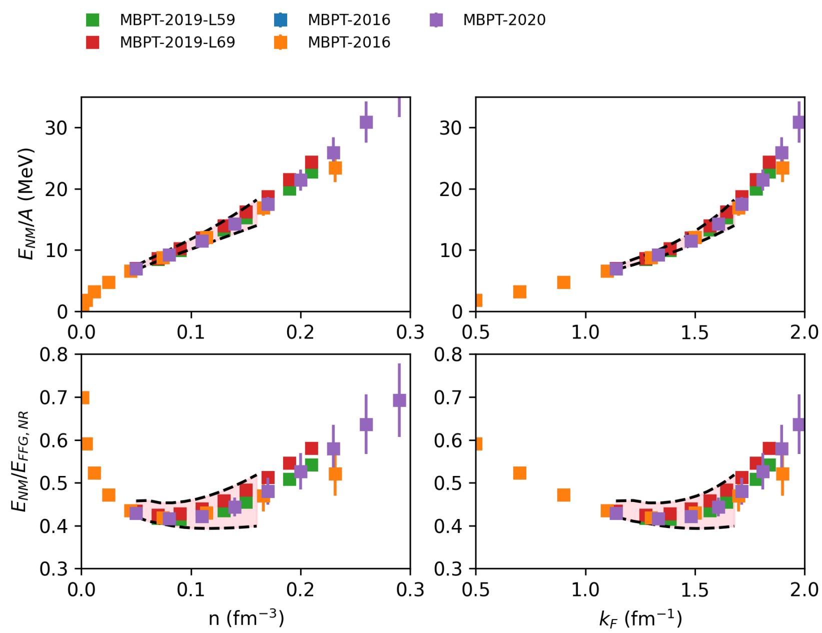

Fig. 6: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.

- **den** (*None or numpy array, optional.*) – if not None (default), impose the densities.
- **matter** (*str, optional.*) – can be 'NM' (default), 'SM' or 'ESYM'.

Attributes:

Parameters

- **model** (*str, optional.*)
- **between** (*The model to consider. Choose*)
- **nden** (*int, optional.*)
- **consider.** (*The density points to*)
- **ne** (*int, optional.*)
- **direction.** (*The number of intervalle in the energy*)
- **den** (*None or numpy array.*)
- **None** (*If*)
- **densities** (*then the density range is calculated automaticaly. If den = list of*)
- **them.** (*the code will prefer using*)
- **matter** (*'SM' symmetric*)
- **matter**
- **matter**
- **energy.** (*or 'Esym' the symmetry*)

- **e2a_min** (*float, optional.*)
- **default** (*e2a_max is set to be 50 MeV by*)
- **practitionner.** (*or any number passed by the*)
- **e2a_max** (*float, optional.*)
- **default**
- **practitionner.**

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.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_matter_setupMicroBand.py

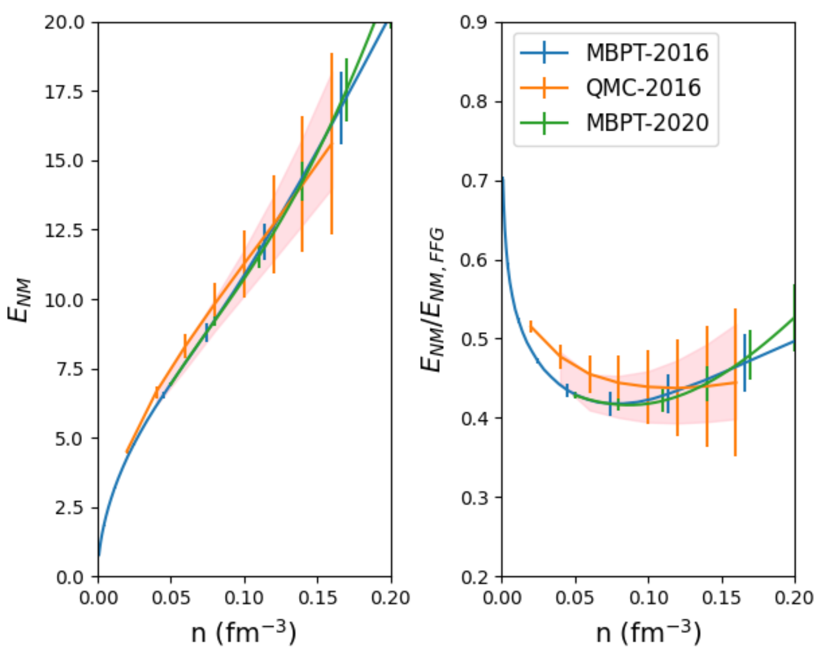

Fig. 7: Uncertainty band in NM obtained from the analysis of different predictions: MBPT-2016, QMC-2016 and MBPT-2020.

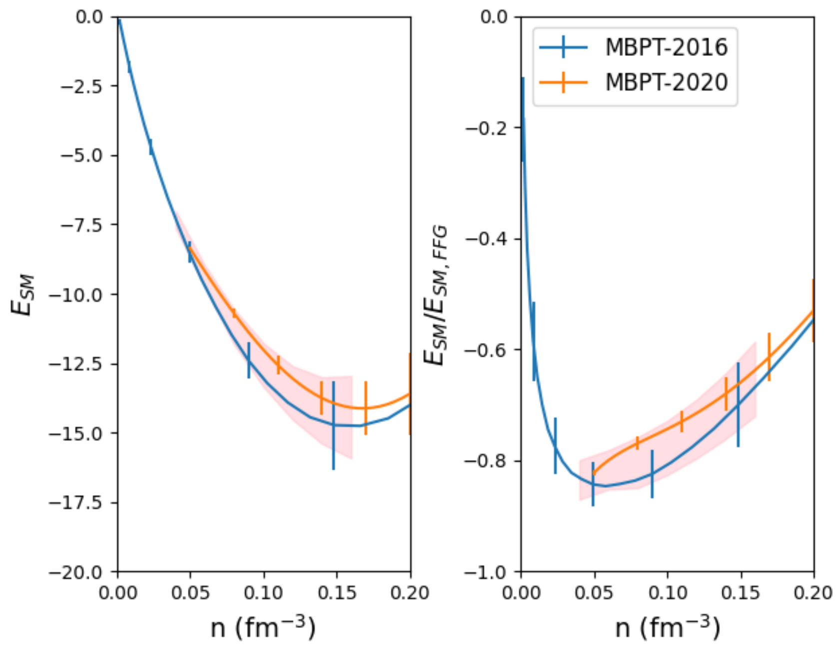

Fig. 8: Uncertainty band in SM obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

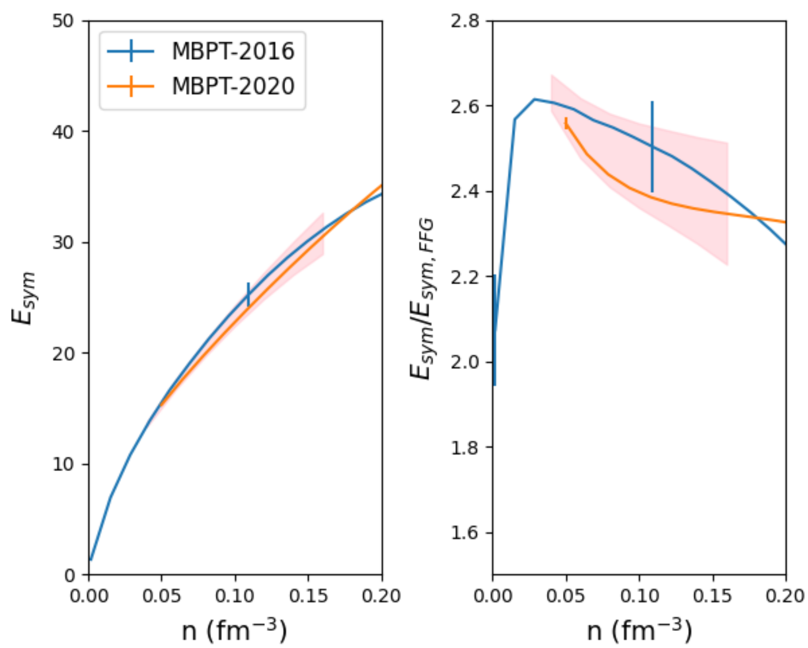

Fig. 9: Uncertainty band for the symmetry energy obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

2.1.4 matter.setupMicroLP

`nucleardatapy.matter.setup_micro_lp.`**`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', '2006-BHF-SM-AV18', '2006-BHF-NM-AV18', '2006-IBHF-SM-AV18', '2006-IBHF-NM-AV18', '2007-BHF-NM-LP-BONNC'.

Returns
The list of models.

Return type
list[str].

`class nucleardatapy.matter.setup_micro_lp.`**`setupMicroLP`**`(`*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', '2006-BHF-SM-AV18', '2006-BHF-NM-AV18', '2006-IBHF-SM-AV18', '2006-IBHF-NM-AV18', '2007-BHF-NM-LP-BONNC'.

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

Attributes:

Parameters

- **model** (*str, optional*)
- **between** (*The model to consider. Choose*)

`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: /nucleardatapy_sample/plots/plot_matter_setupMicroLP.py

2.1.5 matter.setupMicroGap

`nucleardatapy.matter.setup_micro_gap.`**`micro_gap_models`**`(`*matter='NM'*`)`

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: '2008-BCS-NM', '2008-QMC-NM-swave', '2009-DLQMC-NM', '2010-QMC-NM-AV4', '2017-MBPT-NM-GAP-EMG-450-500-N2LO', '2017-MBPT-NM-GAP-EMG-450-500-N3LO', '2017-MBPT-NM-GAP-EMG-450-700-N2LO', '2017-MBPT-NM-GAP-EMG-450-700-N3LO', '2017-MBPT-NM-GAP-EM-500-N2LO', '2017-MBPT-NM-GAP-EM-500-N3LO', '2022-AFDMC-NM' :param matter: matter can be 'NM' (by default) or 'SM'. :type matter: str. :return: The list of models. :rtype: list[str].

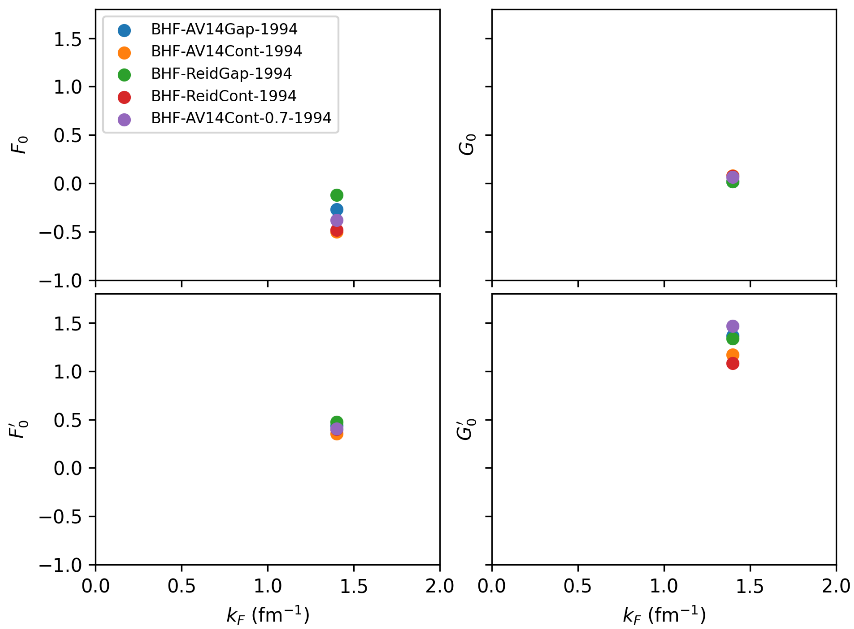

Fig. 10: This figure shows the L=0 Landau parameters in SM for different NN interactions obtained from BHF calculations.

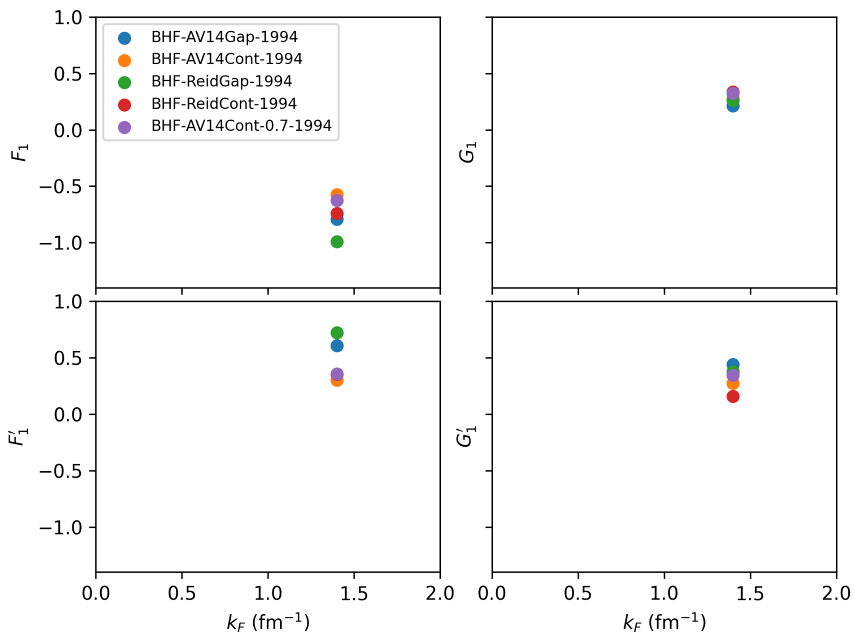

Fig. 11: This figure shows the L=1 Landau parameters in SM for different NN interactions obtained from BHF calculations.

class nucleardatapy.matter.setup_micro_gap.**setupMicroGap**(*model='2008-BCS-NM'*, *matter='NM'*)

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

This choice is defined in *model*, which can chosen among the following choices: '2008-BCS-NM', '2008-QMC-NM-swave', '2009-DLQMC-NM', '2010-QMC-NM-AV4', '2017-MBPT-NM-GAP-EMG-450-500-N2LO', '2017-MBPT-NM-GAP-EMG-450-500-N3LO', '2017-MBPT-NM-GAP-EMG-450-700-N2LO', '2017-MBPT-NM-GAP-EMG-450-700-N3LO', '2017-MBPT-NM-GAP-EM-500-N2LO', '2017-MBPT-NM-GAP-EM-500-N3LO', '2022-AFDMC-NM' :param model: Fix the name of model. Default value: '2008-BCS-NM'. :type model: str, optional.

Attributes:

Parameters

- **model** (*str, optional*)
- **between** (*The model to consider. Choose*)

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: /nucleardatapy_sample/plots/plot_matter_setupMicroGap.py

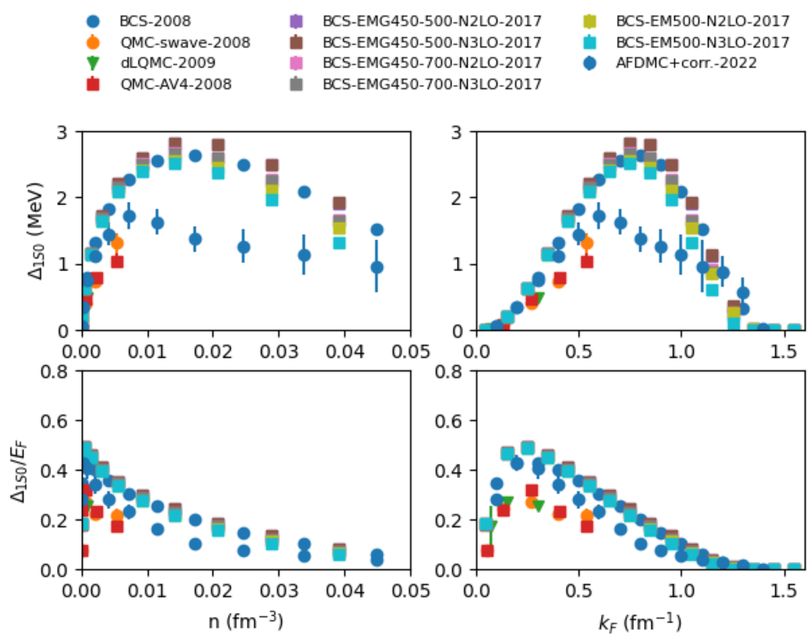

Fig. 12: This figure shows the 1S0 pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

2.1.6 matter.setupMicroEsym

`nucleardatapy.matter.setup_micro_esym.`**`micro_esym_mbs`**`()`

Return a list of many-bodys (mbs) approaches available in this toolkit and print them all on the prompt.

Returns
The list of models with can be 'VAR', 'AFDMC', 'BHF', 'QMC', 'MBPT', 'NLEFT'.

Return type
list[str].

`nucleardatapy.matter.setup_micro_esym.`**`micro_esym_models_mb`**`(`*mb*`)`

Return a list with the name of the models available in this toolkit for a given mb appoach and print them all on the prompt.

Parameters
mb (*str.*) – The mb approach for which there are parametrizations. They should be chosen among the following options: 'VAR', 'AFDMC', 'BHF', 'QMC', 'MBPT', 'NLEFT'.

Returns
The list of parametrizations.

These models are the following ones: If *mb* == 'VAR': '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-APR-fit', If *mb* == 'BHF': '2006-BHF-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', If *mb* == 'MBPT': '2010-MBPT-NM', '2020-MBPT-AM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69' If *mb* == 'NLEFT': '2024-NLEFT-AM',

`nucleardatapy.matter.setup_micro_esym.`**`micro_esym_models_old`**`()`

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-APR-fit', '2006-BHF-AM*', 2016-MBPT-AM', 2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '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].

`class` `nucleardatapy.matter.setup_micro_esym.`**`setupMicroEsym`**`(`*model='1998-VAR-AM-APR', var1=array([0.01, 0.03052632, 0.05105263, 0.07157895, 0.09210526, 0.11263158, 0.13315789, 0.15368421, 0.17421053, 0.19473684, 0.21526316, 0.23578947, 0.25631579, 0.27684211, 0.29736842, 0.31789474, 0.33842105, 0.35894737, 0.37947368, 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-APR-fit', '2006-BHF-AM*', '2016-MBPT-AM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '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:

Parameters

- `model` (*str, optional*)
- `between` (*The model to consider. Choose*)
- `var2` (*var1 and*)
- `np.array([0.1` (*var1 =*)
- `0.15`
- `0.16`
- `0.17`
- `0.2`
- `0.25])`

`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: /nucleardatapy_sample/plots/plot_matter_setupMicroEsym.py

2.1.7 matter.setupPheno

`nucleardatapy.matter.setup_pheno.checkPheno(obj, band, matter)`

Check if the phenomenological EOS is inside the band. Return True if inside the band, otherwise return False.

`nucleardatapy.matter.setup_pheno.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', 'ESkyrme', 'NLRH', 'DDRH', 'DDRHF'.

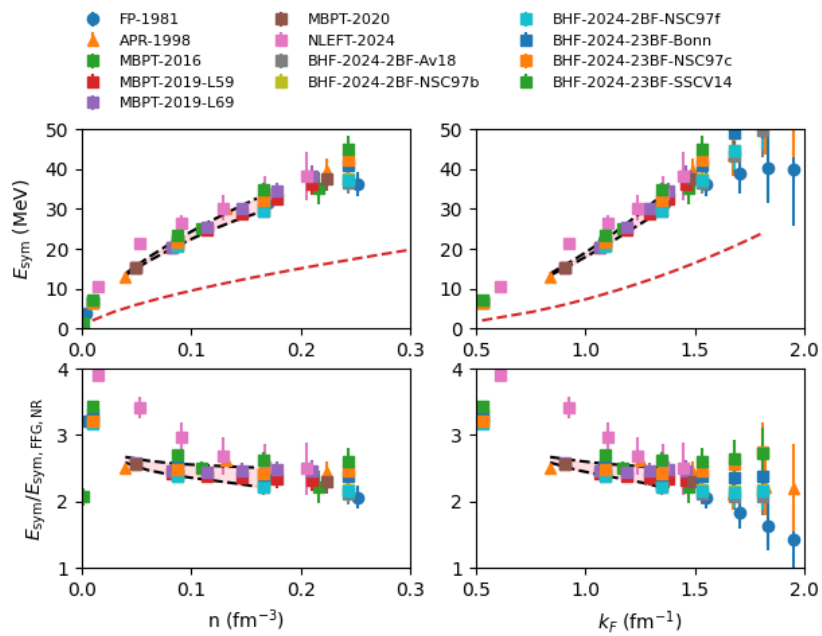

Fig. 13: This figure shows the symmetry energy as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

Return type
list[str].

`nucleardatapy.matter.setup_pheno.`**`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', 'ESkyrme', 'Gogny', 'Fayans', '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* == 'ESkyrme': 'BSk22', 'BSk24', 'BSk25', 'BSk26', 'BSk31', 'BSk32', 'BSkG1', 'BSkG2', 'BSkG3'. If *models* == 'Fayans': 'SLy4', 'SkM*', 'Fy(IVP)', 'Fy(Dr,HDB)', 'Fy(std)', 'SV-min', 'SV-bas', 'SV-K218', 'SV-K226', 'SV-K241', 'SV-mas07', 'SV-mas08', 'SV-mas10',

'SV-sym28', 'SV-sym32', 'SV-sym34', 'SV-kap00', 'SV-kap20', 'SV-kap60'. 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'. :rtype: list[str].

class `nucleardatapy.matter.setup_pheno.`**`setupPheno`**(*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* == 'ESkyrme', *param* can be: 'BSk22', 'BSk24', 'BSk25', 'BSk26', 'BSk31', 'BSk32', 'BSkG1', 'BSkG2', 'BSkG3'.

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

- **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.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_matter_setupPheno.py

2.1.8 matter.setupPhenoEsym

nucleardatapy.matter.setup_pheno_esym.**pheno_esym_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', 'ESkyrme', 'NLRH', 'DDRH', 'DDRHF'.

Return type
list[str].

nucleardatapy.matter.setup_pheno_esym.**pheno_esym_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'.

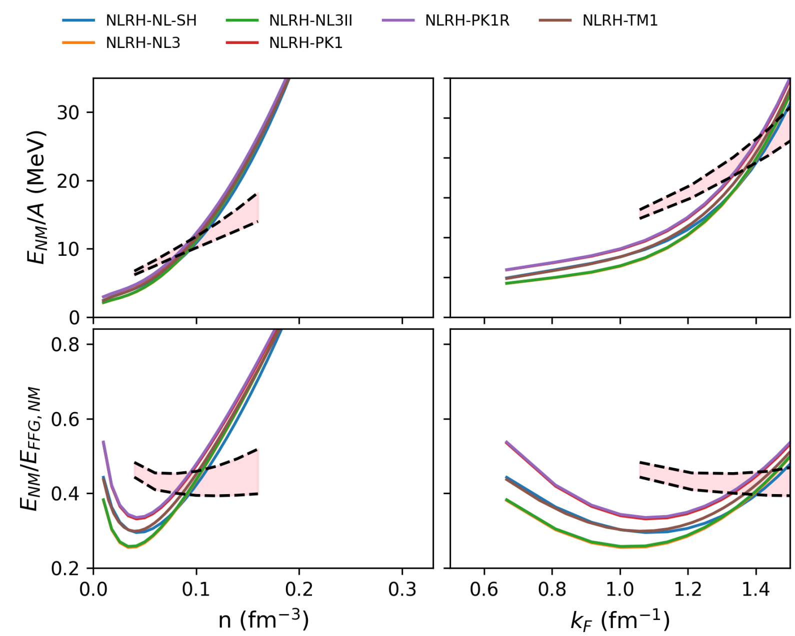

Fig. 14: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree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

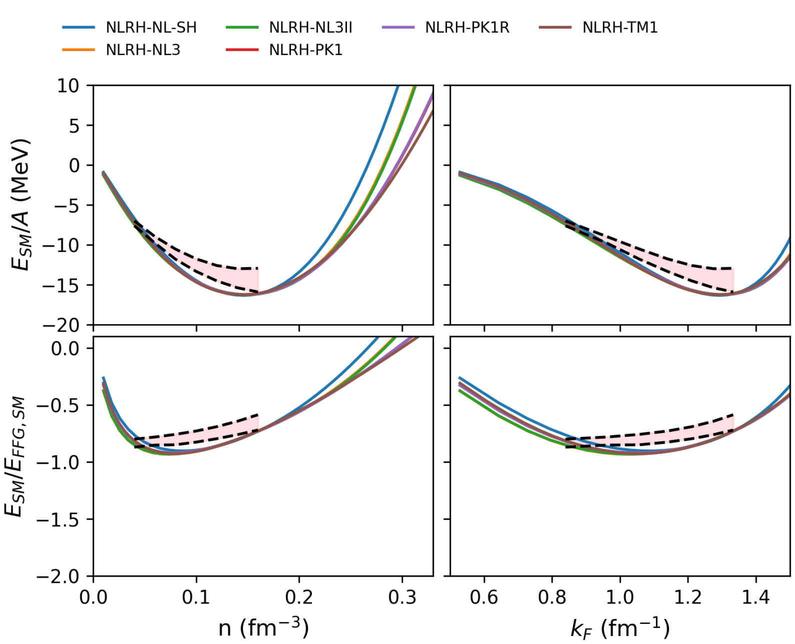

Fig. 15: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic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

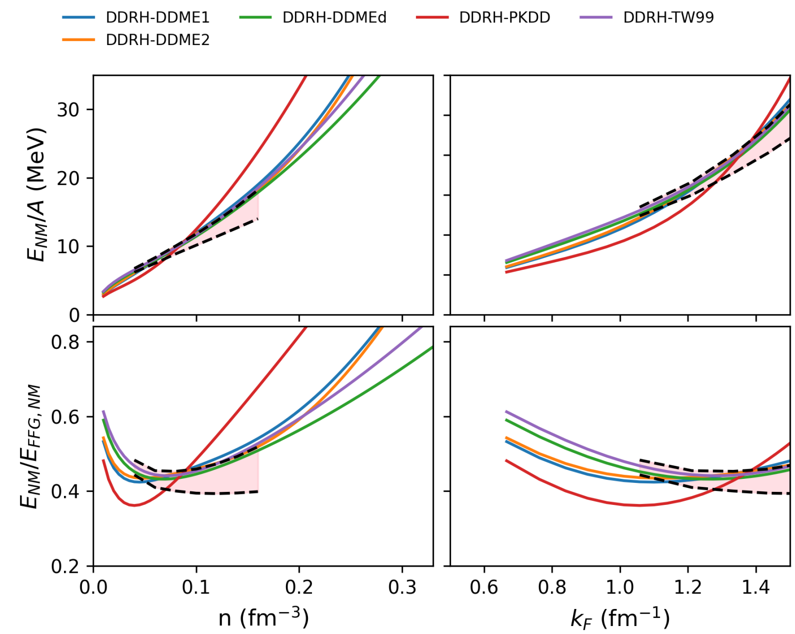

Fig. 16: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

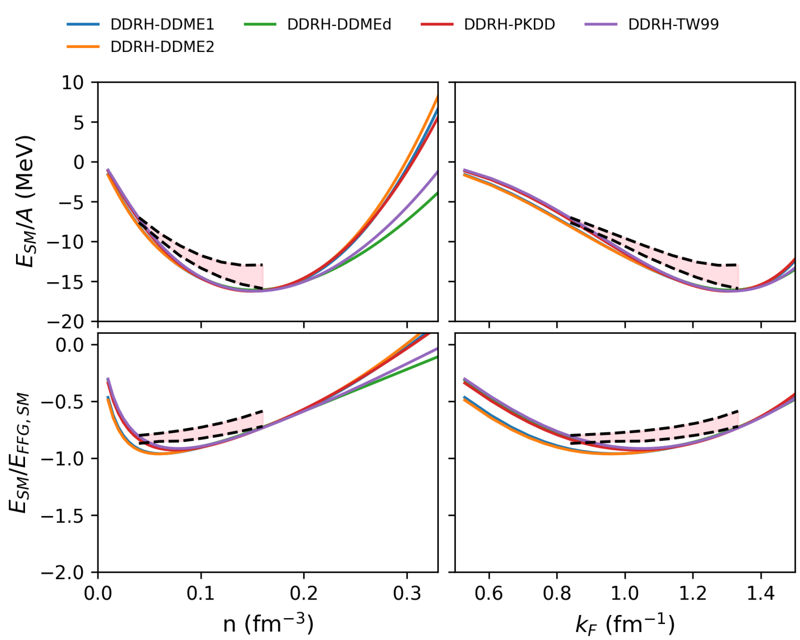

Fig. 17: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

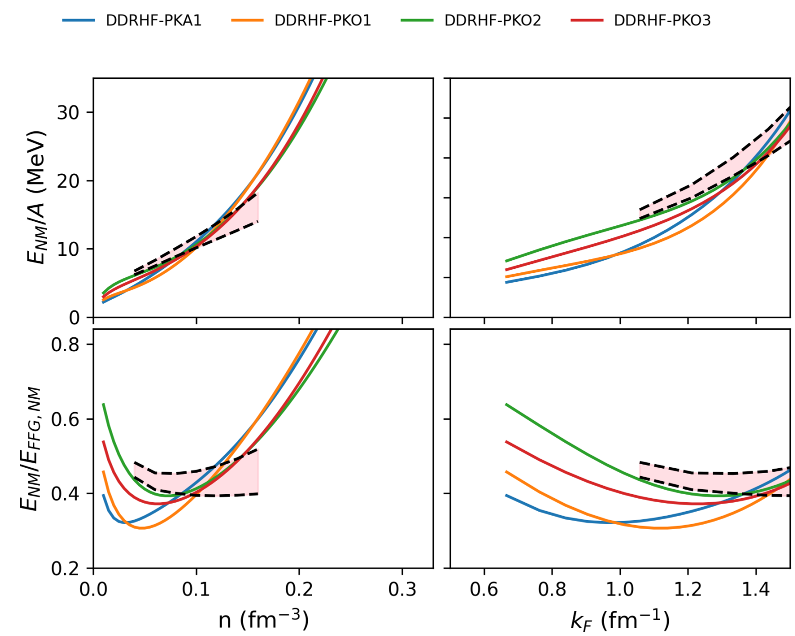

Fig. 18: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

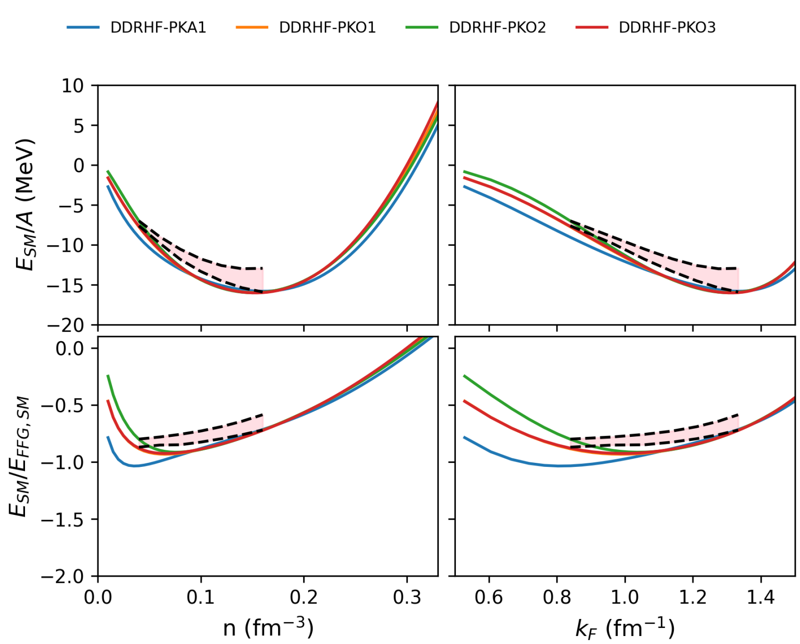

Fig. 19: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock (DDRHF) approach available in the nucleardatapy toolkit.

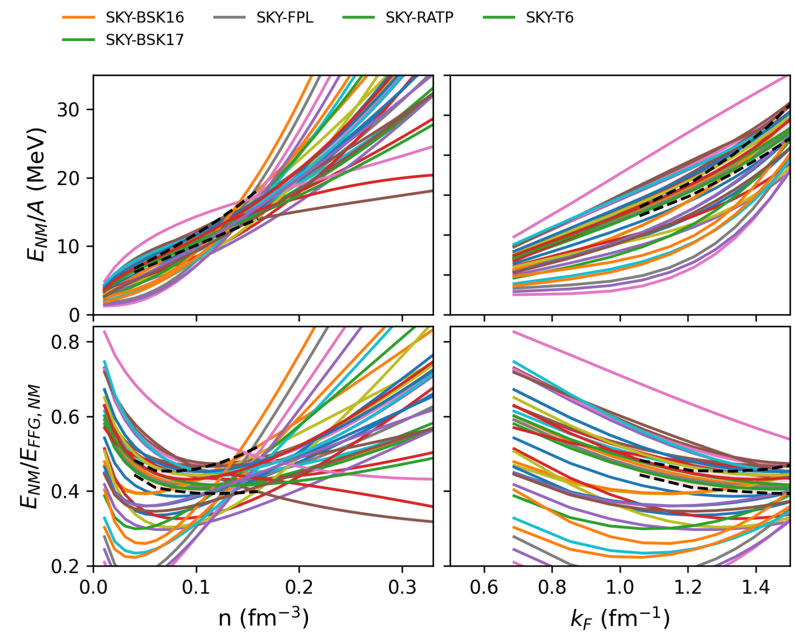

Fig. 20: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interactions available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

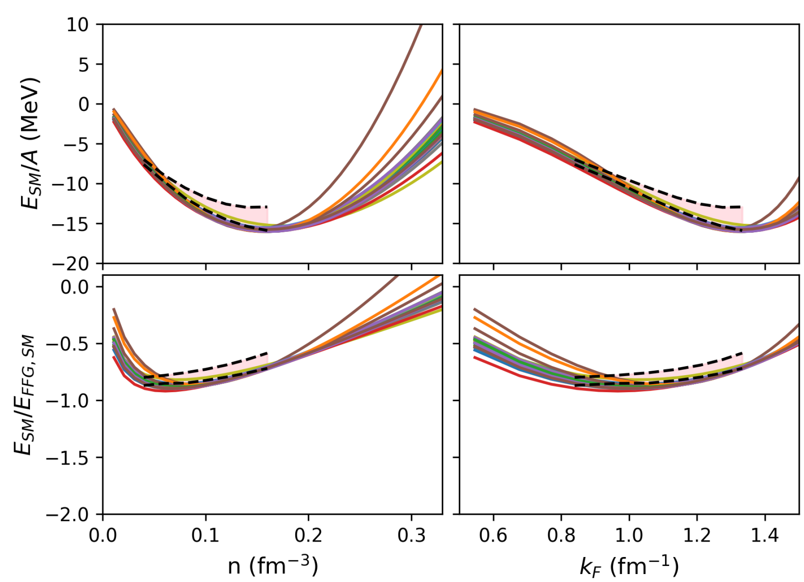

Fig. 21: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interactions available in the nucleardatapy toolkit.

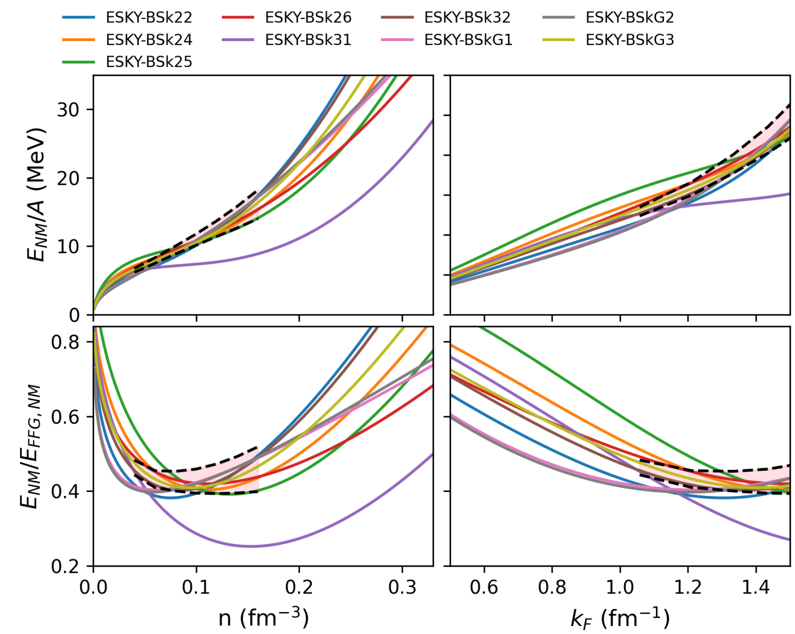

Fig. 22: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the extended ESkyrme interactions available in the nucleardatapy toolkit.

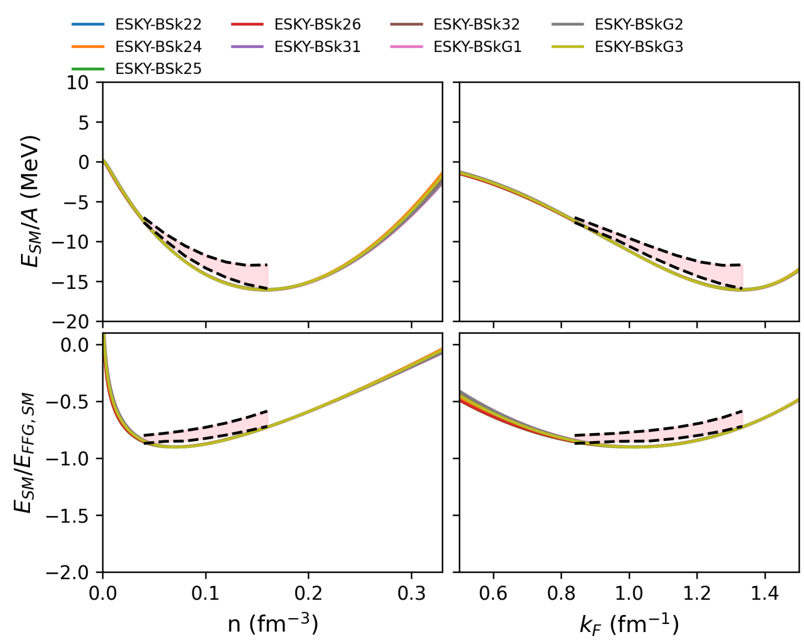

Fig. 23: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the extended ESkyrme interactions available in the nucleardatapy toolkit.

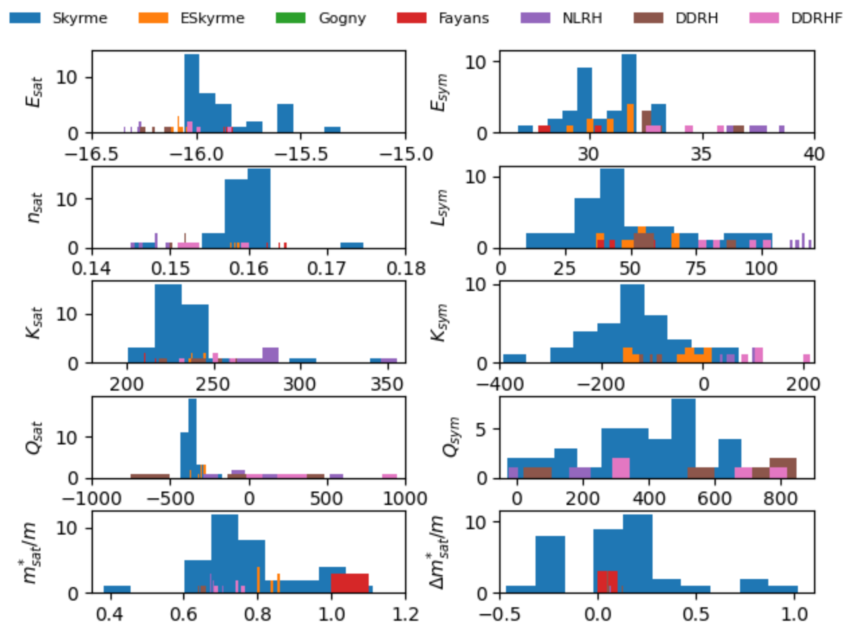

Fig. 24: Distribution of NEP for phenomenological models available in the nucleardatapy toolkit.

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* == 'ESkyrme': 'BSk22', 'BSk24', 'BSk25', 'BSk26', 'BSk31', 'BSk32', 'BSkG1', 'BSkG2', 'BSkG3'. 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].

`class nucleardatapy.matter.setup_pheno_esym.`**`setupPhenoEsym`**(*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* == 'ESkyrme', *param* can be: 'BSk22', 'BSk24', 'BSk25', 'BSk26', 'BSk31', 'BSk32', 'BSkG1', 'BSkG2', 'BSkG3'.

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

- **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:

Parameters

- **model** (*str, optional*)
- **between** (*The model to consider. Choose*)
- **var2** (*var1 and*)
- **np.array([0.1** (*var1 =*)
- **0.15**
- **0.16**
- **0.17**
- **0.2**
- **0.25])**

init_self()

Initialize variables in self.

model

Attribute model.

param

Attribute param.

print_outputs()

Method which print outputs on terminal’s screen.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_matter_setupPhenoEsym.py

2.1.9 matter.setupHIC

nucleardatapy.matter.setup_hic.**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: [‘2002-DLL’, ‘2002-KAON’, ‘2016-FOPI’, ‘2011-FOPI-LAND’, ‘2016-ASY-EOS’ , ‘2021-SPIRIT’,’2019-NP-RATIO’,’2009-ISO-DIFF’].

Returns

The list of constraints.

Return type

list[str].

class nucleardatapy.matter.setup_hic.**setupHIC**(*constraint='2002-DLL'*)

Instantiate the constraints on the EOS from HIC.

This choice is defined in the variable *constraint*.

constraint can chosen among the following ones: [‘2002-DLL’, ‘2016-FOPI’, ‘2002-KAON’,’2009-ISO-DIFF’, ‘2011-FOPI_LAND’ , ‘2016-ASY_EOS’,’2019-NP-RATIO’, ‘2021-SPIRIT’].

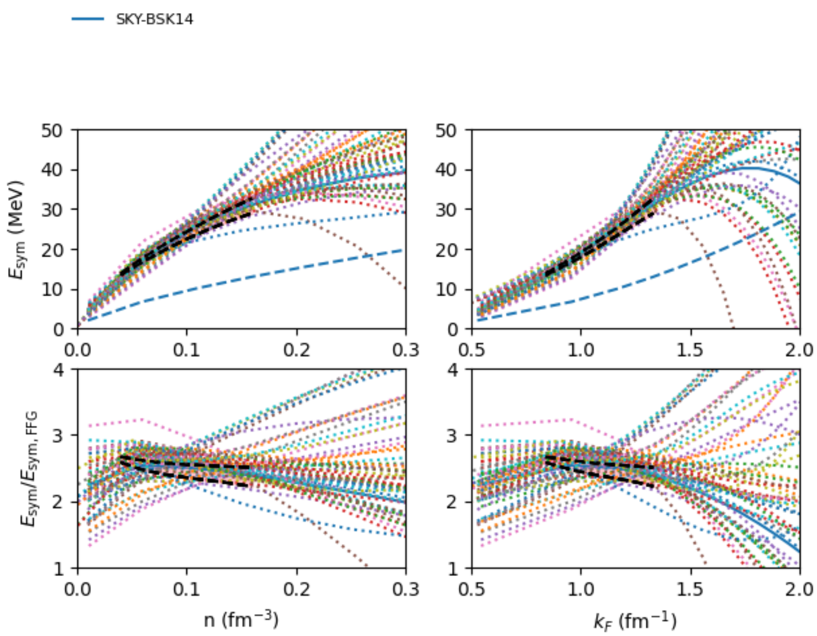

Fig. 25: This figure shows the symmetry energy as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

Parameters
`constraint` (*str, optional.*) – Default value: '2002-DLL'.

Attributes:

`init_self()`
Initialize variables in self.

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

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_matter_setupHIC.py

2.2 EOS

2.2.1 eos.setupAM

class `nucleardatapy.eos.setup_am.setupAM`(*model='1998-VAR-AM-APR'*, *param=None*, *kind='micro'*, *asy=0.0*, *x_mu=0.0*)

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:

Parameters

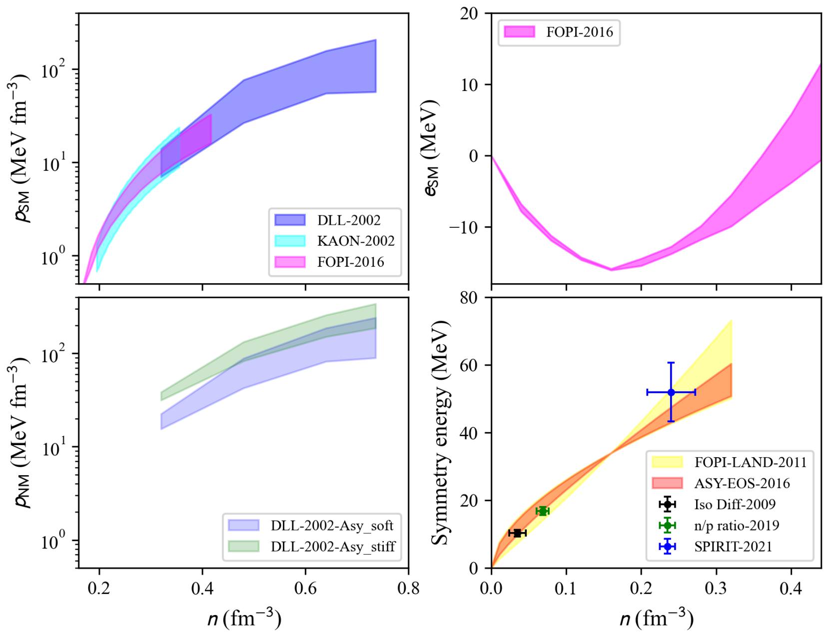

Fig. 26: HIC Experimental constraints for the energy per particle (left) and pressure (right) in SM as a function of the particle density for different analyses available in the *nuda* toolkit.

- **model** (*str, optional*)
- **between** (*The model to consider. Choose*)
- **kind** (*chose between 'micro' or 'pheno'.*)
- **var2** (*var1 and*)
- **np.array([0.1** (*var1 =*)
- **0.15**
- **0.16**
- **0.17**
- **0.2**
- **0.25])**

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: /nucleardatapy_sample/plots/plot_eos_setupAM.py

../version-0.2/nucleardatapy_samples/plots/figs/plot_eos_setupA

Fig. 27: 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.

2.2.2 eos.setupBeta

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_eos_setupBeta.py

../version-0.2/nucleardatapy_samples/plots/figs/plot_eos_setupB

Fig. 28: This figure shows the proton fraction at beta-equilibrium. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

../version-0.2/nucleardatapy_samples/plots/figs/plot_eos_setupB

Fig. 29: This figure shows the electron fraction at beta-equilibrium. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

2.3 Nuc

2.3.1 nuc.setupBEExp

`nucleardatapy.nuc.setup_be_exp.`**`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.nuc.setup_be_exp.`**`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'.

../version-0.2/nucleardatapy_samples/plots/figs/plot_eos_setupB

Fig. 30: This figure shows the muon fraction at beta-equilibrium. (left) Microscopic models and (right) phenomenological models available in the nucleardatapy toolkit.

Return type
list[str].

`class nucleardatapy.nuc.setup_be_exp.setupBEExp(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:

`D3p_n(Zmin=1, Zmax=95)`

Compute the three-points odd-even mass staggering (D3p_n) D_3p^N = (-)**N * (2*E(Z,N)-E(Z,N+1)-E(Z,N-1)) / 2

`D3p_p(Nmin=1, Nmax=95)`

Compute the three-points odd-even mass staggering (D3p_p) D_3p^P = (-)**Z * (2*E(Z,N)-E(Z+1,N)-E(Z-1,N)) / 2

`S2n(Zmin=1, Zmax=95)`

Compute the two-neutron separation energy (S2n) S2n = E(Z,N)-E(Z,N-2)

`S2p(Nmin=1, Nmax=95)`

Compute the two-proton separation energy (S2n) S2p = E(Z,N)-E(Z-2,N)

`Zmax`

maximum charge of nuclei present in the table.

Type
Attribute Zmax

`dist_nbNuc`

attribute number of nuclei discovered per year

`dist_year`

attribute distribution of years

`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.

`isotopes`(*Zmin=1*, *Zmax=95*)

Method which find the first and last isotopes for each Zmin<Z<Zmax.

Parameters

- **Zmin** (*int, optional. Default: 1.*) – Fix the minimum charge for the search of isotopes.
- **Zmax** (*int, optional. Default: 95.*) – Fix the maximum charge for the search of isotopes.

Attributes:

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

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:

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_nuc_setupBEExp.py

../version-0.2/nucleardatapy_samples/plots/figs/plot_nuc_setupB

Fig. 31: The nuclear chart based on AME 2020 table. The different colors correspond to the different measured half-times of nuclei.

../version-0.2/nucleardatapy_samples/plots/figs/plot_nuc_setupB

Fig. 32: Histogram showing the distribution of nuclei per discovery year, since the first one discovered in 1851.

2.3.2 nuc.setupBETheo

`nucleardatapy.nuc.setup_be_theo.`**`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-WS*', '2010-HFB21', '2011-WS3', '2013-HFB22', '2013-HFB23', '2013-HFB24', '2013-HFB25', '2013-HFB26', '2021-BSkG1', '2022-BSkG2', '2023-BSkG3', '2025-BSkG4']

Returns
The list of tables.

Return type
list[str].

`nucleardatapy.nuc.setup_be_theo.`**`conversionMBE`**`(`*M*, *N*, *Z*`)`

Convert the mass excess of a nucleus to its binding energy.

`class nucleardatapy.nuc.setup_be_theo.`**`setupBETheo`**`(`*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-WS*', '2010-HFB21','2011-WS3', '2013-HFB26', '2021-BSkG1', '2022-BSkG2', '2023-BSkG3', '2025-BSkG4']

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

Attributes:

`D3p_n``(`*Zmin=1*, *Zmax=95*`)`

Compute the three-points odd-even mass staggering (D3p_n) D3p^N = (-)**N * (2*E(Z,N)-E(Z,N+1)-E(Z,N-1)) / 2

`D3p_p``(`*Nmin=1*, *Nmax=95*`)`

Compute the three-points odd-even mass staggering (D3p_p) D3p^Z = (-)**Z * (2*E(Z,N)-E(Z+1,N)-E(Z-1,N)) / 2

`S2n``(`*Zmin=1*, *Zmax=95*`)`

Compute the two-neutron separation energy (S2n) S2n = E(Z,N)-E(Z,N+2)

`S2p``(`*Nmin=1*, *Nmax=95*`)`

Compute the two-proton separation energy (S2n) S2p = E(Z,N)-E(Z-2,N)

`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:

`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_S2n`(*Zmin=1*, *Zmax=95*)

Method which find the drip-line nuclei from S2n (neutron side).

Parameters

- **Zmin** (*int, optional. Default: 1.*) – Fix the minimum charge for the search of the neutron drip line.
- **Zmax** (*int, optional. Default: 95.*) – Fix the maximum charge for the search of the neutron drip line.

Attributes:

`drip_S2p`(*Nmin=1*, *Nmax=95*)

Method which find the drip-line nuclei from S2p (proton side).

Parameters

- **Nmin** (*int, optional. Default: 1.*) – Fix the minimum neutron number for the search of the proton drip line.
- **Nmax** (*int, optional. Default: 95.*) – Fix the maximum neutron number for the search of the proton drip line.

Attributes:

`init_self()`

Initialize variables in self.

`print_outputs()`

Method which print outputs on terminal's screen.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_nuc_setupBETheo.py

2.3.3 nuc.setupRchExp

`nucleardatapy.nuc.setup_rch_exp.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].

`class nucleardatapy.nuc.setup_rch_exp.setupRchExp`(*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'.

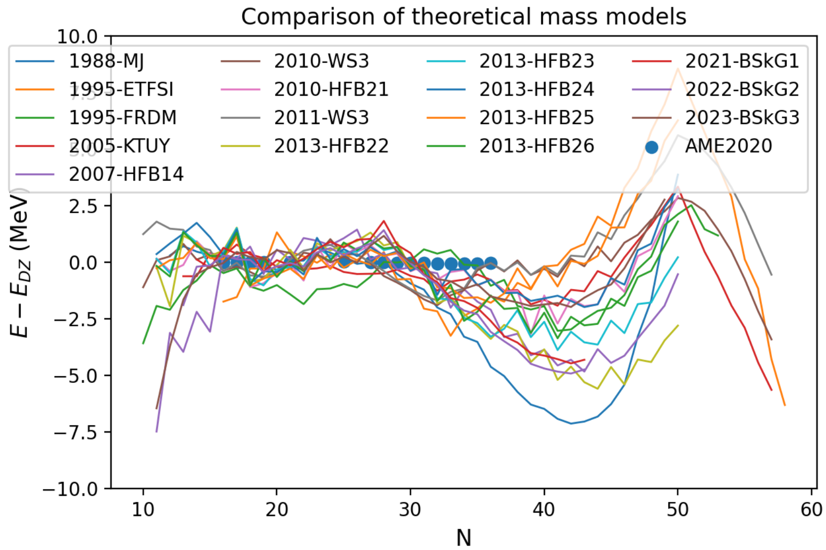

Fig. 33: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 20.

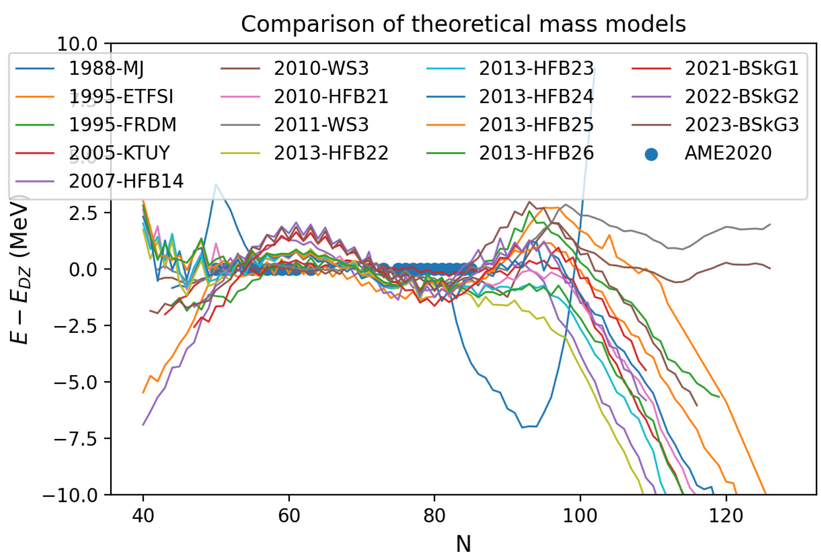

Fig. 34: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 50.

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

Attributes:

Parameters

- **model** (*str, optional*)
- **between** (*The model to consider. Choose*)

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.

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.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_nuc_setupRchExp.py

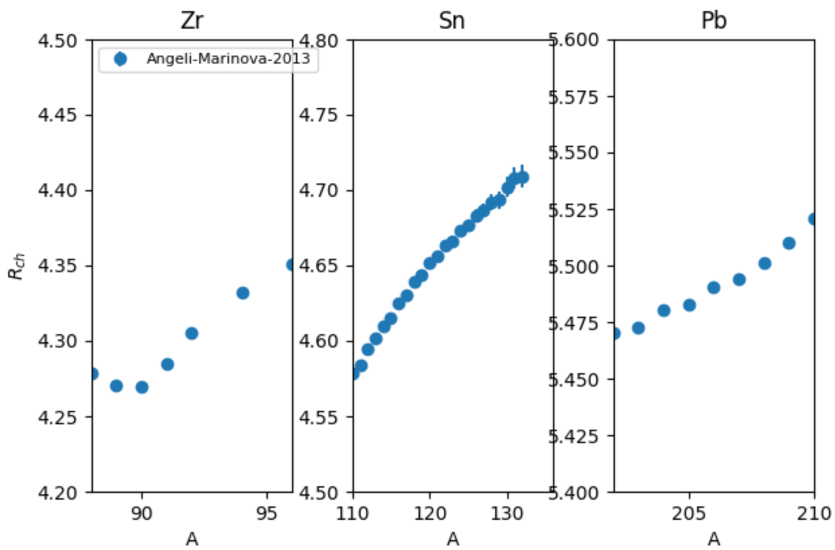

Fig. 35: Charge radii for Zn, Sn, and Pb isotopes and for the models available in the nuda toolkit.

2.3.4 nuc.setupRchTheo

`nucleardatapy.nuc.setup_rch_theo.`**`rch_theo_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].

`class nucleardatapy.nuc.setup_rch_theo.`**`setupRchTheo`**`(`*table='2021-BSkG1'*`)`

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:

Parameters

- **`table`** (*`str, optional`*)
- **`between`** (*`The theoretical table to consider. Choose`*)

`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.

nucA
: Attribute A (mass of the nucleus).

nucN
: Attribute N (number of neutrons of the nucleus).

nucRch
: Attribue 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.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_nuc_setupRchTheo.py

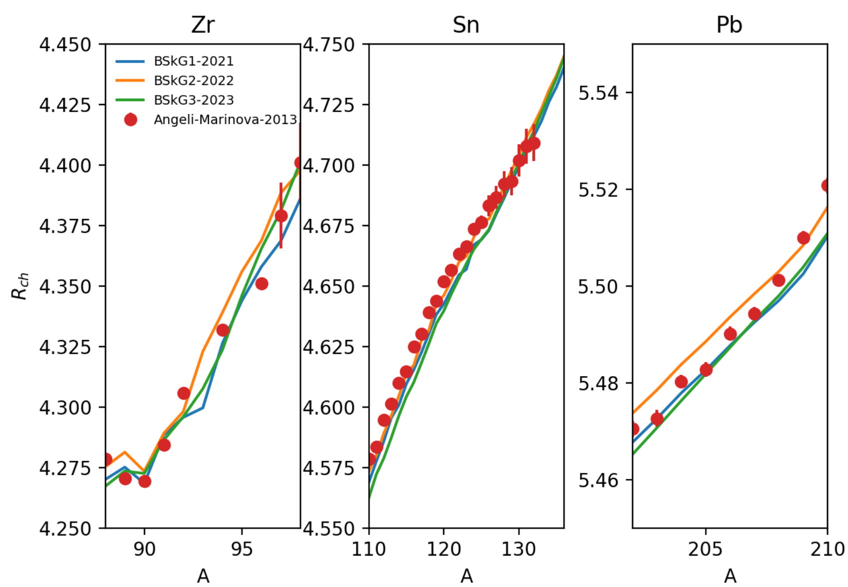

Fig. 36: Charge radii for Zn, Sn, and Pb isotopes and for the theoretical models available in the nuda toolkit.

2.3.5 nuc.setupISGMRExp

nucleardatapy.nuc.setup_isgmr_exp.**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].

class nucleardatapy.nuc.setup_isgmr_exp.**setupISGMRExp**(*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:

Parameters

- **table** (*str, optional*)
- **between** (*The table to consider. Choose*)

E_unit

Attribute energy unit.

average()

Method to average the data when same target is given.

Attributes:

label

Attribute providing the label the data is references for figures.

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.

select(*Zref=50, obs='M12Mm1'*)

Method to select a subset of data.

Parameters

- **Zref** (*int, optional. Default: 1.*) – Fix the reference charge for the search of isotopes.
- **obs** (*str*) – kind of observable to extract: 'M12M0', 'M12Mm1', 'M32M1'.

Attributes:

table

Attribute table.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_nuc_setupISGMRExp.py

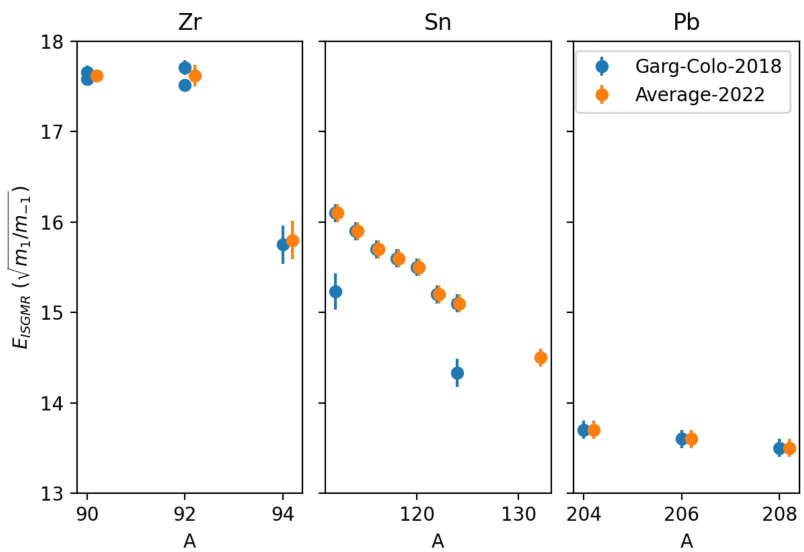

Fig. 37: Experimental ISGMR energies available in the nucleardatapy toolkit.

2.4 Crust

2.4.1 crust.setupCrust

`nucleardatapy.crust.setup_crust.`**`crust_models`**`()`

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].

class `nucleardatapy.crust.setup_crust.`**`setupCrust`**`(`*model='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:

init_self()

Initialize variables in self.

print_outputs()

Method which print outputs on terminal's screen.

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_crust_setupCrust.py

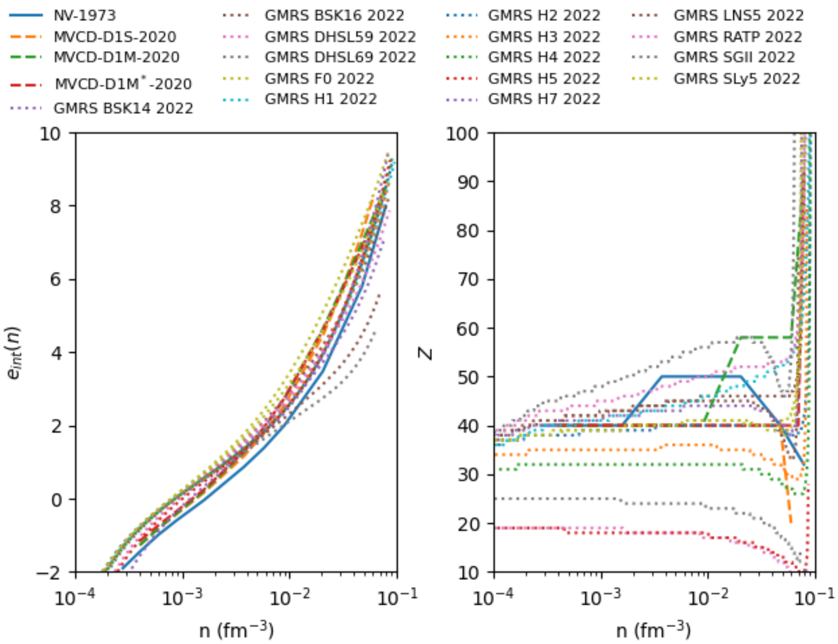

Fig. 38: Properties of the crust as given by the models available in the nuda toolkit.

2.5 Astro

2.5.1 astro.setupGW

`nucleardatapy.astro.setup_gw.`**gw_hyps**(*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].

`nucleardatapy.astro.setup_gw.`**gw_sources**()

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

Returns

The list of sources.

Return type
list[str].

`class nucleardatapy.astro.setup_gw.setupGW(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:

init_self()
Initialize variables in self.

label
Attribute providing the label the data is references for figures.

note
Attribute providing additional notes about the data.

print_latex()
Method which print outputs in table format (latex) on terminal's screen.

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

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

`class nucleardatapy.astro.setup_gw.setupGWAverage(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:

init_self()
Initialize variables in self.

print_latex()
Method which print outputs in table format (latex) on terminal's screen.

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

Here is a figure which is produced with the Python sample: /nucleardatapy_sample/plots/plot_astro_setupGW.py

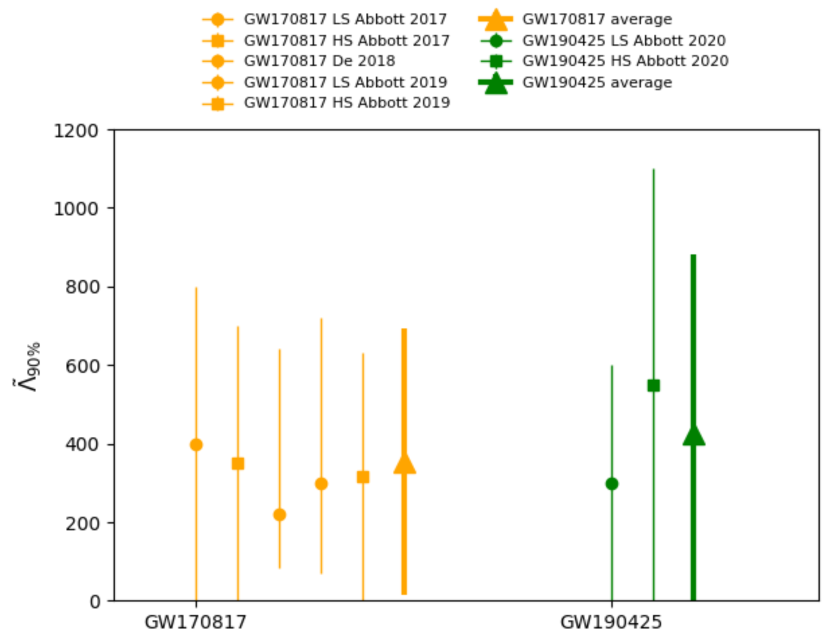

Fig. 39: Estimation of the effective tidal deformability from difference sources (different colors) and different hypotheses on the pulsar spin and the waveform employed for match filtering.

2.5.2 astro.setupMasses

`nucleardatapy.astro.setup_masses.`**`masses_obss`**`(`*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].

`nucleardatapy.astro.setup_masses.`**`masses_sources`**`()`

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

Returns
The list of sources.

Return type
list[str].

class `nucleardatapy.astro.setup_masses.`**`setupMasses`**`(`*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.

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_latex()

Method which print outputs in table format (latex) on terminal's screen.

print_output()

Method which print outputs on terminal's screen.

ref

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

sig_lo

Attribute the negative uncertainty.

sig_up

Attribute the positive uncertainty.

class nucleardatapy.astro.setup_masses.**setupMassesAverage**(*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_latex()

Method which print outputs in table format (latex) on terminal's screen.

print_output()

Method which print outputs on terminal's screen.

Here is a figure which is produced with the Python sample: /nucleardatapy_sample/plots/plot_astro_setupMasses.py

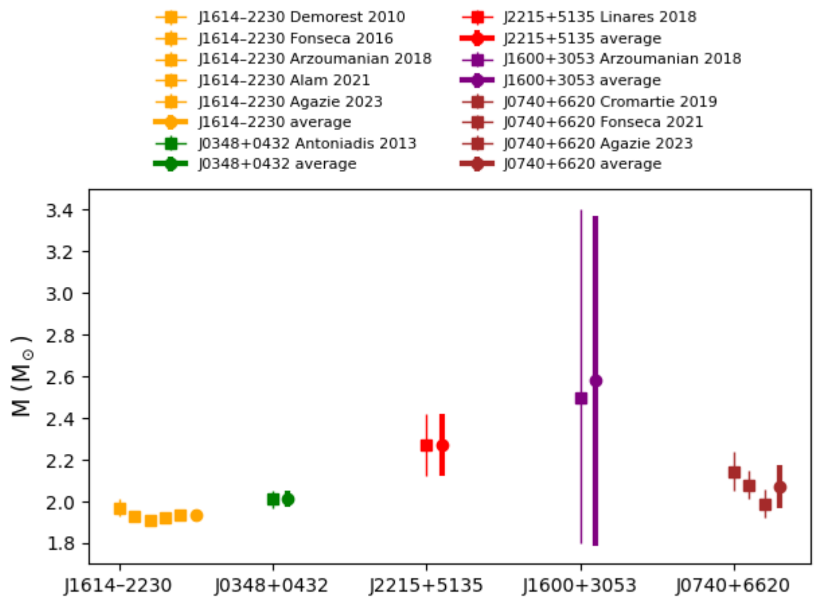

Fig. 40: The masses measured for massive neutron stars is radio-astronomy. The different colors correspond to the different sources.

2.5.3 astro.setupMR

`nucleardatapy.astro.setup_mr.`**`mr_obss`**(*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].

`nucleardatapy.astro.setup_mr.`**`mr_sources`**()

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

Returns
The list of sources.

Return type
list[str].

`class` `nucleardatapy.astro.setup_mr.`**`setupMR`**(*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_lo
Attribute the negative uncertainty.

mass_sig_up
Attribute the positive uncertainty.

note
Attribute providing additional notes about the observation.

print_latex()
Method which print outputs in table format (latex) on terminal's screen.

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

rad
Attribute the observational mass of the source.

rad_sig_lo
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.astro.setup_mr.**setupMRAverage**(*source='J1614–2230'*, *obss=[1, 2]*)
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_latex()
Method which print outputs in table format (latex) on terminal's screen.

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

Here is a figure which is produced with the Python sample: /nucleardatapy_sample/plots/plot_astro_setupMR.py

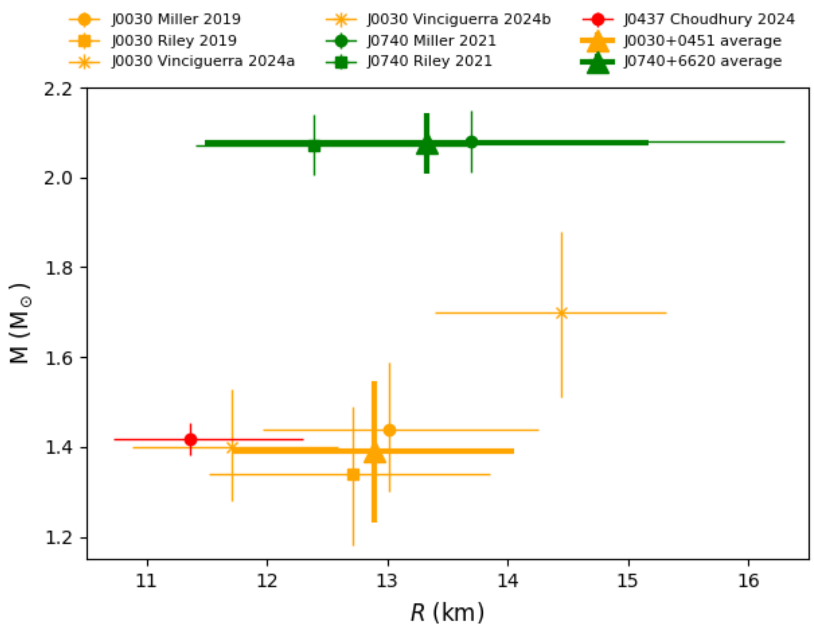

Fig. 41: The masses and radii measured by NICER. The different colors correspond to the different sources.

2.5.4 astro.setupMup

nucleardatapy.astro.setup_mup.**mup_hyps**(*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].

nucleardatapy.astro.setup_mup.**mup_sources**()

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

Returns

The list of sources.

Return type

list[str].

class nucleardatapy.astro.setup_mup.**setupMup**(*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_latex()`
Method which print outputs in table format (latex) on terminal's screen.

`print_output()`
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.astro.setup_mup.**setupMupAverage**(*source='GW170817'*, *hyps=[1]*)

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_latex()`
Method which print outputs in table format (latex) on terminal's screen.

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

Here is a figure which is produced with the Python sample: /nucleardatapy_sample/plots/plot_astro_setupMup.py

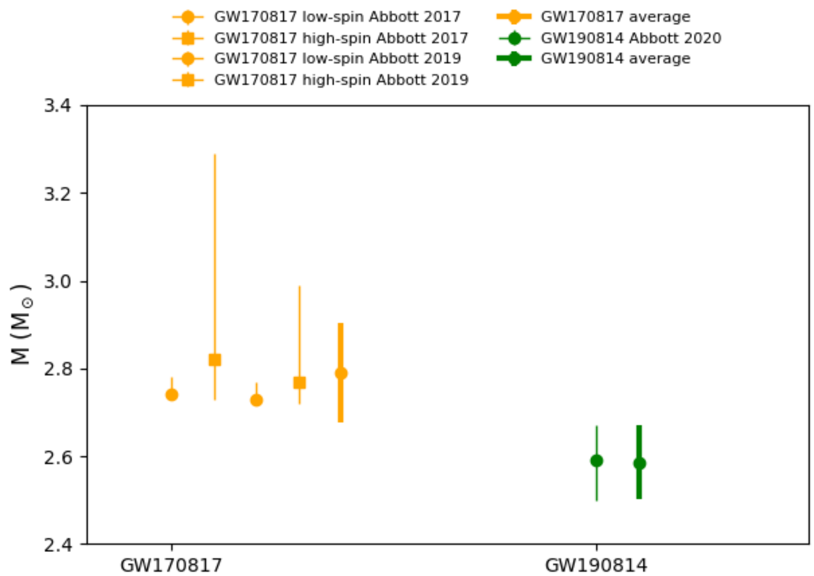

Fig. 42: The upper masses measured from GW observations. The different colors correspond to the different sources.

2.5.5 astro.setupMtov

class `nucleardatapy.astro.setup_mtov.`**`setupMtov`**(*sources_lo=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_latex()`

Method which print outputs in table format (latex) on terminal's screen.

`print_output()`

Method which print outputs on terminal's screen.

Here is a figure which is produced with the Python sample: /nucleardatapy_sample/plots/plot_astro_setupMtov.py

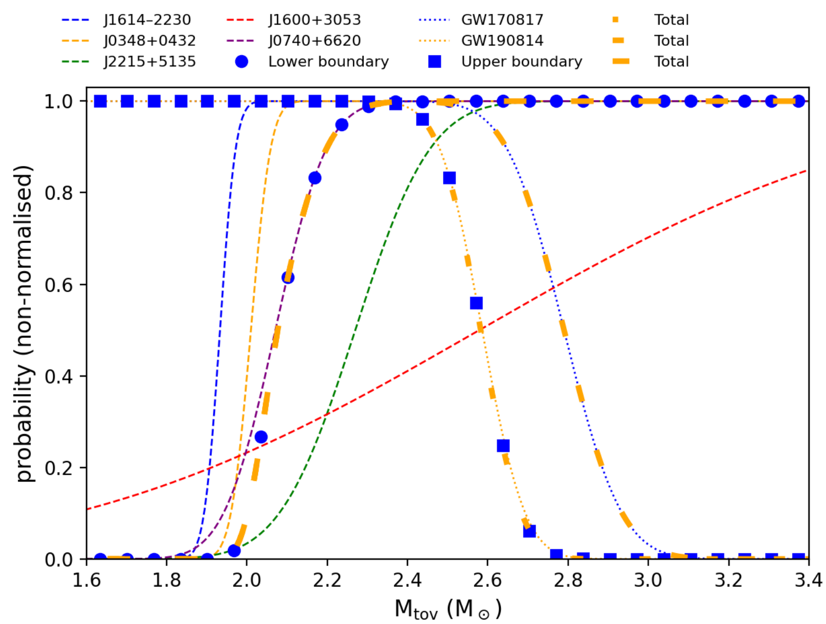

Fig. 43: The probability distribution function for the TOV mass constructed from radio and gravitational-wave observations. The different colors correspond to the different sources.

2.6 Corr

2.6.1 corr.setupEsym

Here are a set of figures which are produced with the Python sample: /nucleardatapy_sample/plots/plot_corr_setupEsym.py

2.6.2 corr.setupEsymLsym

`nucleardatapy.corr.setup_EsymLsym.`**`EsymLsym_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].

`class` `nucleardatapy.corr.setup_EsymLsym.`**`setupEsymLsym`**`(`*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'.

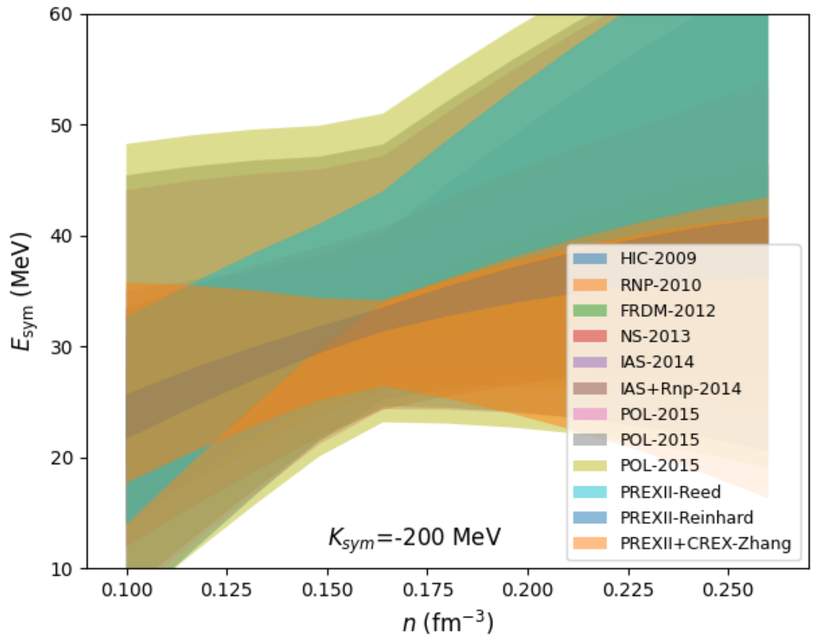

Fig. 44: Uncertainty band for Esym as a function of the density for Ksym=-200 MeV.

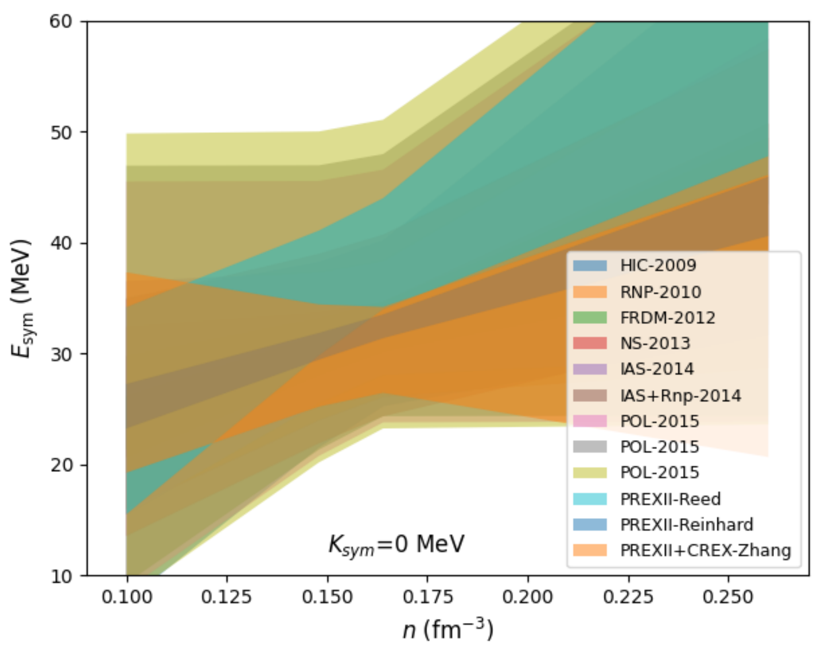

Fig. 45: Uncertainty band for Esym as a function of the density for Ksym=0 MeV.

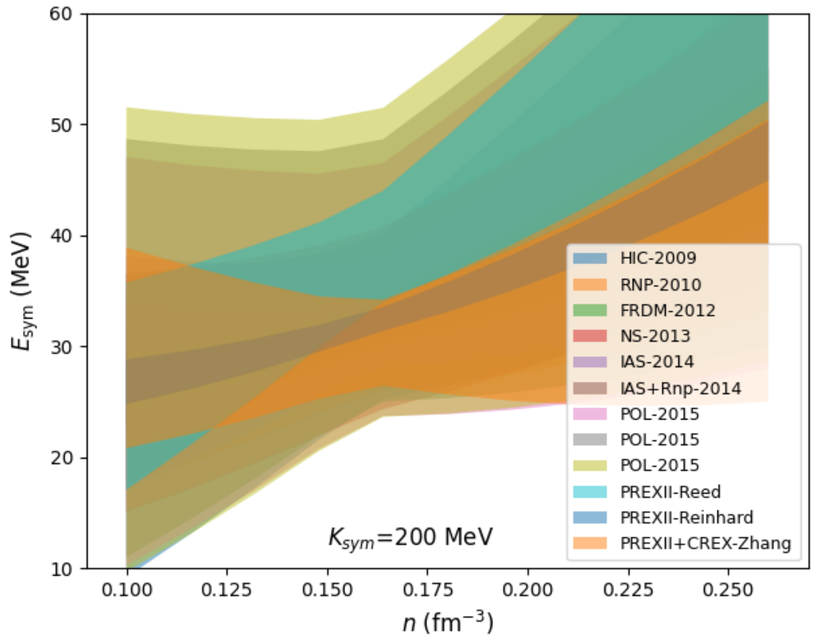

Fig. 46: Uncertainty band for Esym as a function of the density for Ksym=200 MeV.

Parameters

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

Attributes:

constraint

Attribute constraint.

init_self()

Initialize variables in self.

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: /nucleardatapy_sample/plots/plot_corr_setupEsymLsym.py

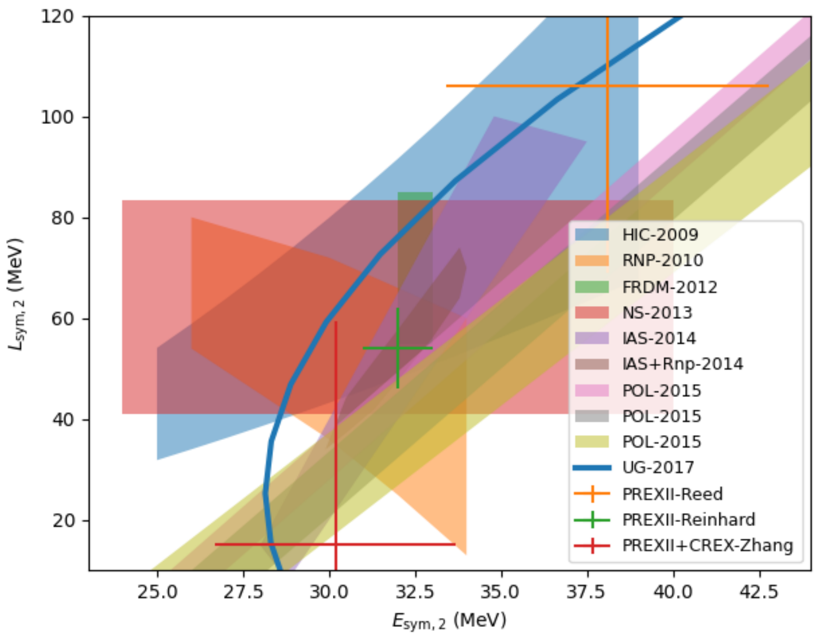

Fig. 47: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nucleardatapy toolkit.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n