nucleardatapy

Release 0.1

Jérôme Margueron, IRL NPA, USA

Jul 13, 2024

CONTENTS

1 Contents **3**

1.1 Usage 3

1.2 API 4

1.3 Miscelaneous 4

2 Complement **7**

2.1 SetupMicro 7

2.2 SetupMicroBand 8

2.3 SetupPheno 11

2.4 SetupEOSHIC 19

2.5 SetupCrust 20

2.6 SetupMassesExp 22

2.7 SetupMassesTheory 26

2.8 SetupRadCh 27

2.9 SetupISGMR 29

2.10 SetupEsymLsym 31

3 Indices and tables **33**

Python Module Index **35**

Index **37**

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_SetupMicro.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.SetupMicro( model = '1998-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 SetupMicro

`class nucleardatapy.setup_micro.SetupMicro(model='1998-VAR-AM-APR')`

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', '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', '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_micro.models_micro()`

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', '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', '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].

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

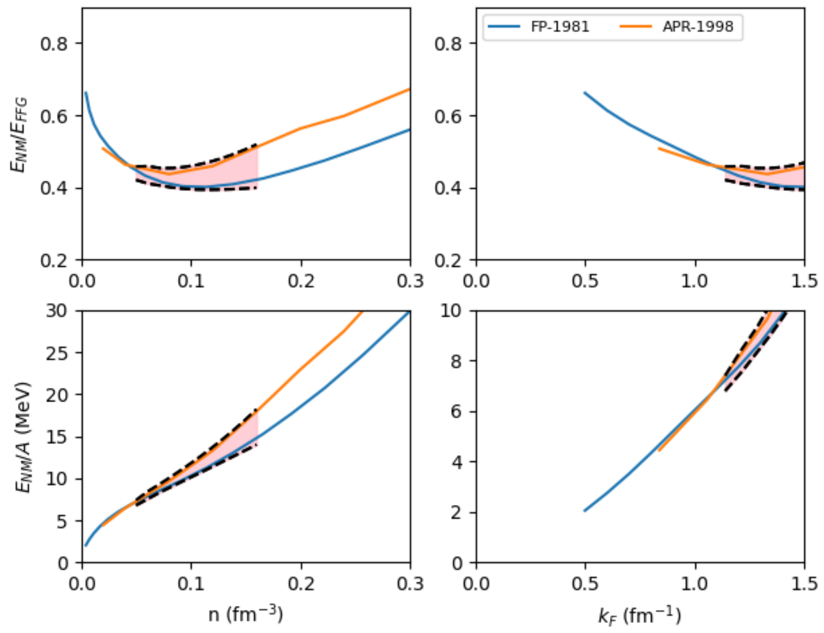

Fig. 1: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the variational models available in the nucleardatapy toolkit.

2.2 SetupMicroBand

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

Attributes:

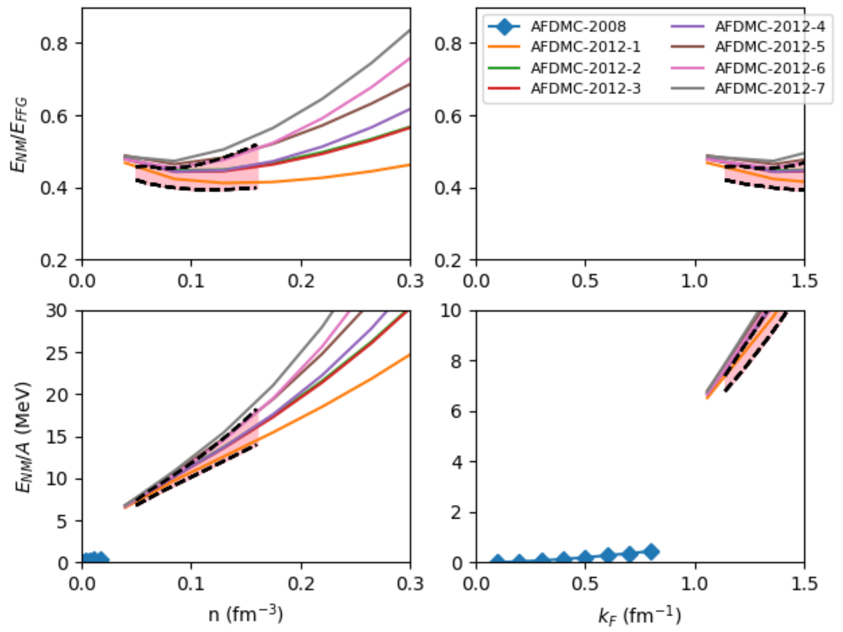

Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

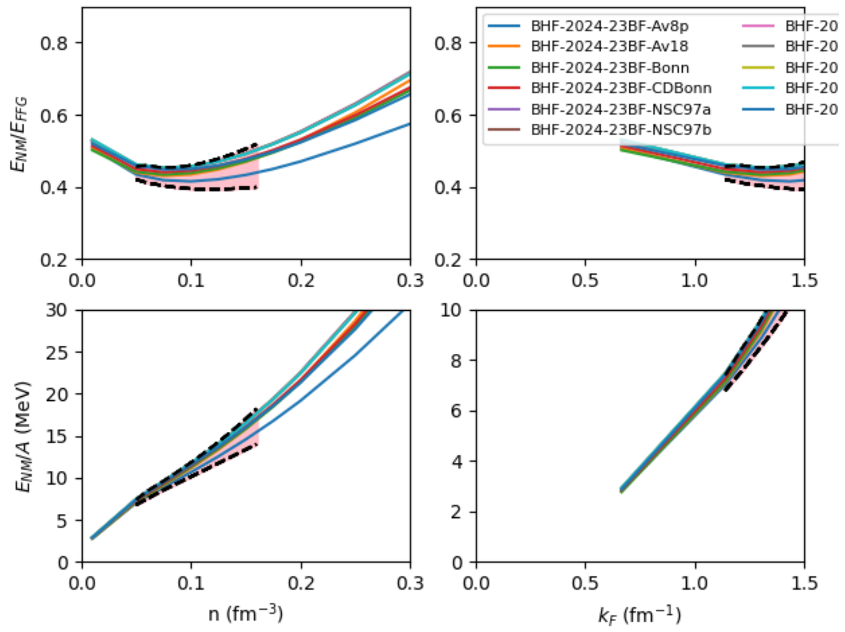

Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

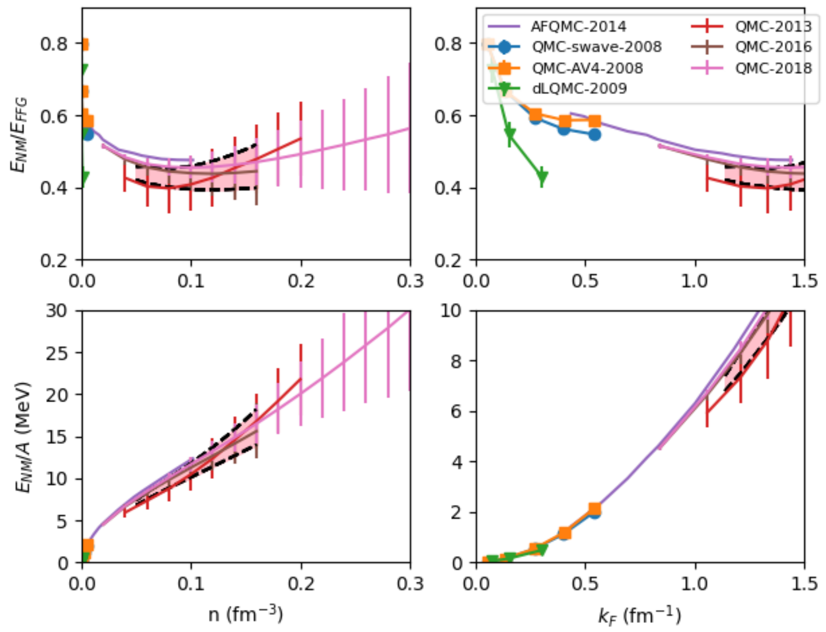

Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

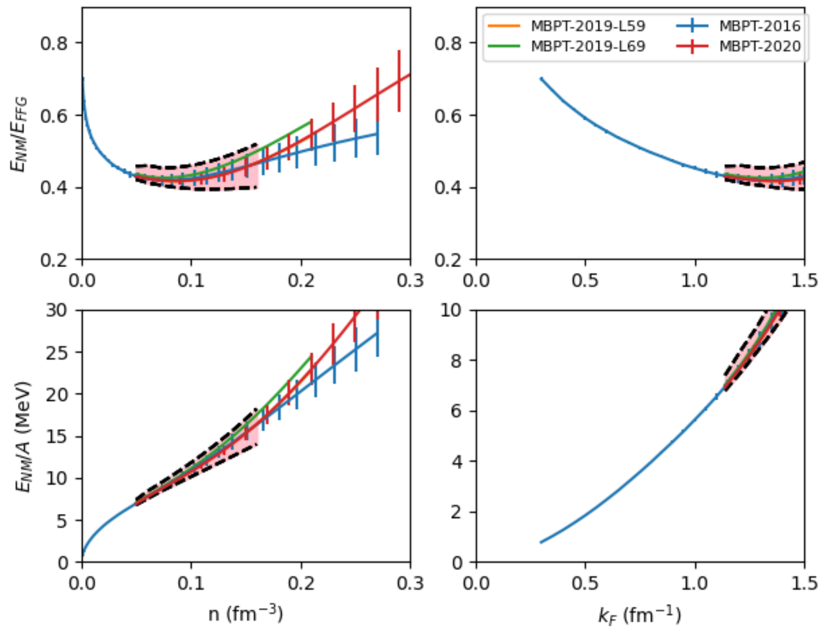

Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.

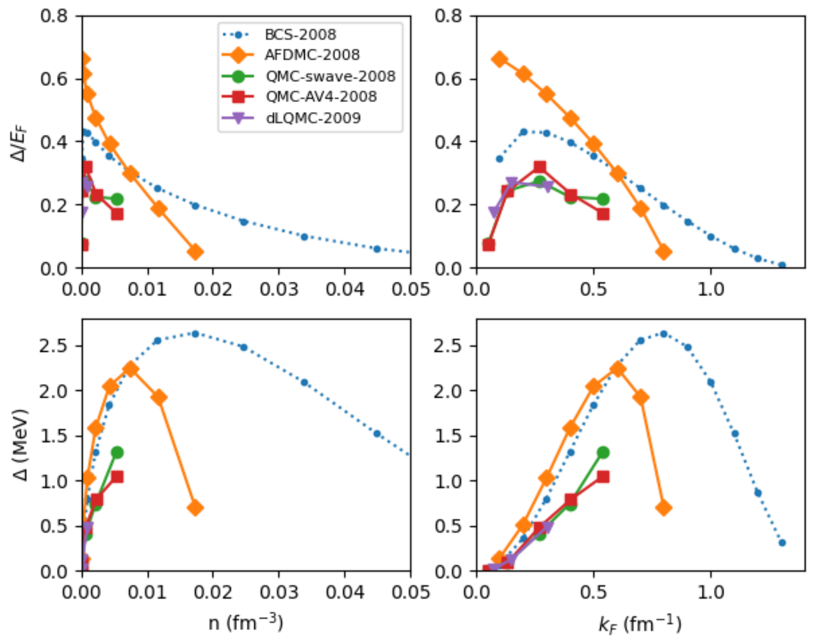

Fig. 6: This figure shows the pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

`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.

2.3 SetupPheno

class `nucleardatapy.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'.

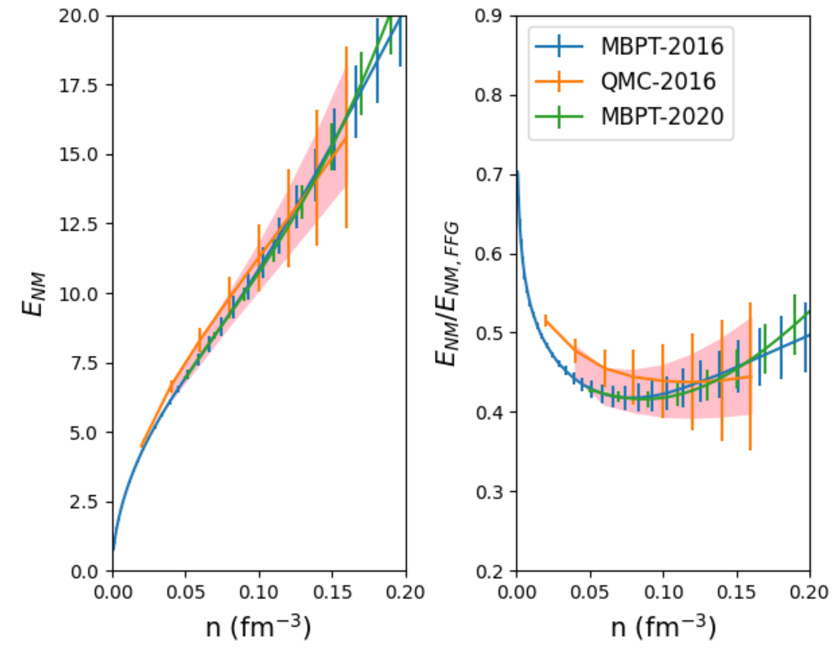

Fig. 7: Uncertainty band in NM obtained from the analysis of different predictions: MBPT-2016, QMC-2016 and MBPT-2020.

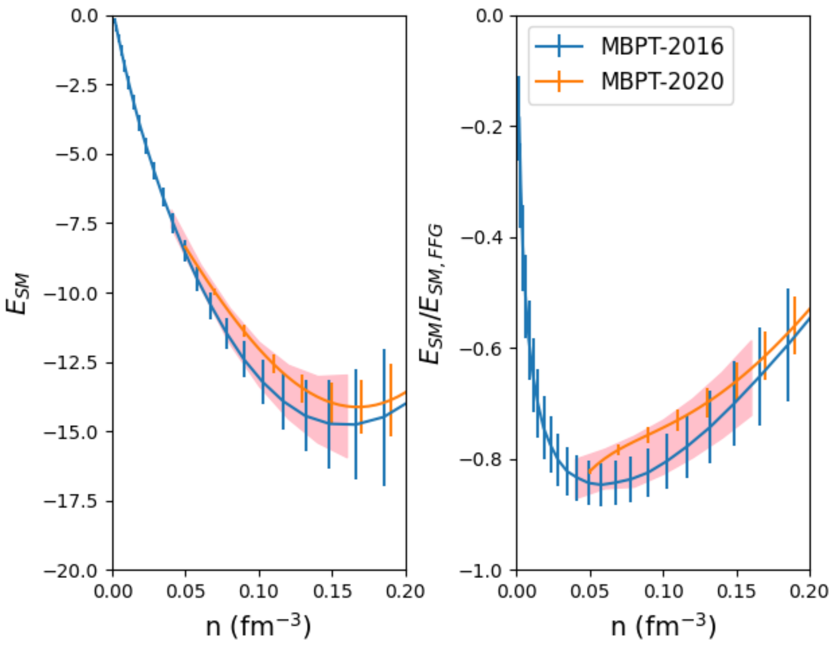

Fig. 8: Uncertainty band in SM obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

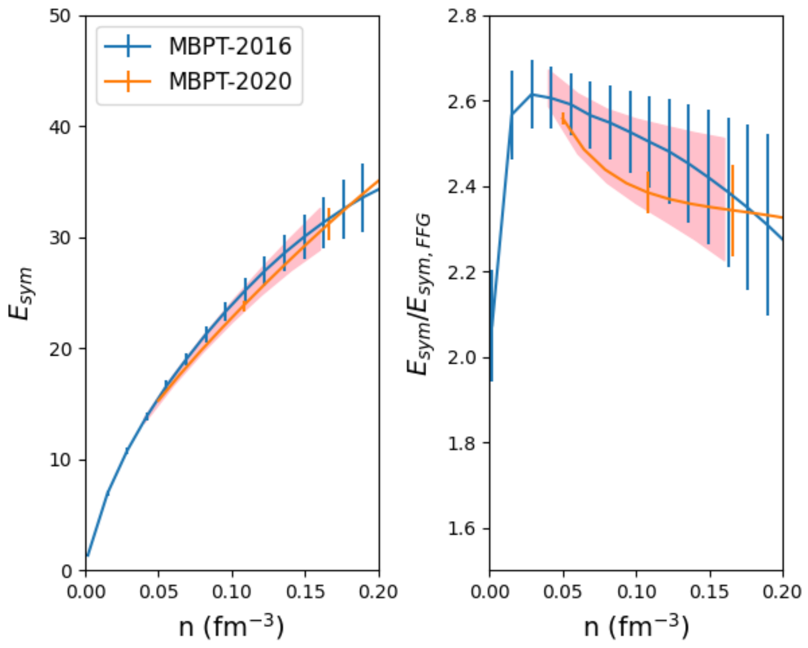

Fig. 9: Uncertainty band for the symmetry energy obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

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:

esym_den

Attribute the density for the symmetry energy.

esym_e2a

Attribute the symmetry energy.

esym_kf

Attribute the Fermi momentum for the symmetry energy.

label

Attribute providing the label the data is references for figures.

model

Attribute model.

nm_cs2

Attribute the neutron matter sound speed (c_s/c)^2.

`nm_den`
: Attribute the neutron matter density.

`nm_e2a`
: Attribute the neutron matter energy per particle.

`nm_gap`
: Attribute the neutron matter pairing gap.

`nm_kfn`
: Attribute the neutron matter neutron Fermi momentum.

`nm_pre`
: Attribute the neutron matter pressure.

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

`param`
: Attribute param.

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

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

`sm_cs2`
: Attribute the symmetric matter sound speed (c_s/c)^2.

`sm_den`
: Attribute the symmetric matter density.

`sm_e2a`
: Attribute the symmetric matter energy per particle.

`sm_gap`
: Attribute the symmetric matter pairing gap.

`sm_kf`
: Attribute the symmetric matter Fermi momentum.

`sm_kfn`
: Attribute the symmetric matter neutron Fermi momentum.

`sm_pre`
: Attribute the symmetric matter pressure.

`nucleardatapy.setup_pheno.models_pheno()`

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_pheno.`**`params_pheno`**`(`*`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_setupPheno.py

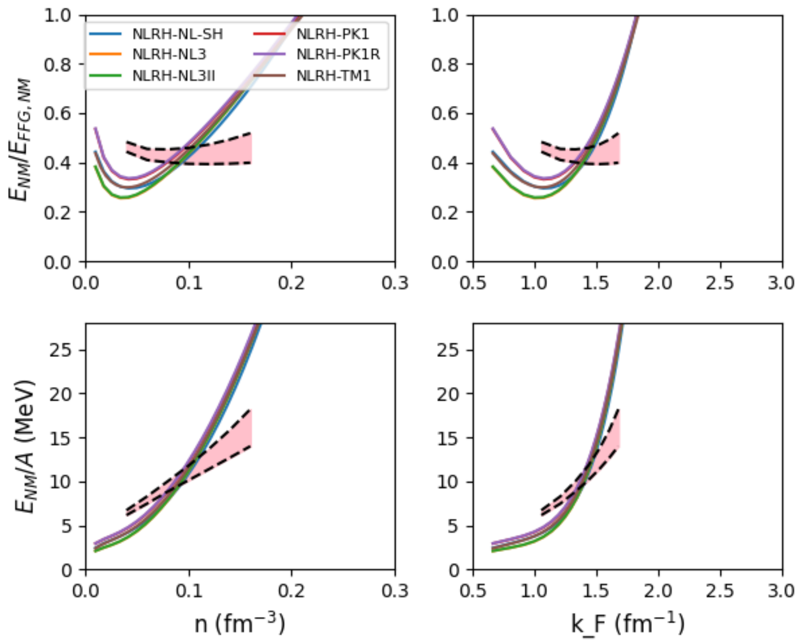

Fig. 10: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.

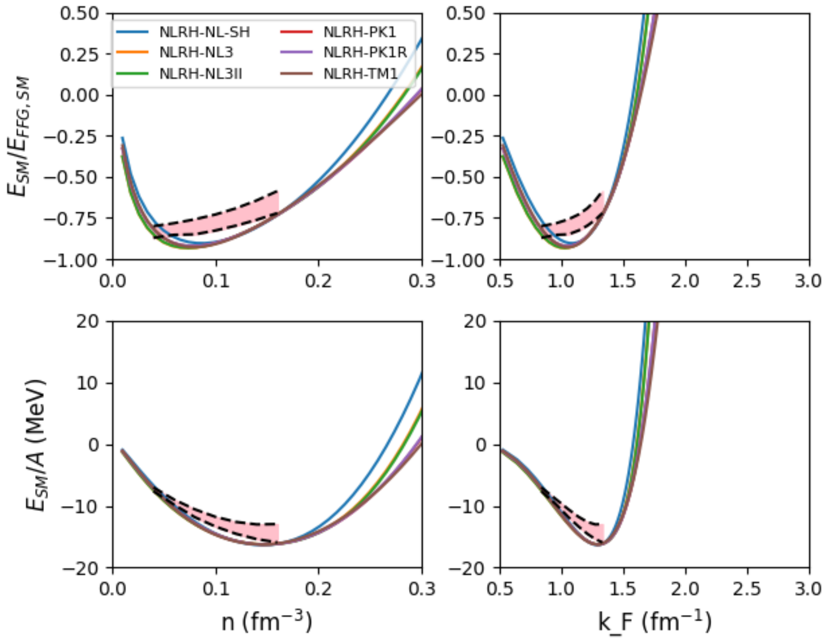

Fig. 11: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.

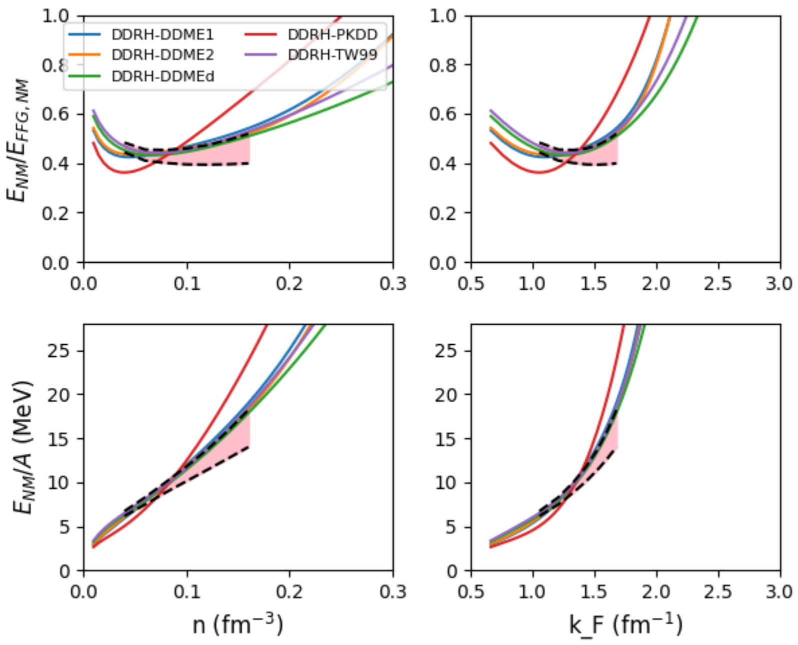

Fig. 12: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree (DDRH) approach available in the nucleardatapy toolkit.

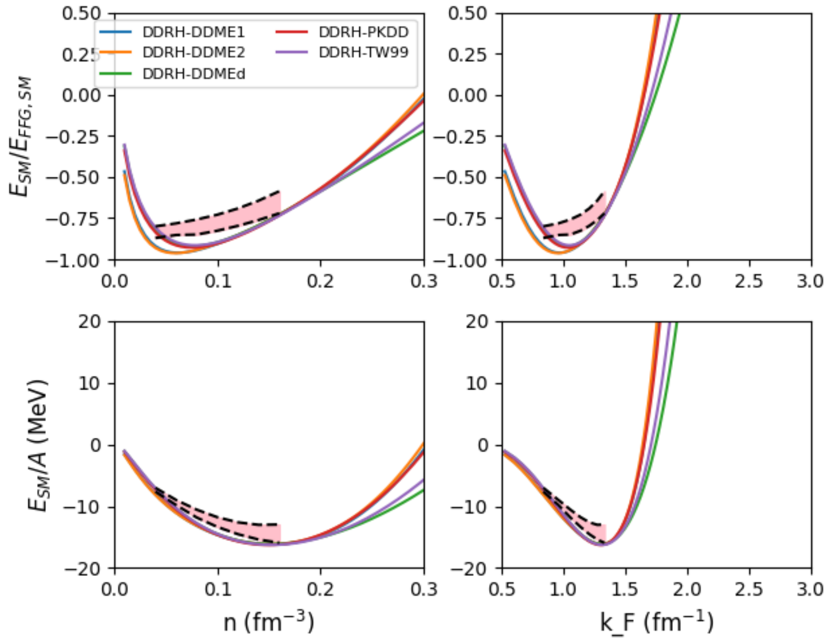

Fig. 13: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

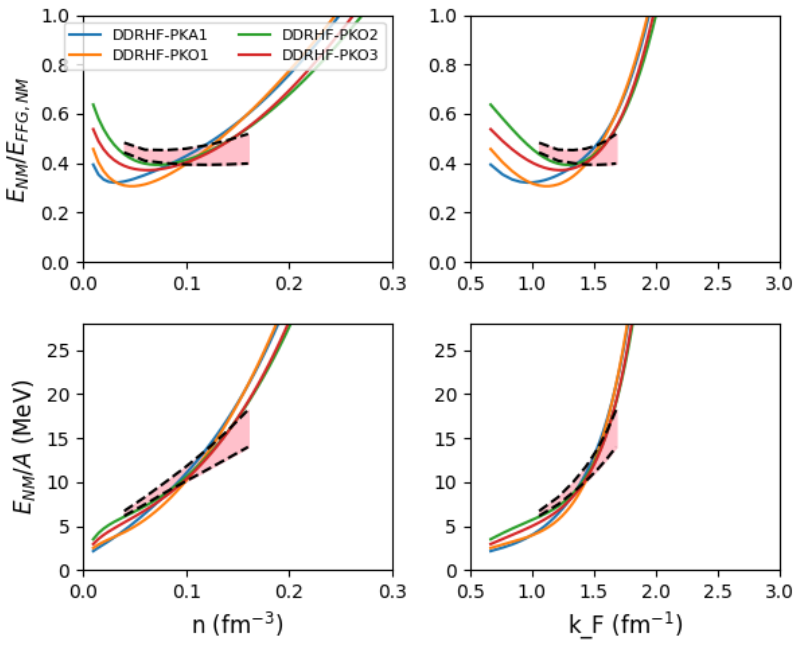

Fig. 14: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

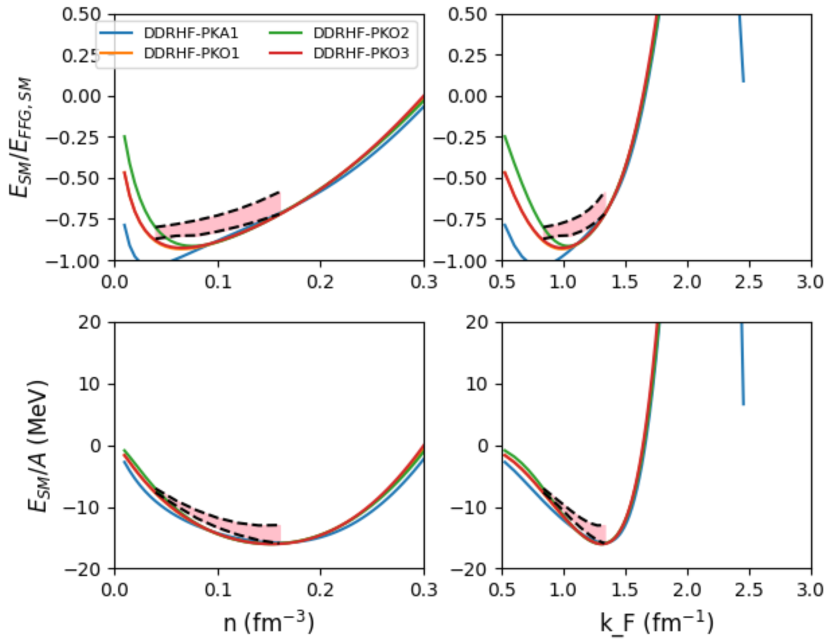

Fig. 15: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

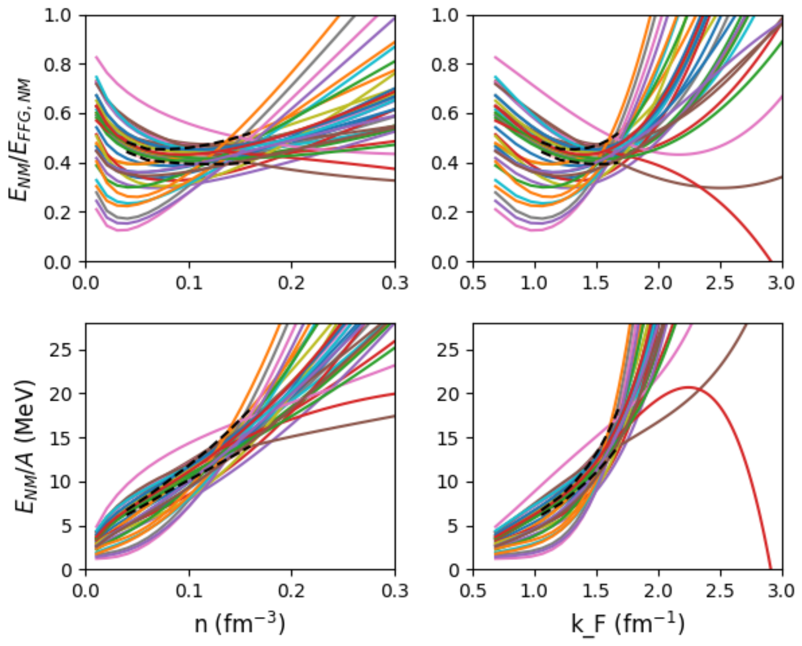

Fig. 16: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

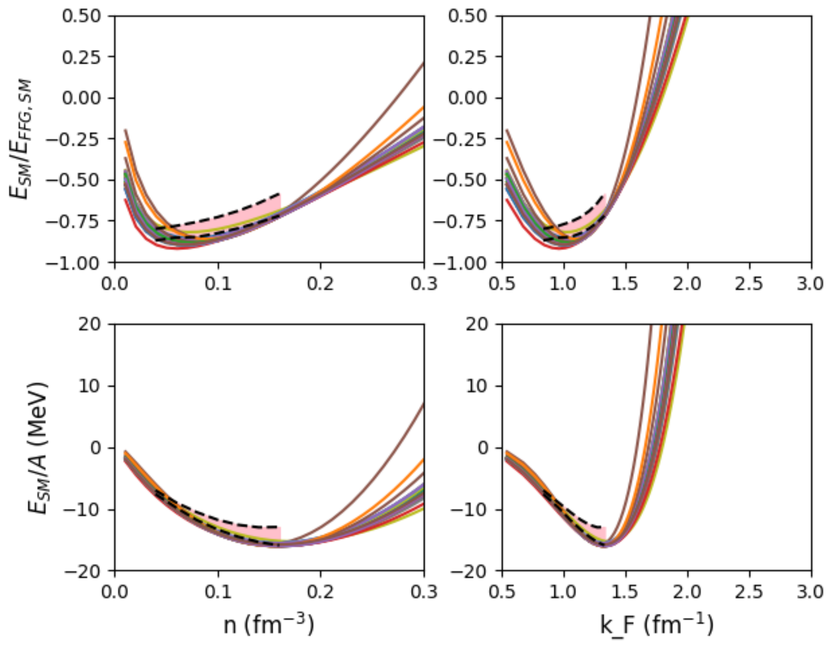

Fig. 17: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

2.4 SetupEOSHIC

class `nucleardatapy.setup_eoshic.`**`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:

`init_self`()

Initialize variables in self.

`print_outputs`()

Method which print outputs on terminal’s screen.

`nucleardatapy.setup_eoshic.`**`constraints_EOSHIC`**()

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].

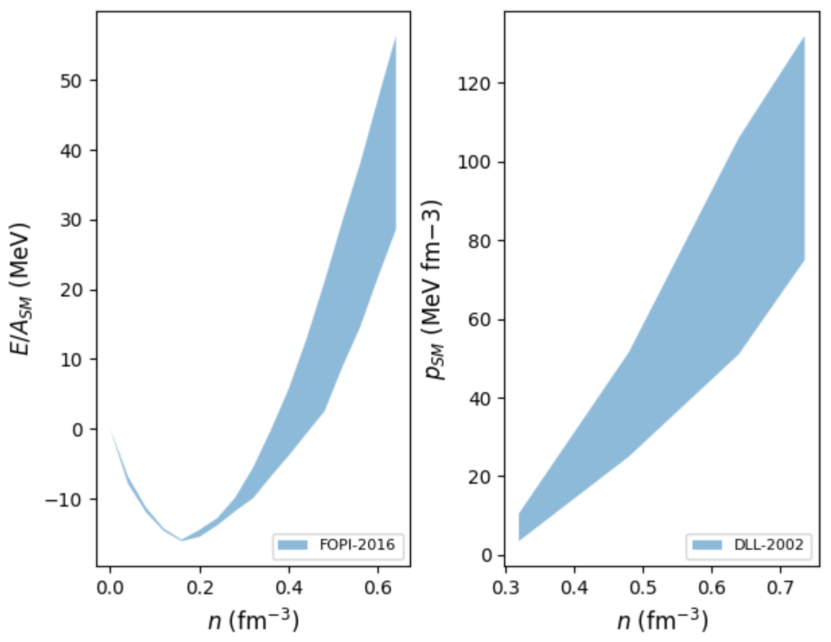

Fig. 18: HIC Experimental constraints for the energy per particle (left) and pressure (right) in SM as a function of the particle density for different analyses available in the *nuda* toolkit.

2.5 SetupCrust

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

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 (mass of the nucleus).

N

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

N_g

Attribute N_g (number of neutrons in the gas).

RWS

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

Z

Attribute Z (charge of the nucleus).

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).

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.

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].

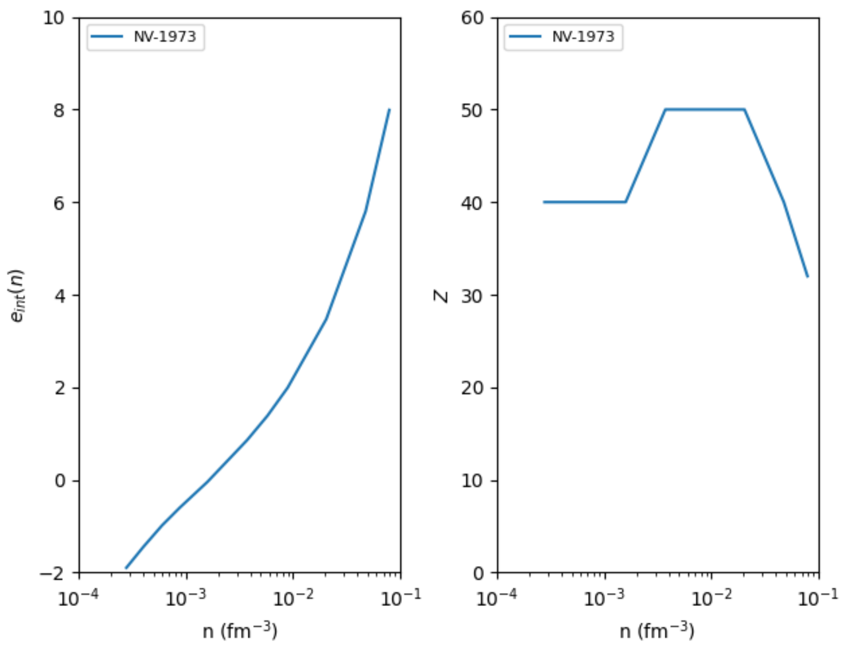

Fig. 19: Properties of the crust as given by the models available in the nuda toolkit.

2.6 SetupMassesExp

class `nucleardatapy.setup_masses_exp.`**`SetupMassesExp`**(*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

`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_masses_exp.tables_masses_exp()`

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_masses_exp.versions_masses_exp`(*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_setupMassesExp.py

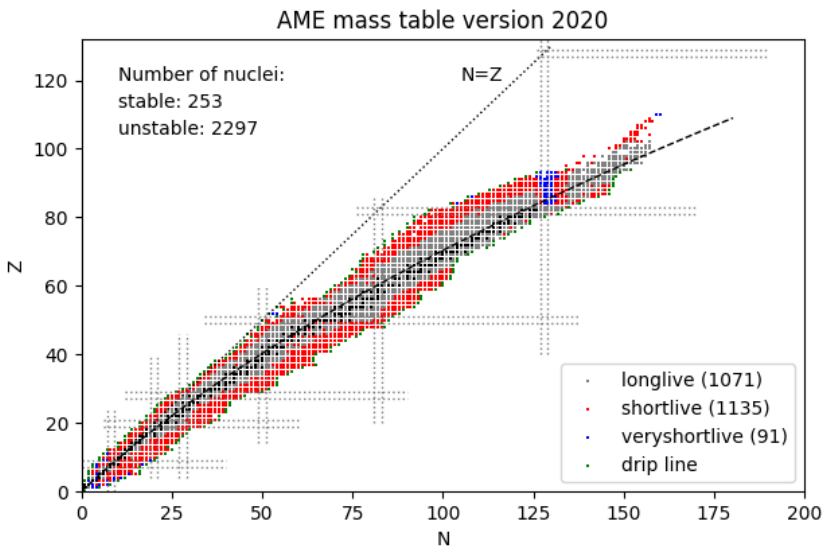

Fig. 20: The nuclear chart based on AME 2020 table. The different colors correspond to the different measured half-times of nuclei.

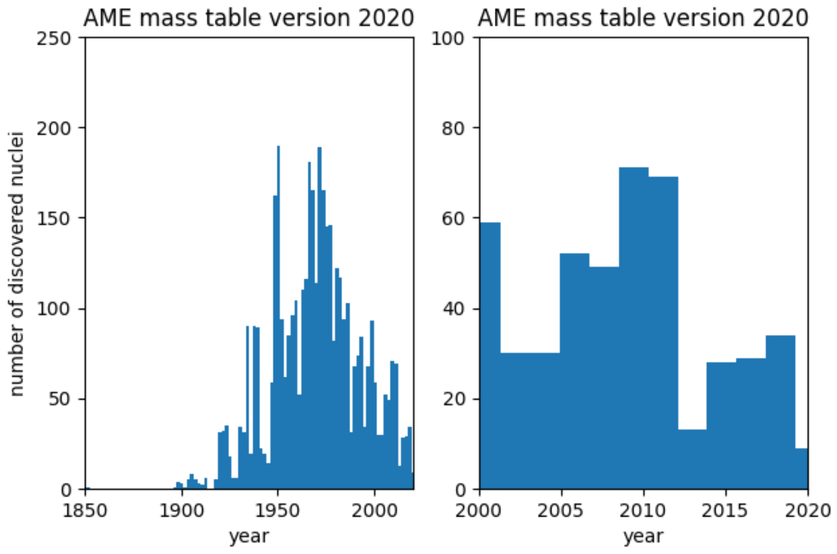

Fig. 21: Histogram showing the distribution of nuclei per discovery year, since the first one discovered in 1851.

2.7 SetupMassesTheory

class nucleardatapy.setup_masses_theory.**SetupMassesTheory**(*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:

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_masses_theory.**tables_masses_theory**()

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-HFB26']

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_setupMassesTheory.py

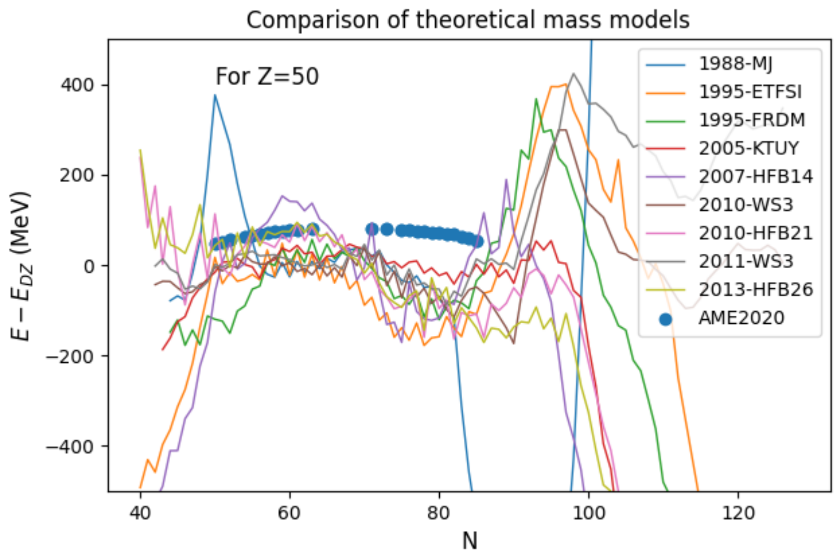

Fig. 22: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 50.

2.8 SetupRadCh

`class nucleardatapy.setup_rad_ch.SetupRadCh(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.

`RadCh_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.

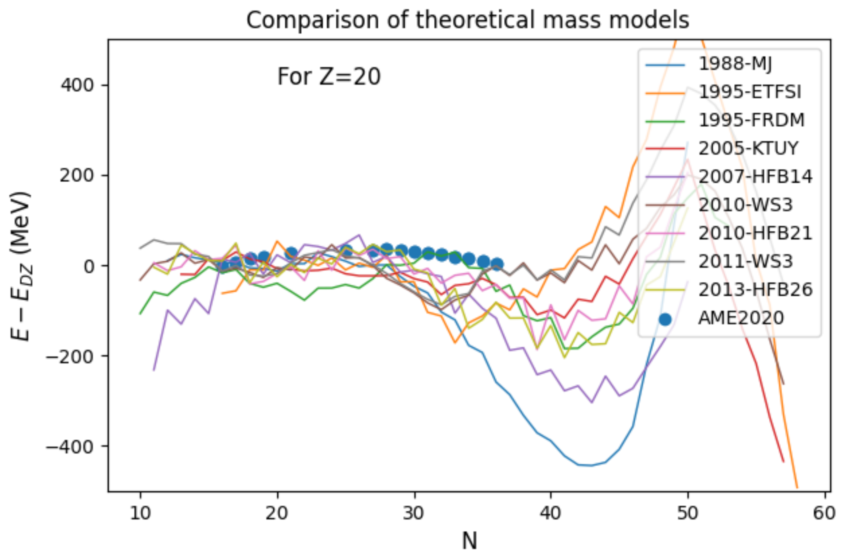

Fig. 23: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 20.

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_rad_ch.**tables_rad_ch()**

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].

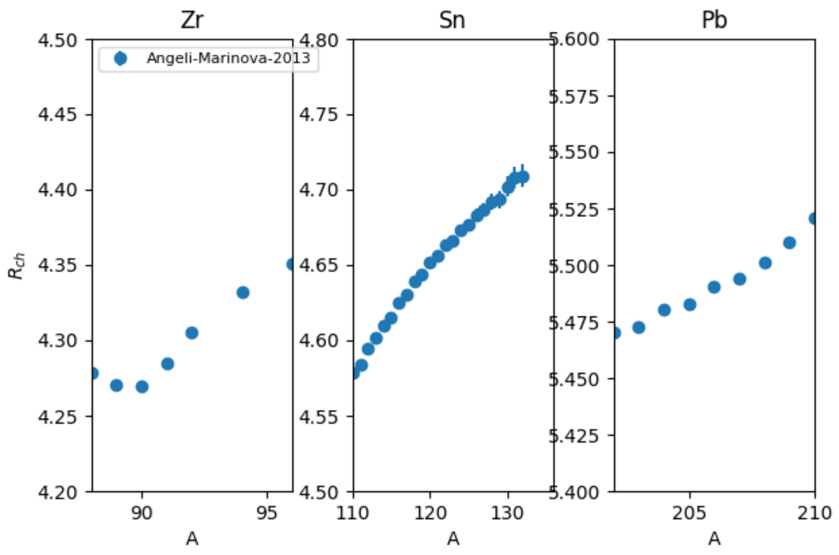

Fig. 24: Charge radii for Zn, Sn, and Pb isotopes and for the models available in the nuda toolkit.

2.9 SetupISGMR

class nucleardatapy.setup_ISGMR.**SetupISGMR**(*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'.

Attributes:

A

Attribute A (mass of the nucleus).

E_cen

Attribute energy centroid.

E_erra

Attribute list with + and - uncertainty

E_errm

Attribute (-) uncertainty in the energy centroid.

E_errp

Attribute (+) uncertainty in the energy centroid.

E_errs

Attribute symmetrised uncertainty (average between + and - uncertainty).

`E_unit`
Attribute energy unit.

`Z`
Attribute Z (charge of the nucleus).

`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.

`table`
Attribute table.

`nucleardatapy.setup_ISGMR.`**`tables_isgmr`**`()`

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'.

Returns
The list of tables.

Return type
list[str].

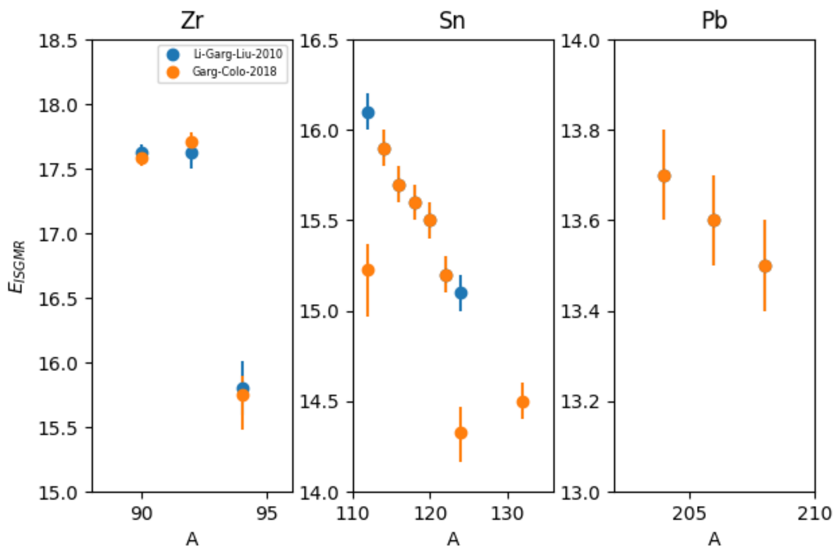

Fig. 25: ISGMR energies available in the nucleardatapy toolkit.

2.10 SetupEsymLsym

class `nucleardatapy.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'.

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.

`nucleardatapy.setup_EsymLsym.constraints_EsymLsym()`

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].

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

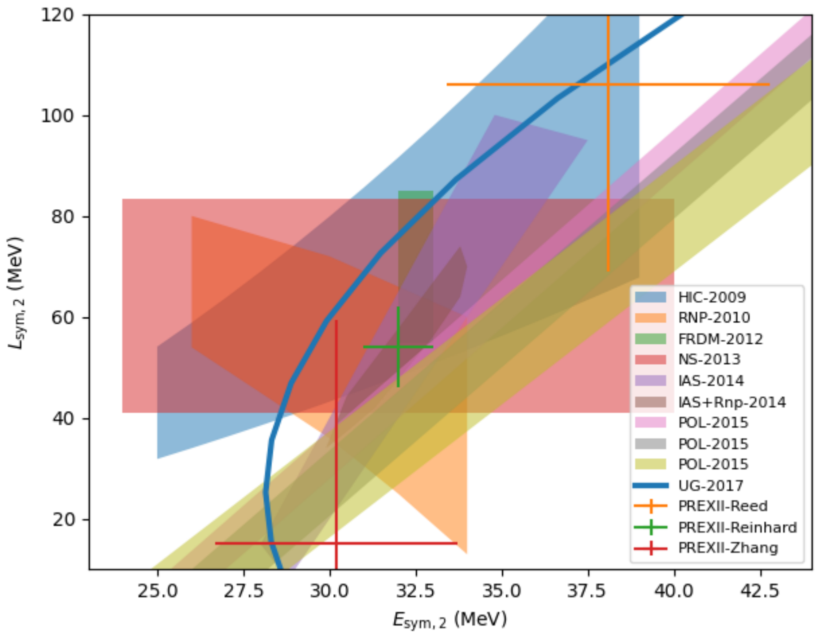

Fig. 26: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nucleardatapy toolkit.

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

nucleardatapy, 4
nucleardatapy.setup_crust, 20
nucleardatapy.setup_eoshic, 19
nucleardatapy.setup_EsymLsym, 31
nucleardatapy.setup_ISGMR, 29
nucleardatapy.setup_masses_exp, 22
nucleardatapy.setup_masses_theory, 26
nucleardatapy.setup_micro, 7
nucleardatapy.setup_micro_band, 8
nucleardatapy.setup_pheno, 11
nucleardatapy.setup_rad_ch, 27

INDEX

A

A (*nucleardatapy.setup_crust.SetupCrust attribute*), 20
A (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
alpha (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31

C

constraint (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
constraints_EOSHIC() (*in module nucleardatapy.setup_eoshic*), 19
constraints_EsymLsym() (*in module nucleardatapy.setup_EsymLsym*), 31

D

den (*nucleardatapy.setup_crust.SetupCrust attribute*), 20
den (*nucleardatapy.setup_micro_band.SetupMicroBand attribute*), 8
den_cgs (*nucleardatapy.setup_crust.SetupCrust attribute*), 20
den_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
diff() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 26
diff_exp() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 26
dist_nbNuc (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 22
dist_year (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 22
drip() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 22
drip() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 26

E

e2a_int (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
e2a_int2 (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
e2a_int_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
e2a_rm (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
E_cen (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
E_erra (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
E_errm (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
E_errp (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
E_errs (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
E_unit (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 29
Esym (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
esym_den (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
esym_e2a (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
Esym_err (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
esym_kf (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
Esym_max (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
Esym_min (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31

F

flagI (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
flagInterp (*nucleardatapy.setup_masses_exp.SetupMassesExp at-*

tribute), 23

I

init_self() (*nucleardatapy.setup_eoshic.SetupEOSHIC method*), 19
init_self() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 26
init_self() (*nucleardatapy.setup_micro.SetupMicro method*), 7
init_self() (*nucleardatapy.setup_micro_band.SetupMicroBand method*), 11

L

label (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
label (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 30
label (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
label (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
label (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 27
Lsym (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
Lsym_err (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
Lsym_max (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
Lsym_min (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31

M

matter (*nucleardatapy.setup_micro_band.SetupMicroBand attribute*), 11
model (*nucleardatapy.setup_micro.SetupMicro attribute*), 7
model (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
models (*nucleardatapy.setup_micro_band.SetupMicroBand attribute*), 11
models_crust() (*in module nucleardatapy.setup_crust*), 21
models_micro() (*in module nucleardatapy.setup_micro*), 7
models_pheno() (*in module nucleardatapy.setup_pheno*), 14
module
    nucleardatapy, 4
    nucleardatapy.setup_crust, 20
    nucleardatapy.setup_eoshic, 19
    nucleardatapy.setup_EsymLsym, 31
    nucleardatapy.setup_ISGMR, 29
    nucleardatapy.setup_masses_exp, 22
    nucleardatapy.setup_masses_theory, 26
    nucleardatapy.setup_micro, 7
    nucleardatapy.setup_micro_band, 8
    nucleardatapy.setup_pheno, 11
    nucleardatapy.setup_rad_ch, 27
mu_n (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
mu_p (*nucleardatapy.setup_crust.SetupCrust attribute*), 21

N

N (*nucleardatapy.setup_crust.SetupCrust attribute*), 20
N_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 20
nbLine (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nbNuc (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nden (*nucleardatapy.setup_micro_band.SetupMicroBand attribute*), 11
nm_cs2 (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
nm_den (*nucleardatapy.setup_pheno.SetupPheno attribute*), 13
nm_e2a (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
nm_gap (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
nm_kfn (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
nm_pre (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
note (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
note (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 30
note (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
note (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
note (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 27
nucA (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucA (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 27
nucBE (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23

nucBE_err (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucHT (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucleardatapy
  module, 4
nucleardatapy.setup_crust
  module, 20
nucleardatapy.setup_eoshic
  module, 19
nucleardatapy.setup_EsymLsym
  module, 31
nucleardatapy.setup_ISGMR
  module, 29
nucleardatapy.setup_masses_exp
  module, 22
nucleardatapy.setup_masses_theory
  module, 26
nucleardatapy.setup_micro
  module, 7
nucleardatapy.setup_micro_band
  module, 8
nucleardatapy.setup_pheno
  module, 11
nucleardatapy.setup_rad_ch
  module, 27
nucN (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucN (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 28
nucRch (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 28
nucRch_err (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 28
nucStbl (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucSymb (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucSymb (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 28
nucYear (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucZ (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
nucZ (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 28

P

param (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
params_pheno() (*in module nucleardatapy.setup_pheno*), 14
print_outputs() (*nucleardatapy.setup_crust.SetupCrust method*), 21
print_outputs() (*nucleardatapy.setup_eoshic.SetupEOSHIC method*), 19
print_outputs() (*nucleardatapy.setup_EsymLsym.SetupEsymLsym method*), 31
print_outputs() (*nucleardatapy.setup_ISGMR.SetupISGMR method*), 30
print_outputs() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 23
print_outputs() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 26
print_outputs() (*nucleardatapy.setup_micro.SetupMicro method*), 7
print_outputs() (*nucleardatapy.setup_micro_band.SetupMicroBand method*), 11
print_outputs() (*nucleardatapy.setup_pheno.SetupPheno method*), 14
print_outputs() (*nucleardatapy.setup_rad_ch.SetupRadCh method*), 28

R

R_unit (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 27
RadCh_isotopes() (*nucleardatapy.setup_rad_ch.SetupRadCh method*), 27
ref (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 31
ref (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 30
ref (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 23
ref (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
ref (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 28
RWS (*nucleardatapy.setup_crust.SetupCrust attribute*), 20

S

select() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 23
select_year() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 24
SetupCrust (*class in nucleardatapy.setup_crust*), 20

SetupEOSHIC (*class in nucleardatapy.setup_eoshic*), 19
SetupEsymLsym (*class in nucleardatapy.setup_EsymLsym*), 31
SetupISGMR (*class in nucleardatapy.setup_ISGMR*), 29
SetupMassesExp (*class in nucleardatapy.setup_masses_exp*), 22
SetupMassesTheory (*class in nucleardatapy.setup_masses_theory*), 26
SetupMicro (*class in nucleardatapy.setup_micro*), 7
SetupMicroBand (*class in nucleardatapy.setup_micro_band*), 8
SetupPheno (*class in nucleardatapy.setup_pheno*), 11
SetupRadCh (*class in nucleardatapy.setup_rad_ch*), 27
sm_cs2 (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
sm_den (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
sm_e2a (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
sm_gap (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
sm_kf (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
sm_kfn (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14
sm_pre (*nucleardatapy.setup_pheno.SetupPheno attribute*), 14

T

table (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 30
tables_isgmr() (*in module nucleardatapy.setup_ISGMR*), 30
tables_masses_exp() (*in module nucleardatapy.setup_masses_exp*), 24
tables_masses_theory() (*in module nucleardatapy.setup_masses_theory*), 26
tables_rad_ch() (*in module nucleardatapy.setup_rad_ch*), 28

V

versions_masses_exp() (*in module nucleardatapy.setup_masses_exp*), 24

X

xn (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
xn_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
xp (*nucleardatapy.setup_crust.SetupCrust attribute*), 21
xpn_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 21

Z

Z (*nucleardatapy.setup_crust.SetupCrust attribute*), 20
Z (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 30
Zmax (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 22