

---

# **nucleardatapy**

***Release 0.1***

**Jérôme Margueron, IRL NPA, USA**

**Jul 05, 2024**



# CONTENTS

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Usage . . . . .	3
1.2	API . . . . .	4
1.3	Miscellaneous . . . . .	4
<b>2</b>	<b>Complement</b>	<b>7</b>
2.1	SetupMicro . . . . .	7
2.2	SetupPheno . . . . .	8
2.3	SetupEsymLsym . . . . .	11
2.4	SetupMasses . . . . .	16
2.5	SetupRadCh . . . . .	19
2.6	SetupISGMR . . . . .	20
<b>3</b>	<b>Indices and tables</b>	<b>23</b>
	<b>Python Module Index</b>	<b>25</b>
	<b>Index</b>	<b>27</b>



**nucleardatapy** (/in short nuda/) is a Python library for nuclear physicists facilitating the access to theoretical or experimental nuclear data. It is specifically designed for equation of state practitioners 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 explicitly. 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.

---



## 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 Miscellaneous

### 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 functioning. 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.



## COMPLEMENT

### 2.1 SetupMicro

**class** nucleardatapy.setup\_micro.**SetupMicro**(*model*='1998-VAR-AM-APR')

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

This choice is defined in *model*, which can be 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-DHSL59', '2019-MBPT-AM-DHSL69', '2020-MBPT-AM'.

#### Parameters

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

#### Attributes:

##### **den\_max**

Attribute maximum of the density (SM and NM).

##### **den\_min**

Attribute minimum of the density (SM and NM).

##### **esym\_den**

Attribute density array for esym.

##### **esym\_e2a**

Attribute energy per particle for esym.

##### **esym\_kf**

Attribute Fermi momentum array for esym.

##### **kf\_max**

Attribute maximum of the Fermi momentum.

##### **kf\_min**

Attribute minimum of the Fermi momentum.

##### **label**

Attribute providing the label the data is references for figures.

##### **model**

Attribute model.

**nm\_chempot**

Attribute neutron matter chemical potential.

**nm\_chempot\_err**

Attribute uncertainty in the neutron matter chemical potential.

**nm\_den**

Attribute neutron matter density.

**nm\_den\_max**

Attribute the maximum of the neutron matter density.

**nm\_den\_min**

Attribute the minimum of the neutron matter density.

**nm\_e2a**

Attribute neutron matter energy per particle.

**nm\_e2a\_err**

Attribute uncertainty in the neutron matter energy per particle.

**nm\_e2v**

Attribute neutron matter energy per unit volume.

**nm\_e2v\_err**

Attribute uncertainty in the neutron matter energy per unit volume.

**nm\_effmass**

Attribute neutron matter effective mass.

**nm\_gap**

Attribute neutron matter pairing gap.

**nm\_gap\_err**

Attribute uncertainty in the neutron matter pairing gap.

**nm\_kfn**

Attribute neutron matter Fermi momentum.

**nm\_pre**

Attribute neutron matter pressure.

**nm\_pre\_err**

Attribute uncertainty in the neutron matter pressure.

**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 cited.

**sm\_chempot**

Attribute symmetric matter chemical potential.

**sm\_chempot\_err**

Attribute uncertainty in the symmetric matter chemical potential.

**sm\_den**

Attribute symmetric matter density.

**sm\_den\_max**

Attribute the maximum of the symmetric matter density.

**sm\_den\_min**

Attribute the minimum of the symmetric matter density.

**sm\_e2a**

Attribute symmetric matter energy per particle.

**sm\_e2a\_err**

Attribute uncertainty in the symmetric matter energy per particle.

**sm\_e2v**

Attribute symmetric matter energy per unit volume.

**sm\_e2v\_err**

Attribute uncertainty in the symmetric matter energy per unit volume.

**sm\_effmass**

Attribute symmetric matter effective mass.

**sm\_gap**

Attribute symmetric matter pairing gap.

**sm\_gap\_err**

Attribute uncertainty in the symmetric matter pairing gap.

**sm\_kfn**

Attribute symmetric matter Fermi momentum.

**sm\_pre**

Attribute symmetric matter pressure.

**sm\_pre\_err**

Attribute uncertainty in the symmetric matter pressure.

**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-DHSL59', '2019-MBPT-AM-DHSL69', '2020-MBPT-AM'.

**Returns**

The list of models.

**Return type**

list[str].

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

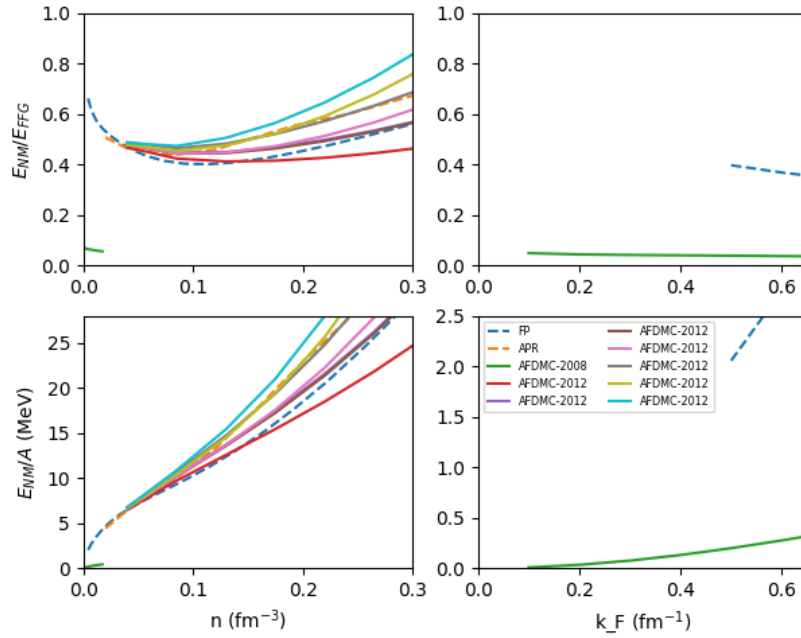


Fig. 1: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

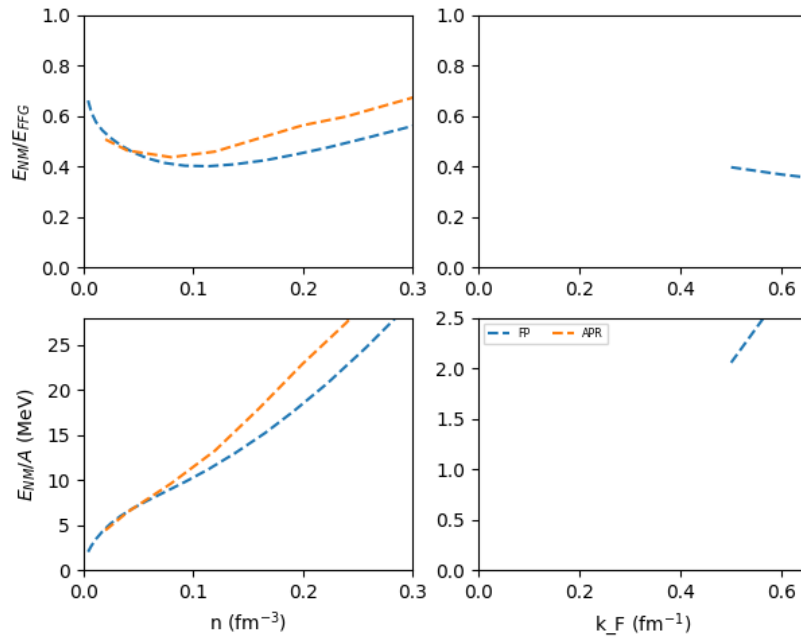


Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

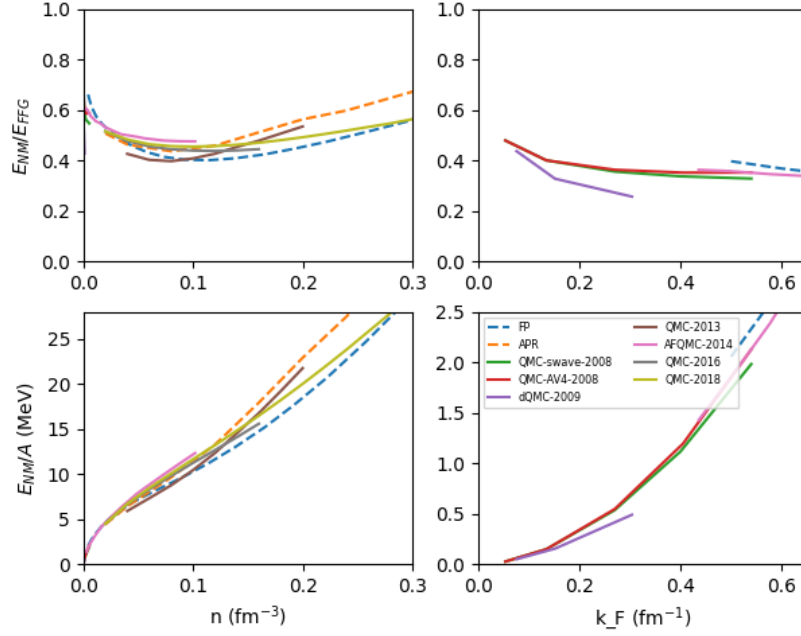


Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

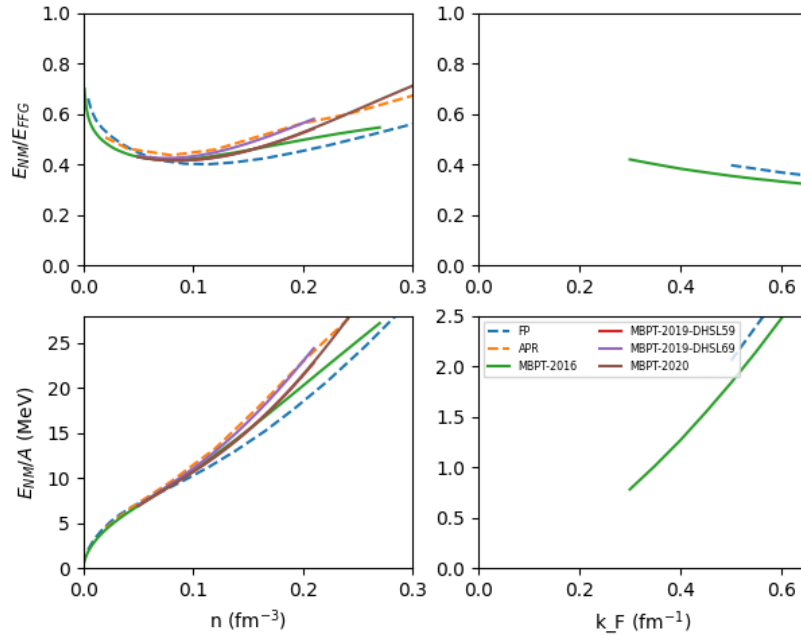
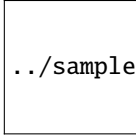


Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.



../samples/nucleardatapy\_plots/figs/plot\_SetupMicro\_gap\_NM.png

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

## 2.2 SetupBand

**class** nucleardatapy.setup\_band.**SetupBand**(models, nden=10, ne=1000)

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

### 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.

### Attributes:

#### 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 chosen by the toolkit practitioner. This choice is defined in the variables *model* and *param*.

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

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

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

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

### Parameters

- **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:****label**

Attribute providing the label the data is references for figures.

**model**

Attribute model.

**nm\_cs2**

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

**nm\_den**

Attribute neutron matter density.

**nm\_e2a**

Attribute neutron matter energy per particle.

**nm\_gap**

Attribute neutron matter pairing gap.

**nm\_kfn**

Attribute neutron matter Fermi momentum.

**nm\_pre**

Attribute 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 symmetric matter sound speed  $(c_s/c)^2$ .

**sm\_den**

Attribute symmetric matter density.

**sm\_e2a**

Attribute symmetric matter energy per particle.

**sm\_gap**

Attribute symmetric matter pairing gap.

**sm\_kfn**

Attribute symmetric matter Fermi momentum.

**sm\_pre**

Attribute 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', 'SKSIGMA', '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`

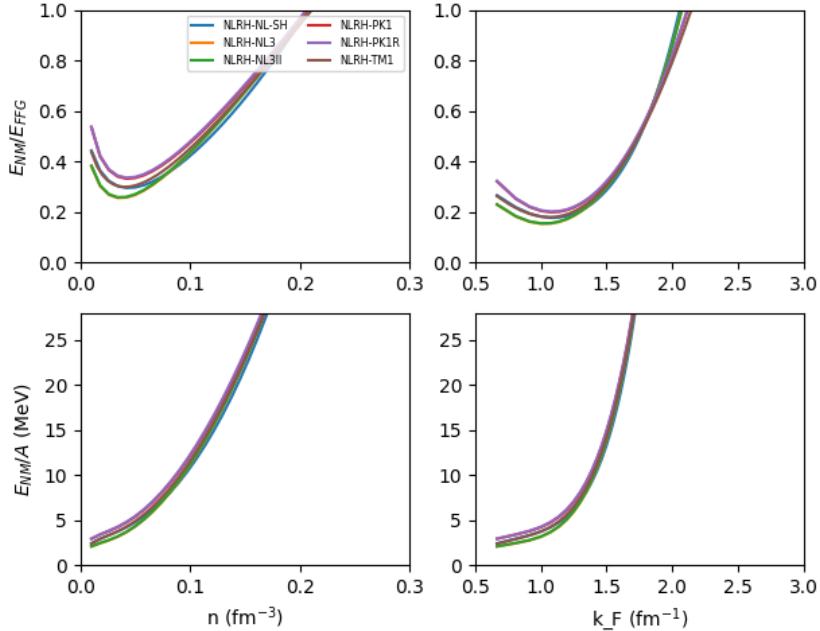


Fig. 6: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.

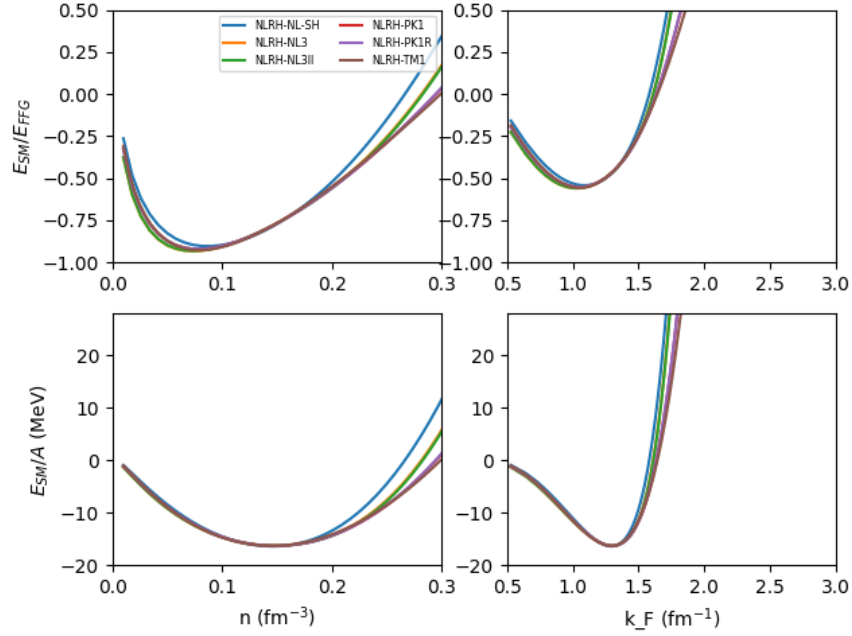


Fig. 7: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.

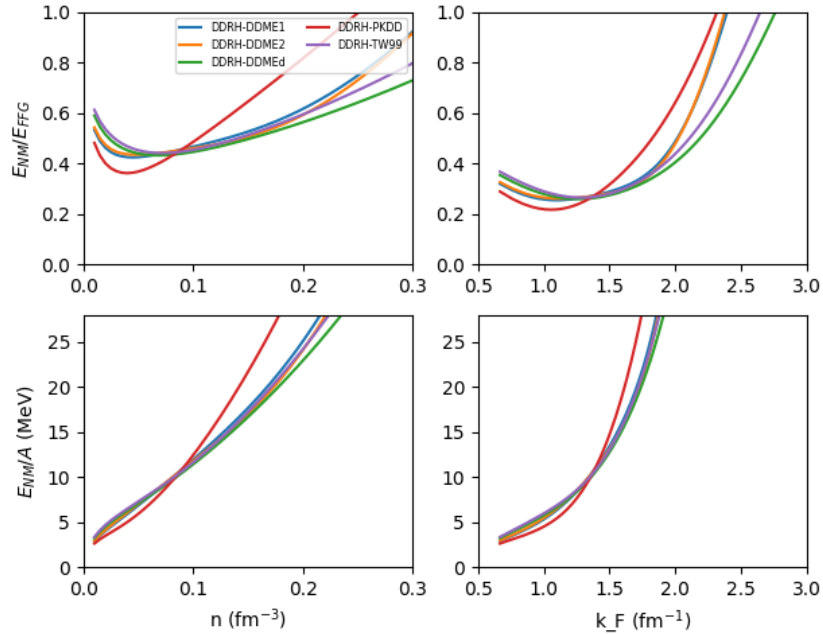


Fig. 8: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree (DDRH) approach available in the nucleardatapy toolkit.

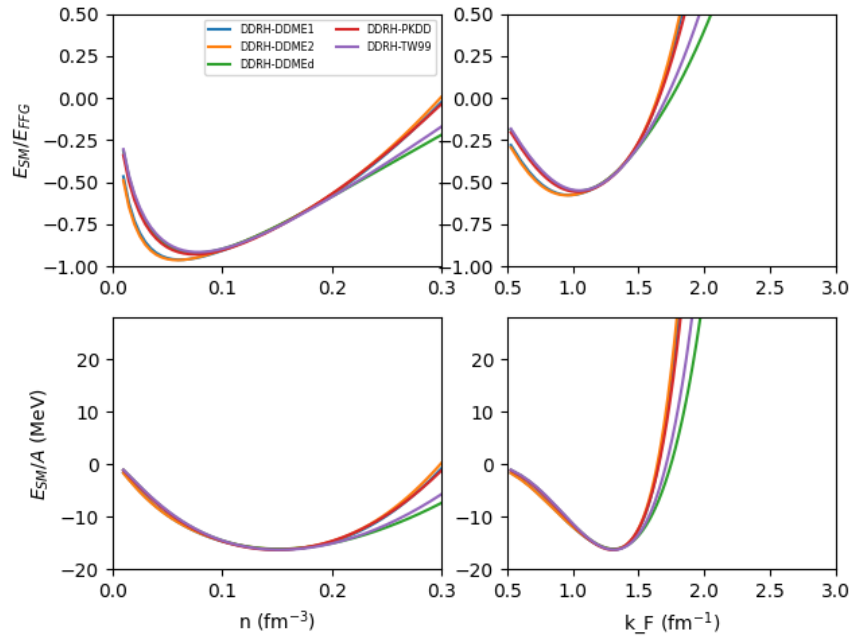


Fig. 9: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree (DDRH) approach available in the nucleardatapy toolkit.

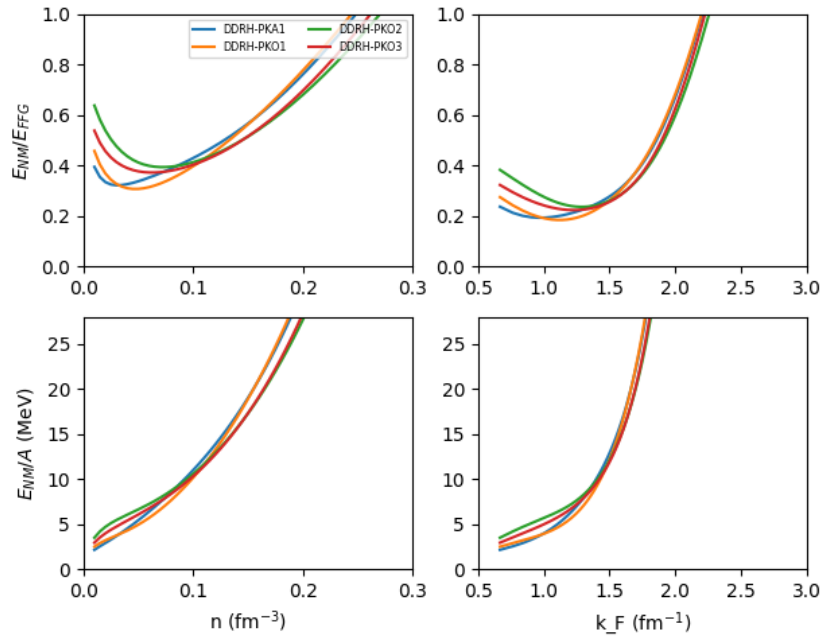


Fig. 10: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

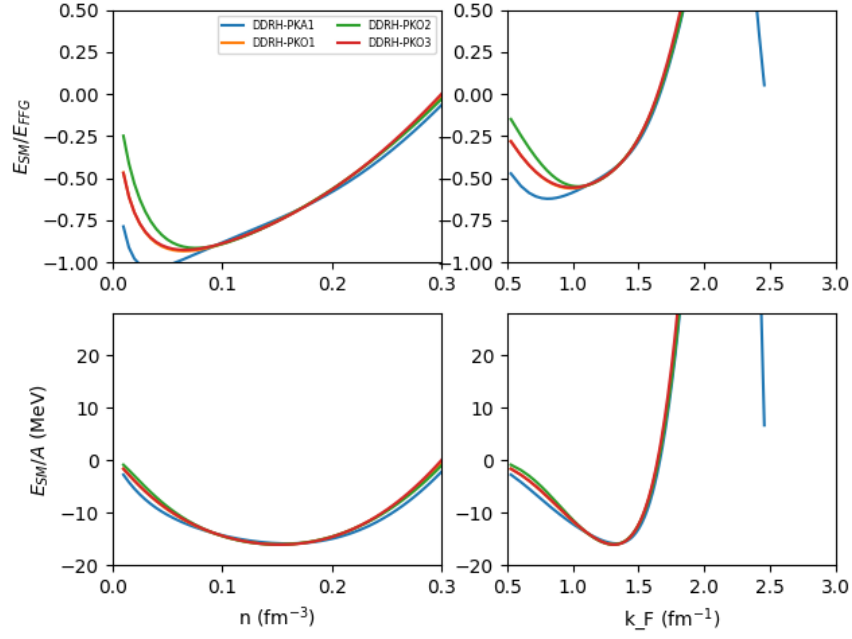


Fig. 11: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

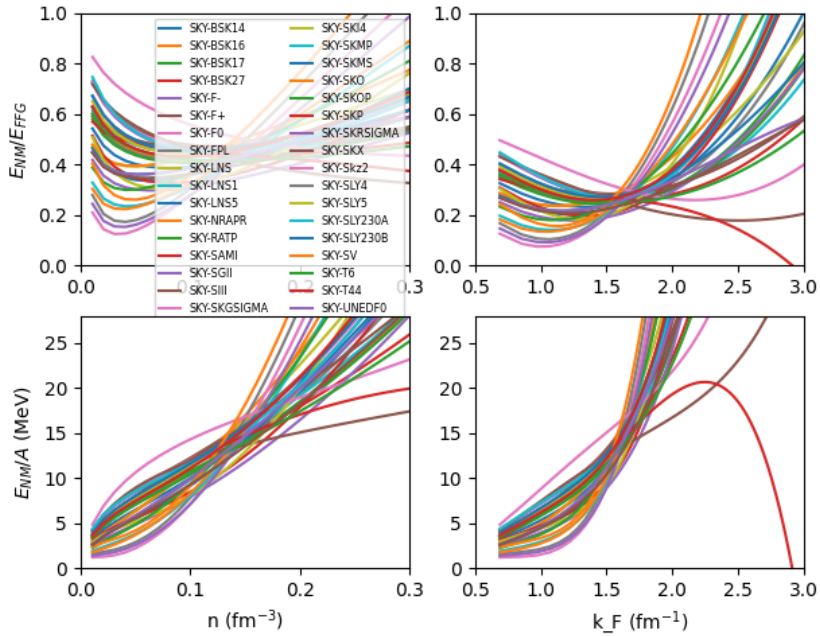


Fig. 12: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

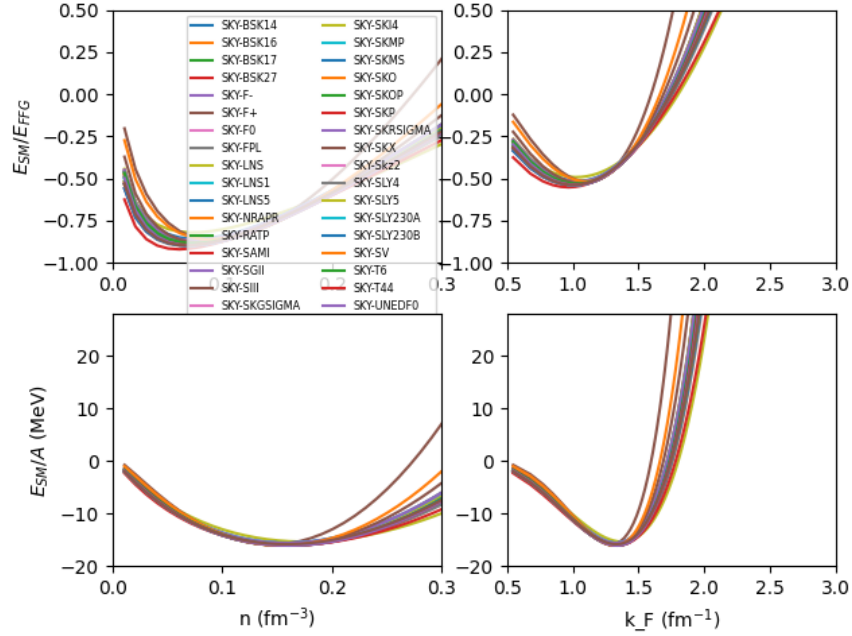


Fig. 13: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

## 2.4 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-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.

#### Lsym

Attribute Lsym.

#### Lsym\_err

Attribute with uncertainty in Lsym.

#### 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 cited.

**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', '2021-PREXII-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`

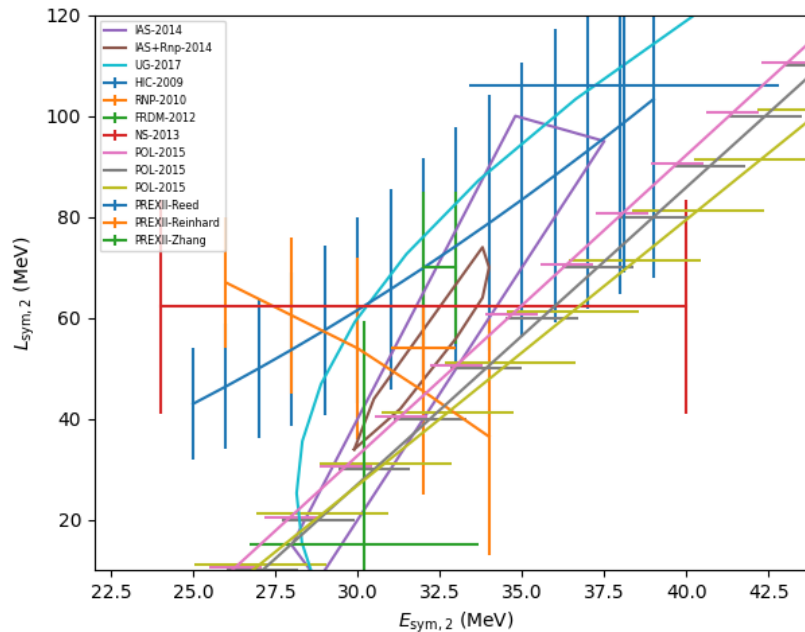


Fig. 14: This figure shows the  $E_{\text{sym},2}$  versus  $L_{\text{sym},2}$  correlation for the different constraints available in the nucleardatapy toolkit.

## 2.5 SetupMasses

**class** nucleardatapy.setup\_masses.**SetupMasses**(*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 be 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:

#### A

Attribute A (mass of the nucleus).

#### BE

Attribute BE (Binding Energy) of the nucleus.

#### BE\_err

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

#### HT

Attribute HT (half-Time) of the nucleus.

#### I

Attribute I.

#### Interp

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

#### N

Attribute N (number of neutrons of the nucleus).

#### Z

Attribute Z (charge of the nucleus).

#### Zmax

maximum charge of nuclei present in the table.

#### Type

Attribute Zmax

#### 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.

#### 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.

**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'.
- **every** (*int, optional. Default every = 1.*) – consider only 1 out of *every* nuclei in the table.

**stbl**

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

**symb**

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

**year**

Attribute year of the discovery of the nucleus.

**nucleardatapy.setup\_masses.tables\_masses()**

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.versions\_masses**(*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_setupMasses.py`

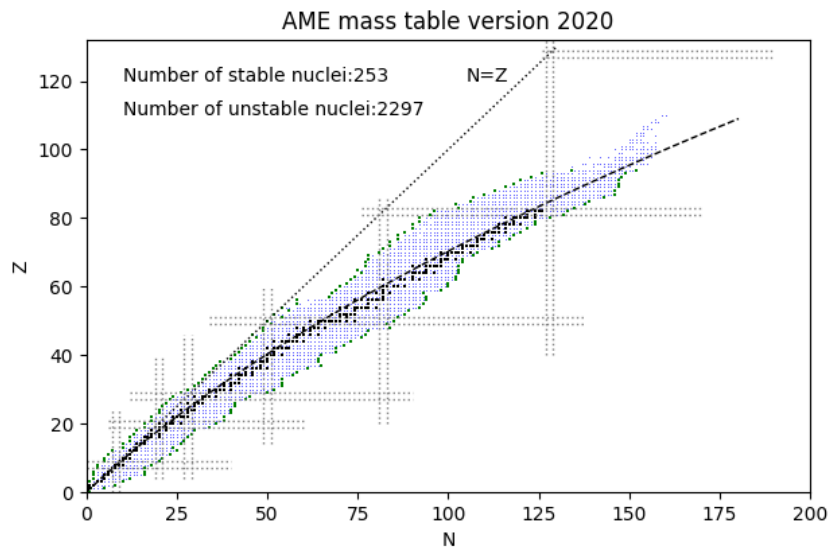


Fig. 15: This figure shows the nuclear chart based on AME 2020 table.

## 2.6 SetupRadCh

**class** `nucleardatapy.setup_rad_ch.SetupRadCh`(*table*='2013-Angeli')

Instantiate the object with charge radii chosen from a table.

This choice is defined in the variable *table*.

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

### Parameters

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

### Attributes:

#### A

Attribute A (mass of the nucleus).

#### N

Attribute N (number of neutrons of the nucleus).

#### R\_ch

Attribute R\_ch (charge radius) in fm.

#### R\_ch\_err

Attribute uncertainty in R\_ch (charge radius) in fm.

#### R\_unit

Attribute radius 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.

**symb**

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

**nucleardatapy.setup\_rad\_ch.tables\_rad\_ch()**

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

**Returns**

The list of tables.

**Return type**

list[str].

## 2.7 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 choosen 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].

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### n

- nucleardatapy, 4
- nucleardatapy.setup\_band, ??
- nucleardatapy.setup\_EsymLsym, 11
- nucleardatapy.setup\_ISGMR, 20
- nucleardatapy.setup\_masses, 16
- nucleardatapy.setup\_micro, 7
- nucleardatapy.setup\_pheno, 8
- nucleardatapy.setup\_rad\_ch, 19





## A

A (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 20  
A (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 16  
A(*nuclear\_datapy.setup\_rad\_ch.SetupRadCh* attribute), 19

## B

BE (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 16  
BE\_err (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 17

## C

constraint (*nuclear\_datapy.setup\_EsymLsym.SetupEsymLsym* attribute), 16  
constraints\_EsymLsym() (in module *nuclear\_datapy.setup\_EsymLsym*), 16

## D

drip() (*nuclear\_datapy.setup\_masses.SetupMasses* method), 17

## E

E\_cen (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 20  
E\_erra (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 21  
E\_errm (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 21  
E\_errp (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 21  
E\_errs (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 21  
E\_unit (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 21  
Esym (*nuclear\_datapy.setup\_EsymLsym.SetupEsymLsym* attribute), 11  
Esym\_err (*nuclear\_datapy.setup\_EsymLsym.SetupEsymLsym* attribute), 11

## H

HT (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 17

## I

I (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 17  
Interp (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 17

## L

label (*nuclear\_datapy.setup\_EsymLsym.SetupEsymLsym* attribute), 16  
label (*nuclear\_datapy.setup\_ISGMR.SetupISGMR* attribute), 21  
label (*nuclear\_datapy.setup\_masses.SetupMasses* attribute), 18  
label (*nuclear\_datapy.setup\_micro.SetupMicro* attribute), 7  
label (*nuclear\_datapy.setup\_pheno.SetupPheno* attribute), 10  
label (*nuclear\_datapy.setup\_rad\_ch.SetupRadCh* attribute), 20  
Lsym (*nuclear\_datapy.setup\_EsymLsym.SetupEsymLsym* attribute), 16  
Lsym\_err (*nuclear\_datapy.setup\_EsymLsym.SetupEsymLsym* attribute), 16

## M

model (*nuclear\_datapy.setup\_micro.SetupMicro* attribute), 7  
model (*nuclear\_datapy.setup\_pheno.SetupPheno* attribute), 10  
models\_micro() (in module *nuclear\_datapy.setup\_micro*), 8  
models\_pheno() (in module *nuclear\_datapy.setup\_pheno*), 11  
module  
    *nuclear\_datapy*, 4  
    *nuclear\_datapy.setup\_EsymLsym*, 11  
    *nuclear\_datapy.setup\_ISGMR*, 20

nucleardatapy.setup\_masses, 16  
 nucleardatapy.setup\_micro, 7  
 nucleardatapy.setup\_pheno, 8  
 nucleardatapy.setup\_rad\_ch, 19

## N

N (*nucleardatapy.setup\_masses.SetupMasses* attribute), 17  
 N (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 19  
 nbLine (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18  
 nbNuc (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18  
 nm\_cs2 (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 nm\_den (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 nm\_den (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 nm\_e2a (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 nm\_e2a (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 nm\_e2a\_err (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 nm\_gap (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 nm\_gap (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 nm\_kfn (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 nm\_kfn (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 nm\_pre (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 nm\_pre (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 nm\_pre\_err (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 note (*nucleardatapy.setup\_EsymLsym.SetupEsymLsym* attribute), 16  
 note (*nucleardatapy.setup\_ISGMR.SetupISGMR* attribute), 21  
 note (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18  
 note (*nucleardatapy.setup\_micro.SetupMicro* attribute), 7  
 note (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 note (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20  
 nucleardatapy  
   module, 4  
 nucleardatapy.setup\_EsymLsym

  module, 11  
 nucleardatapy.setup\_ISGMR  
   module, 20  
 nucleardatapy.setup\_masses  
   module, 16  
 nucleardatapy.setup\_micro  
   module, 7  
 nucleardatapy.setup\_pheno  
   module, 8  
 nucleardatapy.setup\_rad\_ch  
   module, 19

## P

param (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 params\_pheno() (in module *nucleardatapy.setup\_pheno*), 11  
 print\_outputs() (*nucleardatapy.setup\_EsymLsym.SetupEsymLsym* method), 16  
 print\_outputs() (*nucleardatapy.setup\_ISGMR.SetupISGMR* method), 21  
 print\_outputs() (*nucleardatapy.setup\_masses.SetupMasses* method), 18  
 print\_outputs() (*nucleardatapy.setup\_micro.SetupMicro* method), 8  
 print\_outputs() (*nucleardatapy.setup\_pheno.SetupPheno* method), 10  
 print\_outputs() (*nucleardatapy.setup\_rad\_ch.SetupRadCh* method), 20

## R

R\_ch (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20  
 R\_ch\_err (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20  
 R\_unit (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20  
 ref (*nucleardatapy.setup\_EsymLsym.SetupEsymLsym* attribute), 16  
 ref (*nucleardatapy.setup\_ISGMR.SetupISGMR* attribute), 21  
 ref (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18  
 ref (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8  
 ref (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10  
 ref (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20

## S

`select()` (*nucleardatapy.setup\_masses.SetupMasses* method), 18

`SetupEsymLsym` (class in *nucleardatapy.setup\_EsymLsym*), 11

`SetupISGMR` (class in *nucleardatapy.setup\_ISGMR*), 20

`SetupMasses` (class in *nucleardatapy.setup\_masses*), 16

`SetupMicro` (class in *nucleardatapy.setup\_micro*), 7

`SetupPheno` (class in *nucleardatapy.setup\_pheno*), 8

`SetupRadCh` (class in *nucleardatapy.setup\_rad\_ch*), 19

`sm_cs2` (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10

`sm_den` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`sm_den` (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10

`sm_e2a` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`sm_e2a` (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10

`sm_e2a_err` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`sm_gap` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`sm_gap` (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 10

`sm_kfn` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`sm_kfn` (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 11

`sm_pre` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`sm_pre` (*nucleardatapy.setup\_pheno.SetupPheno* attribute), 11

`sm_pre_err` (*nucleardatapy.setup\_micro.SetupMicro* attribute), 8

`stbl` (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18

`symb` (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18

`symb` (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20

## T

`table` (*nucleardatapy.setup\_ISGMR.SetupISGMR* attribute), 21

`tables_isgmr()` (in module *nucleardatapy.setup\_ISGMR*), 21

`tables_masses()` (in module *nucleardatapy.setup\_masses*), 18

`tables_rad_ch()` (in module *nucleardatapy.setup\_rad\_ch*), 20

## V

`versions_masses()` (in module *nucleardatapy.setup\_masses*), 18

## Y

`year` (*nucleardatapy.setup\_masses.SetupMasses* attribute), 18

## Z

`Z` (*nucleardatapy.setup\_ISGMR.SetupISGMR* attribute), 21

`Z` (*nucleardatapy.setup\_masses.SetupMasses* attribute), 17

`Z` (*nucleardatapy.setup\_rad\_ch.SetupRadCh* attribute), 20

`Zmax` (*nucleardatapy.setup\_masses.SetupMasses* attribute), 17