# nucleardatapy

Release 0.1

Jérôme Margueron, IRL NPA, USA

## **CONTENTS**

1	Cont	tents	3
	1.1	Usage	3
	1.2	API	4
	1.3	Miscelaneous	4
2	Com	plement	7
	2.1	SetupMicroMatter	7
	2.2		8
	2.3		13
	2.4		14
	2.5	SetupHICMatter	17
	2.6		22
	2.7	SetupMassesExp	24
	2.8	SetupMassesTheory	28
	2.9		29
	2.10	1	31
	2.11	SetupEsymLsym	33
	2,11	ScupEsymEsym	33
3	Indic	ces and tables	35
Рy	thon I	Module Index	37
In	dex		39

**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 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 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

2 CONTENTS

**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.

1.3. Miscelaneous 5

## COMPLEMENT

## 2.1 SetupMicroMatter

class nucleardatapy.setup\_micro\_matter.SetupMicroMatter(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-NSC97b', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23

#### **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.

## $\verb|nucleardatapy.setup_micro_matter.models_micro_matter()|\\$

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-6', '2012-AFDMC-NM-6', '2012-AFDMC-NM-7', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '2022-AFDMC-NM', '2024-NLEFT-AM', '2024-BHF-AM-2BF-Av8p',

'2024-BHF-AM-2BF-AV18', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-CDBONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97c', '2024-BHF-AM-2BF-SSCV14', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-SSCV14', '2024-BHF-AM-23BF-AV8p', '2024-BHF-AM-23BF-AV18', '2024-BHF-AM-23BF-BONN', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy\_plots/plot\_setupMicro.py

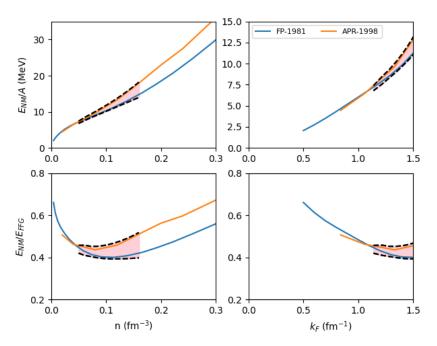


Fig. 1: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the variational models available in the nucleardatapy toolkit.

## 2.2 SetupMicroMatterBand

class nucleardatapy.setup\_micro\_matter\_band.SetupMicroMatterBand(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'.

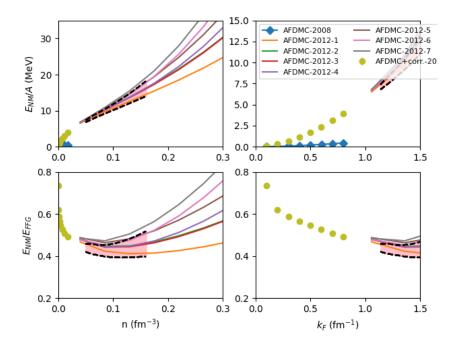


Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

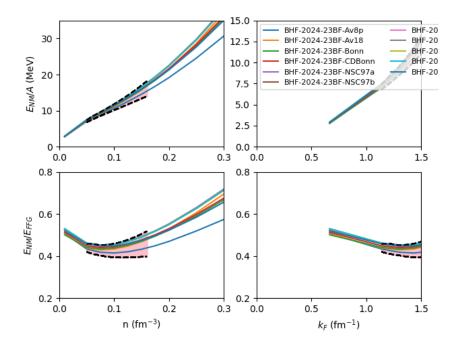


Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

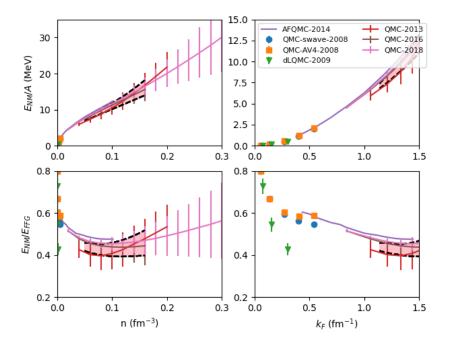


Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

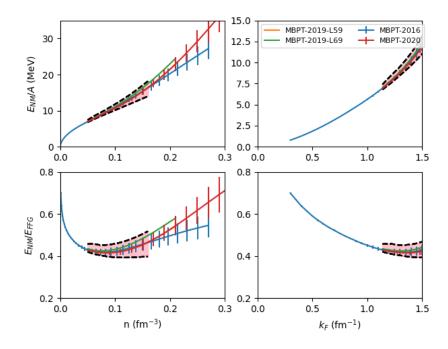


Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.

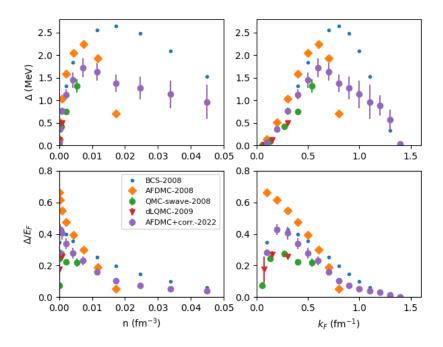


Fig. 6: This figure shows the pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

#### **Attributes:**

#### den

Attribute a set of density points.

## init\_self()

Initialize variables in self.

### matter

Attribute matter str.

#### models

Attribute model.

## nden

Attribute number of points in density.

## print\_outputs()

Method which print outputs on terminal's screen.

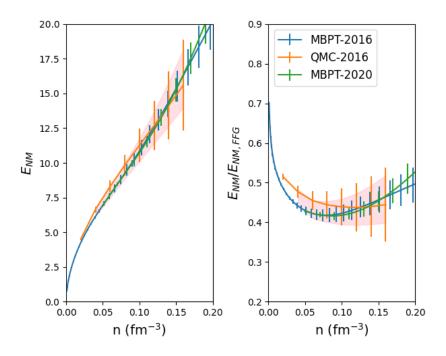


Fig. 7: Uncertainty band in NM obtained from the analysis of different predictions: MBPT-2016, QMC-2016 and MBPT-2020.

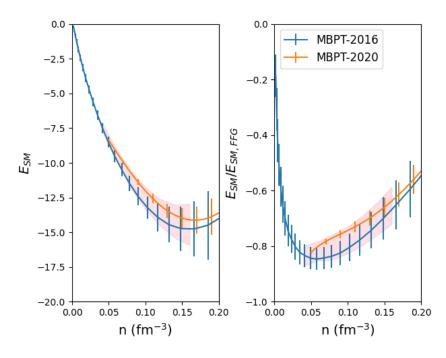


Fig. 8: Uncertainty band in SM obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

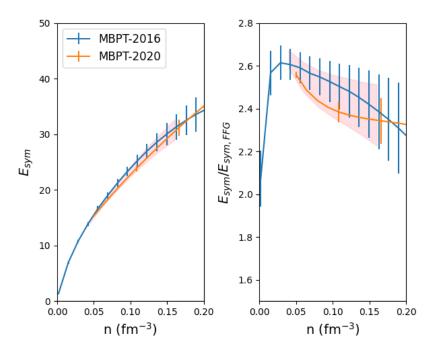


Fig. 9: Uncertainty band for the symmetry energy obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

## 2.3 SetupMicroLP

class nucleardatapy.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-REID-GAP', '1994-BHF-SM-LP-REID-CONT', '1994-BHF-SM-LP-AV14-CONT-0.7'.

#### **Parameters**

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

## **Attributes:**

#### init\_self()

Initialize variables in self.

### model

Attribute model.

### print\_outputs()

Method which print outputs on terminal's screen.

#### nucleardatapy.setup\_micro\_LP.models\_micro\_LP()

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-CONT', '1994-BHF-SM-LP-AV14-CONT-0.7'.

#### Returns

The list of models.

2.3. SetupMicroLP 13

## Return type

list[str].

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy\_plots/plot\_setupLP.py

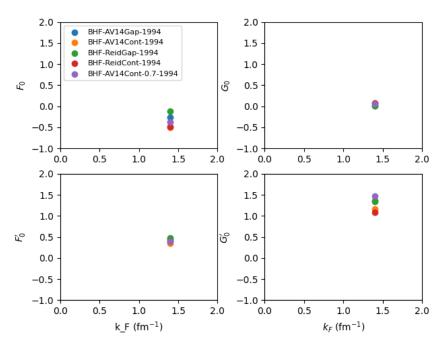


Fig. 10: This figure shows the L=0 Landau parameters in SM for different NN interactions obtained from BHF calculations.

## 2.4 SetupPhenoMatter

class nucleardatapy.setup\_pheno\_matter.SetupPhenoMatter(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', 'FDL', '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:** 

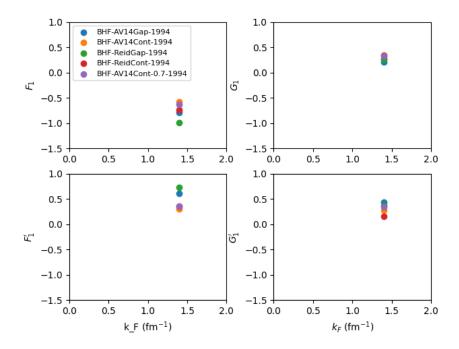


Fig. 11: This figure shows the L=1 Landau parameters in SM for different NN interactions obtained from BHF calculations.

#### Esat

Attribute the NEP.

#### 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\_matter.models\_pheno\_matter()

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\_matter.params\_pheno\_matter(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', 'PK01', 'PKO2', 'PKO3'.

## Return type

list[str].

Here are a set of figures which are produced with the Python sample: /sam-ple/nucleardatapy\_plots/plot\_setupPhenoMatter.py

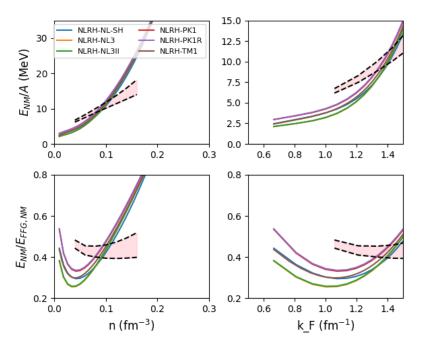


Fig. 12: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree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

## 2.5 SetupHICMatter

class nucleardatapy.setup\_hic\_matter.SetupHICMatter(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\_hic\_matter.constraints\_HIC\_matter()

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' ].

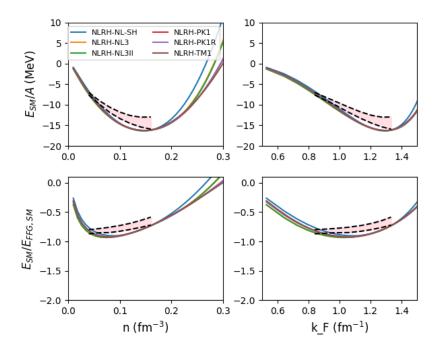


Fig. 13: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic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

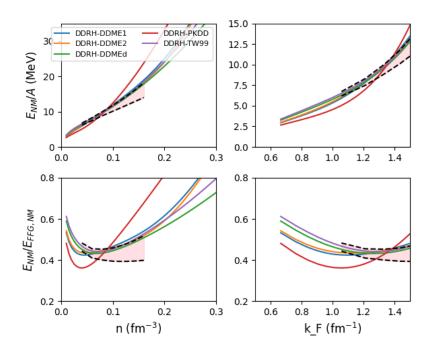


Fig. 14: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

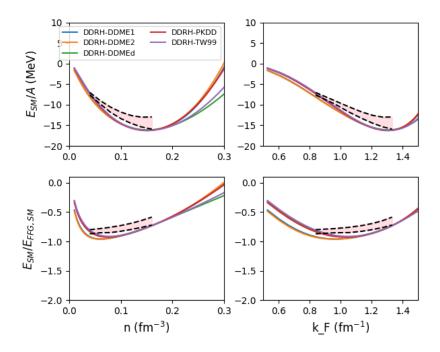


Fig. 15: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

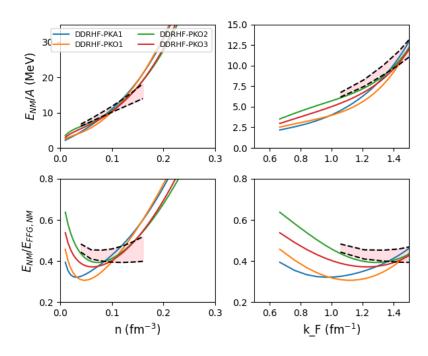


Fig. 16: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

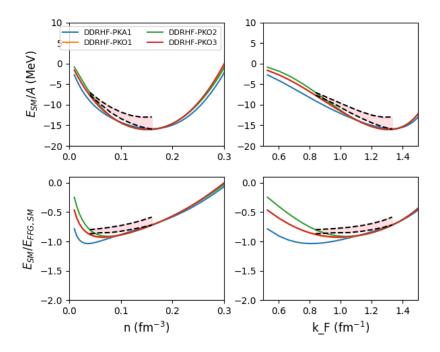


Fig. 17: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

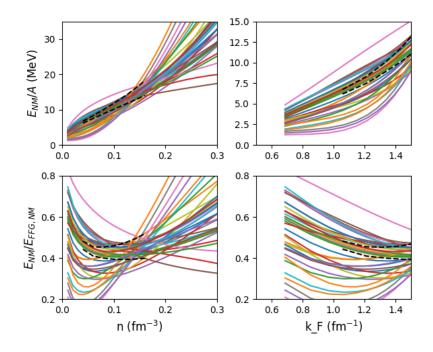


Fig. 18: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

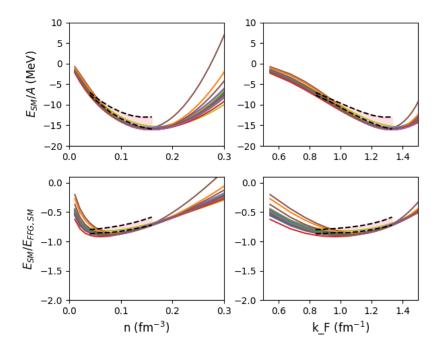


Fig. 19: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

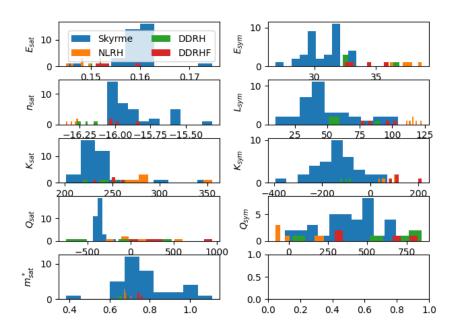


Fig. 20: Distribution of NEP for phenomenological models available in the nucleardatapy toolkit.

#### Returns

The list of constraints.

#### **Return type**

list[str].

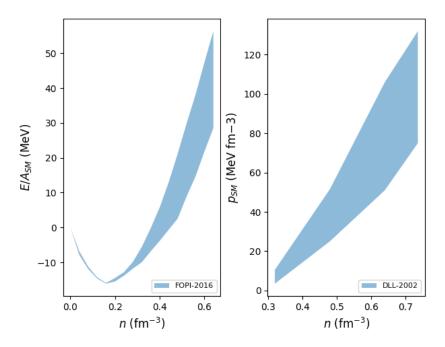


Fig. 21: HIC Experimental constraints for the energy per particle (left) and pressure (right) in SM as a function of the particle density for different analyses available in the *nuda* toolkit.

## 2.6 SetupCrust

class nucleardatapy.setup\_crust.SetupCrust(modcrust='1973-Negele-Vautherin')

Instantiate the properties of the crust for the existing models.

This choice is defined in the variable *crust*.

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

#### **Parameters**

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

### **Attributes:**

A

Attribute A (mass of the nucleus).

N

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

#### N\_bound

Attribute N\_bound (number of bound neutrons).

 $N_g$ 

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.

хp

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].

23 2.6. SetupCrust

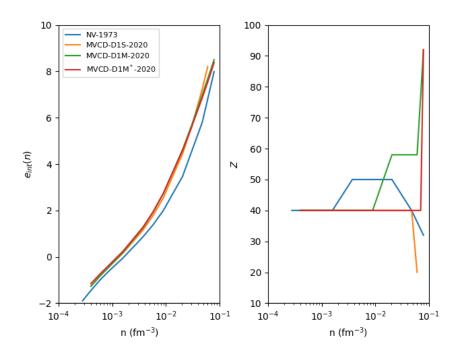


Fig. 22: Properties of the crust as given by the models available in the nuda toolkit.

## 2.7 SetupMassesExp

 $\textbf{class} \ \ \textbf{nucleardatapy.setup\_masses\_exp.SetupMassesExp} (\textit{table='AME'}, \textit{version='2020'})$ 

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: /sam-ple/nucleardatapy\_plots/plot\_setupMassesExp.py

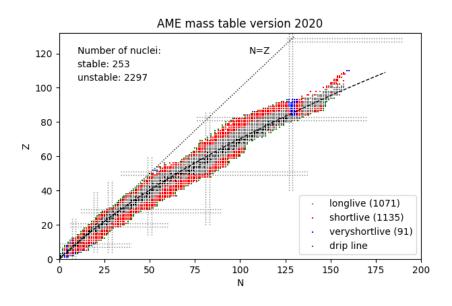


Fig. 23: The nuclear chart based on AME 2020 table. The different colors correspond to the different measured half-times of nuclei.

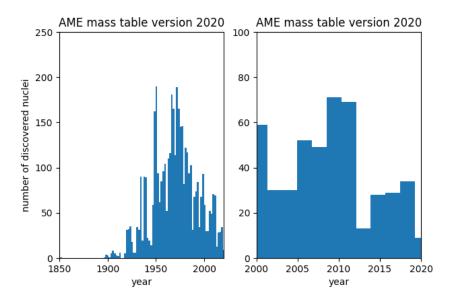


Fig. 24: Histogram showing the distribution of nuclei per discovery year, since the first one discovered in 1851.

## 2.8 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 *ver-sion\_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: /sam-ple/nucleardatapy\_plots/plot\_setupMassesTheory.py

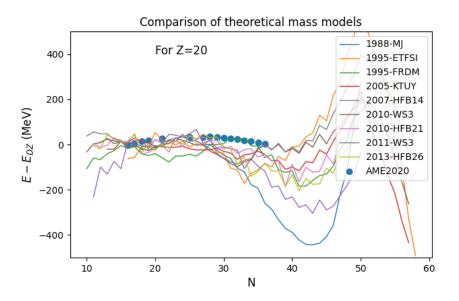


Fig. 25: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 20.

## 2.9 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.

2.9. SetupRadCh 29

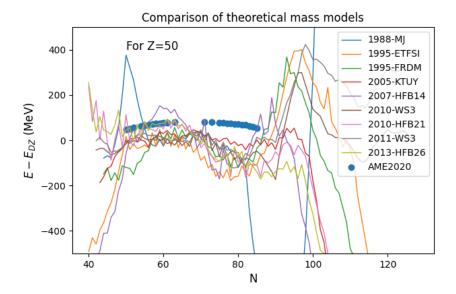


Fig. 26: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 50.

#### nucA

Attribute A (mass of the nucleus).

## nucN

Attribute N (number of neutrons of the nucleus).

### nucRch

Attribue R\_ch (charge radius) in fm.

#### nucRch\_err

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

## nucSymb

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

#### nucZ

Attribute Z (charge of the nucleus).

### print\_outputs()

Method which print outputs on terminal's screen.

### ref

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

## nucleardatapy.setup\_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].

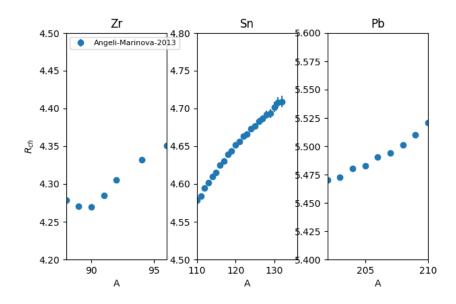


Fig. 27: Charge radii for Zn, Sn, and Pb isotopes and for the models available in the nuda toolkit.

## 2.10 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', '2018-ISGMR-GARG-LATEX'.

#### **Attributes:**

#### E\_unit

Attribute energy unit.

## label

Attribute providing the label the data is references for figures.

#### note

Attribute providing additional notes about the data.

#### nucA

Attribute A (mass of the nucleus).

## nucM12Mm1\_cent

Attribute energy centroid.

#### nucM12Mm1\_errm

Attribute (-) uncertainty in the energy centroid.

2.10. SetupISGMR 31

#### nucM12Mm1\_errp

Attribute (+) uncertainty in the energy centroid.

## nucSymbol

Attribute the symbol of the element.

#### nucZ

Attribute Z (charge of the nucleus).

## print\_outputs()

Method which print outputs on terminal's screen.

## ref

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

#### table

Attribute table.

## nucleardatapy.setup\_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', '2018-ISGMR-GARG-LATEX'.

#### Returns

The list of tables.

#### **Return type**

list[str].

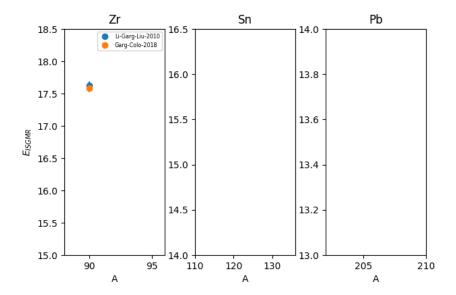


Fig. 28: ISGMR energies available in the nucleardatapy toolkit.

## 2.11 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-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: /sam-ple/nucleardatapy\_plots/plot\_setupEsymLsym.py

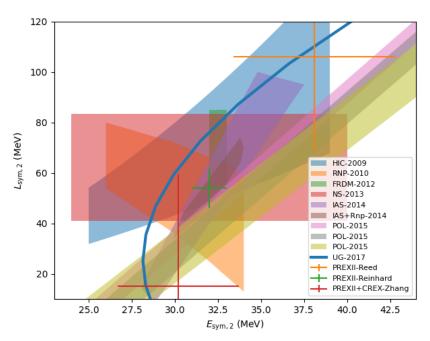


Fig. 29: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nuclear-datapy toolkit.

## **CHAPTER**

## **THREE**

## **INDICES AND TABLES**

- genindex
- modindex
- search

## **PYTHON MODULE INDEX**

#### n

```
nucleardatapy,4
nucleardatapy.setup_crust,22
nucleardatapy.setup_EsymLsym,33
nucleardatapy.setup_hic_matter,17
nucleardatapy.setup_ISGMR,31
nucleardatapy.setup_masses_exp,24
nucleardatapy.setup_masses_theory,28
nucleardatapy.setup_micro_LP,13
nucleardatapy.setup_micro_matter,7
nucleardatapy.setup_micro_matter,7
nucleardatapy.setup_micro_matter_band,8
nucleardatapy.setup_pheno_matter,14
nucleardatapy.setup_rad_ch,29
```

38 Python Module Index

## **INDEX**

A	e2a_int2 (nucleardatapy.setup_crust.SetupCrust at-
A (nucleardatapy.setup_crust.SetupCrust attribute), 22	tribute), 23
alpha (nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute), 33	e2a_int_g (nucleardatapy.setup_crust.SetupCrust at- tribute), 23
unione), 55	e2a_rm (nucleardatapy.setup_crust.SetupCrust at-
C	tribute), 23
constraint (nucleardat-	E_unit (nucleardatapy.setup_ISGMR.SetupISGMR attribute), 31
apy.setup_EsymLsym.SetupEsymLsym at- tribute), 33	Esat (nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 14
<pre>constraints_EsymLsym() (in module nucleardat- apy.setup_EsymLsym), 33</pre>	Esym (nucleardatapy.setup_EsymLsym.SetupEsymLsym
constraints_HIC_matter() (in module nucleardat-	attribute), 33 esym_den (nucleardat-
apy.setup_hic_matter), 17	apy.setup_pheno_matter.SetupPhenoMatter
D	attribute), 15 esym_e2a (nucleardat-
den (nucleardatapy.setup_crust.SetupCrust attribute), 23	any satur phono matter Setun Phono Matter
den (nucleardatapy.setup_micro_matter_band.SetupMicro.	MatterBandattribute), 15
attribute), 11	Esym_err (nucleardat-
den_cgs (nucleardatapy.setup_crust.SetupCrust at- tribute), 23	apy.setup_EsymLsym.SetupEsymLsym at- tribute), 33
<pre>den_g (nucleardatapy.setup_crust.SetupCrust attribute),</pre>	esym_kf (nucleardatapy.setup_pheno_matter.SetupPhenoMatter
diff() (nucleardatapy.setup_masses_theory.SetupMasses	The synt max (nucleardat-
method), 28	apy.setup_EsymLsym.SetupEsymLsym at-
diff_exp() (nucleardat-	tribute), 33
apy.setup_masses_theory.SetupMassesTheory	Esym_min (nucleardat-
method), 28 dist_nbNuc (nucleardat-	apy.setup_EsymLsym.SetupEsymLsym at-
apy.setup_masses_exp.SetupMassesExp at-	tribute), 33
tribute), 24	F
dist_year (nucleardat-	flagI (nucleardatapy.setup_masses_exp.SetupMassesExp
apy.setup_masses_exp.SetupMassesExp at-	attribute), 25
tribute), 24	flagIntern (nucleardat
drip() (nucleardatapy.setup_masses_exp.SetupMassesExp method), 24	apy.setup_masses_exp.SetupMassesExp at-
<pre>drip() (nucleardatapy.setup_masses_theory.SetupMasses</pre>	Theory tribute), 25
method), 28	
E	init_self() (nucleardat-
	apy.setup_hic_matter.SetupHICMatter
e2a_int (nucleardatapy.setup_crust.SetupCrust at- tribute), 23	method), 17

init_s	elf() (nucleardat- apy.setup_masses_theory.SetupMassesTheory	<pre>models_pheno_matter() (in module nucleardat- apy.setup_pheno_matter), 16</pre>
	method), 28	module
init_s		nucleardatapy, 4
_	apy.setup_micro_LP.SetupMicroLP method),	nucleardatapy.setup_crust,22
	13	nucleardatapy.setup_EsymLsym, 33
init_s	elf() (nucleardat-	nucleardatapy.setup_hic_matter, 17
_	apy.setup_micro_matter.SetupMicroMatter	nucleardatapy.setup_ISGMR, 31
	method), 7	nucleardatapy.setup_masses_exp, 24
init_s		nucleardatapy.setup_masses_theory, 28
	apy.setup_micro_matter_band.SetupMicroMatte	
	method), 11	nucleardatapy.setup_micro_matter,7
	<i>"</i>	nucleardatapy.setup_micro_matter_band,8
L		nucleardatapy.setup_pheno_matter, 14
lahel	(nucleardatapy.setup_EsymLsym.SetupEsymLsym	nucleardatapy.setup_rad_ch, 29
IUDCI	attribute), 33	mu_n (nucleardatapy.setup_crust.SetupCrust attribute),
label	(nucleardatapy.setup_ISGMR.SetupISGMR	23
IUDCI	attribute), 31	<pre>mu_p (nucleardatapy.setup_crust.SetupCrust attribute),</pre>
label(	nucleardatapy.setup_masses_exp.SetupMassesExp	
	attribute), 25	N
Tabel(	nucleardatapy.setup_pheno_matter.SetupPhenoMo	M (nucleandatamy setup, emust Setup Court attribute) 22
	attribute), 15	n (nuclearadiapy.setup_crusi.setupCrusi attribute), 22
label	(nucleardatapy.setup_rad_ch.SetupRadCh attribute), 29	tribute), 22
Lsym	(nucleardatapy.setup_EsymLsym.SetupEsymLsym	N_g (nucleardatapy.setup_crust.SetupCrust attribute), 22
	attribute), 33	nbLine (nucleardatapy.setup_masses_exp.SetupMassesExp
Lsym_e		attribute), 25
	apy.setup_EsymLsym.SetupEsymLsym attribute), 33	nbNuc (nucleardatapy.setup_masses_exp.SetupMassesExp attribute), 25
Lsym_m	ax (nucleardat-	${\tt nden}  (nuclear datapy. setup\_micro\_matter\_band. Setup Micro Matter Band)$
	apy.setup_EsymLsym.SetupEsymLsym at-	attribute), 11
	tribute), 33	$\verb nm_cs2  (nuclear datapy.setup\_pheno\_matter.SetupPhenoMatter $
Lsym_m	in (nucleardat-	attribute), 15
	apy.setup_EsymLsym.SetupEsymLsym attribute), 33	nm_den (nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 15
		$\verb nm_e2a  (nuclear datapy.setup\_pheno\_matter.Setup Pheno Matter$
M		attribute), 15
matter	(nucleardatapy.setup_micro_matter_band.SetupMattribute), 11	icToMaire(purleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 15
	nucleardatapy.setup_micro_LP.SetupMicroLP at-	nm_kfn (nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 15
model(	nucleardatapy.setup_micro_matter.SetupMicroMa attribute), 7	ttenm_pre(nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 15
model (	nucleardatapy.setup_pheno_matter.SetupPhenoMo	
model (	attribute), 15	attribute), 33
models	(nucleardatapy.setup_micro_matter_band.SetupM	icPoMatte(pucleardatapy.setup_ISGMR.SetupISGMR at- tribute), 31
	attribute), 11	note (nucleardatapy.setup_masses_exp.SetupMassesExp
models	_crust() (in module nucleardat- apy.setup_crust), 23	attribute), 25
models	_micro_LP() (in module nucleardat-	note (nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 16
modala	<pre>apy.setup_micro_LP), 13 _micro_matter() (in module nucleardat-</pre>	note (nucleardatapy.setup_rad_ch.SetupRadCh at-
mouers	apy.setup_micro_matter), 7	tribute), 29

40 Index

nucA (nucleardatapy.setup_ISGMR.SetupISGMR at- tribute), 31	nucStb1 (nucleardatapy.setup_masses_exp.SetupMassesExp attribute), 25
<pre>nucA (nucleardatapy.setup_masses_exp.SetupMassesExp</pre>	nucSymb (nucleardatapy.setup_masses_exp.SetupMassesExp
attribute), 25	attribute), 25
nucA (nucleardatapy.setup_rad_ch.SetupRadCh attribute), 29	<pre>nucSymb (nucleardatapy.setup_rad_ch.SetupRadCh at- tribute), 30</pre>
nucBE (nucleardatapy.setup_masses_exp.SetupMassesExp attribute), 25	nucSymbol (nucleardatapy.setup_ISGMR.SetupISGMR attribute), 32
nucBE_err (nucleardat-	nucYear (nucleardatapy.setup_masses_exp.SetupMassesExp
apy.setup_masses_exp.SetupMassesExp at-	attribute), 25
tribute), 25	nucZ (nucleardatapy.setup_ISGMR.SetupISGMR at-
$\verb+nucht+ (nuclear datapy.setup\_masses\_exp.SetupMassesExp+$	tribute), 32
attribute), 25	nucZ (nucleardatapy.setup_masses_exp.SetupMassesExp
nucleardatapy	attribute), 25
module, 4	nucZ (nucleardatapy.setup_rad_ch.SetupRadCh at-
nucleardatapy.setup_crust	tribute), 30
module, 22	D
nucleardatapy.setup_EsymLsym	P
module, 33	<pre>param(nucleardatapy.setup_pheno_matter.SetupPhenoMatter</pre>
nucleardatapy.setup_hic_matter	attribute), 16
module, 17	<pre>params_pheno_matter() (in module nucleardat-</pre>
nucleardatapy.setup_ISGMR	apy.setup_pheno_matter), 16
module, 31	<pre>print_outputs() (nucleardat-</pre>
nucleardatapy.setup_masses_exp	apy.setup_crust.SetupCrust method), 23
module, 24	print_outputs() (nucleardat-
nucleardatapy.setup_masses_theory module, 28	apy.setup_EsymLsym.SetupEsymLsym method), 33
<pre>nucleardatapy.setup_micro_LP</pre>	print_outputs() (nucleardat-
module, 13	apy.setup_hic_matter.SetupHICMatter
nucleardatapy.setup_micro_matter	method), 17
module, 7	<pre>print_outputs()</pre>
nucleardatapy.setup_micro_matter_band	apy.setup_ISGMR.SetupISGMR method),
module, 8	32
nucleardatapy.setup_pheno_matter	<pre>print_outputs() (nucleardat-</pre>
module, 14	apy.setup_masses_exp.SetupMassesExp
nucleardatapy.setup_rad_ch	method), 25
module, 29	<pre>print_outputs()</pre>
nucM12Mm1_cent (nucleardat-	apy.setup_masses_theory.SetupMassesTheory
apy.setup_ISGMR.SetupISGMR attribute),	method), 28
31	print_outputs() (nucleardat-
nucM12Mm1_errm (nucleardat-	apy.setup_micro_LP.SetupMicroLP method),
apy.setup_ISGMR.SetupISGMR attribute),	13
31	print_outputs() (nucleardat-
nucM12Mm1_errp (nucleardat-	apy.setup_micro_matter.SetupMicroMatter
apy.setup_ISGMR.SetupISGMR attribute),	method), 7
31	print_outputs() (nucleardat-
nucN (nucleardatapy.setup_masses_exp.SetupMassesExp	apy.setup_micro_matter_band.SetupMicroMatterBand
attribute), 25	method), 11
nucN (nucleardatapy.setup_rad_ch.SetupRadCh at- tribute), 30	print_outputs() (nucleardat-
nucRch (nucleardatapy.setup_rad_ch.SetupRadCh	apy.setup_pheno_matter.SetupPhenoMatter
attribute), 30	method), 16
nucRch_err (nucleardatapy.setup_rad_ch.SetupRadCh	print_outputs() (nucleardat-
attribute), 30	apy.setup_rad_ch.SetupRadCh method), 30
/,	50

Index 41

R	sm_kfn(nucleardatapy.setup_pheno_matter.SetupPhenoMatter		
R_unit (nucleardatapy.setup_rad_ch.SetupRadCh	attribute), 16		
attribute), 29	sm_pre(nucleardatapy.setup_pheno_matter.SetupPhenoMatter		
RadCh_isotopes() (nucleardat-	attribute), 16		
apy.setup_rad_ch.SetupRadCh method), 29	Т		
ref (nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute), 33	table (nucleardatapy.setup_ISGMR.SetupISGMR attribute), 32		
ref (nucleardatapy.setup_ISGMR.SetupISGMR at- tribute), 32	tables_isgmr() (in module nucleardat- apy.setup_ISGMR), 32		
ref (nucleardatapy.setup_masses_exp.SetupMassesExp attribute), 25	tables_masses_exp() (in module nucleardat- apy.setup_masses_exp), 26		
	rtables_masses_theory() (in module nucleardat-		
attribute), 16	apy.setup_masses_theory), 28		
ref (nucleardatapy.setup_rad_ch.SetupRadCh attribute), 30	tables_rad_ch() (in module nucleardat- apy.setup_rad_ch), 30		
RWS (nucleardatapy.setup_crust.SetupCrust attribute), 22	V		
S	•		
select() (nucleardat-	versions_masses_exp() (in module nucleardat- apy.setup_masses_exp), 26		
apy.setup_masses_exp.SetupMassesExp method), 25	X		
select_year() (nucleardat-	xn (nucleardatapy.setup_crust.SetupCrust attribute), 23		
apy.setup_masses_exp.SetupMassesExp method), 26	xn_bound (nucleardatapy.setup_crust.SetupCrust at- tribute), 23		
SetupCrust (class in nucleardatapy.setup_crust), 22	xp (nucleardatapy.setup_crust.SetupCrust attribute), 23		
SetupEsymLsym (class in nucleardat-	xpn_bound (nucleardatapy.setup_crust.SetupCrust at-		
<pre>apy.setup_EsymLsym), 33 SetupHICMatter (class in nucleardat-</pre>	tribute), 23		
SetupHICMatter (class in nucleardat- apy.setup_hic_matter), 17	Z		
SetupISGMR (class in nucleardatapy.setup_ISGMR), 31	_		
SetupMassesExp (class in nucleardat-	Z (nucleardatapy.setup_crust.SetupCrust attribute), 23 Zmax (nucleardatapy.setup_masses_exp.SetupMassesExp		
apy.setup_masses_exp), 24	attribute), 24		
SetupMassesTheory (class in nucleardat-			
apy.setup_masses_theory), 28			
SetupMicroLP (class in nucleardat-			
apy.setup_micro_LP), 13 SetupMicroMatter (class in nucleardat-			
apy.setup_micro_matter), 7			
SetupMicroMatterBand (class in nucleardat-			
apy.setup_micro_matter_band), 8			
SetupPhenoMatter (class in nucleardat-			
apy.setup_pheno_matter), 14 SetupPadCh (class in nucleardatany setup rad_ch) 29			
SetupRadCh (class in nucleardatapy.setup_rad_ch), 29 sm_cs2 (nucleardatapy.setup_pheno_matter.SetupPhenoMatter			
attribute), 16			
sm_den(nucleardatapy.setup_pheno_matter.SetupPhenoMatter			
attribute), 16			
sm_e2a (nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute), 16			
<pre>sm_gap (nucleardatapy.setup_pheno_matter.SetupPhenoMatter</pre>			
<pre>sm_kf (nucleardatapy.setup_pheno_matter.SetupPhenoMattribute), 16</pre>	tter		

42 Index