# nucleardatapy

Release 0.1

Jérôme Margueron, IRL NPA, USA

## **CONTENTS**

1	Cont	ents	3			
	1.1	Usage	3			
	1.2	API	4			
	1.3	Miscelaneous	4			
2	Com	plement	7			
	2.1	SetupEOSFFG	7			
	2.2	SetupEOSMicro	9			
	2.3	SetupEOSMicroBand	14			
	2.4	SetupEOSMicroLP	15			
	2.5	SetupEOSPheno	17			
	2.6	SetupEOSHIC	20			
	2.7	SetupEOSEsym	25			
	2.8	SetupNucBEExp	26			
	2.9	SetupNucBETheo	30			
	2.10	SetupNucRchExp	32			
	2.11	SetupNucISGMRExp	34			
	2.12	SetupCrust	36			
	2.13	SetupAstroMasses	37			
	2.14	SetupAstroMtot	40			
	2.15	SetupAstroMtov	41			
	2.16	SetupAstroGW	42			
	2.17	SetupCorEsymLsym	43			
	2.17	Scupeoilsymesym	73			
3	Indic	ees and tables	47			
Рy	Python Module Index					
In	dex		51			

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 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.



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

#### **1** 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 SetupEOSFFG

class nucleardatapy.setup\_eos\_ffg.SetupEOSFFG(den, delta)

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

#### **Parameters**

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

#### **Attributes:**

#### delta

Attribute isospin parameter

den

Attribute isoscalar density

#### den\_n

Attribute neutron density

### den\_p

Attribute proton density

#### e2a\_int

Attribute FFG energy per particle

#### e2v\_int

Attribute FFG energy per unit volum

## eF\_n

Attribute neutron Fermi energy

## eF\_p

Attribute proton Fermi energy

#### esym

Attribute FFG symmetry energy

## esym2

Attribute FFG quadratic contribution to the symmetry energy

#### esym4

Attribute FFG quartic contribution to the symmetry energy

kf

Attribute Fermi momentum

kf\_n

Attribute neutron Fermi momentum

kf\_p

Attribute proton Fermi momentum

label

Attribute providing the label the data is references for figures.

note

Attribute providing additional notes about the data.

pre

Attribute FFG pressure

print\_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup\_eos\_ffg.den(kf)

Density as a function of the Fermi momentum.

#### **Parameters**

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

nucleardatapy.setup\_eos\_ffg.den\_n(kf\_n)

Neutron density as a function of the neutron Fermi momentum.

#### **Parameters**

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

nucleardatapy.setup\_eos\_ffg.eF\_n(kf\_n)

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

#### **Parameters**

**kf\_n** (float or numpy vector of real numbers.) — neutron Fermi momentum.

nucleardatapy.setup\_eos\_ffg.effg(kf)

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

#### **Parameters**

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

nucleardatapy.setup\_eos\_ffg.effg\_NM(kf\_n)

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

#### **Parameters**

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

nucleardatapy.setup\_eos\_ffg.effg\_SM(kf)

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

#### **Parameters**

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

nucleardatapy.setup\_eos\_ffg.esymffg(kf)

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

#### **Parameters**

 ${f kf}$  (float or numpy vector of real numbers.) – Fermi momentum.

nucleardatapy.setup\_eos\_ffg.kf(den)

Fermi momentum as a function of the density.

#### **Parameters**

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

nucleardatapy.setup\_eos\_ffg.kf\_n(den\_n)

Neutron Fermi momentum as a function of the neutron density.

#### **Parameters**

den\_n (float or numpy vector of real numbers.) - neutron density.

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

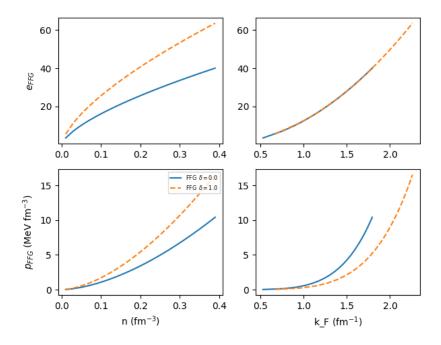


Fig. 1: This figure shows the free Fermi gas energy (top) and pressure (bottom) in symmetric matter (SM) (Blue solid line) and neutron matter (NM) (orange dashed line) as function of the particle density (left) and Fermi momentum (right).

## 2.2 SetupEOSMicro

class nucleardatapy.setup\_eos\_micro.SetupEOSMicro(model='1998-VAR-AM-APR', var1=array([0.01,

0.01393939, 0.01787879, 0.02181818, 0.02575758, 0.02969697, 0.03363636, 0.03757576, 0.04151515, 0.04545455, 0.04939394, 0.053333333, 0.05727273, 0.06121212, 0.06515152, 0.06909091, 0.0730303, 0.0769697, 0.08090909, 0.08484848, 0.08878788, 0.09272727, 0.096666667, 0.10060606, 0.10454545. 0.10848485, 0.11242424, 0.11636364, 0.12030303, 0.12424242, 0.12818182, 0.13212121, 0.13606061, 0.14, 0.14393939, 0.14787879, 0.15181818, 0.15575758, 0.15969697, 0.16363636, 0.16757576, 0.17151515, 0.17545455, 0.17939394, 0.18333333, 0.18727273, 0.19121212, 0.19515152, 0.19909091, 0.2030303, 0.2069697, 0.21090909, 0.21484848, 0.21878788, 0.22272727, 0.22666667, 0.23060606, 0.23454545, 0.23848485, 0.24242424, 0.24636364, 0.25030303, 0.25424242, 0.25818182, 0.26212121, 0.26606061, 0.27, 0.27393939, 0.27787879, 0.28181818, 0.28575758, 0.28969697, 0.29363636, 0.29757576, 0.30151515, 0.30545455, 0.30939394, 0.31333333, 0.31727273, 0.32121212, 0.32515152, 0.32909091, 0.3330303, 0.3369697, 0.34090909, 0.34484848, 0.34878788, 0.35272727, 0.35666667, 0.36060606, 0.36454545, 0.36848485, 0.37242424, 0.37636364, 0.38030303, 0.38424242, 0.38818182, 0.39212121, 0.39606061, 0.41), var2=0.0)

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

This choice is defined in *model*, which can chosen among the following choices: '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-iAPR', '2006-BHF-AM\*', '2008-BCS-NM', '2008-AFDMC-NM', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2012-AFDMC-NM-1', '2012-AFDMC-NM-2', '2012-AFDMC-NM-3', '2012-AFDMC-NM-4', '2012-AFDMC-NM-5', '2012-AFDMC-NM-6', '2012-AFDMC-NM-7', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '2022-AFDMC-NM', '2024-BHF-AM-2BF-AV8p', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97d', '2024-BHF-AM-23BF-AV8p', '2024-BHF-AM-23BF-NSC97a', '202

#### **Parameters**

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

### **Attributes:**

#### init\_self()

Initialize variables in self.

#### model

Attribute model.

#### print\_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup\_eos\_micro.eos\_micro\_models()

Return a list with the name of the models available in this toolkit and print them all on the prompt. These models are the following ones: '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-APRfit', '2006-BHF-AM\*', '2008-BCS-NM', '2008-AFDMC-NM', '2012-AFDMC-NM-1', '2012-AFDMC-N NM-2', '2012-AFDMC-NM-3', '2012-AFDMC-NM-4', '2012-AFDMC-NM-5', '2012-AFDMC-NM-6', '2012-AFDMC-NM-7', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '2022-AFDMC-NM', '2024-NLEFT-AM', '2024-BHF-AM-2BF-Av8p', '2024-BHF-AM-2BF-Av18', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-CDBONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97c', '2024-BHF-AM-2BF-NSC97d', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-23BF-Av8p', '2024-BHF-AM-23BF-Av18', '2024-BHF-AM-2025-BHF-AM-2025-BHF-AW18', '2024-BHF-AM-2025-BHF-AW18', '2024-BHF-AM-2025-BHF-AW18', '2024-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-2025-BHF-AM-202 AM-23BF-BONN', '2024-BHF-AM-23BF-CDBONN', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97d', '2024-BHF-AM-23BF-NSC97e', '2024-BHF-AM-23BF-NSC97f', '2024-BHF-AM-23BF-SSCV14', '2024-BHF-AM-23BFmicro-Av18', '2024-BHF-AM-23BFmicro-BONNB', '2024-BHF-AM-23BFmicro-NSC93' :return: The list of models. :rtype: list[str].

nucleardatapy.setup\_eos\_micro.eos\_micro\_models\_group\_NM(group)

nucleardatapy.setup\_eos\_micro.eos\_micro\_models\_group\_SM(group)

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

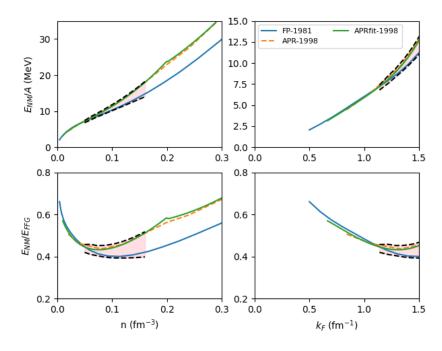


Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the variational models available in the nucleardatapy toolkit.

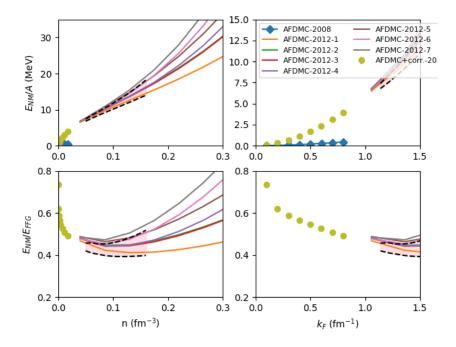


Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

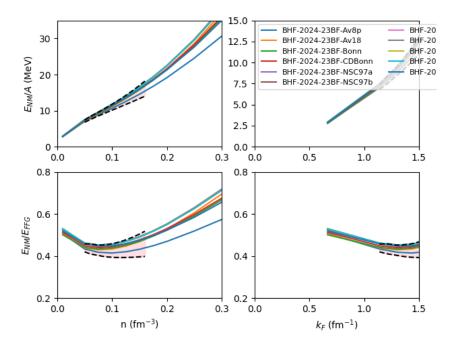


Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

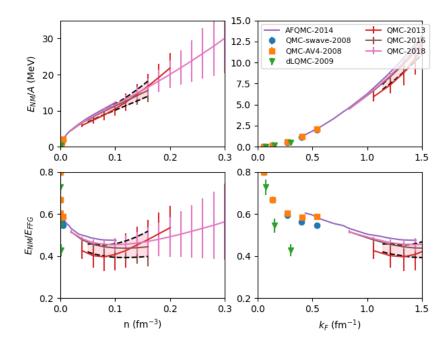


Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

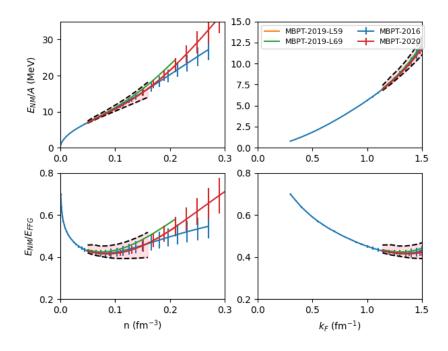


Fig. 6: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.

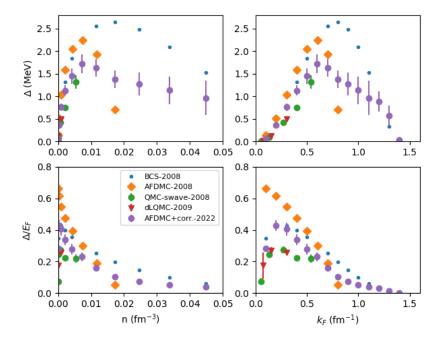


Fig. 7: This figure shows the pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

## 2.3 SetupEOSMicroBand

class nucleardatapy.setup\_eos\_micro\_band.SetupEOSMicroBand( $models=['2016-MBPT-AM'], nden=10, ne=200, den=None, matter='NM', e2a\_min=-20.0, e2a\_max=50.0$ )

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

## **Parameters**

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

#### **Attributes:**

#### den

Attribute a set of density points.

#### init\_self()

Initialize variables in self.

#### matter

Attribute matter str.

#### models

Attribute model.

#### nden

Attribute number of points in density.

#### print\_outputs()

Method which print outputs on terminal's screen.

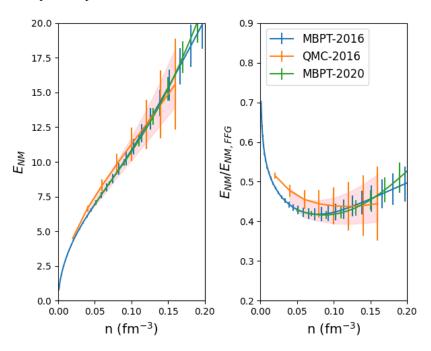


Fig. 8: Uncertainty band in NM obtained from the analysis of different predictions: MBPT-2016, QMC-2016 and MBPT-2020.

## 2.4 SetupEOSMicroLP

class nucleardatapy.setup\_eos\_micro\_lp.SetupEOSMicroLP(model='1994-BHF-SM-LP-AV14-GAP')

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

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

#### **Parameters**

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

#### **Attributes:**

#### init\_self()

Initialize variables in self.

#### model

Attribute model.

## print\_outputs()

Method which print outputs on terminal's screen.

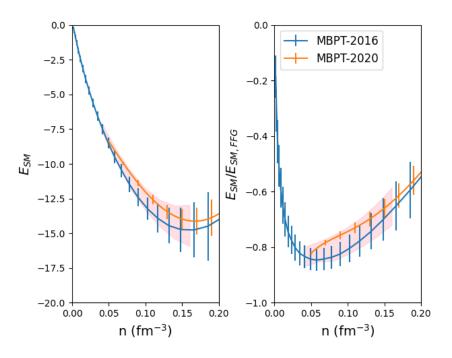


Fig. 9: Uncertainty band in SM obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

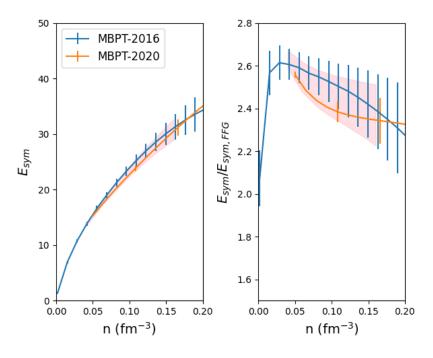


Fig. 10: Uncertainty band for the symmetry energy obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

nucleardatapy.setup\_eos\_micro\_lp.eos\_micro\_LP\_models()

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

#### Returns

The list of models.

#### Return type

list[str].

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

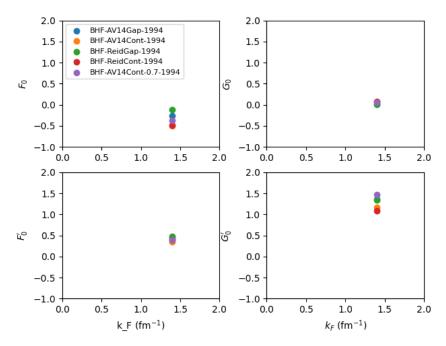


Fig. 11: This figure shows the L=0 Landau parameters in SM for different NN interactions obtained from BHF calculations.

## 2.5 SetupEOSPheno

class nucleardatapy.setup\_eos\_pheno.SetupEOSPheno(model='Skyrme', param='SLY5')

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

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

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

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

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

#### **Parameters**

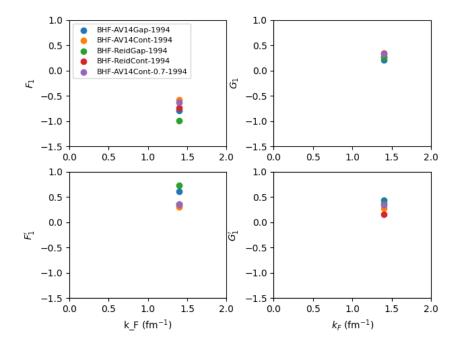


Fig. 12: This figure shows the L=1 Landau parameters in SM for different NN interactions obtained from BHF calculations.

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

## **Attributes:**

#### 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\_eos\_pheno.eos\_pheno\_models()

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

#### Returns

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

#### **Return type**

list[str].

## $\verb|nucleardatapy.setup_eos_pheno.eos_pheno_params| (model)$

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

#### **Parameters**

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

#### Returns

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

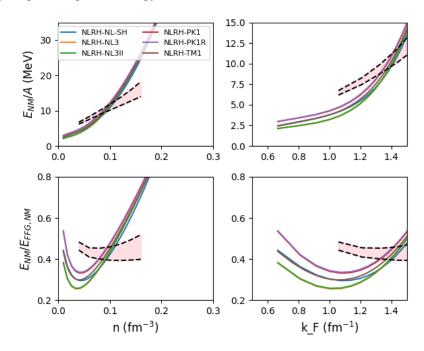


Fig. 13: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree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

## 2.6 SetupEOSHIC

class nucleardatapy.setup\_eos\_hic.SetupEOSHIC(constraint='DLL-2002')

Instantiate the constraints on the EOS from HIC.

This choice is defined in the variable constraint.

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

#### Parameters

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

**Attributes:** 

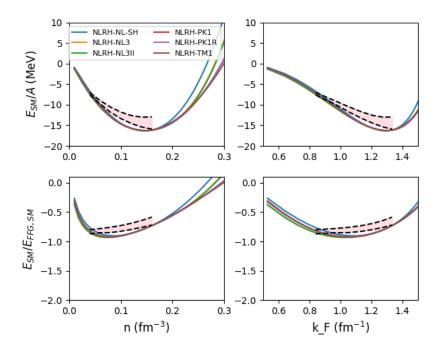


Fig. 14: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic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

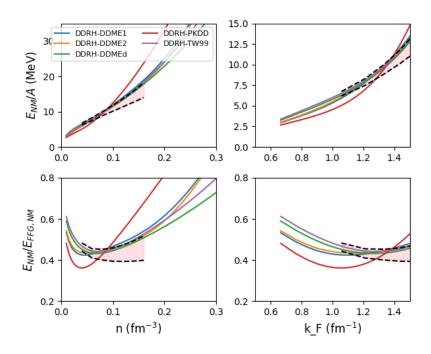


Fig. 15: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

2.6. SetupEOSHIC 21

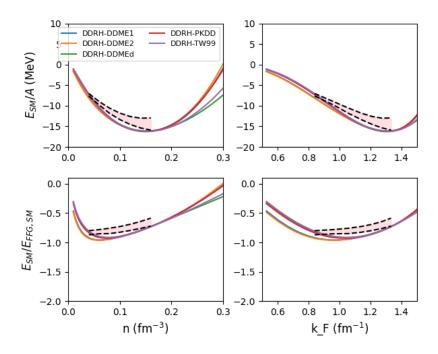


Fig. 16: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

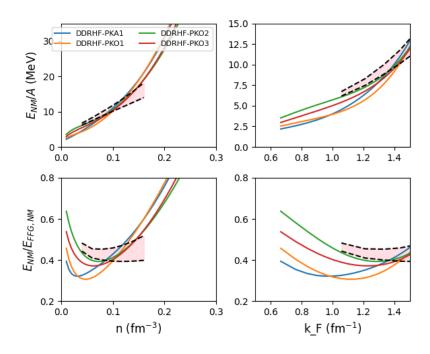


Fig. 17: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

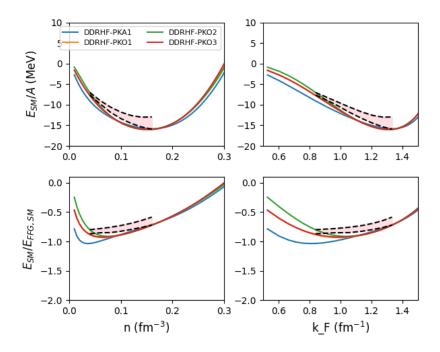


Fig. 18: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

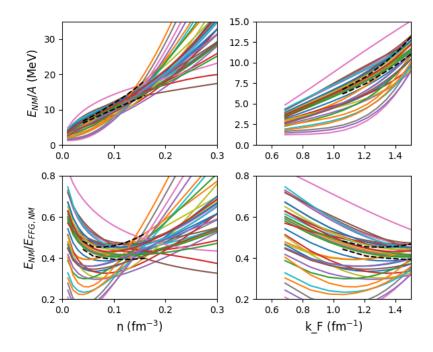


Fig. 19: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

2.6. SetupEOSHIC 23

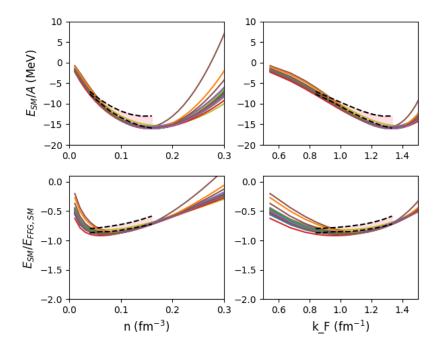


Fig. 20: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

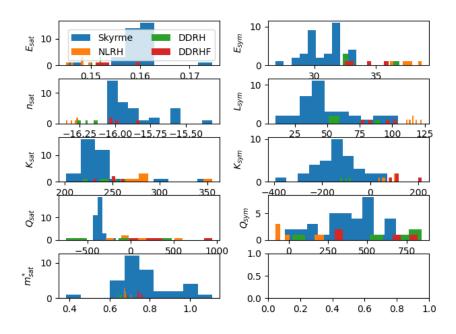


Fig. 21: Distribution of NEP for phenomenological models available in the nucleardatapy toolkit.

## init\_self()

Initialize variables in self.

## print\_outputs()

Method which print outputs on terminal's screen.

#### nucleardatapy.setup\_eos\_hic.eos\_hic\_constraints()

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

#### **Returns**

The list of constraints.

## Return type

list[str].

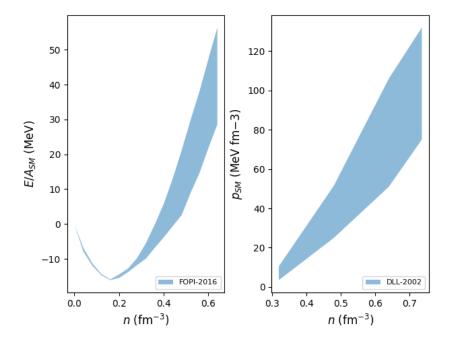


Fig. 22: HIC Experimental constraints for the energy per particle (left) and pressure (right) in SM as a function of the particle density for different analyses available in the *nuda* toolkit.

## 2.7 SetupEOSEsym

class nucleardatapy.setup\_eos\_esym.SetupEOSEsym(constraint='2014-IAS', Ksym=0.0)

Instantiate the values of Esym and Lsym from the constraint.

#### **Parameters**

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

#### Returns

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

#### constraint

Attribute the constraint

#### esym\_e2a\_max

Attribute the maximal symmetry energy

## esym\_e2a\_min

Attribute the minimal symmetry energy

#### print\_outputs()

Method which print outputs on terminal's screen.

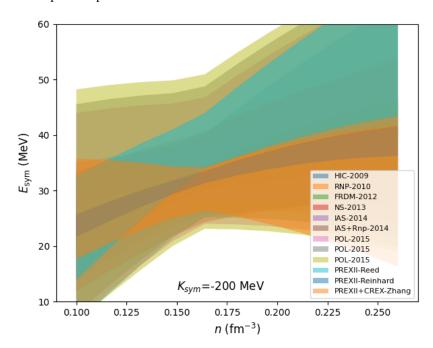


Fig. 23: Uncertainty band for Esym as a function of the density for Ksym=-200 MeV.

## 2.8 SetupNucBEExp

class nucleardatapy.setup\_nuc\_be\_exp.SetupNucBEExp(table='AME', version='2020')

Instantiate the experimental nuclear masses from AME mass table.

This choice is defined in the variables table and version.

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

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

## Parameters

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

#### **Attributes:**

#### **Zmax**

maximum charge of nuclei present in the table.

## Type

Attribute Zmax

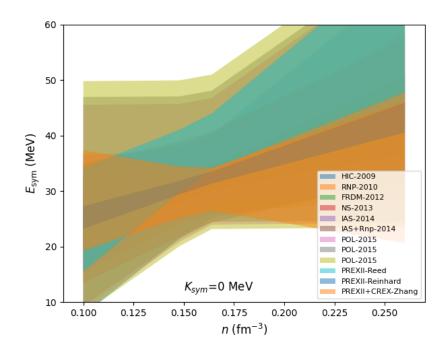


Fig. 24: Uncertainty band for Esym as a function of the density for Ksym=0 MeV.

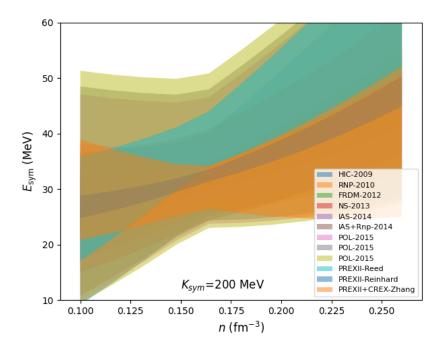


Fig. 25: Uncertainty band for Esym as a function of the density for Ksym=200 MeV.

#### dist\_nbNuc

attribute number of nuclei discovered per year

#### dist\_year

attribute distribution of years

#### drip(Zmax=95)

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

#### **Parameters**

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

#### **Attributes:**

#### flagI

Attribute I.

### flagInterp

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

#### label

Attribute providing the label the data is references for figures.

#### nbLine

Attribute with the number of line in the file.

#### nbNuc

Attribute with the number of nuclei read in the file.

### note

Attribute providing additional notes about the data.

#### nucA

Attribute A (mass of the nucleus).

#### nucBE

Attribute BE (Binding Energy) of the nucleus.

#### nucBE\_err

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

#### nucHT

Attribute HT (half-Time) of the nucleus.

#### nucN

Attribute N (number of neutrons of the nucleus).

#### nucStb1

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

#### nucSymb

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

## nucYear

Attribute year of the discovery of the nucleus.

#### nucZ

Attribute Z (charge of the nucleus).

#### print\_outputs()

Method which print outputs on terminal's screen.

#### ref

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

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

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

#### **Parameters**

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

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

#### **Attributes:**

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

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

## **Parameters**

- year\_min
- year\_max
- **state** (*str*, *optional*. *Default* '*gs*'.) select the kind of state. If state='gs', select nuclei measured in their ground state.

#### **Attributes:**

## nucleardatapy.setup\_nuc\_be\_exp.nuc\_be\_exp\_tables()

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

#### Returns

The list of tables.

## Return type

list[str].

#### nucleardatapy.setup\_nuc\_be\_exp.nuc\_be\_exp\_versions(table)

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

#### **Parameters**

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

## Returns

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

## Return type

list[str].

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

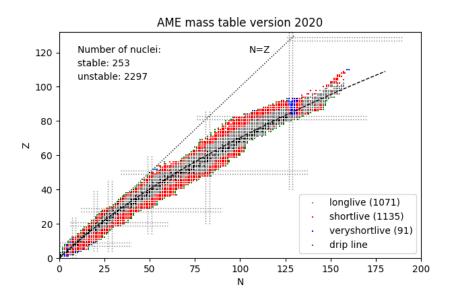


Fig. 26: The nuclear chart based on AME 2020 table. The different colors correspond to the different measured half-times of nuclei.

## 2.9 SetupNucBETheo

class nucleardatapy.setup\_nuc\_be\_theo.SetupNucBETheo(table='1995-DZ')

Instantiate the theory nuclear masses.

This choice is defined in the variable table.

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

#### **Parameters**

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

#### **Attributes:**

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

Method calculates the difference between a given mass model and table\_ref.

## **Parameters**

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

## **Attributes:**

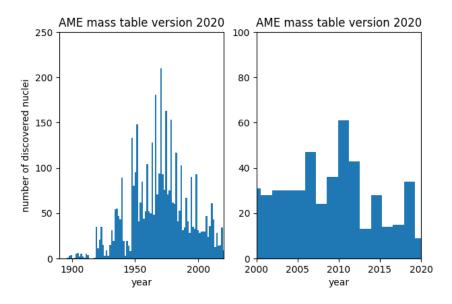


Fig. 27: Histogram showing the distribution of nuclei per discovery year, since the first one discovered in 1851.

## diff\_exp(table\_exp, version\_exp, Zref=50)

Method calculates the difference between a given experimental mass (identified by *table\_exp* and *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\_nuc\_be\_theo.nuc\_be\_theo\_tables()

Return a list of the tables available in this toolkit for the masses predicted by theoretical approaches and print them all on the prompt. These tables are the following ones: [ '1988-MJ', '1995-DZ', '1995-ETFSI', '1995-FRDM', '2005-KTUY', '2007-HFB14', '2010-WS3', '2010-HFB21', '2011-WS3', '2013-HFB22', '2013-HFB23', '2013-HFB24', '2013-HFB25', '2013-HFB26', '2021-BSkG1', '2022-BSkG2', '2023-BSkG3', '2024-BSkG4']

#### 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\_setupNucBETheo.py

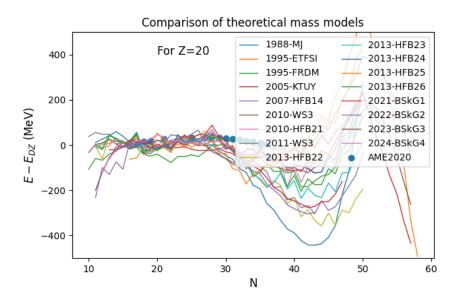


Fig. 28: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 20.

## 2.10 SetupNucRchExp

class nucleardatapy.setup\_nuc\_rch\_exp.SetupNucRchExp(table='2013-Angeli')

Instantiate the object with charge radii choosen from a table.

This choice is defined in the variable *table*.

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

#### **Parameters**

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

#### **Attributes:**

## R\_unit

Attribute radius unit.

### Rch\_isotopes(Zref=50)

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

#### label

Attribute providing the label the data is references for figures.

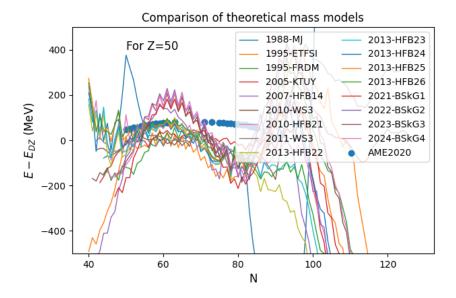


Fig. 29: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 50.

#### note

Attribute providing additional notes about the data.

# nucA

Attribute A (mass of the nucleus).

# nucN

Attribute N (number of neutrons of the nucleus).

# nucRch

Attribue R\_ch (charge radius) in fm.

# nucRch\_err

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

# nucSymb

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

# nucZ

Attribute Z (charge of the nucleus).

# print\_outputs()

Method which print outputs on terminal's screen.

## ref

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

# nucleardatapy.setup\_nuc\_rch\_exp.nuc\_rch\_exp\_tables()

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

#### Returns

The list of tables.

# **Return type**

list[str].

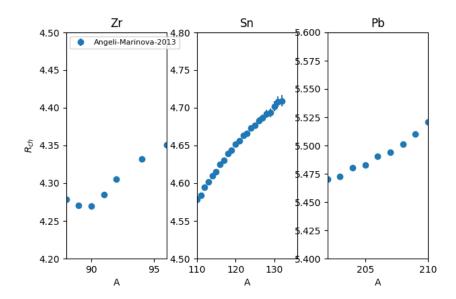


Fig. 30: Charge radii for Zn, Sn, and Pb isotopes and for the models available in the nuda toolkit.

# 2.11 SetupNuclSGMRExp

class nucleardatapy.setup\_nuc\_isgmr\_exp.SetupNucISGMRExp(table='2018-ISGMR-GARG')

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

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

## **Parameters**

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

# **Attributes:**

# E\_unit

Attribute energy unit.

## label

Attribute providing the label the data is references for figures.

#### note

Attribute providing additional notes about the data.

### nucA

Attribute A (mass of the nucleus).

#### nucM12Mm1\_cent

Attribute energy centroid.

# nucM12Mm1\_errm

Attribute (-) uncertainty in the energy centroid.

## nucM12Mm1\_errp

Attribute (+) uncertainty in the energy centroid.

# nucSymbol

Attribute the symbol of the element.

# nucZ

Attribute Z (charge of the nucleus).

# print\_outputs()

Method which print outputs on terminal's screen.

# ref

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

# table

Attribute table.

# nucleardatapy.setup\_nuc\_isgmr\_exp.nuc\_isgmr\_exp\_tables()

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

#### Returns

The list of tables.

# Return type

list[str].

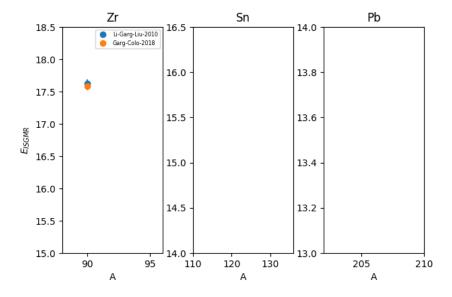


Fig. 31: Experimental ISGMR energies available in the nucleardatapy toolkit.

# 2.12 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:
      Α
           Attribute A (total number of nucleons of the WS cell).
      A_bound
           Attribute A_bound (mass of the cluster).
           Attribute the asymmetry parameter for bound particles.
      N
           Attribute N (total number of neutrons of the WS cell).
      N_bound
           Attribute N_bound (number of neutrons in the cluster).
      N_g
           Attribute N_g (number of neutrons in the gas).
      RWS
           Attribute the radius of the WS cell (in fm).
      Rcl
           Attribute the radius of the cluster (in fm).
      Z
           Attribute Z (total number of protons of the WS cell).
      Z_bound
           Attribute Z_bound (charge of the cluster).
      den
           Attribute the density of the system (in fm^-3).
      den_cgs
           Attribute the density of the system (in cm^-3).
      den_g
           Attribute the approximate density of neutron in the gas (in fm-3).
      e2a_int
           Attribute the internal energy (in MeV).
      e2a_int2
```

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

### e2a\_int\_g

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

## e2a\_rm

Attribute the rest mass energy (in MeV).

#### mu\_n

Attribute the neutron chemical potential (in MeV).

#### mu\_p

Attribute the proton chemical potential (in MeV).

#### n\_g

Attribute n\_g (neutron density in the gas).

# print\_outputs()

Method which print outputs on terminal's screen.

xn

Attribute the fraction of neutrons.

#### xn\_bound

Attribute the fraction of bound neutrons.

хp

Attribute the fraction of protons.

### xp\_bound

Attribute the fraction of bound protons.

# xpn\_bound

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

```
nucleardatapy.setup_crust.models_crust()
```

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

# Returns

The list of tables.

# Return type

list[str].

# 2.13 SetupAstroMasses

class nucleardatapy.setup\_astro\_masses.SetupAstroMasses(source='J1614-2230', obs=1)

Instantiate the observational mass for a given source and obs.

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

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

obs depends on the chosen source.

## **Parameters**

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

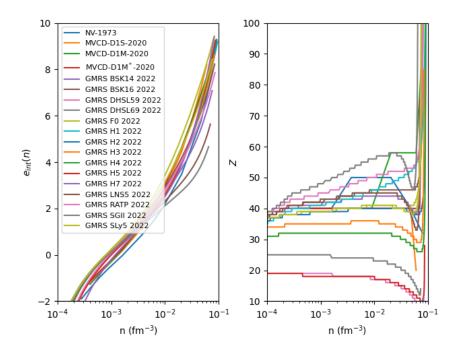


Fig. 32: Properties of the crust as given by the models available in the nuda toolkit.

#### **Attributes:**

#### label

Attribute providing the label the data is references for figures.

#### latexCite

Attribute latexCite.

#### mass

Attribute the observational mass of the source.

### note

Attribute providing additional notes about the observation.

# print\_outputs()

Method which print outputs on terminal's screen.

## ref

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

# sig\_do

Attribute the negative uncertainty.

# sig\_up

Attribute the positive uncertainty.

# class nucleardatapy.setup\_astro\_masses.SetupAstroMassesAverage(source='J1614-2230')

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

This choice is defined in the variable *source*.

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

## **Parameters**

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

#### **Attributes:**

# print\_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup\_astro\_masses.astro\_masses()

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

#### **Returns**

The list of sources.

# Return type

list[str].

nucleardatapy.setup\_astro\_masses.astro\_masses\_source(source)

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

#### **Parameters**

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

#### Returns

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

# **Return type**

list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy\_plots/plot\_setupAstroMasses.py

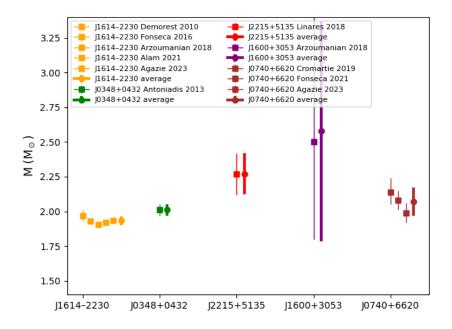


Fig. 33: The masses measured for massive neutron stars is radio-astronomy. The different colors correspond to the different sources.

# 2.14 SetupAstroMtot

```
class nucleardatapy.setup_astro_mtot.SetupAstroMtot(source='GW170817', hyp=1)
```

Instantiate the total mass for a given source and hyptheses.

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

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

hyp depends on the chosen hypotheses.

#### **Parameters**

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

#### **Attributes:**

#### label

Attribute providing the label the data is references for figures.

# latexCite

Attribute latexCite.

#### mtot

Attribute the observational mass of the source.

#### note

Attribute providing additional notes about the observation.

## print\_outputs()

Method which print outputs on terminal's screen.

# ref

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

#### sig\_do

Attribute the negative uncertainty.

## sig\_up

Attribute the positive uncertainty.

# class nucleardatapy.setup\_astro\_mtot.SetupAstroMtotAverage(source='GW170817')

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

This choice is defined in the variable source.

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

#### **Parameters**

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

# **Attributes:**

# print\_outputs()

Method which print outputs on terminal's screen.

# nucleardatapy.setup\_astro\_mtot.astro\_mtot()

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

# Returns

The list of sources.

# Return type

list[str].

nucleardatapy.setup\_astro\_mtot.astro\_mtot\_source(source)

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

#### **Parameters**

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

### Returns

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

### **Return type**

list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy\_plots/plot\_setupAstroMtot.py

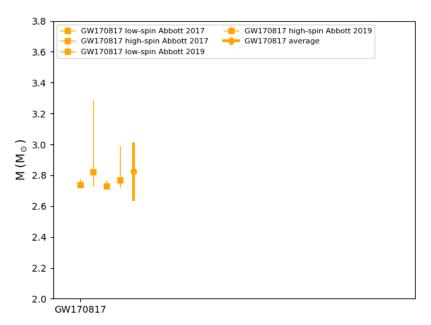


Fig. 34: The total mass measured fo binary neutron star mergers. The different colors correspond to the different sources.

# 2.15 SetupAstroMtov

```
class nucleardatapy.setup_astro_mtov.SetupAstroMtov(sources\_do=array(['J1614-2230'], dtype='<U10'), sources\_up=array(['GW170817'], dtype='<U8'))
```

Instantiate the observational mass for a given source and obs.

This choice is defined in the variable source.

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

## **Parameters**

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

**Attributes:** 

# print\_outputs()

Method which print outputs on terminal's screen.

Here is a figure which is produced with the Python sample: /sample/nucleardatapy\_plots/plot\_setupAstroMtov.py

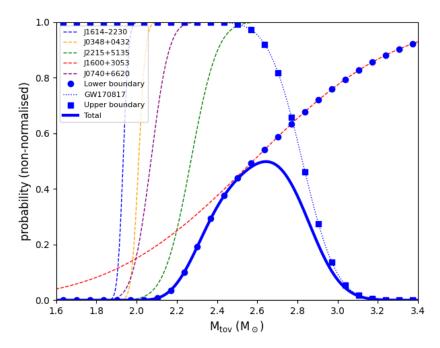


Fig. 35: The probability distribution function for the TOV mass constructed from radio and gravitational-wave observations. The different colors correspond to the different sources.

# 2.16 SetupAstroGW

class nucleardatapy.setup\_astro\_gw.SetupAstroGW(source='GW170817', hyp=1)

Instantiate the tidal deformability for a given source and obs.

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

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

obs depends on the chosen source.

#### **Parameters**

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

# **Attributes:**

## label

Attribute providing the label the data is references for figures.

# lambda\_sig\_do

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

# lambda\_sig\_up

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

# latexCite

Attribute latexCite.

#### note

Attribute providing additional notes about the data.

#### print\_outputs()

Method which print outputs on terminal's screen.

ref

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

# class nucleardatapy.setup\_astro\_gw.SetupAstroGWAverage(source='GW170817')

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

This choice is defined in the variable source.

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

#### **Parameters**

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

#### **Attributes:**

#### print\_outputs()

Method which print outputs on terminal's screen.

```
nucleardatapy.setup_astro_gw.astro_gw()
```

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

# Returns

The list of sources.

# Return type

list[str].

nucleardatapy.setup\_astro\_gw.astro\_gw\_source(source)

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

### **Parameters**

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

# Returns

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

# **Return type**

list[str].

Here is a figure which is produced with the Python sample: /sample/nucleardatapy\_plots/plot\_setupAstroGW.py

# 2.17 SetupCorEsymLsym

 $\verb|nucleardatapy.setup_CorEsymLsym.CorEsymLsym_constraints||||$ 

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

## Returns

The list of constraints.

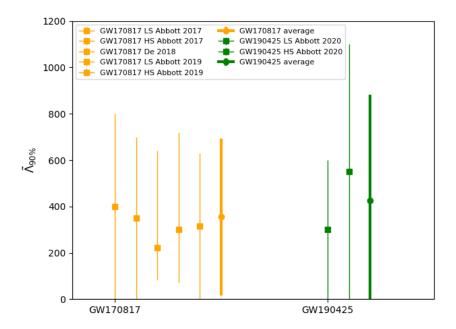


Fig. 36: Estimation of the effective tidal deformability from difference sources (different colors) and different hypotheses on the pulsar spin and the waveform employed for match filtering.

# Return type

list[str].

class nucleardatapy.setup\_CorEsymLsym.SetupCorEsymLsym(constraint='2014-IAS')

Instantiate the values of Esym and Lsym from the constraint.

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

### **Parameters**

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

#### **Attributes:**

#### **Esym**

Attribute Esym.

# Esym\_err

Attribute with uncertainty in Esym.

# Esym\_max

Attribute max of Esym.

# Esym\_min

Attribute min of Esym.

# Lsym

Attribute Lsym.

# Lsym\_err

Attribute with uncertainty in Lsym.

# Lsym\_max

Attribute max of Lsym.

# Lsym\_min

Attribute min of Lsym.

# alpha

Attribute the plot alpha

### constraint

Attribute constraint.

# label

Attribute providing the label the data is references for figures.

# note

Attribute providing additional notes about the constraint.

# print\_outputs()

Method which print outputs on terminal's screen.

# ref

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

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

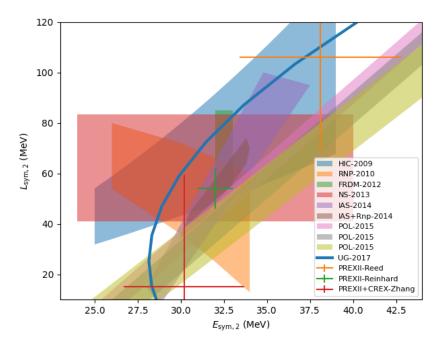


Fig. 37: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nuclear-datapy toolkit.

# **CHAPTER**

# **THREE**

# **INDICES AND TABLES**

- genindex
- modindex
- search

# **PYTHON MODULE INDEX**

# n

```
nucleardatapy, 4
nucleardatapy.setup_astro_gw, 42
nucleardatapy.setup_astro_masses, 37
{\tt nucleardatapy.setup\_astro\_mtot}, 40
nucleardatapy.setup_astro_mtov, 41
nucleardatapy.setup_CorEsymLsym, 43
nucleardatapy.setup_crust, 36
nucleardatapy.setup_eos_esym, 25
nucleardatapy.setup_eos_ffg, 7
nucleardatapy.setup_eos_hic, 20
nucleardatapy.setup_eos_micro, 9
nucleardatapy.setup_eos_micro_band, 14
nucleardatapy.setup_eos_micro_lp, 15
nucleardatapy.setup_eos_pheno, 17
nucleardatapy.setup_nuc_be_exp, 26
nucleardatapy.setup_nuc_be_theo, 30
nucleardatapy.setup_nuc_isgmr_exp, 34
nucleardatapy.setup_nuc_rch_exp, 32
```

50 Python Module Index

# **INDEX**

A	den_n (nucleardatapy.setup_eos_ffg.SetupEOSFFG at-
A (nucleardatapy.setup_crust.SetupCrust attribute), 36 A_bound (nucleardatapy.setup_crust.SetupCrust at- tribute), 36 alpha (nucleardatapy.setup_CorEsymLsym.SetupCorEsym attribute), 45	tribute), 7  den_n() (in module nucleardatapy.setup_eos_ffg), 8  den_p (nucleardatapy.setup_eos_ffg.SetupEOSFFG at- tribute), 7  diff() (nucleardatapy.setup_nuc_be_theo.SetupNucBETheo
astro_gw() (in module nucleardatapy.setup_astro_gw), 43 astro_gw_source() (in module nucleardat- apy.setup_astro_gw), 43 astro_masses() (in module nucleardat- apy.setup_astro_masses), 39 astro_masses_source() (in module nucleardat- apy.setup_astro_masses), 39 astro_mtot() (in module nucleardat- apy.setup_astro_mtot), 40 astro_mtot_source() (in module nucleardat- apy.setup_astro_mtot), 41  C	method), 30  diff_exp() (nucleardat- apy.setup_nuc_be_theo.SetupNucBETheo method), 30  dist_nbNuc (nucleardat- apy.setup_nuc_be_exp.SetupNucBEExp at- tribute), 26  dist_year (nucleardat- apy.setup_nuc_be_exp.SetupNucBEExp at- tribute), 28  drip() (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp method), 28  drip() (nucleardatapy.setup_nuc_be_theo.SetupNucBETheo method), 31
constraint (nucleardat- apy.setup_CorEsymLsym.SetupCorEsymLsym attribute), 45 constraint (nucleardat- apy.setup_eos_esym.SetupEOSEsym attribute),	E  e2a_int (nucleardatapy.setup_crust.SetupCrust attribute), 36  e2a_int (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 7
25 CorEsymLsym_constraints() (in module nucleardat-apy.setup_CorEsymLsym), 43	e2a_int2 (nucleardatapy.setup_crust.SetupCrust at- tribute), 36 e2a_int_g (nucleardatapy.setup_crust.SetupCrust at-
D  delta (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 7  den (nucleardatapy.setup_crust.SetupCrust attribute), 36  den (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 7	tribute), 36  e2a_rm (nucleardatapy.setup_crust.SetupCrust attribute), 37  e2v_int (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 7  E_unit (nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp
tribute), 7 den (nucleardatapy.setup_eos_micro_band.SetupEOSMicro_attribute), 14 den() (in module nucleardatapy.setup_eos_ffg), 8 den_cgs (nucleardatapy.setup_crust.SetupCrust attribute), 36 den_g (nucleardatapy.setup_crust.SetupCrust attribute), 36	

effg_NM() (in module nucleardatapy.setup_eos_ffg), 8	flagInterp (nucleardat-
<pre>effg_SM() (in module nucleardatapy.setup_eos_ffg), 8 eos_hic_constraints() (in module nucleardat-</pre>	apy.setup_nuc_be_exp.SetupNucBEExp at- tribute), 28
apy.setup_eos_hic), 25	
eos_micro_LP_models() (in module nucleardat-	
apy.setup_eos_micro_lp), 15	I_bound (nucleardatapy.setup_crust.SetupCrust at-
eos_micro_models() (in module nucleardat-	tribute), 36
apy.setup_eos_micro), 10	<pre>init_self()</pre>
eos_micro_models_group_NM() (in module nuclear-	apy.setup_eos_hic.SetupEOSHIC method),
datapy.setup_eos_micro), 11	20
eos_micro_models_group_SM() (in module nuclear-	init_self() (nucleardat-
datapy.setup_eos_micro), 11	apy.setup_eos_micro.SetupEOSMicro method),
eos_pheno_models() (in module nucleardat-	10
apy.setup_eos_pheno), 19	init_self() (nucleardat-
eos_pheno_params() (in module nucleardat- apy.setup_eos_pheno), 19	apy.setup_eos_micro_band.SetupEOSMicroBand method), 14
Esat (nucleardatapy.setup_eos_pheno.SetupEOSPheno	<pre>init_self() (nucleardat-</pre>
attribute), 18	apy.setup_eos_micro_lp.SetupEOSMicroLP
Esym(nucleardatapy.setup_CorEsymLsym.SetupCorEsymL	,,
attribute), 44	init_self() (nucleardat-
esym (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 7	apy.setup_nuc_be_theo.SetupNucBETheo method), 31
esym2 (nucleardatapy.setup_eos_ffg.SetupEOSFFG at- tribute), 7	K
esym4 (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 7	kf (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 8
esym_den (nucleardat-	kf() (in module nucleardatapy.setup_eos_ffg), 9
apy.setup_eos_pheno.SetupEOSPheno at- tribute), 18	kf_n (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 8
esym_e2a (nucleardat-	kf_n() (in module nucleardatapy.setup_eos_ffg), 9
apy.setup_eos_pheno.SetupEOSPheno at- tribute), 18	kf_p (nucleardatapy.setup_eos_ffg.SetupEOSFFG attribute), 8
esym_e2a_max (nucleardat-	annome), o
<pre>apy.setup_eos_esym.SetupEOSEsym attribute), 25</pre>	
esym_e2a_min (nucleardat-	label (nucleardatapy.setup_astro_gw.SetupAstroGW at-
apy.setup_eos_esym.SetupEOSEsym attribute),	tribute), 42
26	label (nucleardatapy.setup_astro_masses.SetupAstroMasses attribute), 38
Esym_err (nucleardat-	label (nucleardatapy.setup_astro_mtot.SetupAstroMtot
<pre>apy.setup_CorEsymLsym.SetupCorEsymLsym attribute), 44</pre>	attribute), 40
<pre>esym_kf (nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute), 18</pre>	attribute), 45
Esym_max (nucleardat-	label (nucleardatapy.setup_eos_ffg.SetupEOSFFG at-
apy.setup_CorEsymLsym.SetupCorEsymLsym	tribute), 8
attribute), 44	label (nucleardatapy.setup_eos_pheno.SetupEOSPheno
Esym_min (nucleardat-	attribute), 18
<pre>apy.setup_CorEsymLsym.SetupCorEsymLsym attribute), 44</pre>	label (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute), 28
esymffg() (in module nucleardatapy.setup_eos_ffg), 8	label (nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute), 34
F	<pre>label(nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp</pre>
flagI (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute), 28	attribute), 32

lambda_sig_do (nucleardat-	nucleardatapy.setup_eos_micro_band,14
apy.setup_astro_gw.SetupAstroGW attribute),	<pre>nucleardatapy.setup_eos_micro_lp, 15</pre>
42	nucleardatapy.setup_eos_pheno,17
lambda_sig_up (nucleardat-	<pre>nucleardatapy.setup_nuc_be_exp, 26</pre>
apy.setup_astro_gw.SetupAstroGW attribute),	<pre>nucleardatapy.setup_nuc_be_theo, 30</pre>
42	<pre>nucleardatapy.setup_nuc_isgmr_exp, 34</pre>
latexCite (nucleardat-	<pre>nucleardatapy.setup_nuc_rch_exp, 32</pre>
apy.setup_astro_gw.SetupAstroGW attribute),	mtot (nucleardatapy.setup_astro_mtot.SetupAstroMtot
42	attribute), 40
latexCite (nucleardat-	<pre>mu_n (nucleardatapy.setup_crust.SetupCrust attribute),</pre>
apy.setup_astro_masses.SetupAstroMasses	37
attribute), 38	<pre>mu_p (nucleardatapy.setup_crust.SetupCrust attribute),</pre>
latexCite (nucleardat-	37
<pre>apy.setup_astro_mtot.SetupAstroMtot at- tribute), 40</pre>	N
Lsym(nucleardatapy.setup_CorEsymLsym.SetupCorEsymL	Lswmucleardatapy.setup_crust.SetupCrust attribute), 36
attribute), 44	N_bound (nucleardatapy.setup_crust.SetupCrust at-
Lsym_err (nucleardat-	tribute), 36
apy.setup_CorEsymLsym.SetupCorEsymLsym	N_g (nucleardatapy.setup_crust.SetupCrust attribute), 36
attribute), 44	n_g (nucleardatapy.setup_crust.SetupCrust attribute), 37
Lsym_max (nucleardat-	nbLine (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp
apy.setup_CorEsymLsym.SetupCorEsymLsym	attribute), 28
attribute), 44	
	nbNuc (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp
· ·	attribute), 28
<pre>apy.setup_CorEsymLsym.SetupCorEsymLsym attribute), 45</pre>	nden (nucleardatapy.setup_eos_micro_band.SetupEOSMicroBand attribute), 14
M	nm_cs2 (nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute), 18
mass (nucleardatany setup, astro, masses Setup Astro Mass	esnm_den (nucleardatapy.setup_eos_pheno.SetupEOSPheno
attribute), 38	attribute), 18
matter(nucleardatapy.setup_eos_micro_band.SetupEOS	MproBandnucleardatapy.setup_eos_pheno.SetupEOSPheno
attribute), 14	attribute), 18
model (nucleardatapy.setup_eos_micro.SetupEOSMicro	$\verb"nm_gap" (nuclear data py.set up\_eos\_pheno. Set up EOSP heno$
attribute), 10	attribute), 19
model (nucleardatapy.setup_eos_micro_lp.SetupEOSMicr	OPA_kfn (nucleardatapy.setup_eos_pheno.SetupEOSPheno
attribute), 15	attribute), 19
attribute), 18	nm_pre (nucleardatapy.setup_eos_pheno.SetupEOSPheno attribute), 19
models(nucleardatapy.setup_eos_micro_band.SetupEOS	Mn95Bamucleardatapy.setup_astro_gw.SetupAstroGW at-
attribute), 14	tribute), 43
models_crust() (in module nucleardat- apy.setup_crust), 37	note (nucleardatapy.setup_astro_masses.SetupAstroMasses attribute), 38
module	note (nucleardatapy.setup_astro_mtot.SetupAstroMtot
nucleardatapy,4	attribute), 40
nucleardatapy.setup_astro_gw,42	note (nucleardatapy.setup_CorEsymLsym.SetupCorEsymLsym
nucleardatapy.setup_astro_masses,37	attribute), 45
${\tt nucleardatapy.setup\_astro\_mtot}, 40$	note (nucleardatapy.setup_eos_ffg.SetupEOSFFG
<pre>nucleardatapy.setup_astro_mtov, 41</pre>	attribute), 8
<pre>nucleardatapy.setup_CorEsymLsym, 43</pre>	note (nucleardatapy.setup_eos_pheno.SetupEOSPheno
nucleardatapy.setup_crust,36	attribute), 19
nucleardatapy.setup_eos_esym,25	<pre>note (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp</pre>
<pre>nucleardatapy.setup_eos_ffg,7</pre>	attribute), 28
<pre>nucleardatapy.setup_eos_hic, 20</pre>	$\verb"note" (nuclear data py.setup\_nuc\_isgmr\_exp.SetupNuc ISGMRExp") \\$
nucleardatany setup eos micro 9	attribute), 34

<pre>note(nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp</pre>	<pre>module, 26 nucleardatapy.setup_nuc_be_theo</pre>
<pre>nuc_be_exp_tables() (in module nucleardat-</pre>	module, 30
apy.setup_nuc_be_exp), 29	<pre>nucleardatapy.setup_nuc_isgmr_exp</pre>
<pre>nuc_be_exp_versions() (in module nucleardat-</pre>	module, 34
apy.setup_nuc_be_exp), 29	nucleardatapy.setup_nuc_rch_exp
<pre>nuc_be_theo_tables() (in module nucleardat-</pre>	module, 32
apy.setup_nuc_be_theo), 31	nucM12Mm1_cent (nucleardat-
<pre>nuc_isgmr_exp_tables() (in module nucleardat- apy.setup_nuc_isgmr_exp), 35</pre>	apy.setup_nuc_isgmr_exp.SetupNucISGMRExp attribute), 34
** *	nucM12Mm1_errm (nucleardat-
apy.setup_nuc_rch_exp), 33	apy.setup_nuc_isgmr_exp.SetupNucISGMRExp
nucA (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp	attribute), 35
attribute), 28	nucM12Mm1_errp (nucleardat-
$\verb+nucA+(nuclear datapy.setup\_nuc\_isgmr\_exp.SetupNuclSGM)$	RExp apy.setup_nuc_isgmr_exp.SetupNucISGMRExp
attribute), 34	attribute), 35
$\label{eq:nuc-rch} \textbf{nuc} \textbf{A} (nuclear datapy. setup\_nuc\_rch\_exp. SetupNucRchExp\\ attribute), 33$	attribute), 28
nucBE (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute), 28	nucN (nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute), 33
	$\verb+nucRch+ (nuclear datapy.setup\_nuc\_rch\_exp.SetupNucRchExp+$
apy.setup_nuc_be_exp.SetupNucBEExp at-	attribute), 33
tribute), 28	nucRch_err (nucleardat-
nucHT (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute), 28	apy.setup_nuc_rch_exp.SetupNucRchExp attribute), 33
nucleardatapy module, 4	nucStb1 (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute), 28
<pre>nucleardatapy.setup_astro_gw module, 42</pre>	nucSymb (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp attribute), 28
<pre>nucleardatapy.setup_astro_masses   module, 37</pre>	nucSymb (nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp attribute), 33
nucleardatapy.setup_astro_mtot	nucSymbol (nucleardat-
module, 40	apy.setup_nuc_isgmr_exp.SetupNucISGMRExp
nucleardatapy.setup_astro_mtov	attribute), 35
module, 41	nucYear (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp
nucleardatapy.setup_CorEsymLsym	attribute), 28
module, 43	nucZ (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp
nucleardatapy.setup_crust	attribute), 28
module, 36	nucZ (nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp
nucleardatapy.setup_eos_esym	attribute), 35
module, 25	<pre>nucZ(nucleardatapy.setup_nuc_rch_exp.SetupNucRchExp</pre>
<pre>nucleardatapy.setup_eos_ffg module, 7</pre>	auribuie), 55
nucleardatapy.setup_eos_hic	P
module, 20	param (nucleardatapy.setup_eos_pheno.SetupEOSPheno
nucleardatapy.setup_eos_micro	attribute), 19
module, 9	pre (nucleardatapy.setup_eos_ffg.SetupEOSFFG at-
nucleardatapy.setup_eos_micro_band	tribute), 8
module, 14	print_outputs() (nucleardat-
<pre>nucleardatapy.setup_eos_micro_lp</pre>	apy.setup_astro_gw.SetupAstroGW method),
module, 15	43
nucleardatapy.setup_eos_pheno	<pre>print_outputs()</pre>
module, 17	apy.setup_astro_gw.SetupAstroGWAverage
<pre>nucleardatapy.setup_nuc_be_exp</pre>	method), 43

<pre>print_outputs()</pre>	(nucleardat-	R	
apy.setup_astro_masses.SetupAstro	oMasses	R_unit(nucleardatapy.setup_nuc_rch_exp.SetupNuc	cRchExp
method), 38		attribute), 32	ж
<pre>print_outputs()</pre>	(nucleardat-	Rch_isotopes() (nuclear	rdat-
apy.setup_astro_masses.SetupAstro	oMassesAverag	ge apy.setup_nuc_rch_exp.SetupNucRchExp	
method), 39		method), 32	
- "	(nucleardat-	Rcl (nucleardatapy.setup_crust.SetupCrust attribute	), 36
apy.setup_astro_mtot.SetupAstroM	tot method),	ref (nucleardatapy.setup_astro_gw.SetupAstro	
40		attribute), 43	
• •	(nucleardat-	ref (nucleardatapy.setup_astro_masses.SetupAstroM	<i>lasses</i>
apy.setup_astro_mtot.SetupAstroM	totAverage	attribute), 38	
method), 40		ref (nucleardatapy.setup_astro_mtot.SetupAstroMto	ot at-
- "	(nucleardat-	tribute), 40	
apy.setup_astro_mtov.SetupAstroM	ltov	${\tt ref}$ (nuclear datapy. setup _CorEsymLsym. Setup CorE	symLsym
method), 41	(	attribute), 45	
- "	(nucleardat-	ref (nucleardatapy.setup_eos_pheno.SetupEOSP	heno
apy.setup_CorEsymLsym.SetupCor	rEsymLsym	attribute), 19	
<pre>method), 45 print_outputs()</pre>	(nucleardat-	ref (nucleardatapy.setup_nuc_be_exp.SetupNucBl	EExp
apy.setup_crust.SetupCrust method	*	attribute), 29	
print_outputs()	(nucleardat-	ref (nucleardatapy.setup_nuc_isgmr_exp.SetupNucl	SGMRExp
apy.setup_eos_esym.SetupEOSEsy	`	attribute), 35	
26	m memoa),	ref (nucleardatapy.setup_nuc_rch_exp.SetupNucRch	nExp
<pre>print_outputs()</pre>	(nucleardat-	attribute), 33	26
apy.setup_eos_ffg.SetupEOSFFG	method),	RWS (nucleardatapy.setup_crust.SetupCrust attribute	), 30
8	,,	S	
<pre>print_outputs()</pre>	(nucleardat-		7
apy.setup_eos_hic.SetupEOSHIC	method),	select() (nuclear	raat-
25		apy.setup_nuc_be_exp.SetupNucBEExp	
<pre>print_outputs()</pre>	(nucleardat-	method), 29 select_year() (nuclear	rdat
apy.setup_eos_micro.SetupEOSMi	cro method),	apy.setup_nuc_be_exp.SetupNucBEExp	raai-
10		method), 29	
	(nucleardat-	SetunAstroGW (class in nucleardatany setup astro	aw)
apy.setup_eos_micro_band.SetupE	COSMicroBand	42	-811),
method), 15		SetupAstroGWAverage (class in nuclear	rdat-
	(nucleardat-	apy.setup_astro_gw), 43	
apy.setup_eos_micro_lp.SetupEOS	MicroLP	SetupAstroMasses (class in nuclear	rdat-
method), 15	( 1 1 .	apy.setup_astro_masses), 37	
print_outputs()	(nucleardat-	SetupAstroMassesAverage (class in nuclear	rdat-
apy.setup_eos_pheno.SetupEOSPh method), 19	eno	apy.setup_astro_masses), 38	
print_outputs()	(nucleardat-	SetupAstroMtot (class in nuclear	rdat-
apy.setup_nuc_be_exp.SetupNucBi	`	$apy.setup\_astro\_mtot), 40$	
method), 29	LLxp	SetupAstroMtotAverage (class in nuclear	rdat-
print_outputs()	(nucleardat-	apy.setup_astro_mtot), 40	
apy.setup_nuc_be_theo.SetupNucB	*	SetupAstroMtov (class in nuclear	rdat-
method), 31		apy.setup_astro_mtov), 41	
<pre>print_outputs()</pre>	(nucleardat-	SetupCorEsymLsym (class in nuclear	rdat-
apy.setup_nuc_isgmr_exp.SetupNu	cISGMRExp	apy.setup_CorEsymLsym), 44	_
method), 35	•	SetupCrust (class in nucleardatapy.setup_crust), 3	
<pre>print_outputs()</pre>	(nucleardat-	SetupEOSEsym (class in nucleardatapy.setup_eos_es	<i>ym)</i> ,
apy.setup_nuc_rch_exp.SetupNuck	RchExp	SetupEOSFFG (class in nucleardatapy.setup_eos_ffg	7
method), 33		SetupEOSFFG (ctass in nucleardatapy.setup_eos_jjg SetupEOSHIC (class in nucleardatapy.setup_eos_hic	

```
Z
                                           nucleardat-
SetupEOSMicro
                      (class
                                  in
         apy.setup_eos_micro), 9
                                                        Z (nucleardatapy.setup_crust.SetupCrust attribute), 36
                          (class
                                           nucleardat-
SetupEOSMicroBand
                                    in
                                                        Z bound
                                                                    (nucleardatapy.setup_crust.SetupCrust
         apy.setup_eos_micro_band), 14
                                                                  tribute), 36
SetupEOSMicroLP
                        (class
                                   in
                                           nucleardat-
                                                        Zmax (nucleardatapy.setup_nuc_be_exp.SetupNucBEExp
         apy.setup eos micro lp), 15
                                                                  attribute), 26
SetupEOSPheno
                      (class
                                           nucleardat-
                                  in
         apy.setup_eos_pheno), 17
SetupNucBEExp
                      (class
                                  in
                                           nucleardat-
         apy.setup_nuc_be_exp), 26
SetupNucBETheo
                       (class
                                   in
                                           nucleardat-
         apy.setup_nuc_be_theo), 30
                                           nucleardat-
SetupNucISGMRExp
                         (class
                                    in
         apy.setup_nuc_isgmr_exp), 34
SetupNucRchExp
                       (class
                                           nucleardat-
                                   in
         apy.setup_nuc_rch_exp), 32
sig_do(nuclear datapy.setup\_astro\_masses.SetupAstroMasses
         attribute), 38
\verb|sig_do| (nuclear datapy.setup\_astro\_mtot.Setup Astro Mtot|
         attribute), 40
sig\_up(nucleardatapy.setup\_astro\_masses.SetupAstroMasses
         attribute), 38
sig_up (nucleardatapy.setup_astro_mtot.SetupAstroMtot
         attribute), 40
sm_cs2 (nuclear datapy. setup_eos_pheno. Setup EOSPheno
         attribute), 19
sm\_den(nuclear datapy.setup\_eos\_pheno.SetupEOSPheno)
         attribute), 19
sm_e2a(nucleardatapy.setup_eos_pheno.SetupEOSPheno
         attribute), 19
sm_gap (nucleardatapy.setup_eos_pheno.SetupEOSPheno
         attribute), 19
sm_kf (nucleardatapy.setup_eos_pheno.SetupEOSPheno
         attribute), 19
sm_kfn (nuclear datapy. setup_eos_pheno. SetupEOSPheno
         attribute), 19
\verb|sm_pre| (nuclear datapy.setup\_eos\_pheno.SetupEOSPheno|
         attribute), 19
Т
table(nucleardatapy.setup_nuc_isgmr_exp.SetupNucISGMRExp
         attribute), 35
X
xn (nucleardatapy.setup_crust.SetupCrust attribute), 37
xn_bound (nucleardatapy.setup_crust.SetupCrust at-
         tribute), 37
xp (nucleardatapy.setup_crust.SetupCrust attribute), 37
xp_bound (nucleardatapy.setup_crust.SetupCrust at-
         tribute), 37
xpn_bound (nucleardatapy.setup_crust.SetupCrust at-
         tribute), 37
```