nucleardatapy

Release 0.1

Jérôme Margueron, IRL NPA, USA

Nov 11, 2024

CONTENTS

1 Contents **3**
- 1.1 Usage . . . 3
- 1.2 API . . . 4
- 1.3 Miscelaneous . . . 4

2 Complement **7**
- 2.1 SetupMicroMatter . . . 7
- 2.2 SetupMicroMatterBand . . . 8
- 2.3 SetupMicroLP . . . 14
- 2.4 SetupPhenoMatter . . . 14
- 2.5 SetupHICMatter . . . 17
- 2.6 SetupCrust . . . 22
- 2.7 SetupMassesExp . . . 25
- 2.8 SetupMassesTheory . . . 27
- 2.9 SetupRadCh . . . 29
- 2.10 SetupISGMR . . . 31
- 2.11 SetupEsymLsym . . . 32
- 2.12 SetupAstroMasses . . . 34
- 2.13 SetupAstroMtot . . . 36
- 2.14 SetupAstroMtov . . . 38
- 2.15 SetupAstroGW . . . 38

3 Indices and tables **41**

Python Module Index **43**

Index **45**

nucleardatapy (/in short nuda/) is a Python library for nuclear physicists facilitating the access to theoretical or experimental nuclear data. It is specificaly designed for equation of state practitionners interested in the modeling of neutron stars, and it offers *simple* and *intuitive* APIs.

All data are provided with their reference, so when using these data in a scientific paper, reference to data should be provided explicitely. The reference to this toolkit could be given, but it should not mask the reference to data.

This python toolkit is designed to provide: 1) microscopic calculations in nuclear matter, 2) phenomenological predictions in nuclear matter, 3) experimental data for finite nuclei.

Check out the *Usage* section for further information, including how to *install* the project.

Note

This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

To use nucleardatapy, first download the .zip file from the git repository, or clone it in your local computer:

```
$ git clone https://github.com/jeromemargueron/nucleardatapy
```

If you have downloaded the .zip file, you can unzip it anywhere in your local computer:

```
$ unzip nucleardatapy.zip
```

Then, in all cases, you shall enter into the new folder */nucleardatapy*:

```
$ cd nucleardatapy
```

and launch the install script:

```
$ bash install.sh
```

This will copy the Python toolkit into $HOME/mylib/ as well as a few samples. It will also give you the content of the global variable NUCLEARDATAPY_TK. If you edit install.sh, you can change the version (by default it is set to the latest one) as well as the destination folder (by default it is $HOME/mylib).

Finally, you will have to create the global variable NUCLEARDATAPY_TK with its right content. If you do not want to create it each time you open a new terminal, then you can define it in your .profile or .zprofil or .bash file as:

```
export NUCLEARDATAPY_TK=$HOME/mylib/nucleardatapy
```

Note

The exact path to write above is given at the end of the installation.

1.1.2 Use nucleardatapy

Go to the folder *mylib/nucleardatapy/samples/nucleardatapy_samples/* and try that:

```
$ python3 sample_SetupMicroMatter.py
```

1.1.3 Test nucleardatapy

A set of tests can be easily performed. They are stored in tests/ folder.

```
$ bash run_tests.sh
```

1.1.4 Get started

How to obtain microscopic results for APR equation of state:

```
import os
nucleardatapy_tk = os.getenv('NUCLEARDATAPY_TK')
sys.path.insert(0, nucleardatapy_tk)

import nucleardatapy as nuda

mic = nuda.SetupMicroMatter( model = '1998-VAR-AM-APR' )

mic.print_outputs( )
```

1.2 API

nucleardatapy	This module provides microscopic, phenomenological and experimental data constraints.

1.2.1 nucleardatapy

This module provides microscopic, phenomenological and experimental data constraints.

1.3 Miscelaneous

1.3.1 Contributing

For the moment, contributions are based on co-optation among the team.

To make contribution easy, we all work in the *main* branch and we shall therefore remember to pull before working and pulling after, with a running version. For long developments, you can work in a local folder (in *mylib* for instance) and copy your contribution to the GitHub folder once you are sure it is functionning. So the final step should last less than 5 minutes, and can be safely done between a pull and before a push. Since we are not numerous, we hope that no one will work in the same part of the code at the same time (i.e. between a pull and a push). It is probably the simpler way to proceed.

Once the toolkit is released, the rules to contribute will be changing. A team of developpers should be defined and a generic email to contact them should be created. Here is a suggestion to contribute after the release.

This file describes how new contributors to the project can start contributing.

Two ways:

You can provide your data and interacting with one of our developer.

You can also join the developing team and extend the functionality of this toolkit.

Provide your data:

Please contact the developer team directly by shooting an email to TBC.

Then you can interact directly with one of our developer and provide your data. You will not be able to push your data to the repository, but an updated version of the toolkit will contain your new data.

Join the team:

Please contact the developer team directly by shooting an email to TBC. Explain the reason why you wish to join the team and if you have ideas about extending the functionality of the toolkit.

Once in the team, a branch will be dedicated to your contribution. You could show it during our virtual meetings, and your contribution will be merged to the new version of the toolkit.

1.3.2 License

TBC.

1.3.3 Report issues

For the current version, we report issues chatting among us. Once this toolkit is released, we should setup a way that users could contact us and report issues or difficulties in installing or using the toolkit.

1.3.4 Thanks

A special thanks to all contributors who accepted to share their results in this toolkit.

CHAPTER TWO

COMPLEMENT

2.1 SetupMicroMatter

class nucleardatapy.setup_micro_matter.**SetupMicroMatter**(*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, 0.05333333, 0.05727273, 0.06121212, 0.06515152, 0.06909091, 0.0730303, 0.0769697, 0.08090909, 0.08484848, 0.08878788, 0.09272727, 0.09666667, 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, 0.4])*, *var2=0.0*)

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

This choice is defined in *model*, which can chosen among the following choices: '1981-VAR-AM-FP', '1998-VAR-AM-APR', '1998-VAR-AM-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-NLEFT-AM', '2024-BHF-AM-2BF-Av8p', '2024-BHF-AM-2BF-Av18', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-CDBONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97c', '2024-BHF-AM-2BF-NSC97d', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-2BF-SSCV14', '2024-BHF-AM-23BF-Av8p', '2024-BHF-AM-23BF-Av18', '2024-BHF-AM-23BF-BONN', '2024-BHF-AM-23BF-CDBONN', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97d', '2024-BHF-AM-23BF-NSC97e', '2024-BHF-AM-23BF-NSC97f', '2024-BHF-AM-23BF-SSCV14'

Parameters

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

Attributes:

`init_self()`

Initialize variables in self.

`model`

Attribute model.

`print_outputs()`

Method which print outputs on terminal's screen.

`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', '1998-VAR-AM-APRfit', '2006-BHF-AM*', '2008-BCS-NM', '2008-AFDMC-NM', '2012-AFDMC-NM-1', '2012-AFDMC-NM-2', '2012-AFDMC-NM-3', '2012-AFDMC-NM-4', '2012-AFDMC-NM-5', '2012-AFDMC-NM-6', '2012-AFDMC-NM-7', '2008-QMC-NM-swave', '2010-QMC-NM-AV4', '2009-DLQMC-NM', '2010-MBPT-NM', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2019-MBPT-AM-L59', '2019-MBPT-AM-L69', '2020-MBPT-AM', '2022-AFDMC-NM', '2024-NLEFT-AM', '2024-BHF-AM-2BF-Av8p', '2024-BHF-AM-2BF-Av18', '2024-BHF-AM-2BF-BONN', '2024-BHF-AM-2BF-CDBONN', '2024-BHF-AM-2BF-NSC97a', '2024-BHF-AM-2BF-NSC97b', '2024-BHF-AM-2BF-NSC97c', '2024-BHF-AM-2BF-NSC97d', '2024-BHF-AM-2BF-NSC97e', '2024-BHF-AM-2BF-NSC97f', '2024-BHF-AM-2BF-SSCV14', '2024-BHF-AM-23BF-Av8p', '2024-BHF-AM-23BF-Av18', '2024-BHF-AM-23BF-BONN', '2024-BHF-AM-23BF-CDBONN', '2024-BHF-AM-23BF-NSC97a', '2024-BHF-AM-23BF-NSC97b', '2024-BHF-AM-23BF-NSC97c', '2024-BHF-AM-23BF-NSC97d', '2024-BHF-AM-23BF-NSC97e', '2024-BHF-AM-23BF-NSC97f', '2024-BHF-AM-23BF-SSCV14', '2024-BHF-AM-23BFmicro-Av18', '2024-BHF-AM-23BFmicro-BONNB', '2024-BHF-AM-23BFmicro-NSC93' :return: The list of models. :rtype: list[str].

`nucleardatapy.setup_micro_matter.`**`models_micro_matter_group_NM`**`(`*group*`)`

`nucleardatapy.setup_micro_matter.`**`models_micro_matter_group_SM`**`(`*group*`)`

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

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*`)`

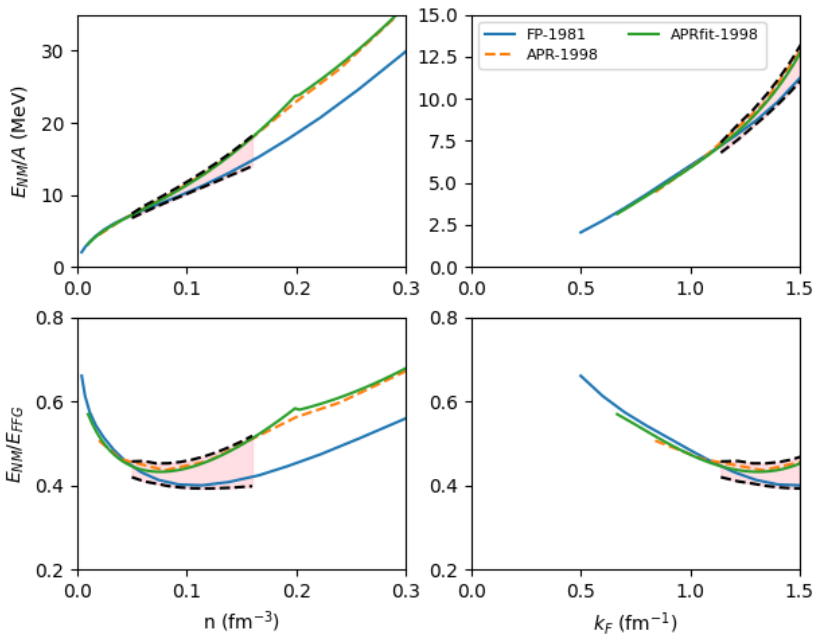

Fig. 1: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the variational models available in the nucleardatapy toolkit.

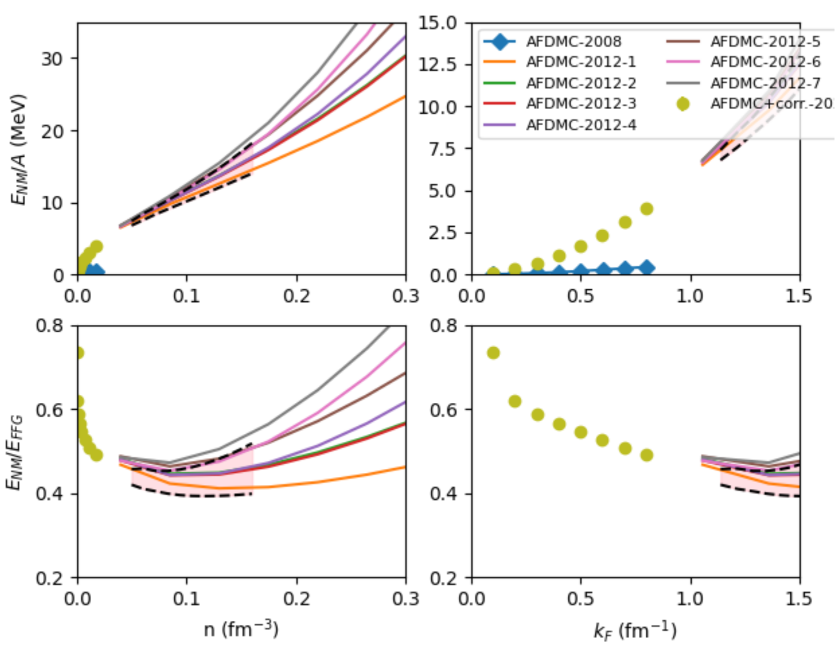

Fig. 2: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the AFDMC models available in the nucleardatapy toolkit.

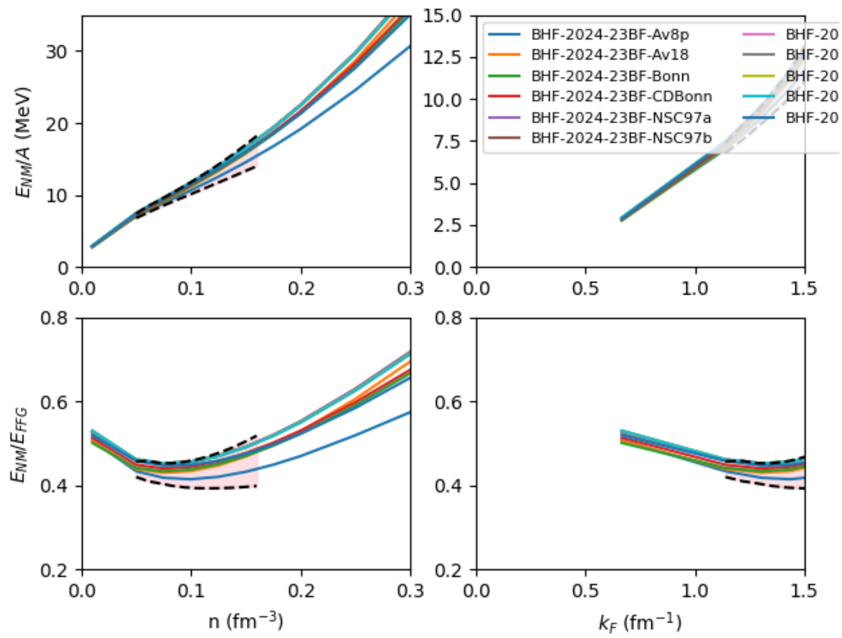

Fig. 3: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the BHF models available in the nucleardatapy toolkit.

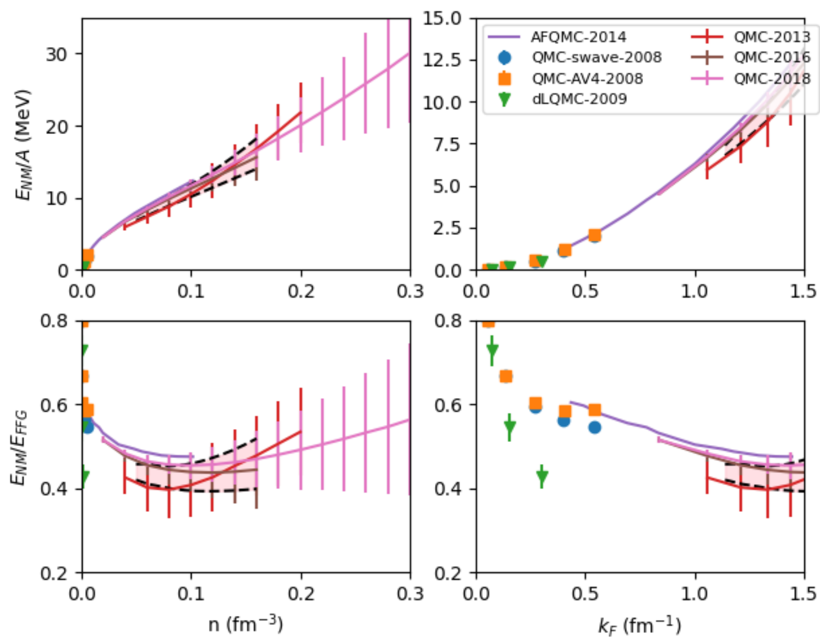

Fig. 4: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the QMC models available in the nucleardatapy toolkit.

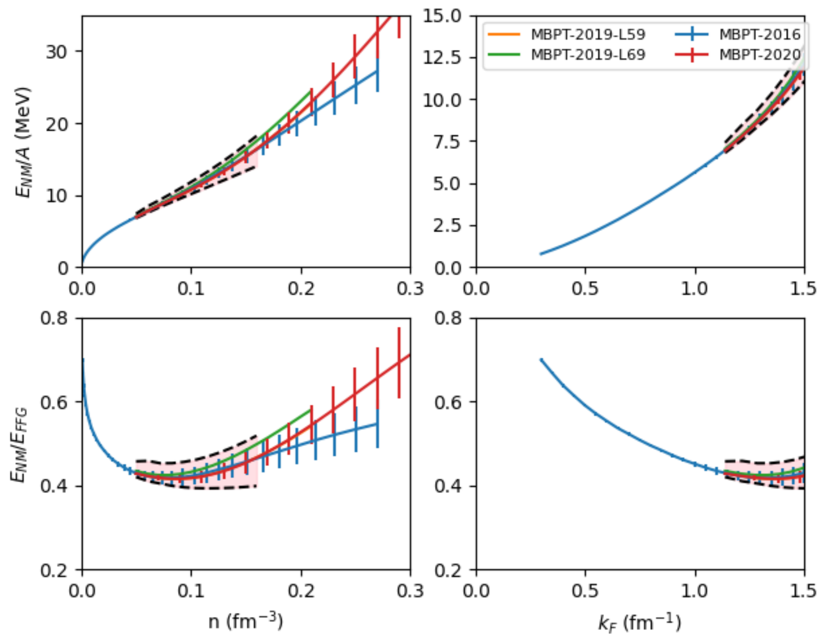

Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the MBPT models available in the nucleardatapy toolkit.

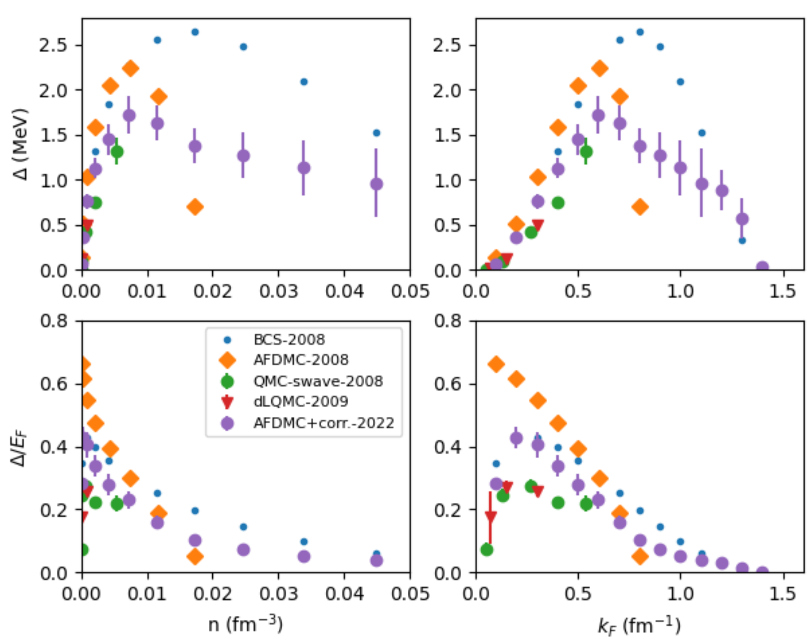

Fig. 6: This figure shows the pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

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.

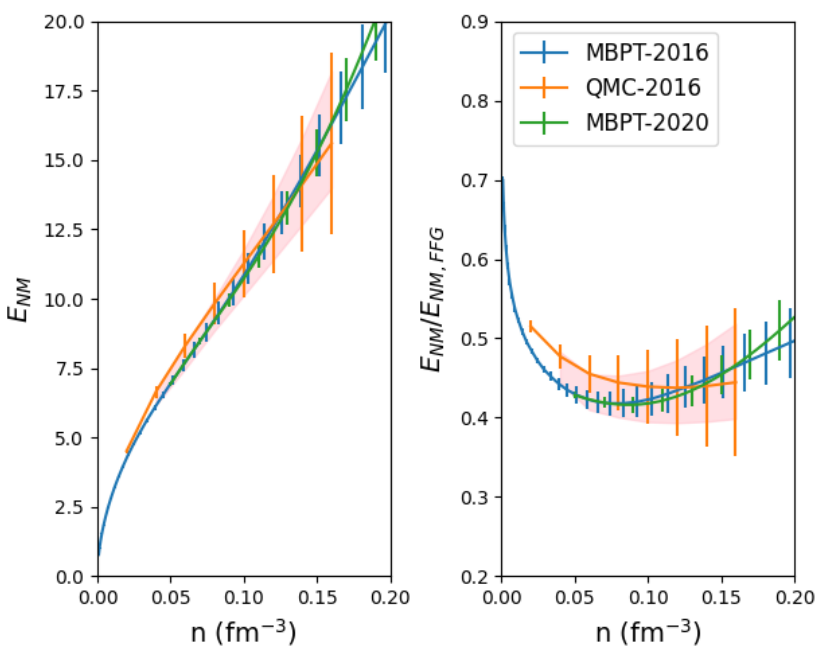

Fig. 7: Uncertainty band in NM obtained from the analysis of different predictions: MBPT-2016, QMC-2016 and MBPT-2020.

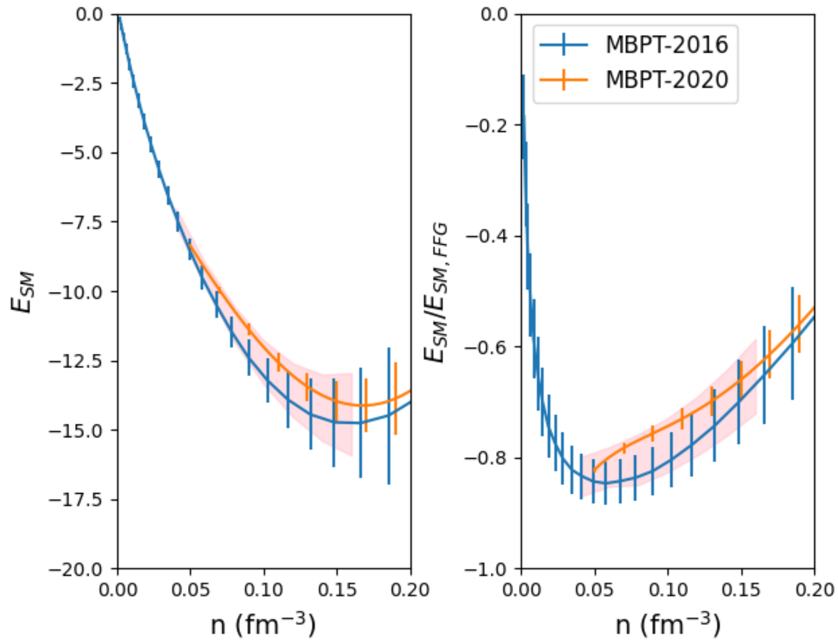

Fig. 8: Uncertainty band in SM obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

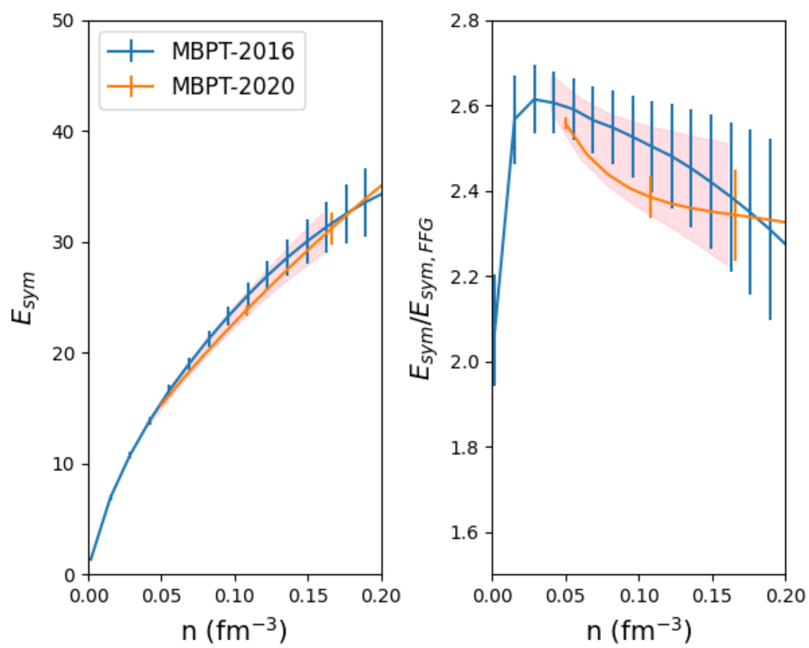

Fig. 9: Uncertainty band for the symmetry energy obtained from the analysis of different predictions: MBPT-2016 and MBPT-2020.

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-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.

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-GAP', '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: /sample/nucleardatapy_plots/plot_setupLP.py

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', 'FPL', 'LNS', 'LNS1', 'LNS5', 'NRAPR', 'RATP', 'SAMI', 'SGII', 'SIII', 'SKGSIGMA', 'SKI2', 'SKI4', 'SKMP', 'SKMS', 'SKO', 'SKOP', 'SKP', 'SKRSIGMA', 'SKX', 'Skz2', 'SLY4', 'SLY5', 'SLY230A', 'SLY230B', 'SV', 'T6', 'T44', 'UNEDF0', 'UNEDF1'.

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

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

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

Parameters

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

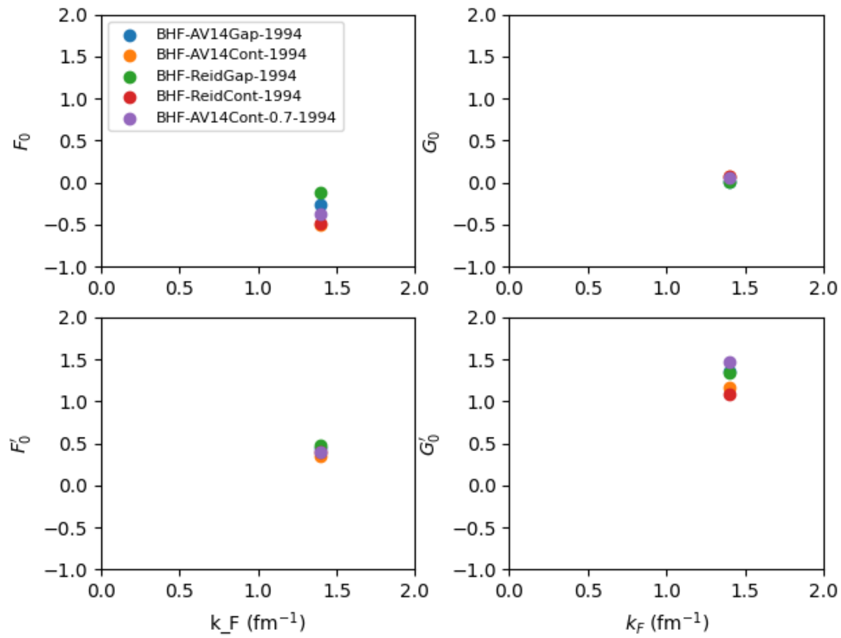

Fig. 10: This figure shows the L=0 Landau parameters in SM for different NN interactions obtained from BHF calculations.

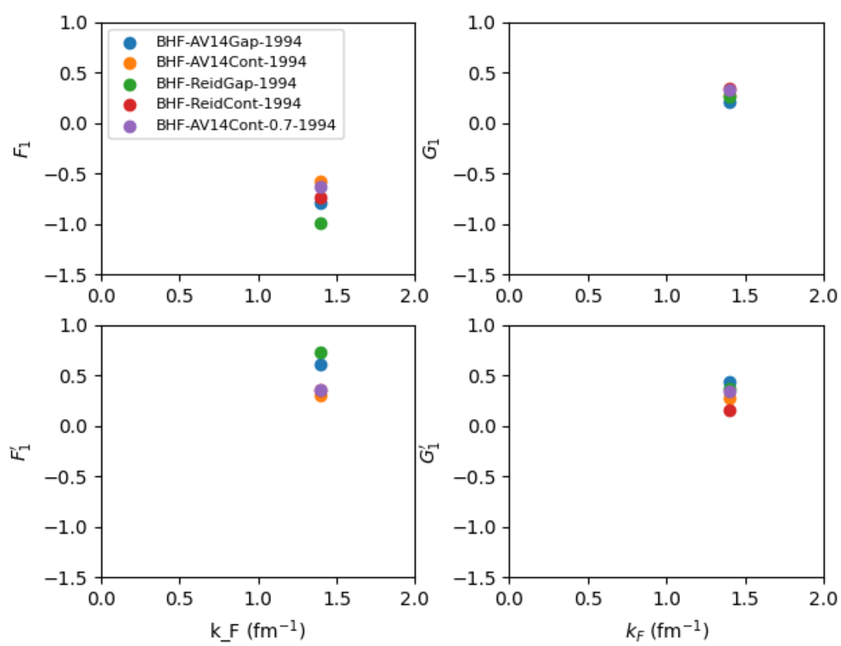

Fig. 11: This figure shows the L=1 Landau parameters in SM for different NN interactions obtained from BHF calculations.

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_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', 'PKO1', 'PKO2', 'PKO3'.

Return type
list[str].

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

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.

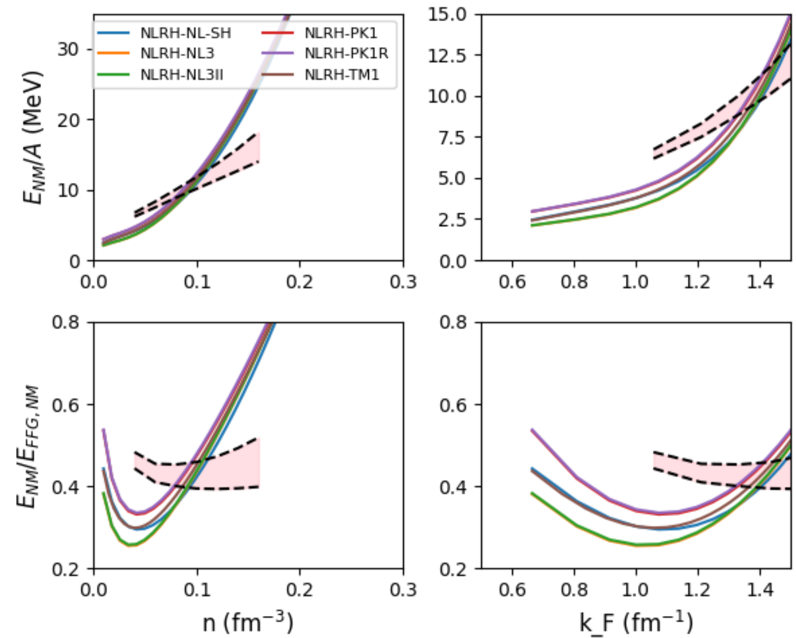

Fig. 12: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree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

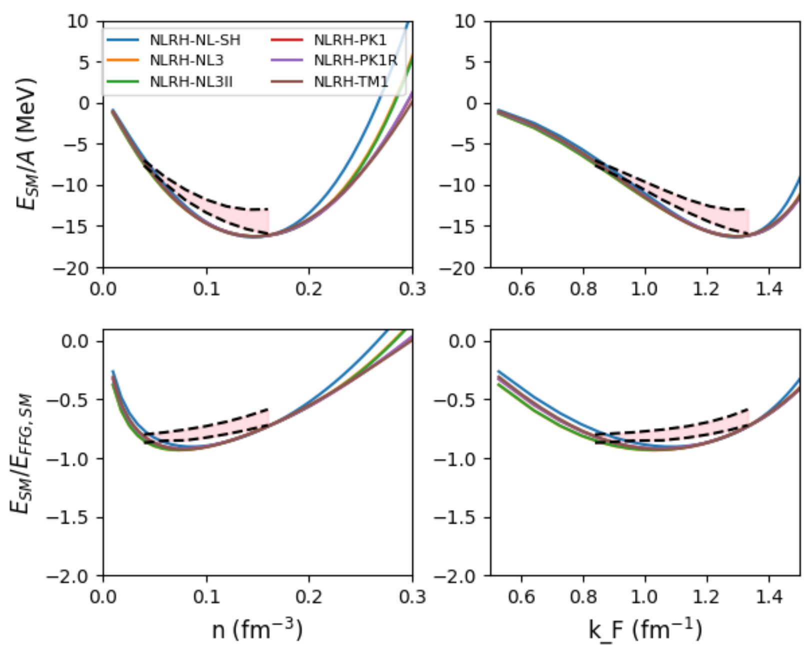

Fig. 13: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic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

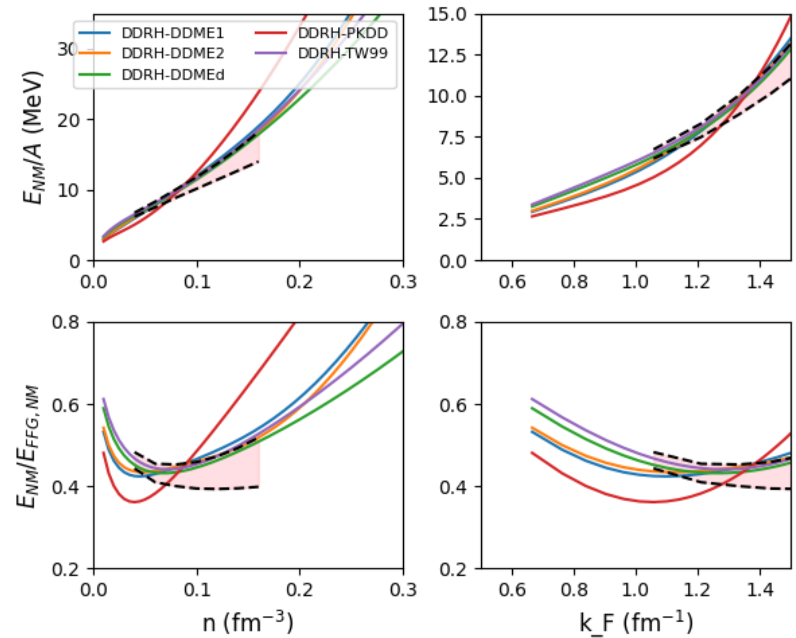

Fig. 14: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

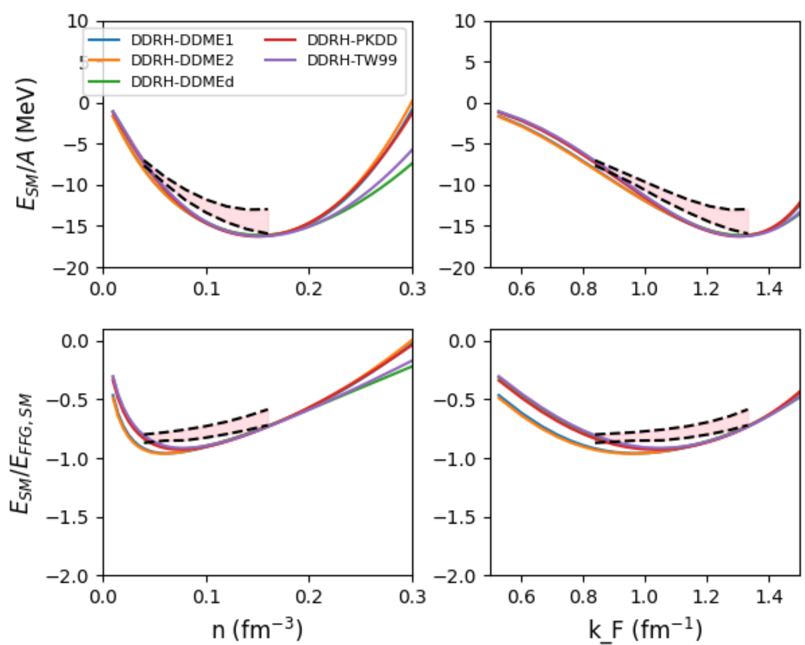

Fig. 15: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

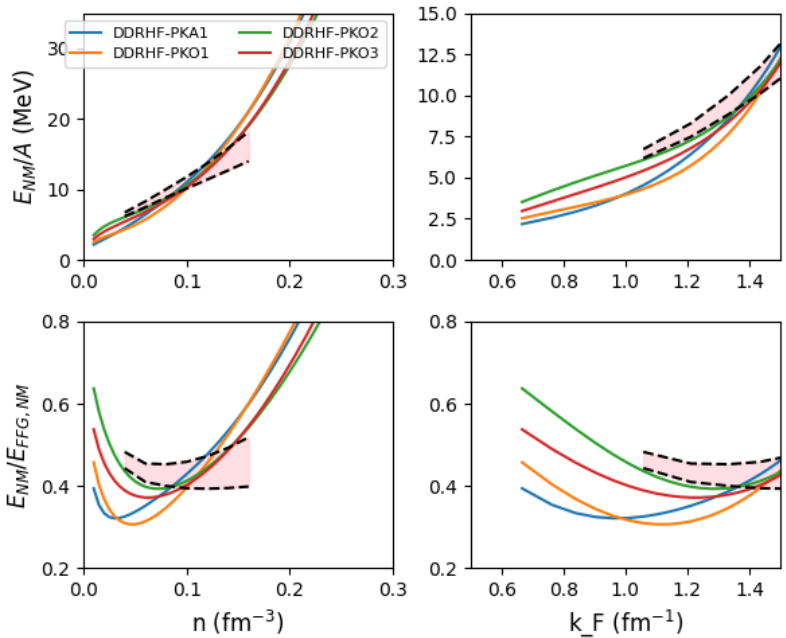

Fig. 16: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

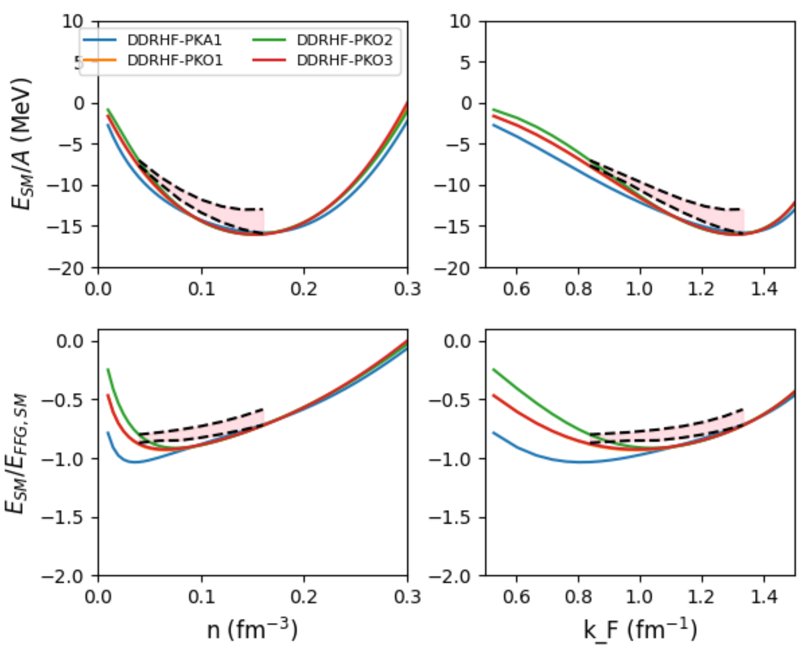

Fig. 17: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree-Fock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

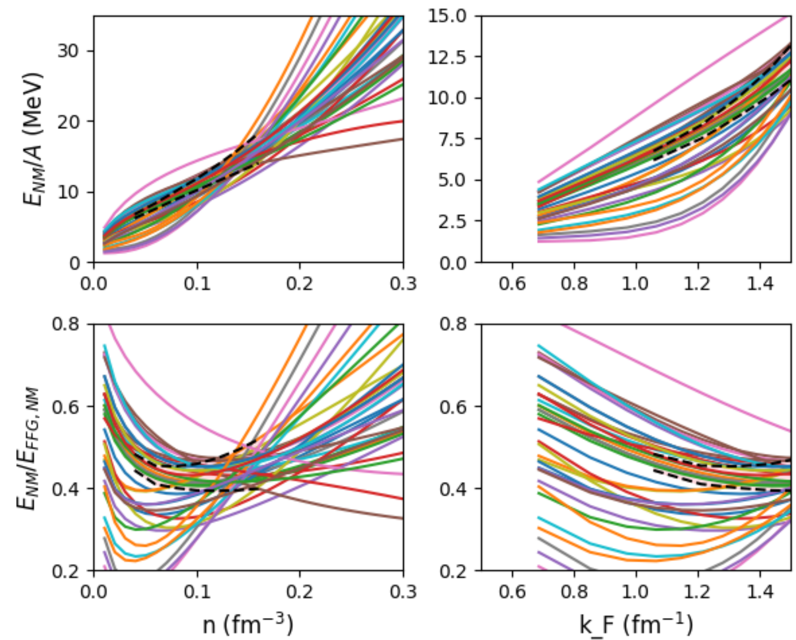

Fig. 18: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

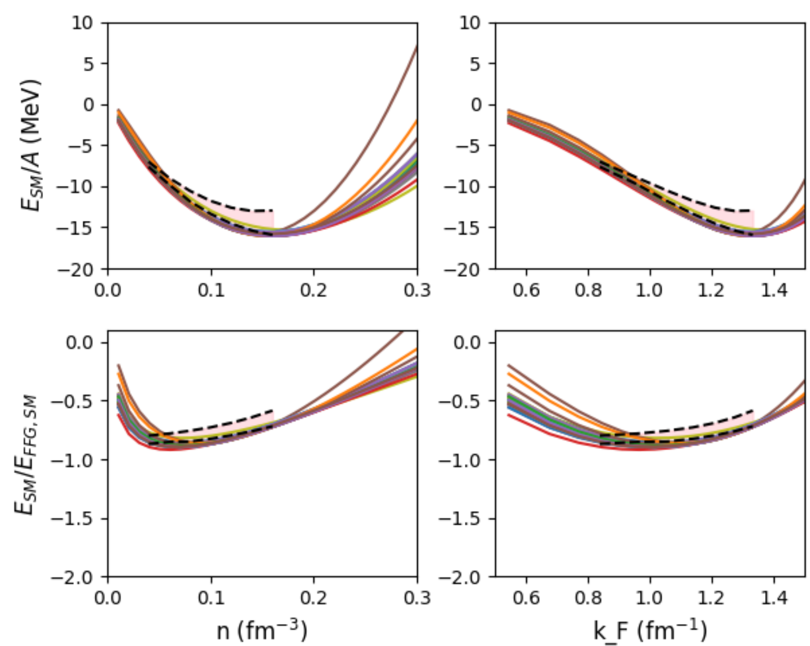

Fig. 19: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on the standard Skyrme interaction available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

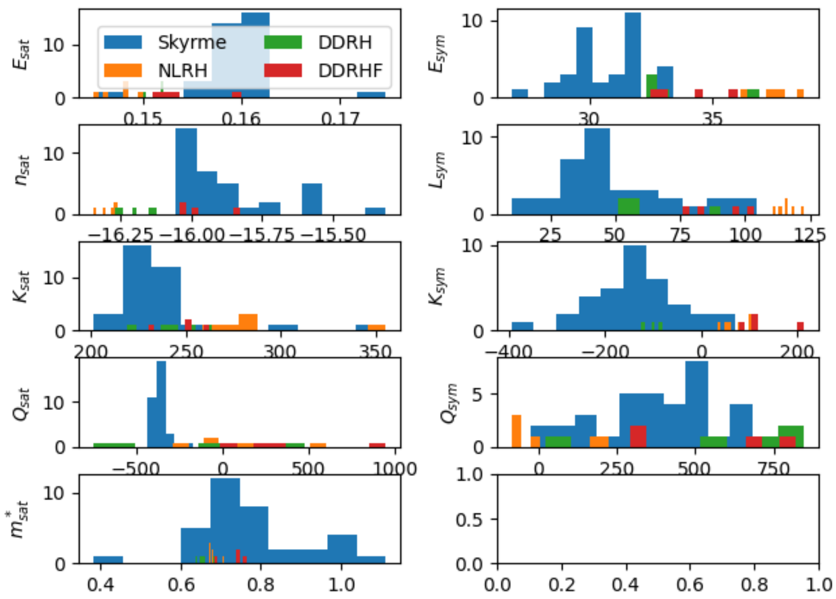

Fig. 20: Distribution of NEP for phenomenological models available in the nucleardatapy toolkit.

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'].

Returns

The list of constraints.

Return type

list[str].

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).

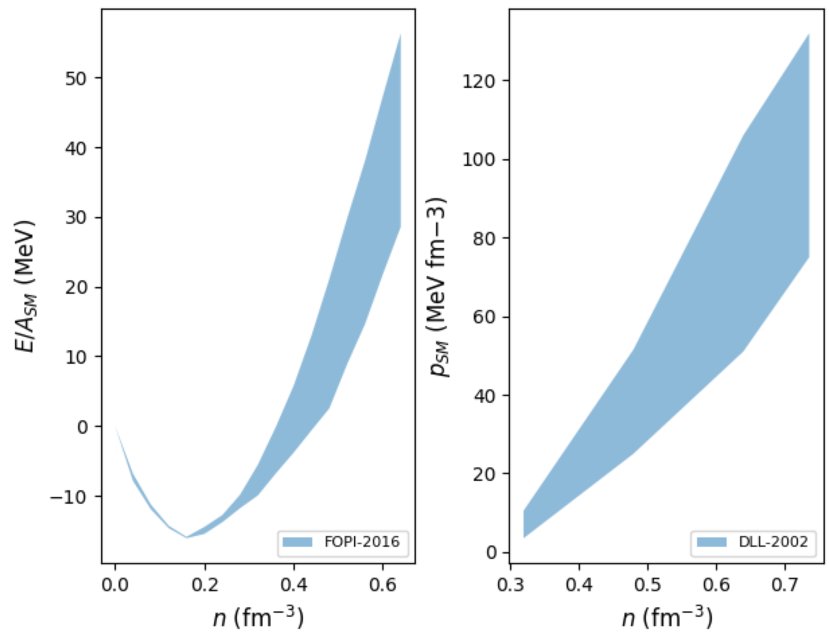

Fig. 21: HIC Experimental constraints for the energy per particle (left) and pressure (right) in SM as a function of the particle density for different analyses available in the *nuda* toolkit.

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.

xp
Attribute the fraction of protons.

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

nucleardatapy.setup_crust.**models_crust**()

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

Returns
The list of tables.

Return type
list[str].

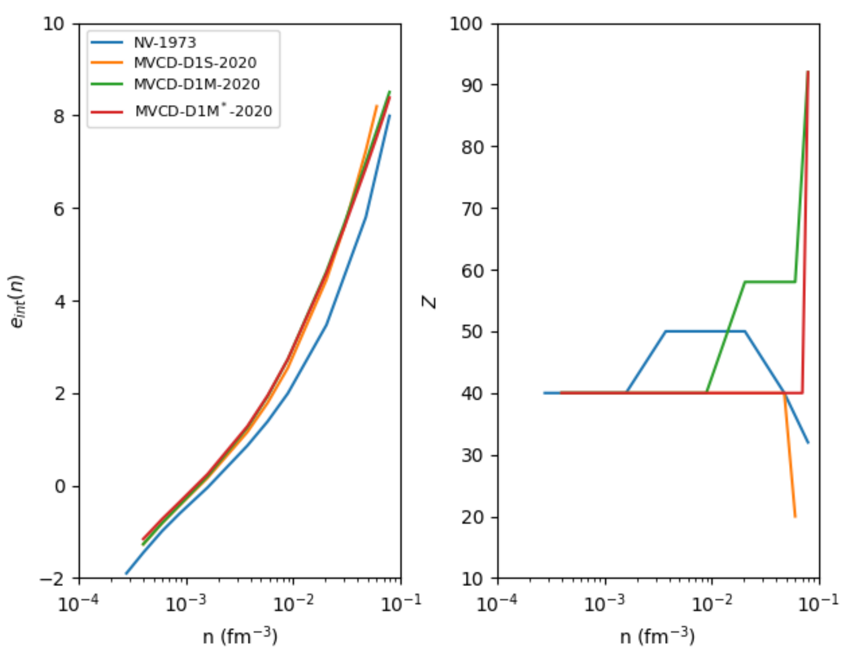

Fig. 22: Properties of the crust as given by the models available in the nuda toolkit.

2.7 SetupMassesExp

class nucleardatapy.setup_masses_exp.**SetupMassesExp**(*table='AME'*, *version='2020'*)

Instantiate the experimental nuclear masses from AME mass table.

This choice is defined in the variables *table* and *version*.

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

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

Parameters

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

Attributes:

Zmax

maximum charge of nuclei present in the table.

Type

Attribute Zmax

dist_nbNuc

attribute number of nuclei discovered per year

dist_year

attribute distribution of years

drip(*Zmax=95*)

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

Parameters

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

Attributes:

flagI

Attribute I.

flagInterp

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

label

Attribute providing the label the data is references for figures.

nbLine

Attribute with the number of line in the file.

nbNuc

Attribute with the number of nuclei read in the file.

note

Attribute providing additional notes about the data.

nucA

Attribute A (mass of the nucleus).

`nucBE`

Attribute BE (Binding Energy) of the nucleus.

`nucBE_err`

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

`nucHT`

Attribute HT (half-Time) of the nucleus.

`nucN`

Attribute N (number of neutrons of the nucleus).

`nucStbl`

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

`nucSymb`

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

`nucYear`

Attribute year of the discovery of the nucleus.

`nucZ`

Attribute Z (charge of the nucleus).

`print_outputs()`

Method which print outputs on terminal's screen.

`ref`

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

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

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

Parameters

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

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

Attributes:

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

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

Parameters

- **year_min**
- **year_max**
- **state** (*str, optional. Default 'gs'.*) – select the kind of state. If state='gs', select nuclei measured in their ground state.

Attributes:

`nucleardatapy.setup_masses_exp.`**`tables_masses_exp`**`()`

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

Returns
The list of tables.

Return type
list[str].

`nucleardatapy.setup_masses_exp.`**`versions_masses_exp`**`(`*table*`)`

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

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

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

Return type
list[str].

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

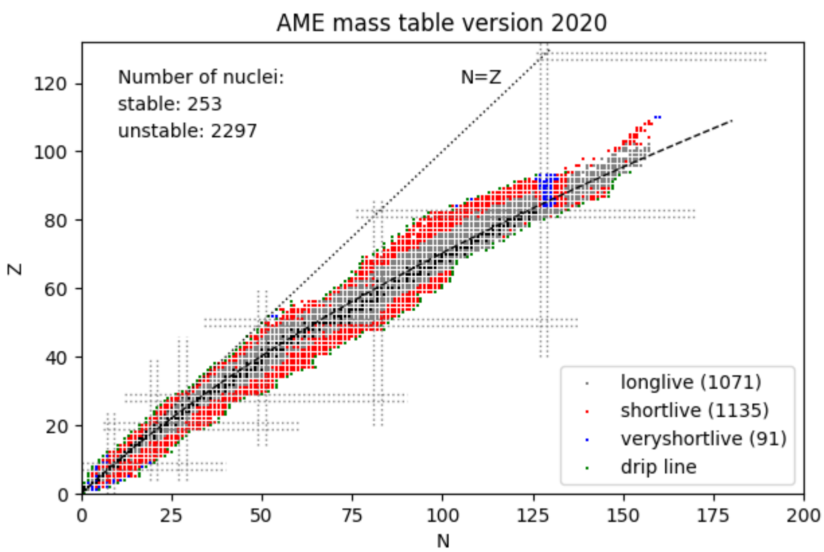

Fig. 23: The nuclear chart based on AME 2020 table. The different colors correspond to the different measured half-times of nuclei.

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*.

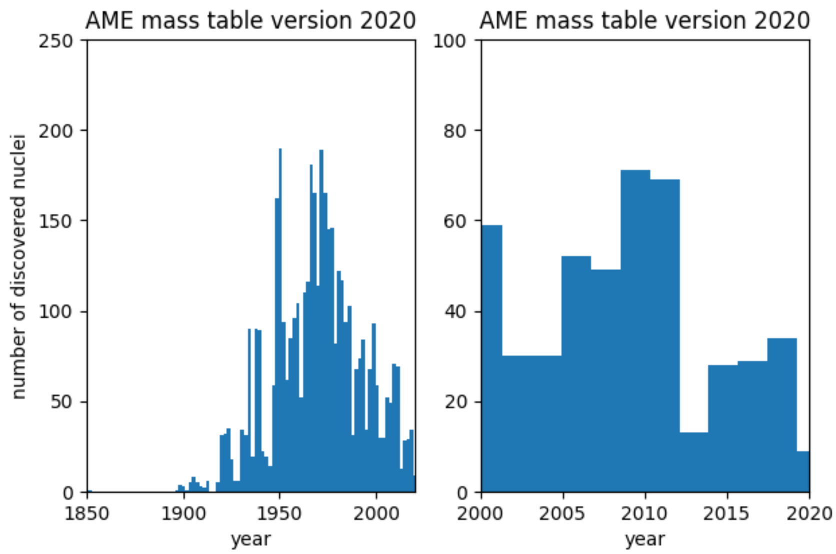

Fig. 24: Histogram showing the distribution of nuclei per discovery year, since the first one discovered in 1851.

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

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

Attributes:

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

Method calculates the difference between a given mass model and table_ref.

Parameters

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

Attributes:

diff_exp(*table_exp*, *version_exp*, *Zref=50*)

Method calculates the difference between a given experimental mass (identified by *table_exp* and *version_exp*) and table_ref.

Parameters

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

Attributes:

drip(*Zmax=95*)

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

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

drip line.

Attributes:

init_self()

Initialize variables in self.

print_outputs()

Method which print outputs on terminal's screen.

nucleardatapy.setup_masses_theory.**tables_masses_theory**()

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

Returns

The list of tables.

Return type

list[str].

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

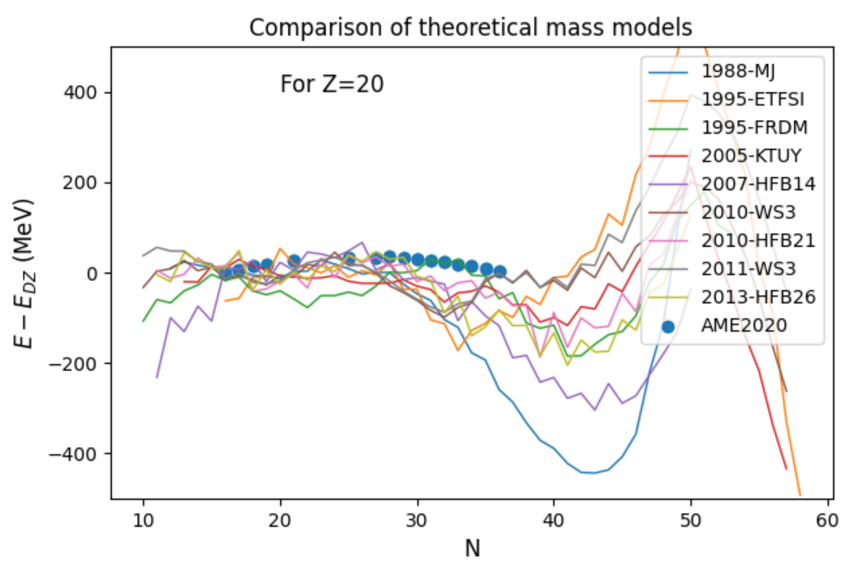

Fig. 25: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 20.

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'.

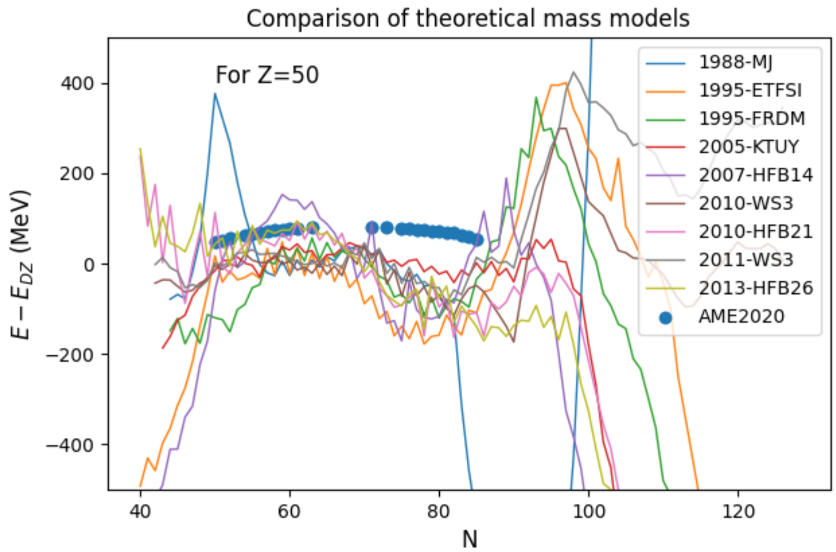

Fig. 26: Differences between binding energies predicted by different models with respect to the one predicted by Duflo-Zuker for Z = 50.

Parameters

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

Attributes:

`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.

`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].

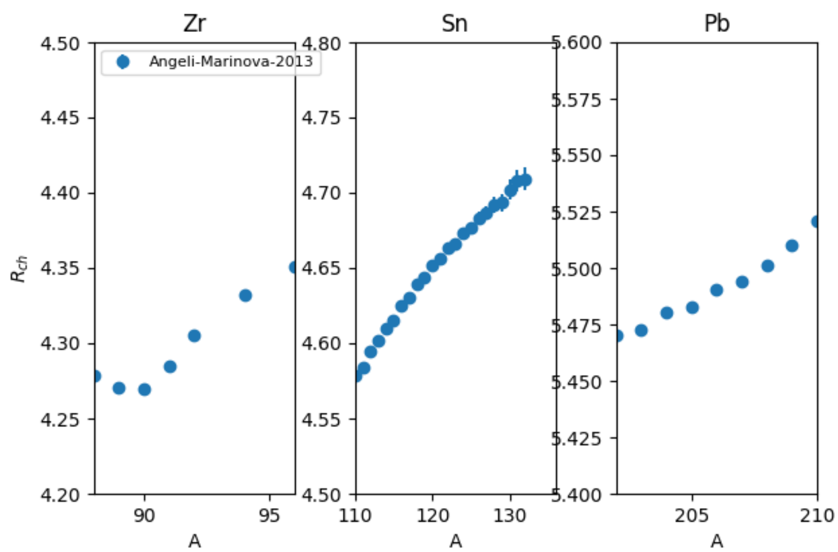

Fig. 27: Charge radii for Zn, Sn, and Pb isotopes and for the models available in the nuda toolkit.

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.

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].

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:

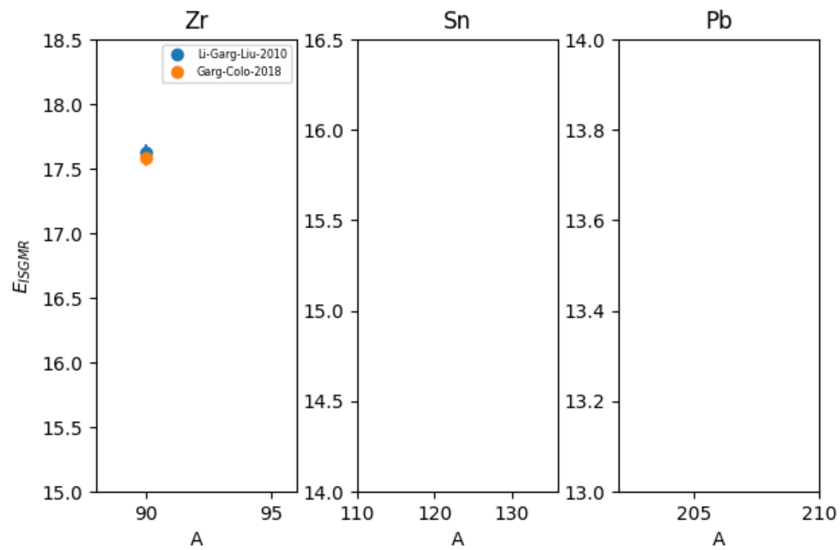

Fig. 28: ISGMR energies available in the nucleardatapy toolkit.

Esym
Attribute Esym.

Esym_err
Attribute with uncertainty in Esym.

Esym_max
Attribute max of Esym.

Esym_min
Attribute min of Esym.

Lsym
Attribute Lsym.

Lsym_err
Attribute with uncertainty in Lsym.

Lsym_max
Attribute max of Lsym.

Lsym_min
Attribute min of Lsym.

alpha
Attribute the plot alpha

constraint
Attribute constraint.

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

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

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

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

`nucleardatapy.setup_EsymLsym.constraints_EsymLsym()`

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

Returns
The list of constraints.

Return type
list[str].

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

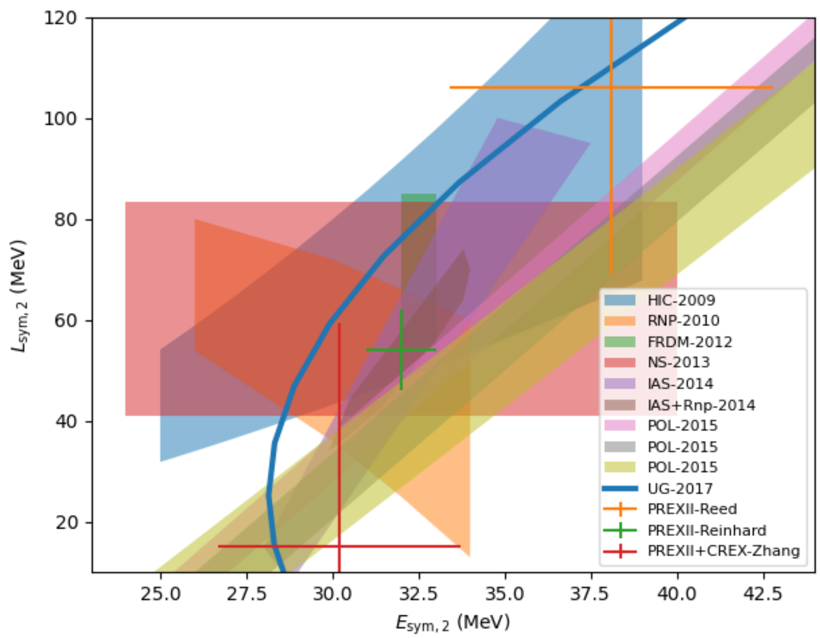

Fig. 29: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nucleardatapy toolkit.

2.12 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.

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

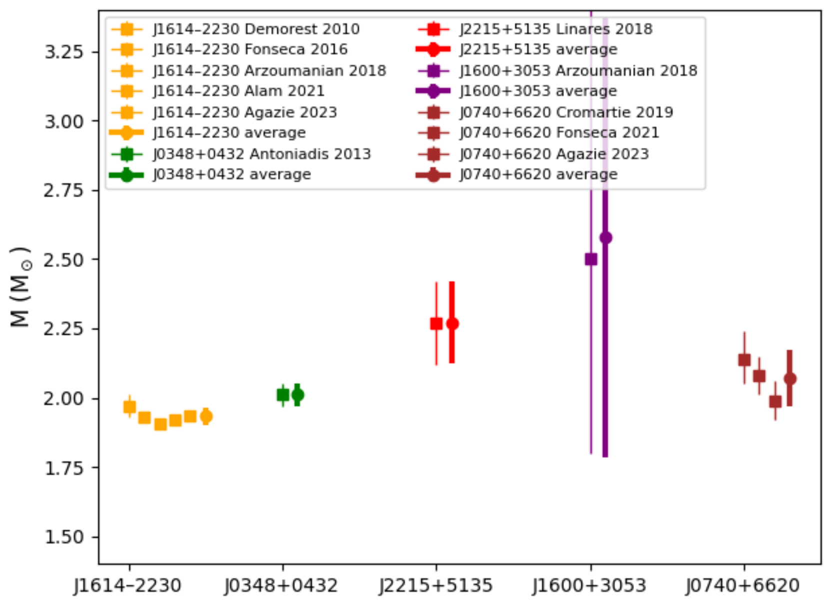

Fig. 30: The masses measured for massive neutron stars is radio-astronomy. The different colors correspond to the different sources.

2.13 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

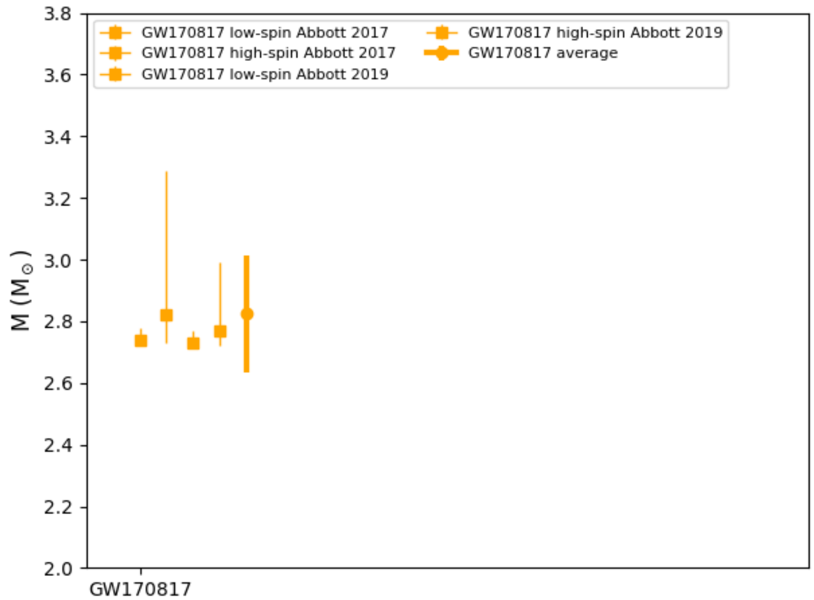

Fig. 31: The total mass measured fo binary neutron star mergers. The different colors correspond to the different sources.

2.14 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

2.15 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.

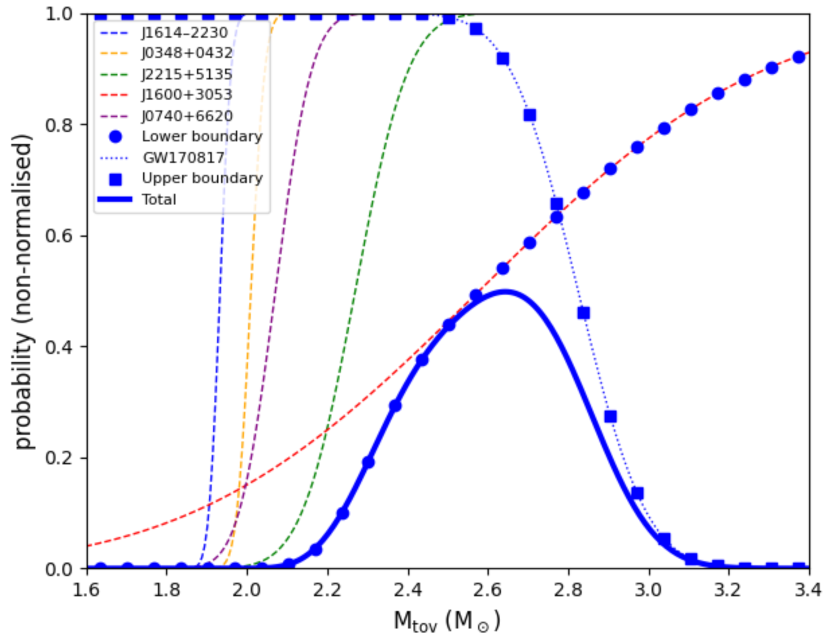

Fig. 32: The probability distribution function for the TOV mass constructed from radio and gravitational-wave observations. The different colors correspond to the different sources.

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

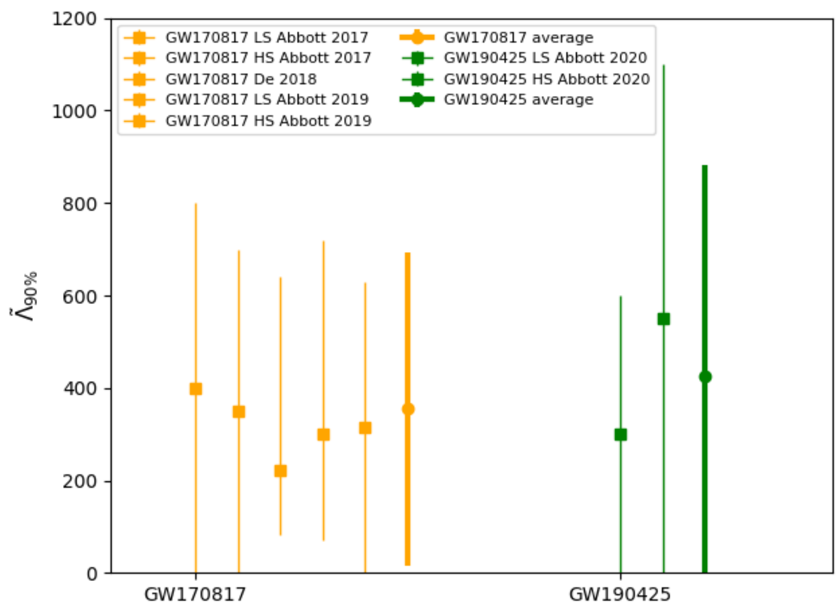

Fig. 33: Estimation of the effective tidal deformability from difference sources (different colors) and different hypotheses on the pulsar spin and the waveform employed for match filtering.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

nucleardatapy, 4
nucleardatapy.setup_astro_gw, 38
nucleardatapy.setup_astro_masses, 34
nucleardatapy.setup_astro_mtot, 36
nucleardatapy.setup_astro_mtov, 38
nucleardatapy.setup_crust, 22
nucleardatapy.setup_EsymLsym, 32
nucleardatapy.setup_hic_matter, 17
nucleardatapy.setup_ISGMR, 31
nucleardatapy.setup_masses_exp, 25
nucleardatapy.setup_masses_theory, 27
nucleardatapy.setup_micro_LP, 14
nucleardatapy.setup_micro_matter, 7
nucleardatapy.setup_micro_matter_band, 8
nucleardatapy.setup_pheno_matter, 14
nucleardatapy.setup_rad_ch, 29

INDEX

A

A (*nucleardatapy.setup_crust.SetupCrust attribute*), 22
alpha (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
astro_gw() (*in module nucleardatapy.setup_astro_gw*), 40
astro_gw_source() (*in module nucleardatapy.setup_astro_gw*), 40
astro_masses() (*in module nucleardatapy.setup_astro_masses*), 35
astro_masses_source() (*in module nucleardatapy.setup_astro_masses*), 35
astro_mtot() (*in module nucleardatapy.setup_astro_mtot*), 37
astro_mtot_source() (*in module nucleardatapy.setup_astro_mtot*), 37

C

constraint (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
constraints_EsymLsym() (*in module nucleardatapy.setup_EsymLsym*), 34
constraints_HIC_matter() (*in module nucleardatapy.setup_hic_matter*), 22

D

den (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
den (*nucleardatapy.setup_micro_matter_band.SetupMicroMatterBand attribute*), 12
den_cgs (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
den_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
diff() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 28
diff_exp() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 28
dist_nbNuc (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
dist_year (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
drip() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 25
drip() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 28

E

e2a_int (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
e2a_int2 (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
e2a_int_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
e2a_rm (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
E_unit (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 31
Esat (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
Esym (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 32
esym_den (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
esym_e2a (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
Esym_err (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
esym_kf (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
Esym_max (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
Esym_min (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33

F

flagI (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
flagInterp (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25

I

init_self() (*nucleardatapy.setup_hic_matter.SetupHICMatter method*), 17
init_self() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 29
init_self() (*nucleardatapy.setup_micro_LP.SetupMicroLP method*), 14
init_self() (*nucleardatapy.setup_micro_matter.SetupMicroMatter method*), 8
init_self() (*nucleardatapy.setup_micro_matter_band.SetupMicroMatterBand method*), 12

L

label (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 39
label (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
label (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 36
label (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
label (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
label (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
label (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
label (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
lambda_sig_do (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 39
lambda_sig_up (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 39
latexCite (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 39
latexCite (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
latexCite (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 37
Lsym (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
Lsym_err (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
Lsym_max (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
Lsym_min (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33

M

mass (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
matter (*nucleardatapy.setup_micro_matter_band.SetupMicroMatterBand attribute*), 12
model (*nucleardatapy.setup_micro_LP.SetupMicroLP attribute*), 14
model (*nucleardatapy.setup_micro_matter.SetupMicroMatter attribute*), 8
model (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
models (*nucleardatapy.setup_micro_matter_band.SetupMicroMatterBand attribute*), 12
models_crust() (*in module nucleardatapy.setup_crust*), 24
models_micro_LP() (*in module nucleardatapy.setup_micro_LP*), 14
models_micro_matter() (*in module nucleardatapy.setup_micro_matter*), 8
models_micro_matter_group_NM() (*in module nucleardatapy.setup_micro_matter*), 8
models_micro_matter_group_SM() (*in module nucleardatapy.setup_micro_matter*), 8
models_pheno_matter() (*in module nucleardatapy.setup_pheno_matter*), 17
module
- nucleardatapy, 4
- nucleardatapy.setup_astro_gw, 38
- nucleardatapy.setup_astro_masses, 34
- nucleardatapy.setup_astro_mtot, 36
- nucleardatapy.setup_astro_mtov, 38
- nucleardatapy.setup_crust, 22
- nucleardatapy.setup_EsymLsym, 32
- nucleardatapy.setup_hic_matter, 17
- nucleardatapy.setup_ISGMR, 31
- nucleardatapy.setup_masses_exp, 25
- nucleardatapy.setup_masses_theory, 27
- nucleardatapy.setup_micro_LP, 14
- nucleardatapy.setup_micro_matter, 7

nucleardatapy.setup_micro_matter_band, 8
nucleardatapy.setup_pheno_matter, 14
nucleardatapy.setup_rad_ch, 29
mtot (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 37
mu_n (*nucleardatapy.setup_crust.SetupCrust attribute*), 24
mu_p (*nucleardatapy.setup_crust.SetupCrust attribute*), 24

N

N (*nucleardatapy.setup_crust.SetupCrust attribute*), 22
N_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 22
N_g (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
nbLine (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
nbNuc (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
nden (*nucleardatapy.setup_micro_matter_band.SetupMicroMatterBand attribute*), 12
nm_cs2 (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
nm_den (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
nm_e2a (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
nm_gap (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
nm_kfn (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
nm_pre (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
note (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 39
note (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
note (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 37
note (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 33
note (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
note (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
note (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
note (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
nucA (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
nucA (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
nucA (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
nucBE (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25
nucBE_err (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucHT (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucleardatapy
module, 4
nucleardatapy.setup_astro_gw
module, 38
nucleardatapy.setup_astro_masses
module, 34
nucleardatapy.setup_astro_mtot
module, 36
nucleardatapy.setup_astro_mtov
module, 38
nucleardatapy.setup_crust
module, 22
nucleardatapy.setup_EsymLsym
module, 32
nucleardatapy.setup_hic_matter
module, 17
nucleardatapy.setup_ISGMR
module, 31
nucleardatapy.setup_masses_exp
module, 25
nucleardatapy.setup_masses_theory
module, 27
nucleardatapy.setup_micro_LP
module, 14
nucleardatapy.setup_micro_matter
module, 7
nucleardatapy.setup_micro_matter_band
module, 8
nucleardatapy.setup_pheno_matter
module, 14
nucleardatapy.setup_rad_ch
module, 29
nucM12Mm1_cent (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
nucM12Mm1_errm (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
nucM12Mm1_errp (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
nucN (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucN (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30

nucRch (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
nucRch_err (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
nucStbl (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucSymb (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucSymb (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
nucSymbol (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
nucYear (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucZ (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
nucZ (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
nucZ (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30

P

param (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
params_pheno_matter() (*in module nucleardatapy.setup_pheno_matter*), 17
print_outputs() (*nucleardatapy.setup_astro_gw.SetupAstroGW method*), 39
print_outputs() (*nucleardatapy.setup_astro_gw.SetupAstroGWAverage method*), 40
print_outputs() (*nucleardatapy.setup_astro_masses.SetupAstroMasses method*), 35
print_outputs() (*nucleardatapy.setup_astro_masses.SetupAstroMassesAverage method*), 35
print_outputs() (*nucleardatapy.setup_astro_mtot.SetupAstroMtot method*), 37
print_outputs() (*nucleardatapy.setup_astro_mtot.SetupAstroMtotAverage method*), 37
print_outputs() (*nucleardatapy.setup_astro_mtov.SetupAstroMtov method*), 38
print_outputs() (*nucleardatapy.setup_crust.SetupCrust method*), 24
print_outputs() (*nucleardatapy.setup_EsymLsym.SetupEsymLsym method*), 34
print_outputs() (*nucleardatapy.setup_hic_matter.SetupHICMatter method*), 17
print_outputs() (*nucleardatapy.setup_ISGMR.SetupISGMR method*), 32
print_outputs() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 26
print_outputs() (*nucleardatapy.setup_masses_theory.SetupMassesTheory method*), 29
print_outputs() (*nucleardatapy.setup_micro_LP.SetupMicroLP method*), 14
print_outputs() (*nucleardatapy.setup_micro_matter.SetupMicroMatter method*), 8
print_outputs() (*nucleardatapy.setup_micro_matter_band.SetupMicroMatterBand method*), 12
print_outputs() (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter method*), 16
print_outputs() (*nucleardatapy.setup_rad_ch.SetupRadCh method*), 31

R

R_unit (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 30
RadCh_isotopes() (*nucleardatapy.setup_rad_ch.SetupRadCh method*), 30
ref (*nucleardatapy.setup_astro_gw.SetupAstroGW attribute*), 39
ref (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
ref (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 37
ref (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 34
ref (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
ref (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 26
ref (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
ref (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 31
RWS (*nucleardatapy.setup_crust.SetupCrust attribute*), 23

S

select() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 26

select_year() (*nucleardatapy.setup_masses_exp.SetupMassesExp method*), 26
SetupAstroGW (*class in nucleardatapy.setup_astro_gw*), 38
SetupAstroGWAverage (*class in nucleardatapy.setup_astro_gw*), 39
SetupAstroMasses (*class in nucleardatapy.setup_astro_masses*), 34
SetupAstroMassesAverage (*class in nucleardatapy.setup_astro_masses*), 35
SetupAstroMtot (*class in nucleardatapy.setup_astro_mtot*), 36
SetupAstroMtotAverage (*class in nucleardatapy.setup_astro_mtot*), 37
SetupAstroMtov (*class in nucleardatapy.setup_astro_mtov*), 38
SetupCrust (*class in nucleardatapy.setup_crust*), 22
SetupEsymLsym (*class in nucleardatapy.setup_EsymLsym*), 32
SetupHICMatter (*class in nucleardatapy.setup_hic_matter*), 17
SetupISGMR (*class in nucleardatapy.setup_ISGMR*), 31
SetupMassesExp (*class in nucleardatapy.setup_masses_exp*), 25
SetupMassesTheory (*class in nucleardatapy.setup_masses_theory*), 27
SetupMicroLP (*class in nucleardatapy.setup_micro_LP*), 14
SetupMicroMatter (*class in nucleardatapy.setup_micro_matter*), 7
SetupMicroMatterBand (*class in nucleardatapy.setup_micro_matter_band*), 8
SetupPhenoMatter (*class in nucleardatapy.setup_pheno_matter*), 14
SetupRadCh (*class in nucleardatapy.setup_rad_ch*), 29
sig_do (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
sig_do (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 37
sig_up (*nucleardatapy.setup_astro_masses.SetupAstroMasses attribute*), 35
sig_up (*nucleardatapy.setup_astro_mtot.SetupAstroMtot attribute*), 37
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
sm_gap (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 16
sm_kf (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 17
sm_kfn (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 17
sm_pre (*nucleardatapy.setup_pheno_matter.SetupPhenoMatter attribute*), 17

T

table (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 32
tables_isgmr() (*in module nucleardatapy.setup_ISGMR*), 32
tables_masses_exp() (*in module nucleardatapy.setup_masses_exp*), 27
tables_masses_theory() (*in module nucleardatapy.setup_masses_theory*), 29
tables_rad_ch() (*in module nucleardatapy.setup_rad_ch*), 31

V

versions_masses_exp() (*in module nucleardatapy.setup_masses_exp*), 27

X

xn (*nucleardatapy.setup_crust.SetupCrust attribute*), 24
xn_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 24
xp (*nucleardatapy.setup_crust.SetupCrust attribute*), 24
xpn_bound (*nucleardatapy.setup_crust.SetupCrust attribute*), 24

Z

Z (*nucleardatapy.setup_crust.SetupCrust attribute*), 23
Zmax (*nucleardatapy.setup_masses_exp.SetupMassesExp attribute*), 25