nucleardatapy

Release 0.1

Jérôme Margueron, IRL NPA, USA

Jun 27, 2024

CONTENTS

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.

CONTENTS

1.1 Usage

1.1.1 Installation

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

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

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

```
$ unzip nucleardatapy.zip
```

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

```
$ cd nucleardatapy
```

and launch the install script:

```
$ bash install.sh
```

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

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

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

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

1.1.2 Use nucleardatapy

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

```
$ python3 sample_SetupMicro.py
```

1.1.3 Test nucleardatapy

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

```
$ bash run_tests.sh
```

1.1.4 Get started

How to obtain microscopic results for APR equation of state:

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

import nucleardatapy as nuda

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

mic.print_outputs( )
```

1.2 API

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

1.2.1 nucleardatapy

This module provides microscopic, phenomenological and experimental data constraints.

1.3 Miscelaneous

1.3.1 Contributing

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

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

will work in the same part of the code at the same time (i.e. between a pull and a push). It is probably the simpler way to proceed.

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

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

Two ways:

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

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

Provide your data:

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

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

Join the team:

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

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

1.3.2 License

TBC.

1.3.3 Report issues

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

1.3.4 Thanks

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

COMPLEMENT

2.1 SetupMicro

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

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

This choice is defined in *model*, which can chosen among the following choices: '1998-VAR-AM-APR', '2008-AFDMC-NM', '2008-QMC-NM-swave', '2008-QMC-NM-AV4', '2009-dQMC-NM', '2010-NM-Hebeler', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2020-MBPT-AM-DHSL59', '2020-MBPT-AM-DHSL69', '2023-MBPT-AM'.

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

Attributes:

label

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

model

> Attribute model.

nm_den

> Attribute neutron matter density.

nm_e2a

> Attribute neutron matter energy per particle.

nm_e2a_err

> Attribute uncertainty in the neutron matter energy per particle.

nm_gap

> Attribute neutron matter pairing gap.

nm_kfn

> Attribute neutron matter Fermi momentum.

nm_pre

> Attribute neutron matter pressure.

nm_pre_err

> Attribute uncertainty in the neutron matter pressure.

note

Attribute providing additional notes about the data.

print_outputs()

Method which print outputs on terminal's screen.

ref

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

sm_den

Attribute symmetric matter density.

sm_e2a

Attribute symmetric matter energy per particle.

sm_e2a_err

Attribute uncertainty in the symmetric matter energy per particle.

sm_gap

Attribute symmetric matter pairing gap.

sm_kfn

Attribute symmetric matter Fermi momentum.

sm_pre

Attribute symmetric matter pressure.

sm_pre_err

Attribute uncertainty in the symmetric matter pressure.

nucleardatapy.setup_micro.**models_micro()**

Return a list with the name of the models available in this toolkit and print them all on the prompt. These models are the following ones: '1998-VAR-AM-APR', '2008-AFDMC-NM', '2008-QMC-NM-swave', '2008-QMC-NM-AV4', '2009-dQMC-NM', '2010-NM-Hebeler', '2013-QMC-NM', '2014-AFQMC-NM', '2016-QMC-NM', '2016-MBPT-AM', '2018-QMC-NM', '2020-MBPT-AM-DHSL59', '2020-MBPT-AM-DHSL69', '2023-MBPT-AM'.

> **Returns**
>
> The list of models.
>
> **Return type**
>
> list[str].

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

2.2 SetupPheno

class nucleardatapy.setup_pheno.**SetupPheno**(*model='Skyrme'*, *param='SLY5'*)

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

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

If *models* == 'NLRH', *param* can be: 'NL-SH', 'NL3', 'NL3II', 'PK1', 'PK1R', 'TM1'.
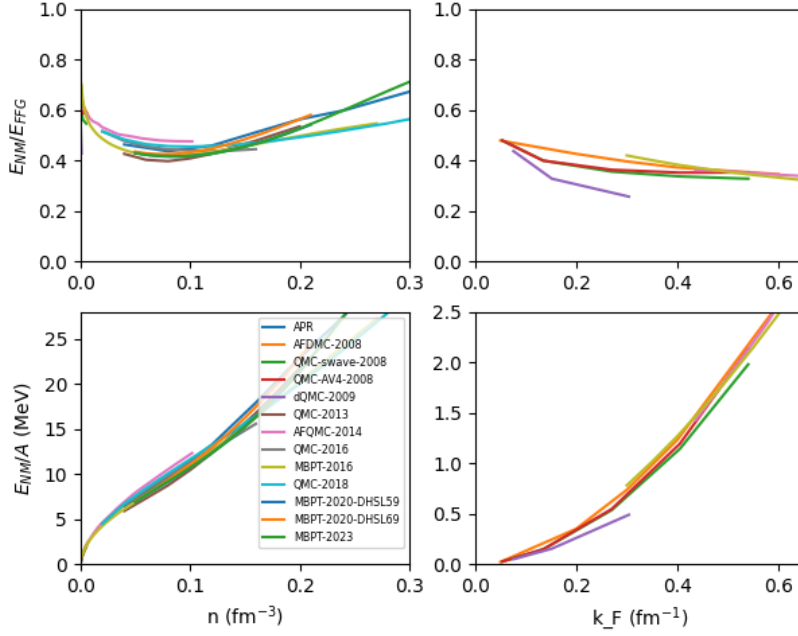
Fig. 1: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of model available in the nucleardatapy toolkit.
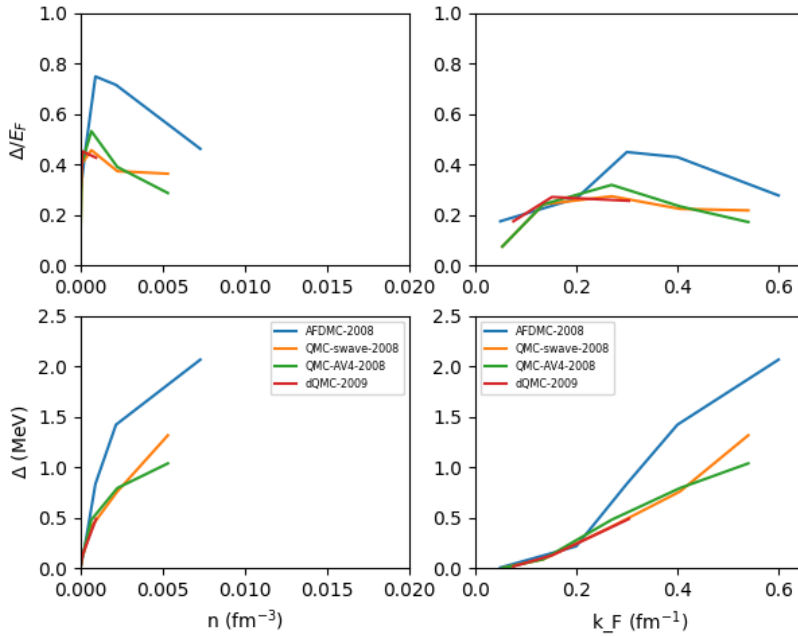

Fig. 2: This figure shows the pairing gap in neutron matter (NM) over the Fermi energy (top) and the pairing gap (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the models available in the nucleardatapy toolkit.

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

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

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

Attributes:

label

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

model

> Attribute model.

nm_cs2

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

nm_den

> Attribute neutron matter density.

nm_e2a

> Attribute neutron matter energy per particle.

nm_gap

> Attribute neutron matter pairing gap.

nm_kfn

> Attribute neutron matter Fermi momentum.

nm_pre

> Attribute neutron matter pressure.

note

> Attribute providing additional notes about the data.

param

> Attribute param.

print_outputs()

> Method which print outputs on terminal's screen.

ref

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

sm_cs2

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

sm_den

> Attribute symmetric matter density.

sm_e2a

> Attribute symmetric matter energy per particle.

`sm_gap`

   Attribute symmetric matter pairing gap.

`sm_kfn`

   Attribute symmetric matter Fermi momentum.

`sm_pre`

   Attribute symmetric matter pressure.

`nucleardatapy.setup_pheno.`**`models_pheno`**`()`

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

> **Returns**
>
>   The list of models with can be 'Skyrme', 'NLRH', 'DDRH', 'DDRHF'.
>
> **Return type**
>
>   list[str].

`nucleardatapy.setup_pheno.`**`params_pheno`**`(`*model*`)`

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

> **Parameters**
>
>   **`model`**`(`*str.*`)` – The type of model for which there are parametrizations. They should be chosen among the following options: 'Skyrme', 'NLRH', 'DDRH', 'DDRHF'.
>
> **Returns**
>
>   The list of parametrizations. If *models* == 'skyrme': 'BSK14', 'BSK16', 'BSK17', 'BSK27', 'F-', 'F+', 'F0', 'FPL', 'LNS', 'LNS1', 'LNS5', 'NRAPR', 'RATP', 'SAMI', 'SGII', 'SIII', 'SKGSIGMA', 'SKI2', 'SKI4', 'SKMP', 'SKMS', 'SKO', 'SKOP', 'SKP', 'SKRSIGMA', 'SKX', 'Skz2', 'SLY4', 'SLY5', 'SLY230A', 'SLY230B', 'SV', 'T6', 'T44', 'UNEDF0', 'UNEDF1'. If *models* == 'NLRH': 'NL-SH', 'NL3', 'NL3II', 'PK1', 'PK1R', 'TM1'. If *models* == 'DDRH': 'DDME1', 'DDME2', 'DDMEd', 'PKDD', 'TW99'. If *models* == 'DDRHF': 'PKA1', 'PKO1', 'PKO2', 'PKO3'.
>
> **Return type**
>
>   list[str].

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

2.3 SetupEsymLsym

`class` `nucleardatapy.setup_EsymLsym.`**`SetupEsymLsym`**`(`*constraint='2014-IAS'*`)`

   Instantiate the values of Esym and Lsym from the constraint.

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

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

Attributes:
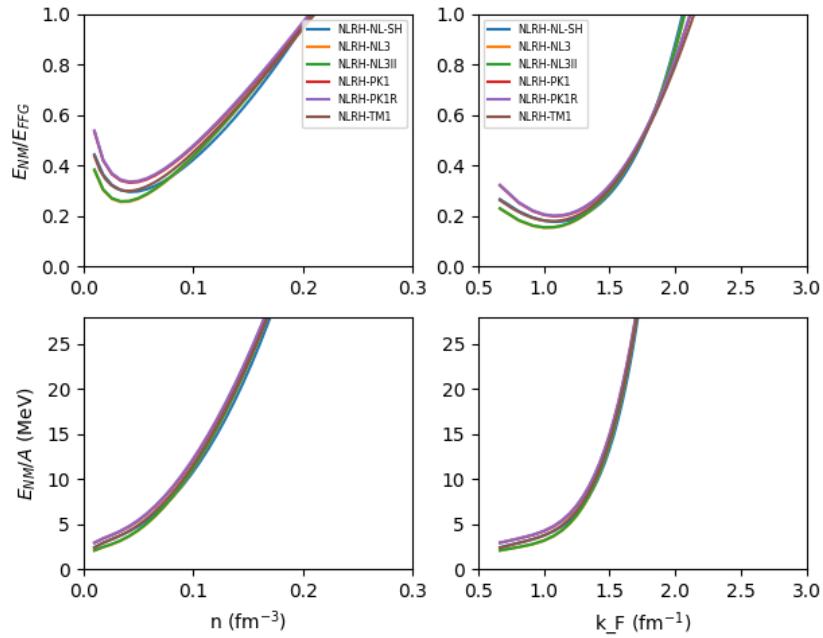
`Esym`

   Attribute Esym.

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 complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.
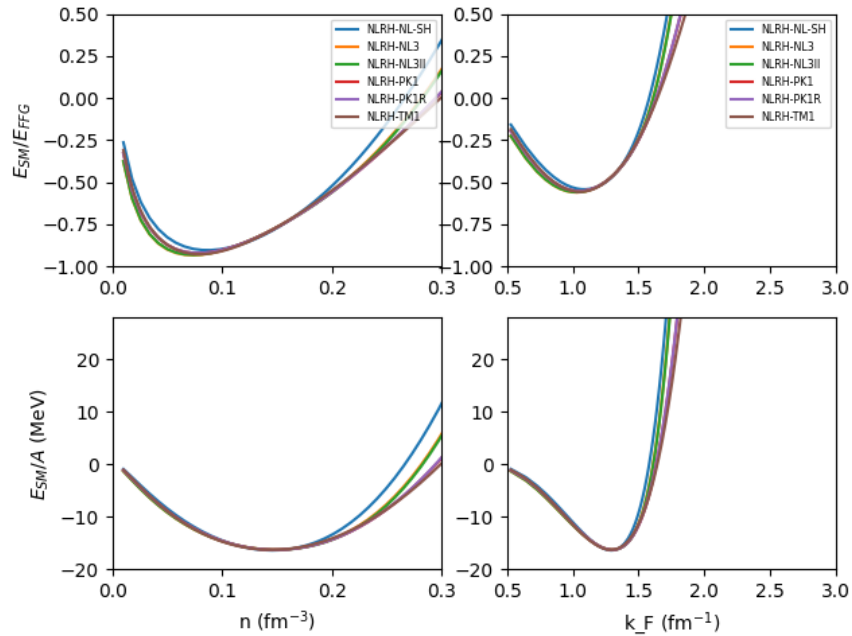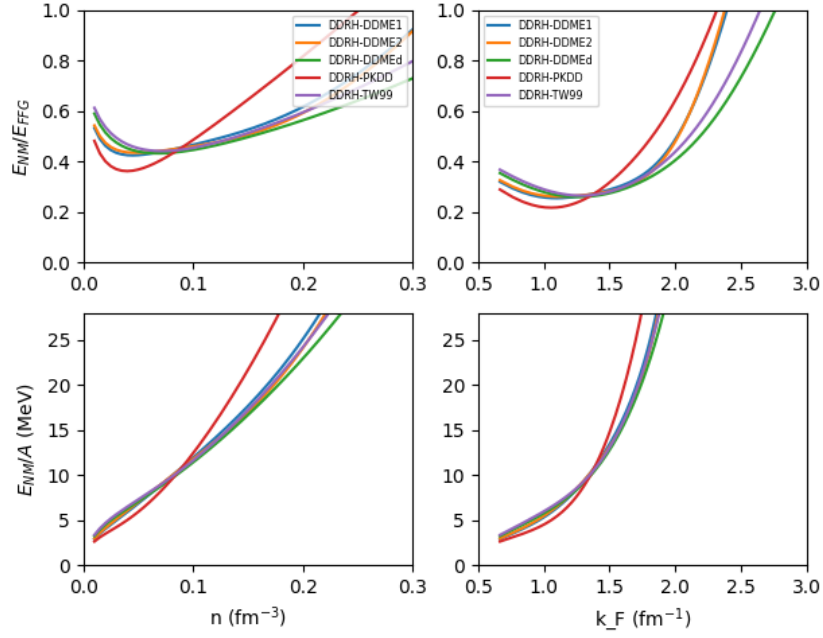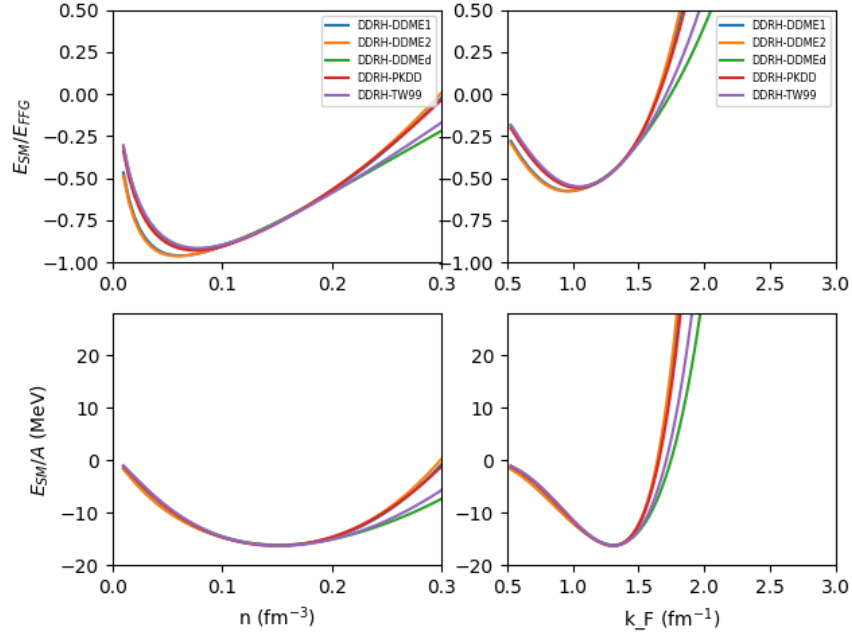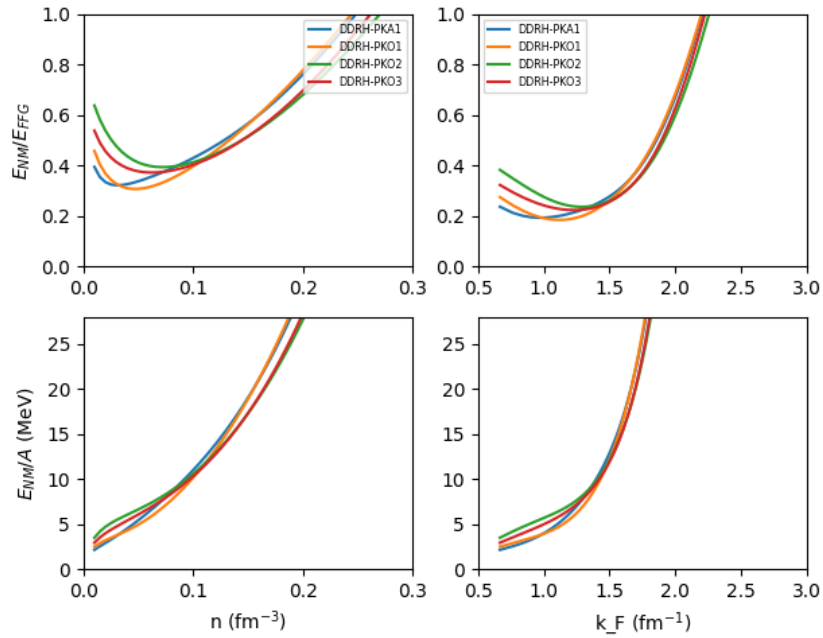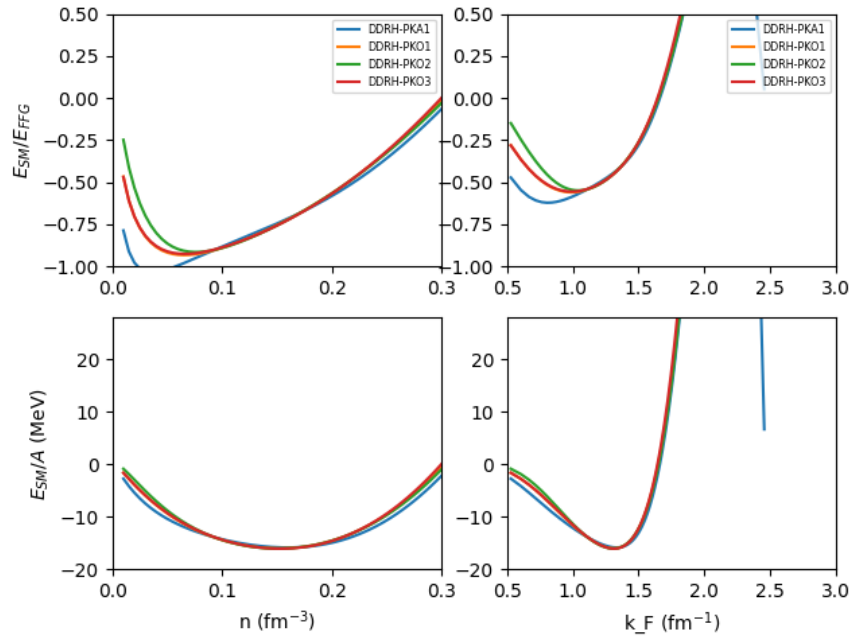

Fig. 4: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on non-linear meson(s) relativistic Hartree (NLRH) approach available in the nucleardatapy toolkit.
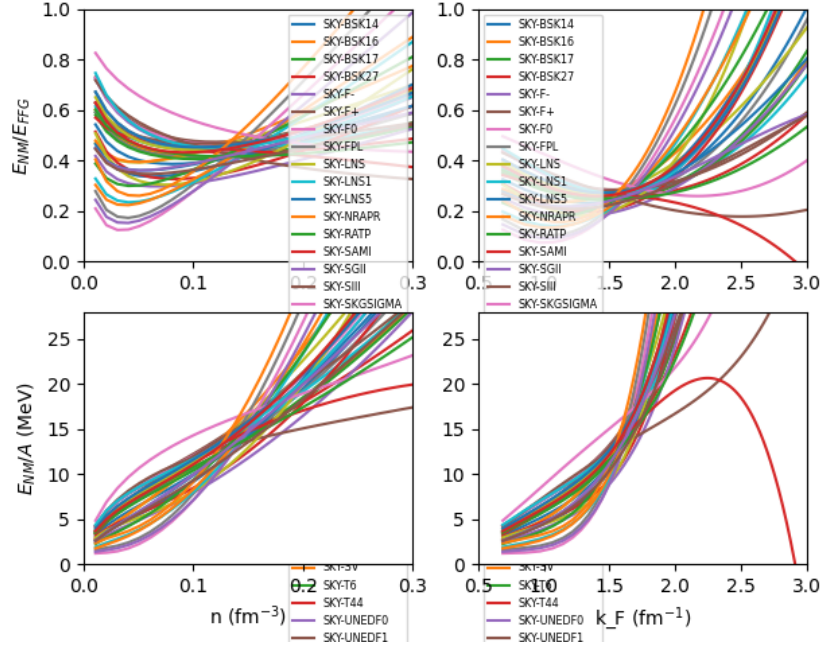
Fig. 5: This figure shows the energy in neutron matter (NM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree (DDRH) approach available in the nucleardatapy toolkit.
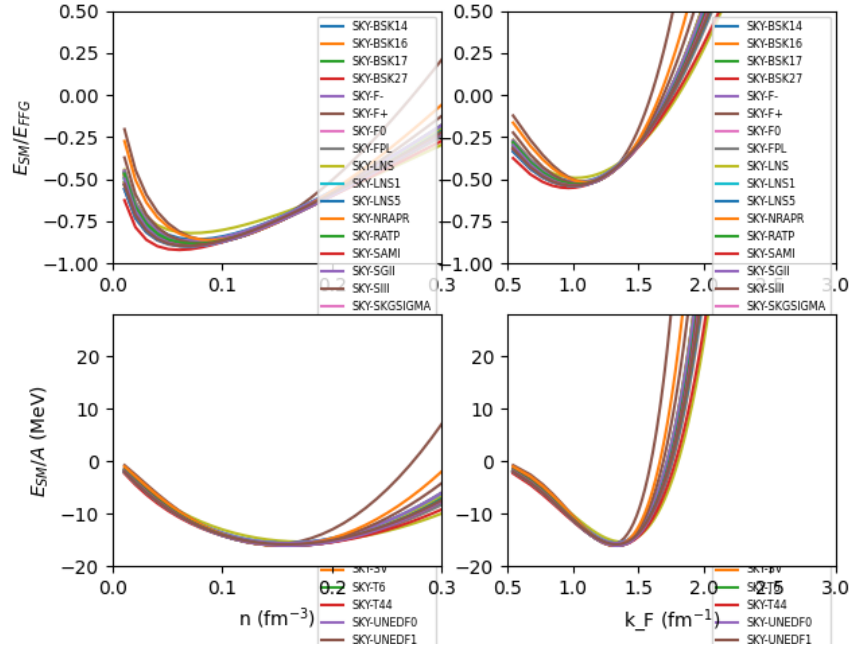

Fig. 6: This figure shows the energy in symmetric matter (SM) over the free Fermi gas energy (top) and the energy per particle (bottom) as function of the density (left) and the neutron Fermi momentum (right) for the complete list of phenomenological models based on density-dependent relativistic Hartree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

Fig. 7: 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. 8: 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. 9: 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. 10: 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.

Esym_err

> Attribute with uncertainty in Esym.

Lsym

> Attribute Lsym.

Lsym_err

> Attribute with uncertainty in Lsym.

constraint

> Attribute constraint.

label

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

note

> Attribute providing additional notes about the constraint.

print_outputs()

> Method which print outputs on terminal's screen.

ref

> Attribute providing the full reference to the paper to be 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', '2021-PREXII-Zhang'; and print them all on the prompt.
>
> > **Returns**
> >
> > > The list of constraints.
> >
> > **Return type**
> >
> > > list[str].

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

2.4 SetupMasses

class nucleardatapy.setup_masses.**SetupMasses**(*table='AME'*, *version='2020'*)

> Instantiate the experimental nuclear masses from AME mass table.
>
> This choice is defined in the variables *table* and *version*.
>
> *table* can chosen among the following ones: 'AME'.
>
> *version* can be chosen among the following choices: '2020', '2016', '2012'.
>
> > **Parameters**
> >
> > > - **table** (`str`, `optional.`) – Fix the name of *table*. Default value: 'AME'.
> > > - **version** (`str`, `optional.`) – Fix the name of *version*. Default value: 2020'.
>
> **Attributes:**
>
> **A**
>
> > Attribute A (mass of the nucleus).
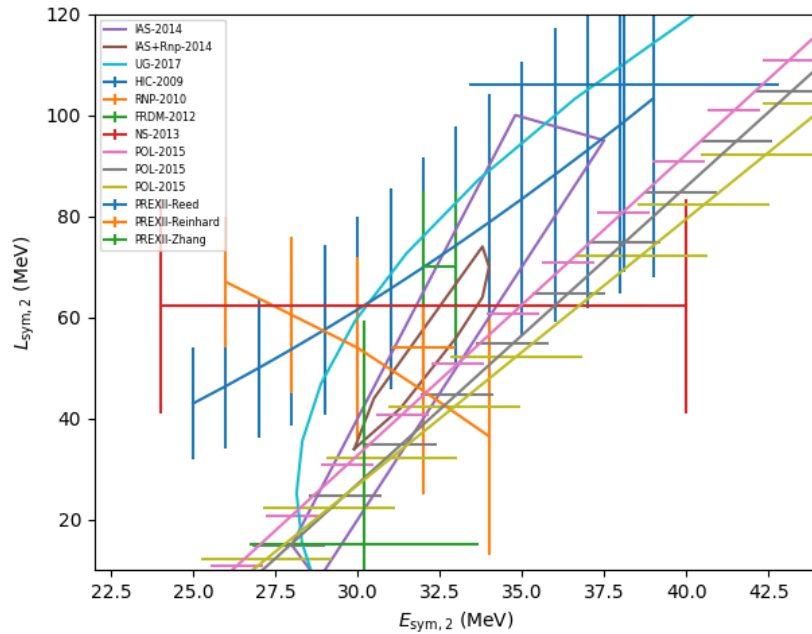
Fig. 11: This figure shows the Esym,2 versus Lsym,2 correlation for the different constraints availble in the nuclear-datapy toolkit.

BE

>   Attribute BE (Binding Energy) of the nucleus.

BE_err

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

HT

>   Attribute HT (half-Time) of the nucleus.

I

>   Attribute I.

Interp

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

N

>   Attribute N (number of neutrons of the nucleus).

Z

>   Attribute Z (charge of the nucleus).

Zmax

>   maximum charge of nuclei present in the table.

> >   **Type**
> >   >   Attribute Zmax

drip(*Zmax=95*)

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

Parameters

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

label

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

nbLine

> Attribute with the number of line in the file.

nbNuc

> Attribute with the number of nuclei read in the file.

note

> Attribute providing additional notes about the data.

print_outputs()

> Method which print outputs on terminal's screen.

ref

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

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

> Method which select some nuclei from the table according to some criteria.
>
> **Parameters**
>
> - **interp** (`str, optional. Default = 'n'.`) – If interp='n', exclude the interpolated nuclei from the selected ones. If interp='y' consider them in the table, in addition to the others.
>
> - **state** (`str, optional. Default 'gs'.`) – select the kind of state. If state='gs', select nuclei measured in their ground state.
>
> - **nucleus** (`str, optional. Default 'unstable'.`) – 'unstable'.
>
> - **every** (`int, optional. Default every = 1.`) – consider only 1 out of *every* nuclei in the table.

stbl

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

symb

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

year

> Attribute year of the discovery of the nucleus.

nucleardatapy.setup_masses.**tables_masses**()

> Return a list of the tables available in this toolkit for the experimental masses and print them all on the prompt. These tables are the following ones: 'AME'.
>
> **Returns**
>> The list of tables.
>
> **Return type**
>> list[str].

nucleardatapy.setup_masses.`versions_masses`(*table*)

> Return a list of versions of tables available in this toolkit for a given model and print them all on the prompt.
>
> > **Parameters**
> > > **table** (`str.`) – The table for which there are different versions.
> >
> > **Returns**
> > > The list of versions. If table == 'AME': '2020', '2016', '2012'.
> >
> > **Return type**
> > > list[str].

Here are a set of figures which are produced with the Python sample: /sample/nucleardatapy_plots/plot_setupMasses.py
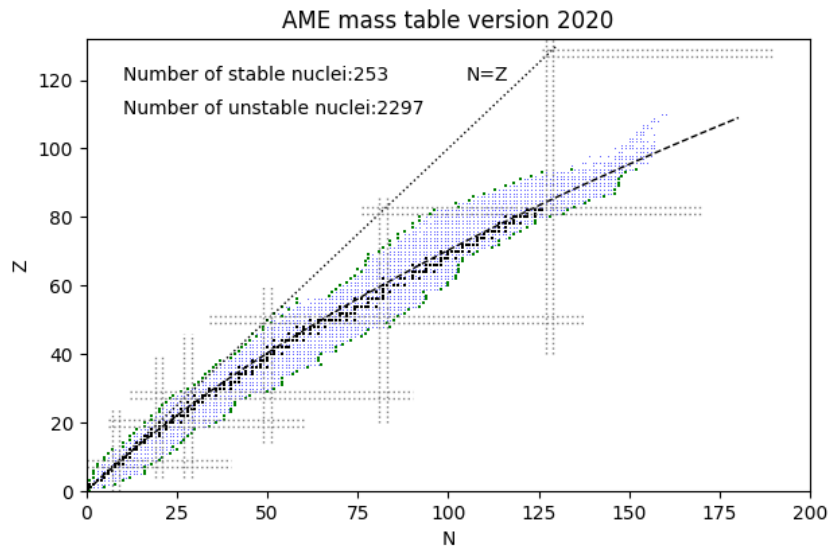
Fig. 12: This figure shows the nuclear chart based on AME 2020 table.

2.5 SetupRadCh

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

> Instantiate the object with charge radii choosen from a table.
>
> This choice is defined in the variable *table*.
>
> The tables can chosen among the following ones: '2013-Angeli'.
>
> > **Parameters**
> > > **table** (`str, optional.`) – Fix the name of *table*. Default value: '2013-Angeli'.
>
> **Attributes:**
>
> **A**
>
> > Attribute A (mass of the nucleus).

N

>   Attribute N (number of neutrons of the nucleus).

R_ch

>   Attribue R_ch (charge radius) in fm.

R_ch_err

>   Attribue uncertainty in R_ch (charge radius) in fm.

R_unit

>   Attribute radius unit.

Z

>   Attribute Z (charge of the nucleus).

label

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

note

>   Attribute providing additional notes about the data.

print_outputs()

>   Method which print outputs on terminal's screen.

ref

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

symb

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

nucleardatapy.setup_rad_ch.**tables_rad_ch**()

>   Return a list of the tables available in this toolkit for the charge 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].

2.6 SetupISGMR

class nucleardatapy.setup_ISGMR.**SetupISGMR**(*table='2018-ISGMR-GARG'*)

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

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

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

>   **Attributes:**

>   **A**

>   >   Attribute A (mass of the nucleus).

E_cen

    Attribute energy centroid.

E_erra

    Attribute list with + and - uncertainty

E_errm

    Attribute (-) uncertainty in the energy centroid.

E_errp

    Attribute (+) uncertainty in the energy centroid.

E_errs

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

E_unit

    Attribute energy unit.

Z

    Attribute Z (charge of the nucleus).

label

    Attribute providing the label the data is references for figures.

note

    Attribute providing additional notes about the data.

print_outputs()

    Method which print outputs on terminal's screen.

ref

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

table

    Attribute table.

nucleardatapy.setup_ISGMR.**tables_isgmr**()

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

    Returns

        The list of tables.

    Return type

        list[str].

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

N

N (*nucleardatapy.setup_masses.SetupMasses attribute*), 17

N (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 19

nbLine (*nucleardatapy.setup_masses.SetupMasses attribute*), 18

nbNuc (*nucleardatapy.setup_masses.SetupMasses attribute*), 18

nm_cs2 (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

nm_den (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

nm_den (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

nm_e2a (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

nm_e2a (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

nm_e2a_err (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

nm_gap (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

nm_gap (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

nm_kfn (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

nm_kfn (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

nm_pre (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

nm_pre (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

nm_pre_err (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

note (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 16

note (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 21

note (*nucleardatapy.setup_masses.SetupMasses attribute*), 18

note (*nucleardatapy.setup_micro.SetupMicro attribute*), 7

note (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

note (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 20

nucleardatapy
module, 4

nucleardatapy.setup_EsymLsym
module, 11

nucleardatapy.setup_ISGMR
module, 20

nucleardatapy.setup_masses
module, 16

nucleardatapy.setup_micro
module, 7

nucleardatapy.setup_pheno
module, 8

nucleardatapy.setup_rad_ch
module, 19

P

param (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

params_pheno() (*in module nucleardatapy.setup_pheno*), 11

print_outputs() (*nucleardatapy.setup_EsymLsym.SetupEsymLsym method*), 16

print_outputs() (*nucleardatapy.setup_ISGMR.SetupISGMR method*), 21

print_outputs() (*nucleardatapy.setup_masses.SetupMasses method*), 18

print_outputs() (*nucleardatapy.setup_micro.SetupMicro method*), 8

print_outputs() (*nucleardatapy.setup_pheno.SetupPheno method*), 10

print_outputs() (*nucleardatapy.setup_rad_ch.SetupRadCh method*), 20

R

R_ch (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 20

R_ch_err (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 20

R_unit (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 20

ref (*nucleardatapy.setup_EsymLsym.SetupEsymLsym attribute*), 16

ref (*nucleardatapy.setup_ISGMR.SetupISGMR attribute*), 21

ref (*nucleardatapy.setup_masses.SetupMasses attribute*), 18

ref (*nucleardatapy.setup_micro.SetupMicro attribute*), 8

ref (*nucleardatapy.setup_pheno.SetupPheno attribute*), 10

ref (*nucleardatapy.setup_rad_ch.SetupRadCh attribute*), 20