Appendix B.

Numerical Workflows – RNSA

To support reproducible research all the the scripts developed for this Thesis are publicly available and can be found on GitHub [64] with the name RNSA (RNS Analyzer) and license GNU GPL v3 [98]. RNSA has been built for GNU/Linux environments, but some scripts may also work on different platforms, provided that the minimum requirements are available and are correctly setup. The following pieces of software are needed:

- Python $\geq$ 3.4.0
- NumPy $\geq$ 1.12
- matplotlib $\geq$ 2.0.0
- matplotlib2tikz $\geq$ 0.6.2
- LaTeX $\geq$ 3.14159265
- gawk $\geq$ 4.1.1
- RNS $\geq$ 4.0
- HTCondor $\geq$ 8.4.0

This appendix is meant to be a documentation for using RNSA and thus for reproducing the results presented in this Thesis.

In the following, the symbol $\star$ indicates that the argument is required to run the script and the symbol $ means that the command should be executed in a shell.

All the scripts have an inline help which provides a brief description of all the different flags and arguments that the program supports. To print this help it suffices to run the command with the flag `-h` or `--help`.

B.1. Equilibrium Models

Constructing equilibrium sequences is only the first step toward the study of the stability properties of rotating neutron stars by means of the Sorkin's theorem and much more analysis is required. RNSA provides the tools to accomplish both the tasks: construction and study of equilibrium solutions. The suite is mainly composed by five scripts that have to be run in the following order:

1. `condor_parser` to compute equilibrium models using HTCondor and RNS
2. `extractor` to produce *plot-friendly* data files
3. `plotter` to plot the data and find the turning points
4. `instabilizer` to compute with greater accuracy the turning points's models
5. `instaplotter` to plot physical quantities for solutions on the turning points's line

In addition, there is another useful script called `instacomparer` that plots every possible combination of physical quantities for models computed with `instabilizer`. This script provides a clear insight of the reciprocal dependencies between the various quantities making simpler to find new interesting relations as the ones of section 2.6.2.

The following sections provide simple user guides for the various scripts and in some case explain details on the algorithm used, for example to find the turning points.

B.1.1. RNS

RNS computes equilibrium models for rapidly rotating relativistic compact stars. Some previous releases are open source and are available at the webpage [99], where a detailed manual can be found. Full support to the differential rotation (one- and three-parameters laws) has been added to RNS in the version `4.0` which is not publicly available yet.

Check this if RNS becomes part of ET

In this section a very minimal user guide for RNS 4.0 is presented covering only the features used in the Thesis.

RNS is a `C` program that can be compiled with the included `Makefile` on most modern systems, provided that a `C` compiler is available (as the GNU `C` compiler `gcc` or the Intel `C` compiler `icc`). The compilation produces an executable `rns` that has to be run via command line with the parameters of the equilibrium model to be computed specified through appropriate flags.

RNS 4.0 requires a tabulated equation of state in a specific format which can be constructed using the utility `HnG` included with the code. This latter takes as input a two-columns table, the first one is the central energy density (in $\mathrm{g\,cm^{-3}}$) and the second one is the pressure (in $\mathrm{dynes\,cm^{-3}}$), and produce a table formatted in the proper way for RNS 4.0. Moreover, an utility `eosify` is included in RNSA to convert an EOS table from the format compatible with the previous versions of RNS with the one required for latest release.

The EOS file has to be specified with the flag `-f` as:

`-f` *path of the* EOS *table*

It is also possible to work with polytropic equations of state, to do this the flag `-q poly` has to be used, with the desired index specified with `-N`:

`-N` *index of the polytropic* EOS

The second required parameter to be supplied is the central energy density that is set with `-e` and with units $\mathrm{g\,cm^{-3}}$ or dimensionless units for polytropic EOS:

`-e` *central energy density* ϵ_c

If not otherwise specified RNS computes the uniformly rotating model and differential rotation has to be enabled with:

```
-R diff
```

RNS 4.0 supports both the one-parameter law (section 2.6) and the three-parameters law (section 2.7) and uses the former or the latter depending on the other given parameters. To compute a solution with the one-parameter the value of $\hat{A}$ has to be set with the flag `-A`.

`-A` *rotation parameter* $\hat{A}$

To enable the 3-parameters differential rotation in addition to `-A` there are two more parameters that have to be given: $\hat{A}_2^2$ which is set with the flag `-D`[1] and β with the flag `-b`.

`-A` *rotation parameter* $\hat{A}_1$

`-D` *rotation parameter* $\hat{A}_2^2$

`-b` *rotation parameter* β

Finally, to fully specify a model a `task` must be chosen (which amounts to specify another parameter). RNS supports the following tasks:

Rewrite the following part

- With the flag `-r 1` the static spherical (TOV) model is computed
- With the flag `-s 3` the mass-shedding model is computed
- With the flag `-s 1` a model with fixed rest mass is computed. To specify the mass (in solar masses) the flag `-p` is used.
- With the flag `-s 2` a model with fixed angular is computed. To specify the angular (in dimensionless or polytropic units) the flag `-p` is used.

Moreover, it is usually convenient to reduce the verbosity of the program with the flag `-d 0` so that the output is only a single line with the physical properties of the solution. In this form the output is easily manipulable with scripts.

`-d` *level of verbosity (0 or 1, defualt: 1)*

RNS 4.0 has a large number of features that have not been mentioned in this minimal guide because they have not been used in this Thesis but a complete explanation can be found in the official manual. Among the other parameters that can be set to configure RNS there are as example the accuracy (settable with `-a`), the tolerance in finding the desired value of the physical quantity (that can be set with `-x`) or a relaxation factor. This latter parameter is set with `-c` and takes values from 0 to 1 and can help the convergence for unstable models or very stiff equations of state.

`-a` *accuracy (default:* 10^{-7}*)*

[1] Attention: the flag `-A` sets the parameter $\hat{A}_1$ but `-D` sets the *square* of $\hat{A}_2$!

`-x` *tolerance (default:* 10^{-4}*)*

`-x` *relaxation factor (default:* 1.0*, no relaxation)*

A complete example to compute a nonrotating model with EOS C and central energy density $3 \times 10^{15}\,\mathrm{g\,cm^{-3}}$ is:

```
$ rns -f eosC -e 3e15 -d 0 -r 1
```

With rest mass $2.0\,M_\odot$, and uniformly rotating:

```
$ rns -f eosC -e 3e15 -d 0 -s 1 -p 2
```

Enabling the differential rotation with the one-parameter law and the parameter $\hat{A} = 1.0$:

```
$ rns -f eosC -R diff -A 1 -e 3e15 -d 0 -s 1 -p 2
```

Using the three-parameters law with $\hat{A}_1 = 1.0$ and $\hat{A}_2 = 1.5$ and $\beta = 0.6$:

```
$ rns -f eosC -R diff -A 1 -D 2.25 -b 0.6 -e 3e15 -d 0 -s 1 -p 2
```

To compute a differentially rotating polytrope:

```
$ rns -q poly -N 1 -e 0.42 -d 0 -r 0.75 -A 1
```

B.1.2. condor_parser

Although RNS is a computational inexpensive code, a large number of models has to be computed to sample the parameter space with a fine mesh. For this reason, it is convenient to run RNS on as many CPU cores as possible, where each core computes a single model. For this Thesis HTCondor [100] has been used to manage the distribution of the jobs among the cores of the LCM cluster in Milan. Most importantly, HTCondor overcomes a problem in automating RNS: to compute a sequence with RNS it is necessary to know in which energy range models can be found and trying to use RNS to construct a sequence in a larger range leads to no results at all. This is a major issue because there is no straightforward way to determine that range. It is possible to avoid the problem with HTCondor: RNS is used to find single models with defined energy, and then one constructs sequences with only the converging models. In this way, RNS is used to compute solutions in a fixed range of energies and when it fails to converge because it reaches the maximum number of iterations or time it means that for that energy the model is not defined. For example, if HTCondor is instructed to use RNS to calculate solutions with energy between $0.5 \times 10^{15}\,\mathrm{g\,cm^{-3}}$ and $5 \times 10^{15}\,\mathrm{g\,cm^{-3}}$ it produces results only if the energy is $\epsilon_c \in (1 \times 10^{15}\,\mathrm{g\,cm^{-3}}, 3 \times 10^{15}\,\mathrm{g\,cm^{-3}})$ which means that outside this range no equilibrium model can be found. In this way no knowledge *a priori* of the validity range is required, and hence it is possible to compute a large number of models with an automatized script.

RNSA includes a script called `condor_parser` that automatizes the procedure of preparing and submitting jobs to HTCondor making easy to construct of equilibrium sequences of any kind, eliminating the problem of having to find the range of definition of a sequence. The script supports many possible usages depending on the task that has to be accomplished but in most of the cases it suffices to combine different flags and arguments to successfully run `condor_parser`.

Before the very first execution there are two parameters that have to set manually inside the script:

- `exec_path`: is the full path of the RNS executable obtained by compiling the source code with `condor_compile`

- `eos_path`: is the full path of the directory that contains the tables for the equations of state.

Not setting these parameter yields an error. To run the script the name of the EOS has to be given. For tabulated equation of states this is done with the flag `-f`:

`-f` *name of* EOS *(inside the* `eos_path`*)*

For polytropic equations of state this is achieved by:

`-p -q` *index of the polytrope*

It is necessary also to specify what kind of sequences have to be constructed, which amounts to choosing a the `task` among the ones supported by RNS (section B.1.1). This is set with the flag `-t`:

`-t` *task (static, kepler, rmass, jmoment)*

Finally, the extremes of the energy range (in $\mathrm{g\,cm^{-3}}$ or polytropic units) and the number of sampling point are to be supplied with the flags `-e1`, `-e2` and `-n`:

`-e1` *lowest central energy density*

`-e2` *highest central energy density*

`-n` *number of sampling points*

In this Thesis sequences have been always sampled with 500 points.

If the specified task is `rmass` or `jmoment` the number of different values of this parameter has to be given with the flag `-N`:

`-N` *number of sequences*

Furthermore, the range of the parameter has to be specified with appropriate flags `-m1`, `-m2` for the mass (in solar masses) and `-j1`, `-j2` for the angular momentum (in dimensionless units or polytropic units).

`-m1` *lowest rest mass*

`-m2` *highest rest mass*

`-j1` *lowest angular momentum*

`-j2` *highest angular momentum*

If not specified, `condor_parser` assumes uniform rotation and appropriate flags have to be used for enabling differential rotation (table B.1). Another default behavior for `condor_parser` is to leave RNS running for 10 minutes before assuming that it has not converged and thus stopping it, but this time can customized editing the variable `periodic_remove` inside the code. It is possible that in some rare case where the convergence is extremely difficult this time limit produces a *false negative* result which means that it RNS had been left running for an indeterminate amount of time it would have converged. Despite this remote possibility, in most real-life cases if a large amount of models is sampled *losing* some point will not cause any harm. Empirically it has been found that with the machines used for this Thesis 10 minutes is enough time for producing most of the convergences but not too much time is wasted in non converging executions.

Fix this sentence!

The output produced by RNS with `condor_parser` is stored in folders with a nested structure that can be exploited to identify exactly which physical parameters have been used to obtain a certain result. If `condor_parser` finds that the output folder already exists it will delete it without asking. `condor_parser` can produce gigabytes of output if many sequences are to be computed, due to the fact that RNS calculate also the configuration of the star for using it in the Einstein Toolkit and it save it in folders called `Output_Star`.Those data are not needed to study equilibrium models and thus it is better to delete them, saving storage. An utility called `stardestroyer` included with RNSA can be left running in background: it will delete every folder named `Output_Star` every 15 minutes. Another trick used to save storage is to delete the `condor.submit` file for every sequence and replace it with a `command.submit` file which contains only the arguments given to RNS for that particular sequence and not the whole storage-consuming submit script.

An example of command that samples with 500 points a static sequence with equation of state L and central energy $\epsilon_c \in (5 \times 10^{14}\,\mathrm{g\,cm^{-3}}, 5 \times 10^{15}\,\mathrm{g\,cm^{-3}})$:

```
$ condor_parser -n 500 -e1 5e14 -e2 5e15 -f eosL -t static -c \
  -o eosL_static
```

To compute 16 sequences with fixed rest mass rotating uniformly:

```
$ condor_parser -n 500 -e1 5e14 -e2 5e15 -f eosL -t rmass \
  -N 16 -m1 2.0 -m2 3.5 -o eosLU -c
```

To construct sequences of constant rest mass differentially rotating with the one-parameter rotation law and different values of $\hat{A}$. With the one-parameter law the flags `-A1` and `-A2` define the range of parameters to be sampled with a number of points specified by `-na`:

```
$ condor_parser -N 16 -n 500 -e1 5e14 -e2 5e15 -f eosL -t rmass \
  -m1 2.0 -m2 3.5 -o eosLD -R -na 4 -A1 0.5 -A2 2.0 -c
```

`-A1` *lowest rotation parameter* $\hat{A}$

`-A2` *highest rotation parameter* $\hat{A}$

`-na` *number of sampling points for the rotation parameter* $\hat{A}$

An example to compute sequences of constant rest mass differentially rotating with the three-parameters rotation law and different values of the parameters:

```
$ condor_parser -c -n 500 -e1 5e14 -e2 5e15 -f eosL -t rmass \
  -N 8 -m1 2.0 -m2 3.5 -o eosLD3 -R3 -na1 4 -na2 4 -nb 4 \
  -A1i 0.5 -A1f 2.0 -A2i 0.5 -A2f 2.0 -B1 0.2 -B2 1.0
```

`-A1i` *lowest rotation parameter* $\hat{A}_1$

`-A1f` *highest rotation parameter* $\hat{A}_1$

`-na1` *number of sampling points for the rotation parameter* $\hat{A}_1$

`-A2i` *lowest rotation parameter* $\hat{A}_2$

`-A2i` *highest rotation parameter* $\hat{A}_2$

`-na2` *number of sampling points for the rotation parameter* $\hat{A}_2$

`-B1` *lowest rotation parameter* β

`-B2` *highest rotation parameter* β

`-nb` *number of sampling points for the rotation parameter* β

If the number of models (`-N`, `-na`, `-na1`, `-na2`, `-nb`) is not specified `condor_parser` assumes that is only one, so for example:

```
$ condor_parser -c -n 500 -e1 5e14 -e2 5e15 -f eosL -t rmass \
  -N 8 -m1 2.0 -m2 3.5 -o eosLD3 -R -A1 1.0 -A2 2.0
```

computes 8 sequences with constant rest mass differentially rotating with the parameter $\hat{A} = 1.0$ and the instruction `-A2 2.0` is ignored.

There is two important differences to be noted; first with `condor_parser` both the rotation parameters $\hat{A}_1$ and $\hat{A}_2$ have to be given not squared, and second for the three-parameters rotation law `-A1` and `-A2` refer to $\hat{A}_1$ and $\hat{A}_2$ where for the one parameter they refer to the lowest and highest values assumed by the only parameter $\hat{A}$.

There are some RNS's options, like tolerance accuracy, that can be configured inside the code. To do that it suffices to add the appropriate flags with the desired value to the variable `arguments` so that every job launched by HTCondor has that additional flags.

B.1.3. extractor

The output of `condor_parser` is stored `.out` files, which are one-line files with fifteen columns each one with a different physical quantity. Table B.3 shows which quantity is in every column and table B.2 explains the notation used.

The `.out` files are stored in folders with a tree structure: each level of the tree has a parameter fixed, which are in order: $\hat{A}_1$, $\hat{A}_2$, β, M_0 or J and finally the last level contains the `.out` files, one for every central energy in the specified range. In this way, any given folder of the tree contains one or more sequences of equilibrium models. This articulated structure is useful because given a `.out` file path it is possible to extract easily the parameters (EOS, $\hat{A}_1, \hat{A}_2, \beta, M_0(J), \epsilon_c$) of the single model. Parameters not used (like

Table B.1.: Arguments explanation for `condor_parser`. The star symbol indicates that the argument is required to run the script. The table is subdivided in three groups: the first group is for specifying many sequences with J or M_0 fixed uniformly rotating, the second for computing many groups of sequences differentially rotating with various values for the rotation parameter $\hat{A}$, and the third for the most general case with many groups of sequences with different values of the three parameters $\hat{A}_1$, $\hat{A}_2$ and β.

Argument	Quantity
`-f, --eos`	Equation of state name, eg. eosL
`-p, --poly`	Enable polytropic EOS
`-q, --index`	Polytropic index
`-t, --task` ⋆	Sequence to compute: `static` nonrotating `kepler` mass-shedding `rmass` with constant rest mass `jmoment` with constant angular momentum
`-o, --out`	Folder where to save output. If not provided the default name is based on the time
`-c, --condor`	Submit the jobs to HTCondor
`-n, --nmodels` ⋆	Number of models per sequence
`-e1, --energy1` ⋆	Lowest central energy density ϵ_c in g/cm²
`-e2, --energy2` ⋆	Highest central energy density ϵ_c in g/cm²
`-N, --nsequences`	Number of sequences
`-m1, --mass1`	Lowest rest mass M_0 in $M_\odot$
`-m2, --mass2`	Highest rest mass M_0 in $M_\odot$
`-j1, --jmoment1`	Lowest angular momentum J in $c/M_\odot^2$
`-j2, --jmoment2`	Highest angular momentum J in $c/M_\odot^2$
`-R, --diff`	Enable differential rotation (one-parameter)
`-na, --namodels`	Number of different values of $\hat{A}$
`-A1, --rotation1`	Lowest rotation parameter $\hat{A}$
`-A2, --rotation2`	Highest rotation parameter $\hat{A}$
`-R3, --diff3`	Enable differential rotation (three-parameters)
`-na1, --namodels1`	Number of different values of $\hat{A}_1$
`-A1i, --a1initial`	Lowest rotation parameter $\hat{A}_1$
`-A1f, --a1final`	Highest rotation parameter $\hat{A}_1$
`-na2, --namodels2`	Number of different values of $\hat{A}_2$
`-A2i, --a2initial`	Lowest rotation parameter $\hat{A}_2$
`-A2f, --a2final`	Highest rotation parameter $\hat{A}_2$
`-nb, --nbmodels`	Number of different values of β
`-B1, --beta1`	Lowest rotation parameter β
`-B2, --beta2`	Highest rotation parameter β

$\hat{A}_1, \hat{A}_2$ and β for uniform rotation) are set to zero. This structure is used for every model besides the static solution that is treated differently because it does not require a such involved structure.

This structure is too complicated to be handled directly so it is convenient to gather the physical data of interest corresponding a single sequence (with varying central energy) in single files. To do this task it is possible to use the `extractor` script, which extracts data of interest from the single `.out` files and gather them in `.dat` files for each sequence, one for each desired quantity, such as `gmass.dat` for the gravitational mass.[2]

To run `extractor` at least one target folder has to be given with the flag `-f`:

`-f` *path of the target folder 1, path of the target folder 2, ...*

The quantities to be extracted have to be specified with appropriate flags (table B.4). An example of usage is:

```
$ extractor -f eosC eosL -e -m
```

This command produces a single `energy.dat` and `gmass.dat` for each sequence for the data found in the folders `eosC` and `eosL`. Since `extractor` takes care of invalid files (such the one that are produced in case of maximum number of iteration reached) or empty files (such the one that are produces in case of convergence not reached) the files produces with `extractor` are ready to be plotted.

To delete every table produced with `extractor` there is a `clean` command which acts on the folder specified with the `-f` flags, for example:

```
$ extractor -f eosC eosL -c
```

If `extractor` finds already existing tables it overwrites them without asking.

[2]This step could be avoided if one extracts the desired data only before plotting them (in the `plotter` script), but this latter way would imply having to deal with many thousands of files and the input/output overhead would produce noticeable decrease in the performances.

Table B.2.: The quantities with a shorthand are tho ones that can be plotted with the plotting scripts and the abbreviation are the ones conventionally used in RNSA. GRV2 and GRV3 are a relativistic generalizations of the virial theorem valid for arbitrary asymptotically flat spacetimes [101–103]. They are routinely used for checking the accuracy of numerical methods.

Quantity	Symbol	Shorthand	Unit
Central energy density	ϵ_c/c^2	`energy`	$[10^{15}\,\mathrm{g\,cm^{-3}}]$
Maximal energy density	ϵ_{max}/c^2	`maxenergy`	$[10^{15}\,\mathrm{g\,cm^{-3}}]$
Gravitational mass	M	`gmass`	$[M_\odot]$
Rest mass	M_0	`rmass`	$[M_\odot]$
Angular momentum	J	`jmoment`	$[c/GM_\odot{}^2]$
Rotational binding ratio	$T/\lvert W\rvert$	`twratio`	[ ]
Central angular velocity	Ω_c	`comega`	$[\mathrm{s^{-1}}]$
Maximal angular velocity	Ω_{max}	`maxomega`	$[\mathrm{s^{-1}}]$
Equatorial angular velocity	Ω_e	`eomega`	$[\mathrm{s^{-1}}]$
Kepler angular velocity	Ω_K		$[\mathrm{s^{-1}}]$
Equatorial radius	R_e	`radius`	[km]
Equatorial coordinate radius	r_e		[km]
Two-dimensional virial identity	GRV2$/c^2$		$[10^{15}\,\mathrm{g\,cm^{-3}}]$
Three-dimensional virial identity	GRV3$/c^2$		$[10^{15}\,\mathrm{g\,cm^{-3}}]$
Polar equatorial ratio	r	`rratio`	[ ]

Table B.3.: Example of `.out` file, that is `STDOUT` of RNS when the flag `-d 0` is given. The file is only on a single row but for typographical convenience it is presented spitted on two rows. The meaning and the units are listed in table B.2. RNS produces also a detailed output of the configuration of the solution and it saves it inside the folder `Output_Star`, which contains useful data such as the tables of rest mass density, angular velocity, enthalpy as a function of the radius.

	ϵ_c	$\epsilon_{\max}$	M	M_0	J	$T/\lvert W\rvert$	Ω_c	$\Omega_{\max}$	...
...	Ω_e	Ω_K	R_e	r_e	GRV2	GRV3	r		

Table B.4.: Arguments explanation for `extractor` with the list of all the quantities that is possible to extract. The star symbol indicates that the argument is required to run the script. If `extractor` finds already existing tables it overwrites them without asking. The script parse the output produced by `condor_parser` to produce tables of the physical quantities of single sequences.

Argument	Quantity
`-f, --files` $\star$	Folders where data are stored
`-c, --clean`	Delete existing `.dat` files
`-e, --energy`	Extract central energy density
`-em, --maxenergy`	Extract max energy density
`-m, --gmass`	Extract gravitational mass
`-m0, --rmass`	Extract rest mass
`-j, --jmoment`	Extract angular momentum
`-tw, --twratio`	Extract $T/\lvert W\rvert$ ratio
`-co, --comega`	Extract central angular velocity
`-mo, --maxomega`	Extract max angular velocity
`-eo, --eomega`	Extract equatorial angular velocity
`-R, --radius`	Extract equatorial radius
`-r, --rratio`	Extract polar equatorial ratio
`-a, --all`	Alias for `-e -em -m -j -tw -co -mo -eo -R -r`

B.1.4. plotter

Because of the ramified structure of the output of `condor_parser` and `extractor` is not straightforward to plot and analyze the data without a proper interface, which in RNSA is provided by `plotter`. The script has two main goals: to draw a quick plot of a given pair of physical quantities for the sequences computed with `condor_parser`, and to find their turning points. Since generally this second feature is not required, for example when plotting the gravitational mass as a function of the equatorial radius (as in figure 1.1 and figure 2.6a), a flag `--notur` is provided to prevent `plotter` from calculating the turning points. In this Thesis the turning points are calculated only with this kind of sequences and thus with central energy density on the x axis and gravitational mass on the y axis.

`--notur` *disable the computation of the turning points*

Turning points

The turning points of sequences of equilibrium models with constant rest mass (or angular momentum) and varying central energy density are of great importance in the first part of this Thesis, so they have been computed with special care.

In a wide region near the turning point the M_0 (or J) constant sequences have vanishing slope so it is not accurate to estimate the turning point by taking the minimum or the maximum of the sampled data as it is can result in an incorrect value of central energy for the turning point (unless a fine mesh is used). For this reason, a polynomial fit is performed and the turning point is found by taking the first and derivative of the interpolated polynomial and imposing the minimum or maximum condition. No numerical error is introduced in taking the derivatives since the function is a polynomial so that they can be computed analytically. To find the degree of the polynomial that should be used to interpolate the sample data a routine called `best_poly_fit` tries to fit the data with polynomial of increasing degree until or a maximum degree is reached (which is fixed at 15 by default, but this value can be customized within the code, when the function `best_poly_fit` is called) or there are available to few points to compute a meaningful interpolation. The

chosen degree is the one that minimizes the residual squares of the fit. Then the turning point is checked with some test to verify that is not spurious.

The tests performed are:

- The sequence must have at least 25 points
- The turning point must be not too close to a border of the sequence (at least 3 %). It they were at one extremum it would have meant that there are convergence problems
- The turning points must have greater gravitational mass than the TOV point with the same central energy
- The turning points must have energy density between the minimum converging model's of the same sequences and the maximum's

In this way even with a coarse mesh (so for the first rough estimations) the locations of the turning points are always correctly found.

A quantity must have been extracted with `extractor` in order to be plotter. Moreover, to specify the folder that contains the data the flag `-f` is used:

`-f` *path of the data folder*

To run `plotter` the two quantities on the x and y axis have to be chosen with the flags `-x` and `-y`. The names accepted are the ones with an abbreviation in table B.2:

`-x` *quantity on the x axis*

`-y` *quantity on the y axis*

Since the static solution is treated differently by `condor_parser`, the path of its data folder has to be given explicitly. This is done because usually the static solution is computed once and processed with `extractor` but it is plotted every time with many different rotating solutions that are in different folders. The static solution is set with the flag `--static`

`--static` *path of the folder that contains the static solution*

The are other flags to save the output in different forms or to set name and title. The list of supported flags is in table B.5. A complete example of execution is:

```
$ plotter -f eosL --static eosL_static -x energy -y gmass \
  -S -s -l -o
```

This will plot the gravitational mass versus central energy plot showing it via graphical inferface, and saving it in a PDF and a TikZ-ready file and it computes the turning points saving the output in a `energyvsgmass.output` and `energyvsgmass.turning` file. The exported plots are saved in the folder where data are so the one specified with the `-f` flag.

The latter file is ready to be used as input for `instabiler` that takes care of computing every detail of the turning points's models.

Table B.5.: Arguments explanation for `plotter`. The star symbol indicates that the argument is required to run the script. Plotter is used for plotting any two given quantities (from the ones in table B.2) for an arbitrary large number of data files produced with `condor_parser` and processed with `extractor`. If not specified `plotter` tries also to calculate the turning points of the sequences and this operation has been designed to work with `energy` on the x axis and `gmass` on the y axis, for every other quantity it might be ill defined. If a title is not provided the plot will not have any title. The exported plots are saved in the folder where data are so the one specified with the `-f` flag.

Flag	Meaning
`-f, --files` ⋆	Folders where data are stored
`-x, --xaxis` ⋆	Choose quantity on the x axis
`-y, --yaxis` ⋆	Choose quantity on the y axis
`--static` ⋆	Folder that contains the nonorotating solution
`-n, --name`	Set plot's name. If it's not provided the default is `xaxisvsyaxis` where `xaxis` and `yaxis` are the previous arguments
`-s, --save`	Save the plot to `name.pdf` in PDF format
`-S, --show`	Show the plot via using a graphic interface
`-l, --latex`	Export to a TikZ format in `name.tikz`
`-o, --output`	Print numerical output on `STDOUT` and on a file named `name.output` and produce a table `name.turning` for `instabilizer`
`-t, --title`	Set plot's title
`--notur`	Disable the computation of turning points

B.1.5. instabilizer

Since the turning points found with `plotter` are usually not among the models computed with `condor_parser` because they are the result of a interpolation, they are to be computed again with RNS. For this purpose the utility `instabilizer` reads the `.turning` file in a folder and compute with RNS every model it finds storing the output in files called after the values of the physical parameter. `instabilizer` reads the `.turning` file which is produced using the flag `-o` or `--output` in the `plotter` program. If `plotter` has been run with a custum name (`-n, --name` argument) that name has to be specified here with the same flag. This file is a ten column file as the one reported in table B.6. This file contains redundant information as there are four columns that have constant value. Those column are in the file to improve the readability since they describe what is the quantity on their right. The structure of the file is the most general possible since it is designed for the three-parameters rotation law, but when a parameter is not needed it is automatically set to zero.

To correctly use `instabilizer` it has to be configured by setting up the `eos_path` inside the code with the path of the directory that contains the EOS tables and usually is the `condor_parser`'s `eos_path` (section B.1.2).

Then, both the path of data folder and the name of the equation of state have to be given with the flags `-f` and `-e`:

`-f` *path of the data folder*

`-e` *name of the equation of state inside* `eos_path`

Table B.6.: Example `.turning` file. The abbreviation *Let.* stands for *letter* as in that field there is always the same letter. It is there to improve the human-readability of the file, so that it is not necessary to memorize the meaning of each column to understand the file.

ϵ_c	M	J or M	J or M	A1	$\hat{A}_1$	A2	$\hat{A}_2$	B	β
$[10^{15}\,\mathrm{g\,cm^{-3}}]$	$[M_\odot]$	Let.		Let.	[ ]	Let.	[ ]	Let.	[ ]
2.67	2.054	J	2.25	A1	0.83	A2	2.0	B	0.6
...	...	...	...	A1	...	A2	...	B	...

The usage is simple, for example:

```
$ instabilizer -f eosC_J -e eosC
```

This command read the `.turning` file inside the folder `eosC_J` and computes with RNS the models specified inside. Running `instabilizer` a second time deletes every previous result in the folder. Moreover, the execution of this script can take hours to complete if the number of models to compute is large as it runs only on a single thread.

The output of `instbilizer` is stored in the input's folder and consits in many `inst_` files whose name depends on the parameter of the model.

Table B.7.: Arguments explanation for `instabilizer`. This script uses RNS to compute the models defined in a `.turning` file produced with `plotter`. In this way even the turning points's models are completly knwon.

Argument	Meaning
`-f, --files` $\star$	Folder where data are stored
`-e, --eos` $\star$	Name of equation of state
`-n, --name`	Name of the `turning` file produced by `plotter` if it has been run with `-n, --name`

B.1.6. instaplotter

`instaplotter` is a version of the `plotter` script designed for models computed with `instabilizer`. This script reads the files produced in the specific format by `instabilizer` and plots two given quantities. Its usage is very close to the `plotter`'s. The main difference is that `instaplotter` does not need to compute turning points, but it has support for plotting multiple folders and rescaling the curves according to given quantities. This feature is essential for finding the universal relations but has to be configured inside the script. For example, to scale both the plotted quantities by their TOV value the code is:

```
if (args.rescale):
    y = y/tovy
    x = x/tovx
```

To scale differently those lines of `instaplotter.py` have to be manually edited. Moreover, the commented part at the end of the code provides examples of interpolation of the EOS-independent laws both using polynomials (using NumPy's `polyfit`) or custom functions (with NumPy's `curve_fit`) and errors's bars evaluation.

As it is for `plotter`, to specify the folder that contains the data the flag `-f` is used:

`-f` *path of the data folder*

To run `instaplotter` the two quantities on the x and y axis have to be chosen with the flags `-x` and `-y`. The names accepted are the ones with an abbreviation in table B.2:

`-x` *quantity on the x axis*

`-y` *quantity on the y axis*

Table B.8.: Arguments explanation for `instaplotter`. The star symbol indicates that the argument is required to run the script. This script plots any two given quantities from models computed with `instabilizer` and rescale them according to a predefined law defined inside the program that has to be customized.

Flag	Meaning
`-f, --files` ⋆	Folders where data are stored
`-x, --xaxis` ⋆	Choose quantity on the x axis
`-y, --yaxis` ⋆	Choose quantity on the y axis
`-n, --name`	Set plot's name. If it's not provided the default is `xaxisvsyaxis` where `xaxis` and `yaxis` are the previous arguments
`-s, --save`	Save the plot on a file named `name.pdf`
`-S, --show`	Show the plot via using a graphic interface
`-l, --latex`	Export to a `TikZ` format in `name.tikz`
`-r, --rescale`	Rescale curves according to the instructions in the code
`-o, --output`	Print numerical output on `STDOUT` and on a file named `name.output`
`-t, --title`	Set plot's title

B.1.7. instacomparer

`instacomparer` is an utility that provides a full insight on the properties of a set of models computed with `instabilizer` by plotting every possible combination of two physical quantities. To run `instacomparer` one or more data folders have to be specified with the flag `-f`:

`-f` *path of the target folder 1*, *path of the target folder 2*, ...

In addition to the input folders, a path and a name for the output file has to be supplied with the flag `-o`:

`-o` *path of the output file*

Its usage is simple and does not allow any special customization, an example is:

```
$ instacomparer -f eosC eosL -o plots.pdf
```

This command produces a PDF file in the present working directory named `plots.pdf`. If a file with the same name exists `instacomparer` overwrites it without asking. The output file consists in a multipage document where every page is a plots two different quantities with every sequence found in the folder. In order to save storage only one of the two possible combinations of two quantities is plotted and the figures are saved with a resolution of 150 DPI (Dots Per Inch). The quantities plotted by `instacomparer` with their units are the ones that have an abbreviation in table B.2.

Table B.9.: Arguments explanation for `instacomparer`. This script produces a multipage PDF with every possible combination of two physical quantities, providing a clear insight on the reciprocal dependencies.

Argument	Meaning
`-f, --files` ⋆	Folders where data are stored
`-o, --output` ⋆	Set name and path of the output file