The Schneider-Roberts-Ott Equation of State Code

Andre da Silva Schneider (andschn@caltech.edu)
Luke F. Roberts (robertsl@nscl.msu.edu)
Christian D. Ott (cott@tapir.caltech.edu)
https://bitbucket.org/andschn/sroeos
https://stellarcollapse.org/SROEOS
SROEOS@stellarcollapse.org

Last updated June 29, 2017.

Abstract

This is the user guide to the Schneider-Roberts-Ott (SRO) equation of state code described in Schneider, Roberts, & Ott (2017) [1]. Here we provide details on how to compile and run the various parts of the SRO code. We also discuss the details of their inputs and outputs.

Contents

1 Introduction

The Schneider-Roberts-Ott (SRO) equation of state (EOS) code is intended to build EOS tables of hot dense matter for astrophysical simulations. The development of the SRO EOS was supported by the National Science Foundation Theoretical and Computational Astrophysics Network (TCAN) award NSF AST-1333520.

Our work, described in Reference [1] (SRO hereafter), builds upon the seminal work of Lattimer and Swesty (L&S hereafter) [2] and presents a generalized method to generate EOSs using a compressible non-relativistic liquid-drop model with Skyrme interactions.

At low densities, one may choose to transition from the L&S-type EOS, which uses a single nucleus approximation (SNA), to a nuclear statistical equilibrium (NSE) description of an ensemble of nuclei. To simplify its usage, the SRO code is separated into three different parts:

(1) The single nucleus approximation (SNA) code that builds a nuclear EOS using a compressible non-relativistic liquid-drop model with Skyrme interactions. The SNA code does not add electrons, positrons, and photons.

(2) The nuclear statistical equilibrium (NSE) code, which builds a nuclear EOS for a mixture of non-interacting nuclei. The SNA code also does not add electrons, positrons, and photons.

(3) The `MERGE` code which combines the SNA and/or the NSE EOSs and adds electrons, positrons, and photons. If so desired, `MERGE` can also run on pure SNA or pure NSE tables.

IMPORTANT: We recommend that the user read the description of the `MERGE` code first before generating SNA and NSE tables. Care must be taken when setting up the SNA and NSE table resolutions and ranges. This is discussed in Section 6.2.2.

The reason for our choice of having three separate parts is that one may want to separately build SNA and NSE tables and change the parameters used to merge different EOSs without having to rebuild them. This is discussed in detail in Section VII.A. of SRO [1].

The SRO code is written using a modern, modular, and OpenMP-threaded Fortran 90+ framework and is made publicly available at https://stellarcollapse.org/SROEOS under the GNU General Public License version 3 (GPL 3).

Some files and subroutines used in the SRO code are from other open-source codes and were slightly modified to suit our needs. These are:

1. The Timmes & Arnett EOS is used to compute the EOS of electrons and photons. For details see Appendix A of SRO and Reference [3].
2. The Fermi integrals and their inverses are computed using the subroutines of Fukushima [4, 5].
3. To solve the system of Equations (38) in SRO, we use the equation solver package "*nleqslv*" of Hasselman [6].
4. Derivatives of thermodynamical quantities w.r.t. parameters that are a function of coupled independent variables are computed using the LU decomposition subroutines of the L&S code available at http://www.astro.sunysb.edu/dswesty/lseos.html.

Besides the SRO code, we also provide open-source Fortran 90 and C++ HDF5 table readers that interpolate between calculated values.

2 Disclaimer and Getting Help

The SRO EOS code is open source under the GNU General Public License version 3 and may be used at your own risk. The code comes with absolutely no warranty. While we will always try to help users, we are unable to guarantee that we will be able to provide support or help if you run into problems running it. If you decide to use the SRO EOS code or tables generated with it in published work, it is your responsibility to test the code and ensure its physical correctness and consistency in the application problem you are using it for.

As a prerequisite to running the SRO EOS code, you should be familiar and comfortable with the Linux/Unix-style command line and file systems (including symbolic links), with compilers and Makefiles, and with installing external libraries either by hand or via your operating system's package manager.

If you run into physics problems, find bugs, or inconsistencies, please let us know by sending email to this address:

SROEOS@stellarcollapse.org

Please understand that our resources and time are limited and we may not be able to help you with basic compilation problems, coding problems, or with problems with external libraries.

3 Requirements, Dependencies, and `make.inc` for Compilation

When making EOS tables, we recommend that you use a computer system with ample compute power (many CPU cores) and with at least 32 GB of RAM to build EOS tables. The default table ranges and resolutions in the SNA and NSE parts lead to tables of around 7 GB that will be merged to a final standard-resolution table of around 700 MB. The SNA code is the slowest component and typically requires a few hours using 20 CPU cores to build the default SNA table.

The SRO EOS code requires the following software on your system:

1. The git version control system. We recommend git version 2 and higher. If it is not already installed, try installing it via your operating system's package manager or get it from https://git-scm.com.
2. Fortran 90+, C++, and C compilers, such as the `gcc` compiler suite available at https://gcc.gnu.org or through your operating system's package manager. Your compiler suite must support OpenMP parallelization (most current ones do).
3. The `HDF5` library, available at https://support.hdfgroup.org/HDF5/. It must be compiled with Fortran 90+ support.
4. The `BLAS` library, available at http://www.netlib.org/blas/ or through your operating system's package manager.
5. The `LAPACK` library, available at http://www.netlib.org/lapack/ or through your operating system's package manager.

Note that, in general, the `HDF5`, `LAPACK`, and `BLAS` libraries must be compiled with the same compiler(s) (and compiler version(s)) used to compile the SRO EOS code components.

Should you decide to compile some of the above libraries yourself, we strongly recommend that you disable shared libraries and go with static libraries instead. This will greatly reduce potential headaches with dynamically linked symbols that are not found. Also, as a general guideline, if you can link the SRO EOS code components statically, do so. Linking statically may not be possible on MacOS.

Once you have setup all software/libraries, you need to create the `make.inc` file in the root directory of the SRO EOS code. This file is read by the SRO EOS Makefiles to compile the code components `SNA`, `NSE`, and `MERGE`. We provide various examples for Linux and MacOS systems in the directory `make_inc_examples`.

4 The Single Nucleus Approximation Code `SNA`

The `SNA` code (residing in the `SNA` subdirectory) is used to compute the EOS of hot dense matter using a compressible non-relativistic liquid-drop model with Skyrme interactions. It does not

include electrons, positrons, and photons.

The SNA code computes the most favorable composition of nuclear matter for a given Skyrme parametrization for the nucleon-nucleon interactions for a range of proton fraction y (identical to electron fraction Y_e because of charge neutrality), temperature T, and number density n. A discussion of the code library dependencies, its input and output as well as units used are detailed in the rest of this section.

4.1 Compiling and Running the SNA Code

Make sure your system satisfies the requirements outline in §3 and that your `make.inc` file is set up correctly.

Compiling the SNA code should now be as simple as typing on your terminal from the root directory of the SRO EOS:

```
> cd SNA
> make
```

This will create two executables in the SNA directory, `main` and `test`. Their inputs and outputs are discussed in Sections 4.2 and 4.3, respectively.

IMPORTANT: When solving for the EOS, the code will make use of OpenMP parallelization. It will use as many OpenMP threads as specified by the system environment variable `OMP_NUM_THREADS`. Make sure to check what this is set to and adjust it, if you want to use less or more threads. We do not recommend using more threads than available physical CPU cores on your machine.

We carried out tests of both executables using both the GNU and the Intel compiler suite. Even though all computations are performed using double precision arithmetic, small fluctuations in some of the results, usually of $\mathcal{O}(10^{-7})$ or less, may occur depending on the compiler used and the optimization level flag `-O#` chosen in the `make.inc`. Experiments show that these small differences result from the computation of some second derivatives when obtaining surface properties of nuclear matter.

4.2 Input

The `main` and `test` executables take two input files, with one of the input files being the same for both: `input/skyrme.in`. The `main` executable uses as its second input the `input/space.in` file, which contains details on the ranges and resolutions in density, temperature, and proton fraction for the resulting EOS. The `test` executable uses as its second input the `input/test.in` file, which contains a single point in density, temperature, and proton fraction, and, if desired, an initial guess for the solution. We discuss these input files in the following.

4.2.1 The Skyrme input `input/skyrme.in`

The `input/skyrme.in` file contains four required and three optional Fortran namelists that specify the Skyrme parametrization being used and other EOS parameters and code control flags. The code will read `input/skyrme.in`, but the file itself is a symbolic link. By default it points

to input/list_SKRA.in, which is the file specifying the SKRA parametrization. If a different parametrization is desired, the symbolic link can be changed to point to one of the other parameterizations that we provide (or to a user-customized file). In the input directory, we provide parameter sets for the following parametrizations (as described in the SRO EOS paper [1]): KDE0v1, LNS, LS220, LS220*, NRAPR, SKRA, SkT1, SkT1*, Skxs20, SLy4, and SQMC700. See the SRO paper [1] for a discussion of these parametrizations.

The Fortran namelists in the input/skyrme.in file are

1. TABLE_INPUT (required),
2. SKYRME_TYPE (required),
3. SKYRME_COEFFICIENTS (required),
4. SURFACE_PROPERTIES (required),
5. HEAVY_NUCLEI_SIZE (optional),
6. STIFFEN_EOS (optional),
7. TABLE_OUTPUT (optional).

The TABLE_INPUT Namelist

The TABLE_INPUT namelist contains information about the type of output files to write, their names, and locations. An example of the TABLE_INPUT namelist, taken from input/list_SKRA.in file, is the following:

```
&TABLE_INPUT
  filename_base     = "SKRA",
  output_directory  = "EOS_SKRA",
  make_hdf5_table   = .true.,
  write_solutions_to_file = .false.,
  redefine_print_parameters = .false.,
/
```

Note that the only parameter of this list relevant to the test executable is output_directory as that is where the code's output, if any, is written. For the main executable, the only required parameters are output_directory and filename_base. The logical parameters are set by default to the values given above and only need to be explicit if changed. The parameter redefine_print_parameters is used to limit the quantities printed to the SNA EOS table. Its usage is described alongside the TABLE_OUTPUT namelist below.

The parameter make_hdf5_table sets whether to write an HDF5 table in the output_directory directory. The table name starts with filename_base followed by information about the table resolution in $\log_{10}$(density), $\log_{10}$(temperature), and proton fraction. Next is information on the git revision. If the code to make the table was compiled from a repository with local changes, the short git hash is prepended with a captial "M". If the repository is all fully committed, the short git hash has no prefix. Finally, the name contains the date the table was generated. For example, using the default input/skyrme.in and input/space.in files, a table name could look

lik this: SKRA_rho427_temp169_ye67_gitMd089cf6_20170624.h5. Note that the table is only written after all solutions have been computed. Also note that we are including the full source code and all inputs needed to generate the table. See Section 8 for details on this.

The details of the contents (including units) of the HDF5 output file is discussed in Section 4.3.2.

The write_solutions_to_file variable determines whether to write the independent variables that solve the system of equations and other debugging information for each point in phase space to an ASCII file. By the default it is set to false. If write_solutions_to_file = .true., the code creates the directories SOL and NO_SOL inside the output_directory. An output file for each proton fraction y is created in SOL in which the solutions to Equations (38) in SRO [1], as well as the solution for the uniform system discussed in Section 4.3, are written for each temperature and density. To debug issues at density, temperature, and proton fraction points where no solutions were found, an output file for each proton fraction y is created in NO_SOL. If no solution is found for a given point, a detailed error of the likely issue is stored in the corresponding y file. We provide a detailed description of the output files in Section 4.3.

The SKYRME_TYPE Namelist

The SKYRME_TYPE namelist is used to give a few details on the input Skyrme parametrization. An example is shown below.

```
&SKYRME_TYPE
  Skyrme_parametrization_type  = 0,
  non_local_terms              = 1, ! (default is 1)
  Use_default_constants        = 0, ! (default is 0)
  High_density_stiffening      = .FALSE., ! (default is .FALSE.)
/
```

The integer parameter Skyrme_parametrization_type can assume the values 0 and 1:

- Skyrme_parametrization_type = 0 ⇒ uses "standard" Skyrme terms, *i. e.*, the Skyrme terms usually found in the literature. The standard Skyrme parameters are x_j, t_j (with $j = 0, 1, 2, 3i$), and σ_i. Here, i is the number of non-local terms given by the parameter non_local_terms. For almost every parameterization found in the literature $i = 1$. In the code, the "standard" terms are converted to the "simplified" Skyrme terms using Equation (14) of SRO [1].
- Skyrme_parametrization_type = 1 ⇒ reads as input the "simplified" Skyrme parameters a, b, c_i, d_i, δ, α_1, α_2, and $q_{tt'}$ for $t, t' = p, n$ (see Equation 14 of [1]). Any values not given are assumed to be zero. Again, i is the number of non-local terms given by the parameter non_local_terms.

The next control in the SKYRME_TYPE namelist is Use_default_constants. Its value can be 0 or 1. If Use_default_constants = 0 then experimental values for the neutron and proton masses are used. Also, the neutron proton mass difference $\Delta = m_n - m_p$ is obtained self-consistently and the binding energy of alpha particles is determined with respect to that of a gas of free neutrons, $B_\alpha = 30.8866\,\text{MeV}$. Otherwise, if Use_default_constants = 1 then $m_n = m_p$, $\Delta = 1.29\,\text{MeV}$, and $B_\alpha = 28.3\,\text{MeV}$. The latter choice is consistent with that used by L&S [2].

Finally, High_density_stiffening can assume the values .true. or .false.. If set to .false., no modifications to the Skyrme parameterization given in the namelist SKYRME_COEFFICIENTS is performed. If set to .true., then extra terms are added to the Skyrme parametrizations that stiffen the EOS at high densities. The stiffening follows the discussion of Section V A of SRO [1] and is controlled by the namelist SKYRME_STIFFENING discussed below.

The SKYRME_COEFFICIENTS Namelist

The SKYRME_COEFFICIENTS namelist contains the Skyrme parameters. Its format for the case Skyrme_parametrization_type = 0 and non_local_terms = 1 for the SKRA parametrization is shown below.

```
 &SKYRME_COEFFICIENTS
  Coeff_t0            = -2895.4000d0, ! in MeV fm^3
  Coeff_t1            =   405.5000d0, ! in MeV fm^5
  Coeff_t2            =   -89.1000d0, ! in MeV fm^5
  Coeff_t3            = 16660.0000d0, ! in MeV fm^(3+3*sigma)
  Coeff_x0            =     0.0800d0, ! dimensionless
  Coeff_x1            =     0.0000d0, ! dimensionless
  Coeff_x2            =     0.2000d0, ! dimensionless
  Coeff_x3            =     0.0000d0, ! dimensionless
  Coeff_sigma         =     0.1422d0, ! dimensionless
/
```

If the standard Skyrme parameters are given as shown here (Skyrme_parametrization_type = 0), the simplified parameters (a, b, c_i, d_i, δ_i, α_1, α_2, and $q_{tt'}$) are calculated using Equations (14) and (27) of SRO [1].

In the Skyrme_parametrization_type = 1 case, the simplified Skyrme parameters must be given in the SKYRME_COEFFICIENTS namelist. Parameters that are not explicitly set in the SKYRME_COEFFICIENTS namelist are assumed to be zero. An example for the LS220 parametrization is shown in input/list_LS220.in file.

The SURFACE_PROPERTIES Namelist

The SURFACE_PROPERTIES namelist controls how the surface properties are obtained (see Section II.B. of SRO [1]). In can also contain information on the proton-fraction dependence of the critical temperature (see Equation 22 of SRO [1]). For the SKA case, we have:

```
 &SURFACE_PROPERTIES
  fit_surface    = .false.,
  Surface_sigma  =   1.124937302481D+00, ! in Mev fm^(-2)
  Surface_q      =   2.398029067503D+01, ! dimensionless
  Surface_p      =   1.707004256669D+00, ! dimensionless
  Surface_lambda =   3.702622653011D+00, ! dimensionless
/
```

If fit_surface = .false., then the code will use the specified precomputed surface parameters (see Section II.B. of SRO [1]) as in the above. We provide precalculated surface parameter values for

all included parameterizations. Note that the provided values may change slightly $\mathcal{O}(10^{-6}-10^{-7})$, depending on the compiler and optimization flag used, since the calculation in sensitive to round-off error. We do not expect differences of this order to significantly alter the EOS.

If `fit_surface = .true.`, then the code will compute the surface parameters, which can take $\sim 2-5$ minutes, which is significant if one is simply interested in running the `test` code. `fit_surface=.true.` must be used for any parameterization for which the surface parameters are not known. Once the surface fit for a given parametrization is found, the user should look up its values in the `output_directory/Surface_properties.dat` file, change `fit_surface = .true.` to `fit_surface = .false.` in the `list.in` file and copy the values of the fit parameters to the `SURFACE_PROPERTIES` namelist described below. This prevents the surface properties from being calculated every time the user wants to run a test or remake a table reusing a given parametrization. For the parametrizations available in the `input` directory, the surface fittings are already provided.

Note that in the `SURFACE_PROPERTIES` namelist shown above, the critical temperature $T_c(y_i)$ (Equation 22 of SRO [1]) fitting is performed by the code since, by default, `fit_temperature = .true.`. However, this calculation is extremely fast we do not bother do add the precalculated values for the parameterization in Equation (22). However, in the `input/list_LS220star.in` input of the LS220* case (see [1]), we set `fit_temperature = .false.` and specify the critical temperature fit coefficients explicitly to use the same fit as L&S.

The *optional* `HEAVY_NUCLEI_SIZE` Namelist

In Section 2.9 of L&S [2], the size of heavy nuclei in the translational part of the free energy is fixed to $A = A_0 = 60$. Though we compute A self-consistently, we also allow for an optional a `HEAVY_NUCLEI_SIZE` namelist in case one wants to compute the EOS using the prescription of L&S. This requires adding the `HEAVY_NUCLEI_SIZE` namelist to `input/skyrme.in`:

```
&HEAVY_NUCLEI_SIZE
  fix_heavy_nuclei_size = .true.,
  A00 = 60.d0,
/
```

Note that `A00` is the desired fixed value for heavy nuclei, A_0. If `A00` is too small (`A00`$\lesssim 30$) or too large (`A00`$\gtrsim 120$), problems can occur in the EOS calculation.

The *optional* `SKYRME_STIFFENING` Namelist

As discussed in Section V.A. of SRO [1], most Skyrme parametrizations are unable to produce neutron stars with masses as high as those observed in nature. Thus, we add the option to include extra terms to the Skyrme parametrization that stiffens the EOS at high densities. These terms are added in the namelist `SKYRME_STIFFENING`. The extra terms are determined following Section V.A. of SRO [1] (see Equation 56). The stiffening exponent δ_2 is given by the parameter `stiffening_exponent`, while the ratio shown in Equation (56) of SRO [1] is given by `stiffening_ratio`. Finally, the weight w between the extra c_2 and d_2 terms is controlled by the parameter `stiffening_symmetry` in such a way that $c_2 + d_2 = wc_2 + (1-w)d_2$. For the SkT1* parametrization discussed in SRO [1] the namelist has the form

```
&SKYRME_STIFFENING
  stiffening_exponent = 5.00d0,
  stiffening_ratio    = 0.01d0,
  stiffening_symmetry = 1.00d0,
/
```

The *optional* TABLE_OUTPUT Namelist

The final (optional) namlist is TABLE_OUTPUT. By default all possible output variables are written to the HDF5 table. However, by adding the TABLE_OUTPUT namelist and selecting redefine_print_parameters = .true., one may select a set of variables to *not* be written to the tables. The output variables are discussed in Section 4.3.3 of this user guide.

For example, one of the written variables is the pressure P, which appears in the HDF5 table as p. To not write the pressure variable to the final table add the namelist

```
&TABLE_OUTPUT
  print_p = .false.,
/
```

For other variables simply add print_(variable name) = .false., where variable name is one of the variables discussed in Section 4.3.3.

4.2.2 SNA test input

The test executable takes as input the namelist input_test_list in input/test.in. This namelist determines the point in density, temperature, and proton fraction where to calculate the EOS. If desired, an initial guess can be given for the variables that are used in the solution of the EOS for uniform and non-uniform nuclear matter.

```
&input_test_list
! point in phase space
proton_fraction =  30.0D-02, ! proton fraction for test
temperature     =  1.00D-00, ! temperature in MeV for test
density         =  1.00D-03, ! density in fm^-3 for test
! use initial guesses?
guess                 = .true.,
uniform_sol_guess     = -6.5d0,
non_uniform_sol_guess = -9.0D0, -110.0D0, -2.15D0,
/
```

As discussed in L&S [2] the solution for the uniform system is solved in terms of the independent variable $y_p = n_{po}/n$ such that

$$n_{no} = n\frac{y_p(2 - nv_\alpha(1-y)) + 2(1-2y)}{2 - nv_\alpha y}\,. \tag{1}$$

The notation used here is the same as in SRO [1]. From the value of y_p one obtains the neutron and proton densities, n_{no} and n_{po}, respectively. Round-off errors make it difficult to find solutions for uniform proton rich matter ($y > 0.50$) for most densities n and temperatures T when using the above independent parameter. Thus, unlike L&S, for proton rich matter, we choose the independent variable to be $y_n = n_{no}/n$, which implies

$$n_{po} = n\frac{y_n(2 - nv_\alpha y) - 2(1 - 2y)}{2 - nv_\alpha(1 - y)}\,. \tag{2}$$

Thus, if desired, when running the `test` code one may add an initial guess to y_p or y_n through the parameter `uniform_sol_guess`.

Similarly, a guess of the independent variables $\zeta = [\log_{10}(n_{no}), \log_{10}(n_{po}), \log_{10}(u)]$ that determine the solution to the non-uniform system (see Section III of SRO [1] may be added with the parameter `uniform_sol_guess`.

We have tried our best to implement a method that will find a solution over the entire reasonable space in density, temperature, and proton fraction. However, convergence cannot be guaranteed and the code may fail for some choices of density, temperature, and proton fraction.

Also, in rare cases we have observed that the code returns a solution which does solve the systems of equations, but returns a non-physical condition. We have tried to eliminate those from the output and look for realistic solutions. However, this is not always possible and the user sohuld be aware that unphysical solutions, though very unlikely, may pop up from time to time. This is we advise the user to carefully test the resulting tables.

Finally, to avoid underflows/overflows we limit the absolute values of some quantities to a maximum of 10^{100} and a minimum of 10^{-100}. To the best of our knowledge, this does not affect the final solutions. Another option would be to change from double precision to quadruple precision arithmetic. However, this would be cumbersome and would render the code very slow.

Even if no initial guess is given, the SNA `test` code should be able to find solutions for most parametrizations within the limits:

- $10^{-14}\,\mathrm{fm}^{-3} \lesssim n \lesssim 10\,\mathrm{fm}^{-3}$;
- $10^{-4}\,\mathrm{MeV} \lesssim T \lesssim 300\,\mathrm{MeV}$;
- $0.001 \lesssim y \lesssim 0.7$.

The code can become slow at very low temperatures and very low densities.

4.2.3 SNA Table Ranges and Resolution Input File `input/space.in`

The `main` executable also takes as input the namelist `space_list` contained in `input/space.in`. This namelist determines the range and spacing in $\log_{10} n$ (density n given in fm^{-3}), $\log_{10} T$ (temperature T given in MeV), and proton fraction y that will be covered by the output table. The example below should be self-explanatory. Note that there are two methods for determining the spacing for each of the three variables and that we commented out the superseding option.

```
&space_list
yp_min      =  0.00499D0,       ! minimum proton fraction
```

```
yp_max      =  0.66499D0,       ! maximum proton fraction
steps_in_Yp =  67,              ! steps in Yp every 0.01
! Yp_spacing = 0.01D0,          ! spacing in Yp
! (if used Yp_spacing supersedes option for steps_per_hundredth_in_yp)

log10n_min  = -13.20d0,         ! log10 of minimum density in fm^-3
log10n_max  =   1.0d0,          ! log10 of maximum density in fm^-3
steps_per_decade_in_n  =  30,   ! steps per decade in density
! log10n_spacing = 0.03333333d0,! spacing in log10(n)
! (if used log10n_spacing supersedes option for steps_per_decade_in_n)

log10T_min  =  -3.1d0,          ! log10 of minimum density in fm^-3
log10T_max  =   2.5d0,          ! log10 of maximum density in fm^-3
steps_per_decade_in_T  = 30,    ! steps per decade in density
! log10T_spacing = 0.03333333d0,! spacing in log10(T)
! (supersedes log10T_spacing option for steps_per_decade_in_n)
/
```

4.3 Output

Both the `test` and the `main` executables produce output with information on the employed Skyrme parametrization. Below is a list of output files written to the directory defined by the `output_directory` in the `TABLE_INPUT` namelist.

- `Skyrme_coefficients.dat` contains the Skyrme parametrization used.
- `Nuclear_Properties.dat` contains the parameters of the expansion of the specific nuclear energy around nuclear saturation density, Equations (44-47) in SRO, the parameters of the symmetry energy expansion (48-51) in SRO, and information about the effective masses of nucleons for symmetric nuclear matter at nuclear saturation density, and alpha particle mass, volume, and binding energy.
- `symmetry.dat` contains the symmetry energy of uniform nuclear matter as a function of density, see Equation (49) of SRO [1].
- `surface_density.dat` contains y_i, T (in MeV), the densities $\log_{10} n_{ni}$, $\log_{10} n_{pi}$, $\log_{10} n_{no}$, and $\log_{10} n_{po}$ which are solutions to Equations (23) and (24) of SRO [1] and the residue of the four equations.
- `surface_tension.dat` uses the density results given in `surface_density.dat` and provides y_i, T (in MeV), the parameters r_n, a_n, and a_t (in fm), which minimize the surface tension, the derivatives of $\sigma(y_i, T)$ w.r.t. r_n, a_n, and a_p, and the surface tension $\sigma(y_i, T)$ in MeV fm^2. Note that r_p is set to zero by default. r_t and a_t and the surface tension are discussed around Equations (25-28) of SRO [1].
- `surface_tension_fit.dat` contains y_i, T, the normalized surface tension $\sigma(y_i, T)/\sigma_s$, the fitting $\sigma_{\rm fit}(y_i, T)/\sigma_s$ given by Equations (19) and (20) of SRO [1], and the ratio $\sigma_{\rm fit}(y_i, T)/\sigma(y_i, T)$.

- `Surface_fit.dat` contains the parameters σ_s, λ, q, and p of the surface fit, Equations (19) and (20) in SRO [1], as well as the surface properties A_S and S_S in Equation (53) of SRO [1].
- `Critical_temperature_fit.dat` provides the parameters T_{c0}, a_c, b_c, c_c, and d_c that fit the critical temperature $T_c(y_i)$ for which two phases of nuclear matter with different densities and proton fractions coexist. See Equation (22) of SRO [1].

Note that the files related to the surface fit will only be written if explicit calculation of the surface fit takes place, *i.e.*, if `fit_surface = .true.` in the `SURFACE_PROPERTIES` namelist.

The other output depends on whether `test` or `main` is run. If `test` is run, output about the search for solution, possible errors found, and ground state of the system, if found, is printed to the standard output.

If `main` is run, then an `HDF5` file is output, unless `make_hdf5_table = .false.`.

4.3.1 Output of `SNA test`

The test output should be mostly self-explanatory. However, some error messages still need to be implemented. Units are appropriate combinations of MeV and fm unless otherwise stated explicitly.

4.3.2 Output of `SNA main`

The `main` executable creates a directory with the name that is specified in the `output_directory` parameter of namelist `TABLE_INPUT`.

If `write_solutions_to_file = .true.` in namelist `TABLE_INPUT`, then the code creates two subdirectories `SOL` and `NO_SOL`. Although this should work smoothly, some Fortran compilers have problems with the system calls and are unable to create directories and subdirectories. If that is the case, the user will need to create the necessary directories manually.

If `write_solutions_to_file = .true.`, in the `SOL` directory, a file with the solutions to the uniform and non-uniform systems is written for each proton fraction. The explanation of the columns of these files is given below. In the `NO_SOL` directory, a file for each proton fraction is created and if neither a uniform nor a non-uniform solution is found for a particular combinations of proton fraction, density, and temperature, their values and possible errors encountered are written to the file. The explanation of the columns is given below.

The files written in the `output_directory/SOL/` directory have 17 columns. They are:

1.–3. Density (fm^{-3}), temperature (MeV), and proton fraction y.

4. Independent variable y_p (y_n) that solves the uniform system if $y < 0.50$ (if $y > 0.50$). Set to zero if uniform solution not found or not searched for.

5.–7. Independent variables $\zeta = [\log_{10}(n_{no}), \log_{10}(n_{po}), \log_{10}(u)]$ that solve the non-uniform system. Densities are assumed to be in fm^{-3}. Set to zero if non-uniform solution not found or not searched for.

8.–9. Specific free energy in MeV for the uniform and non-uniform systems. Set to 10^{99} if solution not found or not searched for.

10.–11. Dot product of residue for the equations solved for uniform and non-uniform systems. Set to 10^{10} if solution not found or not searched for.

12.–13. Number of times tried to search for uniform and non-uniform solutions.

14.–15. Logical (T or F) indicating whether uniform and non-uniform solutions were found.

16.–17. Logical (T or F) indicating whether uniform and non-uniform solutions were searched for.

The files written in the `output_directory/NO_SOL/` directory have 9 columns. They are:

1.–3. Density (fm^{-3}), temperature (MeV), and proton fraction y.

4.–5. Dot product of the residue of the equations solved for the uniform and non-uniform systems. Set to 10^{10} if solution not found or not searched for.

6.–7. Logical (T or F) indicating whether uniform and non-uniform solutions were found.

8.–9. Logical (T or F) indicating whether uniform and non-uniform solutions were searched for.

4.3.3 SNA HDF5 output

The `HDF5` output file, unless limited by the `TABLE_OUTPUT` namelist discussed above, contains information about the following quantities.

1. Scalar integer `pointsrho`. Number of table points in logarithmic density.
2. Scalar integer `pointstemp`. Number of table points in logarithmic temperature.
3. Scalar integer `pointsye`. Number of table points in proton fraction.
4. 1D Array `logn` with size `pointsrho`. Contains the values for $\log_{10} n$, where n is density in fm^{-3}.
5. 1D Array `logtemp` with size `pointstemp`. Contains the values for $\log_{10} T$, where T is temperature in MeV.
6. 1D Array `ye` with size `pointsye`. Contains the values for the proton fraction.
7. 3D Array `p` with size (`pointsrho`,`pointstemp`,`pointsye`). Note that all 3D arrays described in the following have the same size. Contains the pressure P in units of $\mathrm{MeV\,fm}^{-3}$ for each point (n, T, y) in the table.
8. 3D Array `s`. Contains the specific entropy in $k_B\,\mathrm{baryon}^{-1}$.
9. 3D Array `e`. Contains the specific internal energy in $\mathrm{MeV\,baryon}^{-1}$ with respect to the free neutron mass.
10. 3D Array `muh`. Contains $\hat{\mu} = \mu_n - \mu_p$ in MeV. Note that $\hat{\mu}$ includes the neutron–proton mass difference.
11. 3D Array `mun`. Contains μ_n in MeV with respect to the free neutron mass.

12. 3D Array `mup`. Contains μ_p in MeV with respect to the free neutron mass.
13. 3D Array `dpdn`. Contains $dP/dn|_{T,y}$ in MeV.
14. 3D Array `dsdn`. Contains $ds/dn|_{T,y}$ in $k_B\,\mathrm{fm}^3\,\mathrm{baryon}^{-1}$.
15. 3D Array `dmudn`. Contains $d\mu_h/dn|_{T,y}$ in $\mathrm{MeV}\,\mathrm{fm}^3$.
16. 3D Array `dpdT`. Contains $dP/dT|_{n,y}$ in fm^{-3}.
17. 3D Array `dsdT`. Contains $ds/dT|_{n,y}$ in $k_B\,\mathrm{baryon}^{-1}\,\mathrm{MeV}^{-1}$.
18. 3D Array `dmudT`. Contains $d\mu_h/dT|_{n,y}$. Dimensionless.
19. 3D Array `dpdy`. Contains $dP/dy|_{n,T}$ in $\mathrm{MeV}\,\mathrm{fm}^{-3}$.
20. 3D Array `dsdy`. Contains $ds/dy|_{n,T}$ in $k_B\,\mathrm{baryon}^{-1}$.
21. 3D Array `dmudy`. Contains $d\mu_h/dy|_{n,T}$ in MeV.
22. 3D Array `zbar`. Contains the charge number Z of the representative heavy nucleus.
23. 3D Array `abar`. Contains the mass number A of the representative heavy nucleus.
24. 3D Array `xn`. Contains the neutron mass fraction.
25. 3D Array `xp`. Contains the proton mass fraction.
26. 3D Array `xa`. Contains the alpha particle mass fraction.
27. 3D Array `xh`. Contains the heavy nuclei mass fraction.
28. 3D Array `u`. Contains the occupied volume fraction by heavy nuclei u.
29. 3D Array `r`. Contains the occupied generalized radius of heavy nuclei r in fm.
30. 3D Array `meff_n`. Contains the effective mass of unbound neutrons m_n^* in MeV.
31. 3D Array `meff_p`. Contains the effective mass of unbound protons m_p^* in MeV.

Note that the `SNA` code also includes additional datasets in the `HDF5` file that contain the full `SNA` source code and all inputs necessary to reproduce the results stored in the `SNA` EOS table. See Section 8 for details on this feature.

5 The Nuclear Statistical Equilibrium Code `NSE`

The `NSE` code used to compute the EOS of an ensemble of nuclei in nuclear statistical equilibrium resides in the directory `NSE`.

The `NSE` code works by computing the most favorable composition of nuclei (the one with the lowest Helmholtz free energy) for a user-specified ensemble of nuclei (and nucleons) and their masses and partition functions. Details are discussed in the SRO paper [1]. NSE can be computed for a single point (the `test` case) or for a 3D space of points (`main` case) in proton fraction y, temperature T, and density n.

5.1 Compiling and Running the NSE Code

Once all dependencies outlined in Section 3 are installed and the `make.inc` file is set up in the SRO EOS root directory, compiling the `NSE` code should be as simple as

```
> cd NSE
> make
```

This will create two executables in the `NSE` directory: `main` and `test`. Their inputs and outputs are discussed in Section 5.2 and 5.3, respectively.

IMPORTANT: When solving for the EOS, the code will make use of OpenMP parallelization. It will use as many OpenMP threads as specified by the system environment variable `OMP_NUM_THREADS`. Make sure to check what this is set to and adjust it, if you want to use less or more threads. We do not recommend using more threads than available physical CPU cores on your machine.

We tested the `NSE` code with various versions of the GNU compiler suite and the Intel compiler suite.

5.2 Input

The `main` (`test`) executable takes five (four) input files. Three of the input files are the same for both codes – from the `NSE` root directory: `input/list.in`, `input/partition.in`, and `input/isotope_masses.in`. The `main` executable takes two other inputs: the `input/output.in` and `input/space.in` files, which contain, respectively, details on the output to be written by the code and the space in proton fraction, density, and temperature to be covered by the output `NSE` table. The `test` executable reads the `input/test.in` file, which contains a single point in proton fraction, density, and temperature to compute the NSE EOS. We discuss these input files in detail below.

5.2.1 The Mass Table and Partition Function Input

The mass table input, `input/isotopes.in`, contains information about the names of available nuclides and their proton, neutron, and mass number, as well as their mass excess and binding energies. **We advise the user to NOT modify this file.** Information about the nuclide partition functions contained in the `input/partition.in` file. **Again, we do NOT recommend the user to modify this file**. The data in both of these files are obtained from the file `chem/data/isotopes.data` of the open-source MESA stellar evolution code [7], revision r7503, which contains information for 7851 nuclides. The mass table contains 6 columns with nuclide properties: a short name descriptor, mass number, proton number (charge), neutron number, mass excess w.r.t. ^{12}C, and the nuclear binding energy. The excess mass, binding energies and partition function are from the JINA REACLIB library [8]. Here the binding energies are obtained w.r.t. symmetric nuclear matter. However, to be consistent with the `SNA` code, we subtract the neutron-proton mass difference from the resulting free energies.

5.2.2 Input Nuclides

To construct the NSE table, one must provide a list of nuclides in file input/list.in. This file is actually a symbolic link that points to a nuclide list file in the tables directory (see below).

When making a custom input/list.in file, an important restriction to keep in mind is that the chosen nuclides must be present in the input/isotopes.in and input/partition.in files. For obvious reasons, we recommend protons, neutrons, and alpha particles to be among the nuclides included when putting together input/list.in. The input/list.in file must have four columns: the nuclide name, which should match the name in the input/isotopes.in file, its mass number, proton (charge) number, and neutron number. **Because of the way the code is structured, the nuclide list must be ordered by increasing proton (charge) number first and second by increasing neutron number.** We provide in the tables directory the nuclide lists used in the SRO paper and some other nuclide lists inspired by those used by MESA [7].

5.2.3 NSE test input

The test executable takes as input the namelist input_test_list, which is part of the input/test.in file. This namelist determines the point in proton fraction, density, and temperature where to calculate the NSE EOS.

```
&input_test_list
! point in phase space
proton_fraction =  30.0D-02, ! proton fraction for test
temperature     =  1.00D-00, ! temperature in MeV for test
density         =  1.00D-03, ! density in fm^-3 for test
/
```

5.2.4 NSE Table Ranges and Resolution Input File

As in the SNA code, the main NSE executable also takes as input a namelist space_list that is in the file input/space.in (path relative to the NSE root directory; this file is distinct from its SNA counterpart). This namelist determines the range and spacing in $\log_{10} n$ (density n given in fm^{-3}), $\log_{10} T$ (temperature T given in MeV), and proton fraction y that is covered by the output NSE table. The example below should be self-explanatory. Note that there are two methods for determining the spacing for each of the three variables and that we comment out the superseding option. We do not try to obtain the NSE EOS at densities larger than nuclear saturation density $n_0 \simeq 0.16\,\mathrm{fm}^{-3}$ since we expect matter to be a uniform gas of protons and neutrons there. In fact, the NSE EOS if likely unreliable for densities of $\gtrsim n_0/10$ since exotic nuclei dominate in this region and nuclear interactions play an important role and are not included in our NSE framework.

```
&space_list
yp_min      =  0.00499D0,       ! minimum proton fraction
yp_max      =  0.66499D0,       ! maximum proton fraction
steps_in_Yp =  67,              ! steps in Yp every 0.01
! Yp_spacing = 0.01D0,          ! spacing in Yp
```

```
! (if used Yp_spacing supersedes option for steps_per_hundredth_in_yp)

log10n_min  = -13.20d0,         ! log10 of minimum density in fm^-3
log10n_max  =  -1.0d0,          ! log10 of maximum density in fm^-3
steps_per_decade_in_n  =  30,   ! steps per decade in density
! log10n_spacing = 0.03333333d0,! spacing in log10(n)
! (if used log10n_spacing supersedes option for steps_per_decade_in_n)

log10T_min  =  -3.1d0,          ! log10 of minimum density in fm^-3
log10T_max  =   2.5d0,          ! log10 of maximum density in fm^-3
steps_per_decade_in_T  = 30,    ! steps per decade in density
! log10T_spacing = 0.03333333d0,! spacing in log10(T)
! (if used log10T_spacing supersedes option for steps_per_decade_in_T)
/
```

The input/output.in file contains the TABLE_INPUT namelist. Similarly to the SNA case, this namelist contains information about the type of output files to write, their names, and the directory they are written to. Below is an example of the TABLE_INPUT namelist.

```
&TABLE_INPUT
  make_hdf5_table = .true.,
  write_solutions_to_file = .false.,
  filename_base   = "NSE_0023",  ! base for HDF5 table name
  output_directory= "NSE_0023",  ! directory for ascii output files
/
```

The logical parameters in the namelist are set by default to the values given above and only need to be explicitly set if changed. Note that we have chosen the 0023 in the filename to correspond to the default number of NSE nuclides used. Of course, larger NSE networks are possible and should be considered.

The write_solutions_to_file variable determines whether to write the neutron and proton chemical potentials, $\tilde{\mu}_n$ and $\tilde{\mu}_p$, respectively, that determine the nuclear statistical equilibrium and other debugging information for each point in density, temperature and proton fraction to an ASCII file. By default it is set to false.

The parameter make_hdf5_table sets whether to write an HDF5 table in the output_directory directory. The table name starts with filename_base followed by information about the table resolution in $\log_{10}$(density), $\log_{10}$(temperature), and proton fraction. Next is information on the git revision. If the code to make the table was compiled from a repository with local changes, the short git hash is prepended with a captial "M". If the repository is all fully committed, the short git hash has no prefix. Finally, the name contains the date the table was generated. For example, a table name could look like this: NSE_0023rho367_temp169_ye67_gitMd089cf6_20170624.h5. Note that the table is only written after all solutions have been computed. Also note that we are including the full source code and all inputs needed to generate the table. See Section 8 for details on this.

The details of the contents (including units) of the HDF5 output file are discussed in Section 5.3.2.

5.3 NSE Output

5.3.1 Output of Test

The test output should be mostly self-explanatory. However, some error messages still need to be implemented. Units are appropriate combinations of MeV and fm unless otherwise stated explicitly.

5.3.2 Output of main

The main executable creates a directory with the name that is specified in the output_directory parameter of namelist TABLE_INPUT.

The following applies only if write_solutions_to_file = .true. in the TABLE_INPUT namelist:

In the output_directory, the code creates three subdirectories SOL, NO_SOL, and ERROR. Although all of these directories should automatically be created by the code, some Fortran compilers have problems with the system calls and are unable to create directories and subdirectories. If that is the case, the user will need to create the necessary directories manually.

In the SOL directory, files for each proton fraction computed containing the neutron and proton chemical potentials ($\tilde{\mu}_n$ and $\tilde{\mu}_p$) that determine the composition of isotopes given in nuclear statistical equilibrium is given. In the NO_SOL directory, a file for each proton fraction is created and if the solution found is not within the desired tolerance. In the ERROR directory, a file for each proton fraction is created and an error message is written to the corresponding file if there is a problem solving the system of equations. An explanation of the columns in the files of each of these folders is given below.

The files written in the output_directory/SOL/ directory have 13 columns. They are:

1.–3. Density (fm^{-3}), temperature (MeV), and proton fraction y.

4.–5. Neutron and proton chemical potentials, $\tilde{\mu}_n$ and $\tilde{\mu}_p$. Set to zero if a solution is not found.

6. Accuracy of solution defined by $|1 - x_n - x_p - x_\alpha - x_l - x_h|$.
   - x_n is the mass fraction of neutrons.
   - x_p is the mass fraction of protons.
   - x_α is the mass fraction of alpha particles.
   - x_l is the mass fraction of light nuclei. Light nuclei are defined as nuclei with proton number $Z \leq 6$, not including neutrons, protons, and alpha particles.
   - x_h is the mass fraction of heavy nuclei. Heavy nuclei areas defined as nuclei with proton number $Z > 6$.

7.–11. The mass fractions x_n, x_p, x_α, x_l, x_h. Set to 10^{-99} if lower than that.

12. Number of calls to the function `nr_iterate` that solves the system of equations.

13. Number of calls to the function `reorder` that orders nuclei by their abundances.

The files written in the output_directory/NO_SOL/ directory have the same columns as the files in output_directory/SOL/. The difference is that a line is only written if the solution is not found or not whithin the accepted tolerance, $|1 - x_n - x_p - x_\alpha - x_l - x_h| < 10^{-6}$.

The files written in the output_directory/ERROR/ directory have four columns. The columns are only written in case the subroutine `local_nse` is unable to find a combination of nuclides in NSE. The columns are

1.–3. Density (fm^{-3}), temperature (MeV), and proton fraction y.

4. Error type:
   - 1, if the number of iterations trying to compute the solution is larger than the allowed maximum, 600.
   - 2, if the Jacobian of the system of equations is singular.

5.3.3 NSE HDF5 output

The `HDF5` output file contains the following datasets:

1. Scalar integer `pointsrho`. Number of table points in logarithmic density.
2. Scalar integer `pointstemp`. Number of table points in logarithmic temperature.
3. Scalar integer `pointsye`. Number of table points in proton fraction.
4. 1D Array `logn` with size `pointsrho`. Contains the values for $\log_{10} n$, where n is density in fm^{-3}.
5. 1D Array `logtemp` with size `pointstemp`. Contains the values for $\log_{10} T$, where T is temperature in MeV.
6. 1D Array `ye` with size `pointsye`. Contains the values for the proton fraction.
7. 3D Array `p` with size (`pointsrho`,`pointstemp`,`pointsye`). Note that all 3D arrays described in the following have the same size. Contains the pressure P in units of $\mathrm{MeV\,fm}^{-3}$ for each point (n, T, y) in the table.
8. 3D Array `s`. Contains the specific entropy in $k_B\,\mathrm{baryon}^{-1}$.
9. 3D Array `e`. Contains the specific internal energy in $\mathrm{MeV\,baryon}^{-1}$.
10. 3D Array `muh`. Contains $\hat{\mu} = \mu_n - \mu_p$ in MeV. Note that here μ_n and μ_p are obtained with respect to the neutron vacuum mass.
11. 3D Array `mun`. Contains μ_n in MeV.
12. 3D Array `mup`. Contains μ_p in MeV.
13. 3D Array `dpdn`. Contains $dP/dn|_{T,y}$ in MeV.
14. 3D Array `dsdn`. Contains $ds/dn|_{T,y}$ in $k_B\,\mathrm{fm}^3\,\mathrm{baryon}^{-1}$.
15. 3D Array `dmudn`. Contains $d\mu_h/dn|_{T,y}$ in $\mathrm{MeV\,fm}^3$.

16. 3D Array `dpdT`. Contains $dP/dT|_{n,y}$ in fm^{-3}.
17. 3D Array `dsdT`. Contains $ds/dT|_{n,y}$ in $k_B\,\mathrm{baryon}^{-1}\,\mathrm{MeV}^{-1}$.
18. 3D Array `dmudT`. Contains $d\mu_h/dT|_{n,y}$. Dimensionless.
19. 3D Array `dpdy`. Contains $dP/dy|_{n,T}$ in $\mathrm{MeV}\,\mathrm{fm}^{-3}$
20. 3D Array `dsdy`. Contains $ds/dy|_{n,T}$ in $k_B\,\mathrm{baryon}^{-1}$.
21. 3D Array `dmudy`. Contains $d\mu_h/dy|_{n,T}$ in MeV.
22. 3D Array `xn`. Contains the neutron mass fraction.
23. 3D Array `xp`. Contains the proton mass fraction.
24. 3D Array `xa`. Contains the alpha particle mass fraction.
25. 3D Array `xh`. Contains the heavy nuclei mass fraction.
26. 3D Array `xl`. Contains the heavy nuclei mass fraction.
27. 3D Array `zh`. Contains the charge number Z_h averaged over the present heavy nuclei.
28. 3D Array `ah`. Contains the mass number A_h averaged over the present heavy nuclei.
29. 3D Array `zl`. Contains the charge number Z_l averaged over the present light nuclei.
30. 3D Array `al`. Contains the mass number A_l averaged over the present light nuclei.

Note that the `NSE` code also includes additional datasets in the `HDF5` file that contain the full `NSE` source code and all inputs necessary to reproduce the results stored in the `NSE` EOS table. See Section 8 for details on this feature.

6 The Code to Merge Tables: `MERGE`

The `MERGE` code (residing in the `MERGE` subdirectory) is used to combine the HDF5 tables output by the `SNA` and `NSE` codes into a new table using the procedure described in Section VII.A. of SRO [1]. The `MERGE` code also adds photon, electron, and positron contributions to the EOS.

6.1 Compiling and Running the `MERGE` Code

Make sure your system satisfies the requirements outline in §3 and that your `make.inc` file is set up correctly. Given that, compiling the `MERGE` code should now be as simple as typing in your terminal from the root directory of the SRO EOS:

```
> cd MERGE
> make
```

This will create a single executable in the `MERGE` directory called `merge_eos`. Its input and output are discussed in Sections 6.2 and 6.3, respectively.

IMPORTANT: When creating the combined EOS table, the code will make use of OpenMP parallelization. It will use as many OpenMP threads as specified by the system environment

variable OMP_NUM_THREADS. Make sure to check what this is set to and adjust it, if you want to use less or more threads. We do not recommend using more threads than available physical CPU cores on your machine.

6.2 Input

The merge_eos executable takes three input files, input/tables.in, input/space.in, and input/transition.in. They are discussed next.

6.2.1 The input/tables.in file

input/tables.in contains one namelist, tables. An example is given below.

```
&tables
merge_base = 'SKRA_0023',
sna_eos = '../SNA/EOS/SKRA/SKRA_rho427_temp169_ye67_gitMd089cf6_20170624.h5',
nse_eos = '../NSE/NSE_0023/NSE_0023_rho367_temp169_ye67_gitMd089cf6_20170624.h5',
only_sna = .FALSE.,
only_nse = .FALSE.,
/
```

This namelist tables contains five parameters. Three parameters are character strings: merge_base is a base for the file name of the final table (we recommend including the Skyrme parametrization and the number of NSE nuclides [if any]), sna_eos is the path to the SNA table, nse_eos is the path to the NSE table.

The two remaining parameters are logical, only_sna and only_nse determine whether the final HDF5 table is obtained only in the SNA or only in the NSE approximation. Note that both options cannot be set to .true. simultaneously.

Note that in the above case, the resulting output table will be named SKRA_0023_rho415_temp163_ye66_gitMd089cf6_20170624.h5. Note that the M behind git indicates that the table was generated from a not fully committed git repository. The 7 following characters are the short git hash. The final set of numbers is the date the table was created at in YYYY-MM-DD format.

6.2.2 The input/space.in file

As the SNA and NSE codes, the MERGE code also takes as input a file named input/space.in. This file has a namelist space_list that specifies the ranges and resolutions in density, temperature, and proton fraction for the final merged table. An example is shown below.

```
&space_list
yp_min      =  0.005D0,       ! minimum proton fraction
yp_max      =  0.655D0,       ! maximum proton fraction
steps_in_Yp =  66,              ! steps in Yp every 0.01
! Yp_spacing = 0.01D0,          ! spacing in Yp
```

```
! (if used Yp_spacing supersedes option for steps_per_hundredth_in_yp)

log10n_min  = -13.0d0,          ! log10 of minimum density in fm^-3
log10n_max  =   0.8d0,          ! log10 of maximum density in fm^-3
steps_per_decade_in_n  =  30,   ! steps per decade in density
! log10n_spacing = 0.03333333d0,! spacing in log10(n)
! (if used log10n_spacing supersedes option for steps_per_decade_in_n)

log10T_min  =  -3.0d0,          ! log10 of minimum density in fm^-3
log10T_max  =   2.4d0,          ! log10 of maximum density in fm^-3
steps_per_decade_in_T  = 30,    ! steps per decade in density
! log10T_spacing = 0.03333333d0,! spacing in log10(T)
! (supersedes log10T_spacing option for steps_per_decade_in_n)
/
```

Before getting into the details, we want to point out the following:

> **IMPORTANT:** The `MERGE` code interpolates linearly in $\log_{10} n$, $\log_{10} T$ and y to fill points that are not exactly aligned with entries in the SNA and NSE tables. Because of this, great care must be taken to set up density, temperature, and proton fraction grids for SNA, NSE, and merged tables. We recommend to *always* make SNA and NSE tables that are higher resolution than the desired resolution for the final merged table. For the merged tables we make available at https://stellarcollapse.org/SROEOS, we used SNA and NSE tables that have twice the resolution of the merged tables.

While in the `SNA` and `NSE` codes the specified ranges are limited by physical significance and computational limitations, for the `MERGE` code the ranges are limited by the ranges of the SNA and NSE tables being merged.

If the option `only_sna` (`only_nse`) is set to `.true.`, then the space in density, temperature, and proton fraction space given in the namelist `space_list` for the `MERGE` code should be completely inside the space used by the `space_list` namelist used to generate the SNA (NSE) table.

In the standard case that SNA and NSE tables are merged, the range requirements change slightly:

(1) Our merging procedure is only a function of density. Hence, `yp_min` and `log10T_min` in `MERGE` should be larger than or equal to the largest value of these parameters used for the SNA and/or NSE tables.

(2) Similarly, the parameters `yp_max` and `log10T_max` used in the `MERGE` code should be smaller than or equal to the smallest value of these parameters used for the SNA and/or NSE tables.

(3) We always assume that the NSE table is at a lower density than the SNA table. Hence, the requirements for the density are such that `log10n_min` for `MERGE` is larger than or equal to the value of that parameter used to generate the NSE table. Similarly, `log10n_max` used for `MERGE` should be smaller than or equal to the value of this parameter used to generate the SNA table.

We discuss in Section 6.2.3 how the maximum density of the NSE table and minimum density in the SNA table are limited by the transition density, its width, and the merge tolerance.

6.2.3 The input/transition.in file

The file input/transition.in contains a namelist transition that specifies the transition density n_t from SNA to NSE and the transition width n_δ, see Section VII.A. of SRO [1]. We also define a tolerance ϵ_t used for the merge function that will be discussed in the following. An example of the transition namelist is shown below.

```
&transition
  n_transition = -4.00d0,   ! Log10(transition density in fm^-3)
  n_delta      =  0.33d0,   ! transition thickness parameter
  n_tolerance  =  1.d-04,   ! tolerance in transition function
/
```

The function we use to merge the SNA and NSE Helmholtz free energies,

$$F_{\rm MIX} = \chi(n) F_{\rm SNA} + [1-\chi(n)] F_{\rm NSE}\,, \tag{3}$$

depends on the function

$$\chi(n) = \frac{1}{2}\left[1 + \tanh\left(\frac{\log_{10}(n) - \log_{10}(n_t)}{n_\delta}\right)\right]\,. \tag{4}$$

Since $\chi(n)$ only reaches 0 or 1 asymptotically. Hence, to speed up the MERGE code and avoid having to determine the SNA and NSE EOS contributions in regions where they are very small, we limit the range to compute $F_{\rm MIX}$ to densities $n_- < n < n_+$ where

$$\log_{10} n_\pm = n_t \mp n_\delta \tan^{-1}\left(2\epsilon_t - 1\right)\,. \tag{5}$$

Thus, when $\chi(n) \leq \epsilon_t$ the EOS is only given by the NSE EOS, while if $\chi(n) \geq 1-\epsilon_t$ the EOS is given by the SNA EOS.

Given these constraints, when merging an SNA with an NSE EOS, the choices of n_t, n_δ, and ϵ_t should be such that $\log_{10} n_+ <$ log10n_max for NSE table and $\log_{10} n_- >$ log10n_min for the SNA table. Even though the MERGE code does warn the user of any issues with the parameters chosen in the transition namelist, it is up to the user to chose transition parameters that satisfy the constraints discussed. We refer the user to Section VII.A. of SRO [1] and recommend that the user carefully test the effects on their application problem of variations of the above transition parameters.

6.3 Output

The merge_eos executable creates an HDF5 table that combines the SNA and NSE tables with the electron and photon EOSs. The units now are, in most cases, in cgs units. The quantities output in the final table are listed below.

1. Scalar integer pointsrho. Number of table points in logarithmic density.

2. Scalar integer `pointstemp`. Number of table points in logarithmic temperature.
3. Scalar integer `pointsye`. Number of table points in proton fraction.
4. 1D Array `logrho` with size `pointsrho`. Contains the values for $\log_{10}\rho$, where ρ is the baryon density in $\mathrm{g\,cm^{-3}}$. The single baryon rest mass is the free neutron mass.
5. 1D Array `logtemp` with size `pointstemp`. Contains the values for $\log_{10}T$, where T is temperature in MeV.
6. 1D Array `ye` with size `pointsye`. Contains the values for the proton fraction.
7. 3D Array `logpress` with size (`pointsrho`,`pointstemp`,`pointsye`). Contains $\log_{10}P$ for each point (ρ, T, y) in the table where P is the pressure in units of $\mathrm{dyne\,cm^{-2}}$. Note that all 3D arrays described in the following have the same size.
8. 3D Array `entropy`. Contains the specific entropy in $k_B\,\mathrm{baryon}^{-1}$.
9. 3D Array `logenergy`. Contains $\log_{10}(\epsilon+\epsilon_0)$ where ϵ is the specific internal energy in $\mathrm{erg\,g^{-1}}$ with respect to the free neutron mass and $\epsilon_0 = 20\,\mathrm{MeV} = 1.91313\times10^{19}\,\mathrm{erg\,g^{-1}}$ is an energy shift set to guarantee $\epsilon+\epsilon_0 > 0$.
10. Scalar integer `energy_shift`. Contains the energy shift $\epsilon_0 = 20\,\mathrm{MeV} = 1.91313\times10^{19}\,\mathrm{erg\,g^{-1}}$ in cgs units.
11. 3D Array `muhat`. Contains $\hat{\mu} = \mu_n - \mu_p$ in MeV. Note that $\hat{\mu}$ includes the neutron–proton mass difference.
12. 3D Array `mu_n`. Contains the neutron chemical potential μ_n in MeV with respect to the free neutron mass.
13. 3D Array `mu_p`. Contains the proton chemical potential μ_p in MeV with respect to the free neutron mass.
14. 3D Array `mu_e`. Contains the electron chemical potential μ_e in MeV including the electron rest mass.
15. 3D Array `munu`. Contains the neutrino chemical potential $\mu_\nu = \mu_n - \mu_p + \mu_e$ in MeV.
16. 3D Array `dedt`. Contains $d\epsilon/dT|_{\rho,y}$ in units of $\mathrm{erg\,g^{-1}\,K^{-1}}$.
17. 3D Array `dpdrhoe`. Contains $dP/d\rho|_{\epsilon,y}$ in units of $\mathrm{cm^2\,s^{-2}}$.
18. 3D Array `dpderho`. Contains $dP/d\epsilon|_{\rho,y}$ in units of $\mathrm{g\,cm^{-3}}$.
19. 3D Array `gamma`. Contains the dimensionless adiabatic index $\Gamma = d\log P/d\log\rho|_s$.
20. 3D Array `cs2`. Contains the speed of sound squared in units of $\mathrm{cm^2\,s^{-2}}$.
21. 3D Array `zbar`. Contains the average charge number Z_h of heavy nuclei. For the NSE EOS only isotopes with $Z > 6$ are considered "heavy". See the discussion in Section 5.3.2.
22. 3D Array `abar`. Contains the average mass number A_h of heavy nuclei. For the NSE EOS only isotopes with $Z > 6$ are considered "heavy". See the discussion in Section 5.3.2.

23. 3D Array `zlbar`. Contains the average charge number Z_l of light nuclei. For the NSE EOS only isotopes with $Z \leq 6$ excluding protons, neutrons, and alpha particles are considered "light". See the discussion in Section 5.3.2.

24. 3D Array `albar`. Contains the average charge number A_l of light nuclei. For the NSE EOS only isotopes with $Z \leq 6$ excluding protons, neutrons, and alpha particles are considered "light". See the discussion in Section 5.3.2.

25. 3D Array `xn`. Contains the neutron mass fraction.

26. 3D Array `xp`. Contains the proton mass fraction.

27. 3D Array `xa`. Contains the alpha particle mass fraction.

28. 3D Array `xh`. Contains the heavy nuclei mass fraction.

29. 3D Array `xl`. Contains the light nuclei mass fraction.

30. 3D Array `u`. Contains the occupied volume fraction by heavy nuclei u. Note that as of now we do not use an excluded volume for the nuclei in the NSE EOS and, thus, its contribution is zero.

31. 3D Array `r`. Contains the occupied generalized radius of heavy nuclei r in fm. Set to zero for the NSE EOS.

32. 3D Array `meff_n`. Contains the effective mass of unbound neutrons m_n^* in MeV. Set to zero for the NSE EOS.

33. 3D Array `meff_p`. Contains the effective mass of unbound protons m_p^* in MeV. Set to zero for the NSE EOS.

Note that the `MERGE` code also includes additional datasets in the `HDF5` file that contain the full `MERGE` source code and all inputs necessary to reproduce the results stored in the merged EOS table. It also includes all source code and inputs needed to reproduce the NSE and SNA parts of the EOS. See Section 8 for details on this feature.

7 EOS Driver and Interpolation Codes

The HDF5 EOS tables generated by the SRO EOS code use the same overall format and the same physical units for thermodynamic quantities as the legacy EOS tables available from https://stellarcollpase.org/equationofstate in the format originally chosen by O'Connor & Ott (2010, [9]). Routines that can handle the legacy EOS tables will be able to handle SRO tables.

That said, a few caveats need to be kept in mind:

(1) The SRO EOS provides additional variables that are not in the legacy O'Connor & Ott tables (see Section 6.3). Readers and interpolation routines need adjustment in order to be able to provide these additional variables.

(2) The legacy O'Connor & Ott tables include the speed of sound without relativistic corrections. However, the new SRO EOS tables include it with these corrections. SRO tables include an integer flag called `have_rel_cs2` in the HDF5 file that, if set to `1` (as is the default for

SRO tables), indicates that the speed of sound is already relativistic and does not need to be corrected by the application code. The legacy tables do not have such a dataset, so application code can judge from its absence that the speed of sound needs correction.

Specifically, we are talking about the following:

$$c_s^2 = \frac{c^2}{h}\frac{dP}{d\rho}\bigg|_s , \tag{6}$$

with $h = c^2 + \epsilon + p/\rho$ and ϵ being the specific internal energy and c being the speed of light.

We provide two different example codes for using SRO EOS tables. They implement linear interpolation (and extrapolation) and root finding for inverting $\epsilon(\rho, T, Y_e)$ to find the temperature at fixed ρ, Y_e after the update of the internal energy by the (radiation-)hydrodynamics scheme.

7.1 `Fortran 90` Interface/Driver/Interpolator

This code is written in `Fortran 90` and is a somewhat updated version of the code provided at https://stellarcollapse.org/equationofstate.

The `Fortran 90` code is available as open source via a `git` repository hosted on `bitbucket.org`. To obtain the code, execute

```
git clone https://bitbucket.org/zelmani/eosdriverfortran
```

We do not recommend using the `Fortran 90` code in simulations. Much of it dates back more than a decade. It is inefficient and a display of rather poor software engineering. (But it works).

Note that the current versions of this EOS interface/driver/interpolation implementation supports handling the relativistic speed of sound corrections as described above.

7.2 C/C++ Interface/Driver/Interpolator

For high-performance applications, we recommend the more recent and much better designed C/C++ routines available at

```
git clone https://bitbucket.org/zelmani/eosdrivercxx
```

These routines have been optimized for speed (including vectorization) and perform up to a factor of three faster than the legacy `Fortran 90` routines.

Note that the current versions of this EOS interface/driver/interpolation implementation supports handling the relativistic speed of sound corrections as described above.

8 Code and Data Provenance: `Formaline`

Provenance is a word used in computational science (and in some other research areas). It describes the full history of a (computational) scientific result. **Knowing the provenance of a simulation result or dataset makes it fully reproducible**. The provenance of a calculation does not only include the released open-source code, but also the full information on the exact source code that was used, perhaps even the way it was compiled and where it was run, the parameters that were

set for the calculation, and all initial conditions and other ingredients that were used in whatever form (and also their respective provenances).

Historically, EOS tables for core-collapse supernova and neutron star merger simulations were stored in proprietary (binary) formats that did not provide the user with information on the table format and on how the table was generated, which code and parameters were used. This made it often impossible to reproduce how a given table was generated and, in the worst case, made it impossible to use the table if the format was no longer known. HDF5 solves the latter problem, but by itself does not provide information on how the table was generated.

The SRO EOS code uses the Formaline approach to boundle source code and input parameters with its output. Formaline was pioneered by Erik Schnetter for the Cactus Computational Toolkit (http://www.cactuscode.org). In its original form, Formaline stored a tar.gz file of the Cactus source code in the Cactus executable. Upon execution, the tar.gz file is written to the output directory. We have extended this concept to include the SRO source code and all input files in HDF5 SRO EOS tables.

8.1 Formaline-preserved Code and Inputs in HDF5 Tables.

SNA tables contain the HDF5 datasets

- SNA-skyrme.in – the SNA Skyrme input file input/skyrme.in, discussed in Section 4.2.1.
- SNA-space.in – the SNA table ranges and resolution input file input/space.in, discussed in Section 4.2.3.
- SNA-src.tar.gz – the complete SNA source tree, make.inc, Makefile, and git hash.

NSE tables contain the HDF5 datasets

- NSE-isotope-list.in – the list of isotopes used for the NSE table, file input/list.in, discussed in Section 5.2.2.
- NSE-isotope-properties.in – isotope data table, file input/isotopes.in, discussed in Section 5.2.1.
- NSE-output-list.in – the NSE table output namelist file, input/output.in, discussed in Section 5.2.4
- NSE-partition.in – – isotope partition function data table, file input/partition.in, discussed in Section 5.2.1.
- NSE-space.in – the NSE table ranges and resolution input file input/space.in, discussed in Section 5.2.4.
- NSE-src.tar.gz – the complete NSE source tree, make.inc, Makefile, and git hash.

The MERGE code (see Section 6) generates the final EOS tables. It copies the above datasets from the input SNA and/or NSE tables and places them in the output HDF5 table. It adds the following datasets specific to the MERGE code:

- MERGE-tables.in – MERGE table output details and paths to SNA and NSE tables being merged, file input/tables.in, discussed in Section 6.2.1.

- `MERGE-space.in` – the `MERGE` table ranges and resolution input file `input/space.in`, discussed in Section 6.2.2.
- `MERGE-transition.in` – transition parameters used to merge the `SNA` and `NSE` tables, file `input/transition.in`, discussed in Section 6.2.3.
- `MERGE-src.tar.gz` – the complete `MERGE` source tree, `make.inc`, Makefile, and git hash.

8.2 Extracting `Formaline`-preserved Information

The SRO EOS comes with a simple Formaline extraction code written in C++. It resides in the `formaline_extract` directory in the EOS root directory and uses the EOS-wide `make.inc`.

Compiling. Go into `formaline_extract` and type `make`. This creates the exectuable `extract_from_h5`.

Usage. It's as simple as it gets:

```
./extract_from_h5 [HDF5 filename] [Dataset Name]
```

Recall that you can always use `h5ls` to learn about the contents of an `HDF5` file.

9 References

[1] A. S. Schneider, L. F. Roberts, and C. D. Ott. A New Open-Source Nuclear Equation of State Framework based on the Liquid-Drop Model with Skyrme Interaction. *Phys. Rev. C*, (to be submitted), 2017.

[2] J. M. Lattimer and F. D. Swesty. A Generalized Equation of State for Hot, Dense Matter. *Nucl. Phys. A*, 535:331 – 376, 1991.

[3] F. X. Timmes and D. Arnett. The Accuracy, Consistency, and Speed of Five Equations of State for Stellar Hydrodynamics. *Astrophys. J. Suppl. Ser.*, 125:277–294, November 1999.

[4] T. Fukushima. Precise and fast computation of inverse fermi–dirac integral of order 1/2 by minimax rational function approximation. *Appl. Math. Comput.*, 259:698 – 707, 2015.

[5] T. Fukushima. Precise and fast computation of fermi–dirac integral of integer and half integer order by piecewise minimax rational approximation. *Appl. Math. Comput.*, 259:708 – 729, 2015.

[6] B. Hasselman. Package "nleqslv", 2016.

[7] B. Paxton, L. Bildsten, A. Dotter, F. Herwig, P. Lesaffre, and F. Timmes. Modules for Experiments in Stellar Astrophysics (MESA). *Astrophys. J. Suppl. Ser.*, 192:3, January 2011.

[8] Richard H. Cyburt, A. Matthew Amthor, Ryan Ferguson, Zach Meisel, Karl Smith, Scott Warren, Alexander Heger, Rob D. Hoffman, Thomas Rauscher, Alexander Sakharuk, Hendrik Schatz, F. K. Thielemann, and Michael Wiescher. The JINA REACLIB Database: Its Recent Updates and Impact on Type-I X-ray Bursts. *Astrophys. J. Suppl. Ser.*, 189(1):240, 2010.

[9] E. O'Connor and C. D. Ott. A New Open-Source Code for Spherically Symmetric Stellar Collapse to Neutron Stars and Black Holes. *Class. Quantum Grav.*, 27(11):114103, June 2010.