Manual for occup_3d_ext and its auxiliary utilities

Johann Weber

September 29, 2021

Contents

1 Introduction **4**

2 Requirements **4**

3 Compiling **5**

4 Testing the program **5**

5 File formats **6**
5.1 The main input file (input) 6
5.2 The file describing the sources (input) 6
5.3 The file describing the density structure (input) 7
5.4 The files describing the ionization structure (output) 7
5.5 The files describing the thermal energy and the temperature (output) 8
5.6 Logging of the timesteps in `time.txt` (output) 8
5.7 Results for the escape fractions (output) 8

6 Option reference for `occup_3d_ext` **9**
6.1 Command line options 9
6.2 Options in input file 10
6.2.1 abund_rel_to_H 10
6.2.2 add_heating 10
6.2.3 add_heating_filename 10
6.2.4 Arions 10
6.2.5 aritficially_fix_temperature 11
6.2.6 ArtoH 11
6.2.7 Artosolar 11
6.2.8 Cions 11
6.2.9 c_correct_on 12
6.2.10 charge_transfer 12
6.2.11 compute_temperature 12
6.2.12 CtoH 12
6.2.13 Ctosolar 13
6.2.14 diffuse 13
6.2.15 diffuse_random 13
6.2.16 diffuse_random_rays 13
6.2.17 diffuse_threshold 13
6.2.18 f_a 14
6.2.19 factor 14
6.2.20 filling_factor 14
6.2.21 H_0 15

6.2.22 HetoH . . . 15
6.2.23 Hetosolar . . . 15
6.2.24 ifrit_filename . . . 15
6.2.25 initial_temperature . . . 15
6.2.26 include_Ar . . . 16
6.2.27 include_C . . . 16
6.2.28 include_He . . . 16
6.2.29 include_metals . . . 16
6.2.30 include_N . . . 17
6.2.31 include_Ne . . . 17
6.2.32 include_O . . . 17
6.2.33 include_S . . . 17
6.2.34 initial_redshift . . . 18
6.2.35 l_cell . . . 18
6.2.36 last_step . . . 18
6.2.37 log_diffuse . . . 18
6.2.38 log_heating_cooling . . . 18
6.2.39 lower_energy_limit . . . 19
6.2.40 mask_filename . . . 19
6.2.41 mergespec . . . 19
6.2.42 Neions . . . 19
6.2.43 NetoH . . . 19
6.2.44 Netosolar . . . 20
6.2.45 nH_complete_scalar . . . 20
6.2.46 nHI_fraction . . . 20
6.2.47 no_source_filename . . . 20
6.2.48 NtoH . . . 21
6.2.49 Ntosolar . . . 21
6.2.50 num_threads . . . 21
6.2.51 o_abundan . . . 21
6.2.52 Oions . . . 22
6.2.53 o_thermal_pressure . . . 22
6.2.54 o_write_escape_fraction . . . 22
6.2.55 old_solar_abundances . . . 23
6.2.56 omega_m . . . 23
6.2.57 optabu_3d . . . 23
6.2.58 OtoH . . . 23
6.2.59 Otosolar . . . 24
6.2.60 periodic . . . 24
6.2.61 photoheat_metals . . . 24
6.2.62 read_from_file . . . 25
6.2.63 rep_output . . . 25
6.2.64 Sions . . . 25
6.2.65 source_filename . . . 25
6.2.66 start_ionized . . . 26
6.2.67 StoH . . . 26
6.2.68 Stosolar . . . 26
6.2.69 t_end . . . 26
6.2.70 t_start . . . 27
6.2.71 t_step_max . . . 27
6.2.72 use_mask . . . 27
6.2.73 trace_expansion . . . 27
6.2.74 verbose_stdout . . . 28
6.2.75 x_max . . . 28

6.2.76 y_max . . . 28
6.2.77 z_max . . . 29

7 3dhydro_ext **29**
7.1 Variables that affect all density distributions . . . 29
7.1.1 ensure_minimum . . . 29
7.1.2 macro_filling_factor . . . 29
7.1.3 nHI_fraction . . . 30
7.1.4 nhyd . . . 30
7.1.5 nhyd_fill . . . 30
7.1.6 random_mode . . . 31
7.1.7 x_center . . . 31
7.1.8 x_max . . . 31
7.1.9 y_center . . . 31
7.1.10 y_max . . . 32
7.1.11 z_center . . . 32
7.1.12 z_max . . . 32
7.2 Homogeneous density distributions (`mode="HOMOGENEOUS"`) . . . 32
7.3 Exponential, spherically symmetric density distribution(`mode="EXPONENTIALRAD"`) 33
7.4 Exponential, plane-parallel density distribution(`mode="EXPONENTIALPLANE"`) . . . 33
7.5 Spherically symmetric power-law density distribution with cutoff(`mode="POWERRADCUTOFF"`) 33
7.6 Plane-parallel symmetric power-law density distribution with cuttoff (`mode="POWERPLANECUTTOFF"`) 33
7.7 Spherically symmetric Gaussian density distribution(`mode="GASUSSIAN"`) . . . 34
7.8 Fractal density structure(`mode="FRACTAL"`) . . . 34

8 Auxiliary utilities **35**
8.1 create_temperature . . . 35
8.2 nebstat . . . 35
8.3 radiusbench . . . 36
8.4 spectinfo . . . 36

A Description of the namelist-based hydro code **37**
A.1 Structure of the 3dhydro_ext_namelist input and output files . . . 37
A.2 Variables that affect all density distributions . . . 37
A.2.1 ensure_minimum . . . 37
A.2.2 macro_filling_factor . . . 37
A.2.3 nHI_fraction . . . 38
A.2.4 nhyd . . . 38
A.2.5 nhyd_fill . . . 38
A.2.6 random_mode . . . 39
A.2.7 x_center . . . 39
A.2.8 x_max . . . 39
A.2.9 y_center . . . 39
A.2.10 y_max . . . 40
A.2.11 z_center . . . 40
A.2.12 z_max . . . 40
A.3 Homogeneous density distributions(`mode="HOMOGENEOUS"`) . . . 40
A.4 Exponential, spherically symmetric density distribution(`mode="EXPONENTIALRAD"`) 41
A.5 Exponential, plane-parallel density distribution(`mode="EXPONENTIALPLANE"`) . . . 41
A.6 Spherically symmetric power-law density distribution with cutoff(`mode="POWERRADCUTOFF"`) 41
A.7 Plane-parallel symmetric power-law density distribution with cuttoff(`mode="POWERPLANECUTTOFF"`) 41
A.8 Spherically symmetric Gaussian density distribution(`mode="GASUSSIAN"`) . . . 42
A.9 Fractal density structure(`mode="FRACTAL"`) . . . 42

B Parallelization of the code and performance aspects **43**
B.1 Motivation 43
B.2 Parallelization of occup_3d_ext and performance measurements 43
B.3 Tests 44

1 Introduction

occup_3d_ext is a 3D radiative transfer code able to account for atomic gas. It currently considers hydrogen, helium and the ionization stages I to IV of carbon, nitrogen, oxygen, neon and sulfur. Implementation details have been described in detail in Weber et al. [2013] and Weber et al. [2015]. In this manual we will focus on the usage of the program.

2 Requirements

occup_3d_ext has been developed in Fortran 2003, but the only Fortran 2003 specific features used so far are the calls to the subroutine get_command_argument, the function command_argument_count (both of these subprograms are related to the handling of command line options) and the flush statement. Thus each Fortran 95 compatible compiler that additionally supports these extensions may be able to generate the occup_3d_ext program.
The program has been tested with the Intel Fortran compiler[1] (Versions later than 9) and the GNU Fortran compiler gfortran which is freely available for all relevant platforms. Details about gfortran can be found at
http://gcc.gnu.org/wiki/GFortran.
occup_3d_ext can in principle be compiled for 32 bit systems, but large simulations (e.g including 101^3 grid cells) that account for the ionization structure of metals and/or the escape fractions will only run on 64 bit machines with a sufficient amount of RAM.
The program has been tested on Linux and Windows 7. As there no operating system dependent calls, it is likely to work on other operating systems like OS X or BSD-type unixes.
The only external dependency of the program is the LAPACK library. This may be available from your compiler supplier, either included with your compiler (as it is the case for Intel Fortran Version 11 or later) or as a separate package (e.g. for the older versions of Intel Fortran). Alternatively it is possible the download the reference implantation from NIST available from
http://www.netlib.org/lapack/index.html.
This version may however be slower than the platform-specific versions of the compiler versions. LAPACK is also provided in the repositories of the major Linux distributions.
In order to utilize parallelization on multi-processor machines, occup_3d_ext relies on OpenMP (http://http://openmp.org/wp/)
directives (see also Appendix B). On compilers that do not support OpenMP, occup_3d_ext processes will not take profit form multi-core CPUs[2]. Using an OpenMP-compatible compiler is therefore strongly recommended. OpenMP is supported by most of the recent Fortran compilers. gfortran supports OpenMP in version 4.2 or later.
To compile the documentation a LaTeX compiler is required. Both latex and pdflatex can be used, but hyperlinks and cross-references are only supported if either pdflatex is used or the Postscript file generated by latex and dvips is converted into a PDF file afterwards. The documentation requires the latex packages a4wide, hyperref, inputenc, graphicx, and natbib to be installed.

[1] http://software.intel.com/en-us/fortran-compilers
[2] Of course it is still be possible to run several instances of occup_3d_ext (i.e. different simulations) in parallel.

3 Compiling

Provided with the source code is a Makefile. You will probably have to modify the Makefile to set the correct library paths, commands and compiler options.
The Makefile currently includes several build targets:

- `make binary` Creates the binary file using Intel Fortran.
- `make binary_gfortran` The system is built using the GNU Fortran compiler. This option may be removed in the future. It is recommended to modify the variables of the `binary target` such that they meet your build environment.
- `make all` This option currently corresponds to make_binary (using the Intel Compiler)[3]
- `make clean` Removes all but the source files.

There also exists a Visual Studio 2005 project file (adapted to Version 9 of the Intel Fortran compiler) in the directory
`WMbasic_IVF\Occup` to create `occup_3d_ext`, wheras the VS 2005 projects to create the auxiliary programs are located in
`WMbasic_IVF\Utilities\Occup_3d_ext_utils`.
Like the Makefile, these will probably have to be modified in order the run on your system.
If your development environment is neither Visual Studio nor does include the `make` command you have to care about the correct order of the compilation and linking processes yourself.
There is no installation that copies the compiled program and the documentation to an "usual" location in the file system like `/usr/local/bin` or `/usr/share/doc` or sets the `$PATH` (MS Windows: `%Path`). If setting the paths is desired, this has to be done manually by the user. The only environment variables that are evaluated are the variables that control the behavior of OpenMP. The environment variable `$OMP_NUM_THREADS` is an integer, which determines the number of parallel OpenMP threads. It should have the same value as the `num_threads` variable in the input file. If the values differ, the program will still run correctly, but the performance will be poor. So far the have been no benchmarks of the effects of the other OpenMP-related environment variables.

4 Testing the program

The `testcases` directory contains the input files required to test the program after it has been compiled. The test case represents a homogeneous H II region surrounding a a 40 kK dwarf star. Both the nebular and the stellar metallicity corresponds to the solar value (The values for the "solar metallicity" correspond to the ones in Asplund et al. 2009, the properties of the corresponding stellar model are described in Weber et al. 2015). Simulated is one octant of the H II region. The input files are

- `input_values`: This file contains the namelist with the simulation parameters.
- `sources.txt`: File containing the position, the radius, the lifetime and the name of the file with the SED of the source
- `MERGESPEC`: File containing the stellar SED.

The structure of the files will be explained in detail in the next section.
The program is finally run by
`<Path to executable>``/occup_3d_ext < input_values .`

[3]Note that the LaTeXsources of this document are no longer contained in the same directory as the Fortran source files, but instead in the directory `$ug/manual_3d_ext` of the `Doc` module. This document can be rebuilt by calling `latex manual.tex` or `pdflatex manual.tex` three times (which is necessary for the generation of the correct citations and references). The hyperlinks within this document only work in PDF files, but not in Postscript or DVI files.

The execution time for the testcase is approximately 40 minutes for a single thread on a Intel Xeon (Core 2 Architecture) when the Intel Fortran Compiler V 13 is used and the optimizations are activated. For convenience, there exist scripts that automatically run the test case and prepare the output files for `gnuplot` (using `radiusbench`), which is then applied to generate plots that show the abundances and the temperature as a function of the distance from the source. The UNIX script is located in the `testcases/ref_3d_ext` subdirectory and called `run_ext_3d_testcase`. The Windows batchfile, called `run_ext_3d_testcase.bat`, is located in `WMbasic_IVF\Utilities\bat_local_use` . The names of the output files of the scrips have the structure `comparison-`*xx*`.ps`, the reference files are called `reference-`*xx*`.ps`. Here *xx* is either the name of the corresponding chemical symbol or `temp` in the case of the file showing the temperature structure.

5 File formats

5.1 The main input file (input)

In the main input file, the variables which specify the behavior of the program are set. The format of the corresponds to a Fortran namelist. A sample file called `input_values`[4] is provided with the source code.
The file starts with the line
`&input_values`

Then the options and simulation parameters are given their values via pairs of the form *variable* = *value*, where each of the assignments is followed by a comma and, if desired to improve readability of the input file, a "`&`", followed by a newline. More information about namelists can be obtained, for instance, from `http://de.wikibooks.org/wiki/Fortran:_Fortran_95:_Ein-_und_Ausgabe` (in German). Note that even the last assignment has to be followed by a comma. The notation of the values corresponds to the notation that is used in Fortran source code. The file is terminated by a slash.
If no value is assigned to a variable in the input file, the default values will be used. For some variables there are no default values, which may lead to undefined behavior. However, these variables are only evaluated if certain options are switched on. For details see the description of the corresponding options in Sect. 6.2.

5.2 The file describing the sources (input)

The sources of ionization are usually (but see below) described in a separate file, with the default name `sources.txt`. A sample is provided with the source code. In the main input file the name of input file for the sources of ionization can be changed by modifying `source_filename`. The structure of the file is as follows:

- The first line of the file can contain an arbitrary text/comment/explanations and is not evaluated.
- The second line contains the (integer) number of sources.
- The third line is a comment again.
- The fourth line has the structure
  x y z radius spectrum start stop
  Here the integer value x represents the x-index of the cell the source is located in. The x values have to be within the interval $[-x\_max : +x\_max]$. y and z represent the indexes in the other dimensions. *radius* is the radius of the source in solar radii. If the source is not

[4] As the file is passed to the program via standard input, the name of the file does not matter

a spherical object, a representative radius has to be chosen. *spectrum* defines a file, which contains the description of the SED of the ionizing source. `occup_3d_ext` reads files, whose formats corresponds the format of the `MERGESPEC` files that are a output of the WM-Basic code, cf. Pauldrach et al. [2001] and Hoffmann et al. [2012]. It `mergespec=.false.`, the program uses the format, that has been used in previous version of `occup_3d_ext`. The file `ps4e+4.txt` that is included with the source code (but not used in the default testcase) is in the format of the older versions. These file start with 16 lines that are either empty or may be used for comments. The next 2001 lines contain two columns, where the first columns is the wavelength in Å, the second column is the corresponding Eddington flux H_ν in $\mathrm{erg\,s^{-1}\,sr^{-1}\,cm^{-3}}$. It is not currently possible to use both types of file formats in the same simulation.

- For each source, there follow two lines where the first one is a comment and the second one describes the source in the format described above.
- If the option no_source_filename is set to `.true.`, the sources are not read from a separate file, but from the same file that contains the namelist `input_values`, for the setup of the simulation runs.

5.3 The file describing the density structure (input)

If `read_from_file=.true.` the user has to specify the density structure of the gas in a file, whose name is specified in the option `ifrit_filename`[5]. In the first line the dimension of the simulated volume in units of grid cells is set. Note that the absolute size of the simulation volume additionally depends on the option `l_cell`, which specified the length of an edge of a single grid cell in parsec. Because of the symmetry of the coordinate system – the cell with the coordinates $(0, 0, 0)$ is located in the center of the simulated volume – the first line has the form
$2 \times \mathtt{x_{max}} + 1 \;\; 2 \times \mathtt{y_{max}} + 1 \;\; 2 \times \mathtt{z_{max}} + 1$. For example, if x_max = 20, y_max = 30 and z_max = 40, then the first line is
`41 61 81`
The remaining lines contain in their first row the total hydrogen number density and in their second row the number density of neutral hydrogen in $\mathrm{cm^{-3}}$ at the beginning of the simulation. The order of the grid cells is optimized for Fortran, i.e. the loop over the different z-values is the outermost loop and the loop over the x-values is the innermost loop. The file may contain no empty lines or comments and end with a newline character.
Note that it is not possible to set the initial conditions for the ionization structures of elements other than hydrogen. To do so the `-resume` command line option has to be used.

5.4 The files describing the ionization structure (output)

The structure of the files which describe the ionization is compatible to IFrIT. In the first line the dimension of the volume is specified in Sect. 5.3. For each grid cell, the first n columns contain the number densities (in units of $\mathrm{cm^{-3}}$) for the density of the ions E (i.e. neutral atoms of an element E) to E^{+n-1}. The last column contains the total number density of the atoms/ions belonging or the corresponding element. Currently supported are hydrogen (H and H^+), helium (He, and He^+ and He^{2+}), and the most important metals C, N, O, Ne, S and Ar.
Additionally a file with the suffix `.e.txt` is written which contains the electron number density in $\mathrm{cm^{-3}}$.
There currently exist problems to visualize the output data with newer Linux-Versions of IFrIT if the locale is set to German or another language that uses commas a decimal delimiters. The prob-

[5] IFrIT is a visualization program by Nickolay Gnedin, which concentrates on the visualization of ionization fronts and density structures. It can be obtained for free from `https://sites.google.com/site/ifrithome/` and is available for Windows and for Unixes. On the homepage one can also find the documentation of the program and its file formats.

lem can be solved by setting the environment variables `LANG=en_US.UTF-8` *and* `LC_NUMERIC=en_US.UTF-8`*. The Windows version of IFrIT seems to be unaffected by this problem.*

5.5 The files describing the thermal energy and the temperature (output)

If `compute_temperature=.true.` files are written that describe the temperature structure of the simulated file.

- The file with the suffix `.temp.txt` is an IFrIT compatible file that contains in its first column the temperature of the cells before the past timestep in K, in its second column the temperature after the past timestep in K, an in its third column the mean value of the temporal derivative of the temperature $\frac{\mathrm{d}T}{\mathrm{d}t}$ in units of K/yr.
- The IFrIT file with the suffix `.energy.txt` contains in its only column the density of the thermal energy in units of erg cm^{-3}.
- The files with the ending `.xx.heatcool.txt`, where `xx` is the name of the element contain the heating and the cooling rates per element. In the IFrIT compatible files the first column contains the heating rates, the second column the cooling rates, and the third column the difference between the heating and the cooling rates. The used units are erg $\mathrm{s}^{-1}\mathrm{cm}^{-3}$.

5.6 Logging of the timesteps in `time.txt` (output)

In the file `time.txt` the timesteps are logged. The file format is not compatible to IFrIT, as the simulated time is a global property of the system and not a property that differs from cell to cell. The first column contains the length of the past timestep in years, the second column the total simulated time since the start of the simulated process and the third the counter for the timesteps. The fourth column is the wallclock time required per iteration cycle, the fifth column the CPU time per iteration cycle. A comparison of the two latter values indicates how well the code is parallelized (see Sect. B). If `trace_expansion=.true.`, the sixth column contains the redshift at the given time and the counter for the timesteps is written in the seventh column.
The second to last line contains the wall-clock time required to compute the past iteration step, the last line the amount of CPU time needed. Both values are given in seconds. Note that in some cases there might be negative value for the wall-clock time. The reason in an integer overflow affecting the Fortran routine `system_clock`. As this affects only a small fraction of all timesteps, `time.txt` is still useful to analyze the performance. However, in a numerical data analysis program that works with data from `out.txt`, these effects have to be accounted for. It is not clear yet, how this problem can be fixed in a portable way in the Fortran source code.
If the program program is restarted using the `-resume` command line option, the currently a new `time.txt` file such that the log for the previous timesteps is deleted. A workaround is to rename the older version before restarting the program. For more details about restarting a program, see Sect. 6.1.

5.7 Results for the escape fractions (output)

If `o_write_escape_fraction=.true.` the escape fraction from the volume is written as a function of the wavelength in files with the suffix `.escape.txt`. The escape fraction is computed separately for each of the sources. The first column contains the wavelength in Å (the wavelength grid corresponds to the wavelength grid internally used by the radiative transfer procedure of `occup_3d_ext`.
Using `gnuplot`[6] it is possible to plot the escape fractions for the different sources using the `index` option of the `plot` command. `index 0` plots the escape fraction of the first source, `index 1` the escape fraction of the second source and so on.

[6] `http://www.gnuplot.info`

6 Option reference for occup_3d_ext

6.1 Command line options

Currently, there are two ways to call occup_3d_ext.

- The first one does not require a command line option, but the namelist with the simulation parameters (see Sect. 6.2) has to be passed via standard input:

  $PATH_TO_BINARY/occup_3d_ext < input_file

  Here $PATH_TO_BINARY is the path, where the executable is located. Note that by default there is no environment variable where the path is located. So the path has to be passed to the command line manually. Alternatively it is possible to set such an environment variable in a startup file like .bashrc. You just have to type the name of the program, if the name of the directory, where the binary of occup_3d_ext is located is added to the $PATH (UNIX-like operating systems) or %Path (Windows) environment variable.

  The namelist with the input parameters is usually stored in a file called input_values, (one is supplied with the source code), but this is by no means a requirement.

- If the program has been aborted, e.g. because of technical problems or because the computing resources have been needed for other program runs, it is possible to restart the program such that the date form the most recent set of output files are used.

  The command to restart the program is

  $PATH_TO_BINARY/occup_3d_ext -resume *filename_without_ending* < input_file

  filename_without_ending means the part of the name, that is common to all the output files including the file number for the counter, but excluding the subsequent dot.

  The following output files have to be present and complete:

  - *filename_without_ending*.H.txt
  - *filename_without_ending*.He.txt
  - *filename_without_ending*.e.txt
  - time.txt

  If include_metals=.true, a restart additionally requires the following files:

  - *filename_without_ending*.C.txt
  - *filename_without_ending*.N.txt
  - *filename_without_ending*.O.txt
  - *filename_without_ending*.Ne.txt
  - *filename_without_ending*.S.txt
  - *filename_without_ending*.Ar.txt

  If compute_temperature=.true, a restart additionally requires the file *filename_without_ending*.temp.txt and if trace_expansion =.true. the file time.txt has to be generated with a run, where the same option is required. The values form read these files are used as the new "initial conditions" for the resumed run.

  The program is not able to find the correct re-entry point automatically. If the program is restarted, the file-counter starts at 1000 again, but the simulated time and the redshift are continued correctly. In the first step to the new run, the time-step size corresponds to the minimal value t_start.

  In principle it is possible to artificially generate or modify the start files with another program. This is likely to lead to inconsistent results if a program is restarted, but can also be used to define the initial conditions of the simulations in more detail than it can be specified in the namelist with the input parameters.

6.2 Options in input file

6.2.1 abund_rel_to_H

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If true, the abundances of helium and the considered metals have to be given relative to the abundance of hydrogen. This has to be done using the *Elementname*`toH` options. `add_heating_filename`.
- *caveats:* If this option is set, the default value for the relative abundances is 10^{12} helium and all considered metals.
  If this option is set, the values for the *Elementname*`tosolar` values are ignored without a warning.

6.2.2 add_heating

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `compute_temperature=.true.`
- *description:* If true, for each of the cells the artificial heating rate is computed as a function of the electron density, described by a polynomial of degree 3. The coefficients of the polynomial are read in from the file specified in `add_heating_filename`.
- *caveats:* The coefficients of the polynomial remain the same throughout the entire simulation. Furthermore they do not depend on temperature.

6.2.3 add_heating_filename

- *data type:* `character (len=3000)`
- *default value:* none
- *requires:* `add_heating=.true.`
- *description:* The name of the file containing the polynomial coefficients. The order of the cells corresponds to the one of the IFrIT uniform scalar data text files. Each entry contains of 4 columns. The formula for the heating rate per volume unit is

$$Q = C_1 \cdot n_\mathrm{e}^3 + C_2 \cdot n_\mathrm{e}^2 + C_3 \cdot n_\mathrm{e} + C_4, \tag{1}$$

  where Q is in units of $\left[\mathrm{erg\,s^{-1}\,cm^{-3}}\right]$, and n_e is the electron number density in $\left[\mathrm{cm^{-3}}\right]$. C_n is the value entry in the nth column of the file.
- *caveats:* The coefficients of the polynomial remain the same throughout the entire simulation. Furthermore they do not depend on temperature.

6.2.4 Arions

- *data type:* `integer`
- *data range:* $3 \leq$ `Arions` ≤ 4
- *default value:* 4
- *description:* Number of considered Ar ions, starting with the neutral stage and reaching to the `Arions`-1 times ionzed stage.

6.2.5 aritficially_fix_temperature

- *data type:* `logical`
- *default value:* `.false.`
- *requires:*
- *description:* Sets the temperature to a fixed value of `aritficially_fix_temperature` Kelvin.
- *caveats:* Not thoroughly tested yet.

6.2.6 ArtoH

- *data type:* `real`
- *data range:* `ArtoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio $n(\mathrm{Ar})/n(\mathrm{H})$.

6.2.7 Artosolar

- *data type:* `real`
- *data range:* `Artsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{Ar})/n(\mathrm{H})$ the compared to the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. By setting `old_solar_abundances=.true.`, the values given by Grevesse and Sauval [1998] are used.

6.2.8 Cions

- *data type:* `integer`
- *data range:* $3 \leq$ `Cions` ≤ 4
- *default value:* 4
- *description:* Number of considered C ions, starting with the neutral stage and reaching to the `Cions`-1 times ionized stage.

6.2.9 c_correct_on

- *data type:* `logical`
- *default value:* `.true.`
- *requires:*
- *description:* The ray-tracing is stopped for cells, that could not be reached by the radiation of a source as $\Delta s > c \cdot t_{\mathrm{source}}$, where Δs is the distance between the source and the cell, c is the speed of light and t_{source} is the current lifetime of the source.
- *caveats:* While this option prevents ionization fronts to expand faster than the speed of light, it does not trace the "history" of the emitted photons, i.e. the code does not determine where the time position $\vec{s}(t)$ of photons emitted at t_{emit} as a function of time. The code therefore is not able to trace the radiation of sources that change their emission properties during timescales that are shorter than the light travel time as it is the case for supernova explosions or quasars with varying luminosity.

6.2.10 charge_transfer

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `include_metals=.true.`
- *description:* If the option is set to `.true.`, charge transfer processes are included for the computation of the ionization and recombination rates.
- *caveats:* Currently only the process H I/O II ¡–¿ H II/O I is implemented. Further tests are required.

6.2.11 compute_temperature

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If the option is set to `.true.` the thermal energy content of each grid cell is updated and the temperature recomputed after each timestep.
- *caveats:* The computation of the temperature does not account for the effects caused by the cosmological expansion.

6.2.12 CtoH

- *data type:* `real`
- *data range:* `CtoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio $n(\mathrm{C})/n(\mathrm{H})$.

6.2.13 Ctosolar

- *data type:* `real`
- *data range:* `Ctsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{C})/n(\mathrm{H})$ compared to the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. By setting `old_solar_abundances=.true.`, the values given by Grevesse and Sauval [1998] are used.

6.2.14 diffuse

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If this option is set, the diffuse radiation field is computed explicitly. Else the on-the-spot approximation is used for photons absorbed by H I, He I, and He II.
- *caveats:* The implementation is still work in progress and the code is so far not able to produce accurate results.

6.2.15 diffuse_random

- *data type:* `logical`
- *default value:* `.true.`
- *requires:* `diffuse=.true.`
- *description:* If this option is set, the directions of the rays emitted by each cell are random. This avoids systematic errors, but may induce "random noise"
- *caveats:* So far, always random directions are chosen. An non-random method will be implemented in the future.

6.2.16 diffuse_random_rays

- *data type:* `integer`
- *data range:* `diffuse_random_rays` ≥ 1
- *default value:* 1
- *requires:* `diffuse=.true.`, `diffuse_random=.true.`
- *description:* Number of rays emitted per cell to describe the diffuse radiation field using the "random-directions" approach.

6.2.17 diffuse_threshold

- *data type:* `real`
- *data range:* $0 \leq$ `diffuse_threshold` ≤ 1
- *default value:* 1.0^{-3}
- *description:* The diffuse radiation field is only computed if the electron density divided by the hydrogen density is larger than `diffuse_threshold`.

6.2.18 f_a

- *data type:* `real`
- *data range:* `f_a` > 0
- *default value:* 0.1
- *requires:*
- *description:* `f_a` controls the number of rays emitted per source in our ray-tracing approach. The number of rays N_{rays} is computed as

$$N_{\text{rays}} = 12 \times 4^{\left\uparrow \ln\left(4\pi \texttt{f\_a} \cdot \left((2\texttt{x\_max}+1)^2+(2\texttt{y\_max}+1)^2+(2\texttt{z\_max}+1)^2\right)\right)/\ln(4) \right\uparrow}, \tag{2}$$

  where the arrows indicate that the value is rounded to the next larger integer.

6.2.19 factor

- *data type:* `real`
- *data range:* `factor` > 0
- *default value:* 0.25
- *requires:*
- *description:* This value controls the lengths of the time-steps. The length Δt of the timesteps is computed by

$$\Delta t = \texttt{factor} \times \max\left(\frac{n_e}{\Delta n_e/\Delta t_{\text{prev}}}, \frac{n_{\text{HI}}}{\Delta n_e/\Delta t_{\text{prev}}}\right), \tag{3}$$

  where Δt is the length of the next timestep, n_e is the current number density of electrons, n_{HI} the current number density of neutral hydrogen, Δn_e the change of the electron density during the past timestep and Δt_{prev} is the length of the previous timestep. This computation is repeated for each cell of the simulation volume. The smallest value that is obtained will be used as the length of the next time step.

- *caveats:* If the simulation includes the computation of the temperature, we additionally try to limit the temperature change per timestep to 500 K by computing

$$\Delta t_T = \delta t_{\text{prev}} / \frac{\Delta T_{\text{prev}}}{500\,\text{K}}, \tag{4}$$

  where ΔT_{prev} is the maximal value change of temperature in a cell of the simulation value during the past time step. The value of 500 K is currently fixed in the source code. This is likely to be changed in the future. For the length of the next timestep, the smaller of the results of Eq. 3 and Eq. 4 is chosen.

6.2.20 filling_factor

- *data type:* `real`
- *data range:* $0 <$ `filling_factor` ≤ 1
- *default value:* 1
- *description:* Filling factor of the gas.
- *caveats:* Currently, the filling factor is assumed to be the same in the entire simulated volume, which might be a be unrealistic for actual physical systems.

6.2.21 H_0

- *data type:* `real`
- *data range:* `H_0` > 1
- *default value:* 71.0
- *description:* Hubble constant H_0 in $\mathrm{km} \cdot \mathrm{s}^{-1} \cdot \mathrm{Mpc}^{-1}$.

6.2.22 HetoH

- *data type:* `real`
- *data range:* `HetoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio $n(\mathrm{He})/n(\mathrm{H})$.

6.2.23 Hetosolar

- *data type:* `real`
- *data range:* `Hetsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{He})/n(\mathrm{H})$ compared to the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. By setting `old_solar_abundances=.true.`,
  the values given by Grevesse and Sauval [1998] are used.

6.2.24 ifrit_filename

- *data type:* `character (len=3000)`
- *default value:* STWITHD3D
- *description:* Name of the file for the input of the density structure. The default file name corresponds to the output of the `3dhydro_ext` program, `STWITHD3D`. For details see Section 5 and Section 7. The file format is compatible to the visualization program "IFrIT" provided by Nickolay Gnedin.

6.2.25 initial_temperature

- *data type:* `real`
- *data range:* `Ctsolar` > 0
- *default value:* 7500.0
- *requires:*
- *description:* Temperature of the gas at the beginning of the first timestep. The temperature is assumed to be equal for all grid points. Note that the value will be ignored, if the program is restarted with `-resume`. In this case, the program uses the temperature structure read in from the output file that the resumed run is based on

6.2.26 include_Ar

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, Ar is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of Ar is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.27 include_C

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, C is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of C is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.28 include_He

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, He is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of He is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.29 include_metals

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is set the ionization stages I to IV of carbon, nitrogen, oxygen, neon and sulfur are computed.
- *caveats:* Still work to be done concerning the atomic data.

6.2.30 include_N

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, N is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of N is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.31 include_Ne

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, Ne is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of Ne is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.32 include_O

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, O is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of O is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.33 include_S

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If this option is *not* set, S is not considered in the computation of the rate equations or in the radiative transfer.
- *caveats:* Currently, the memory that saves the abundances of S is still occupied and the corresponding output files are written. The abundances of each of the considered ionization stages are set to $10^{-150}\,\mathrm{cm}^{-3}$. The rationale for this behavior is to maintain compatibility with post-processing tools. It may be changed in the future.

6.2.34 initial_redshift

- *data type:* `real`
- *data range:* `initial_redshift` > 0
- *default value:* (none)
- *requires:* `trace_expansion=.true.`
- *description:* Redshift at the beginning of the simulated time.

6.2.35 l_cell

- *data type:* `real`
- *data range:* `l_cell` > 0
- *default value:* 1.0
- *description:* Length of a single grid cell in parsec.

6.2.36 last_step

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If the option is set, all values are written to files after completing the last time step. In this case the prefix of the file name is not the simulated time but the word `final`.
- *caveats:* This option has no effect if the program is stopped externally, e.g. by pressing CTRL+C or by invoking the `kill` command.

6.2.37 log_diffuse

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `diffuse=.true.`
- *description:* If this option the energy and photon emission by the recombination of hydrogen in the frequency range considered by the program is computed and written to standard output.
- *caveats:* May reduce the performance of the program. Not thoroughly tested yet.

6.2.38 log_heating_cooling

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `compute_temperature=.true.`
- *description:* The contributions of the different elements to the heating and cooling (in $\mathrm{erg} \cdot \mathrm{s}^{-1} \cdot \mathrm{cm}^{-3}$ are written in files with the suffixes `.xxheatcool.txt`, where `xx` is the name of the element in lower-case letters.
- *caveats:* Not thoroughly tested yet. Not all processes are considered yet. For example, the heating by ionization of metals is currently considered to be zero. For the file format, see Sect. 5.5.

6.2.39 lower_energy_limit

- *data type:* `real`
- *data range:* $0 <$ `lower_energy_limit` < 1
- *default value:* 83363.7/109678.8
- *description:* Sets the low value for the lowest photon energy considered.

6.2.40 mask_filename

- *data type:* `character(len=3000)`
- *default value:* `none`
- *requires:* `use_mask=.true.`
- *description:* Name of the file that contains the map for the cells that are "masked" out from the computation of the temperature and ionization structure as explained in more detail in the description of `use_mask`.
- *caveats:*

6.2.41 mergespec

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If this option is set, files with the WM-basic MERGESPEC format are used instead of of the format previously used by Sebastian Knogl's and Johann Weber's 3d Code.
- *caveats:* So far it is not possible to mix MERGESPEC-type values with the file format previously used by the 3d code.

6.2.42 Neions

- *data type:* `integer`
- *data range:* $3 \leq$ `Neions` ≤ 4
- *default value:* 4
- *description:* Number of considered C ions, starting with the neutral stage and reaching to the `Cions`-1 times ionized stage.

6.2.43 NetoH

- *data type:* `real`
- *data range:* `NetoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio n(Ne)/n(H).

6.2.44 Netosolar

- *data type:* `real`
- *data range:* `Netsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{Ne})/n(\mathrm{H})$ the compared to the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. By setting `old_solar_abundances=.true.`, the values given by Grevesse and Sauval [1998] are used.

6.2.45 nH_complete_scalar

- *data type:* `real`
- *data range:* `nH_complete_scalar` > 0
- *default value:* 10.0
- *requires:* `read_from_from_file=.false.`
- *description:* Number density of hydrogen in cm^{-3}. The number density refers to the total amount of hydrogen, i.e. it is the sum of the number densities of H I and H$\dot{\mathrm{II}}$. The same value is set for all grid cells. If `filling_factor` is smaller than one, the value refers to the mean number density within the volume, i.e. in the filled parts of the volume the number density of hydrogen is

$$\frac{\texttt{nH\_complete\_scalar}}{\texttt{filling\_factor}}. \tag{5}$$

6.2.46 nHI_fraction

- *data type:* `real`
- *data range:* $0 <$ nHI_fraction < 1
- *default value:* 0.9990
- *requires:* `read_from_from_file=.false.`
- *description:* Fraction of neutral hydrogen when compared to the total number density of hydrogen. The value is set for all grid cells.

6.2.47 no_source_filename

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If this option is activated, the data describing the ionizing sources are no longer read from the file specified by `source_filename`, but read from the file, that contains the options and is passed to `occup_3d_ext` via the standard input. The format of the data corresponds to that in the external file. There has to be a blank line between the namelist and the start of the list of sources[7].

[7] The list of sources usually starts wit a comment. If this is not the case there are two blank lines between the end of the namelist and the first entry of the file describing the sources. In this case the first entry is the number of sources.

- *caveats:* If this option is activated, the value of `source_filename` is ignored.
- *see also:* Description of the external file, that describes the sources.

6.2.48 NtoH

- *data type:* `real`
- *data range:* `NtoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio $n(\mathrm{N})/n(\mathrm{H})$.

6.2.49 Ntosolar

- *data type:* `real`
- *data range:* `Ntsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{N})/n(\mathrm{H})$ compared to the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. by setting `old_solar_abundances=.true.`,
  the values given by Grevesse and Sauval [1998] are used.

6.2.50 num_threads

- *data type:* `integer`
- *data range:*
- *default value:* 1
- *requires:* The code has to be generated with an compiler that supports OpenMP.
- *description:* Number of parallel OpenMP-threads used by the program. The maximal speed is reached if the number of threads corresponds to the number of CPU cores. An useful value is the number of available CPU cores minus the number of CPUs required for other purposes.
- *caveats:* Especially for relatively small simulated systems (in terms of grid cells), the performance may not be completely linear for a large value of `num_threads`. However for a grid with 101×101 cells, we found a good scalability for up to 12 threads (see B).

6.2.51 o_abundan

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *requires:*

- *description:* If this option is set, the metallicity is read from the `ABUNDAN` file generated by the `abund` program of WMbasic (The `ABUNDAN` file has to be located in the current directory). If the option is set, the values of the `Xtosolar` variables (where X is the element symbol of the corresponding element) are ignored. If a program is continued based on the output data of a former run using the `-resume` command line option, the element abundances will be read directly from the output of a former run. Thus neither the values of `Xtosolar` nor the values from the `ABUNDAN` file will be considered. If `optabu3d=.false.` one abundancd (relative to the number density of hydrogen) per element is read and the same chemical composition is assumed for each of the volume elements. Else the abundances will be read for each depth point.
- *caveats:* In `ABUNDAN` the abundances are written for all elements with atomic numbers from $Z = 1$(hydrogen) to $Z = 30$ (zinc). It is not possible to shorten the file, even if only a part of these elements are actually considered.

6.2.52 Oions

- *data type:* `integer`
- *data range:* $3 \leq$ `Oions` ≤ 4
- *default value:* 4
- *description:* Number of considered O ions, starting with the neutral stage and reaching to the `Arions`-1 times ionzed stage.

6.2.53 o_thermal_pressure

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *requires:* `compute_temperature=.true.`
- *description:* Computes the thermal pressure for each grid cell in units of $\mathrm{dyn\,cm^{-2}}$
- *caveats:* Accurate hydrodynamic computations would require the determination of the temperature and the chemical state (including metals) outside the ionized region. These computations are currently not done.

6.2.54 o_write_escape_fraction

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `periodic=.false.`
- *description:* Writes the escape fraction, defined by the fraction of the radiation escaping from the simulated volume, as a function of wavelength in a file with the name suffix `.escape.txt` after each timestep.
- *caveats:* For periodic boundary conditions, the escape fraction is per definition equal zero. Therefore this option may not be combined with `periodic`.

6.2.55 old_solar_abundances

- *data type:* `logical`
- *default value:* `.false.`
- *requires:*
- *description:* If this option sets the solar abundances given by Grevesse and Sauval [1998] instead of the newer ones by Asplund et al. [2009] are used.
- *caveats:*

6.2.56 omega_m

- *data type:* `real`
- *data range:* $0 <$ `omega_m` < 1
- *default value:* 0.269
- *requires:* `trace_expansion=.true.`
- *description:* Density (baryonic and dark) matter divided by the critical density of the universe.

6.2.57 optabu_3d

- *data type:* `logical`
- *data range:*
- *default value:* `.false`
- *requires:* `opt_abundan=.true.` If `optabu_3d` is set, the abundances are read separately for each of the depth points.
- *caveats:* For large simulation volumes the required
- *description:* `ABUNDAN` input files become large. For example, for an simulation volume containing 101^3 grid cells the required memory is $101^3 \times 30 \times 8$ bytes $\approx 235\,\mathrm{MB}$.

6.2.58 OtoH

- *data type:* `real`
- *data range:* `OtoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio $n(\mathrm{O})/n(\mathrm{H})$.

6.2.59 Otosolar

- *data type:* `real`
- *data range:* `Otsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{S})/n(\mathrm{H})$ compared to the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. by setting `old_solar_abundances=.true.`, the values given by Grevesse and Sauval [1998] are used.

6.2.60 periodic

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `o_write_escape_fraction=.false.`
- *description:* Enables periodic boundary conditions.
- *caveats:* The total distance that can crossed by rays emitted the sources is currently limited (in the different directions to) $2 \times x_{\max}$, $2 \times y_{\max}$ and $2 \times z_{\max}$, even if they leave the volume through one side and enter again though the opposing side. This is likely to be sufficient for the computation of the ionized volumes (reionization !), but there may be deviations in optically thin media, where the mean free path considerably exceeds the size of the edges of the considered volume. Do not try to compute the Lyman-α forest with the current implementation. "Longer" rays are in principle relatively simply to implement, but currently have no priority.
  If `periodic=.true.` it does not make sense to set `o_write_escape_fraction=.true.`.

6.2.61 photoheat_metals

- *data type:* `logical`
- *default value:* `.false.`
- *requires:* `compute_temperature=.true.`
- *description:* Enables the computation of the photoheating processes of metal atoms (and ions that can be further ionized).
- *caveats:* In the test case (H II region arround a D-40 star with solar metallicity of both the star and the gas) the consideration of the photoheating of metal atoms lead to an increase of the mean temperature (weighted by the square of the electron density) by just 30 K. It should also be noted that we currently do not consider the recombination cooling of metal ions. Setting `photoheat_metals` to `.true.` increases the required CPU time to compute the testcase by approximately 13 %.

6.2.62 read_from_file

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *requires:*
- *description:* Reads the density structure of the simulated volume from the specified file. The data format (cf. Sect. 5) of the file is compatible to the visualization program "IFrIT" provided by Nickolay Gnedin . Then first row contains the total number density of hydrogen, while the second contains the number density of neutral hydrogen atoms.
- *caveats:* So far it is not possible to specify the ionization structure of helium and metals at the start of the simulation. run It is just possible to start with mostly neutral or, if `start_ionized=.true.`, with mostly ionized gas.

6.2.63 rep_output

- *data type:* `integer`
- *data range:*
- *default value:* 1
- *requires:*
- *description:* The output files are written once after `rep_output` timesteps. The intention is to reduce the amount of used disk memory and the number of output files.

6.2.64 Sions

- *data type:* `integer`
- *data range:* $3 \leq$ `Sions` ≤ 4
- *default value:* 4
- *description:* Number of considered S ions, starting with the neutral stage and reaching to the `Sions`-1 times ionzed stage.

6.2.65 source_filename

- *data type:* `character (len=3000)`
- *data range:*
- *default value:* (none)
- *requires:*
- *description:* Specifies the name of the file that describes the sources. The file format is described in Sect. 5.
- *caveats:* If the option `no_source_filename` is set to `.true.`, the content of `source_filename` is ignored and the data are read in from the same file as the namelist containing the options.

6.2.66 start_ionized

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *requires:*
- *description:* If this option is activated, hydrogen and helium are assumed to be almost completely ionized (each of the other ionization stages has an abundance (relative to the total abundance of the element) of 10^{-10}). If the option is not set, the simulations start with the specified ionization fraction (either from `nHI_fraction` or from the file defined by `ifrit_filename`.

6.2.67 StoH

- *data type:* `real`
- *data range:* `StoH` > 0
- *default value:* 1.0E-12
- *requires:* `include_metals=.true.`
- *description:* Number ratio $n(\mathrm{S})/n(\mathrm{H})$.

6.2.68 Stosolar

- *data type:* `real`
- *data range:* `Stsolar` > 0
- *default value:* 1.0
- *requires:* `include_metals=.true.`, `abund_rel_to_H=.false.`
- *description:* Number ratio $n(\mathrm{S})/n(\mathrm{H})$ in compared to the the solar value.
- *caveats:* The absolute values given for the solar abundances vary in different publications. Per default, solar abundances as given by Asplund et al. [2009] are assumed. By setting `old_solar_abundances=.true.`, the values given by Grevesse and Sauval [1998] are used.

6.2.69 t_end

- *data type:* `real`
- *data range:*
- *default value:* 1.0^5
- *requires:*
- *description:* Time range of the simulation in years.

6.2.70 t_start

- *data type:* `real`
- *data range:*
- *default value:* 1.0^6
- *requires:*
- *description:* Length of the initial timestep in seconds. This value also corresponds to the minimal length of any further timestep.

6.2.71 t_step_max

- *data type:* `real`
- *data range:*
- *default value:* $4 \cdot 10^{12}$
- *requires:*
- *description:* Maximal length of a timestep in seconds.

6.2.72 use_mask

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *requires:*
- *description:* If this option is set, a file in the `IFrIT` uniform data textformat is chosen (the name is specified by the option `mask_filename`), that indicates, for which grid cells the temperature structure and (optionally) the ionization structure is not not recomputed. These cells are also not considered for the recomputation of the length of the subsequent timestep. The value 1 means that neither the ionization nor the temperature structure are recomputed. The value 10 leads to a behavior of the program where the ionization state is recomputed, but not the temperature structure. If the value of the cell in the file is 0 the cell is treated as "normal" cell where the temperature and ionization structure of the gas is recomputed and which is considered for the computation of the length of the next time step. All of the cells are still considered in the radiative transfer procedure. Thus the option `use_mask` can for example, be used to simulate volumes that contain "vacuum", i.e. cells that are optically extremely thin thin such that they do not influence the behavior of the other cells and therefore need not to be considered[8].

6.2.73 trace_expansion

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *requires:* `initial_redshift` has to be set to the redshift at the beginning of the simulated period.

[8] Due to numerical reasons it is not possible to assign a value for the density of exactly zero to a grid cell.

- *description:* Traces the expansion of space during the simulated time, by modifying the re-computing the value of `l_cell` after each timestep.
- *caveats:* The cosmological redshift of the ionizing radiation is not considered. Furthermore `occup_3d_ext` currently does not account for the effects of the expansion of space on the temperature structure. The formula used for the computation of the expansion assumes that the contribution of relativistic particles to the total energy density of the universe is negligible. This is a good approximation for $z < 100$, i.e. for the entire age where star formation in the universe is likely to occur.

6.2.74 verbose_stdout

- *data type:* `logical`
- *data range:*
- *default value:* `.false.`
- *description:* If the option is activated, the ionization fractions (relative to the total amount of the element within the simulated volume) are written to the standard output for each iteration step. Additionally, the maximal, the minimal, and the mean temperature (weighted by the square of the electron number density) of the gas are written to standard output.
- *caveats:* The CPU time required to compute the ionization fractions is approximately 8% of the total CPU time required by the program. As the corresponding routines are not explicitly parallelized, the effect on the wallclock times is in most cases larger (9% for one test run on a 4 Core machine with code compiled by Intel Fortran 13).

6.2.75 x_max

- *data type:* `integer`
- *data range:* `x_max` ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal x-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the x indexes range form $-$`x_max` to $+-$`x_max`. This means that the x-axis is divided into $2\,$`x_max` $+ 1$ segments.

6.2.76 y_max

- *data type:* `integer`
- *data range:* `y_max` ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal y-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the y indexes range form $-$`y_max` to $+-$`y_max`. This means that the y-axis is divided into $2\,$`y_max` $+ 1$ segments.

6.2.77 z_max

- *data type:* `integer`
- *data range:* `z_max` ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal z-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the z indexes range form $-$`z_max` to $+-$`z_max`. This means that the z-axis is divided into $2\,$`z_max` $+ 1$ segments.

7 3dhydro_ext

The aim of `3dhydro_ext` is to allow the generation of files that contain the description of different types of density structures. By default, the name of the output file of `3dhydro` is `STWITHD3DR`. This name is also the default value for the option ifrit_filename of the main program that specifies the name of the file that includes the density structure. However, the user is free to set ifrit_filename to another value. If the option `read_from_file` is set to `.false.`, the content of the output file will be ignored. In this case a homogeneous density structure will be assumed, which is controlled by the options `nH_complete_scalar` and `nHI_fraction`. Note that the program is not be able to generate arbitrary density structures. For complex density structures it will be necessary to generate the corresponding files using individual programs.
The input file has the structure

```
blablabla
```

The type of the density distribution is determined by the value of the variable `mode`. Each type of density distribution requires a set of additional variables, which are also read from the namelist[9]. First, we will explain some variables that affect all of the density distributions. Then we will explain variables that affect only some of the density distributions.
3dhydro Cartesian uses a Cartesian coordinate system where unity corresponds to the length of an edge of a grid cell, independently of the actual distance represented by this grid cell. If it is intended do apply scales that are provided in different units (e.g. parsec)

7.1 Variables that affect all density distributions

7.1.1 ensure_minimum

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If `ensure_minimum` is set, values for hydrogen number densities below nhyd_fill will be prevented.

7.1.2 macro_filling_factor

- *data type:* `real`
- *data range:* `l_cell` $0 \leq$ `macro_filling_factor` ≤ 1
- *default value:* 1.0
- *requires:* `random_mode=.true.`

[9] If a variable is not needed for the chosen type of density distribution, it is still read, but its value is ignored.

- *description:* The fraction of grid cells where the density corresponds to the one of the included density distributions. The rest of the cells will be reset to nhyd_fill.
- *caveats:* -The `macro_filling_factor` of `3dhydro` is different from the `filling_factor` of `occup_3d_ext`, where the microclumping approximation is applied, i.e. the clumps/inhomogeneities are assumed to be unresolved (much smaller than the grid cells) and optically thin.

7.1.3 nHI_fraction

- *data type:* `real`
- *data range:* `l_cell` $0 \leq$ `nHI_fraction` ≤ 1
- *default value:* 10.0
- *description:* Fraction of hydrogen atoms that are neutral. This value currently is the same for all grid cells.
- *caveats:* This value sets only the ionization fraction of hydrogen, but does not set the relative abundances of the various ionization stages of helium and the metals as initial conditions for the. simulation. `occup_3d_ext` assumes helium and the metals to be almost neutral[10] by default. If `start_ionized= .true.`, they are almost fully ionized. To provide different initial conditions for the ionization structure of helium and metals (and the temperature structure)

7.1.4 nhyd

- *data type:* `real`
- *data range:* `l_cell` > 0
- *default value:* 10.0
- *description:* Characteristic number density of hydrogen for a the corresponding density distribution. The exact interpretation of `nhyd` depends on the selected density distribution and will be explained in the corresponding subsections.

7.1.5 nhyd_fill

- *data type:* `real`
- *data range:* `l_cell` > 0
- *default value:* 10^{-2}
- *description:* The density of the gas in cells that are not otherwise filled. These cells are
  - Cells that are not filled by the spheres or ellipses crated with `mode=SPHERE` or `mode=ELLIPSOID`.
  - Cells that are reset, if `random_mode` is set to `.true.`.
  - If the density of one of the exponential, power-law or Gaussian distribution would lead to values below `nhyd_fill`, but simultaneously `ensure_minimum` is set to `.true.`

[10] All other ionization stages are assumed to have an abundance of 10^{-10} relative to the total abundance of the corresponding elements.

7.1.6 random_mode

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If activated, for each of the grid cells a random number between 0 and 1 is selected. If this number is smaller or equal than filling_factor, the density of the grid cell is kept. Else it will be reset to the value of nhyd_fill.

7.1.7 x_center

- *data type:* `integer`
- *data range:* `-x_max` $\leq$ `x_center` $\leq$`x_max`
- *default value:* 0
- *requires:*
- *description:* x-coordinate of the reference point $\vec{r_0}$ of the density distribution. It is always located in the center of a grid cell, which has to be addressed by an integer value. The exact interpretation of `x_center` depends on the selected density distribution and will be explained in the corresponding subsections.

7.1.8 x_max

- *data type:* `integer`
- *data range:* `x_max` ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal x-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the x indexes range form $-$`x_max` to $+-$ `x_max`. This means that the x-axis is divided into $2\,$`x_max` $+ 1$ segments (The variable has the same meaning as in `occup_3d_ext`).

7.1.9 y_center

- *data type:* `integer`
- *data range:* `-y_max` $\leq$ `y_center` $\leq$`y_max`
- *default value:* 0
- *requires:*
- *description:* y-coordinate of the reference point $\vec{r_0}$ of the density distribution. It is always located in the center of a grid cell, which has to be addressed by an integer value. The exact interpretation of `y_center` depends on the selected density distribution and will be explained in the corresponding subsections.

7.1.10 y_max

- *data type:* integer
- *data range:* y_max ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal y-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the y indexes range from $-$y_max to $+-$ y_max. This means that the y-axis is divided into $2\,$y_max $+ 1$ segments (The variable has the same meaning as in occup_3d_ext).

7.1.11 z_center

- *data type:* integer
- *data range:* -z_max $\leq$ z_center $\leq$z_max
- *default value:* 0
- *requires:*
- *description:* z-coordinate of the reference point $\vec{r_0}$ of the density distribution. It is always located in the center of a grid cell, which has to be addressed by an integer value. The exact interpretation of z_center depends on the selected density distribution and will be explained in the corresponding subsections.

7.1.12 z_max

- *data type:* integer
- *data range:* z_max ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal z-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the z indexes range from $-$z_max to $+-$ z_max. This means that the z-axis is divided into $2\,$z_max $+ 1$ segments (The variable has the same meaning as in occup_3d_ext).

7.2 Homogeneous density distributions (mode="HOMOGENEOUS")

As mentioned above, homogeneous density distributions can be set up directly in occup_3d_ext using the options nH_complete_scalar and nHI_fraction. Alternatively, the homogenous output can be written into the output file by setting mode="HOMOGENEOUS". The option needed to set up the density distribution is

- nhyd: Number density of hydrogen atoms in cm^{-3}.

7.3 Exponential, spherically symmetric density distribution (mode="EXPONENTIALRAD")

In this case the hydrogen number density in a grid cell with the positions x, y and z is computed by

$$n_H(x,y,z) = \texttt{nyd} \cdot \exp\left(-\frac{|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|}{\texttt{dens\_scale}}\right) \tag{6}$$

Here x_center, y_center, and z_center have the usual meaning and are the components of the vector . Here nhyd is the number density in the cell where the refence point is located. $|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|$ is the distance of the center of the cell with the indexes x, y and z (in parsec). The real value dens_scale (> 0) Is the characteristic distance scale, where the density is reduced to $\frac{1}{\mathrm{e}}$. Note that the program only considers the distances between the centers to the cells for the determination of the hydrogen density within the cells and there is no interpolation scheme. the results may therefore differ somewhat from the results from programs that assume smooth density distributions.

7.4 Exponential, plane-parallel density distribution (mode="EXPONENTIALPLANE")

I The orientation of the reference plane is defined by the option plane, which contains a single character. If plane="X", then the position of the plane is defined by $x =$ x_center. For plane="Y" and plane="Z" the positions of the symmetry planes are analogously defined by $y =$ y_center and $z =$ z_center resp.
Now we describe the density distribution for the example of a plane defined by a constant the x-axis (the formula has an analogous structure for planes defined by constant y or constant z)

$$n_H(x) = \texttt{nhyd} \cdot \exp\left(-\frac{|x - \texttt{x\_center}|}{\texttt{dens\_scale}}\right) \tag{7}$$

Here the real value dens_scale is the scale height of the density distribution (in parsec). Like in the plane-parallel case, only the centers of the cells are considered.

7.5 Spherically symmetric power-law density distribution with cutoff (mode="POWERRADCUTOFF")

In this case the hydrogen number density in a grid cell with the indices x, y and z is computed by

$$n_H(x,y,z) = \min\left(\texttt{cutoff}, \texttt{nhyd} \cdot \left(\frac{|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|}{\texttt{dens\_scale}}\right)^{-\texttt{dens\_exponent}}\right) \tag{8}$$

Here cutoff is maximal density, i.e. if the formula for the density sructure (second argument of min in Eq. 8) leads to a value larger than the cutoff-value it will be replaced by cutoff. $|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|$ is the distance of the center of the cell with the indexes x, y and z (in parsec). The real value dens_scale (> 0) has been provided to enable a more convenient notation of the density distribution.

7.6 Plane-parallel symmetric power-law density distribution with cuttoff (mode="POWERPLANECUTTOFF")

The orientation of the reference plane is defined by the option plane, which contains a single character. If plane="X", then the position of the plane is defined by $x =$ x_center. For plane="Y" and plane="Z" the positions of the symmetry planes are analogously defined by $y =$ y_center and $z =$ z_center resp.

Now we describe the density distribution for the example of a plane defined by a constant the x-axis (the formula has an analogous structure for planes defined by constant y or constant z)

$$n_H(x,y,z) = \min\left(\texttt{cutoff}, \texttt{nhyd} \cdot \left(\frac{|x - \texttt{x\_center}|}{\texttt{dens\_scale}}\right)^{-\texttt{dens\_exponent}}\right) \tag{9}$$

The real value `dens_scale` (> 0) has been provided to enable a more convenient notation of the density distribution.

7.7 Spherically symmetric Gaussian density distribution (`mode="GASUSSIAN"`)

In this case the hydrogen number density in a grid cell with the positions x, y and z is computed by

$$n_H(x,y,z) = \texttt{nyd} \cdot \exp\left(-\frac{|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|^2}{\texttt{sigma}^2}\right) \tag{10}$$

Here `x_center`, `y_center`, and `z_center` have the usual meaning. Here nhyd is the number density in the cell where the refence point is located. $|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|$ is the distance of the center of the cell with the indexes x, y and z (in parsec). `sigma` is a `real` value (> 0) and corresponds to the "standard deviation". Note that the program only considers the distances between the centers to the cells for the determination of the hydrogen density within the cells and there is no interpolation scheme. the results may therefore differ somewhat from the results from programs that assume smooth density distributions.

7.8 Fractal density structure (`mode="FRACTAL"`)

The algorithm for the generation of the density distribution is a multi-tier Monte Carlo Code. The simulated volume is divided into cubic grid cells of equal size, which correspond to the ones used for the radiative transfer. At the highest level N_1 points are distributed randomly in the entire volume. They can be located anywhere within a grid cell. For the next step N_2 points are placed around each of the points at the highest level. For each dimension, they are placed in a interval of $\pm\frac{\Delta_2}{2}$ around each of the highest level points Δ_1 is the entire length of an edge of the simulated volume. Each position within the interval is equally probable. The points of the second generation are placed in a cube around the points of the first generation. The quotient $\Delta_{1,2} = \frac{\Delta_1}{\Delta_2}$ can calculated as follows:

$$f = \frac{\log N}{\log \Delta_{12}} \quad \Longrightarrow \quad \Delta_{1,2} = N^{\frac{1}{f}} \quad , \tag{11}$$

where f is called fractal dimension and it is stored in the `real` variable `fractal_dimension`. If this fractal dimension would have been used on each of the scales, N had to be chosen as N_1 for the calculation of Δ_2. Yet in the simulation, N has a constant value of 32, independent of N_1, leading to a smoother distribution for higher numbers of N_1 (the number of first-generation points is stored in the integer variable `frac_tier_1`) while maintaining the fractal dimension on the smaller scale. As the method for determining the position of the third-generation points in analogous and the one of the larger-order tiers is analogous, the algorithm is implemented in an recursive way until the final generation is reached. The number of generations is stored in the `real` variable `frac_tiers.` The boundary conditions are periodic. This means that if the algorithm would result in a position outside the simulated range in x_n dimension, it is transfered inside the volume and given the position it had if x_n-axis the coordinate system was sifted in the direction of the particle by the length of the edge. The periodic boundary conditions make sure that the overall number of perticles does not depend on the positions of the selected points.

The fractal contribution of a single cell can be obtained by normalizing the included number $N_{5,cell}$ of points of the last generation with the mean value of all cells and multiplying the result with the given mean density of the fractal component of the DIG, which is stored in `nhyd.`. The values are then added to `nhyd_fill`.

8 Auxiliary utilities

Along with `occup_3d_ext` several utilities are provided that can be used for benchmarking purposes, postprocessing of the simulation results and to modify the initial conditions.

8.1 create_temperature

`create_temperature` provides a file that corresponds to the output files for the temperature (see Sect. 5.5). The intention of the program is to resume a run with the input option `compute_temperature=.true.` based on data of a run where `compute_temperature` has been set to `.false.`.

- *synopsis:*

  `echo` *x_max y_max z_max temperature* | `create_temperature` > outfile

  Here `x_max` is the maximal x-index of a grid cell, `y_max` is the maximal y-index of a grid cell, and `z_max` is the maximal x-index of a grid cell (see also the description of the corresponding options). *temperature* is the temperature of the gas in K.

- *output:* The first and the second column both contain in the temperature, whist the third column contains the number 0.00. The same values are assigned to all grid cells. The reason is that as the temperature structure is artificially generated, no "temperature" evolution could be traced. The new program run will use the temperature as initial condition.

- *caveats:* After the start of the new `occup_3d_ext` run, *ionization structure and temperature are not consistent !*. The option can therefore only be used if only the "stationary state" of the H II region is important. There are still some unresolved numerical issues. *Use at your own risk !*

8.2 nebstat

`nebstat` is a post-processing tool written in Fortran90, that prints statistical information about the simulated volume at a given timestep.

- *requires:* Data which have been generated with an run of `occup_3d_ext`, where the input variables `include_metals` and `compute_temperature` had both been set to `.true.`.

- *synopsis:*

  `nebstat` *filename_without_ending cellsize_in_pc*

  Here *filename_without_ending* means the part of the name, that is common to all the output files including the file number for the counter, but excluding the subsequent dot (cf. also Sect. 6.1).

- *output:* `nebstat` writes the following information to standard output:

  - The number fraction of the different ionization stages relative to the total abundance of the elements.
  - The fraction of grid cells, where a given ionization stages is the most abundant.

- The total emission of recombination and collisionally excited lines within the simulation values (in units of $\mathrm{erg\,s^{-1}\,cm^{-3}}$ and in comparison to the Hβ emission).
- The maximal value of emission among the cells per volume for some important recombination and collisionally excited lines (in units of $\mathrm{erg\,s^{-1}\,cm^{-3}}$ and in comparison to the Hβ emission).
- Maximal linestrength of different lines for projections along the coordinate axes

Additionally simulated narrowband filter images are created and written to IFrIT compatible files, one per emission line/emission line multiplet and direction of the projection.

8.3 radiusbench

`radiusbench` takes IFrIT files as input and writes the contained physical values in dependence of the distance from the center of the simulated value.

- *synopsis:*

  `radiusbench` *cellsize data_cols source_x source_y source_z infile outfile*

  Here *cellsize* is the size of a grid cell in parsec, *data_cols* is the number of columns in the input file, *infile* the name of the input file and *outfile* the name of the output file. *x_source*, *y_source* and *z_source* are the indizes of the cells where the source is located.

- *output:* The output corresponds to the input file, but additionally contains in its first column the distance between the center of the grid cell and the center of the central grid cell in units of parsec. The file does not contain an header specifying the extent of the volume and is therefore of compatible visualizations with IFrIT. The main application of the is program is to perform benchmarks.

- *caveats:* The program output contains the distance from the source, regardless of the position of the sources. If the source is not located in the center of the volume, the results may therefore be misleading.

8.4 spectinfo

`spectinf` is a tool to obtain statistical information about the synthetic spectra generated by `WM-Basic`[11].

- *synopsis:*

  cat mergespec | `spectinfo`

  Here *cat* is a program that takes a text file as input and writes it non-interactively to standard output. This may be the `cat` command for Unix or the `type` command for the windows command line shell. It is not possible to use interactive programs like `more`, `less`, or any text editor. *mergespec* is a copy of the file containing the model spectrum. The format has to correspond exactly to the `WM-Basic` output file `MERGESPEC`.

- output: The results are written to standard output. They contain the luminosity in erg per second and compared with the solar luminosity and the effective temperature that would be obtained from the spectrum.

  Furthermore the output contains the ionizing fluxes for several important ions, both as the common logarithms of the ionizing fluxes in photons per second and compared to a blackbody radiator with the same radius and effective temperature. For the He^+-ionizing fluxes the output contains the value for the correct ionization threshold, and for thresholds 2 % above (He II(+)) and below (He II(-)) the ionization energy of He II.

[11] cf. Pauldrach et al. [2001]

A Description of the namelist-based hydro code

The "hydro" code for the 3d radiative transfer has been rewritten in order to maintain compatibility with the `fsplit` command of the `zz2` package. A description of the new code can be found here. However, this lead to compromises concerning the fexibility of the code, as the number of arguments is now limited, whereas the namelist approach allows in principle for an arbitrary number of switches and options[12] Thus, the description of the namelist-based code is still included in this appendix. the source-code of the namelist-based code has been renamed to `3dhydro_ext_namelist.f90` and is included in the `zz2_3d_ext` subdirectory. To avoid the confusion with having two differnt programs with different interfaces doing essentially the same, it is, by default, not built when the `make` command is run (or the build process is started from the corresponding Visual Studio Solution file). However, it can still be built using the command
`make 3dhyro_ext_namelist`

A.1 Structure of the 3dhydro_ext_namelist input and output files

The aim of `3dhydro_ext` is to allow the generation of files that contain the description of various types of density structures. By default, the name of the output file of `3dhydro_ext_namelist` is `STWITHD3DR`. This name is also the default value for the option ifrit_filename of the main program that specifies the name of the file that includes the density structure. However, the user is free to set ifrit_filename to another value. If the option `read_from_file` is set to `.false.`, the content of the output file will be ignored. In this case a homogeneous density structure will be assumed, which is controlled by the options `nH_complete_scalar` and `nHI_fraction`. Note that the program is not be able to generate arbitrary density structures. For complex density structures it will be necessary to generate the corresponding files using individual programs.
The input file has the structure

blablabla

The type of the density distribution is determined by the value of the variable `mode`. Each type of density distribution requires a set of additional variables, which are also read from the namelist[13]. First, we will explain some variables that affect all of the density distributions. Then we will explain variables that affect only some of the density distributions.
3dhydro Cartesian uses a Cartesian coordinate system where unity corresponds to the length of an edge of a grid cell, independently of the actual distance represented by this grid cell. If it is intended do apply scales that are provided in different units (e.g. parsec)

A.2 Variables that affect all density distributions

A.2.1 ensure_minimum

- *data type:* `logical`
- *default value:* `.true.`
- *description:* If `ensure_minimum` is set, values for hydrogen number densities below nhyd_fill will be prevented.

A.2.2 macro_filling_factor

- *data type:* `real`
- *data range:* `l_cell` $0 \leq$ `macro_filling_factor` ≤ 1

[12] Both approaches do, however, not support the processing of array-like data, such as a density field containing a number of clumps, where the properties of each of the clumps are set explicitly).
[13] If a variable is not needed for the chosen type of density distribution, it is still read, but its value is ignored.

- *default value:* 1.0
- *requires:* `random_mode=.true.`
- *description:* The fraction of grid cells where the density corresponds to the one of the included density distributions. The rest of the cells will be reset to nhyd_fill.
- *caveats:* -The `macro_filling_factor` of `3dhydro` is different from the `filling_factor` of `occup_3d_ext`, where the microclumping approximation is applied, i.e. the clumps/inhomogeneities are assumed to be unresolved (much smaller than the grid cells) and optically thin.

A.2.3 nHI_fraction

- *data type:* `real`
- *data range:* `l_cell` $0 \leq$ `nHI_fraction` ≤ 1
- *default value:* 10.0
- *description:* Fraction of hydrogen atoms that are neutral. This value currently is the same for all grid cells.
- *caveats:* This value sets only the ionization fraction of hydrogen, but does not set the relative abundances of the various ionization stages of helium and the metals as initial conditions for the. simulation. `occup_3d_ext` assumes helium and the metals to be almost neutral[14] by default. If `start_ionized= .true.`, they are almost fully ionized. To provide different initial conditions for the ionization structure of helium and metals (and the temperature structure)

A.2.4 nhyd

- *data type:* `real`
- *data range:* `l_cell` > 0
- *default value:* 10.0
- *description:* Characteristic number density of hydrogen for a the corresponding density distribution. The exact interpretation of `nhyd` depends on the selected density distribution and will be explained in the corresponding subsections.

A.2.5 nhyd_fill

- *data type:* `real`
- *data range:* `l_cell` > 0
- *default value:* 10^{-2}
- *description:* The density of the gas in cells that are not otherwise filled. These cells are
  - Cells that are not filled by the spheres or ellipses crated with `mode=SPHERE` or `mode=ELLIPSOID`.
  - Cells that are reset, if `random_mode` is set to `.true.`.
  - If the density of one of the exponential, power-law or Gaussian distribution would lead to values below `nhyd_fill`, but simultaneously `ensure_minimum` is set to `.true.`

[14] All other ionization stages are assumed to have an abundance of 10^{-10} relative to the total abundance of the corresponding elements.

A.2.6 random_mode

- *data type:* `logical`
- *default value:* `.false.`
- *description:* If activated, for each of the grid cells a random number between 0 and 1 is selected. If this number is smaller or equal than filling_factor, the density of the grid cell is kept. Else it will be reset to the value of nhyd_fill.

A.2.7 x_center

- *data type:* `integer`
- *data range:* `-x_max` $\leq$ `x_center` $\leq$`x_max`
- *default value:* 0
- *requires:*
- *description:* x-coordinate of the reference point $\vec{r_0}$ of the density distribution. It is always located in the center of a grid cell, which has to be addressed by an integer value. The exact interpretation of `x_center` depends on the selected density distribution and will be explained in the corresponding subsections.

A.2.8 x_max

- *data type:* `integer`
- *data range:* `x_max` ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal x-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the x indexes range from $-$`x_max` to $+-$ `x_max`. This means that the x-axis is divided into $2\,$`x_max` $+ 1$ segments (The variable has the same meaning as in `occup_3d_ext`).

A.2.9 y_center

- *data type:* `integer`
- *data range:* `-y_max` $\leq$ `y_center` $\leq$`y_max`
- *default value:* 0
- *requires:*
- *description:* y-coordinate of the reference point $\vec{r_0}$ of the density distribution. It is always located in the center of a grid cell, which has to be addressed by an integer value. The exact interpretation of `y_center` depends on the selected density distribution and will be explained in the corresponding subsections.

A.2.10 y_max

- *data type:* integer
- *data range:* y_max ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal y-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the y indexes range from $-$y_max to $+-$ y_max. This means that the y-axis is divided into $2\,$y_max $+ 1$ segments (The variable has the same meaning as in occup_3d_ext).

A.2.11 z_center

- *data type:* integer
- *data range:* -z_max $\leq$ z_center $\leq$z_max
- *default value:* 0
- *requires:*
- *description:* z-coordinate of the reference point $\vec{r_0}$ of the density distribution. It is always located in the center of a grid cell, which has to be addressed by an integer value. The exact interpretation of z_center depends on the selected density distribution and will be explained in the corresponding subsections.

A.2.12 z_max

- *data type:* integer
- *data range:* z_max ≥ 0
- *default value:* 15
- *requires:*
- *description:* Sets the maximal z-index for a cell. Note that all indexes of the cell in the center of the volume is 0, and the z indexes range from $-$z_max to $+-$ z_max. This means that the z-axis is divided into $2\,$z_max $+ 1$ segments (The variable has the same meaning as in occup_3d_ext).

A.3 Homogeneous density distributions (mode="HOMOGENEOUS")

As mentioned above, homogeneous density distributions can be set up directly in occup_3d_ext using the options nH_complete_scalar and nHI_fraction. Alternatively, the homogenous output can be written into the output file by setting mode="HOMOGENEOUS". The option needed to set up the density distribution is

- nhyd: Number density of hydrogen atoms in cm^{-3}.

A.4 Exponential, spherically symmetric density distribution (mode="EXPONENTIALRAD")

In this case the hydrogen number density in a grid cell with the positions x, y and z is computed by

$$n_H(x,y,z) = \texttt{nyd} \cdot \exp\left(-\frac{|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|}{\texttt{dens\_scale}}\right) \tag{12}$$

Here x_center, y_center, and z_center have the usual meaning and are the components of the vector . Here nhyd is the number density in the cell where the refence point is located. $|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|$ is the distance of the center of the cell with the indexes x, y and z (in parsec). The real value dens_scale (> 0) Is the characteristic distance scale, where the density is reduced to $\frac{1}{\mathrm{e}}$. Note that the program only considers the distances between the centers to the cells for the determination of the hydrogen density within the cells and there is no interpolation scheme. the results may therefore differ somewhat from the results from programs that assume smooth density distributions.

A.5 Exponential, plane-parallel density distribution (mode="EXPONENTIALPLANE")

I The orientation of the reference plane is defined by the option plane, which contains a single character. If plane="X", then the position of the plane is defined by $x =$ x_center. For plane="Y" and plane="Z" the positions of the symmetry planes are analogously defined by $y =$ y_center and $z =$ z_center resp.
Now we describe the density distribution for the example of a plane defined by a constant the x-axis (the formula has an analogous structure for planes defined by constant y or constant z)

$$n_H(x) = \texttt{nhyd} \cdot \exp\left(-\frac{|x - \texttt{x\_center}|}{\texttt{dens\_scale}}\right) \tag{13}$$

Here the real value dens_scale is the scale height of the density distribution (in parsec). Like in the plane-parallel case, only the centers of the cells are considered.

A.6 Spherically symmetric power-law density distribution with cutoff (mode="POWERRADCUTOFF")

In this case the hydrogen number density in a grid cell with the indices x, y and z is computed by

$$n_H(x,y,z) = \min\left(\texttt{cutoff}, \texttt{nhyd} \cdot \left(\frac{|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|}{\texttt{dens\_scale}}\right)^{-\texttt{dens\_exponent}}\right) \tag{14}$$

Here cutoff is maximal density, i.e. if the formula for the density structure (second argument of min in Eq. 14) leads to a value larger than the cutoff-value it will be replaced by cutoff. $|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|$ is the distance of the center of the cell with the indexes x, y and z (in parsec). The real value dens_scale (> 0) has been provided to enable a more convenient notation of the density distribution.

A.7 Plane-parallel symmetric power-law density distribution with cutoff (mode="POWERPLANECUTTOFF")

The orientation of the reference plane is defined by the option plane, which contains a single character. If plane="X", then the position of the plane is defined by $x =$ x_center. For plane="Y" and plane="Z" the positions of the symmetry planes are analogously defined by $y =$ y_center and $z =$ z_center resp.

Now we describe the density distribution for the example of a plane defined by a constant the x-axis (the formula has an analogous structure for planes defined by constant y or constant z)

$$n_H(x,y,z) = \min\left(\texttt{cutoff}, \texttt{nhyd} \cdot \left(\frac{|x - \texttt{x\_center}|}{\texttt{dens\_scale}}\right)^{-\texttt{dens\_exponent}}\right) \tag{15}$$

The real value `dens_scale` (> 0) has been provided to enable a more convenient notation of the density distribution.

A.8 Spherically symmetric Gaussian density distribution (`mode="GASUSSIAN"`)

In this case the hydrogen number density in a grid cell with the positions x, y and z is computed by

$$n_H(x,y,z) = \texttt{nyd} \cdot \exp\left(-\frac{|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|^2}{\texttt{sigma}^2}\right) \tag{16}$$

Here `x_center`, `y_center`, and `z_center` have the usual meaning. Here nhyd is the number density in the cell where the reference point is located. $|\mathbf{r_{x,y,z}} - \mathbf{r_{center}}|$ is the distance of the center of the cell with the indexes x, y and z (in parsec). `sigma` is a `real` value (> 0) and corresponds to the "standard deviation". Note that the program only considers the distances between the centers to the cells for the determination of the hydrogen density within the cells and there is no interpolation scheme. the results may therefore differ somewhat from the results from programs that assume smooth density distributions.

A.9 Fractal density structure (`mode="FRACTAL"`)

The algorithm for the generation of the density distribution is a multi-tier Monte Carlo Code. The simulated volume is divided into cubic grid cells of equal size, which correspond to the ones used for the radiative transfer. At the highest level N_1 points are distributed randomly in the entire volume. They can be located anywhere within a grid cell. For the next step N_2 points are placed around each of the points at the highest level. For each dimension, they are placed in a interval of $\pm\frac{\Delta_2}{2}$ around each of the highest level points Δ_1 is the entire length of an edge of the simulated volume. Each position within the interval is equally probable. The points of the second generation are placed in a cube around the points of the first generation. The quotient $\Delta_{1,2} = \frac{\Delta_1}{\Delta_2}$ can calculated as follows:

$$f = \frac{\log N}{\log \Delta_{12}} \quad \Longrightarrow \quad \Delta_{1,2} = N^{\frac{1}{f}} \quad , \tag{17}$$

where f is called fractal dimension and it is stored in the `real` variable `fractal_dimension`. If this fractal dimension would have been used on each of the scales, N had to be chosen as N_1 for the calculation of Δ_2. Yet in the simulation, N has a constant value of 32, independent of N_1, leading to a smoother distribution for higher numbers of N_1 (the number of first-generation points is stored in the integer variable `frac_tier_1`) while maintaining the fractal dimension on the smaller scale. As the method for determining the position of the third-generation points in analogous and the one of the larger-order tiers is analogous, the algorithm is implemented in an recursive way until the final generation is reached. The number of generations is stored in the `real` variable `frac_tiers.` The boundary conditions are periodic. This means that if the algorithm would result in a position outside the simulated range in x_n dimension, it is transferred inside the volume and given the position it had if x_n-axis the coordinate system was sifted in the direction of the particle by the length of the edge. The periodic boundary conditions make sure that the overall number of particles does not depend on the positions of the selected points.

The fractal contribution of a single cell can be obtained by normalizing the included number $N_{5,cell}$ of points of the last generation with the mean value of all cells and multiplying the result with the given mean density of the fractal component of the DIG, which is stored in `nhyd.`. The values are then added to `nhyd_fill`.y

B Parallelization of the code and performance aspects

B.1 Motivation

Even if the 3D radiative transfer is solved using the approximations as discussed the "on-the-spot" approximation, it requires a large computational effort that scales at best linearly with the number of resolution elements, such that, for example, the computational effort to solve a simulation grid with 100^3 grid cells is at least 8 times larger than for the same simulation resolved in only 50^3 grid cells. The factor is likely to be considerably larger if the aim is to trace the temporal evolution of the system as the a smaller grid cell is crossed by an ionization front in a shorter time and therefore smaller time-steps have to be chosen. To be able to perform reasonably resolved simulations it is therefore essential that the code is "parallelized" which means that the computations are distributed among several CPUs located on a single computer and/or several computing nodes.

B.2 Parallelization of `occup_3d_ext` and performance measurements

As we run our simulations on multicore-workstations and our aim was a portable code, we have decided to parallelize our code using OpenMP. directives. We parallelized the radiative transfer routines as well as the solvers that compute the ionization and the temperature structure of the gas.
The radiative transfer has been parallelized such that if there are n processes, each of the these performs the radiative transfer for one n-th of the rays. The rays are distributed such that neighboring rays are in most cases distributed to different rays. This is done using the HEALPiX numeration scheme for the different directions (cf. Górski et al. 2005). The aim of this distribution is to minimize the effects of direction-dependent lengths of the rays which occur if a source is close to a border of the simulation volume. As the radiative transfer is halted when the border of the simulation volume is reached, the computational effort to trace shorter rays is smaller when compared to longer rays. Thus distributing the rays such that most neighboring rays belong to the same process leads to different run times of the processes, such that at the end of the radiation transfer routine, the processes tracing the short rays would be idle until the processes tracing the longer rays were finished.
Concerning the computation of the ionization and temperature structure we parallelize over the different $z-$indexes of the gird cells as the loop over these indexes is the outermost loop and therefore the computational effort to required for the synchronization of the processes is minimized. The disadvantage of this approach is that it scales not well if the number of processes is no longer small when compared to the number of cells along the z-axis. The reason is that the largest number of cells that have to be computed by an individual process to total time, is

$$N_{\mathrm{max\_per\_thread}} = N_x \cdot N_y \cdot \left\uparrow \frac{N_z}{N_{\mathrm{threads}}} \right\uparrow , \tag{18}$$

where N_x, N_y, and N_z are the number of cells along the coordinate axes and N_{threads} is the number of parallel threads. The arrows indicate that if the fraction is not equal to a integer, it has to the rounded to the next larger number. We show $N_{\mathrm{max\_per\_thread}}$ as a function of N_{threads} in Fig 1. The maximal time required for an individual thread corresponds to the total time to complete the computation of the corresponding loop, because no code after the loop can be executed before each of the processes has completed.

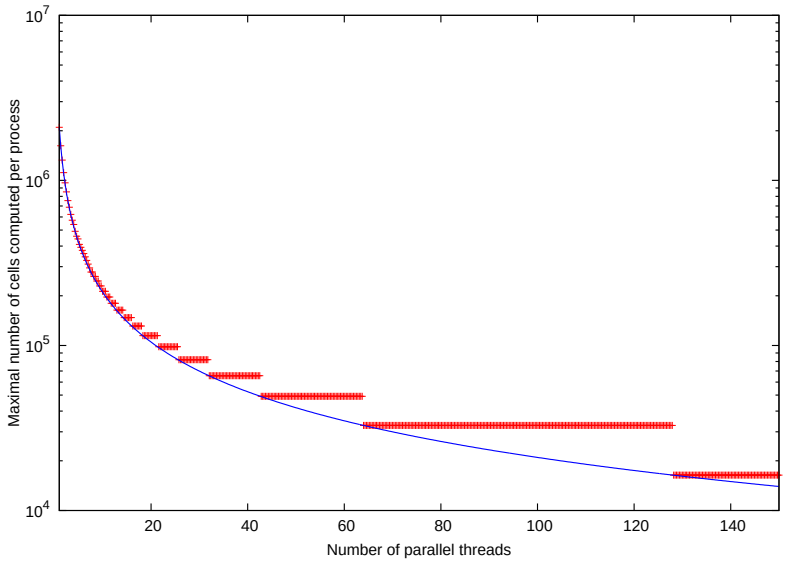

Figure 1: The position of the crosses corresponds to the maximal number of grid cells whose properties have to be computed by a individual thread as a function of the number of parallel process. As all cells with the same z-index are always computed by the same process, the maximal amount of grid cells that has to be computed by an individual process is no longer indirectly proportional to the number of parallel processes if the number of parallel processes is not small compared to the number of cells along the z-axis. For comparison, the blue line shows an ideal scaling behavior.

B.3 Tests

To test how the run time of our program scales with the number of parallel processes we have repeated the same simulation with 1 to 12 parallel processes[15]. The test simulation consists of a box with 41^3 cells, a simulated volume of $(31\,\mathrm{pc})^3$ a homogeneous hydrogen number density of $10\,\mathrm{cm}^{-3}$ and a chemical composition that corresponds to the solar values. As ionizing source we have chosen a 40 kK dwarf model with solar metallicity. The radiation field of the stars has been split into 196 608 rays. The simulation included computation of the temperature structure of the gas and the ionization structure of the contained metals. The simulated time was 30 000 yrs.. In each of our runs, the simulation was finished after 23 iteration steps [16]. In Fig. 2 we show the execution speed of the simulation (in units of iteration cycles per hour of wallclock time) as a function of the number of threads and compare it with the ideal case of a performance that scales linearly with the number of threads. In Fig. 3 we show the efficiency, i.e. the quotient of the measeured and the ideal performance. We find that the efficency drops to approximately 90 % when the number of processes is increased from 1 to 2. In the range from 2 to 12 processes the efficiency decreases more slowly until a value of approximately 80 % is reached.

References

M. Asplund, N. Grevesse, A. J. Sauval, and P. Scott. The Chemical Composition of the Sun. 47: 481–522, September 2009. doi: 10.1146/annurev.astro.46.060407.145222.

K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartel-

[15]The test have been performed on a workstation with 12 physical cores. So we have chosen 12 as the maximal number of parallel processes.

[16]We started with a fully ionized H II region which allows for larger timesteps.

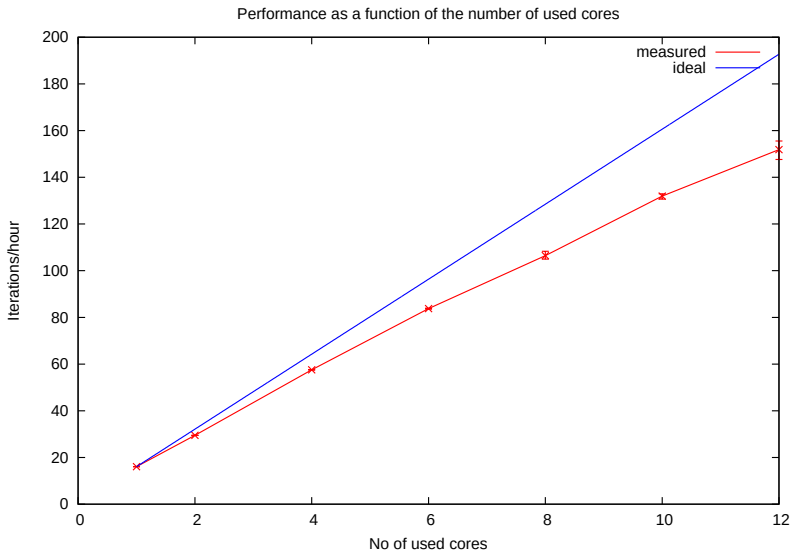

Figure 2: Performance of the simulation in iteration cycles per hour as a function of the number of computing threads. The red line shows the measured values. For the measurements with at least parallel processes we have performed several runs. In the case of 12 processes we found variations of the run time of approximately 2 %. The blue lines shows a linear scaling of the performance for one process with the number of processes, which would describe the performance for an ideal parallelization.

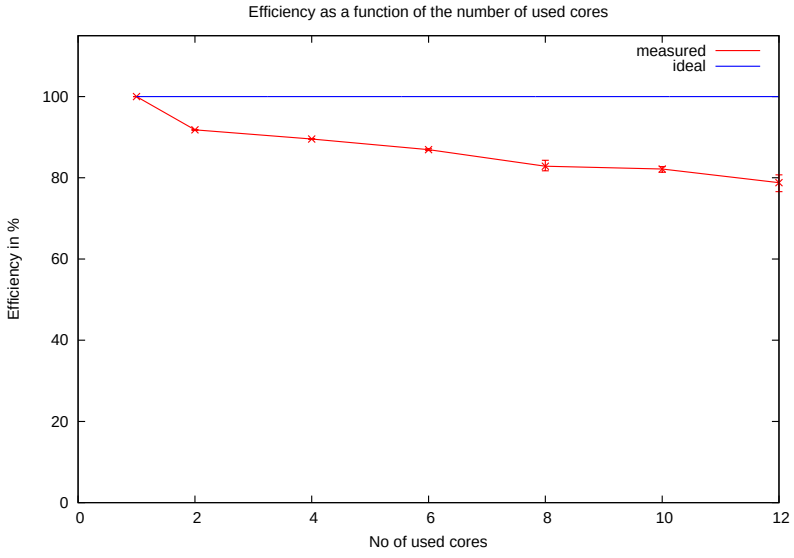

Figure 3: The efficiency of the parallelization drops to approximately 80 % for 12 parallel processes.

mann. HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *ApJ*, 622:759–771, April 2005. doi: 10.1086/427976.

N. Grevesse and A. J. Sauval. Standard Solar Composition. *Space Sci. Rev.*, 85:161–174, may 1998.

T. L. Hoffmann, S. Lieb, A. W. A. Pauldrach, H. Lesch, P. J. N. Hultzsch, and G. T. Birk. Numerical models for the diffuse ionized gas in galaxies. I. Synthetic spectra of thermally excited gas with turbulent magnetic reconnection as energy source. *Astronomy and Astrophysics*, 544: A57, August 2012. doi: 10.1051/0004-6361/200811512.

A. W. A. Pauldrach, T. L. Hoffmann, and M. Lennon. Radiation-driven winds of hot luminous stars. XIII. A description of NLTE line blocking and blanketing towards realistic models for expanding atmospheres. *Astronomy and Astrophysics*, 375:161–195, August 2001. doi: 10.1051/0004-6361:20010805.

J. A. Weber, A. W. A. Pauldrach, J. S. Knogl, and T. L. Hoffmann. Three-dimensional modeling of ionized gas. I. Did very massive stars of different metallicities drive the second cosmic reionization? *Astronomy and Astrophysics*, 555:A35, July 2013. doi: 10.1051/0004-6361/201220897.

J. A. Weber, A. W. A. Pauldrach, and T. L. Hoffmann. Three-dimensional modeling of ionized gas. II. Spectral energy distributions of massive and very massive stars in stationary and time-dependent modeling of the ionization of metals in HII regions. *ArXiv e-prints*, January 2015.