# MURaM Parameter file

Matthias Rempel

September 19, 2025

## 1 Introduction

Values in [] indicate default settings. Note that most parameters are mandatory, i.e. the code will terminate if they are missing in the parameters.dat files.

## 2 Grid Parameters

This set of parameters defines grid size and extent, as well domain decomposition:

**NDIM (int) [3]:** Grid dimensions (1-3), currently only 3 has been validated for the GPU version of the code

**gxmin (double[NDIM]) [0,0,0]:** Lower domain extent for each dimension [cm]

**gxmax (double[NDIM]) [1,1,1]:** Upper domain extent for each dimension [cm]

**gsize (int[NDIM]) [1,1,1]:** Grid size for each dimension. The combination of gxmax, gxmin and gsize defines the grid spacing.

**ghosts (int[NDIM]) [0,0,0]:** Ghost cells for each direction. The currently implemented 4th order stencil requires 2 ghost cells.

**periods (int[NDIM]) [0,0,0]:** Grid directions with periodic treatment. *It is the responsibility of the user to implement proper boundary conditions for the non-periodic grid directions.* Currently non-periodic boundary conditions are only available for the x-direction (vertical)

**pardim (int[NDIM]) [0,0,0]:** No longer used (was needed when code used auto domain decomposition). Superseded by procs.

**procs (int[NDIM]) [1,1,1]:** Number of MPI processes for each dimension. Make sure the total number of MPI ranks is equal to the number of MPI processes specified in batch script, otherwise the code terminates.

# 3 Run Parameters

These settings determine the runtime behavior, such as time stepping and IO frequency:

**anlfile (char) [anls.log]:** Name of file for logging output from AnalyzeSolution (global quantities like min, max, rms values).

**resfile (char) [result]:** Name of the MHD state vector file

**backfile (char) [backup.dat]:** Name of file logging last written snapshot, including iteration, time, pressure and entropy at bottom boundary

**maxiter (int) [1,000,000]:** Maximum number of iterations

**anlfreq (int) [1,000,000]:** Frequency for writing analysis file (anlfile)

**resfreq (int) [1,000,000]:** Frequency for writing full output (MHD state vector + EOS and other diagnostic variables)

**backfreq (int) [1,000,000]:** Frequency for rolling backups, writes only restart (MHD variables + hyperbolic heat conduction), these files are erased when the next one is written. Only files written at resfreq are kept.

**outcad (double) [0]:** If set to a value $> 0$, full result files are written at this time cadence. The numerical time step is adjusted to hit this frequency exactly

**slicefreq (int) [1,000,000]:** Frequency for writing various 2D and 1D files. These include tau-slices, xy, xz, yz (2D) slices, emission measure (2D + temperature) and horizontal averages (1D)

**dt (double) [0]:** If set to a value $> 0$ the code uses a fixed time-step. The code still checks if the CFL condition is met and will terminate if the choice of dt violates the CFL condition

**Tmax (double) [1]:** Maximum physical time for simulation

**CFL (double) [0.8]:** CFL number, the numerical scheme of MURaM (4th order finite difference + 4 sub step time-stepping scheme) can run with CFL numbers of up to 2. This is typically only recommended for an-isotropic grid spacing (like $\Delta x < 0.5(\Delta y, \Delta z)$). For isotropic grids we recommend CFL $< 1.5$.

**CFL_tvd (double) [0.95]:** The slope-limited diffusion scheme can only work with CFL number $< 1$. If the code uses CFL numbers $> 1$, this leaves 2 options: (1) call the TVD scheme with a reduced diffusivity (may cause some spurious oscillations) or (2) call the TVD scheme twice with half the time-step. CFL_tvd is the CFL number at which the switch over between these two options is happening, i.e. for CFL $\lesssim$ CFL_tvd the code stays in regime (1), for CFL $>$ CFL_tvd in regime (2)

**maxWtime (double) [1e7]:** The wallclock time at which the code will terminate. If maxWtime is set to a negative number the code will write out a restart file before terminating

**comment (char) [ ]:** Optional comment, currently not used for anything

**verbose (int) [0]:** Verbosity of comments written into the runtime logfile. The maximum setting is 4

**path_3D (char) [ ]:** Path for storing all 3D variables, a separation of the IO paths can be useful on Lustre filesystems, where larger 3D cubes may require different stipe sizes and counts than the smaller 2D and 1D files for best IO performance

**path_2D (char) [ ]:** Path for storing all 2D and 1D variables (slice files, emission measure, horizontal averages)

**eos_name (char) [eos.dat]:** Equation of state table

**kappa_name (char) [kappa.dat]:** Opacity table

**eos_output (int[15]) [1,1,0,0,....]:** Optional output of derived variables. By default the code writes out temperature and pressure, additional quantities depend on the implemented physics, check the routine eos_output (in io_xysl.C) for further detail

**diagnostics (int) [0]:** Enables optional diagnostic output for variables that are easier to compute at runtime.

**diag_output (int[11]) [0,0,....]:** Diagnostic variables to be written. Currently implemented are the terms in the energy equation including numerical resistive and viscous as well as ambipolar heating. Note that numerical heating terms cannot be easily computed from the regular output and need to be written out at runtime if needed

**HAVG (int) [0]:** Horizontally averaged quantities. Check the routine analysis_hmean* for what is implemented

**DEM (int) [0]:** Output of emission measure. If enabled, EM, vlos, vrms and effective fill factor will be written for $\lg(T) > 4.5$ for the 3 coordinate directions

**RT_HAVG (int) [0]:** Output of horizontally averaged radiative transfer quantities. [This was mostly used for debugging and has not been checked for correctness recently]

# 4 Physics Parameters

The physics data structure collects all parameters that enter the equations solved by the MURaM code. Physics does include here also numerical terms such as the semi-relativistic Alfven speed limiter, numerical diffusivities, div B cleaning etc.

## 4.1 Physics

**param_gravity (double) [0]:** Value of gravitational acceleration $[cm^2/s]$. The code assumes that the direction of gravity is in the x-direction. A positive value means gravity pointing in the negative x-direction

**param_va_max (double) [1e7] and param_va_adjust (double) [0]:** Meaning depends on the setting of param_va_adjust: (0) Maximum allowed value for the Alfvén velocity is given by param_va_adjust [cm/s], which is limited through the Boris correction (semi-relativistic MHD with a reduced speed of light). For this setting advection velocities are further limited to param_va_max/3. ($\geq 1$) The Alfvén velocity is dynamically adjusted, in this case param_va_max is the minimum value for the dynamically adjusted maximum Alfvén speed. The dynamically adjusted Alfvén velocity is given by max(param_va_max, param_va_adjust*3*v_max, param_va_adjust*2*cs_max)

**param_max_fill (double) [1000]:** When using param_va_adjust $> 0$, the code also imposes a dynamical ceiling on v and eint/rho. This dynamical ceiling is raised when more than param_max_fill grid points exceed current values and lowered if less than param_max_fill/2 grid points reach current values. The purpose of this dynamical ceiling is to prevent severe time-step constraints from a small value of grid points with extreme values

**param_spitzer (double) [0]:** Value of the coefficient for the Spitzer conductivity $\kappa_s =$ param_spitzer $T^{2.5}$. Typical value is $10^{-6}$

**param_eta (double) [0]:** Magnetic diffusivity [$cm^2/s$]. By default the code uses numerical magnetic diffusivity computed through a slope-limited diffusion scheme. param_eta allows to add an explicit homogeneous magnetic diffusivity

**param_ambipolar (double) [0], param_ambfac_max (double) [1e12], param_ambvel_max (double) [1e7]:** Parameters controlling a hyperbolic implementation of ambipolar diffusion. [The current implementation is outdated and needs to be updated. Don't use!]

## 4.2 Boundaries

**bnd_top (double) [0]:** Determines treatment of upper boundary. A value of 0 leads to a closed top boundary (antisymmetric vertical mass flux). A value of 1 leads to a half open boundary at which upflows are allowed (symmetric mass flux) and downflows are suppressed (antisymmetric mass flux). A choice in the range (0 ... 1) uses a half open boundary with an imposed flow velocity limit (relative to the max velocity found in the simulation domain). This choice was implemented to prevent situations at which time-step limiting strong outflows develop at the top boundary.

**bnd_pot (double) [0]:** A choice of 0 uses a vertical magnetic field boundary, a choice of 1 a potential field extrapolation, which is updated every sub-step of the 4-step time integration scheme. A choice of $n \leq 2$ will only compute the potential field every $n-1$ full time step to save computing time. This is typically not necessary if the HeFFTe library is used (2D domain decomposition for FFTs), but can help with the FFTW library that uses only a 1D domain decomposition for FFTs and has worse scaling.

**bnd_bcrit (double) [1e10]:** Transition from an open to a closed bottom boundary when the vertical magnetic field amplitude exceeds this value. The massflux boundary transitions from symmetric to antisymmetric for all three mass flux components.

**bnd_eps_top (double) [0]:** "Hot plate" top boundary. This is useful for domains that include a transition region, but are too small to produce a self-sustained corona. This boundary condition imposes a specific internal energy of the given value to mimic a hot corona. This makes only sense if both heat conduction and optically thin radiative loss are enabled.

**bnd_fem (double) [0]:** Placeholder for flux emergence boundary, currently not implemented

## 4.3 Numerical Diffusivity

Numerical diffusivities are computed in MURaM using a dimensional split scheme after the full MHD updated, i.e. they are added as a filtering step after the 4 step time integration scheme is complete. The diffusivities are computed using the following expressions (see Rempel (2014)). The first step is computed extrapolated values at cell interfaces:

$$u_l = u_i + 0.5\,\Delta u_i \tag{1}$$

$$u_r = u_{i+1} - 0.5\,\Delta u_{i+1}\ . \tag{2}$$

The reconstruction slopes $\Delta u_i$ are computed using the monotonized central difference limiter, given by

$$\Delta u_i = \text{minmod}\,[(u_{i+1} - u_{i-1})/2, \tag{3}$$

$$2\,(u_{i+1}-u_i), 2\,(u_i-u_{i-1})]\ .$$

Numerical fluxes at cell interfaces follow from:

$$f_{i+\frac{1}{2}} = -\frac{1}{2}\,\mathrm{tvd_{coeff}}\,\max(c_i, c_{i+1})\,\Phi_h(u_r-u_l, u_{i+1}-u_i)\cdot(u_r-u_l)\ . \tag{4}$$

Here $c_i$ is the characteristic velocity given by (for clarity we leave out here details about the Boris correction, but note that Alfvén velocity is limited)

$$c = |\mathbf{v}| + \sqrt{(\mathrm{tvd_{cs}}\,C_s)^2 + V_A^2} \tag{5}$$

The function $\Phi_h$ is given by

$$\Phi_h = \max\left[0, 1+\mathrm{tvd_h}\left(\frac{u_r-u_l}{u_{i+1}-u_i}-1\right)\right] \tag{6}$$

The primary control parameters for the numerical diffusivity are $\mathrm{tvd_{coeff}}$, $\mathrm{tvd_{cs}}$, and $\mathrm{tvd_h}$. Since the simulation domains are strongly stratified, the code can uses different setting in different parts of the domain to allow for more flexibility.

While the above numerical dissipation scheme is based on TVD schock capturing schemes such as the $2^{nd}$ order TVD Lax-Friedrichs scheme, the TVD property is only (approximately) maintained when using $\mathrm{tvd_{cs}} = 1$ and $\mathrm{tvd_h} = 0$. Using settings of $\mathrm{tvd_{cs}} < 1$ and $\mathrm{tvd_h} > 0$ can dramatically reduce numerical diffusivity, but this comes at the expense of potentially creating strong oscillations near discontinuities. Since we maintain the full numerical diffusivity near monotonicity changes the numerical scheme does however remain stable in those circumstances (providing that settings in Tcheck prevent negative internal energies). It is therefore critical to carefully evaluate which settings are appropriate for the setup considered. On the one hand, for any shock dominated problem (e.g. 1D Brio Wu shock tube test setups) only $\mathrm{tvd_{cs}} = 1$ and $\mathrm{tvd_h} = 0$ should be used, on the other hand, for simulations of the convection zone and photosphere a setting of $\mathrm{tvd_{cs}} = 0.2$ and $\mathrm{tvd_h} = 2$ is fine. For the more shock dominated chromosphere we recommend $\mathrm{tvd_h} = 1$.

**tvd_h (double[4]) [2,2,2,2]:** Value of $\mathrm{tvd_h}$ for the four regions (bottom boundary, lower domain, upper domain, top boundary)

**tvd_cs (double[4]) [0.2,0.2,0.2,0.2]:** Value of $\mathrm{tvd_{cs}}$ for the four regions (bottom boundary, lower domain, upper domain, top boundary)

**tvd_rholev (double) [1e-11]:** Density value that discriminates the lower and upper domain for the above settings. Since physical quantities are horizontally homogeneous (e.g. magnetized vs un magnetized regions), a density based delimiter is more meaningful than a constant height

**tvd_coeff (double[4]) [1,1,1,1]:** Value of $\text{tvd}_{\text{coeff}}$ for [$\varrho$, $\mathbf{v}$, $\varepsilon$, $\mathbf{B}$]. Note that values ¡1 typically lead to instability unless other explicit diffusivities are added for that variable. An example would be setting param_eta and tvd_coeff =[1,1,1,0] to disable numerical resistivity. If the desire is to lower numerical diffusivity increase tvd_h and lower tvd_cs.

**tvd_rholog (int) [1]:** Apply numerical diffusivities to $\log \varrho$ and $\log \varepsilon$

**tvd_qrho (double) [10]:** Limit jumps in density between neighboring grid cells to this value by increasing mass diffusivity (through setting $\Delta u_i = 0$)

**tvd_Bpar (double) [0.2]:** Reduce magnetic diffusivity in the direction of the magnetic field (this term is typically not needed and produces a larger $\nabla \cdot \mathbf{B}$ error that needs to be cleaned again)

**tvd_vhyp (double) [1]:** An additional $4^{th}$ order hyperdiffusivity in the x-direction (i.e. vertical direction). The hyperdiffusivity is added applied to $\varrho$, $\mathbf{v}$, $\varepsilon$. The hyperdiffusivity scales with $|\mathbf{v}|$. This term allows to reduce even-odd grid oscillation in the x-direction that are too small to cause monotonicity changes and go mostly undetected by the slope-limited diffusion scheme.

**tvd_Qdiff_bnd (double) [1]:** This parameter allows to reduce or disable the heating from numerical resistivity and viscosity at the top boundary.

**tvd_pm_v (double) [1]:** This parameter allows to set the numerical viscosity different from the chosen tvd_h value, i.e. the code uses for the velocity field a value of tvd_pm_v * tvd_h. This is applied only in the upper domain (i.e. above tvd_rho_lev).

**tvd_pm_B (double) [2]:** This parameter allows to set the numerical resistivity different from the chosen tvd_h value, i.e. the code uses for the magnetic field a value of tvd_pm_B * tvd_h. This is applied only in the upper domain (i.e. above tvd_rho_lev).

**tvd_vmax_lim (double) [0.75]:** If the velocity exceed this threshold (specified to the vmax threshold either set through the tchk_vmax or the dynamically adjusted vmax ceiling when using param_va_adjust = 1), the numerical viscosity is increased by setting the reconstruction slopes to zero.

**tvd_CME_thresh (double) [100]:** This setting allows to enhance the numerical diffusivity in regions where $|\mathbf{v}| >$ tvd_CME_thresh $* C_{\text{fast}}$, a condition that separates out the cool dense cores of coronal mass ejections. Note that the code is stable without this setting, but it increases the smoothness of mass ejecta.

**tvd_h_bnd (double[2]) [0.01, 0.01]:** Defines the width of the boundary region at bottom and top of domain. A value $< 1$ is interpreted relative to the vertical domain extent, a value $> 1$ is interpreted as number of grid cells in the boundary regions.

**tvd_visc_bnd (double[2]) [1, 1]:** tvd_coeff for velocity in boundary regions (mostly relevant if tvd_coeff is set to a value $< 1$). A value $> 1$ sets the tvd_coeff to 1 and in addition reduces the reconstruction slope in the boundary region by a factor of (tvd_visc_bnd-1). A value of 2 will maximize numerical viscosity at the boundary.

**tvd_eta_bnd (double[2]) [1, 1]:** tvd_coeff for magnetic field in boundary regions (mostly relevant if tvd_coeff is set to a value $< 1$). A value $> 1$ sets the tvd_coeff to 1 and in addition reduces the reconstruction slope in the boundary region by a factor of (tvd_eta_bnd-1). A value of 2 will maximize numerical resistivity at the boundary.

### 4.4 div B cleaning

In MURaM $\nabla \cdot \mathbf{B}$ is controlled to the hyperbolic divergence cleaning of Dedner (2000). We did separate the divergence cleaning from the integration of the MHD equations in order to allow for more control, i.e. the divergence cleaning can be called multiples times to allow for a stronger reduction of the $\nabla \cdot \mathbf{B}$ error.

**divB_switch (int) [1]:** (0) disable, (1) enable $\nabla \cdot \mathbf{B}$ cleaning

**divB_itmax (int) [5]:** Maximum number of iterations in $\nabla \cdot \mathbf{B}$ cleaning scheme

**divB_err (double) [0.2]:** Factor by which the maximum $\nabla \cdot \mathbf{B}$ error in the domain is reduced, i.e. a value of 0.2 means that the scheme iterates until the maximum $\nabla \cdot \mathbf{B}$ error is reduced by a factor of 5 or the maximum iteration count is reached

### 4.5 Tcheck

**tchk_eps_min (double) [1e11]:** Minimum value for the specific internal energy $\varepsilon = E_{\rm int}/\varrho$. This is mostly relevant to very cold regions that form in the upper photosphere/lower chromosphere due to adiabatic expansion.

**tchk_rho_min (double) [1e-16]:** Minimum value for the density.

**tchk_eps_max (double) [4e15]:** Maximum value for the specific internal energy $\varepsilon = E_{\rm int}/\varrho$.

**tchk_vmax (double) [1e7]:** Maximum value for the flow velocity. If the flow velocity is limited, the removed kinetic energy is removed from the system.

### 4.6 Damping

The code damps the "box mode" by adding the terms:

$$\frac{\partial \varrho v_x}{\partial t} = [\ldots] - \frac{\varrho v_x}{\tau_{\rm dmp}} \tag{7}$$

$$\frac{\partial E_{\rm plasma}}{\partial t} = [\ldots] - \frac{\varrho v_x^2}{\tau_{\rm dmp}} \tag{8}$$

The damping time scale $\tau_{\rm dmp}$ is a function of depth and is adjusted based on the amplitude of the box mode through ($\langle \ldots \rangle$ denotes the horizontal average, $\varrho_{\rm bot}$ the mass density at the bottom of the domain):

$$\frac{1}{\tau_{\rm dmp}} = \min \left[ \frac{1}{\tau_{\rm min}}, \frac{1}{\tau_{\rm ref}} \left( \frac{\langle \varrho v_x \rangle}{v_{\rm ref}\, \varrho_{\rm bot}} \right)^4 \right] \tag{9}$$

Most of the damping is applied near the bottom boundary of the domain (largest mode mass).

**dmp_switch (int) [0]:** Enable/disable damping of the "box mode", i.e. oscillation that lifts the content of the entire simulation domain up and down.

**dmp_tau_ref (double) [1e3]:** Reference damping time scale

**dmp_vel_ref (double) [1e3]:** Reference velocity amplitude (corresponding to mode amplitude at the bottom of the domain) to which the reference damping time scale is applied

**dmp_tau_min (double) [1e2]:** Shortest allowed damping time scale

### 4.7 Radiative Transfer

**rt_update (int) [1]:** Update RT only every rt_update time steps. This is only happening if that does not violate the RT time step constraint.

**rt_tau_min (double) [1e-8]:** Taper off $Q_{\rm rad}$ for optical depths smaller than this value

**rt_tr_tem (double) [2e4] and rt_tr_pre (double) [1e2]:** Define position of transition region ( T > rt_tr_tem and p < rt_tr_pre ) at which RT is switched off.

**rt_pre_cut (double) [1e2]:** Taper off optically thin radiative loss for p > rt_pre_cut. Note that this can lead to numerical instabilities for flare simulations, there it is better to switch off optically thin loss based on the position of the transition region.

**rt_tstep (double) [0]:** Set a minimum time step for RT relative to the MHD time step. If the RT time step falls below this value, $Q_{\rm rad}$ is truncated in regions that cause the time step constraint.

**rt_cfl (double) [0.5]:** CFL number for radiative time step: $dt_{\rm rad} = {\rm CFL}_{\rm rt}\, E_{\rm int}/Q_{\rm rad}$

**rt_type (double) [0]:** [Currently not used]

**rt_epsilon (double) [0]:** [Currently not used]

**rt_iout (double) [0]:** Type of diagnostic intensity output: (0) zero$^{th}$ (continuum) opacity bin, (1) 500nm continuum intensity, (2) bolometric intensity. Note that for gray simulations (0 and (2) are identical.

**ext_cor (double) [0]:** (1) Switch on optically thin loss in corona

### 4.8 Slice IO

These parameters control the slice IO, belong logically more into the Run data structure but ended up here

**sl_collect (int) [0]:** (0) Write every time step in separate file; (1) write all time steps into same file [This has not been used for a long time and needs to be tested, restart of code can lead to duplicate entries and termiation of code during write can lead to corrupted files, strongly recommend (0)].

**sl_I_out (int) [0]:** (1) Write intensity files (vertical ray only , see rt_iout for additional options). If (0) is selected the intensity can also be included in either tau or yz slices.

**sl_tau (int) [0]:** Number of tau slices

**tau_lev (double):** Optical depth position for tau slices

**tau_var (int[14]):** Variables to be written (up to 14): [$\varrho$, $v_x$, $v_y$, $v_z$, $\varepsilon$, $B_x$, $B_y$, $B_z$, $|\mathbf{v}|$, $|\mathbf{B}|$, $T$, $P$, $I$, tau surface]

**sl_xy (int) [0]:** Number of xy slices

**xy_lev (double):** Grid positions position for xy slices

**xy_var (int[12]):** Variables to be written (up to 12): [$\varrho$, $v_x$, $v_y$, $v_z$, $\varepsilon$, $B_x$, $B_y$, $B_z$, $|\mathbf{v}|$, $|\mathbf{B}|$, $T$, $P$]

**sl_xz (int) [0]:** Number of xz slices

**xz_lev (double):** Grid positions position for xz slices

**xz_var (int[12]):** Variables to be written (up to 12): [$\varrho$, $v_x$, $v_y$, $v_z$, $\varepsilon$, $B_x$, $B_y$, $B_z$, $|\mathbf{v}|$, $|\mathbf{B}|$, $T$, $P$]

**sl_yz (int) [0]:** Number of yz slices

**yz_lev (double):** Grid positions position for yz slices

**yz_var (int[13]):** Variables to be written (up to 13): [$\varrho$, $v_x$, $v_y$, $v_z$, $\varepsilon$, $B_x$, $B_y$, $B_z$, $|\mathbf{v}|$, $|\mathbf{B}|$, $T$, $P$, $I$]

# 5 Parameters currently hardcoded

## 5.1 IO (io_xysl.C)

**mpi_io_in (int) [1]:** Use MPIIO for reading 3D cubes (data is gathered in z and written out in parallel within xy processor slice, different xy processors slices write different variables in parallel)

**mpi_io_out (int) [1]:** Use MPIIO for writing 3D cubes

**blocksize (int) [8]:** If not using parallel MPIIO within xy processor slice, serial write xy slice with blocksize in the z direction (prevent out of memory for root process performing serial IO)

## 5.2 EOS (cons_to_prim.C)

For out of table bound values the equation of state transitions to an ideal gas. For this purpose $\gamma = 1.65$ and the mean molecular weight $\mu = 0.62$ are defined in the routine cons_to_prim.C. For maximum consistency these values should be updated based of the chosen equation of state.

## 5.3 RT (rt.cc, rt.h)

The radiation transport scheme uses an iterative solver. The intensity in each direction is iterated until the relative error across processor boundaries is below a set threshold. This threshold is defined in rt.h (#define threshold 1.0E-4). Another parameter is the number of rays per octant (#define NMU 3). The code uses Carlson quadratures and this can be increased to 6 or 10 if necessary, see also Section **??**.

# 6 Files needed to run the code

## 6.1 Parameters file

The parameters.dat file contains the parameters described in sections (2-4). Most of the commonly used parameters are mandatory (i.e. the code will terminate if they are not set), a few loss commonly used parameters are optional. [We should go through this and decide if the choices for what is mandatory and what is optional are sensible. Maybe it is safest to have everything mandatory?]

## 6.2 Equation of state tables

The EOS tables are set by the parameter "eos_name" in parameters.dat. They include tabulated functions $T = T(\varrho, \varepsilon)$, $P = P(\varrho, \varepsilon)$, $s = s(\varrho, \varepsilon)$ as well as inverse tables $\varrho = \varrho(s, P)$ and $\varepsilon = \varepsilon(s, P)$ that are needed for the bottom boundary condition. In addition the tables also provide electron density, ion mass density and the ion neutral collisions for ambipolar diffusion as function of $\varrho$ and $\varepsilon$. [The ambipolar quantities are missing from the latest EOS tables!]

## 6.3 Opacity tables

Tge opacity tables are set by the parameter "kap_name" in parameters.dat. The opacity tables contain the number of opacity bins and for each bin opacity and source function as function of $T$ and $P$. The tables may also include opacity and source function for 500nm continuum, which are used for diagnostic intensity output (see also parameter rt_iout) and optical depth scale.

## 6.4 Optically thin radiative loss tables

Optically thin radiative losses are given by the file "Radloss_Chianti.dat". This file provides losses tabulated as function of temperature, the rather weak density dependence is not considered. This is only used when the setting "ext_cor = 1" is chosen. [Should also add name for this file in parameters.dat so that loss functions for different coronal abundances could be selected without renaming the file]

## 6.5 Quadrature for radiation transport

The file "carlson3.dat" contains the weights and angles for the Carlson quadrature using 3 rays per octant. Different Carlson quadratures can be choses by changing "NMU" to 6 or 10 in "rt.h" and providing the appropriate "carlson6.dat" or "carlson10.dat" files.

### 6.6 Backup file

The "backup.dat" file keeps track of the last written restart snapshot. The file contains iteration, time, $P_{\rm bot}$, $s_{\rm bot}$. When setting $P_{\rm bot}$ or $s_{\rm bot}$ to a negative value that code will compute these values from the snapshot. $P_{\rm bot}$ will be set to the mean pressure in the lowest mist domain cell extrapolated to the boundary (half a grid spacing down), $s_{\rm bot}$ will be set to the average entropy in inflow regions if there are inflows, or simply to the mean entropy otherwise.

### 6.7 Potential kernels

Some older formulations of the potential field top boundary require "PSF-kernel-*-*.dat" files that contain pre-computed potential field kernels (a python code for that is provided). When using the HeFFTe FFT library and the routine "potential_sd_heffte_kernels.C" this is no longer required. The code will compute these kernels at the beginning and store them for the remainder of the computation.