# Chapter 1

# How to use

## 1.1 Install/Compile

The MUSIC code can be compiled using standard `cmake`. If you choose this option, please make sure that `cmake` is installed in your system. After downloading the `MUSIC` package, you can create a `build/` folder under the root directory. Go into the build folder, `cd build/`, and then type `cmake ..` to generate configuration and make files. `cmake` will find the compiler and dependent library information in your local systems. After it is done, type `make` under the `build/` folder to compile the code. Once `MUSIC` is complied, use `make install` to copy the executable program to the root directory. At this stage, `MUSIC` is all set and ready to run.

Alternatively, one can use the `GNUmakefile` in the root directory to compile the source code. The `MUSIC` requires using a MPI C++ compiler and the GSL library. In this case, one needs to specify the name of the complier in the `GNUmakefile` under the `src` folder. To compile the source code, one can simply type `make clean` and then `make` under the root directory of the code package. If the code is compiled successfully, an executable file `mpihydro` will be generated in the root directory.

Hydrodynamic simulations are usually performed under the root directory. To start a simulation, one can type the command `mpirun -np num_of_cores ./mpihydro inputfile`. All the parameters are specified in the `inputfile`.

To perform one entire simulation, from initial condition to final particle spectra, one needs to run `MUSIC` sequentially with mode 2, mode 3, mode 4, and mode 14. Mode 13 is optional if one wants to get particle spectra and flow coefficients of thermal emitted hadrons.

## 1.2 Parameters

Here are the parameters as they are called in the `inputfile`. Every parameter has its default value in the code. All the default values are set in the class `ReadInParameters` in the source file `read_in_parameters.cpp`. If one does not want to change the default value of some parameters, these parameters do no need to be appeared in the `inputfile`. Note that the file has to end with the line `EndOfData`.

One can find some example input files in the `/examples_inputs` folder. Alternatively, one can use the `python` script `utilities/generate_music_inputfile.py` to generate input files for `MUSIC`.

**Control parameters:**

`echo_level`
control the mount of messages output to screen

- scale from `1 - 9`

`mode`
This sets the mode the program is supposed to run in.

- `1`: Does everything. Evolution. Computation of thermal spectra. Resonance decays.
- `2`: Hydrodynamic evolution
- `3`: Compute thermal spectra (please make sure surface.dat has been already generated from mode 2)
- `4`: Resonance decays (please make sure mode 3 has been properly done)
- `13`: Compute observables from previously-computed thermal spectra (please make sure mode 3 has been properly done)
- `14`: Compute observables from post-decay spectra (please make sure mode 4 has been properly done)

**Parameters for Initial Conditions:**

`Initial_profile`
set type of initial condition

- `0`: for Gubser flow tests
- `1`: Optical Glauber model
- `3`: Monte-Carlo Glauber model
- `8`: Read in initial profile from a file specified by parameter `Initial_Distribution_Filename` (e.g. IP-Glasma)

`initialize_with_entropy`
decides on whether to scale the entropy density or the enrgy density with the number of wounded nucleons/binary collisions

- `0`: scale energy density
- `1`: scale entropy density

`Initial_Distribution_Filename`
the filename of initial profile (for `Initial_profile == 8`)

- relative directory path for the initial condition

`s_factor`
normalization factor for the energy density

- `any number` $>= 0$.

`Eta_plateau_size`
determines the width of the flat region symmetrical around $\eta = 0$ ($\eta_0$)

- `any number` $>= 0.$

`Eta_fall_off`
determines the width of the half-Gaussians on each side of the plateau ($\sigma_\eta$)

- `any number` $> 0.$

`binary_collision_scaling_fraction`
determines the fraction of which part of the entropy/energy density to scale with the number of binary collisions ($\alpha$).

- `any number`: between 0. and 1.

`Maximum_energy_density`
determines the maximum energy density at zero impact parameter given in [GeV/fm$^3$]
(for `initialize_with_entropy = 0`)
or the maximum entropy density at zero impact parameter given in [1/fm$^3$]
(for `initialize_with_entropy = 1`)

- `any number` $> 0.$

`rho_b_max`
is the maximum baryon density for zero impact parameter. The shape of the $\rho_B$ distribution is the same as that for the energy/entropy density distribution

- `any number` $> 0.$ (don't use 0, however if the EOS does not support finite baryon chemical potential, it does not matter what you put)

`Impact_parameter`
is the impact parameter $b$ in [fm]

- `any number` $>= 0.$

`SigmaNN`
is the nucleon-nucleon cross section in [mb]

- `any number` $> 0.$

`Target`
is the target nucleus' name

- `any name` of nucleus as defined in `known_nuclei.dat`

`Projectile`
is the projectile nucleus' name

- `any name` of nucleus as defined in `known_nuclei.dat`

## Parameters for Hydrodynamic Evolution:

`boost_invariant`
whether the simulation is boost-invariant

- `1` Yes
- `0` No

If `boost_invariant==1`, `Grid_size_in_eta` will be set to 1. Please assign only 1 cpu core to run the code.

`EOS_to_use`
determines the equation of state to use in the evolution

- `0` ideal gas
- `1` EOS-Q from `azhydro`
- `2` lattice EOS `s95p-v1` from Huovinen and Petreczky
- `3` lattice EOS `s95p-v1-PCE150` from Huovinen and Petreczky with partial chemical equilibrium (PCE) at $T_{\rm chem} = 150$ MeV.
  (see `https://wiki.bnl.gov/TECHQM/index.php/QCD_Equation_of_State`)
- `4` lattice EOS `s95p-v1-PCE155` from Huovinen and Petreczky with partial chemical equilibrium (PCE) at $T_{\rm chem} = 155$ MeV.
- `5` lattice EOS `s95p-v1-PCE160` from Huovinen and Petreczky with partial chemical equilibrium (PCE) at $T_{\rm chem} = 160$ MeV.
- `6` lattice EOS `s95p-v0-PCE165` from Huovinen and Petreczky with partial chemical equilibrium (PCE) at $T_{\rm chem} = 165$ MeV.
- `7` lattice EOS `s95p-v1.2` from Huovinen and Petreczky where the hadronic phase contains the same species of resonances as in the UrQMD model

`Initial_time_tau_0`
is the initial time in [fm]

- `any number` $> 0$.

`Delta_tau`
is the time step to use in [fm].

- `any number` $> 0$.

`Total_evolution_time_tau`
is the total evolution time in [fm]. In case of `Do_FreezeOut_Yes_1_No_0 == 1`, evolution will halt erlier if all cells are frozen out.

- `any number` $> 0$.

`Viscosity_Flag_Yes_1_No_0`
flag to turn on and off viscosities in the evolution

- `1` Yes
- `0` No

`Include_Shear_Visc_Yes_1_No_0`
flag to turn on and off shear viscous effect

- `1` Yes
- `0` No

`Shear_to_S_ratio`
The value of constant $\eta/s$

- `any number` $\geq 0$

`T_dependent_Shear_to_S_ratio`
flag to use the hard-coded temperature dependent $\eta/s(T)$. The temperature dependence of $\eta/s(T)$ is specified in the `dissipative.cpp`

- `1` Yes
- `0` No

`Include_Bulk_Visc_Yes_1_No_0`
include bulk viscous effect using a hard-coded temperature dependent $\zeta/s(T)$

- `1` Yes
- `0` No

`Include_second_order_terms`
include second order non-linear coupling terms

- `1` Yes
- `0` No

`Minmod_Theta`
is the $\theta$ parameter in the min-mod like limiter

- `any number` $1. \leq \theta \leq 2.$

`Runge_Kutta_order`
is the order of the Runge-Kutta solver

- `any integer` in $\{1, 2\}$

`Grid_size_in_eta`
is the number of cells in the $\eta$ direction

- `any integer` $>= 1$

`Grid_size_in_x`
is the number of cells in the $x$ direction

- `any integer` $>= 1$

`Grid_size_in_y`
is the number of cells in the $y$ direction

- `any integer` $>= 1$

`Eta_grid_size`
is the extend of the lattice in $\eta$-units

- `any number` $> 0$.

`X_grid_size_in_fm`
is the extend of the lattice in $x$ in [fm]

- `any number` $> 0$.

`Y_grid_size_in_fm`
is the extend of the lattice in $y$ in [fm]

- `any number` $> 0$.

`output_hydro_debug_info`
flag to output evolution info

- `1` Yes
- `0` No

`output_evolution_data`
flag to output evolution history to file (e.g. temperature, flow velocities of most space-time points)

- `1` Yes
- `0` No

`outputBinaryEvolution`
output evolution file in binary format

- `1` Output evolution file in binary format
- `0` Output evolution file in plain text format

`output_evolution_every_N_eta`
output evolution file every Neta steps

- `any integer` $> 0$

`output_evolution_every_N_y`
output evolution file every Ny steps

- `any integer` $> 0$

`output_evolution_every_N_x`
output evolution file every Nx steps

- `any integer` $> 0$

`output_evolution_every_N_timesteps`
output evolution every Ntime steps

- `any integer` $> 0$

## Parameters for Freeze-out and Cooper-Frye:

`Do_FreezeOut_Yes_1_No_0`
flag to find freeze-out surface

- `1` Yes
- `0` No

`use_eps_for_freeze_out`
flag to use energy density as criteria to find freeze-out surface

- `0` Use temperature
- `1` Use energy density

`epsilon_freeze`
is the freeze-out energy density, at which the system is frozen out, given in [GeV/fm$^3$]

- `any number` $> 0$.

`T_freeze`
is the freeze-out energy density, at which the system is frozen out, given in [GeV/fm$^3$]

- `any number` $> 0$.

`Include_deltaf`
flag to include shear delta f correction in the Cooper-Frye formula

- `1` Yes
- `0` No

`Inlucde_deltaf_bulk`
flag to include bulk delta f correction in the Cooper-Frye formula

- `1` Yes
- `0` No

`freeze_out_method`
determines which method to use for freeze-out.

- `2`: Schenke's more complex method

`average_surface_over_this_many_time_steps`
the corse-graining step in the tau for hyper-surface finding algorithm

- `any integer` $> 0$

`particle_spectrum_to_compute`
This decides on which spectrum to compute.

- 0: Do all up to `number_of_particles_to_include`
- `any number`: Do the particle with this (internal) id.

`number_of_particles_to_include`
This determines up to which particle in the list spectra should be computed (`mode==3`) or resonances should be included (`mode==4`)

- `any number`: current maximum = 320.

`pseudo_steps`
number of rapidity slices when computing thermal particle spectra

- `any integer` $> 0$

`pseudofreeze`
flag to compute particle spectra in equally-spaced rapidity or pseudorapidity

- `0` calculate particle spectra in equally-spaced rapidity
- `1` calculate particle spectra in equally-spaced pseudorapidity

`max_pseudorapidity`
the range of pseudo-rapidity (`-maximal_rapidity`, `maximal_rapidity`) in which particle spectra are computed

- `any number` $\geq 0$

`pt_steps`
number of points in $p_T$ at which thermal particle spectra are computed

- `any integer` $> 0$

`min_pt`
the minimum $p_T$ value for particle spectra

- `any number` $> 0$

`max_pt`
the maximum $p_T$ value for particle spectra

- `any number` $> 0$

`phi_steps`
number of points in $\phi_p$ in $(0, 2\pi)$ at which thermal particle spectra are computed

- `any integer` $> 0$

## Parameters for Post Freeze-out Analysis

`dNdy_nrap`
number of slices in (pseudo-)rapidity to compute particle's $dN/dy$ or $dN/d\eta$ distribution

- `any integer` $> 0$

`dNdy_eta_min`
The minimum value of $dN/d\eta$ distribution

- `any number`

`dNdy_eta_max`
The maximum value of $dN/d\eta$ distribution

- `any number`

`dNdy_y_min`
The minimum value of $dN/dy$ distribution

- `any number`

`dNdy_y_max`
The maximum value of $dN/dy$ distribution

- `any number`

`dNdyptdpt_eta_min`
The minimum value of the pseudo-rapidity integration range for particle $p_T$-spectra

- `any number`

`dNdyptdpt_eta_max`
The maximum value of the pseudo-rapidity integration range for particle $p_T$-spectra

- `any number`

`dNdyptdpt_y_min`
The minimum value of the rapidity integration range for particle $p_T$-spectra

- `any number`

`dNdyptdpt_y_max`
The maximum value of the rapidity integration range for particle $p_T$-spectra

- `any number`

## 1.3 Output

**Mode 2:**

After `MUSIC` finishes mode 2, hyper-surface files (if `Do_FreezeOut_Yes_1_No_0` is set to 1) will be generated. The format of the hyper-surface file `surface.dat` is as follows,

$\tau_f$, $x_f$, $y_f$, $\eta_f$, $d^3\sigma_\tau$, $d^3\sigma_x$, $d^3\sigma_y$, $d^3\sigma_\eta$, $u^\tau$, $u^x$, $u^y$, $\tilde{u}^\eta$, $e_f$, $T_f$, $\mu_B$, $\frac{e_f+P_f}{T_f}$, $\pi^{\tau\tau}$, $\pi^{\tau x}$, $\pi^{\tau y}$, $\tilde{\pi}^{\tau\eta}$, $\pi^{xx}$, $\pi^{xy}$, $\tilde{\pi}^{x\eta}$, $\pi^{yy}$, $\tilde{\pi}^{y\eta}$, $\tilde{\pi}^{\eta\eta}$, $\Pi$ (if `Include_Bulk_Visc_Yes_1_No_0` is set to 1)

Here the spatial position of the freeze-out surface ($\tau_f$, $x_f$, $y_f$, $\eta_f$) is in the unit fm. The surface normal vector $d^3\sigma_\mu$ is in the unit of fm$^3$. For the third component of the flow velocity, $\tilde{u}^\eta = \tau u^\eta$. The freeze-out energy density $e_f$ is in the unit 1/fm$^4$. The units of the freeze-out temperature $T_f$ and baryon chemical potential $\mu_B$ are 1/fm. For the shear stress tensor, all the components carry unit 1/fm$^4$, in where $\tilde{\pi}^{(\tau,x,y)\eta} = \tau\pi^{(\tau,x,y)\eta}$ and $\tilde{\pi}^{\eta\eta} = \tau^2\pi^{\eta\eta}$. The bulk pressure $\Pi$ (1/fm$^4$) is only output when `Include_Bulk_Visc_Yes_1_No_0` is set to 1.

**Mode 3:**

After `MUSIC` finishes mode 3, thermal particle spectra $dN/(dyp_Tdp_Td\phi_p)$ will be output to a file `yptphiSpectra.dat`. The information about how the data is stored in this file is printed in a file `particleInformation.dat`. In this file, the first column is particle's monte-carlo id, the second column is the computed maximum rapidity for the thermal particle spectra $dN/(dyp_Tdp_Td\phi_p)$, the third column is the number of points in rapidity or pseudo-rapidity direction in $dN/(dyp_Tdp_Td\phi_p)$. The lattice points along this direction range from [-maximum rapidity, +maximum rapidity]. The fourth and fifth columns are the minimum and maximum values of the transverse momentum $p_T$ evaluated in $dN/(dyp_Tdp_Td\phi_p)$, respectively. The sixth column is the number of points taken in $p_T$. The last column records the number of points evaluated in $\phi_p$. In the file `yptphiSpectra.dat`, every particle's $dN/(dyp_T dp_T d\phi_p)$ is stored as a $(N_y N_{p_T} \times N_{\phi_p}$ matrix.

**Mode 4:**

After `MUSIC` finishes mode 4, the feed-down contributions were computed for every species of particles. Two new files `FyptphiSpectra.dat` and `FparticleInformation.dat` are generated. The formats of these two files are exactly the same as the ones generated in Mode 3.

**Mode 13:**

After `MUSIC` finishes mode 13, thermal particle spectra and $v_n$ results are generated in a folder `outputs/`. If the folder does not exist, `MUSIC` will create the folder. In default, the particle spectra and vn of $\pi^+$, $\pi^-$, $K^+$, $K^-$, $p$, $\bar{p}$ are analyzed. The header of each file explains the format of the file.

**Mode 14:**

After `MUSIC` finishes mode 14, final particle spectra and $v_n$ results are generated in a folder `outputs/`. If the folder does not exist, `MUSIC` will create the folder. The formats of the files are exactly the same as the ones output in Mode 13. In order to distinguish the outputs from the two modes, an extra letter "F" is added in the front of the filenames for the results analyzed in Mode 14.

## 1.4 Code testing

`MUSIC` provides some testing cases to validate the code in the package. The testing files are stored in the `/tests/` folder.

### 1.4.1 Gubser flow test

One can perform the Gubser flow test for the `MUSIC` code. This is very handy if one wants to implement some customized modifications of the source codes. To perform the test, one can type the following commands. For ideal hydro test, `mpirun -np 1 ./mpihydro tests/Gubser_flow/music_input_Gubser_ideal` and for viscous hydro test, `mpirun -np 1 ./mpihydro tests/Gubser_flow/music_input_Gubser`. After the hydrodynamic simulation is done, one can use the `python` script `make_plots_ideal.py` and `make_plots.py` in the `/test/Gubser_flow` folder to generate comparison plots. One needs to have the `matplotlib` library installed to generate the comparison plots.