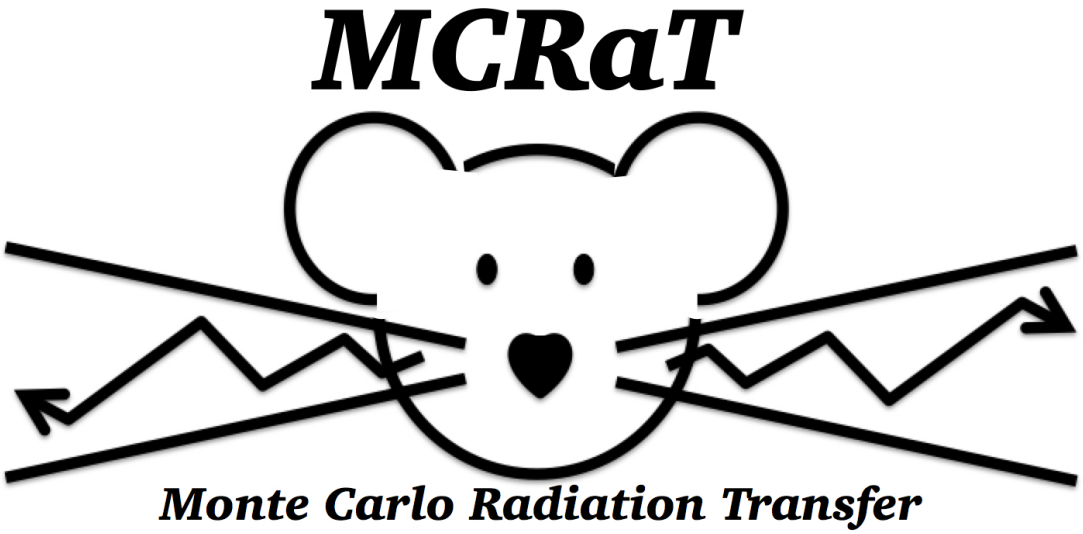


Monte Carlo Radiation Transfer (MCRaT) User's Guide

Tyler Parsotan

Contents

1 Introduction

The Monte Carlo Radiation Transfer (MCRaT; pronounced *Em-Cee-Rat*) code is a next generation radiation transfer code that can be used to analyze the radiation signature expected from astrophysical outflows. The code is written in C and uses the Message Passing Interface (MPI) library for inter-process communication, the Open Multi-Processor (OpenMP) library for intra-node communication, the GNU Scientific Library, and the HDF5 library for parallel I/O.

MCRaT injects photons into a hydro simulation, using the FLASH or the PLUTO codes, and individually propagates and compton scatters the photons through the fluid until the end of the simulation. This process of injection and propagating occurs for a user specified number of times until there are no more photons to be injected. Users can then construct light curves and spectra with the MCRaT calculated results. The hydro simulation used in this version of MCRaT must be in 2D, however, the photon propagation and scattering is done in 3D by assuming cylindrical symmetry.

The code was initially written in python by Dr. Davide Lazzati as a proof of concept. The code was then translated into C by Tyler Parsotan and made to use the OpenMP, MPI, HDF5, and GNU Scientific libraries. MCRaT is highly parallelized and is easy to set up and use.

There are also python files with documented functions to process the MCRaT output located at: `https://github.com/parsotat/ProcessMCRaT`.

In using MCRaT and the ProcessMCRaT codes, we ask that you cite the following papers: Lazzati (2016); Parsotan & Lazzati (2018); Parsotan et al. (2018, 2020).

2 Getting Started

This section describes how to get the MCRaT code as well as the required libraries necessary to compile MCRaT. This section will also cover the necessary steps to run the compiled MCRaT code.

2.1 Required Libraries

MCRaT requires a number of libraries in order to compile properly. These are:

- MPI
- OpenMP
- GNU Scientific Library (GSL)
- HDF5

The MPI library can be an any implementation such as OpenMPI (open-mpi.org) or one that is specific to a given supercomputer. Following the steps to install the library should allow it to work properly for compiling MCRaT. The OpenMP library (openmp.org) is automatically included with Intel and GNU compilers. The GNU Scientific Library (GSL) can be downloaded from gnu.org/software/gsl/. Following the included instructions should install GSL in the default directory, allowing it to be found by the compiler. The HDF5 library can be found at support.hdfgroup.org/HDF5/ where version 1.10 or greater should be downloaded and installed. The parallel portion of HDF5 needs to also be installed which is included in the instructions for installing HDF5.

2.2 Downloading and Compiling MCRaT

Once the required libraries have been installed, the current MCRaT code can be downloaded from github.com/lazzati-astro/MCRaT/. There is a Makefile which makes compiling the code significantly easier. In the directory that all the MCRaT files were downloaded into, simply typing `make` will compile the MCRaT code into a binary file called `MCRAT`. Typing `make MERGE` will compile the included MCRaT post-processing code, which will be covered in later in this section, into a binary file called `MERGE`.

The Makefile has to be configured for a variety of specifics related to the user's system. The `CC` option in the makefile can be changed to the user's compiler of choice such as gcc or icc. The `HDF_INSTALL` line is set to the directory that the HDF5 library is installed in. The `EXTLIB` line is set to the directories of any external libraries that the user may have. For example, if the GSL library cannot be found by the compiler, the user can put the directory of the library in this line of the Makefile in order to properly compile MCRaT. The `CFLAGS` feeds any compiler options to the compiler such as optimizations, and the switch to compile MCRaT with OpenMP capabilities, which is necessary to exploiting the hybrid parallelization available in the code. If the user is using gcc, the `-fopenmp` does not need to be changed, however, if the user is using icc, `-fopenmp` needs to be changed to `-qopenmp`. The `INCLUDE` line can be changed to include any `include/` directories to help the compiler, and the same for the `LIBSHDF` in relation to any shared libraries. The `LIB`, `DEPS`, and `OBJ` lines should not need to be changed. Once the lines in the Makefile appropriately reflect the users configurations, typing `make` and `make MERGE` should properly compile the MCRaT code as well as its post-processing code, `MERGE`.

2.3 The mcrat_input.h file

Once compilation is successful, the user will need to decide where MCRaT's output will go. The user needs to make a directory within the directory that contains the hydro simulation frames. This subdirectory is where the parameter file for MCRaT must be placed. This is the same directory in which MCRaT will do all of its work, including making its own directories and output files within the aforementioned directories. Once this directory is created, the user must modify the `mcrat_input.h` file in order for the code to know where to find all the hydro files and the MCRaT parameter file. The constant `FILEPATH` should be modified with the directory of the flash simulation files while the constant `FILEROOT` should contain the name of the hydro simulation files without any frame numbers. Since the FLASH and PLUTO-CHOMBO simulation files are typically hdf5 files, MCRaT not need the file extension explicitly. The constant `MC_PATH` should be the name of the directory, located within the `FILEPATH` directory, that holds the MCRaT parameter file named `mc.par`. An example file structure, with the values for each variable in MCRaT, is shown below:

```
/dir/to/hydro/simulation/ (FILEPATH)
├── dir_with_MCRaT_parameter_file/ (MC_PATH)
│   └── mc.par
└── example_hydro_file_name_0001 (FILEROOT=example_hydro_file_name_)
```

The other variable that can be changed in MCRaT is `SIMULATION_TYPE` which should be set as `SCIENCE`, if using MCRaT for a production run, or `SPHERICAL_OUTFLOW/CYLINDRICAL_OUTFLOW/`

which is for testing MCRaT with these types of outflows (which will be covered in a later section). Once all the variables have been appropriately changed, by replacing the strings appropriately in the `mcrat.c` file, the code needs to be recompiled.

Additionally, there are a number of switches that the user can specify to the program to tell it whether they want to consider the effects of polarization, with the `STOKES_SWITCH`, and one that specifies whether the user wants MCRaT to save the comoving 4 momenta of each photon, with the `COMV_SWITCH`. These switches can be set to `OFF`, which means do not save the comoving 4 momenta and/or save and consider the effects of polarization, or they can be set to `ON`, which tells the program to consider polarization and/or save the stokes parameter associated with each photon or save each photon's 4 momentum.

The `SIM_SWITCH` tells MCRaT what hydro simulation is being used. The options are `FLASH` and `PLUTO_CHOMBO`, for the FLASH hydrodynamic code and the PLUTO hydro code with CHOMBO AMR respectively. The switch `DIM_SWITCH` tells MCRaT the dimensionality of the hydro simulation that is used. This switch can be set to `TWO`, for 2D, `TWO_POINT_FIVE`, for 2.5D, or `THREE`, for 3D. The geometry can also be specified through the `GEOMETRY` define statement. The options are `SPHERICAL`, `CARTESIAN`, `CYLINDRICAL`, and `POLAR`. The length, pressure, and density scales of the hydro simulation also has to be specified by setting the values of `HYDRO_L_SCALE`, `HYDRO_P_SCALE`, and `HYDRO_D_SCALE`, respectively, to the normalization that is used in the hydro simulation.

In order to enable the consideration of thermal synchrotron photons in the MCRaT simulation the `CYCLOSYNCHROTRON_SWITCH` needs to be set to `ON`, otherwise this switch should be set to `OFF`. If it is set to be `ON`, the user needs to define other variables. One variable that needs to be defined is `CYCLOSYNCHROTRON_REBIN_E_PERC` which provides MCRaT with an upper limit for the number of synchrotron photons to emit. This quantity is calculated as the percentage of the maximum number of photons the user wants injected (see the `[MCRaT Photon Block]` portion of the example mc.par file in section 2.4). If `CYCLOSYNCHROTRON_REBIN_E_PERC` is set to 0.5 and the value of the 17^{th} line is 3000, the upper limit for the number of synchrotron emitted photons is 1500. This variable also defines the number of energy bins that MCRaT will create when it rebins the synchrotron photons, thus in the aforementioned example, the number of energy bins that the photons will be rebinned into is 1500 as well. MCRaT will also rebin the photons into spatial bins and the size of these bins are specified using the `CYCLOSYNCHROTRON_REBIN_ANG` variable which has units of degrees. In considering synchrotron emission, the user needs to specify how the magnetic field is calculated. This is done through the `B_FIELD_CALC` switch which can be set to: `INTERNAL_E`, where the magnetic energy density is set equal to the internal energy of the outflow, `TOTAL_E`, where the magnetic field is calculated from the total energy of the outflow, or `SIMULATION` where the code uses the magnitude of the magnetic field that is acquired from a simulation. If `B_FIELD_CALC` is set to `TOTAL_E` or `INTERNAL_E`, the user needs to set the `EPSILON_B` switch, which represents the fraction of the total outflow energy that is in the magnetic field. More information about the implementation of thermal synchrotron emission in the algorithm of the code can be found in section 3.2. More information regarding the physics of the implementation of thermal synchrotron emission and absorption can be found in section 4.4.

Another helpful switch is the `SAVE_TYPE` switch, with tells MCRaT to include the photon type in the saved datasets.

The parameters that can be set and the appropriate values that each parameter can be set as are as follows:

Parameter	Acceptable Values
SIMULATION_TYPE	SCIENCE, CYLINDRICAL_OUTFLOW, SPHERICAL_OUTFLOW, STRUCTURED_SPHERICAL_OUTFLOW
FILEPATH	"/dir/to/hydro/simulation/ "
FILEROOT	"example_hydro_file_name "
MC_PATH	"dir_with_MCRaT_parameter_file/"
SIM_SWITCH	FLASH, PLUTO_CHOMBO
STOKES_SWITCH	ON, OFF (Defaults to OFF)
COMV_SWITCH	ON, OFF (Defaults to OFF)
SAVE_TYPE	ON, OFF (Defaults to OFF)
DIMENSIONS	TWO, TWO_POINT_FIVE, or THREE
GEOMETRY	CARTESIAN, SPHERICAL, CYLINDRICAL (only in 2D), POLAR (only in 3D)
HYDRO_L_SCALE	$(-\infty, +\infty)$
HYDRO_D_SCALE	$(-\infty, +\infty)$
MCPAR	"mcpar_filename.par"
CYCLOSYNCHROTRON_SWITCH	ON, OFF (Defaults to OFF)
If CYCLOSYNCHROTRON_SWITCH is ON:	
CYCLOSYNCHROTRON_REBIN_E_PERC	(0.0, 1.0] (Default is 0.1)
CYCLOSYNCHROTRON_REBIN_ANG	$(-\infty, +\infty)$ (Default is 0.5 degrees)
B_FIELD_CALC	INTERNAL_E TOTAL_E (Default) SIMULATION
If B_FIELD_CALC is TOTAL_E or INTERNAL_E:	
EPSILON_B	(0.0, 1.0] (Default is 0.5)

2.4 The MCRaT Parameter File

MCRaT requires the user to provide information in the form of a parameter file named `mc.par` which should be placed in the appropriate directory, as is described in the previous section. An example `mc.par` file, named `sample_mc.par` is included in the MCRaT github repository. An example with a description of each line, is as follows:

```
[Hydro/MHD Simulation Block]

5.        # Number of frames per second of hydro simulation (likely always the same)
3000   # Last available hydro simulation frame  (get from the last file in the data folder)
0 5e12   # Max r0 coordinate limits of hydro simulation
0 2.5e13  # Max r1 coordinate limit of hydro simulation
0 2e13   # Max r2 coordinate limit of hydro simulation (if simulation is 3D)

[MCRaT Injection Angles Block]

0.                # The minimum off-axis angle to inject photons (in degrees)
6.                # The maximum off-axis angle to inject photons (in degrees)
3.       # Number of angle bins to consider
200 200 200       # Frame at which photon injection starts for each angle bin
199 199 299       # Number of frames for which photons are injected for each angle bin
1e11 1.5e12 2e12  # The radius at which the photons are injected for each angle bin

[MCRaT Photon Block]

b # Type of spectrum we inject with, w=wien b=blackbody
1000 # Min number of photons
5000 # Max number of photons

[Initialization/Continuation Block]

i # Initialize or continue simulation (i=initialize (delete all files) c=continue)
```

The format of the `mc.par` file must follow what is shown above with each block in its respective order and each block title separated by a blank line from the contents of the block. Furthermore, the coordinate limits for r2 must always be there, even if the hydro simulation that is being analyzed with MCRaT is in 2D. The entries for each line that takes multiple values, such as the coordinate limits, must have each value separated by a space.

The first block provides information on the hydro/MHD simulation that MCRaT is analyzing. The first line denotes the number of hydro frames per second, or fps. The second line provides the file number of the last hydro simulation frame available in the analysis. The next three lines provide the lower and upper limits of the hydro coordinate. Even if the hydro simulation that is being analyzed is in 2D, the last line must still be there. In the example above, the r0 coordinate extends from 0 to 5e12 in the units provided by `HYDRO_L_SCALE` or, in the case of angles, the units used by the hydro simulation. The table below denotes what coordinates r0, r1, and r2 correspond to for each coordinate system and number of dimensions as well as their corresponding units.

	2D	3D
Coordinate System	(r0, r1, r2)	(r0, r1, r2)
Cartesian	(x,y,-)	(x,y,z)
Spherical	(r,θ, -)	(r,θ, ϕ)
Polar	-	(r, ϕ, z)
Cylindrical	(r,z)	-

The second block, called the `[MCRaT Injection Angles Block]`, tells MCRaT what angle ranges (in degrees) it should consider when injecting photons, as provided by the first two lines of the block, and what radii to inject photons for each angle range. The third line of this block denotes the number of angle ranges. In this example, we want MCRaT to inject photons from 0∘-6∘ and we want there to be 3 angle bins. For each angle bin, we provide the frame from which the photon injections will start, $\mathrm{f_{start}}$, the number of frames that we will inject photons into, df, and the radius where the photons will be injected in the frame, R_{inj}. Thus, in the example provided, there have to be 3 numbers provided in each of the last 3 lines of the block. If we wanted N angle bins, then the next 3 lines should contain N numbers for each angle bin. Each number needs to be separated by a space.

These last 3 lines dictate what time in the simulation you begin injecting photons. This is calculated as

$$t_{\mathrm{start}} = \frac{\mathrm{f_{start}}}{\mathrm{fps}} - \frac{R_{\mathrm{inj}}}{c}$$

Building on the above formula, the time in which photons stop being injected are calculated as

$$t_{\mathrm{end}} = \frac{\mathrm{f_{start}} + \mathrm{df}}{\mathrm{fps}} - \frac{R_{\mathrm{inj}}}{c}$$

These calculations are the same for photons injected at all angles. The number of angle bins and the number of frames that photons will be injected into (for each angle bin) are important for MCRaT's parallelism. This will be covered in the next section.

In the next block, labeled `[MCRaT Photon Block]`, we specify the initial condition of the MCRaT simulation and the number of photons that we want MCRaT to consider. The first line denotes whether we want the injected photons to be drawn from a blackbody or Wien spectrum, based on the optical depth being either $\gtrsim 10^3$ or ~ 100, respectively. The next two lines specify the minimum and maximum number of photons that MCRaT injects into a given frame, which helps to balance Poisson error from the Monte Carlo method with the run time of the MCRaT code.

The final block allows the user to continue the MCRaT code from a previous point, if the value is set to "c", or it allows the user to initalize (by erasing the full working directory of MCRaT) or start a simulation by setting it to "i". **This is an important parameter to check to prevent accidentally erasing work that has already been done.**

2.5 Understanding MCRaT's Parallelism

The first step that MCRaT takes in breaking up the problem is breaking it up into pieces based on the range of angles that photons will be injected into, which the user specifies in the `[MCRaT Injection Angles Block]` in the `mc.par` file. As the example `mc.par` above shows, MCRaT will break up the problem into 3 pieces initially. Then, for each angle range ($0^\circ - 2^\circ$, $2^\circ - 4^\circ$, etc.) the code will then break up the problem based upon how many times photons will be injected, which is 1 plus the values provided in the `mc.par` file; this is because the photons are injected in the frames $[\mathrm{f_{start}}, \mathrm{f_{start}} + 1, ..., \mathrm{f_{start}} + \mathrm{df} - 1, \mathrm{f_{start}} + \mathrm{df}]$, which gives us this n+1 behavior. These pieces of information are important for understanding how the processes are divided with MCRaT. An example will be given in the next section.

2.6 Running MCRaT

In this section we show how to run MCRaT and allocate the appropriate number of cores in order to optimize MCRaT's parallelism. We will use the example `mc.par` file in Section 2.4 for this section.

Running the MCRaT code is as simple as doing `mpiexec -np N ./MCRAT` where N is the total number of MPI processes that will be created. The OpenMP parallelism is controlled by setting the OpenMP environment variable `OMP_NUM_THREADS`. We recommended that `OMP_NUM_THREADS` be set to 1 for a newly started MCRaT simulation, which essentially turns off the OpenMP parallelism.

The total number of MPI processes, N, is initially divided by the set of angles that photons will be injected into. For the example `mc.par`, there are 3 sets of angles that will be considered which leads to the N MPI process initially being divided by 3. The $N_\theta = N/3$ MPI processes will then be used to inject photons in the user specified frames in parallel. The example `mc.par` file shows that for the first angle bin there will be $199 + 1$ photon injections, as was described in the last section. Thus, the number of frames that each MPI process will inject photons into becomes $\mathrm{f_{inj}} = (199 + 1)/N_\theta$. If we set $N = 150$, $N_\theta = 50$ and the number of frames that each MPI process will inject photons into becomes 4. Alternatively, if we want each MPI process to only inject photons into one frame, $N_\theta = 200$ which means that $N = 600$. Calculating N, by following these examples, shows how the problem is broken up between angles and photon injections. If the number of MPI processes available for the user to use does not divide well into the size of the problem, in terms of angle ranges and frames in which photons are injected, MCRaT will prioritize breaking up the problem evenly based on the set of angles. Thus, N should at least be evenly divisible by the number of angle ranges specified by the `mc.par`; Once N_θ is calculated, MCRaT will distribute the number of frames in which photons will be injected accordingly to calculate $\mathrm{f_{inj}}$.

Another option, when dealing with a limited number of available cores, is running MCRaT on a small portion of the problem. For example instead of doing 3 sets of angles at 2° intervals all at once, simply run 1 set of angles from $0^\circ - 2^\circ$, and when that simulation is complete run the second set from $2^\circ - 4^\circ$, etc. The increased parallelism in decreasing $\mathrm{f_{inj}}$ by using this method may speed up the simulation compared to running the whole problem at once. This is also advantageous due to the fact that as the MPI processes begin to complete their portion of the simulation, the user may restart MCRaT using only the appropriate number of MPI processes left that have calculations left to do. Thus, this feature lets a given MCRaT simulation (e.g. $0^\circ - 2^\circ$) use less MPI processes and resources over time, allowing another full simulation (e.g. $2^\circ - 4^\circ$) to start running using the rest of the available resources. We will go over this feature in a later section.

2.7 MCRaT Output Files

Once MCRaT is run, it will create subdirectories dedicated to the calculations and results of each angle range. Using the example `mc.par` file, the directory tree will now look like:

Within each angle subdirectory, each MPI process, labeled process i, creates 3 files. The mc_output_i.log file prints exactly what the MCRaT code is doing for that given MPI process. This allows us to monitor the progress of the MCRaT simulation. This can be easily done by using Linux commands such as `less mc_output_`i`.log`. The mc_proc_i.h5 is a HDF5 file that contains all of the calculation results of the MCRaT MPI process. The mc_chkpt_i.dat file contains the necessary data for each MPI process to continue

```
/dir/to/hydro/simulation/
├── example_hydro_file_name_0001
└── dir_with_MCRaT_parameter_file/
    ├── mc.par
    ├── 0.0-2.0/
    │   ├── mc_proc_i.h5
    │   ├── mc_chkpt_i.dat
    │   └── mc_output_i.log
    ├── 2.0-4.0/
    │   ├── mc_proc_i.h5
    │   ├── mc_chkpt_i.dat
    │   └── mc_output_i.log
    └── 4.0-6.0/
        ├── mc_proc_i.h5
        ├── mc_chkpt_i.dat
        └── mc_output_i.log
```

from the last point if MCRaT is interrupted for any reason. This is covered in the next section.

When all of the MPI processes in a given angle subdirectory have completed their calculations, they will produce HDF5 files with the results of the calculations for each file in the simulation. The results start with the frame specified in the `[MCRaT Injection Angles Block]` of the `mc.par` file and go up to the frame specified in the next line of the `mc.par` file block. These files will be named `mcdata_`f`.h5` where f is the frame number of the hydrodynamic simulation.

2.8 Restarting MCRaT

If the MCRaT code gets interrupted by running out of allocated time on a cluster or any other type of issue, the code is easily continued. It may also use less MPI processes when it is restarted based on the number of MPI processes in the original simulation run that still have calculations to complete.

In order to ensure that the code knows to continue a MCRaT simulation, the last block of the `mc.par` file must be set to "c". **This is important to check since not changing this parameter means that all of the simulation progress will be erased and a completely new simulation will be started, thus nullifying the attempt to save time and resources by continuing a MCRaT simulation.**

Once this parameter has been changed, the user needs to identify how many total MPI processes have completed their calculations. This is done by running"

```
tail -n 1 dir_with_MCRaT_parameter_file/angle_dir/mc_output_* |grep
"completed" |wc -l
```

in each angle range directory created by MCRaT. Add all the printed values and subtract from the original number of MPI processes, N, that the MCRaT simulation was originally started with. This is the number that has to replace N in the new run of MCRaT. From here, MCRaT automatically distributes the given number of processes among the MPI processes in each angle subdirectory that still need to run calculations.

This feature of only continuing the MCRaT simulation with the necessary number of MPI processes means that as the simulation progresses, fewer resources will be used which is invaluable for running jobs in a pipeline type of fashion and for conserving resources such as SBU allocations.

If a certain process is also taking a long time to conduct its simulation it is possible to modify the mc.par file to specify that the simulation should inject fewer photons, to speed up the calculation and force the process' simulation portion to restart. To force the restart, delete that process' mc_chkpt file and it's mc_proc file. This is exactly what the `mcrat_msp.sh` shell script does.

2.8.1 Using The mcrat_msp.sh Shell Script

The mcrat_msp.sh shell script determines which MCRaT processes have a very large number of scatterings and allows the user to delete those process' mc_chkpt and mc_proc files, if the process' photons have undergone a number of scatterings greater than some user specified number of scatterings. This would then force a new restart for those processes with less injected photons. It also allows the user to refine the mc.par file to set a new number of injected photons for these same processes that will be restarted the next time the user starts mcrat with the 'c' flag (to continue a simulation). **This is important to change once MCRaT has been restarted and the photons injected. If it isn't, other processes, which don't have so many scatterings, will inject photons within the new range specified by the mc.par file (if applicable, which it should be on a relatively small cluster where the user can't assign one photon injection frame to each process). To do this, once the simulation restarts and is running smoothly, the user has to end it, modify the minimum and maximum range of injected photons to the original numbers, and restart the simulation once again. (We are currently working on a better way of implementing this feature.) The user must remember to reset the restart flag in the mc.par file to 'c' otherwise all the simulation data files will be erased and all progress lost.** Opening the mcrat_msp.sh file in a text editor will bring up different flags for the shell script and give examples of how to properly use the function.

2.9 Using Hybrid-Parellelization

As was mentioned earlier, it is not advised to enable hybrid parallelization at the start of a new MCRaT simulation, especially if limited resource availability is an issue. This is due to the fact that the OpenMP threads may interfere with one another and neighboring MPI processes, thus slowing the overall code. To prevent this, set `OMP_NUM_THREADS` to 1.

If MCRaT is being used to continue a simulation, there is a good chance that some of the MPI processes have finished and the remaining MPI processes are running slower than the rest due to a variety of factors, which will be covered later. This is where hybrid parallelization helps to ensure that MCRaT completes its calculations is a reasonable amount of time. These MPI processes can be sped up by using the OpenMP threads. The number of threads per MPI process can be set with the `OMP_NUM_THREADS` parameter. Even in this case, it is important to make sure that threads are not interfering with one another and undermining any potential performance gain.

2.10 Post-Processing MCRaT Data

If the user is simply running running a simulation with one angle range under consideration, there is no post-pocessing to be done. The MCRaT produced `mcdata_`f`.h5` files contain all the information for the photons (which will be covered in the next section).

If not we need to be able to merge all the simulation results for photons injected in all angle ranges; following the example `mc.par` file, we need to merge the data in angles $0^\circ - 2^\circ$, $2^\circ - 4^\circ$, etc. This is done is with the MERGE code that is included with MCRaT. To run merge, the user simply types `mpiexec -np N ./MERGE`. In this case N needs to be a multiple of the number of subdirectories the MCRaT code has created and it has to be greater than or equal to the number of subdirectories; of course, the larger the number of MPI processes used, the less work each process has to do, making the post-processing code faster. In the example `mc.par` file there are 3 directories created, so N has to be a multiple of 3 in this case. The minimum value of N can be 3; however, since the MERGE code is also parallelized with OpenMPI, the larger the number of processes used, the less time it takes to merge all of the MCRaT data. If $N = 6$, 3 processes will be allocated to merge the data from the first half of the frames from 200 to 3000, as specified in the `mc.par` file, and the other 3 will merge the data from the last half of the range of frames. If $N = 12$, the number of frames from 200 to 3000 will be divided into quarters and 3 processes will deal with the first quarter, while a second set of 3 processes will deal with the second quarter, so on and so forth.

The merged data is placed into another folder that the code creates named `ALL_DATA/`.

The standard work flow for MCRaT is as follows:

1. Compile the MCRaT and MERGE codes
2. Write the `mc.par` file in the correct directory
3. Run MCRaT
4. Use MERGE to post-process the MCRaT results

This work flow can be practiced using a spherical or cylindrical outflow problem that is included in MCRaT. These simulations can be run using any hydro simulation because MCRaT reads in the data from the hydro simulation and over writes it (in memory, not in the hydro file) with the correct values assuming a spherical or cylindrical outflow.

In order to run a spherical outflow simulation simply change the `SIMULATION_TYPE` parameter in `mcrat_inputs.h` file to `SPHERICAL`. Then recompile the code and run it as we outlined previously. In order to run a cylindrical outflow simulation, change the `SIMULATION_TYPE` parameter to `CYLINDRICAL`. These test simulations allow the user to become acquainted with MCRaT and test that MCRaT produces the expected results for these type of outflows.

3 The MCRaT Code

This section describes the algorithm and output of MCRaT. It will cover the steps that MCRaT takes to inject and scatter photons. A more in depth description of the physics is included later in this manual.

3.1 Assumptions about the Hydro Simulation

MCRaT has to make two important assumptions about the hydro simulation. These assumptions are that:

1. The velocity in the hydro simulation is normalized by the speed of light, c.
2. The pressure is radiation dominated. As such, we can calculate the temperature using $p = \frac{1}{3}aT^4$, where a is the radiation density constant.

3.2 Algorithm

MCRaT first reads the data from the `mc.par` file and then appropriately divides the MPI processes based on the number of angle ranges, if starting a new simulation, or, for continuing a simulation, reads the mc_chkpt file in each directory to determine which MPI processes still have work to complete and then distributes the MPI processes.

Each MPI process is assigned a range of frames in which the process injects photons into and then propagates through all the hydro simulation frames until the last hydro simulation frame, which is specified in the `mc.par` file. The photons have a `photon` data structure that fully describes each photon. This includes the photon's lab frame 4-momenta, the x, y, and z coordinates of the photon, the Stokes parameters of the photon, the weight of the photon, the number of times the photon has scattered, and the index of the hydro fluid element that the photon is located within.

In order to inject the photons, MCRaT reads in the first hydro frame, that the photons will be injected into, and chooses hydro elements that have radii, r, such that $R_{\rm inj} - c/(2 * \text{fps}) < r < R_{\rm inj} + c/(2 * \text{fps})$, where c is the speed of light, $R_{\rm inj}$ is the injection radius and fps is the hydro simulation frames per second, both specified in `mc.par`. These hydro elements also have to be within the angle range that the MPI process is injecting photons within. Thus, we choose a slab of hydro elements to inject photons into. The number of photons that would be expected from the $i^{\rm th}$ hydro element, n_i, is calculated using the energy density of the element and by assuming a weight for each MCRaT photon which is taken from the `mc.par` file. This is further explained in Section 4.1. Once the expected number of photons is calculated, the actual number is acquired by assuming that the photon injection is random, which means that we acquire the actual number of photons injected into the $i^{\rm th}$ hydro element, N_i by randomly drawing from a Poisson distribution with a mean of n_i. The total number of photons that would be injected across all the chosen hydro elements becomes $N = \sum N_i$. If N is within the minimum and maximum number of photons that the user wants, which is specified in the `mc.par` file, the calculation for the number of photons is complete. If not, then the code continues to recalculates n_i using a new weight for each photon, then recalculates N_i and N, and after rechecks so see if N is within the user specified number of photons again. This loop is exited when N falls within the user specified number of desired photons. If the code needs to inject more photons to meet this condition, it will adjust the weights to be smaller and vice versa.

Once N and N_i have been determined, the N_i photons are randomly placed within the $i^{\rm th}$ hydro element, with a random azimuthal angle uniformly distributed between 0 and 2π (since we assume cylindrical symmetry for the 2D hydro simulation), and they are given an energy based on the comoving temperature of the hydro element, thus the photons are injected in the comoving frame of the outflow. If the user specified that the

photons should be injected with a Wien spectrum then the frequencies, ν, are assigned to the N_i photons by sampling from a Wien spectrum defined by the i^{th} fluid element's comoving temperature. The same thing is done with a Blackbody spectrum if the user specified that the photons should be injected as a Blackbody spectrum. This sampling is done by using the acceptance-rejection Monte Carlo sampling of the spectra.

For their 4-momenta, the photons are assigned a random azimuthal angle, ϕ, from a uniform distribution between 0 and 2π, as well as a polar angle, which is calculated as $\theta = \arccos(2\xi - 1)$, where ξ is a random number drawn from a uniform distribution between 0 and 1. The i^{th} injected photon 4-momentum becomes:

$$p_i^\mu = \frac{h\nu_i}{c} \begin{pmatrix} 1 \\ \sin(\theta_i)\cos(\phi_i) \\ \sin(\theta_i)\sin(\phi_i) \\ \cos(\theta_i) \end{pmatrix}$$

After the 4-momenta are set, the photons are boosted into the lab (observer) frame using the velocity of the fluid element that the photons are located within. The lab frame 4-momenta are saved to each photon's `photon` data structure, as well as the photon's position and weight. The number of scatterings is initialized to 0 and the stokes parameters are initialized to 0 except for s0 which is set to 1. These photons are also denoted as type 'i', since they are the original injected photons.

Once the photons have been injected into the simulation and their 4-momenta have been determined, the scattering process through the hydrodynamic (HD) simulation occurs. MCRaT reads in a HD simulation frame, starting from the frame in which the photons were initially injected into up until the last HD simulation frame available, and determines which HD fluid elements each photon is in. MCRaT then calculates the scattering path length of each photon. In order to reduce memory and optimize the code, MCRaT determines the minimum and maximum radii of the photons, $R_{\min}$ and $R_{\max}$ respectively, and selects a slab of hydro fluid elements with radii, r_i such that

$$R_{\min} - \alpha\frac{c}{\text{fps}} < r_i < R_{\max} + \alpha\frac{c}{\text{fps}}$$

where α is a multiplicative factor that can be increased to ensure that the number of chosen hydro elements is greater than 0. The code finds which fluid element each photon is located within, and saves the index to the `photon` data structure. This data field only gets updated if the photon propagates outside of the fluid element. If this happens, then the code finds and saves the index of the new fluid element that the photon is located within. Sometimes, the code is not able to find a fluid block that a given photon is located within. This may happen due to a photon propagating outside of the range of radii of the chosen slab of hydro element, and is very rare. To deal with this, we set the scattering path length, l, to an arbitrary large value to ensure that the photon doesn't scatter, which would inaccurately change its energy. If the fluid element is correctly identified, the scattering path length is calculated as

$$l = -\frac{m_p}{\rho\sigma_{\text{T}}(1 - \beta\cos\theta_{fp})}\ln\xi$$

where we have inverted the exponential probability distribution for a photon scattering in a medium. Here, ξ is a random number uniformly distributed between 0 and 1, m_p is the proton mass, σ_{T} is the Thompson cross section, ρ is the lab frame density of the hydro

fluid element that the photon is located within, β is the velocity of the fluid element normalized by the speed of light (this should be already done in the hydro code), and θ_{fp} is the angle between the fluid element's and the photon's velocity vectors. From l, the time for each photon to scatter is calculated as $t_{\rm s} = l/c$, where c is the speed of light. The photon with the smallest t_s is assumed to scatter first, however that is not necessarily true since we use the full Klein-Nishina cross section in the electron rest frame to actually determine if the scattering occurs. The positions of all the photons are advanced by the time for the photon of interest to scatter, t_s.

In determining whether a photon will scatter or not, we first boost the chosen photon with the smallest t_s to the fluid rest frame and produce an electron by sampling the Maxwell-Boltzmann or Maxwell-Juttner distributions for non-relativistic and relativistic gases. The comoving temperature, T_i', of the fluid element is used to distinguish between which distribution is used. If $T_i' \geq 1 \times 10^7$ K the Maxwell-Juttner distribution is used, otherwise the Maxwell-Boltzmann distribution is used. When the Maxwell-Boltzmann distribution is used we choose 3 random x, y and z velocities from gaussian distributions with a standard deviation of $k_B T_i'/m_e$, where k_B is the Boltzmann constant and m_e is the mass of the electron. The velocities are normalized by the speed of light and used to calculate the Lorentz factor of the electron in the fluid frame. To get the full 4-mometum of the electron, the azimuthal angle of the electron's direction fo travel is drawn from a uniform distribution between 0 and 2π and the relative polar angle (between the electron velocity vector and the incoming photon's velocity vector) is acquired by acceptance-rejection sampling the distribution $(1-\beta\cos(\theta))\sin(\theta)$. We then perform rotations based on the velocity vector of the photon that will scatter to ensure that the electron velocity is oriented appropriately relative to the photon's velocity.

Once the electron has been initialized, the photon and the electron are boosted into the electron rest frame. Then the photon velocity vector is rotated such that it is moving directly along the x-axis towards the stationary electron at the origin of the coordinate system. We then use the Klein Nishina (KN) cross section including the effects of polarization, $\sigma_{\rm KN}$, to determine if the photon will actually scatter. The ratio $\sigma_{\rm KN}/\sigma_T$ is calculated and a random number, ξ, from a uniform distribution between 0 and 1 is drawn. If $\xi > \sigma_{\rm KN}/\sigma_T$ the photon does not scatter, otherwise it does scatter with the electron.

If the photon scatters with the electron, a random θ and ϕ are drawn from their respective distributions that are derived from the KN cross section including polarization. The photon's new 4-momentum and stokes parameters are calculated and then the scattered photon is rotated to its original orientation with respect to the previously stationary electron. The photon is then Lorentz boosted twice, back to the lab frame. The process of keeping track of the Stokes parameters and calculating them is covered in subsection 4.3.

If the photon does not scatter, then the next photon with the second shortest time to scatter is chosen and the process to create an electron and determine if the photon will scatter repeats. This loop continues until a photon scatters or all of the photons have been determined to not scatter.

While this loop is occurring, the photons in the simulation are being propagated incrementally by the change in scattering time for each photon until the time for the next hydro simulation frame is reached. The same thing occurs when a photon does scatter, the photons get propagated through space using the scattering time and the process of determining each photons' time to scatter, etc. repeats until the time corresponding to

the next hydro frame is reached.

Once the time limit for the next hydro fluid frame is reached, MCRaT saves all the photons' information in the `photon` data structure into the HDF5 file under a group with the same HD frame number that the photons were being scattered in. Once the data is written, a checkpoint file is saved for MCRaT to be able to continue the simulation if it were interrupted. Then, the next simulation frame is loaded and the process of determining photons' time to scatter, actual probability of scattering and resulting scattered 4-momenta until the next HD simulation frame time is reached. This repeats until the last HD simulation frame is reached.

If the user is taking thermal synchrotron photons under consideration, then the thermal synchrotron photons are emitted into the flow in the frame after the initial set of photons are injected into the flow. The emission is done in the comoving frame of the flow. The number of photons emitted is calculated and iterated over in the same manner as the injection of photons. In the thermal synchrotron case the number of photons emitted in the i^{th} fluid element is calculated by integrating the low energy tail of the blackbody spectrum divided by $h\nu$, corresponding to the temperature of that fluid element, from 10 Hz to the cyclotron frequency of that cell. Another difference is that the total number of photons is limited by the `CYCLOSYNCHROTRON_REBIN_E_PERC` switch times the value of the 17^{th} line in the `MCPAR` file. The last major difference is that the emitted synchrotron photons are set to have the energy corresponding to the cyclotron frequency of the i^{th} fluid element that they are placed in.

These synchrotron photons are placed at the center of their corresponding fluid elements. If, when MCRaT is calculating the scattering path length of each photon, one of these synchrotron photons is determined to scatter, it is placed at a random position within its cell, then another synchrotron photon is created to replace it in the pool of photons at the center of the fluid element. After, the scattering process occurs.

In order to keep the number of photons manageable, MCRaT checks the number of scattered synchrotron photons every 1000 scatterings. If the number is smaller than the 17^{th} line in the `MCPAR` file (the maximum number of photons the user wants injected in each frame) then nothing happens, but if the number if larger then MCRaT rebins the photons. The rebinning is done in energy and in space with the number of energy bins, N_E, calculated as `CYCLOSYNCHROTRON_REBIN_E_PERC`$\times N_{\max}$, where `CYCLOSYNCHROTRON_REBIN_E_PERC` is a switch that the user sets in the `mcrat_input.h` file and $N_{\max}$ is the value of the 17^{th} line in the `MCPAR` file. The number of spatial bins, N_θ, is calculated as $\Delta\theta$/`CYCLOSYNCHROTRON_REBIN_ANG`, where $\Delta\theta$/`CYCLOSYNCHROTRON_REBIN_ANG` is the angular difference between the largest photon location in theta and the smallest photon location in theta and `CYCLOSYNCHROTRON_REBIN_ANG` is is a switch that the user sets in the `mcrat_input.h` file. $\Delta\theta$/`CYCLOSYNCHROTRON_REBIN_ANG` will most likely be the range of angles that the photons are injected into so that use can anticipate this value a priori. If $N_E \times N_\theta > N_{\max}$, then MCRaT will throw an error pointing this out. Otherwise the rebinning occurs by averaging photons' values within a given energy and spatial bin. The scattering and general steps of MCRaT continues and iterates until the time for the next hydro fluid frame is reached. Before the photons are saved, any photons that have energies that are smaller than the energy corresponding to the thermal synchrotron energy of the cell they are located in gets absorbed and becomes a null photon.

If the MPI process has more photons to inject, then the process of injecting/emitting the photons into the next frame and propagating those photons through the hydro simulation repeats. If all of the MPI processes have finished injecting and propagating photons

for the hydro simulation frames assigned to them, the code combines all of the photons' data from each MPI process into `mcdata_`i`.h5` files where i is the hydro simulation frame number. The output is covered in the next section.

3.3 Saved Quantities

Each MCRaT MPI process outputs 3 files. These files are named `mc_proc_`i`.h5`, `mc_chkpt_`i`.dat`, and `mc_output_`i`.log` where i is the number for the MPI process producing those files. As the end of the simulation, MCRaT produces the actual data files named `mcdata_`f`.h5` where f is the hydro simulation file frame.

The `mc_chkpt_`i`.dat` file is a binary file that contains the data from the MPI process that allows it to continue the simulation if MCRaT gets interrupted for any reason. This information is: the MPI process' range of hydro frames to inject photons into, the frame that the current set of photons were injected in, the frame in which the photons were recently scattered in, the current time of the MCRaT simulation, the number of photons that the code is keeping track of, and the `photon` data structure for each photon in the simulation.

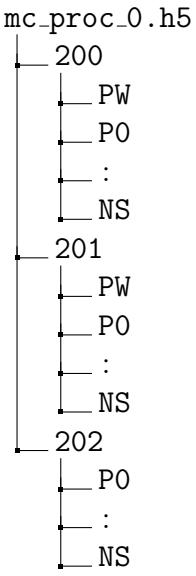
The `mc_proc_`i`.h5` file is a HDF5 file that contains all the data for the photons that the i^{th} MPI process injects and propagates through the hydro simulation. The data is organized by the hydro simulation frame which constitutes a HDF5 group. Within the group, there are the following datasets, depending on if `STOKES_SWITCH`, `COMV_SWITCH` or `SAVE_TYPE` ids are set to ON or OFF.

- PW - The photon weight for any photons that were injected into the frame. If there were no photons injected, then this data set does not exist.
- P0 - the energy of the photons (in erg/c)
- P1 - the magnitude of the photons' 4-momentum in the x direction (in erg/c)
- P2 - the magnitude of the photons' 4-momentum in the y direction (in erg/c)
- P3 - the magnitude of the photons' 4-momentum in the z direction (in erg/c)
- COMV_P0 - the comoving energy of the photons (in erg/c)
- COMV_P1 - the magnitude of the photons' comoving 4-momentum in the x direction (in erg/c)
- COMV_P2 - the magnitude of the photons' comoving 4-momentum in the y direction (in erg/c)
- COMV_P3 - the magnitude of the photons' comoving 4-momentum in the z direction (in erg/c)
- R0 - the x position of the photons (in cm)
- R1 - the y position of the photons (in cm)
- R2 - the z position of the photons (in cm)
- S0 - the normalized I Stokes parameter for the photons

- S1 - the normalized Q Stokes parameter for the photons
- S2 - the normalized U Stokes parameter for the photons
- S3 - the normalized V Stokes parameter for the photons
- NS - the cumulative number of scattering each photon has undergone
- PT - the photon type of each photon (these are saved as ASCII decimal numbers that can be mapped to a character)

The currently available photon types are 'i' for the injected photons and 'c' for thermal synchrotron photons.

These data sets include information for all of the photons after they were scattered through the given hydro simulation frame. The order of the data is based on the chronological order in which each set of photons were injected. For the example `mc.par` file in subsection 2.4, if the user uses N MPI processes such that each process injects photons into 2 frames, the `mc_proc_0.h5` file would look like:



where all the datasets in group 200 would have only n_0 elements, where n_0 is the number of photons first injected into hydro frame 200. If we let n_1 be the number of photons injected in frame 201, the dataset PW in group 201 will have n_1 elements while the rest of the datasets will have n_0 elements, corresponding to the information for the photons first injected in frame 200 that are propagated through frame 201, *followed by* n_1 elements for the newly injected photons in frame 201.

The last file produced by the MCRaT MPI processes are named `mc_output_`i`.log`. These are files that output the progress of the given process, allowing the user to keep track of how far along MCRaT is in the radiation transfer simulation and how much more work the code has left to complete. This file contains the number of scatterings that the code completes in increments of 1000 scatterings as well as the average photon energy in units of ergs, the comoving temperature, in Kelvin, of the hydro fluid element that was used in the most recent scattering, and the most recent time step for a photon to scatter and the current time in the simulation. When all the scatterings are completed for a given frame, the code prints out the average cumulative number of scatterings that each photon has undergone, as well as the the maximum and minimum number of scatterings for the photons.The code also prints out the average radius of the MCRaT photons, the simulation time and the frames left to inject photons into and the last hydro simulation frame.

When the MCRaT MPI process has completed the Monte Carlo scattering portion of the code it will print: "Process i has completed the MC calculation." where i is the ID of the MPI process. When all of the MPI processes are complete with the Monte Carlo scattering they will collectively produce the `mcdata_`f`.h5` files which contain the merged data from each MPI process within a given angle range.

These `mcdata_`f`.h5` files are formatted in the same way that the HDF5 groups are formatted in the `mc_proc_`i`.h5` files with the exception of the PW dataset being included in each frame, regardless of whether additional photons were injected into the simulation at that frame or not.

3.4 Post-processing Algorithm

When the user has completed running MCRaT for each angle range that they are interested in, the data for every simulated photon needs to be merged. The MERGE code that is included with the download of MCRaT uses parallel HDF5 I/O to accomplish and speed up this post-processing.

The code takes the MPI processes and distributes them evenly among the MCRaT produced angle subdirectories. If there are n_θ angle subdirectories, the MPI processes are grouped into sets of n_θ, with each MPI process operating withing a given subdirectory. Then, the code distributes the hydro simulation frames that need to be created among the groups of n_θ MPI processes. It also counts how many processes were originally used by MCRaT in each directory so if MCRaT is independently run 3 different times for 3 different ranges of angles, but with each of the 3 runs using a different number of MPI processes, the post-processing code will still work.

These processes open the `mc_proc_`i`.h5` files and tally how many photons will be merged to produce the final `mcdata_`f`.h5` file in the `ALL_DATA` directory. Using this information, the code constantly ensures that any files created by MERGE files that were created, before the code was terminated for any reason, are not corrupted; thus the MERGE code can be stopped and started again without worrying about corrupting data.

The code iterates over the photons injected from first to last in the MCRaT simulation, by reading the data from the `mc_proc_`i`.h5` files. These photon data are then collectively appended to the final `mcdata_`f`.h5` file.

4 Physics Included in MCRaT

This section describes the physics of the MCRaT code injecting photons and scattering them through the hydro simulation. We also go into the physics of emitting thermal synchrotron photons into the outflow and how we rebin these photons to ensure that the number of photons doesn't increase without bound.

4.1 Injecting Photons

The code selects HD fluid elements at a given radius and calculates the expected number of photons in each element as

$$n_i = \frac{\xi T_i'^3 \Gamma_i}{w} dV_i \tag{1}$$

where ξ, the number density coefficient which is used to calculate the number density of photons as $n_\gamma = \xi T'^3$, is 20.29 for a Blackbody spectrum or 8.44 for a Wien spectrum, T_i' is the comoving temperature of the fluid element, Γ_i is the Lorentz factor of the element, w is the weighting factor of each injected photon, and dV_i is the volume of the fluid element.

The comoving temperature is calculated as

$$T_i' = \left(\frac{3p_i}{a}\right)^{\frac{1}{4}} \tag{2}$$

where p_i is the pressure of the fluid element, and a is the radiation density constant.

If the HD simulations used are 2D axis-symmetric, we assume cylindrical symmetry and calculate $dV_i = 2\pi x_i dA_i$ where x_i is the distance of the fluid element from the y-axis of the simulation and dA_i is the area of the element.

Once the number of photons is calculated, we draw a random number from a Poisson distribution using n_i as the average value in order to get the actual number of photons that will be injected into the i^{th} HD element, N_i. The weight, w, for a given set of injected photons can be adjusted accordingly in order to inject more or less photons into the simulation; this allows us to conserve energy and increase photon statistics for time resolved spectra. This weight is essentially how many real photons each MCRaT photon represents.

The total number of photons that would be injected across all the chosen hydro elements becomes $N = \sum N_i$. If N is within the minimum and maximum number of photons that the user wants, which is specified in the `mc.par` file, the calculation for the number of photons is complete. If not, then the code continues to recalculates n_i using a new weight for each iteration, then recalculates N_i and N, and after rechecks to see if N is within the user specified number of photons again. This loop is exited when N falls within the user specified number of desired photons. If the code needs to inject more photons to meet this condition, it will adjust the weights to be smaller and vice versa.

Once N and N_i have been determined, the N_i photons are placed at the center of the i^{th} hydro element, with a random azimuthal angle uniformly distributed between 0

and 2π (since we assume cylindrical symmetry for the 2D hydro simulation), and they are given an energy based on the comoving temperature of the hydro element, thus the photons are injected in the comoving frame of the outflow. If the user specified that the photons should be injected with a Wien spectrum then the frequencies, ν_i, are assigned to the N_i photons by sampling from a Wien spectrum defined by the i^{th} fluid element's comoving temperature. The same thing is done with a Blackbody spectrum if the user specified that the photons should be injected as a Blackbody spectrum. This sampling is done by using the acceptance-rejection Monte Carlo sampling of the spectra.

For their 4-momenta, the photons are assigned a random azimuthal angle, ϕ, from a uniform distribution between 0 and 2π, as well as a polar angle, which is calculated as $\theta = \arccos(2\xi - 1)$, where ξ is a random number drawn from a uniform distribution. The i^{th} injected photon 4-momentum becomes:

$$p_i^\mu = \frac{h\nu_i}{c} \begin{pmatrix} 1 \\ \sin(\theta_i)\cos(\phi_i) \\ \sin(\theta_i)\sin(\phi_i) \\ \cos(\theta_i) \end{pmatrix}$$

After the 4-momenta are set, the photons are boosted into the lab frame using the velocity of the fluid element that the photons are located within. The lab frame 4-momenta are saved to each photon's `photon` data structure, as well as the photon's position and weight. The number of scatterings is initialized to 0 and the stokes parameters are initialized to 0 except for s0 which is set to 1.

4.2 Scattering Photons

In order to scatter the photons, we first need to determine the distance that each photon will propagate before scattering. Thus we calculate each photon's mean free path and choose a random number that will distribute it according to an exponential distribution. We calculate the distance each photon will propagate before scattering as:

$$l = -\frac{m_p}{\rho\sigma_{\text{T}}(1 - \beta\cos\theta_{fp})}\ln\xi$$

where, ξ is a random number uniformly distributed between 0 and 1, m_p is the proton mass, σ_{T} is the Thompson cross section, ρ is the lab frame density of the hydro fluid element that the photon is located within, β is the velocity of the fluid element normalized by the speed of light (this should be already done in the hydro code), and θ_{fp} is the angle between the fluid element's and the photon's velocity vectors. From l, the time for each photon to scatter is calculates as $t_{\text{s}} = l/c$, where c is the speed of light. The photon with the smallest t_s is assumed to scatter first, however that is not necessarily true since we use the full Klein-Nishina cross section in the electron rest frame to actually determine if the scattering occurs. The positions of all the photons are advanced by the time for the photon of interest to scatter, t_s.

In determining whether a photon will scatter or not, we first boost the chosen photon with the smallest t_s to the fluid rest frame and produce an electron by sampling the Maxwell-Boltzmann or Maxwell-Juttner distributions for non-relativistic and relativistic gases. The comoving temperature, T_i', of the fluid element is used to distinguish between which distribution is used. If $T_i' \geq 1 \times 10^7$ K the Maxwell-Juttner distribution is used, otherwise the Maxwell-Boltzmann distribution is used. When the Maxwell-Boltzmann

distribution is used we choose 3 random x, y and z velocities from gaussian distributions with a standard deviation of $k_B T_i'/m_e$, where k_B is the Boltzmann constant and m_e is the mass of the electron. The velocities are normalized by the speed of light and used to calculate the Lorentz factor of the electron in the fluid frame. To get the full 4-mometum of the electron, the azimuthal angle of the electron's direction of travel is drawn from a uniform distribution between 0 and 2π and the polar angle is acquired by acceptance-rejection sampling of the polar angle distribution $(1 - \beta\cos(\theta))\sin(\theta)$. We then perform rotations based on the velocity vector of the photon that will scatter to ensure that the electron velocity is oriented relative to the photon's velocity.

Once the electron has been initialized, the photon and the electron are boosted into the electron rest frame. Then, the photon velocity vector is rotated such that it is directly along the x-axis with the stationary electron at the origin of the coordinate system. We then use the Klein Nishina (KN) cross section, $\sigma_{\rm KN}$, to determine if the photon will actually scatter. Using $\sigma_{\rm T}$ as the Thomson cross section, the ratio $\sigma_{\rm KN}/\sigma_T$ is calculated and a random number, ξ, from a uniform distribution between 0 and 1 is drawn. If $\xi > \sigma_{\rm KN}/\sigma_T$ the photon does not scatter, otherwise it does scatter with the electron.

If $\xi \leq \sigma_{\rm KN}/\sigma_{\rm T}$ then we sample the differential KN cross section to determine the angles, $\theta_{\rm sc}$ and $\phi_{\rm sc}$, that the photon will be scattered into. The differential cross section given by Lundman et al. (2014) is

$$\frac{{\rm d}\sigma_{\rm KN}}{{\rm d}\Omega}(\theta_{\rm sc}, \phi_{\rm sc}) = \frac{r_0^2}{2}\left(\frac{\epsilon}{\epsilon_0}\right)^2 \times \left\{\frac{\epsilon_0}{\epsilon} + \frac{\epsilon}{\epsilon_0} - \sin^2\theta_{\rm sc}\left(1 - q\cos 2\phi_{\rm sc} + u\sin 2\phi_{\rm sc}\right)\right\} \tag{3}$$

where r_0 is the classical electron radius, ϵ_0 is the incoming photon energy scaled by the electron rest mass, $\epsilon = \epsilon_0/(1 + \epsilon_0[1 - \cos\theta_{\rm sc}])$ is the scattered photons energy scaled by the electron rest mass, and q and u are the stokes parameters discussed further in Section 4.3. The outgoing photon's $\theta_{\rm sc}$ is acquired by rejection sampling the differential cross section integrated over $\phi_{\rm sc}$ following the method outlined by Mathews (2013) for maximum efficiency. We acquire $\phi_{\rm sc}$ by choosing a random uniformly distributed value between [0, 2π] when $q = u = 0$, otherwise we apply a rejection sampling method to the differential KN cross section. The case of $q = u = 0$ removes the KN cross section's dependence on $\phi_{\rm sc}$, which allows us to choose a value between [0, 2π].

We outline our method of sampling Equation 3 to acquire $\phi_{\rm sc}$ when the incoming photon is polarized. The azimuthal angle that gives the maximum of the KN differential cross section, $\phi_{\rm m}$, can be solved as

$$\phi_{\rm m} = \frac{1}{2}\left|\arctan\left(\frac{-u}{q}\right)\right| \tag{4}$$

We then plug $\phi_{\rm m}$ and the previously acquired $\theta_{\rm sc}$ into the KN differential cross section to get the normalization for the rejection sampling method, $\frac{{\rm d}\sigma_{\rm KN}}{{\rm d}\Omega}(\theta_{\rm sc}, \phi_{\rm m})$. We proceed as follows:

(1) Draw a random number ξ uniformly distributed between 0 and 1

(2) Draw a random $\phi_{\rm sc}$ uniformly distributed between 0 and 2π

(3) Calculate $\frac{{\rm d}\sigma_{\rm KN}}{{\rm d}\Omega}(\theta_{\rm sc}, \phi_{\rm sc})$

(4) Determine if $\xi \leq [\frac{{\rm d}\sigma_{\rm KN}}{{\rm d}\Omega}(\theta_{\rm sc}, \phi_{\rm sc})][\frac{{\rm d}\sigma_{\rm KN}}{{\rm d}\Omega}(\theta_{\rm sc}, \phi_{\rm m})]^{-1}$

(5) If the above condition is met, accept the value of $\phi_{\rm sc}$, otherwise repeat the sampling

The photon's new 4-momentum and stokes parameters are calculated and then the scattered photon is rotated to its original orientation with respect to the previously stationary electron. The photon is then Lorentz boosted twice, back to the lab frame.

If the photon does not scatter, then the next photon with the second shortest time to scatter is chosen and the process to create an electron and determine if the photon will scatter repeats. This loop continues until a photon scatters or all of the photons have been determined to not scatter.

4.3 Polarization

The Stokes parameters are a vector, $S = (I, Q, U, V)$ that holds information about the polarization of electromagnetic radiation. I is the intensity of the electromagnetic radiation, Q and U describe the orientation of the polarization ellipse, and V describes the ratio of the principal axis of the polarization ellipse (Rybicki & Lightman, 1979). We follow the convention set by McMaster (1961) and Lundman et al. (2014) where $Q = +1$ is oriented with the y-axis of the Stokes plane and $Q = -1$ is oriented with the x-axis of the Stokes plane. The $+U$ axis is rotated 45° clockwise with respect to the $+Q$ axis and the $-U$ axis is rotated 45° clockwise with respect to the $-Q$ axis. Furthermore, we normalize the Stokes parameters such that $I = 1$ at all times giving us $s = (1, Q/I, U/I, V/I) = (1, q, u, v)$. In the code, it is stored as $s = (S0, S1, S2, S3)$. In our simulations, we only consider linear polarization which means that we ignore any contribution by v (S0). This is appropriate since we assume that electron spins, which directly affect v, are isotropically distributed.

In MCRaT all photons are initialized to have no polarization; thus, we set $s = (1, 0, 0, 0)$. Each photon becomes 100% polarized from the very first scattering that it undergoes, which does not bias the results.

In order to deal with polarization, we follow the prescription given by Lundman et al. (2014). We lorentz boost the photons' 4 momenta from the lab frame to the fluid frame, then from the fluid frame to the electron rest frame. In the electron rest frame we conduct our scattering using the full Klein Nishina (KN) cross section and then scatter the Stokes parameters using Fanno's Matrix. Afterwards, we deboost the photons from the electron rest frame and the fluid rest frame back to the lab frame.

In the lab frame, the Stokes plane is oriented such that the $+Q$ axis is pointing in the $-\hat{\phi}$ direction and the $-Q$ axis is pointing in the $-\hat{\theta}$ direction. The $+U$ axis is rotated 45° clockwise with respect to the $+Q$ axis and the $-U$ axis is rotated 45° clockwise with respect to the $-Q$ axis. χ is measured clockwise from the $+Q$ axis towards the $+U$ axis. $\hat{\phi}$ and $\hat{\theta}$ are the orthonormal azimuthal and polar unit vectors in a spherical coordinate system where the radial unit vector is parallel to the photon's momentum vector.

Each boost to another reference frame entails rotating the Stokes plane using the Muller rotation matrix (McMaster, 1961) given by

$$\mathbf{M}[\phi] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\phi & -\sin 2\phi & 0 \\ 0 & \sin 2\phi & \cos 2\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

where the angle of rotation, ϕ, corresponds to the rotation that orients the y-axis of the Stokes plane perpendicular to the photon three momentum and the velocity vector of the

frame that the photon will be boosted into. The equation for ϕ is given in Appendix B of Lundman et al. (2014). After each boost, we rotate the Stokes plane again to ensure that the y-axis of the Stokes plane is perpendicular to the plane formed by the reference frame's z-axis and the photon's three momentum.

We conduct a scattering of the photon's 4 momenta as outlined in section 4.2. Lorentz boosting to all the proper frames and sampling the KN cross section.

Once the scattering is completed, and still in the electron rest frame after undoing all of the rotations that placed the photon along the x axis moving towards the electron at the origin of the coordinate system, we conduct the scattering of the Stokes parameters. First, we use the Muller matrix to rotate the Stokes plane such that the Stokes plane's y-axis is perpendicular to the plane formed by the incoming and outgoing photon three momenta (McMaster, 1961; Lundman et al., 2014). Then, we use Fanno's matrix, $T[\theta_{\rm sc}, \epsilon_0, \epsilon]$, (McMaster, 1961) to determine the resulting polarization, $\tilde{s} = T[\theta_{\rm sc}, \epsilon_0, \epsilon]s$. Fanno's matrix is

$$T[\theta_{\rm sc}, \epsilon_0, \epsilon] = \begin{pmatrix} 1 + \cos^2\theta_{\rm sc} + (\epsilon_0 - \epsilon)(1 - \cos\theta_{\rm sc}) & \sin^2\theta_{\rm sc} & 0 \\ \sin^2\theta_{\rm sc} & 1 + \cos^2\theta_{\rm sc} & 0 \\ 0 & 0 & 2\cos\theta_{\rm sc} \end{pmatrix} \quad (6)$$

where, similar to Krawczynski (2011), we have excluded the factors in front since they cancel out when we normalize the scattered Stokes parameters by I as mentioned above. We have also excluded the fourth row and column of the matrix since we only consider linear polarization. In this case, $\theta_{\rm sc}$ is acquired by calculating the angle between the photon's original incoming 3 momentum and the new outgoing 3 momentum. We calculate this as:

$$\theta_{\rm sc} = \arccos(\overrightarrow{p_{\rm in}} \cdot \overrightarrow{p_{\rm out}}) \quad (7)$$

Once the new stokes parameters are acquired, they are normalized by dividing by $S0$. Then, the stokes parameters are rotated to ensure that the y-axis of the Stokes plane is perpendicular to the plane formed by the electron rest frame's z-axis and the outgoing photon's 3 momentum. After the photon is deboosted back to the lab frame and the stokes parameters are rotated for each lorentz boost.

We reproduce Depaola's (2003) result, as shown in Figure 1, which verifies the sampling algorithm of the differential KN cross section. Figure 1 shows the resulting modulation curve when our Klein Nishina (KN) cross section is monte carlo sampled in black. The analytic profile of the cross section as a function of $\phi_{\rm sc}$ calculated by Depaola (2003) is shown as the solid blue line. They sample the KN cross section for a photon beam with 100% polarization along the $+Q$ direction, with an energy of 100 keV and $85° < \theta_{\rm sc} < 90°$. Accounting for the differences in the positive stokes parameters in the convention used, we are able to use our method of sampling the stokes parameter to acquire the proper distribution.

Additionally, reproducing Krawczynski's (2011) results, in Figure 2, tests the Lorentz transform portion of MCRaT. Krawczynski (2011) scattered a beam of photons, 100% polarized along the $+Q$ axis with frequency $\omega = 10^{12}$ Hz, with a beam of electrons moving at a Lorentz factor of $\gamma = 100$. Following their setup, we produce distributions of the resulting lab frame quantities which we show in Figure 2. These distributions are identical, with the exception of normalization, to the distribution acquired by Krawczynski (2011) in their Figure 6. The difference in the signs of Q and U in our distribution with respect to Krawczynski's (2011) Figure 6 is due to the differences in the orientation of the $+Q$ and $+U$ axis of the Stokes plane.

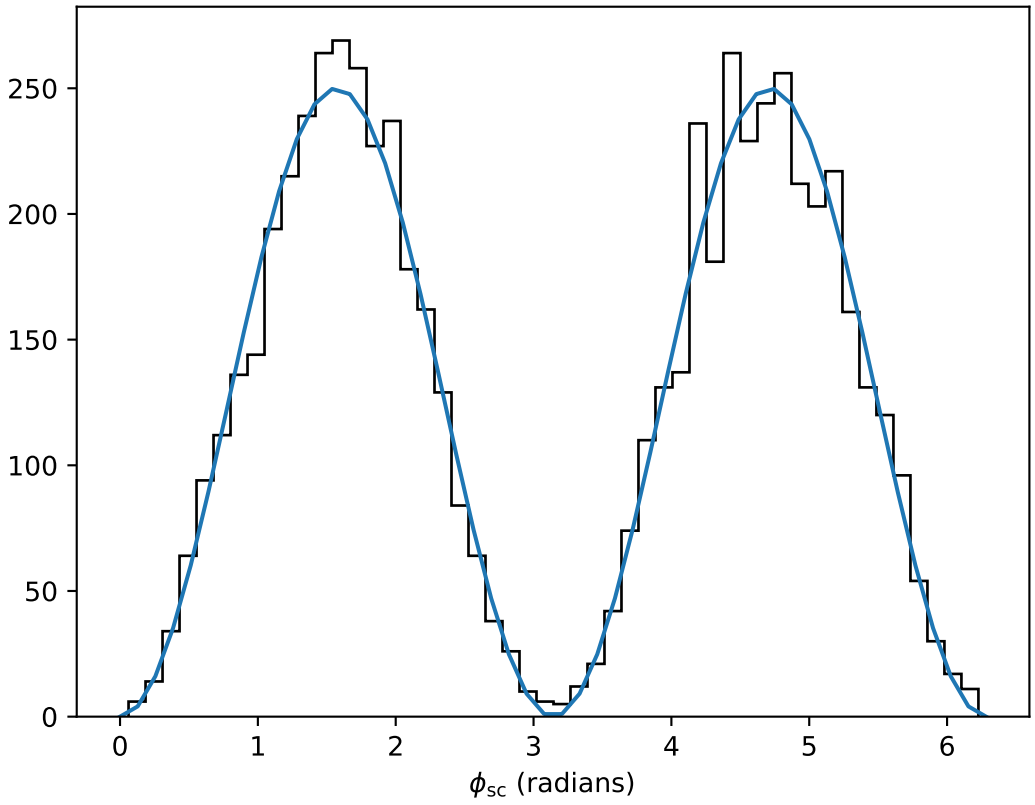


Figure 1: Plot of the distributed ϕ_{sc} from sampling the Klein Nishina cross section compared to the analytic cross section acquired by Depaola (2003). The distribution is shown in black and the analytic profile is shown by the blue curve.

In order to test the code globally, we reproduce the results of Lundman et al. (2014). To do so, we imposed the analytic jet structure provided by Lundman et al. (2014) on some FLASH HD simulation frames. This is similar to Lazzati (2016) simulating a variety of outflows by imposing an analytic solution onto hydrodynamic simulation files. The domain of this simulation is 2.5×10^{13} cm along the direction of the jet and 5×10^{12} cm along the x axis. We used $\sim 6 \times 10^5$ photons to conduct our code validation for a wide structured jet with $\theta_j = 0.1$ radians ($\sim 5.7°$), $\Gamma_0 = 100$ and $L = 3 \times 10^{50}$ erg/s. This is the same case exhibited in Lundman et al.'s (2014) wide jet with the exception of the value of L that we use, which was chosen to maximize the number of photons that reached the photosphere before they approached the edge of the domain of the hydrodynamic simulation. There are two major differences between the simulation conducted by Lundman et al. (2014) and the MCRaT simulation: 1) MCRaT uses the full Klein Nishina Cross section to determine if photons scatter while Lundman et al.'s (2014) simulation uses the Thomson cross section and 2) The photons in MCRaT are not permitted to immediately escape to infinity if the randomly drawn optical depth is small enough while photons in Lundman et al.'s (2014) simulation are allowed to do so.

In Figure 3 we show the results of our validation. The polarization acquired by Lundman et al. (2014) is shown as the red dotted line, the black points with 1σ error bars show the polarization acquired by MCRaT, and the grey dotted line shows $\Pi = 0\%$

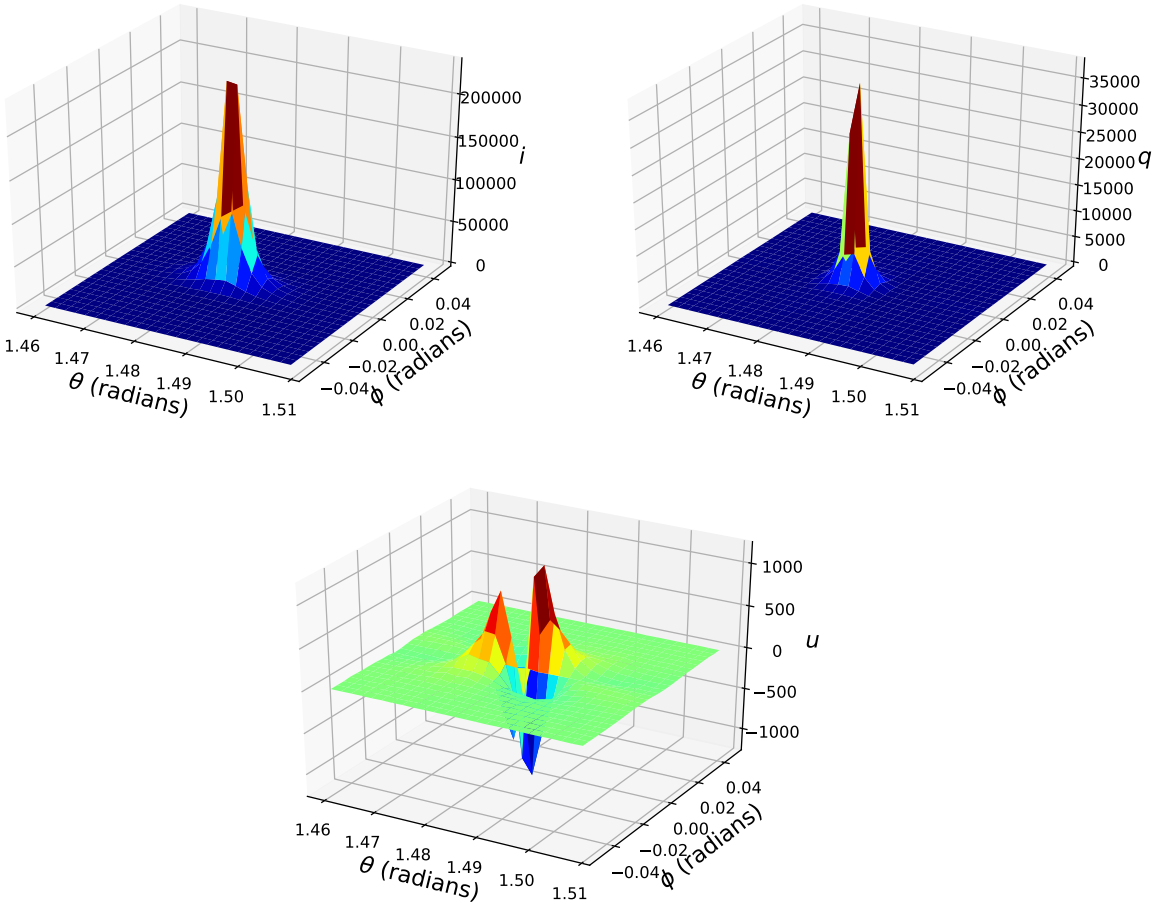


Figure 2: The distribution of the stokes parameters as a function of the scattered photon's θ and ϕ values. These distributions are morphologically similar to the distribution acquired by Krawczynski (2011) in thier Figure 6.

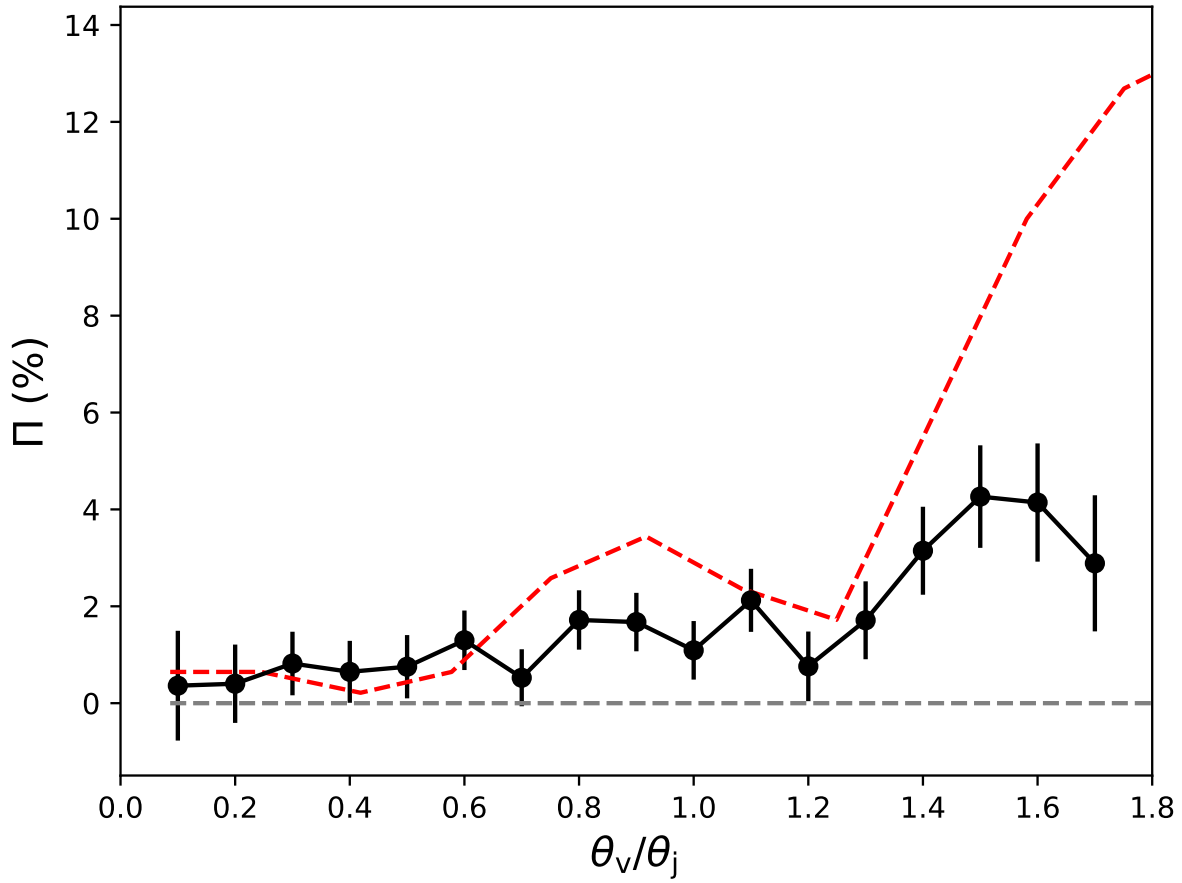


Figure 3: A comparison between the polarization acquired by Lundman et al. (2014) for a structured jet with $\theta_j = 0.1$ radians $\Gamma_0 = 100$ and $L = 3 \times 10^{50}$ erg/s, shown by the red dashed line, and the MCRaT acquired polarization shown by the black points with 1σ error bars. The dashed grey line denotes $\Pi = 0\%$ for reference. We find agreement until $\theta_v/\theta_j \approx 1.6$ ($\theta_v \approx 9°$) where the MCRaT photons are no longer decoupled from the flow by the time they reach the edge of the simulation domain.

for reference. In addition to finding that the Stokes u parameter vanishes ($\sum u \approx 0$), which is expected for an axis symmetric jet (Ito et al., 2014; Lundman et al., 2014), we find that MCRaT is able to recover Lundman et al.'s (2014) polarization profile relatively well. We are also able to recover the change in the sign of the stokes Q parameter at $\theta_{\rm v}/\theta_{\rm j} \sim 1.2$ that Lundman et al. (2014) find in their results. The MCRaT result is slightly lower than the polarization acquired by Lundman et al. (2014) due to the fact that the analytic jet profile is mapped onto a discretized grid; this has the effect of decreasing the gradients in the jet profile that would contribute to a larger polarization. Furthermore, Lundman et al.'s (2014) polarization profile contains a minimum of 200 photons in each angle bin, while the results of the MCRaT validation contain at least ~ 10000 photons in each angle bin, increasing the general precision of the MCRaT results.

For $\theta_{\rm v}/\theta_{\rm j} \gtrsim 1.6$ ($\theta_{\rm v} \approx 9^\circ$) the MCRaT polarization begins to decrease again, coming into strain with what is expected. This is due to the fact that the simulation files that we impose the analytic jet equations onto have a finite domain. Even when the photons reach the edge of the domain (2.5×10^{13} cm) at $\theta_{\rm v} \gtrsim 9^\circ$, they haven't fully decoupled from the photosphere (which would be located at $r \sim 1 \times 10^{14}$ cm (Lundman et al., 2014)) thus decreasing the detected polarization.

4.4 Cyclo-synchrotron Radiation

4.4.1 Emitting Cyclo-synchrotron (CS) Photons

The emission of the cyclo-synchrotron (CS) photons is similar to injecting photons into the outflow. We iteratively calculate the number of photons to emit as synchrotron photons in the frame and assign them random 4 momenta with an energy corresponding to the local cyclotron frequency. We consider the source of CS photons to be constant in time so we emit a constant number and if any synchrotron photon gets upscattered, we replace it with another synchrotron photon.

These photons are emitted into a region that the original injected photons are passing through so it accounts for the CS photons that would be upscattered in the flow during the time of interest of the original injected photons.

The maximum number of photons that are emitted, $N_{e,max}$, are calculated as the percentage of the maximum number of photons the user wants injected (see line 17 of the example mc.par file in section 2.4). For example, if `CYCLOSYNCHROTRON_REBIN_E_PERC` is set to 0.5 and the value of the 17$^{\rm th}$ line is 3000, the upper limit for the number of synchrotron emitted photons is 1500. The number density of CS photons, $n_{s,i}$, in a given cell is acquired by integrating the blackbody photon number spectrum up to the cyclotron frequency, ν_B. We calculate the number density as

$$n_{s,i} = \int_0^{\nu_B} \frac{u_\nu}{h\nu} d\nu \frac{dV_i}{w} = \int_0^{\nu_B} \frac{8\pi\nu^2}{c^3(e^{\frac{h\nu}{kT_i'}} - 1)} d\nu \frac{dV_i}{w} \tag{8}$$

where w is the weight of each CS photon, dV_i is the volume of the HD cell, T_i is the temperature of the same cell, h is Planck's constant and

$$\nu_B = \frac{eB}{2\pi m_e c} \tag{9}$$

where e is the electron charge and B is the magnetic field within the cell. Once $n_{s,i}$ is calculated, we draw a random number from a Poisson distribution with a mean of $n_{s,i}$ to

acquire the total actual number of photons that are emitted into the cell. We get the total number of emitted photons, N_e, by summing over each cell's $n_{s,i}$. If $N_e > N_{e,max}$, then the w is made larger and then the aforementioned calculations are made. If $N_e \leq N_{e,max}$ then the code proceeds to assign the CS photons a random 4 momentum and assigns them to be at the center of their respective hydrodynamic cell.

When a CS photon gets scattered, it is placed randomly within its HD cell and the scattering proceeds as it normally would. Then another CS photon gets added to the pool of photons at the center of their respective HD cell with the proper ν_B.

4.4.2 Calculating B

MCRaT provides 2 ways to calculate the magnetic field in the outflow, in order to calculate ν_B. It either uses the internal energy fo the flow to calculate B, when the `B_FIELD_CALC` switch is set to `INTERNAL_E`, or it uses the total energy of the outflow to calculate B, when the `B_FIELD_CALC` switch is set to `TOTAL_E`.

- Total Energy

  For a given plasma, the luminosity from matter and radiation can be written as $L = 4\pi(\rho c^2 + 4p)\Gamma^2 c r^2$, where c is the speed of light, ρ is the plasma density, p is the pressure, Γ is the bulk Lorentz factor, and r is the radius of the plasma away from its point of origin. The luminosity due to the magnetic field is $L_B = 4\pi(B^2/8\pi)\Gamma^2 c r^2$ where B is the magnetic field of the plasma. The fraction of energy between the magnetic outflow luminosity and the total outflow luminosity can be written as

  $$\epsilon_B = \frac{L_B}{L} = \frac{B^2}{8\pi(\rho c^2 + 4p)} \tag{10}$$

  If the user supplies ϵ_B to MCRaT via the `EPSILON_B` parameter then the code uses the above formula to solve for the magnetic field in each HD cell.

- Internal Energy

  In the internal energy case, the magnetic field energy density is equated to the electron thermal kinetic energy.

  $$\frac{B^2}{8\pi} = \frac{3}{2} n k T' \epsilon_B \tag{11}$$

  where n is the number density of electrons, k is the boltzmann constant and T is the plasma temperature.

MCRaT can also read in the magnetic field from a magneto-hydrodynamic simulation and use the magnitude of the magnetic field in calculating ν_B. This option is used when `B_FIELD_CALC` is set to `SIMULATION`.

4.4.3 Rebinning The Scattered CS Photons

The MCRaT code spatially rebins the CS photons in order to keep the number of photons that it needs to track down to a reasonable amount, which helps decrease computational time. In order to rebin the photons, MCRaT finds the photon with the smallest position in the polar direction, $\theta_{\min}$, and the photon with the largest position in the polar direction, $\theta_{\max}$. These typically correspond to the values that were specified in the `mc.par` file that

is read in by MCRaT. Then the size of each spatial bin, $\Delta\theta$ is set by the user through the `CYCLOSYNCHROTRON_REBIN_ANG` parameter in the `mcrat_input.h` file. The number of spatial bins, N_θ is then is determined as

$$N_\theta = \frac{\theta_{\max} - \theta_{\min}}{\Delta\theta} \tag{12}$$

The photons are also binned into energy bins, where the size of the energy bins, ΔE is determined as

$$\Delta E = \frac{\log(E_{\max}) - \log(E_{\min})}{N_E} \tag{13}$$

where $E_{\max}$ and $E_{\min}$ are the minimum and maximum energies of the CS photons and N_E is the number of energy bins acquired by multiplying the `CYCLOSYNCHROTRON_REBIN_E_PERC` value times the value of the 17^{th} line of the `mc.par` file, N_{17}.

To ensure that the rebinning is efficient, MCRaT checks that $N_\theta N_E < N_{17}$. If it is not, the the code throws an error. The user can then change either parameter to ensure that the aforementioned condition is met.

When rebinning the photons in polar angle and energy, the following values of photons that fall within a given energy and θ bin are averaged and then assigned to a new rebinned photon (the quantities for this rebinned photon are represented with a tilde). For a given energy bin i and θ bin j, the new rebinned photon would have the following four momentum

$$\tilde{p}^{\mu}_{i,j} = \frac{< E >}{c} \begin{pmatrix} 1 \\ \sin(< \theta >)\cos(< \phi >) \\ \sin(< \theta >)\sin(< \phi >) \\ \cos(< \theta >) \end{pmatrix}$$

where θ and ϕ are the directional angles of the 4 momentum and the $< \cdots >$ denote weighted averages of the various quantities over the photons that fall within bin (i, j). The stokes parameters for the rebinned photon becomes $\tilde{s} = (1, < q >, < u >, < v >)$. The photon weight becomes $\tilde{w} = \sum w_{i,j}$. The position vector of the rebinned photon becomes

$$\overrightarrow{\tilde{r}}_{i,j} =< r > \begin{pmatrix} \sin(< \theta_p >)\cos(< \phi > - < \Delta\tilde{\phi}_p >) \\ \sin(< \theta_p >)\sin(< \phi > - < \Delta\tilde{\phi}_p >) \\ \cos(< \theta_p >) \end{pmatrix}$$

where $< r >$ is the average radius of the photons that lie in bin (i, j), $< \theta_p >$ is the average polar angle of the photons with the given bins and , for these same photons, $< \Delta\tilde{\phi}_p >$ is the average displacement angle taken across each photon's 4-momentum azimuthal angle, ϕ, and its positional azimuthal angle, ϕ_p, that is $\Delta\tilde{\phi}_p = \phi - \phi_p$. This consideration of the displacement in the positional and 4-momentum azimuthal angles is necessary to ensure that the rebinned photons are still in appropriate equilibrium with the flow.

4.4.4 Absorption of CS Photons

At the end of each HD frame, MCRaT proceeds to absorb any photons that have a frequency $\nu \leq \nu_B$ of the HD cell that they are located within. As a result, all of the CS photons that were emitted and never upscattered become absorbed and any photons that have been scattered but still satisfy $\nu \leq \nu_B$ are also absorbed.

References

Depaola, G. 2003, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 512, 619

Ito, H., Nagataki, S., Matsumoto, J., et al. 2014, The Astrophysical Journal, 789, 159

Krawczynski, H. 2011, The Astrophysical Journal, 744, 30

Lazzati, D. 2016, The Astrophysical Journal, 829, 76

Lundman, C., Pe'er, A., & Ryde, F. 2014, Monthly Notices of the Royal Astronomical Society, 440, 3292

Mathews, K. 2013, Nuclear Science and Engineering, 173, 207

McMaster, W. H. 1961, Reviews of Modern Physics, 33, 8

Parsotan, T., & Lazzati, D. 2018, The Astrophysical Journal, 853, 8

Parsotan, T., López-Cámara, D., & Lazzati, D. 2018, The Astrophysical Journal, 869, 103

—. 2020, The Astrophysical Journal, 896, 139

Rybicki, G. B., & Lightman, A. P. 1979, Radiative processes in astrophysics