Draft version July 8, 2014
Preprint typeset using LaTeX style emulateapj v. 8/13/10

GENERAL RELATIVISTIC POLARIZED RADIATIVE TRANSFER CODE ASTRORAY. USER GUIDE

Roman V. Shcherbakov
Department of Astronomy, University of Maryland, College Park, MD 20742-2421, USA
Hubble Fellow

Draft version July 8, 2014

ABSTRACT

Version 1.0 of the code is publicly released. The code is provided under the terms of GNU Public License v. 3.0. No technical support is provided.

1. GENERAL IDEAS

The technique and the application of the code are described in Shcherbakov & Huang (2011) (later SH11) and Shcherbakov et al. (2012) (later SPM12), respectively. The code performs radiative transfer on top of three-dimensional general relativistic magneto hydrodynamic (3D GRMHD) simulations such as the ones described in Penna et al. (2010); Tchekhovskoy et al. (2011); McKinney et al. (2012); Narayan et al. (2012). Radiation is not expected to significantly alter the dynamical quantities of the so-called radiatively inefficient accretion flows (RIAFs) (Narayan et al. 1998; Quataert 2001). Therefore, one can perform the numerical simulations without cooling and then perform the radiative transfer in the post-processing.

The radiative transfer of polarized cyclo-synchrotron is performed. Version 1.0 only handles thermal distribution of electrons at the moment, but non-thermal particles can be readily introduced. The general radiative transfer equation in plain geometry is given by formula (17) in SH11, where emissivities $\varepsilon_{I,Q,V}$, absorptivities $\alpha_{I,Q,V}$, and rotativities $\rho_{Q,V}$ are given by formulas (21)-(25). Note, that the expression (20) for emissivities is valid for any particle distribution, while the expressions (22) and (23) for absorptivities and rotativities are only valid for thermal particles. Non-thermal rotativities are computed in Huang & Shcherbakov (2011). The correspondent *Mathematica 9* script can be found at `http://astroman.org/Faraday_conversion/`

The proper extension of radiative transfer to GR is described in SH11. We switch from the Stokes vector $\mathbf{S}$ to the vector of photon occupation numbers $\mathbf{N}$ (eq. (55) in SH11). Radiative transfer is conducted along the geodesic path of massless particles via the technique called "ray tracing". This technique is best applied, when the emission/absorption/rotation coefficients depend only on the local dynamical properties of the medium and not on the local properties of radiation field. It is possible to use the applied technique to Compton scattering as well, but then we need to compute and retain in computer memory the radiation field at each point. Ray tracing consists of solving for the photon geodesic from the picture plane/telescope/observer's plane in the direction of the black hole (BH). The geodesics either go back to infinity or fall onto the BH. Then the relevant equations on the evolution of Stokes occupation numbers are solved from the end points on the geodesics back to the observer's plane. The resultant polarized intensities on the plane constitute an image. Integrated over the observer's angle intensities give the total fluxes (Rybicki & Lightman 1979).

shcher@gmail.com

1.1. *Extension to large radius*

2. COMPILATION AND BASIC USAGE

2.1. *Compilation*

The code can be successfully compiled in GNU compiler "gcc", Intel Compiler "icc", and (with minor modifications) in Microsoft Visual Studio ("icl") The recommended software versions to ensure compilation and fast execution are: Visual Studio 2010, "gcc" version 4.6 or higher, "icc" version 11.0 or higher.

When submitting a job with *QSUB* on a supercomputer such as Deepthought at UMD or Nautilus at NICS the environment variable $PBS\_ARRAYID$ is read. It corresponds to the index of a batch job. Remember that some systems have a low limit on the number of concurrent jobs. The correspondent environment variable is $LSB\_JOBINDEX$ on *BSUB* system. The variable name should be double-checked on any supercomputer of interest. Whenever a code is run on a regular computer, the environment variable $PBS\_ARRAYID$ needs to be defined.

3. SETTING UP AND RUNNING THE CODE ON NAUTILUS

3.1. *Prerequisites/Tricks*

1. If one cannot login through Nautilus, then login through Kraken. They have a shared file system.

2. Nautilus often does not like it, when one logs in from a laptop and drops a connection after 1min of inactivity. To avoid that add "export TMOUT=86400" in ".bash_profile" file in home directory.

3. For compilation load intel/12.1.233 module with a command "module load intel/12.1.233". This compiler produces the fastest code - about 0.5% faster than GCC 4.6.3. Make sure no other intel or GCC compiler (apart from default GCC 4.3.4) is loaded by executing command "module list" and scanning its output. For automatic dealing with modules add "module unload intel/11.1.038" and "module load intel/12.1.233" lines to *.modules* file in the home directory, then relogin to Nautilus.

4. Check the size/file count quotas for home directory. They are typically found in documentation. Exceeding the disk space of all home directories brings Nautilus down.

3.2. *Directories and Files*

1. Create directory with any name (I use name *rt*) for radiative transfer code in home directory (*/nics/d/home/USERNAME/* on Nautilus). Create directory with any name (I use name *analysis*) for analysis in either the home directory or (better) in a work directory (*/lustre/medusa/USERNAME/* on Nautilus). Make sure the variable **dir** in *win_lin_Jon.c* points at analysis directory.

2. Copy the code w/ all files into *∼/rt/*. Copy 5 files with radiative transfer coefficients for thermal plasma to *analysis/* folder.

3. Create a directory *analysis/SIMULATION/*, where *SIMULATION* is a name of a particular GRMHD simulation of interest, i.e. "thickdisk7". Add this name as a part of **astr** array in *win_lin_Jon.c* file, add a correspondent spin value into **atab** array. One may need to correspondingly increase the sizes of those array. Make sure the values of **rlen**, **thlen**, and **phlen** correspond to radial, poloidal, and toroidal resolutions, respectively. The code will not complain about a problem, but a correspondent *Mathematica 9* script will.

4. Copy *dxdxp.dat* and *TsmapSIMULATION.dat* files into *analysis/SIMULATION/* directory. If one doesn't have the files, then one needs to generate them (see below).

5. Make sure that ALL *fieldlineXXXX.bin* files of interest exist in (or linked to) **adir**/$SIMULATION$/**fieldstr** directory. Those variables are defined in *win_lin_Jon.c* file. One may need to change **adir** and **fieldstr** variables to point to the correct location.

6. If one needs to compile with "gcc" and not with "icc", then copy *∼/rt/Makefile_gcc* into *∼/rt/Makefile*.

7. Compile the code with "make -B" command from *∼/rt/* directory. It should take under 30 seconds.

8. Make sure that every directory in the tree and every file has a group read permission.

3.3. *Job submission and execution*

1. Each supercomputer has its job submission system (BSUB/QSUB/other) and its own features/bugs. Always submit jobs through the submission system and not directly. One cannot directly copy the submission files from one system to another. Always RTFM first. Erroneous submission may bring supercomputer down.

2. The examples of QSUB scripts for Nautilus are in *linux_Nautilus* in GIT repository. The examples of BSUB scripts are in *linux_Odyssey* in GIT repository.

3. Set **nthreads** in *win_lin_Jon.c* to the maximum number of OpenMP threads the system supports /or/ on shared system machines set it to a smaller number, for which the code runs efficiently. **nthreads** = 16 is typically a safe bet.

4. Make sure the number of CPUs in the batch job file is equal to **nthreads** as **ncpus** = 16(= **nthreads**).

5. Make sure there is enough memory in the system. The code needs about $2 * \mathbf{fdiff} * (fieldlinesize) +$ 4 GB of shared RAM, where the size of *fieldlineXXXX.bin* file is used.

6. Translate the memory requirement into a line in the job submission file.

7. For each change in "c" or "cpp" files run "make -B". Always check that *∼/rt/transfer_quad* executable has a later modification time than any other code file.

8. For QSUB system take care of the batch variable by submitting as "qsub -t H G SCRIPTNAME", where **H** is the first index for $PBS\_ARRAYID$ and **G** is the last index. For provided Nautilus scripts the submission lines are the last (commented) lines of the scripts.

9. For a quick test one might submit *quall2.csh* as "qsub -t 30-30 quall2.csh". A quick test typically starts running immediately after submission and

10. Make sure to not exceed the suggested system I/O bandwidth. Limit the number of concurrent jobs with heavy I/O (especially in *m_sear* mode). Too much I/O will slow down the supercomputer and may bring it down.

11. For some time after submission keep checking the job status with, e.g. "showq -n". If the job starts running or disappears, then check the outputs.

12. First outputs to check are the "error" and "output" files in *∼/rt/* directory. Communication errors typically can be ignored, segmentation faults cannot. If one didn't modify the code, then a segmentation fault means that some input files are missing/have wrong names/are in wrong places. Other type of mistakes are typically caught on a debug stage.

13. If the code actually works (the "error" and "output" files have zero sizes for at least a couple of minutes), then check the output in *analysis/* folder. *m_quick* mode generates first output within a minute, while *m_sear* mode generates first output after 1hr. Read the first output files to check that they look ok and make sure the code doesn't need to be restarted. That saves tons of one's time and computational time.

3.4. *Other*

1. Define **rgrav** in *transfer_quad.cpp* and the proportionality coefficient for the final flux in *intensity.cpp* to correspond to the physical problem at hand. Make sure **rgrav** is self-consistently used as either M or $2M$. At present **rgrav** $= 2GM/c^2$.

2. Copy one of the late-time *fieldlineXXXX.bin* files on a laptop/desktop. Copy *gdump.bin* file from **adir** + $SIMULATION$ + **fieldstr** to laptop/desktop.

3. Run a *Mathematica 9* script on those files to create *dxdxp.dat* file. Run a series of *Mathematica 9* tests to see that we understand the structure of *fieldlineXXXX.bin* file and that everything is consistent.

4. Run radiative transfer code in *m_ts* mode by first creating a correspondent batch job. The output is a file of average temperature/density profiles *analysis/SIMULATION/TsmapSIMULATIONx.dat*.

5. Rename that file into *analysis/SIMULATION/TsmapSIMULATION.dat* file. This extra step is implemented to avoid accidentally overwriting the file of averaged profiles.

6. Figure out the frequencies, for which we want to compute the radiative transfer. Figure out the fluxes at these frequencies. Enter everything into **sftab** variable in *transfer_quad.cpp*. While a set of frequencies is obligatory, the fluxes are not obligatory unless a model-fitting routine is invoked.

7. Come up with a fiducial set of parameters **rhonor**, **heat**, **th** and run radiative transfer code in *m_quick* mode. One needs to set **fdiff** $=$ **0** and define **thlimit** and **isBcut** variables. Also set **fmin**, **fmax**, and **ind** to some values. Recompile the code each time the parameters are changed.

8. Run *Mathematica 9* script to plot the lightcurve and perform any other analysis one needs.

9. Try to manually change three parameters for a quick computation in order to produce a spectrum, which resembles the observed spectrum.

10. Define the χ^2 computation and a set of residuals. Run the code in *m_sear* mode and find a local minimum. Recompile the code after any parameters were changed. A simplified version of *m_sear* is implemented at present: it might not find a global minimum, especially if the polarized quantities are fitted for. A full version would include running *m_sear* for a set of fixed inclination angles and searching in space of **rhonor** and **heat** only. It is recommended that **ind** $= 21$ is set for this step.

11. Run the code in *m_imag* mode and create image files. Run the correspondent *Mathematica 9* script to visualize the images.

12. Run the code in *m_quick* for maximum **ind**, run correspondent *Mathematica 9* script and enjoy the full variable lightcurve.

13. Run the code in *m_imag* for maximum **ind**, run correspondent *Mathematica 9* script to create all images. Use, e.g. *JPGvideo* program to make movies. Upload movies to Youtube.

4. INPUTS

The code has several inputs and several modes of operation.

4.1. *GRMHD simulation snapshots*

The files *fieldlineXXXX.bin* are the simulation snapshots. They contain the full 3D dynamical quantities: arbitrarily normalized density ρ, internal energy density u, 4-velocity u^ν, 3-vector of lab-frame magnetic field B_i (see Penna et al. 2010), and two more quantities, which we do not need. All quantities are 4-byte real numbers The simulations are performed on a distorted spherical grid, modified Kerr-Schild (MKS) coordinates. The effective dimensions are **rlen**, **thlen**, and **phlen** in radial, poloidal, and toroidal directions, respectively. *fieldlineXXXX.bin* files contain the preamble of the length which changed over the years. Thus, start reading the file from the position

$$p = filesize - rlen * thlen * phlen * 11 * 4. \quad (1)$$

The velocity of matter approaches the speed of light near the event horizon. Therefore, as the photons propagate within the flow, the flow itself has enough time to change. The information about the flow at that later time is stored in a *fieldlineYYYY.bin* file with a different $YYYY > XXXX$ number. The emissivity etc. at that later time is determined by quantities in *fieldlineYYYY.bin*. The code allows to read several *fieldlineXXXX.bin* files from the storage and perform self-consistent radiative transfer calculations, where the numerical simulation effectively evolves as the light propagates through the flow.

4.2. *Coordinates and transformations*

Since the simulation grid is not spherical, the distortions need to be quantified. First, we should know the coordinates of grid points. Second, we need to know the Jacobian matrix of transformation between the distorted coordinates and the unmodified Kerr-Schild (KS) coordinates. The metric tensor in KS system is analytic and is computed directly within the code. Both coordinates and Jacobian matrices are parts of *gdump.bin* file. However, I don't want to read that file (or even a part of it) within the code, since the format of that file may also change with time. Instead, I read 1/**phlen**'s part of *gdump.bin* in *Mathematica 9* , then extract the coordinates and the Jacobian matrix at each point. Both are written into *dxdxp.dat* file, which has a small size. The *Mathematica 9* code to produce *dxdxp.dat* from *gdump.bin* is the auxiliary part of the full code.

The transformation of coordinates is the correspondence of the consecutive number of a grid cell $(nr, n\theta, n\phi)$ to "real" MKS coordinates (r, θ, ϕ). The transformation of $n\phi$ to ϕ is strictly linear. The transformation of nr to

r is close to exponential. Coordinate θ in the function of both nr and $n\theta$ in the latest versions of Jon's HARM code. That latter dependence is highly sophisticated and hard to invert, which makes us create *dxdxp.dat* binary file as opposed to analytic computations of coordinates and the transformation matrices.

4.3. *Average density and temperature profiles*

The numerical simulations output the internal energy density at each point. However, those are the electrons, which radiate. The electron temperature T_e is expected to be much lower than the proton temperature T_p close to the BH (Narayan & Yi 1995). The Coulomb collisions or the plasma processes might not be strong enough to equilibrate the temperatures, and the two-temperature flow arises. Electrons can be cooler, because they can cool (Drappeau et al. 2012), because their heat capacity is much higher at relativistic temperatures (Shcherbakov & Baganoff 2010), because viscosity is expected to mainly heat the protons (Narayan & Yi 1995; Sharma et al. 2007).

It is not really possible to compute the temperature of electrons from the first principles, or prove that the distribution is thermal. Some researchers set the electron temperature to be a constant fraction of proton temperature $T_p/T_e = \text{const}$ (Mościbrodzka et al. 2009; Dexter et al. 2010). However, I employ a more sophisticated technique. I follow Sharma et al. (2007) and compute T_e starting from the outer simulation boundary some $10^5 r_{\rm g}$ away from the center. Such computation includes the Coulomb collisions, relativistic heat capacity of electrons, redistribution of viscous energy dissipation between species $Q_e/Q_i = C(T_e/T_i)^{0.5}$, where C is a constant. The code computes correspondence of T_e on u in the equatorial plane over the averaged profiles of density and temperature. Then it draws T_e from that correspondence to u at any point of the simulation (see section 4.2 of SPM12). Those averaged radial profiles of density and temperature are created by *m_ts* routine (see below) and stored in *TsmapNAME.dat* text file, where *NAME* is the simulation name.

4.4. *Cyclo-synchrotron emissivities and rotativities*

A crucial part of GR polarized radiative transfer is the correct absorptivities/emissivities/rotativities. Those are computed by a separate *Mathematica 9* script. Those coefficients are typically function of three parameters: electron temperature $\theta_e = k_b T_e/(m_e c^2)$ (or the whole particle distribution), angle between the magnetic field line and the direction of $\mathbf{k}$ vector θ_{kB}, and the ratio of cyclotron frequency to observed frequency ν_B/ν. The assumed momentum distribution of particles is isotropic. However, the number of parameters can be reduced in certain approximations down to two for emissivity/absorptivity, and down to one for rotativities (Faraday rotation and conversion) (see SPM12). The relevant text files are two-parametric *lookupjIy.dat* file for total emissivity, *lookupjQy.dat* file for linearly polarized emissivity, *lookupjVy.dat* for circularly polarized emissivity, *lookupjQa.dat* for Faraday conversion coefficient, and *lookupjVa.dat* for Faraday rotation coefficient. It is redundant to have two dimensions for rotativities, but this is done for the sake of uniformity.

5. MODES OF OPERATION

5.1. *Usage*

The usage is *transfer_quad A B C D* with defined $PBS\_ARRAYID$. The parameter A chooses a spin of the flow or chooses one of the predefined models. $A-1$ is the index in **atab**, **astr**, and **ncuttab** arrays defined in *win_lin_NAME.c* files. The fourth parameter D chooses the operation mode (the mode is sometimes chosen in *transfed_quad* routine itself for debugging purposes). Parameter B means the number of separate points of time, over which the computation of image/intensity etc. are performed in all modes except *m_surf*. It coincides with the number of simulation snapshots for **fdiff** $= 0$. In mode *m_surf* the parameter B switched between surfing different pairs of parameters: $B = 1 \Rightarrow$ **tth** *and* **rhonor**, $B = 2 \Rightarrow$ **rhonor** *and* **heat**, $B = 3 \Rightarrow$ **heat** *and* **tth**. Parameter C means different things for different D.

The radiative transfer code has seven modes of operation at present. They serve the tasks of computing the auxiliary quantities, computing the spectrum for a single set of flow parameters, exploring the full parameters space, searching for a local minimum of χ^2/dof in the parameter space, thoroughly surfing the region of parameter space, creating a flow image, performing convergence studies. The flow parameters are the spin a_*, the inclination angle θ, the accretion rate $\dot{M}$, the heating constant C or the ratio of temperatures T_i/T_e at certain radius. For each set of parameters we compute the spectrum/image at several times correspondent to times of simulation snapshots with numbers from **fmin** to **fmax** with a step **sep**. For example, **fmin** $= 6950$, **fmax** $= 9950$, and **sep** $= 150$ were used in Shcherbakov et al. (2012) to compute the spectrum at 21 points of time. Parameter **fdiff** determines in which range of times Δt around the simulation snapshot times t_0 the simulation is taken to evolve as the rays propagate. For example, **fdiff** $= 60$ means that snapshots *fieldlineXXXX.bin* with numbers from $XXXX_0 - fdiff$ to $XXXX_0 + fdiff$ are employed to compute the spectrum at a time correspondent to $XXXX_0$. The value **fdiff** $= 0$ corresponds to "fast light propagation", i.e. the simulation is not evolved as the light propagates along geodesics. In such case at each time the spectrum/image is simulated for a single simulation snapshot. Large values of **fdiff** may not work because of memory limitations. Some time ago all simulated spectra used to be computed based on single snapshots/averaged model. The routines for such computations were called *s_name*. Now the routines are called *m_name* to indicate the computations are always performed on multiple snapshots/at multiple points of time. The range of frequencies for the radiative transfer can be chosen in any routine with variables **kmin** and **kmax**.

The "main" function is contained in *transfer_quad* file, whose primary goal is to read the command line parameters and decide on the mode of operation. That file also defines global variables, and a couple of core functions, which are to be discussed below.

5.2. *m_space*

This routine allows for a given spin **sp** $\sim a_*$ and inclination angle **th** $= \theta$ to surf the parameter space of

heating constant **heat** $= C$. For each value of **heat** we search for the value of density normalization **rhonor** (which is proportional to the accretion rate $\dot{M}$), which provides the best χ^2 fit to the data. Then we explore the values of **rhonor** near that best value. The values of heating constant **heat** roll from 0.75 down to 0.15. The routine is run for a range of inclination angles **th**. Such technique allows for thorough exploration of the entire parameter space to make sure no local minimum is missed. Looking at **rhonor** values near the minimum allows to immediately estimate the confidence interval, e.g., based on $\Delta\chi^2$ technique. Search for the best combination of parameters and the estimate of the confidence interval are done by a separate *Mathematica 9* scripts.

5.3. *m_quick*

A simple computation of a spectrum for a single set of parameters **sp**, **th**, **rhonor**, **heat**, **fdiff** within the range from **fmin** to **fmax**. The result is written into *quickaSTRING.dat* file. The routine is convenient to use for debugging the code.

5.4. *m_conv*

A range of convergence tests/tests on sensitivity to auxiliary flow parameters is conducted by this routine. The descriptions and ideas behind such tests can be found in the Table 3 and the Appendix of Shcherbakov et al. (2012). The sample spectrum is computed first similar to *m_quick* routine. Then one parameter is changed and the spectrum is recomputed. Then the change in reduced chi-squared, χ^2_H/dof defined by formula (A1), is computed the quantify the changes in the spectrum. In the present version tests (1) and (2) explores the number of geodesics given by $\mathbf{snxy}^2$ to compute the spectrum. Tests (3) and (4) check how far from the BH the radiative transfer should start on geodesics, which go through the horizon. Tests (5) and (6) test the size of the emitting region. Tests (7) to (9) explore the changes in the slopes of extensions of quantities to large radius (changes in magnetic field slope might not be consistently implemented at present). Tests (10) and (11) explore the number of spectra computed over a long period of time to represent the true mean spectrum over that period of time. Tests (12) to (14) explore for how long one needs to have the simulation evolve simultaneously with the propagation of light rays. Tests (15) to (38) explore the sensitivity to small changes in main parameters **heat**, **rhonor**, and **th**. These latter tests are useful to figure out the test steps in **heat**, **rhonor**, and **th** to use in minimization routines.

5.5. *m_surf*

Explores the rectangular grid in any two out of three parameters **heat**, **rhonor**, and **th**. Computation is useful to show the contours of $\Delta\chi^2$ and illustrate the confidence intervals.

5.6. *m_imag*

This routine is very similar to *m_quick*. However, instead of computing the spectrum, we compute the image (2D table) of Stokes parameters in the picture plane. A separate *Mathematica 9* routine is used to draw the actual images. The same routine can technically compute the image and the spectrum. However, the resolution is higher for an image, which makes the calculation slower. Thus, I use *m_quick* to quickly compute the spectra over many frequencies and *m_imag* to compute the images over a few frequencies.

5.7. *m_sear*

This routine searches for a local minimum with the help of a steepest descent method. First, I set up some initial conditions, which does not have to be provide a fit. Then I explore the initial conditions, where one of the parameters is changed, i.e., **heat** changed to **heat**$(1 +$ **dheat**) etc. The steepest descent methods searches for a local minimum until the algorithm converges (as given by **ddh**, **ddr** and **dth** variables), or until 20 iterations are performed. The algorithm is known to be stuck in a limit cycle at low **heat** values, thus the limit on the number of iterations is important.

5.8. *m_ts*

This routine reads multiple simulation snapshots and finds the average of density and internal energy density around the equatorial plane in θ, over all ϕ, for each radius r. The resultant profiles are sorted to be homogeneous, which is needed to define a single-valued $Te(u)$ function. The sorted radial profile is written into *TsmapxSTRING.dat* file.

6. CORE ROUTINES

In this section we describe the routines and functions, which define and perform the computations. All equations are derived in separate *Mathematica 9* scripts, which are to be shared as well.

6.1. *init*

6.2. *intensity*

Performs the computation of intensity along a pre-computed set of geodesics.

6.3. *imaging*

This routine is very similar to *intensity*. Performs the computation of intensity along a pre-computed set of geodesics.

6.4. *evalpointzero*

Main routine

6.5. *solvetrans*

Frond-end routine for solving radiative transfer equations

6.6. *transnew*

Radiative transfer equations in polarized polar coordinates

6.7. *geoint*

Frond-end routine for solving for the geodesics

6.8. *geodes*

Equations, which define the geodesic line as well as two parallel-propagated vectors.

6.9. *emis*

Look-up of emissivity/Faraday rotation & conversion coefficients on 2D grid.

6.10. *Function solte in transfer_quad*

The evolution equations on proton and electron temperatures. The equations are solved within *init* function.

7. CODE VARIABLES

8. SUPPORTING MATHEMATICA 8 SCRIPTS

A lot of scripts are to be grouped into a single structured file *check_GRMHD_code.nb* in *math* directory on GIT. It consists of variety of scripts, which compute auxiliary quantities, derive some parts of ASTRORAY code to be directly copied into C++, and check GRMHD code. **P**lease, run "Cell-¿Delete all output" in *Mathematica 9* before committing the file to GIT. The data processing and QPO computations are stored in separate *Mathematica 9* scripts to be shared later.

8.1. *check_GRMHD_code.nb*

8.1.1. *Emis functions*

The script "Computing emissivities/rotativities" does what its title says. The precise relativistic thermal electron distribution is used in all computations. It is easy to explicitly change the distribution to compute emissivities. However, if a new distribution is non-thermal, then the absorptivities need to be computed separately, stored as another set of tables and read by the code.

The Faraday rotation and conversion for thermal distribution is computed in my paper (Shcherbakov 2008). It is somewhat harder to compute Faraday rotation and conversion for non-thermal particle distribution. One needs to follow my other paper Huang & Shcherbakov (2011). The code from that paper is available online.

A perfect future particle distribution is the relativistic Lorentzian distribution (Shcherbakov 2009)

$$f(p) \propto \left(1+\frac{\sqrt{1+p^2}-1}{\kappa T}\right)^{-\kappa+1}. \quad (2)$$

In the limit $\kappa->\infty$ the distribution becomes the thermal distribution with temperature $T = k_B T_e/(m_e c^2)$. For small κ the distribution is the power-law with a flexible amount of low-energy particles, controlled by the parameter T. Thus, a single distribution can be used everywhere! One does not have to artificially merge the thermal and non-thermal distributions and increase the number of free parameters.

8.1.2. *Evalpointzero functions*

A particularly useful script is "Checking gdump/fieldline, making dxdxp -¿ for Jon's jet simulations with QPOs (i.e. thickdisk7)". This script reads *gdump.bin* and a single *fieldlineXXXX.bin* files from a specified place on disk. The script performs a number of tests to ensure the format of each file is properly decoded and that the encoded data are correctly understood. Then the script computes *dxdxp.dat* file, which is required for the ASTRORAY to work.

The script in "Limiting the polar region" checks how bad the polar region looks: I make various ContourPlots.

REFERENCES

????
08. 1
Dexter, J., Agol, E., Fragile, P. C., & McKinney, J. C. 2010, ApJ, 717, 1092
Drappeau, S., Dibi, S., Dexter, J., Markoff, S., & Fragile, P. C. 2012, ArXiv e-prints
Huang, L., & Shcherbakov, R. V. 2011, MNRAS, 416, 2574
McKinney, J. C., Tchekhovskoy, A., & Blandford, R. D. 2012, MNRAS, 423, 3083
Mościbrodzka, M., Gammie, C. F., Dolence, J. C., Shiokawa, H., & Leung, P. K. 2009, ApJ, 706, 497
Narayan, R., Mahadevan, R., & Quataert, E. 1998, in Theory of Black Hole Accretion Disks, ed. M. A. Abramowicz, G. Bjornsson, & J. E. Pringle, 148
Narayan, R., Sadowski, A., Penna, R. F., & Kulkarni, A. K. 2012, ArXiv e-prints
Narayan, R., & Yi, I. 1995, ApJ, 452, 710
Penna, R. F., McKinney, J. C., Narayan, R., Tchekhovskoy, A., Shafee, R., & McClintock, J. E. 2010, MNRAS, 408, 752
Quataert, E. 2001, in Astronomical Society of the Pacific Conference Series, Vol. 224, Probing the Physics of Active Galactic Nuclei, ed. B. M. Peterson, R. W. Pogge, & R. S. Polidan, 71
Rybicki, G. B., & Lightman, A. P. 1979, Radiative Processes in Astrophysics (New York: Wiley)
Sharma, P., Quataert, E., Hammett, G. W., & Stone, J. M. 2007, ApJ, 667, 714
Shcherbakov, R. V. 2008, ApJ, 688, 695
—. 2009, Physics of Plasmas, 16, 032104
Shcherbakov, R. V., & Baganoff, F. K. 2010, ApJ, 716, 504
Shcherbakov, R. V., & Huang, L. 2011, MNRAS, 410, 1052
Shcherbakov, R. V., Penna, R. F., & McKinney, J. C. 2012, ApJ, 755, 133
Tchekhovskoy, A., Narayan, R., & McKinney, J. C. 2011, MNRAS, 418, L79