Cosmic Rays Self Propagation Escaping and Confinement TRAnsport code (CR SPECTRA)

Loann Brahimi

February 11, 2020

Contents

1 Introduction & Quick tutorial

1.1 Presentation of the CR SPECTRA code

1.2 Structure of the code

The ./CR_SPECTRA/ code is structured in the following way. In the ./CR_SPECTRA/ folder, we have two Python3.7 files : "namelist.py" and "setup.py" which allow to generate the initial conditions of the problem such as : the ISM model, the CRs initial pressure and turbulent background. These files make use of the folder "tools" which contains all the stuff in term of mathematics and physical models in order to generate the initial conditions. The ./CR_SPECTRA/src folder contains the C++ core of the code which allows to solve numerically the propagation of CRs and the turbulence it generate according to the initial ISM conditions. The "./CR_SPECTRA/data_analysis" folder contains some files which allows to read the outputs of the code.

1.2.1 The input files

The input files contains the files **./CR_SPECTRA/namelist.py** and **./CR_SPECTRA/setup.py** which allow to generate the initial coniditions of the simulation and produce the initial data in the work folder. These data are located in the folder **WorkFolder/data_ini** and are named : **Alfven.dat** (which corresponds to the Alfven velocity in space and energy), **B.dat** which corresponds to the background magnetic field in space, **damping.dat** which corresponds to the self-generated turbulence damping rate in space and energy, **DBohm.dat** which correspond to the Bohm diffusion coefficient in space and energy, **DCRpara.dat** which correspond to the background diffusion coefficient of CRs in space and energy, **E.dat** which correspond to the energy scale in ergs, **Im.dat** which corresponds to the backward Alfvén perturbations energy density in space and energy, **Ip.dat** which corresponds to the forward Alfvén perturbations energy density in space and energy, **Pcr.dat** which corresponds to the protons pressure in space and energy, **Pe.dat** which corresponds to the electrons pressure in space and energy and **X.dat** which corresponds to the space scale grid in cm.

A **./WorkFolder/parameters.dat** file is also generated and put in the work-folder. It contains other important informations for the simulation.

1.2.2 The calculation files

All the calculation files can be found in the folder **./CR_SPECTRA/src**, they play the following roles

- **main.cpp** : This file is the main file of the simulation which contains the time-loop.
- **constants.h** : This file contains all the important parameters of the simulation such as the CRs injection model, solvers properties, output parameters ...
- **cr_source.h** : This file contains all the CRs injection models functions
- **solver1D.h** : This file contains all the solver functions of the simulation
- **tools.h & mathematics.h** : These files contains important mathematical functions used in the simulation
- **fwritter.h & freader.h** : These files allow to read/write data
- **out.h** : This file allows to write the ouput data
- **logmaker.h** : This file allows to create the log output of the code to show during the simulation

1.2.3 The output files

During the simulation, the code will print outputs. They will be printed in the folder **./WorkFolder/data_out/**. And they are of the form :

VariableName_XXXXX.dat

where XXXXX correspond to the number of ouput.

1.2.4 The tools files

1.3 The initial conditions python files

The initial conditions files are located in the main folder : "./CR_SPECTRA/" and are named as following :

→ namelist.py

→ setup.py

They need to be executed with Python3.7 using the terminal command

```
python ./setup.py
```

Let is describe both files. First, we start with the "namelist.py" file

namelist.py This file contains all the variables allowing to basically structure the simulation. It is divided in different part : the **Output folder creator** part which allows to create the output folder and contains the following variables

- **folder_name** : Name of the current work folder.
- **folder_path** : The relative (or absolute) path of the work folder on your computer.

The code will create the folder a the location you choosed and in this folder it will create sub-folders "data_out", "data_ini", "data_analysis" and "tools" in order to put the initial conditions and read the outputs of the simulation. The **Grid parameters** part leads the size of the simulation grid and contains the following variables

- **NX, NE** : Correspond to the number of bins of space and energy respectively in terms of power of two. For example, NX = 12 corresponds to $2^{12} = 4096$ bins.
- **Xmin, Xmax** : Correspond to the minimum (maximum) value of the space grid bin.
- **xgridtype** : Corresponds to the type of the grid, cartesian is the only choice for instance.
- **Emin, Emax** : Correspond to the minimum (maximum) value of the energy grid bin.
- **egridtype** : Corresponds to the type of the grid, logspace is the only choice for instance.
- **box_center** : Corresponds to the space position center of the CRs source according to the **Xmin** and **Xmax** values. Example, if Xmin = 200.*cst.pc and Xmax = 2200.*cst.pc, if box_center = 100.*cst.pc, the CRs source will be outside the simulation box which will lead to an error in the code.

All these variables allow to build a space and energy grid according to the **grid** function in the file **./CR_SPECTRA/tools/d1_grid_generator.py**. The **Other terms** part allows to select or unselect some turbulent damping parameters **in_damping** (Ion-Neutral damping), **lz_damping** (Lazarian damping, see Lazarian 2016) and **nlld_damping** (Non-linear Landau damping, see Wiener et al. 2013). The variables **Pcr_1GeV** and **Pe_1GeV** allow to fix the Protons and electrons pressure at 1 GeV (in CGS units). The **ISM Structure** part allows to generate the ISM model as a combinaison of phase layers listed in the variable **phase**. The layers are defined as a list of the following variables :

- **ism.layer_phase_name** : This variable can be (HII, WIM, WNM, CNM, DiM ...) and correspond to a python dictionnary of phase defined in the file **./CR_SPECTRA/tools/phases_collection.py**. You can add your own phases with their own parameters.

- **dict{Xmin = a, Xmax = b}** : Corresponds to a Python dictionnary describing the minimum position and maximum position of the given phase in the simulation box. You need to fill all the box with a phase, otherwise the code will lead to an error.

- **getVAE, ism.layer_phase_name** : Except the name to the layer, you don't need to change this variable. It allows to obtain a smoother value of the Alfvén velocity between the different phases. Indeed, without this function, one can observe strong jumps in the Alfvén velocity between two phases which can lead to a numerical error.

Finally, the variable **smooth_width_transition** allows to fix the transition size between the different layers in order to avoid jumps. It is suggested to avoid values lower than 10 pc for a spatial grid of 2048 bins.

setup.py The **setup.py** file allows to generate the initial CRs and turbulence conditions and write the input files in the chosen work folder. It is divided in different part such as **Informations coming from the namelist** which retrieve all the data coming from the file **namelist.py** in the same folder, the **Initial ISM conditions** which you can adapt with your own models and which is basically generate the following variables

- **d00** [list, X] : Corresponds to a background value of the CRs diffusion coefficient. This value is only space dependent and can be refined according to a model of turbulence in the different ISM phases.

- **D, Db** [list, (E, X)]: Corresponds to the value of the CRs diffusion coefficient (Bohm diffusion coefficient).

- **Ip, Im** [list, (E, X)] : Corresponds to the value of the magnetic perturbations energy density of foward (backward) Alfvén waves ($I = (\delta B/B_0)^2$).

- **VA** [list, (E, X)] : Corresponds to the vectorial Alfvén velocity

- **gamma_in, gamma_lazarian, gamma_nlld, gamma_tot** [list, (E, X)] : Corresponds to the damping terms in space and energy of the generated resonnant Alfvén waves (Im and Ip) and $\Gamma_{\rm tot} = \Gamma_{\rm in} + \Gamma_{\rm lz} + \Gamma_{\rm nlld}$.

- **Pcr, Pe** [list, (E, X)] : Correspond to the protons and electrons pressure.

This part contains two important double loops. The first one allow to calculate the turbulent background while the second one allows to calculate the damping terms and the initial protons and electrons space and energy distribution (only of the background, not the source). The **Write the initial conditions** and **Copy tools and analysis files** allow to put in the work folder all the important files for the simulation.

1.4 The initial condition C++ file

The inital condition C++ file are found in the folder **CR_SPECTRA/src/**. It is named **constants.h** and allows to fix all the important parameters of the simulation such as : the parallelisation of the code, the equations terms to use, the CRs injection parameters (CR Source) and the data output frequency and the simulation duration. The file is decompsed in different parameters : The first part is the constant values of the simulation which can be refined if necessary of even changed into an SI metric system. The second is the **Solver options**. Let remind quickly the equations solved by the CR_SPECTRA code

$$\underbrace{\frac{\partial P_{\rm cr}}{\partial t}}_{\rm CR.0}+\underbrace{V_{\rm A}\frac{\partial P_{\rm cr}}{\partial x}}_{\rm CR.1} = \underbrace{\frac{\partial}{\partial x}D\frac{\partial P_{\rm cr}}{\partial x}}_{\rm CR.2}+\underbrace{\frac{E}{3}\frac{\partial V_{\rm A}}{\partial x}\frac{\partial P_{\rm cr}}{\partial E}}_{\rm CR.3}-\underbrace{\frac{E}{3}\frac{\partial V_{\rm A}}{\partial E}\frac{\partial P_{\rm cr}}{\partial x}}_{\rm CR.4}-\underbrace{\frac{4}{3}\frac{\partial V_{\rm A}}{\partial x}P_{\rm cr}}_{\rm CR.5}+\underbrace{Q_{\rm cr}}_{\rm CR.6}$$

$$\underbrace{\frac{\partial P_{\rm e}}{\partial t}}_{\rm E.0}+\underbrace{V_{\rm A}\frac{\partial P_{\rm e}}{\partial x}}_{\rm E.1}=\underbrace{\frac{\partial}{\partial x}D\frac{\partial P_{\rm e}}{\partial x}}_{\rm E.2}+\underbrace{\frac{E}{3}\frac{\partial V_{\rm A}}{\partial x}\frac{\partial P_{\rm e}}{\partial E}}_{\rm E.3}-\underbrace{\frac{E}{3}\frac{\partial V_{\rm A}}{\partial E}\frac{\partial P_{\rm e}}{\partial x}}_{\rm E.4}-\underbrace{\frac{4}{3}\frac{\partial V_{\rm A}}{\partial x}P_{\rm e}}_{\rm E.5}+\underbrace{Q_{\rm e}}_{\rm E.6}+\underbrace{Q_{\rm sync}}_{\rm E.7}$$

$$\underbrace{\frac{\partial I_p}{\partial t}}_{\rm IP.0}+\underbrace{V_{\rm A}\frac{\partial I_p}{\partial x}}_{\rm IP.1}=\underbrace{-I_p\frac{\partial V_{\rm A}}{\partial x}}_{\rm IP.2}+\underbrace{(\Gamma_g-\Gamma_d)(I_p-I_p^0)}_{\rm IP.3}$$

$$\underbrace{\frac{\partial I_m}{\partial t}}_{\rm IM.0}+\underbrace{V_{\rm A}\frac{\partial I_m}{\partial x}}_{\rm IM.1}=\underbrace{-I_m\frac{\partial V_{\rm A}}{\partial x}}_{\rm IM.2}+\underbrace{(\Gamma_g-\Gamma_d)(I_m-I_m^0)}_{\rm IM.3}$$

In the **constants.h** file, each term relies to the following parameters whose their value can be 1 (for on) and 0 (for off). Here P refers to the Protons pressure (Pcr) and also the electrons pressure (Pe), I refers to the fowarding (Ip) and backwarding (Im) Alfvén turbulence.

- **solver_PAdvection** [CR.1 & E.1] : Explicit

- **solver_PDiffusion** [CR.2 & E.2]: Fully Implicit
- **solver_PAdvection2** [CR.4 & E.4]: Explicit
- **solver_PAdvectionE** [CR.3 & E.3]: Explicit
- **solver_PSource1** [CR.5 & E.5]: Source
- **solver_PSource2** [CR.6 & E.6]: Source
- **solver_PeAdvectionE2** [E.7] : Explicit
- **solver_IAdvection** [IP.1 & IM.1]: Explicit
- **solver_ISource1** [IP.2 & IM.2]: Source
- **solver_IDampGrowth** [IP.3 & IM.3]: Source

The part **Run & Output parameters** provides th following parameters

- **nproc** [integer] : Number of processors if OpenMP is activated. Otherwise, we recommend to let it equals to 0.
- **t_data_out_min** [double] : Instant - dt of the first output of the code
- **t_data_out_max** [double] : Instant of the last output of the code
- **number_out_data** [int] : Number of output data we want between **t_data_out_min** and **t_data_out_max**
- **time_distrib_of_data** [int] (0: linspace, 1: logspace) : Corresponds to the way (in terms of time) we want to output the simulation data
- **log_first_data** [double] : Parameters allowing to choose the first time of data ouput in the case where **time_distrib_of_data** = 1 because t = 0 does not exist in a such case.
- **delta_log_output** [int] : This parameter specifies the number of time-step of the code between each log output.
- **Tmax** [double] : Corresponds to the maximum time of the simulation.

The part **SNR Properties** deals with the CRs injection models. The parameters are the following

- **Esn** [double] : Corresponds to the total energy released from the SNR (in unit of 1e51 erg)

- **Mej** [double] : Corresponds to the total mass released by the SNR in Sun mass units
- **xi_n** [double] : Parameter = 1 for solar abundance metallicity
- **phi_c** [double] : Actual thermal conductivity, the Sptitzer (1962) value
- **bbeta** [double] :
- **C06** [double] :
- **xhi_cr** [double] : Efficiency of CRs acceleration (0.1 by default)
- **xhi_0** [double] :
- **gam** [double] : CRs injection energy power law index (2.2 by default)
- **Emin** [double] : Minimum value of energy of accelerated CRs
- **delta** [double] : Free parameter for the CRs escape time calculation (from Celli et al. 2019)
- **t_start_injection** [double] : Time start CRs injection function (10^{-6} kyrs)
- **t_end_injection** [double] : in tesc[E] units, Time end CRs injection function (number of tesc)
- **injection_function_width** [double] : Allows to calculate the Half-width value of the CRs time injection function : $\sigma = t_{\rm esc}/\text{width}$
- **injection_function_norm** [integer] : Number of bins for the numerical integral in order to normalize the injection function
- **r_snr_thickness** [double] : = R_SNR(t)/by the value you chose, it allows to smooth the injection shape of CRs

2 Models & Theories

2.1 The interstellar medium

2.2 The CRs injection and propagation

3 Numerical methods

4 Examples of applications