User guide (1D LRNe model)

This is a guide to intall and use guangqi. You will be able to reproduce a light curve that resemble AT2019zhd by following this guid. If you find anything suspicious, please let me know.

Installation

Prerequisite: Fortran, Lapack, parallel HDF5 (v1.14.2), MPI (openmpi-4.1.6), and Petsc (v3.19.4)

On Ubuntu, fortran is a part of the GNU compiler. Install gcc to get gfortran.

The zeroth step is to make sure that your computer has gfortran, gcc compilers, and lapack. Download lapack here https://netlib.org/lapack/ and compile to get liblapack.a and librefblas.a. Copy them to your /usr/local/lib

The first step is to install Openmpi https://www.open-mpi.org/

The configuration is:

./configure --prefix=/usr/local/openmpi

The second step is to install the parallel HDF5
https://www.hdfgroup.org/downloads/hdf5/source-code/
The configuration is:

CC=/usr/local/openmpi/bin/mpicc FC=/usr/local/openmpi/bin/mpif90 ./configure --enable-parallel --enable-fortran --prefix=/usr/local/hdf5

Petsc is here https://petsc.org/release/overview/

./configure --prefix=/home/(yourusername)/petsc --with-blas-lib=/usr/local/lib/librefblas.a --with-lapack-lib=/usr/local/lib/liblapack.a --with-mpi-dir=/usr/local/openmpi

Setup the run directory

After getting the code, unzip and cd into guangqi1d, copy the makefile with

cp makefiles/makefile.rmhd.gfortran Makefile

The key environment variables are PETSC_DIR, HDF5, and openmpi. Make sure they are correct.

Create a directory with name obj to store object files.

mkdir obj

Then

cd modules

Establish a soft link to the problem module. Choose the prolem module you want to use, for the 1D luminous red novae problem, do

ln -s lrne problem

Then

cd problem

The code will look for makefile.problem during the compilation.

Now, you can switch to guangqi directory and enter "make". If everything is correct, the compilation should start. After the compilation, a program called guangqi will be generated. Copy the excutable to the problem directory.

cp guangqi modules/problem

Now, cd to the problem directory.

cd modules/problem

You can type the following to create the model that reproduce the light curve of AT2019zhd by typing

python generate_at2019zhd_model.py

then, a new directory named model99 is created

Now go to model99 by

cd model99

modify the environment variable path_tables in global.data so that the program can find the opacity tables. Then, you can run the code with ./model99 (if your file is not appears to be an excutable, chmod +x model99 first)

The code will look for global.data for general setups and problem.data for the problem dependent setups. The 1D LRNe model will also look for bcinput.dat for boundary conditions.

The profile results are stored in out/*.h5 and the history (including the light curve information) is stored in history.data (Currently, the three columns are the simulation time, the luminosity, and the adiabatic sound speed of the inner boundary). You can modify the output information of history.data it in guangqi/modules/ejecta_1d/problem.f90

After you run one simulation, you can type

python at2019zhd.py 99

to compare your result to the observed result of at2019zhd.

Setup the problem

1D problem has an inner boundary and an outer boundary. The computational domain and physics are defined in global.data.

n_domain: defines the inner boundary and outer boundary radii.

tfinal: end of the simulation time, the starting time is 0 or a restart time.

CFL: CFL number <1, in this problem, use 0.4 or less if you want to resolve a strong shock.

nframe: how many .h5 files do you want to create. They are evenly spaced in time.

refine_type: static means no adaptive mesh refinement (AMR), adaptive mean fully AMR, mixed means you will specify some region with SMR and also have AMR.

nrefine_region: how many SMR region are there? They are specified in $refinement in global.data

max_refine_level: max AMR and SMR level.

restart: restart calculation. Sometimes you do not wish to run from the beginning.

restart_iframe: the number of frame you want to restart.

igravity =1: there is a point gravity source at r=0

iradiation: 0 means no radiation transfer, 4 has radiation transfer. Other numerics are not accepted.

nd: number of dimension. This is a 1D problem so nd=1

nx: resolution of the base level

blk_size_nx: The code has block structure, one block has blk_size_nx of cells. nx should be divisable of blk_size_nx

maw: mean atomic weight in perfect gas, effective when using gamma law gas

gamma_gas: gamma law gas

About the makefile

In the problem directory /modules/lrne, there is a makefile, it defines the EoS. When ieos=2, we use the realistic EoS, called in src/eos.f90, and the gamma_gas in global.data is not used. When using a realistic EoS, we need to set isolver=2 as well, called in src/hydro.f90. You can select a particular EoS among four eosmodules, by changing ieosmodule=1,2,3,4, called in src/eos_analytic.

The iopacity=1 means realistic opacity tables, which combines Malygin et al. (2014) and Semenov et al. (2003). Other chose of opacity tables, or analytic values can be done by setting iopacity=3, called in src/radiation_comm_functions.f90. By doing this, you can change user_kr(rho,tgas) and user_kp(rho,Erad,tgas) in the problem.f90 to define the opacity.

irecord=1 can let the code save some high frequency data by calling assemble_record_array(record_array) in rmhd.f90. The subroutine is defined in problem.f90 by the user, remember that the record_length is predefined by the user and you can change it in problem.data or problem.f90.

About the output variables

The output variables are declared in output_var_info.dat, and defined in src/io_out.f90. The surface data is stored by write_blk_dset_xsurface_parallel and the volume center data is stored by write_blk_dset_cell_parallel. You can modify the output_var_info.dat if you want to

output other data (remember to change the first number in output_var_info.dat which is the number of variables to output). You can also define more varibales to output in io_out.f90, but remember to allocate and deallocate them in data_structure.f90.

Data analysis

The output data is in the out directory. Guangqi uses hdf5 to save data, you can learn how to read hdf5 data with python here https://docs.h5py.org/en/stable/

The functions that help to read the file is in guangqi/scripts/assemble_1d_data.py.
For example, you could cd into problem/model99/out and do

python

import h5py

f=h5py.File('lrne00000.h5','r') to list the objects in lrne00000.h5

f['blk00001'].keys() to list the object in blk00001

Each blk stores some varibles, including

['Erad', 'Fradx', 'H2', 'HI', 'HII', 'HeI', 'HeII', 'HeIII', 'Planck', 'Rosseland', 'aradx', 'cpu_id', 'egv', 'entropy', 'mesh_x', 'pres', 'rho', 'temp', 'vx', 'x_center']