Hi-COLA Quick Start Guide

Hi-COLA is not yet fully automated – for now, you first need to run the Hi-COLA frontend separately to generate two data files describing the background cosmological expansion and pre-force quantities (see main documentation for a description). These files are then used as input for our modified version of the FML COLA solver. These two tasks are described in Sections 2 & 3 respectively. This is also represented schematically in Fig. 1.

1 How to install Hi-COLA

1.1 Dependencies

Hi-COLA has the following dependencies:

- NumPy* (1.23.5)
- SciPy* (1.10.0)
- SymPy* (1.11.1)
- ConfigObj* (5.0.8)
- FFTW (3.3.8)
- GSL (2.4)
- Lua (5.4.1)

*Many of these packages either come by default with environments like Anaconda, or can be installed using Conda. The version numbers indicated in brackets are those for which Hi-COLA is known to work with.

1.2 Installation instructions

Firstly, we would recommend creating an environment for Hi-COLA, to help protect against clashes in dependencies with other modules on your machine, if relevant[1]. Navigate to the root directory when you want to install Hi-COLA. Then clone the GitHub repo using either:

- `git clone https://github.com/Hi-COLACode/Hi-COLA`
- `git clone git@github.com:Hi-COLACode/Hi-COLA.git`

[1]An example of doing so in Anaconda can be found at `https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html` Creating an environment that installs all default Anaconda packages, and then installing any of the above dependencies that were left out should suffice, for instance.

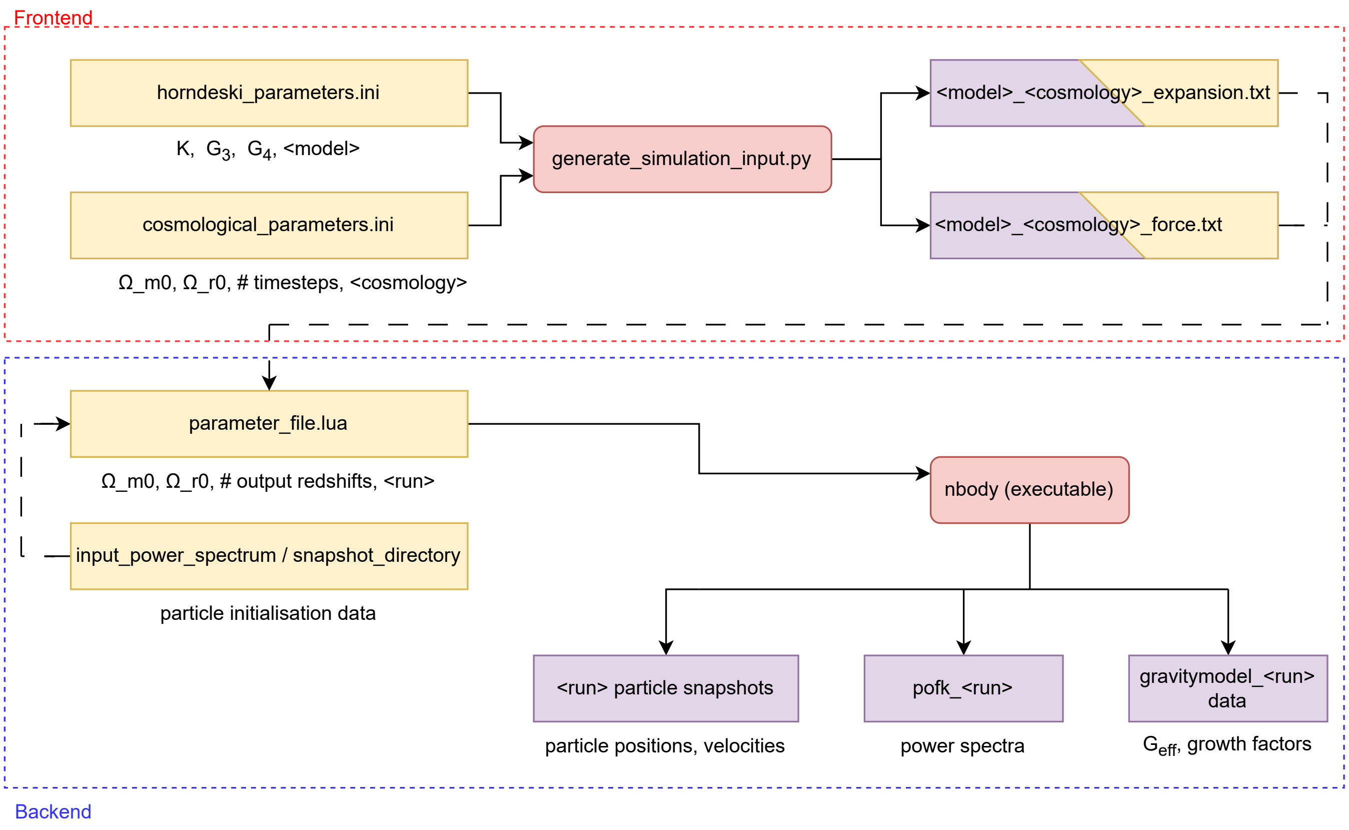
Frontend
horndeski_parameters.ini
K, G3, G4, <model>
cosmological_parameters.ini
Ω_m0, Ω_r0, # timesteps, <cosmology>
generate_simulation_input.py
<model>_<cosmology>_expansion.txt
<model>_<cosmology>_force.txt
parameter_file.lua
Ω_m0, Ω_r0, # output redshifts, <run>
input_power_spectrum / snapshot_directory
particle initialisation data
nbody (executable)
<run> particle snapshots
particle positions, velocities
pofk_<run>
power spectra
gravitymodel_<run> data
Geff, growth factors
Backend

Figure 1: Flowchart showing general process of a Hi-COLA run. The Hi-COLA Frontend is the top half of this figure (encased in the dashed red box), and the Hi-COLA Backend is the lower portion (encased in the dashed blue box). Objects in hard-edged yellow rectangles are generally parameter or input files, whereas objects in soft-edged red rectangles are 'engine' files that take in input files, and produce outputs, shown in hard-edged purple boxes. The arrows indicate where files are fed into or what is produced. Some input files are fed to engine files via other parameter files, and these are indicated by dashed arrows. Additionally, the outputs of the Frontend are in turn used as inputs for the Backend, indicated by the dual colours. Below the rectangles are some text captions that give some examples of what is contained in the files. Note: `cosmological_parameters.ini` is also referred to as `numerical_parameters.ini` in this guide.

Navigate to the main `Hi-COLA` directory which should be `<install_dir>/Hi-COLA/HiCOLA/FML/COLASolver`.

Before compiling `Hi-COLA`, we need to customise the `Makefile` to your machine. To do this, first set `FML_INCLUDE=<install_dir>/Hi-COLA/HiCOLA/`. Next you need to point the various `_INCLUDE` and `_LIB` settings to where FFTW, GSL, and LUA are installed on your machine (load the corresponding modules if you're using a High Performance Cluster). Note that you can set `FFTW_OMP_LINK = -lfftw3_threads` or `FFTW_OMP_LINK = -lfftw3_omp` depending on whether `USE_OMP` was set to `false` or `true` respectively[2].

Now compile using `make clean; make`. This will create an executable called `nbody` within the `FML/COLASolver` directory.

Error notes:

- If you are trying to get a minimum working example of `Hi-COLA` running on a laptop or desktop, you may wish to disable all OpenMP and MPI options (unless you know how to use these on your system).
- We are aware of an error generated by the C compiler Clang on some Apple Macs, of the form:
  `<install_dir>/HiCOLA/FML/FriendsOfFriends/FoF.h:186:59: error: template parameter redefines default argument template <class T, int NDIM, class FoFHaloClass = FoFHalo<T, NDIM>>`.

  For this we suggest the following fix. Open the file `<install_dir>/HiCOLA/FML/FriendsOfFriends/FoF.h`, and change line 186 to:
  `template <class T, int NDIM, class FoFHaloClass>`

2 How to run the frontend to generate input for the FML COLA solver

Navigate to `<install_dir>/` and run `python3 setup.py install --user`. We recommend creating an environment before doing this installation. To run the frontend, we need to specify two .ini parameter files, examples of which can be found in `<install_dir>/HiCOLA/Parameter_files/Frontend_parameter_files`. The first should follow the format of the `horndeski_parameters.ini` template, and the second the format of the `numerical_parameters.ini` template. The former specifies aspects of the gravity model to be run. The latter specifies settings for integration, such as the number of outputs, maximum redshift up to which to evolve to, and cosmological parameter values.

Ready-to-run example input files are provided in the directory `Frontend_parameter_files/examples`. The file `WMAP_cosmological_parameters.ini` is a specific example of the generic `numerical_parameters.ini` template, customised to a WMAP cosmology. Any of the files in the `cubic_Galileon` and `ESS` folders can be used as an example `horndeski_parameters.ini`. Note that you may wish to edit the output directory specified near the top of the `cubic_Galileon` and `ESS` model files.

In the next two subsections we give brief instructions on how to customise these input files.

2.1 Customising `horndeski_parameters.ini`

`horndeski_parameters.ini` is the first parameter file that is fed to `generate_simulation_input.py`; it is where the functional form of the reduced Horndeski model of choice is specified. Open a copy of the template `<install_dir>/Hi-COLA/HiCOLA/Parameter_files/Frontend_parameter_files/horndeski_parameters.ini` with a different name e.g. `horndeski_model_new_params.ini`. First, give your new model a name via `model_name` – this will be used as a prefix for the expansion and force output files. Then specify the path to the directory where you want the output files to be saved.

[2]For reference, the combination, `USE_OMP = false` and `FFTW_OMP_LINK = -lfftw3_threads` is known to work.

Next, we specify the Horndeski Lagrangian functions. The functional dependence of these functions should follow the dependence[3] specified in the appendix of the manual (section 8), or alternatively section 2 of arXiv:2209.01666. The expected symbols used to construct the Horndeski functions are ϕ, X and any symbols used to denote real-valued constants. The symbols used to specify these model constants must then be declared in the theory symbols section (see the comments in the file for the syntax).

Next, set the parameter `f_phi` to a value between 0 and 1 (inclusive, see eq.2.26 of arXiv:2209.01666), which determines the fraction of total dark energy in the form of the Horndeski scalar field, with the remaining portion ($1 -$ `f_phi`) allocated to a cosmological constant. For example, `f_phi=1` indicates all of the dark energy sector is constituted by the scalar and there is no cosmological constant.

We also need to specify the initial conditions Hubble0, which sets the value for the normalised Hubble function[4] at $z = 0$, and ϕ_0', which sets the value for the scalar field derivative at $z = 0$. Finally, as we obtain a constraint equation from the Horndeski field equations (eq 2.3 in arXiv:2209.01666), we can use this to fix one of the free parameters or initial conditions of the model. Refer to the comments in the template file to understand how to choose which initial condition or model parameter is fixed via the field constraint equation. Examples of `horndeski_parameters.ini` are provided in `<install_dir>/Hi-COLA/HiCOLA/Parameter_files/Frontend_parameter_files/examples/` for the cubic Galileon and Extended Shift Symmetric models.

2.2 Customising `numerical_parameters.ini`

The second parameter file required to run the frontend should follow the format of `numerical_parameters.ini` template. This file generally controls the cosmological parameters (Ω_{i0}) and the integration settings (such as the number of output redshifts, the redshift to evolve up until, etc.) Open a copy of this template with a different name e.g. `numerical_new_params.ini`. First, set `cosmo_name` – this will be added as a suffix after `model_name` in the output file names.

The next set of variables determine settings for the integration peformed by `SciPy`. As explained in the template file, these determine the maximum redshift up to which to integrate, the number of redshifts to output values at, the option to suppress `lsoda` warnings (useful if you know to ignore the warnings `lsoda` gives when quantities appear to rapidly increase or decrease). The `GR_flag` variable, when set to `True`, changes the integration run into one for ΛCDM. Naturally, this means that the Horndeski settings specified in the `horndeski_model_new_params.ini` file will be ignored (with the exception of the model name and output variables)! Lastly, set the initial values for the cosmological density parameters at $z = 0$. These can be either be specified as density parameters divided by the dimensionless Hubble rate, Ω_{i0}/h^2, or simply directly, Ω_{i0}. The former style of specifying the density parameters only takes effect if the variables for the latter style are set to `None`, i.e. $\Omega_{i0} = \text{None}$. An example for `numerical_parameters.ini` is provided in `<install_dir>/Hi-COLA/HiCOLA/Parameter_files/Frontend_parameter_files/examples/` for cosmological parameter values as measured by WMAP9.

2.3 Running the frontend

The run command is then (for example):

```
python3 -m HiCOLA.Frontend.generate_simulation_input A B
```

where `A` is the path to `horndeski_model_new_params.ini` and `B` the path to `numerical_new_params.ini`.

This will generate an expansion file(`<model_name>_<cosmo_name>_expansion.txt`) and a pre-force file (`<model_name>_<cosmo_name>_force.txt`) to be used as input for a `Hi-COLA` `FML` simulation. The expansion file consists of three columns of data describing the expansion history in the format $[a, E, E'/E]$, that is: the scale factor,

[3]Note that the code operates in terms of *massless* variables; we strongly recommend the reader reads the material recommended above to understand how these are constructed before attempting to specify a new model. Otherwise, *caveat emptor*.

[4]We keep this variable's name generic because it is possible to rescale the Hubble function if one's reduced Horndeski theory possesses rescaling symmetries. Normally, in the absence of any rescaling, the default assumption is that a user of `Hi-COLA` is working with $E := H/H_0$ (therefore `Hubble0` $\equiv E_0 = 1$ would be the appropriate value to set).

the Hubble rate, and the logarithmic derivative of the Hubble factor with respect to the logarithm of the scale factor, $d\ln H/d(\ln a)$. The pre-force file describes the time-dependence of the screening and coupling quantities necessary for computing the screened fifth forces, and consists of three columns of data in the format $[a, \chi/\delta_{\mathrm{m}}, \beta]$. See section 3 of the manual, and section 3.1 of arXiv:2209.01666for the definitions and meanings of χ/δ_{m} and β.

3 How to run a simulation with the FML COLA solver

3.1 Setting up the FML COLA solver parameter file

Navigate to `<install_dir>/HiCOLA/Parameter_files/Backend_parameter_files/`. Make a copy of the `Hi-COLA_params.lua` template with a new name e.g. `sim_for_my_new_model.lua` to edit. You may also wish to edit the directory for output files.

To set the background expansion history to the one produced by the Hi-COLA frontend, set `cosmology_model="HiCOLA"` and point `HiCOLA_expansion_filename` to the desired expansion file generated in section 2 above. Similarly, to set the gravitational forces to the ones produced by the Hi-COLA frontend, set `gravity_model="HiCOLA"` and point `HiCOLA_preforce_filename` to the desired pre-force files generated in section 2 above. Note that you will see sections in the `.lua` file for DGP, Jordan-Brans-Dicke gravity, symmetron gravity etc. – these are hard-coded models from the original FML implementation, and can be ignored for native Hi-COLA runs.

It is important to set the cosmological parameters here to be the same as those in the frontend python file you used to generate the expansion/pre-force files. The exception is `cosmology_OmegaLambda`[5], which should be set to the total dark energy i.e. $\Omega_{\mathrm{DE}0} = \Omega_{\Lambda 0} + \Omega_{\phi 0}$ rather than just $\Omega_{\Lambda 0}$. That is, ensure `cosmology_OmegaLambda = 1 - Omega_r0 - Omega_m0`, where the `Omega_m0, Omega_r0` were set in `numerical_new_parameters.ini`. **Note also that massive neutrinos are not currently included in the Hi-COLA frontend**, so you should set `cosmology_OmegaMNu=0.0`. Be aware that if these quantities do not sum to 1, then FML will assume there must be some curvature (which the frontend does not currently allow). In section 5 of the manual we give a list of default values to use in the backend parameter file should one want some guidance.

3.2 How to generate initial conditions

There are two options for supplying initial conditions to the backend: using a matter power spectrum, or using particle snapshots.

Firstly, the matter power spectrum method. We suggest using a linear power spectrum that has been 'backscaled' from the target output redshift z_{target} (e.g. $z = 0$) to the initial simulation redshift z_{ini}. This ensures the effects of radiation between z_{ini} and z_{target}, which are not otherwise included in the forward evolution performed by the simulation, are already present in the initial conditions. Ideally, the target matter power spectrum should be generated using an Einstein-Boltzmann solver that has been modified to include the impact of Horndeski gravity, such as `hi_class`.

The backscaling of this target power spectrum can then be peformed using the inbuilt methods of FML. In the FML COLA solver parameter file, set `ic_type_of_input = "transferinfofile"` or `ic_type_of_input = "matterpowerspectrum"` to match the type of input file your Einstein-Boltzmann solver has generated, then point `ic_input_filename` to the location of this generate input file. Then set `ic_input_redshift = <`z_{target}`>` and `ic_initial_redshift = <`z_{ini}`>`, and ensure `ic_use_gravity_model_GR = false`. We recommend setting `ic_random_field_type = "gaussian"`, although the FML code can also produce non-Gaussian ICs. The default settings are `ic_fix_amplitude = false` and `ic_reverse_phases = false`, although see the full user manual for a discussion of fixed and paired ICs.

[5]This is because what is called `cosmology_OmegaLambda` in the original FML code becomes the sum of the true cosmological constant (if present) and the scalar field energy density at $z = 0$ in Hi-COLA.

If you don't have access to an Einstein-Boltzmann solver that can handle your particular Horndeski model, you can manually do the following *approximate* backscaling based on a ΛCDM power spectrum:

$$P_{\rm L}^{\rm MG,quasi-back}(k, z_{\rm ini}) = \left[\frac{D_1^{\rm MG}(k, z_{\rm ini})}{D_1^{\Lambda\rm CDM}(k, z_{\rm target})}\right]^2 P_{\rm L}^{\Lambda\rm CDM}(k, z_{\rm target})$$

where the linear growth factors D_1 match those compute internally by `FML`. Note that the final results of this approximate scaling are slightly less accurate than the method of the previous paragraph.

The second option is to read in a particle snapshot in GADGET format that explicitly delineates the initial positions and velocities of the particles. This snapshot may have been generated from some other code, for instance, an N-body simulation that one wishes to generate `Hi-COLA` results to compare with. To utilise this option, set `ic_type_of_input = read_particles`. Then set `ic_reconstruct_gadgetfilepath` to the path where the GADGET files are located. Note that the path should end in `/gadget`, so that the backend locates the GADGET files correctly.

3.3 Running the `FML` COLA solver

If necessary (e.g. if running on a cluster), load your FFTW, GSL and Lua modules. The basic run command is `mpirun -np <number of cores> nbody <params.lua>`. The output will be saved in the directory specified by the `output_folder` variable in `params.lua`. This will include files that contain information on the effective gravitational constant, split by scale (filename format: `gravitymodel*`), power spectra split by redshift (filename format: `pofk*`) and directories that include gadget files encoding particle positions and velocities (if `output_particles` in `params.lua` was set to `true`).