Hi-COLA User Manual

For instructions on installing and running Hi-COLA, please see the Quickstart guide included in the GitHub repo. This document provides more detail on the structure and solution methods of Hi-COLA. The developer team can be contacted at: team.hicola@gmail.com.

Usage: We provide Hi-COLA free for public use. We ask that you kindly cite the paper referenced below in all works made using this code. We will call this the 'code presentation paper' in the rest of the manual.

"Hi-COLA: *Fast, approximate simulations of structure formation in Horndeski gravity*", Bill Wright, Ashim Sen Gupta, Tessa Baker, Georgios Valogiannis and Bartolomeo Fiorini.

Cite as: Bill S. Wright et al., JCAP 03 (2023) 040.

ArXiv: https://arxiv.org/abs/2209.01666

Published version: https://iopscience.iop.org/article/10.1088/1475-7516/2023/03/040

BibTeX entry:
@article{Wright_2023,
doi = {10.1088/1475-7516/2023/03/040},
url = {https://dx.doi.org/10.1088/1475-7516/2023/03/040},
year = {2023},
month = {mar},
publisher = {IOP Publishing},
volume = {2023},
number = {03},
pages = {040},
author = {Bill S. Wright and Ashim Sen Gupta and Tessa Baker and Georgios Valogiannis and Bartolomeo Fiorini and The LSST Dark Energy Science collaboration},
title = {Hi-COLA: fast, approximate simulations of structure formation in Horndeski gravity},
journal = {Journal of Cosmology and Astroparticle Physics} }

Contents

1 Hi-COLA philosophy

The primary purpose of Hi-COLA is to enable rapid exploration of the effects of different modified gravity models on nonlinear large-scale structure scales. Notably, the word 'different' here does **not** mean 'the user selects from a menu of pre-defined gravity models'. Instead, the goal of Hi-COLA is to enable the user to freely specify their own gravity model for simulation 'on the fly'. Furthermore, we wished this task to be as easy as possible for someone who does not have extensive experience in cosmological simulations, e.g. a user with a primarily theoretical/model-building background.

For this reason, Hi-COLA has been designed to accept input in the form of a Lagrangian for gravity, immediately connecting the code to the way new models are naturally formulated. Now, given the near-infinite flexibility that Lagrangians possess, some degree of restriction had to be imposed in the name of computational feasibility. For this reason, Hi-COLA has been designed to handle general gravity models that fall within the reduced Horndeski class – see section 2 of the accompanying release paper referenced above for a detailed description.

With this choice of the reduced Horndeski class, the acronym Hi-COLA was born ('Horndeski gravity In the COmoving Lagrangian Acceleration method'). We may seek to generalise the code's capabilities beyond the parent reduced Horndeski Lagrangian in future releases.

Important note: the current release of Hi-COLA is only suitable for use in theories which implement Vainshtein screening. Chameleon screened-theories cannot yet be fully realised. Lifting this restriction is one of the developer team's top priorities, to be in-keeping with our philosophy above. We hope to make this feature public in a future release of Hi-COLA; this documentation and the code website will make it clear when this has happened.

2 The structure of Hi-COLA

Hi-COLA contains two major building blocks: 1) a *frontend*, and 2) the COLA solver itself.

1. The frontend receives the input gravity model from the user, and is responsible for computing all necessary quantities related to the cosmological expansion history of that model (we will refer to this later as 'pre-computation'). It does this via the use of symbolic algebra routines (courtesy of Ashim Sen Gupta) that manipulate the user's chosen Lagrangian algebraically, before converting those algebraic outputs into solvable numerical functions.

2. These precomputed quantities are then passed into a COLA solver suite, adapted from the FML library by Hans Winther. This part of the code is effectively the true simulation, making use of the COLA algorithm (arXiv:1301.0322) to produce reasonably accurate outputs at modest computational cost.

At present, the code blocks 1) and 2) are to be run in turn, manually. The frontend can be used in standalone mode to compute the background expansion history of a user-specified model. This can useful to check a model produces

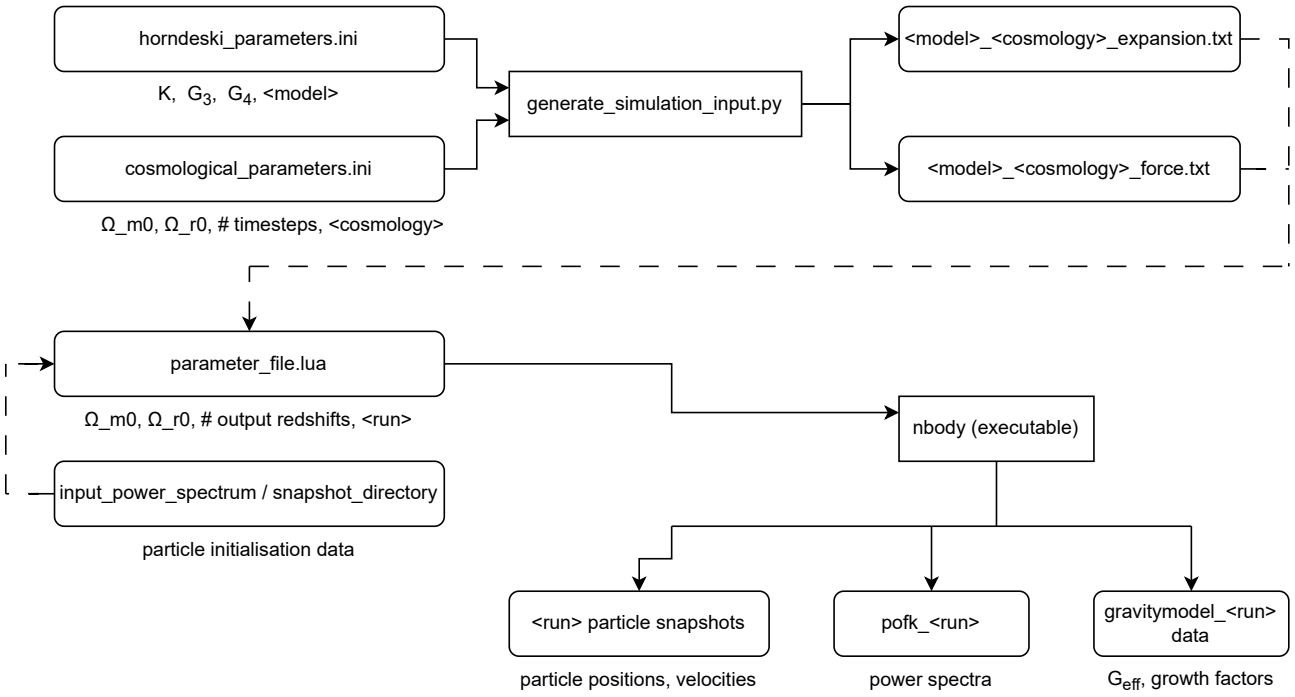

Figure 1: Flowchart showing general process of a `Hi-COLA` run. Objects in rounded rectangles are generally parameter, input or output files, whereas objects in hard-edged rectangles are 'engine' files that take in files in rounded rectangles, and produce outputs - as indicated by the arrows. Some input files are fed to engine files via other parameter files, and these are indicated by dashed arrows. Below the rectangles are some text captions that give some examples of what is contained in the files. The `Hi-COLA` frontend is the top half of this figure (above the horizontal dashed line), and the `Hi-COLA` backend is the lower portion.

a cosmological expansion history consistent with some chosen requirements; see section 5.1 of the code presentation paper for an example of what constitutes a minimal, viable expansion.

A diagram showing the inputs, outputs and flow of key quantities around `Hi-COLA` is shown in Fig. 1.

3 Modified forces and expansion in `Hi-COLA`

Three key ingredients are needed to perform a COLA simulation:

1. The expansion of the simulation box.
2. The forces between particles.
3. First and second-order growth factors computed in Lagrangian Perturbation Theory (LPT).

For 1), we only need $E = H/H_0$ and E'/E, which the `Hi-COLA` frontend computes directly by solving the modified Friedmann equations appropriate for the user-specified theory (see section 2.1 of the code presentation paper, also Appendix 8 of this document).

For 2), our expression for the modified force law, including the screened fifth force, is

$$F_{\rm tot} \quad = \quad F_{\rm N} \frac{G_{G_4}}{G_{\rm N}} \left[1 + \beta\, S\right], \tag{1}$$

where β is the coupling (a function of redshift only), and S is the screening coefficient defined by

$$S = \frac{2}{\chi}\left(\sqrt{1+\chi} - 1\right).$$

See section 3.1 in the code presentation paper referenced above for derivation of these expressions. Here χ is a combination of redshift-dependent background quantities, multiplied by the density perturbation δ_m. The `Hi-COLA` frontend precomputes the quantities β and $\chi/\delta_{\rm m}$ for the user-specified theory, since these only require solution of the Friedmann equations. The density $\delta_{\rm m}$ is computed in the simulation from the distribution of particles, and together with the frontend inputs is assembled to form the screening factor S above. With S in hand, via Eq.(1) we can compute the screened fifth forces between particles in the simulation.

Now for 3). In `Hi-COLA` the first and second order growth factors D_1 & D_2 are computed in LPT by solving:

$$D_1'' = \frac{3}{2}\frac{\Omega_{\rm m0}}{E^2 a^3}\frac{G_{\rm eff}}{G_{\rm N}} D_1 - \left(2 + \frac{E'}{E}\right) D_1',$$
$$D_2'' = \frac{3}{2}\frac{\Omega_{\rm m0}}{E^2 a^3}\frac{G_{\rm eff}}{G_{\rm N}} \left[D_2 - D_1^2\right] - \left(2 + \frac{E'}{E}\right) D_2',$$

where $G_{\rm eff}/G_{\rm N} = 1 + \beta$. These equations assume a linear modification to gravity; to be more accurate, the equation for the second order growth factor should include the leading order screening impact. However, we have confirmed that this assumption in the LPT equations does not significantly affect the power spectra produced by the COLA simulation code for sensible simulation parameter choices. This is because the impact of LPT is subdominant to the impact of forces between particles in the simulations computed using Eq. (1) that do include screening.

For further information about LPT equations and their use in COLA with modified gravity, see arXiv:1703.00879 and arXiv:1705.08165.

4 Which parts of the original `FML` code have been modified?

Here we list the key structural changes made in `Hi-COLA` to the COLA solver provided with the FML library. The `Hi-COLA` frontend is original work unique to this codebase.

- A new HiCOLA cosmology option via the new file `cosmology_HiCOLA.h` allows the code to read the expansion history from a user-specified file (which we generate using the frontend).
- A new HiCOLA gravity model option via the new file `gravitymodel_HiCOLA.h` has been added. This new file allows the code to read the pre-force quantities (β & $\chi/\delta_{\rm m}$) from a user-specified file (generated using the frontend). These quantities are described in section 3.
- Modifications to `Main.c` which instructs the backend to spline the pre-force quantities sent by the frontend when the `gravitymodel_HiCOLA.h` option is triggered.
- Adjustments have been made to the LPT equations in `COLA.h` to make them consistent with Eq. (4.10) of 1703.00879, which are then compatible with the form presented in section 3.

5 Recommended FML COLA solver parameters

Here we provide some default parameter settings for a typical COLA simulation. These should be entered in the `.lua` parameter file supplied to the modified COLA executable. See the Quickstart guide to understand the contents of this `.lua` file and how it enters the `Hi-COLA` pipeline.

- A boxsize of $L = 512$ Mpc/h or 1024 Mpc/h are common choices for cosmological simulations of this type.
- A particle number $N_{\rm part} = L^3$ ensures decent mass resolution.
- The resolution of the force mesh grid: $N_{\rm force\ mesh} = 3N_{\rm part}^{1/3}$ – `https://arxiv.org/abs/1509.04685` investigated the importance of the ratio $N_{\rm force\ mesh}/N_{\rm part}^{1/3}$; finding that a ratio of 3 works well for COLA simulations.
- $N_{\rm timesteps} = 50$ should give decent small-scale accuracy when combined with our other recommended settings i.e. set `timestep_nsteps = {50}`.

  Increasing $N_{\rm timesteps}$ will improve the accuracy of the results at small scales, but will also increase the runtime of the simulation. Depending on other simulation parameter settings, there can also be diminishing returns to the impact of increasing $N_{\rm timesteps}$ on the accuracy at small scales, as discussed below.
- `https://arxiv.org/abs/1509.04685` identified that the best choice for timestep spacing is linear in scale factor a i.e. set `timestep_scalefactor_spacing = "linear"` in the COLA parameter file.

6 Fixed and paired initial conditions

Traditionally, to reduce the impact of cosmic variance on the real-space matter power spectrum at large scales, one would need to run many simulations each with a different realisation of standard (i.e. neither fixed nor paired) Gaussian ICs. This would be achieved by setting `ic_random_field_type = "gaussian"`, `ic_fix_amplitude = false`, and `ic_reverse_phases = false`, while varying `ic_random_seed` and averaging the power spectra across the realisations.

However, it's also possible to use paired-fixed ICs [arXiv:1511.04090,arXiv:1603.05253] to reduce the impact of cosmic variance with only two simulations. This is achieved by setting `ic_fix_amplitude = true` and running one simulation with `ic_reverse_phases = false` and a second with `ic_reverse_phases = true`, while keeping `ic_random_seed` the same in both. The average of the power spectra from these two paired-fixed simulations should have significantly reduced cosmic variance in the real-space matter power spectrum at large scales, equivalent to the average of power spectra from many traditional Gaussian realisations. However, users should be aware that this paired-fixed method is not suitable for computing covariance matrices [arXiv:1903.08518].

Also see `https://fml.wintherscoming.no/randomfield.php`.

7 Understanding the Hi-COLA output

The `Hi-COLA` frontend will produce two text files describing the cosmological background expansion and quantities needed for the force computation between particles, see Figure 1. These serve as inputs for the modified COLA simulation. The outputs of the COLA simulation (aka the backend) itself are:

- `Hi-COLA` will produce a file named `cosmology_<sim_name>.txt` containing the background quantities as a function of time in the following order $[a, E, E'/E, \Omega_{\rm m}, \Omega_{\rm r,tot}, \Omega_\Lambda]$. E and E'/E are guaranteed to match those of the input file produced by the frontend. $\Omega_{\rm m}$ and $\Omega_{\rm r,tot}$ are accurate as long as you set consistent values of $\Omega_{\rm b0}$, $\Omega_{\rm c0}$, and Ω_{r0} in the frontend and `FML` COLA solver parameter files. For now, in our `Hi-COLA` case, Ω_Λ will

not be correct and should be ignored (this is a legacy output of the original FML). The quantity $1-\Omega_{\rm m}-\Omega_{\rm r,tot}$ will give the **sum** of the energy density in a cosmological constant (if present) and the Horndeski scalar field as a function of redshift.

- Hi-COLA will also produce several files named gravitymodel_<sim_name>_k<value>.txt that contain information about the strength of gravity and linear and non-linear growth as functions of time for a particular scale. Each file is for a different fixed scale indicated by the file name. The first three columns $[a, G_{\rm eff}/G_{\rm N}, D_1]$ are the most relevant.

- Do I need to save the particle data?

  Particle data files are very large, and can easily start to take up excessive amounts of space on your machine.

  If you are only interested in the matter power spectrum, and are happy using FML's inbuilt power spectrum computer, then you generally do not need to save the particle data files.

  However, there are some cases where you may wish to save the particle data files. For example, you may wish to save the snapshot at $z_{\rm ini}$ in case you or someone else later wants to run a simulation using identical initial conditions. Additionally, if you want to do any post-processing, such as visualising the density field, creating halo catalogues etc, then it is vital to save the particle data. To do this, set `output_particles = true`. When `true`, also specify the redshifts at which to save particle data, the directory in which to save them, and the file format for the snapshots (`output_redshifts, output_folder, output_fileformat` respectively).

- Which power spectrum files have shot noise included? This is controlled by the pofk_subtract_shotnoise variable in parameter_file.lua. When set to true, shot noise is subtracted from the power spectrum calculated by the backend.

- What scale should I trust my power spectrum to?

  Clearly, the answer depends on what level of accuracy you need. However, it also depends on three key factors, two of which you have some control over through the values of the simulation parameters you choose.

  The first factor is the number of timesteps and how they are spaced in redshift. The more timesteps the better, although there are diminishing returns, and the 2LPT part of COLA approach ensures the results are accurate on large scales even with a small number of timesteps. See typical recommended settings in section 5.

  The second factor is the force resolution, which is set by the relative size of a force mesh cell and thus depends on the ratio $L/N_{\rm force\ mesh}$. L and $N_{\rm force\ mesh}$ are controlled by simulation_boxsize and force_nmesh respectively in the FML COLA solver parameter file. See typical recommended settings in section 5.

  The third factor is the impact of the approximations we have used to compute the screened fifth forces for Horndeski gravity. These are fundamental to the Hi-COLA code and cannot be easily changed. See the code presentation paper for discussion of these, and a validation exercise in Cubic Galileon gravity.

- What is the mass resolution?

  The mass resolution is set by the mass of a single particle in the simulation, which is given by $m_{\rm part} = \Omega_{\rm m0}\rho_{\rm crit,0}[L/N_{\rm part}]^3$ which can be rewritten $m_{\rm part} = 8.517\times10^{10}[\Omega_{\rm m0}/0.3][L/h^{-1}Gpc/N_{\rm part}/1000]^3 h^{-1}M_\odot$. L and $N_{\rm part}^{1/3}$ are controlled by simulation_boxsize and particle_Npart_1D respectively in the FML COLA solver parameter file.

8 Appendix on frontend equations of motion

Here we outline the equations solved for by the frontend of Hi-COLA. For a more contextual reading of what these equations are eventually used for, one can refer to arXiv:2209.01666. In brief, the frontend solves for the isotropic and homogeneous background for theories described by the following action:

$$S = \int d^4x\sqrt{-g}\left(\tilde{G}_4(\tilde{\phi})R + \tilde{K}(\tilde{\phi},\tilde{X}) - \tilde{G}_3(\tilde{\phi},\tilde{X})\Box\tilde{\phi} - M_{\rm P}^2\Lambda + \mathcal{L}_m\right). \tag{2}$$

We call this class of scalar-tensor theories the reduced Horndeski class. These are Horndeski theories where gravitational waves travel at lightspeed. Quantities in eq. (2) are denoted with tildes to indicate they carry dimensions. The code works with dimensionless analogues of these quantities, and here we will denote them without tildes.

Let us review the mass dimensions[1] of the various components of the action.

$$[\tilde{\phi}] = M \tag{3}$$
$$[\tilde{K}] = M^4 \tag{4}$$
$$[\tilde{G}_3] = M \tag{5}$$
$$[\tilde{G}_4] = M^2 \tag{6}$$

We define dimensionless versions of these objects through the following equations:

$$\tilde{\phi} = M_s\phi \tag{7}$$
$$\tilde{K} = M_K^2H_0^2K \tag{8}$$
$$\tilde{G}_3 = M_{G3}G_3 \tag{9}$$
$$\tilde{G}_4 = M_{G4}^2G_4 \tag{10}$$
$$H = H_0E. \tag{11}$$

The quantities of the form M_a are the mass scales of the associated functions. When we take ratios of mass scales we use the following notation to summarise them: $M_a/M_b \equiv M_{ab}$. We call quantities of this form 'mass ratios'. The next task is to define a dimensionless analogue of $\tilde{X}$. First, let us take a moment to check what the mass dimension of $\tilde{X}$ is,

$$[\partial_t] = T^{-1} = M \tag{12}$$
$$\implies [\tilde{X}] = ([\partial_t][\tilde{\phi}])^2 \tag{13}$$
$$= \frac{M^2}{T^2} = M^4 \tag{14}$$

We can change variables to $x = \ln(a)$ (which is dimensionless), which will help expose what to define as X. Derivatives with respect to x are denoted by a prime ($'$).

$$\tilde{X} = \frac{1}{2}\cdot(\partial_t\tilde{\phi})^2 \tag{15}$$
$$= \frac{M_s^2}{2}\cdot(\partial_t\phi)^2 \tag{16}$$

$$\tilde{X} = \frac{M_s^2H^2}{2}(\partial_x\phi)^2 \equiv \frac{M_s^2H^2}{2}(\phi')^2 \tag{17}$$

[1]Note, we work in geometric units where $c = h = 1$, which has the effect of allowing us to freely interchange length, time and inverse mass dimensions, $L = T = M^{-1}$. This follows from $[c] = LT^{-1}$, and $[h] = MT$ if $c = 1$ is assumed and used to exchange L for T.

Rather than having part of the mass dimension carried by H, which is a function, we can instead define X through

$$X = \frac{H^2}{2H_0^2}(\phi')^2 \equiv \frac{E^2(\phi')^2}{2} \tag{18}$$

so that

$$\tilde{X} = M_s^2 H_0^2 X. \tag{19}$$

We are now in a position to use these dimensionless quantities to construct dimensionless analogues of derivatives of the Horndeski functions, etc., as shown in the set of equations starting with eq. (20). In eq. (20) we show the steps for how this is done, with all the other quantities being constructed in a similar fashion.

$$\begin{aligned} K_X &= M_K^4 \tilde{K}_{\tilde{X}} \partial_X \tilde{X} \\ &= M_K^4 \tilde{K}_{\tilde{X}} \partial_X \left(\frac{X}{M_s^2 H_0^2} \right) = \frac{M_K^4}{M_s^2 H_0^2} \cdot \tilde{K}_{\tilde{X}} = \frac{M_{\text{K-new}}^2}{M_s^2} \tilde{K}_{\tilde{X}} \end{aligned} \tag{20}$$

$$K_\phi = \frac{M_K^4}{M_s} \cdot \tilde{K}_{\tilde{\phi}} \tag{21}$$

$$K_{XX} = \left(\frac{M_K}{M_s H_0} \right)^4 \cdot \tilde{K}_{\tilde{X}\tilde{X}} \tag{22}$$

$$K_{X\phi} = \frac{M_K^4}{M_s^3 H_0^2} \cdot \tilde{K}_{\tilde{X}\tilde{\phi}} \tag{23}$$

$$G_{3X} = \frac{M_{G3}}{M_s^2 H_0^2} \cdot \tilde{G}_{3\tilde{X}} \tag{24}$$

$$G_{3XX} = \frac{M_{G3}}{M_s^4 H_0^4} \cdot \tilde{G}_{3\tilde{X}\tilde{X}} \tag{25}$$

$$G_{3\phi} = \frac{M_{G3}}{M_s} \cdot \tilde{G}_{3\tilde{\phi}} \tag{26}$$

$$G_{3\phi X} = \frac{M_{G3}}{M_s^3 H_0^2} \cdot \tilde{G}_{3\tilde{\phi}\tilde{X}} \tag{27}$$

$$G_{3\phi\phi} = \frac{M_{G3}}{M_s^2} \cdot \tilde{G}_{3\tilde{\phi}\tilde{\phi}} \tag{28}$$

$$G_{4\phi} = \frac{M_{G4}^2}{M_s} \cdot \tilde{G}_{4\tilde{\phi}} \tag{29}$$

$$G_{4\phi\phi} = \frac{M_{G4}^2}{M_s^2} \tilde{G}_{4\tilde{\phi}\tilde{\phi}} \tag{30}$$

$$G_{4\phi X} = \frac{M_{G4}^2}{M_s^3 H_0^2} \tilde{G}_{4\tilde{\phi}\tilde{X}} \tag{31}$$

$$G_{4X} = \frac{M_{G4}^2}{M_s^2 H_0^2} \cdot \tilde{G}_{4\tilde{X}} \tag{32}$$

The equations of motion the frontend solves for numerically, using `odeint` from the package `SciPy` are as shown in eq. (33), (34) and eq. (39). In `Hi-COLA`, the file containing the functions that encode these expressions is `expression_builder.py` in the `frontend` directory. For example, the code analogue of eq. (33) is the `EprimeEODEDRHS` function.

The dynamical Friedmann equation is

$$\frac{E'}{E} = \left(1 + \frac{M_{G3G4}M_{sG4}}{2}\frac{\tilde{X}\tilde{G}_{3\tilde{X}}}{\tilde{G}_4}\frac{\tilde{B}_1}{\tilde{A}} - \frac{\tilde{G}_{4\tilde{\phi}}}{2\tilde{G}_4}\frac{\tilde{B}_1}{\tilde{A}}\right)^{-1}\Bigg\{ \\ - \frac{1}{4\tilde{G}_4}\left[\frac{M_{KG4}^2}{E^2}\tilde{K} - \frac{2M_{sG4}M_{G3G4}}{E^2}\tilde{X}\tilde{G}_{3\tilde{\phi}} + 4\tilde{G}_{4\tilde{\phi}}\tilde{\phi}' + \frac{4}{E^2}\tilde{X}\tilde{G}_{4\tilde{\phi}\tilde{\phi}}\right] \\ - \frac{1}{2}\left(\frac{M_{pG4}^2(\Omega_r - 3\Omega_\Lambda)}{2\tilde{G}_4} + 3\right) - M_{G3G4}M_{sG4}\frac{\tilde{X}\tilde{G}_{3\tilde{X}}}{2\tilde{G}_4}\frac{\tilde{B}_2}{\tilde{A}} + \frac{\tilde{G}_{4\tilde{\phi}}}{2\tilde{G}_4}\frac{\tilde{B}_2}{\tilde{A}}\Bigg\} \tag{33}$$

As a reminder, quantities like M_{G3G4} denote the mass ratio M_{G3}/M_{G4}. The non-dynamical Friedmann equation is

$$1 = \Omega_m + \Omega_r + \Omega_\phi + \Omega_\Lambda, \tag{34}$$

where

$$\Omega_\phi = (\Omega_r + \Omega_m + \Omega_\Lambda)\left(\frac{M_{pG4}^2}{2G_4} - 1\right) + \frac{1}{3G_4}\left[\frac{M_{KG4}^2}{E^2}XK_X - \frac{M_{KG4}^2}{2E^2}K + 3M_{G3s}M_{sG4}X\phi'\tilde{G}_{3\tilde{X}}\right. \\ \left. - \frac{M_{G3G4}M_{sG4}}{E^2}XG_{3\phi} - 3\phi' G_{4\phi}\right] \tag{35}$$

which is used as a constraint equation to fix one of the initial conditions of the given reduced Horndeski model. Finally, for the equation of motion for the scalar field, first define

$$\tilde{A} = M_{Ks}^2\tilde{K}_{\tilde{X}} - M_{G3s}\tilde{G}_{3\tilde{\phi}} + 2\tilde{X}M_{G3s}\tilde{G}_{3\tilde{\phi}\tilde{X}} \\ + E^2\left(6\tilde{\phi}'\left[M_{G3s}\tilde{G}_{3\tilde{X}} + \tilde{X}M_{G3s}\tilde{G}_{3\tilde{X}\tilde{X}}\right] + \tilde{\phi}'^2\left\{M_{Ks}^2\tilde{K}_{\tilde{X}\tilde{X}} - 2M_{G3s}\tilde{G}_{3\tilde{\phi}\tilde{X}}\right\}\right) \tag{36}$$

and

$$B_1 = 6M_{G3s}\tilde{X}\tilde{G}_{3\tilde{X}} - 6M_{G4s}^2\tilde{G}_{4\tilde{\phi}} \tag{37}$$

$$B_2 = 3\tilde{\phi}'\left[M_{Ks}^2\tilde{K}_{\tilde{X}} - 2M_{G3s}\tilde{G}_{3\tilde{\phi}} + 2M_{G3s}\tilde{X}\tilde{G}_{3\tilde{\phi}\tilde{X}}\right] + (\tilde{\phi}')^2\left[M_{Ks}^2\tilde{K}_{\tilde{X}\tilde{\phi}} - 2M_{G3s}\tilde{G}_{3\tilde{\phi}\tilde{\phi}}\right] - \frac{M_{Ks}^2}{E^2}\tilde{K}_{\tilde{\phi}} \\ - 12M_{G4s}^2\tilde{G}_{4\tilde{\phi}} + 18M_{G3s}\tilde{X}\tilde{G}_{3\tilde{X}} + \frac{2M_{G3s}\tilde{X}\tilde{G}_{3\tilde{\phi}\tilde{\phi}}}{E^2} \tag{38}$$

so that the equation of the motion for the scalar field is

$$\phi'' = -\frac{\tilde{B}}{\tilde{A}} - \frac{E'}{E}\phi' \tag{39}$$

Along with these equations of motion we also use the evolution equations for the matter and radiation density parameters,

$$\Omega_m' = -\Omega_m \left(3 + \frac{2E'}{E}\right) \tag{40}$$

$$\Omega_r' = -\Omega_r \left(4 + \frac{2E'}{E}\right). \tag{41}$$

Therefore, the initial conditions to solve for the background are $\phi_0', \Omega_{m0}, \Omega_{m0}$, with $E_0 = 1$ by definition. In addition to these are any model parameters that are introduced in the definitions of K, G_3, G_4. Since eq. (34) is a constraint equation, it can be used to fix one of the initial conditions or model parameters. However, note that the initial conditions may be different if one opts to exploit a rescaling symmetry of a given reduced Horndeski model, in that this may lead to cases where one must specify the initial condition for the re-scaled Hubble function.

To give an example, one can consider a cubic Galileon (defined by $K, G_3 \propto X$ and $G_4 = 1/2$) with the additional constraint that it possesses a de Sitter-space limit at late times. This is achieved by demanding $E'(a_{\rm dS}) = \phi''(a_{\rm dS}) = 0$. By applying these conditions to our equations of motion we find 2 extra constraint equations, leading to a reduction of the number of model parameters. On top of this though, these de Sitter constraint equations also encourage a rescaling of the cubic Galileon parameters, and the variables for the Hubble function and the scalar field. Therefore, in this scenario one may no longer have the advantage of using E to represent the Hubble rate, and having its initial condition fixed by definition. See Section 2.3 of arXiv:2209.01666 for the details of this rescaling procedure.

For this reason, the names of the variables in the frontend `.ini` parameter files are generic, e.g. the initial condition for the Hubble function is called `Hubble0` as opposed to `E_0`, to account for the fact that one may choose to rescale quantities to forms different from those outlined here. It should be noted that ultimately the way in which the frontend 'knows' which rescaling is in use comes from what initial condition was supplied. We can say `Hi-COLA` used E for the Hubble rate if the value for `Hubble0` was set as 1, for instance.

Once the frontend has solved for the background quantities, the next step is to compute the pre-force quantities that make up the fifth force expression. We say pre-force because the fifth force has some density dependence, and this is something that must come from the backend during the hybrid N-body simulation. Thus, at this stage we are only able to compute the portion of the fifth force that depends only on the background. These quantities are β, the coupling factor, and χ/δ, the ratio of the screening factor and the overdensity. For a rundown of the steps to derive the fifth force expression, see section 3.1 of arXiv:2209.01666. The code equivalents are defined in the same manner and with the same names; the functions can be found in `expression_builder.py`. For example, the function `calB` is the same quantity as the calligraphic B, i.e. $\mathcal{B}$, defined in eq. 3.7. The key quantities required by the backend are a, E, $E'/E, \beta, \chi/\delta$. These are saved in text files and passed to the backend for use in the hybrid N-body simulation part of the `Hi-COLA` process.