MG-PICOLA Documentation

Hans Winther

June 21, 2018

Introduction

This code is based on the L-PICOLA code written by Cullan Howlett & Marc Manera. For a documentation of L-PICOLA see the L-PICOLA GITHub page. The code is still under development so it's a bit messy. I plan to clean it up and put up a clean and stable version in due time.

Compilation

Requires the FFTW3 library (needs to be compiled with –enable-float to use the SINGLE_PRECISION option).

Requires GSL (GNU Scientific Library)

Set desired options and the model in the parameterfile and run make. Alternatively compile as make Makefile MODEL=MYMODEL (MYMODEL = FOFR, DGP, BRANSDICKE, MBETA etc.). The code is run as mpirun -np 1 MG_PICOLA_MODEL paramfile.txt. See paramfiles for some example parameter-files. Note that the parmeterfile must contain all parameters asked for (and nothing else)!

Random notes

The scale-dependent version needs the define SCALEDEPENDENT. This version requires several Fourier transforms per time-step which makes the code 5 times slower.

The lightcone version of the code have not been tested, but should work fine for the scale-independent version of the case (i.e. using LCDM growth-factors).

Some optimizations should be done in the SCALEDEPENDENT versions with respect to Output(). Currently we recompute the LPT fields twice here. Should be changed.

When running with a modified model note that what the code calls ΛCDM is the cosmological model which has no modifications to the growth-rate of perturbations (i.e. not fifth-forces). However if the background is modified then this will be what we call

ΛCDM.

The original fitting functions for the ΛCDM growth-functions can be found as ***_LCDMFit** in **src/cosmo.c**.

The time-variable in the integration of growth-factors is $\mathrm{d}y = \frac{\mathrm{d}a}{E(a)a^3}$.

Functions of scale (Fourier wavenumber) require k to be in units of h/Mpc ($2\pi/B$ times the integer wave-number where B is the box-size).

Memory monitoring is done by wrapping around malloc and free. Use my_malloc to allocate and my_free to free memory to keep track of the memory we are using.

Modifying the code

If one wants to add a new model then one likely only needs to modify the file **src/user_defined_functions.h**. This requires the user to specify functions like $\mu(k,a) = \frac{G_{\rm eff}(k,a)}{G}$, the Hubble-function $E(a) = H(a)/H_0$ and it's derivatives, a screening method and screening-function (if the model has screening) plus adding relevant parameters to the code.

If one needs to do any initialization, computing different stuff in a new model then this can be done in the function **src/user_defined_functions.h::init_modified_version()**.

Parameter file

The code uses the same parameter-file as `PICOLA` with some additions. Note that even if some of the features below and not used the way the parameters are read requires them to be in the parameter-file. In this case the particular values does not matter so just add them with arbitrary values. The main flags are:

- **modified_gravity_active = 0, 1** Main flag to turn on or off the modifications of gravity.
- **include_screening = 0, 1** Main flag to turn on or off the screening method of Winther & Ferreira 2015.

Initial conditions

- **input_pofk_is_for_lcdm = 0, 1** The input power-spectrum is assumed to be for ΛCDM and we will use the MG growth-factor to rescale it.
- **input_sigma8_is_for_lcdm = 0, 1** Only used if input_pofk_is_for_lcdm = 1. Useful if one wants to do MG and ΛCDM simulations with exactly the same IC. The power-spectrum is normalized according to the linear ΛCDM model. This means the linear σ_8 for a MG run will be different than what is in the parameter-file.

- **inverted_initial_condition = 0, 1** We multiply $\delta(k)$ by -1 if 1.
- **amplitude_fixed_initial_condition = 0, 1** Use amplitude fixed IC. We enforce $|\delta(k)|^2 = P(k)$.

It has been shown that inverted and amplitude fixed IC does not lead to biased results, but significantly reduces cosmic variance in most clustering statistics. Suggested use: run pairs of simulations. For a given seed run one simulation with inverted IC = 0 and one simulation with IC = 1 (both with amplitude fixing turned on) and average observables over these two simulations.

The code can also read IC from file:

- **ReadParticlesFromFile = 0, 1** Main flag to read Ramses / Gadget snapshots and using this as the IC instead of generating the IC in the code. Useful to run COLA sim ulations with the same IC as previous full N-body simulations.
- **TypeInputParticleFiles = integer** Ramses is 1, ascii is 2 and gadget is 3.
- **NumInputParticleFiles = integer** How many Ramses / Gadget / Ascii files there are to read.
- **InputParticleFileDir = string** Path to the folder containing the snapshot.
- **InputParticleFilePrefix = string** Prefix of the particle-filenames, e.g. gadget in /InputParticleFileDir/gadget.0. Ignored for Ramses-files as we assume it's the standard 'part_0000X.out0000Y' format.
- **RamsesOutputNumber = integer** The number X in part_0000X.out0000Y. Ignored for ascii / gadget

Halo Finding

The code has the FoF halo-finder MatchMaker[1] written by David Alonso included. This has not been properly tested yet so use with care. It also require extra memory (duplicating pos,vel and ID). This is only an issue if we run without MEMORY_MODE. To use this option compile the code with the define MATCHMAKER_HALOFINDER and add the following options to the parameterfile:

- **mm_run_matchmaker = 0, 1** Main flag to turn on halo-finding. Will compute the halo-catalog every time we output.
- **mm_output_pernode = 0, 1** One output-file per node (1) or one file in total (0).
- **mm_output_format = 0, 1, 2** Ascii (0), Fits (1) or Binary (2). Fits require the cfitsio library plus being compiled with the define MATCHMAKER_USEFITS.
- **mm_min_npart_halo = int** The minimum number of particles in the FoF groups for we to define it a halo.

[1] See https://github.com/damonge/MatchMaker

- **mm_linking_length = float** The FoF linking length in terms of the mean inter-particle distance (0.2 is a commonly used value).

- **mm_dx_extra_mpc = float** Size of buffer region in the same units as the boxsize. 3.0 Mpc$/h$ is a safe value to use.

Power-spectrum evaluation

The code can do a simple power-spectrum evaluation. This requires the code to be compiled with the define COMPUTE_POFK. If this define is not set then the parameters below should not be in the parameter-file. If the values entered does not make sense the code will adjust this to the fiducial value. The computation is done right after we have computed $\delta(k)$ which is needed for forces so there is basically no cost associated with computing this.

- **pofk_compute_every_step = 0, 1** Main flag to turn on off $P(k)$ evaluation at every step.

- **pofk_bintype = integer** Linear spacing (0) or logarithmic spacing (1) of the bins. Put to 0 to use fiducial value [LINEAR]

- **pofk_nbins = integer** Number of bins between kmin and kmax. Put to 0 to use fiducial value [Nmesh]

- **pofk_kmin = float** Minimum wavenumber in h/Mpc. Fiducial value [0.0]

- **pofk_kmax = float** Maximum wavenumber in h/Mpc (should be smaller than $\sim \sqrt{3} \cdot 2\pi/\text{Box} \cdot \text{Nmesh}$). Put to 0 to use fiducial value [$2\pi/B \cdot$ Nmesh]

- **pofk_subtract_shotnoise = 0, 1** Flag to turn on or off shotnoise subtraction.

The code can also compute the first multipole moments P_0, P_2, P_4 of the redshift space power-spectrum in the global parallel plane approximation (we average over 2 axes) $P_\ell(k) = (2\ell+1)\left\langle L_\ell(\mu)|\delta(k)|^2\right\rangle$ where L_ℓ is the Legendre polynomial and $\mu = k_z/k = \cos\theta$. This requires 2 extra FFTs plus one temporary grid. For the scale-dependent version of the code, if the COLA method is in use, we require [P[i].dDdy] to contain dD/dy apposed to ΔD as needed for the time-stepping. This is to be able to add the COLA velocity field to the particles. This is availiable when we output so it's a bit cheaper to compute it there.

- **pofk_compute_rsd_pofk = 0, 1, 2** Main flag to turn on off RSD $P_\ell(k)$ evaluation. Compute RSD power-spectrum at every step (1), every time we output (2) or not at all (0).

NB: if we compute this at every step then the velocity are one half time-step behind the positions so there RSD $P(k)$ will not be 100% correct. If we compute when we output then everything is synchronized. In addition some power-spectrum estimation codes for post-processing of the data is included, see the SimplePofk folder.

Massive neutrinos

Include massive neutrinos (only for the scaledependent version of the code) by compiling with MASSIVE_NEUTRINOS and SCALEDEPENDENT. The current version requires CAMB (or similar, but read routines for other formats not inclued yet) transfer functions for massive neutrinos. Will update this with a version using growth-functions at a later stage.

- **nu_include_massive_neutrinos = 0, 1** Main flag to include massive neutrinos.
- **nu_sum_mass = float** The sum of neutrino masses in eV. This is translated into $\Omega_\nu = \frac{\sum m_\nu}{93.14 h^2 \mathrm{eV}}$ and the CDM+baryon density is put to $\Omega_{cb} = \Omega - \Omega_\nu$.
- **nu_transfer_info_file = string** Path to the info-file which again contains a list of filenames with redshifts to CAMB transfer-functions. These is a simple script in [camb_data] that can be used to run CAMB to generate the transfer-functions and generate this file.

NB: as the code is currently written the σ_8 value in the parameter-file needs to be computed from the CDM+baryon density and not the total matter density (the value CAMB outputs when transfer_power_var = 8).

w_0,w_a parametrization for background

The often used parametrization for the dark energy equation of state $w(a) = w_0 + w_a(1-a)$ is included in the code and is activated if we compile with the define EQUATIONOFSTATE_PARAMETRIZATION. For the parameter-file one must add:

- **w_0 = float** The value of w at $z = 0$.
- **w_a = float** The derivative $-dw/da$ at $z = 0$.

DGP model

The normal-branch DGP model with a ΛCDM background. The requires the code to be compiled with the define DGPGRAVITY and a choice of smoothing-filter GAUSSIANFILTER (standard), TOPHATFILTER or SHARPKFILTER.

- **Rsmooth = float** The radius in (Mpc$/h$) of the Fourier space smoothing kernel which we use to smooth the density field with for density-screening. The smoothing kernel itself is set in the Makefile (gaussian, tophat or sharp-k).
- **rcH0_DGP = float** The crossover-scale $r_c H_0/c$. Typical values for cosmological simulations are in the range $0.5 - 5$, i.e. $r_c = 1.5 - 15$ Gpc/h.

$f(R)$ gravity

This is the Hu-Sawicky $f(R)$ model. If true growth-factors then this requires the code to be compiled with the define SCALEDEPENDENT (in addition to FOFRGRAVITY). Otherwise use the flag **use_lcdm_growth_factors = 1**.

- **fofr0 = float** The value of f_R in the cosmological background today. Typical values for comological simulations are $10^{-4} - 10^{-6}$.
- **n_fofr = float** The value of n in the Hu-Sawicky $f(R)$ function. $n = 1$ is the fiducial value and the most commonly used in the literature.

$\{m(a), \beta(a)\}$ models

This is implemented in the code, and one should just have to specify $m^2(a)$, $dm^2(a)/da$ and $\beta(a)$ in **src/user_defined_functions.h** and add relevant parameters to the parameter-file to be able to use this. The integration of $\phi(a)$ needed for the screening method is done automatically (in **src/cosmo.c::compute_phi_of_a()**) however one should double check that this gives the correct results. This requires the code to be compiled with the define MBETAMODEL.

The example model implemented this way is the symmetron for which we need the following parameters:

- **assb_symm = float** The symmetry braking scalefactor (no modifications at all for $a < a_{\rm ssb}$)
- **range_symm = float** The range of the field in Mpc$/h$.
- **beta_symm = float** The coupling strength

Note that $f(R)$ discussed above can also be implemented this way.

Jordan-Brans-Dicke model

The constrained Jordan-Brans-Dicke model. When solving the background and scalar field equation we enforce $G_{\rm eff}/G = \frac{1}{\phi}\frac{4+3\omega_{\rm BD}}{4+3\omega_{\rm BD}}$ to be unity at $a = 1$. This is done by taking in physical density parameters and solving the background equations selecting the initial scalar field value so that $\phi(a = 1)$ has the desired value. The hubble parameter is a derived quantity. This requires the code to be compiled with the define BRANSDICKE.

- **wBD = float** The JBD parameter.
- **Omegah2 = float** The physical matter density parameter
- **Omegarh2 = float** The physical radiation density parameter
- **Omegavh2 = float** The physical dark energy density parameter

Note that the Omega and HubbleParam in the parameter-file is ignored and recalculated from the parameters above.

Other important things

- The non-gaussian initial condition generations has not been tested and might require additional modifications in general (especially wrt.the transfer-function that is currently used here ; this will for sure have to be changed for massive neutrinos).
- The lightcone option should work well for scaleindependent growth. The modifications needed for scale-dependent growth has not been implemented (and this would likely have to be done approximately).