The documentation of GDSimulation code for doing lattice simulations of the evolution of scalar fields in an expanding Universe

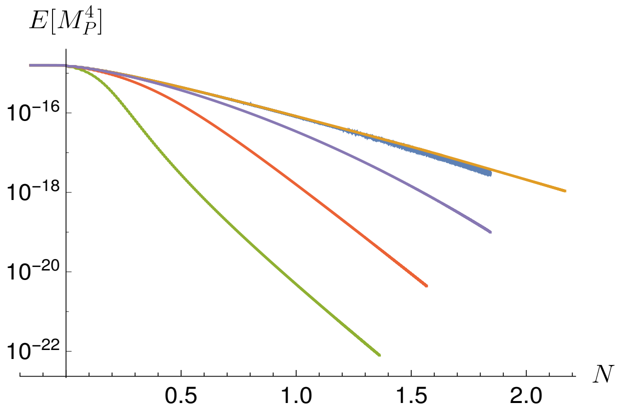
Michał Wieczorek

Contents

1 Overview

GDSimulation is a python program for doing lattice simulations of the evolution of scalar fields in an expanding Universe. The program enables to perform the simulations for the fields-space manifolds characterized by the non-zero curvature. Up to our knowledge, it is the first program with this feature, which uses the symplectic numerical method. The usage of such method is crucial to preserve the high accuracy of the obtained solution in case of very long simulations with the number of time steps to be of order 10^6 or higher.

There are many computer codes written to track the fields evolution during or after inflation on the lattice (see e.g [1] for a recent review). However, most of them can be applied only to models with canonical kinetic term in the Lagrangian (e.g. [2, 3, 4]). One exception is GABE [5]. This code is designed to make simulations for models with non-canonical kinetic terms. However, its time evolution scheme is based on the Runge-Kutta method, which is not symplectic and turned out to generate huge errors in the long simulations. Using a non-symplectic method significantly decreases accuracy of the long-time simulations and spoils energy conservation. To advocate this statement on Graph 1 we present the plot of energy density of inflaton after the end of inflation for α-attractor single-field T-model. We compare the evolution of this density in homogeneous approximation (i.e. using the homogeneous initial conditions) obtained with highly accurate *Mathematica* ODE solver, with symplectic *Lattice Easy* [3] solver and with modified *Lattice Easy* solver using Runge Kutta method for different values of time step. The obtained results show how important it is to use the symplectic method.



Graph 1: Comparison of the energy density evolution in homogeneous approximation obtained with highly accurate *Mathematica* ODE solver, with symplectic *Lattice Easy* solver and with modified *Lattice Easy* solver using Runge Kutta method for different values of time step.

The original motivation for creating the *GDSimulation* program was to simulate the geometrical destabilization of inflation at the very end of inflationary epoch for two fields α-attractor models. Such simulations were done for the first time (the results are described

in [6]) and enabled to show, that for small values of α the geometrical destabilization mixed with the parametric resonance mechanism leads to instantaneous reheating - the very desirable behavior for inflationary models.
Later the program was used to simulate more general case of geometrical destabilization for single field inflationary models, which are coupled to the second scalar field via the kinetic interaction term in Lagrangian. Again, for the first time it enabled to obtain the precise quantitative description of this phenomenon.

The potential applicability of *GDSimulation* code considerably exceeds the described research. Because of that we decided to make it publicly available. The program can be downloaded from the following link of GitHub repository. It can be freely used or modified as long as you give us credit by citing our GitHub page.

The content of this document is organized as follows. In chapter 2 we provide the necessary informations to use the program *GDSimulation.* In chapter 3 we describe the details of the used numerical method. Finally in chapter 4 we show the discretization technique crucial for the proper initialization of the quantum perturbations of scalar fields and describe the way we set those initial conditions in our program.

2 How to use the *GDSimulation* code

2.1 The general form of the Einstein-Hilbert action, for which *GDSimulation* code can be used.

To use our code for your own research project, it is necessary to know, when this code can be used. In this subsection we present the general form of the problem, which our program can potentially address.

The general form of the Einstein-Hilbert action, for which *GDSimulation* code can be used without significant modifications has the following form.

$$S = \int d^4x\sqrt{-g}\left[\frac{1}{2}\mathcal{R} - \frac{1}{2}f(\chi)(\partial_\mu\phi)(\partial^\mu\phi) - \frac{1}{2}(\partial_\mu\chi)(\partial^\mu\chi) - V(\phi,\chi)\right]. \tag{1}$$

In the program we assumed that the space-time is spatially homogeneous, isotropic and flat[1], i.e. we have:

$$ds^2 = a^2(-d\tau^2 + d\mathbf{x}^2), \tag{2}$$

where a is the scale factor describing the expansion of the Universe. For the determinant of the metric g and the Rici scalar $\mathcal{R}$ it implies:

$$\sqrt{-g} = a^4 \qquad \text{and} \qquad \mathcal{R} = 6\frac{a''}{a^3}. \tag{3}$$

[1]In Ref. [4], it is explained why this assumption is a good approximation even in case of very effective fragmentation of the inflaton condensate.

The function $f(\chi)$, while being nontrivial, makes the fields-space curvature to be nonzero. In principle it would be easy to adjust the code for the slightly more general action:

$$S = \int d^4x\sqrt{-g}\left[\frac{1}{2}\mathcal{R} - \frac{1}{2}f(\chi)(\partial_\mu\phi)(\partial^\mu\phi) - \frac{1}{2}h(\phi)(\partial_\mu\chi)(\partial^\mu\chi) - V(\phi,\chi)\right]. \qquad (4)$$

In practice however, by suitable fields transformation it is usually possible to make the kinetic term of one of the fields to have the canonical form and we found such generalization to be unnecessary complication.
Nevertheless it could have the potential value to enable function f to depend on ϕ. In this case however, the lack of generality has deeper reason. It turns out that if f would be ϕ-dependent, the proposed numerical method would be non-symplectic and there does not exist the known symplectic method in such case.

2.2 Defining your own model.

There are only two files, which should be modified by the user - the configuration file and the file *Model.py* placed in folder */gd_ simulation.* In this subsection we will describe the second of these files, where the model is defined.
This file contains the class *Model* with a bunch of static methods. Those methods are the functions defining the potential of your model, its first and second derivatives as well as the function $f(\chi)$ and its derivative. The method defining function f is called *non_ canonical_ multiplier.*
Apart of the *Model* class the parameters specific to the model are assigned above the class definition. The list of these parameters is defined in the configuration file (more about it in the next subsection). All these parameters should be multiplied by the appropriate power of the Planck mass parameter (MPl) depending on their mass dimension.

2.3 Defining parameters for the simulation.

The file, where all the parameters are defined is the configuration file. The example of this file is placed in folder */configs* and is called *ExemplaryConfig.* The parameters in these file are grouped according to their functionality and properties.

The first group are *cut_ off_ dependent* parameters. This group consists of the following parameters:

cut-off - it is used only to write down the informations about the particular run of the simulation and therefore you can provide it in any units you want.

rescaling_ parameter - it is the value of Planck Mass in terms of units used in the program. Those units are chosen in such a way, that the lattice spacing (i.e. the minimal distance between two points in the lattice) is equal 1. Since in our program the cut-off is identical to the biggest wave number possibly realized in the given lattice, the lattice spacing directly corresponds to the cut-off. Therefore also the chosen units are cut-off dependent and the value of Planck Mass in those units has to be precomputed before performing the simulations for the particular cut-off. After reading the section 4, it should be clear, how the lattice spacing and the value of cut-off are related.

The second group of the parameters are the *model_ parameters*. All these parameters should be provided in the appropriate power of Planck units depending on their mass dimension. They are automatically rescaled before the simulation. The first four parameters in this group are the initial background homogeneous values of fields and their velocities. So far we used the program only in case of vanishing homogeneous initial conditions for field χ. It should work however in the general case.
The fifth parameter is scale factor, with we advise to initialize with unity.
Finally, in the *model_ parameters* the subgroup of *specific_ model_ parameters* is defined. Those are all the parameters specific to the considered model.

The third group of parameters are the *error_ analysis_ parameters*. The error analysis is based on comparing the solution obtained with the 2nd order and 4th order integrator. The parameters in this group are:

τ_1 - We advise to set it to be of order 10^{-4} or smaller and not greater than the inverse of the number of time steps. The precise value depends strongly on the instability of the system for the given model. In general decreasing this parameter make the computation more accurate and more time consuming.

τ_2 - should be smaller than the quarter of τ_1 but greater than $\tau_1/20$ to avoid unnecessary granulation in time.

The last group of parameters are *technical_ parameters*. Those are the following parameters:

continuation - can be True or False depending weather we want to run the new simulation or continue the old one (in such case the initial conditions are created based on the results of the previous simulation and the new results are appended to the old files while appropriate).

N - the linear lattice size.

steps_ number - the number of time steps.

startIter - it is used only if *continuation=True* and denotes the number of step of the previous simulations from which the new simulations should start. By default the data, which enables the continuation of simulation is written every 10000 time steps, hence *startIter* should be the multiplication of 10000 (this number can be changed in the code in the function Integrator.integrate()).

initial_ time_ step - the initial time step. We advise to set it to be of order 10^{-4} or smaller. To small initial time step will be changed quickly due to error analysis, which increases time step if it is unnecessary to keep it small.

output_ dir_ path - it defines the path to the directory, where you want to have the files with the output.

write_ separate_ potential_ energy - it can be set to True if the potential of the considered model can be written as a sum of its part depending on one field and the part depending on the second field (it should be expressed accordingly in the file *Model.py*). In this case the evolution of two part of the potential energy will be written separately, which will enable the better analysis of the results. This parameter should be *False*, if the potential is not separable in the described way.

special_comment - a string added to the names of output files which do not depend on parameter

parameters_included_in_files_names - the names of parameters (grouped according to their placement in configuration file) whose values should be included in the names of output files (it will be done automatically)

2.4 Output files

The results of the simulation are saved in many different files. The names of all these files are designed to provided the information about their content and the parameters used in the particular simulation (it is defined in configuration file which parameters are the parts of the names of the files). During the simulation, the following output files are produced:

chiMagnitudeData_... - the file with the average of absolute values of field χ (second column in file) for every time step as a function of scale factor (first column in a file). The values of fields are provided in program units (they have to be divided by Planck mass in the same units to obtain the values in Planck units).

dtList_... - the files with the times steps used during the simulations (it can be useful for error analysis and appropriate adjustment of error bound)

EfoldsBoxPlotList_... - the file with the e-folds number since the beginning of the simulations at which the fields and energy density values for every point in the lattice are saved.

energyData_... - the file with different components of energy. It has five or six columns depending if we want to have the values of ϕ-dependent and χ-dependent parts of potential energy density divided or saved as a sum. Different variables are saved in the columns in the following order: scale factor, ϕ kinetic energy density, χ kinetic energy density, ϕ gradient energy density, χ gradient energy density, ϕ potential energy density and χ potential energy density. The last to columns can be replaced by the overall potential energy density if required. The values of the energy densities are provided in the Planck units (i.e. Planck mass to the 4th power).

finalaPiaData_... - the files with scale factor and its conjugate momentum. The part of the names of these files is the time step number for which they were saved. Those files are used to continue the simulation from this time step.

finalChiFieldData_... - the files with values of field χ for every point in the lattice for the given time step. The part of the names of these files is the time step number for which they were saved. Those files are used to continue the simulation from this time step.

finalChiPiFieldData_... - the files with values of the conjugate momentum of field χ for every point in the lattice for the given time step. The part of the names of these files is the time step number for which they were saved. Those files are used to continue the simulation from this time step.

finalPhiFieldData_... - the files with values of field ϕ for every point in the lattice for the given time step. The part of the names of these files is the time step number for which they were saved. Those files are used to continue the simulation from this time step.

finalPhiPiFieldData_... - the files with values of the conjugate momentum of field ϕ for every point in the lattice for the given time step. The part of the names of these files is the time step number for which they were saved. Those files are used to continue the simulation from this time step.

PeriodicChiFieldData_... - the files with values of field χ for every point in the lattice for the given e-folds number after the start of the simulation. The part of the names of these files is the value of e-folds number for which they were saved. Those files are saved more often than the final state files and are used to investigate the space-time evolution of fields during simulation.

PeriodicPhiFieldData_... - the files with values of field ϕ for every point in the lattice for the given e-folds number after the start of the simulation. The part of the names of these files is the value of e-folds number for which they were saved. Those files are saved more often than the final state files and are used to investigate the space-time evolution of fields during simulation.

PeriodicEnergyData_... - the files with values of the overall energy density for every point in the lattice for the given e-folds number after the start of the simulation. The part of the names of these files is the value of e-folds number for which they were saved. Those files are saved more often than the final state files and are used to investigate the space-time evolution of energy density during simulation.

2.5 Running the simulation

To run the simulation from the command line in the application directory *gd_simulation* one has to type:

```
python app.py path_to_your_configuration_file
```

where you should provide the path to your configuration file.

2.6 Making plots

To analyze the results obtained in the simulations, in our repository we provide the exemplary script for making the animation of the time evolution of field χ on the two dimensional intersection of the original lattice. The script for plotting other quantities can be created analogously. To create the animation from the command line in the *PlotsMaking* directory one has to type:

```
python AnimationDensityPlot.py path_to_your_configuration_file
```

where you should provide the path to the configuration file that you used for the simulation. The animation is saved in the same output directory as the data from simulation.

2.7 Power spectra

In our repository we provide the script for computation of power spectra of both fields. To run the script from the command line in the *Postprocessing* directory one has to type:

```
python PowerSpectrum.py path_to_your_configuration_file
```

where you should provide the path to the configuration file that you used for the simulation. The files with power spectra for both fields and for all e-folds numbers for which the complete values of fields on the lattice were saved are created. Those files are saved in the same output directory as the data from simulation. Each of these files consists of two columns: the wave number k (provided in Planck units) and the power spectrum (i.e. $|\phi_k|^2$ and $|\chi_k|^2$) for fields ϕ and χ, respectively.

3 Description of the method used in the program

3.1 The discretized action and Hamiltonian

Our goal is to solve numerically equations which come from the action

$$S = \int d^4x\sqrt{-g}\left[\frac{1}{2}\mathcal{R} - \frac{1}{2}f(\chi)(\partial_\mu\phi)(\partial^\mu\phi) - \frac{1}{2}(\partial_\mu\chi)(\partial^\mu\chi) - V(\phi,\chi)\right]. \tag{5}$$

As noted previously, we assume here that the space-time is spatially homogeneous, isotropic and flat, i.e. we have

$$ds^2 = a^2(-d\tau^2 + d\mathbf{x}^2), \tag{6}$$

which implies

$$\sqrt{-g} = a^4 \qquad \text{and} \qquad \mathcal{R} = 6\frac{a''}{a^3}. \tag{7}$$

After discretization in space the action (5) can be written as:

$$\begin{aligned} S &= (dx)^3\int \mathcal{L}d\tau = \\ &= (dx)^3\int\Bigg[-3a'^2V_L + \sum_{\vec{x}}\frac{a^2}{2}\Bigg(f(\chi_{\vec{x}})\Bigg((\phi'_{\vec{x}})^2 - \frac{G(\phi,\vec{x})}{(dx)^2}\Bigg) + \\ &+ \Bigg((\chi'_{\vec{x}})^2 - \frac{G(\chi,\vec{x})}{(dx)^2}\Bigg) - a^2V(\phi_{\vec{x}},\chi_{\vec{x}})\Bigg)\Bigg]d\tau, \end{aligned} \tag{8}$$

where $(dx)^3V_L$ equals the volume of the periodic lattice and

$$G(Y,\vec{x}) = \frac{1}{2}\sum_{x_1-1}^{x_1+1}\sum_{x_2-1}^{x_2+1}\sum_{x_3-1}^{x_3+1} c_{d(\alpha)}(Y_\alpha - Y_0)^2 \tag{9}$$

is the second order discretization of the squared spatial gradient operator

$$(\nabla Y)^2(\vec{x}) \simeq \frac{G(Y,\vec{x})}{(dx)^2} \tag{10}$$

with $c_1 = 1$ and $c_0 = -6$ (see [4]).
After the Legendre transformation we obtain the following Hamiltonian density:

$$\mathcal{H} = -\frac{p_a^2}{12V_L} + \sum_{\vec{x}} a^4 \left(\frac{\pi_{\phi,\vec{x}}^2}{2a^6 f(\chi_{\vec{x}})} + \frac{\pi_{\chi,\vec{x}}^2}{2a^6} + f(\chi_{\vec{x}}) \frac{G(\phi,\vec{x})}{2(dx)^2 a^2} + \frac{G(\chi,\vec{x})}{2(dx)^2 a^2} + V(\phi_{\vec{x}}, \chi_{\vec{x}}) \right), \tag{11}$$

where canonical momenta are defined by the formulae

$$p_a \equiv \frac{\partial \mathcal{L}}{\partial a'} = -6a' V_L, \quad \pi_{\phi,\vec{x}} \equiv \frac{\partial \mathcal{L}}{\partial \phi'} = a^2 f(\chi_{\vec{x}}) \phi'_{\vec{x}} \quad \text{and} \quad \pi_{\chi,\vec{x}} \equiv \frac{\partial \mathcal{L}}{\partial \chi'} = a^2 \chi'_{\vec{x}}. \tag{12}$$

3.2 Discretization scheme

The time upgrade scheme is based on the fact that we can divide this Hamiltonian into four parts

$$\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2 + \mathcal{H}_3 + \mathcal{H}_4 \tag{13}$$

in such the way that none of the parts depends on both the field and its canonical momentum. The fact, that such division is possible is crucial for construction of symplectic method. An example of such division is

$$\mathcal{H}_1 \equiv -\frac{p_a^2}{12V_L}, \tag{14}$$

$$\mathcal{H}_2 \equiv \sum_{\vec{x}} a^4 \left(\frac{\pi_{\phi,\vec{x}}^2}{2a^6 f(\chi_{\vec{x}})} \right), \tag{15}$$

$$\mathcal{H}_3 \equiv \sum_{\vec{x}} a^4 \left(\frac{\pi_{\chi,\vec{x}}^2}{2a^6} \right) \tag{16}$$

and

$$\mathcal{H}_4 \equiv \sum_{\vec{x}} a^4 \left(f(\chi_{\vec{x}}) \frac{G(\phi,\vec{x})}{2(dx)^2 a^2} + \frac{G(\chi,\vec{x})}{2(dx)^2 a^2} + V(\phi_{\vec{x}}, \chi_{\vec{x}}) \right). \tag{17}$$

To solve the Hamiltonian system numerically, we define for time step $h = \delta\tau$ the transformations

$$\Phi_1(h) : \left(a, p_a, \phi_{\vec{x}}, \pi_{\phi,\vec{x}}, \chi_{\vec{x}}, \pi_{\chi,\vec{x}} \right) \rightarrow \left(a + \frac{\partial \mathcal{H}_1}{\partial p_a} h, p_a, \phi_{\vec{x}}, \pi_{\phi,\vec{x}}, \chi_{\vec{x}}, \pi_{\chi,\vec{x}} \right), \tag{18}$$

$$\Phi_2(h) : \left(a, p_a, \phi_{\vec{x}}, \pi_{\phi,\vec{x}}, \chi_{\vec{x}}, \pi_{\chi,\vec{x}} \right) \rightarrow \left(a, p_a - \frac{\partial \mathcal{H}_2}{\partial a} h, \phi_{\vec{x}} + \frac{\partial \mathcal{H}_2}{\partial \pi_{\phi,\vec{x}}} h, \pi_{\phi,\vec{x}}, \chi_{\vec{x}}, \pi_{\chi,\vec{x}} - \frac{\partial \mathcal{H}_2}{\partial \chi_{\vec{x}}} h \right), \tag{19}$$

$$\Phi_3(h):\Big(a,p_a,\phi_{\vec{x}},\pi_{\phi,\vec{x}},\chi_{\vec{x}},\pi_{\chi,\vec{x}}\Big)\to\Big(a,p_a-\frac{\partial\mathcal{H}_3}{\partial a}h,\phi_{\vec{x}},\pi_{\phi,\vec{x}},\chi_{\vec{x}}+\frac{\partial\mathcal{H}_3}{\partial\pi_{\chi,\vec{x}}}h,\pi_{\chi,\vec{x}}\Big) \tag{20}$$

and

$$\Phi_4(h):\Big(a,p_a,\phi_{\vec{x}},\pi_{\phi,\vec{x}},\chi_{\vec{x}},\pi_{\chi,\vec{x}}\Big)\to\Big(a,p_a-\frac{\partial\mathcal{H}_4}{\partial a}h,\phi_{\vec{x}},\pi_{\phi,\vec{x}}-\frac{\partial\mathcal{H}_4}{\partial\phi_{\vec{x}}}h,\chi_{\vec{x}},\pi_{\chi,\vec{x}}-\frac{\partial\mathcal{H}_4}{\partial\chi_{\vec{x}}}h\Big). \tag{21}$$

Then the first order symplectic numerical method (see Ref. [7]) has the form

$$\tilde{\Phi}(h)=\Phi_4(h)\circ\Phi_3(h)\circ\Phi_2(h)\circ\Phi_1(h). \tag{22}$$

With that and its adjoint (again, see Ref. [7]) we can create the second order symplectic method for this system, namely

$$\Phi(h)\equiv\tilde{\Phi}^*(h/2)\circ\tilde{\Phi}(h/2)=\Phi_1(h/2)\circ\Phi_2(h/2)\circ\Phi_3(h/2)\circ\Phi_4(h)\circ\Phi_3(h/2)\circ\Phi_2(h/2)\circ\Phi_1(h/2). \tag{23}$$

It can be written as the following set of explicit upgrades:

$$a_{n+1/2}=a_n+\frac{h}{2}\frac{\partial\mathcal{H}_1}{\partial p_a}(p_{a,n}) \tag{24}$$

$$\tilde{p}_{a,n+1/2}=p_{a,n}-\frac{h}{2}\frac{\partial\mathcal{H}_2}{\partial a}(a_{n+1/2},\pi_{\phi,n},\chi_n) \tag{25}$$

$$\tilde{\pi}_{\chi,\vec{x},n+1/2}=\pi_{\chi,\vec{x},n}-\frac{h}{2}\frac{\partial\mathcal{H}_2}{\partial\chi_{\vec{x}}}(a_{n+1/2},\pi_{\phi,n},\chi_n) \tag{26}$$

$$\phi_{\vec{x},n+1/2}=\phi_{\vec{x},n}+\frac{h}{2}\frac{\partial\mathcal{H}_2}{\partial\pi_{\phi,\vec{x}}}(a_{n+1/2},\pi_{\phi,n},\chi_n) \tag{27}$$

$$\tilde{\tilde{p}}_{a,n+1/2}=\tilde{p}_{a,n+1/2}-\frac{h}{2}\frac{\partial\mathcal{H}_3}{\partial a}(a_{n+1/2},\tilde{\pi}_{\chi,n+1/2}) \tag{28}$$

$$\chi_{\vec{x},n+1/2}=\chi_{\vec{x},n}+\frac{h}{2}\frac{\partial\mathcal{H}_3}{\partial\pi_{\chi,\vec{x}}}(a_{n+1/2},\tilde{\pi}_{\phi,n+1/2}) \tag{29}$$

$$\tilde{\tilde{p}}_{a,n+1}=\tilde{\tilde{p}}_{a,n+1/2}-h\frac{\partial\mathcal{H}_4}{\partial a}(a_{n+1/2},\phi_{n+1/2},\chi_{n+1/2}) \tag{30}$$

Note that this procedure needs only one array of numbers for every variable and for its associated canonical momenta. These variables are modified consecutively.
Based on this upgrade scheme we can construct higher order methods [2]. The simulations, which results are shown in this paper used the 4-th order integration scheme with:

$$\Phi_{4\text{th}}(h)=\Phi(z_1h)\circ\Phi(z_0h)\circ\Phi(z_1h), \tag{31}$$

where

$$z_0\equiv\frac{-2^{1/3}}{2-2^{1/3}}\quad\text{and}\quad z_1\equiv\frac{1}{2-2^{1/3}}. \tag{32}$$

4 The initial conditions for perturbations

4.1 The Fourier transform on the lattice

While making the lattice simulations we have to deal with the representation of the infinite space by the finite number of points. What is more, the analytic form of the initial conditions is provided in Fourier space and has to be dicretized as well.
To present and explain the way it affects the form of the power spectrum, let us begin with the definition of the discrete Fourier transform. The usual Fourier transform is usually defined for square integrable function $f : \mathbb{R} \to \mathbb{C}$:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x\xi} dx, \qquad f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i x\xi} d\xi. \tag{33}$$

One motivation behind the Fourier transform comes from Fourier series. Namely if f is periodic, one can write:

$$f(x) = \frac{1}{L} \sum_{k \in \frac{2\pi}{L}\cdot\mathbb{Z}} e^{ikx} \hat{f}_k \tag{34}$$

$$f_k = \frac{1}{L} \int_0^L e^{-ikx} f(x) dx \tag{35}$$

If we make f periodic (or with compact support), it implies discretization in k-space. And vice-versa, if we make discretization in k-space, it implies periodicity (or compact support) in x-space. In case of lattice simulations we have both - the discretization in k-space and in x-space. Therefore in such simulations we take into account the finite number of the momentum modes uniformly distributed in k-space with the implicit cut-off imposed by the spatial discretization.
To be more specific, for 3-dimensional cubic lattice with N^3 uniformly distributed points, length L and lattice spacing $h \equiv \frac{L}{N}$, we get (in 3D) the following approximation of the equation (35):

$$f_{\bar{k}} = h^3 \sum_{\bar{x}} e^{-i\bar{k}\bar{x}} f_{\bar{x}}, \qquad \bar{x} = h\overline{(j_1, j_2, j_3)}, \quad j_i \in \{1, ..., N\}, \quad h = \frac{L}{N} \tag{36}$$

and the analogue of the equation (34):

$$f_{\bar{x}} = \frac{1}{L^3} \sum_{\bar{k}} e^{i\bar{k}\bar{x}} f_{\bar{k}}, \qquad \bar{k} = \overline{(k_1, k_2, k_3)}, \quad k_i \in \frac{2\pi}{L}\{-\frac{N}{2} + 1, ..., -1, 0, 1, ..., \frac{N}{2}\} \tag{37}$$

It gives us the momentum cutoff: $k_{\text{max}} = \frac{\sqrt{3}\pi}{h}$. Very often, while doing the numerical simulations, it is computationally efficient to have $h = 1$ (it simplifies the gradient computation and Fourier transforms). Therefore for given physical cutoff k_{max} we have to write all dimensional quantities (i.e. fields, there derivatives etc.) in such units, that $k_{\text{max}} = \sqrt{3}\pi$. Then we obtain $N = L$ and the equations (36) and (37) simplify to:

$$\hat{f}_{\bar{k}} = \sum_{\bar{x}} e^{-i\bar{k}\bar{x}} f_{\bar{x}}, \qquad \bar{x} = \overline{(j_1, j_2, j_3)}, \quad j_i \in \{1, ..., N\}, \quad h = \frac{L}{N} \tag{38}$$

$$f_{\bar{x}} = \frac{1}{N^3}\sum_{\bar{k}} e^{i\bar{k}\bar{x}} f_{\bar{k}}, \qquad \bar{k} = \overline{(k_1, k_2, k_3)}, \quad k_i \in \frac{2\pi}{N}\{-\frac{N}{2}+1, ..., -1, 0, 1, ..., \frac{N}{2}\} \tag{39}$$

Those two equations are in agreement with the formulations of discrete Fourier transforms for example in python (see [8]). For codes that do not use units such that $h = 1$, one has to adjust the interpretation of f_x and f_k, which are provided by these discrete transforms. In the next chapter we will write h explicitly to track the general rescaling of dimensional quantities.

4.2 The primordial perturbations in discrete setup

Now, let us begin to think about f_x and f_k (from now on we will write k and x instead of $\bar{k}$ and $\bar{x}$) as about field perturbation and its Fourier component respectively. For simplicity let us focus on the sub-horizon modes. For the other modes the analysis is analogous. In continuum setup the initial conditions would be characterized by $|f_k| = \frac{1}{\sqrt{2k}}$. It turns out that it is not true in dicretized space. Let us explain why.
The origin of the form of the sub-horizon modes power spectrum comes from the quantum nature of the perturbations. For such modes the field perturbation can be expressed as the operator in an infinite space:

$$\hat{f}(x) = \frac{1}{\sqrt{(2\pi)^3}}\int \frac{d^3k}{\sqrt{2|k|}}\left(\hat{a}^-_k e^{-i(|k|t-\bar{k}\bar{x})} + \hat{a}^+_k e^{i(|k|t-\bar{k}\bar{x})}\right); \quad [\hat{a}^-_k, \hat{a}^+_{k'}] = \delta(k-k') \tag{40}$$

One of the physically meaningful quantities, that can be computed now is:

$$\langle 0|(\hat{f}(x))^2|0\rangle = \frac{1}{2(2\pi)^3}\int d^3k \int d^3k' \frac{1}{|kk'|}\delta(k-k') = \frac{4\pi}{2(2\pi)^3}\int k dk = \frac{1}{2}\frac{k^2_{max}}{(2\pi)^2} \tag{41}$$

Now, let us try to compute the same quantity in the (random) discrete case, assuming that $\langle |f_k|^2\rangle = \frac{1}{2k}$. By discrete Perceval identity:

$$\sum_{\bar{x}} |f_x|^2 = \frac{1}{N^3h^6}\sum_{\bar{k}} |f_k|^2 \tag{42}$$

the following holds:

$$\begin{aligned} \langle |f_x|^2\rangle &= \frac{h^3}{L^3}\sum_{\bar{x}} |f_x|^2 = \frac{1}{L^3N^3h^3}\sum_{\bar{k}} |f_k|^2 \approx \frac{4\pi}{L^3N^3h^3}\sum_{l=1}^{N/2} l^2 |f_{\frac{2\pi}{L}l}|^2 = \\ &= \frac{4\pi}{L^3N^3h^3}\frac{L}{2\pi}\sum_{l=1}^{N/2}\frac{l^2}{2l} \approx \frac{1}{N^2h^2L^3}\frac{N}{2}\frac{N}{4} = \frac{1}{8h^2L^3} = \frac{1}{2L^3}\frac{k^2_{max}}{(2\pi)^2} \end{aligned} \tag{43}$$

Above, the first approximate equality is true up to some constant of order 1, since instead of the sum over the cube, we take the sum over the ball of radius k_{max}. The result of

(43) is different from the result (41) by the factor of L^3.

Now comes the important assumption: *We want to make $\langle |f_x|^2 \rangle$ to be independent of L and N and we want the result of (43) to be the same as (41).*
Therefore in discrete setup we assume $|f_k| = \frac{L^{3/2}}{\sqrt{2k}}$. This assumption is consistent with the initial conditions, which are provided in LatticeEasy as well as those provided in DEFROST. The obvious consequence of the modification of f_k, is that taking the inverse Fourier transform of the initial conditions for field f_x, we expect to get the power spectrum close to the function $|f_k|^2 = \frac{L^3}{2k}$ instead of $|f_k|^2 = \frac{1}{2k}$.

4.3 The adiabatic approximation for the initial conditions of perturbations

In our simulations not all of the modes are sub-horizon. Therefore in the program we set the initial conditions which are based on the adiabatic approximation:

$$f_\phi(k,\tau_0) = \frac{1}{\sqrt{2\omega_{\phi,k}}}\exp^{-2i\pi\alpha}, \qquad f'_\phi(k,\tau_0) = -i\sqrt{\frac{\omega_{\phi,k}}{2}}\exp^{-2i\pi\alpha} \tag{44}$$

and

$$f_\chi(k,\tau_0) = \frac{1}{\sqrt{2\omega_{\chi,k}}}\exp^{-2i\pi\beta}, \qquad f'_\chi(k,\tau_0) = -i\sqrt{\frac{\omega_{\chi,k}}{2}}\exp^{-2i\pi\beta}, \tag{45}$$

where

$$\omega^2_{\phi,k} \equiv k^2 + a^2\Big(V_{\phi\phi} + 2H^2\Big), \quad \omega^2_{\chi,k} \equiv k^2 + a^2\Big(V_{\chi\chi} + 2H^2 + \frac{1}{2}\dot{\phi}^2\mathbb{R}\Big) \tag{46}$$

and α and β are the random phases. In the definition of $\omega^2_{\chi,k}$ by $\mathbb{R}$ we denoted the fields space curvature, which in our case equals to the second derivative of the non-canonical multiplier of kinetic term $f(\chi)$ (that is, why you should provide this function in the file *Model.py*).
The performed tests suggest, that the obtained results are not affected by the inaccuracy of the adiabatic approximation, especially, if the majority of the considered modes are initially sub-horizon.

4.4 The precise form of the initial conditions for perturbations used in the code

We initialize the perturbations to be the Gaussian random variables, which is consistent with the negligible amount of predicted non-gaussianities [9], [10]. We constrain the modes for $\bar{k}$ and $-\bar{k}$ to be conjugate to each other. This way in spatial coordinates we obtain the real initial conditions. The procedure we used to set initial conditions in our program is the same as in *LatticeEasy* [3]. To be more precise we set:

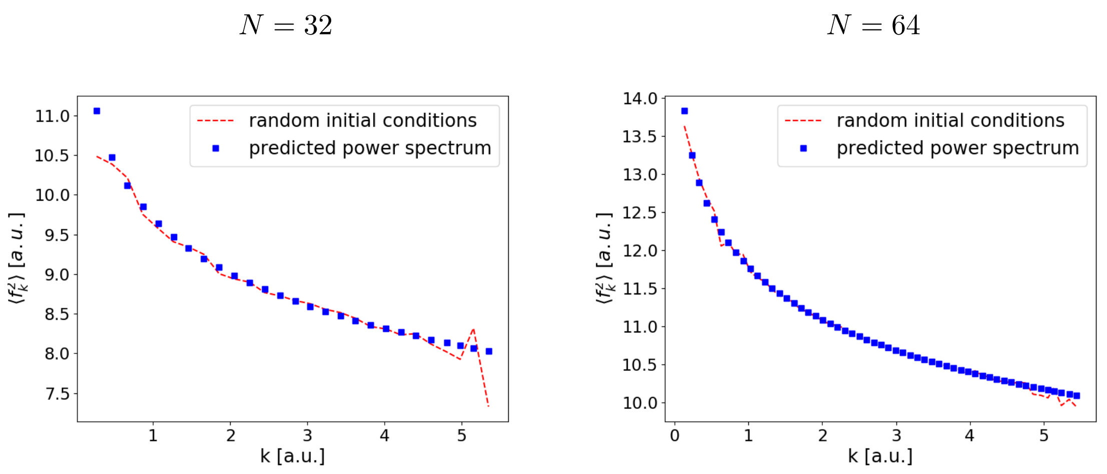
$$f_k(\tau_0) \equiv \frac{N^{3/2}}{2\sqrt{\omega_k}}\Big[\sqrt{-\ln X_1}\exp(2i\pi\theta_1) + \sqrt{-\ln X_2}\exp(2i\pi\theta_2)\Big] \tag{47}$$

and

$$f'_k(\tau_0) \equiv \frac{i\omega_k N^{3/2}}{2\sqrt{\omega_k}} \left[\sqrt{-\ln X_1} \exp(2i\pi\theta_1) - \sqrt{-\ln X_2} \exp(2i\pi\theta_2) \right], \tag{48}$$

where $X_1, X_2, \theta_1, \theta_2$ are independent random variables with uniform probability distribution over the interval $[0,1]$. The factor $N^{3/2}$ comes from the rescaling we have discussed. Replacing f_k by the fields initial conditions $f_\phi(k)$ and $f_\chi(k)$ we substitute $\omega_k \equiv \omega_{\phi,k}$ and $\omega_k \equiv \omega_{\chi,k}$, respectively and use the fact, that we intialize the scale factor $a(\tau_0) = 1$. The details can be found in the code.
To test this procedure on Graph 2 we present the comparison between the power spectrum obtained from the random initial conditions set in the program and its predicted expectation value. The units used on the graph are the program units, in which the lattice spacing h satisfies the condition $h = 1$.

The full initial conditions for the fields on the lattice are obtained by performing the inverse Fourier transform of the described Fourier modes of the perturbations and adding the homogeneous initial conditions defined in the configuration file.



Graph 2: *The comparison between the power spectrum obtained from the random initial conditions set in the program and its predicted expectation value for two different sizes of the lattice: $N = 32$ and $N = 64$.*

References

[1] M. A. Amin, M. P. Hertzberg, D. I. Kaiser and J. Karouby, Int. J. Mod. Phys. D **24** (2014) 1530003 [arXiv:1410.3808 [hep-ph]].

[2] J. Sainio, JCAP **1204** (2012) 038 [arXiv:1201.5029 [astro-ph.IM]].

[3] G. N. Felder and I. Tkachev, Comput. Phys. Commun. **178** (2008) 929 doi:10.1016/j.cpc.2008.02.009 [hep-ph/0011159].

[4] A. V. Frolov, JCAP **0811** (2008) 009 doi:10.1088/1475-7516/2008/11/009 [arXiv:0809.4904 [hep-ph]].

[5] *"http://cosmo.kenyon.edu/gabe.html"*

[6] T. Krajewski, K. Turzyński and M. Wieczorek, arXiv:1801.01786 [astro-ph.CO].

[7] E. Hairer, *Lecture notes on Geometric Numerical Integration: "http://www.unige.ch/ hairer/poly_ geoint/week2.pdf"*

[8] *https://docs.scipy.org/doc/numpy/reference/routines.fft.html*

[9] V. Acquaviva, N. Bartolo, S. Matarrese, A. Riotto, Nucl. Phys. **B667** (2003)

[10] N. Bartolo, E. Komatsu, S. Matarrese, A. Riotto, Phys. Rept. **402** (2004)