Lattice Inflation Manual

目次

はじめに

本マニュアルは単一場インフレーション・axion-U(1) インフレーションの Lattice シミュレーションコード「SFInflation」「axion-U(1)」を理解するためのマニュアル。一応 Lattice を知らない人向けだが、プログラミング言語についての知識は範囲外。

コードは C++ で書かれていて LATTICEEASY[1] のコードを適宜変更したものを用いている。LATTICEEASY のコードについての詳細は `https://www.felderbooks.com/latticeeasy/latticeeasydocs/latticeeasydocs.html` を参照。C++ の知識については私は「ロベールの C++ 入門講座」で勉強した。

インフレーションの Lattice シミュレーションとして主に参考にしたのは Angelo Caravano 氏の Simulating the inflationary Universe: from single-field to the axion-U(1) model[2] あるいは [3, 4, 5]。その他の参考は LATTICEEASY や CosmoLattice[6]。

注意事項として LatticeQCD とは根本的に違い、おそらく N 体シミュレーションと呼ばれるものに近い方の Lattice です（あんまわかってない)。また、コードは「美しさ」の「う」の字くらいしかないので大いに改善の余地が残されていることをご了承ください。最後に、本マニュアルのページ数が膨大になってしまいましたが、これはコードレビューで全コードを記載したせいで、実際に読むことになるページ数は 100 に満たないと思いますのでご安心ください。

本マニュアルの構成は以下の通り。

- 第 I 部：Lattice シミュレーションの一般論
- 第 II 部：単一場インフレーションのシミュレーションとコード解説
- 第 III 部：axion-U(1) インフレーションのシミュレーションとコード解説
- 第 IV 部：シミュレーションを実行する際の方法・注意事項
- 付録：計算過程の明記など

記法・規約

本論文を通して用いる単位系は、自然単位系 $c = k_B = \hbar = 1$・換算プランク質量を $M_{\rm pl} = 1$ とする単位系である。

時間変数は共形時間 τ を用い、共形時間に関する微分はドット $\dot{}$ で表し、プログラム上の無次元化された共形時間 $\tau_{\rm pr}$ に関する微分はプライム $'$ で表す。このようにプログラム上で扱う無次元化された量には”pr”の添字をつける。

議論すべき重要な仮定は曲率ゆらぎを無視していること。

用いる物理定数は以下の通り。

- 換算 Planck 質量

$$M_{\rm pl} \equiv \sqrt{\frac{1}{8\pi G}} = 2.435 \times 10^{18}\ \text{GeV}$$

- 太陽質量

$$M_\odot = 1.988 \times 10^{30}\ \text{kg}$$

第Ⅰ部

一般論

第 1 章

Lattice シミュレーションとは

1.1 概要

Lattice シミュレーションとは、有限な長さの一辺を持つ立方領域上での連続場の動力学を数値的にシミュレーションする手法であり、連続な立方領域を N^3 個の格子点に離散化し各格子点上で運動方程式を解く。

格子を特徴づけるのに一般的に用いられる量は

- L：立方領域の一辺の（共動的な）長さ
- N：一辺の格子点数
- dx：（共動的な）格子間隔。$dx = \frac{L}{N-1} \approx \frac{L}{N}$

の 3 つである。

連続的な立方領域の位置ベクトル

$$\boldsymbol{x} = \sum_i x_i \boldsymbol{e}_i \quad , \quad x_i = [0, L]$$

に対し、離散的な立方領域つまり（実）格子空間の位置ベクトルを

$$\boldsymbol{n} = \sum_i n_i \boldsymbol{e}_i \quad , \quad n_i \in \{0, 1, \cdots, N-1\} \tag{1.1}$$

と定義する。但し基底ベクトル $\boldsymbol{e}_i$ の定義は変更されていて、Lattice 上では格子間隔 dx に等しい大きさを持つ軸ベクトルとなる（格子で見れば 1 マス分なのでここでは基底ベクトルと呼んでいる）。

これらを踏まえて、図 1.1 に $N = 4$ の場合の 3 次元格子を示しておく。

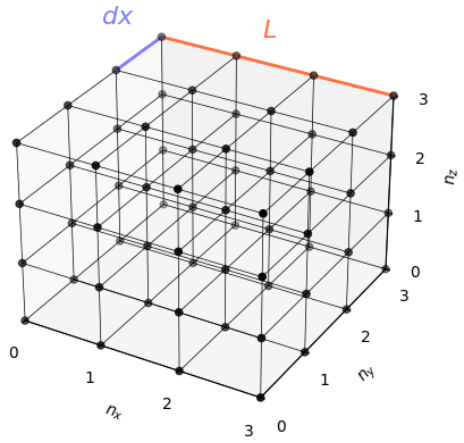

図 1.1　$N = 4$ の場合の 3 次元格子

1.2　場の定義

時空上の連続場の値 $f(\tau, \boldsymbol{x})$ は、格子シミュレーションでは各格子点上で定義する。

$$f(\tau, \boldsymbol{x}) \quad \longrightarrow \quad f(\tau, \boldsymbol{n})$$

以下では場合に応じて時間依存性の表記を省略する。このようにスカラー場は N^3 個の値を持ち、各格子点上で運動方程式を解くことで場の発展を追う。格子空間の境界条件として周期的境界条件を課し、場の満たす条件式は

$$f(\tau, \boldsymbol{n}) = f(\tau, \boldsymbol{n} + N\boldsymbol{e}_x) = f(\tau, \boldsymbol{n} + N\boldsymbol{e}_y) = f(\tau, \boldsymbol{n} + N\boldsymbol{e}_z) \tag{1.2}$$

となる。

ゆらぎによって場は各格子点上で異なる値を持ち、背景量を取り出したければ格子平均値を計算すれば良い。

$$\bar{f}(\tau) \equiv \langle f(\tau) \rangle = \frac{1}{N^3} \sum_{\boldsymbol{n}} f(\tau, \boldsymbol{n}) \tag{1.3}$$

以降、格子シミュレーションの文脈においては $\langle \cdots \rangle$ を格子平均の意味で用いる。

一方、ゆらぎの値を取り出したければ平均値を差し引いて

$$\delta f(\tau, \boldsymbol{n}) \equiv f(\tau, \boldsymbol{n}) - \bar{f}(\tau) \tag{1.4}$$

で計算される。これは宇宙論的摂動論におけるゆらぎの定義（空間依存しない平均値からのずれ）である。

1.3　Lattice シミュレーションの利点

格子シミュレーションでは各格子点で異なる場の値を取ることが許されているので、ゆらぎが場の値に原理的に組み込まれている。このゆらぎは値が大きくなることが許され、線形摂動論が破れるような場合も取り扱うことができ非摂動論的な計算が可能である。

しかし注意が必要なこともあり、ゆらぎが十分小さい時は線形摂動論と同様の結果が得られると期待されるが、プログラムで扱う変数（の型）が持つ有効数字の桁によって、ゆらぎの値が何桁まで信頼できるか決定される。例えば C++ の float 型は約 8 桁の有効数字を持ち、この 8 桁が場の値の有効桁となる。ここでゆらぎが平均値から 4 桁小さい場合、ゆらぎの有効桁は 4 桁となる。

$$f(\tau, \boldsymbol{n}) = 1.0002745 \quad \longrightarrow \quad \delta f(\tau, \boldsymbol{n}) = 2.745 \times 10^{-4}$$

このようにゆらぎの有効桁は変数の型よりも少ないため、ゆらぎが小さい場合（厳密には時間発展の最中に 1 度でも小さくなった場合）は信頼できる桁に注意が必要である。

1.4　リスケーリング

プログラム上で数値的に扱うために時空座標を無次元化（リスケーリング）する必要がある。ここで導入するリスケーリングの定義は [1] や [2] とは異なり、私のオリジナルの必要最小限な定義である。リスケーリングパラメータは無次元化のための質量次元を持つ量（インフラトン質量 m）とスケール因子の冪を調整する s の 2 つで

$$d\tau_{\rm pr} = ma^s d\tau, \quad \boldsymbol{x}_{\rm pr} = m\boldsymbol{x} \tag{1.5}$$

のように無次元化する。私は $s = 1$ を主に用いるが、以下では s の値を指定せずに計算を進める（s の選択がシミュレーションに与える影響はあまりわかっていない）。

この無次元化によって、立方領域の一辺の長さ L や格子間隔 dx はプログラム上では

$$L_{\rm pr} = mL, \qquad dx_{\rm pr} = mdx \tag{1.6}$$

で表される。また時間微分も変更を受け、共形時間 τ による微分 (ドット $\dot{}$) とプログラム

時間 $\tau_{\rm pr}$ による微分（プライム $'$）の関係は[*1]

$$\dot{a} = ma^s a', \qquad \ddot{a} = m^2 a^{2s}\left[a'' + s\frac{(a')^2}{a}\right] \tag{1.7}$$

$$\dot{f} = ma^s f', \qquad \ddot{f} = m^2 a^{2s}\left[f'' + s\frac{a'}{a}f'\right] \tag{1.8}$$

ただし f は場や時間依存の量を表す。空間微分演算子は

$$\nabla_{\rm pr} = \frac{\nabla}{m} \tag{1.9}$$

の関係がある。

プログラム上のエネルギー密度と圧力を以下で定義する。

$$\rho_{\rm pr} \equiv \frac{\rho}{m^2 a^{2s}}, \qquad p_{\rm pr} \equiv \frac{p}{m^2 a^{2s}} \tag{1.10}$$

これに伴って、インフラトンポテンシャルに対しても

$$V(\phi)_{\rm pr} \equiv \frac{V(\phi)}{m^2 a^{2s}} \tag{1.11}$$

を定義する。

ちなみに LATTICEEASY では、場の値の規格化・空間微分でスケール因子の冪を調整するというさらに多いリスケーリングパラメータを用いている。

[*1] 計算は付録を参照

第 2 章

離散化

本章では Lattice シミュレーションに伴う離散化の影響・実装方法をフーリエ変換と微分演算子に関して説明する。2.1 節で離散フーリエ変換を定義、2.2 節で微分演算子の離散化法を説明、2.3 節で物理的な運動量に対応する Lattice 上の有効運動量を定義、2.4 節で物理的なフーリエモードと Lattice 上の離散フーリエモードの対応関係を導出、2.5 で Lattice 上の Horizon-Crossing や Lattice シミュレーションの限界を議論する。

2.1 離散フーリエ変換

2.1.1 定義

Lattice シミュレーションでは実空間を離散化して実空間上で運動方程式を解くが、ゆらぎのパワースペクトルなどを計算するためにはフーリエ空間に移る必要がある。そのためにフーリエ変換を離散化するが、まずは逆格子ベクトル $\boldsymbol{\kappa}$ を定義する。

$$\begin{aligned} \boldsymbol{\kappa}(\boldsymbol{m}) &= \frac{2\pi}{L}\boldsymbol{m}, \quad m_i \in \left\{ -\frac{N}{2}+1, ..., \frac{N}{2} \right\} \\ \boldsymbol{\kappa}_{\mathrm{pr}}(\boldsymbol{m}) &= \frac{\boldsymbol{\kappa}}{m} = \frac{2\pi}{L_{\mathrm{pr}}}\boldsymbol{m} \end{aligned} \tag{2.1}$$

これを踏まえて離散フーリエ変換 (Discrete Fourier Transformation, DFT) とその逆変換を以下のように定義する。[*1]

$$f(\tau, \boldsymbol{\kappa}) = \sum_{\boldsymbol{n}} f(\tau, \boldsymbol{n}) e^{\frac{2\pi i}{N} \boldsymbol{m} \cdot \boldsymbol{n}} \tag{2.2}$$

$$f(\tau, \boldsymbol{n}) = \frac{1}{N^3} \sum_{\boldsymbol{m}} f(\tau, \boldsymbol{\kappa}) e^{-\frac{2\pi i}{N} \boldsymbol{m} \cdot \boldsymbol{n}} \tag{2.3}$$

上記のように、実空間上とフーリエ空間上の場の区別を引数で行うこととする。実空間での場はリスケーリングの影響を受けず、DFT における位相が

$$\frac{2\pi}{N} \boldsymbol{m} \cdot \boldsymbol{n} = \boldsymbol{\kappa} \cdot dx \boldsymbol{n} = \boldsymbol{\kappa}_{\mathrm{pr}} \cdot dx_{\mathrm{pr}} \boldsymbol{n}$$

でリスケーリングの影響を受けないことから、フーリエモードもリスケーリングの影響を受けない。

$$f_{\mathrm{pr}}(\boldsymbol{n}) = f(\boldsymbol{n}) \quad , \quad f_{\mathrm{pr}}(\boldsymbol{\kappa}) = f(\boldsymbol{\kappa}) \tag{2.4}$$

フーリエモード $f(\boldsymbol{\kappa})$ は一般に複素数であり、N^3 個の格子点を考慮するとその自由度は $2N^3$ である。しかし格子空間あるいは実空間における場の値 $f(\boldsymbol{n})$ が実数である場合、その自由度は N^3 でありフーリエモードの自由度も N^3 に削減される。$f(\boldsymbol{n})$ が実場であることによる $f(\boldsymbol{\kappa})$ に対する条件式は

$$f^*(\boldsymbol{\kappa}) = f(-\boldsymbol{\kappa}) \quad , \quad \text{ただし } \frac{2\pi}{L} \left(-\frac{N}{2} \right) \equiv \frac{2\pi}{L} \frac{N}{2} \tag{2.5}$$

である。[*2]

2.1.2　逆格子ベクトルの取り方の等価性

(2.1) でベクトル $\boldsymbol{m}$ の各成分は正負で定義した。これは（空間 2 次元の場合に置き換えれば）図 2.1 の右に対応し、逆格子ベクトルの各大きさに対して全方向を考慮している。しかし $m_i \in \{0, \cdots, N-1\}$ で定義する方法もあり、これは逆格子ベクトルの各成分が正の領域のみを考えていて図 2.1 の左に対応する。これらの逆格子ベクトルの定義は、フー

[*1] 計算は付録を参照
[*2] 計算は付録を参照

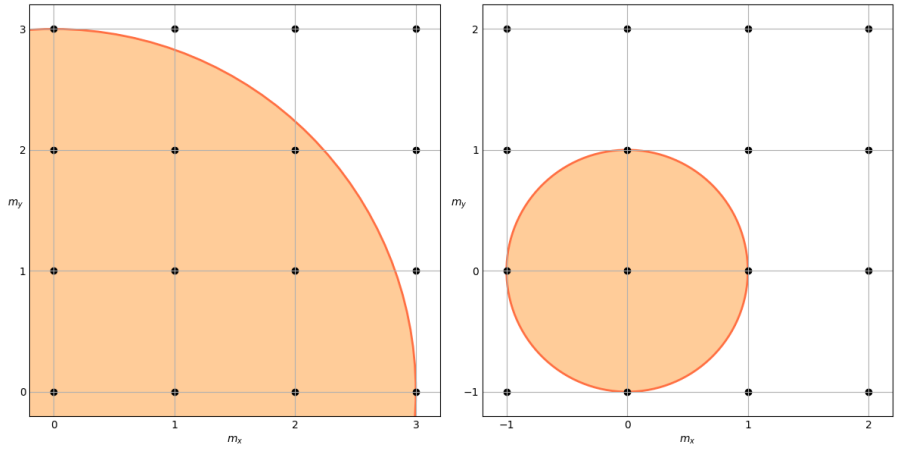

図 2.1　$N = 4$ かつ 2 次元での逆格子ベクトルのイメージ。正の成分（左）と全方向網羅（右）。

リエモードと後に定義する有効運動量ベクトルにおいては等価である。なぜならフーリエ変換の位相について

$$\begin{aligned}\exp\left[\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}\right] &= \exp\left[\frac{2\pi i}{N}(\boldsymbol{m}-\boldsymbol{N}+\boldsymbol{N})\cdot\boldsymbol{n}\right] \\ &= \exp\left[\frac{2\pi i}{N}(\boldsymbol{m}-\boldsymbol{N})\cdot\boldsymbol{n}\right]\exp\left[2\pi i(n_x+n_y+n_z)\right] \\ &= \exp\left[\frac{2\pi i}{N}(\boldsymbol{m}-\boldsymbol{N})\cdot\boldsymbol{n}\right]\end{aligned}$$

の関係があることから、$m_i \in \left\{\frac{N}{2}+1,\cdots,N-1\right\}$ に対して (2.1) の $m_i \in \left\{-\frac{N}{2}+1,\cdots,-1\right\}$ が対応している。私が (2.1) の定義を用いるのは、同じ $|\boldsymbol{\kappa}|$ の値を持つ逆格子ベクトルあるいはフーリエモードのサンプル数を増やすためである。

2.1.3　FFT

プログラム上で DFT を実行することを考えると、愚直に (2.2)(2.3) をそのまま実装した場合は処理に長い時間がかかる。そこで高速フーリエ変換 (Fast Fourier Transformation, FFT) がよく用いられる。これは DFT を高速で処理するための離散フーリエ変換の 1 つの表現であり、格子点数 N が 2 の累乗の時のみ成立する。Lattice シミュレーションで $N = 512, 1024$ などがよく用いられるのはこの FFT の制限によるものである。以下で 1 次元の場合の FFT の表式を導く。

FFT では N 点の DFT を $N/2$ 点の 2 つの DFT に分解する。複素関数 h_k の離散複素

フーリエ変換は

$$
\begin{aligned}
H_k &= \sum_{n=0}^{N-1} W^{kn} h_n \quad , \quad W = e^{\frac{2\pi i}{N}} \\
&= \sum_{n=0}^{\frac{N}{2}-1} W^{k(2n)} h_{2n} + \sum_{n=0}^{\frac{N}{2}-1} W^{k(2n+1)} h_{2n+1} \\
&= \sum_{n=0}^{\frac{N}{2}-1} e^{\frac{2\pi i}{\frac{N}{2}} kn} h_{2n} + W^k \sum_{n=0}^{\frac{N}{2}-1} e^{\frac{2\pi i}{\frac{N}{2}} kn} h_{2n+1} \\
&\equiv H_k^e + W^k H_k^o
\end{aligned}
$$

ここで上付き添字の”e”は偶数 (even) を表し、”o”は奇数 (odd) を表す。$N = 2^l$ で累乗で表されれば、上の分解を繰り返すことで和の記号を省略することができる。簡単な例として $N = 4$ の場合を具体的に見ると

$$
\begin{aligned}
H_k &= \sum_{n=0}^{\frac{N}{2}-1} e^{\frac{2\pi i}{\frac{N}{2}} kn} h_{2n} + W^k \sum_{n=0}^{\frac{N}{2}-1} e^{\frac{2\pi i}{\frac{N}{2}} kn} h_{2n+1} \\
&= \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{2}} k(2n)} h_{2(2n)} + \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{2}} k(2n+1)} h_{2(2n+1)} \\
&\quad + W^k \left[\sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{2}} k(2n)} h_{2(2n)+1} + \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{2}} k(2n+1)} h_{2(2n+1)+1} \right] \\
&= \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{4}} kn} h_{4n} + W^{2k} \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{4}} kn} h_{4n+2} \\
&\quad + W^k \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{4}} kn} h_{4n+1} + W^{3k} \sum_{n=0}^{\frac{N}{4}-1} e^{\frac{2\pi i}{\frac{N}{4}} kn} h_{4n+3} \\
&\overset{N=4}{=} h_0 + W^{2k} h_2 + W^k h_1 + W^{3k} h_3 \\
&\equiv H_k^{ee} + W^{2k} H_k^{eo} + W^k H_k^{oe} + W^{3k} H_k^{oo}
\end{aligned}
$$

最終的な h の下付き添字と H の上付き添字には関係があり、「e, o の順番を反転させ、$e = 0, o = 1$ として 2 進数を作り、10 進数に直すと h の下付き添字に等しくなる」。コードは LATTICEEASY のものを用いたが、アルゴリズムはあまりわかっていないので説明しない。

2.2　微分演算子

格子シミュレーションでは各格子点上で場の値を定義し運動方程式を解くが、異なる点での場の値は独立に発展するわけではない。なぜなら空間微分項によって、隣接格子点の情報が運動方程式に組み込まれるからである。これを実現するために、連続空間における空間微分演算子を離散化する必要がある。離散化に伴って微分は差分で定義されるが、その差分の取り方には任意性がある。ここではよく用いられる差分の取り方を列挙しておく。

1 階微分を離散化する場合、解像度（実際の連続空間を離散化した時のピクセルあるいは dx の小ささ、dx が小さいほど連続空間に近づき解像度が高くなる。モザイクアートのモザイク数が多いほど写真に近づくことと同じ。）を最大化するためには、注目格子点 $\boldsymbol{n}$ の最隣接格子点 $\boldsymbol{n} \pm \boldsymbol{e}_i$ を用いればよく

$$\partial_i f(\boldsymbol{n}) = \frac{f(\boldsymbol{n}+\boldsymbol{e}_i) - f(\boldsymbol{n}-\boldsymbol{e}_i)}{2dx} \tag{2.6}$$

となる。プログラム上で実装する場合は

$$\partial_{i,\mathrm{pr}} f(\boldsymbol{n}) = \frac{f(\boldsymbol{n}+\boldsymbol{e}_i) - f(\boldsymbol{n}-\boldsymbol{e}_i)}{2dx_{\mathrm{pr}}} \tag{2.7}$$

のようにプログラム上の格子間隔 dx_{pr} を用いれば良い。

(2.6) の 1 階差分をもう一度適用して得られるラプラシアンの離散化は

$$\nabla^2 f(\boldsymbol{n}) = \frac{1}{(2dx)^2} \sum_{i=x,y,z} \sum_{c=\pm 2} [f(\boldsymbol{n}+c\boldsymbol{e}_i) - f(\boldsymbol{n})] \tag{2.8}$$

である。しかしこの定義ではラプラシアンの解像度が最大化されていない。ラプラシアンの解像度が最大化されるような定義は最隣接格子点を用いて

$$\nabla^2 f(\boldsymbol{n}) = \frac{1}{dx^2} \sum_{i=x,y,z} \sum_{c=\pm 1} [f(\boldsymbol{n}+c\boldsymbol{e}_i) - f(\boldsymbol{n})] \tag{2.9}$$

と表される。

2.3　有効運動量

格子上の有効運動量ベクトルを、物理的な運動量ベクトルに対応するものとして定義する。連続空間では微分演算子のフーリエ変換で運動量ベクトルが現れるので、有効運動量ベクトルも同様に微分演算子の離散フーリエ変換で定義する。思想は連続空間でのフーリエモードの運動方程式を再現することで、そうすれば観測量であるパワースペクトルを再現できると期待する。(2.9) に対する有効運動量ベクトルは

$$
\begin{aligned}
\mathrm{DFT}[\nabla^2 f(\boldsymbol{n})](\boldsymbol{\kappa}) &= \sum_{\boldsymbol{n}} \frac{1}{dx^2} \sum_{i=x,y,z} \left[f(\boldsymbol{n}+\boldsymbol{e}_i) + f(\boldsymbol{n}-\boldsymbol{e}_i) - 2f(\boldsymbol{n}) \right] e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} \\
&= \frac{1}{dx^2} \sum_{\boldsymbol{n}} \sum_{i=x,y,z} \Big[f(\boldsymbol{n}+\boldsymbol{e}_i) e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot(\boldsymbol{n}+\boldsymbol{e}_i-\boldsymbol{e}_i)} \\
&\quad + f(\boldsymbol{n}-\boldsymbol{e}_i) e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot(\boldsymbol{n}+\boldsymbol{e}_i-\boldsymbol{e}_i)} - 2f(\boldsymbol{n}) e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} \Big] \\
&= \frac{1}{dx^2} \sum_{\boldsymbol{n}} \sum_{i=x,y,z} \Big[f(\boldsymbol{n}) e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot(\boldsymbol{n}-\boldsymbol{e}_i)} \\
&\quad + f(\boldsymbol{n}) e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot(\boldsymbol{n}+\boldsymbol{e}_i)} - 2f(\boldsymbol{n}) e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} \Big] \\
&= \frac{1}{dx^2} f(\boldsymbol{\kappa}) \sum_{i=x,y,z} \left[2\cos\left(\frac{2\pi m_i}{N}\right) - 2 \right] \\
&= -\left[\frac{2}{dx} \sqrt{\sin^2\left(\frac{\pi m_x}{N}\right) + \sin^2\left(\frac{\pi m_y}{N}\right) + \sin^2\left(\frac{\pi m_z}{N}\right)} \right]^2 f(\boldsymbol{\kappa})
\end{aligned}
$$

より

$$
[k_{\mathrm{eff}}(\boldsymbol{m})]_i = \frac{2}{dx} \sin\left(\frac{\pi m_i}{N}\right) \tag{2.10}
$$

同様に (2.6)(2.8) に対する有効運動量ベクトルは

$$
[k_{\mathrm{eff}}(\boldsymbol{m})]_i = \frac{1}{dx} \sin\left(\frac{2\pi m_i}{N}\right) \tag{2.11}
$$

である。dx が共動的な長さであるので、これらの運動量も共動的な量である。

有効運動量ベクトルのリスケーリングは逆格子ベクトルと同様に

$$
\boldsymbol{k}_{\mathrm{eff,pr}} = \frac{\boldsymbol{k}_{\mathrm{eff}}}{m} \tag{2.12}
$$

2.4　フーリエモードの対応関係

物理的には無限空間かつ連続空間でフーリエモード $f(\boldsymbol{k})$ が計算される。一方、Lattice シミュレーションでは有限空間かつ離散空間で離散フーリエモード $f(\boldsymbol{\kappa})$ が計算される。ここでは $f(\boldsymbol{k})$ と $f(\boldsymbol{\kappa})$ の対応関係を求める。

まず有限化に伴う補正を考える。ある量に対して 2 乗の空間平均が体積 V に依存しないことを要求する [1]。

$$\begin{aligned}\langle f(\boldsymbol{x})^2\rangle &= \frac{1}{V}\int d^3x \int \frac{d^3k_1}{(2\pi)^3}\int \frac{d^3k_2}{(2\pi)^3} f(\boldsymbol{k}_1)f(\boldsymbol{k}_2)e^{-i(\boldsymbol{k}_1+\boldsymbol{k}_2)\cdot\boldsymbol{x}} \\ &= \frac{1}{V}\int \frac{d^3k}{(2\pi)^3} f(\boldsymbol{k})f(-\boldsymbol{k}) \\ &= \frac{1}{V}\frac{1}{(2\pi)^3}\int d^3k |f(\boldsymbol{k})|^2\end{aligned}$$

ただし 2 つ目の $=$ では Dirac のデルタ関数を用い、3 目の $=$ では実空間での場の値 $f(\boldsymbol{x})$ が実数であることから得られる $f^*(\boldsymbol{k}) = f(-\boldsymbol{k})$ を用いた。ここで結果が体積 V に依存しないためには $|f(\boldsymbol{k})| \propto V^{\frac{1}{2}} = L^{\frac{3}{2}}$ である必要がある。

次に離散化に伴う補正を考える。

$$f(\boldsymbol{k}) = \int d^3x f(\boldsymbol{x})e^{i\boldsymbol{k}\cdot\boldsymbol{x}} \approx (dx)^3 \sum_{\boldsymbol{n}} e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} = (dx)^3 f(\boldsymbol{\kappa})$$

より離散化に伴って dx^3 の因子が必要である。

最終的に、物理的なフーリエモードと Lattice 上のフーリエモードの関係は

$$f(\boldsymbol{\kappa}) = \frac{L^{\frac{3}{2}}}{dx^3} f(\boldsymbol{k}) = m^{\frac{3}{2}} \frac{L_{\mathrm{pr}}^{\frac{3}{2}}}{dx_{\mathrm{pr}}^3} f(\boldsymbol{k}) \tag{2.13}$$

この関係は、以降でシミュレーションにおけるゆらぎの初期条件を求める際や、シミュレーションのパワースペクトルから物理的なパワースペクトルを得る際に用いる。

2.5　解像度と Horizon-Crossing

Lattice シミュレーションにおける「解像度」とは dx のことである。Lattice シミュレーションでは例えば $\frac{dx}{2}$ の距離スケールのゆらぎは原理上扱うことができず、格子間隔 dx が Lattice シミュレーションで扱える最小の距離スケールである。dx が小さければ小さい程解像度が高く、解像度を上げるには格子点数 N を大きくする必要がある。

インフレーションの Lattice シミュレーションでは、L あるいは dx が共動的な長さを表す（物理的な長さを表すようにする設定もあるのかも）ため、上で述べた解像度があっという間に低く（悪く）なってしまい長い間の時間発展を追うことはできない。追うことができるのは数 e-folds で、インフレーション開始から終了まで Lattice シミュレーションすることは実質的に不可能である（無限のメモリ容量と無限の計算力があればその限りではないけれども）。

具体的には、共動的な距離スケール l と Horizon スケールは

$$l \ : \ \frac{1}{aH}$$

の比較で行い、インフレーション中はスケール因子が（宇宙時間 t に関しては）指数関数的に増加するため、ホライズンスケールが急激に小さくなり共動的な距離スケール l がすぐ super-horizon になる。Lattice シミュレーションでは数 e-folds のうちに最小の距離スケール dx が super-horizon になり、sub-horizon の物理が追えなくなってしまう。ただし、インフレーションにおける量子ゆらぎが sub-horizon で古典的に扱って良いかどうかは議論の余地あり。

次に Lattice シミュレーションにおける Horizon-Crossing を説明する。理論的によく議論される Horizon-Crossing は、共動的な運動量スケール k と Horizon スケールの逆数で

$$\begin{aligned} k < aH \qquad & \mathrm{Super-Horizon} \\ k > aH \qquad & \mathrm{Sub-Horizon} \end{aligned}$$

のように評価される。2.3 節で述べたように、Lattice シミュレーションでは物理的な運動量に対応するものとして有効運動量が存在するので、Lattice シミュレーションにおけ

る Horizon-Crossing は有効運動量 $k_{\rm eff}$ で評価する。プログラム上の量で表せば

$$k_{\rm eff,pr} < a^{s-1}a' \qquad {\rm Super-Horizon}$$
$$k_{\rm eff,pr} > a^{s-1}a' \qquad {\rm Sub-Horizon}$$

で評価する。

Lattice シミュレーションでは小さい $k_{\rm eff}$ から順に Horizon-Crossing していき、最後に最大の $k_{\rm eff}$ が Horizon-Crossing する。この最大の $k_{\rm eff}$ が Horizon-Crossing すると、Sub-Horizon の物理を見ることができなくなるので、これを Lattice シミュレーションが追える最終時刻とする。つまり、「最大の $k_{\rm eff}$ が Horizon-Crossing する時刻あたりまで Lattice シミュレーションで追える」。また、Sub-Horizon の物理を見るために「シミュレーション開始時刻で、最小の $k_{\rm eff}$ が Horizon スケール程度であるように設定する」。以上 2 つの基準を考慮すれば、開始時刻での Hubble パラメータの値から L がおおよそ決定され、N に応じて最大の $k_{\rm eff}$ が決まりシミュレーションで追える期間が決定される。

具体的には $k_{\rm eff,min} \sim \frac{2\pi}{L}$ なので

$$k_{\rm eff,min} \lesssim [aH]_{\rm ini} \quad \Longleftrightarrow \quad L \gtrsim \frac{2\pi}{H_{\rm ini}}$$

で L が決まり（上の条件は最小の $k_{\rm eff}$ がシミュレーション開始時刻で少し Super-Horizon になるようにしている）、このように選んだ L と使用しているコンピュータで可能な最大の N から

$$k_{\rm eff,max} \sim \frac{1}{dx}$$

のように最大の $k_{\rm eff}$ が決まる。この $k_{\rm eff,max}$ が Horizon-Crossing する e-folds は、インフレーションが slow-roll していて Hubble パラメータが一定であると近似（仮定）して

$$k_{\rm eff,max} \sim a_{\rm end}H_{\rm ini} \quad \Longleftrightarrow \quad a_{\rm end} \sim \frac{1}{H_{\rm ini}dx}$$
$$\Longleftrightarrow \quad N_{e,\rm end} \sim \ln\left(\frac{1}{H_{\rm ini}dx}\right)$$

でおおよそ決定される。厳密には $k_{\rm eff,max}$ の表式（あるいは $k_{\rm eff}$ の定義）に依存している。上の式から分かることは、格子点数 N を 2 倍したとしても $N_{e,\rm end}$ は $\ln 2 \approx 0.3$ しか延長されないということである。最後に注意書きとして、本節では次章で述べるスケール因子の初期条件 $a_{\rm ini} = 1$ を用いている。

第 3 章

初期条件

本章では Lattice シミュレーションにおける背景的な量（時間変数 τ, a, H, N_e、場の平均値）とゆらぎについて初期値の取り方を説明する。ゆらぎについては CosmoLattice[6] と同じ手法を用いて決定する。3.1 節で背景的な量の初期条件を説明し、3.2 節でゆらぎのフーリエモードから実空間での場の値を得る方法を説明し、3.3 節でゆらぎのフーリエモードの初期条件を説明する。

3.1　背景量

・スケール因子

Lattice シミュレーション開始時のスケール因子は 1 と規格化する。

$$a_{\rm ini} = 1 \tag{3.1}$$

・e-folds

スケール因子の初期値に合わせて、Lattice シミュレーションにおける e-folds の初期値は 0 である。

$$N_{e,\rm ini} = 0 \tag{3.2}$$

・共形時間

共形時間は de-Sitter 宇宙の関係式から決定する。

$$\tau_{\rm ini} = -\frac{1}{a_{\rm ini} H_{\rm ini}} = -\frac{1}{H_{\rm ini}} \tag{3.3}$$

de-Sitter 宇宙の関係式を用いるため、Lattice シミュレーション開始時には slow-roll 近似が十分良い精度で成立している必要がある。

・Hubble パラメータ

Hubble パラメータは Friedmann 方程式から決めるが、ゆらぎの寄与は無視する。

$$H_{\rm ini} = \sqrt{\frac{\rho_{\rm ini}}{3}} = m\sqrt{\frac{\rho_{\rm pr,ini}}{3}} \tag{3.4}$$

・スケール因子の微分

スケール因子の微分はリスケーリングの影響を受ける。スケール因子のプログラム時間による微分は

$$a'_{\rm ini} = \frac{\dot{a}_{\rm ini}}{ma^s_{\rm ini}} = \frac{a^2_{\rm ini}}{ma^s_{\rm ini}} H_{\rm ini} = \frac{H_{\rm ini}}{m} = \sqrt{\frac{\rho_{\rm pr,ini}}{3}} \tag{3.5}$$

・場の平均値

場の値は背景量とゆらぎから初期値を決定する（後述）。インフラトン場に関する背景量 $\bar{\phi}, \dot{\bar{\phi}}$ は、背景インフレーションから決める。この背景場の値によって Lattice シミュレーション開始時の e-folds が決定される。場の平均値 $\bar{\phi}$ はリスケーリングの影響を受けないが、場の微分は時間微分からリスケーリングの影響を受ける。インプットパラメータである場の微分の初期値 $\dot{\bar{\phi}}_{\rm ini}$ に対して、プログラム上の実装は

$$\bar{\phi}'_{\rm ini} = \frac{\dot{\bar{\phi}}_{\rm ini}}{ma^s_{\rm ini}} = \frac{\dot{\bar{\phi}}_{\rm ini}}{m} \tag{3.6}$$

3.2　実空間での場の値を求める方法

上で述べたように、場の初期値は背景量とゆらぎから決定する。ゆらぎの初期条件の決定方法は次節で述べるが、結果としてはゆらぎの離散フーリエ変換 $\delta f(\boldsymbol{\kappa})$ とその時間微分 $\delta f'(\boldsymbol{\kappa})$ が与えられる。これらから実空間の場の値を求めるために、まずゆらぎのフーリエ変換を逆離散フーリエ変換 (2.3) して実空間のゆらぎを求める。このようにして得られた各格子点でのゆらぎに背景量を加えることで場の初期値を決定する。

$$\delta\phi(\boldsymbol{\kappa}) \xrightarrow{(2.3)} \delta\phi(\boldsymbol{n}) \xrightarrow{+\bar{\phi}} \bar{\phi} + \delta\phi(\boldsymbol{n}) \equiv \phi(\boldsymbol{n})$$
$$\delta\phi'(\boldsymbol{\kappa}) \xrightarrow{(2.3)} \delta\phi'(\boldsymbol{n}) \xrightarrow{+\bar{\phi}'} \bar{\phi}' + \delta\phi'(\boldsymbol{n}) \equiv \phi'(\boldsymbol{n})$$

3.3　ゆらぎ

Lattice シミュレーションにおけるゆらぎの初期条件として「解析的な計算を再現できるようなゆらぎであること」を課す。具体的には、ゆらぎの主な観測量であるパワースペクトルを再現するような量子ゆらぎを初期条件として課す。私のコードではインフラトンゆらぎの初期条件に slow-roll 近似の解析解である Hankel 関数を用いたが、これはつまり slow-roll 近似と線形摂動論が良い精度で成立している時刻を Lattice シミュレーションの開始時刻に設定する必要があることを指す。

本節ではモード関数として指数関数を採用し、確率密度分布がガウス分布となるようなゆらぎの実装方法を説明する。モード関数の一般化は第 2 部と第 3 部で行う。

3.3.1　解析解を再現する確率分布

解析的に得られるインフラトンパワースペクトルは sub-horizon で

$$\mathcal{P}_{\delta\phi}(k) = |\delta\phi(k)|^2 = \frac{1}{2a^2\sqrt{k^2+m_{\rm eff}^2}} \quad , \quad m_{\rm eff}^2 = a^2\frac{d^2V(\phi)}{d\phi^2} \tag{3.7}$$

(無次元パワースペクトルでは k^3 が掛かり $\mathcal{P} \propto k^2$ となる)。ここでインフラトンゆらぎのモード関数は

$$\delta\phi(\tau,k) = \frac{1}{a\sqrt{2}(k^2+m_{\rm eff}^2)^{\frac{1}{4}}} e^{-i\sqrt{k^2+m_{\rm eff}^2}\tau} \tag{3.8}$$

である[*1]。

(3.7) の値を分散として出すような $|\delta\phi(k)|$ の分布は Rayleigh 分布

$$\mathrm{PDF}[|\delta\phi|] = \left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} |\delta\phi| \exp\left[-\frac{1}{2}\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1}|\delta\phi|^2\right] \tag{3.9}$$

である[*2]。ただし”確率変数”の範囲は $0 \leq |\delta\phi|$ である。コード上で欲しいのは複素数 $\delta\phi(k)$ であり、$|\delta\phi(k)|$ が上記の Rayleigh 分布になるような $\delta\phi(k)$ の分布は $N(0, \frac{\mathcal{P}_{\delta\phi}}{2})$ の複素ガウス分布である。ここで言う複素ガウス分布とは、複素数 $\delta\phi(k)$ の実部 $\mathrm{Re}[\delta\phi(k)]$ と虚部 $\mathrm{Im}[\delta\phi(k)]$ が各々独立に $N(0, \frac{\mathcal{P}_{\delta\phi}}{2})$ に従うという意味である。

[*1] 計算は付録を参照

[*2] 計算は付録を参照

$\mathrm{Re}[\delta\phi(k)], \mathrm{Im}[\delta\phi(k)]$ の同時確率分布は[*3]

$$\mathrm{PDF}\left[\mathrm{Re}[\delta\phi(k)], \mathrm{Im}[\delta\phi(k)]\right] = \frac{1}{2\pi\frac{\mathcal{P}_{\delta\phi}}{2}} \exp\left[-\frac{(\mathrm{Re}[\delta\phi(k)])^2 + (\mathrm{Im}[\delta\phi(k)])^2}{2\frac{\mathcal{P}_{\delta\phi}}{2}}\right] \tag{3.10}$$

3.3.2　ガウス分布の実装

(3.10) でフーリエモードの実部と虚部がガウス分布に従えば良いことがわかった。ここでは一様乱数 X, Y からガウス乱数（ガウス分布に従う確率変数）を得る公式を求める。

一様乱数の生成

LATTICEEASY で用いられている生成方法と同じく、「PM(Park-Miller) minimum standard random number generator」を用いる。これは

$$a = 7^5 = 16807 \quad , \quad m = 2^{31} - 1 = 2147483647$$

を用いて

$$X_{n+1} = (a \times X_n) \bmod m$$

によって疑似乱数を生成する方法である（線形合同法と似てる）。ただし、プログラム上では数値が大きくオーバーフロー（変数の最大値を上回り正しくない結果を出すこと）してしまうので

$$q = 127773 \quad , \quad r = 2836 \qquad (m = aq + r)$$

を用いて

$$X_{n+1} = \begin{cases} a \times [X_n \bmod q] - r \times [\text{quotient of } X_n/q] & X_n \geq 0 \\ a \times [X_n \bmod q] - r \times [\text{quotient of } X_n/q] + m & X_n \leq 0 \end{cases}$$

を計算することで同様の結果が得られる。

[*3] 計算は付録を参照

一様乱数からガウス乱数の生成公式

X, Y を [0,1] の一様乱数とする。同時確率分布は

$$\mathrm{PDF}[X, Y] = 1 \quad , \quad (0 \leq X, Y \leq 1) \tag{3.11}$$

であり、天下り的に

$$r = \sqrt{-2 \ln X} \quad , \quad \theta = 2\pi Y \tag{3.12}$$

と変数変換すると

$$\begin{aligned}
\int_0^1 dX \int_0^1 dY &= \int_0^1 dX \frac{e^{\ln X}}{X} \int_0^1 dY \\
&= \int_1^0 -\frac{dX}{X} e^{-(-\ln X)} \int_0^1 dY \frac{2\pi}{2\pi} \\
&= \int_0^\infty dr r e^{-\frac{r^2}{2}} \frac{1}{2\pi} \int_0^{2\pi} d\theta \\
&= \int_0^\infty dr r \int_0^{2\pi} d\theta \frac{1}{2\pi} e^{-\frac{r^2}{2}} \\
&= \int_{-\infty}^\infty dx \int_{-\infty}^\infty dy \frac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}
\end{aligned}$$

最後の「=」で極座標からデカルト座標

$$x = r\cos\theta \quad , \quad y = r\sin\theta$$

に座標変換した。

x, y は分散 1 のガウス分布に従うので、(3.10) のガウス分布のためには標準偏差 $\sqrt{\frac{\mathcal{P}_{\delta\phi}}{2}}$ を掛けて

$$\begin{aligned}
\mathrm{Re}[\delta\phi(k)] &= \sqrt{\mathcal{P}_{\delta\phi}} \sqrt{-\ln X} \cos(2\pi Y) \\
\mathrm{Im}[\delta\phi(k)] &= \sqrt{\mathcal{P}_{\delta\phi}} \sqrt{-\ln X} \sin(2\pi Y)
\end{aligned}$$

よってシミュレーションで実装すべき複素数 $\delta\phi(k)$ は

$$\delta\phi(k) = \sqrt{\mathcal{P}_{\delta\phi}} \sqrt{-\ln X} e^{i2\pi Y} \tag{3.13}$$

ここで (3.7) より $\sqrt{\mathcal{P}_{\delta\phi}}$ は解析的な $|\delta\phi(k)|$ であるのでモード関数で表され、以降はモード関数を指定すれば (3.13) に則ってガウス分布の $\delta\phi(k)$ を実装できる。

3.3.3　等方的初期条件

(3.13) の実装では $e^{-i\sqrt{k^2+m_{\rm eff}^2}\tau}$ の波を考えており、これは 1 方向に進む波のみを考慮していることに対応する（ゆらぎの初期条件に対して解析的なモード関数は絶対値で効いてくるため指数関数の位相の符号の違いは (3.13) には現れないが、場の微分のゆらぎの初期条件ではその違いが明示的に効いてくる、後述する）。1 方向に進む波では宇宙原理の等方性に逆らうため、CosmoLattice[6] でも採用されている「等方的初期条件」を用いる。等方的初期条件では上述した波と逆方向の $e^{i\sqrt{k^2+m_{\rm eff}^2}\tau}$ も考慮する。この時の $\delta\phi(k)$ は

$$\delta\phi(k) = \frac{1}{\sqrt{2}}\sqrt{\mathcal{P}_{\delta\phi}}\left(\sqrt{-\ln X_1}e^{i2\pi Y_1} + \sqrt{-\ln X_2}e^{i2\pi Y_2}\right) \tag{3.14}$$

ここで $\sqrt{2}$ で割ったのは、ガウス分布の加法性を考慮して $|\delta\phi(k)|$ の分散が $\mathcal{P}_{\delta\phi}$ となるようにするためである。

ガウス分布の加法性とは、2 つのガウス分布 $N(\mu_1,\sigma_1^2), N(\mu_2,\sigma_2^2)$ に従う確率変数 x_1, x_2 に対して x_1+x_2 が $N(\mu_1+\mu_2, \sigma_1^2+\sigma_2^2)$ に従うことである。これを考慮して (3.14) は分散 $\mathcal{P}_{\delta\phi}$ に従う。式的には

$$x_{1,2} = \sqrt{\frac{\mathcal{P}_{\delta\phi}}{2}}\sqrt{-\ln X_{1,2}}e^{i2\pi Y_{1,2}}$$

の実部と虚部が各々 $N(0, \frac{\mathcal{P}_{\delta\phi}}{4})$ に従うため、(3.14) つまり x_1+x_2 の実部と虚部は各々 $N(0, \frac{\mathcal{P}_{\delta\phi}}{2})$ に従う。よって (3.14) の $\delta\phi(k)$ は、分散が $\mathcal{P}_{\delta\phi}$ となる $|\delta\phi(k)|$ の Rayleigh 分布を導く。

3.3.4　場の微分のゆらぎ

場の微分のゆらぎはモード関数の微分から決める。(3.8) のモード関数に対しては

$$\begin{aligned}
\partial_\tau\delta\phi(k) &= \partial_\tau\left[\frac{1}{a\sqrt{2}(k^2+m_{\rm eff}^2)^{\frac{1}{4}}}e^{\pm i\sqrt{k^2+m_{\rm eff}^2}\tau}\right] \\
&= \left[-\mathcal{H} - \frac{\partial_\tau(k^2+m_{\rm eff}^2)}{4(k^2+m_{\rm eff}^2)} \pm i\sqrt{k^2+m_{\rm eff}^2}\right]\frac{1}{a\sqrt{2}(k^2+m_{\rm eff}^2)^{\frac{1}{4}}}e^{\pm i\sqrt{k^2+m_{\rm eff}^2}\tau} \\
&\approx \left[-\mathcal{H} \pm i\sqrt{k^2+m_{\rm eff}^2}\right]\frac{1}{a\sqrt{2}(k^2+m_{\rm eff}^2)^{\frac{1}{4}}}e^{\pm i\sqrt{k^2+m_{\rm eff}^2}\tau}
\end{aligned}$$

と計算される。ただし LATTICEEASY[1] と同様に 2 項目の時間依存性 $\partial_\tau(k^2+m_{\rm eff}^2)$ を無視した。

(3.14) のように等方的初期条件を考慮して

$$\partial_\tau \delta\phi(k) = -\mathcal{H}\delta\phi(k) - i\sqrt{k^2+m_{\rm eff}^2}\sqrt{\frac{\mathcal{P}_{\delta\phi}}{2}}\left(\sqrt{-\ln X_1}e^{i2\pi Y_1} - \sqrt{-\ln X_2}e^{i2\pi Y_2}\right) \tag{3.15}$$

ここで右辺の $\delta\phi(k)$ は場のゆらぎで実装した (3.14) の値である。

3.3.5　リスケーリング

得られた表式をプログラム上で実装するためのリスケーリングを考慮する。(2.13) よりプログラム上で実装するフーリエモードは

$$\begin{aligned}\delta\phi(\boldsymbol{\kappa}) &= \frac{1}{\sqrt{2}}|\delta\phi|_{\rm pr}\left(\sqrt{-\ln X_1}e^{i2\pi Y_1} + \sqrt{-\ln X_2}e^{i2\pi Y_2}\right)\\ |\delta\phi|_{\rm pr} &= m\frac{L_{\rm pr}^{\frac{3}{2}}}{dx_{\rm pr}^3}\frac{1}{\sqrt{2}}\left(k_{\rm eff,pr}^2 + \frac{m_{\rm eff}^2}{m^2}\right)^{-\frac{1}{4}}\end{aligned} \tag{3.16}$$

ここで 1 式目はインフラトンゆらぎのモード関数に依らない一般的な表式で、2 式目の $|\delta\phi|_{\rm pr}$ は sub-horizon 極限のモード関数 (3.8) を用いリスケーリング (2.12)(2.13) を考慮した表式である。またスケール因子の初期値が 1 であることも用いた。

プログラム上で実装するゆらぎの微分は

$$\begin{aligned}\delta\phi'(\boldsymbol{\kappa}) = &-a'_{\rm ini}\delta\phi(\boldsymbol{\kappa})\\ &- i\sqrt{k_{\rm eff,pr}^2 + \frac{m_{\rm eff}^2}{m^2}}\frac{|\delta\phi|_{\rm pr}}{\sqrt{2}}\left(\sqrt{-\ln X_1}e^{i2\pi Y_1} - \sqrt{-\ln X_2}e^{i2\pi Y_2}\right)\end{aligned} \tag{3.17}$$

である。

一様乱数 X_1, X_2, Y_1, Y_2 は各 $\boldsymbol{\kappa}$ に対して異なる値を用いる。また、$\delta\phi'$ で用いる X_1, X_2, Y_1, Y_2 は同じ $\boldsymbol{\kappa}$ に対して $\delta\phi$ と同じ値を用いる。この場合、ゆらぎの生成に使用している自由度（一様乱数の数）は $2N^3$ であり、$\delta\phi(\boldsymbol{n}), \delta\phi'(\boldsymbol{n})$ の自由度の数と一致する。$2N^3$ の数え方は、1 つの $\boldsymbol{\kappa}$ に対して $\delta\phi, \delta\phi'$ で 4 つであり、$\boldsymbol{\kappa}$ が格子点数と同数の N^3 個あるので $4N^3$。最後にフーリエモードの条件式 (2.5) から、実際に値を決めるの（独立な値）は $\kappa_z > 0$ のフーリエモードのみであり、最終的な自由度は半分の $2N^3$ である。（LATTICEEASY[1] では正負方向の波の振幅を一致させるために $X_1 = X_2$ としているが、この場合は自由度が少ないためにガウス分布が正しく実現できないと考えられている）

第 4 章

時間発展の方法

本章では Lattice シミュレーション（より一般には常微分方程式の数値計算）で用いる時間発展あるいは時間積分の手法の概要を説明する。ただし具体的に実装する式などは第 II 部と題 III 部に記す。4.1 節で時間発展の概要を説明し、4.2 節以降で各手法について説明する。

4.1　一般論

常微分方程式の解法はいわゆる初期値問題であり、（場あるいは関数の）初期値を与えれば微分方程式（つまり時間発展の振る舞い）によって初期時刻から dt 後の値が得られ、これを繰り返せば原理的に任意の時刻における値を得ることができる。解析的に解いた場合は $dt \to 0$ の極限での解を得ることができていて厳密解となるが、数値的に解く場合は $dt \to 0$ の極限を取ることが不可能であるため厳密解からの誤差が生じる。この誤差は時間発展を繰り返すたびに積み重なっていく。以降は 1 回の時間発展における有限な時間間隔を Δt のように Δ を用いて表記する。（この段落の日本語変かも）

時間発展に用いる手法は様々あり、各手法で得られる数値解が厳密解からどの程度外れているのかを「何次の精度」という言葉で表す。これは 1 回の時間発展で

$$1 \text{ step Error} \equiv |y(t+\Delta t) - y_{i+1}| = \mathcal{O}(\Delta t^{p+1}) \tag{4.1}$$

の誤差を生じる場合に「p 次の精度」があると言う。ここで $y(t)$ が厳密解で y_i が数値解である。つまり 1 回の時間発展で誤差が時間間隔 Δt の $p+1$ 乗のオーダーである場合に p 次の精度がある。なぜ p 次なのかというと、初期時刻から最終時刻まで N 回時間発展

した時の大域誤差（最終的な誤差）が

$$\mathcal{O}(\Delta t^{p+1}) \times N = C\Delta t^p(\Delta t N) = C\Delta t^p \Delta t_{\rm tot} \tag{4.2}$$

と表されるからである。大域誤差に対して 1 回の時間発展で現れる誤差 (4.1) を局所誤差と呼ぶ。（ここの内容はどこぞの web サイトを参照した）

時間発展の手法は例えば

- Euler 法（1 次）
- Leap-Frog 法
- Runge-Kutta 法

が存在する。私のコードで用いているのは 4 次の Runge-Kutta 法である。また、私のコードで用いている時間変数は共形時間 τ（プログラム上ではこれに対応するプログラム時間 $\tau_{\rm pr}$）であり、その 1 回の時間間隔 $\Delta\tau_{\rm pr}$ は一定にしている。

時間発展の手法は大抵の場合 1 階微分方程式に対する解法となっている。しかし物理学において解くべきは時間に関する 2 階微分方程式

$$\frac{d^2x(t)}{dt^2} = f(t,x) \tag{4.3}$$

で、これに 1 階微分方程式の解法を適用するために $v(t) = \frac{dx}{dt}$ と $x(t)$ を独立に扱う。こうすることで (4.3) は

$$\frac{dx(t)}{dt} = v(t) \quad , \quad \frac{dv(t)}{dt} = f(t,x) \tag{4.4}$$

のように 2 つの 1 階微分方程式になる。

4.2　Euler 法

4.2.1　概要

ここでは最も単純な「1 つの 1 変数 1 階微分方程式」に対する Euler 法を説明する。Lattice シミュレーションで用いるための一般化は次の 4.2.2 節で説明する。ここで考える微分方程式を

$$\frac{dy(t)}{dt} = f(t, y) \tag{4.5}$$

とする。Euler 法では時間微分を離散化して差分で定義し

$$\begin{aligned} f(t, y) &= \frac{y(t + \Delta t) - y(t)}{\Delta t} \\ \therefore\ y(t + \Delta t) &= y(t) + \Delta t f(t, y(t)) \end{aligned} \tag{4.6}$$

このような上式 1 式目の差分の取り方を「前進差分」と呼び、2 式目の Euler 法を「前進 Euler 法」と呼ぶ。この方法では時刻 t の情報のみから時刻 $t + \Delta t$ の情報を得ることができるようになっていて、「陽解法」と呼ぶ。

一方、以下のような「後退差分」で差分を定義する方法もあり

$$\begin{aligned} f(t, y) &= \frac{y(t) - y(t - \Delta t)}{\Delta t} \\ y(t + \Delta t) &= y(t) + \Delta t f(t + \Delta t, y(t + \Delta t)) \end{aligned} \tag{4.7}$$

このような Euler 法を「後退 Euler 法」と呼ぶ。この方法では、時刻 $t + \Delta t$ の情報を得るのに時刻 t の情報だけでは足りず他の方程式を解く必要がある。これを陽解法に対して「陰解法」と呼ぶ。

Euler 法の精度は 1 次であり、Δt を小さくしてもあまり誤差は小さくならない。これは (4.6) が 1 次の Taylor 展開

$$y(t + \Delta t) \approx y(t) + \Delta t \frac{dy}{dt} \tag{4.8}$$

に対応しているからである。また、厳密解の近似として考えれば

$$\begin{aligned} y(t + \Delta t) &= y(t) + \int_0^{\Delta t} d\tilde{t} f(t + \tilde{t}, y(t + \tilde{t})) \\ &\approx y(t) + \Delta t f(t, y(t)) \end{aligned} \tag{4.9}$$

でこれはつまり図 4.1 のような近似をしていると解釈できる。

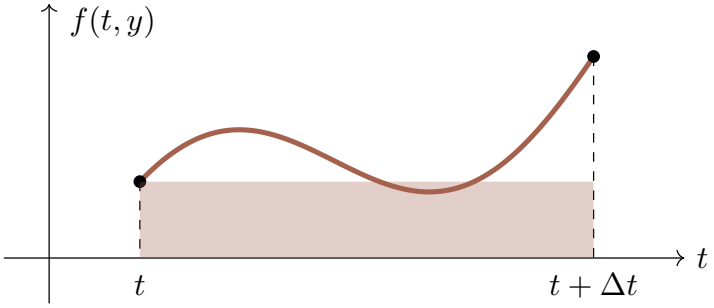

図 4.1　Euler 法の解釈（これほど f が変化するような Δt は大きすぎて本当は良くない）

4.2.2　実装

ここでは Lattice シミュレーションの時間発展に Euler 法を適用することを考える。4.1 節の最後で述べたように、1 つの 2 階微分方程式を 2 つの 1 階微分方程式に分解する。つの場 $\phi(\tau, \boldsymbol{n})$ を考えた場合、各格子点で場の運動方程式を解くため $2N^3$ 個の 1 階微分方程式が存在する（ϕ に対して N^3 個と ϕ' に対して N^3 個）。さらにスケール因子 a とその微分 a' に対する 1 階微分方程式が存在するため、合計 $2N^3+2$ 個の 1 階微分方程式に対して Euler 法を適用する。

$2N^3+2$ 個の微分方程式を

$$\frac{da(\tau_{\rm pr})}{d\tau_{\rm pr}} = a' \quad , \quad \frac{da'(\tau_{\rm pr})}{d\tau_{\rm pr}} = F^a(\tau_{\rm pr}) \tag{4.10}$$

$$\frac{d\phi(\tau_{\rm pr}, \boldsymbol{n})}{d\tau_{\rm pr}} = \phi'(\boldsymbol{n}) \quad , \quad \frac{d\phi'(\tau_{\rm pr}, \boldsymbol{n})}{d\tau_{\rm pr}} = F^\phi(\tau_{\rm pr}, \boldsymbol{n}) \tag{4.11}$$

とすると、時刻 $\tau_{\rm pr}$ の情報 $a(\tau_{\rm pr}), a'(\tau_{\rm pr}), \phi(\tau_{\rm pr}, \boldsymbol{n}), \phi'(\tau_{\rm pr}, \boldsymbol{n})$ から

$$\begin{aligned}
a(\tau_{\rm pr}+\Delta\tau_{\rm pr}) &= a(\tau_{\rm pr}) + \Delta\tau_{\rm pr} a'(\tau_{\rm pr}) \\
a'(\tau_{\rm pr}+\Delta\tau_{\rm pr}) &= a'(\tau_{\rm pr}) + \Delta\tau_{\rm pr} F^a(\tau_{\rm pr}) \\
\phi(\tau_{\rm pr}+\Delta\tau_{\rm pr}, \boldsymbol{n}) &= \phi(\tau_{\rm pr}, \boldsymbol{n}) + \Delta\tau_{\rm pr} \phi'(\tau_{\rm pr}, \boldsymbol{n}) \\
\phi'(\tau_{\rm pr}+\Delta\tau_{\rm pr}, \boldsymbol{n}) &= \phi'(\tau_{\rm pr}, \boldsymbol{n}) + \Delta\tau_{\rm pr} F^\phi(\tau_{\rm pr}, \boldsymbol{n})
\end{aligned} \tag{4.12}$$

と得られる。注意が必要なのは、例えば F^a が ϕ' に依存し F^ϕ が a' に依存している場合、a' を発展させるために $\phi'(\tau_{\rm pr})$ を用いれば今度は ϕ' を発展させるための $a'(\tau_{\rm pr})$ が存在しなくなってしまう。このような問題の解決のためには a, a', ϕ, ϕ' の値を保存しておくダミー配列が必要になり、（特に N^3 個の要素を持つ ϕ, ϕ' によって）メモリが圧迫されるので Lattice シミュレーションでは致命的である。

4.3　Leap-Frog 法

4.3.1　概要

Leap-Frog 法はもともと 2 階微分方程式に適用するような手法である（1 階微分方程式に適用する Leap-Frog 法は知らないだけであるかも）。ただし 2 つの 1 階微分方程式に分解することは他の手法と同様である。ここでは (4.4) に対する Leap-Frog 法を説明する。Leap-Frog 法は位置 x と速度 v の参照時刻を $\frac{\Delta t}{2}$ だけずらし、交互に時間発展させる手法であり (4.4) から

$$\begin{aligned} x(t+\Delta t) &= x(t) + \Delta t v\left(t+\frac{\Delta t}{2}\right) \\ v\left(t+\frac{\Delta t}{2}\right) &= v\left(t-\frac{\Delta t}{2}\right) + \Delta t f(t,x) \end{aligned} \tag{4.13}$$

のように時間発展後の値を得ることができる。ただし一般的には $f(t,x)$ が速度 v にも依存する場合が考えられ、そのような場合には (4.13)2 式目最終項が時刻 t の情報だけで表されなくなり、Leap-Frog 法の実装が困難になる。図 4.2 に Leap-Frog 法の発展方法のイメージを示す。

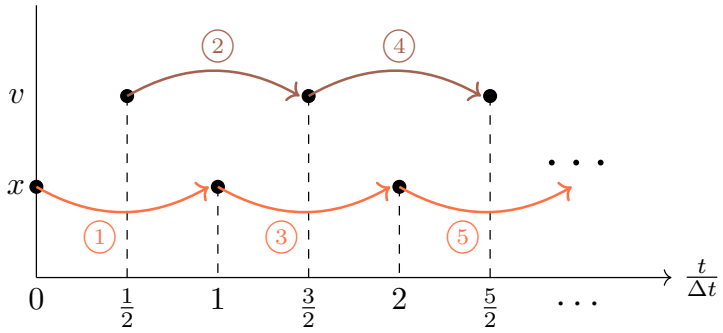

図 4.2　Leap-Frog 法のイメージ図

Leap-Frog 法の精度は 2 次であり、Runge-Kutta 法にはない「symplectic 性（エネルギー保存が良い性質）」を持っている。また、Leap-Frog 法の 4 次精度への一般化があり吉田法と呼ばれている。吉田法では

$$\begin{aligned} C &= [\ 0.675,\ -0.175,\ -0.175,\ 0.675\ ] \\ D &= [\ 1.35,\ -1.7,\ 1.35\ ] \end{aligned}$$

を用いて少々複雑な発展方法を実行する。図 4.2 に対応する発展のイメージは である。

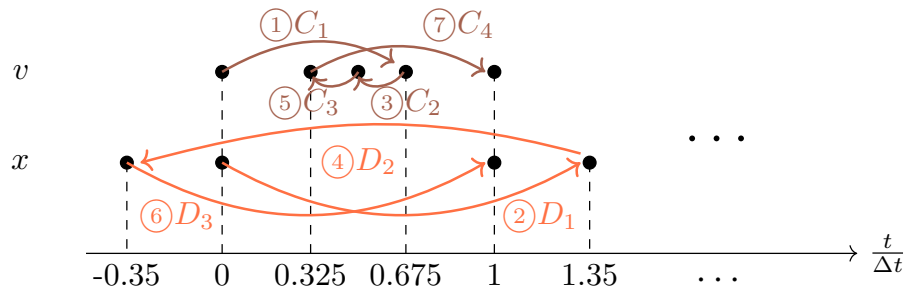

図 4.3　吉田法のイメージ図（1 回の Δt 発展）

吉田法では Δt の発展開始時に x, v の参照時刻が等しく、Leap-Frog 法のようにずらす必要がない。

4.3.2　実装

Euler 法と同様に (4.10)(4.11) に対する時間発展後の表式は

$$
\begin{aligned}
a(\tau_{\rm pr}+\Delta\tau_{\rm pr}) &= a(\tau_{\rm pr}) + \Delta\tau_{\rm pr} a'\left(\tau_{\rm pr}+\frac{\Delta\tau_{\rm pr}}{2}\right) \\
a'\left(\tau_{\rm pr}+\frac{\Delta\tau_{\rm pr}}{2}\right) &= a'\left(\tau_{\rm pr}-\frac{\Delta\tau_{\rm pr}}{2}\right) + \Delta\tau_{\rm pr} F^a(\tau_{\rm pr}) \\
\phi(\tau_{\rm pr}+\Delta\tau_{\rm pr}, \boldsymbol{n}) &= \phi(\tau_{\rm pr}, \boldsymbol{n}) + \Delta\tau_{\rm pr} \phi'\left(\tau_{\rm pr}+\frac{\Delta\tau_{\rm pr}}{2}, \boldsymbol{n}\right) \\
\phi'\left(\tau_{\rm pr}+\frac{\Delta\tau_{\rm pr}}{2}, \boldsymbol{n}\right) &= \phi'\left(\tau_{\rm pr}-\frac{\Delta\tau_{\rm pr}}{2}, \boldsymbol{n}\right) + \Delta\tau_{\rm pr} F^\phi(\tau_{\rm pr}, \boldsymbol{n})
\end{aligned}
\tag{4.14}
$$

となる。Leap-Frog 法では、Euler 法で問題となったようなダミー配列によるメモリ圧迫の心配はない。なぜなら参照時刻をずらしているおかげで交互に時間発展させることができるからである。この利点は 4.3.1 節で注意した「$f(t,x)$ が v に依存しない場合」によって保証されているものであるが、多分この条件は往々にして満たせない。（エイスケさんのコードではなんか解決できてた。忘れた。）

4.4　4 次の Runge-Kutta 法

4.4.1　概要

4 次の Runge-Kutta 法では 1 回の Δt 時間発展を行うのに、4 回の細かい時間発展を計算する。(4.5) に対する 4 次の Runge-Kuttah 法は

$$y(t+\Delta t) = y(t) + \Delta t \left(\frac{S_1}{6} + \frac{S_2}{3} + \frac{S_3}{3} + \frac{S_4}{6} \right) \tag{4.15}$$

ここで $S_1 \sim S_4$ は

$$\begin{aligned} S_1 &= f(t, y(t)) \qquad , \quad S_2 = f\left(t + \frac{\Delta t}{2}, y(t) + \frac{\Delta t}{2} S_1 \right) \\ S_3 &= f\left(t + \frac{\Delta t}{2}, y(t) + \frac{\Delta t}{2} S_2 \right) \quad , \quad S_4 = f(t + \Delta t, y(t) + \Delta t S_3) \end{aligned} \tag{4.16}$$

で求められる量である。

4.4.2　実装

上で見た最も単純な場合の Runge-Kutta 法 (4.15)(4.16) でもある程度複雑だが、Lattice シミュレーションに適用する場合にはより複雑になる。(4.10)(4.11) に対する時間発展の表式は

$$\begin{aligned} a(\tau_{\rm pr} + \Delta\tau_{\rm pr}) &= a(\tau_{\rm pr}) + \Delta\tau_{\rm pr} \left(\frac{S_1^a}{6} + \frac{S_2^a}{3} + \frac{S_3^a}{3} + \frac{S_4^a}{6} \right) \\ a'(\tau_{\rm pr} + \Delta\tau_{\rm pr}) &= a'(\tau_{\rm pr}) + \Delta\tau_{\rm pr} \left(\frac{S_1^{a'}}{6} + \frac{S_2^{a'}}{3} + \frac{S_3^{a'}}{3} + \frac{S_4^{a'}}{6} \right) \\ \phi(\tau_{\rm pr} + \Delta\tau_{\rm pr}, \boldsymbol{n}) &= \phi(\tau_{\rm pr}, \boldsymbol{n}) + \Delta\tau_{\rm pr} \left(\frac{S_1^{\phi}}{6} + \frac{S_2^{\phi}}{3} + \frac{S_3^{\phi}}{3} + \frac{S_4^{\phi}}{6} \right) \\ \phi'(\tau_{\rm pr} + \Delta\tau_{\rm pr}, \boldsymbol{n}) &= \phi'(\tau_{\rm pr}, \boldsymbol{n}) + \Delta\tau_{\rm pr} \left(\frac{S_1^{\phi'}}{6} + \frac{S_2^{\phi'}}{3} + \frac{S_3^{\phi'}}{3} + \frac{S_4^{\phi'}}{6} \right) \end{aligned} \tag{4.17}$$

である。ただし S は各変数に対して (4.16) と同様に

$$\begin{aligned} S_1^a &= a' \qquad , \quad S_2^a = a' + \frac{\Delta\tau_{\rm pr}}{2} S_1^{a'} \\ S_3^a &= a' + \frac{\Delta\tau_{\rm pr}}{2} S_2^{a'} \quad , \quad S_4^a = a' + \Delta\tau_{\rm pr} S_3^{a'} \end{aligned} \tag{4.18}$$

$$
\begin{aligned}
S_1^{a'} &= F^a(a, a', \phi, \phi') \\
S_2^{a'} &= F^a\left(a + \frac{\Delta\tau_{\rm pr}}{2} S_1^a, a' + \frac{\Delta\tau_{\rm pr}}{2} S_1^{a'}, \phi + \frac{\Delta\tau_{\rm pr}}{2} S_1^{\phi}, \phi' + \frac{\Delta\tau_{\rm pr}}{2} S_1^{\phi'}\right) \\
S_3^{a'} &= F^a\left(a + \frac{\Delta\tau_{\rm pr}}{2} S_2^a, a' + \frac{\Delta\tau_{\rm pr}}{2} S_2^{a'}, \phi + \frac{\Delta\tau_{\rm pr}}{2} S_2^{\phi}, \phi' + \frac{\Delta\tau_{\rm pr}}{2} S_2^{\phi'}\right) \\
S_4^{a'} &= F^a\left(a + \Delta\tau_{\rm pr} S_3^a, a' + \Delta\tau_{\rm pr} S_3^{a'}, \phi + \Delta\tau_{\rm pr} S_3^{\phi}, \phi' + \Delta\tau_{\rm pr} S_3^{\phi'}\right)
\end{aligned}
\tag{4.19}
$$

$$
\begin{aligned}
S_1^{\phi} &= \phi' \qquad\qquad , \quad S_2^{\phi} = \phi' + \frac{\Delta\tau_{\rm pr}}{2} S_1^{\phi'} \\
S_3^{\phi} &= \phi' + \frac{\Delta\tau_{\rm pr}}{2} S_2^{\phi'} \quad , \quad S_4^{\phi} = \phi' + \Delta\tau_{\rm pr} S_3^{\phi'}
\end{aligned}
\tag{4.20}
$$

$$
\begin{aligned}
S_1^{\phi'} &= F^{\phi}(a, a', \phi, \phi') \\
S_2^{\phi'} &= F^{\phi}\left(a + \frac{\Delta\tau_{\rm pr}}{2} S_1^a, a' + \frac{\Delta\tau_{\rm pr}}{2} S_1^{a'}, \phi + \frac{\Delta\tau_{\rm pr}}{2} S_1^{\phi}, \phi' + \frac{\Delta\tau_{\rm pr}}{2} S_1^{\phi'}\right) \\
S_3^{\phi'} &= F^{\phi}\left(a + \frac{\Delta\tau_{\rm pr}}{2} S_2^a, a' + \frac{\Delta\tau_{\rm pr}}{2} S_2^{a'}, \phi + \frac{\Delta\tau_{\rm pr}}{2} S_2^{\phi}, \phi' + \frac{\Delta\tau_{\rm pr}}{2} S_2^{\phi'}\right) \\
S_4^{\phi'} &= F^{\phi}\left(a + \Delta\tau_{\rm pr} S_3^a, a' + \Delta\tau_{\rm pr} S_3^{a'}, \phi + \Delta\tau_{\rm pr} S_3^{\phi}, \phi' + \Delta\tau_{\rm pr} S_3^{\phi'}\right)
\end{aligned}
\tag{4.21}
$$

である。ただし a, a', ϕ, ϕ' は全て時刻 $\tau_{\rm pr}$ における値であり、(4.20)(4.21) の S は本来 $\boldsymbol{n}$ を引数に持ち各格子点について定義される量である。

Euler 法でも述べたように、正しく時間発展させるためには a, a', ϕ, ϕ' の値を保存しておくダミー配列が必要になる。以下で (4.17) から (4.21) の計算をする手順を説明しつつ、必要になるダミー配列の要素数も求める。

①まず時刻 $\tau_{\rm pr}$ の情報 a, a', ϕ, ϕ' を持っているとして、$S_1^a, S_1^{a'}, S_1^{\phi}, S_1^{\phi'}$ を計算する。この時、計算した S_1 の値を保存しておくために $2N^3 + 2$ 個の変数を持つダミー配列（$S_1^a, S_1^{a'}$ が 1 個ずつ、$S_1^{\phi}, S_1^{\phi'}$ が N^3 個ずつ）が必要になる。

$$
a, a', \phi, \phi' \quad \rightarrow \quad S_1^a, S_1^{a'}, S_1^{\phi}, S_1^{\phi'} \text{ (Dummy List 1)}
$$

②元々持っていた時刻 $\tau_{\rm pr}$ の情報 a, a', ϕ, ϕ' と先ほど計算した $S_1^a, S_1^{a'}, S_1^{\phi}, S_1^{\phi'}$ から $S_2^a, S_2^{a'}, S_2^{\phi}, S_2^{\phi'}$ を計算する。この時、計算した S_2 の値を保存しておくために $2N^3 + 2$ 個の変数を持つ新たなダミー配列が必要になる。ここで時刻 $\tau_{\rm pr}$ の情報 a, a', ϕ, ϕ' を書き換えていけないのは (4.17) や今後の S の計算でこれらの情報が必要になるからで、

$S_1^a, S_1^{a'}, S_1^\phi, S_1^{\phi'}$ を書き換えていけないのは S_2 の計算で S_1 の情報が必要だからである。

$$a, a', \phi, \phi' \text{ and } S_1^a, S_1^{a'}, S_1^\phi, S_1^{\phi'} \quad \to \quad S_2^a, S_2^{a'}, S_2^\phi, S_2^{\phi'} \text{ (Dummy List 2)}$$

③元々持っていた時刻 $\tau_{\rm pr}$ の情報 a, a', ϕ, ϕ' と先ほど計算した $S_2^a, S_2^{a'}, S_2^\phi, S_2^{\phi'}$ から $S_3^a, S_3^{a'}, S_3^\phi, S_3^{\phi'}$ を計算する。この時 $S_1^a, S_1^{a'}, S_1^\phi, S_1^{\phi'}$ は (4.17) の時間発展を除けばもう使わないので、S_1 の情報を $2N^3+2$ 個の変数を持つ新たなダミー配列に保存し、S_3 の情報を元々 S_1 を保存していたダミー配列に上書きする。

$$\begin{aligned} S_1^a, S_1^{a'}, S_1^\phi, S_1^{\phi'} \text{ (Dummy List 1)} \quad &\to \quad \frac{S_1^a}{6}, \frac{S_1^{a'}}{6}, \frac{S_1^\phi}{6}, \frac{S_1^{\phi'}}{6} \text{ (Dummy List 3)} \\ a, a', \phi, \phi' \text{ and } S_2^a, S_2^{a'}, S_2^\phi, S_2^{\phi'} \quad &\to \quad S_3^a, S_3^{a'}, S_3^\phi, S_3^{\phi'} \text{ (Dummy List 1)} \end{aligned}$$

④元々持っていた時刻 $\tau_{\rm pr}$ の情報 a, a', ϕ, ϕ' と先ほど計算した $S_3^a, S_3^{a'}, S_3^\phi, S_3^{\phi'}$ から $S_4^a, S_4^{a'}, S_4^\phi, S_4^{\phi'}$ を計算する。この時、先程と同じように S_2 の情報を S_1 の情報が入っているダミー配列に追加し、S_4 の情報を元々 S_2 を保存していたダミー配列に上書きする。

$$\begin{aligned} S_2 \text{ (Dummy List 2)} \quad &\to \quad \frac{S_1}{6} + \frac{S_2}{3} \text{ (Dummy List 3)} \\ a, a', \phi, \phi' \text{ and } S_3^a, S_3^{a'}, S_3^\phi, S_3^{\phi'} \quad &\to \quad S_4^a, S_4^{a'}, S_4^\phi, S_4^{\phi'} \text{ (Dummy List 2)} \end{aligned}$$

⑤最後に S_3, S_4 の情報を S_1 と S_2 の情報が入っているダミー配列に追加し、この配列と時刻 $\tau_{\rm pr}$ の情報 a, a', ϕ, ϕ' を用いて (4.17) を通して時刻 $\tau_{\rm pr} + \Delta\tau_{\rm pr}$ の情報を得る。これで得られる情報で時刻 $\tau_{\rm pr}$ の配列を上書きして良い。

$$\begin{aligned} S_3,\ S_4 \text{ (Dummy List 1, 2)} \quad &\to \quad \frac{S_1}{6} + \frac{S_2}{3} + \frac{S_3}{3} + \frac{S_4}{6} \text{ (Dummy List 3)} \\ &\to \quad a, a', \phi, \phi' \text{ at } \tau_{\rm pr} + \Delta\tau_{\rm pr} \text{ (Main List)} \end{aligned}$$

上記のような手順で $\Delta\tau_{\rm pr}$ の時間発展を計算するが、ここで必要になったダミー配列の容量（メモリ）は変数 a, a', ϕ, ϕ' の 3 つ分である。つまり 4 次の Runge-Kutta 法を実行するためには、扱う変数（スケール因子と場の値）の 4 倍のメモリが必要になる。このメモリ圧迫は Lattice シミュレーションでは致命的だよね（泣）。

第 5 章

アウトプット

本章では Lattice シミュレーションで得られたプログラム上の値から、物理量を取り出すための方法を説明する。5.1 節で取り出す背景量とその取り出し方を説明し、5.3 節でゆらぎの PDF（確率分布関数）の計算方法を説明し、5.3 節で物理的なパワースペクトルの計算方法を説明する。

5.1 背景量

Lattice シミュレーションにおいて場に関する背景量は、各格子点での値を足し合わせた後に格子点数で割るという格子平均で計算される。計算式は

$$\langle f \rangle \equiv \frac{1}{N^3} \sum_{\boldsymbol{n}} f(t, \boldsymbol{n}) \tag{5.1}$$

である。以降はよく出力される量について、プログラム上の値から計算する方法を記す。

- e-folds

Lattice シミュレーション開始から経過した e-folds は

$$N_e = \ln \left(\frac{a}{a_{\mathrm{ini}}} \right) = \ln a \tag{5.2}$$

・Hubble パラメータ

出力する Hubble パラメータはインフラトン質量で無次元化する。(1.7) を用いて

$$\frac{H}{m} = \frac{\partial_t a}{ma} = \frac{\partial_\tau a}{ma^2} = \frac{ma^s a'}{ma^2} = a^{s-2}a' \tag{5.3}$$

・場の平均値

場 ϕ の値はリスケーリングの影響を受けないので (5.1) の格子平均を用いて

$$\bar{\phi} = \langle\phi\rangle \tag{5.4}$$

・場の微分の平均値

場 ϕ の微分は時間微分からリスケーリングの影響を受けるので (1.8) を考慮する。また Hubble パラメータと同様にインフラトン質量で無次元化して

$$\frac{\overline{\partial_t\phi}}{m} = \frac{\langle\partial_\tau\phi\rangle}{ma} = \frac{ma^s\langle\phi'\rangle}{ma} = a^{s-1}\langle\phi'\rangle \tag{5.5}$$

・キュムラント

確率変数 X とその平均値 $\overline{X}$ に対して、$\langle(X-\overline{X})^n\rangle$ に関する量を n 次のキュムラントと呼ぶ。1 次のキュムラントは平均値、2 次のキュムラントは分散、3 次のキュムラントは歪度 (skewness)、4 次のキュムラントは尖度 (kurtosis) に関係する。ガウス分布の 3 次以上のキュムラントは全て 0 となるため（ただし偶数次のキュムラントに対しては特別な定義が必要)[*1]、キュムラントによって非ガウス性を数値化することができる。

Lattice シミュレーションにおいては統計的平均の $\langle\cdots\rangle$ が空間平均と等しいと仮定（近似）するので、場 ϕ の n 次のキュムラントは

$$n \text{ th cumulant} = \frac{1}{N^3}\sum_{\boldsymbol{n}}(\phi-\bar{\phi})^n \tag{5.6}$$

[*1] 計算は付録を参照

で計算する。ちなみに（エルゴード性を完全に信頼した場合の）統計的公式、例えば $\sigma^2 = \langle X^2 \rangle - \langle X \rangle^2$、を用いて Lattice シミュレーションのキュムラントを計算すると 3 次以上で正しい結果を出さなくなってしまう[*2]。よって (5.6) のような愚直な格子平均で計算する。（多分エルゴード性を信頼する計算を何回も行うことによるもの、あるいは別の原因）

5.2　ゆらぎの PDF

場 ϕ のゆらぎの PDF(Probability Distribution Function) はその定義から、実空間でのゆらぎの値を数えれば良い。先に平均 $\overline{\phi}$ と標準偏差 $\sigma_\phi = \sqrt{\langle (\phi - \overline{\phi})^2 \rangle}$ を (5.4)(5.6) に従って計算する。次に標準偏差で規格化した各格子点でのゆらぎの値

$$\frac{\delta\phi(\boldsymbol{n})}{\sigma_\phi} = \frac{\phi(\boldsymbol{n}) - \overline{\phi}}{\sigma_\phi} \tag{5.7}$$

を計算し、どの数値範囲に属するかを数える。この数値範囲あるいは bin 幅は任意だが、私は 0.1 で計算した。つまり $[0, 0.1), [0.1, 0.2), \cdots$ のような bin でカウントした。具体的なコードの処理は第 I,II 部を参照すべき。

[*2] 計算は付録を参照

5.3　パワースペクトル

本節では場 f の（無次元）パワースペクトルを計算する方法を説明する。まず LATTICEEASY[1] で用いられている計算方法を説明し、続いて [2] で改良された計算方法を見、その後オリジナルの計算方法を見る。無次元パワースペクトルを $P(k)$ と表記し、パワースペクトルを $\mathcal{P}(k)$ と表記する。

まずは LATTICEEASY で用いられている計算方法を説明する。場 f に対して解析的なパワースペクトルは、フーリエモード $f(k)$ を用いて

$$\mathcal{P}_f(k) = |f(k)|^2$$

で計算される。フーリエモードの関係式 (2.13) を考慮すると、解析的なパワースペクトルを再現するためには、Lattice 上の離散フーリエモード $f(\kappa)$ を用いて

$$\mathcal{P}_f(k) = \frac{dx^6}{L^3}|f(\kappa)|^2 = \frac{dx^6_{\rm pr}}{m^3 L^3_{\rm pr}}|f(\kappa)|^2 \tag{5.8}$$

ただし、解析的にはフーリエモードが運動量ベクトルの大きさ k のみに依存する場合が多く（絶対かも）、一方で Lattice 上ではフーリエモードは逆格子ベクトルの方向にも依存する $f(\boldsymbol{\kappa})$。

このことを考慮して、Lattice 上で逆格子ベクトルの大きさのみに依存する量を取り出すために逆格子ベクトルの方向で平均を取る。つまり

$$\mathcal{P}_f(k) = \frac{dx^6_{\rm pr}}{m^3 L^3_{\rm pr}} \frac{\sum_\kappa |f(\boldsymbol{\kappa})|^2}{\sum_\kappa} \tag{5.9}$$

で $\sum_\kappa$ は大きさ κ を持つ任意の方向の逆格子ベクトル $\boldsymbol{\kappa}$ での和を表す。この和は連続空間では半径 k の球面積分に対応し、格子上は離散化されているため幅を持たせて球殻で平均を取る。平均を取る球殻の幅は逆格子ベクトルの大きさの最小値 $\frac{2\pi}{L}$ とする。つまり、

$$\sum_\kappa = \sum_{[\frac{2\pi}{L}i, \frac{2\pi}{L}(i+1))}, \qquad |\boldsymbol{\kappa}| \times \frac{L}{2\pi} \in [i, i+1) \tag{5.10}$$

と表せる。(2.1) より、逆格子ベクトルの係数 $\frac{2\pi}{L}$ を除いたベクトル $\boldsymbol{m}$ で表せば

$$\sum_{|\boldsymbol{m}|} = \sum_{[i,i+1)}, \qquad |\boldsymbol{m}| \in [i, i+1) \tag{5.11}$$

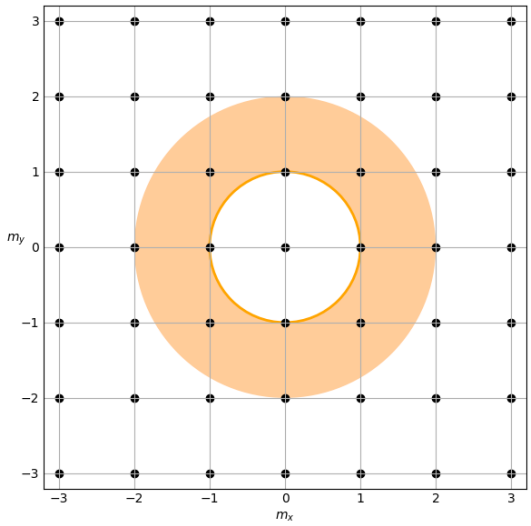

図 5.1　(5.11) の和の取り方のイメージ。この図では空間が 2 次元の場合で、$|\boldsymbol{m}| \in [1, 2)$ の領域を表している。

となる。図 5.1 に和の取り方のイメージ図を示す。無次元パワースペクトルに直すには逆格子ベクトルの 3 乗を掛け算してやれば良い。

最終的に、LATTICEEASY[1] で用いられている無次元パワースペクトルの計算方法は

$$P_f(k) = P_f(\kappa) = \frac{dx_{\mathrm{pr}}^6}{L_{\mathrm{pr}}^3} \frac{\kappa_{\mathrm{pr}}^3}{2\pi^2} \frac{\sum |f(\boldsymbol{\kappa})|^2}{\sum} \tag{5.12}$$

$$\kappa_{\mathrm{pr}} = \frac{\sum |\boldsymbol{\kappa}_{\mathrm{pr}}|}{\sum} \quad , \quad \sum = \sum_{|\boldsymbol{m}| \in [i, i+1)}$$

である。

次に [2] で用いられている計算方法を説明する。2.3 節で述べたように、物理的な運動量に対応するものは Lattice 上の有効運動量ベクトル $\boldsymbol{k}_{\mathrm{eff}}$ である（と信じている）。このことを考慮し、(5.12) において平均を取った量 $\frac{\sum |f|^2}{\sum}$ は物理的な運動量 k_{eff} における情報だとした。つまり、無次元パワースペクトルは

$$
\begin{aligned}
P_f(k) = & P_f(k_{\mathrm{eff}}) = \frac{dx_{\mathrm{pr}}^6}{L_{\mathrm{pr}}^3} \frac{k_{\mathrm{eff,pr}}^3}{2\pi^2} \frac{\sum |f(\boldsymbol{\kappa})|^2}{\sum} \\
k_{\mathrm{eff,pr}} = & \frac{\sum |\boldsymbol{k}_{\mathrm{eff,pr}}|}{\sum} \quad , \quad \sum = \sum_{|\boldsymbol{m}| \in [i, i+1)}
\end{aligned}
\tag{5.13}
$$

(5.12) からの計算上の変更点は、見ている運動量スケールを $\kappa \to k_{\mathrm{eff}}$ と読み替えただけで、κ^3 の部分と横軸の κ を k_{eff} にすれば良い。$\sum |f|^2$ の計算の仕方は変更していない。

次にオリジナルの計算方法を説明する。(5.13) では変更されなかった $\sum |f|^2$（厳密には $\sum_{|\boldsymbol{m}| \in [i, i+1)}$ の和の取り方）を変更する。もともとこの和は、運動量の大きさ k に幅を持たせて球殻で平均を取るためのものであった。運動量 k に対応するものが有効運動量 k_{eff} であることを考慮すれば、k_{eff} に幅を持たせて球殻で平均を取るのが良いと考えられる。よって

$$
\begin{aligned}
P_f(k) = & P_f(k_{\mathrm{eff}}) = \frac{dx_{\mathrm{pr}}^6}{L_{\mathrm{pr}}^3} \frac{k_{\mathrm{eff,pr}}^3}{2\pi^2} \frac{\sum |f(\boldsymbol{k}_{\mathrm{eff}})|^2}{\sum} \\
k_{\mathrm{eff,pr}} = & \frac{\sum |\boldsymbol{k}_{\mathrm{eff,pr}}|}{\sum} \quad , \quad \sum = \sum_{|\boldsymbol{k}_{\mathrm{eff}}| \times \frac{L}{2\pi} \in [i, i+1)}
\end{aligned}
\tag{5.14}
$$

と表される。（(5.13) と (5.14) の結果はあまり変わらなかった）

第 II 部

単一場インフレーション

第 6 章

セットアップと実装表式

本章では、単一場インフレーションの Lattice シミュレーションで用いた詳細なセットアップと、実際に用いた実装表式を説明する。第 I 部で既に説明した内容については触れない。6.1 節で用いたセットアップのまとめを説明、6.2 節で初期条件の説明、6.3 節で実装する発展方程式の導出、6.4 節で出力する量の説明、6.5 節でインフラトンポテンシャルの計算、6.6 節でインプットするべきパラメータの説明をする。

ここでは概念的なセットアップと理論的な計算を記すため、具体的（あるいは技術的）なコードの解説は 7 章と 8 章を参照すべき。

6.1　セットアップのまとめ

ここで扱うのは単一場インフレーションでラグランジアンは

$$\mathcal{L} = \sqrt{-g}\left[-\frac{1}{2}g^{\mu\nu}(\partial_\mu\phi)(\partial_\nu\phi) - V(\phi)\right] \tag{6.1}$$

である。インフラトン場 ϕ は実スカラー場の場合を考えているので、プログラム上で扱う変数はインフラトン場 ϕ・その時間微分 ϕ'・スケール因子 a・その時間微分 a' である。全格子点数が N^3 個であることを考慮すれば、変数の数は $2N^3 + 2$ 個である。

インフラトンゆらぎは場の値 ϕ, ϕ' に含まれているが、計量ゆらぎについては小さいと考えて無視し、平坦な FLRW 計量を考える。

$$ds^2 = a^2[-d\tau^2 + \delta_{ij}dx^i dx^j] \tag{6.2}$$

これは線形摂動論において計量ゆらぎは slow-roll パラメータに比例するので、インフラトンが十分 slow-roll していれば計量ゆらぎは無視できるからである [2]。今は計量ゆらぎ

を無視していて、この近似のためにシミュレーション中は十分 slow-roll している必要がある。

空間の有限化に伴うインフラトン場の境界条件は周期的境界条件 (1.2) を課し、微分演算子の離散化と対応する有効運動量ベクトルの定義は (2.9)(2.10)

$$\nabla^2 f(\boldsymbol{n}) = \frac{1}{dx^2} \sum_{i=x,y,z} \sum_{c=\pm 1} [f(\boldsymbol{n} + c\boldsymbol{e}_i) - f(\boldsymbol{n})] \tag{6.3}$$

$$[k_{\mathrm{eff}}(\boldsymbol{m})]_i = \frac{2}{dx} \sin\left(\frac{\pi m_i}{N}\right) \tag{6.4}$$

を用いる。ここで解像度の高いラプラシアンの定義を用いた理由は、後に見るように発展方程式に 1 階微分が現れずラプラシアンのみが現れるからで、1 階微分の離散化を定義する必要がないためである。

時間発展の方法は 4 次の Runge-Kutta 法（4.4 節）を用いる。

6.2　初期条件

6.2.1　背景量

背景量の初期値は 3.1 節で説明した通りである。ただし、Hubble パラメータの初期値に用いるエネルギー密度の表式は明らかにしていなかったのでここで見る。前述したように、エネルギー密度に対するゆらぎの寄与は無視するので、運動エネルギーとポテンシャルエネルギー（つまりインフラトン場に関する背景量 $\overline{\phi}_{\mathrm{ini}}, \overline{\phi'}_{\mathrm{ini}}$ のみ）から決まり

$$\rho_{\mathrm{pr,ini}} = \frac{1}{2}\overline{\phi'}^2_{\mathrm{ini}} + V_{\mathrm{pr}}(\overline{\phi}_{\mathrm{ini}}) \tag{6.5}$$

である。ここでスケール因子の初期値が 1 であることを用いた。ゆらぎの寄与を無視しているので、インプットパラメータである $\overline{\phi}_{\mathrm{ini}}, \overline{\phi'}_{\mathrm{ini}}$ から簡単に計算できる。よってハッブルパラメータの初期値は

$$H_{\mathrm{ini}} = m\sqrt{\frac{\overline{\phi'}^2_{\mathrm{ini}} + 2V_{\mathrm{pr}}(\overline{\phi}_{\mathrm{ini}})}{6}} \tag{6.6}$$

6.2.2　インフラトンゆらぎ

3.3 節で量子ゆらぎを再現するインフラトンゆらぎの実装方法を見た。私は単一場インフレーションのシミュレーション「SFInflation」で、モード関数として Hankel 関数を用いる。これは 3.3 節で用いた sub-horizon 極限のモード関数 (3.8) の一般化であり、super-horizon のゆらぎも考慮できるモード関数である。ただし Hankel 関数を用いた場合でも、slow-roll 近似を課しているため Lattice シミュレーション開始時に十分 slow-roll かつ線形摂動論が有効でなければならないことは変わらない。

Hankel 関数を用いた場合の解析的なインフラトンパワースペクトルは

$$\mathcal{P}_{\delta\phi}(k) = \frac{-\pi\tau}{4a^2}|H_\nu^{(1)}(-k\tau)|^2 \tag{6.7}$$

であり、インフラトンゆらぎのモード関数は

$$\delta\phi(k) = \frac{\sqrt{-\pi\tau}}{2a}H_\nu^{(1)}(-k\tau) \tag{6.8}$$

である。ここで ν は slow-roll の 0 次近似で評価して

$$\nu = \sqrt{\frac{9}{4} - \frac{m_{\rm eff}^2}{H^2}} = \sqrt{\frac{9}{4} - \frac{m^2}{H_{\rm ini}^2}\frac{m_{\rm eff}^2}{m^2}} \quad , \quad m_{\rm eff}^2 = \frac{d^2V(\phi)}{d\phi^2} \tag{6.9}$$

を用いる[*1]。Hankel 関数の引数を χ で表すと (2.12)(3.3)(3.4) より

$$\chi = \frac{k_{\rm eff}}{H_{\rm ini}} = \frac{k_{\rm eff,pr}}{a'_{\rm ini}} = k_{\rm eff}\sqrt{\frac{3}{\rho_{\rm pr,ini}}} \tag{6.10}$$

のようにプログラム上の値で与えられる。

この解析解を (3.16) に用いて、プログラム上で実装するインフラトンゆらぎ（のフーリエモード）の初期条件は

$$\begin{aligned}\delta\phi(\boldsymbol{\kappa}) &= \frac{1}{\sqrt{2}}|\delta\phi|_{\rm pr}\left(\sqrt{-\ln X_1}e^{i2\pi Y_1} + \sqrt{-\ln X_2}e^{i2\pi Y_2}\right)\\ |\delta\phi|_{\rm pr} &= m\frac{L_{\rm pr}^{\frac{3}{2}}}{dx_{\rm pr}^3}\frac{1}{2}\sqrt{\frac{\pi}{a'_{\rm ini}}}|H_\nu(\chi)|\end{aligned} \tag{6.11}$$

ただし Hankel 関数の第 1 種と第 2 種を表す上付き添字 (1), (2) は、この式では違いが現れないので省略した。第 1 種 Hankel 関数と第 2 種 Hankel 関数は各々 sub-horizon 極限

[*1] 計算は付録を参照

で $e^{\mp ik\tau}$ に比例するため、3.3.3 節の等方的初期条件から両方の寄与を考慮している。これは時間微分のゆらぎを考える時に重要になる。

インフラトン場の時間微分のゆらぎは 3.3.4 節のようにモード関数の微分から求める。Hankel 関数は Bessel 関数 J_ν と Nuemann 関数 N_ν を用いて

$$H_\nu^{(1)} = J_\nu + iY_\nu \quad , \quad H_\nu^{(2)} = J_\nu - iY_\nu$$

と書け、これらの満たす微分漸化式を用いて計算される (3.17) に対応するプログラム上の微分ゆらぎは

$$\begin{aligned}\delta\phi'(\boldsymbol{\kappa}) = &\left[\left(\nu - \frac{3}{2}\right) a'_{\rm ini} - k_{\rm eff,pr}\frac{J_\nu J_{\nu-1} + Y_\nu Y_{\nu-1}}{|H_\nu|^2}\right]\delta\phi(\boldsymbol{\kappa}) \\ &+ ik_{\rm eff,pr}\frac{J_{\nu-1}Y_\nu + J_\nu Y_{\nu-1}}{|H_\nu|^2}\frac{|\delta\phi|_{\rm pr}}{\sqrt{2}}\left[\sqrt{-\ln X_1}e^{i2\pi Y_1} - \sqrt{-\ln X_2}e^{i2\pi Y_2}\right]\end{aligned} \tag{6.12}$$

ここで Hankel 関数・Bessel 関数・Neumann 関数の引数は全て χ である[*2]。

6.3　発展方程式

本節では Lattice シミュレーションで実装する発展方程式を記す（導出は付録を参照）。(6.1) のラグランジアンから Euler-Lagrange 方程式を求めるだけだが、摂動論とは異なり Lattice では場はそのまま扱う。つまり ϕ の発展方程式は

$$\partial_\mu \frac{\partial\mathcal{L}}{\partial(\partial_\mu\phi)} - \frac{\partial\mathcal{L}}{\partial\phi} = 0$$

であり、$\overline{\phi}$ と $\delta\phi$ の方程式に分解する必要はない。上式から得られるインフラトン場 ϕ の発展方程式は

$$\partial_\tau^2\phi + 2\mathcal{H}\partial_\tau\phi - \nabla^2\phi + a^2\frac{dV}{d\phi} = 0 \tag{6.13}$$

プログラム上の値に直すと

$$\phi'' = -(s+2)\frac{a'}{a}\phi' + \frac{\nabla^2_{\rm pr}\phi}{a^{2s}} - a^2\frac{dV_{\rm pr}}{d\phi} \tag{6.14}$$

この式の右辺が (4.11) の $F^\phi(\tau_{\rm pr}, \boldsymbol{n})$ に対応する。またこの発展方程式は各格子点の $\phi(\tau_{\rm pr}, \boldsymbol{n})$ に対して定義されるため、方程式の数で言えば N^3 個ある。

[*2] 計算は付録を参照

スケール因子の発展方程式は Einstein 方程式であり、今回は空間成分の方程式（一般に Friedmann 方程式と呼ばれない方）を用いる。なぜなら自然に 2 階微分が現れるからである（Axion-U(1) インフレーションの文脈では別の理由もあるが）。

$$\frac{\partial_\tau^2 a}{a} = \frac{a^2}{6}(\rho - 3p) \tag{6.15}$$

プログラム上の値に直すと

$$a'' = -s\frac{(a')^2}{a} + \frac{a^3}{6}\left\langle \rho_{\rm pr} - 3p_{\rm pr} \right\rangle \tag{6.16}$$

ここで $\langle \cdots \rangle$ は格子平均 (5.1) であり、エネルギー密度と圧力は

$$\rho_{\rm pr} = \frac{(\phi')^2}{2a^2} - \frac{\phi \nabla_{\rm pr}^2 \phi}{2a^{2s+2}} + V_{\rm pr} \tag{6.17}$$

$$p_{\rm pr} = \frac{(\phi')^2}{2a^2} + \frac{\phi \nabla_{\rm pr}^2 \phi}{6a^{2s+2}} - V_{\rm pr} \tag{6.18}$$

である。

スケール因子の発展に用いなかった Friedmann 方程式

$$\mathcal{H}^2 = \frac{a^2}{3}\rho \tag{6.19}$$

は時間発展に伴って破れないかどうか調べておく。具体的には

$$E \equiv \frac{3\mathcal{H}^2}{a^2\rho} = \frac{3\left(\frac{ma^s a'}{a}\right)^2}{a^2 m^2 a^{2s}\left\langle \rho_{\rm pr} \right\rangle} = \frac{3(a')^2}{a^4\left\langle \rho_{\rm pr} \right\rangle} \tag{6.20}$$

という値が 1 から外れないかどうかを調べる。このような、整合性のために調べておく条件（式）を「Consistency Condition」と呼ぶこととする。

6.4　アウトプット

本節では出力する物理量を説明する。5 章で述べた e-folds・Hubble パラメータ・インフラトン場に関する平均値・インフラトン場に関するキュムラント・インフラトンゆらぎの PDF・インフラトンゆらぎのパワースペクトルに加え、プログラム時間 $\tau_{\rm pr}$・スケール因子 a・slow-roll パラメータを出力する。$\tau_{\rm pr}, a$ はプログラム上の値をそのまま出力するが、slow-roll パラメータは少々式変形が必要になる。

slow-roll パラメータ ϵ, η として信じる定義は

$$\epsilon \equiv -\frac{\partial_t H}{H^2} \quad , \quad \eta \equiv \frac{d\ln\epsilon}{dN_e} \tag{6.21}$$

とする。これらは時間のみに依存する背景量であることを注意しておく（定義というか出発点となる表式を変えると、どのタイミングで格子平均を取るか変わる。結果がそのタイミングに依らない場合に良い Lattice シミュレーションと言えるのだろうけど、、、）。slow-roll パラメータは無次元量なのでリスケーリングの影響を受けず、$\epsilon = \epsilon_{\rm pr}, \eta = \eta_{\rm pr}$ である。注意が必要なのは、「リスケーリングの影響を受けないこと」と「プログラム上の値で書き直すこと」は別である。

slow-roll パラメータ ϵ は Consistency Condition(6.20) が十分良く成立していると仮定して

$$\epsilon = \frac{3}{2}\left(1 + \frac{\langle p_{\rm pr}\rangle}{\langle \rho_{\rm pr}\rangle}\right) \tag{6.22}$$

(6.21) から解析的な（not Lattice の）計算を進めた表式 $\epsilon = \frac{3}{2}\left(1+\frac{p}{\rho}\right)$ を定義とすると、Lattice で計算するべきは $\epsilon = \frac{3}{2}\left(1+\langle\frac{p}{\rho}\rangle\right)$ のように格子平均のタイミングが異なる。

slow-roll パラメータ η は

$$\begin{aligned}\eta = &\left[-3 - 3\frac{\langle(\phi')^2\rangle}{a^2\langle\rho_{\rm pr}+p_{\rm pr}\rangle}\right] + 2\epsilon - 2\frac{a}{a'}\frac{1}{\langle\rho_{\rm pr}+p_{\rm pr}\rangle}\langle\frac{dV_{\rm pr}}{d\phi}\phi'\rangle \\ &- \frac{\langle\phi\nabla^2_{\rm pr}\phi\rangle}{3a^{2s+2}\langle\rho_{\rm pr}+p_{\rm pr}\rangle} + \frac{4}{3a^{2s+1}a'}\frac{\langle\phi'\nabla^2_{\rm pr}\phi\rangle}{\langle\rho_{\rm pr}+p_{\rm pr}\rangle}\end{aligned} \tag{6.23}$$

ここで支配的な寄与をするのは 1 行目である。

6.5　インフラトンポテンシャル

本節ではインフラトンポテンシャル $V(\phi)$ の具体的な表式と、対応するプログラム上の表式 $V_{\rm pr}, \frac{dV_{\rm pr}}{d\phi}, \frac{d^2V_{\rm pr}}{d\phi^2}$ を求める。$V_{\rm pr}$ はエネルギー密度 (6.17) と圧力 (6.18) で用いる。$\frac{dV_{\rm pr}}{d\phi}$ はインフラトン場の発展方程式 (6.14) で用いる。$\frac{d^2V_{\rm pr}}{d\phi^2}$ はゆらぎの初期条件 (6.9) で用いる。

6.5.1　Chaotic ポテンシャル

$$V(\phi) = \frac{1}{2}m^2\phi^2 \tag{6.24}$$

に対して (1.11) より

$$\begin{aligned} V_{\rm pr} &= \frac{\phi^2}{2a^{2s}} \\ \frac{dV_{\rm pr}}{d\phi} &= \frac{\phi}{a^{2s}} \\ \frac{d^2V_{\rm pr}}{d\phi^2} &= \frac{1}{a^{2s}} \quad \overset{\text{initial}}{\to} \quad 1 \end{aligned} \tag{6.25}$$

ただし 2 階微分はゆらぎの初期条件で用いるので (3.1) を使った。

6.5.2　Step ポテンシャル

$$V(\phi) = \frac{1}{2}m^2\phi^2\left[1 + s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right] \tag{6.26}$$

に対しては、双曲線関数の位相を $\theta = \frac{\phi-\phi_{\rm step}}{d}$ と表記すると

$$\begin{aligned} V_{\rm pr} &= \frac{\phi^2}{2a^{2s}}(1 + s\tanh\theta) \\ \frac{dV_{\rm pr}}{d\phi} &= \frac{\phi}{a^{2s}}(1 + s\tanh\theta) + \frac{\phi}{2a^{2s}}\frac{\phi}{d}\frac{s}{\cosh^2\theta} \\ \frac{d^2V_{\rm pr}}{d\phi^2} &= 1 + s\tanh\theta + \frac{s\frac{\phi}{d}}{\cosh^2\theta}\left(2 - \frac{\phi}{d}\tanh\theta\right) \end{aligned} \tag{6.27}$$

ただし 2 階微分は既に (3.1) を用いた表式である[*3]。

[*3] 計算は付録を参照

6.6　インプットパラメータ

本節では単一場インフレーションの Lattice シミュレーションを実行するのに必要なインプットパラメータを説明する。「型 変数名：説明」のフォーマットで説明する。型がないものはマクロ名である（マクロの説明は省略する）。重要なものや、よく値を操作するものは赤字で記載する。
Lattice 特有のパラメータは

- double L：立方領域の一辺の（共動的な）プログラム上の長さの値
- int N：一辺の格子点数（2 の累乗）
- int num_thread：OpenMP で用いるスレッド数（コンピュータごとに可能な最大値が異なるので注意）

物理的なパラメータは

- MODEL：インフラトンポテンシャルの選択（0 → chaotic, 1 → step）
- double m：換算プランク質量単位のインフラトン質量
- NDIMS：空間次元、3 にしておくべき
- int nflds：場の数、1 にしておくべき
- double initfield[]：インフラトン場の背景量初期値（換算プランク質量単位）
- double initderivs[]：インフラトン場微分の背景量初期値（換算プランク質量単位）
- bool hankel：インフラトンゆらぎの初期条件（true → Hankel 関数, false → 指数関数）
- double kcutoff：運動量カットオフ（多分使用しない、LATTICEEASY の名残）

シミュレーションに必要なパラメータは

- double dt：1 回の時間発展に用いる時間間隔のプログラム上の値 $\Delta\tau_{\rm pr}$
- double tf：シミュレーション終了時のプログラム上の時刻
- int seed：一様乱数 X, Y を生成する時のシード値

- int continue_run：Lattice シミュレーションを続けて行う場合に、何回目のシミュレーションかを表す数（基本 0 にしておく、LATTICEEASY の名残）
- char alt_extension[]：出力する dat ファイルの名前に付ける文字（設定しなくて

良い）

- int noutput_times：（背景量の）値の出力を行う回数
- int print_interval：CLI に現在の $\tau_{\rm pr}$ を出力する際の（我々の世界での）時間間隔
- int screen_updates：CLI に現在の $\tau_{\rm pr}$ を出力するかどうか（1 → 出力する）
- double checkpoint_interval：dat ファイルを flush する（書き出す）プログラム上の時間間隔
- double store_lattice_times[]：触らぬ神に祟りなし、LATTICEEASY の名残

- int smeansvars：インフラトン場に関する背景量 (means_0.dat) を出力するかどうか（1 → 出力する）
- int sexpansion：スケール因子に関する量 (sf_0.dat) を出力するかどうか（1 → 出力する）
- double t_start_output：出力をいつから始めるか（0 にしておくべき）
- int senergy：インフラトン場のエネルギーに関する量 (energy_0.dat) を出力するかどうか（1 → 出力する）
- double tenergy：インフラトン場のエネルギーに関する量の出力をいつから始めるか
- int sspectra：インフラトンゆらぎのパワースペクトル (spectra0_0.dat, spectra_eff0_0.dat) を出力するかどうか（1 → 出力する）
- double tspectra：インフラトンゆらぎのパワースペクトルの出力をいつから始めるか
- int spdf：インフラトンゆらぎの PDF(pdf0_0.dat) を出力するかどうか（1 → 出力する、ただし sspectra=1 でなければいけない）
- int pdfrange：インフラトンゆらぎの PDF の横軸 $\frac{\delta\phi}{\sigma}$ の最大値

「model.hpp」ヘッダファイルでインプットするパラメータは

- double rescale_B：(1.5) の位置リスケーリングパラメータ（インフラトン質量を用いるのがデフォルト）
- double rescale_s：(1.5) の時間微分リスケーリングパラメータ（1 がデフォルト）
- double $\cdots$：インフラトンポテンシャルのパラメータ群

第 7 章

ファイル構造

本章では単一場インフレーションを Lattice シミュレーションする「SingleInflation」に含まれるファイルあるいはディレクトリの構造を説明する。構造のイメージを図 7.1 に示す。

単一場インフレーションの Lattice シミュレーションをするためのディレクトリ（フォルダ）「SingleInflation」は 2 つのディレクトリから構成されている。1 つは Lattice シミュレーションを実行するための C++ コードファイルと、シミュレーションによって出力される dat ファイルを含む「SFInflation」である。もう 1 つは、Lattice シミュレーションによって出力された dat ファイルから結果のグラフを描画するための python コードファイルを含む「visualize」である。

7.1 節では Lattice シミュレーションを実行するための「SFInflation」ディレクトリについて説明し、7.2 節では dat ファイルからグラフを描画するための「visualize」ディレクトリについて説明する。

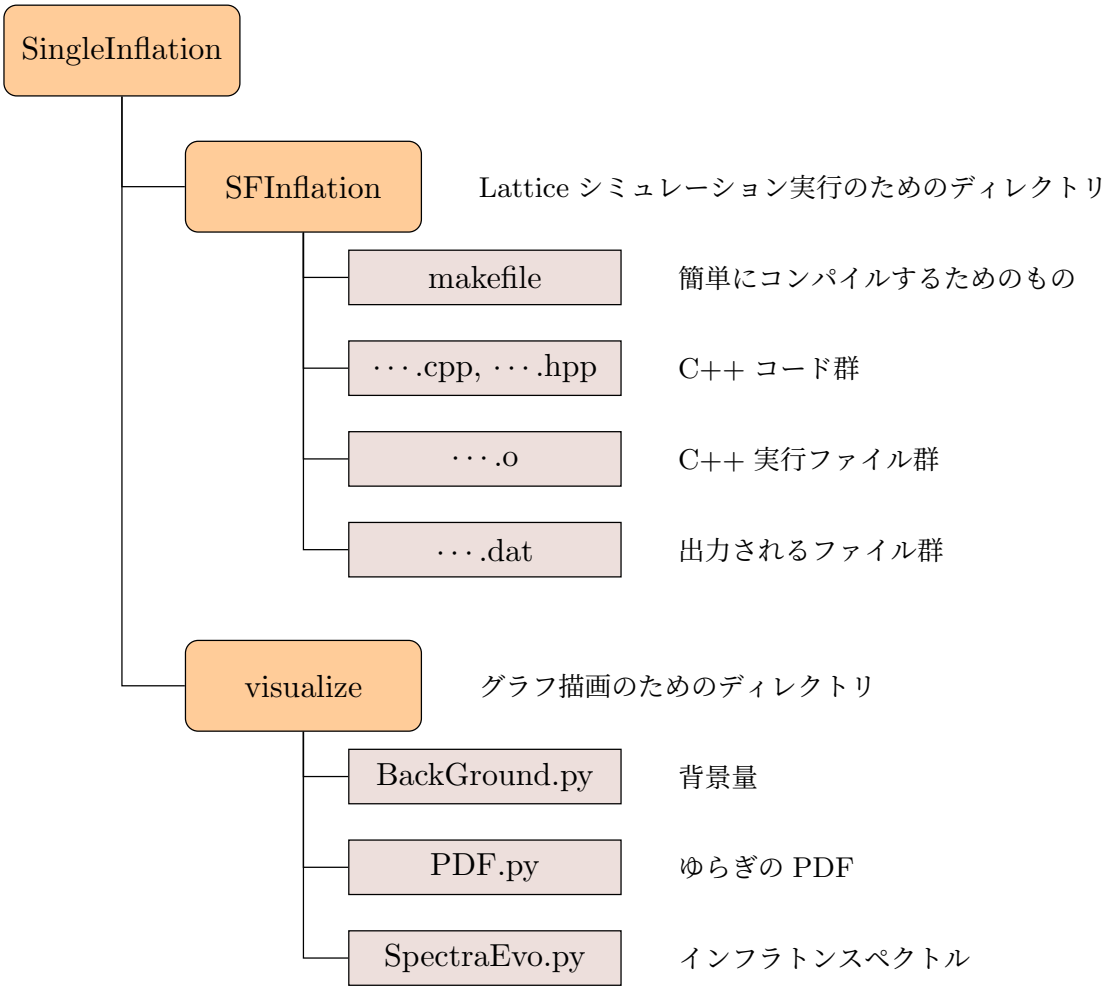

図 7.1　ファイル構造の概要図

7.1　SFInflation ディレクトリ

本節では Lattice シミュレーションを実行するための C++ コードファイル（cpp 拡張子のソースファイルと hpp 拡張子のヘッダファイル）について、各ファイルの役割等を説明する。C++ コードファイルの依存関係を図 7.2 に示す。

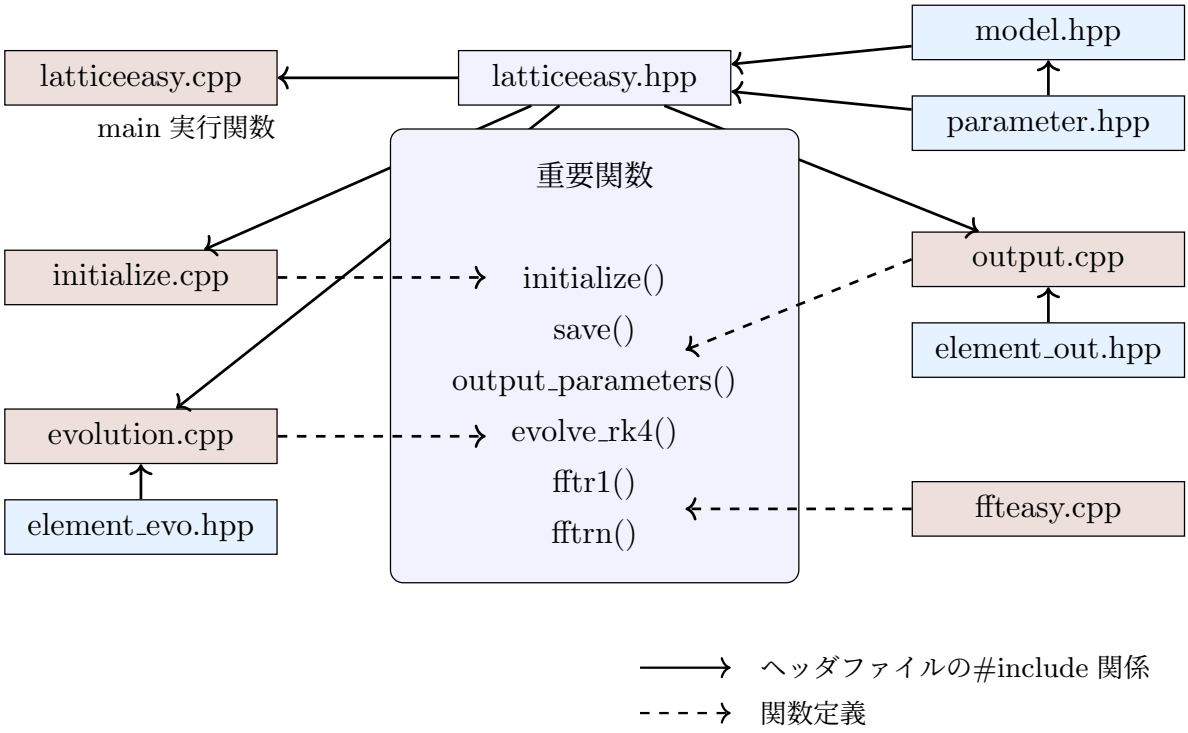

図 7.2　Lattice シミュレーションのための C++ コードファイルの依存関係。ヘッダファイル#include の矢印は IT 業界の付け方と逆かもしれないが、自分はこっちの方がわかりやすいのでこうしてます。関数定義の矢印は、latticeeasy.hpp ヘッダファイルで宣言し、各ソースファイルで実装している。

①まず見るべきは図 7.2 の上部にあるファイル群であり、「latticeeasy.cpp」が Lattice シミュレーションそのものを実行している（つまり main 実行関数を含む）ソースファイルである。またこのファイルで（インプットパラメータを除く）global 変数も宣言（定義）している。

②main 実行関数の内側で使用するいくつかの関数は「latticeeasy.hpp」ヘッダファイルで宣言していて、これらの関数の実装は他の各ソースファイルで行っている（図 7.2 の重要関数のこと）。また latticeeasy.hpp に include しているヘッダファイルが 2 つあり、

「parameter.hpp」と「model.hpp」である。parameter.hpp ではインプットパラメータに対応する global 変数を宣言していて、model.hpp ではインフラトンポテンシャルに関する関数を実装している。”この 2 つのヘッダファイルを include した latticeeasy.hpp ヘッダファイル”をソースファイルに#include することで、そのソースファイルでインプットパラメータやインフラトンポテンシャルの関数を使用できるようになる。

③main 実行関数で用いる、初期化のための関数 initialize() を実装しているのは「initialize.cpp」ソースファイルである。ここで背景量やゆらぎの初期化を実行している。

④main 実行関数で用いる、1 回の時間発展（4 次の Runge-Kutta 法）を行う関数 evolve_rk4() を実装しているのは「evolution.cpp」ソースファイルである。このファイルに#include している「element_evo.hpp」ヘッダファイルは evolve_rk4() でよく用いられる基本的な関数を実装したファイルである。

⑤main 実行関数で用いる、Lattice シミュレーションの結果を出力するための関数 save() と output_parameters() を実装しているのは「output.cpp」ソースファイルである。このファイルに#include している「element_out.hpp」ヘッダファイルは save() でよく用いられる基本的な関数を実装したファイルである。

⑥main 実行関数で用いる、FFT を行うための関数 fftr1() と fftrn() を実装しているのは「ffteasy.cpp」ソースファイルである。

Lattice シミュレーションのセットアップを変更したい場合は、インプットパラメータを含む「parameter.hpp」とインフラトンポテンシャルを含む「model.hpp」の中身を書き換えるだけで良い（これは LATTICEEASY[1] と同じ）。ただし、時間発展の手法などを変更したければその限りではない。

7.2　visualize ディレクトリ

本節ではグラフを描画するための visualize ディレクトリについて説明する。visualize ディレクトリに含まれる各 python ファイルと、SFInflation ディレクトリに含まれる Lattice シミュレーションで出力される dat ファイルの依存関係を図 7.3 に示す。

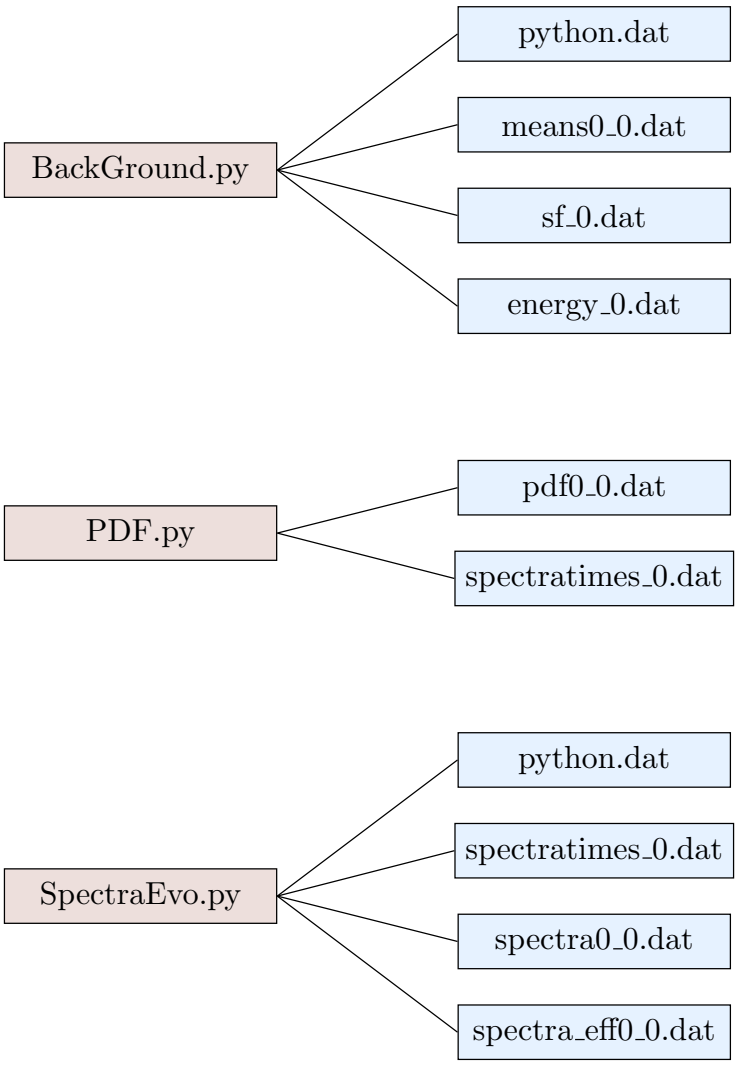

図 7.3　visualize ディレクトリに含まれるファイルの依存関係

「BackGround.py」は背景量のグラフを描画するためのファイルであり、「python.dat」「means0_0.dat」「sf_0.dat」「energy_0.dat」のデータ（値）を用いる。「python.dat」は Lattice シミュレーションのセットアップや初期条件のデータが入っている。「means0_0.dat」は場の値に関する平均量（場の平均値・キュムラントなど）のデータが入っている。「sf_0.dat」はスケール因子に関するデータが入っている。「energy_0.dat」はエネルギー密度や slow-roll パラメータに関するデータが入っている。

「PDF.py」はインフラトンゆらぎの PDF を描画するためのファイルであり、「pdf0_0.dat」「spectratimes_0.dat」のデータ（値）を用いる。「pdf0_0.dat」はインフラトンゆらぎの値を標準偏差で規格化した $\frac{\delta\phi}{\sigma}$ に対して、値 (bin の代表値) とその値を取る確率が入っている。「spectratimes_0.dat」はインフラトンスペクトル（とゆらぎの PDF）のデータを出力した時刻（e-folds）が入っている。

「SpectraEvo.py」はインフラトンゆらぎの無次元パワースペクトルの時間発展をグラフに描画するためのファイルであり、「python.py」「spectratimes_0.dat」「spectra0_0.dat」「spectra_eff_0.dat」のデータ（値）を用いる。「spectra0_0.dat」と「spectra_eff_0.dat」はインフラトンゆらぎのフーリエモードの 2 乗 $|\delta\phi(\boldsymbol{\kappa})|^2$ に関するデータが入っている。

第 8 章

コードレビュー

本章では「SFInflation」ディレクトリの各コード（但し.o 拡張子の実行ファイルはおそらく機械語なので除く）と「visualize」ディレクトリの各 python コードの説明を行う。8.1 節では g++ で簡単にコンパイルするための「makefile」を説明、8.2 節では Lattice シミュレーションを実行するための C++ コード群を説明、8.3 節では Lattice シミュレーションで出力される dat ファイルの内容を説明、8.4 節では dat ファイルからグラフを描画するための Python コードを説明する。

技術的な話ばかりで冗長になるので、スキップして第 III 部あるいは第 IV 部に飛ぶのを推奨する。自分でコードを書き換える際には本章を参考にされたし。プログラム上のコメントアウトは青色で表記する。やる気が出たら書きます。

8.1　makefile

Code 8.1　makefile

```
1 COMPILER = g++
2 FLAGS = -O3 -fopenmp
3 HEADA = latticeeasy.hpp model.hpp parameters.hpp
4 all: $(HEADA) element_out.hpp element_evo.hpp ffteasy.o latticeeasy.o
      evolution.o initialize.o output.o
5 $(COMPILER) $(FLAGS) ffteasy.o latticeeasy.o evolution.o initialize.o
      output.o -lm -o latticeeasy
6
7 clean:
8         rm -f *.o out output.txt latticeeasy *~ *.dat core*
9 cleaner:
10        rm -f *.o out output.txt latticeeasy *~ *.dat *.img
11
12 ffteasy.o: ffteasy.cpp
13        $(COMPILER) -c $(FLAGS) ffteasy.cpp
14 latticeeasy.o: latticeeasy.cpp $(HEADA)
15        $(COMPILER) -c $(FLAGS) latticeeasy.cpp
16 evolution.o: evolution.cpp latticeeasy.cpp $(HEADA) element_evo.hpp
17        $(COMPILER) -c $(FLAGS) evolution.cpp
18 initialize.o: initialize.cpp latticeeasy.cpp $(HEADA)
19        $(COMPILER) -c $(FLAGS) initialize.cpp
20 output.o: output.cpp latticeeasy.cpp $(HEADA) element_out.hpp
21        $(COMPILER) -c $(FLAGS) output.cpp
```

mekefile は簡単にコンパイルするためのファイルで、ソースファイル (cpp) とヘッダファイル (hpp) を複数用いる際のリンクを設定できる。

1 ～ 3 行目はこのファイル上の変数定義で、各々コンパイラ・コンパイルオプション・よく用いるヘッダファイルを表している。定義した変数を用いるには「$(変数名)$」と表記する。2 行目のコンパイルオプション変数において、「-03」は良く用いられる自動で効率化してくれる（？）オプション、「-fopenmp」は OpenMP を用いるオプションである。

4,5 行目はコンパイルの命令で、4 行目には全てのヘッダファイルと全ての実行ファイ

ル、5 行目にはコンパイラとコンパイルオプションと全ての実行ファイルを書く。「-lm -o latticeeasy」はコンパイルしたプログラムの名前を設定していて、Lattice シミュレーションを実行する時に「./latticeeasy」と命令すれば良くなる。

7 ～ 10 行目は、Lattice シミュレーションで作成された dat ファイルを削除するための命令「make clean」あるいは「make cleaner」を定義している。

12 ～ 21 行目は各々実行ファイルを設定している。各実行ファイルの設定は、1 行目に「実行ファイル名: 関係するファイル名全て」を書き、2 行目に「コンパイラ -c コンパイルオプション メインとなるソースファイル名」を書く。関係するファイルとは、メインとなるソースファイル・このソースファイルで実装した関数を利用しているファイル・インクルードされるヘッダファイルである。例えば initialize() は latticeeasy.cpp で用いるので、関係するファイルに latticeeasy.cpp を含める。

8.2　C++ コード

8.2.1　latticeeasy.cpp

本節では Lattice シミュレーションの main 実行関数を実装している「latticeeasy.cpp」を説明する。

Code 8.2　latticeeasy.cpp

```
1  /*
2  LATTICEEASY consists of the C++ files ‘‘latticeeasy.cpp,’’
3  ‘‘initialize.cpp,’’ ‘‘evolution.cpp,’’ ‘‘output.cpp,’’
4  ‘‘latticeeasy.h,’’ ‘‘parameters.h,’’. (The distribution also
       includes
5  the file ffteasy.cpp but this file is distributed separately and
6  therefore not considered part of the LATTICEEASY distribution in
       what
7  follows.) LATTICEEASY is free. We are not in any way, shape, or
       form
8  expecting to make money off of these routines. We wrote them for
       the
9  sake of doing good science and we’re putting them out on the
       Internet
10 in case other people might find them useful. Feel free to
       download
11 them, incorporate them into your code, modify them, translate the
12 comment lines into Swahili, or whatever else you want. What we do
        want
13 is the following:
14 1) Leave this notice (i.e. this entire paragraph beginning with
15 ‘‘LATTICEEASY consists of...’’ and ending with our email
       addresses) in
16 with the code wherever you put it. Even if you’re just using it
17 in-house in your department, business, or wherever else we would
       like
18 these credits to remain with it. This is partly so that people
       can...
19 2) Give us feedback. Did LATTICEEASY work great for you and help
```

```
20 your work? Did you hate it? Did you find a way to improve it, or
21 translate it into another programming language? Whatever the case
22 might be, we would love to hear about it. Please let us know at
       the
23 email address below.
24 3) Finally, insofar as we have the legal right to do so we forbid
25 you to make money off of this code without our consent. In other
       words
26 if you want to publish these functions in a book or bundle them
       into
27 commercial software or anything like that contact us about it
28 first. We'll probably say yes, but we would like to reserve that
29 right.
30
31 For any comments or questions you can reach us at
32 gfelder@email.smith.edu
33 Igor.Tkachev@cern.ch
34 */
35
36 #include "latticeeasy.hpp"
37
38
39 double f[nflds][N][N][N],fd[nflds][N][N][N];
40
41 double t,t0;
42
43 double a=1.,ad=0.;
44 double asub1=1.,adsub1=0.;
45 double asub2=1.,adsub2=0.;
46 double ka=0.,kad=0.;
47
48 double hubble_init=0.;
49 int run_number;
50 char mode_[10]="w";
51 double rescaling=1.;
52 char ext_[500]="_0.dat";
53 // false->no exit true->exit
54 bool exitpara = false;
```

```
55
56
57 int main()
58 {
59   int numsteps=0,output_interval;
60   int update_time;
61
62   /* DEBUG: seed */
63   if(seed<1)
64     printf("ERROR: The parameter seed has been set to %d. For correct
         output set seed to a positive integer.",seed);
65
66   /* Initialization */
67   initialize();
68   t=t0;
69
70   output_interval = (int)((tf-t0)/dt)/noutput_times + 1;
71   update_time=time(NULL)+print_interval;
72
73   double (*fsub1)[N][N][N] = (double(*)[N][N][N])malloc(sizeof(double
         )*nflds*N*N*N);
74   double (*fdsub1)[N][N][N] = (double(*)[N][N][N])malloc(sizeof(
         double)*nflds*N*N*N);
75   double (*fsub2)[N][N][N] = (double(*)[N][N][N])malloc(sizeof(double
         )*nflds*N*N*N);
76   double (*fdsub2)[N][N][N] = (double(*)[N][N][N])malloc(sizeof(
         double)*nflds*N*N*N);
77   double (*kf)[N][N][N] = (double(*)[N][N][N])malloc(sizeof(double)*
         nflds*N*N*N);
78   double (*kfd)[N][N][N] = (double(*)[N][N][N])malloc(sizeof(double)*
         nflds*N*N*N);
79
80   /* Evolution */
81   while(t<=tf)
82   {
83     evolve_rk4(dt,SUB,kf,kfd);
84
85     /* Save values */
```

```
86      numsteps++;
87      if(numsteps%output_interval == 0 && t<tf)
88        save(0);
89
90      /* Command line output */
91      if(time(NULL) >= update_time)
92      {
93        if(screen_updates)
94          printf("%f\n",t);
95        update_time += print_interval;
96      }
97
98      /* Free memory with error */
99      if(exitpara)
100     {
101       FREE;
102       output_parameters();
103       printf("ERROR: LATTICEEASY program unfinished\n");
104       exit(1);
105     }
106   }
107
108   /* Post Processing */
109   printf("OK: Saving final data\n");
110   save(1);
111   output_parameters();
112   FREE;
113   printf("OK: LATTICEEASY program finished\n");
114
115   return(0);
116 }
```

1 ～ 34 行目は LATTICEEASY で指定されている使用権限に関するコメントで 12.1 節で説明する。36 行目は「latticeeasy.hpp」ヘッダファイルのインクルード記述である。ヘッダファイルをインクルードすることでその部分にヘッダファイルの記述が差し込まれる。

39 ～ 54 行目は global 変数（インプットパラメータ以外）の宣言・定義である。基本

的にいじらない方が良い。f と fd は場の値と場の微分の値を格納するリストで、[場の種類（今はインフラトンの 1 種）][x 座標][y 座標][z 座標] を表している。t はプログラム上の時間 $\tau_{\rm pr}$ を表し、t0 はプログラム上の初期時刻 $\tau_{\rm pr,ini}$ を表す。a,ad はスケール因子とそのプログラム時間微分 a, a' であり、asub1,adsub1,asub2,adsub2,ka,kad は 4.4 節で説明したように 4 次の Runge-Kutta 法で必要になる 3 つのダミー変数である。hubble_init は Hubble パラメータの初期値、run_number は何回目の Lattice シミュレーションか、rescaling はプログラムでリスケーリングする因子 (LATTICEEASY の名残で使わない)、exp_は作成する dat ファイルの拡張子を表す。exitpara はプログラムを中断する (true) か否 (false) かを表す変数で、プログラム上で勝手に変更される。

57 ∼ 116 行目は Lattice シミュレーションの main 実行関数である。62 ∼ 64 行目は一様乱数 X, Y を生成するためのシード値が正であることを確かめるためのデバッグ処理である。66 ∼ 68 行目は初期化処理を行っている。70 行目で output_interval にデータを出力する時間間隔の値を代入している。71 行目で update_time に CLI 上で経過時間を表示する次の時刻を代入している。73 ∼ 78 行目で場の値を格納するための配列を動的に確保していて、fsub1 と fdsub1 は 4.4 節における Dummy List1 に対応し、fsub2 と fdsub2 は Dummy List2 に対応し、kf と kfd は Dummy List3 に対応する。80 ∼ 106 行目で時間発展のループを実行していて、時間発展・値の出力・CLI への経過時間の出力・エラー時のメモリ解放の順で記述している。108 ∼ 113 行目は時間発展が終わった後の値の出力とメモリ解放を行なっている。115 行目の return(0) は main 実行関数に書かなきゃいけないやつ、正常終了を表す。

8.2.2　latticeeasy.hpp

8.2.3　parameters.hpp

本節ではインプットパラメータの値を操作するための「parameters.hpp」を説明する。

Code 8.3　parameters.hpp

```
1  #ifndef _PARAMETERS_
2  #define _PARAMETERS_
3
4
5  #define MODEL 0 // Inflaton potential choice (0->chaotic 1->step)
6  const double m =0.51e-5; // Inflaton mass[M_pl]
7
8  // Adjustable run parameters
9  #define NDIMS 3
10 const int N = 128;
11 const int nflds = 1;
12 #if MODEL==0
13 const double L = 1.4;
14 const double initfield[]={14.5};
15 const double initderivs[]={-0.8152128};
16 #elif MODEL==1
17 const double L = 0.6;
18 const double initfield[]={14.5};
19 const double initderivs[]={-0.8233865};
20 #endif
21 const double dt = .0001;
22 const double tf=1.2;
23 const int seed=2;
24
25 const bool hankel=true; // Inflaton fluctuation initial setup
26 const double kcutoff=0; // Momentum for initial lowpass filter.
       Set to 0 to not filter
27
28 // If and how to continue previous runs.
29 // If no grid image is available in the run directory then a new
```

```
      run will be started irrespective of continue_run.
30 // 0=Start a new run at t=0. 1=Continue old run, appending new
      data to old data files. 2=Continue old run, creating new data
       files for all output. (Old ones will not in general be
      overwritten.)
31 const int continue_run=0;
32
33 // Variables controlling output
34 const char alt_extension[]="";
35 const int noutput_times=1000;
36 const int print_interval=1;
37 const int screen_updates=1;
38 const double checkpoint_interval=0.1;
39 const double store_lattice_times[]={0.}; // An optional list of
      times at which to close the grid image file and open a new
      one.
40 // The variables s<name> control what will be saved (1=save, 0=
      don't save)
41 const int smeansvars=1;
42 const int sexpansion=1;
43 // The following calculations are performed at intervals given by
       checkpoint_interval
44 const double t_start_output=0.;
45 const int senergy=1;
46   const double tenergy=t_start_output;
47 const int sspectra=1;
48   const double tspectra=t_start_output;
49   const int spdf=1;
50   const int pdfrange=5;
51
52 /*
53 const int scheckpoint=0;
54   const double tcheckpoint=t_start_output;
55 */
56
57 // The number of threads in openMP
58 const int num_thread=8;
59
```

```
60 #endif
```

1,2,60 行目は 2 重インクルードを防ぐためのマクロであり、ヘッダファイル（hpp 拡張子）に書いておくべきもの。5 行目はインフラトンポテンシャルを決定するマクロであり、6.5.1 節の Chaotic ポテンシャルにしたければ 0、6.5.2 節の Step ポテンシャルにしたければ 1 にすれば良い。6 行目は換算プランク質量単位でのインフラトン質量の値である。9 行目は空間次元を決定するマクロである 3 にしておくべき。10 行目は 1 辺の格子点数であり、FFT の実行のためには 2 の累乗でなければならない（128 だと 5 分くらいでシミュレーションが終わると思います)。11 行目は場の数を指定していて、単一場インフレーションでは 1 にしておくべき。12 ∼ 20 行目はインフラトンポテンシャルで場合分けした $L_{\rm pr}$ と場の初期値と場の微分の初期値を決定している。このパラメータ値は [2] で実行されていたシミュレーションの値である。21 行目はプログラム上の時間発展の間隔を指定している。22 行目は Lattice シミュレーションを終了するプログラム上の時刻を指定している。23 行目は一様乱数 X, Y を生成する時のシード値を指定していて、自然数を入れるべきである。25 行目はインフラトンゆらぎの初期条件に Hankel 関数と指数関数のどちらを用いるかを決定する変数（定数）を指定している。26 行目は LATTICEEASY の名残であり操作すべきでない。

28 ∼ 31 行目は何回目の Lattice シミュレーションかを指定する数であり、基本的に 0 にしておくべき。

8.2.4　model.hpp

8.2.5　initialize.cpp

8.2.6　evolution.cpp, element_evo.hpp

8.2.7　output.cpp, element_out.hpp

8.2.8　ffteasy.cpp

本節では FFT と逆 FFT を実行するための関数を実装している「ffteasy.cpp」を説明する。これは浮動小数点数の型を float 型から倍の精度を持つ double 型に変更した以外は、LATTICEEASY のコードをそのまま用いている。アルゴリズムについてはあまり理解していないのでおそらくほとんど説明しない。また、可読性のためにコメントアウトをいくつか省略してある。

Code 8.4　ffteasy.cpp

```
1  /*
2  FFTEASY consists of the four C functions fftc1, fftcn, fftr1, and
3  fftrn. FFTEASY is free. I am not in any way, shape, or form
       expecting
4  to make money off of these routines. I wrote them because I
       needed
5  them for some work I was doing and I'm putting them out on the
6  Internet in case other people might find them useful. Feel free
       to
7  download them, incorporate them into your code, modify them,
       translate
8  the comment lines into Swahili, or whatever else you want. What I
        do
9  want is the following:
10 1) Leave this notice (i.e. this entire paragraph beginning with
11 ``FFTEASY consists of...'' and ending with my email address) in
       with
12 the code wherever you put it. Even if you're just using it in-
       house in
13 your department, business, or wherever else I would like these
       credits
14 to remain with it. This is partly so that people can...
15 2) Give me feedback. Did FFTEASY work great for you and help your
16 work? Did you hate it? Did you find a way to improve it, or
       translate
17 it into another programming language? Whatever the case might be,
        I
```

```
18 would love to hear about it. Please let me know at the email
       address
19 below.
20 3) Finally, insofar as I have the legal right to do so I forbid
       you
21 to make money off of this code without my consent. In other words
        if
22 you want to publish these functions in a book or bundle them into
23 commercial software or anything like that contact me about it
24 first. I'll probably say yes, but I would like to reserve that
       right.
25
26 For any comments or questions you can reach me at
27 gfelder@email.smith.edu.
28 */
29
30 /* These declarations are put here so you can easily cut and
       paste them into your program. */
31 void fftc1(double f_arg[], int N_arg, int skip, int forward);
32 void fftcn(double f_arg[], int ndims, int size[], int forward);
33 void fftr1(double f_arg[], int N_arg, int forward);
34 void fftrn(double f_arg[], double fnyquist[], int ndims, int size[],
       int forward);
35
36 #include <stdlib.h>
37 #include <stdio.h>
38 #include <math.h>
39
40 struct complex
41 {
42   double real;
43   double imag;
44 };
45
46 /*
47 Do a Fourier transform of an array of N complex numbers separated
        by steps of (complex) size skip.
48 The array f should be of length 2N*skip and N must be a power of
```

```
      2.
49 Forward determines whether to do a forward transform (1) or an
      inverse one(-1)
50 */
51 void fftc1(double f_arg[], int N_arg, int skip, int forward)
52 {
53   int b,index1,index2,trans_size,trans;
54   double pi2 = 4.*asin(1.);
55   double pi2n,cospi2n,sinpi2n;
56   struct complex wb;
57   struct complex temp1,temp2;
58   struct complex *c = (struct complex *)f_arg;
59
60   /* Place the elements of the array c in bit-reversed order */
61   for(index1=1,index2=0;index1<N_arg;index1++)
62   {
63     for(b=N_arg/2;index2>=b;b/=2)
64       index2-=b;
65     index2+=b;
66     if(index2>index1)
67     {
68       temp1 = c[index2*skip];
69       c[index2*skip] = c[index1*skip];
70       c[index1*skip] = temp1;
71     }
72   }
73
74   /* Next perform successive transforms of length 2,4,...,N using
        the Danielson-Lanczos formula */
75   for(trans_size=2;trans_size<=N_arg;trans_size*=2)
76   {
77     pi2n = forward*pi2/(double)trans_size;
78     cospi2n = cos(pi2n);
79     sinpi2n = sin(pi2n);
80     wb.real = 1.;
81     wb.imag = 0.;
82     for(b=0;b<trans_size/2;b++)
83     {
```

```
84        for(trans=0;trans<N_arg/trans_size;trans++)
85        {
86          index1 = (trans*trans_size+b)*skip;
87          index2 = index1 + trans_size/2*skip;
88          temp1 = c[index1];
89          temp2 = c[index2];
90          c[index1].real = temp1.real + wb.real*temp2.real - wb.imag*
                temp2.imag;
91          c[index1].imag = temp1.imag + wb.real*temp2.imag + wb.imag*
                temp2.real;
92          c[index2].real = temp1.real - wb.real*temp2.real + wb.imag*
                temp2.imag;
93          c[index2].imag = temp1.imag - wb.real*temp2.imag - wb.imag*
                temp2.real;
94        }
95        temp1 = wb;
96        wb.real = cospi2n*temp1.real - sinpi2n*temp1.imag;
97        wb.imag = cospi2n*temp1.imag + sinpi2n*temp1.real;
98      }
99    }
100
101   /* For an inverse transform divide by the number of grid points
          */
102   if(forward<0.)
103   {
104     for(index1=0;index1<skip*N_arg;index1+=skip)
105     {
106       c[index1].real /= N_arg;
107       c[index1].imag /= N_arg;
108     }
109   }
110 }
111
112 /*
113 Do a Fourier transform of an ndims dimensional array of complex
        numbers
114 Array dimensions are given by size[0],...,size[ndims-1]. Note
        that these are sizes of complex arrays.
```

```
115 The array f should be of length 2*size[0]*...*size[ndims-1] and
        all sizes must be powers of 2.
116 Forward determines whether to do a forward transform (1) or an
        inverse one(-1)
117 */
118 void fftcn(double f_arg[], int ndims, int size[], int forward)
119 {
120   int i,j,dim;
121   int planesize=1,skip=1;
122   int totalsize=1;
123
124   for(dim=0;dim<ndims;dim++)
125     totalsize *= size[dim];
126
127   for(dim=ndims-1;dim>=0;dim--)
128   {
129     planesize *= size[dim];
130     for(i=0;i<totalsize;i+=planesize)
131       for(j=0;j<skip;j++)
132         fftc1(f_arg+2*(i+j),size[dim],skip,forward);
133     skip *= size[dim];
134   }
135 }
136
137 /*
138 Do a Fourier transform of an array of N real numbers
139 N must be a power of 2
140 Forward determines whether to do a forward transform (>=0) or an
        inverse one(<0)
141 */
142 void fftr1(double f_arg[], int N_arg, int forward)
143 {
144   int b;
145   double pi2n = 4.*asin(1.)/N_arg,cospi2n=cos(pi2n),sinpi2n=sin(pi2n);
146   struct complex wb;
147   struct complex temp1,temp2;
148   struct complex *c = (struct complex *)f_arg;
149
```

```
150   if(forward==1)
151     fftc1(f_arg,N_arg/2,1,1);
152
153   wb.real = 1.;
154   wb.imag = 0.;
155   for(b=1;b<N_arg/4;b++)
156   {
157     temp1 = wb;
158     wb.real = cospi2n*temp1.real - sinpi2n*temp1.imag;
159     wb.imag = cospi2n*temp1.imag + sinpi2n*temp1.real;
160     temp1 = c[b];
161     temp2 = c[N_arg/2-b];
162     c[b].real = .5*(temp1.real+temp2.real + forward*wb.real*(temp1.
          imag+temp2.imag) + wb.imag*(temp1.real-temp2.real));
163     c[b].imag = .5*(temp1.imag-temp2.imag - forward*wb.real*(temp1.
          real-temp2.real) + wb.imag*(temp1.imag+temp2.imag));
164     c[N_arg/2-b].real = .5*(temp1.real+temp2.real - forward*wb.real*(
          temp1.imag+temp2.imag) - wb.imag*(temp1.real-temp2.real));
165     c[N_arg/2-b].imag = .5*(-temp1.imag+temp2.imag - forward*wb.real
          *(temp1.real-temp2.real) + wb.imag*(temp1.imag+temp2.imag));
166   }
167   temp1 = c[0];
168   c[0].real = temp1.real+temp1.imag;
169   c[0].imag = temp1.real-temp1.imag;
170
171   if(forward==-1)
172   {
173     c[0].real *= .5;
174     c[0].imag *= .5;
175     fftc1(f_arg,N_arg/2,1,-1);
176   }
177 }
178
179 /*
180 Do a Fourier transform of an ndims dimensional array of real
      numbers
181 Array dimensions are given by size[0],...,size[ndims-1]. All
      sizes must be powers of 2.
```

```
182 The (complex) nyquist frequency components are stored in fnyquist
        [size[0]][size[1]]...[2*size[ndims-2]]
183 Forward determines whether to do a forward transform (1) or an
        inverse one(-1)
184 */
185 void fftrn(double f_arg[], double fnyquist[], int ndims, int size[],
        int forward)
186 {
187   int i,j,b;
188   int index,indexneg=0;
189   int stepsize;
190   int N_arg=size[ndims-1];
191   double pi2n = 4.*asin(1.)/N_arg,cospi2n=cos(pi2n),sinpi2n=sin(pi2n);
192   struct complex wb;
193   struct complex temp1,temp2;
194   struct complex *c = (struct complex *)f_arg, *cnyquist = (struct
          complex *)fnyquist;
195   int totalsize=1;
196   int *indices= (int *) malloc(ndims*sizeof(int));
197   if(!indices)
198   {
199     printf("Error allocating memory in fftrn routine. Exiting.\n");
200     exit(1);
201   }
202
203   size[ndims-1] /= 2;
204   for(i=0;i<ndims;i++)
205   {
206     totalsize *= size[i];
207     indices[i] = 0;
208   }
209
210   if(forward==1)
211   {
212     fftcn(f_arg,ndims,size,1);
213     for(i=0;i<totalsize/size[ndims-1];i++)
214       cnyquist[i] = c[i*size[ndims-1]];
215   }
```

```
216
217   for(index=0;index<totalsize;index+=size[ndims-1])
218   {
219     wb.real = 1.;
220     wb.imag = 0.;
221     for(b=1;b<N_arg/4;b++)
222     {
223       temp1 = wb;
224       wb.real = cospi2n*temp1.real - sinpi2n*temp1.imag;
225       wb.imag = cospi2n*temp1.imag + sinpi2n*temp1.real;
226       temp1 = c[index+b];
227       temp2 = c[indexneg+N_arg/2-b];
228       c[index+b].real = .5*(temp1.real+temp2.real + forward*wb.real*(
              temp1.imag+temp2.imag) + wb.imag*(temp1.real-temp2.real));
229       c[index+b].imag = .5*(temp1.imag-temp2.imag - forward*wb.real*(
              temp1.real-temp2.real) + wb.imag*(temp1.imag+temp2.imag));
230       c[indexneg+N_arg/2-b].real = .5*(temp1.real+temp2.real - forward
              *wb.real*(temp1.imag+temp2.imag) - wb.imag*(temp1.real-temp2
              .real));
231       c[indexneg+N_arg/2-b].imag = .5*(-temp1.imag+temp2.imag -
              forward*wb.real*(temp1.real-temp2.real) + wb.imag*(temp1.
              imag+temp2.imag));
232     }
233     temp1 = c[index];
234     temp2 = cnyquist[indexneg/size[ndims-1]];
235     c[index].real = .5*(temp1.real+temp2.real + forward*(temp1.imag+
            temp2.imag));
236     c[index].imag = .5*(temp1.imag-temp2.imag - forward*(temp1.real-
            temp2.real));
237     cnyquist[indexneg/size[ndims-1]].real = .5*(temp1.real+temp2.real
            - forward*(temp1.imag+temp2.imag));
238     cnyquist[indexneg/size[ndims-1]].imag = .5*(-temp1.imag+temp2.imag
             - forward*(temp1.real-temp2.real));
239
240     /* Find indices for positive and single index for negative
            frequency. In each dimension indexneg[j]=0 if index[j]=0,
             indexneg[j]=size[j]-index[j] otherwise. */
241     stepsize=size[ndims-1];
```

```
242     for(j=ndims-2;indices[j]==size[j]-1 && j>=0;j--)
243     {
244       indices[j]=0;
245       indexneg -= stepsize;
246       stepsize *= size[j];
247     }
248     if(indices[j]==0)
249       indexneg += stepsize*(size[j]-1);
250     else
251       indexneg -= stepsize;
252     if(j>=0)
253       indices[j]++;
254   }
255
256   if(forward==-1)
257     fftcn(f_arg,ndims,size,-1);
258
259   size[ndims-1] *= 2;
260 }
```

1 ～ 28 行目は LATTICEEASY で指定されている使用権限に関するコメントで 12.1 節で説明する。30 ～ 34 行目はこのソースファイルで実装する関数の宣言（関数プロトタイプ）である。36 ～ 38 行目は C++ のライブラリをインクルードしている。40 ～ 44 行目は複素数を表現する構造体の定義をしている。

46 ～ 110 行目は 1 次元複素数のフーリエ変換を行う関数「fftc1()」を実装している。112 ～ 135 行目は n 次元複素数のフーリエ変換を行う関数「fftcn()」を実装している。137 ～ 177 行目は 1 次元実数のフーリエ変換を行う関数「fftr1()」を実装している。179 ～ 260 行目は n 次元実数のフーリエ変換を行う関数「fftrn()」を実装している。これらの関数には引数「int forward」が必要であり、1 を与えればフーリエ変換、-1 を与えれば逆フーリエ変換を実行する。

このコードで実行されるフーリエ変換の定義は (2.2)(2.3) であり、それを 1 次元かつ $N = 4$ の簡単な場合で確認する。実行した結果は以下のようになった。

Code 8.5　fftr1() の挙動の確認

```
1 int main() {
```

```
2   double list[4]={1.,10.,100.,1000.};
3   fftr1(list,4,1);
4   std::cout << list[0] << " "
5             << list[1] << " "
6             << list[2] << " "
7             << list[3] << std::endl;
8   fftr1(list,4,-1);
9   std::cout << list[0] << " "
10            << list[1] << " "
11            << list[2] << " "
12            << list[3] << std::endl;
13 }
14 ------------------------------------------------
15 1111 -909 -99 -990
16 1 10 100 1000
```

つまりフーリエ変換で実装されている処理は

$$\{f_0, f_1, f_2, f_3\} \ \rightarrow \ \{f_0 + f_1 + f_2 + f_3,\ f_0 - f_1 + f_2 - f_3,\ f_0 - f_2,\ f_1 - f_3\}$$

である。f のフーリエ変換を F と表すとすると、解析的には

$$\begin{aligned}
F_0 &= \sum_{i=0\sim3} f_i e^{\frac{2\pi i}{4} 0\times i} = f_0 + f_1 + f_2 + f_3 \\
F_1 &= \sum_{i=0\sim3} f_i e^{\frac{2\pi i}{4} 1\times i} = f_0 + if_1 - f_2 - if_3 \\
F_2 &= \sum_{i=0\sim3} f_i e^{\frac{2\pi i}{4} 2\times i} = f_0 - f_1 + f_2 - f_3 \\
F_3 &= \sum_{i=0\sim3} f_i e^{\frac{3\pi i}{4} 0\times i} = f_0 - if_1 - f_2 + if_3
\end{aligned}$$

であり、実装されている処理は

$$\begin{aligned}
\{f_0, f_1, f_2, f_3\} \ \rightarrow \ & \left\{F_0,\ F_2,\ \frac{F_1 + F_3}{2},\ -i\frac{F_1 - F_3}{2}\right\} \\
= \ & \{F_0,\ F_2,\ \mathrm{Re}[F_1],\ \mathrm{Im}[F_1]\}
\end{aligned}$$

である。定義 (2.2)(2.3) の位相を逆に考えてもいいのだが、その場合実装される処理は $\mathrm{Im}[F_1]$ ではなく $-\mathrm{Im}[F_1] = \mathrm{Im}[F_3]$ となり、不自然なのでここでは (2.2)(2.3) だと思うこととする。

このように ffteasy.cpp で実装される離散フーリエ変換において、f_k は実部と虚部で表される。つまり f_k と f_{-k} の自由度 2 つを $\mathrm{Re}[f_k]$ と $\mathrm{Im}[f_k]$ として表している。この構造は、初期条件でインフラトンゆらぎのフーリエモードを作成する時に理解しておく必要がある（3 次元での構造は面倒なので解説しない）。

8.3　dat ファイル

8.3.1　info.dat

8.4　Python コード

8.4.1　BackGround.py

8.4.2　PDF.py

8.4.3　SpectraEvo.py

第 III 部

Axion-U(1) インフレーション

第 9 章

セットアップと実装表式

本章では、axion-U(1) インフレーションの Lattice シミュレーションで用いた詳細なセットアップと、実際に用いた実装表式を説明する。第 I 部で既に説明した内容については触れないが、第 II 部は読まなくとも良いように第 II 部と被っている内容をもう一度同じく説明する。9.1 節で用いたセットアップのまとめ、9.2 節で初期条件、9.3 節で実装する発展方程式、9.4 節で出力する量、9.5 節でインフラトンポテンシャル、9.6 節でインプットするべきパラメータ、9.7 節で運動量カットオフについて説明をする。

ここでは概念的なセットアップと理論的な計算を記すため、具体的（あるいは技術的）なコードの解説は 10 章と 11 章を参照すべき。

9.1 セットアップのまとめ

ここで扱うのは axion-U(1) インフレーションでラグランジアンは

$$\mathcal{L} = \sqrt{-g}\left[-\frac{1}{2}g^{\mu\nu}(\partial_\mu\phi)(\partial_\nu\phi) - V(\phi) - \frac{1}{4}F^{\mu\nu}F_{\mu\nu} - \frac{\alpha}{4f}\tilde{F}^{\mu\nu}F_{\mu\nu}\right] \tag{9.1}$$

である。α は無次元パラメータ、f は axion 崩壊定数、$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$ はゲージ場 A_μ の強さテンソル、$\tilde{F}^{\mu\nu}$ は双対テンソル

$$\tilde{F}^{\mu\nu} = \frac{\tilde{\epsilon}^{\mu\nu\alpha\beta}}{2\sqrt{-g}}F_{\mu\nu} \tag{9.2}$$

であり $\tilde{\epsilon}$ は $\tilde{\epsilon}^{0123} = 1$ の完全反対称な記号である。インフラトン場（axion 場）ϕ は実スカラー場の場合を考えているので、プログラム上で扱う変数はインフラトン場 ϕ・その時間微分 ϕ'・ゲージ場 A_μ・その時間微分 A'_μ・スケール因子 a・その時間微分 a' である。

全格子点数が N^3 個であることを考慮すれば、変数の数は $10N^3+2$ 個（ゲージ固定条件を課してゲージ場の成分 1 つを落とせば $8N^3+2$ 個）である。

場のゆらぎは場の値 $\phi, \phi', A_\mu, A'_\mu$ に含まれているが、計量ゆらぎについては小さいと考えて無視し、平坦な FLRW 計量を考える。

$$ds^2 = a^2[-d\tau^2 + \delta_{ij}dx^i dx^j] \tag{9.3}$$

これは線形摂動論において計量ゆらぎは slow-roll パラメータに比例するので、インフラトンが十分 slow-roll していれば計量ゆらぎは無視できるからである [2]。今は計量ゆらぎを無視していて、この近似のためにシミュレーション中は十分 slow-roll している必要がある。

ゲージ固定条件は Lorentz ゲージと Coulomb ゲージの 2 つが主な選択肢であり、（今のところ）Lorentz ゲージ条件を課す。（だけれども Coulomb ゲージの方がメモリ的にも処理速度的にも経済的な気がする）また、インフラトン速度は正（$\partial_t\bar{\phi} > 0$）になるようにする。

空間の有限化に伴う場の境界条件は周期的境界条件 (1.2) を課し、微分演算子の離散化と対応する有効運動量ベクトルの定義は (2.6)(2.8)(2.11)

$$\partial_i f(\boldsymbol{n}) = \frac{f(\boldsymbol{n}+\boldsymbol{e}_i) - f(\boldsymbol{n}-\boldsymbol{e}_i)}{2dx} \tag{9.4}$$

$$\nabla^2 f(\boldsymbol{n}) = \frac{1}{(2dx)^2} \sum_{i=x,y,z} \sum_{c=\pm 2} [f(\boldsymbol{n}+c\boldsymbol{e}_i) - f(\boldsymbol{n})] \tag{9.5}$$

$$[k_{\mathrm{eff}}(\boldsymbol{m})]_i = \frac{1}{dx} \sin\left(\frac{2\pi m_i}{N}\right) \tag{9.6}$$

を用いる。ここで解像度の低いラプラシアンの定義を用いたのは、ゲージ場の増加を Lattice シミュレーションで正しく見るためであり、そのためには 1 階微分とラプラシアンの有効運動量ベクトルを一致させる必要がある [2]。その思惑は、フーリエモードの運動方程式を連続空間と格子空間で一致させるためで、ゲージモードの運動方程式に 1 階微分とラプラシアンが現れることが原因だそうだ。解像度の高いラプラシアン (2.9) を用いた場合、有効運動量ベクトルを一致させるためには 1 階微分を $\boldsymbol{n} \pm \frac{1}{2}\boldsymbol{e}_i$ での差分で定義する必要があり、これらの格子点は Lattice 上で定義されないので解像度の高いラプラシアンは用いることができない。有効運動量ベクトルにはカットオフを設ける。これについては [2] と 9.7 節を参照すべし。

時間発展の方法は 4 次の Runge-Kutta 法（4.4 節）を用いる。

9.2　初期条件

9.2.1　背景量

背景量の初期値は 3.1 節で説明した通りである。追加で考えるべきゲージ場については背景量を 0 とし、ゆらぎのみ考える。また、Hubble パラメータの初期値に用いるエネルギー密度の表式は明らかにしていなかったのでここで見る。前述したように、エネルギー密度に対するゆらぎの寄与は無視するので、運動エネルギーとポテンシャルエネルギー（つまりインフラトン場に関する背景量 $\overline{\phi}_{\rm ini}, \overline{\phi'}_{\rm ini}$ のみ）から決まり

$$\rho_{\rm pr,ini} = \frac{1}{2}\overline{\phi'}^2_{\rm ini} + V_{\rm pr}(\overline{\phi}_{\rm ini}) \tag{9.7}$$

である。ここでスケール因子の初期値が 1 であることを用いた。ゆらぎの寄与を無視しているので、インプットパラメータである $\overline{\phi}_{\rm ini}, \overline{\phi'}_{\rm ini}$ から簡単に計算できる。よってハッブルパラメータの初期値は

$$H_{\rm ini} = m\sqrt{\frac{\overline{\phi'}^2_{\rm ini} + 2V_{\rm pr}(\overline{\phi}_{\rm ini})}{6}} \tag{9.8}$$

9.2.2　インフラトンゆらぎ

3.3 節で量子ゆらぎを再現するインフラトンゆらぎの実装方法を見た。私は単一場インフレーションのシミュレーションと同じくモード関数として Hankel 関数を用いる。これは 3.3 節で用いた sub-horizon 極限のモード関数 (3.8) の一般化であり、super-horizon のゆらぎも考慮できるモード関数である。ただし Hankel 関数を用いた場合でも、slow-roll 近似を課しているため Lattice シミュレーション開始時に十分 slow-roll かつ線形摂動論が有効でなければならないことは変わらない。

Hankel 関数を用いた場合の解析的なインフラトンパワースペクトルは

$$\mathcal{P}_{\delta\phi}(k) = \frac{-\pi\tau}{4a^2}|H^{(1)}_\nu(-k\tau)|^2 \tag{9.9}$$

であり、インフラトンゆらぎのモード関数は

$$\delta\phi(k) = \frac{\sqrt{-\pi\tau}}{2a}H^{(1)}_\nu(-k\tau) \tag{9.10}$$

である。ここで ν は slow-roll の 0 次近似で評価して

$$\nu = \sqrt{\frac{9}{4} - \frac{m_{\rm eff}^2}{H^2}} = \sqrt{\frac{9}{4} - \frac{m^2}{H_{\rm ini}^2}\frac{m_{\rm eff}^2}{m^2}} \quad , \quad m_{\rm eff}^2 = \frac{d^2V(\phi)}{d\phi^2} \tag{9.11}$$

を用いる[*1]。Hankel 関数の引数を χ で表すと (2.12)(3.3)(3.4) より

$$\chi = \frac{k_{\rm eff}}{H_{\rm ini}} = \frac{k_{\rm eff,pr}}{a'_{\rm ini}} = k_{\rm eff}\sqrt{\frac{3}{\rho_{\rm pr,ini}}} \tag{9.12}$$

のようにプログラム上の値で与えられる。

この解析解を (3.16) に用いて、プログラム上で実装するインフラトンゆらぎ（のフーリエモード）の初期条件は

$$\begin{aligned}
\delta\phi(\boldsymbol{\kappa}) &= \frac{1}{\sqrt{2}}|\delta\phi|_{\rm pr}\left(\sqrt{-\ln X_1}e^{i2\pi Y_1} + \sqrt{-\ln X_2}e^{i2\pi Y_2}\right) \\
|\delta\phi|_{\rm pr} &= m\frac{L_{\rm pr}^{\frac{3}{2}}}{dx_{\rm pr}^3}\frac{1}{2}\sqrt{\frac{\pi}{a'_{\rm ini}}}|H_\nu(\chi)|
\end{aligned} \tag{9.13}$$

ただし Hankel 関数の第 1 種と第 2 種を表す上付き添字 (1), (2) は、この式では違いが現れないので省略した。第 1 種 Hankel 関数と第 2 種 Hankel 関数は各々 sub-horizon 極限で $e^{\mp ik\tau}$ に比例するため、3.3.3 節の等方的初期条件から両方の寄与を考慮している。これは時間微分のゆらぎを考える時に重要になる。

インフラトン場の時間微分のゆらぎは 3.3.4 節のようにモード関数の微分から求める。Hankel 関数は Bessel 関数 J_ν と Nuemann 関数 N_ν を用いて

$$H_\nu^{(1)} = J_\nu + iY_\nu \quad , \quad H_\nu^{(2)} = J_\nu - iY_\nu$$

と書け、これらの満たす微分漸化式を用いて計算される (3.17) に対応するプログラム上の微分ゆらぎは

$$\begin{aligned}
\delta\phi'(\boldsymbol{\kappa}) = &\left[\left(\nu - \frac{3}{2}\right)a'_{\rm ini} - k_{\rm eff,pr}\frac{J_\nu J_{\nu-1} + Y_\nu Y_{\nu-1}}{|H_\nu|^2}\right]\delta\phi(\boldsymbol{\kappa}) \\
&+ ik_{\rm eff,pr}\frac{J_{\nu-1}Y_\nu + J_\nu Y_{\nu-1}}{|H_\nu|^2}\frac{|\delta\phi|_{\rm pr}}{\sqrt{2}}\left[\sqrt{-\ln X_1}e^{i2\pi Y_1} - \sqrt{-\ln X_2}e^{i2\pi Y_2}\right]
\end{aligned} \tag{9.14}$$

ここで Hankel 関数・Bessel 関数・Neumann 関数の引数は全て χ である[*2]。

[*1] 計算は付録（第 II 部の計算）を参照

[*2] 計算は付録（第 II 部の計算）を参照

9.2.3　ゲージ場のゆらぎ

ゲージ場のゆらぎもインフラトンゆらぎと同様に実装する。ただし時間成分 A_0, A_0' は 0 とする。これは（多分）Lattice シミュレーション開始時刻で Coulomb ゲージを成立させるためで、なぜ Coulomb ゲージを成立させたいのかというと、ゲージ場ゆらぎのモード関数として採用する式（Coulomb 波動関数）が Coulomb ゲージを課して導かれた結果だからである。この条件によって、Coulomb ゲージと Lorentz ゲージのどちらを Lattice シミュレーションで課すとしても初期条件は統一的に扱える。よってここではゲージ固定条件による場合分けは行わない。

解析的なモード関数を導入する前にゲージ場の空間成分 $\boldsymbol{A}$ を量子場として展開しておく

$$\hat{\boldsymbol{A}}(\tau, \boldsymbol{x}) = \sum_{\lambda=\pm} \int \frac{d^3k}{(2\pi)^3} \left[\hat{b}_{\boldsymbol{k},\lambda} \boldsymbol{\epsilon}_\lambda(\boldsymbol{k}) A_\lambda(\tau, \boldsymbol{k}) e^{-i\boldsymbol{k}\cdot\boldsymbol{x}} + h.c. \right] \tag{9.15}$$

ここで λ は 2 つの円偏光を表す添字、$\hat{b}$ はゲージ場の消滅演算子、$\boldsymbol{\epsilon}$ は偏光ベクトル、A_λ は各偏光のモード関数である。円偏光の偏光ベクトルが満たす式は

$$\begin{gathered} \boldsymbol{k} \cdot \boldsymbol{\epsilon}_\pm(\boldsymbol{k}) = 0 \quad , \quad \boldsymbol{\epsilon}^*_{\lambda_1}(\boldsymbol{k}) \cdot \boldsymbol{\epsilon}_{\lambda_2}(\boldsymbol{k}) = \delta_{\lambda_1,\lambda_2} \\ \boldsymbol{k} \times \boldsymbol{\epsilon}_\pm(\boldsymbol{k}) = \mp ik \boldsymbol{\epsilon}_\pm(\boldsymbol{k}) \end{gathered} \tag{9.16}$$

このうち 1 式目が $A_0 = 0$ とした時の Coulomb ゲージ条件 $\nabla \cdot \boldsymbol{A} = 0$ に対応する（なのでやはり Coulomb ゲージで Lattice を定式化した方が良いのでは?）。線形摂動論を用いると、このゲージ偏光モード関数 $A_\lambda(\tau, \boldsymbol{k})$ に対する運動方程式が得られ、その解は Coulomb 波動関数 F, G を用いて

$$A_\pm(\tau, k) = \frac{1}{\sqrt{2k}} \left[G_0(\mp\xi, -k\tau) + iF_0(\mp\xi, -k\tau) \right] \tag{9.17}$$

と書ける。ただしこの解は $\xi = \text{const}$ の場合に対する解であり、ξ は

$$\xi \equiv -a\tau \frac{\alpha \partial_t \overline{\phi}}{2f} \quad \xrightarrow{\text{de-Sitter space}} \quad \frac{\alpha \partial_t \overline{\phi}}{2fH} \tag{9.18}$$

この定義は ξ が正になるように取っているが、ξ の定義・インフラトン速度の向き（正か負か）・量子場展開における指数関数の位相の符号、の定義の仕方（符号の決め方）次第で不安定なゲージ場の偏光が変わる。今の定義では ” − ” 偏光のゲージ場が不安定である。

Sub-Horizon 極限のモード関数

(9.17) の解は Sub-Horizon の極限で

$$A_{\pm}(\tau,k) \quad \xrightarrow{-k\tau\to\infty} \quad \frac{1}{\sqrt{2k}}e^{-ik\tau} \tag{9.19}$$

のように 3.3.1 節で用いたインフラトンゆらぎのモード関数と類似した表式になるので（というより (9.17) を求める時の初期条件で (9.19) を課したので、本当は順番が逆だが）、まずはこの表式に対するゲージ場の初期条件の実装表式を求める。

インフラトンゆらぎに対しては、解析的な $|\delta\phi(k)|^2$ を再現するようにゆらぎを生成した。ゲージ場に対しては、$|A_+(k)|^2, |A_-(k)|^2$ を再現するようにゆらぎを生成する。(3.16) と同様にして、プログラム上で実装するゲージ場の各偏光フーリエモード関数は

$$\begin{aligned} A_{\pm}(\boldsymbol{\kappa}) &= \frac{1}{\sqrt{2}}|A|_{\rm pr}(\sqrt{-\ln X_{\pm,1}}e^{i2\pi Y_{\pm,1}} + \sqrt{-\ln X_{\pm,2}}e^{i2\pi Y_{\pm,2}}) \\ |A|_{\rm pr} &= m\frac{L_{\rm pr}^{\frac{3}{2}}}{dx_{\rm pr}^3}\frac{1}{\sqrt{2k_{\rm eff,pr}}} \end{aligned} \tag{9.20}$$

ここで、一様乱数 X, Y は各 $\boldsymbol{\kappa}$・各偏光ごとに異なる値を用いる（1 つの偏光モードでスカラー場 1 つと同じ自由度を使っている）。

また、プログラム上で実装するゲージ場の微分は

$$A'_{\pm}(\boldsymbol{\kappa}) = -ik_{\rm eff,pr}\frac{|A|_{\rm pr}}{\sqrt{2}}(\sqrt{-\ln X_{\pm,1}}e^{i2\pi Y_{\pm,1}} - \sqrt{-\ln X_{\pm,2}}e^{i2\pi Y_{\pm,2}}) \tag{9.21}$$

である[*3]。インフラトンゆらぎと同様に、一様乱数は (9.20) で用いた値を（同じ $\boldsymbol{\kappa}$、同じ偏光のものを）用いる。

[*3] 計算は付録を参照

Coulomb 波動関数

より一般的なモード関数の解析解である Coulomb 波動関数を用いる場合 (9.17) のゲージ場の実装表式を求める。(9.17) は $\xi = \text{const}$ の仮定の下で導かれた解であるので、Lattice シミュレーション開始時に十分 $\xi = \text{const}$ の近似が有効でなければならない。ただし ξ の表式から、この条件はインフラトンが十分 slow-roll していれば満たされるので、結局インフラトンが slow-roll している時刻から Lattice シミュレーションを開始すれば良いだけである。

まず Coulomb 波動関数の引数をプログラム上の値で表す。第 1 引数 ξ は (9.18) より

$$\xi_{\rm ini} = \frac{\alpha}{2f}\frac{m a_{\rm ini}^{s-1}\overline{\phi}'_{\rm ini}}{H_{\rm ini}} = \frac{\alpha\overline{\phi}'_{\rm ini}}{2f a'_{\rm ini}} \tag{9.22}$$

であり、第 2 引数 $-k\tau$ は χ(9.12) を用いる。

ここまで踏まえて解析解 (9.17) を用いて (9.20) と同様にすると、プログラム上で実装するゲージ場の各偏光フーリエモード関数は

$$\begin{aligned} A_\pm(\boldsymbol{\kappa}) &= \frac{1}{\sqrt{2}}|A_\pm|_{\rm pr}\left(\sqrt{-\ln X_{\pm,1}}e^{i2\pi Y_{\pm,1}} + \sqrt{-\ln X_{\pm,2}}e^{i2\pi Y_{\pm,2}}\right) \\ |A_\pm|_{\rm pr} &= m\frac{L_{\rm pr}^{\frac{3}{2}}}{dx_{\rm pr}^3}\sqrt{\frac{G_0^2(\mp\xi_{\rm ini},\chi)+F_0^2(\mp\xi_{\rm ini},\chi)}{2k_{\rm eff,pr}}} \end{aligned} \tag{9.23}$$

ここで $\sqrt{G_0^2(\mp\xi_{\rm ini},\chi)+F_0^2(\mp\xi_{\rm ini},\chi)}$ はインフラトンゆらぎ (9.13) における $|H_\nu(\chi)|$ に対応する部分である。Hankel 関数と同様に (9.17) で記した $G+iF$ が $e^{-ik\tau}$ の波に対応し、$G-iF$ が $e^{ik\tau}$ の波に対応するので、以下の時間微分の計算では少々注意が必要。

プログラム上で実装するゲージ場の微分は

$$\begin{aligned} A'_\pm(\boldsymbol{\kappa}) = &\left[k_{\rm eff,pr}\sqrt{1+\xi_{\rm ini}^2}\frac{G_0G_1+F_0F_1}{G_0^2+F_0^2} \pm k_{\rm eff,pr}\xi_{\rm ini} - a'_{\rm ini}\right]A_\pm(\boldsymbol{\kappa}) \\ &- \frac{i}{\sqrt{2}}\frac{k_{\rm eff,pr}}{\sqrt{G_0^2+F_0^2}}\frac{|A_\pm|_{\rm pr}}{\sqrt{G_0^2+F_0^2}}\left[\sqrt{-\ln X_{\pm,1}}e^{i2\pi Y_{\pm,1}} - \sqrt{-\ln X_{\pm,2}}e^{i2\pi Y_{\pm,2}}\right] \end{aligned} \tag{9.24}$$

である[*4]。ここで Coulomb 波動関数の引数は全て $(\mp\xi_{\rm ini},\chi)$ である。インフラトンゆらぎと同様に、一様乱数は (9.20) で用いた値を（同じ $\boldsymbol{\kappa}$、同じ偏光のものを）用いる。

[*4] 計算は付録を参照

モード関数からベクトルへ

上の議論でゲージ場の各偏光フーリエモードの実装表式を求めた。3.2 節で述べたように、Lattice シミュレーションではゆらぎのフーリエモードを逆離散フーリエ変換することで実空間のゆらぎを求める。ゲージ場はベクトル場であるので、この手続きを x, y, z の各成分に対して行うが、先ほど得たのは偏光（2 成分）のフーリエモードである。よって偏光のフーリエモードから x, y, z の各フーリエモードを求める必要がある。つまり

$$A_{\pm}(\boldsymbol{\kappa}) \quad \xrightarrow{\text{polarization vector}} \quad \boldsymbol{A}(\boldsymbol{\kappa})$$

の手続きが必要である。これは偏光ベクトル $\boldsymbol{\epsilon}_{\pm}(\boldsymbol{\kappa})$ を掛けることで得られる。

$$\boldsymbol{A}(\boldsymbol{\kappa}) = \boldsymbol{\epsilon}_{+}(\boldsymbol{\kappa})A_{+}(\boldsymbol{\kappa}) + \boldsymbol{\epsilon}_{-}(\boldsymbol{\kappa})A_{-}(\boldsymbol{\kappa}) \tag{9.25}$$

つまり偏光ベクトルの表式を指定してやれば、$A_{\pm}(\boldsymbol{\kappa})$ から (9.25) によって $\boldsymbol{A}(\boldsymbol{\kappa})$ が求められ、x, y, z の各成分に対して逆離散フーリエ変換することで $\boldsymbol{A}(\boldsymbol{n})$ を求めることができる。

それで、問題となる偏光ベクトルの表式だが、(9.16) を満たす必要がある。ただし Lattice 上で $\boldsymbol{k}$ に対応するものは有効運動量 $\boldsymbol{k}_{\text{eff}}$ であるので、Lattice 上の偏光ベクトルが満たす式は

$$\begin{gathered}\boldsymbol{k}_{\text{eff}} \cdot \boldsymbol{\epsilon}_{\pm}(\boldsymbol{k}_{\text{eff}}) = 0 \quad , \quad \boldsymbol{\epsilon}^{*}_{\lambda_1}(\boldsymbol{k}_{\text{eff}}) \cdot \boldsymbol{\epsilon}_{\lambda_2}(\boldsymbol{k}_{\text{eff}}) = \delta_{\lambda_1,\lambda_2} \\ \boldsymbol{k}_{\text{eff}} \times \boldsymbol{\epsilon}_{\pm}(\boldsymbol{k}_{\text{eff}}) = \mp ik\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k}_{\text{eff}})\end{gathered} \tag{9.26}$$

これを満たす具体的な表式は

$$\boldsymbol{\epsilon}_{\pm} = \begin{cases} \frac{1}{\sqrt{2}}(1, \pm i, 0) & k_{\text{eff},1} = k_{\text{eff},2} = 0 \\ \frac{\left(-k_{\text{eff},1}k_{\text{eff},3} \pm ik_{\text{eff},2}k_{\text{eff}}, -k_{\text{eff},2}k_{\text{eff},3} \mp ik_{\text{eff},1}k_{\text{eff}}, k_{\text{eff}}^2 - k_{\text{eff},3}^2\right)}{\sqrt{2k_{\text{eff}}^2(k_{\text{eff}}^2 - k_{\text{eff},3}^2)}} & \text{それ以外} \end{cases} \tag{9.27}$$

である[*5]。実際に実装する際は k_{eff} は全て $k_{\text{eff,pr}}$ で実装するが、上式では可読性の都合上省略した（無次元量なので「(9.27) を実装する」という言い方をしたとしても正しい）。ちなみに [2] では $\boldsymbol{k}_{\text{eff}}$ ではなく $\boldsymbol{\kappa}$ を用いて定義していた。

[*5] 計算は付録を参照

9.3　発展方程式

本節では Lattice シミュレーションで実装する発展方程式を記す（導出は付録を参照）。(9.1) のラグランジアンから Euler-Lagrange 方程式を求めるだけだが、摂動論とは異なり Lattice では場はそのまま扱う。つまり ϕ の発展方程式は

$$\partial_\mu \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi)} - \frac{\partial \mathcal{L}}{\partial \phi} = 0$$

であり、$\overline{\phi}$ と $\delta\phi$ の方程式に分解する必要はない。上式から得られるインフラトン場 ϕ の発展方程式は

$$\partial_\tau^2 \phi + 2\mathcal{H}\partial_\tau \phi - \nabla^2 \phi + a^2 \frac{dV}{d\phi} = -\frac{1}{a^2}\frac{\alpha}{f}(\nabla \times \boldsymbol{A}) \cdot (\partial_\tau \boldsymbol{A} - \nabla A_0) \tag{9.28}$$

プログラム上の値に直すと

$$\phi'' = -(s+2)\frac{a'}{a}\phi' + \frac{\nabla_{\rm pr}^2 \phi}{a^{2s}} - a^2 \frac{dV_{\rm pr}}{d\phi} - \frac{\alpha}{f}\frac{1}{a^{2s+2}}(\nabla_{\rm pr} \times \boldsymbol{A}) \cdot (a^s \boldsymbol{A}' - \nabla_{\rm pr} A_0) \tag{9.29}$$

Coulomb ゲージ（$\nabla \cdot \boldsymbol{A} = 0,\ A_0 = 0$）を採用した場合は

$$\phi'' = -(s+2)\frac{a'}{a}\phi' + \frac{\nabla_{\rm pr}^2 \phi}{a^{2s}} - a^2 \frac{dV_{\rm pr}}{d\phi} - \frac{\alpha}{f}\frac{1}{a^{s+2}}(\nabla_{\rm pr} \times \boldsymbol{A}) \cdot \boldsymbol{A}' \tag{9.30}$$

これらの式の右辺が (4.11) の $F^\phi(\tau_{\rm pr}, \boldsymbol{n})$ に対応する。またこの発展方程式は各格子点の $\phi(\tau_{\rm pr}, \boldsymbol{n})$ に対して定義されるため、方程式の数で言えば N^3 個ある。

ゲージ場の発展方程式も Euler-Lagrange 方程式から

$$\begin{cases} \nabla^2 A_0 - \partial_\tau(\nabla \cdot \boldsymbol{A}) + \dfrac{\alpha}{f}(\nabla\phi) \cdot (\nabla \times \boldsymbol{A}) = 0 \\ \partial_\tau^2 \boldsymbol{A} - \nabla^2 \boldsymbol{A} - \dfrac{\alpha}{f}\partial_\tau \phi(\nabla \times \boldsymbol{A}) + \dfrac{\alpha}{f}(\nabla\phi) \times (\partial_\tau \boldsymbol{A} - \nabla A_0) = 0 \end{cases} \tag{9.31}$$

これらはゲージ固定条件を課していない方程式である。リスケーリングを考慮して、Lorentz ゲージ ($\partial^\mu A_\mu = 0$) を課すと

$$\begin{cases} A_0'' = -s\dfrac{a'}{a}A_0' + \dfrac{\nabla_{\rm pr}^2 A_0}{a^{2s}} + \dfrac{\alpha}{f}\dfrac{1}{a^{2s}}(\nabla_{\rm pr}\phi) \cdot (\nabla_{\rm pr} \times \boldsymbol{A}) \\ \boldsymbol{A}'' = -s\dfrac{a'}{a}\boldsymbol{A}' + \dfrac{\nabla_{\rm pr}^2 \boldsymbol{A}}{a^{2s}} + \dfrac{\alpha}{f}\dfrac{\phi'}{a^s}(\nabla_{\rm pr} \times \boldsymbol{A}) - \dfrac{\alpha}{f}\dfrac{1}{a^{2s}}(\nabla_{\rm pr}\phi) \times (a^s \boldsymbol{A}' - \nabla_{\rm pr} A_0) = 0 \end{cases} \tag{9.32}$$

一方、Coulomb ゲージを課すと A_0 は消去できて

$$\begin{cases} (\nabla_{\rm pr}\phi)\cdot(\nabla_{\rm pr}\times\boldsymbol{A})=0 \\ \boldsymbol{A}''=-s\dfrac{a'}{a}\boldsymbol{A}'+\dfrac{\nabla^2_{\rm pr}\boldsymbol{A}}{a^{2s}}+\dfrac{\alpha}{f}\dfrac{\phi'}{a^s}(\nabla_{\rm pr}\times\boldsymbol{A})-\dfrac{\alpha}{f}\dfrac{1}{a^s}(\nabla_{\rm pr}\phi)\times\boldsymbol{A}'=0 \end{cases} \tag{9.33}$$

となる。実際には 1 式目は時間発展には用いず、この関係式が破られないかを確かめるだけである。

スケール因子の発展方程式は Einstein 方程式であり、今回は空間成分の方程式（一般に Friedmann 方程式と呼ばれない方）を用いる。なぜなら自然に 2 階微分が現れるからと、ゲージ場が $p=\frac{1}{3}\rho$ を満たすために $\langle\rho-3p\rangle$ に寄与せず実装が簡単になるからである。

$$\frac{\partial_\tau^2 a}{a}=\frac{a^2}{6}(\rho_\phi-3p_\phi) \tag{9.34}$$

プログラム上の値に直すと

$$a''=-s\frac{(a')^2}{a}+\frac{a^3}{6}\left\langle\rho_{\phi,{\rm pr}}-3p_{\phi,{\rm pr}}\right\rangle \tag{9.35}$$

ここで $\langle\cdots\rangle$ は格子平均 (5.1) であり、インフラトンのエネルギー密度と圧力は

$$\rho_{\phi,{\rm pr}}=\frac{(\phi')^2}{2a^2}-\frac{\phi\nabla^2_{\rm pr}\phi}{2a^{2s+2}}+V_{\rm pr} \tag{9.36}$$

$$p_{\phi,{\rm pr}}=\frac{(\phi')^2}{2a^2}+\frac{\phi\nabla^2_{\rm pr}\phi}{6a^{2s+2}}-V_{\rm pr} \tag{9.37}$$

ゲージ場のエネルギー密度は

$$\rho_{A,{\rm pr}}=\frac{1}{2a^{2s+4}}\left[\sum_i(a^sA_i'-\partial_{{\rm pr},i}A_0)^2+\sum_{i,j}(\partial_{{\rm pr},i}A_j\partial_{{\rm pr},i}A_j-\partial_{{\rm pr},i}A_j\partial_{{\rm pr},j}A_i)\right] \tag{9.38}$$

である。これはゲージ固定条件を課していない表式であり、Coulomb ゲージを課せば

$$\rho_{A,{\rm pr}}=\frac{1}{2a^{2s+4}}\left[\sum_i(a^sA_i')^2+\sum_{i,j}(\partial_{{\rm pr},i}A_j\partial_{{\rm pr},i}A_j-\partial_{{\rm pr},i}A_j\partial_{{\rm pr},j}A_i)\right] \tag{9.39}$$

ゲージ場の圧力は前述したように $p_{A,{\rm pr}}=\frac{1}{3}\rho_{A,{\rm pr}}$ である。

スケール因子の発展に用いなかった Friedmann 方程式

$$\mathcal{H}^2 = \frac{a^2}{3}\rho \tag{9.40}$$

は時間発展に伴って破れないかどうか調べておく。具体的には

$$E \equiv \frac{3\mathcal{H}^2}{a^2\rho} = \frac{3\left(\frac{ma^s a'}{a}\right)^2}{a^2 m^2 a^{2s}\left\langle\rho_{\rm pr}\right\rangle} = \frac{3(a')^2}{a^4\left\langle\rho_{\rm pr}\right\rangle} \tag{9.41}$$

という値が 1 から外れないかどうかを調べる。ここで $\rho_{\rm pr} = \rho_{\phi,{\rm pr}} + \rho_{A,{\rm pr}}$ は全エネルギー密度である。このような、整合性のために調べておく条件（式）を「Consistency Condition」と呼ぶこととする。

もう 1 つの Consistency Condition としてゲージ固定条件を調べる。Lorentz ゲージ条件は

$$G \equiv \frac{\partial^\mu A_\mu}{\sqrt{\sum_\rho |\partial^\rho A_\rho|^2}} = \frac{-\partial_\tau A_0 + \nabla\cdot\boldsymbol{A}}{\sqrt{|\partial_\tau A_0|^2 + |\partial_i A_i|^2}} = \frac{-a^s A_0' + \nabla_{\rm pr}\cdot\boldsymbol{A}}{\sqrt{a^{2s}|A_0'|^2 + |\partial_{{\rm pr},i} A_i|^2}} \tag{9.42}$$

という値が 1 より十分小さいことを確認しておく。Coulomb ゲージ条件は

$$G \equiv \frac{\nabla\cdot\boldsymbol{A}}{\sqrt{\sum_i |\partial_i A_i|^2}} = \frac{\nabla_{\rm pr}\cdot\boldsymbol{A}}{\sqrt{\sum_i |\partial_{{\rm pr},i} A_i|}} \tag{9.43}$$

という値が 1 より十分小さいことを確認しておく。Coulomb ゲージ条件を課す場合は、さらにもう 1 つの式も破れないかを調べておくべきである。これはゲージ場の時間成分 A_0 に関する運動方程式に対応する量であり

$$F \equiv \frac{(\nabla_{\rm pr}\phi)\cdot(\nabla_{\rm pr}\times\boldsymbol{A})}{\sqrt{\sum_i |\partial_{{\rm pr},i}\phi(\nabla_{\rm pr}\times\boldsymbol{A})_i|^2}} \tag{9.44}$$

が 1 より十分小さいことを確認しておく。（今のところ、Lattice 平均は最後に取るようにしている）

9.4　アウトプット

本節では出力する物理量を説明する。5 章で述べた e-folds・Hubble パラメータ・インフラトン場に関する平均値・インフラトン場（これは曲率ゆらぎと等価）に関するキュムラント・インフラトンゆらぎ（これは曲率ゆらぎと等価）の PDF・インフラトンゆらぎ（これは曲率ゆらぎと等価）のパワースペクトルに加え、プログラム時間 $\tau_{\rm pr}$・スケール因子 a・slow-roll パラメータ・有効結合定数 ξ・Backreaction の強さに関する無次元量・ゲージ場の各偏光モードスペクトル・PBH 存在量 $f_{\rm PBH}$ を出力する。このうち $\tau_{\rm pr}, a$ はプログラム上の値をそのまま出力する。

9.4.1　slow-roll パラメータ

slow-roll パラメータ ϵ, η として信じる定義は

$$\epsilon \equiv -\frac{\partial_t H}{H^2} \quad , \quad \eta \equiv \frac{d\ln\epsilon}{dN_e} \tag{9.45}$$

とする。これらは時間のみに依存する背景量であることを注意しておく（定義というか出発点となる表式を変えると、どのタイミングで格子平均を取るか変わる。結果がそのタイミングに依らない場合に良い Lattice シミュレーションと言えるのだろうけど、、、）。slow-roll パラメータは無次元量なのでリスケーリングの影響を受けず、$\epsilon = \epsilon_{\rm pr}, \eta = \eta_{\rm pr}$ である。注意が必要なのは、「リスケーリングの影響を受けないこと」と「プログラム上の値で書き直すこと」は別である。

slow-roll パラメータ ϵ は Consistency Condition(6.20) が十分良く成立していると仮定して

$$\epsilon = \frac{3}{2}\left(1 + \frac{\langle p_{\rm pr}\rangle}{\langle \rho_{\rm pr}\rangle}\right) \tag{9.46}$$

(6.21) から解析的な（not Lattice の）計算を進めた表式 $\epsilon = \frac{3}{2}\left(1 + \frac{p}{\rho}\right)$ を定義とすると、Lattice で計算するべきは $\epsilon = \frac{3}{2}\left(1 + \langle\frac{p}{\rho}\rangle\right)$ のように格子平均のタイミングが異なる。

9.4.2　相互作用に関する量

9.4.3　PBH 関係量

axion-U(1) インフレーションでは曲率ゆらぎが増加するため、PBH が形成される可能性がある。本節ではこのことに基づいて、Lattice シミュレーションで得られた曲率ゆらぎの振幅から PBH 存在量 $f_{\rm PBH}$ を数値的に計算する手法を説明する（ただし曲率ゆらぎはガウス分布に従うことを仮定する）。PBH 存在量 $f_{\rm PBH}$ を求めるための式は

$$f_{\rm PBH} \approx \left(\frac{\beta}{5.9\times10^{-9}}\right)\left(\frac{\gamma}{0.2}\right)^{\frac{1}{2}}\left(\frac{g_*}{106.75}\right)^{-\frac{1}{4}}\left(\frac{M_{\rm PBH}}{M_\odot}\right)^{-\frac{1}{2}} \tag{9.47}$$

であり、インプットパラメータである割合 γ と形成時の有効自由度 g_* を除いて、形成される PBH の質量 $M_{\rm PBH}$ と形成時の存在量 β を求める必要がある。以下ではこれらを求める手法と必要なインプットパラメータを記す。

PBH 質量

PBH 質量は共動的な運動量スケール k との関係

$$M_{\rm PBH} \approx 2.8\times10^{-12}\left(\frac{\gamma}{0.2}\right)\left(\frac{g_*}{106.75}\right)^{-\frac{1}{6}}\left(\frac{k}{10^{12}\ {\rm Mpc}^{-1}}\right)^{-2} M_\odot \tag{9.48}$$

を用いて求める。共動的な運動量スケール k は、PBH が形成される時刻における共動的な Horizon スケールの逆数 aH であり、以降 PBH スケールと呼ぶ。理論的な関係式 $k=aH$ は現在時刻のスケール因子が 1 であるという規格化のもとで（多分）成立していて、Lattice シミュレーションでのスケール因子の規格化と異なるため単純に $(aH)_{\rm pr}$ として計算しても理論的な PBH スケールを求めることはできない。そこで CMB スケールを参照スケールとして導入して

$$\begin{aligned}\frac{k}{k_{\rm CMB}} &= \frac{aH}{(aH)_{\rm CMB}} = e^{N_e(k)-N_{e,\rm CMB}}\frac{H}{H_{\rm CMB}} \\ &= e^{N_{e,\rm ini}-N_{e,\rm CMB}+\Delta N_e}\frac{H}{m}\left(\frac{H_{\rm CMB}}{m}\right)^{-1}\end{aligned} \tag{9.49}$$

のように e-folds で表せば、スケール因子の規格化に関係なく計算することができる。ここで $N_{e,\rm ini}$ は格子シミュレーション開始時の e-folds であり、$N_{e,\rm CMB}$ は CMB スケール $k_{\rm CMB}$ が super-horizon になった時の e-folds である。また ΔN_e は Lattice シミュレーション開始から、注目スケール k が super-horizon になるまでの e-folds である。(9.49) により、Lattice 上の有効運動量 $k_{\rm eff}$ が Lattice シミュレーション上で super-horizon に

なる時刻（つまり $k_{\mathrm{eff,pr}} = (aH)_{\mathrm{pr}}$ となる時刻）における $\Delta N_e, \frac{H}{m}$ を求めれば、k_{eff} に対応する k が求められる。上式におけるインプットパラメータは $N_{e,\mathrm{ini}} - N_{e,\mathrm{CMB}}$, $\frac{H_{\mathrm{CMB}}}{m}$, k_{CMB} であり、前者 2 つは背景インフレーションの発展から求める。

次に Lattice シミュレーションにおいて、各 k_{eff} に対応する $\Delta N_e, \frac{H}{m}$ を求める手法を述べる。前述の通り $k_{\mathrm{eff,pr}} = (aH)_{\mathrm{pr}}$ となる時刻で $\Delta N_e, \frac{H}{m}$ を評価する。ただし Lattice シミュレーションでは、時間発展が離散化されているので直線近似を行う。具体的には、離散的な時間発展によって厳密に $k_{\mathrm{eff,pr}} = (aH)_{\mathrm{pr}}$ となる時刻を求めることはできないので、代わりに直前の時刻における $(\Delta N_{e,\mathrm{pre}}, (aH)_{\mathrm{pr,pre}})$ の値と直後の時刻における $(\Delta N_{e,\mathrm{post}}, (aH)_{\mathrm{pr,post}})$ の値を用いて、その 2 点間を直線で近似して求める。以下に直線近似を表す図を示す。この図をもとに、ΔN_e を求める式は

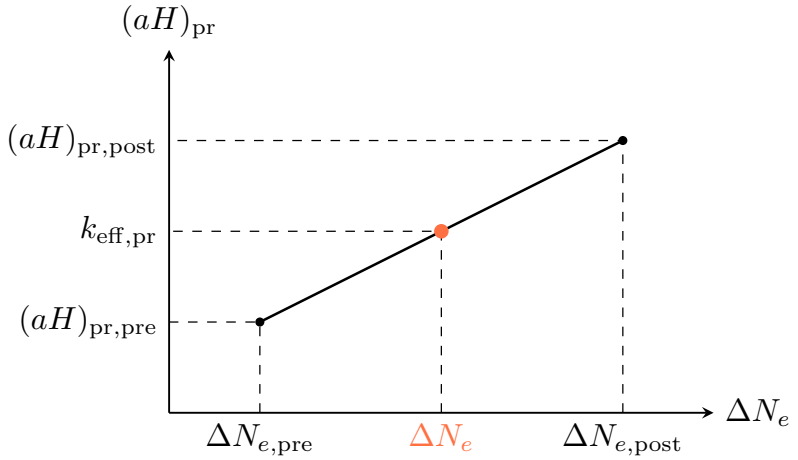

図 9.1　$k_{\mathrm{eff,pr}} = (aH)_{\mathrm{pr}}$ となる時刻 ΔN_e を求めるための直線近似

$$\Delta N_e = \Delta N_{e,\mathrm{pre}} + \frac{\Delta N_{e,\mathrm{post}} - \Delta N_{e,\mathrm{pre}}}{(aH)_{\mathrm{pr,post}} - (aH)_{\mathrm{pr,pre}}} \left[k_{\mathrm{eff,pr}} - (aH)_{\mathrm{pr,pre}}\right] \tag{9.50}$$

である。同様に $\frac{H}{m}$ についても直線近似を用いれば、上で求めた ΔN_e から

$$\frac{H}{m} = \left(\frac{H}{m}\right)_{\mathrm{pre}} + \frac{\left(\frac{H}{m}\right)_{\mathrm{post}} - \left(\frac{H}{m}\right)_{\mathrm{pre}}}{\Delta N_{e,\mathrm{post}} - \Delta N_{e,\mathrm{pre}}} \left[\Delta N_e - \Delta N_{e,\mathrm{pre}}\right] \tag{9.51}$$

と計算できる。このようにして求めた $\Delta N_e, \frac{H}{m}$ とインプットパラメータを用いれば、k_{eff} に対応する共動的運動量スケール k が求められ、(9.48) を用いて k_{eff} に対応する PBH 質量を求めることができる。

PBH 形成時の存在量

形成時の存在量（あるいは Volume fraction）β は、密度ゆらぎがガウス分布に従うと仮定した場合

$$\beta = \gamma \int_{\delta_c}^{\infty} \frac{d\delta}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\delta^2}{2\sigma^2}\right) = \frac{\gamma}{2}\mathrm{erfc}\left(\frac{\delta_c}{\sqrt{2}\sigma}\right) \tag{9.52}$$

で求められる。割合 γ と崩壊の閾値 δ_c はインプとパラメータであるが、smoothing (or filtering) したゆらぎの分散 σ^2 は無次元曲率スペクトル $P_{\mathcal{R}}$ を用いて

$$\sigma^2(k) = \frac{16}{81}\int_0^{\infty} d\ln q \left(\frac{q}{k}\right)^4 \exp\left(-\frac{q^2}{k^2}\right) P_{\mathcal{R}}(q) \tag{9.53}$$

で計算される。ただしガウス型の窓関数を採用した。このように曲率スペクトルの振幅から β が計算できるので、これと上で求めた PBH 質量から f_{PBH} が求められる。以下では Lattice 上で得られた曲率スペクトルから（物理的な？）$\sigma(k)$ を計算する手法を説明する。

運動量ベクトル k, q に Lattice 上で対応するものは有効運動量ベクトルである。有効運動量ベクトルの絶対的な値を知るには (9.49) に従って計算する必要があるが、$\sigma(k)$ を求める際には運動量の比のみが重要になるので、深く考える必要がなく

$$\frac{q}{k} = \frac{q_{\mathrm{eff,pr}}}{k_{\mathrm{eff,pr}}} \tag{9.54}$$

を用いるだけで、理論的な量と Lattice 上の値が結びつく。このように運動量の比で $\sigma(k)$ を表すと

$$\begin{aligned}\sigma^2(k) &= \frac{16}{81}\int_0^{\infty} d\left(\frac{q}{k}\right)\left(\frac{q}{k}\right)^3 \exp\left(-\frac{q^2}{k^2}\right) P_{\mathcal{R}}(q) \\ &= \frac{16}{81}\int_0^{\infty} dx x^3 e^{-x^2} P_{\mathcal{R}}(x)\end{aligned}$$

となる。ただし $x = \frac{q}{k}$ に変数変換し、この x を引数に持つ関数としての無次元曲率スペクトル $P_{\mathcal{R}}\left(\frac{q}{k}\right)$ で表した。

Lattice シミュレーションではもう一つ問題がある。それは、調べることができる運動量スケールが有限であることである（空間の有限化と離散化による弊害)。これを考慮して、積分範囲に上限と下限を設ける。

$$\sigma^2(k_{\mathrm{eff}}) = \frac{16}{81}\int_{x_{\min}}^{x_{\max}} dx x^3 e^{-x^2} P_{\mathcal{R}}(x) \tag{9.55}$$

ただし $x_{\max}$ は $q_{\mathrm{eff},\max}$ に対応し、$x_{\min}$ は $q_{\mathrm{eff},\min} \approx \frac{2\pi}{L}$ に対応する。この近似が $\sigma(k)$ の値に影響を与えないことを確かめるために、被積分関数のスペクトル以外の部分についてその振る舞いを図 9.2 に示す。このように $\sigma(k)$ への寄与が大きいのは $x = 1$ の周り、つ

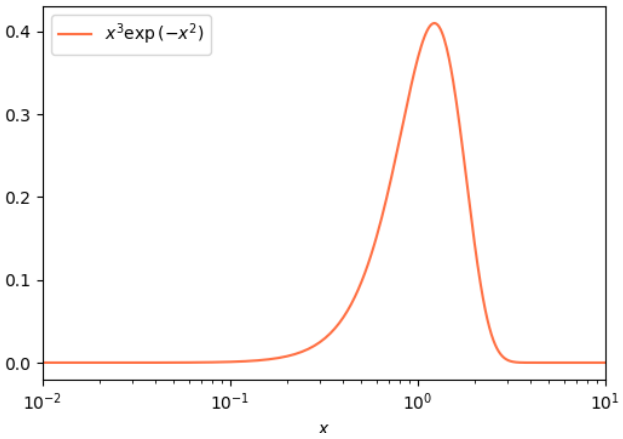

図 9.2　(9.55) の被積分関数 $x^3 e^{-x^2}$ の振る舞い

まり $q_{\mathrm{eff}} = k_{\mathrm{eff}}$ の周りのみであるため、(9.55) の積分範囲 $[x_{\min}, x_{\max}]$ が $x = 1$ 付近を含んでいれば、十分良い精度で $\sigma(k)$ を計算できると考えられる。具体的に $x^3 e^{-x^2} > 10^{-2}$ の領域は $[0.2, 2.7]$ である。このような”良い”積分範囲を確保するために、曲率スペクトルにピークがある場合はそのピークが Lattice シミュレーションで見れる運動量スケールの中央付近に来るようにパラメータを設定する必要がある。f_{PBH} を求めるにあたって重要な、f_{PBH} が大きくなる運動量スケールは無次元曲率スペクトルのピーク付近 $k_{\mathrm{eff}} \approx k_{\mathrm{eff,peak}}$ であるので

$$\sigma^2(k_{\mathrm{eff}} \approx k_{\mathrm{eff,peak}}) = \frac{16}{81} \int_{x_{\min}}^{x_{\max}} dx x^3 e^{-x^2} P_{\mathcal{R}}(x) \tag{9.56}$$

が重要な計算である。この場合 $P_{\mathcal{R}}(x)$ は $x = 1$ 付近にピークを持ち、後は $[x_{\min}, x_{\max}]$ がその周りを含んでいれば良い。

9.5　インフラトンポテンシャル

本節ではインフラトンポテンシャル $V(\phi)$ の具体的な表式と、対応するプログラム上の表式 $V_{\rm pr}, \frac{dV_{\rm pr}}{d\phi}, \frac{d^2V_{\rm pr}}{d\phi^2}$ を求める。$V_{\rm pr}$ はエネルギー密度 (9.36) と圧力 (9.37) で用いる。$\frac{dV_{\rm pr}}{d\phi}$ はインフラトン場の発展方程式 (9.29)(9.30) で用いる。$\frac{d^2V_{\rm pr}}{d\phi^2}$ はゆらぎの初期条件 (9.11) で用いる。

9.5.1　Chaotic ポテンシャル

$$V(\phi) = \frac{1}{2}m^2\phi^2 \tag{9.57}$$

に対して (1.11) より

$$\begin{aligned} V_{\rm pr} &= \frac{\phi^2}{2a^{2s}} \\ \frac{dV_{\rm pr}}{d\phi} &= \frac{\phi}{a^{2s}} \\ \frac{d^2V_{\rm pr}}{d\phi^2} &= \frac{1}{a^{2s}} \quad \overset{\text{initial}}{\to} \quad 1 \end{aligned} \tag{9.58}$$

ただし 2 階微分はゆらぎの初期条件で用いるので (3.1) を使った。

9.5.2　Bumpy Axion ポテンシャル

$$V(\phi) = \frac{1}{2}m^2\phi^2 + \Lambda^4\frac{\phi}{f}\sin\left(\frac{\phi}{f}\right) \tag{9.59}$$

に対しては、$\beta = \frac{\Lambda^4}{m^2f^2}$ として

$$\begin{aligned} V_{\rm pr} &= \frac{\phi^2}{2a^{2s}} + \frac{\beta f\phi}{a^{2s}}\sin\left(\frac{\phi}{f}\right) \\ \frac{dV_{\rm pr}}{d\phi} &= \frac{\phi}{a^{2s}} + \frac{\beta f}{a^{2s}}\left[\sin\left(\frac{\phi}{f}\right) + \frac{\phi}{f}\cos\left(\frac{\phi}{f}\right)\right] \\ \frac{d^2V_{\rm pr}}{d\phi^2} &= 1 + \beta\left[2\cos\left(\frac{\phi}{f}\right) - \frac{\phi}{f}\sin\left(\frac{\phi}{f}\right)\right] \end{aligned} \tag{9.60}$$

ただし 2 階微分は既に (3.1) を用いた表式である[*6]。

[*6] 計算は付録を参照

9.6　インプットパラメータ

本節では axion-U(1) インフレーションの Lattice シミュレーションを実行するのに必要なインプットパラメータを説明する。「型 変数名：説明」のフォーマットで説明する。型がないものはマクロ名である（マクロの説明は省略する）。重要なものや、よく値を操作するものは赤字で記載する。
Lattice 特有のパラメータは

- double L：立方領域の一辺の（共動的な）プログラム上の長さの値
- int N：一辺の格子点数（2 の累乗）
- int num_thread：OpenMP で用いるスレッド数（コンピュータごとに可能な最大値が異なるので注意）

物理的なパラメータは

- BACKREACTION：backreaction の強さの選択、廃止予定（0 → weak, 1 → stronog, 2 → bumpy）
- POLARIZATION：偏光ベクトルの定義選択（0 → [2] に基づいた κ での定義, 1 → (9.27) の $k_{\rm eff}$ による定義）
- MODEL：インフラトンポテンシャルの選択（0 → chaotic, 1 → bumpy）
- double m：換算プランク質量単位のインフラトン質量
- NDIMS：空間次元、3 にしておくべき
- int nflds：場の数、4 か 5 にしておくべき
- double initfield[]：インフラトン場の背景量初期値（換算プランク質量単位）
- double initderivs[]：インフラトン場微分の背景量初期値（換算プランク質量単位）
- double g：結合定数 $\frac{\alpha}{f}$

初期条件に関する物理的なパラメータは

- int seed：一様乱数 X, Y を生成する時のシード値
- bool hankel：インフラトンゆらぎの初期条件（true → Hankel 関数, false → 指数関数）
- bool coulomb_po：正のゲージモード A_+ の初期条件（ture → Coulomb 波動関数, false → 指数関数）

- bool coulomb_ne：負のゲージモード A_+ の初期条件（ture → Coulomb 波動関数, false → 指数関数）
- std::string path_base：クーロン波動関数の値を持つファイルのディレクトリへのパス

PBH 関係量の計算に用いるインプットパラメータは

- double kcmb：CMB の共動的な運動量スケール、0.05 か 0.002 にしておくべき
- double Hcmb：CMB スケールが crossing した時のハッブルパラメータ $\frac{H}{m}$ の値
- double Ncmb：Lattice シミュレーション開始時の e-folds から CMB スケールが crossing した時の e-folds を引いた値 $N_{e,\mathrm{ini}} - N_{e,\mathrm{CMB}}$
- double gamm：horizon 質量の内、PBH 質量になる割合 γ、0.2 にしておくべき
- double gsta：PBH 形成時の有効自由度

運動量カットオフに関係するパラメータは

- int cutoff_method：運動量カットオフの取り方（0 → (9.61), 1 → (9.62)、これらは後述）
- bool cutoff_iniax：インフラトン場の初期条件でカットオフを考慮するかどうか（true → カットオフあり, false → カットオフなし）
- bool cutoff_inigf：ゲージ場の初期条件でカットオフを考慮するかどうか（true → カットオフあり, false → カットオフなし）
- bool cutoff_speax：インフラトン場のパワースペクトルでカットオフを考慮するかどうか（true → カットオフあり, false → カットオフなし）
- bool cutoff_spegf：ゲージ場のパワースペクトルでカットオフを考慮するかどうか（true → カットオフあり, false → カットオフなし）

シミュレーションに必要なパラメータは

- double dt：1 回の時間発展に用いる時間間隔のプログラム上の値 $\Delta\tau_{\mathrm{pr}}$
- double tf：シミュレーション終了時のプログラム上の時刻

- int continue_run：Lattice シミュレーションを続けて行う場合に、何回目のシミュレーションかを表す数（基本 0 にしておく、LATTICEEASY の名残）
- char alt_extension[]：出力する dat ファイルの名前に付ける文字（設定しなくて良い）

- int noutput_times：（背景量の）値の出力を行う回数
- int print_interval：CLI に現在の $\tau_{\rm pr}$ を出力する際の（我々の世界での）時間間隔
- int screen_updates：CLI に現在の $\tau_{\rm pr}$ を出力するかどうか（1 → 出力する）
- double checkpoint_interval：dat ファイルを flush する（書き出す）プログラム上の時間間隔
- double store_lattice_times[]：触らぬ神に祟りなし、LATTICEEASY の名残

- int sbackground：背景量 (background_0.dat) を出力するかどうか（1 → 出力する）
- int sexpansion：スケール因子に関する量 (sf_0.dat) を出力するかどうか（1 → 出力する）
- int senergy：エネルギーに関する量 (energy_0.dat) と相互作用に関する量 (coupling_0.dat) を出力するかどうか（1 → 出力する）
- int sgauge：ゲージ固定条件が満たされているか確認する（1 → 出力する）
- double t_start_output：出力をいつから始めるか（0 にしておくべき）
- int sspectra：インフラトンゆらぎのパワースペクトル (spectra0_0.dat, spectra_eff0_0.dat) を出力するかどうか（1 → 出力する）
- double tspectra：インフラトンゆらぎのパワースペクトルの出力をいつから始めるか
- int spdf：インフラトンゆらぎの PDF(pdf0_0.dat) を出力するかどうか（1 → 出力する、ただし sspectra=1 でなければいけない）
- int pdfrange：インフラトンゆらぎの PDF の横軸 $\frac{\delta\phi}{\sigma}$ の最大値
- int spbh：PBH 関係量 (PBHmass_0.dat) を出力するかどうか（1 → 出力する）

「model.hpp」ヘッダファイルでインプットするパラメータは

- double rescale_B：(1.5) の位置リスケーリングパラメータ（インフラトン質量を用いるのがデフォルト）
- double rescale_s：(1.5) の時間微分リスケーリングパラメータ（1 がデフォルト）
- double ···：インフラトンポテンシャルのパラメータ群

9.7　運動量カットオフ

本節では (9.6) の有効運動量ベクトルを用いたことによる運動量カットオフについて説明する。このカットオフを導入したのは [2] であり、私もそれに倣うがあまりよくわかっていない。

(2.10) のような最大解像度の有効運動量ベクトルの大きさ k_{eff} は $|\boldsymbol{m}|$ に関して増加関数であるが、(9.6) の有効運動量ベクトルの大きさは途中から減少する。その概要図を図

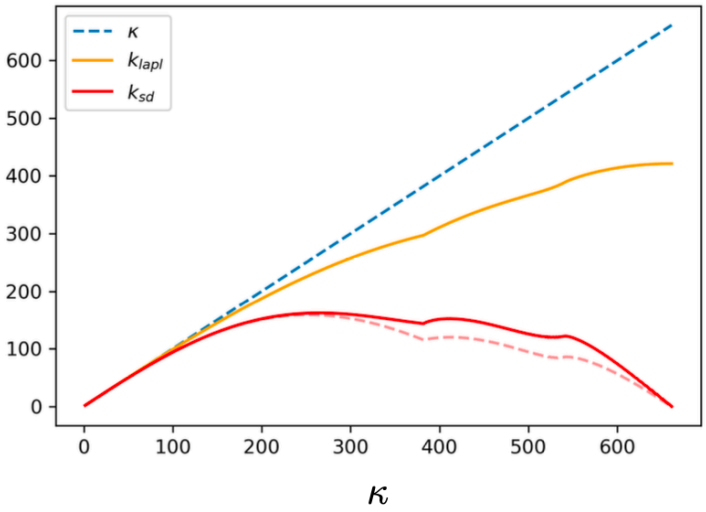

図 9.3　[2] の Figure5.1 から引用。κ（青破線）は逆格子ベクトル、k_{lapl}（黄線）は (2.10) の有効運動量ベクトル、k_{sd}（赤線）は (9.6) の有効運動量ベクトルの大きさを表す。

9.3 に示す。このように axion-U(1) インフレーションで用いる有効運動量ベクトルが減少することは sin 関数の位相が $\frac{2\pi m_i}{N}$ であることから明白で、[2] によると赤線が減少し始めるより左の k_{eff} が物理的・より右の k_{eff} は非物理的としてカットオフを設けるべきだそうだ。

以降は私が行ったカットオフの実行方法を説明する。その方法は単純で、非物理的な k_{eff} を持つフーリエモード $\phi(\boldsymbol{k}_{\mathrm{eff}}), A_\mu(\boldsymbol{k}_{\mathrm{eff}})$ の初期値を 0 とすることと、パワースペクトルの計算でこれらの寄与を 0 とすることである。時間発展は実空間上で行うのでカットオフに関して考慮することはない。

次に、カットオフの閾値の取り方を説明する。図 9.3 において横軸が逆格子ベクトルの大きさ κ であるので、本来は 1 本の実線で k_{eff} が表されるはずはなく、同一の κ に対し

て複数の k_{eff} を取れるはずである（例えば $\boldsymbol{m} = (1,1,0)$ と $\boldsymbol{m} = (2,0,0)$ では k_{eff} の値が異なる）。おそらく [2] では同一の κ で k_{eff} の平均を取っているのだと思う。図 9.4 に各

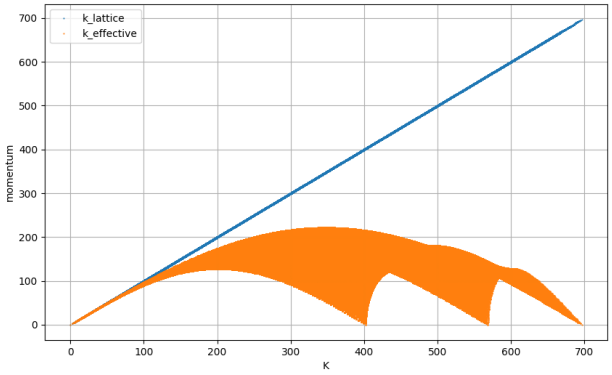

図 9.4　横軸が κ、青線が κ、オレンジ点が (9.6) の有効運動量ベクトルを表す。

k_{eff} の点をプロットしたグラフを示す。3 つの（白い）山？ はそれぞれ左から、2 つの m_i が 0・1 つの m_i が 0・3 つの m_i が有限値を取る場合を表していると予想する。[2] では $\kappa \lesssim 250$ というカットオフを設けていたが、私は保守的に $\kappa \lesssim 200$（一番左の山の頂点）とした。一般の N, L, dx の値に対する公式は

$$|\boldsymbol{m}| \leq \frac{N}{4} \tag{9.61}$$

である。あるいは sin 関数の頂点までが物理的だと思えば

$$m_x,\ m_y,\ m_z \leq \frac{N}{4} \tag{9.62}$$

でも良い。ただし後者のカットオフはサンプル数が多いけれども、$\kappa > \frac{2\pi}{L}\frac{N}{4}$ の κ に注目した時に考慮できていない m_i の組が存在してしまい、良いカットオフの取り方とはあまり思っていない。

9.8　アウトプット

単一場インフレーションの場合に加えて相互作用に関する量とゲージ固定の条件式を出力する。有効結合定数 ξ は (4.18) より

$$\xi = \frac{\alpha\,\langle\dot{\phi}\rangle}{2fH} = \frac{\alpha}{2f}\frac{\frac{\langle\dot{\phi}\rangle}{m}}{\frac{H}{m}} = \frac{\alpha}{2f}\frac{a\,\langle\check{\phi}\rangle}{\check{a}} \tag{9.63}$$

また背景量へのゲージ場の影響を測る条件式 (**??**)(**??**) は

$$\frac{H^2}{26\pi\left\langle\dot{\phi}\right\rangle}\frac{e^{\pi\xi}}{\xi^{\frac{3}{2}}}=\frac{m\left(\frac{H}{m}\right)^2}{26\pi\frac{\left\langle\dot{\phi}\right\rangle}{m}}\frac{e^{\pi\xi}}{\xi^{\frac{3}{2}}}=\frac{ma^{s-3}(\check{a})^2}{26\pi\left\langle\check{\dot{\phi}}\right\rangle}\left(\frac{\alpha}{2f}\frac{a\left\langle\check{\phi}\right\rangle}{\check{a}}\right)^{-\frac{3}{2}}\exp\left(\frac{\alpha}{2f}\frac{a\left\langle\check{\phi}\right\rangle}{\check{a}}\right) \tag{9.64}$$

$$\frac{H}{146}\frac{e^{\pi\xi}}{\xi^{\frac{3}{2}}}=\frac{ma^{s-2}\check{a}}{146}\left(\frac{\alpha}{2f}\frac{a\left\langle\check{\phi}\right\rangle}{\check{a}}\right)^{-\frac{3}{2}}\exp\left(\frac{\alpha}{2f}\frac{a\left\langle\check{\phi}\right\rangle}{\check{a}}\right) \tag{9.65}$$

で計算する。

第 10 章

ファイル構造

本章では Axion-U(1) インフレーションを Lattice シミュレーションする「AxionInflation」に含まれるファイルあるいはディレクトリの構造を説明する。構造のイメージを図 10.1 に示す。

Axion-U(1) インフレーションの Lattice シミュレーションをするためのディレクトリ（フォルダ）「AxionInflation」は 2 つのディレクトリから構成されている。1 つは Lattice シミュレーションを実行するための C++ コードファイルと、シミュレーションによって出力される dat ファイルを含む「Axion-U(1)」である。もう 1 つは、Lattice シミュレーションによって出力された dat ファイルから結果のグラフを描画するための python コードファイルを含む「visualize」である。

10.1 節では Lattice シミュレーションを実行するための「Axion-U(1)」ディレクトリについて説明し、10.2 節では dat ファイルからグラフを描画するための「visualize」ディレクトリについて説明する。

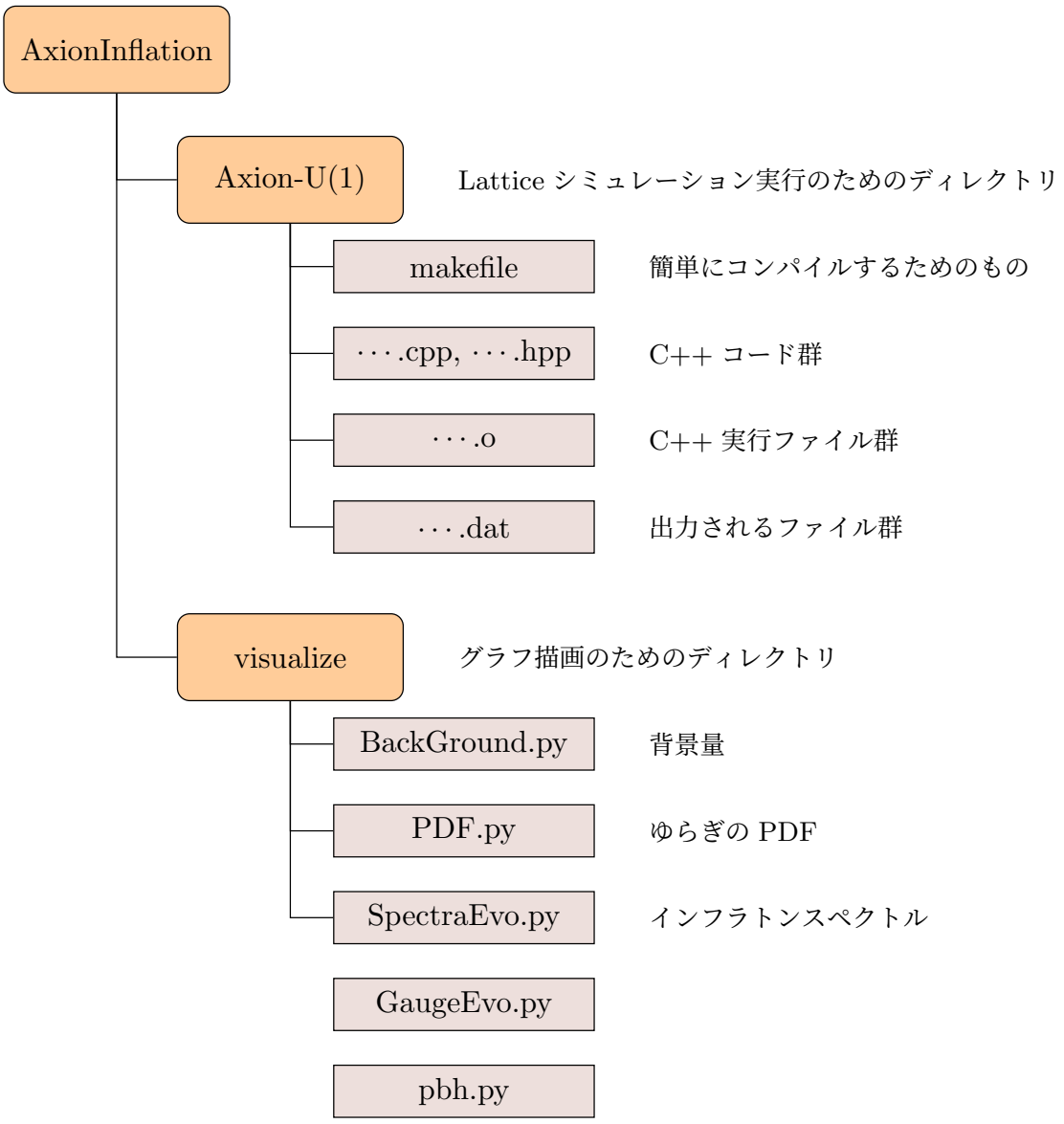

図 10.1　ファイル構造の概要図

10.1　Axion-U(1) ディレクトリ

本節では Lattice シミュレーションを実行するための C++ コードファイル（cpp 拡張子のソースファイルと hpp 拡張子のヘッダファイル）について、各ファイルの役割等を説明する。C++ コードファイルの依存関係を図 10.2 に示す。

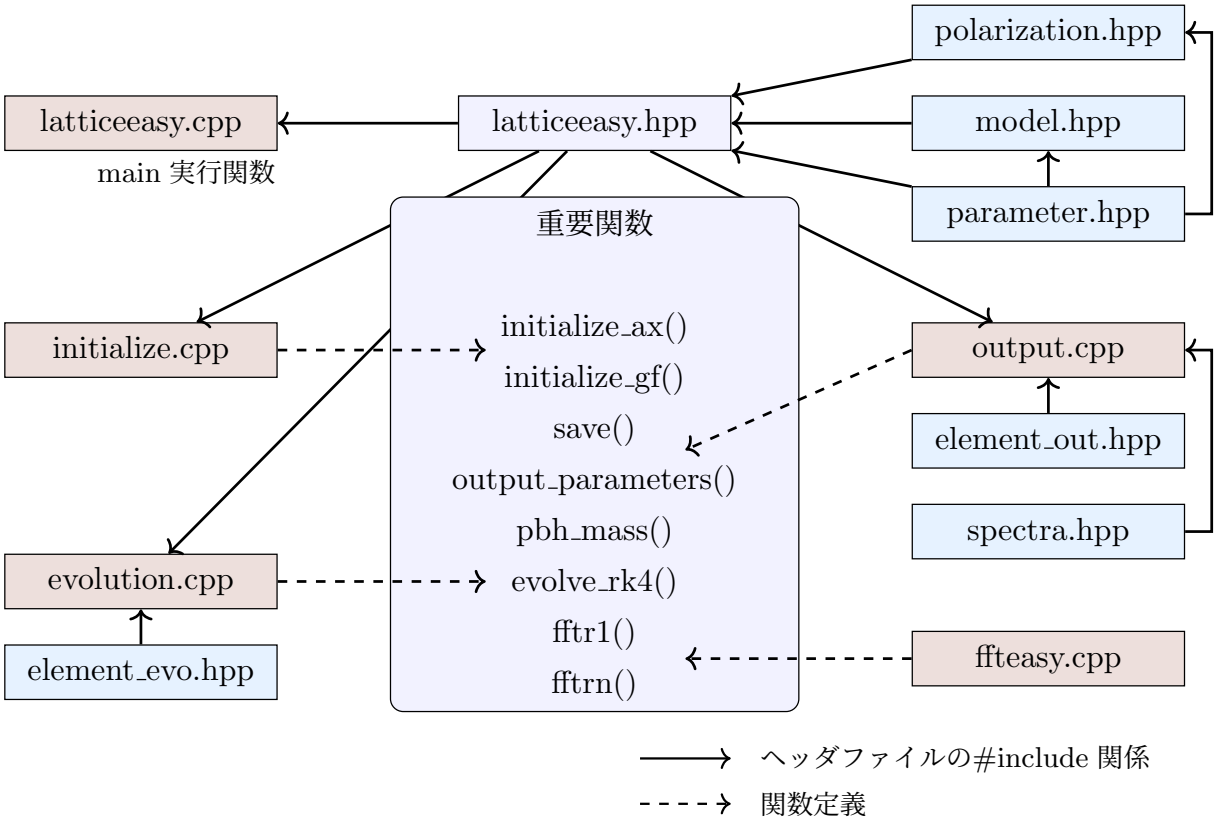

図 10.2　Lattice シミュレーションのための C++ コードファイルの依存関係。ヘッダファイル#include の矢印は IT 業界の付け方と逆かもしれないが、自分はこっちの方がわかりやすいのでこうしてます。関数定義の矢印は、latticeeasy.hpp ヘッダファイルで宣言し、各ソースファイルで実装している。

①まず見るべきは図 10.2 の上部にあるファイル群であり、「latticeeasy.cpp」が Lattice シミュレーションそのものを実行している（つまり main 実行関数を含む）ソースファイルである。またこのファイルで（インプットパラメータを除く）global 変数も宣言（定義）している。

②main 実行関数の内側で使用するいくつかの関数は「latticeeasy.hpp」ヘッダファイ

ルで宣言していて、これらの関数の実装は他の各ソースファイルで行っている（図 10.2 の重要関数のこと）。また latticeeasy.hpp に include しているヘッダファイルが 3 つあり、「parameter.hpp」と「model.hpp」と「polarization.hpp」である。parameter.hpp ではインプットパラメータに対応する global 変数を宣言していて、model.hpp ではインフラトンポテンシャルに関する関数を実装していて、polarization.hpp では（正の）偏光ベクトルの各成分に関する関数を実装している。”この 3 つのヘッダファイルを include した latticeeasy.hpp ヘッダファイル”をソースファイルに#include することで、そのソースファイルでインプットパラメータやインフラトンポテンシャルの関数を使用できるようになる。

③main 実行関数で用いる、初期化のための関数 initialize_ax() と initialize_gf() を実装しているのは「initialize.cpp」ソースファイルである。ここで背景量やゆらぎの初期化を実行している。

④main 実行関数で用いる、1 回の時間発展（4 次の Runge-Kutta 法）を行う関数 evolve_rk4() を実装しているのは「evolution.cpp」ソースファイルである。このファイルに#include している「element_evo.hpp」ヘッダファイルは evolve_rk4() でよく用いられる基本的な関数を実装したファイルである。

⑤main 実行関数で用いる、Lattice シミュレーションの結果を出力するための関数 save() と output_parameters() と pbh_mass() を実装しているのは「output.cpp」ソースファイルである。このファイルに#include している「element_out.hpp」ヘッダファイルは save() でよく用いられる基本的な関数を実装したファイルである。

⑥main 実行関数で用いる、FFT を行うための関数 fftr1() と fftrn() を実装しているのは「ffteasy.cpp」ソースファイルである。

Lattice シミュレーションのセットアップを変更したい場合は、インプットパラメータを含む「parameter.hpp」とインフラトンポテンシャルを含む「model.hpp」の中身を書き換えるだけで良い（これは LATTICEEASY[1] と同じ）。ただし、時間発展の手法などを変更したければその限りではない。

10.2　visualize ディレクトリ

第 11 章

コードレビュー

本章では「Axion-U(1)」ディレクトリの各コード（但し.o 拡張子の実行ファイルはおそらく機械語なので除く）と「visualize」ディレクトリの各 python コードの説明を行う。11.1 節では g++ で簡単にコンパイルするための「makefile」を説明、11.2 節では Lattice シミュレーションを実行するための C++ コード群を説明、11.3 節では Lattice シミュレーションで出力される dat ファイルの内容を説明、11.4 節では dat ファイルからグラフを描画するための Python コードを説明する。本章で見る C++ のコードは Lorentz ゲージ条件を課した場合のシミュレーションコードである。

技術的な話ばかりで冗長になるので、スキップして第 IV 部に飛ぶのを推奨する。自分でコードを書き換える際には本章を参考にされたし。プログラム上のコメントアウトは青色で表記する。また FFT の関数を実装している ffteasy.cpp は 8 章のレビューと同じなのでここでは行わない。やる気が出たら書きます。

11.1　makefile

Code 11.1　makefile

```
1  COMPILER = g++
2  FLAGS = -O3 -fopenmp #-fstack-usage
3  HEADA = latticeeasy.hpp model.hpp parameters.hpp polarization.hpp
4  SOURCE = ffteasy.o latticeeasy.o evolution.o initialize.o output.o
5  all: $(HEADA) spectra.hpp element_evo.hpp element_out.hpp $(SOURCE)
6          $(COMPILER) $(FLAGS) $(SOURCE) -lm -o latticeeasy
7
8  clean:
9          rm -f *.o out output.txt latticeeasy *~ *.dat core*
10 cleaner:
11         rm -f *.o out output.txt latticeeasy *~ *.dat *.img
12
13 ffteasy.o: ffteasy.cpp
14         $(COMPILER) -c $(FLAGS) ffteasy.cpp
15 latticeeasy.o: latticeeasy.cpp $(HEADA)
16         $(COMPILER) -c $(FLAGS) latticeeasy.cpp
17 evolution.o: evolution.cpp $(HEADA) element_evo.hpp
18         $(COMPILER) -c $(FLAGS) evolution.cpp
19 initialize.o: initialize.cpp $(HEADA)
20         $(COMPILER) -c $(FLAGS) initialize.cpp
21 output.o: output.cpp $(HEADA) element_out.hpp spectra.hpp
22         $(COMPILER) -c $(FLAGS) output.cpp
```

mekefile は簡単にコンパイルするためのファイルで、ソースファイル (cpp) とヘッダファイル (hpp) を複数用いる際のリンクを設定できる。

1 ～ 4 行目はこのファイル上の変数定義で、各々コンパイラ・コンパイルオプション・よく用いるヘッダファイル・全実行ファイルを表している。定義した変数を用いるには「$(変数名)$」と表記する。2 行目のコンパイルオプション変数において、「-03」は良く用いられる自動で効率化してくれる（？）オプション、「-fopenmp」は OpenMP を用いるオプションである。コメントアウトしている「-fstack-usage」は各実行ファイルの使用するスタック領域のメモリを確認するためのオプションである（確認方法は 13.2 節を参照）。

5,6 行目はコンパイルの命令で、5 行目には全てのヘッダファイルと全ての実行ファイル、6 行目にはコンパイラとコンパイルオプションと全ての実行ファイルを書く。「-lm -o latticeeasy」はコンパイルしたプログラムの名前を設定していて、Lattice シミュレーションを実行する時に「./latticeeasy」と命令すれば良くなる。

8 ～ 11 行目は、Lattice シミュレーションで作成された dat ファイルを削除するための命令「make clean」あるいは「make cleaner」を定義している。

13 ～ 22 行目は各々実行ファイルを設定している。各実行ファイルの設定は、1 行目に「実行ファイル名: 関係するファイル名全て」を書き、2 行目に「コンパイラ -c コンパイルオプション メインとなるソースファイル名」を書く。関係するファイルとは、メインとなるソースファイル・このソースファイルで実装した関数を利用しているファイル・インクルードされるヘッダファイルである。例えば initialize() は latticeeasy.cpp で用いるので、関係するファイルに latticeeasy.cpp を含める（と思ってたのだけれどどうやら書かなくても良いらしい)。

11.2　C++ コード

11.2.1　latticeeasy.cpp

11.2.2　latticeeasy.hpp

11.2.3　parameters.hpp

11.2.4　model.hpp

11.2.5　polarization.hpp

11.2.6　initialize.cpp

11.2.7　evolution.cpp, element_evo.hpp

11.2.8　output.cpp, element_out.hpp

11.3　dat ファイル

11.4　Python コード

第 IV 部

利用に関して

第 12 章

利用上の注意事項

本章では Lattice シミュレーションコード「SFInflation」「Axion-U(1)」を利用する際の注意事項を説明する。12.1 節では使用するための権限について、12.2 節ではプログラム上の時間間隔が満たすべき条件、12.3 節ではシミュレーションするために必要なメモリの見積もり、12.4 節では動的に確保したメモリがリークする可能性についての注意点を説明する。適宜追加予定。

12.1　使用権限

本節ではシミュレーションコードを使用する際の権限を記しておく。私は LATTICEEASY[1] のコードを書き換えただけなので、その権限は LATTICEEASY 開発者に帰属する（日本語あってる？ ）。その内容は

LATTICEEASY consists of the C++ files ”latticeeasy.cpp,” ”initialize.cpp,” ”evolution.cpp,” ”output.cpp,” ”latticeeasy.h,” ”parameters.h,”. (The distribution also includes the file fftеasy.cpp but this file is distributed separately and therefore not considered part of the LATTICEEASY distribution in what follows.) LATTICEEASY is free. We are not in any way, shape, or form expecting to make money off of these routines. We wrote them for the sake of doing good science and we’re putting them out on the Internet in case other people might find them useful. Feel free to download them, incorporate them into your code, modify them, translate the comment lines into Swahili, or whatever else you want. What we do want is the following:
1) Leave this notice (i.e. this entire paragraph beginning with ”LATTICEEASY consists of...” and ending with our email addresses) in with the code wherever you put it. Even if you’re just using it in-house in your department, business, or wherever else we would like these credits to remain with it. This is partly so that people can...
2) Give us feedback. Did LATTICEEASY work great for you and help your work? Did you hate it? Did you find a way to improve it, or translate it into another programming language? Whatever the case might be, we would love to hear about it. Please let us know at the email address below.
3) Finally, insofar as we have the legal right to do so we forbid you to make money off of this code without our consent. In other words if you want to publish these functions in a book or bundle them into commercial software or anything like that contact us about it first. We’ll probably say yes, but we would like to reserve that right.

For any comments or questions you can reach us at
gfelder@email.smith.edu
Igor.Tkachev@cern.ch

である。曰く、「LATTICEEASY のコードを自由に書き換えて良いが、上の文章をコード中に記載しておくこと。また、書き換えたコードなどをお金儲けに使いたければ必ず連絡し許可を取れ。」だそうだ。

また 2.1.3 節で説明した FFT を実行するためのコード「ffteasy.cpp」にも同様に

> FFTEASY consists of the four C functions fftc1, fftcn, fftr1, and fftrn. FFTEASY is free. I am not in any way, shape, or form expecting to make money off of these routines. I wrote them because I needed them for some work I was doing and I'm putting them out on the Internet in case other people might find them useful. Feel free to download them, incorporate them into your code, modify them, translate the comment lines into Swahili, or whatever else you want. What I do want is the following:
> 1) Leave this notice (i.e. this entire paragraph beginning with "FFTEASY consists of..." and ending with my email address) in with the code wherever you put it. Even if you're just using it in-house in your department, business, or wherever else I would like these credits to remain with it. This is partly so that people can...
> 2) Give me feedback. Did FFTEASY work great for you and help your work? Did you hate it? Did you find a way to improve it, or translate it into another programming language? Whatever the case might be, I would love to hear about it. Please let me know at the email address below.
> 3) Finally, insofar as I have the legal right to do so I forbid you to make money off of this code without my consent. In other words if you want to publish these functions in a book or bundle them into commercial software or anything like that contact me about it first. I'll probably say yes, but I would like to reserve that right.
>
> For any comments or questions you can reach me at
> gfelder@email.smith.edu.

の文章をコード中に残しておく必要がある。

このように、LATTICEEASY あるいは私の書いたコードに対する利用条件はおおよそ「MIT ライセンス」のようなもので、上で記載した文章をコード中に入れておけば自由に利用できる（ただし金儲けのためには直接的な許可を得る必要あり）。ちなみに MIT ライセンスとは、マサチューセッツ工科大学で作成された代表的な寛容型オープンソースライセンスであり、以下の文章を記載すれば自由にコードを利用できるという、緩い利用条件である。

Copyright (c) <YEAR> <COPYRIGHTHOLER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the“Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ただし <YEAR> には著作権発生年を記入し、<COPYRIGHTHOLER> には著作権保持者名を記入する。

12.2　Courant 条件

本節ではプログラム上の値が、時間発展で指数関数的に増加してしまわないための条件である「Courant 条件」（あるいは Courant 安定条件）について説明する。これは単に、離散的な時間発展で時間間隔 Δt が大きすぎると正しく計算できないことを回避するためのもの。その条件は

$$\Delta t < \frac{dx}{\sqrt{\text{spatial dimension}}} \tag{12.1}$$

である。これを満たすように Δt を決定すべきであるが、理想は (12.1) を満たしつつ Δt を変えても（小さくしても）結果が変わらない程度の Δt の値を用いることである。

12.3　必要メモリ

本節では Lattice シミュレーションの実行に必要なコンピュータのメモリの概算を行う。最も大きい寄与は N^3 個の場の値を格納する配列である。

スカラー的な場 1 つに対して、場の値と場の微分の値で $2N^3$ 個の要素が必要であり、私のコードでは 4 次の Runge-Kutta 法を用いているので 4.4 節でも述べたように 4 倍の要素が必要となる。結局 1 つのスカラー的な場に対して $8N^3$ 個の要素が必要である。1 つの要素が占めるメモリは扱っている変数によって変わる。私のコードでは浮動小数点数に double 型（約 16 桁の精度）を用いていて、C++ 言語の double 型は 1 つ当たり 8 B（バイト）である。これを踏まえてスカラー的な場 1 つ当たりが占めるメモリは

$$\text{memory per scalar field} = 8 \times 8N^3 \text{ B} \tag{12.2}$$

例えば、単一場インフレーションを $N = 256$ で実行するのに必要なメモリは

$$8 \times 8 \times (256)^3 \text{ B} \approx 1 \text{ GB} \tag{12.3}$$

また、axion-U(1) インフレーションを $N = 256$ で、Lorentz ゲージで実行するのに必要なメモリは

$$8 \times 5 \times 8 \times (256)^3 \text{ B} \approx 5 \text{ GB} \tag{12.4}$$

である。よって $N = 256$ の Lattice シミュレーションを行うには、8 GB のメモリがあるコンピュータであれば可能である（はず）。

ここで「はず」とはぐらかしたのは、コンピュータのメモリ容量をフルに使えるかどうか怪しいからで、私のコードでは（現状は）場の値の要素数の 1/4 を global 変数として宣言し、残りの 3/4 を local 変数として「動的に（C++ 言語の用語）」確保している。C++ プログラムで使用されるコンピュータのメモリ領域は「テキスト領域」「静的領域」「ヒープ領域」「スタック領域」の 4 つに分けられ、global 変数の確保は静的領域・動的なメモリの確保はヒープ領域・local 変数の確保はスタック領域で行われる。これらの領域の中で静的領域とスタック領域は（おそらく）割り当てられる容量が決まっていて、ヒープ領域は（おそらく）上限いっぱいまで使える。これを踏まえて global 変数として宣言している 1/4 の場の要素数のメモリが、静的領域の容量を超えてしまうとコンピュータのメモリ不足 (segmentation fault) として実行できない可能性がある。解決するには場の値を全て local 変数として「動的に」確保すると良いかもしれない。

12.4　メモリリーク

C++ 言語では大容量の配列を確保するために、動的なメモリの確保を行う。私の Lattice シミュレーションコードでも、場の値を格納する配列のうち 3/4 をのメモリを動的に確保している。動的に確保したメモリは、プログラム上で手動で解放しなければならない（借りたメモリは返さないといけない）。この「動的に確保したメモリの解放」を行わないとそのメモリを再利用できなくなってしまう。このことを「メモリリーク」と呼ぶ。

さて、g++ で実行中のプログラムを停止するには「Ctrl+c」をする必要がある。Lattice シミュレーションの実行には基本的に時間がかかるため、途中で終了するにはこの作業をする必要がある。しかし、私のシミュレーションコードには（というか C++ プログラム全てかもしれないが）、「Ctrl+c」の強制終了コマンドでメモリを解放することができない。よって Lattice シミュレーションを実行する際は「Ctrl+c」の強制終了コマンドは極力使わないように注意すべきである。

ただし ChatGPT に聞いたところ「コンピュータを再起動すればリークしたメモリは初期化される」そうなので、「Ctrl+c」の強制終了コマンドを使用してしまった場合、コンピュータを再起動するべきである。

第 13 章

利用方法

本章では Lattice シミュレーションを実行し、結果のグラフを描画する具体的な方法・手順を説明する。その前にまず 13.1 節でプログラムを作動させるのに必要なアプリケーションを説明、13.2 節で Lattice シミュレーションを実行する方法を説明、13.3 節で得られた dat ファイルからグラフを描画する方法を説明する。

13.1　必要なアプリケーション

以下に Lattice シミュレーションの実行とグラフの描画に必要なアプリケーションを列挙しておく。

- コードエディタ
- C++ コンパイラ
- OpenMP（C++ に内包された API）
- Python 実行環境
- Matplotlib（Python に内包されたライブラリ）

私が使用していた各アプリケーションは「Visual Studio Code (VSCode)」「(homebrew でインストールした）g++」である。Python の実行は VSCode で行った。以下で各アプリケーションについて軽く説明する。

g++

C++ コンパイラとして用いた g++ は、様々なプログラミングコンパイラの集合である GNU Compiler Collection (gcc) の C++ 版であり、macOS における Command Line Interface (CLI) である「Terminal」にて実行した（私は）。具体的な実行方法は 13.2 節で説明する。

注意が必要なのは、macOS にデフォルトでインストールされている gcc は「clang」のもの（？）で古いバージョンらしく、この gcc で実行してもうまく動いてくれない場合がある。そこで「homebrew」という macOS で必須級のパッケージマネージャーから最新版の gcc をインストールする必要がある。ただし Terminal で gcc コマンドを使用しても clang の gcc が呼び出されるように（デフォルトでは）なっているはずなので、「シンボリックリンク」を作成し homebrew 経由の最新版 gcc が呼び出されるように設定を変更する必要がある。homebrew 経由の gcc のバイナリ（？）を確認するには terminal で

Code 13.1　gcc バイナリ確認

```
1 ls /usr/local/bin | grep gcc
2 ls /usr/local/bin | grep g++
```

とする。これで確認した gcc のバージョンに対してシンボリックリンクを作成するには

Code 13.2　シンボリックリンクの作成

```
1 ln -s /usr/local/bin/gcc-00 /usr/local/bin/gcc
2 ln -s /usr/local/bin/g++-00 /usr/local/bin/g++
3 exec $SHELL -l
```

とすれば、gcc コマンドで homebrew 経由のものが呼び出されるようになる。00 にはバージョンを表す数字を入れ、3 行目はシェルの再起動をするコマンドである。シンボリックリンクが適切に作成されているか確かめるには

Code 13.3　シンボリックリンクの確認

```
1 gcc --version
2 g++ --version
```

とすれば良い。また、作成したシンボリックリンクを削除するには

Code 13.4　シンボリックリンクの削除

```
1 unlink /usr/local/bin/gcc
2 unlink /usr/local/bin/g++
```

とすれば良い。ここで記載した gcc への path (/usr/local/bin) は一例。

OpenMP

OpenMP (Open Multi-Processing) は共有メモリ型マシンで並列計算を可能にする API であり、C 言語・C++ 言語・FORTRAN で利用可能である。コンパイラによる自動並列化とは異なり、コードを書くことで並列化したい部分を明示的に指定できる。OpenMP による並列化はスレッド並列化と呼ばれ、並列化する部分で複数のスレッドが同時に計算を行うことで処理速度を上げることができる。詳しくは [8, 9] の web サイト、あるいはコードレビューを参照。

Matplotlib

Matplotlib は Python でグラフを描画するためのライブラリで、もしかしたらデフォルトで Python に付いているわけではないかもしれないのでその場合はインストールが必要。グラフ描画に Python（ひいては Matplotlib）を用いたのは、私が学部時代に授業で習ったグラフ描画がその方法だったからで、グラフが美しく描けるからとか業界ではそれが一般的だからとかそのような理由では全くない。

13.2　Lattice シミュレーションの実行

本節では Lattice シミュレーションを実行する具体的な手順を説明する。私の使用したアプリケーションでの説明をするので、他のアプリケーションを使用する場合はよしなに読み替えてください。

①Lattice シミュレーションを行うディレクトリ「SingleInflation」「AxionInflation」（正確には「SFInflation」「Axion-U(1)」ディレクトリ）を保存した path

スタックサイズの確認・変更

12.3 節で local 変数がスタック領域（のメモリ）で確保されることを述べた。どのコンピュータも、使用できるスタック領域のデフォルトなメモリ容量は決まっていてそれほど大きくない。Terminal でスタック領域（あるいはその他の設定）の設定値を確認するには

Code 13.5　スタックサイズの確認

```
1 ulimit -s
2 ulimit -a
```

とすれば良い。前者がスタック領域のメモリ（kB 単位）を確認するコマンドで、後者が全体的な設定値を確認するコマンドである。私の macbook で確認したところ、デフォルトでは約 8MB であった。

Lattice シミュレーションで少し大きい配列など local 変数の確保でスタック領域のメモリが足りなくなる可能性がある。この時にスタック領域のメモリを手動で変更するには

Code 13.6　スタックサイズの変更

```
1 limit stacksize -----
```

とすれば良い。—-には kB 単位での欲しいメモリ容量を記入する。

実行ファイルが使用するスタック領域メモリの確認

ここでは 11.1 節で述べた「-fstack-usage」オプションを用いて、各実行ファイルに含まれる各関数が使用するスタック領域のメモリを確認する方法を説明する。このオプションをつけてコンパイルすると、実行ファイル（.o 拡張子）の他に stack usage ファイル（.su 拡張子）が作成される。このファイルの中身を見るには

Code 13.7　su ファイルの確認

```
1 cat ----.su
```

とすれば良い。—-にはファイル名を入れる。このようにして表示される情報に各関数が使用するスタック領域メモリが記載されている。この使用メモリがスタックサイズを超えると segmentation fault になる（セグフォになる要因は他にもあるが)。

13.3　グラフの描画

VSCode の Python でグラフを書く場合は、各 python コードに書いている dat ファイルへの path が正しいことを確認した後、VSCode の右上の Run ボタンをポチッとクリックするだけで良い（私の環境ではそれでできるようになっていた)。

付録

付録 A

計算過程の明記

A.1　第 I 部

・(1.7)

$$\begin{aligned}\ddot{a} &= \frac{d}{d\tau}\dot{a} \\ &= ma^s \frac{d}{d\tau_{\rm pr}}(ma^s a') \\ &= m^2 a^s (sa^{s-1}a'a' + a^s a'') \\ &= m^2 a^{2s}\left[a'' + s\frac{(a')^2}{a}\right]\end{aligned}$$

・(1.8)

$$\begin{aligned}\ddot{f} &= \frac{d}{d\tau}\dot{f} \\ &= ma^s \frac{d}{d\tau_{\rm pr}}(ma^s f') \\ &= m^2 a^s (sa^{s-1}a'f' + a^s f'') \\ &= m^2 a^{2s}\left[f'' + s\frac{a'}{a}f'\right]\end{aligned}$$

・(2.2) と (2.3) の DFT が逆変換であること

$$\begin{aligned} f(\boldsymbol{n}) &= \frac{1}{N^3}\sum_{\boldsymbol{m}}\left[\sum_{\tilde{\boldsymbol{n}}} f(\tilde{\boldsymbol{n}})e^{\frac{2\pi i}{N}\boldsymbol{m}\cdot\tilde{\boldsymbol{n}}}\right]e^{-\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} \\ &= \frac{1}{N^3}\sum_{\tilde{\boldsymbol{n}}} f(\tilde{\boldsymbol{n}})\sum_{\boldsymbol{m}}\exp\left[-\frac{2\pi i}{N}\boldsymbol{m}\cdot(\boldsymbol{n}-\tilde{\boldsymbol{n}})\right] \\ &= \frac{1}{N^3}\sum_{\tilde{\boldsymbol{n}}} f(\tilde{\boldsymbol{n}})N^3\delta_{\boldsymbol{n},\tilde{\boldsymbol{n}}} \\ &= f(\boldsymbol{n}) \end{aligned}$$

・(2.5)

$$\begin{aligned} f^*(\boldsymbol{\kappa}) &= \sum_{\boldsymbol{n}} f^*(\boldsymbol{n})e^{-\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} \\ &= \sum_{\boldsymbol{n}} f(\boldsymbol{n})e^{-\frac{2\pi i}{N}\boldsymbol{m}\cdot\boldsymbol{n}} \\ &= \sum_{\boldsymbol{n}} f(\boldsymbol{n})e^{\frac{2\pi i}{N}(-\boldsymbol{m})\cdot\boldsymbol{n}} \\ &= f(-\boldsymbol{\kappa}) \end{aligned}$$

ただし $\boldsymbol{\kappa} = \frac{2\pi}{L}\frac{N}{2}$ に対しては

$$f^*(\boldsymbol{\kappa}) = \sum_{\boldsymbol{n}} f(\boldsymbol{n})e^{-i\pi n_i} = \sum_{\boldsymbol{n}} f(\boldsymbol{n})e^{i\pi n_i} = \sum_{\boldsymbol{n}} f(\boldsymbol{n})$$

で仮想的な $m_i = -\frac{N}{2}$ が対応する。もっと言うとフーリエモードのうち $\boldsymbol{\kappa} = (0,0,0)$, $(0,0,\frac{N}{2}), (0,\frac{N}{2},0), (\frac{N}{2},0,0), (0,\frac{N}{2},\frac{N}{2}), (\frac{N}{2},0,\frac{N}{2}), (\frac{N}{2},\frac{N}{2},0), (\frac{N}{2},\frac{N}{2},\frac{N}{2})$ の 8 つのみが実数で $f^*(\boldsymbol{\kappa}) = f(\boldsymbol{\kappa})$ となる。

・(3.8) の $m_{\rm eff}^2$ ポテンシャル項を残した slow-roll 第 0 近似かつ sub-horizon 極限でのインフラトン運動方程式は

$$\partial_\tau^2 f_k + \left[k^2 + a^2 \frac{d^2V}{d\phi^2}\right] f_k = 0 \qquad (f = a\delta\phi)$$

よって、より sub-horizon の近似を用いた解

$$\delta\phi = \frac{1}{a\sqrt{2k}} e^{-k\tau}$$

の一般化としては

$$\delta\phi = \frac{1}{a\sqrt{2\sqrt{k^2+m_{\rm eff}^2}}} e^{-\sqrt{k^2+m_{\rm eff}^2}\tau} \quad , \quad m_{\rm eff}^2 = a^2\frac{d^2V}{d\phi^2}$$

・(3.9) の分散が $\mathcal{P}_{\delta\phi}$ になること

$$\begin{aligned}
\sigma^2 &= \int_0^\infty dX X^2 \left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} X \exp\left[-\frac{1}{2}\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} X^2\right] \\
&= \left[-X^2 \exp\left[-\frac{1}{2}\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} X^2\right]\right]_0^\infty + 2\int_0^\infty dX X \exp\left[-\frac{1}{2}\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} X^2\right] \\
&= 2\left[-\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)\exp\left[-\frac{1}{2}\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} X^2\right]\right]_0^\infty \\
&= \mathcal{P}_{\delta\phi}
\end{aligned}$$

・(3.10) の複素ガウス分布で $|\delta\phi(k)|$ が (3.9) の Rayleigh 分布に従うこと

$$|\delta\phi(k)|^2 = (\mathrm{Re}[\delta\phi(k)])^2 + (\mathrm{Im}[\delta\phi(k)])^2 \quad , \quad \frac{\mathrm{Im}[\delta\phi(k)]}{\mathrm{Re}[\delta\phi(k)]} = \tan\theta$$

で極座標のように変換すると

$$\begin{aligned}
&\int_{-\infty}^\infty d\mathrm{Re}[\delta\phi(k)] \int_{-\infty}^\infty d\mathrm{Im}[\delta\phi(k)] \frac{1}{\pi\mathcal{P}_{\delta\phi}} \exp\left[-\frac{|\delta\phi(k)|^2}{\mathcal{P}_{\delta\phi}}\right] \\
&\qquad = \int_0^\infty d|\delta\phi(k)| \int_0^{2\pi} d\theta |\delta\phi(k)| \frac{1}{\pi\mathcal{P}_{\delta\phi}} \exp\left[-\frac{|\delta\phi(k)|^2}{\mathcal{P}_{\delta\phi}}\right] \\
&\qquad = \int_0^\infty d|\delta\phi(k)| \left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} |\delta\phi(k)| \exp\left[-\frac{1}{2}\left(\frac{\mathcal{P}_{\delta\phi}}{2}\right)^{-1} |\delta\phi(k)|^2\right]
\end{aligned}$$

・ガウス分布の 3 次のキュムラント

$$\begin{aligned}\langle (X-\overline{X})^3 \rangle &= \int_{-\infty}^{\infty} dX (X-\overline{X})^3 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(X-\overline{X})^2}{2\sigma^2}} \\ &= \int_{-\infty}^{\infty} dx x^3 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\end{aligned}$$

ここで

$$\frac{d}{dx}\left[e^{-\frac{x^2}{2\sigma^2}}\right] = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

より

$$\begin{aligned}\langle (X-\overline{X})^3 \rangle &= \left[\frac{x^2}{\sqrt{2\pi}\sigma}(-\sigma^2) e^{-\frac{x^2}{2\sigma^2}}\right]_{-\infty}^{\infty} + \frac{\sigma}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dx 2x e^{-\frac{x^2}{2\sigma^2}} \\ &= \frac{2\sigma}{\sqrt{2\pi}} \left[-\sigma^2 e^{-\frac{x^2}{2\sigma^2}}\right]_{-\infty}^{\infty} \\ &= 0\end{aligned}$$

・ガウス分布の 4 次のキュムラント

$$\begin{aligned}\langle (X-\overline{X})^4 \rangle &= \int_{-\infty}^{\infty} dX (X-\overline{X})^4 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(X-\overline{X})^2}{2\sigma^2}} \\ &= \int_{-\infty}^{\infty} dx x^4 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \\ &= \left[\frac{x^3}{\sqrt{2\pi}\sigma}(-\sigma^2) e^{-\frac{x^2}{2\sigma^2}}\right]_{-\infty}^{\infty} + \frac{\sigma}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dx 3x^2 e^{-\frac{x^2}{2\sigma^2}} \\ &= 3\sigma^2 \int_{-\infty}^{\infty} dx x^2 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \\ &= 3\sigma^4\end{aligned}$$

A.2　第 II 部

・(6.9)
ポテンシャル項を残した slow-roll 第 0 近似（つまり曲率ゆらぎのみを無視）でのインフラトン運動方程式は

$$\partial_\tau^2 f_k + \left[k^2 - \frac{\partial_\tau^2 a}{a} + a^2 \frac{d^2V}{d\phi^2}\right] f_k = 0 \qquad (f = a\delta\phi)$$

ここで de-Sitter 宇宙の関係式を用いて

$$\begin{aligned}\frac{\partial_\tau^2 a}{a} &= \frac{1}{a}\frac{\partial}{\partial\tau}\left[a\frac{da}{dt}\right] = \frac{d}{dt}(a^2 H) \\ &\approx H\frac{d}{dt}a^2 = 2a^2H^2\end{aligned}$$

と計算され、解

$$f_k(\tau) = \frac{\sqrt{-\pi\tau}}{2} H_\nu^{(1)}(-k\tau)$$

に対する ν は

$$\partial_\tau^2 f_k + \left[k^2 - \frac{\nu^2 - \frac{1}{4}}{\tau^2}\right] f_k = 0$$

と比較して

$$\begin{aligned}\nu^2 &= \frac{1}{4} + \tau^2\left[2a^2H^2 - a^2\frac{d^2V}{d\phi^2}\right] \\ &= \frac{9}{4} - \frac{1}{H^2}\frac{d^2V}{d\phi^2}\end{aligned}$$

・(6.12)
Bessel 関数と Neumann 関数の満たす微分漸化式

$$\frac{dg_\nu(x)}{dx} = g_{\nu-1}(x) - \frac{\nu}{x}g_\nu(x), \qquad g = J, Y, H$$

を用いてモード関数（両方向の波それぞれ）の微分は

$$
\begin{aligned}
\partial_\tau \delta\phi(k) &= \partial_\tau \left[\frac{\sqrt{-\pi\tau}}{2a} H_\nu^{(1,2)}(-k\tau) \right] \\
&= \partial_\tau \frac{\sqrt{-\pi\tau}}{2a} \sqrt{J_\nu^2(-k\tau) + Y_\nu^2(-k\tau)} \exp\left[\pm i \arctan\left(\frac{Y_\nu(-k\tau)}{J_\nu(-k\tau)} \right) \right] \\
&= \left[\frac{-1}{2(-\tau)} - \frac{\partial_\tau a}{a} \right] \frac{\sqrt{-\pi\tau}}{2a} \sqrt{J_\nu^2 + Y_\nu^2} e^{\pm i\theta} \\
&\quad + \frac{\sqrt{-\pi\tau}}{2a} (-k) \frac{\partial}{\partial x} \sqrt{J_\nu^2(x) + Y_\nu^2(x)} \exp\left[\pm i \arctan\left(\frac{Y_\nu(x)}{J_\nu(x)} \right) \right] \\
&= \left[\frac{1}{2\tau} - \mathcal{H} \right] \frac{\sqrt{-\pi\tau}}{2a} \sqrt{J_\nu^2 + Y_\nu^2} e^{\pm i\theta} \\
&\quad - k \frac{\sqrt{-\pi\tau}}{2a} \left[\frac{J_\nu \partial_x J_\nu + Y_\nu \partial_x Y_\nu}{J_\nu^2 + Y_\nu^2} \pm i \frac{J_\nu \partial_x Y_\nu - Y_\nu \partial_x J_\nu}{\left(1 + \frac{Y_\nu^2}{J_\nu^2}\right) J_\nu^2} \right] \sqrt{J_\nu^2 + Y_\nu^2} e^{\pm i\theta} \\
&= \left[\frac{1}{2\tau} - \mathcal{H} - k \frac{J_\nu \left(J_{\nu-1} - \frac{\nu}{-k\tau} J_\nu \right) + Y_\nu \left(Y_{\nu-1} - \frac{\nu}{-k\tau} Y_\nu \right)}{J_\nu^2 + Y_\nu^2} \right] \delta\phi(k) \\
&\quad \mp ik \frac{J_\nu Y_{\nu-1} - J_{\nu-1} Y_\nu}{J_\nu^2 + Y_\nu^2} \delta\phi(k) \\
&= \left[\frac{1}{2\tau} - \mathcal{H} - \frac{\nu}{\tau} - k \frac{J_\nu J_{\nu-1} + Y_\nu Y_{\nu-1}}{J_\nu^2 + Y_\nu^2} \pm ik \frac{J_{\nu-1} Y_\nu + J_\nu Y_{\nu-1}}{J_\nu^2 + Y_\nu^2} \right] \delta\phi(k)
\end{aligned}
$$

初期条件と Lattice 上の有効運動量を考慮すると

$$
\begin{aligned}
\partial_\tau \delta\phi(k) &= \left[-\frac{H_{\rm ini}}{2} - H_{\rm ini} + \nu H_{\rm ini} - k_{\rm eff} \frac{J_\nu J_{\nu-1} + Y_\nu Y_{\nu-1}}{|H_\nu|^2} \right. \\
&\quad \left. \pm ik_{\rm eff} \frac{J_{\nu-1} Y_\nu + J_\nu Y_{\nu-1}}{|H_\nu|^2} \right] \delta\phi(k) \\
&= \left[\left(\nu - \frac{3}{2} \right) H_{\rm ini} - k_{\rm eff} \frac{J_\nu J_{\nu-1} + Y_\nu Y_{\nu-1}}{|H_\nu|^2} \pm ik_{\rm eff} \frac{J_{\nu-1} Y_\nu + J_\nu Y_{\nu-1}}{|H_\nu|^2} \right] \delta\phi(k)
\end{aligned}
$$

リスケーリングと等方的初期条件をを考慮して最終的に

$$
\begin{aligned}
\delta\phi'(\boldsymbol{\kappa}) &= \left[\left(\nu - \frac{3}{2} \right) a'_{\rm ini} - k_{\rm eff,pr} \frac{J_\nu J_{\nu-1} + Y_\nu Y_{\nu-1}}{|H_\nu|^2} \right] \delta\phi(\boldsymbol{\kappa}) \\
&\quad + ik_{\rm eff,pr} \frac{J_{\nu-1} Y_\nu + J_\nu Y_{\nu-1}}{|H_\nu|^2} \frac{|\delta\phi|_{\rm pr}}{\sqrt{2}} \left[\sqrt{-\ln X_1} e^{i2\pi Y_1} - \sqrt{-\ln X_2} e^{i2\pi Y_2} \right]
\end{aligned}
$$

・(6.13)

$$
\begin{aligned}
0 &= \partial_\mu \left[-\sqrt{-g} g^{\alpha\beta} (\partial_\alpha \phi) \delta^\mu_\beta \right] - \sqrt{-g} \left(-\frac{dV}{d\phi} \right) \\
&= \partial_\mu \left[-a^4 g^{\alpha\mu} (\partial_\alpha \phi) \right] + a^4 \frac{dV}{d\phi} \\
&= \partial_\tau \left[a^2 (\partial_\tau \phi) \right] - a^2 \partial_i^2 \phi + a^4 \frac{dV}{d\phi} \\
&= a^2 \partial_\tau^2 \phi + 2a^2 \mathcal{H} \partial_\tau \phi - a^2 \nabla^2 \phi + a^4 \frac{dV}{d\phi} \\
&= a^2 \left[\partial_\tau^2 \phi + 2\mathcal{H} \partial_\tau \phi - \nabla^2 \phi + a^2 \frac{dV}{d\phi} \right]
\end{aligned}
$$

・(6.14)

リスケーリング (1.5)(1.7)(1.8)(1.9)(1.11) を用いて

$$
\begin{aligned}
&\partial_\tau^2 \phi + 2\mathcal{H} \partial_\tau \phi - \nabla^2 \phi + a^2 \frac{dV}{d\phi} \\
&\quad = m^2 a^{2s} \left[\phi'' + s \frac{a'}{a} \phi' \right] + 2 \frac{m a^s a'}{a} m a^s \phi' - m^2 \nabla^2_{\rm pr} \phi + a^2 \frac{d}{d\phi} (m^2 a^{2s} V_{\rm pr}) \\
&\quad = m^2 a^{2s} \left[\phi'' + s \frac{a'}{a} \phi' + 2 \frac{a'}{a} \phi' - \frac{\nabla^2_{\rm pr} \phi}{a^{2s}} + a^2 \frac{dV_{\rm pr}}{d\phi} \right] \\
&\quad = m^2 a^{2s} \left[\phi'' + (s+2) \frac{a'}{a} \phi' - \frac{\nabla^2_{\rm pr} \phi}{a^{2s}} + a^2 \frac{dV_{\rm pr}}{d\phi} \right]
\end{aligned}
$$

・(6.16)

リスケーリング (1.7)(1.10) を用いて

$$
\begin{aligned}
&\partial_\tau^2 a - \frac{a^3}{6} (p - 3p) \\
&\quad = m^2 a^{2s} \left[a'' + s \frac{(a')^2}{a} \right] - \frac{a^3}{6} m^2 a^{2s} \left\langle \rho_{\rm pr} - 3 p_{\rm pr} \right\rangle \\
&\quad = m^2 a^{2s} \left[a'' + s \frac{(a')^2}{a} - \frac{a^3}{6} \left\langle \rho_{\rm pr} - 3 p_{\rm pr} \right\rangle \right]
\end{aligned}
$$

・(6.17)(6.18)
エネルギー運動量テンソルは

$$T_{\mu\nu} = (\partial_\mu\phi)(\partial_\nu\phi) - g_{\mu\nu}\left[\frac{1}{2}g^{\alpha\beta}(\partial_\alpha\phi)(\partial_\beta\phi) + V(\phi)\right]$$

より

$$\begin{aligned}\rho = -T^0{}_0 &= -g^{0\mu}(\partial_\mu\phi)(\partial_0\phi) + \frac{1}{2}g^{\alpha\beta}(\partial_\alpha\phi)(\partial_\beta\phi) + V(\phi)\\ &= a^{-2}(\partial_\tau\phi)^2 - \frac{1}{2}a^{-2}(\partial_\tau\phi)^2 + \frac{1}{2}a^{-2}(\nabla\phi)^2 + V(\phi)\\ &= \frac{(\partial_\tau\phi)^2}{2a^2} + \frac{(\nabla\phi)^2}{2a^2} + V(\phi)\end{aligned}$$

リスケーリング (1.8) から (1.11) を用いて

$$\begin{aligned}\rho_{\rm pr} &= \frac{1}{m^2a^{2s}}\left[\frac{(ma^s\phi')^2}{2a^2} + m^2\frac{(\nabla_{\rm pr}\phi)^2}{2a^2} + m^2a^{2s}V_{\rm pr}\right]\\ &= \frac{(\phi')^2}{2a^2} + \frac{(\nabla_{\rm pr}\phi)^2}{2a^{2s+2}} + V_{\rm pr}\\ &= \frac{(\phi')^2}{2a^2} - \frac{\phi\nabla^2_{\rm pr}\phi}{2a^{2s+2}} + V_{\rm pr}\end{aligned}$$

ただし、最後の「=」で部分積分的な式 $(\nabla_{\rm pr}\phi)^2 = -\phi\nabla^2_{\rm pr}\phi$ が成立することを仮定した。
また

$$\begin{aligned}p = \frac{1}{3}T^i{}_i &= \frac{1}{3}g^{i\mu}(\partial_\mu\phi)(\partial_i\phi) - \left[\frac{1}{2}g^{\alpha\beta}(\partial_\alpha\phi)(\partial_\beta\phi) + V(\phi)\right]\\ &= \frac{1}{3}a^{-2}(\nabla\phi)^2 + \frac{(\partial_\tau\phi)^2}{2a^2} - \frac{(\nabla\phi)^2}{2a^2} - V(\phi)\\ &= \frac{(\partial_\tau\phi)^2}{2a^2} - \frac{(\nabla\phi)^2}{6a^2} - V(\phi)\end{aligned}$$

リスケーリングを考慮して

$$p_{\rm pr} = \frac{(\phi')^2}{2a^2} + \frac{\phi\nabla^2_{\rm pr}\phi}{6a^{2s+2}} - V_{\rm pr}$$

・(6.22)

Friedmann 方程式の Consistency Condition がよく成立していると仮定して

$$\begin{aligned}\epsilon &= -\frac{\partial_t H}{H^2} = -\frac{3}{\langle\rho\rangle}\partial_t\sqrt{\frac{\langle\rho\rangle}{3}} \\ &= -\frac{\sqrt{3}}{\langle\rho\rangle}\frac{\partial_t\langle\rho\rangle}{2\sqrt{\langle\rho\rangle}} = -\frac{\sqrt{3}}{\langle\rho\rangle}\frac{\langle\partial_t\rho\rangle}{2\sqrt{\langle\rho\rangle}} \\ &= -\frac{\sqrt{3}}{\langle\rho\rangle}\frac{\langle -3H(\rho+p)\rangle}{2\sqrt{\langle\rho\rangle}} = \frac{3}{2}\sqrt{3}\frac{H}{\sqrt{\langle\rho\rangle}}\frac{\langle\rho+p\rangle}{\langle\rho\rangle} \\ &= \frac{3}{2}\frac{\langle\rho\rangle+\langle p\rangle}{\langle\rho\rangle} = \frac{3}{2}\left(1+\frac{\langle p\rangle}{\langle\rho\rangle}\right)\end{aligned}$$

ただし 2 行目 2 つ目の「=」で格子平均と時間微分が可換であることを仮定した（これを仮定しないとプログラムの処理が少し面倒くさくなるので仕方なく仮定した)。また 3 行目 1 つ目の「=」で保存則 $\partial_t\rho + 3H(\rho+p) = 0$ が成立していることを仮定した。保存則成立の仮定は Einstein 方程式の 2 本（スケール因子の発展方程式と、Friedmann 方程式の Consistency Condition）が十分良い精度で成立しているという仮定から保証されていると思っている。

・(6.23)

$$\frac{d}{dN_e} = \frac{d}{d\ln a} = \frac{d}{\frac{da}{a}} = a\frac{dt}{da}\frac{d}{dt} = \frac{1}{H}\frac{d}{dt}$$

と η の定義より

$$\begin{aligned}\eta &= \frac{1}{\epsilon}\frac{d\epsilon}{dN_e} = \frac{\partial_t\epsilon}{H\epsilon} = \frac{1}{H\epsilon}\frac{3}{2}\partial_t\frac{\langle p\rangle}{\langle\rho\rangle} = \frac{1}{H\epsilon}\frac{3}{2}\partial_t\left(\frac{\langle p\rangle}{3H^2}\right) \\ &= \frac{1}{2H\epsilon}\left[\frac{\partial_t\langle p\rangle}{H^2} - 2\frac{\partial_t H}{H^2}\frac{\langle p\rangle}{H}\right] = \frac{\partial_t\langle p\rangle}{2H^3\epsilon} + \frac{\langle p\rangle}{H^2} \\ &= \frac{\langle\partial_t p\rangle}{2H^3\epsilon} + 3\frac{\langle p\rangle}{\langle\rho\rangle} = \frac{\langle\partial_t p\rangle}{2H^3\epsilon} + (2\epsilon - 3)\end{aligned}$$

とできる。思想としては、Consistency Condition などを用いて ρ, p の時間微分を極力回避することを心がけた。しかしもっと簡単な計算過程がありそう。圧力 p の時間微分は

$$\begin{aligned}\partial_t p &= \frac{1}{a}\frac{d}{d\tau}\left[\frac{(\partial_\tau\phi)^2}{2a^2} - \frac{(\nabla\phi)^2}{6a^2} - V(\phi)\right] \\ &= \frac{1}{a}\left[\frac{\partial_\tau\phi}{a^2}\partial_\tau^2\phi - \mathcal{H}\frac{(\partial_\tau\phi)^2}{a^2} - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^2} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^2} - \frac{dV}{d\phi}\partial_\tau\phi\right]\end{aligned}$$

ここでインフラトン場の 2 階微分 $\partial_\tau^2\phi$ は発展方程式 (6.13) から

$$\begin{aligned}\partial_t p &= \frac{\partial_\tau\phi}{a^3}\left(-2\mathcal{H}\partial_\tau\phi + \nabla^2\phi - a^2\frac{dV}{d\phi}\right) - \mathcal{H}\frac{(\partial_\tau\phi)^2}{a^3} \\ &\quad - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^3} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^3} - \frac{1}{a}\frac{dV}{d\phi}\partial_\tau\phi \\ &= -3\frac{\mathcal{H}}{a}\frac{(\partial_\tau\phi)^2}{a^2} - 2\frac{dV}{d\phi}\frac{\partial_\tau\phi}{a} \\ &\quad + \frac{\partial_\tau\phi\nabla^2\phi}{a^3} - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^3} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^3}\end{aligned}$$

1 行目が支配的な寄与をし、2 行目は空間微分が含まれているため小さな寄与をする。

1 行目は宇宙時間 t で表記すると多少わかりやすく

$$\partial_t p_{\rm 1st} = -3H(\partial_t\phi)^2 - 2\frac{dV}{d\phi}\partial_t\phi$$

この支配的な寄与のみを考えた時の slow-roll パラメータ η は

$$\eta_{\rm 1st} = \frac{1}{2H^3\epsilon}\left\langle -3H(\partial_t\phi)^2 - 2\frac{dV}{d\phi}\partial_t\phi\right\rangle + 2\epsilon - 3$$

支配的な寄与のみを考えた時（空間微分項を除いた時)、slow-roll パラメータ ϵ が

$$\epsilon_{\rm 1st} = \frac{3\left\langle\rho+p\right\rangle}{2\left\langle\rho\right\rangle} = \frac{\left\langle\rho+p\right\rangle}{2H^2} = \frac{\left\langle\frac{(\partial_\tau\phi)^2}{a^2}\right\rangle}{2H^2} = \frac{\left\langle(\partial_t\phi)^2\right\rangle}{2H^2}$$

と表されるので

$$\begin{aligned}\eta_{\rm 1st} &= \frac{1}{H\left\langle(\partial_t\phi)^2\right\rangle}\left\langle -3H(\partial_t\phi)^2 - 2\frac{dV}{d\phi}\partial_t\phi\right\rangle + 2\epsilon - 3 \\ &= -3 - 2\frac{1}{H\left\langle(\partial_t\phi)^2\right\rangle}\left\langle\frac{dV}{d\phi}\partial_t\phi\right\rangle + 2\epsilon - 3 \\ &= -6 + 2\epsilon - 2\frac{1}{H}\frac{\left\langle\frac{dV}{d\phi}\partial_t\phi\right\rangle}{\left\langle(\partial_t\phi)^2\right\rangle}\end{aligned}$$

で解析的な結果を再現 (?) する。プログラム上の値で書き直すと

$$\eta_{\rm 1st} = -6 + 2\epsilon - 2\frac{a^3}{a'}\frac{\left\langle\frac{dV_{\rm pr}}{d\phi}\phi'\right\rangle}{\left\langle(\phi')^2\right\rangle}$$

支配的でない空間微分項も考慮すると

$$\begin{aligned}\eta &= 2\epsilon - 3 + \frac{1}{2H^3\epsilon}\left[-3\frac{\mathcal{H}}{a}\frac{(\partial_\tau\phi)^2}{a^2} - 2\frac{dV}{d\phi}\frac{\partial_\tau\phi}{a}\right.\\ &\qquad\left. + \frac{\partial_\tau\phi\nabla^2\phi}{a^3} - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^3} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^3}\right]\\ &= 2\epsilon - 3 + \frac{1}{H\langle\rho+p\rangle}\left[-3\frac{\mathcal{H}}{a}\frac{(\partial_\tau\phi)^2}{a^2} - 2\frac{dV}{d\phi}\frac{\partial_\tau\phi}{a}\right.\\ &\qquad\left. + \frac{\partial_\tau\phi\nabla^2\phi}{a^3} - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^3} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^3}\right]\\ &= \left[-3 - 3\frac{\langle(\partial_\tau\phi)^2\rangle}{a^2\langle\rho+p\rangle}\right] + 2\epsilon - 2\frac{1}{H\langle\rho+p\rangle}\langle\frac{dV}{d\phi}\frac{\partial_\tau\phi}{a}\rangle\\ &\qquad + \frac{1}{H\langle\rho+p\rangle}\left[\frac{\partial_\tau\phi\nabla^2\phi}{a^3} - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^3} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^3}\right]\end{aligned}$$

ここで 1 行目が先ほどの支配的な寄与に対応する（上の式で格子平均の $\langle\cdots\rangle$ を忘れているところが沢山あるのでよしなに、、、）。エネルギー密度の計算で仮定した部分積分的な式を空間微分項にも適用して、2 行目は

$$\begin{aligned}&\frac{\partial_\tau\phi\nabla^2\phi}{a^3} - \frac{(\nabla\phi)(\nabla\partial_\tau\phi)}{3a^3} + \mathcal{H}\frac{(\nabla\phi)^2}{3a^3}\\ &\qquad = \frac{\partial_\tau\phi\nabla^2\phi}{a^3} + \frac{\nabla^2\phi\partial_\tau\phi}{3a^3} - \mathcal{H}\frac{\phi\nabla^2\phi}{3a^3}\\ &\qquad = -H\frac{\phi\nabla^2\phi}{3a^2} + \frac{4(\partial_\tau\phi)(\nabla^2\phi)}{3a^3}\end{aligned}$$

であるので最終的に

$$\begin{aligned}\eta = &\left[-3 - 3\frac{\langle(\partial_\tau\phi)^2\rangle}{a^2\langle\rho+p\rangle}\right] + 2\epsilon - 2\frac{1}{H\langle\rho+p\rangle}\langle\frac{dV}{d\phi}\frac{\partial_\tau\phi}{a}\rangle\\ &- \frac{\langle\phi\nabla^2\phi\rangle}{3a^2\langle\rho+p\rangle} + \frac{4\langle(\partial_\tau\phi)(\nabla^2\phi)\rangle}{3a^3H\langle\rho+p\rangle}\end{aligned}$$

プログラム上の値で書き直すと

$$\begin{aligned}\eta = &\left[-3 - 3\frac{\langle(\phi')^2\rangle}{a^2\langle\rho_{\rm pr}+p_{\rm pr}\rangle}\right] + 2\epsilon - 2\frac{a}{a'}\frac{1}{\langle\rho_{\rm pr}+p_{\rm pr}\rangle}\langle\frac{dV_{\rm pr}}{d\phi}\phi'\rangle\\ &- \frac{\langle\phi\nabla^2_{\rm pr}\phi\rangle}{3a^{2s+2}\langle\rho_{\rm pr}+p_{\rm pr}\rangle} + \frac{4}{3a^{2s+1}a'}\frac{\langle\phi'\nabla^2_{\rm pr}\phi\rangle}{\langle\rho_{\rm pr}+p_{\rm pr}\rangle}\end{aligned}$$

ここで分母に来ているエネルギーと圧力の和は

$$\langle\rho_{\rm pr}+p_{\rm pr}\rangle = \langle\frac{(\phi')^2}{a^2} - \frac{\phi\nabla^2\phi}{3a^{2s+2}}\rangle$$

・(6.27)

1 階微分は

$$\begin{aligned}\frac{dV_{\rm pr}}{d\phi} &= \frac{1}{2a^{2s}}\frac{d}{d\phi}\phi^2\left[1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right]\\ &= \frac{\phi}{a^{2s}}\left[1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right]+\frac{\phi^2}{2a^{2s}}\frac{s}{d}\frac{1}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\\ &= \frac{\phi}{a^{2s}}\left[1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right]+\frac{\phi}{2a^{2s}}\frac{\phi}{d}\frac{s}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\end{aligned}$$

2 階微分は

$$\frac{d^2V_{\rm pr}}{d\phi^2} = \frac{1}{a^{2s}}\frac{d}{d\phi}\left[\phi\left\{1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right\}+\frac{\phi^2}{2d}\frac{s}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\right]$$

先にスケール因子の初期条件 (3.1) を使って

$$\begin{aligned}\frac{d^2V_{\rm pr}}{d\phi^2} &= \frac{d}{d\phi}\left[\phi\left\{1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right\}+\frac{\phi^2}{2d}\frac{s}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\right]\\ &= 1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)+\phi\frac{s}{d}\frac{1}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\\ &\quad+\frac{\phi}{d}\frac{s}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}+\frac{\phi^2}{2d}(-2s)\frac{1}{d}\frac{\sinh\left(\frac{\phi-\phi_{\rm step}}{d}\right)}{\cosh^3\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\\ &= 1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)+2\frac{s\frac{\phi}{d}}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\\ &\quad-\frac{\phi}{d}\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\frac{s\frac{\phi}{d}}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\\ &= 1+s\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)+\frac{s\frac{\phi}{d}}{\cosh^2\left(\frac{\phi-\phi_{\rm step}}{d}\right)}\left[2-\frac{\phi}{d}\tanh\left(\frac{\phi-\phi_{\rm step}}{d}\right)\right]\end{aligned}$$

A.3　第 III 部

・(9.21)

$$\partial_\tau A(k) = \partial_\tau \left[\frac{1}{\sqrt{2k}} e^{\pm ik\tau} \right] = \frac{\pm ik}{\sqrt{2k}} e^{\pm ik\tau} = \pm ikA(k)$$

ただし指数関数の肩の ± は円偏光を表しているのではなく、等方的初期条件を考慮する際の正負方向に進む波を表す。リスケーリングを考慮すると、時間微分のリスケーリング m と運動量のリスケーリング m が相殺して

$$A'(\boldsymbol{\kappa}) = -ik_{\mathrm{eff,pr}} \frac{|A|_{\mathrm{pr}}}{\sqrt{2}} (\sqrt{-\ln X_1} e^{i2\pi Y_1} - \sqrt{-\ln X_2} e^{i2\pi Y_2})$$

最初の「-」は 1 項目に対応するのが $e^{-ik\tau}$ の波だからであるが、ガウス分布で作っているので（正負の対称性があるはずなので）無くても良い。

・(9.24)

Coulomb 波動関数の満たす微分漸化式と関係式は

$$\frac{dg_0(\xi,x)}{dx} = \left(\frac{1}{x} + \xi \right) g_0(\xi,x) - \sqrt{1+\xi^2} g_1(\xi,x), \qquad g = F, G$$

$$G_l(\xi,x) \frac{d}{dx} F_l(\xi,x) - F_l(\xi,x) \frac{d}{dx} G_l(\xi,x) = 1$$

である。モード関数（両方向の波それぞれ ± と、各偏光 λ）の微分は

$$\begin{aligned}
\partial_\tau A_\lambda(k) &= \partial_\tau \left[\frac{1}{\sqrt{2k}} [G_0(-\lambda\xi, -k\tau) \pm iF_0(-\lambda\xi, -k\tau)] \right] \\
&= \frac{1}{\sqrt{2k}} \partial_\tau \sqrt{G_0^2(-\lambda\xi,-k\tau) + F_0^2(-\lambda\xi,-k\tau)} \exp\left[\pm i \arctan\left(\frac{F_0(-\lambda\xi,-k\tau)}{G_0(-\lambda\xi,-k\tau)} \right) \right] \\
&= \frac{1}{\sqrt{2k}} (-k) \partial_\chi \sqrt{G_0^2(-\lambda\xi,\chi) + F_0^2(-\lambda\xi,\chi)} \exp\left[\pm i \arctan\left(\frac{F_0(-\lambda\xi,\chi)}{G_0(-\lambda\xi,\chi)} \right) \right] \\
&= \frac{-k}{\sqrt{2k}} \frac{G_0\partial_\chi G_0 + F_0 \partial_\chi F_0}{\sqrt{G_0^2+F_0^2}} \exp\left[\pm i \arctan\left(\frac{F_0}{G_0} \right) \right] \\
&\quad + \frac{-k}{\sqrt{2k}} \sqrt{G_0^2+F_0^2} \frac{\pm i}{1+\frac{F_0^2}{G_0^2}} \frac{G_0\partial_\chi F_0 - F_0\partial_\chi G_0}{G_0^2} \exp\left[\pm i \arctan\left(\frac{F_0}{G_0} \right) \right]
\end{aligned}$$

ただし Coulomb 波動関数の引数は冗長なので省いたのでよしなに。A_λ で表せば

$$\partial_\tau A_\lambda(k) = -k\frac{G_0\partial_\chi G_0 + F_0\partial_\chi F_0}{G_0^2+F_0^2}A_\lambda(k) \mp ik\frac{G_0\partial_\chi F_0 - F_0\partial_\chi G_0}{G_0^2+F_0^2}A_\lambda(k)$$

微分漸化式を用いると 1 項目は

$$\begin{aligned}
&G_0\partial_\chi G_0 + F_0\partial_\chi F_0 \\
&\quad = G_0\left[\left(\frac{1}{\chi}-\lambda\xi\right)G_0 - \sqrt{1+\xi^2}G_1\right] + F_0\left[\left(\frac{1}{\chi}-\lambda\xi\right)F_0 - \sqrt{1+\xi^2}F_1\right] \\
&\quad = \left(\frac{1}{\chi}-\lambda\xi\right)[G_0^2+F_0^2] - \sqrt{1+\xi^2}[G_0G_1+F_0F_1] \\
&\quad = \left(\frac{1}{-k\tau}-\lambda\xi\right)[G_0^2+F_0^2] - \sqrt{1+\xi^2}[G_0G_1+F_0F_1]
\end{aligned}$$

また、関係式を用いると 2 項目は 1 なので

$$\begin{aligned}
\partial_\tau A_\lambda(k) = -k\left[\left(\frac{1}{-k\tau}-\lambda\xi\right) - \sqrt{1+\xi^2}\frac{G_0G_1+F_0F_1}{G_0^1+F_0^2}\right]A_\lambda(k) \\
\mp ik\frac{1}{G_0^2+F_0^2}A_\lambda(k)
\end{aligned}$$

初期条件と Lattice 上の有効運動量を考慮すると

$$\begin{aligned}
\partial_\tau A_\lambda(k_{\rm eff}) = k_{\rm eff}\sqrt{1+\xi_{\rm ini}^2}\frac{G_0G_1+F_0F_1}{G_0^1+F_0^2}A_\lambda(k_{\rm eff}) - H_{\rm ini}A_\lambda(k_{\rm eff}) \\
+\lambda k_{\rm eff}\xi_{\rm ini}A_\lambda(k_{\rm eff}) \mp ik_{\rm eff}\frac{1}{G_0^2+F_0^2}A_\lambda(k_{\rm eff})
\end{aligned}$$

リスケーリングを考慮して

$$\begin{aligned}
A'_\lambda(\boldsymbol{\kappa}) &= \frac{1}{m}\left[mk_{\rm eff,pr}\sqrt{1+\xi_{\rm ini}^2}\frac{G_0G_1+F_0F_1}{G_0^1+F_0^2}A_\lambda(\boldsymbol{\kappa}) - ma'_{\rm ini}A_\lambda(\boldsymbol{\kappa})\right. \\
&\qquad \left.+\lambda mk_{\rm eff,pr}\xi_{\rm ini}A_\lambda(\boldsymbol{\kappa}) \mp imk_{\rm eff,pr}\frac{1}{G_0^2+F_0^2}A_\lambda(\boldsymbol{\kappa})\right] \\
&= k_{\rm eff,pr}\sqrt{1+\xi_{\rm ini}^2}\frac{G_0G_1+F_0F_1}{G_0^1+F_0^2}A_\lambda(\boldsymbol{\kappa}) - a'_{\rm ini}A_\lambda(\boldsymbol{\kappa}) \\
&\qquad + \lambda k_{\rm eff,pr}\xi_{\rm ini}A_\lambda(\boldsymbol{\kappa}) \mp ik_{\rm eff,pr}\frac{1}{G_0^2+F_0^2}A_\lambda(\boldsymbol{\kappa})
\end{aligned}$$

等方的初期条件を考慮して

$$\begin{aligned}
A'_\lambda(\boldsymbol{\kappa}) = \left[k_{\rm eff,pr}\sqrt{1+\xi_{\rm ini}^2}\frac{G_0G_1+F_0F_1}{G_0^1+F_0^2} + \lambda k_{\rm eff,pr}\xi_{\rm ini} - a'_{\rm ini}\right]A_\lambda(\boldsymbol{\kappa}) \\
- i\frac{k_{\rm eff,pr}}{\sqrt{2}}\frac{|A_\lambda|_{\rm pr}}{G_0^2+F_0^2}\left[\sqrt{-\ln X_{\pm,1}}e^{i2\pi Y_{\pm,1}} - \sqrt{-\ln X_{\pm,2}}e^{i2\pi Y_{\pm,2}}\right]
\end{aligned}$$

・(9.27) の定義が関係式 (9.26) を満たすこと

この計算では有効運動量ベクトルの添字”eff”を省略する。

まず偏光ベクトルが運動量ベクトルと直行すること

$$\begin{aligned}\boldsymbol{k}\cdot\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k}) &\propto (-k_1^2k_3 \pm ik_1k_2k) + (-k_2^2k_3 \mp ik_1k_2k) + (k_3k^2 - k_3^3)\\ &= -(k_1^2+k_2^2+k_3^2)k_3 + k_3k^2 = 0\end{aligned}$$

次に規格化

$$\begin{aligned}\boldsymbol{\epsilon}^*_{\lambda_1}(\boldsymbol{k})\cdot\boldsymbol{\epsilon}_{\lambda_2}(\boldsymbol{k}) &= \frac{1}{2k^2(k^2-k_3^2)}[(-k_1k_3-\lambda_1ik_2k)(-k_1k_3+\lambda_2ik_2k)\\ &\quad + (-k_2k_3+\lambda_1ik_1k)(-k_2k_3-\lambda_2ik_1k) + (k^2-k_3^2)^2]\\ &= \frac{1}{2k^2(k^2-k_3^2)}[k_1^2k_3^2 + (\lambda_1-\lambda_2)ik_1k_2k_3k + \lambda_1\lambda_2k_2^2k^2\\ &\quad + k_2^2k_3^2 - (\lambda_1-\lambda_2)ik_1k_2k_3k + \lambda_1\lambda_2k_1^2k^2 + (k^2-k_3^2)^2]\\ &= \frac{1}{2k^2(k^2-k_3^2)}[k_1^2k_3^2 + k_2^2k_3^2 + k_3^2k_3^2 - 2k_3^2k^2 + k^4\\ &\quad + \lambda_1\lambda_2(k_1^2+k_2^2)k^2]\\ &= \frac{1}{2k^2(k^2-k_3^2)}[k^4 - k_3^2k^2 + \lambda_1\lambda_2(k_1^2+k_2^2)k^2]\\ &= \frac{1}{2(k^2-k_3^2)}[k^2 - k_3^2 + \lambda_1\lambda_2(k^2-k_3^2)]\\ &= \frac{1}{2}(1+\lambda_1\lambda_2) = \delta_{\lambda_1,\lambda_2}\end{aligned}$$

最後に外積

$$\begin{aligned}[\boldsymbol{k}\times\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})]_1 &= \frac{1}{\sqrt{2k^2(k^2-k_3^2)}}\Big(k_2(k^2-k_3^2) - k_3(-k_2k_3 \mp ik_1k)\Big)\\ &= \frac{1}{\sqrt{2k^2(k^2-k_3^2)}}\Big(k_2k^2 \pm ik_3k_1k\Big)\\ &= \frac{\mp ik}{\sqrt{2k^2(k^2-k_3^2)}}(-k_1k_3 \pm ik_2k) = \mp ik[\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})]_1\end{aligned}$$

$$\begin{aligned}[\boldsymbol{k}\times\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})]_2 &= \frac{1}{\sqrt{2k^2(k^2-k_3^2)}}\Big(k_3(-k_1k_3 \pm ik_2k) - k_1(k^2-k_3^2)\Big)\\ &= \frac{1}{\sqrt{2k^2(k^2-k_3^2)}}\Big(-k_1k^2 \pm ik_2k_3k\Big)\\ &= \frac{\mp ik}{\sqrt{2k^2(k^2-k_3^2)}}\Big(-k_2k_3 \mp ik_1k\Big) = \mp ik[\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})]_2\end{aligned}$$

$$
\begin{aligned}
[\boldsymbol{k}\times\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})]_3 &= \frac{1}{\sqrt{2k^2(k^2-k_3^2)}}\Big(k_1(-k_2k_3\mp ik_1k)-k_2(-k_1k_3\pm ik_2k)\Big)\\
&= \frac{1}{\sqrt{2k^2(k^2-k_3^2)}}\Big(\mp ik_1^2k\mp ik_2^2k\Big)\\
&= \frac{\mp ik}{\sqrt{2k^2(k^2-k_3^2)}}\Big(k_1^2+k_2^2\Big)\\
&= \frac{\mp ik}{\sqrt{2k^2(k^2-k_3^2)}}\Big(k^2-k_3^2\Big) = \mp ik[\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})]_3
\end{aligned}
$$

これらを合わせて

$$\boldsymbol{k}\times\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k}) = \mp ik\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})$$

運動量ベクトルが z 軸方向、つまり $k_{\mathrm{eff},1}=k_{\mathrm{eff},2}=0$ の場合には

$$\boldsymbol{\epsilon}^*_{\lambda_1}(\boldsymbol{k})\cdot\boldsymbol{\epsilon}_{\lambda_2}(\boldsymbol{k}) = \frac{1}{2}\Big(1-i^2\lambda_1\lambda_2\Big) = \delta_{\lambda_1,\lambda_2}$$

$$
\begin{aligned}
\boldsymbol{k}\times\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k}) &= \frac{1}{\sqrt{2}}\Big(-k_3\times\pm i,k_3,0\Big)\\
&= \frac{k}{\sqrt{2}}\Big(\mp i,1,0\Big)\\
&= \frac{\mp ik}{\sqrt{2}}\Big(1,\pm i,0\Big) = \mp ik\boldsymbol{\epsilon}_{\pm}(\boldsymbol{k})
\end{aligned}
$$

であり、運動量ベクトルと偏光ベクトルの直交性は自明とする。

・(9.28)
(6.13) の計算において

$$V \to V + \frac{\alpha}{4f}\phi \tilde{F}^{\mu\nu}F_{\mu\nu}$$

とすれば良いので、ポテンシャル項に相互作用項を加えて

$$a^2\frac{dV}{d\phi} + a^2\frac{\alpha}{4f}\tilde{F}^{\mu\nu}F_{\mu\nu}$$

とすれば良い。つまりインフラトンの運動方程式は

$$\partial_\tau^2\phi + 2\mathcal{H}\partial_\tau\phi - \nabla^2\phi + a^2\frac{dV}{d\phi} = -a^2\frac{\alpha}{4f}\tilde{F}^{\mu\nu}F_{\mu\nu}$$

となる。Lattice シミュレーションでゲージ場は $A(\tau, \boldsymbol{n})$ として扱うので、右辺の相互作用項を A で表すと

$$\begin{aligned}
\tilde{F}^{\mu\nu}F_{\mu\nu} &= \frac{\tilde{\epsilon}^{\mu\nu\alpha\beta}}{2\sqrt{-g}}(\partial_\alpha A_\beta - \partial_\beta A_\alpha)(\partial_\mu A_\nu - \partial_\nu A_\mu) \\
&= 2\frac{\tilde{\epsilon}^{\mu\nu\alpha\beta}}{\sqrt{-g}}\partial_\alpha A_\beta \partial_\mu A_\nu \\
&= \frac{2}{a^4}\tilde{\epsilon}^{\mu\nu\alpha\beta}\partial_\alpha A_\beta \partial_\mu A_\nu \\
&= \frac{2}{a^4}\left[\tilde{\epsilon}^{0ijk}\partial_j A_k \partial_0 A_i + \tilde{\epsilon}^{i0jk}\partial_j A_k \partial_i A_0 + \tilde{\epsilon}^{ij0k}\partial_0 A_k \partial_i A_j + \tilde{\epsilon}^{ijk0}\partial_k A_0 \partial_i A_j\right] \\
&= \frac{2}{a^4}\tilde{\epsilon}^{0ijk}\left[\partial_j A_k \partial_0 A_i - \partial_j A_k \partial_i A_0 + \partial_0 A_k \partial_i A_j - \partial_k A_0 \partial_i A_j\right] \\
&= \frac{2}{a^4}\tilde{\epsilon}^{0ijk}\left[\partial_j A_k(\partial_0 A_i - \partial_i A_0) + (\partial_0 A_k - \partial_k A_0)\partial_i A_j\right]
\end{aligned}$$

2 項目で $k \to i,\ i \to j,\ j \to k$ とすれば

$$\tilde{F}^{\mu\nu}F_{\mu\nu} = \frac{2}{a^4}\tilde{\epsilon}^{0ijk}\left[\partial_j A_k(\partial_0 A_i - \partial_i A_0) + (\partial_0 A_i - \partial_i A_0)\partial_j A_k\right]$$

Lattice シミュレーションでは古典場として扱う（ここら辺が少し怪しい、sub-horizon で量子ゆらぎとして扱うのが理論的な手続き）ので、2 項目でゲージ場の可換性を認めて

$$\begin{aligned}
\tilde{F}^{\mu\nu}F_{\mu\nu} &= \frac{4}{a^4}\tilde{\epsilon}^{0ijk}\partial_j A_k(\partial_0 A_i - \partial_i A_0) \\
&= \frac{4}{a^4}(\nabla \times \boldsymbol{A}) \cdot (\partial_\tau \boldsymbol{A} - \nabla A_0)
\end{aligned}$$

ここで $A_i \to \boldsymbol{A}$ とした。この相互作用項の表式を用いて

$$\partial_\tau^2\phi + 2\mathcal{H}\partial_\tau\phi - \nabla^2\phi + a^2\frac{dV}{d\phi} = -\frac{1}{a^2}\frac{\alpha}{f}(\nabla\times\boldsymbol{A})\cdot(\partial_\tau\boldsymbol{A} - \nabla A_0)$$

これはゲージ固定を指定していない表式である。

・(9.29)
(6.14) の計算と同様にすれば

$$\phi'' + (s+2)\frac{a'}{a}\phi' - \frac{\nabla_{\rm pr}^2\phi}{a^{2s}} + a^2\frac{dV_{\rm pr}}{d\phi} = \frac{1}{m^2a^{2s}}\left[-\frac{1}{a^2}\frac{\alpha}{f}(\nabla\times\boldsymbol{A})\cdot(\partial_\tau\boldsymbol{A} - \nabla A_0)\right]$$

相互作用項はリスケーリング (1.8)(1.9) より

$$\begin{aligned}
&\frac{1}{m^2a^{2s}}\left[-\frac{1}{a^2}\frac{\alpha}{f}(\nabla\times\boldsymbol{A})\cdot(\partial_\tau\boldsymbol{A} - \nabla A_0)\right] \\
&\qquad = \frac{1}{m^2a^{2s}}\left[-\frac{1}{a^2}\frac{\alpha}{f}(m\nabla_{\rm pr}\times\boldsymbol{A})\cdot(ma^s\boldsymbol{A}' - m\nabla_{\rm pr}A_0)\right] \\
&\qquad = -\frac{\alpha}{f}\frac{1}{a^{2s+2}}(\nabla_{\rm pr}\times\boldsymbol{A})\cdot(a^s\boldsymbol{A}' - \nabla_{\rm pr}A_0)
\end{aligned}$$

最終的に

$$\phi'' = -(s+2)\frac{a'}{a}\phi' + \frac{\nabla_{\rm pr}^2\phi}{a^{2s}} - a^2\frac{dV_{\rm pr}}{d\phi} - \frac{\alpha}{f}\frac{1}{a^{2s+2}}(\nabla_{\rm pr}\times\boldsymbol{A})\cdot(a^s\boldsymbol{A}' - \nabla_{\rm pr}A_0)$$

・(9.31)
ゲージ場を含むラグランジアンは

$$\begin{aligned}
\mathcal{L}_{\rm GF} &= \sqrt{-g}\left[-\frac{1}{4}F^{\mu\nu}F_{\mu\nu} - \frac{\alpha}{4f}\phi\tilde{F}^{\mu\nu}F_{\mu\nu}\right] \\
&= \sqrt{-g}\left[-\frac{1}{4}g^{\mu\alpha}g^{\nu\beta}F_{\alpha\beta}F_{\mu\nu} - \frac{\alpha}{4f}\phi\frac{\tilde{\epsilon}^{\mu\nu\alpha\beta}}{2\sqrt{-g}}F_{\alpha\beta}F_{\mu\nu}\right] \\
&= -\frac{1}{4}\eta^{\mu\alpha}\eta^{\nu\beta}F_{\alpha\beta}F_{\mu\nu} - \frac{\alpha}{8f}\phi\tilde{\epsilon}^{\mu\nu\alpha\beta}F_{\alpha\beta}F_{\mu\nu} \\
&= -\frac{1}{2}\eta^{\mu\alpha}\eta^{\nu\beta}\partial_\alpha A_\beta(\partial_\mu A_\nu - \partial_\nu A_\mu) - \frac{\alpha}{2f}\phi\tilde{\epsilon}^{\mu\nu\alpha\beta}\partial_\alpha A_\beta\partial_\mu A_\nu
\end{aligned}$$

ただし

$$\eta_{\mu\nu} = \frac{g_{\mu\nu}}{a^2} = \begin{pmatrix} -1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}$$

である。このようにゲージ場を含むラグランジアンは計量のスケール因子が相殺し、Minkowski 計量で扱えるので Euler-Lagrange 方程式も Minkowski 計量と同じもので良く

$$
\begin{aligned}
0 &= \partial_\rho \frac{\partial \mathcal{L}_{\mathrm{GF}}}{\partial(\partial_\rho A_\sigma)} \\
&= \partial_\rho \left[-\frac{1}{2}\eta^{\mu\alpha}\eta^{\nu\beta}\delta^\rho_\alpha\delta^\sigma_\beta(\partial_\mu A_\nu - \partial_\nu A_\mu) - \frac{1}{2}\eta^{\mu\alpha}\eta^{\nu\beta}\partial_\alpha A_\beta(\delta^\rho_\mu\delta^\sigma_\nu - \delta^\rho_\nu\delta^\sigma_\mu) \right. \\
&\qquad \left. -\frac{\alpha}{2f}\phi\tilde{\epsilon}^{\mu\nu\alpha\beta}(\delta^\rho_\alpha\delta^\sigma_\beta\partial_\mu A_\nu + \partial_\alpha A_\beta\delta^\rho_\mu\delta^\sigma_\nu) \right] \\
&= \partial_\rho \left[-\frac{1}{2}\eta^{\mu\rho}\eta^{\nu\sigma}(\partial_\mu A_\nu - \partial_\nu A_\mu) - \frac{1}{2}\eta^{\rho\alpha}\eta^{\sigma\beta}\partial_\alpha A_\beta + \frac{1}{2}\eta^{\sigma\alpha}\eta^{\rho\beta}\partial_\alpha A_\beta \right. \\
&\qquad \left. -\frac{\alpha}{2f}\phi\tilde{\epsilon}^{\mu\nu\rho\sigma}\partial_\mu A_\nu - \frac{\alpha}{2f}\phi\tilde{\epsilon}^{\rho\sigma\alpha\beta}\partial_\alpha A_\beta \right] \\
&= \partial_\rho \left[-\frac{1}{2}\eta^{\mu\rho}\eta^{\nu\sigma}(\partial_\mu A_\nu - \partial_\nu A_\mu + \partial_\mu A_\nu - \partial_\nu A_\mu) \right. \\
&\qquad \left. -\frac{\alpha}{2f}\phi\tilde{\epsilon}^{\mu\nu\rho\sigma}(\partial_\mu A_\nu + \partial_\mu A_\nu) \right] \\
&= \partial_\rho \left[\eta^{\mu\rho}\eta^{\nu\sigma}(\partial_\nu A_\mu - \partial_\mu A_\nu) - \frac{\alpha}{f}\phi\tilde{\epsilon}^{\mu\nu\rho\sigma}\partial_\mu A_\nu \right] \\
&= \eta^{\mu\rho}\eta^{\nu\sigma}(\partial_\rho\partial_\nu A_\mu - \partial_\rho\partial_\mu A_\nu) - \frac{\alpha}{f}\tilde{\epsilon}^{\mu\nu\rho\sigma}(\partial_\rho\phi\partial_\mu A_\nu + \phi\partial_\rho\partial_\mu A_\nu)
\end{aligned}
$$

最後の項は $\tilde{\epsilon}^{\mu\nu\rho\sigma}$ の反対称性から 0 で

$$
\eta^{\mu\rho}\eta^{\nu\sigma}(\partial_\rho\partial_\nu A_\mu - \partial_\rho\partial_\mu A_\nu) - \frac{\alpha}{f}\tilde{\epsilon}^{\mu\nu\rho\sigma}\partial_\rho\phi\partial_\mu A_\nu = 0
$$

$\sigma = 0$ の方程式は

$$
\begin{aligned}
\eta^{\mu\rho}(-1)(\partial_\rho\partial_0 A_\mu - \partial_\rho\partial_\mu A_0) - \frac{\alpha}{f}\tilde{\epsilon}^{ijk0}\partial_k\phi\partial_i A_j &= 0 \\
\eta^{\mu\rho}(\partial_\rho\partial_0 A_\mu - \partial_\rho\partial_\mu A_0) + \frac{\alpha}{f}\tilde{\epsilon}^{ijk0}\partial_k\phi\partial_i A_j &= 0 \\
\partial_i\partial_0 A_i - \partial_i\partial_i A_0 - \frac{\alpha}{f}\tilde{\epsilon}^{0kij}\partial_k\phi\partial_i A_j &= 0 \\
\partial_\tau(\nabla\cdot\boldsymbol{A}) - \nabla^2 A_0 - \frac{\alpha}{f}(\nabla\phi)\cdot(\nabla\times\boldsymbol{A}) &= 0 \\
\nabla^2 A_0 - \partial_\tau(\nabla\cdot\boldsymbol{A}) + \frac{\alpha}{f}(\nabla\phi)\cdot(\nabla\times\boldsymbol{A}) &= 0
\end{aligned}
$$

$\sigma = i$ の方程式は

$$\begin{aligned}
0 &= \eta^{\mu\rho}(\partial_\rho\partial_i A_\mu - \partial_\rho\partial_\mu A_i) - \frac{\alpha}{f}\tilde{\epsilon}^{\mu\nu\rho i}\partial_\rho\phi\partial_\mu A_\nu \\
&= -(\partial_0\partial_i A_0 - \partial_0\partial_0 A_i) + (\partial_j\partial_i A_j - \partial_j\partial_j A_i) \\
&\quad - \frac{\alpha}{f}(\tilde{\epsilon}^{0jki}\partial_k\phi\partial_0 A_j + \tilde{\epsilon}^{j0ki}\partial_k\phi\partial_j A_0 + \tilde{\epsilon}^{jk0i}\partial_0\phi\partial_j A_k) \\
&= \partial_\tau^2 A_i - \nabla^2 A_i + \partial_i(\nabla\cdot\boldsymbol{A} - \partial_\tau A_0) \\
&\quad - \frac{\alpha}{f}\tilde{\epsilon}^{0ijk}(\partial_k\phi\partial_0 A_j - \partial_k\phi\partial_j A_0 + \partial_0\phi\partial_j A_k) \\
&= \partial_\tau^2 A_i - \nabla^2 A_i + \partial_i(\nabla\cdot\boldsymbol{A} - \partial_\tau A_0) \\
&\quad - \frac{\alpha}{f}\partial_\tau\phi(\nabla\times\boldsymbol{A})_i + \frac{\alpha}{f}[(\nabla\phi)\times(\partial_\tau\boldsymbol{A} - \nabla A_0)]_i
\end{aligned}$$

ここで 3 項目は Lorentz ゲージと Coulomb ゲージの両方で 0 になるので、今は省略して

$$\partial_\tau^2\boldsymbol{A} - \nabla^2\boldsymbol{A} - \frac{\alpha}{f}\partial_\tau\phi(\nabla\times\boldsymbol{A}) + \frac{\alpha}{f}(\nabla\phi)\times(\partial_\tau\boldsymbol{A} - \nabla A_0) = 0$$

・(9.32)

Lorentz ゲージ条件より

$$\partial^\mu A_\mu = g^{\mu\nu}\partial_\nu A_\mu = a^{-2}(-\partial_\tau A_0 + \nabla\cdot\boldsymbol{A}) = 0$$

を用いると $\sigma = 0$ の方程式は A_0 の 2 階微分方程式でかけて

$$\partial_\tau^2 A_0 - \nabla^2 A_0 - \frac{\alpha}{f}(\nabla\phi)\cdot(\nabla\times\boldsymbol{A}) = 0$$

リスケーリング (1.8)(1.9) を考慮して

$$\begin{aligned}
m^2a^{2s}\left[A_0'' + s\frac{a'}{a}A_0'\right] - m^2\nabla_{\rm pr}^2 A_0 - \frac{\alpha}{f}(m\nabla_{\rm pr}\phi)\cdot(m\nabla_{\rm pr}\times\boldsymbol{A}) = 0 \\
A_0'' + s\frac{a'}{a}A_0' - \frac{\nabla_{\rm pr}^2 A_0}{a^{2s}} - \frac{\alpha}{f}\frac{1}{a^{2s}}(\nabla_{\rm pr}\phi)\cdot(\nabla_{\rm pr}\times\boldsymbol{A}) = 0
\end{aligned}$$

$\sigma = i$ の方程式は変更されず、リスケーリングを考慮して

$$\begin{aligned}
m^2a^{2s}\left[\boldsymbol{A}'' + s\frac{a'}{a}\boldsymbol{A}'\right] - m^2\nabla_{\rm pr}^2\boldsymbol{A} - \frac{\alpha}{f}ma^s\phi'(m\nabla_{\rm pr}\times\boldsymbol{A}) \\
+\frac{\alpha}{f}(m\nabla_{\rm pr}\phi)\times(ma^s\boldsymbol{A}' - m\nabla_{\rm pr}A_0) = 0
\end{aligned}$$

$$\boldsymbol{A}'' + s\frac{a'}{a}\boldsymbol{A}' - \frac{\nabla_{\rm pr}^2\boldsymbol{A}}{a^{2s}} - \frac{\alpha}{f}\frac{\phi'}{a^s}(\nabla_{\rm pr}\times\boldsymbol{A}) + \frac{\alpha}{f}\frac{1}{a^{2s}}(\nabla_{\rm pr}\phi)\times(a^s\boldsymbol{A}' - \nabla_{\rm pr}A_0) = 0$$

・(9.33)(9.44)
Coulomb ゲージ条件より

$$A_0 = \partial_\tau A_0 = 0 \quad , \quad \nabla \cdot \boldsymbol{A} = 0$$

を用いると $\sigma = 0$ の方程式は

$$(\nabla\phi) \cdot (\nabla \times \boldsymbol{A}) = 0$$

リスケーリングを考慮して

$$(\nabla_{\rm pr}\phi) \cdot (\nabla_{\rm pr} \times \boldsymbol{A}) = 0$$

これは発展に伴って破れないかを調べておく必要がある量である。ゲージ条件と同様にすれば

$$F \equiv \frac{(\nabla_{\rm pr}\phi) \cdot (\nabla_{\rm pr} \times \boldsymbol{A})}{\sqrt{\sum_i |\partial_{{\rm pr},i}\phi(\nabla_{\rm pr} \times \boldsymbol{A})_i|^2}}$$

の値を調べるべきである。一方、$\sigma = i$ の方程式は

$$\partial_\tau^2 \boldsymbol{A} - \nabla^2 \boldsymbol{A} - \frac{\alpha}{f}\partial_\tau\phi(\nabla \times \boldsymbol{A}) + \frac{\alpha}{f}(\nabla\phi) \times (\partial_\tau \boldsymbol{A}) = 0$$

リスケーリングを考慮すると

$$\begin{aligned} & m^2 a^{2s}\left[\boldsymbol{A}'' + s\frac{a'}{a}\boldsymbol{A}'\right] - m^2\nabla_{\rm pr}^2\boldsymbol{A} \\ & \quad - \frac{\alpha}{f}ma^s\phi'(m\nabla_{\rm pr} \times \boldsymbol{A}) + \frac{\alpha}{f}(m\nabla_{\rm pr}\phi) \times (ma^s\boldsymbol{A}') = 0 \\ & \boldsymbol{A}'' + s\frac{a'}{a}\boldsymbol{A}' - \frac{\nabla_{\rm pr}^2\boldsymbol{A}}{a^{2s}} - \frac{\alpha}{f}\frac{\phi'}{a^s}(\nabla_{\rm pr} \times \boldsymbol{A}) + \frac{\alpha}{f}\frac{1}{a^s}(\nabla_{\rm pr}\phi) \times \boldsymbol{A}' = 0 \end{aligned}$$

・(9.38)
ゲージ場の作用

$$S_A = \int d^4x\sqrt{-g}\left(-\frac{1}{4}F^{\mu\nu}F_{\mu\nu}\right)$$

を計量に対して変分して

$$\begin{aligned}
\delta S_A &= \int d^4x \left[(\delta\sqrt{-g})\left(-\frac{1}{4}F^{\mu\nu}F_{\mu\nu}\right) + \sqrt{-g}\left(-\frac{1}{4}F_{\mu\nu}F_{\alpha\beta}\right)\delta(g^{\mu\alpha}g^{\nu\beta})\right] \\
&= \int d^4x\sqrt{-g}\left[-\frac{1}{2}g_{\mu\nu}\left(-\frac{1}{4}F^{\alpha\beta}F_{\alpha\beta}\right)\delta g^{\mu\nu}\right. \\
&\qquad\qquad \left. + \left(-\frac{1}{4}F_{\mu\nu}F_{\alpha\beta}\right)(g^{\mu\alpha}\delta g^{\nu\beta} + g^{\nu\beta}\delta g^{\mu\alpha})\right] \\
&= \int d^4x\sqrt{-g}\left[-\frac{1}{2}g_{\mu\nu}\left(-\frac{1}{4}F^{\alpha\beta}F_{\alpha\beta}\right)\delta g^{\mu\nu} + 2\left(-\frac{1}{4}F_{\mu\nu}F_{\alpha\beta}\right)g^{\nu\beta}\delta g^{\mu\alpha}\right] \\
&= \int d^4x\sqrt{-g}\left[-\frac{1}{2}g_{\mu\nu}\left(-\frac{1}{4}F^{\alpha\beta}F_{\alpha\beta}\right) - \frac{1}{2}g^{\alpha\beta}F_{\mu\alpha}F_{\nu\beta}\right]\delta g^{\mu\nu}
\end{aligned}$$

より、ゲージ場のエネルギー運動量テンソルは

$$T_{A,\mu\nu} = g^{\alpha\beta}F_{\mu\alpha}F_{\nu\beta} - \frac{1}{4}g_{\mu\nu}F^{\alpha\beta}F_{\alpha\beta}$$

ここからゲージ場のエネルギー密度は

$$\begin{aligned}
\rho_A &= -T^0{}_0 = -g^{0\mu}g^{\alpha\beta}F_{\mu\alpha}F_{0\beta} + g^{0\mu}\frac{1}{4}g_{\mu0}F^{\alpha\beta}F_{\alpha\beta} \\
&= a^{-2}g^{\alpha\beta}F_{0\alpha}F_{0\beta} + \frac{1}{4}F^{\alpha\beta}F_{\alpha\beta} \\
&= a^{-4}(F_{0i})^2 + \frac{1}{4}g^{\alpha\mu}g^{\beta\nu}F_{\alpha\beta}F_{\mu\nu} \\
&= a^{-4}(F_{0i})^2 - \frac{1}{4}a^{-2}g^{\beta\nu}F_{0\beta}F_{0\nu} + \frac{1}{4}a^{-2}g^{\beta\nu}F_{i\beta}F_{i\nu} \\
&= a^{-4}(F_{0i})^2 - \frac{1}{4}a^{-4}(F_{0i})^2 - \frac{1}{4}a^{-4}(F_{0i})^2 + \frac{1}{4}a^{-4}(F_{ij})^2 \\
&= \frac{1}{2}a^{-4}(F_{0i})^2 + \frac{1}{4}a^{-4}(F_{ij})^2 \\
&= \frac{1}{2a^4}\sum_i(\partial_\tau A_i - \partial_i A_0)^2 + \frac{1}{4a^4}\sum_{i,j}(\partial_i A_j - \partial_j A_i)^2 \\
&= \frac{1}{2a^4}\left[\sum_i(\partial_\tau A_i - \partial_i A_0)^2 + \sum_{i,j}(\partial_i A_j\partial_i A_j - \partial_i A_j\partial_j A_i)\right]
\end{aligned}$$

リスケーリング (1.8)(1.9)(1.10) を考慮すれば

$$\begin{aligned}
\rho_{A,\mathrm{pr}} &= \frac{\rho_A}{m^2 a^{2s}} \\
&= \frac{1}{m^2 a^{2s}} \frac{1}{2a^4} \left[\sum_i (m a^s A_i' - m\partial_{\mathrm{pr},i} A_0)^2 \right. \\
&\qquad \left. + \sum_{i,j} (m\partial_{\mathrm{pr},i} A_j m\partial_{\mathrm{pr},i} A_j - m\partial_{\mathrm{pr},i} A_j m\partial_{\mathrm{pr},j} A_i) \right] \\
&= \frac{1}{2a^{2s+4}} \left[\sum_i (a^s A_i' - \partial_{\mathrm{pr},i} A_0)^2 + \sum_{i,j} (\partial_{\mathrm{pr},i} A_j \partial_{\mathrm{pr},i} A_j - \partial_{\mathrm{pr},i} A_j \partial_{\mathrm{pr},j} A_i) \right]
\end{aligned}$$

・

1 階微分は

$$\begin{aligned}\frac{dV_{\rm pr}}{d\phi} &= \frac{1}{a^{2s}}\frac{d}{d\phi}\left[\frac{\phi^2}{2}+\beta f\phi\sin\left(\frac{\phi}{f}\right)\right]\\ &= \frac{1}{a^{2s}}\left[\phi+\beta f\sin\left(\frac{\phi}{f}\right)+\beta f\phi\frac{1}{f}\cos\left(\frac{\phi}{f}\right)\right]\\ &= \frac{\phi}{a^{2s}}+\frac{\beta f}{a^{2s}}\left[\sin\left(\frac{\phi}{f}\right)+\frac{\phi}{f}\cos\left(\frac{\phi}{f}\right)\right]\end{aligned}$$

2 階微分は

$$\frac{d^2V_{\rm pr}}{d\phi^2} = \frac{1}{a^{2s}}\frac{d}{d\phi}\left[\phi+\beta f\sin\left(\frac{\phi}{f}\right)+\beta f\frac{\phi}{f}\cos\left(\frac{\phi}{f}\right)\right]$$

先にスケール因子の初期条件 (3.1) を使って

$$\begin{aligned}\frac{d^2V_{\rm pr}}{d\phi^2} &= \frac{d}{d\phi}\left[\phi+\beta f\sin\left(\frac{\phi}{f}\right)+\beta f\frac{\phi}{f}\cos\left(\frac{\phi}{f}\right)\right]\\ &= 1+\beta\cos\left(\frac{\phi}{f}\right)+\beta\cos\left(\frac{\phi}{f}\right)-\beta\frac{\phi}{f}\sin\left(\frac{\phi}{f}\right)\\ &= 1+\beta\left[2\cos\left(\frac{\phi}{f}\right)-\frac{\phi}{f}\sin\left(\frac{\phi}{f}\right)\right]\end{aligned}$$

参考文献

[1] G. Felder, I. Tkachev, *LATTICEEASY: A Program for Lattice Simulations of Scalar Fields in an Expanding Universe*, [hep-ph/0011159]

[2] A. Caravano, *Simulating the Inflationary Universe: from single-field to the axion-U(1) model*, [2209.13616]

[3] A. Caravano et al., *Lattice Simulations of Inflation*, [2102.06378]

[4] A. Caravano et al., *Lattice simulations of Abelian gauge fields coupled to axions during Inflation*, [2110.10695]

[5] A. Caravano et al., *Lattice Simulations of Axion-U(1) Inflation*, [2204.12874]

[6] D. G. Figueroa, *The art of simulating the early Universe – Part I*, [2006.15122]

[7] R. Sharma et al., *Lattice simulations of axion-U(1) inflation: gravitational waves, magnetic fields, and black holes*, [2411.04854]

[8] https://www.cms-initiative.jp/ja/events/20130425-katagiri.pdf

[9] http://www.sanko-shoko.net/note.php?id=9twp