LATTICEEASY(日本語訳)

Haruto Yokoyama

2021 年 3 月 18 日

目次

第 1 章

概要

LATTICEEASY は C 発展する宇宙においてスカラー場との相互作用の格子シミュレーションをする C++ のプログラムです。プログラムは簡単にパラメータを変えて動かすことができ、簡単に評価のために新しいモデルを挿入可能なようにデザインされています。LATTICEEASY2.0 ではこれらのシミュレーションを 1-3 の次元で一つの変数を変えるだけで簡単に行うことができます。また、LATTICEEASY の並列処理のバージョンとして CLUSTEREASY があります。

プログラムは Web サイトの `http://www.science.smith.edu/departments/Physics/fstaff/gfelder/latticeeasy` にて使用可能です。これは自由に誰でも使い、修正することが可能です。詳細は第 8 章:基本的には我々のクレジットと連絡先を明記していただければ自由に変更を加えていただいて問題ありません。

もし何らかの質問やコメントがありましたら、gfelder@email.smith.edu までご連絡ください。我々はあなたにとってプログラムがどんな風に動き、あなたが何らかの問題を解決できれば光栄です。バグや改善点などプログラムに関してご連絡いただければ幸いです。

この資料は 4 つのメインセクションに分かれています。4 章は LATTICEEASY の使い方について記載されており、どのようにプログラムをコンパイルし、走らせるか、そしてどのように与えられたモデルに対して適切なパラメータをセットするか、最終的にどのように新しいモデルをプログラムを走らせるために作るかが記載されています。5 章ではプログラムのアウトプットについて記載されています。そこにはアウトプットの関数と出力されるファイルに何が記載されているかを述べています。6 章ではプログラムで用いる式について記載されています。このセクションの大半はプログラムを使う上で必要はありません。しかし、どのようにしてプログラムが動いているのか、何が起こっているのかを知る上では必要となってきます。このセクションで最も重要なパートは変数をプログラムでリスケールすることができることです。式を簡単にするためにプログラムは場や時空に対してリスケールされた値を用い、これらのリスケールについては 6 章にて説明がなされています。これらのリスケールは 4 章や 5 章を通して使われています。なのでそれらのセクションを読むときにより理解理解しやすくなるでしょう。7 章は LATTICEEASY の並列処理について記載されています。

メインセクションに比べてマイナーなセクションがいくらかあります。2 章は記法と慣習につい

て、3 章は LATTICEEASY ファイルのリストとそれぞれが何をしているかについて説明しています。8 章はプログラムの使用方法について。この"クレジット"セクションは latticeeasy.cpp の一番上に記載してください。

第 2 章

記法と慣習

それぞれの値はプランク単位である $M_p \approx 1.22 \times 10^{19}$ GeV で与えられています。FRW 計量は $g_{\mu\nu} = diag\{1, -a^2, -a^2, -a^2\}$ となり、これに対する時間は物理単位でかかれ、距離は共動座標で測られています。プログラムでリスケールされた変数は下付き文字で pr がつけられています。また、繰り返しの変数は特に明記せず和が取られます。上付きドットは実時間での微分を表し、プライム記号はプログラム時間 t_{pr} での微分を表します。

場の一般的な特性は、フィールド変数 f を用いて示されます。f_i のような下付き文字は必要に応じて用いられます。F_k と f_k はそれぞれ連続と離散の f のフーリエ変換を指します。これらに関しては 6.3 にて議論します。

スケールファクターと時間は利便性のため、始め 1 と 0 にセットされています。

この文書は LATTICEEASY に含まれるファイルや、プログラムの変数や関数を参照していることが多くあります。これらのファイル名は `latticeeasy.cpp` のようにタイプライター体で書かれます。プログラムによって作成されるアウトプットファイルも同じ方法で記載されます。プログラム中の関数や変数もタイプライターフォントですが、$nflds$ や $potentialenergy()$ のように斜体になっています。(ただし、プログラム変数が数式の中に出てくる場合は、読みやすいように通常の数式の形式で表示されます。) 関数は常に括弧で表示されますが、関数の引数は通常含まれません。

第 3 章

LATTICEEASY のファイル

LATTICEEASY は以下のファイルから構成されます。

latticeeasy.cpp latticeeasy の $main()$ 関数とグローバル変数の宣言が含まれています。(すべて `latticeeasy.h` で extern として宣言されています。)

initialize.cpp 初期化に関することをする $initialize()$ 関数を含みます。フィールドの揺らぎの初期条件の設定、スケールファクターの微分値の初期値の設定、各種パラメーターの初期化などがあります。

evolution.cpp 場やその微分値やスケールファクターやその微分値の増幅に関する関数が含まれています。またこのファイルは場のエネルギー勾配をも計算する関するが含まれています。

output.cpp 与えられた量を計算し保存する関数が含まれています。これらの量には場の平均値、分散、スケールファクターとその微分値、エネルギー密度、スペクトル、ヒストグラムが含まれています。また、このファイルには、格子のイメージを定期的に保存する機能も含まれています。

fftеasy.cpp フーリエ変換をするコードが含まれています。 FFTEASY は、Gary Felder 氏が執筆した独立したルーチンのセットで、`http://www.science.smith.edu/departments/Physics/fstaff/gfelder/ffteasy` に記載されています。

latticeeasy.h 実行するためのすべてのパラメータが記載されています。グリッドサイズやタイムステップなどの変数に加え、どの出力をどのくらいの頻度で保存するかなどの変数もあります。

model.h それぞれのモデルに関する情報が含まれています。これらのファイルのライブラリは、"models"ディレクトリに含まれています。これらのファイルのどれかをプログラムディレクトリにコピーし、`model.h` という名前に変更することで、そのモデルを実行することができます。

makefile プログラムをコンパイルするためのもの。デフォルトは g++ を使うためにセットされていますが、簡単に修正できます。

第 4 章

LATTICEEASY の使用

基本的なプログラムの構造を以下に示します。特定のモデルに固有の情報を含む `model.h` というファイルがあります。例として、デフォルトのモデルとして記述しておいた 2 つのスカラー場 ϕ と χ のモデルが置かれています。

$$V = \frac{1}{4}\lambda\phi^4 + \frac{1}{2}g^2\phi^2\chi^2. \quad \text{(TWOFLDLAMBDA model)} \tag{4.0.1}$$

動かす新しいモデルプログラムを作るために、あなたは新しいモデルファイルを作成する必要があります。我々は以下でどのようにしてこれを行うのかを説明します。

プログラムを走らせるためにパラメータ (grid size, time step, etc.) をセットし、コンパイルをします。すべての調節可能なパラメータは `parameters.h` に記載されています。したがってユーザーとしてあなたが修正すべきファイルは `parameters.h` と `model.h` の 2 つだけです。`latticeeasy.cpp` のはじめに我々のクレジットさえ記載していただければ。もちろん他のファイルについても修正していただいて構いません。また、このプログラムには多くのコメントが付けられているので、プログラミングの経験が豊富な人であれば、各部分が何をしているのかを理解することができるでしょう。

以下のセクションでは、特定のモデルに対してプログラムを実行する方法と、新しいモデルをプログラムに組み込む方法を順に説明します。

4.1 実在するモデルでプログラムを動かす

プログラムを特定のモデルで実行するように設定するには、そのモデルのヘッダーファイルを他のソースファイルのあるディレクトリにコピーし、`model.h` と名付けます。LATTICEEASY を実行するには、`parameters.h` ファイルにパラメータを設定し、プログラムをコンパイルしてから実行します。 LATTICEEASY で実行するには `parameters.h` というファイルにパラメータを設定し、プログラムをコンパイルして実行するようになっていて簡単そうに見えますが、40 以上のパラメータが設定されています。実際、これらの調整可能なパラメータのほとんどは、プログラムにどのような種類の出力をさせたいかによるものです。4.1.3 節では `parameters.h` にあるすべて

のパラメータについて説明します。4.1.4 節ではグリッドサイズやタイムステップなどの量について、妥当な値を選ぶためのアドバイスをします。最後に、4.1.5 節では、倍精度の実数での実行方法を説明しています。

4.1.1 コンパイル

他のコンパイラを使用している場合は、ファイルを直接コンパイルするか、必要に応じて makefile を編集してください。例えば、makefile の先頭にある "COMPILER" という変数を "g++" 以外のものに変更することができます。最適化をオンにしておくことをお勧めします。現在、makefile は "latticeeasy "という実行ファイルを作成するように設定されており、ソースコードと同じディレクトリで実行できるようになっています。警告：古いバージョンの GNU コンパイラにはバグがあり、それが原因でプログラムがクラッシュすることがあります。g++ を使用していてプログラムが segmentation fault で終了する場合は、g++ -v を実行してコンパイラのバージョンを確認してください。その結果、「egcs-1.0.3 release」と表示された場合は、バグのあるバージョンを使用していることになります。 このバグについての詳細をお知りになりたい方は、Gary Felder（gfelder@email.smith.edu）までメールでお問い合わせください。

4.1.2 プログラムの実行と古い実行への続き

プログラムの実行は、コンパイル時に作成した実行ファイルを実行するだけです。コマンドラインには何のパラメータもありませんので、「./latticeeasy」と入力するだけでプログラムが実行され、カレントディレクトリにすべての出力ファイルが作成されます。

$t = 100$ というプログラムを実行した後，時々その後で何が起こるかを見たいと思うことがあります。これは、最初の実行時にグリッドイメージを保存するように *checkpoint* を設定していれば可能です。単に、$tf = 200$(または任意の値) になるように `parameters.h` を編集し、$continue\_run$ を 1 または 2 に設定して、再実行します。$continue\_run = 1$ の場合、プログラムは新しいデータを前回の実行時の出力ファイルに追加します。$continue\_run = 2$ の場合、新しいファイルは異なる拡張子で作成されます。出力ファイルのデフォルトの拡張子は "_0.dat "ですが、実行の継続のために新しい出力ファイルを作成することを選択した場合、最初の継続では "_1.dat"、次の継続では "_2.dat "という拡張子になります。

プログラムが実行の継続を設定し、カレントディレクトリにグリッドイメージファイルがない場合、単に新しい実行を開始します。プログラムがディレクトリ内に保存された時刻が t_f の値と同じかそれ以降のグリッドイメージファイルを検出した場合は、単に警告メッセージを表示して終了します。

$t = 100$ で実行した後に $t = 200$ で続行しても、結果の出力は直接 $t = 200$ で実行した場合とは必ずしも一致しないことに注意してください。すべての実行でパラメータ $noutput\times$ を 1000 のままにしたとすると、プログラムは 0 から 100 までの約 1000 回で平均と分散を生成し、100 から

200 までの約 1000 回で平均と分散を生成するので、実行を 2 段階で行うと、1 段階で行う場合の 2 倍の出力回数になります。1 回の実行で出力する回数を調整することで、この挙動を変えることができます。実際のフィールド発展はどちらも同じです。