FDPS 講習会の手引

谷川衝、岩澤全規、細野七月、似鳥啓吾、村主崇行、野村昴太郎、坪内美幸、牧野淳一郎

2026 年 9 月 9 日

目次

1 準備

1.1 自分で用意した計算機で実行する場合

https://github.com/FDPS/FDPS から FDPS の最新版をダウンロードし、好きなディレクトリ下で解凍する。これによってディレクトリ FDPS-master が出来る。

以下の方法のいずれかで FDPS の最新バージョンを取得できる。

- ブラウザから
  1. ウェブサイト https://github.com/FDPS/FDPS で”Download ZIP” をクリックし、ファイル FDPS-master.zip をダウンロード
  2. FDPS を展開したいディレクトリに移動し、圧縮ファイルを展開
- コマンドラインから
  - Git を用いる場合：以下のコマンドを実行するとカレントディレクトリにディレクトリ FDPS ができ、その下を Git のレポジトリとして使用できる

```
$ git clone git://github.com/FDPS/FDPS.git
```

以下では、ウェブサイトからダウンロードした場合を想定し、ディレクトリ FDPS-master があるディレクトリの名前を fdps とする。

2 実習本番

実習で行うことは、FDPS を使って実装された重力 N 体シミュレーションコードと SPH シミュレーションコードを使用することである。最初に重力 N 体シミュレーションコード、次に SPH シミュレーションコードを使用する。

なお、実習の際に Makefile を更新し、コンパイルし直すという作業を何回か行うが、ここで注意しなくてはならないのは、**Makefile を編集しただけでは実行ファイルの再作成は行われない**ということである。この場合、きちんと前回作った実行ファイルを明示的に “$ rm ./nbody.out” などで消す必要がある。これを忘れた場合、「make: `nbody.out' は更新済みです」と出る。もし，GNU make を使っている場合は make -B nbody.out などとすることで，ファイルのタイムスタンプを無視して，再コンパイルすることもできる.

2.1 重力 N 体シミュレーションコード

ここでは、重力 N 体シミュレーションコードでの cold collapse を、並列環境無し、OpenMP を用いた並列計算環境、OpenMP + MPI を用いた並列計算環境の 3 つで行う。MPI は、環境があれば行う。

2.1.1 概要

ここでは、用意された重力 N 体シミュレーションコードを動かしてみよう。このコードは、重力多体系のコールドコラプスを計算する。この節でまず行うことは、シリアルコードのコンパイルと実行、出て来た結果の解析である。次にシリアルコードを Phantom-GRAPE や PIKG を用いて高速化して、その速さを体験しよう。最後に OpenMP や MPI を利用して、さらにコードを高速化する。

2.1.2 シリアルコード

以下の手順で本コードを使用できる。

- ディレクトリ `fdps/FDPS-master/sample/c++/nbody` に移動
- `make` を実行
- ジョブの投入
- 結果の解析
- OpenMP/MPI の利用 (オプション)

2.1.2.1 ディレクトリ移動

ディレクトリ `fdps/FDPS-master/sample/c++/nbody` に移動する。

```
$ cd fdps/FDPS-master/sample/c++/nbody
```

2.1.2.2 make の実行

`make` コマンドを実行する。

```
$ make
```

2.1.2.3 計算の実行

まずは、インタラクティブ実行で計算を実行する。これは、生成された実行ファイルの名前をそのまま実行すればよい。

```
$ ./nbody.out
```

正しくジョブが開始すると、以下のような表示がされる。

```
     //==================================\\
     ||                                  ||
     || ::::::: ::::::. ::::::. .::::::. ||
     || ::      ::    : ::    : ::       ||
     || ::::::  ::    : ::::::'  `:::::. ||
     || ::      ::::::' ::      `.....' ||
     ||     Framework for Developing     ||
     ||        Particle Simulator        ||
     ||     Version 8.0 (2025/09)       ||
     \\==================================//

       Home   : https://github.com/fdps/fdps
       E-mail : fdps-support@mail.jmlab.jp
       Licence: MIT (see, https://github.com/FDPS/FDPS/blob/master/LICENSE)
       Note   : Please cite the following papers.
                - Iwasawa et al. (2016, Publications of the Astronomical Society of
                - Namekata et al. (2018, Publications of the Astronomical Society of

       Copyright (C) 2015
         Masaki Iwasawa, Ataru Tanikawa, Natsuki Hosono,
         Keigo Nitadori, Takayuki Muranushi, Daisuke Namekata,
         Kentaro Nomura, Junichiro Makino and many others
******** FDPS has successfully begun. ********
Directory "./result" is successfully made.
This is a sample program of N-body simulation on FDPS!
Number of processes: 1
Number of threads per process: 1
(省略)
```

さらに，標準入出力の最後には以下のようなログが出力されるはずである。energy error は絶対値で 1×10^{-3} のオーダーに収まっていればよい。

```
（省略）
time:  9.5000000 energy error: -3.509366e-03
time:  9.6250000 energy error: -3.582904e-03
time:  9.7500000 energy error: -3.616002e-03
time:  9.8750000 energy error: -3.510957e-03
time: 10.0000000 energy error: -3.707107e-03
MemoryPool::finalize() is completed!
******** FDPS has successfully finished. ********
```

ただし、後述する Phantom-GRAPE や PIKG を用いた場合、energy error 数値は変わるので注意する。

2.1.2.4 結果の解析

ディレクトリ `result` に粒子分布を出力したファイル"000*.dat" ができている。*は 0 から 9 の値で、時刻を表す。出力ファイルフォーマットは 1 列目から順に粒子の ID, 粒子の質量、位置の x, y, z 座標、粒子の x, y, z 軸方向の速度である。

ここで実行したのは、粒子数 1024 個からなる一様球 (半径 3) のコールドコラプスである。コマンドライン上で以下のコマンドを実行すれば、時刻 9 における xy 平面に射影した粒子分布を見ることができる。

```
$ gnuplot
$ plot "result/0009.dat" using 3:4
```

他の時刻の粒子分布をプロットすると、一様球が次第に収縮し、その後もう一度膨張する様子を見ることができる (図 1 参照)。

2.1.3 Phantom-GRAPE の利用

以下では、相互作用計算に Phantom-GRAPE を使う場合について、記述する。この場合、ユーザーはまずは Phantom-GRAPE のコンパイルを行わなければならない。今回使う Phantom-GRAPE のソースコードは、`fdps/FDPS-master/src/phantom_grape_x86/G5/newton/libpg5/` 以下に存在するので、そこまで移動し、コンパイルを行う。以下は、現在 `sample/c++/nbody/` に居る場合の例である。

```
$ cd ../../../src/phantom_grape_x86/G5/newton/libpg5/
$ make
```

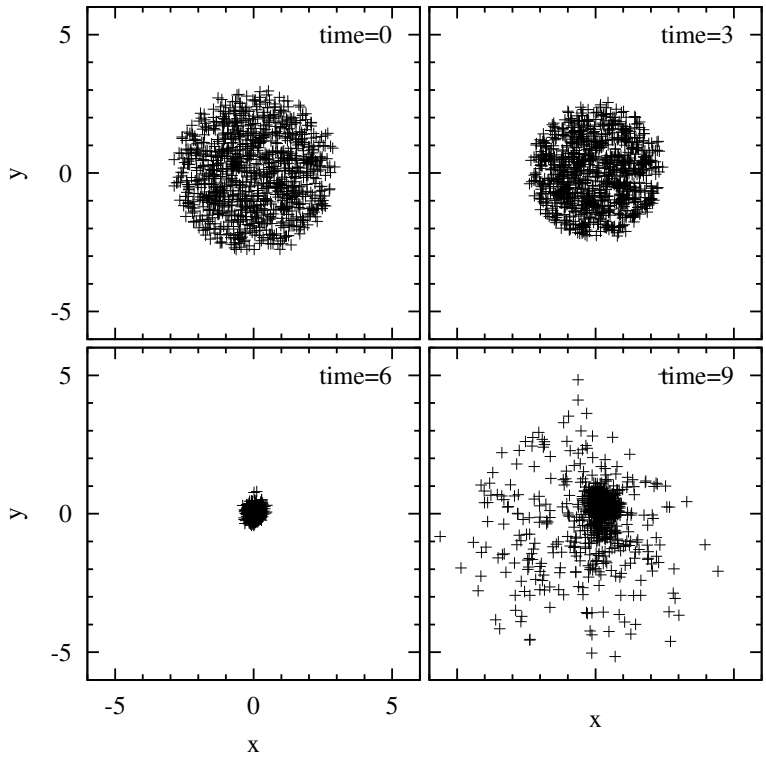

図 1

コンパイルが成功したら、元のディレクトリに戻る。次に、Makefile の修正を行う。Makefile の 13 行目に Phantom-GRAPE を使用するか否かを決定しているスイッチが存在している。このスイッチはデフォルトではコメントアウトされて no になっているため、以下のようにしてコメントアウトを解除する。

```
use_phantom_grape_x86 = yes
```

無事にコンパイルが通れば、以降の実行・解析の手順は同様である。実行直後に次のような表示がされれば、正しく実行ができている。

```
******** FDPS has successfully begun. ********
./result/t-de.dat
Number of processes: 1
Number of threads per process: 1
rsqrt: MSE = 1.158186e-04,  Bias = 8.375360e-08
(以下省略)
```

2.1.4 PIKG の利用

以下では、相互作用計算に PIKG を使う場合について、記述する。この Makefile の修正を行う。Makefile の 14 行目に PIKG を使用するか否かを決定しているスイッチが存在している。このスイッチはデフォルトではコメントアウトされて no になっているため、以下のようにしてコメントアウトを解除する。

```
use_pikg_x86 = yes
```

デフォルトでは，PIKG は通常の C++ コードを生成するようになっている．59・60 行目のコメントアウトを外すことにより，AVX2 向けに最適化されたコードが生成されるのでコンパイルし直して (make -B など) 実行すると速度が変わる．

```
CONVERSION_TYPE = AVX2
CFLAGS += -mavx2 -mfma -ffast-math
```

無事にコンパイルが通れば、以降の実行・解析の手順は同様である。

2.1.5 OpenMP/MPI の利用

OpenMP や MPI を利用する場合について以下に記述する。

- OpenMP のみ使用の場合
  - Makefile の編集
    - マクロ CC に OpenMP 対応の C++ コンパイラを代入する。今回の実習の環境では変更する必要は無い。
    - “CFLAGS += -DPARTICLE_SIMULATOR_THREAD_PARALLEL -fopenmp” の 行のコメントアウトを外す
  - make コマンドを実行する。
  - 実行方法はシリアルコードの場合と同じである。

    ```
    ******** FDPS has successfully begun. ********
    ./result/t-de.dat
    Number of processes: 1
    Number of threads per process: 4
    rsqrt: MSE = 1.158186e-04,  Bias = 8.375360e-08
    (以下省略)
    ```

    見て分かる通り、Number of threads per process: 4 となっている。これで、

4 スレッドでの並列計算が行われている事が確認できた。スレッド数を変更するには、以下のように環境変数 OMP_NUM_THREADS を設定する必要がある。

```
// 恒久的に設定する場合.（bash）
export OMP_NUM_THREADS=4
./nbody.out
```

```
// 一時的に設定する場合
OMP_NUM_THREADS=4 ./nbody.out
```

- OpenMP と MPI の同時使用の場合
  - Makefile の編集
    - マクロ CC に MPI 対応の C++ コンパイラを代入する。今回の実習の環境では mpic++ にする。
    - “CFLAGS += -DPARTICLE_SIMULATOR_THREAD_PARALLEL -fopenmp” の行のコメントアウトを外す (インテルコンパイラの場合は-fopenmp を外す)
    - “CFLAGS += -DPARTICLE_SIMULATOR_MPI_PARALLEL” の行のコメントアウトを外す
  - make コマンドを実行する。
  - システムの MPI 環境の方法で実行する。(例えば、OMP_NUM_THREADS=4 mpirun -np 2 nbody.out) 正しく実行された場合、以下のように表示されるはずである。

```
（省略）
******** FDPS has successfully begun. ********
./result/t-de.dat
./result/t-de.dat
Number of processes: 2
Number of threads per process: 4
rsqrt: MSE = 1.158186e-04,  Bias = 8.375360e-08
rsqrt: MSE = 1.158186e-04,  Bias = 8.375360e-08
（以下省略）
```

見て分かる通り、Number of processes: 2 となっている。これで、2 プロセス 4 スレッドでの並列計算が行われている事が確認できた。

2.1.6 CUDA 版 FDPS による重力 N 体シミュレーション

FDPS の Ver.9 では、Tree 構造や相互作用リストの作成を全て NVIDIA の GPU 上で行うようにできるようになる予定である．ここでは、開発版の FDPS を使って，NVIDIA の GPU を用いた CUDA 版 FDPS による重力 N 体シミュレーションのサンプルコードの動かし方を説明する．NVIDIA の GPU があり，CUDA がインストールされている環境がある人は以下の手順で実際に動作させることができるはずである．

まず，FDPS の開発版 `src_v9_alpha.tar.gz` を講習会ホームページからダウンロード．

ダウンロードしたディレクトに移動し `src_v9_alpha.tar.gz` を解凍し，`src_v9_alpha/nbody_sample` に移動する．

```
$ tar -xzvf src_v9_alpha.tar.gz
$ cd src_v9_alpha/nbody_sample/
```

`nbody_sample` 内の `Makefile` 内の `use_gpu_walk = yes` とし，`CUDA_HOME` に CUDA がインスト―ルされているディレクトリのパスを，`CUDA_ARCH` に GPU のコンピュータケーパビリティを指定する．

```
use_gpu_walk = yes
CUDA_HOME = /usr/local/cuda
CUDA_ARCH = 86
```

編集後，`make` コマンドを実行する．

```
$ make
```

すると，`nbody_gpu_walk.out` という実行ファイルが生成されるはずなので，それを実行すればよい．

```
$ ./nbody_gpu_walk.out
```

2.2 SPH シミュレーションコード

2.2.1 概要

ここでは、SPH シミュレーションコードを動かす。用意されているコードは、断熱スフィアコラプスの計算を行う。この節でまず行うことは、シリアルコードのコンパイルと実行、出て来た結果の解析である。最後に OpenMP や MPI を利用して、さらにコードを高速化する。

2.2.2 シリアルコード

以下の手順で本コードを使用できる。

- ディレクトリ fdps/FDPS-master/sample/c++/nbodysph に移動
- make を実行
- ジョブの投入
- 結果の解析
- OpenMP/MPI の利用 (オプション)

2.2.2.1 ディレクトリ移動

ディレクトリ fdps/FDPS-master/sample/c++/nbodysph に移動する。

2.2.2.2 make の実行

make コマンドを実行する。

2.2.2.3 計算の実行

まずは、インタラクティブ実行で計算を実行する。これは、生成された実行ファイルの名前をそのまま実行すればよい。

```
$ ./sph.out
```

正しくジョブが開始すると、以下のような表示がされる。

```
     //==================================\\
     ||                                  ||
     || ::::::: ::::::. ::::::. .::::::. ||
     || ::      ::    : ::    : ::       ||
     || ::::::  ::    : ::::::'  `::::. ||
     || ::      ::::::' ::      `.....' ||
     ||     Framework for Developing     ||
     ||        Particle Simulator        ||
     ||     Version 8.0 (2025/09)       ||
     \\==================================//

       Home   : https://github.com/fdps/fdps
       E-mail : fdps-support@mail.jmlab.jp
       Licence: MIT (see, https://github.com/FDPS/FDPS/blob/master/LICENSE)
       Note   : Please cite the following papers.
                - Iwasawa et al. (2016, Publications of the Astronomical Society of
                - Namekata et al. (2018, Publications of the Astronomical Society of

       Copyright (C) 2015
         Masaki Iwasawa, Ataru Tanikawa, Natsuki Hosono,
         Keigo Nitadori, Takayuki Muranushi, Daisuke Namekata,
         Kentaro Nomura, Junichiro Makino and many others
******** FDPS has successfully begun. ********
Directory "result" is successfully made.
This is a sample program of Smoothed Particle Hydrodynamics on FDPS!
# of proc is   1
# of thread is 1
//====================================
(省略)
```

ジョブが終了すると、標準入出力の最後には以下のようなログが出力されるはずである。

```
（省略）
//================================
time = 7.6048679454973855e-01, dt = 3.6609410387440453e-03
step = 86
mass-mass0= 0.0000000000e+00, mom-mom0= 1.0867927164e-06 8.0816730376e-
08 8.0827460925e-08, (eng-eng0)/eng0= -1.8010620283e-05
//================================
//================================
time = 7.6414773558848259e-01, dt = 3.6565140441978295e-03
step = 87
mass-mass0= 0.0000000000e+00, mom-mom0= 1.6179565007e-06 1.0922361483e-
06 1.0922466630e-06, (eng-eng0)/eng0= -2.0803238130e-05
//================================
//================================
time = 7.6780424963268046e-01, dt = 3.6460498768130464e-03
step = 88
mass-mass0= 0.0000000000e+00, mom-mom0= 3.2598127928e-07 1.2074604295e-
07 1.2076044857e-07, (eng-eng0)/eng0= 1.0042612504e-06
//================================
MemoryPool::finalize() is completed!
******** FDPS has successfully finished. ********
```

ここでは、質量や運動量、エネルギーなどの誤差が出力される.

2.2.2.4 結果の解析

ディレクトリ `result` にファイルが出力されている。ファイル名は”00**.dat”(* には数字が入る) となっている。ファイル名は時刻を表す。出力ファイルフォーマットは 1 列目から順に粒子の ID、粒子の質量、位置の x, y, z 座標、粒子の x, y, z 軸方向の速度、密度、内部エネルギー、圧力である。

これは 3 次元の Evrard sphere である。以下のコマンドを実行すれば、横軸に r、縦軸に密度の logscale での図が作成される。

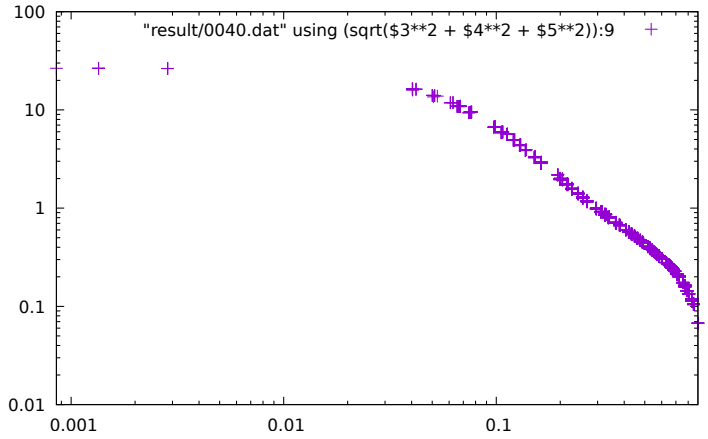

図 2

```
$ gnuplot
$ set logscale
$ plot "result/0040.dat" using (sqrt($3**2 + $4**2 + $5**2)):9
```

正しい答が得られれば、図 2 のような図を描ける。

2.2.3 OpenMP/MPI の利用

OpenMP や MPI を利用する場合を以下に示す。

- OpenMP のみ使用の場合
  - `Makefile` の編集
    - マクロ `CC` に OpenMP 対応の C++ コンパイラを代入する
    - “`CFLAGS += -DPARTICLE_SIMULATOR_THREAD_PARALLEL -fopenmp`” の行のコメントアウトを外す
  - `make` コマンドを実行する。
  - シリアルコードと同じように実行する。正しく実行された場合、以下のように表示されるはずである。

```
This is a sample program of Smoothed Particle Hydrodynamics on FDPS!
# of proc is   1
# of thread is 4
//====================================
```

- OpenMP と MPI の同時使用の場合

- Makefile の編集
  - マクロ CC に MPI 対応の C++ コンパイラを代入する
  - “CFLAGS += -DPARTICLE_SIMULATOR_THREAD_PARALLEL -fopenmp” の行のコメントアウトを外す (インテルコンパイラの場合は-fopenmp を外す)
  - “CFLAGS += -DPARTICLE_SIMULATOR_MPI_PARALLEL” の行のコメントアウトを外す
- make コマンドを実行する。
- システムの MPI 環境の使い方に従って実行する。正しく実行された場合、以下のように表示されるはずである。

```
This is a sample program of Smoothed Particle Hydrodynamics on FDPS!
# of proc is   2
# of thread is 4
//=====================================
```

2.2.4 概要

ここでは、用意された重力 N 体シミュレーションコードを動かしてみよう。このコードは、重力多体系のコールドコラプスを計算する。この節でまず行うことは、シリアルコードのコンパイルと実行、出て来た結果の解析である。次にシリアルコードを Phantom-GRAPE や PIKG を用いて高速化して、その速さを体験しよう。最後に OpenMP や MPI を利用して、さらにコードを高速化する。