

# FDPS ユーザーチュートリアル

FDPS 開発者

## 目 次

1	TODO	4
2	変更記録	5
3	概要	6
4	入門	7
4.1	動作環境	7
4.2	必要なソフトウェア	7
4.2.1	標準機能	7
4.2.1.1	逐次処理	7
4.2.1.2	並列処理	7
4.2.1.2.1	OpenMP	7
4.2.1.2.2	MPI	8
4.2.1.2.3	MPI+OpenMP	8
4.2.2	拡張機能	8
4.2.2.1	Particle Mesh	8
4.3	インストール	8
4.3.1	取得方法	8
4.3.2	ビルド方法	9
4.4	サンプルコードの使用方法	9
4.4.1	重力 $N$ 体シミュレーションコード	9
4.4.1.1	概要	9
4.4.1.2	ディレクトリ移動	10
4.4.1.3	Makefile の編集	10
4.4.1.4	make の実行	10
4.4.1.5	実行	10
4.4.1.6	結果の解析	11
4.4.2	SPH シミュレーションコード	11
4.4.2.1	概要	11
4.4.2.2	ディレクトリ移動	12

4.4.2.3	Makefile の編集 . . . . .	12
4.4.2.4	make の実行 . . . . .	13
4.4.2.5	実行 . . . . .	13
4.4.2.6	結果の解析 . . . . .	13
5	使ってみよう . . . . .	16
5.1	サンプルコードのコンパイルと実行 . . . . .	16
5.2	前提知識 . . . . .	16
5.2.1	Vector 型 . . . . .	16
5.3	固定長 SPH シミュレーションコード . . . . .	16
5.3.1	作業ディレクトリ . . . . .	16
5.3.2	インクルード . . . . .	16
5.3.3	ユーザー定義クラス . . . . .	16
5.3.3.1	概要 . . . . .	16
5.3.3.2	FullParticle 型 . . . . .	16
5.3.3.3	EssentialParticleI 型 . . . . .	16
5.3.3.4	Force 型 . . . . .	16
5.3.3.5	calcForceEpEp 型 . . . . .	16
5.3.4	プログラム本体 . . . . .	16
5.3.4.1	概要 . . . . .	16
5.3.4.2	開始、終了 . . . . .	16
5.3.4.3	初期化 . . . . .	16
5.3.4.3.1	インスタンスの生成 . . . . .	16
5.3.4.3.2	領域クラスの初期化 . . . . .	16
5.3.4.3.3	粒子群クラスの初期化 . . . . .	16
5.3.4.3.4	相互作用ツリークラスの初期化 . . . . .	16
5.3.4.4	ループ . . . . .	16
5.3.4.4.1	領域分割の実行 . . . . .	16
5.3.4.4.2	粒子交換の実行 . . . . .	16
5.3.4.4.3	相互作用計算の実行 . . . . .	16
5.3.5	コンパイル . . . . .	16
5.3.6	実行 . . . . .	16
5.3.7	ログファイル . . . . .	16
5.3.8	可視化 . . . . .	16
5.4	(穴埋め式で)N 体シミュレーションコードを書く . . . . .	16
5.4.1	作業ディレクトリ . . . . .	16
5.4.2	ユーザー定義クラス . . . . .	16
5.4.2.1	概要 . . . . .	16
5.4.2.2	FullParticle 型 . . . . .	16
5.4.2.3	(EssentialParticleI 型) . . . . .	16
5.4.2.4	(EssentialParticleJ 型) . . . . .	16

5.4.2.5	(SuperParticleJ 型)	16
5.4.2.6	Force 型	16
5.4.2.7	calcForceEpEp 型	16
5.4.2.8	(calcForceEpSp 型)	16
5.4.3	プログラム本体	16
5.4.3.1	概要	16
5.4.3.2	全体の初期化	16
5.4.3.3	インスタンスの生成	16
5.4.3.4	(領域クラスの初期化)	16
5.4.3.5	(粒子群クラスの初期化)	16
5.4.3.6	粒子データの入力	16
5.4.3.7	(相互作用ツリークラスの初期化)	16
5.4.3.8	領域分割の実行	16
5.4.3.9	粒子交換の実行	16
5.4.3.10	相互作用計算の実行	16
5.4.3.11	時間積分	16
5.4.3.12	全体の終了	16
5.4.4	ログファイル	16
5.5	(穴埋め式で) 衝突を考慮した $N$ 体シミュレーションコードを書く	16
5.5.1	作業ディレクトリ	16
5.5.2	ユーザー定義クラス	16
5.5.3	ログファイル	16
6	サンプルコード	17
6.1	$N$ 体シミュレーション	17
6.2	SPH シミュレーション	17
6.3	$N$ 体+SPH シミュレーション	17
7	よくあるエラーメッセージ	18
8	ユーザーサポート	19
9	ライセンス	20

# 1 TODO

## 2 変更記録

- 2015/01/25 作成

### 3 概要

本節では、Framework for Developing Particle Simulator (FDPS) の概要について述べる。FDPS は粒子シミュレーションのコード開発を支援するフレームワークである。FDPS が行うのは、計算コストの最も大きな粒子間相互作用の計算と、粒子間相互作用の計算のコストを負荷分散するための処理である。これらはマルチプロセス、マルチスレッドで並列に処理することができる。比較的計算コストが小さく、並列処理を必要としない処理 (粒子の軌道計算など) はユーザーが行う。

FDPS が対応している座標系は、2次元直交座標系と3次元直交座標系である。また、境界条件としては、開放境界条件と周期境界条件に対応している。周期境界条件の場合、 $x$ 、 $y$ 、 $z$  軸方向の任意の組み合わせの周期境界条件を課することができる。

ユーザーは粒子間相互作用の形を定義する必要がある。定義できる粒子間相互作用の形には様々なものがある。粒子間相互作用の形を大きく分けると2種類あり、1つは長距離力、もう1つは短距離力である。この2つの力は、遠くの複数の粒子からの作用を1つの超粒子からの作用にまとめるか (長距離力)、まとめないか (短距離力) という基準でもって分類される。

長距離力には、小分類があり、無限遠に存在する粒子からの力も計算するカットオフなし長距離力と、ある距離以上離れた粒子からの力は計算しないカットオフあり長距離力がある。前者は開放境界条件下における重力やクーロン力に対して、後者は周期境界条件下の重力やクーロン力に使うことができる。後者のためには Particle Mesh 法などが必要となるが、これは FDPS の拡張機能として用意されている。

短距離力には、小分類が4つ存在する。短距離力の場合、粒子はある距離より離れた粒子からの作用は受けない。すなわち必ずカットオフが存在する。このカットオフ長の決め方によって、小分類がなされる。すなわち、全粒子のカットオフ長が等しいコンスタントカーネル、カットオフ長が作用を受ける粒子固有の性質で決まるギャザーカーネル、カットオフ長が作用を与える粒子固有の性質で決まるスキッタカーネル、カットオフ長が作用を受ける粒子と作用を与える粒子の両方の性質で決まるシンメトリックカーネルである。コンスタントカーネルは分子動力学における LJ 力に適用でき、その他のカーネルは SPH などに適用できる。

ユーザーは、粒子間相互作用や粒子の軌道積分などを、C++ 言語を用いて記述する。将来的には Fortran 言語でも記述できるように検討する。

## 4 入門

本節では、FDPS の入門について記述する。FDPS を使用する環境、FDPS に必要なソフトウェア、FDPS のインストール方法、サンプルコードの使用方法、の順で記述する。

### 4.1 動作環境

FDPS は Linux, Mac OS X, Windows などの OS 上で動作する。

### 4.2 必要なソフトウェア

本節では、FDPS を使用する際に必要となるソフトウェアを記述する。まず標準機能を用いるのに必要なソフトウェア、次に拡張機能を用いるのに必要なソフトウェアを記述する。

#### 4.2.1 標準機能

本節では、FDPS の標準機能のみを使用する際に必要なソフトウェアを記述する。最初に逐次処理機能のみを用いる場合（並列処理機能を用いない場合）に必要なソフトウェアを記述する。次に並列処理機能を用いる場合に必要なソフトウェアを記述する。

##### 4.2.1.1 逐次処理

逐次処理の場合に必要なソフトウェアは以下の通りである。

- make
- C++ コンパイラ (gcc バージョン **x.x.x** 以降なら確実)

##### 4.2.1.2 並列処理

本節では、FDPS の並列処理機能を用いる際に必要なソフトウェアを記述する。まず、OpenMP を使用する際に必要なソフトウェア、次に MPI を使用する際に必要なソフトウェア、最後に OpenMP と MPI を同時に使用する際に必要なソフトウェアを記述する。

###### 4.2.1.2.1 OpenMP

OpenMP を使用する際に必要なソフトウェアは以下の通り。

- make
- OpenMP 対応の C++ コンパイラ (gcc version **x.x.x** 以降なら確実)

#### 4.2.1.2.2 MPI

MPI を使用する際に必要なソフトウェアは以下の通り。

- make
- MPI version 1.x 対応の C++ コンパイラ (gcc version x.x.x 以降なら確実)

#### 4.2.1.2.3 MPI+OpenMP

MPI と OpenMP を同時に使用する際に必要なソフトウェアは以下の通り。

- make
- MPI version 1.x と OpenMP に対応の C++ コンパイラ (gcc version x.x.x 以降なら確実)

### 4.2.2 拡張機能

本節では、FDPS の拡張機能を使用する際に必要なソフトウェアについて述べる。FDPS の拡張機能には Particle Mesh がある。以下では Particle Mesh を使用する際に必要なソフトウェアを述べる。

#### 4.2.2.1 Particle Mesh

Particle Mesh を使用する際に必要なソフトウェアは以下の通りである。

- make
- MPI version 1.x と OpenMP に対応の C++ コンパイラ (gcc version x.x.x 以降なら確実)
- FFTW 3.3 以降

### 4.3 インストール

本節では、FDPS のインストールについて述べる。取得方法、ビルド方法について述べる。

#### 4.3.1 取得方法

以下の方法のいずれかで FDPS 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

- Subversion を用いる場合：以下のコマンドを実行するとディレクトリ trunk のしたを Subversion レポジトリとして使用できる

```
$ svn co --depth empty https://github.com/FDPS/FDPS
$ cd FDPS
$ svn up trunk
```

- 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
```

### 4.3.2 ビルド方法

configure などをする必要はない。

## 4.4 サンプルコードの使用方法

本節ではサンプルコードの使用方法について記述する。サンプルコードには重力  $N$  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がある。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の使用について記述する。サンプルコードは拡張機能を使用していない。

### 4.4.1 重力 $N$ 体シミュレーションコード

#### 4.4.1.1 概要

以下の手順で本コードを使用できる。

- ディレクトリ  $\$(FDPS)/sample/nbody$  に移動
- カレントディレクトリにある Makefile を編集
- コマンドライン上で make を実行
- nbody.out ファイルの実行
- 結果の解析

これ以後、ディレクトリ  $\$(FDPS)$  は FDPS の最も上の階層のディレクトリを指す。

#### 4.4.1.2 ディレクトリ移動

ディレクトリ\$(FDPS)/sample/nbody に移動に移動する。

#### 4.4.1.3 Makefile の編集

Makefile の編集項目は以下の通りである。OpenMP と MPI を使用するかどうかで編集方法が変わることに注意。

- OpenMP も MPI も使用しない場合
  - マクロ CC に C++コンパイラを代入する
- OpenMP のみ使用の場合
  - 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 (インテルコンパイラの場合は-fopenmp を外す)
- MPI のみ使用の場合
  - マクロ CC に MPIC++コンパイラを代入する
  - "CFLAGS += -DPARTICLE\_SIMULATOR\_MPI\_PARALLEL" の行のコメントアウトを外す
- OpenMP と MPI の同時使用の場合
  - 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

#### 4.4.1.4 make の実行

make コマンドを実行する。

#### 4.4.1.5 実行

実行方法は以下の通りである。

- MPI を使用しない場合、コマンドライン上で以下のコマンドを実行する

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

- MPI を使用する場合、コマンドライン上で以下のコマンドを実行する

```
$ MPIRUN -np NPROC ./nbody.out
```

ここで、“MPIRUN”には `mpirun` や `mpiexec` などが、“NPROC”には使用する MPI プロセスの数が入る。

正しく終了すると、標準エラー出力に

```
***** FDPS has successfully finished. *****
```

と表示される。

#### 4.4.1.6 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“000x.dat”ができています。x 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)。プロセス数によって時刻 9 の結果は大きく変わるので、時刻 9 における粒子分布が図 1 と異なっても気にする必要はない。

粒子数を 10000 個にして計算を行いたい場合には以下のように実行すればよい (MPI を使用しない場合)。

```
$ ./nbody.out -N 10000
```

### 4.4.2 SPH シミュレーションコード

#### 4.4.2.1 概要

以下の手順で本コードを使用できる。

- ディレクトリ `$(FDPS)/sample/sph` に移動
- カレントディレクトリにある `Makefile` を編集 (後述)

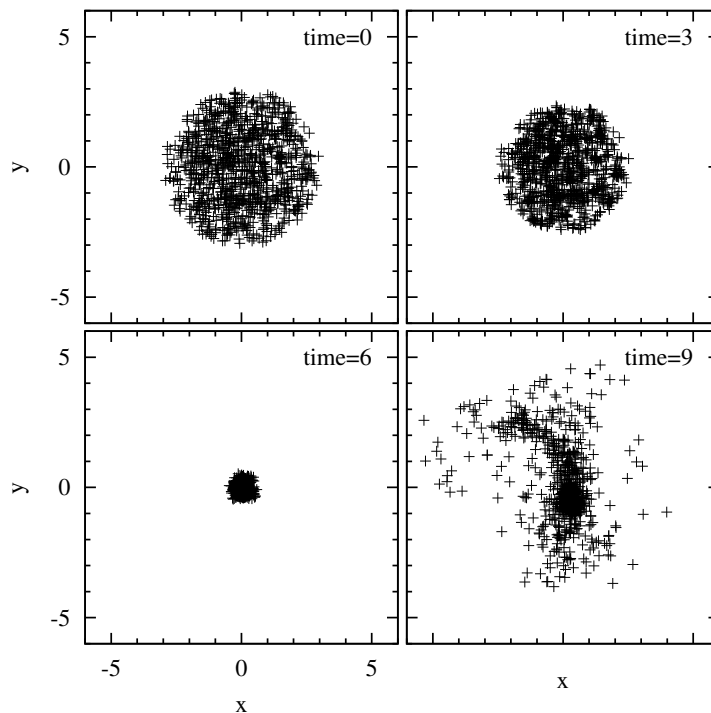


図 1:

- コマンドライン上で make を実行
- sph.out ファイルの実行 (後述)
- 結果の解析 (後述)

#### 4.4.2.2 ディレクトリ移動

ディレクトリ\$(FDPS)/sample/nbody に移動に移動する。

#### 4.4.2.3 Makefile の編集

Makefile の編集項目は以下の通りである。OpenMP と MPI を使用するかどうかで編集方法が変わることに注意。

- OpenMP も MPI も使用しない場合
  - マクロ CC に C++コンパイラを代入する
- OpenMP のみ使用の場合
  - 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 (インテルコンパイラの場合は-fopenmp を外す)
- MPI のみ使用の場合
  - マクロ CC に MPIC++ コンパイラを代入する
  - ”CFLAGS += -DPARTICLE\_SIMULATOR\_MPI\_PARALLEL” の行のコメントアウトを外す
- OpenMP と MPI の同時使用の場合
  - 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

#### 4.4.2.4 make の実行

make コマンドを実行する。

#### 4.4.2.5 実行

実行方法は以下の通りである。

- MPI を使用しない場合、コマンドライン上で以下のコマンドを実行する

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

- MPI を使用する場合、コマンドライン上で以下のコマンドを実行する

```
$ MPIRUN -np NPROC ./sph.out
```

ここで、”MPIRUN”には mpirun や mpiexec などが、”NPROC”には使用する MPI プロセスの数が入る。

#### 4.4.2.6 結果の解析

実行するとディレクトリ result にファイルが出力されている。ファイル名は”00xx.dat”(x 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。

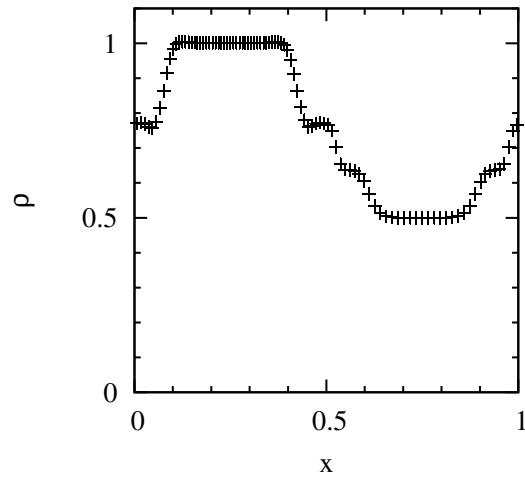


図 2:

これは 3 次元の衝撃波管問題である。コマンドライン上で以下のコマンドを実行すれば、横軸に粒子の  $x$  座標、縦軸に粒子の密度をプロットできる (時刻は 50)。

```
$ gnuplot  
$ plot "result/0040.dat" using 3:9
```

正しい答が得られれば、図 2 のような図を描ける。



## 5 使ってみよう

### 5.1 サンプルコードのコンパイルと実行

### 5.2 前提知識

#### 5.2.1 Vector 型

### 5.3 固定長 SPH シミュレーションコード

#### 5.3.1 作業ディレクトリ

#### 5.3.2 インクルード

#### 5.3.3 ユーザー定義クラス

##### 5.3.3.1 概要

##### 5.3.3.2 FullParticle 型

##### 5.3.3.3 EssentialParticleI 型

##### 5.3.3.4 Force 型

##### 5.3.3.5 calcForceEpEp 型

#### 5.3.4 プログラム本体

##### 5.3.4.1 概要

##### 5.3.4.2 開始、終了

##### 5.3.4.3 初期化

###### 5.3.4.3.1 インスタンスの生成

###### 5.3.4.3.2 領域クラスの初期化

###### 5.3.4.3.3 粒子群クラスの初期化

###### 5.3.4.3.4 相互作用ツリークラスの初期化

##### 5.3.4.4 ループ

###### 5.3.4.4.1 領域分割の実行

###### 5.3.4.4.2 粒子交換の実行

###### 5.3.4.4.3 相互作用計算の実行

#### 5.3.5 コンパイル

#### 5.3.6 実行

#### 5.3.7 ログファイル

#### 5.3.8 可視化

### 5.4 (穴埋め式で)N 体シミュレーションコードを書く



## 6 サンプルコード

### 6.1 $N$ 体シミュレーション

### 6.2 SPH シミュレーション

### 6.3 $N$ 体+SPH シミュレーション

## 7 よくあるエラーメッセージ

## 8 ユーザーサポート

## 9 ライセンス

MIT ライセンスに準ずる。Particle Mesh は GreeM の MIT ライセンスである。