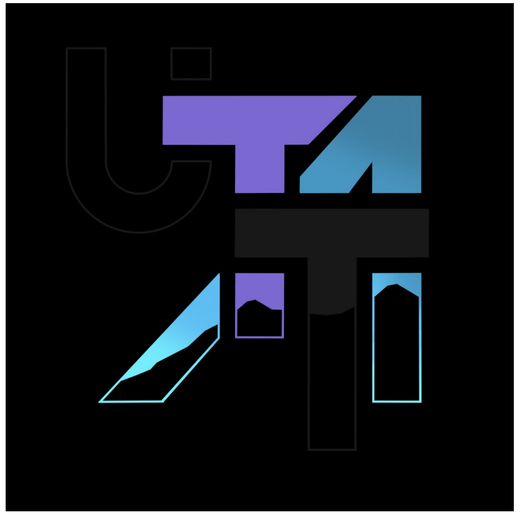
HPC-Phys 勉強会

GPUを用いたAMR法による自己重力流体計算コードの開発

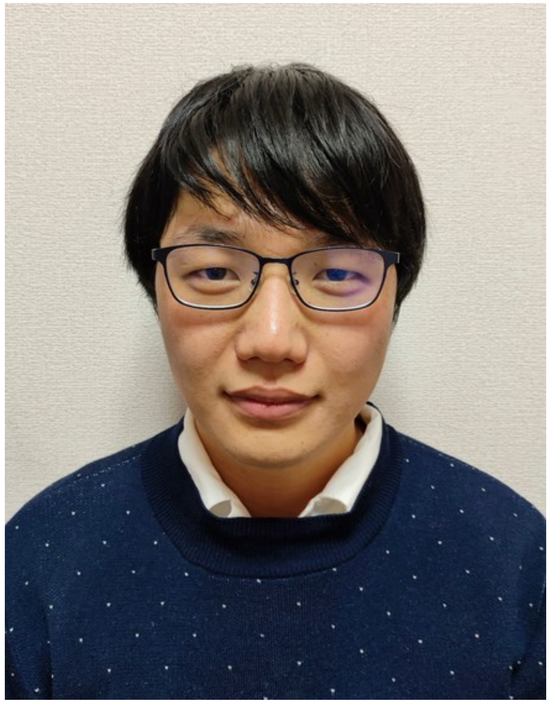
福島 肇 (筑波大学)

共同研究者: 松本倫明(法政大学)

筑波大学
計算科学研究センター
Center for Computational Sciences



自己紹介:　福島　肇　(ふくしま　はじめ)



研究:

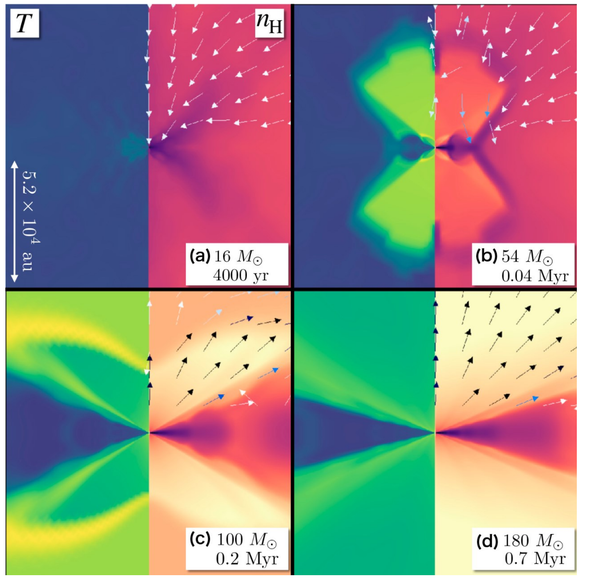
宇宙理論 (星・星団形成、銀河形成)

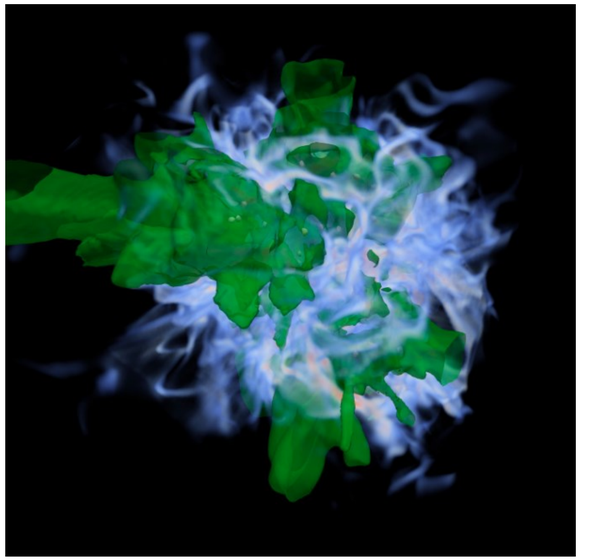
職歴など:

2016-2019　京都大学 (2019.03 博士取得)

2019-　　筑波大学 (研究員・現職)

より詳しくは[こちら]

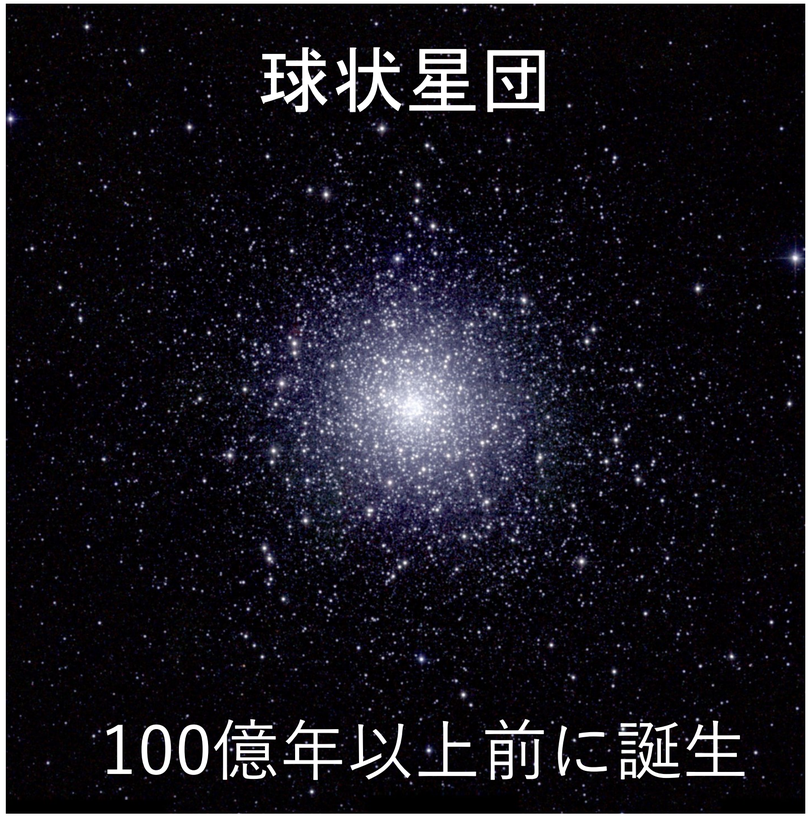




よく使う言語: CUDA, C++, Fortran, Julia, pythonなど

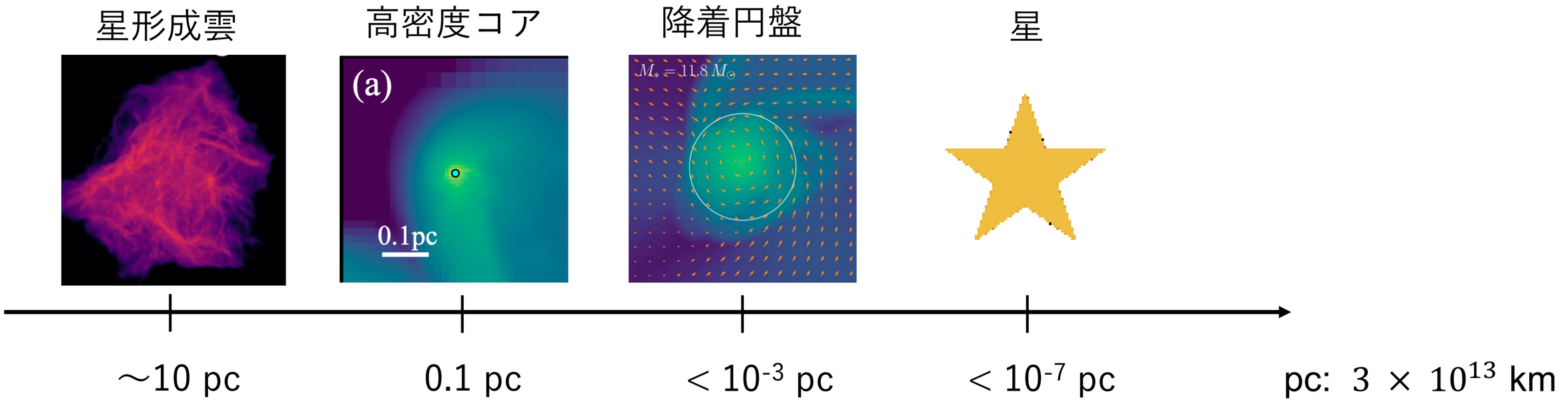
背景: 星団の誕生と種類

星の大部分が星団として誕生



雲から星々がどのように誕生するか、シミュレーションで解明する

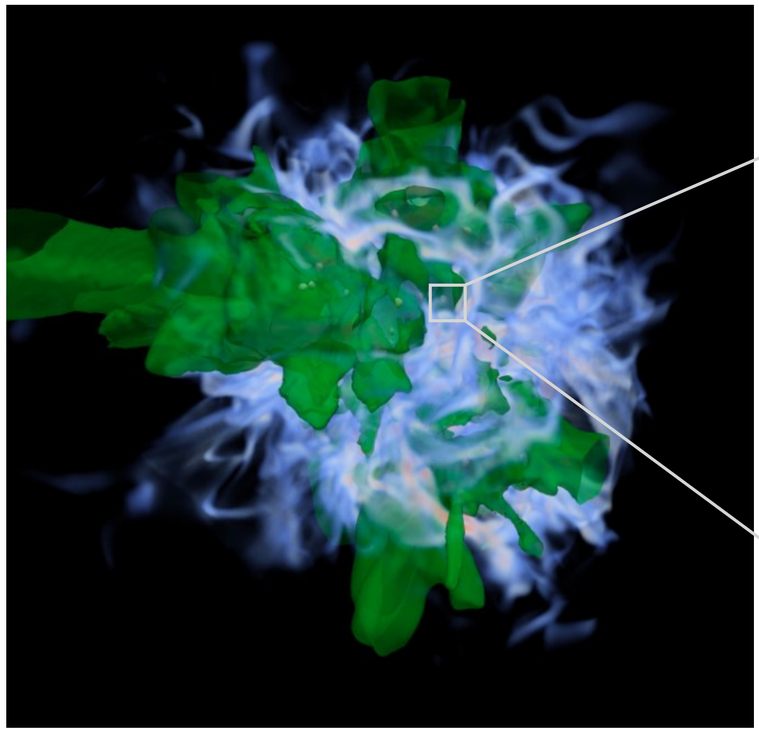
星・星団形成における物理スケール



物理スケールが数桁以上変化する現象を同時に計算する必要がある。

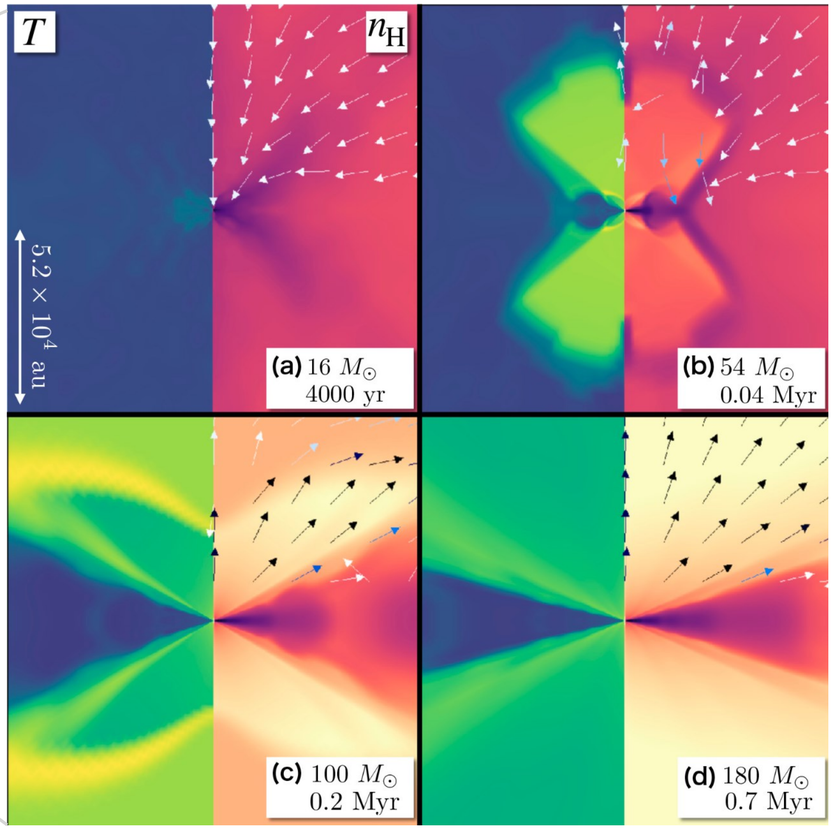
開発の動機: 星・星団形成

星団: 数~100pc



(HF+2020)

降着円盤: 10auくらい?



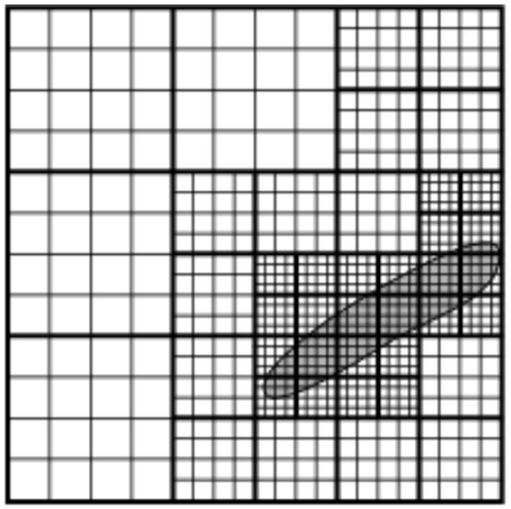
(HF+2020)

星団スケールと星へのガス降着を同時に計算したい

円盤の性質などより議論できるようになるはず

適合格子細分化 (adaptive mesh refinement: AMR)法

AMR格子構造



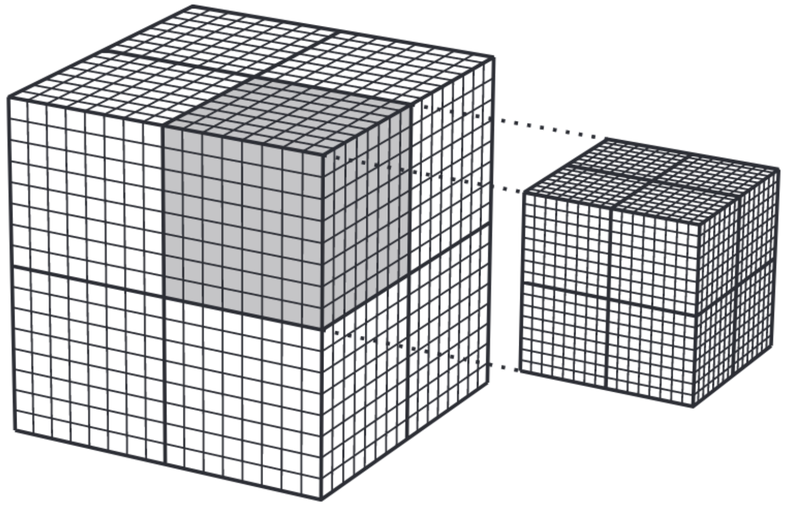
(Matumoto2007)

解像度が必要な領域に選択的に格子を設置する

解像条件:

例) ジーンズ長を解像するセル数を指定する

セルの実際の構造



計算ではグリッド(8x8x8のセルの集合体)
単位にidを割り当てて管理する

星団形成条件の解明

計算コード:

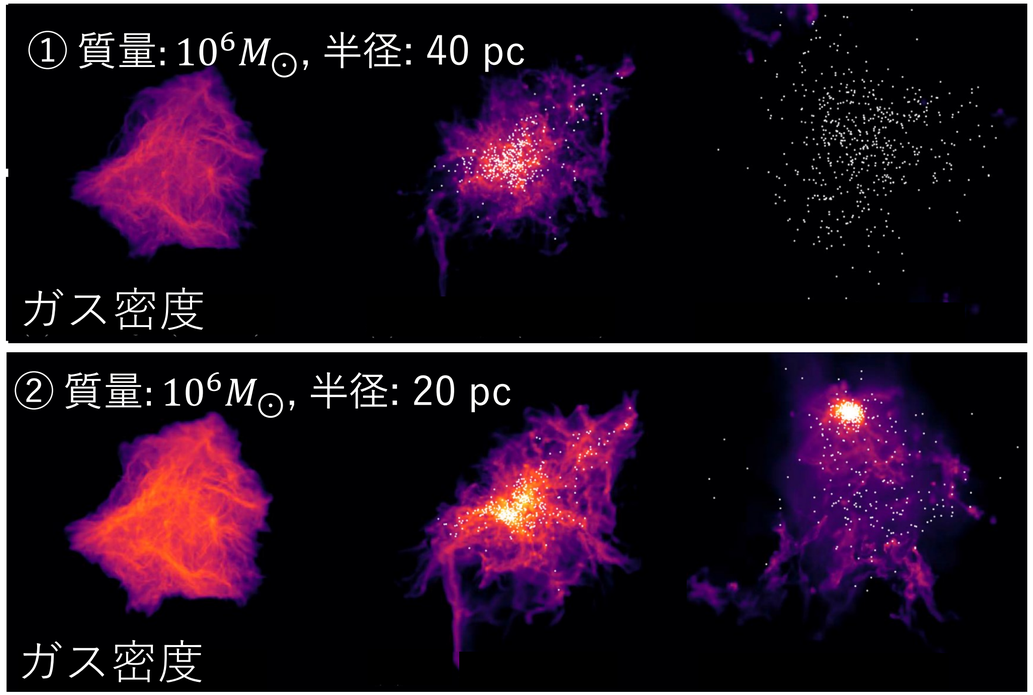
SFUMATO-M1

- 適合格子細分化法
- 流体力学
- 自己重力
- 輻射輸送
- 非平衡化学

SFUMATOコード(Matsumoto 2007)
に輻射輸送&化学反応計算を実装

(Fukushima & Yajima 2021)

輻射流体シミュレーション



$M_\odot$: 太陽質量

pc: 3×10^{13} km

(Fukushima & Yajima 21)

CPUで計算を実行

解像度は0.01pc程度なので、星へのガス降着は計算できなかった

高解像度計算を実施可能な計算コードの開発が必要だった

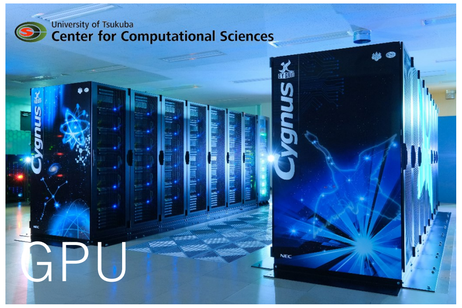
GPU化への取り組み: GPU計算開発の動機

筑波大学計算科学研究センター関連のスーパーコンピュータ:

2022 2023 2024 2025

Oakforest-PACS

MIYABI

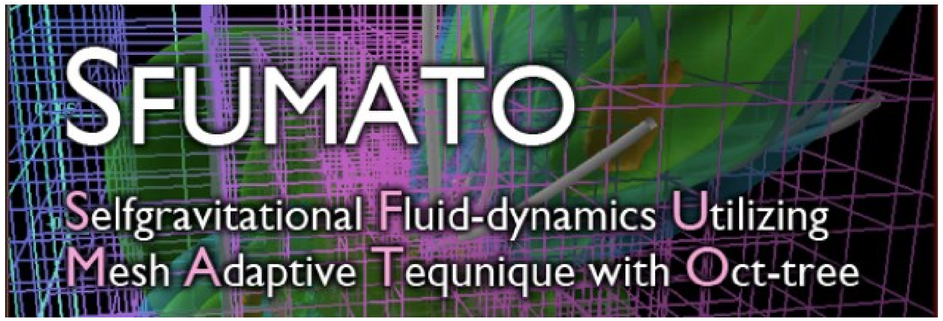


Cygnus

Pegasus

開発開始時(2022年頃)、既にGPUをメインとするスパコンに囲まれていた

SFUMATOのGPU化

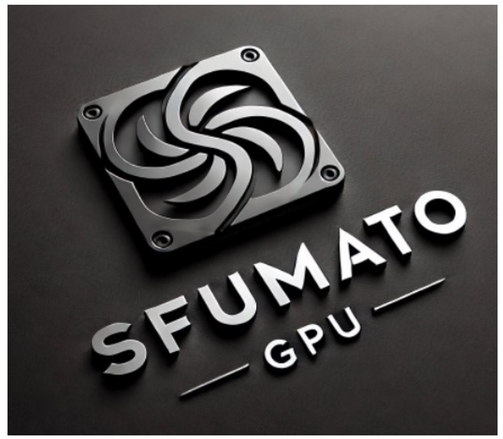


(Matsumoto 2007)

言語: Fortran90

並列化: MPI

CPUのみ



(このロゴはChatGPTで作成)

言語: CUDA / HIP

並列化: MPI (+openmp ?)

CPU + GPU (NVIDIA or AMD)

CUDA(HIP)を選択した理由:

- メモリ管理のしやすさ
- Fortranからの脱却
- CUDAしか使えない..

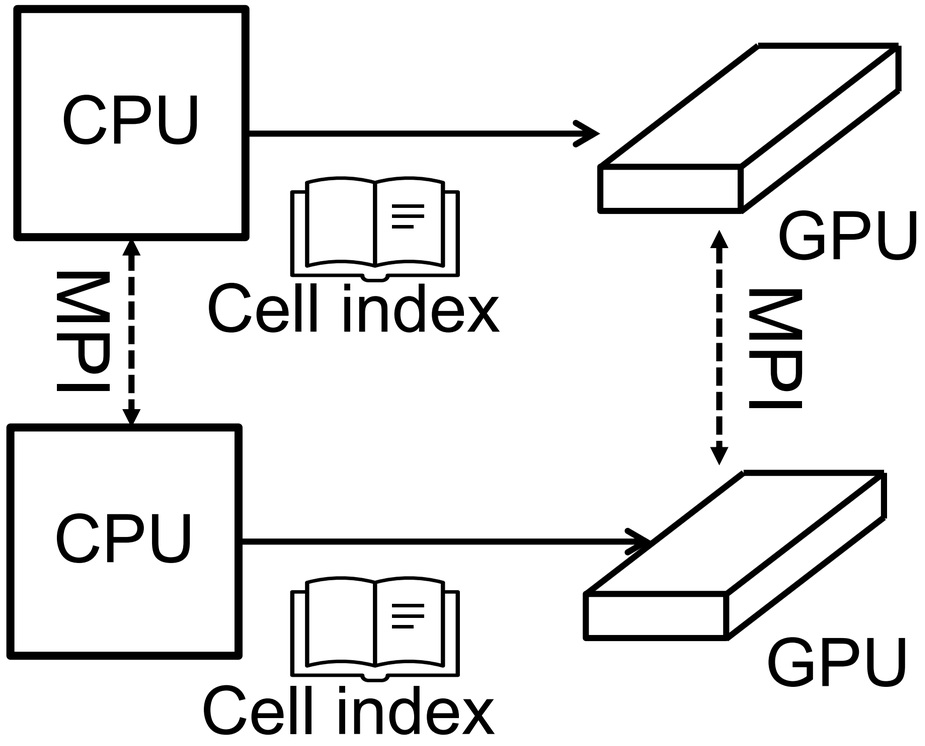
SFUMATO GPUの開発状況:

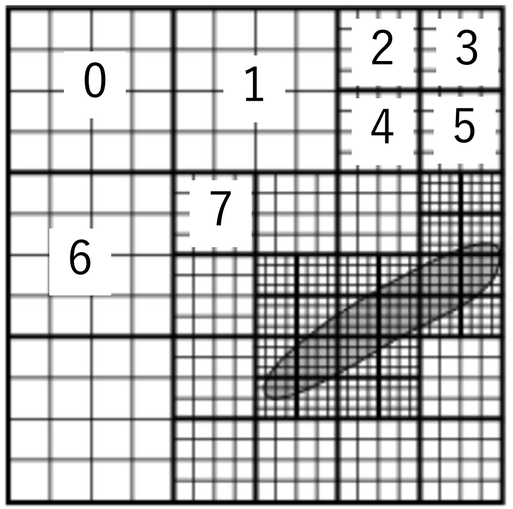
AMR格子生成	<- 完了
流体計算	<- 完了
自己重力	<- 完了
シンク粒子	<- 完了
非平衡化学＋輻射輸送	<- ほぼ完了

計算コードの設計

CPU: グリッドIDの管理人

GPU: セル情報は全てGPUに





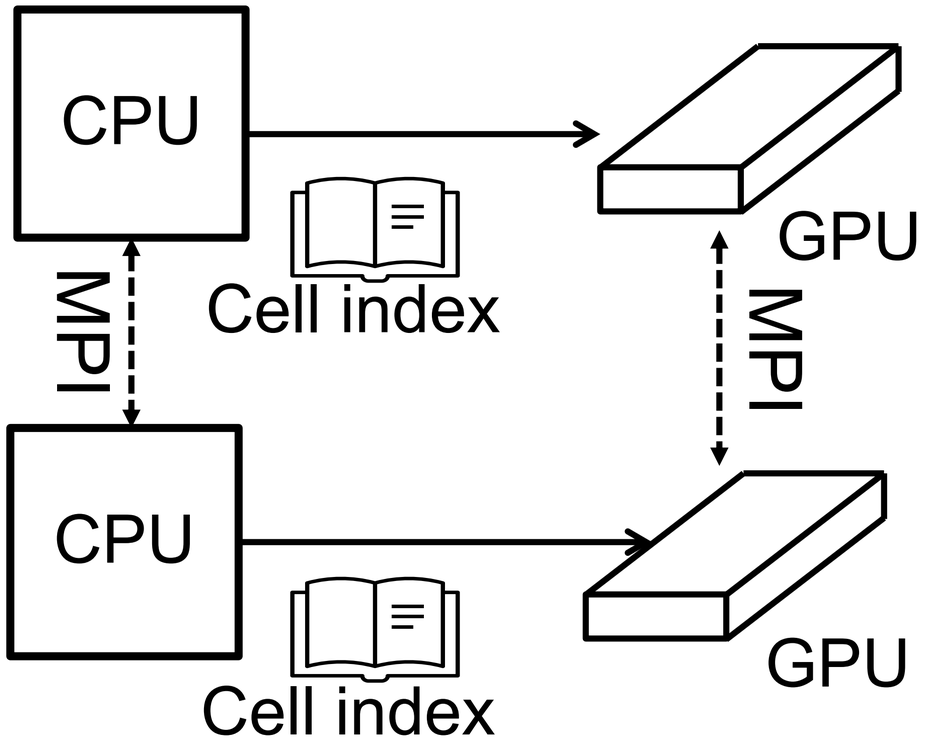
各MPIプロセスはグリッドのID及び、各ランクのグリッドの隣接関係、親子関係を記憶しておく。

GPUはグリッドIDから指定された作業を実施する。

計算コードの設計

CPU: グリッドIDの管理人

CPU: セル情報は全てGPUに



セルの情報は全てGPU上におく(多くて80GB)

置ききれない場合は、Unifined memory (cudaMallocManaged)としてhostに置く設定も追加

cudMemcpyAsyncとの組み合わせでもう少しまじめに改良してもいいのかもしれない。

方程式

流体の式

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0,$$

$$\frac{\partial (\rho \boldsymbol{v})}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v}) + \nabla P = \rho (\boldsymbol{g} + \boldsymbol{f}),$$

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot [(\rho E + P)\,\mathbf{v}] = \rho (\mathbf{g} + \mathbf{f}) \cdot \mathbf{v} + \Gamma - \Lambda,$$

+ 自己重力 (マルチグリッド法)

+ 非平衡化学

+ 輻射輸送

ρ: 密度, v: 速度, P: 圧力, E:全エネルギー, g: 重力, f: 輻射圧

Γ, Λ: 加熱・冷却関数

$$E = \frac{|\mathbf{v}|^2}{2} + (\gamma - 1)^{-1} \frac{P}{\rho},$$

流体計算の時間発展: 袖の値

グリッドIDからデータの送り主と転送先を特定し
リスト化 (CPU)

転送データを準備 (GPU)

データ転送 (MPI)

袖領域に値を代入 (GPU)

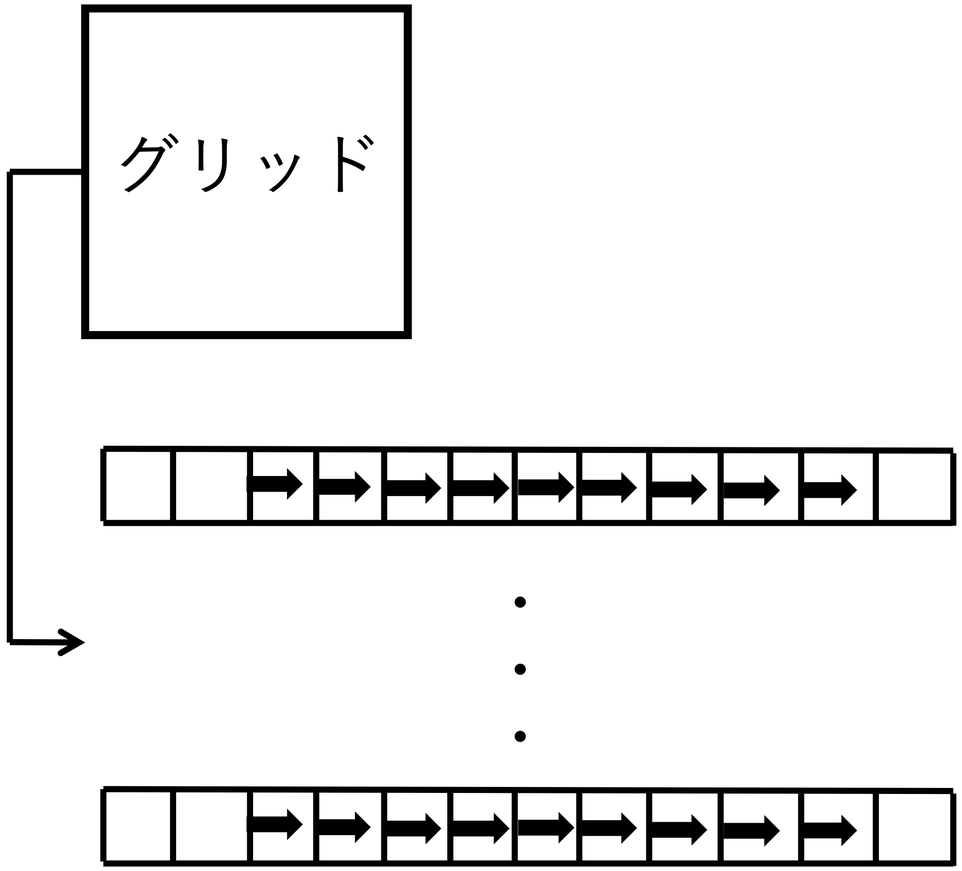
MPIの関数も多くは使える

```
MPI_Isend(h_Bufs[rank], nsend_array[rank], get_mpi_datatype(), rank, tags, MPI_COMM_WORLD, &reqs[rank]);
MPI_Irecv(h_Bufd[rank], nrecv_array[rank], get_mpi_datatype(), rank, tags, MPI_COMM_WORLD, &reqd[rank]);
```

(使えない? MPI_Iallgathervとか, Open MPIのページより)

流体計算の時間発展: 時間発展

x, y, zの３方向について移流を計算する



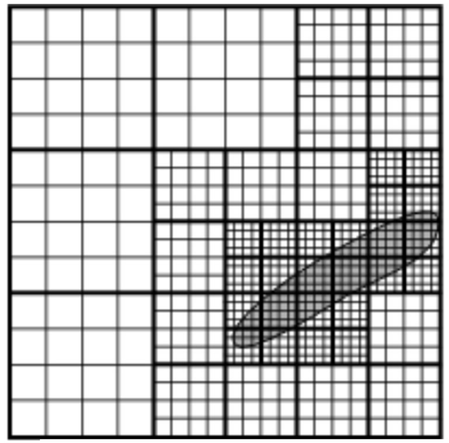
グリッドの数 ×グリッドセル数^2(64)の並列数で実行

１次元方向に順次計算することで流束計算の結果を使い回すことが可能

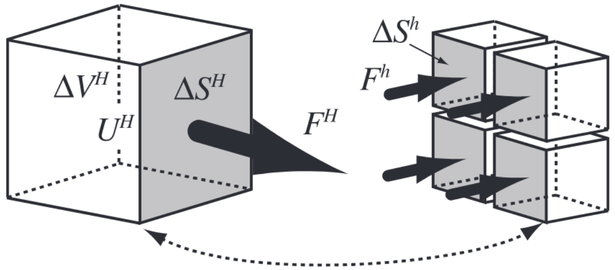
セルごとの実行プラス+shared memoryの使用も検討したが、あまり有効性を見出せなかったのでこの方式を採用した

流体計算の時間発展：流束の補正

格子構造



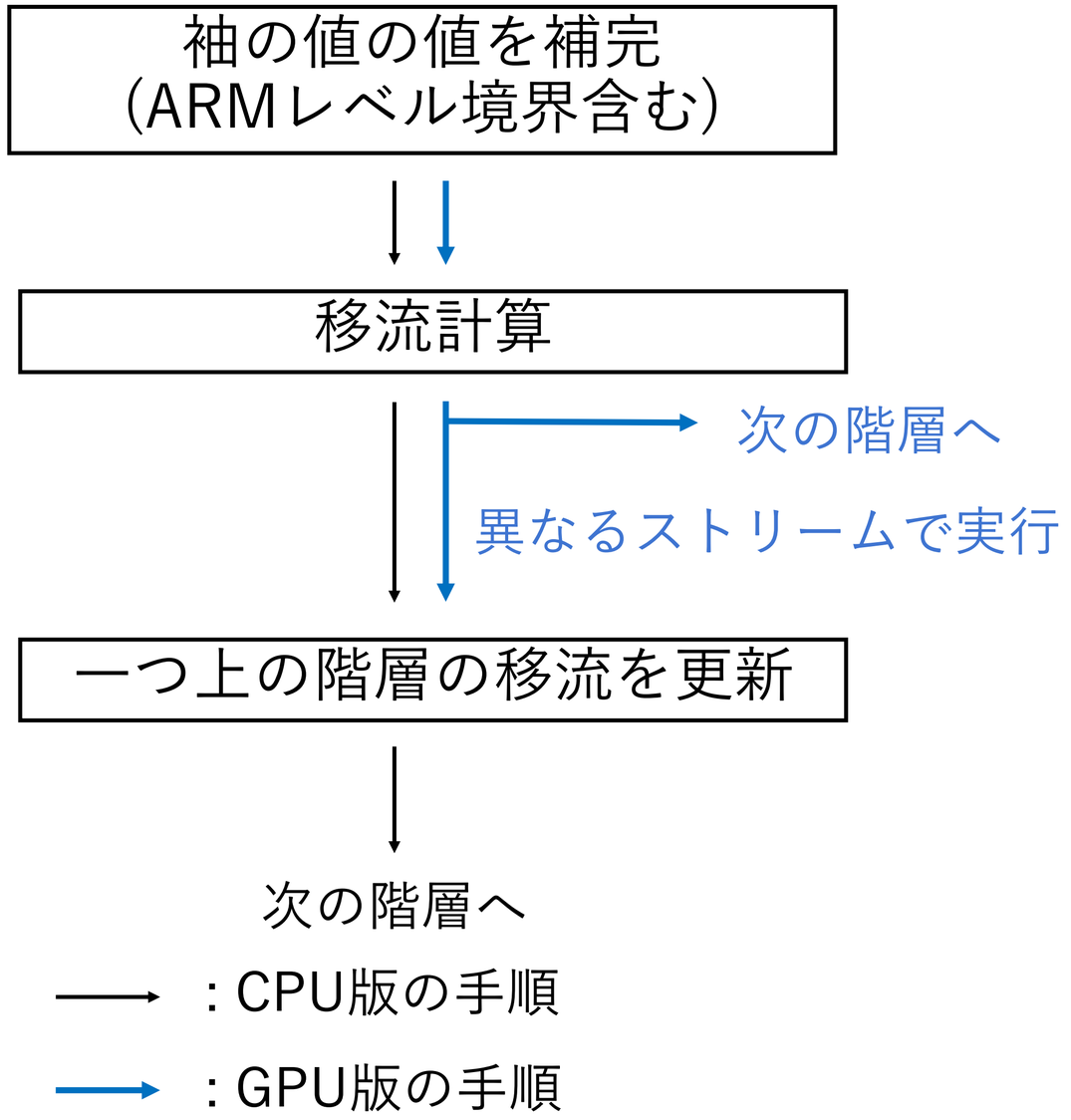
流束補正の様子



(Matsumoto 2007)

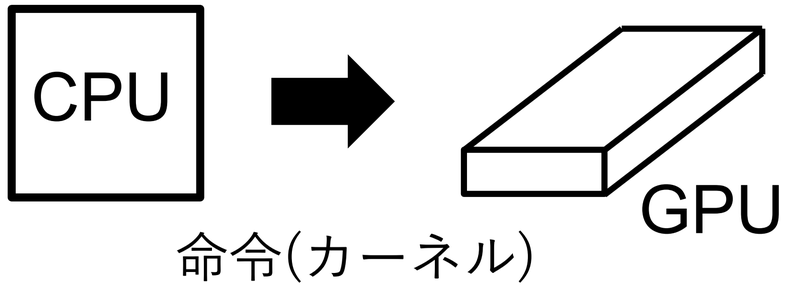
粗密境界での流束の補正が必要となる

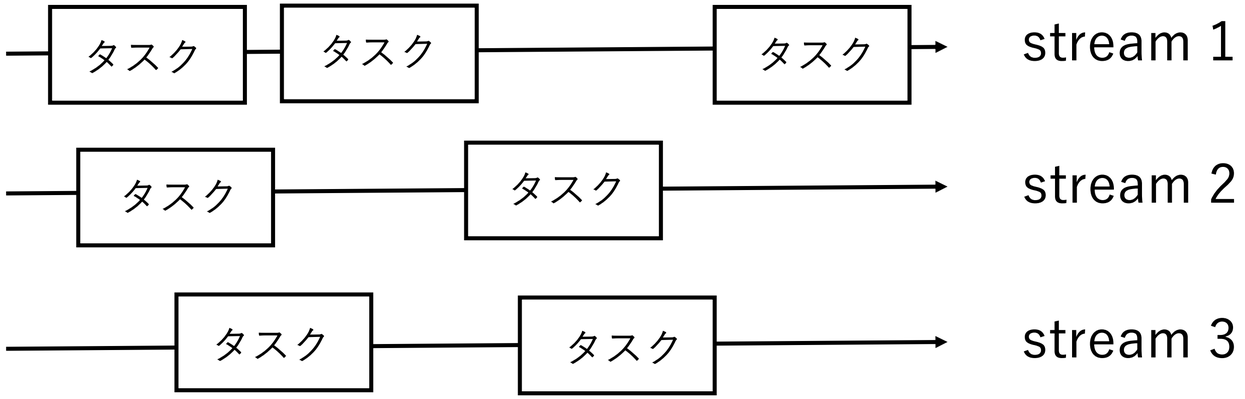
例) 流体移流の手順



GPU化において面倒なところ(初心者的視点)

・StreamとeventのCPU管理





・StreamとMPI通信の併用

Stream上でのタスクの終了を確認しつつ、MPI通信を行う。

・カーネル中でif文を(あまり)使えない

GPU化において面倒なところ(初心者的視点)

- 開発環境の確保 (開発を最も困難にしている点)

  GPUが高価なため個人が簡単に購入できる域を超えかかっている

  研究室以上の単位での環境の整備が必須:

  例) 自分の場合 (テスト計算実行)

  研究室の計算機 (GPU2枚搭載機 × 3 台)
  科研費等で購入したGPU 2 枚(NVIDIAとAMD)

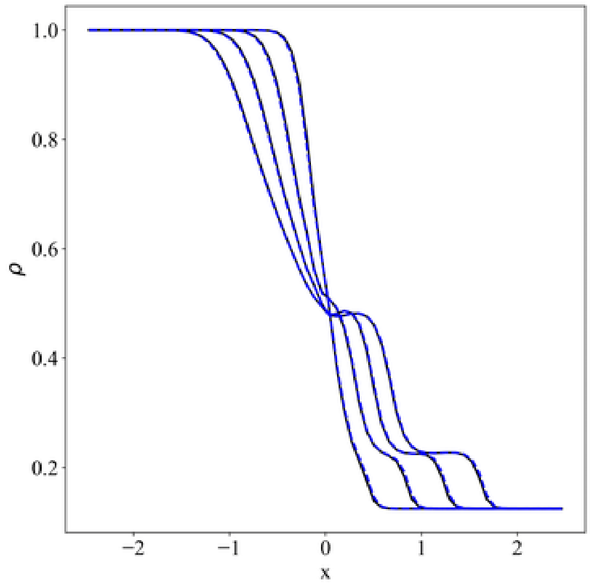
  CfCAのGPUクラスター

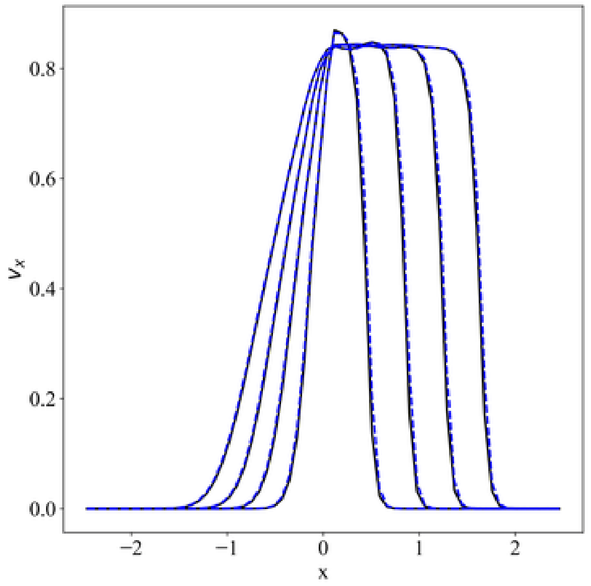
  Cygnus & Pegasus (長時間計算)

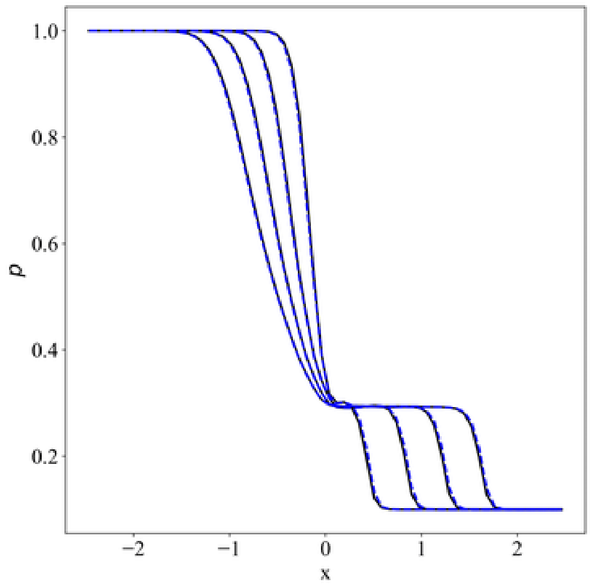
- CUDAの日本語テキストがあまりない（少し古い）

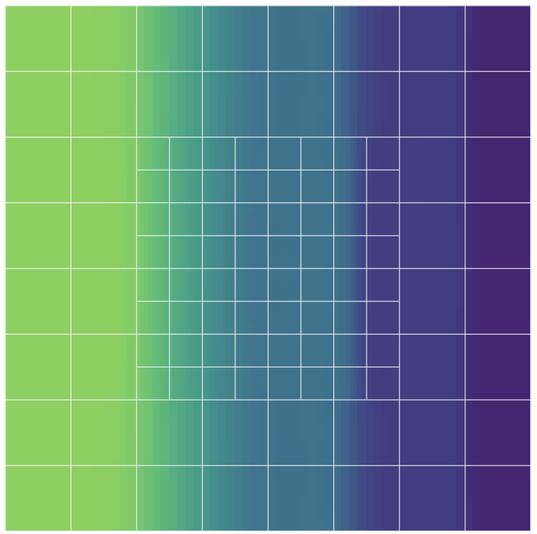
計算の様子

衝撃波管問題



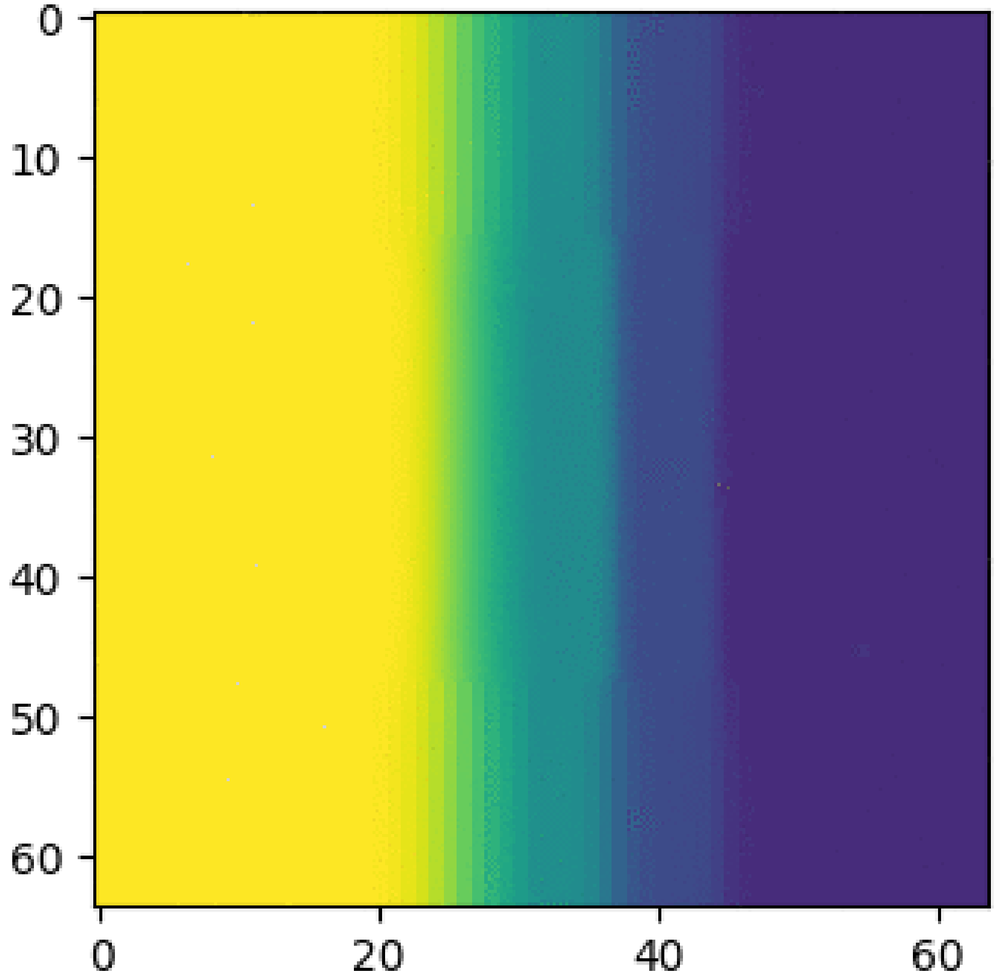




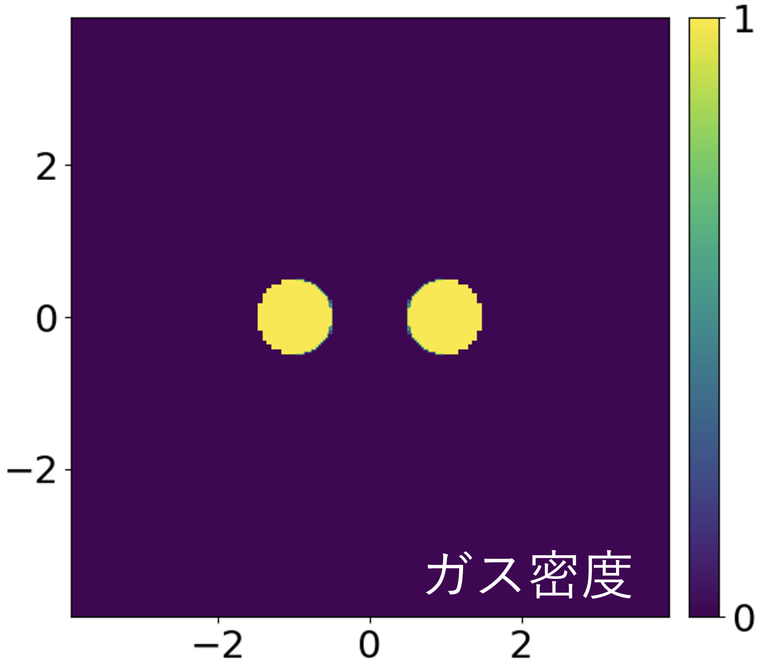


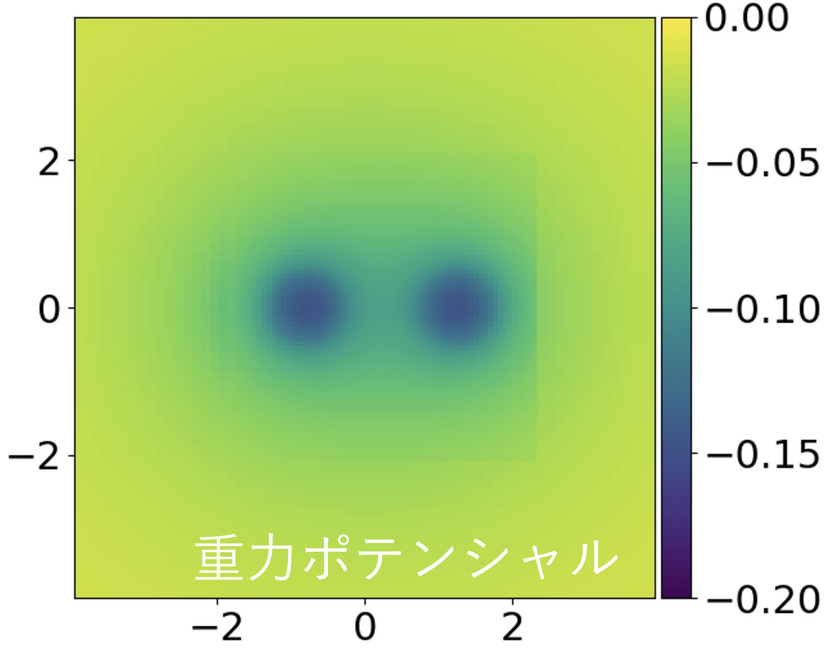
ρ

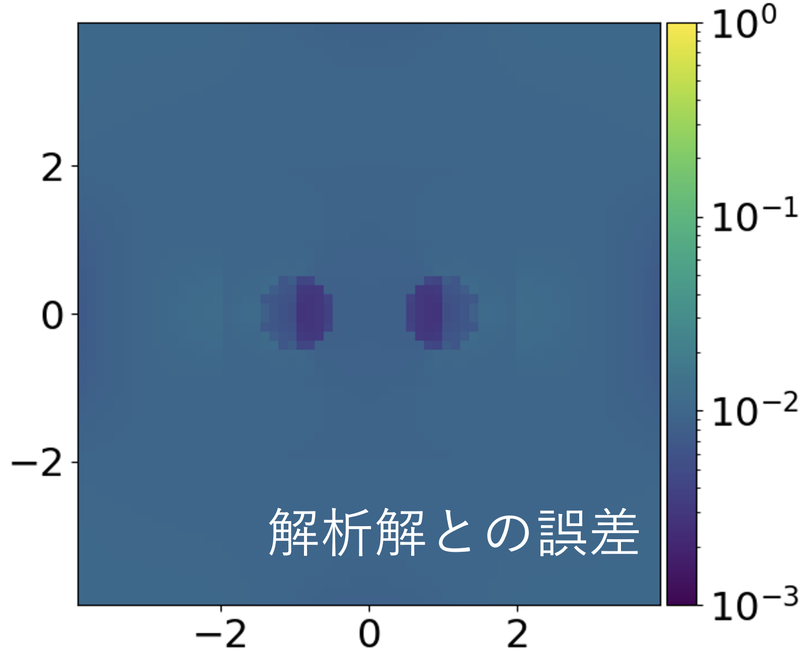
密度



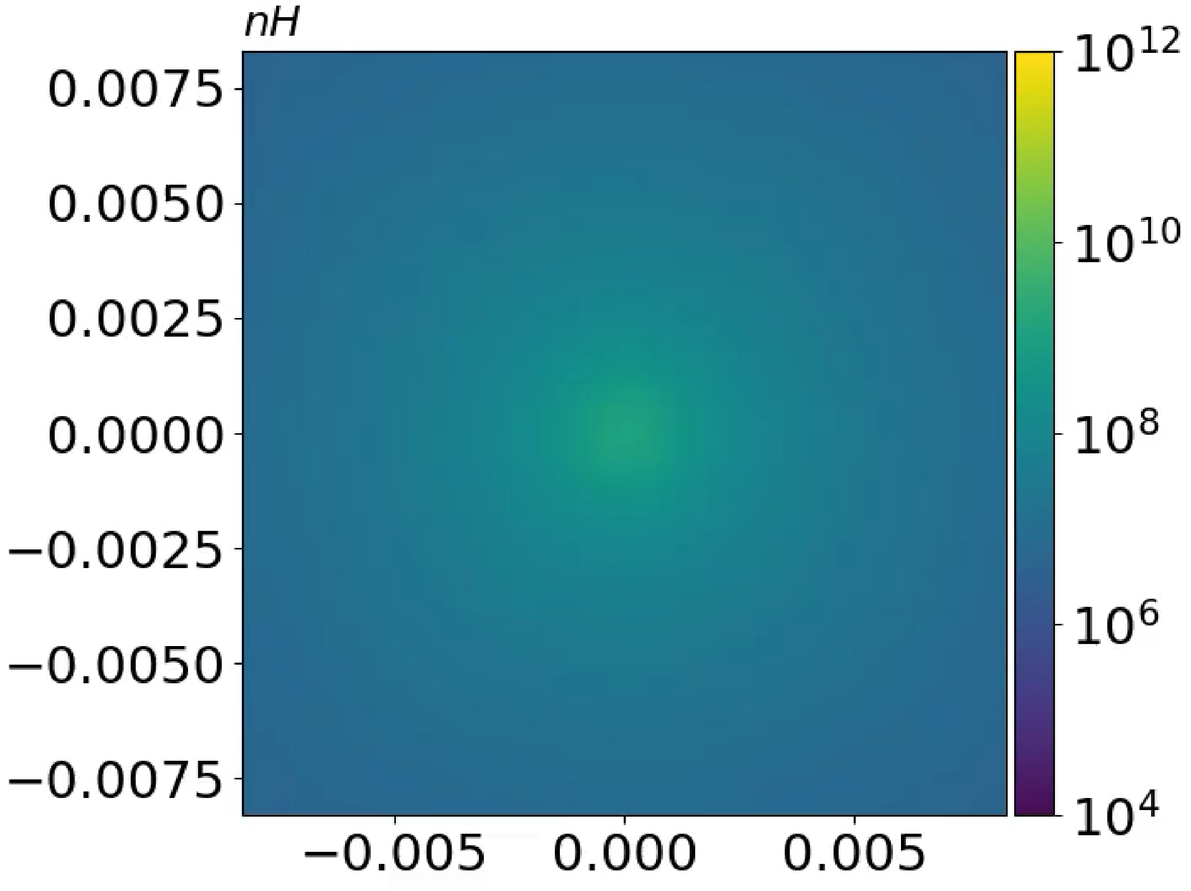
計算の様子　自己重力計算

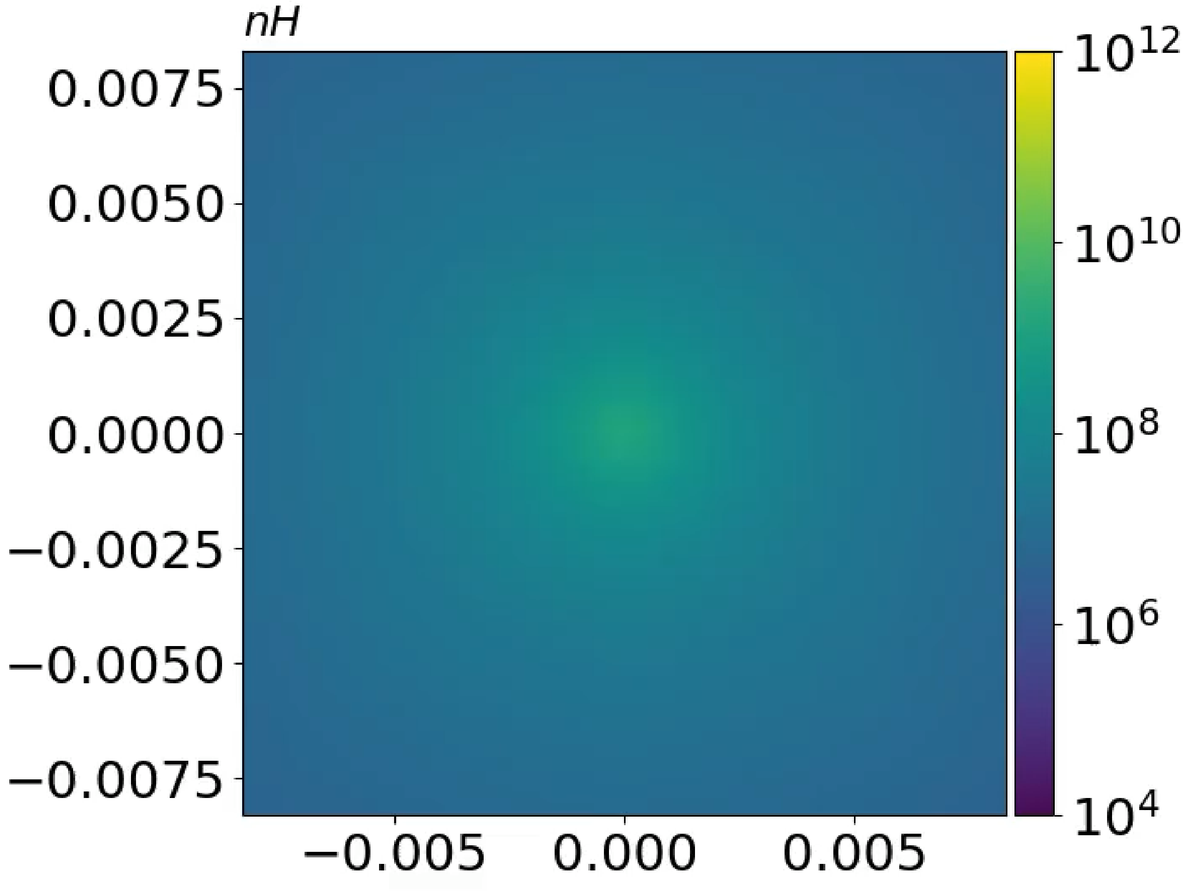






計算の様子: BE球 (自己重力 + 流体)

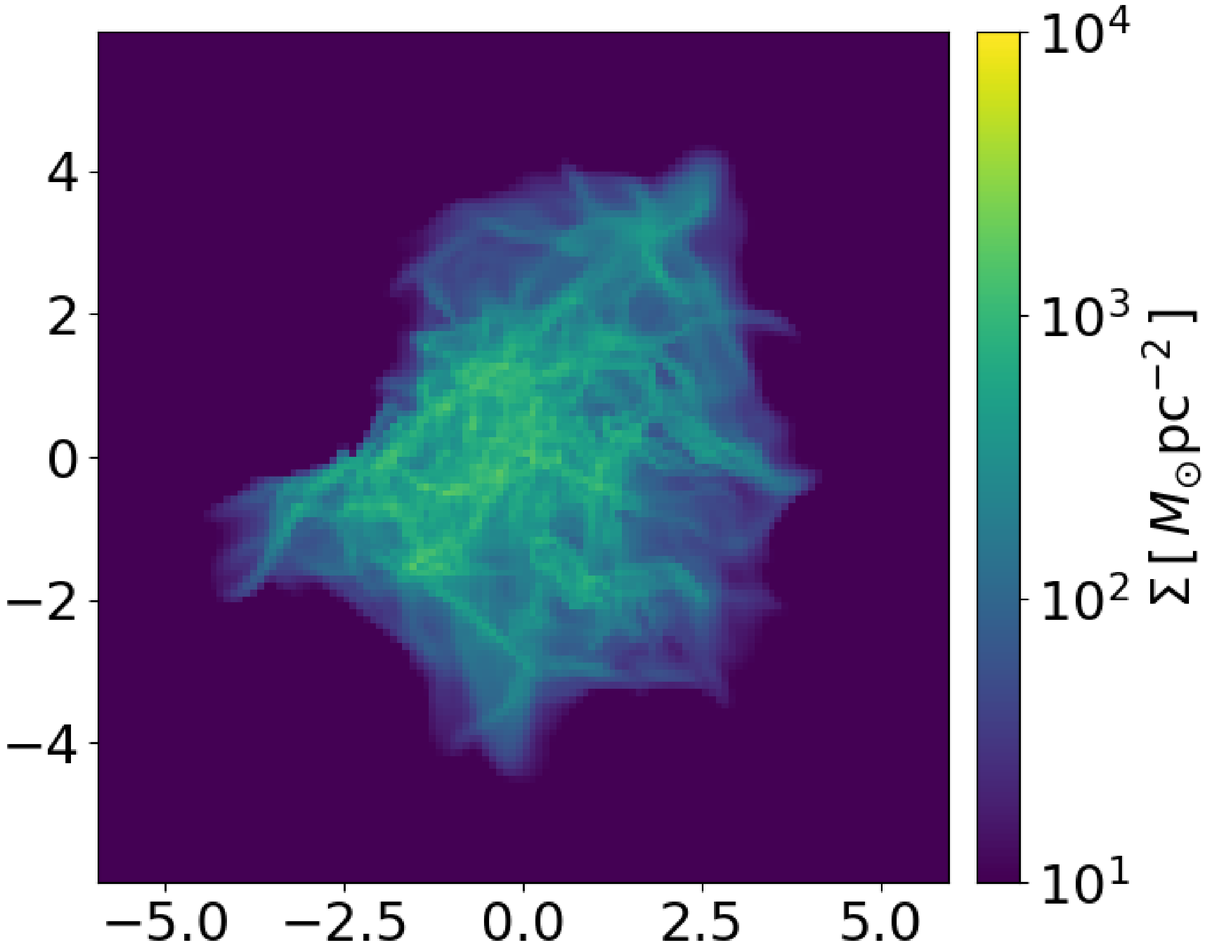




密度進化の様子

計算の様子

GMC (自己重力 + 流体)



(計算途中)

質量: 1.e4 Msun

半径: 3pc

最大解像度: 302 au

最大レベル9

１コア+1GPU計算 (H100, Pegasus)
(終了まで1週間くらい?)

反復計算

エネルギーの式: $\frac{\partial e}{\partial t} = \Gamma - \Lambda$

化学反応の式: $\frac{\partial y_i}{\partial t} = R_i$

Γ, Λ: 加熱・冷却関数, R_i: 反応率

化学種: H, H^+, H_2, e

収束までの反復回数が異なるのでwarp divergenceが起こる

計算ではセルごとに反復計算を実施する必要あり

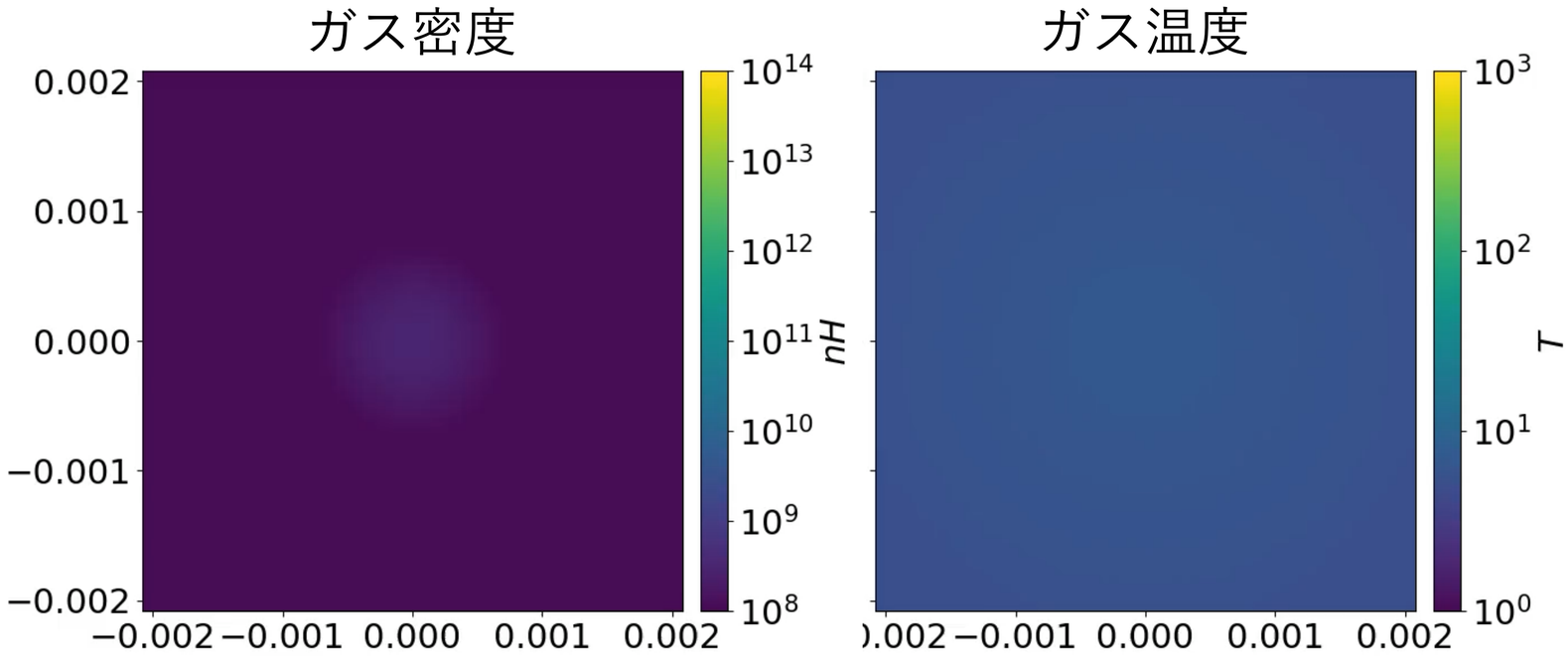
現状のコード

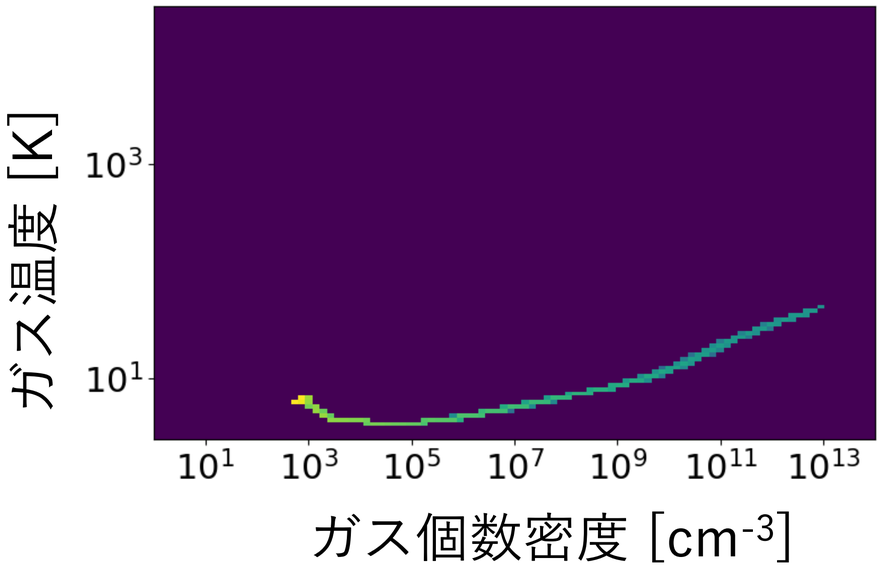
① 全セル一定数回反復計算を行う

② 収束していないセルについては、収束するまで反復計算を繰り返す

（流体の50%くらいのコスト）

BE球の重力収縮計算　流体＋輻射輸送+非平衡化学





まとめ

AMR法を用いた自己重力流体計算コードをGPU上で実行可能とした

　CUDAで再開発を行っている (OpenACCでもよかったはず)

移流計算など、GPUの多並列を活かせる部分は高速化可能 (速度測定等が間に合わずすいません)

反復計算については制約が多いが、とりあえず動かすことは可能

来年度はMIYABIを使用して、数10pcスケールの高解像度星団形成シミュレーションを実施する予定である。