物理学特殊講義I（MC向け集中）
AMR編

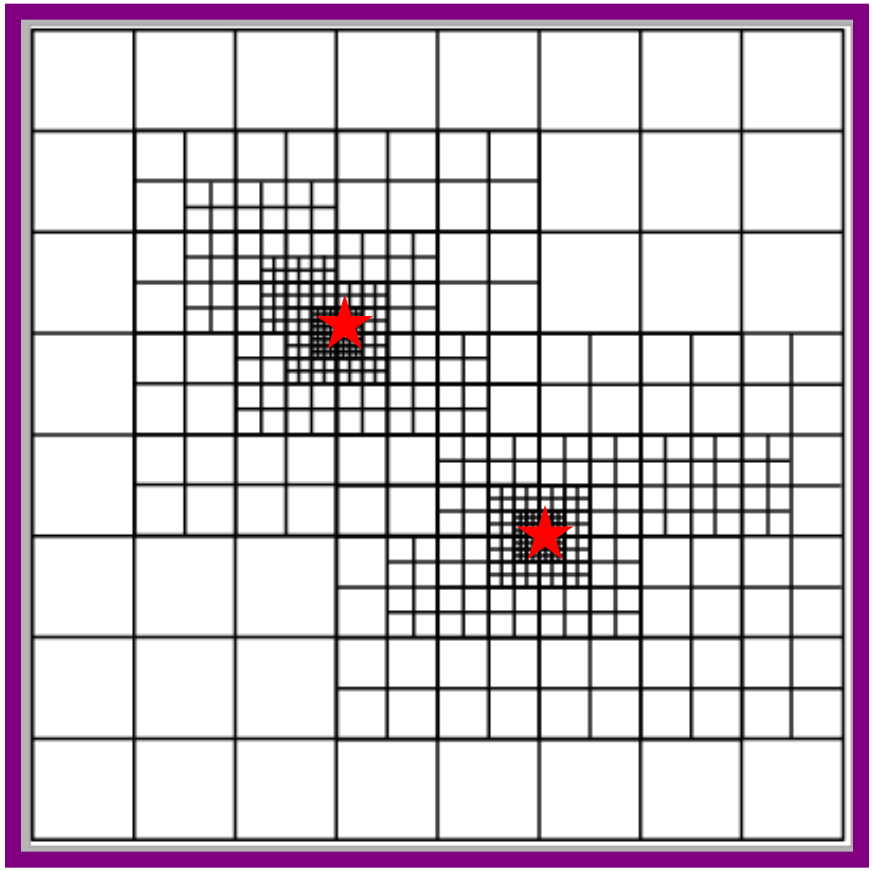
松本倫明（法政大学人間環境学部）

AMRとは

- Adaptive Mesh Refinement (AMR)
  - 適合格子細分化法
  - 解適合格子(Solution Adaptive Mesh)

- 骨子
  - 高解像度が必要な領域を高解像度に。
  - それ以外を低解像度に。
  - 「高解像度が必要な領域」を動的に変更する。
  - トータルで格子点数を節約する。
  - 計算機性能の割に、高解像なシミュレーションが可能。
    例：256^3 -> (256*1024)^3=(1.80144e+16)^3

AMRの模式図（例）
高解像度が必要な領域を高解像度に



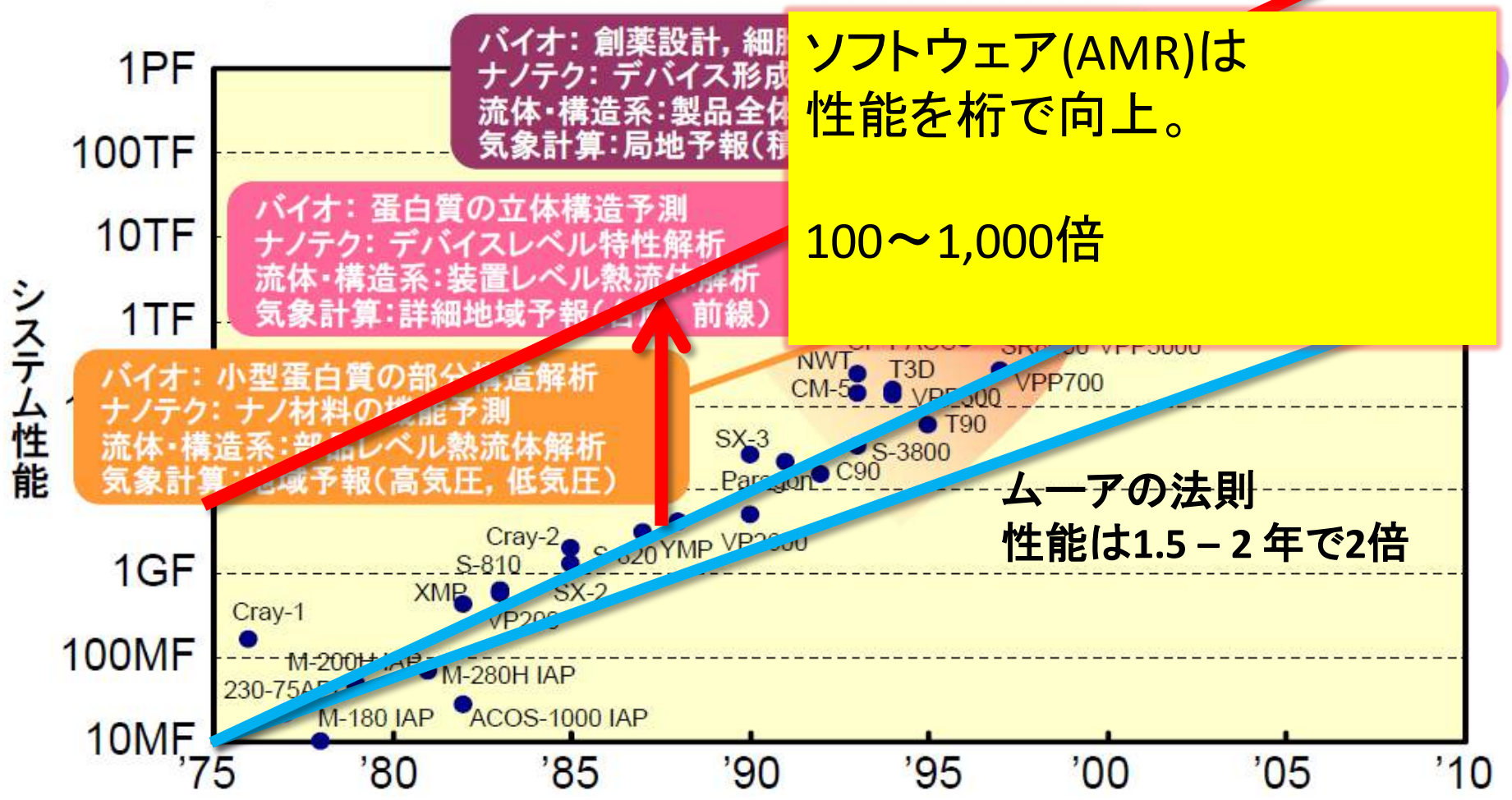
連星系の形成

星の運動とともに
格子を動的に更新する。

AMRを使う目的

- 自己重力による構造形成:
  - 形成される天体を分解したい。
  - 空間スケール$\propto$密度$^{-1/2}$
- 惑星間空間のプラズマ:
  - カレントシートを分解したい。
- 超新星爆発:
  - 燃焼波(デトネーション・デフラグレーション)を分解したい。
- 乱流:
  - パワースペクトルを長いレンジで書きたい。
- つまり、ダイナミックレンジが欲しい。
- 格子点数の割りに高解像度が欲しい。

なぜAMRか
ハードウェア vs ソフトウェア



スーパーコンピュータの性能の推移　日立製作所2004

古い資料でごめんなさい

AMRコードを作るか作らないか
作る場合

- 利点： 新しい物理の導入が容易。
  新しい解析手法の導入が容易。

- 欠点： 膨大なテスト計算の必要性。
  開発時間の必要性

- 開発期間： 流体（MHD）を「書く」だけなら1カ月以内。
  - 松本は流体部分を2週間で集中的に書いた。
  - 自己重力部分は大変だった。

- 効率よく開発するには
  - 一様格子コードでCFDを勉強してからがお勧め（総研大の講義を活用など）。
  - 流体などの既存のソルバを流用する。
  - はじめから並列版を作る。

AMRコードを作るか作らないか
作らない場合

- 既存の公開コードを利用する。
  現在の主流になりつつある。シミュレーションコードの複雑化が原因。

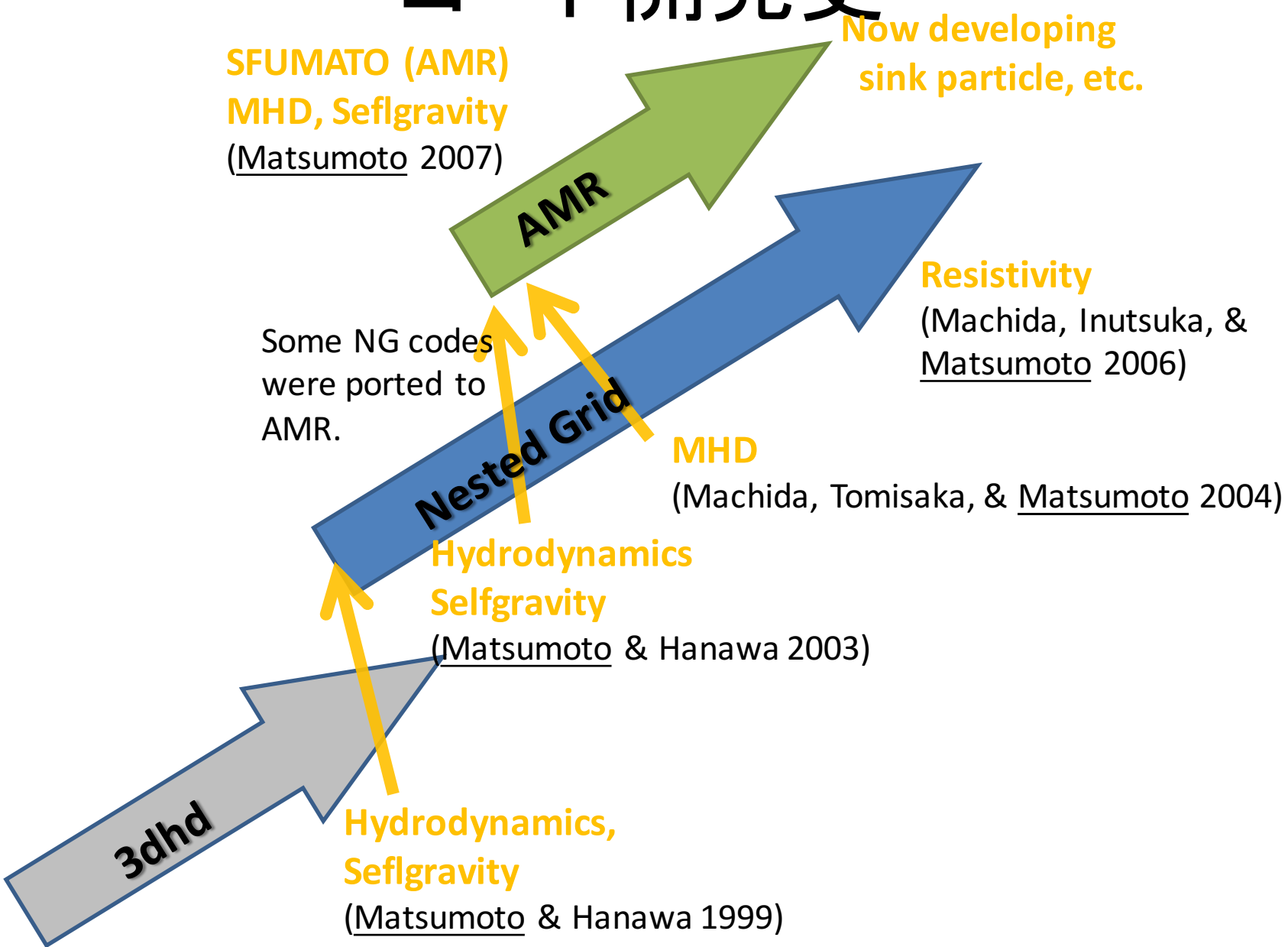
- 利点：　手軽に利用できる。開発の不要（テスト計算は必要）。

- 欠点：　改造の困難さ。新しい物理を導入場合。
  科学計算のコードは汚い！
  （一般のソフトウェアと異なる点）。

- アイディア勝負な人は作らない方が良いと思う。

- 開発者を共同研究者にして、いろいろ頼むのが吉か？
  ブレイクスルーな仕事では、開発者が共同研究者であることが多い。

コード開発史


SFUMATO (AMR)
MHD, Seflgravity
(Matsumoto 2007)
Now developing
sink particle, etc.
AMR
Resistivity
(Machida, Inutsuka, &
Matsumoto 2006)
Some NG codes
were ported to
AMR.
Nested Grid
MHD
(Machida, Tomisaka, & Matsumoto 2004)
Hydrodynamics
Selfgravity
(Matsumoto & Hanawa 2003)
3dhd
Hydrodynamics,
Seflgravity
(Matsumoto & Hanawa 1999)

AMRコードとその他のコードの比較

コード種別	3dhd	Nested grid	AMR
行数	3,524 （自動生成後6,994)	14,037	31,637
言語	Fortran77 + Perl コード自動生成	Fortran77 + 自作cpp	Fortran90
並列化	VPP Fortran 指示行の挿入	なし/自動並列 指示行の挿入	MPI並列
バージョン	-	2.12	3.6
実装機能	流体, 自己重力	HD, MHD, registivity, 自己重力	HD, MHD, 自己重力, sink 粒子

分類

JOURNAL OF COMPUTATIONAL PHYSICS 82, 64–84 (1989)

Local Adaptive Mesh Refinement for Shock Hydrodynamics

M. J. BERGER

Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, 10012 New York

AND

P. COLELLA

Lawrence Livermore Laboratory, Livermore, 94550 California

Received S[illegible]ber [illegible] 198[illegible] Revis[illegible] May[illegible] 1988

全てのAMRは

The aim of this work is the development of an automatic, adaptive mesh refinement strategy for solving hyperbolic conservation laws in two dimensions. There are two main difficulties in doi[illegible] The f[illegible]lem[illegible] the p[illegible] of disco[illegible]ties [illegible] the [illegible]ture and the effect on them [illegible] dis[illegible]nti[illegible]ities in the main. The sec[illegible]d pr[illegible] to [illegible]e algo-rit[illegible]n to minim[illegible]ze [illegible]emo[illegible] and [illegible]rhead. [illegible]s is an [illegible]t [illegible]nt consi[illegible]a[illegible]nd will con[illegible]e to be impor[illegible] as m[illegible]s[illegible]ticat[illegible]gorit[illegible]s [illegible] data st[illegible]s th[illegible] than arrays are developed for use on vector and parallel computers.

この論文起源

AMRの分類

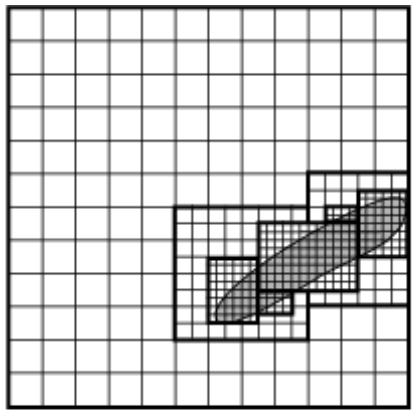
A)　パッチ型ブロック構造格子
- パッチ指向
- 最初のAMR
- Berger & Oliger 1984,
- Berger & Colella 1989

B)　八分木型ブロック構造格子
- 八分木構造

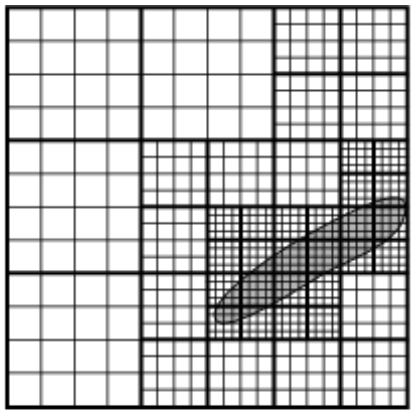
C)　セル分割型格子
- 八分木構造

D)　三角形非構造格子
- あまり用いられていない。
- 機体に沿った境界条件に有利。

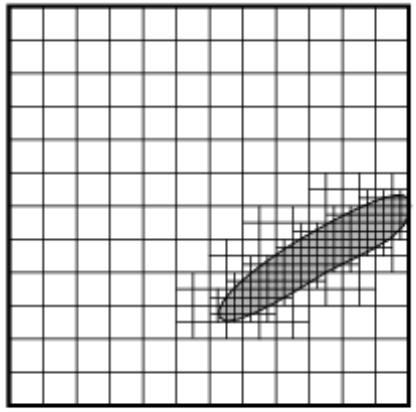
Level = 0 ～ 2



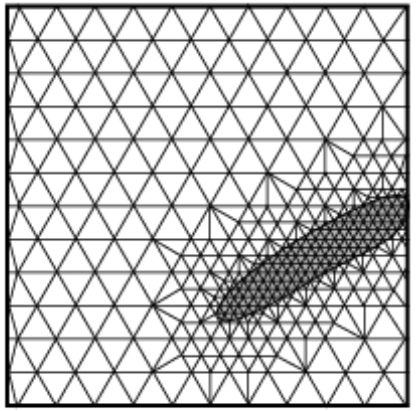
(A)　パッチ型
ブロック構造格子



(B) 八分木型
ブロック構造格子



(C) セル分割型格子



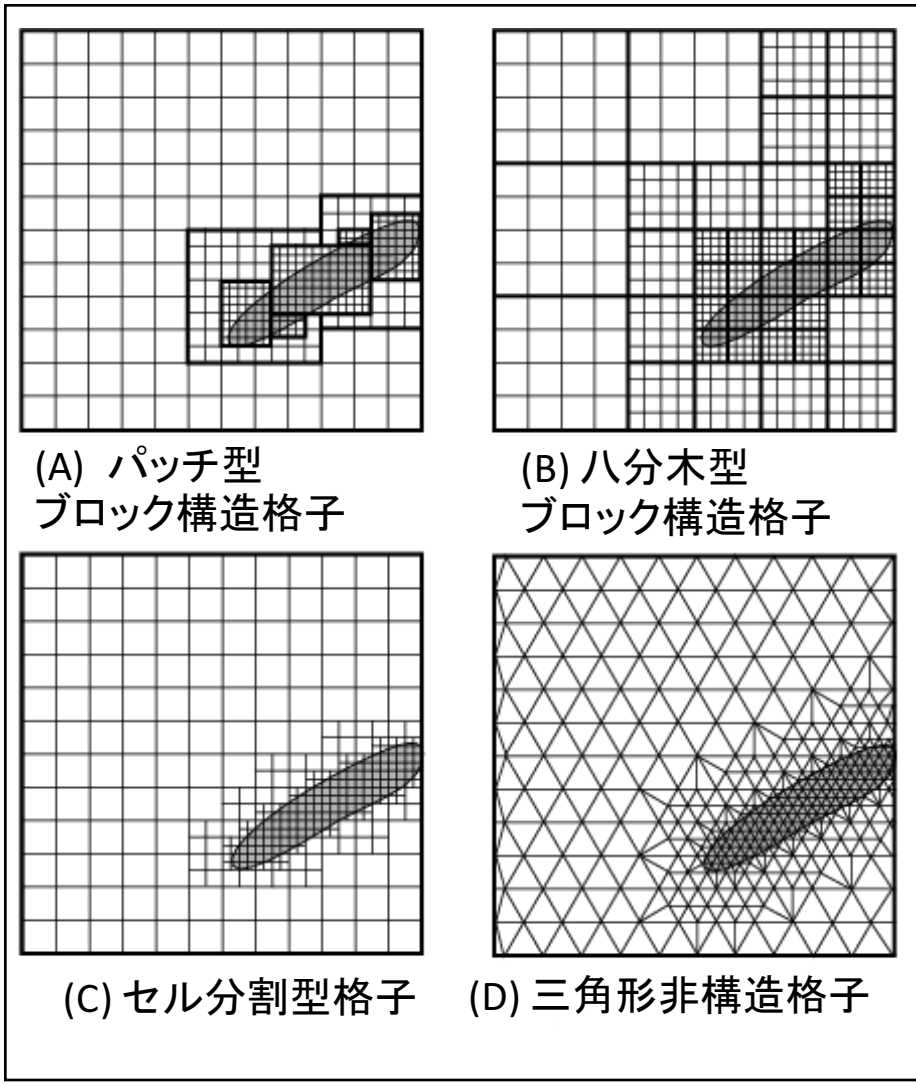
(D) 三角形非構造格子

AMRの分類

A)　パッチ型ブロック構造格子
- セル数は少なめ。
- ブロック配置のアルゴリズムが複雑。
- 袖のセルが少なく、効率が良い。
- メモリを動的に使う。

B)　八分木型ブロック構造格子
- セル数多め。
- ブロック配置のアルゴリズムが単純。
- 袖のセルが多くい。
- メモリを静的に使う。
- キャッシュの有効活用。

C)　セル分割型格子
- セル数は必要最低限。
- オーバーヘッドが大きいか？
- メモリを静的に使う。

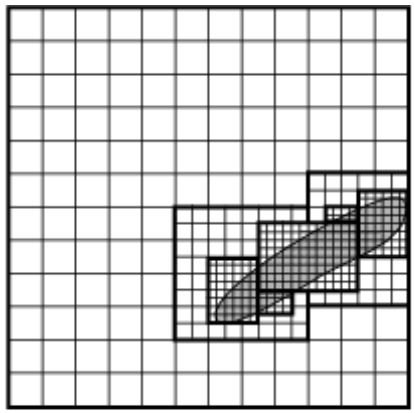


天体物理学における主なAMR（古いかも？）

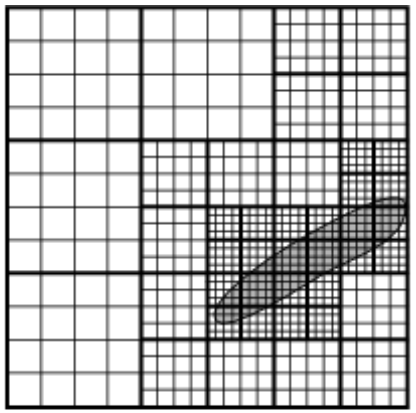
ほかにも沢山。国内にも下記以外に2コード存在。

現在では、MHD と 自己重力は多くのAMRに実装されている。

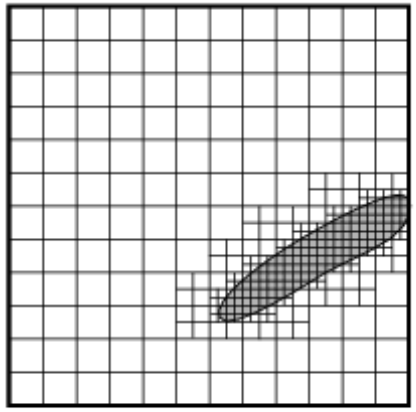
Code name	Author(s)	Main targets	Grid type
ORION	R. Klein	Star formation	A
Enzo	M. Norman	Cosmology	A
FLASH	ASC/U-Chicago	Any	B
BATS-R-US	K. G. Powell	Space weather	B
NIRVANA	U. Ziegler	Any	B
RIEMANN	D. Balsara	ISM	A
RAMSES	R. Teyssier	Cosmology	C
CASTRO	A. S. Almgren CCSE.LBL	Supernovae	A
PLUTO	A. Mignone	Any	A
Athena	J. Stone	Any	Coming seen
SFUMATO	T. Matsumoto	Star formation	B



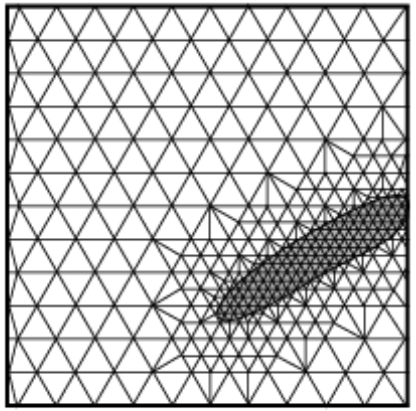
(A) パッチ型ブロック構造格子



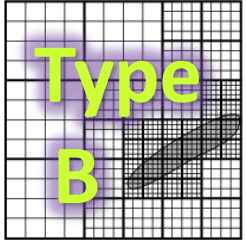
(B) 八分木型ブロック構造格子



(C) セル分割型格子



(D) 三角形非構造格子



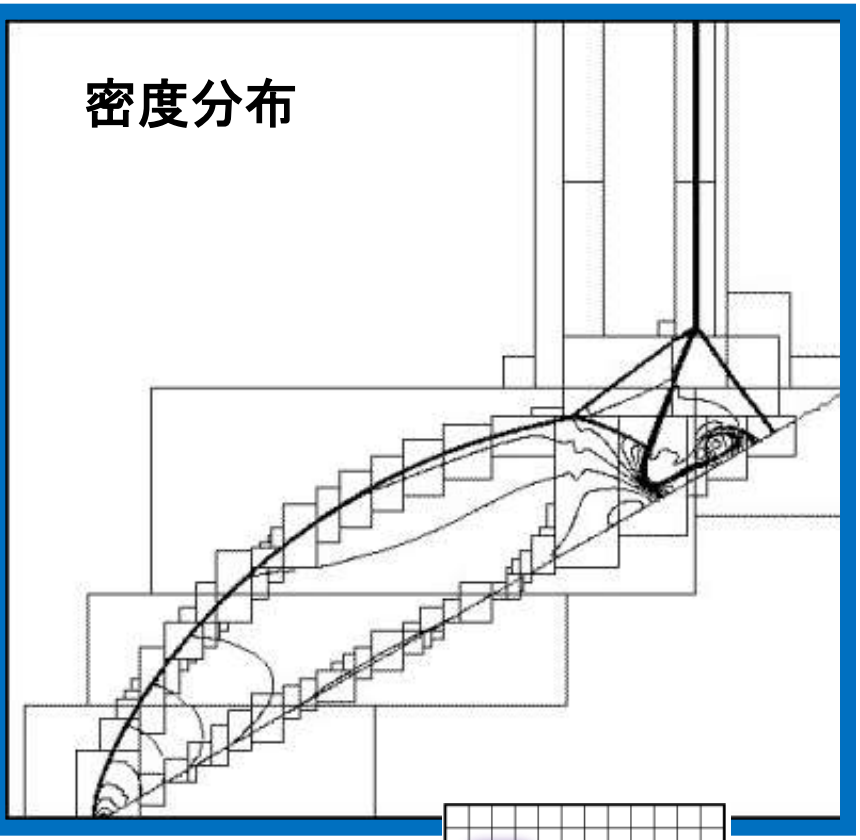
What is Sfumato

Self-gravitational **F**luid-dynamics **U**tiliz
Adaptive **T**echnique with **O**ct-tree.

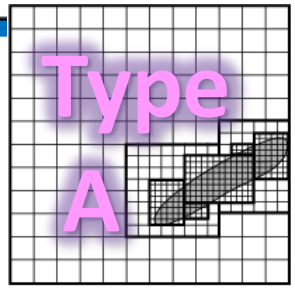
- *Sfumato* は本来、絵画の技法。レオナルド・ダ・ビンチ (1452–1519) によって完成された。
- その後、ルネサンスーバロック期の多くの画家に用いられた。
- モチーフの輪郭をぼかし、**空気を表現**。
- 我々のAMRコードも**ガス(空気)を表現**。
- Matsumotoのアナグラムで**はない**。

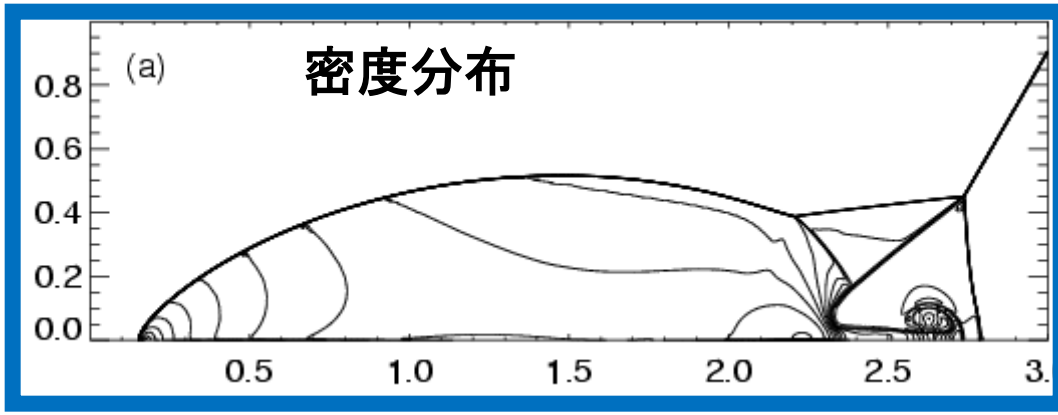
Mona Lisa, Leonardo da Vinci (1503–1507)

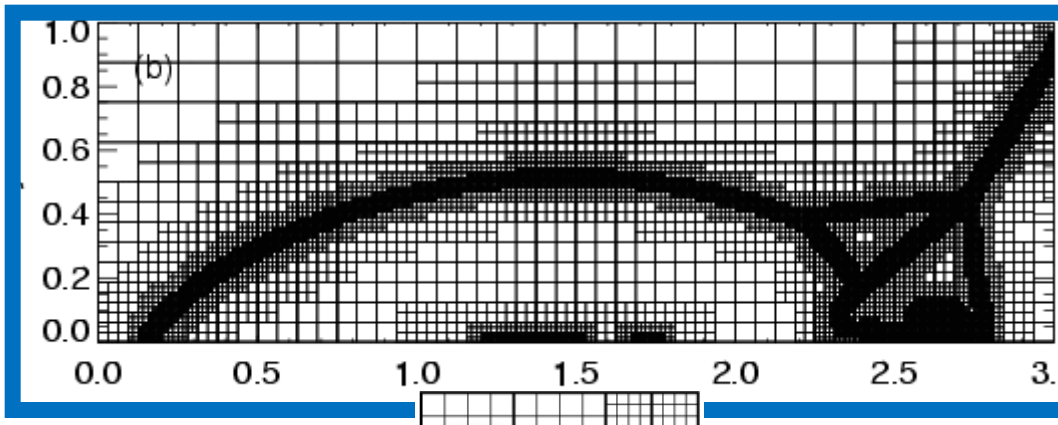
二重マッハ反射問題みるタイプ別の解

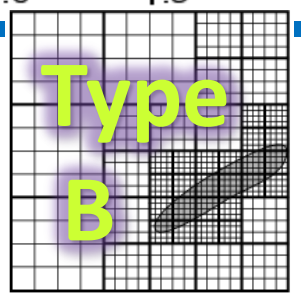


ORION?

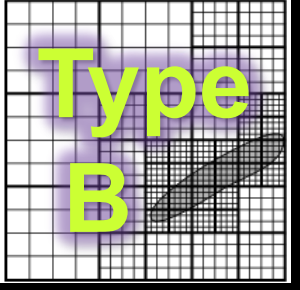




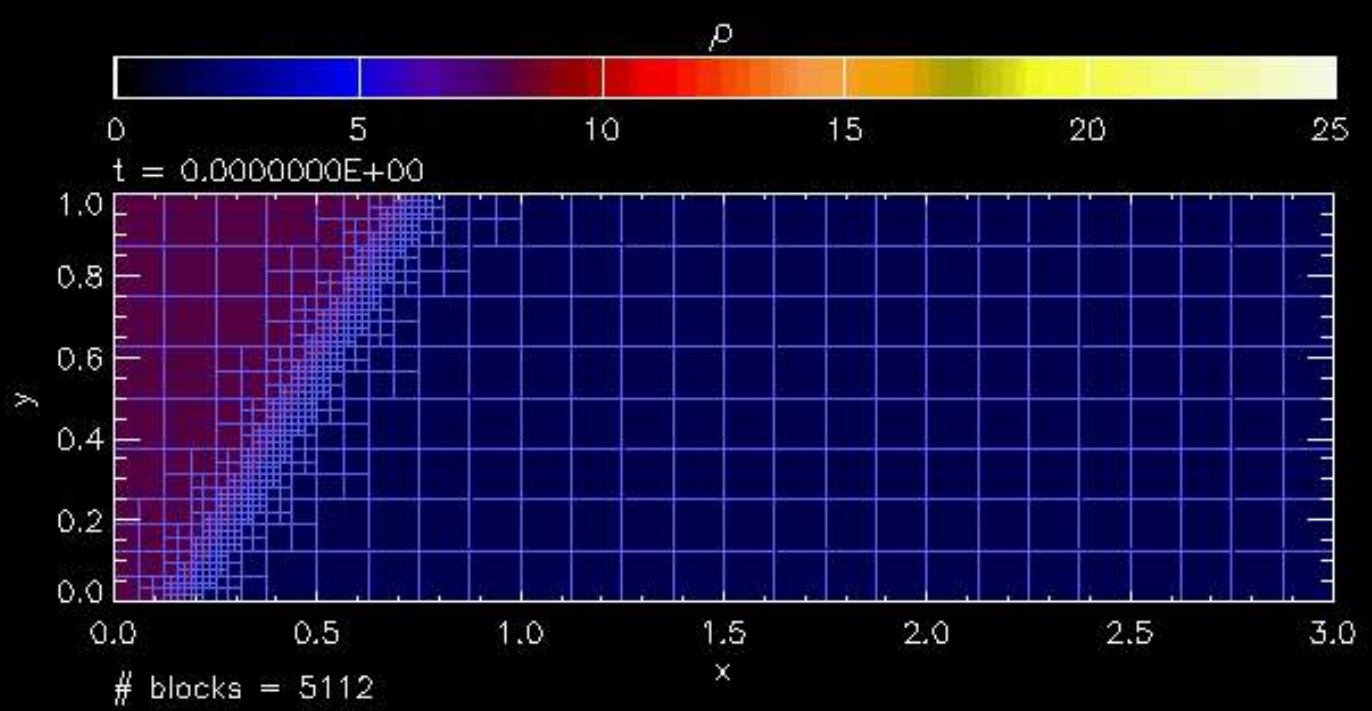




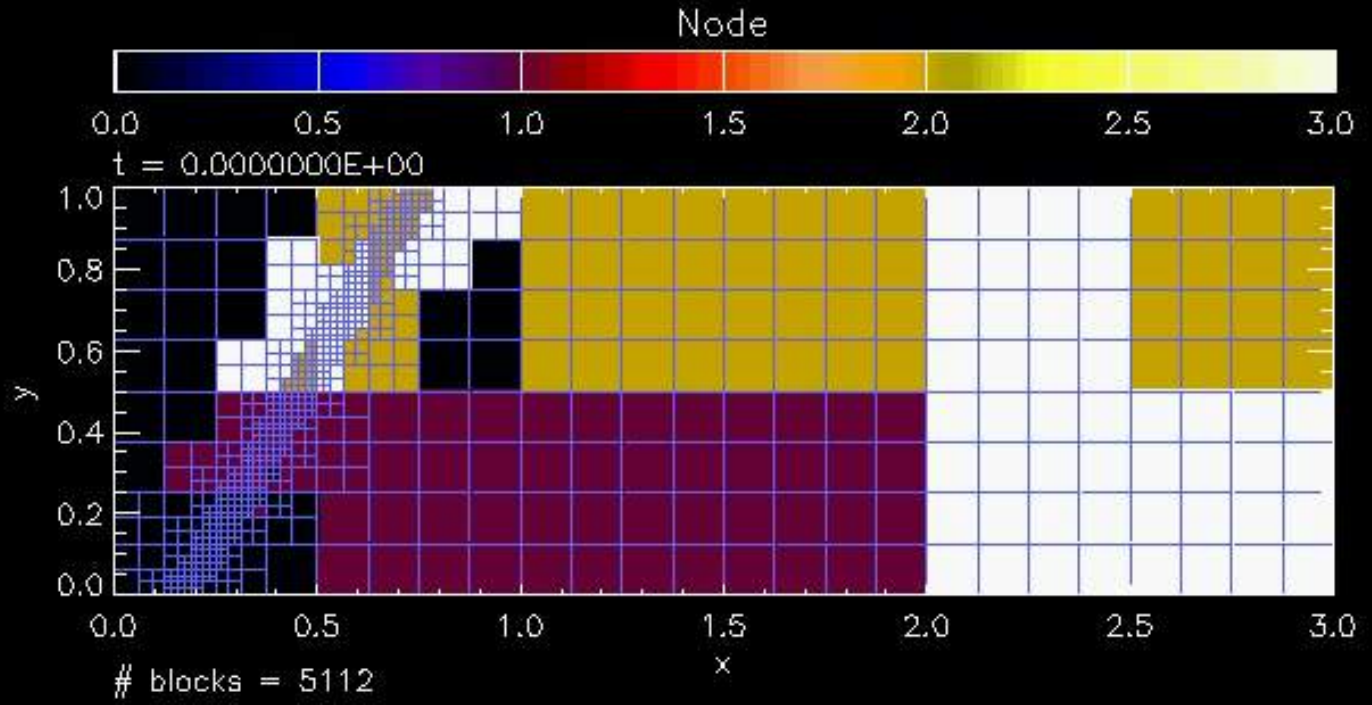
SFUMATO



密度分布



ノード（コア）分
0, 1, 2, 3



SFUMATO
Matsumoto 07

最近の動向

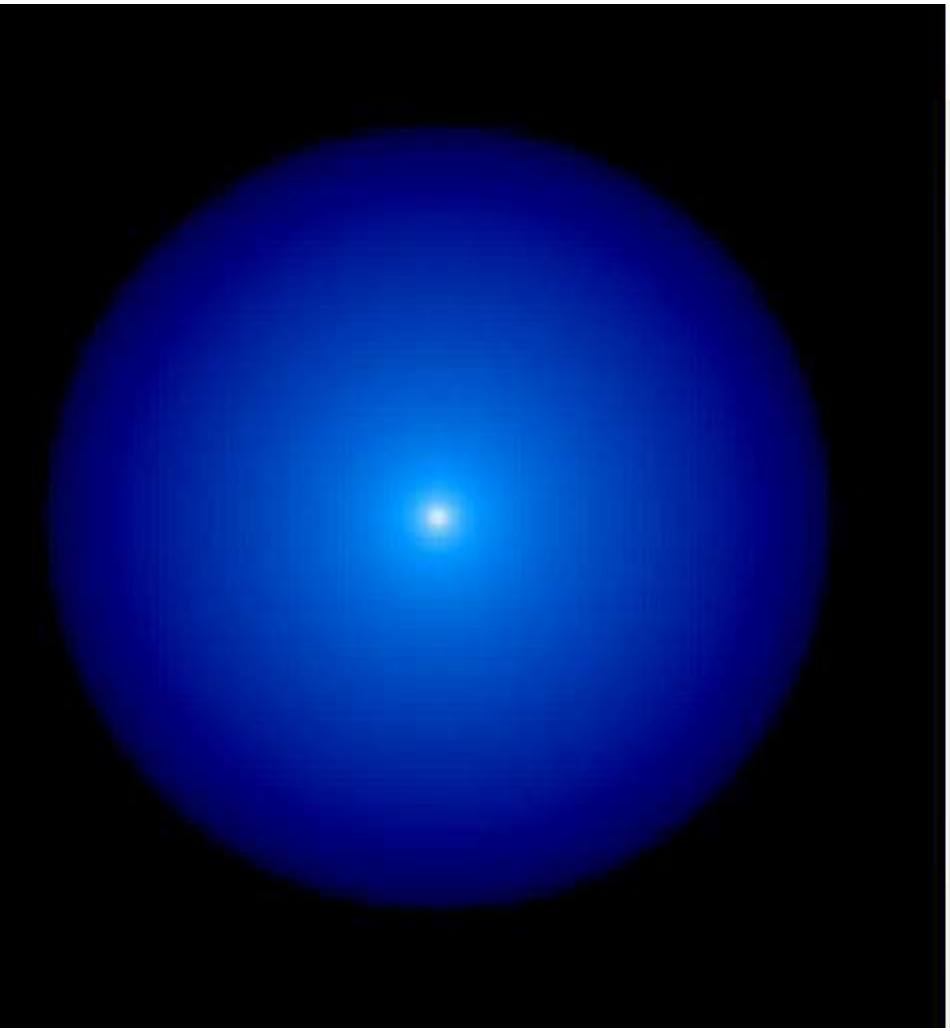
AMRは当たり前の時代になった。

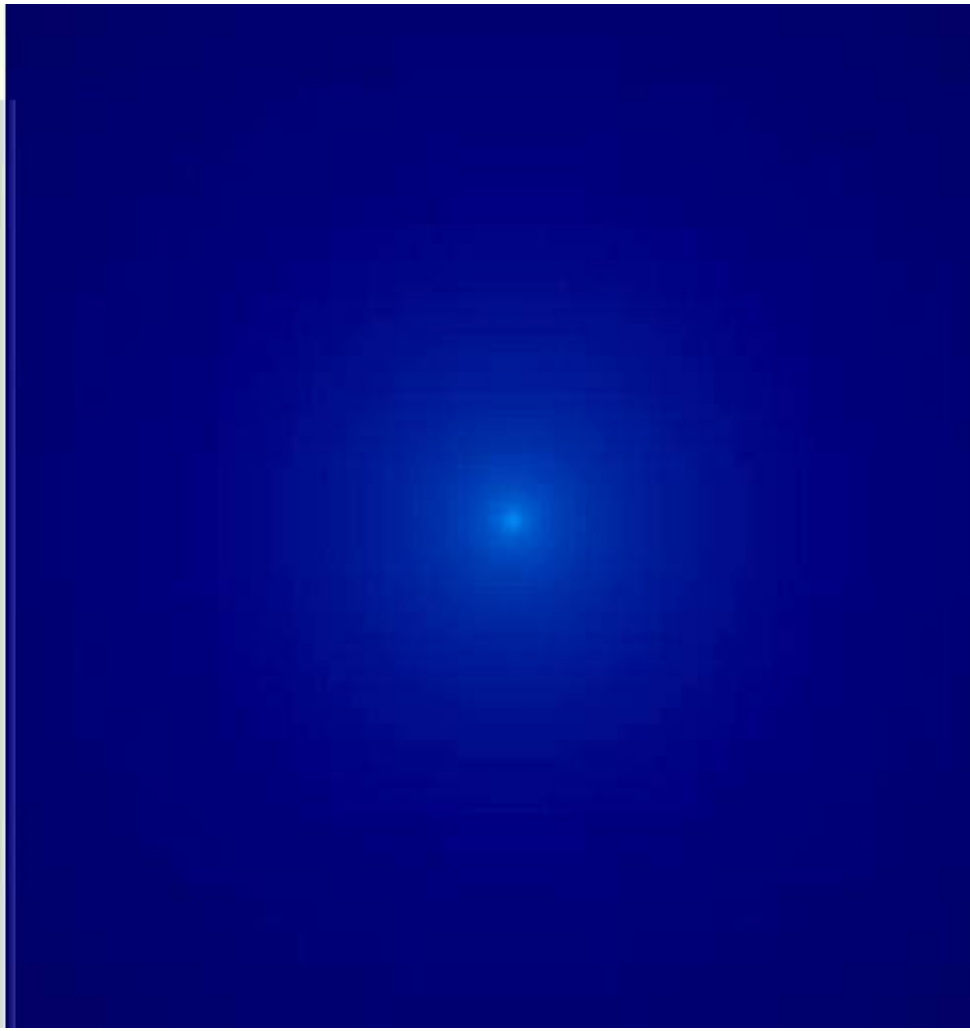
- AMRが特別ではなくなってきた。
  - メッシュ法自己重力系では必須

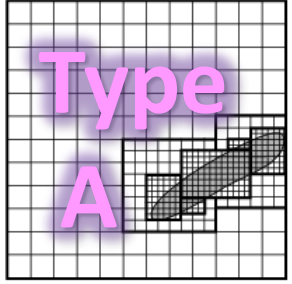
- 方法論
  - 新しい物理モデルで勝負する人
  - アイディアで勝負する人

乱流コアでの多重星形成
ORION: 流体+自己重力+輻射

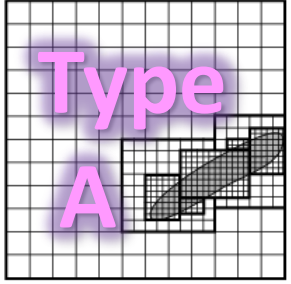
Krumholz, Klein, & McKee (2007)

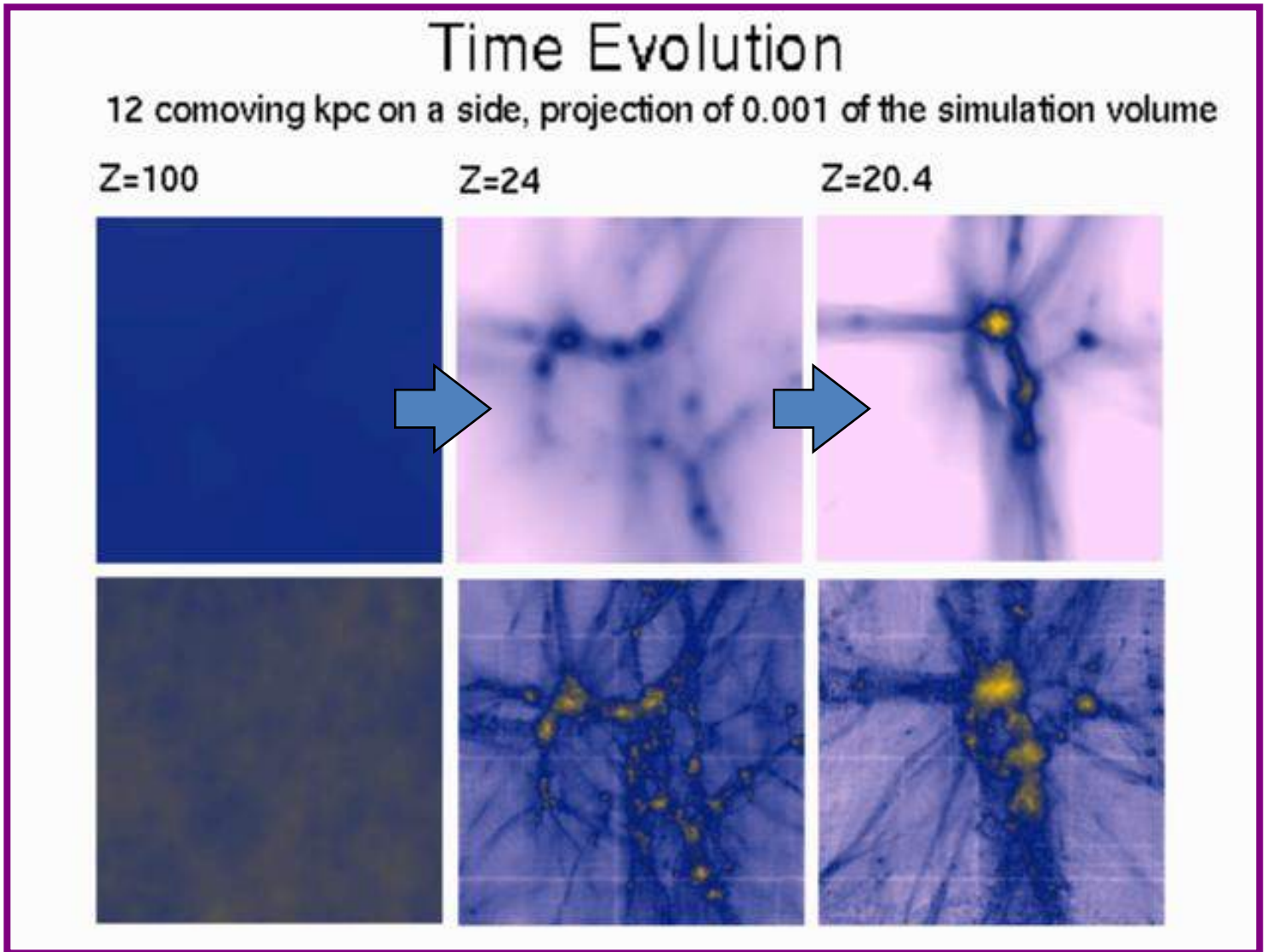






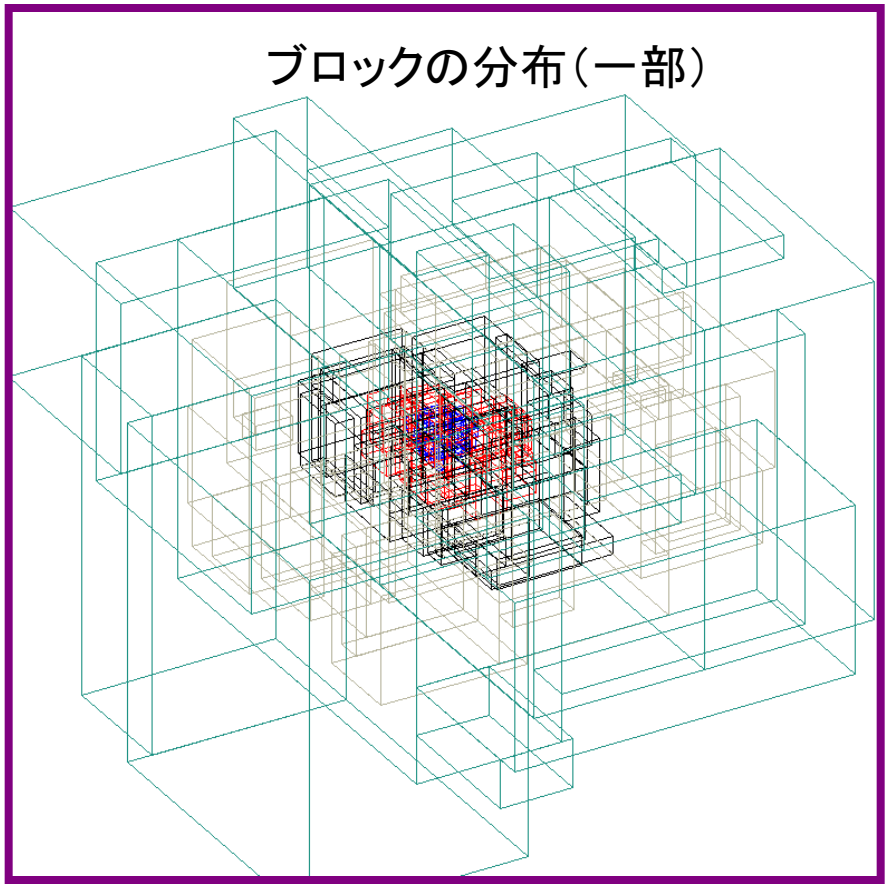
Enzo (M. Norman)
Block-structured grid (patch-oriented type)
First star formation (Abel et al. 2003)

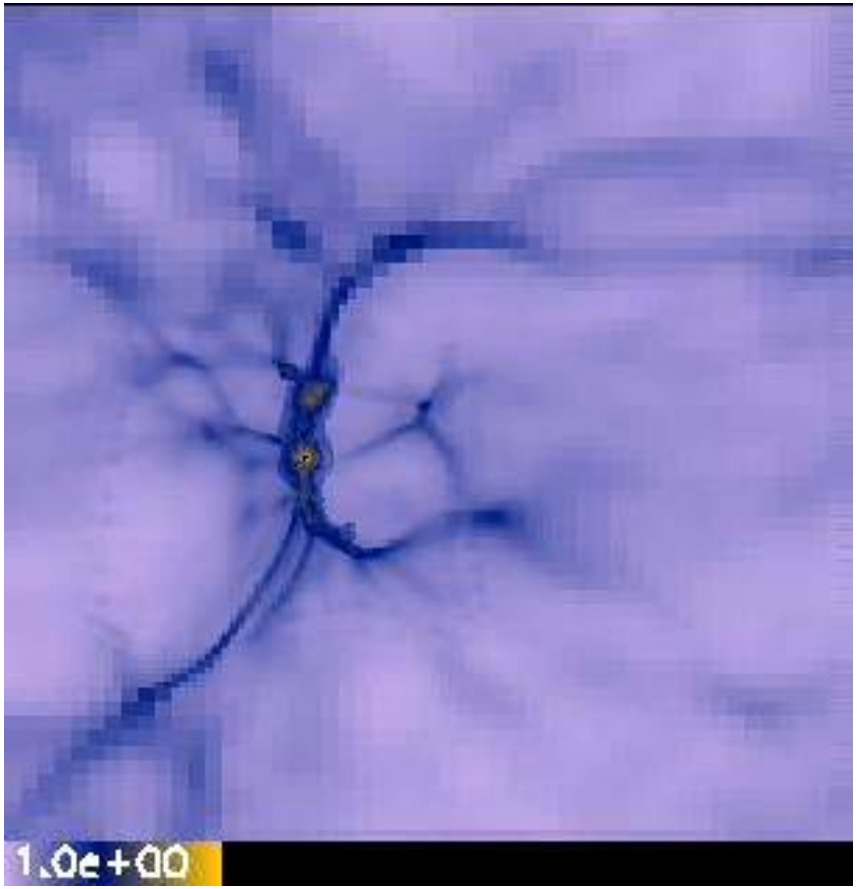




First star formation by Enzo

Abel et al. (2003)



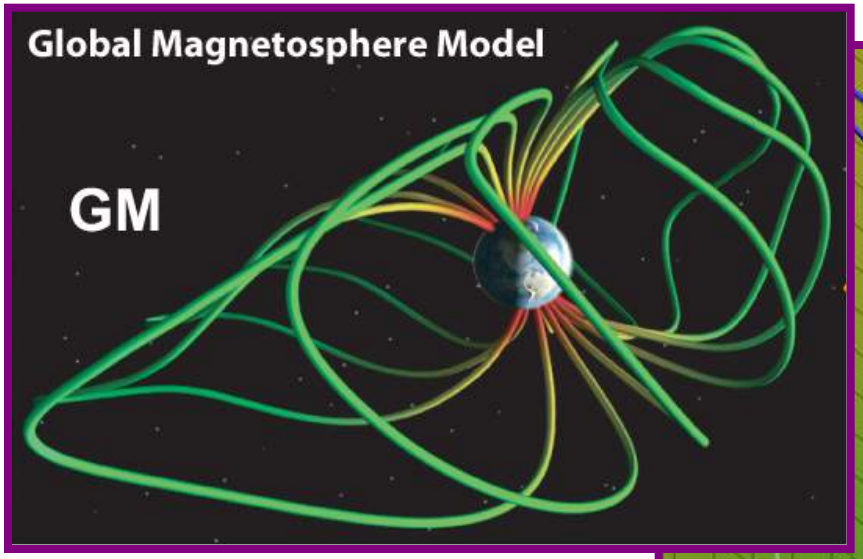


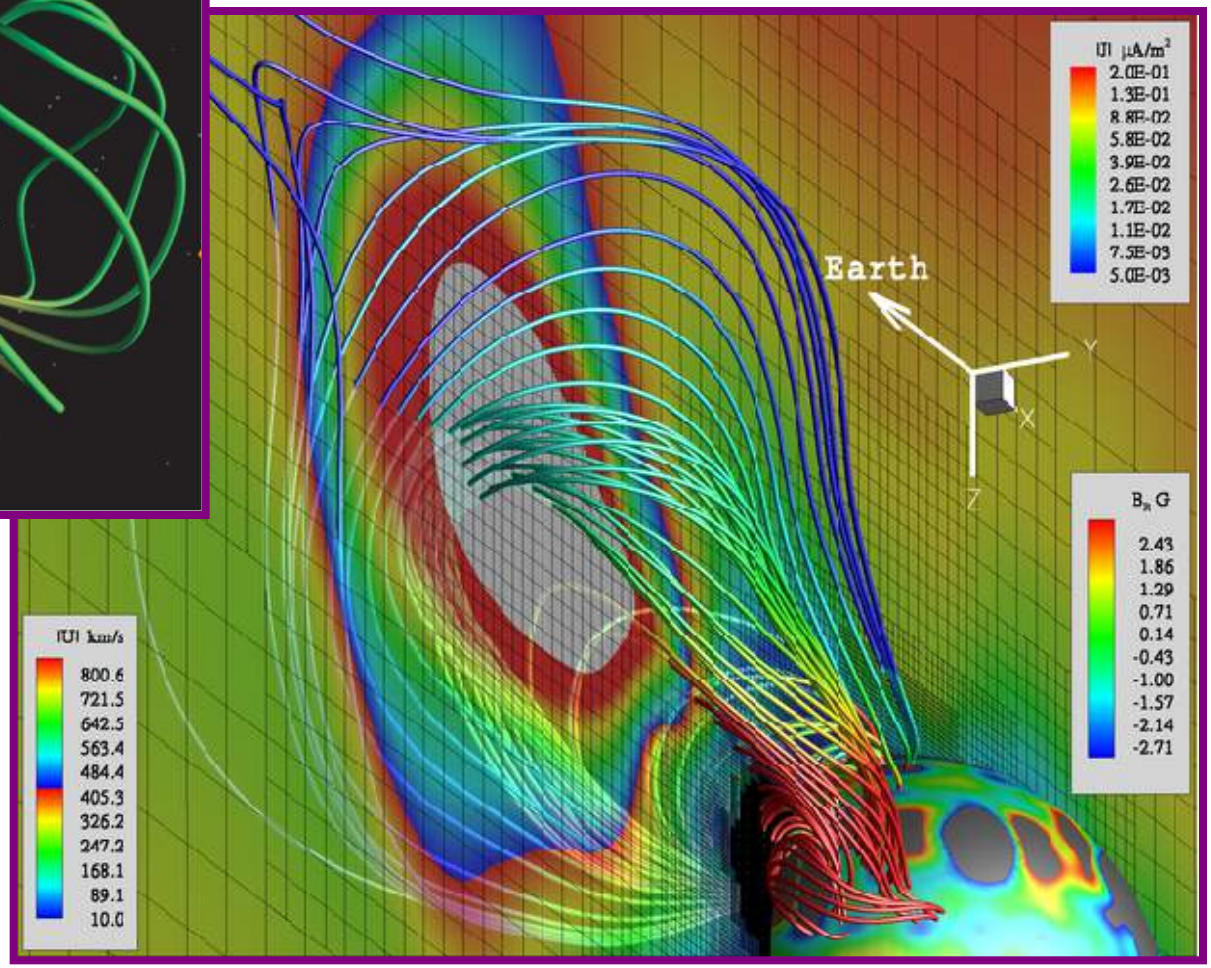
6 kpc ⇒ 100 AU (1,2000倍)

BATS-R-US (K. G. Powell)

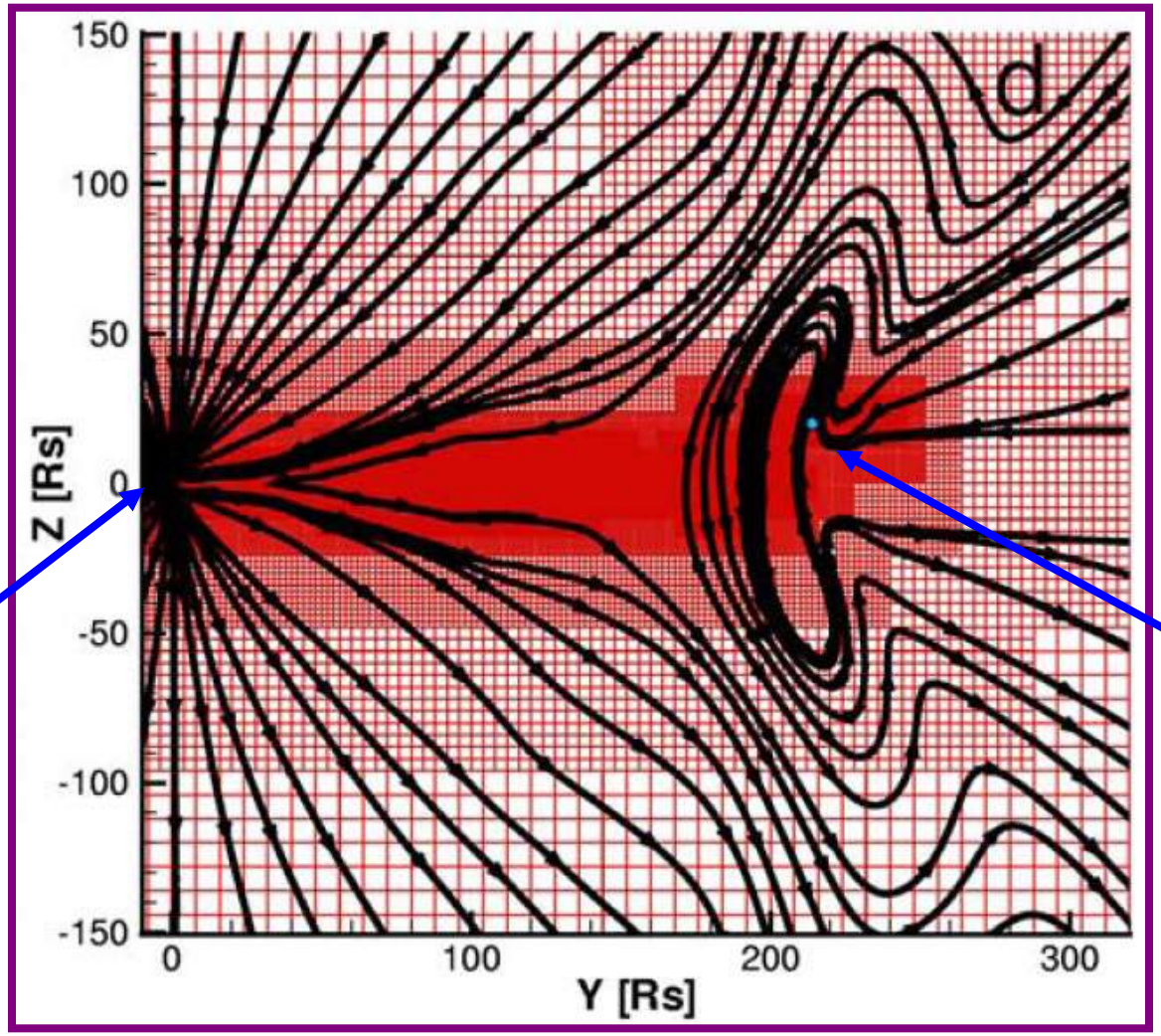
Block-structured grid (self-similar oct-tree type)

Space Weather





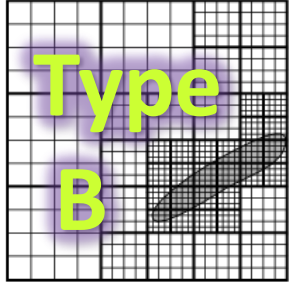
Coronal Mass Ejection by BAT-R-US Manchester IV et al. (2004)

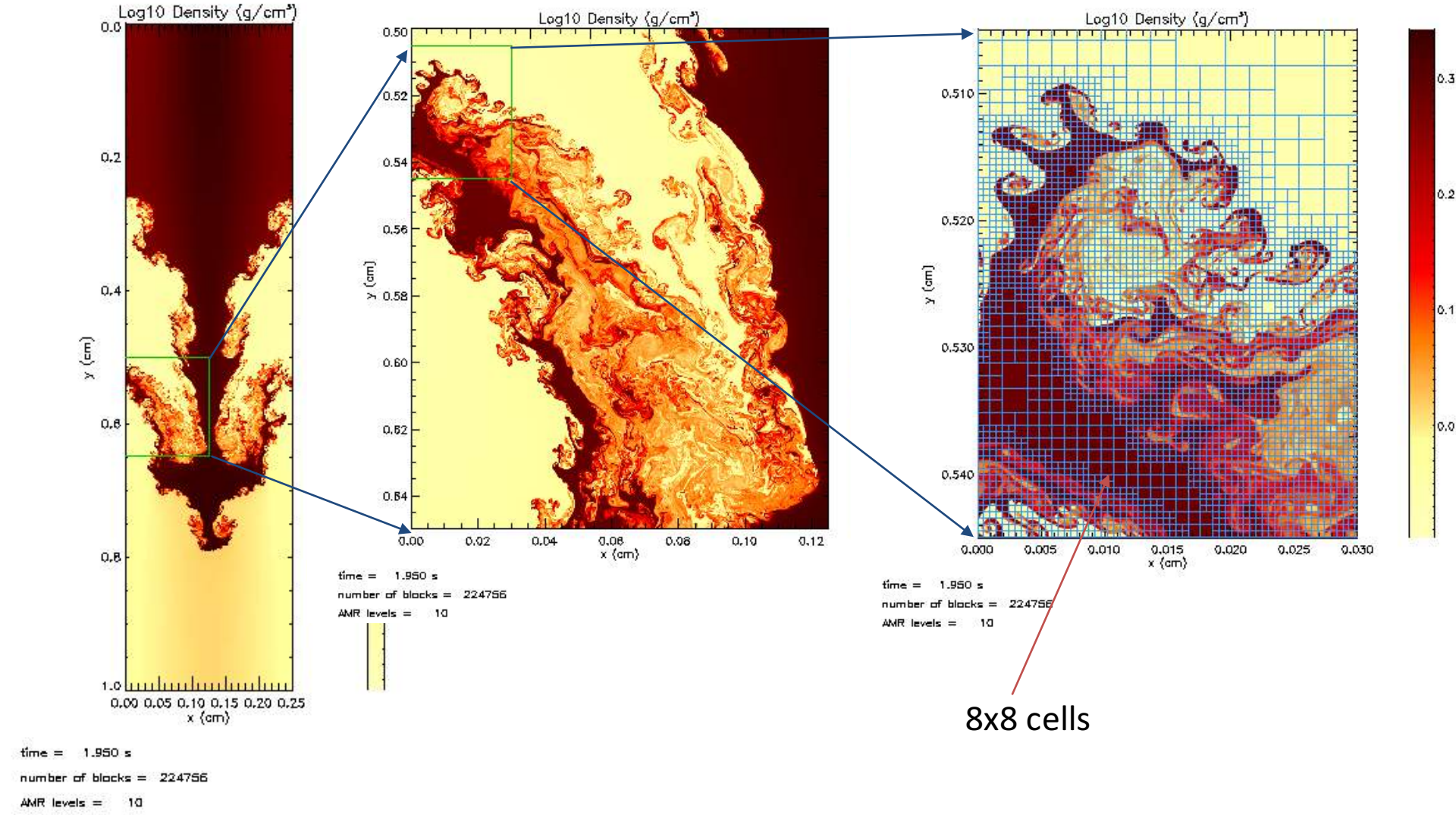


太陽

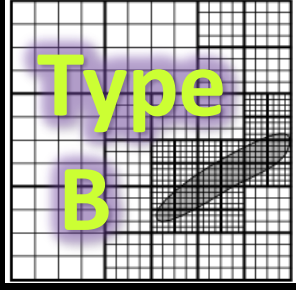
地球

FLASH (ASC, U-Chicago)
Block-structured grid (self-similar oct-tree type)
Rayleigh-Taylor Instability





タイプIa超新星爆発（FLASH）
色：反応率と密度



White Dwarf Detonation

Resolution: 6 km

Initial Bubble Radius: 18 km

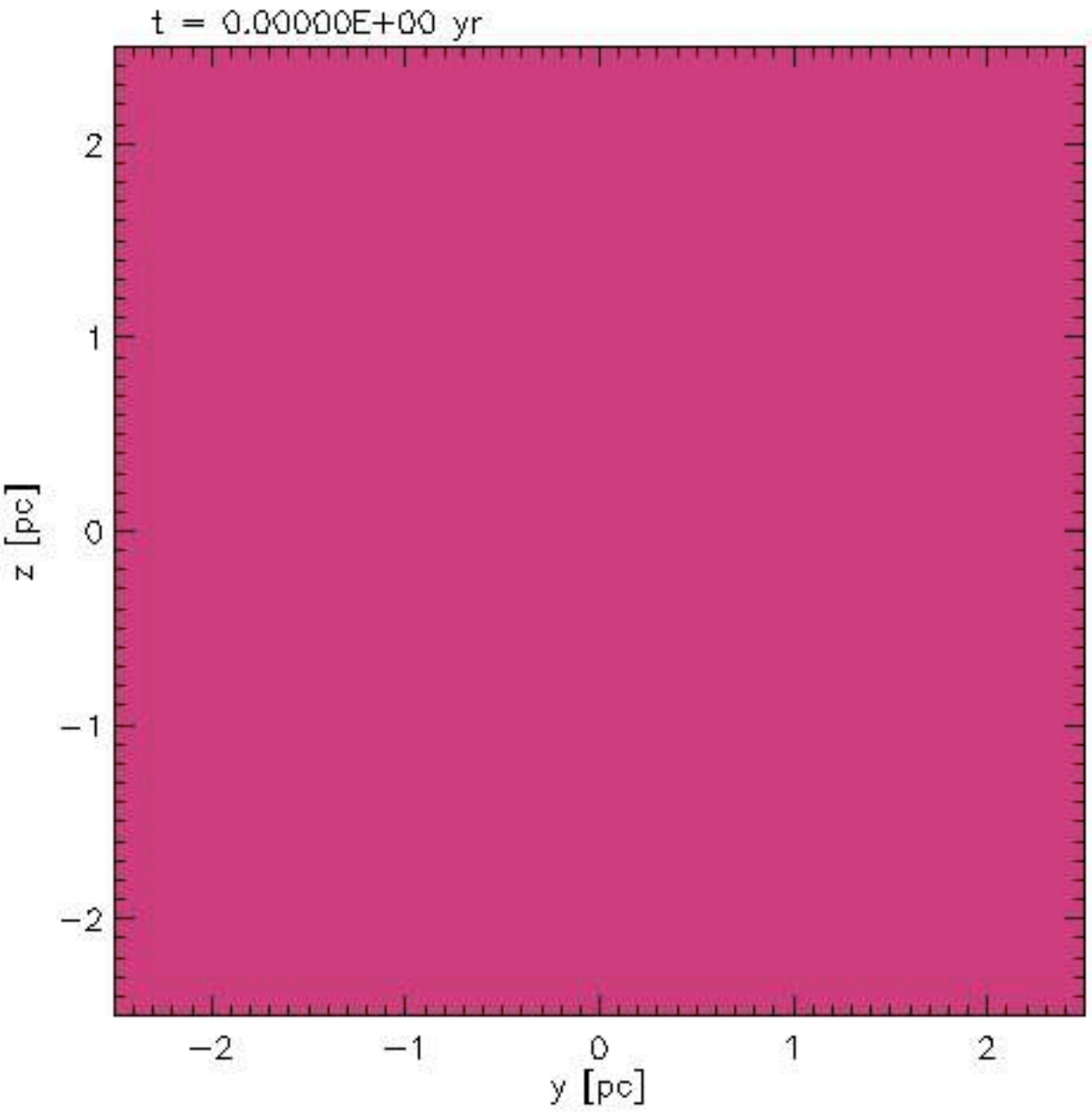
Ignition Offset: 42 km

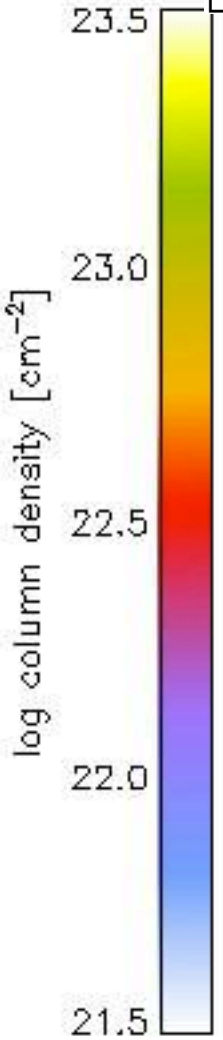
Variable 1: Density [1.5e+07 - 2.0e+07]

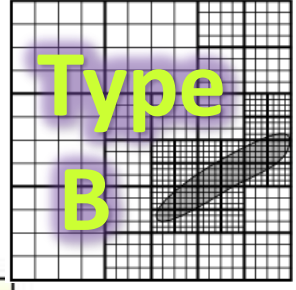
Variable 2: Reaction Progress [0.0 - 1.0]

Jordan et al., 2007　astro-ph/0703573

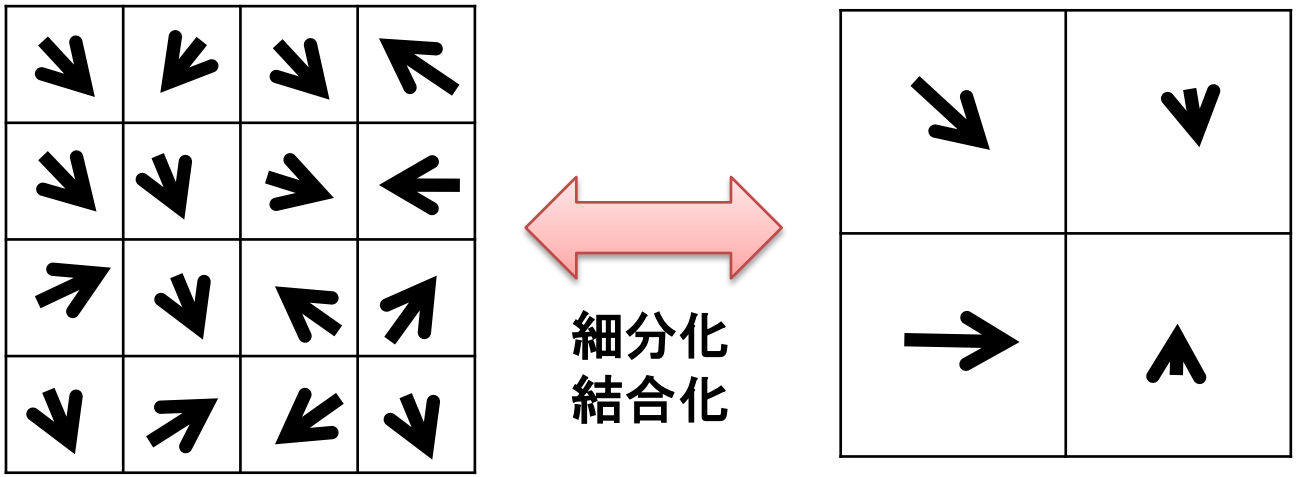
SFUMATOによる乱流星形成シミュレーション
HD + 自己重力







AMRは乱流が不得意であるという「定説」



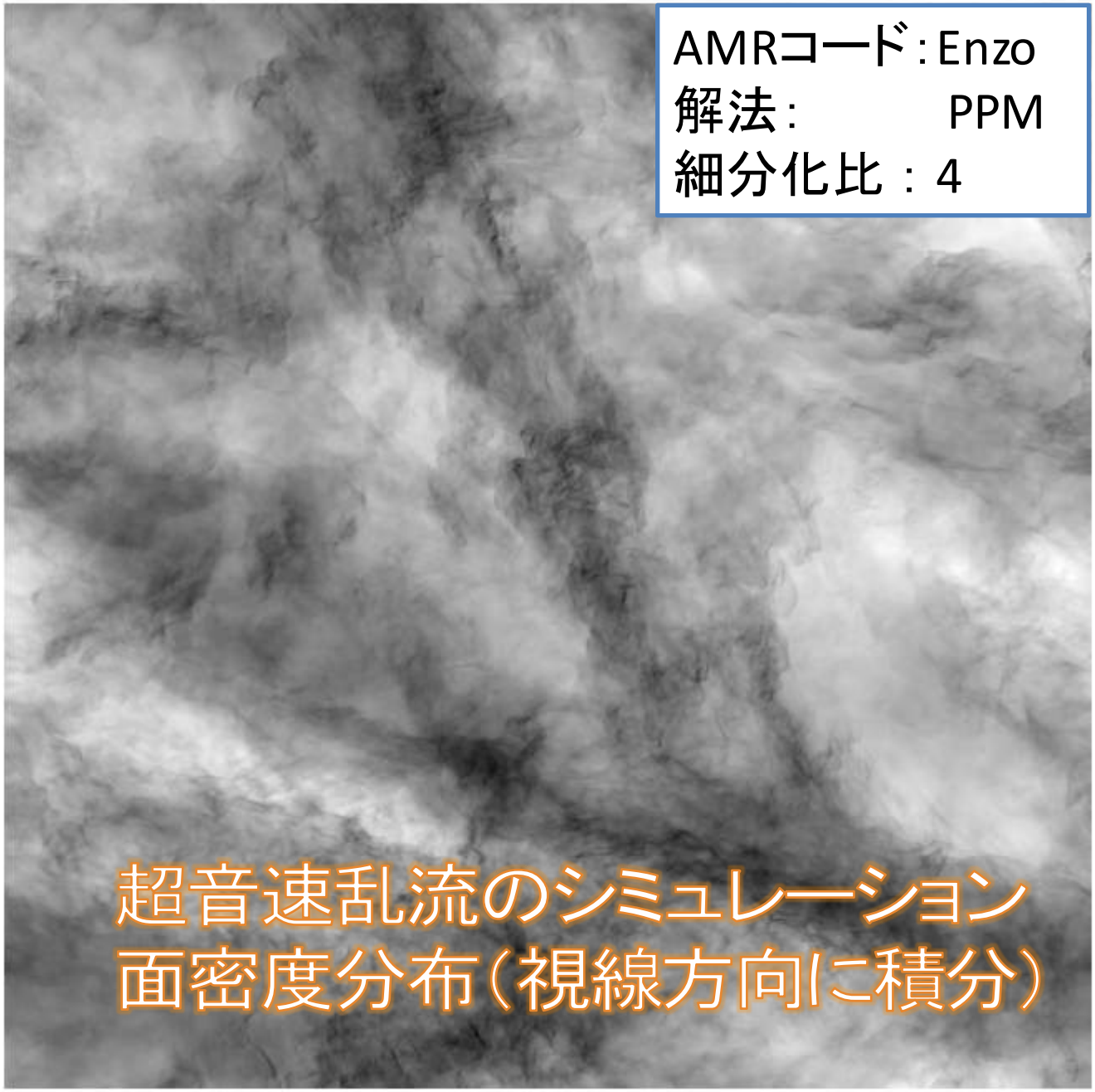
$\vec{v}$ は保存するが、$|\vec{v}|^2$は保存しない。

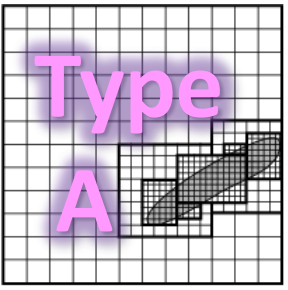
全エネルギー (E_K+E_{th}) は保存する。

運動エネルギー(E_K)が熱エネルギー(E_{th})になる。

AMRコード：Enzo
解法：　　PPM
細分化比：4

分解能
1024^3
相当





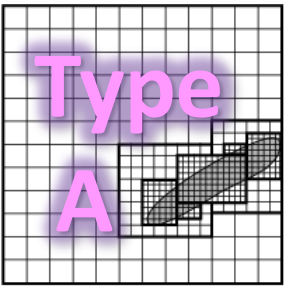
Kritsuk et al. 2006

level 0
256^3

Level 1
1024^3
65%

AMRコード：Enzo
解法：　　PPM
細分化比：4

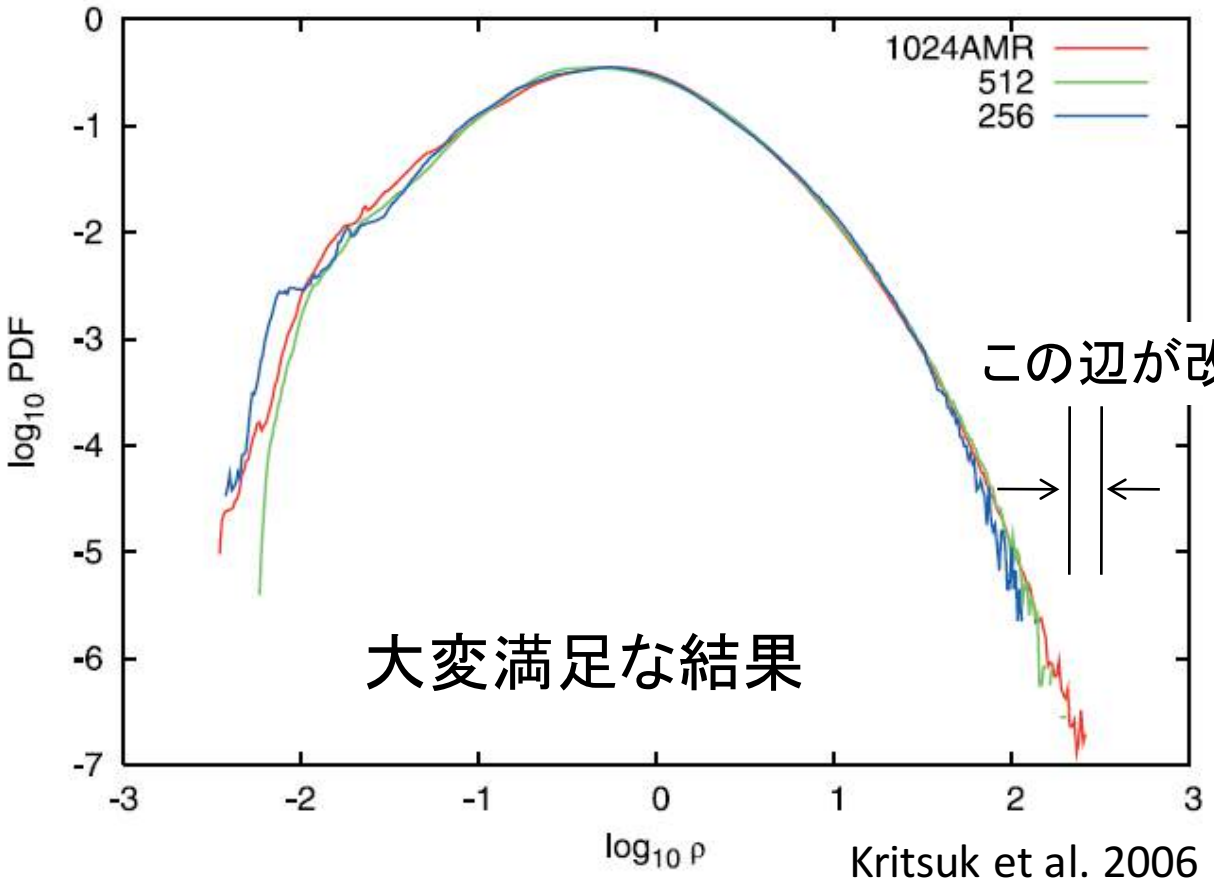
超音速乱流ではOK
亜音速乱流は微妙



超音速乱流のシミュレーション
密度分布（面でスライス）

Kritsuk et al. 2006

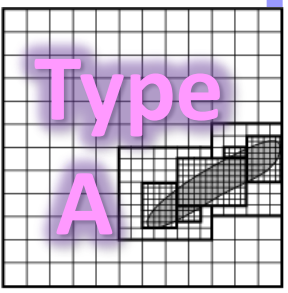
密度のPDF (確率分布関数)
AMRと一様格子の比較



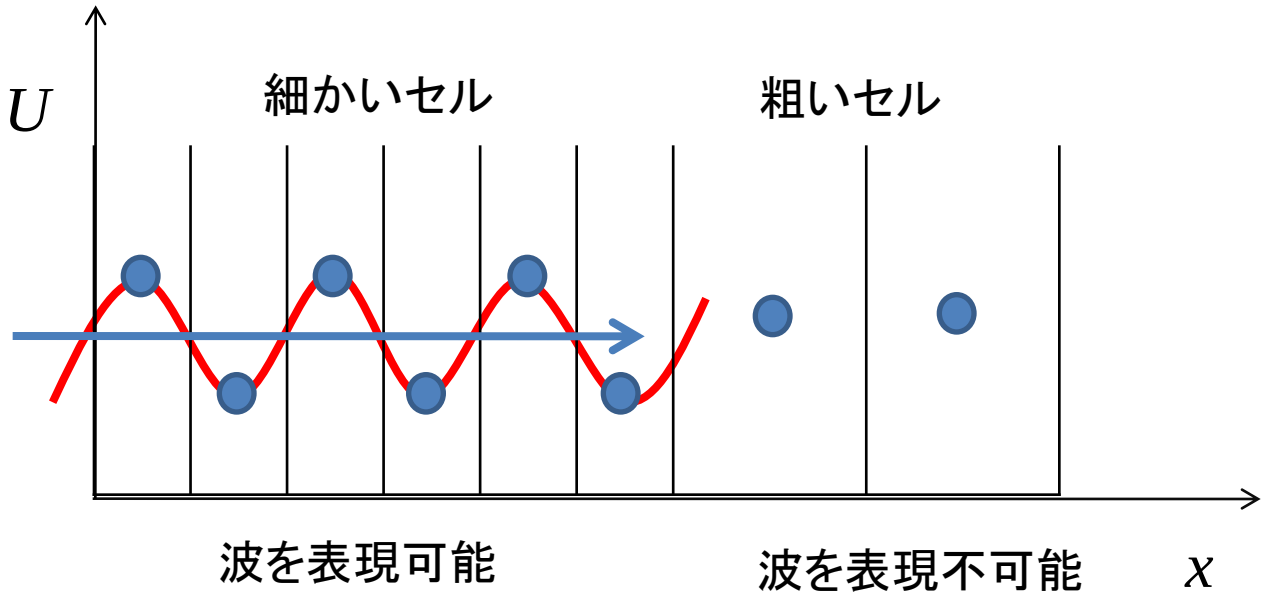
この辺が改善

大変満足な結果

Kritsuk et al. 2006

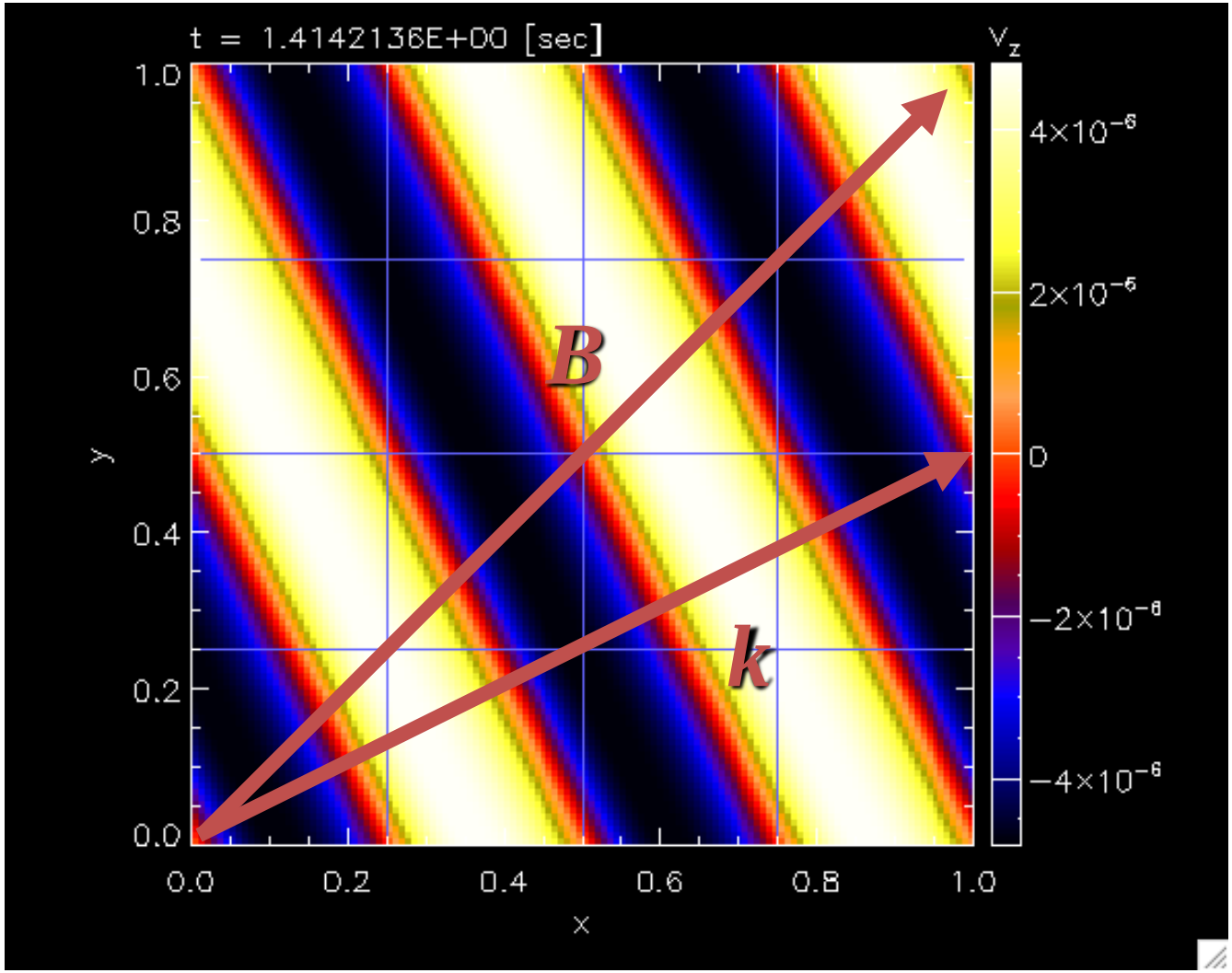


AMRはちゃんと作れば「波は反射しない」という「誤解」



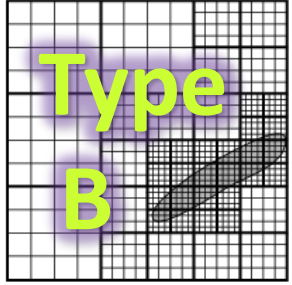
保存形式で解いているけど波のエネルギーはどこへ？
→　反射するしかない

Convergence test for MHD (1/2)
Alfven wave

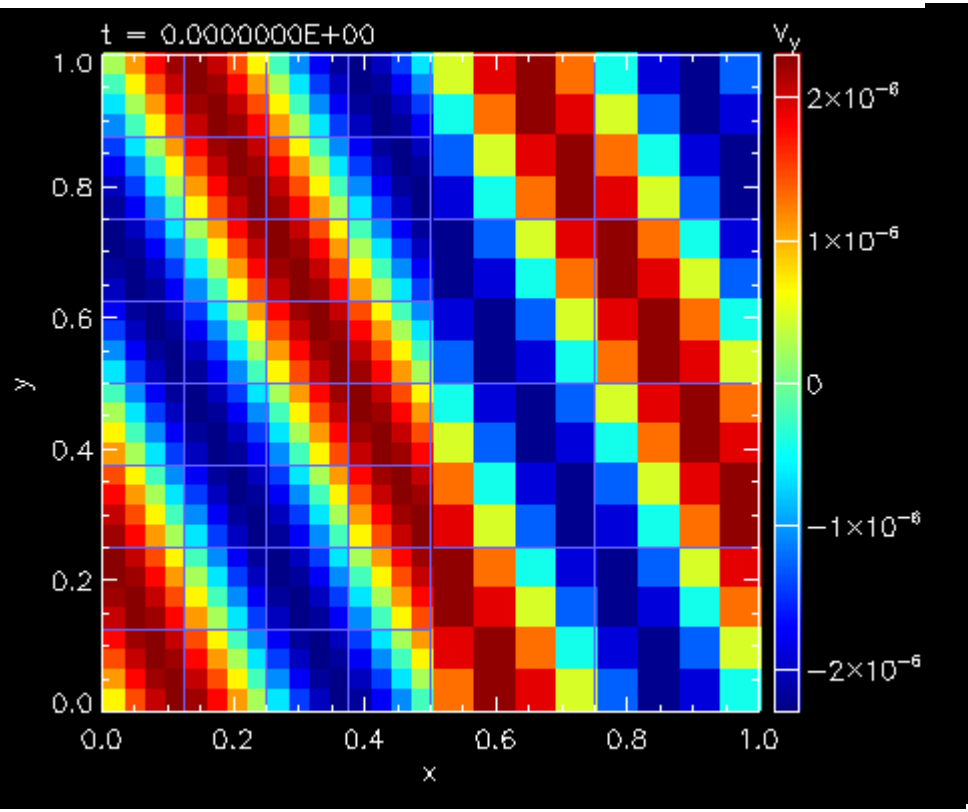


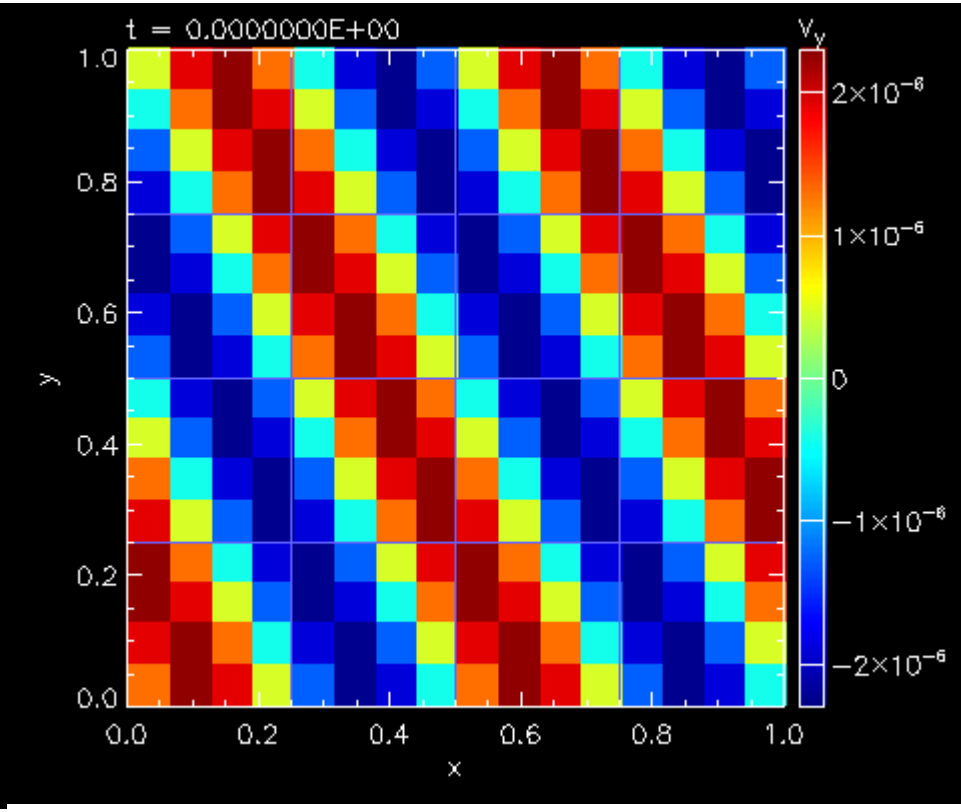
紙面に垂直な
vとBが振動する
sin波

しかし、波は境界で反射する。
Fast wave の反射実験。



SFUMATO

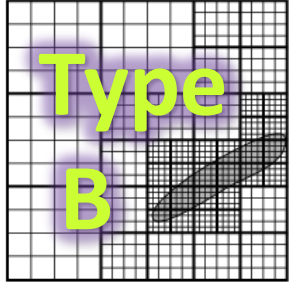




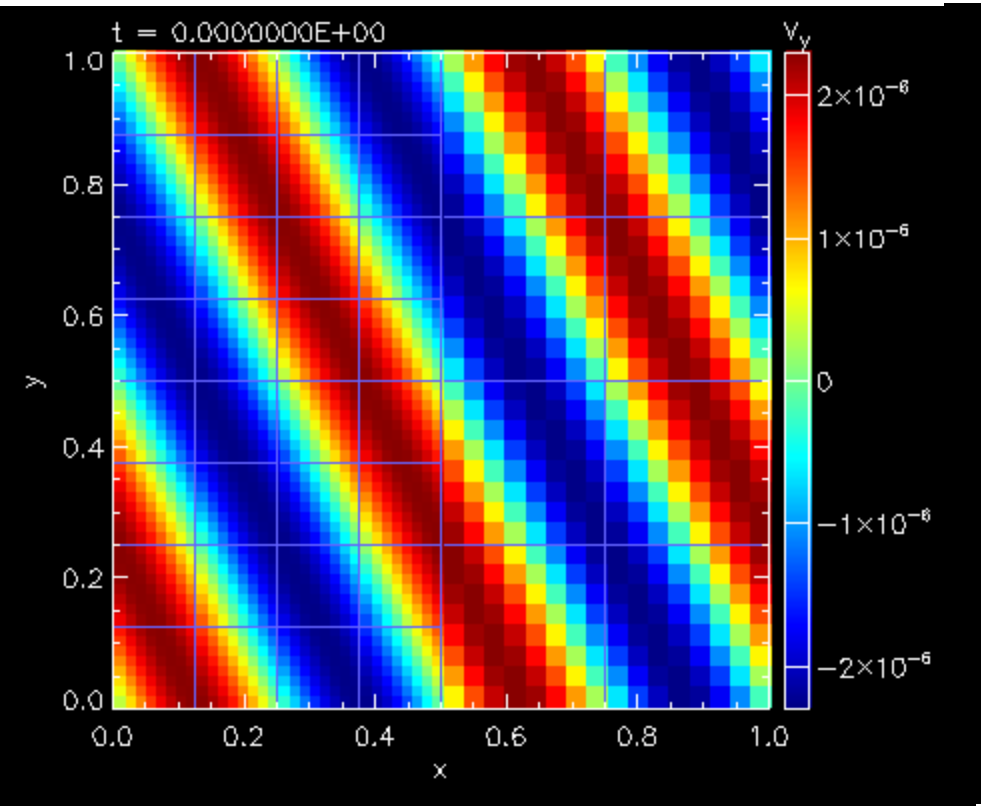
AMR
h = 1/32, 1/16

一様格子
h = 1/16

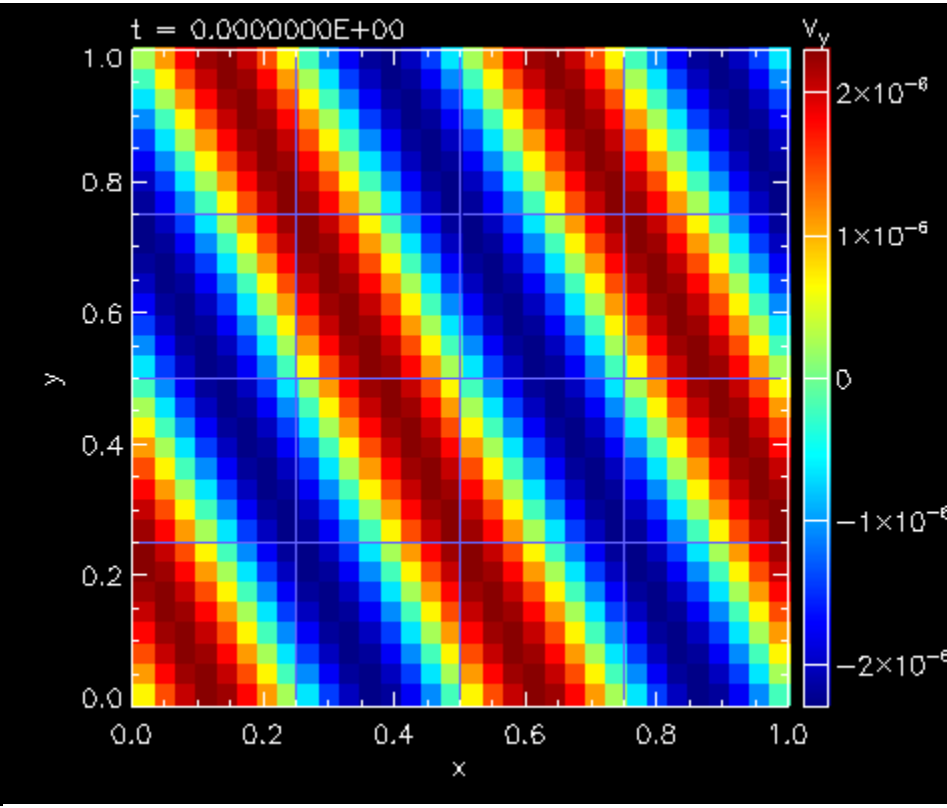
波は境界で反射する。
AMRでは減衰が少ない。



SFUMATO

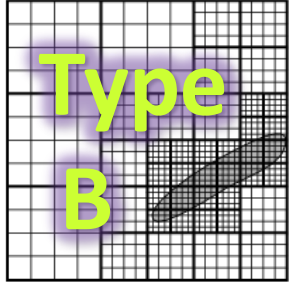


AMR
h = 1/64, 1/32

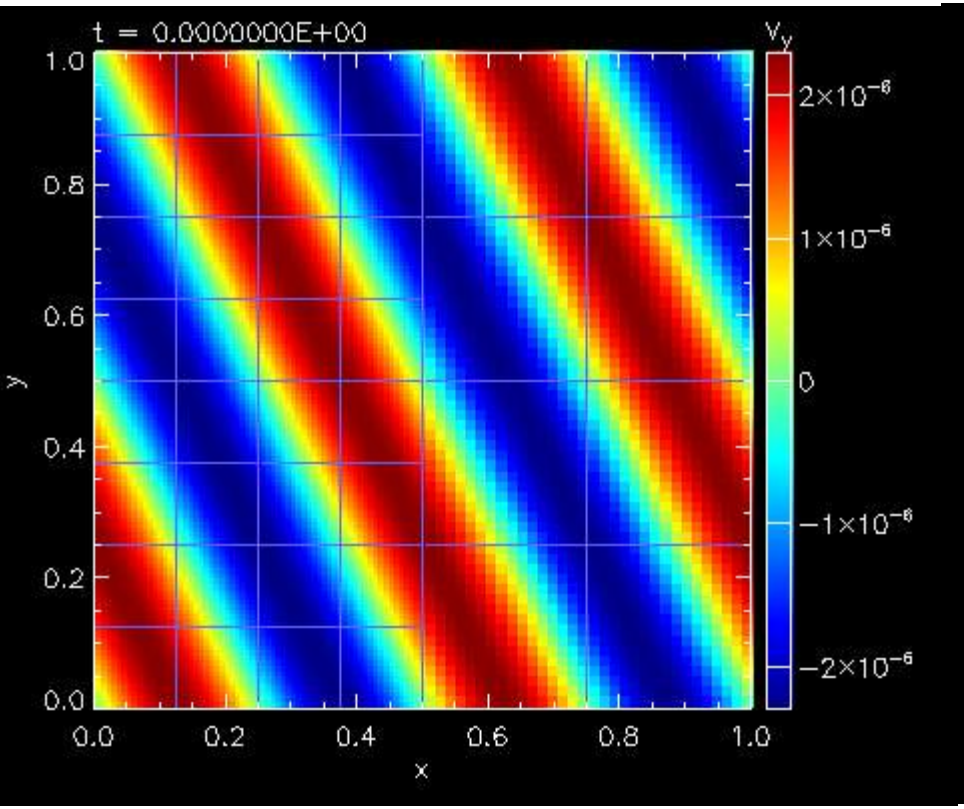


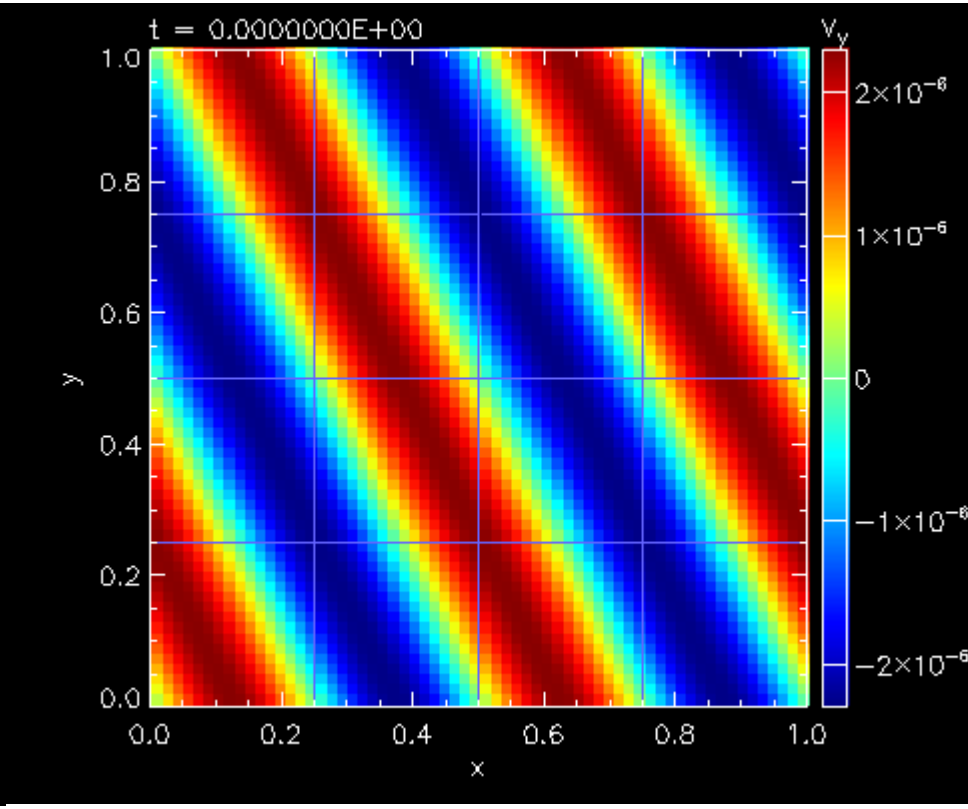
一様格子
h = 1/32

格子が細かくなると
反射は少なくなる。



SFUMATO

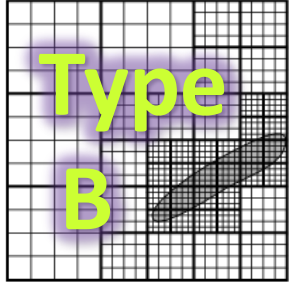




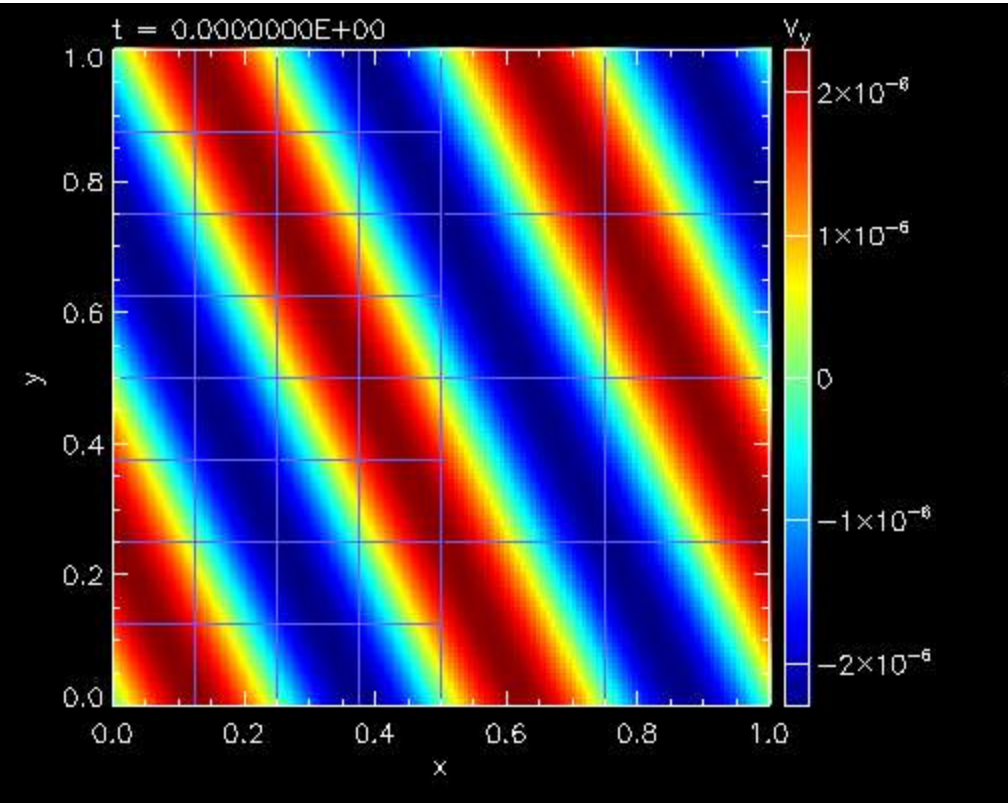
AMR
h = 1/128, 1/64

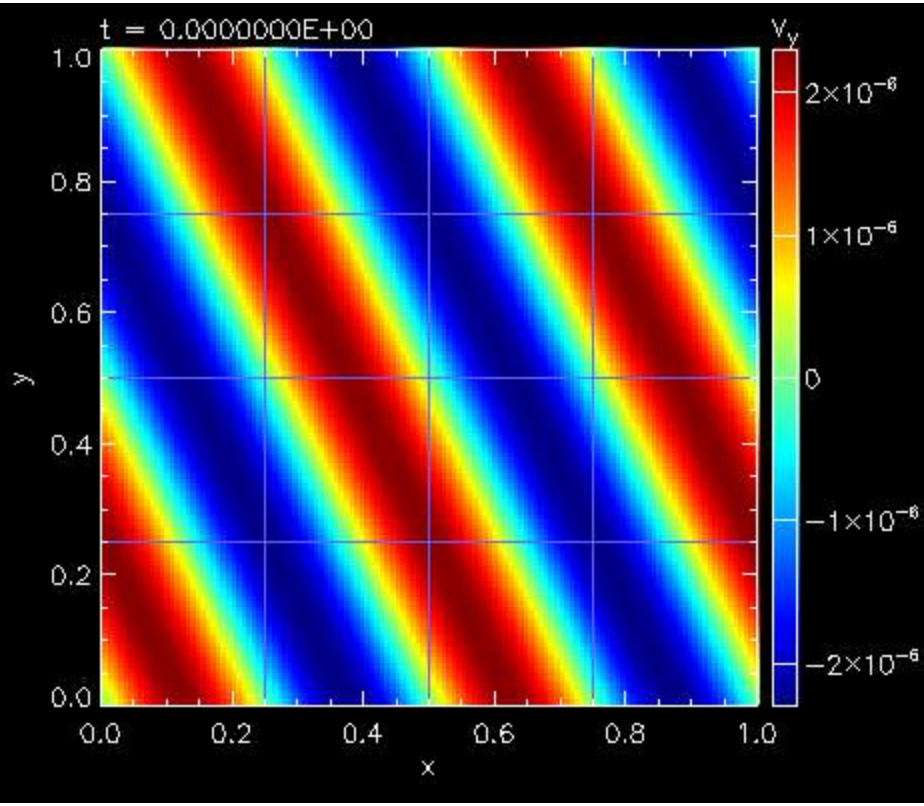
一様格子
h = 1/64

格子が細かくなると、
ほとんど反射は目視できない。



SFUMATO



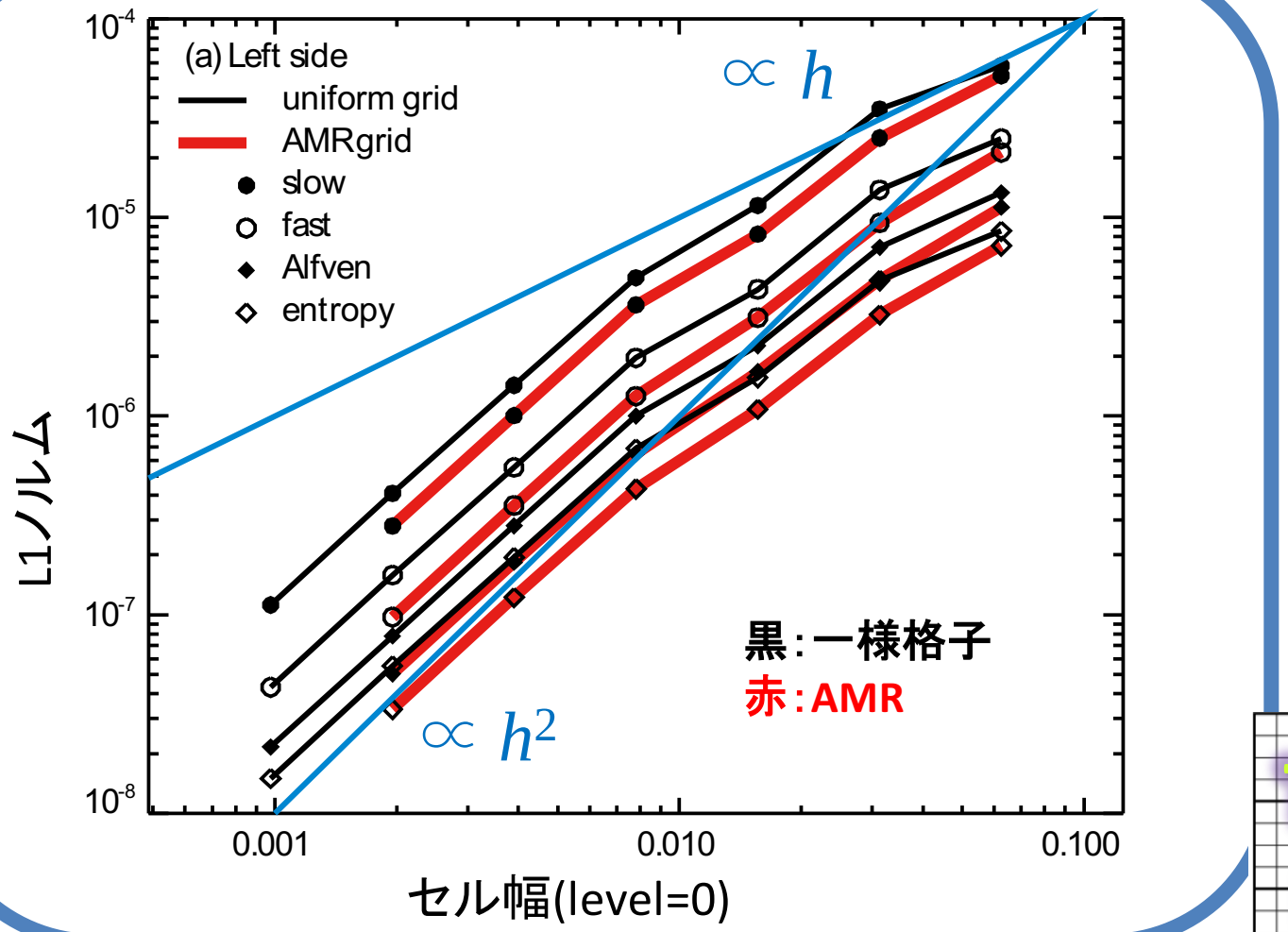


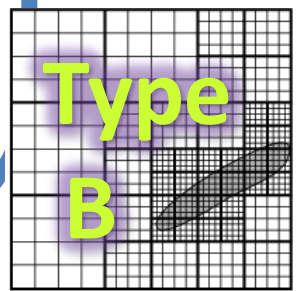
AMR
h = 1/256, 1/128

一様格子
h = 1/128

反射してもスキームの精度を達成する。
２次精度 @ h << 1
１次精度 @ h～1

SFUMATO





解法

AMR上の解法

- ブロック構造格子
  - ブロック自体は普通の一様格子。
  - 問題は、粗細（親子）ブロックの関係（時間的・空間的）
- 流体・MHD（双曲型偏微分方程式） $\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}+\frac{\partial H}{\partial z}=S$
  - 基本コンセプトは Berger & Colella (1989) に尽きる。
  - MHDでは、$\nabla\cdot B$ の処方箋に複数の流儀がある。
  - TVD, Roe法, predictor-corrector 法 (時間空間2次精度)
- 自己重力（楕円型偏微分方程式） $\nabla\Phi=4\pi G\rho$
  - 解像度が異なる格子が混在する下で、解を求める。
    全てのグリッドを同時に更新する反復法が必要。
  - お勧め参考書「MULTIGRID」 Trottenberg et al. (2001)
  - Multigrid 法 (FMG-cycle, V-cycle)

AMRが一様格子と異なる点

- 空間の分割方法（格子の配置）
- 時間の分割方法（時間ステップ）
- 細かい格子と粗い格子の境界の扱い

AMRの分類

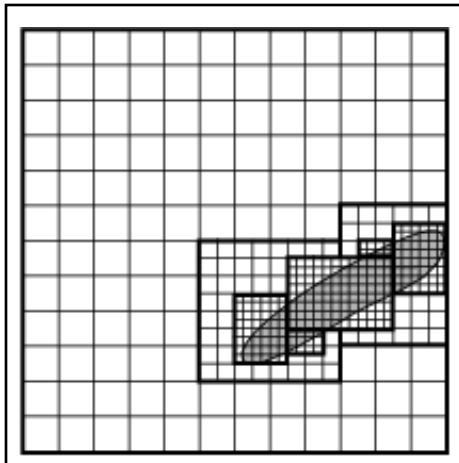
A) ブロック構造格子
- パッチ指向
- 最初のAMR
- Berger & Oliger 1984,
- Berger & Colella 1989

B) 自己相似型ブロック構造格子
- 八分木構造

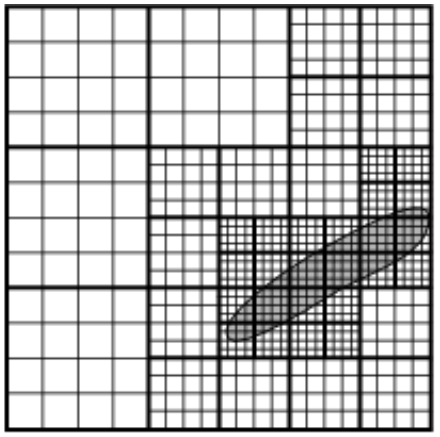
C) 直交非構造格子
- 八分木構造

D) 三角形非構造格子
- あまり用いられていない。
- 機体に沿った境界条件に有利。

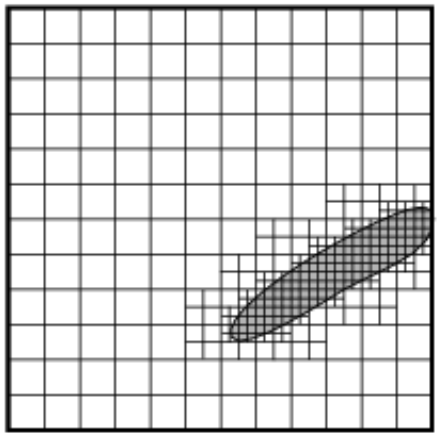
Level = 0 ～ 2



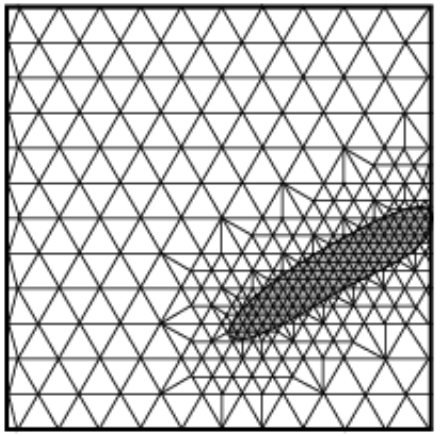
(A) ブロック構造格子



(B) 自己相似型
ブロック構造格子

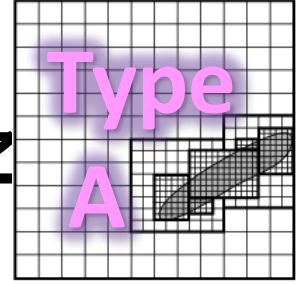


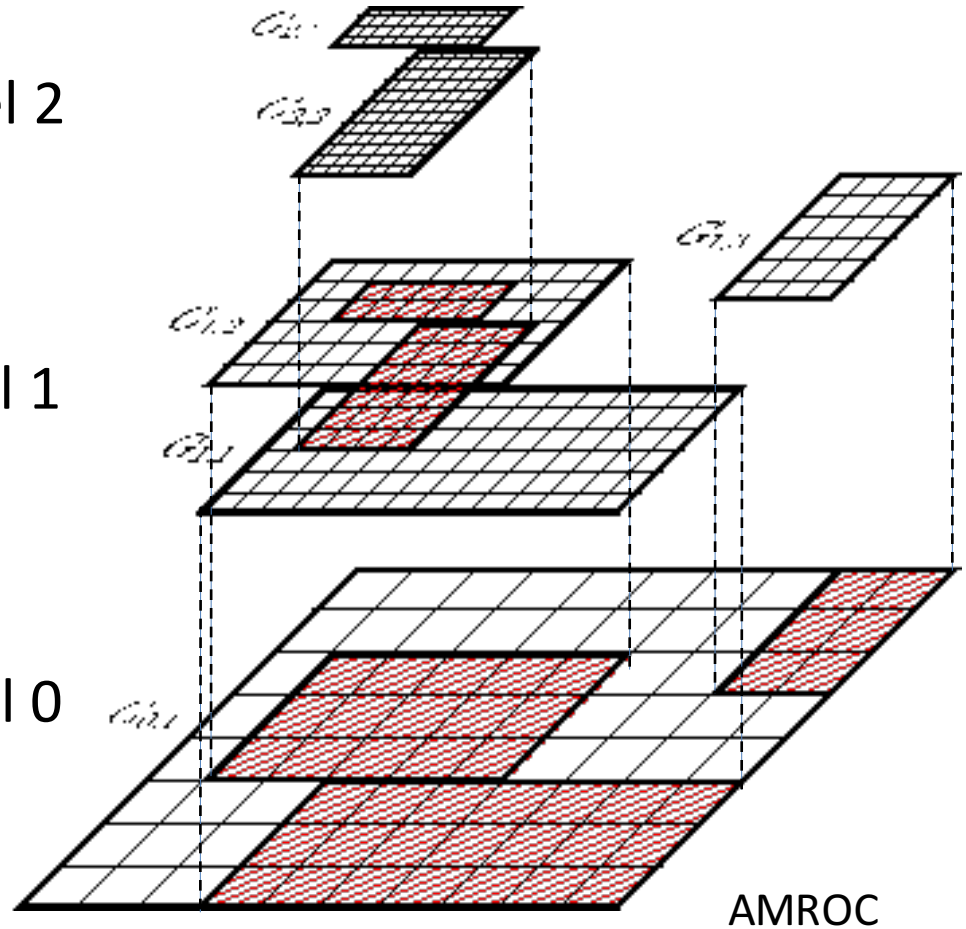
(C) 直交非構造格子



(D) 三角形非構造格子

パッチ指向ブロック構造格子



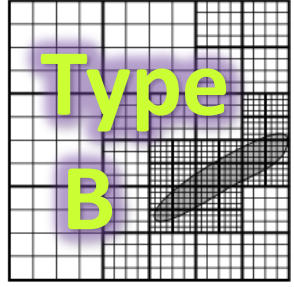


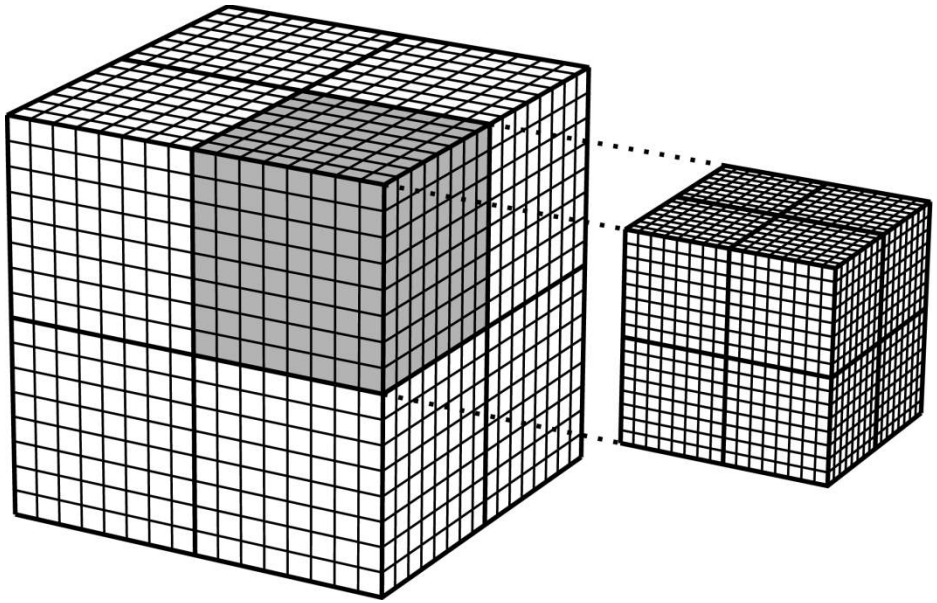
ブロックの大きさは様々（可変長）。

同じレベルのブロックが重なってもOK。

細分化比
= 2, 4, 8…

自己相似型ブロック構造格子





ブロックの大きさは固定（固定長）。

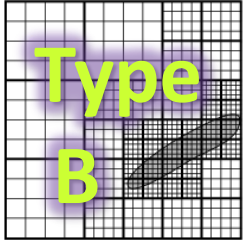
八分木によるデータ管理。

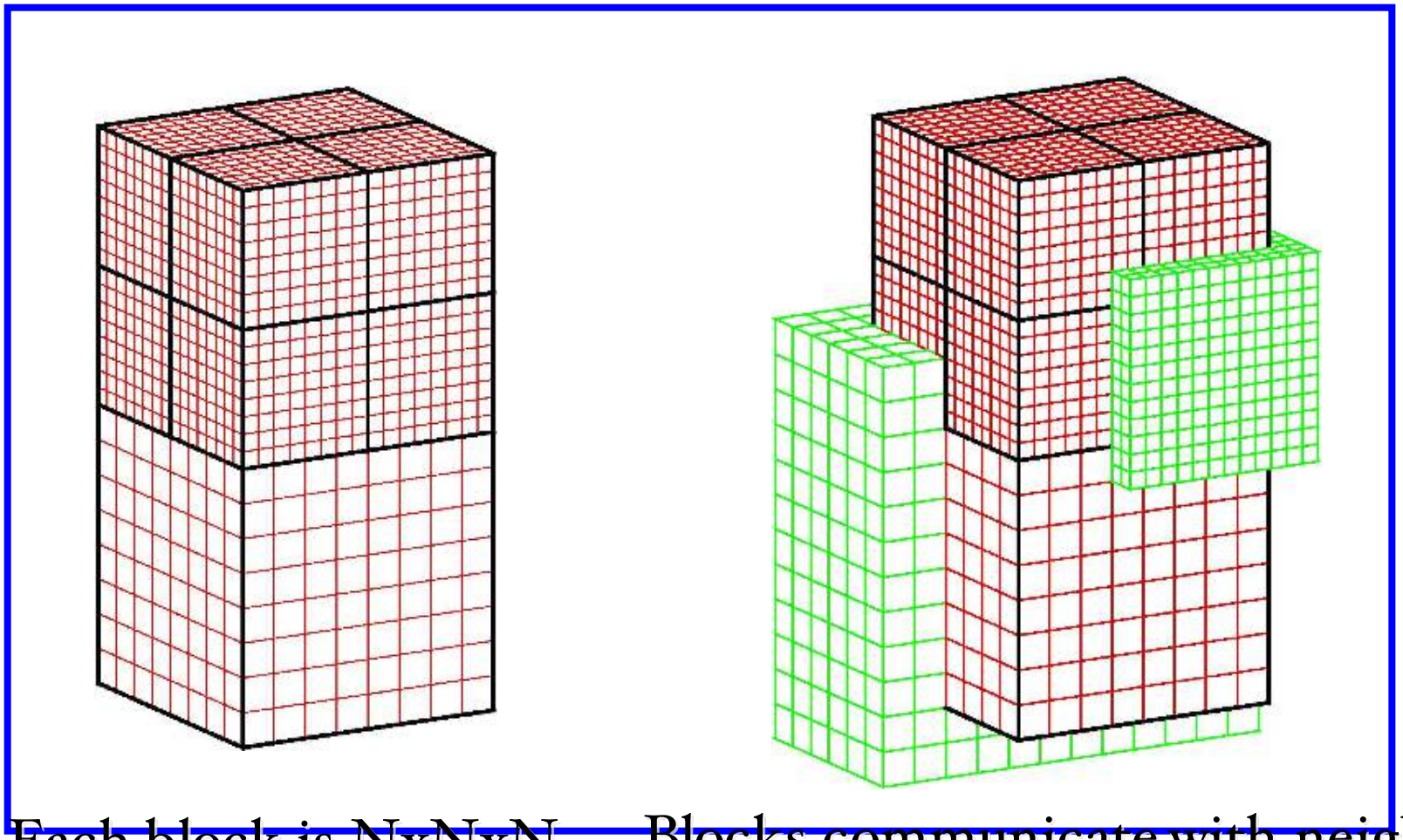
同じレベルのブロックは重ならない。

細分比 = 2

SFUMATO

Block Structure (BAT-R-US)
Self-similar oct tree type





Each block is NxNxN

Blocks communicate with neighbors through “ghost” cells

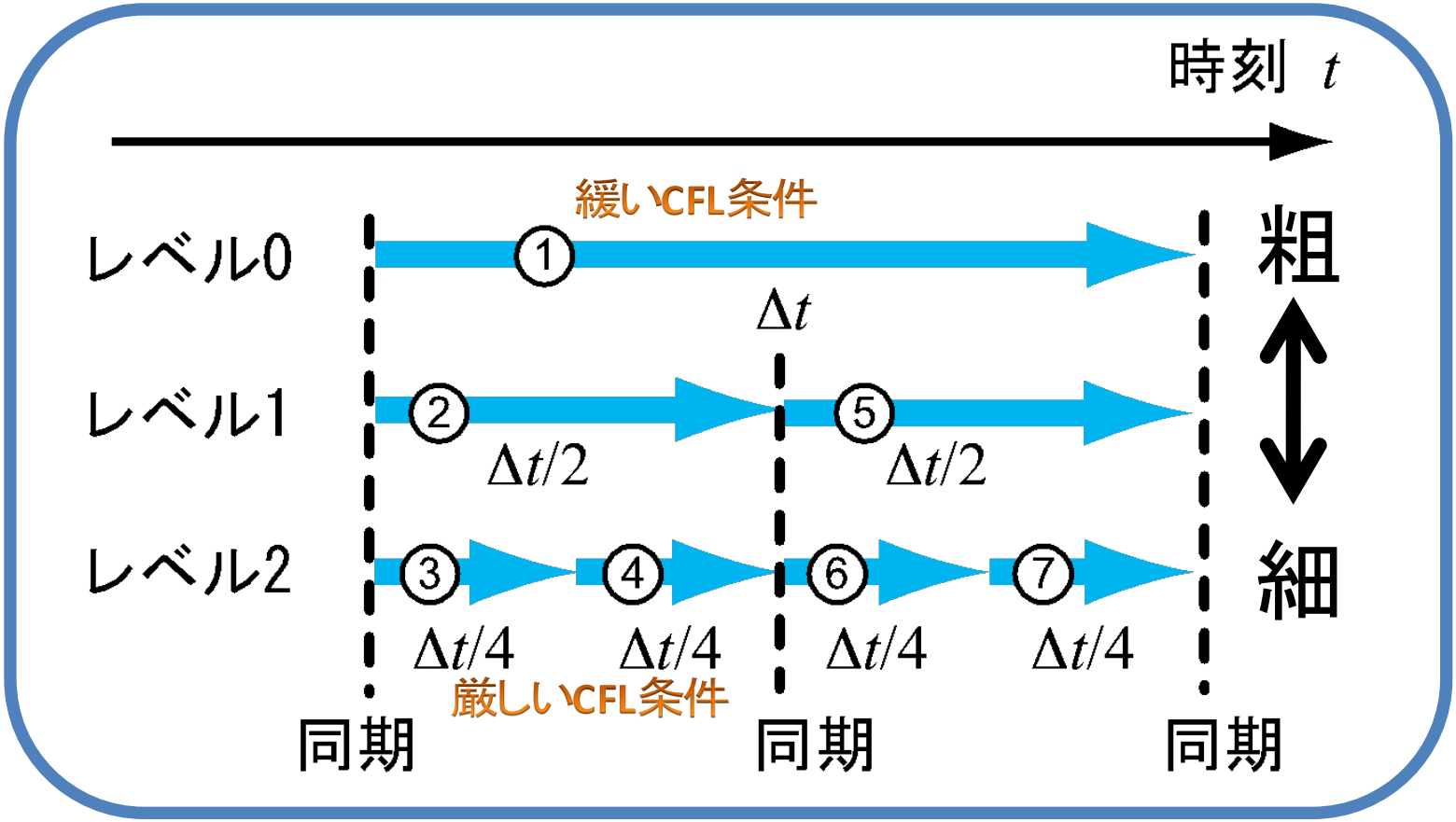
双曲型方程式の解法
AMRにおける時間発展

- Adaptive time step
  - 親のタイムステップ≧子のタイムステップ
  - C.f., Berger & Colella (1989)
  - 自己重力なし
- Synchronous time step
  - 全てのグリッドレベルが同じタイムステップを持つ。
  - C.f., FLASH
  - 自己重力系、輻射流体、（長距離相互作用）

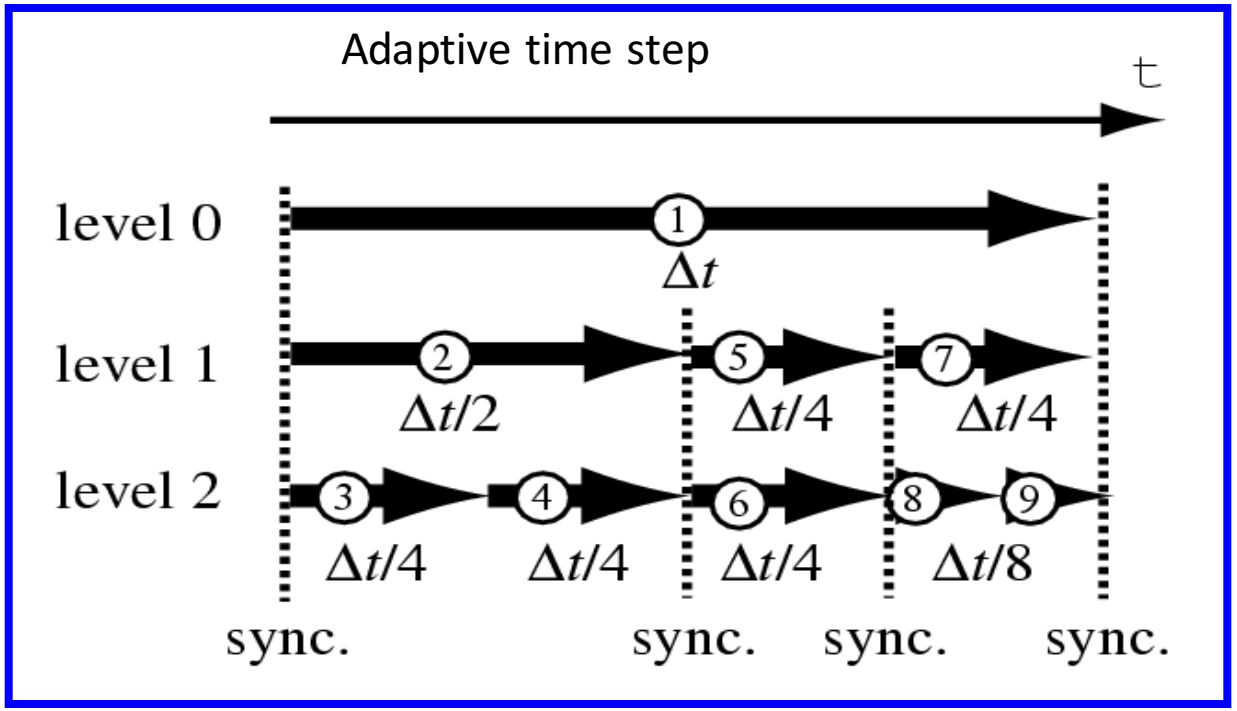
CFL条件

$$\Delta t < \frac{\Delta x}{v}$$

適合時間ステップ(adaptive time step)
時間も適合細分化



適合時間ステップ(adaptive time step) オプション

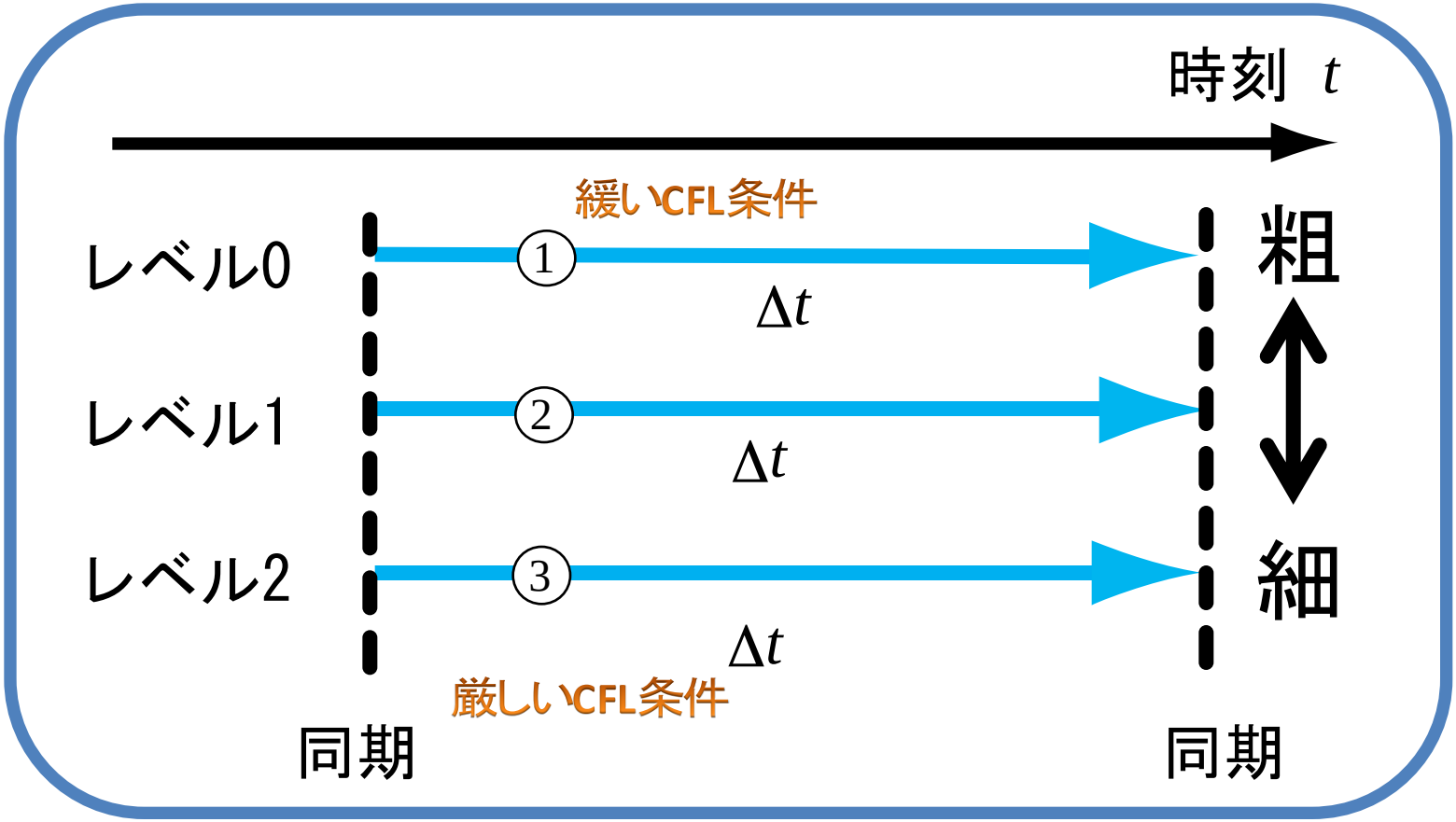


子グリッドの時間刻み $\Delta t_{子} = \Delta t_{親}/2^{n}$ $(n=0, 1, 2, \ldots)$

子グリッドがCFL条件を破るのを防ぐ

子グリッドが律儀に親のCFL条件を守る制限を解除する。

同期時間ステップ(synchronous time step)
時間を適合細分化しない



Adaptive vs synchronous time steps

Adaptive time step

もっとも細かいグリッドレベルが1ステップ進むためのコスト
親のグリッドレベルは1/2ステップ
祖父のグリッドレベルは1/4ステップ

$$1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{l\max}}=\sum_{l=0}^{l\max}\frac{1}{2^{l}}\approx 2 \qquad (l_{\max} >> 1)$$

トータルで2倍コストがかかる

Synchronous time step

もっとも細かいグリッドレベルが1ステップ進むためのコスト
親のグリッドレベルも1ステップ
祖父のグリッドレベルの1ステップ

$$1+1+1+\cdots+1=\sum_{l=0}^{l\max}1=l_{\max}+1\approx l_{\max} \qquad (l_{\max} >> 1)$$

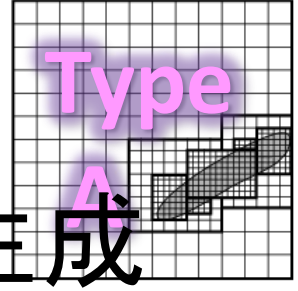
トータルで段数倍コストがかかる

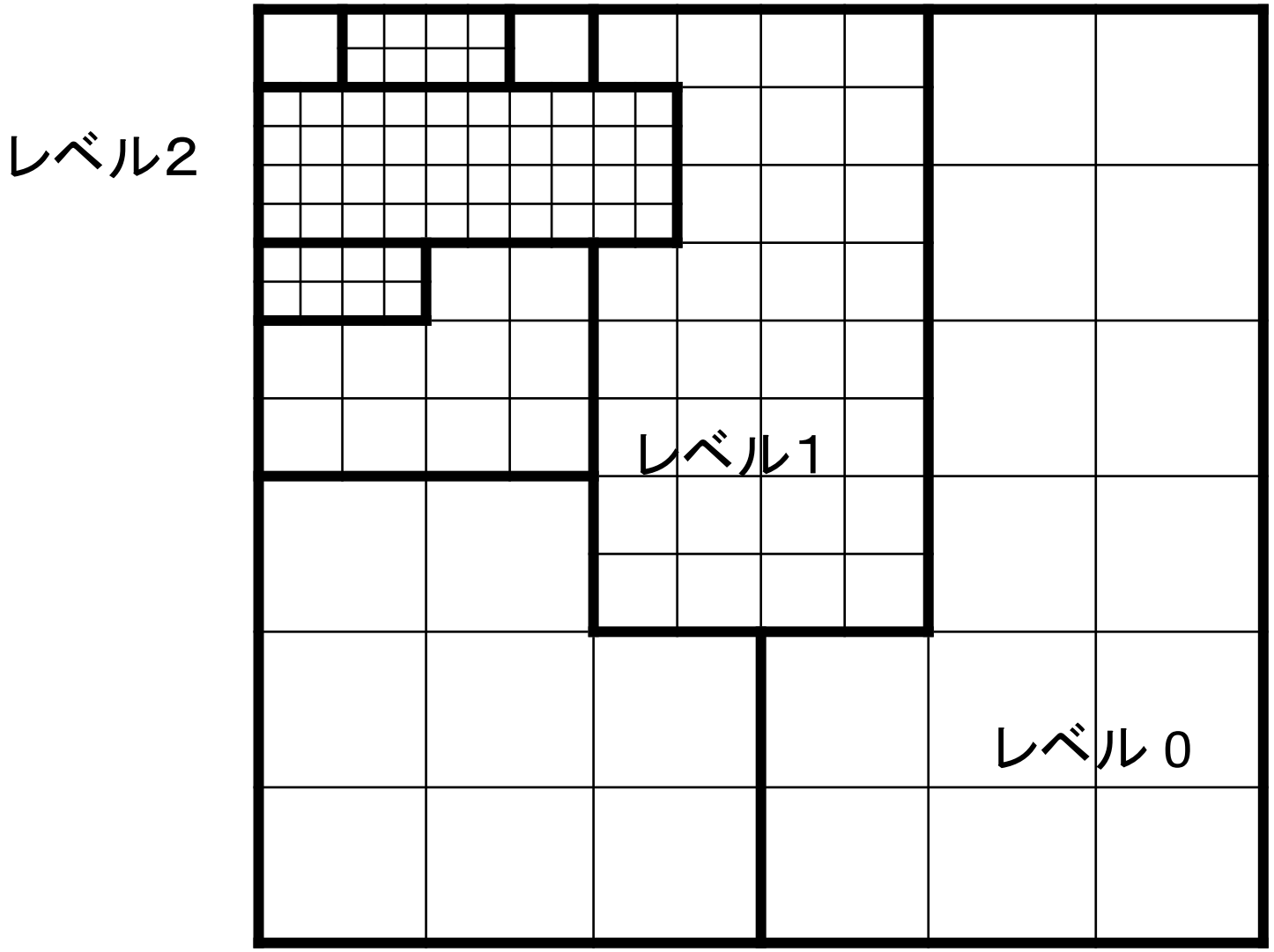
特別な理由がない限り、adaptive time step を採用するべき。
特別な理由の例：自己重力、輻射

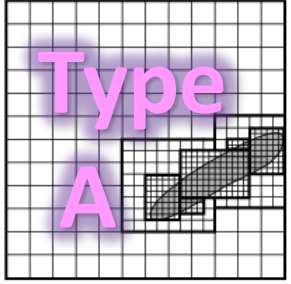
細分化のアルゴリズム

1. 細分化するセルの探査
   a. レベル l に属するセルを順に探査し、細分化条件を満たすセルに印をつける。
   b. 印がついたセルの周囲のセルにも印をつける。
   c. レベル $l+2$ にブロックと重なるセルにも印をつける。
2. ブロック配置の決定
   a. 印がついたセルを含むようにレベル $l+1$ のブロックを作る
3. ブロックの生成
   a. 当該セルがすでに細分化されていれば、その値をコピーする。
   b. 新たに細分化する場合には、レベル l の値を内挿する。
4. 古いブロックの破棄
5. 手順1～4を同期しているレベルで行う。
   a. 粗いレベルから細かいレベルの順に繰り返す。

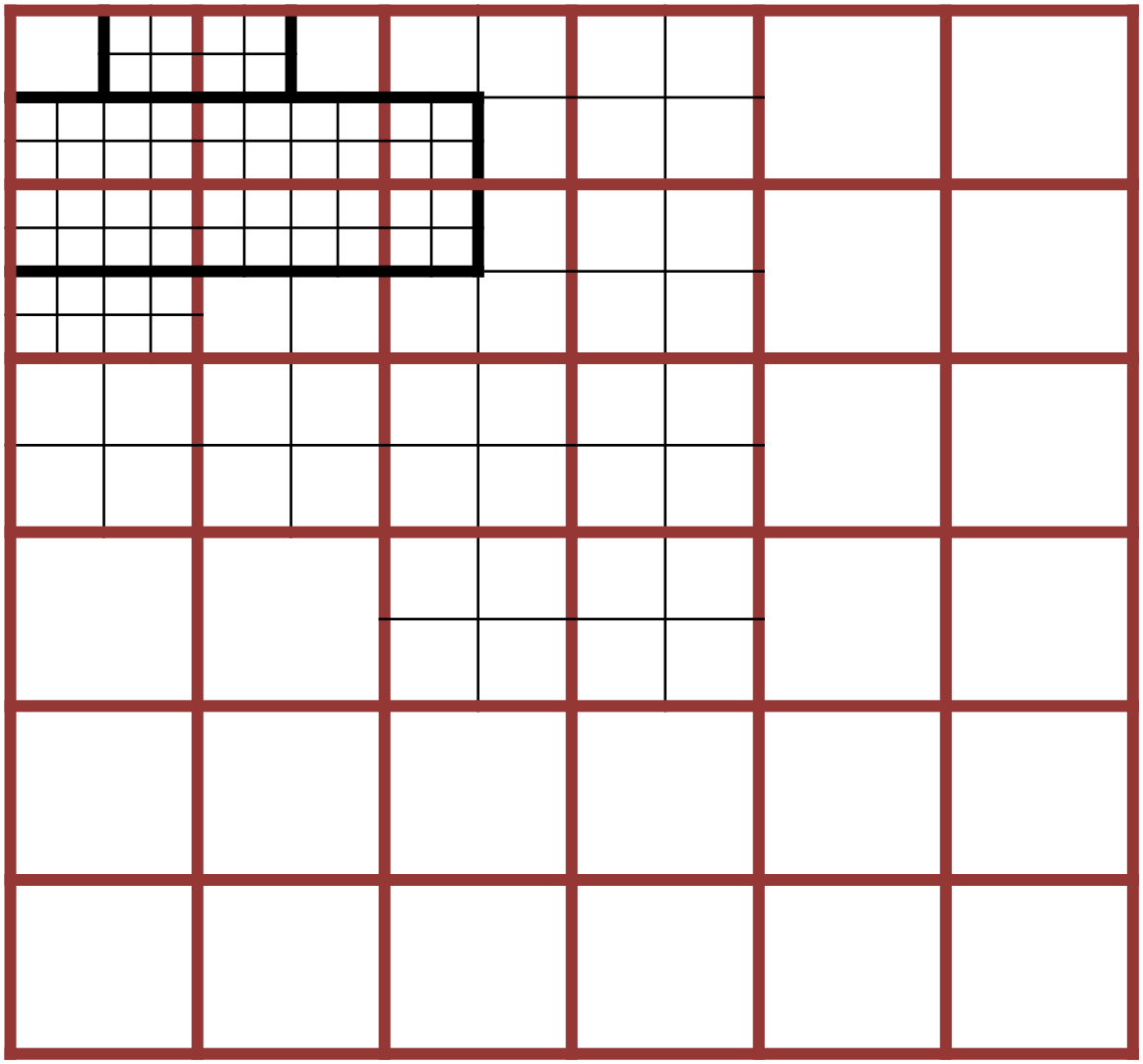
細分化のアルゴリズム
レベル0を細分化してレベル1を生成



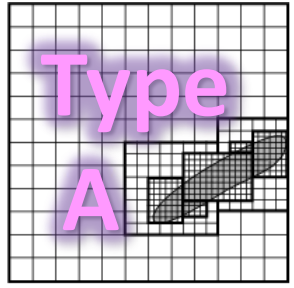


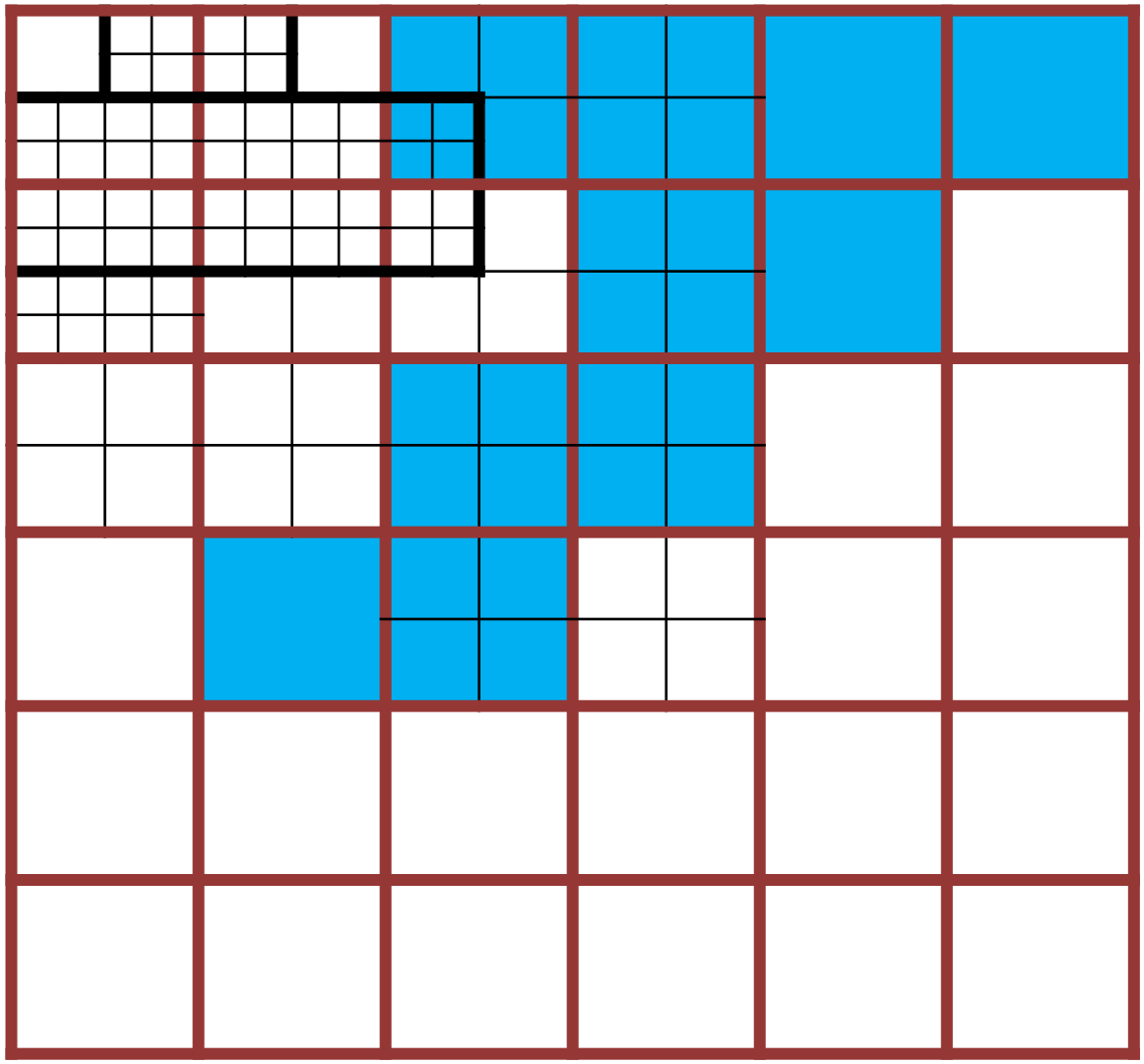


1. レベル0のセルに注目する。

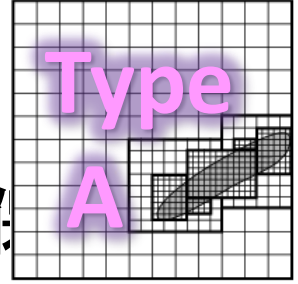


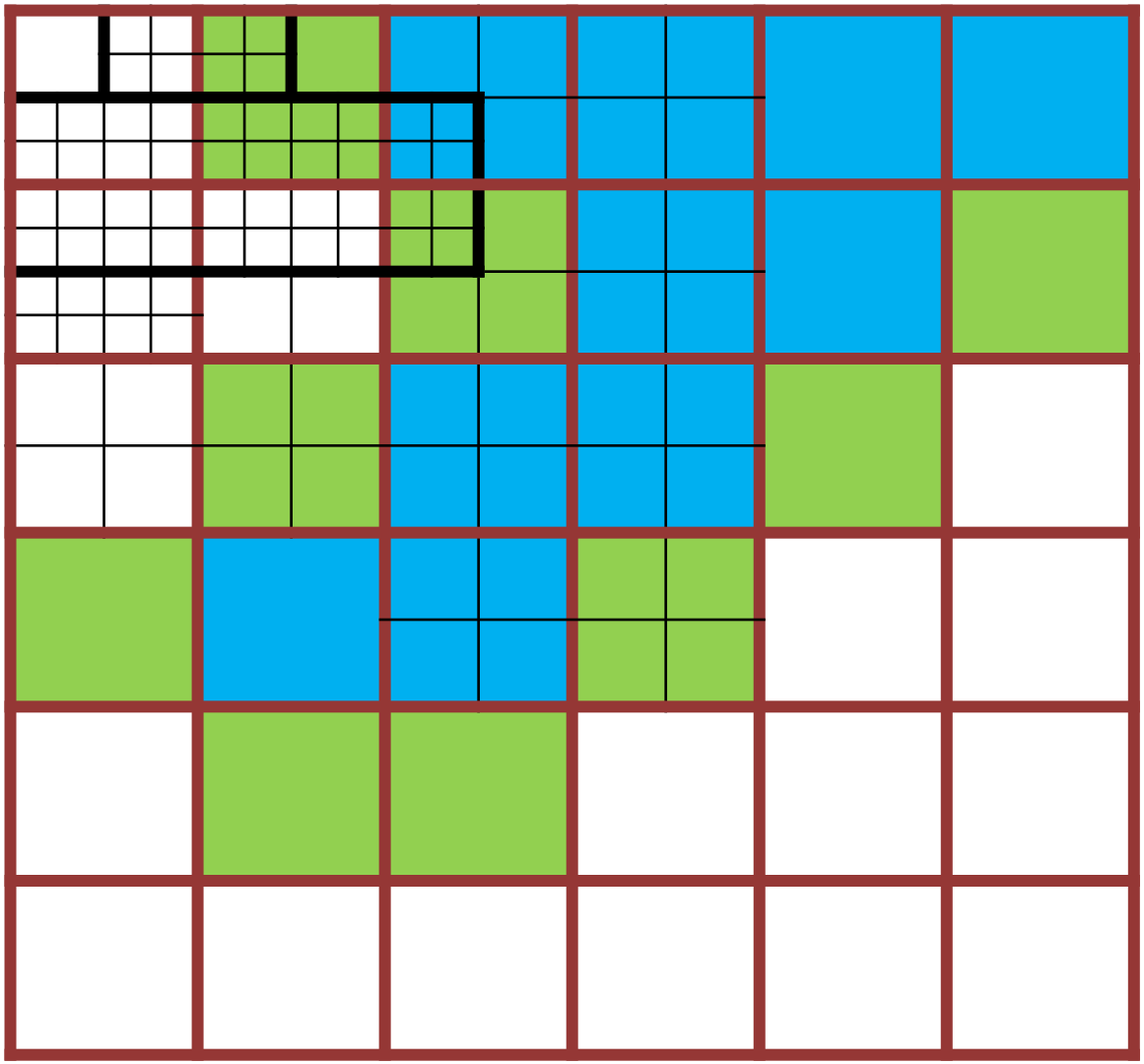
1-a. レベル0の細分化。
レベル0のセルに対して細分化条件を評価する。



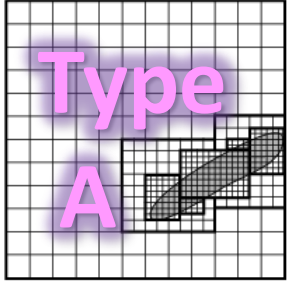


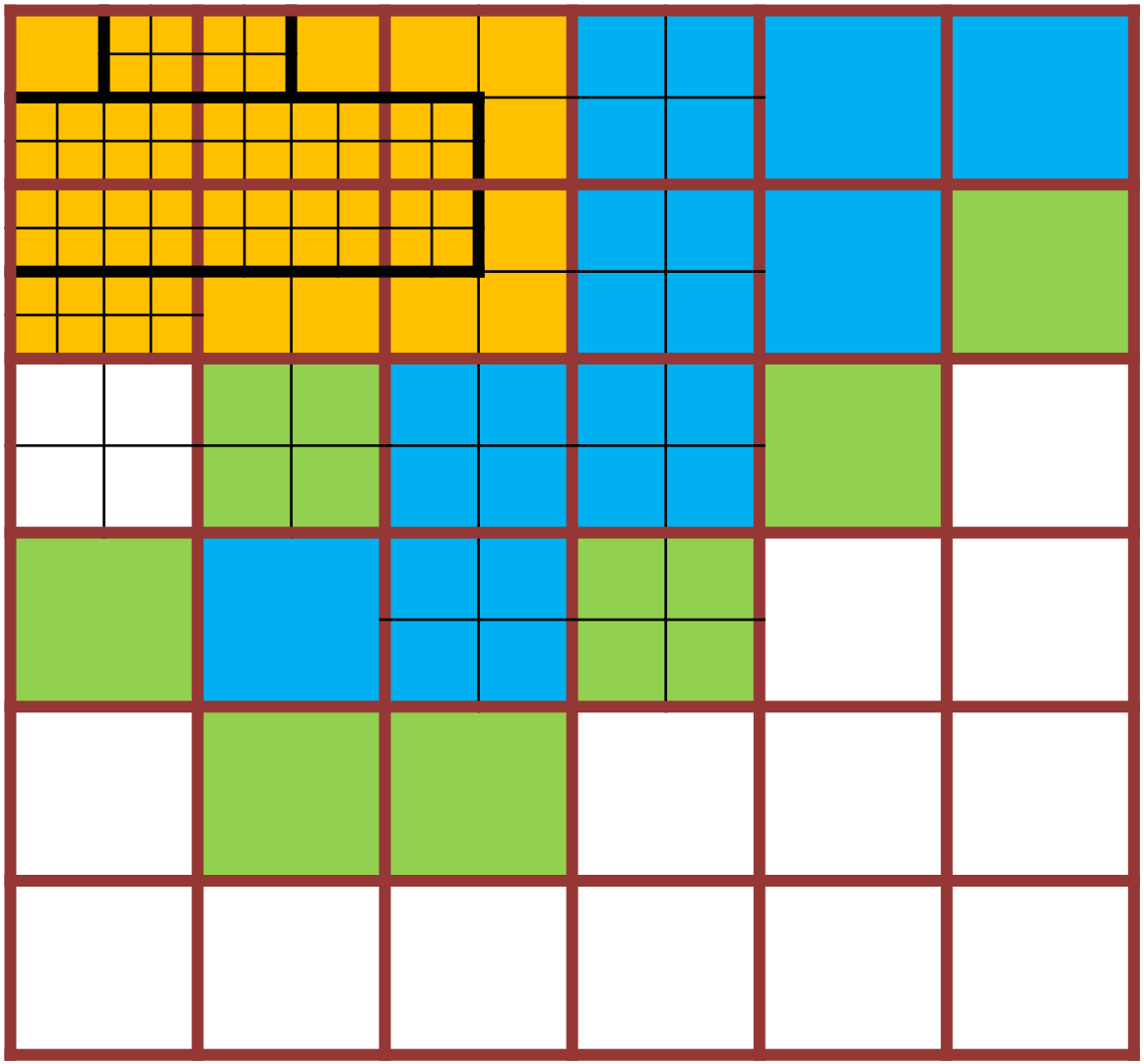
1-b. レベル0の細分化。
周囲の１～２セルにも印をつける。マージンの確保



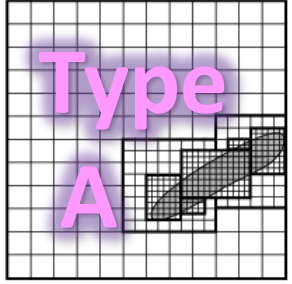


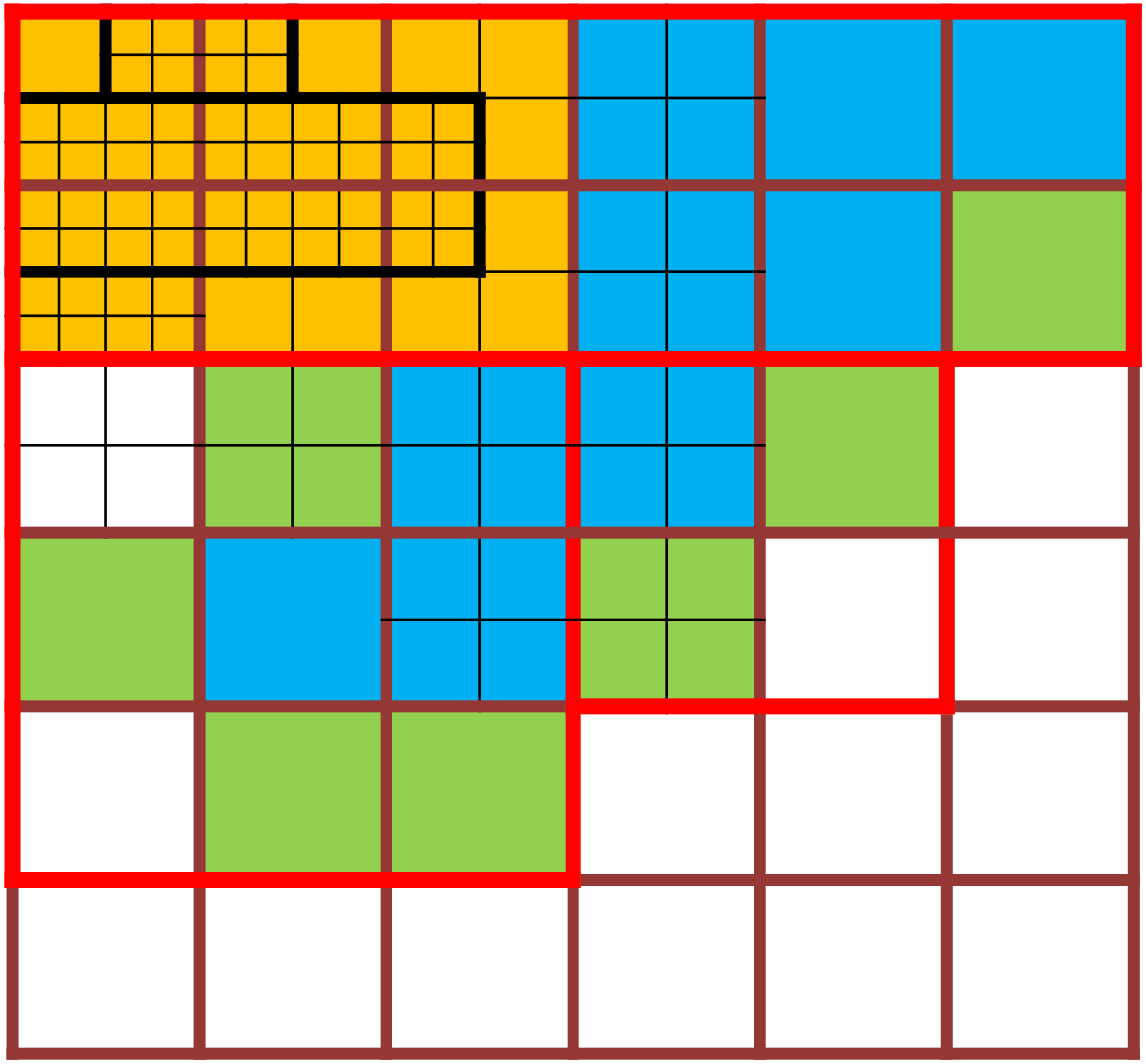
1-c. レベル0の細分化。
レベル2と重なるセルにも印をつける。



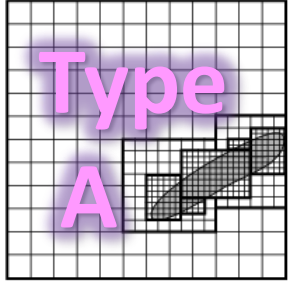


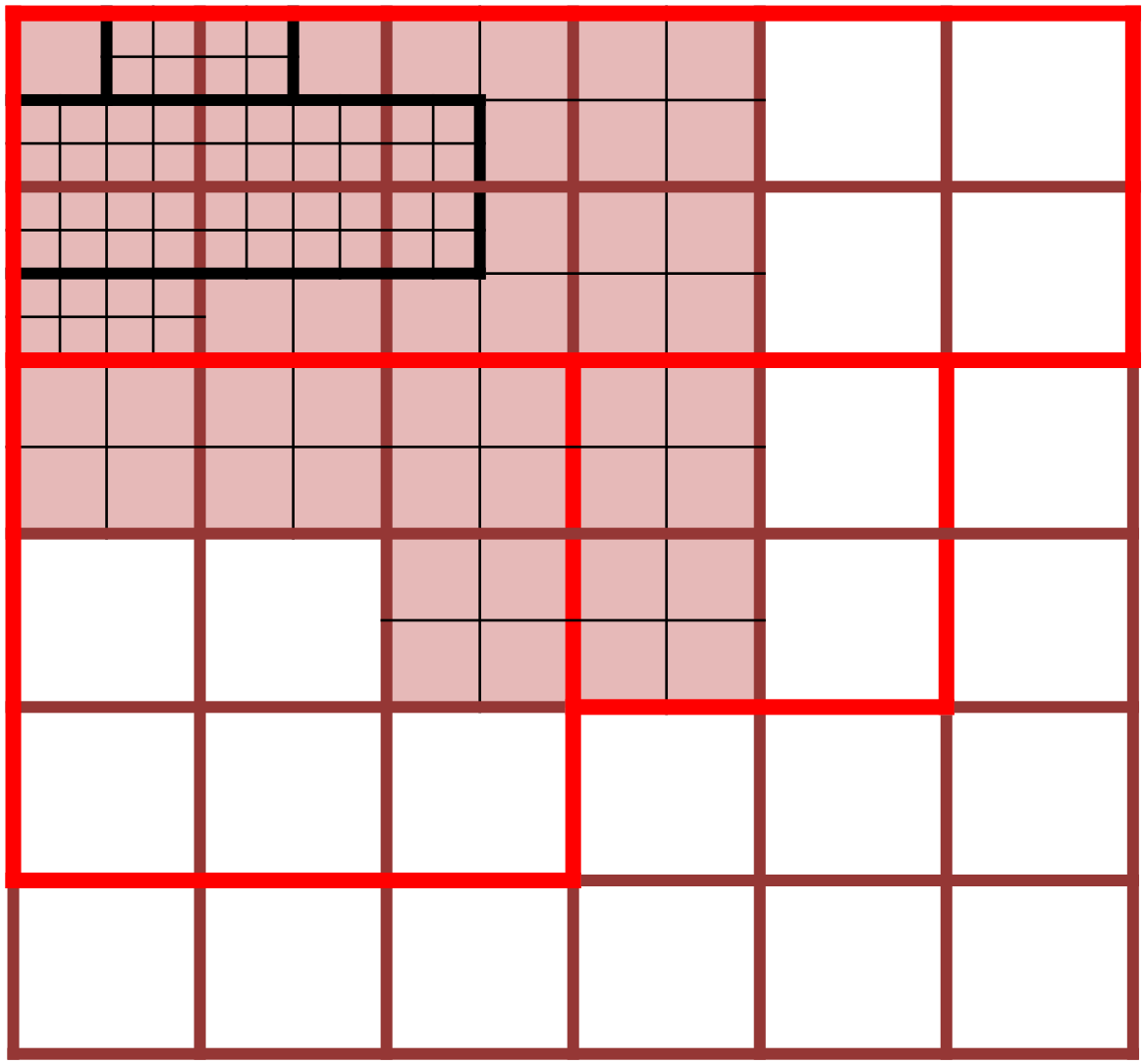
2. レベル1のブロック配置を決定。
ここはいろいろな流儀がある。



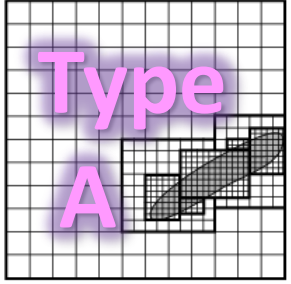


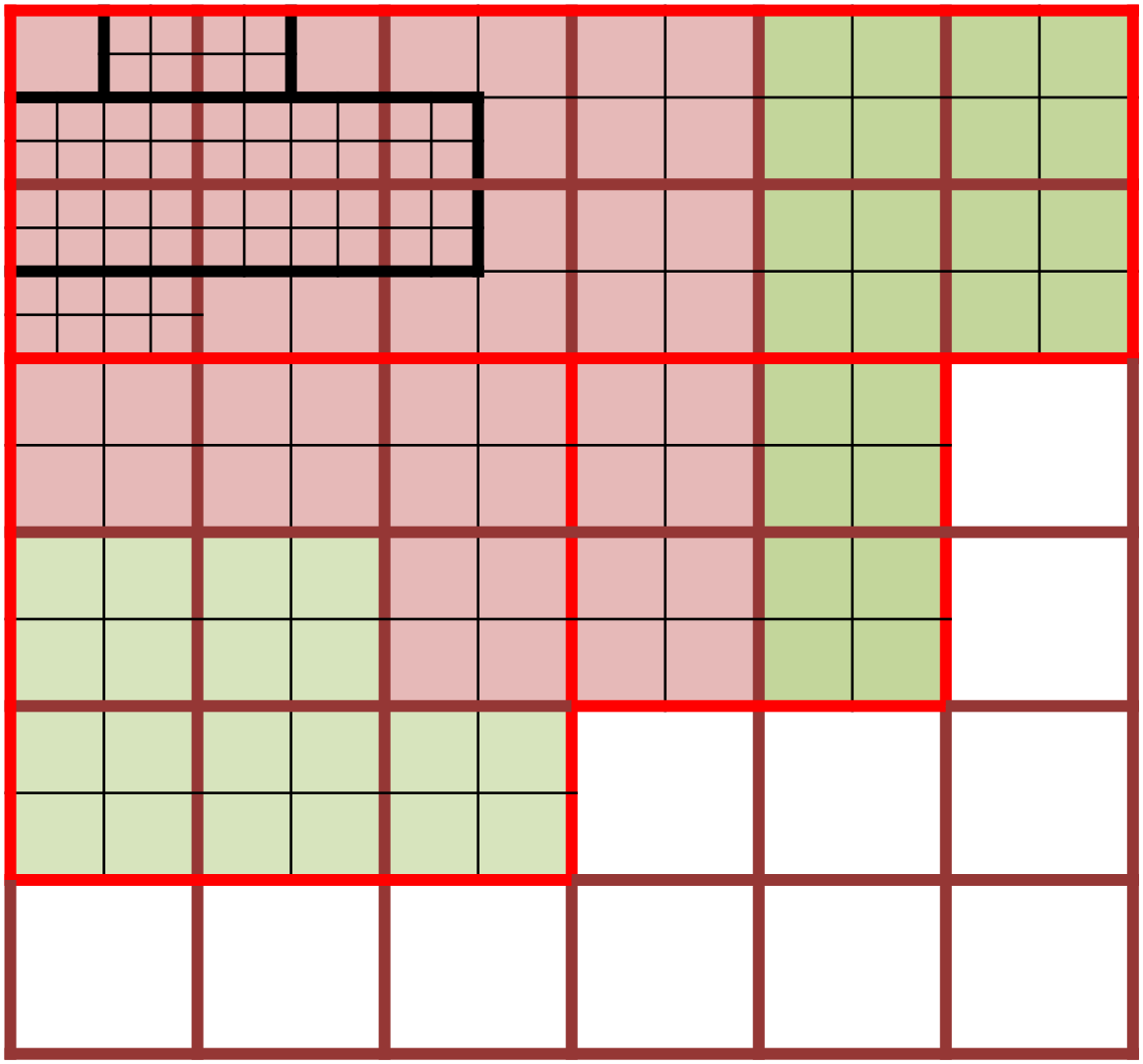
3-a. ブロックの生成。
既存のセルの値をコピー。

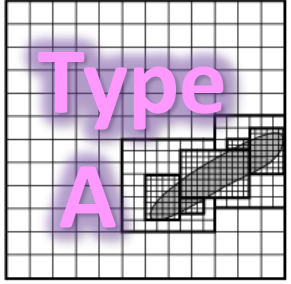




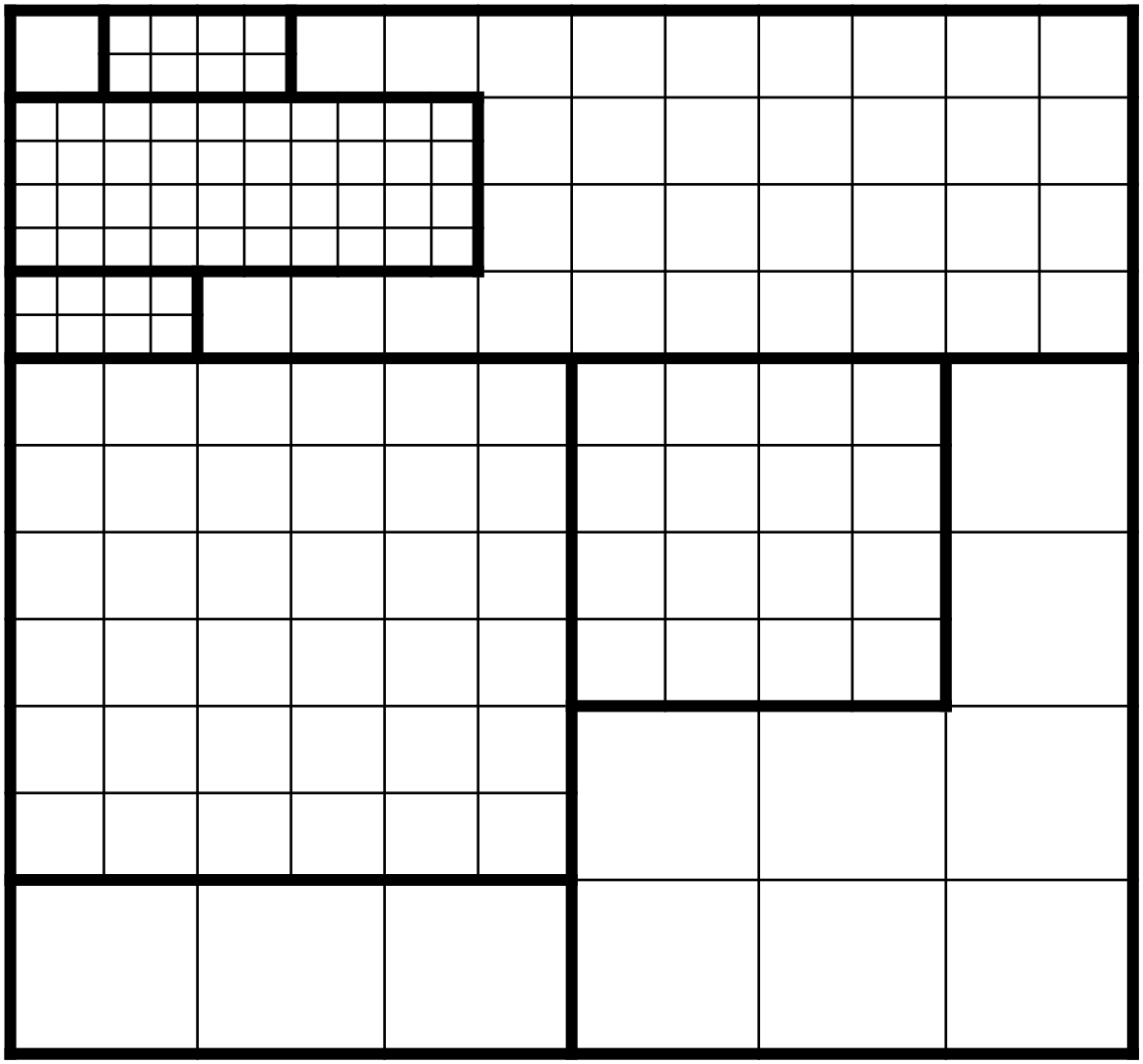
3-b. ブロックの生成。
レベル0から内挿。





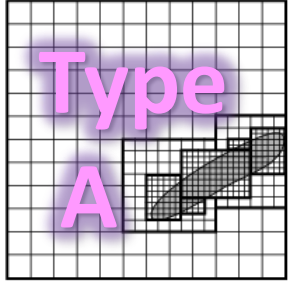


4. 古いブロックを破棄して完成。



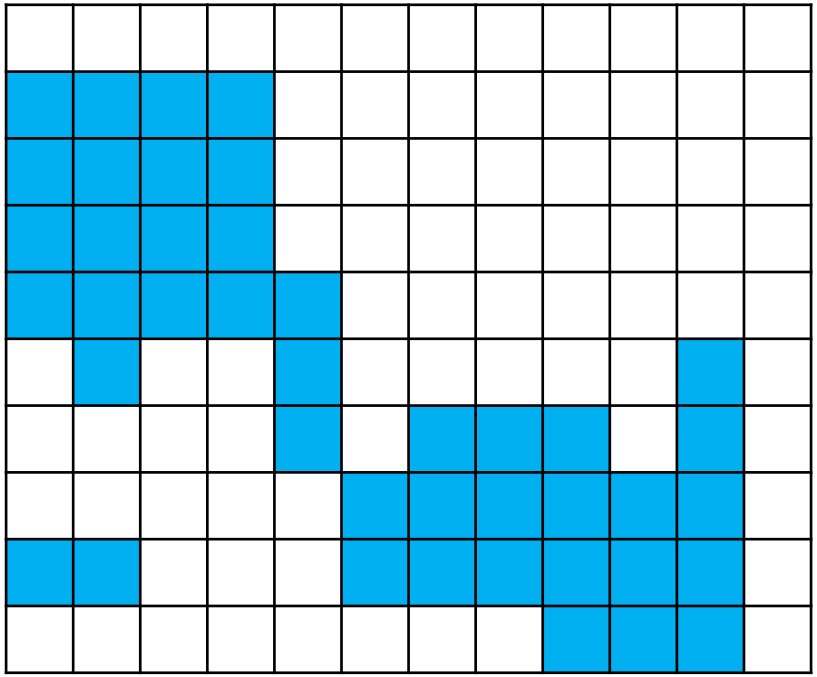
細分化条件
どこを細分化するべきか？

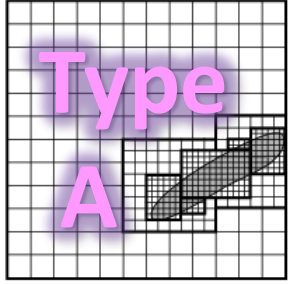
- 何を見たいかに依存する。
- 打切り誤差が閾値を超える部分を細分化 (Berger & Colella 1989)
- 物理量の2階微分が閾値を超える部分を細分化 (Flash など)
- 星形成分野の業界標準
  - Jeans 条件 : Jeans 波長を4メッシュ以上で分解する (Truelove et al.)



ブロックの決定方法

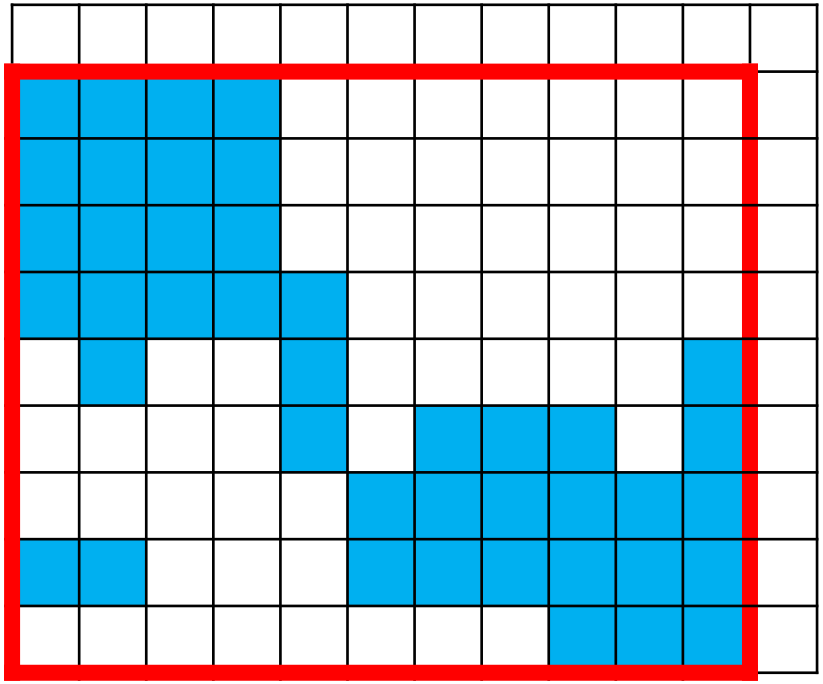
パッチ指向AMRには様々な流儀が存在する。
たとえば、

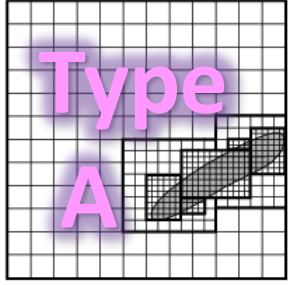




ブロックの決定方法

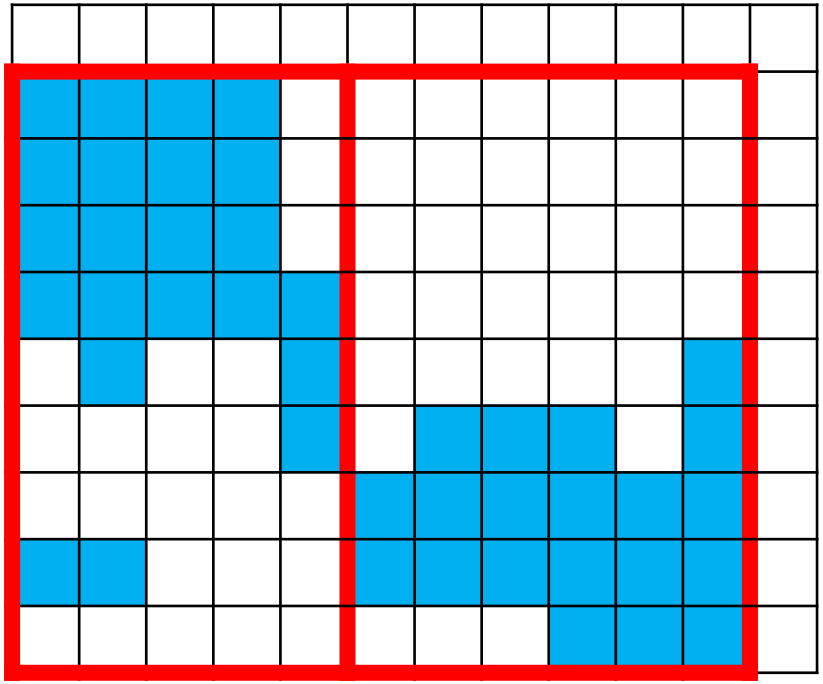
印がついたセルをすべて覆うブロックを考える





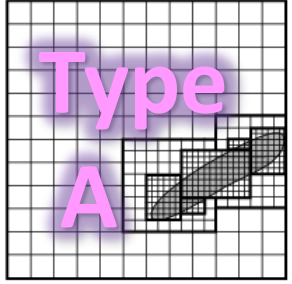
ブロックの決定方法

セル数のヒストグラムを書き、最小値部分でブロックを分割する。



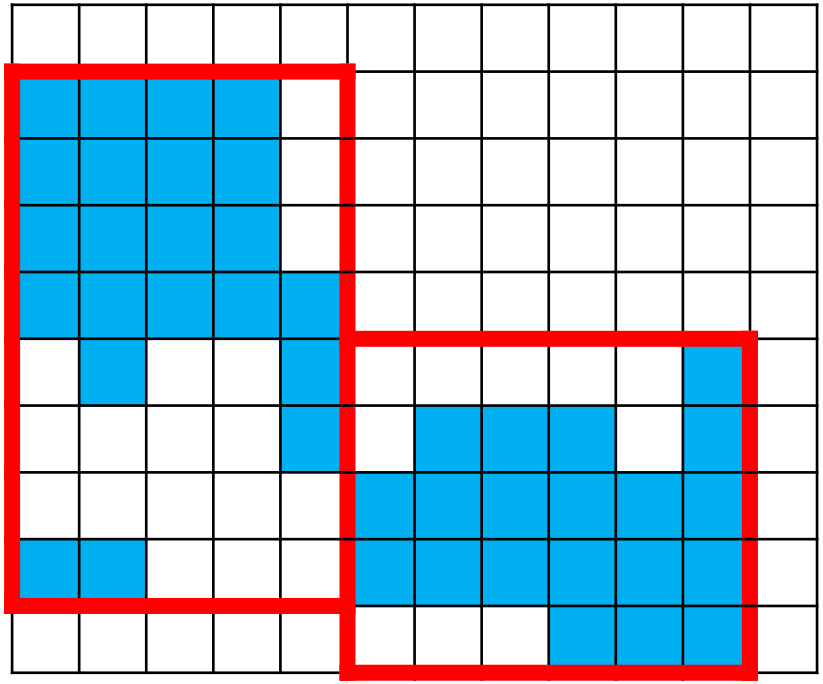
ヒストグラム

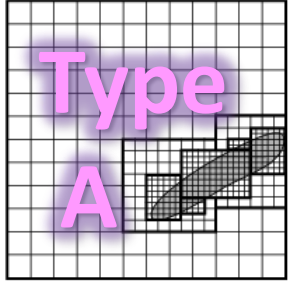
最小



ブロックの決定方法

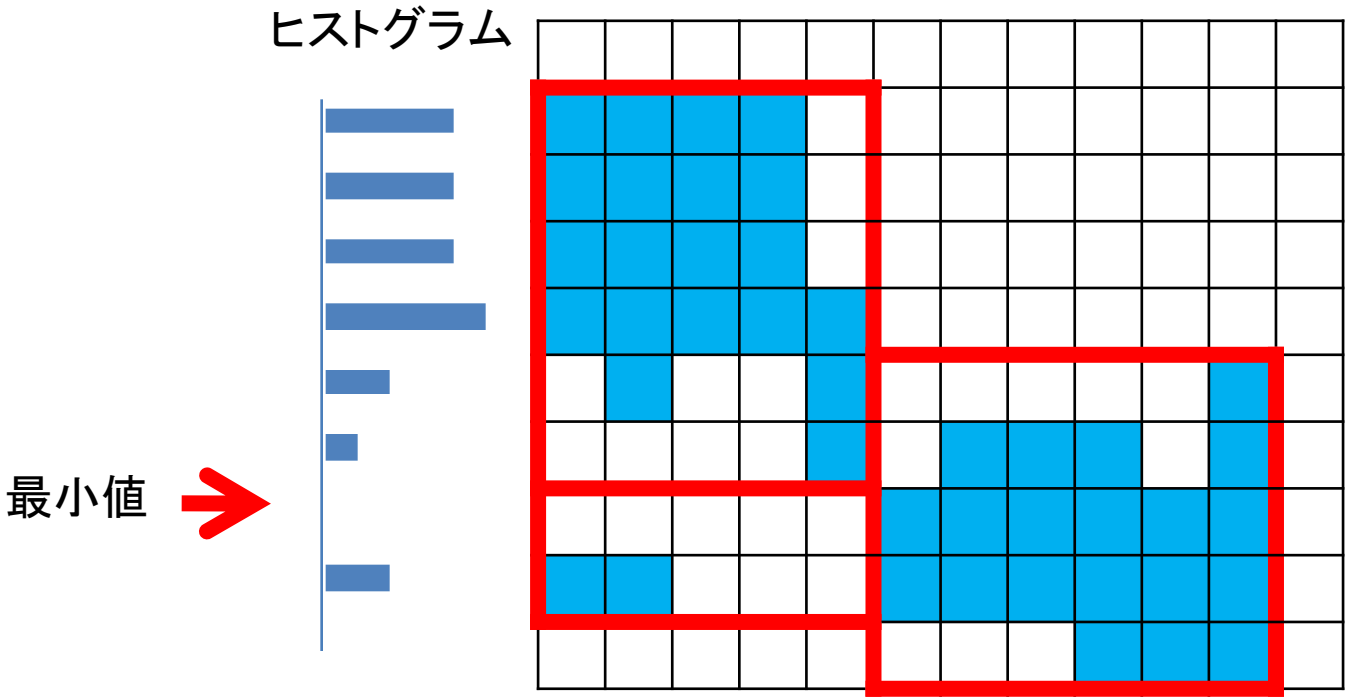
余分な部分を取り除く

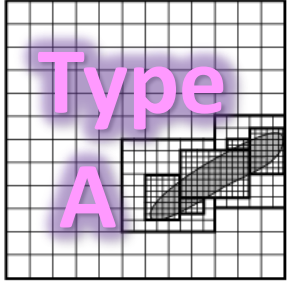




ブロックの決定方法

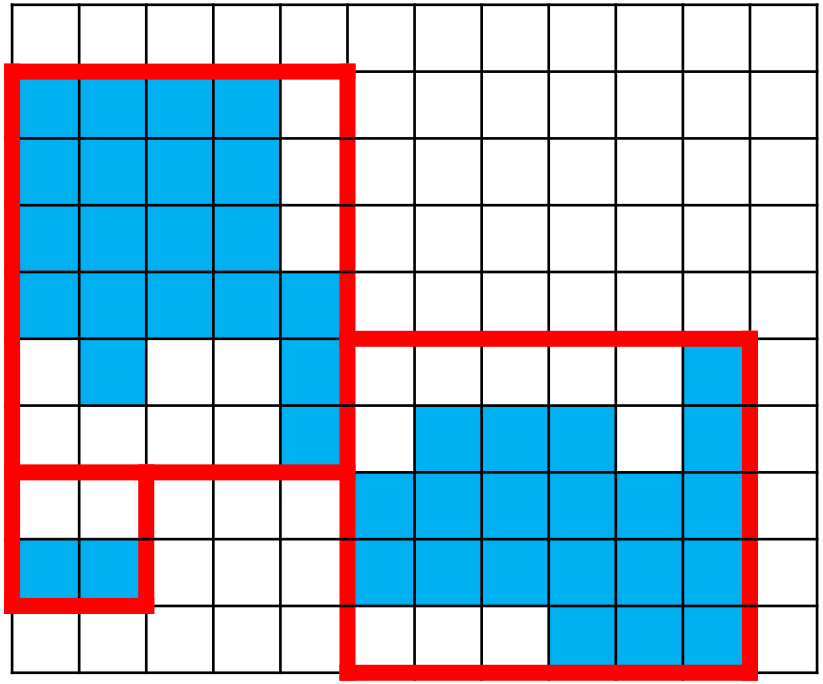
別の方向からセル数のヒストグラムを書き、分割する。





ブロックの決定方法

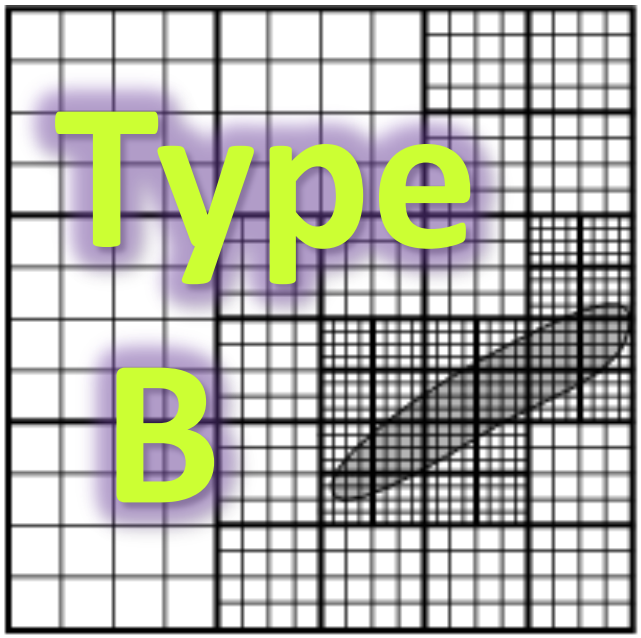
余白を取り除く。



こんなことを繰り返してゆく。

この操作を繰り返すほど、ブロックは小さくなる。
ブロックの最小サイズを決めておこう。

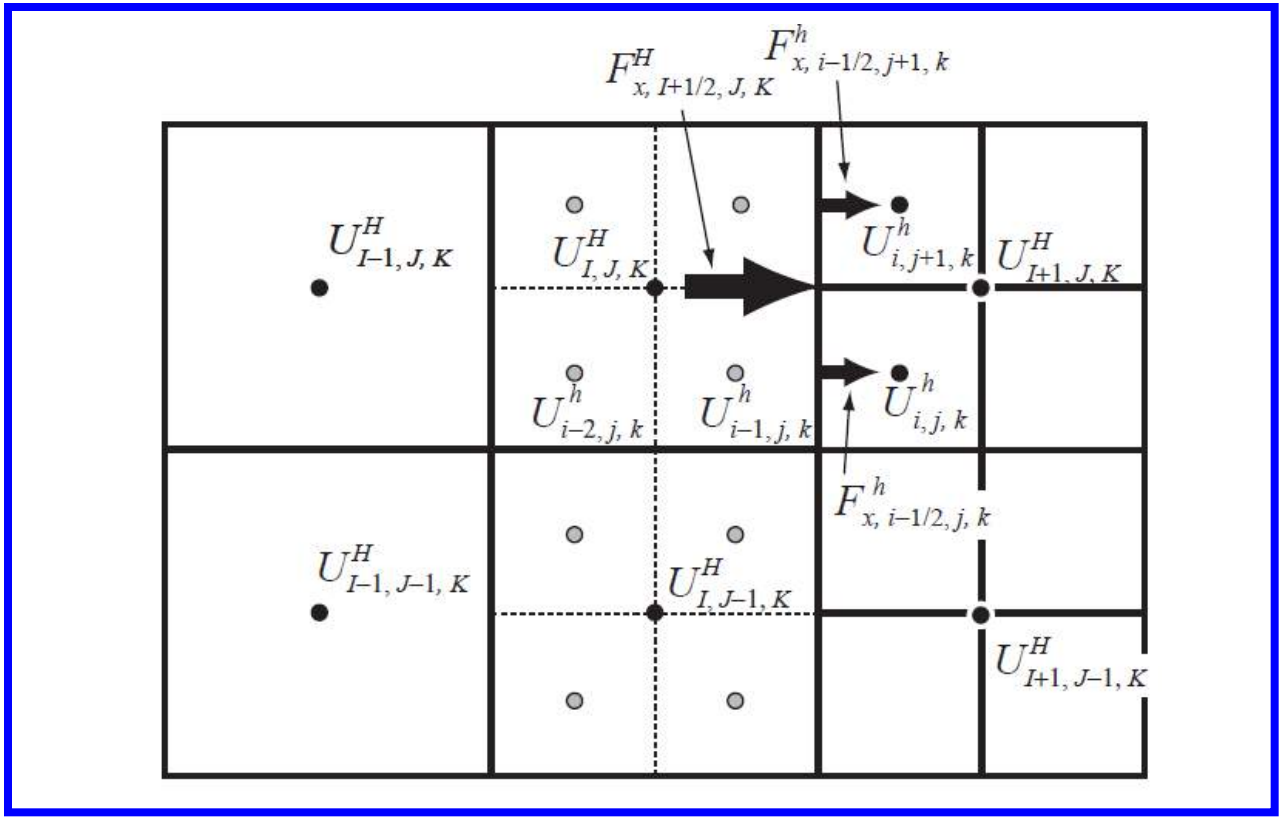
とっても面倒です



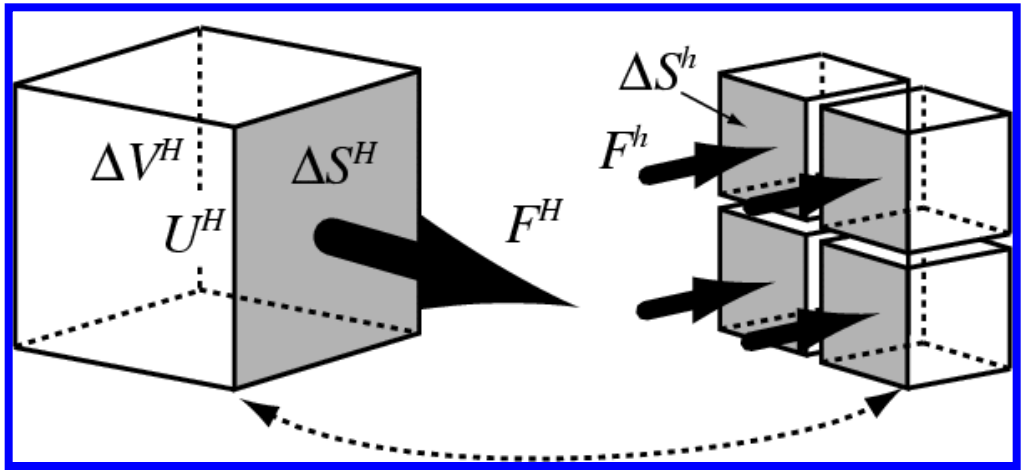
を採用しよう！

双曲型方程式の解法
細粗ブロックの境界

- 粗いブロックと接する**細かいブロック**
  - 粗いブロックを時間的に空間的に補間し、細かいブロックの境界条件にセット。
- 細かいブロックと接する**粗いブロック**
  - 細かいブロックの数値流速を使って、時間発展。



細粗境界での数値流束の保存



数値流束保存 $F^H \Delta S^H \Delta t^H = \sum_{\text{sub-cycle}} \sum_{\text{surface}} F^h \Delta S^h \Delta t^h$ が成立するように、

細かいブロックと隣接した粗いセルを補正する

$$U^{H,\text{new}} = U^H - \frac{1}{\Delta V^H} \left(\sum_{\text{sub-cycle}} \sum_{\text{surface}} F^h \Delta S^h \Delta t^h - F^H \Delta S^H \Delta t^H \right)$$

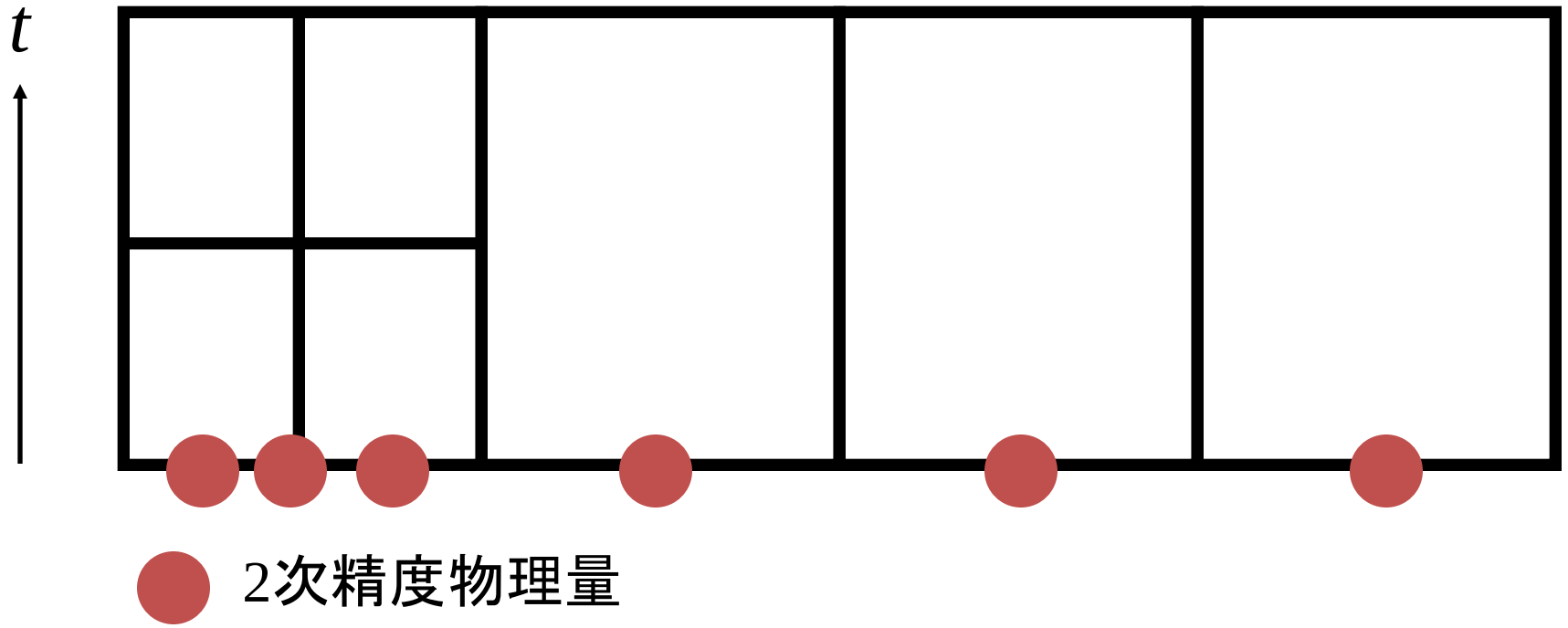
時間推進の方法

1. 時間を進めるべきレベルを探す。
   a. もっとも時間が進んでいないレベルのうち、もっとも粗いレベル。
2. 時間刻み幅dtを求める。
   a. レベル0はクーラン条件から。
   b. それ以外は親のdtの半分
3. 境界条件の設定
   a. 兄弟ブロックからコピー。
   b. 親ブロックから時間・空間的に内挿。
   c. ユーザの境界条件。
4. 通常の方法で時間推進する。
   a. たとえば、predictor-corrector法など。
5. 親レベルと同期した場合：
   a. 自分の解を親セルに代入する。
   b. 隣接する親セルの値を修正する（数値流束保存）。
6. 同期している全てのレベルを用いて細分化をする。

   $l_{\mathrm{sync}} \sim l_{\mathrm{max}} - 1$ を細分化して、
   $l_{\mathrm{sync}} + 1 \sim l_{\mathrm{max}}$ を生成する。
   l_{sync} = 同期している最粗レベル

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

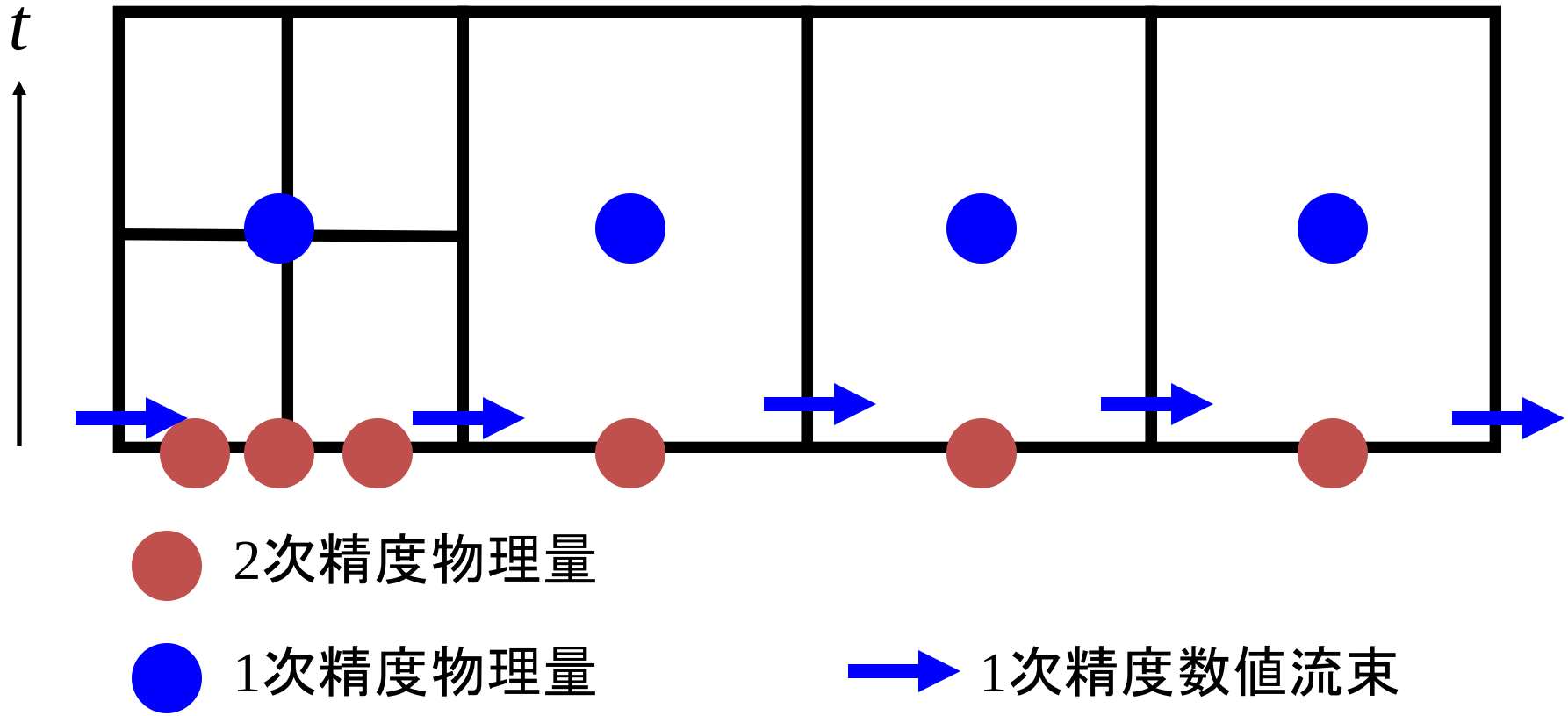
初期



2次精度物理量

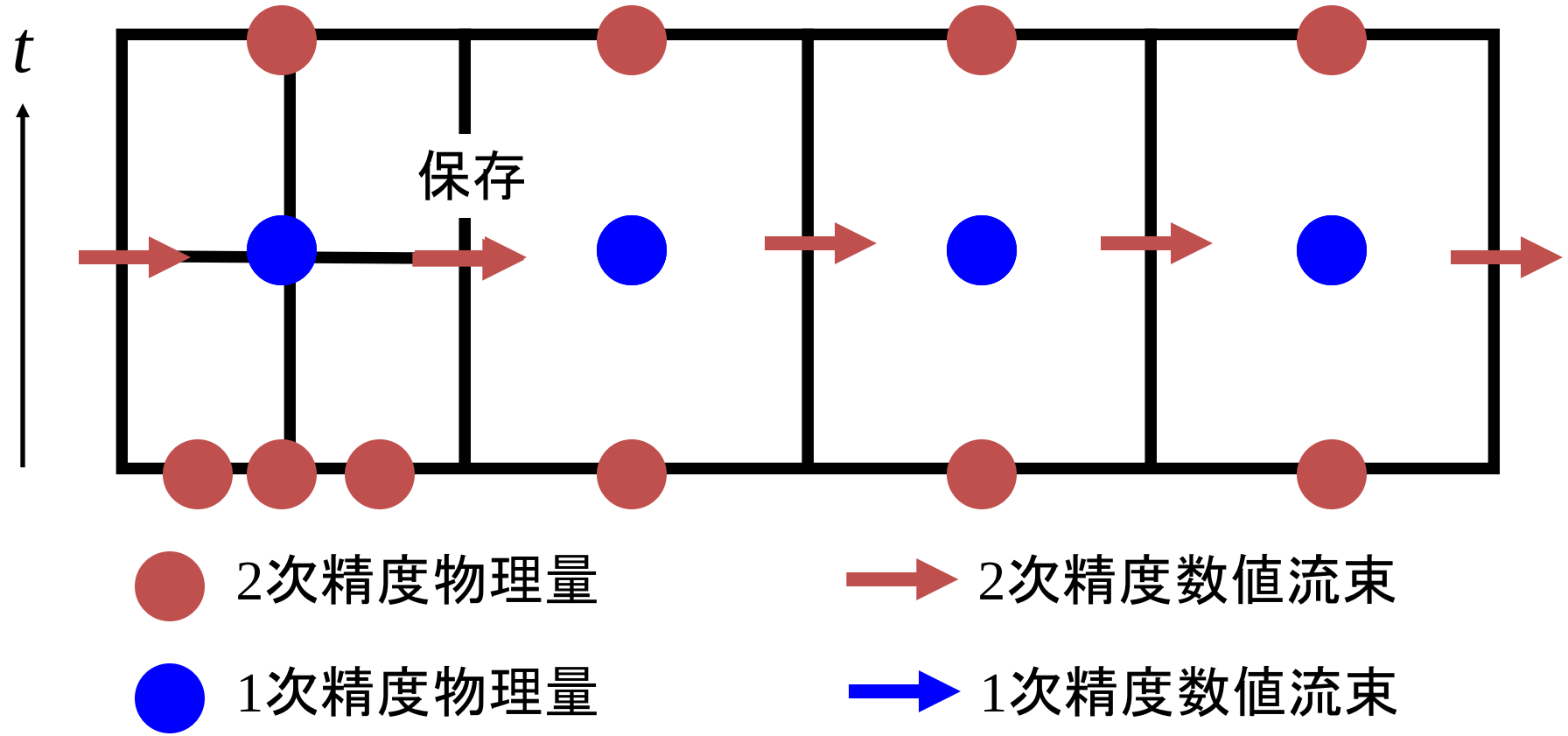
コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

親:予測ステップ



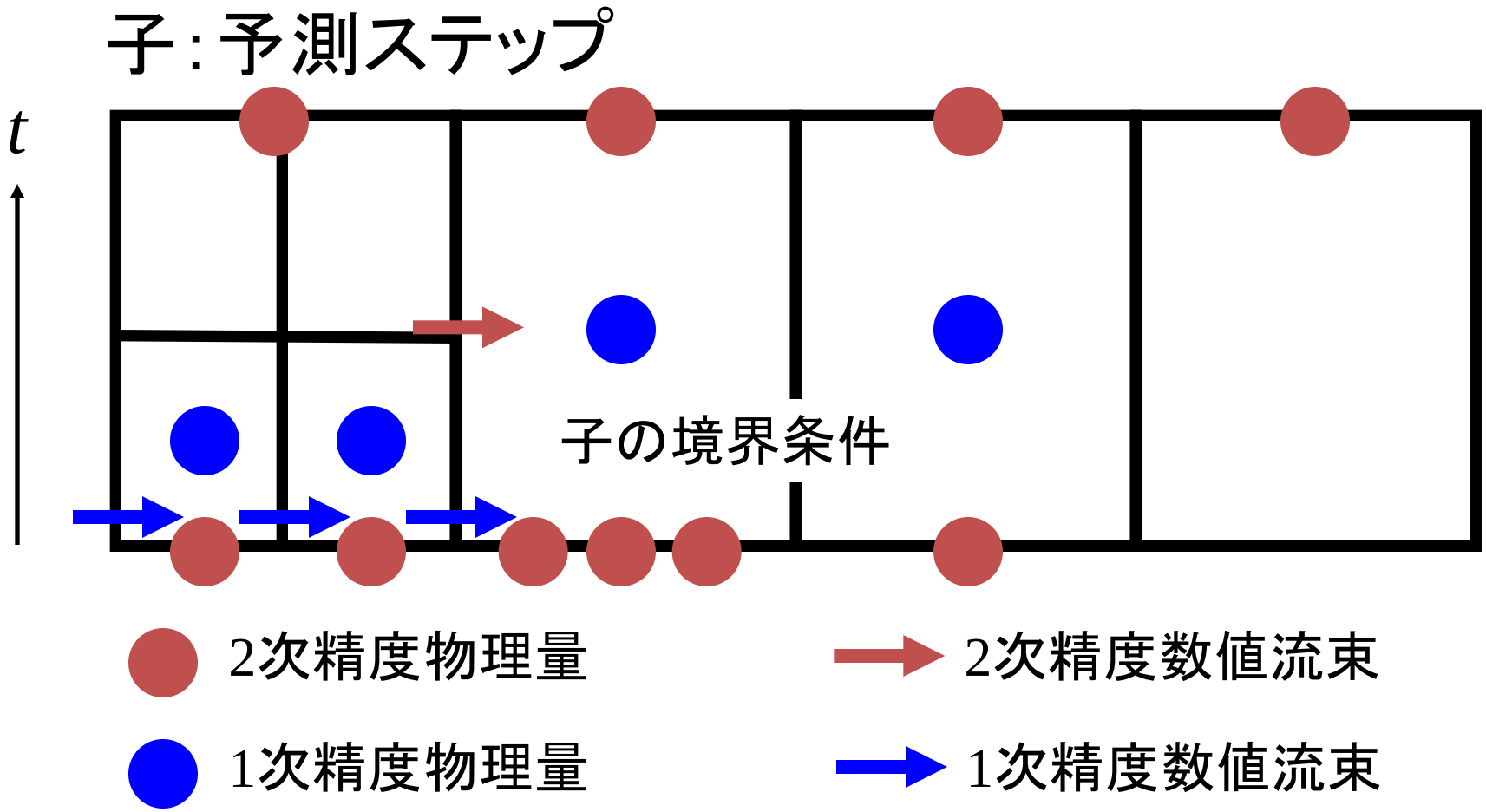
コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

親:修正ステップ



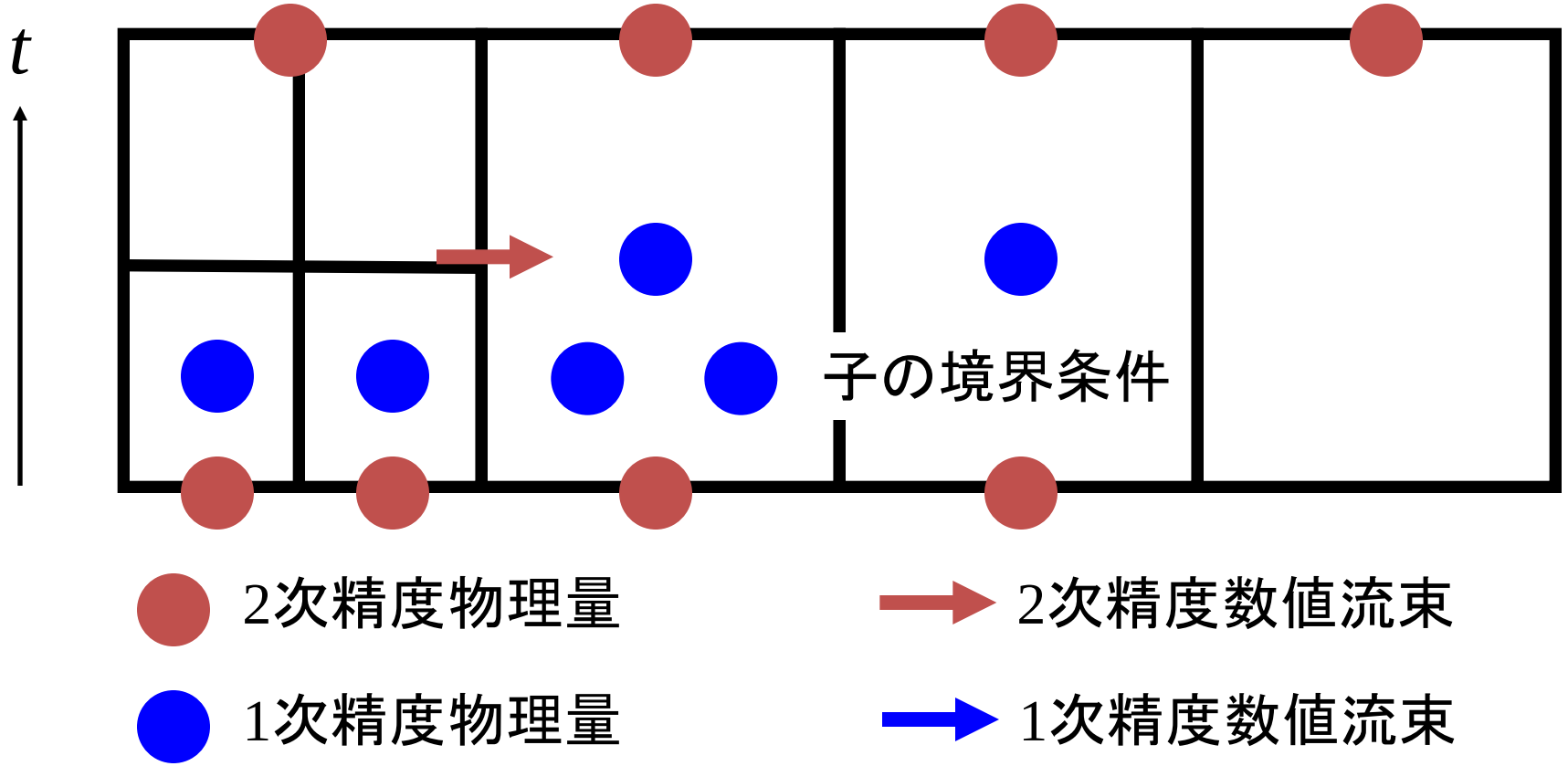
click

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ



コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

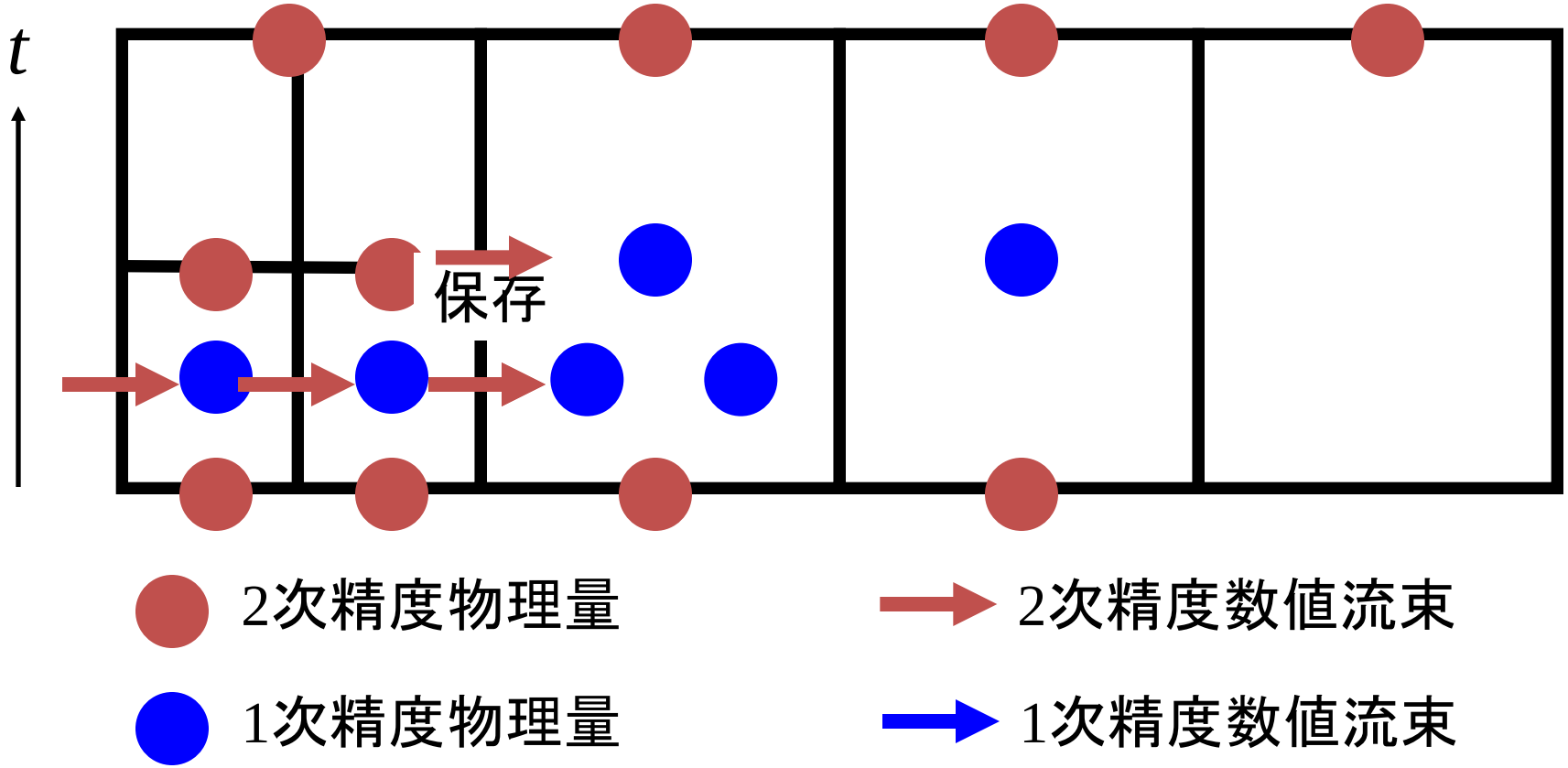
子:修正ステップ



click

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

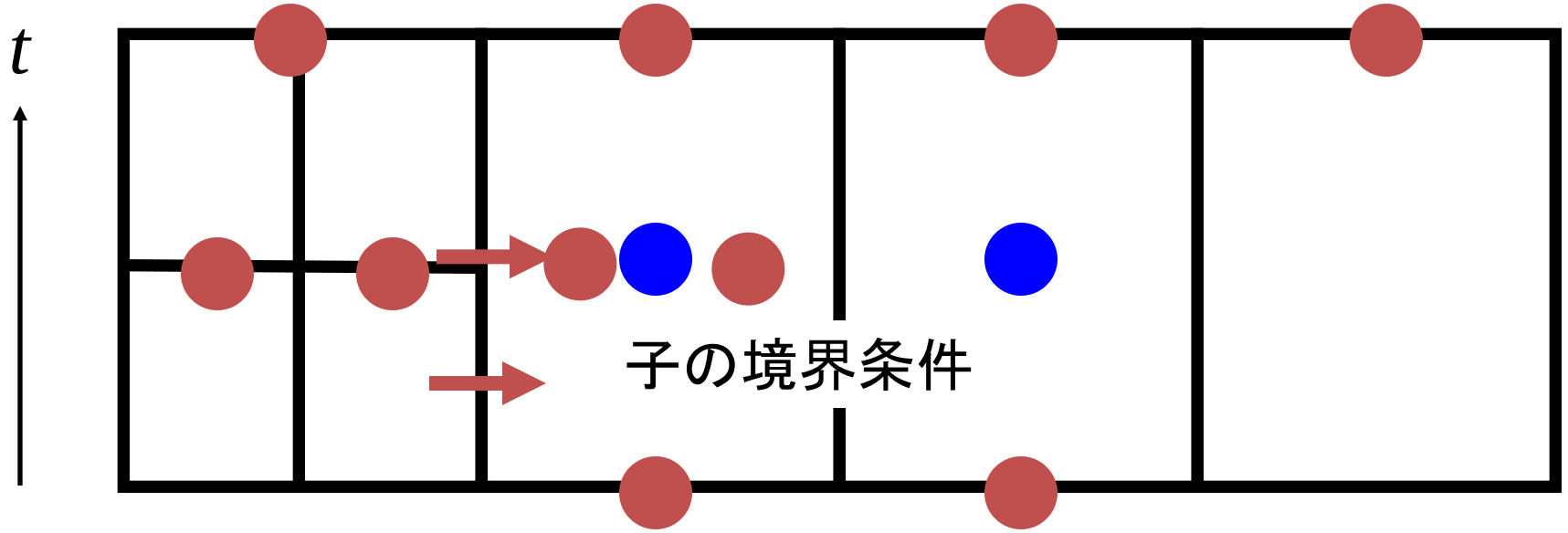
子:修正ステップ



click

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

子:予測ステップ(2回目)



2次精度物理量

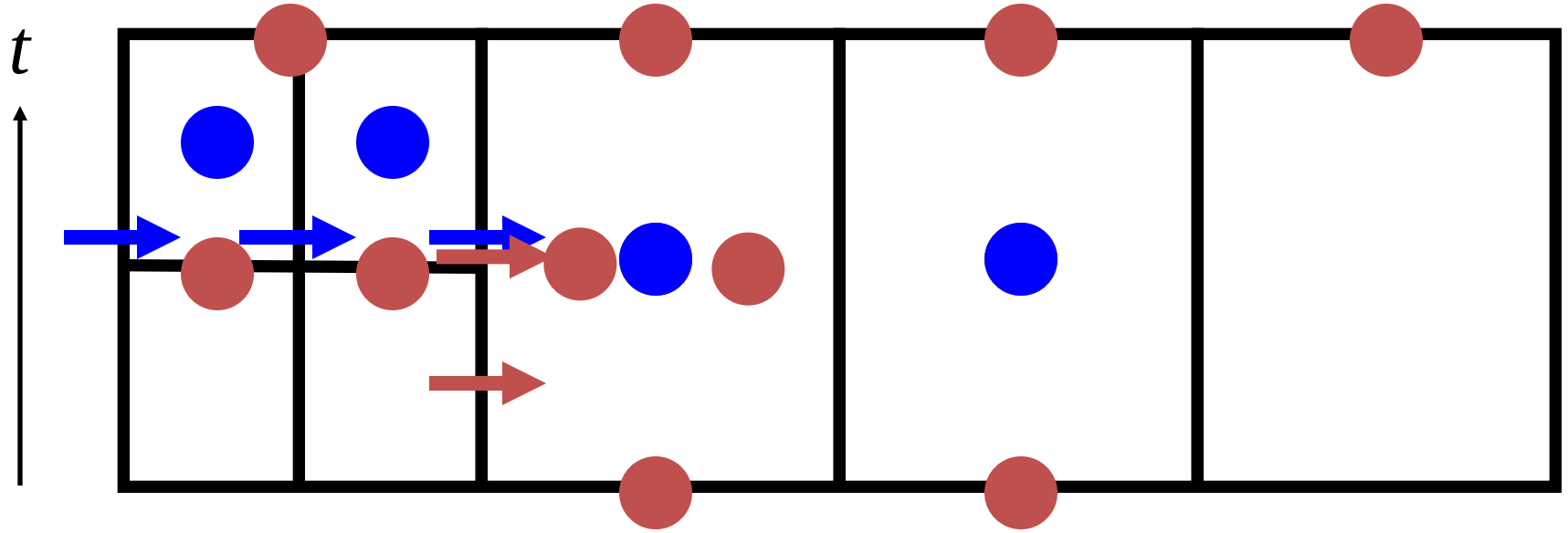
1次精度物理量

2次精度数値流束

1次精度数値流束

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

子:予測ステップ(2回目)



2次精度物理量

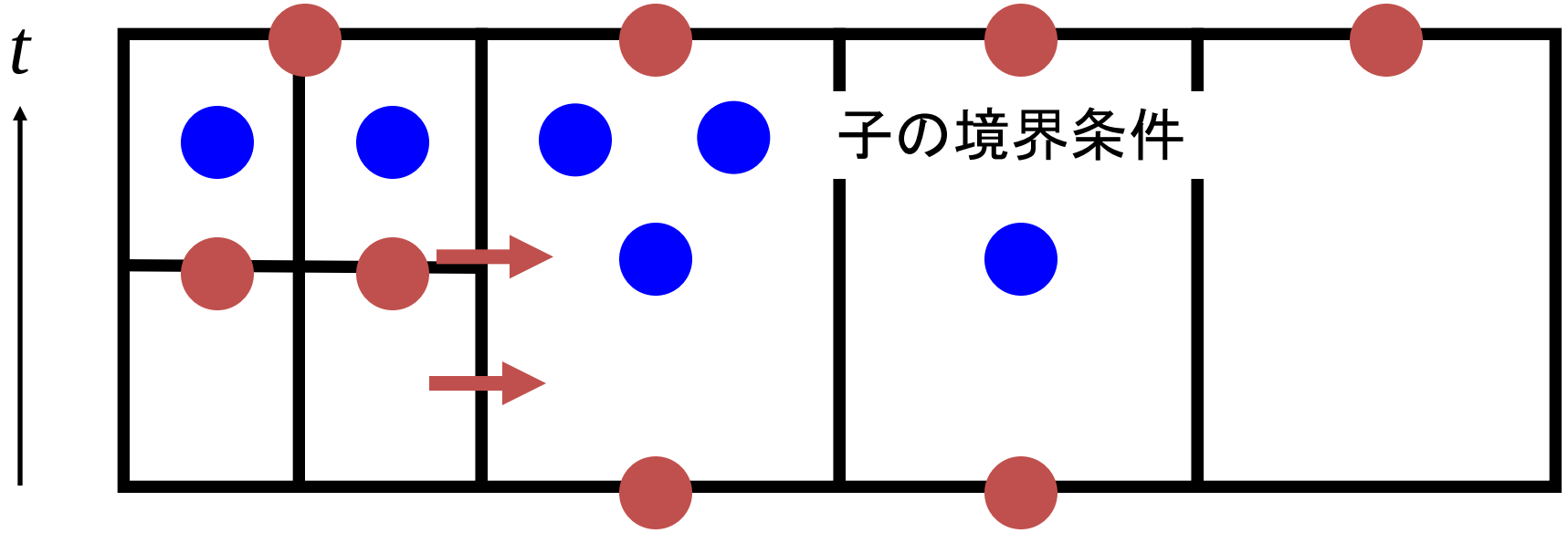
1次精度物理量

2次精度数値流束

1次精度数値流束

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

子:修正ステップ(2回目)



2次精度物理量

1次精度物理量

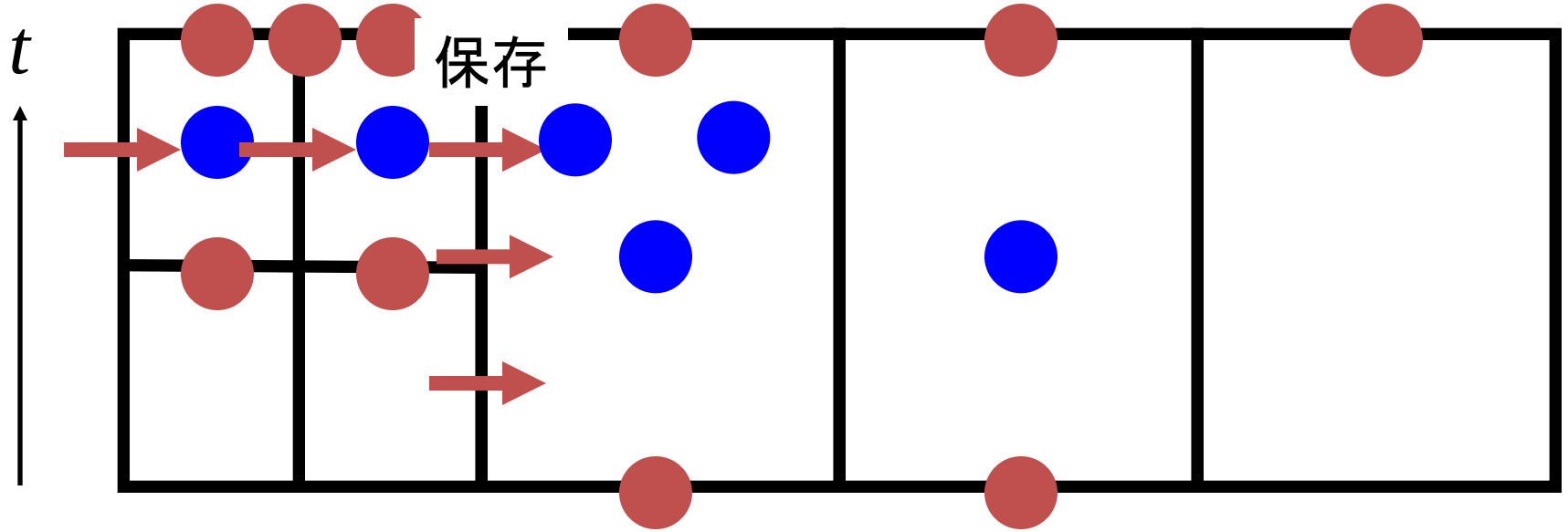
2次精度数値流束

1次精度数値流束

click

コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

子:修正ステップ(2回目)



2次精度物理量

1次精度物理量

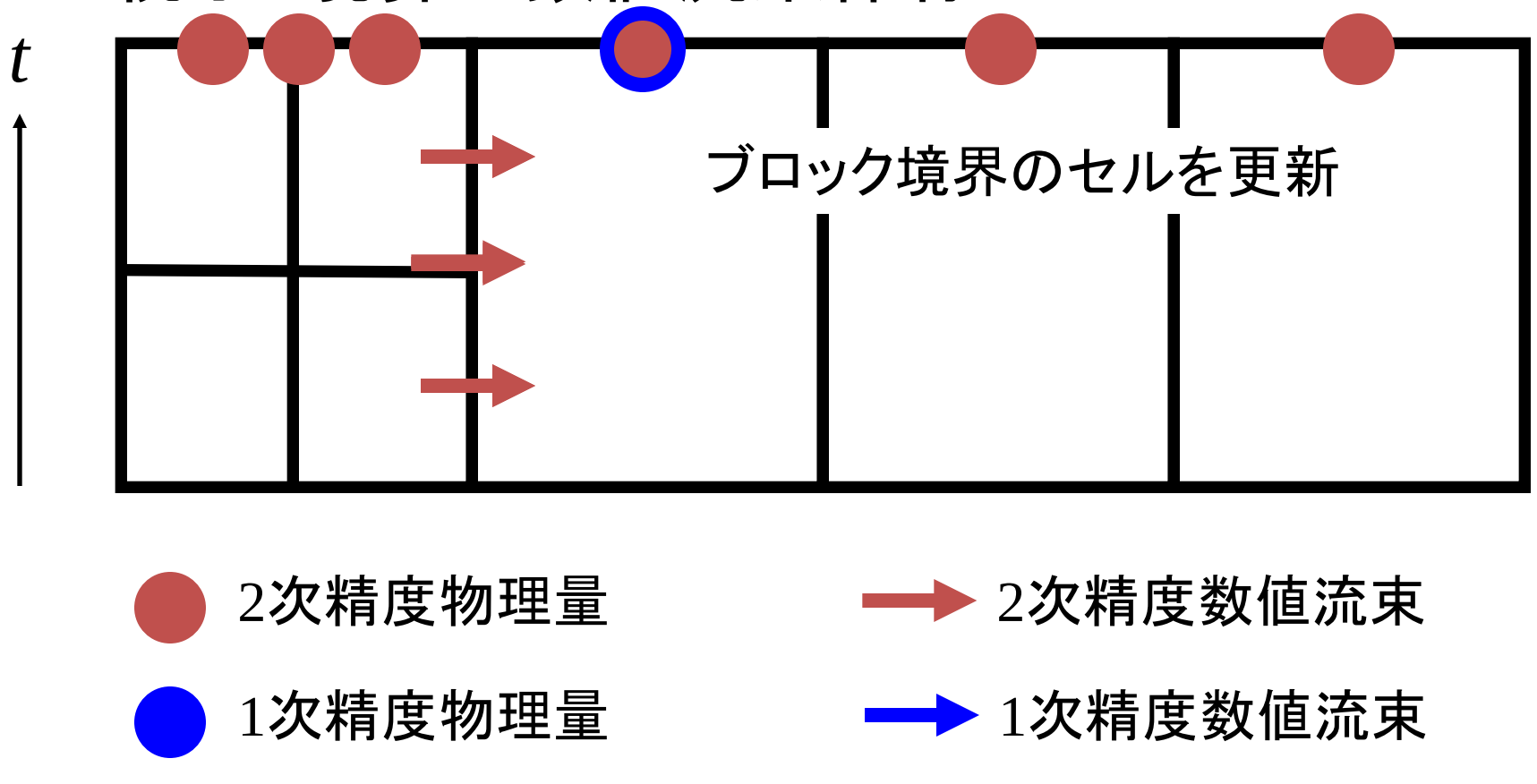
2次精度数値流束

1次精度数値流束

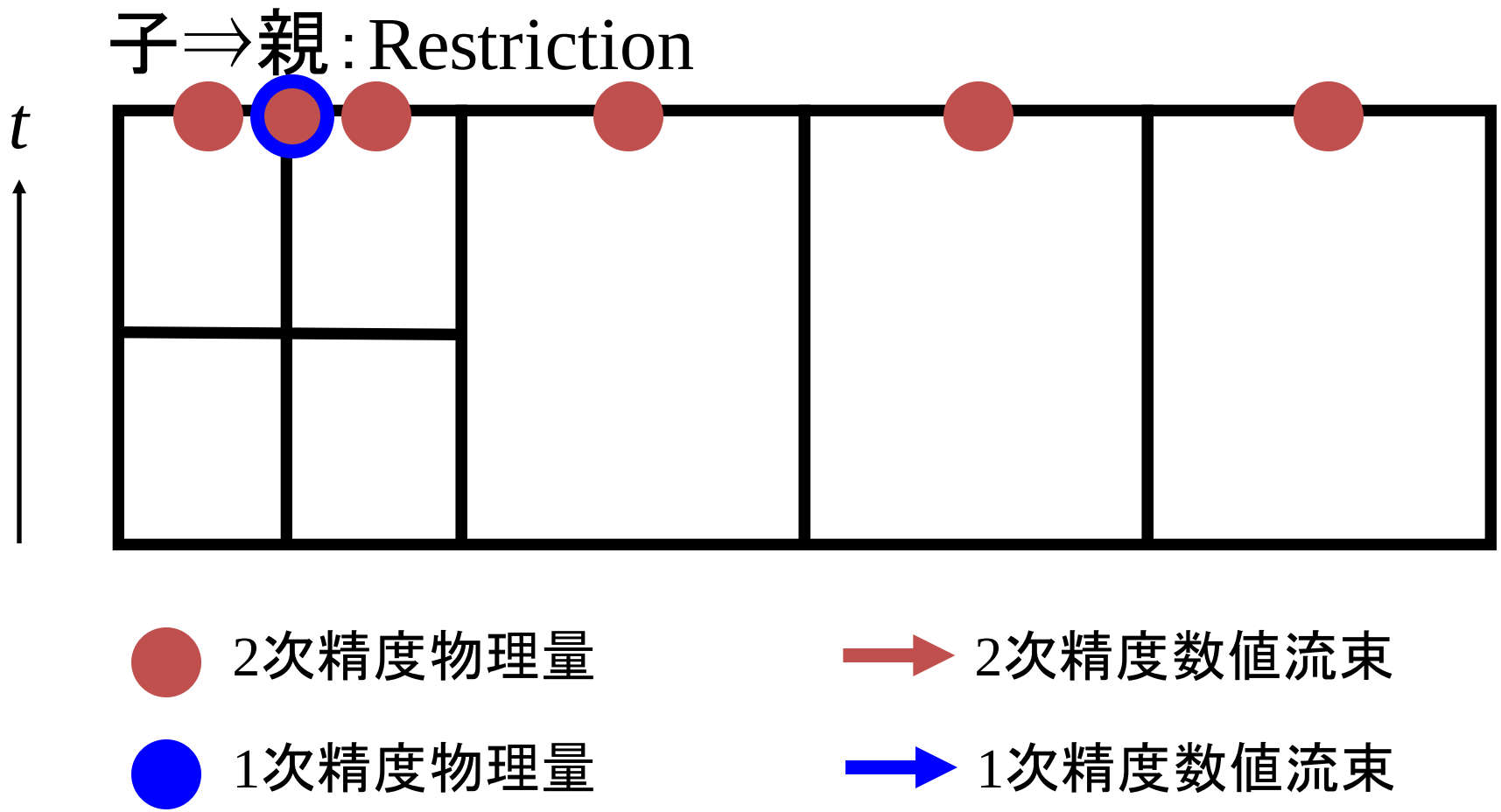
click

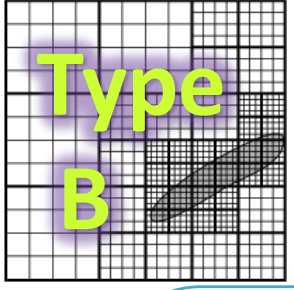
コードの詳細:　流体部分
数値流束補正・マルチタイムステップ

親子:境界で数値流束保存

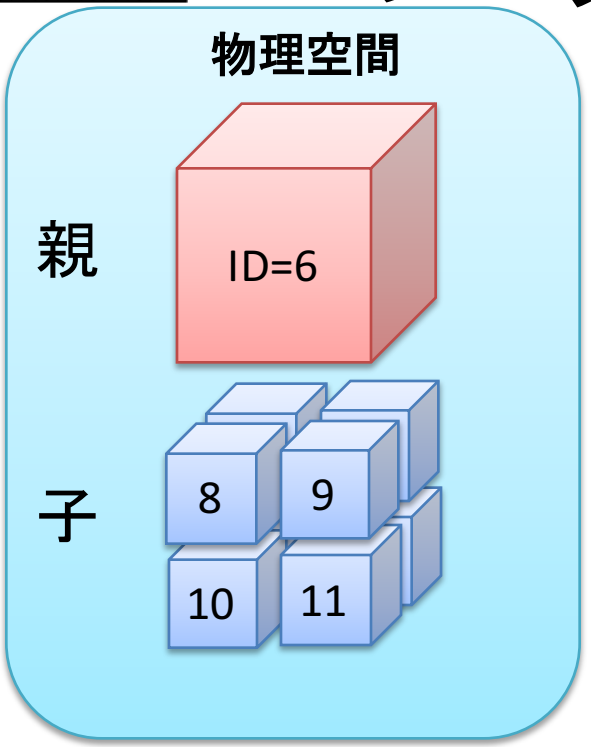


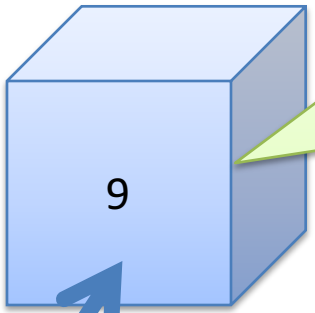
コードの詳細:　流体部分
数値流束補正・マルチタイムステップ





実装
ブロックの八分木構造





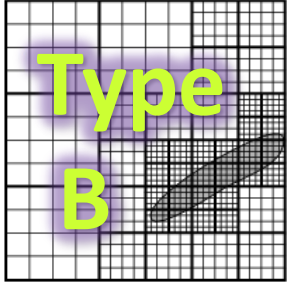
親のID×1
子供のID×8
隣のID×6
グリッドレベル
ブロックの位置(i,j,k)

セル Nx×Ny×Nz 個

並列化の方法

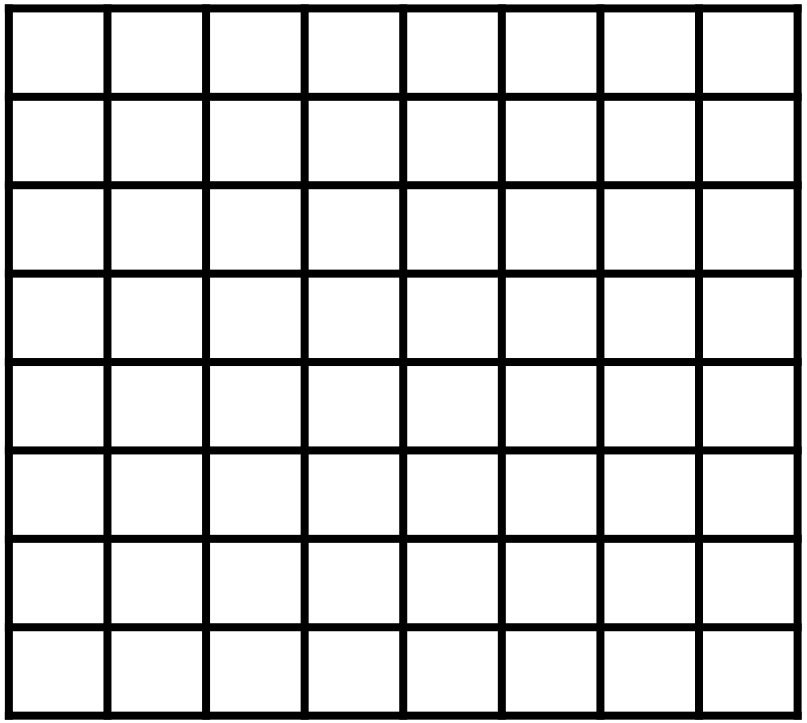
- Block structured gridの場合
  - ブロックをノードに割りつける。
  - 通信量を少なくするために
    - 近所のブロックを同じノードに割り付ける。
    - 近所のブロックを近所のノードに割りつける。

- それ以外の場合
  - よく知りません。

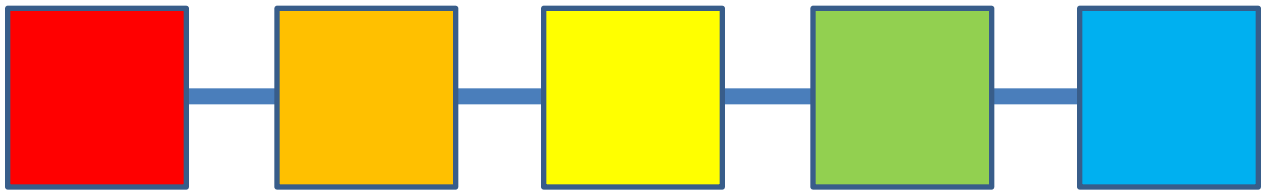


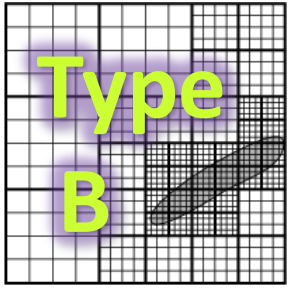
並列化
良くない方法

8^2ブロック
= 64 ブロック
=13 ブロック/ノード × 5ノード
-1ブロック



ノード



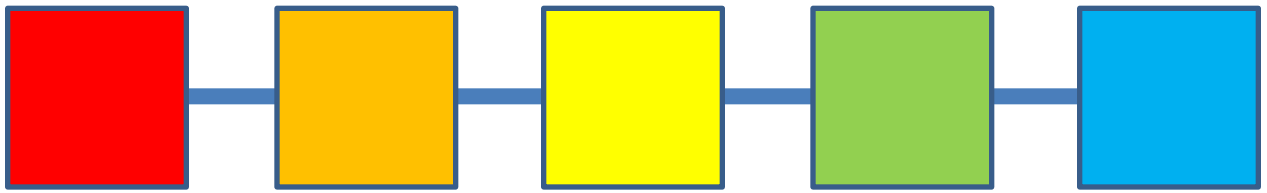


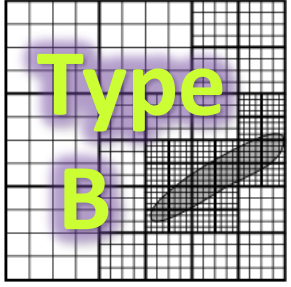
並列化
良くない方法

8^2ブロック
= 64 ブロック
=13 ブロック/ノード × 5ノード
-1ブロック

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

ノード





並列化
良くない方法

8^2ブロック
= 64 ブロック
=13 ブロック/ノード × 5ノード
-1ブロック

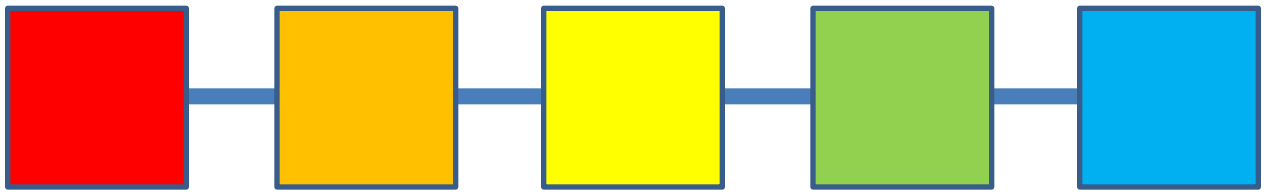
同じノードに割り付けられるブロックが細長く分布。

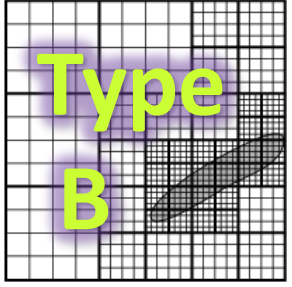
通信量は多い

ブロック数・ノード数が多くなると、不利になる。
泣き別れのブロックなど。

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

ノード

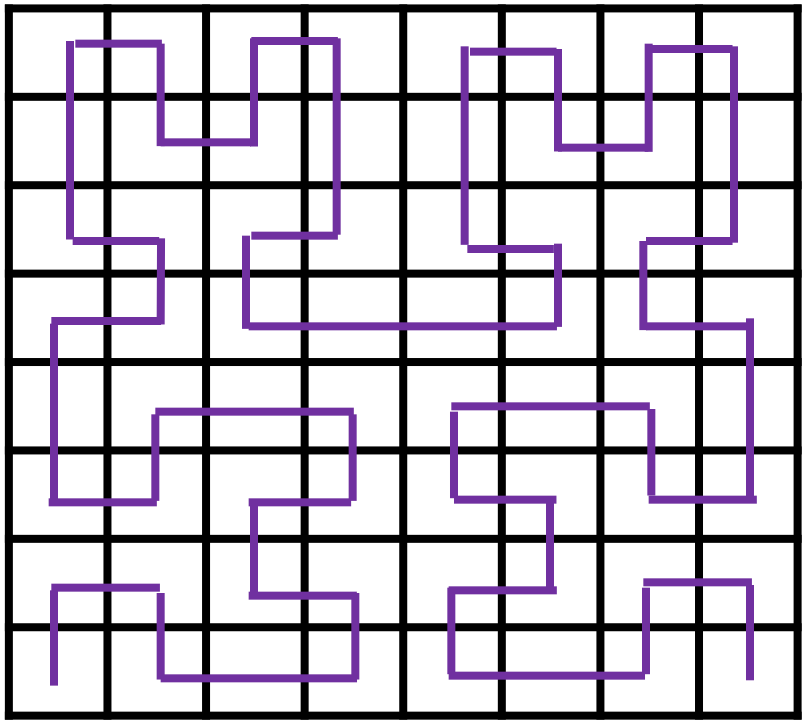




並列化
良い方法

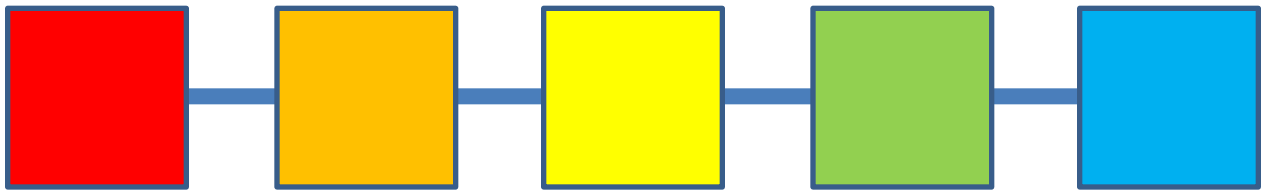
8^2ブロック
= 64 ブロック
=13 ブロック/ノード × 5ノード
-1ブロック

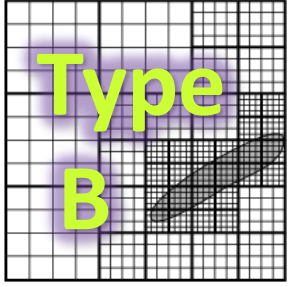
Hillbert 空間充填曲線
によるオーダリング



なるべく近くを通る
一筆書き

ノード





並列化
良い方法

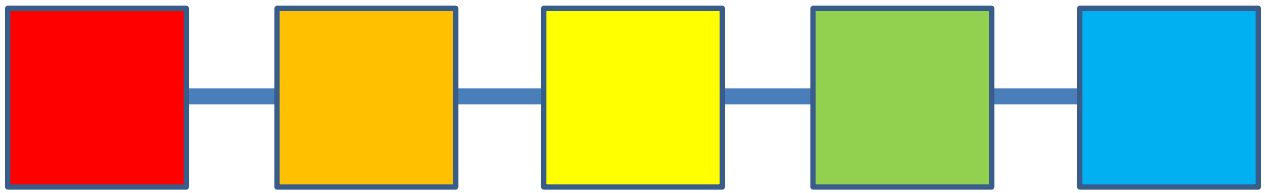
8^2ブロック
= 64 ブロック
=13 ブロック/ノード × 5ノード
-1ブロック

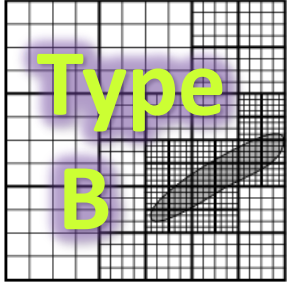
同じノードに割り付けられるブロックがコンパクトに分布。

通信量は少ない。

22	23	26	27	38	39	42	43
21	24	25	28	37	40	41	44
20	19	30	29	36	35	46	45
17	18	31	32	33	34	47	48
16	13	12	11	54	53	52	49
15	14	9	10	55	56	51	50
2	3	8	7	58	57	62	63
1	4	5	6	59	60	61	64

ノード



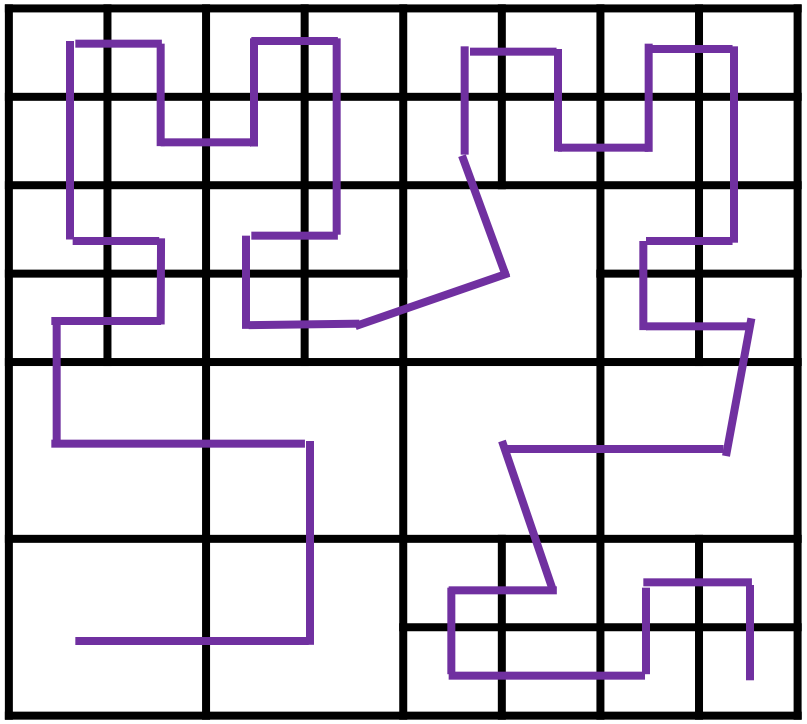


= 43 ブロック
= 8 ブロック/ノード × 5ノード
+3ブロック

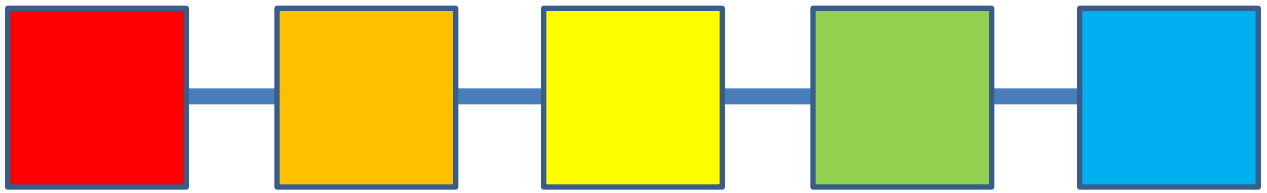
並列化
グリッドレベル横断

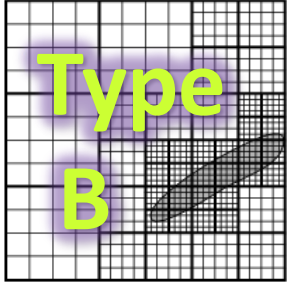
Hillbert 空間充填曲線
によるオーダリング

なるべく近くを通る
一筆書き



ノード





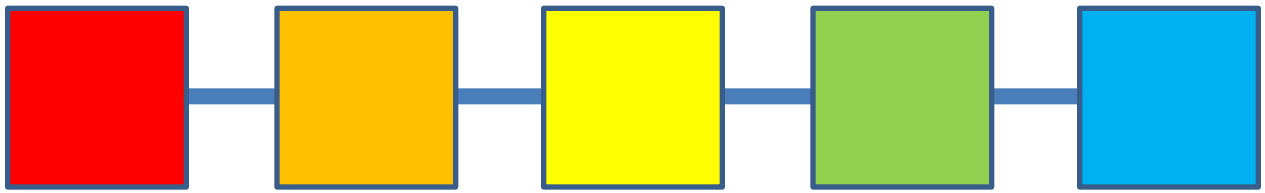
= 43 ブロック
= 8 ブロック/ノード × 5ノード
+3ブロック

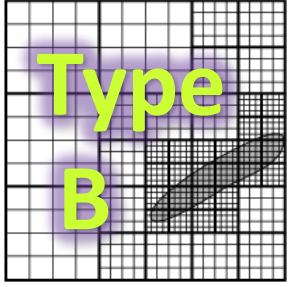
並列化
グリッドレベル横断

Hillbert 空間充填曲線
によるオーダリング

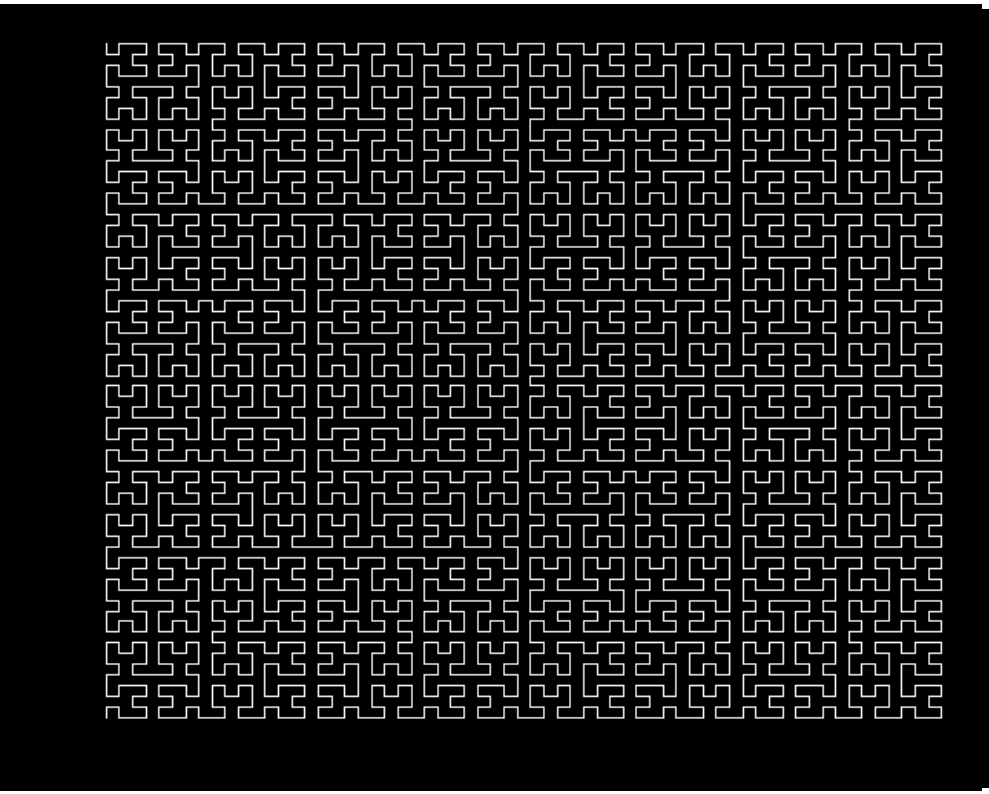
なるべく近くを通る
一筆書き

ノード

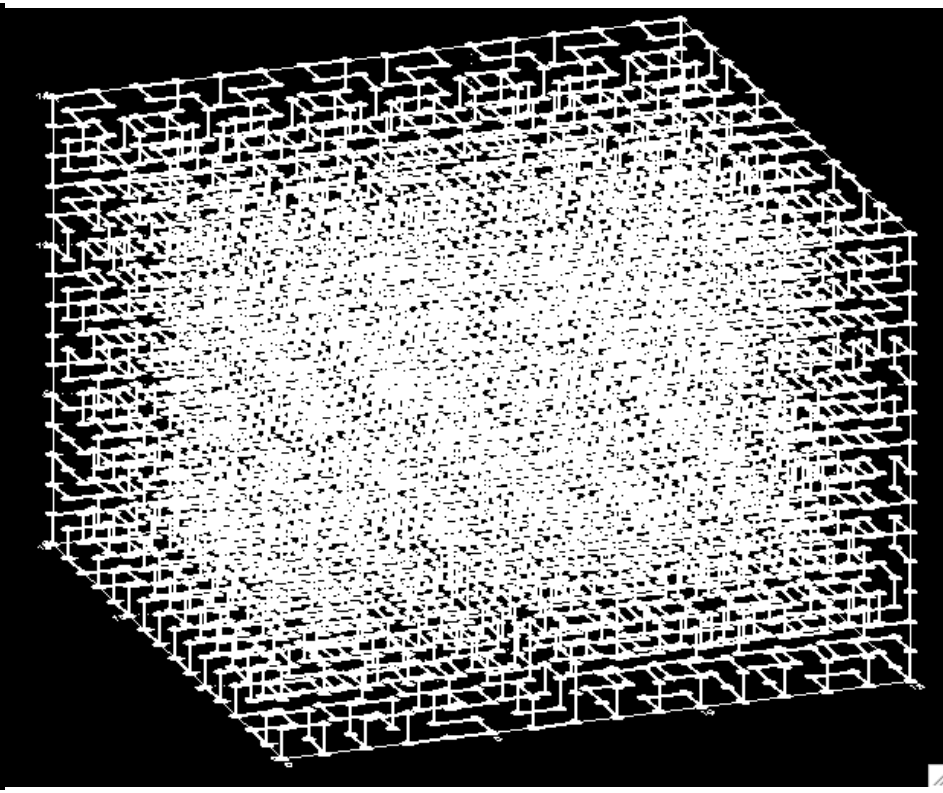




3次元でも同様



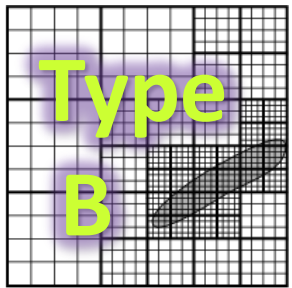
2次元のHillbert 空間充填曲線

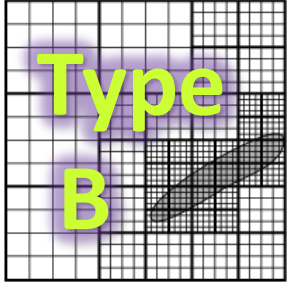


3次元のHillbert 空間充填曲線

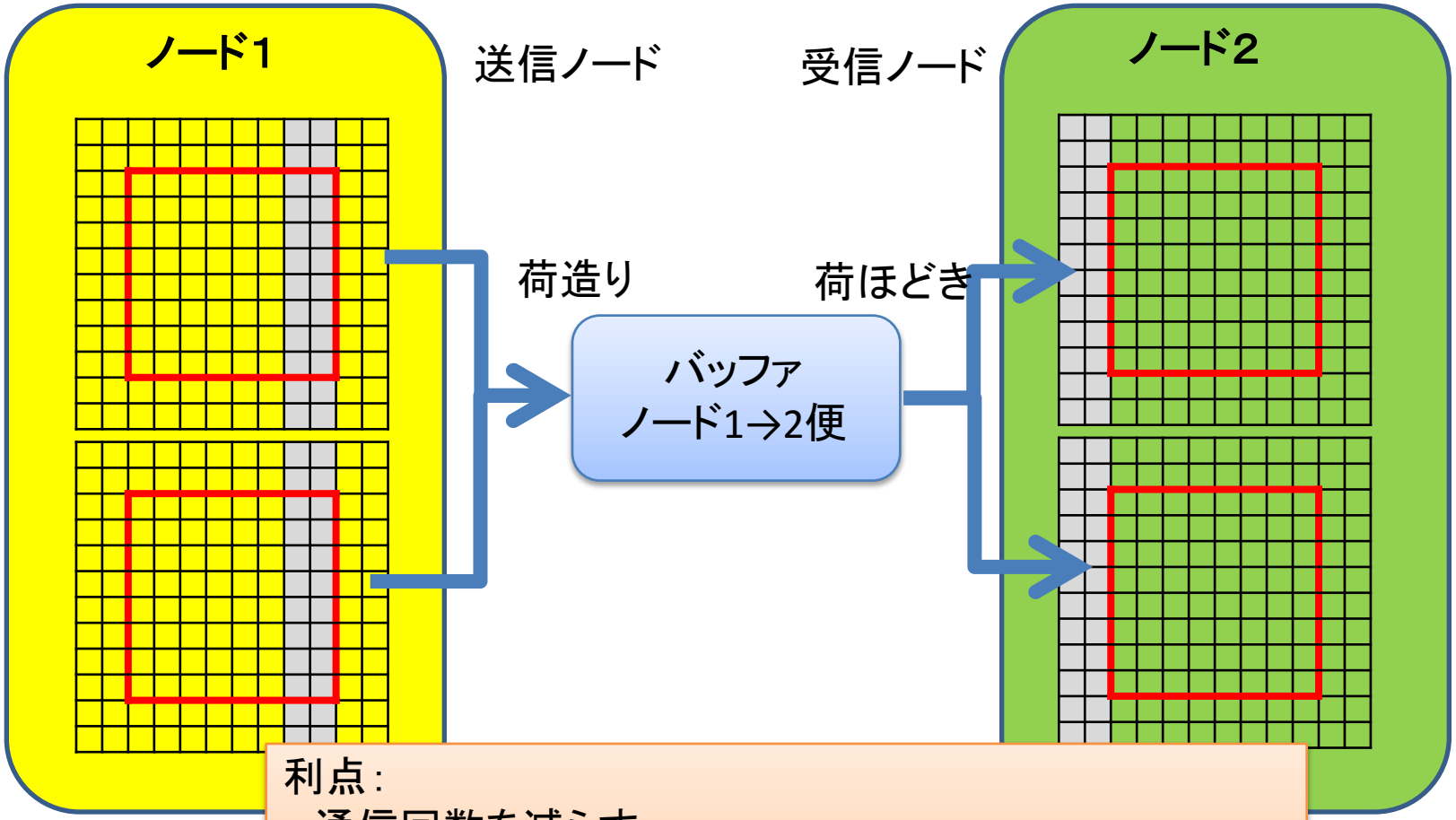
ブロックのオーダリングまとめ

- Hillbert空間充填曲線によるオーダリングは、面倒そうですが、意外に簡単な工夫です。
- グリッドレベルごとに行う。
  - Adaptive time step
- グリッドレベルを横断して行う。
  - Synchronous time step





袖の転送
ノード間をまとめて転送する



利点：
- 通信回数を減らす。
  性能は通信量ではなく通信回数で決まる。
- MPI通信タグの枯渇を回避する。
  ブロックが個別に通信するとタグ数が上限を超える。

おしまい