MACHINE LEARNING, AN EMULATOR, AND THE FUTURE OF COSMOLOGICAL SIMULATIONS

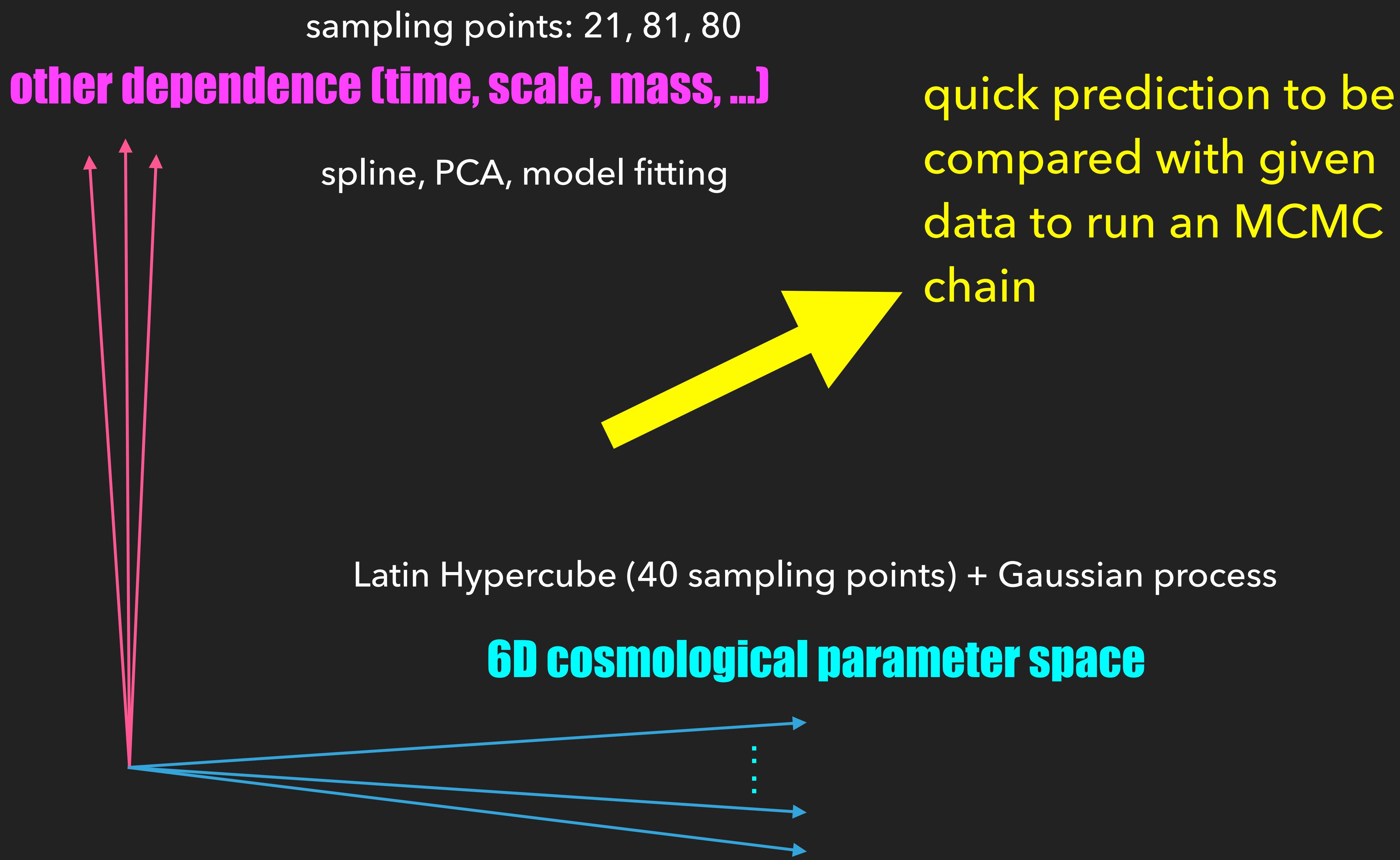
BY U-TOKYO GROUP (YOSHIDA, NISHIMICHI, OOGI)

- Cosmology with Subaru HSC survey
- A large number of cosmological N-body simulations
- Parameter space exploration and the Gaussian process
- Direct integration of collisionless Boltzmann equation
- Future prospects

サブCの活動

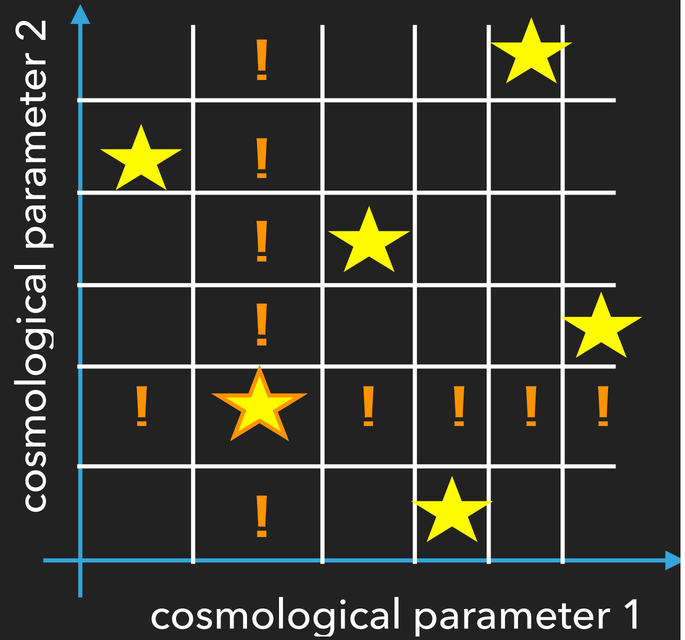
- ~３か月おきに全体会合(2016.3, 6, 9)
各機関報告や、分野連携(BH降着+semi-analytic法など)
を推進. 次回会合を国内WSにすることを検討中
(理論懇、CfCA UM, ポスト京9全体などを避ける)
- ブラソフ、N体、プラズマ、BH各班順調にコード開発
ただし京コンピューター前期は1M nh ほど未使用だった

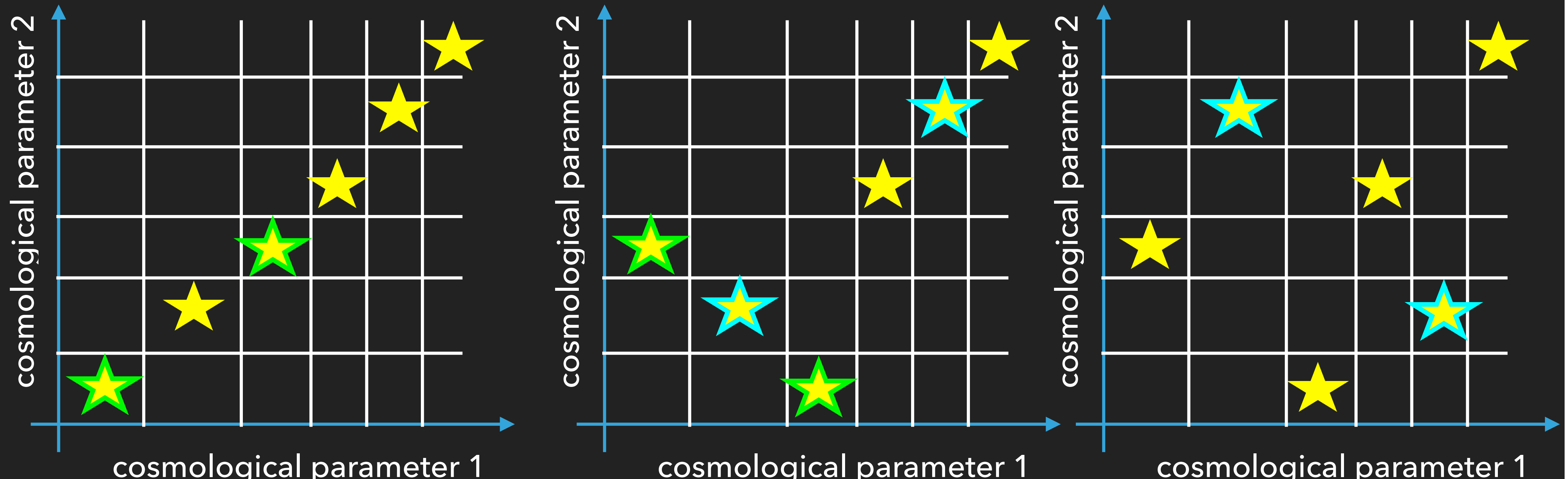
THE SIMULATION DESIGN IN HIGH DIMENSION SPACE



EFFICIENT SAMPLING IN MULTI DIMENSIONAL SPACE: LATIN HYPERCUBE

- Each sample is the only one in each axis-aligned hyperplane containing it
  - One can find many realizations of such design (ex. diagonal design)
  - Impose additional condition such as "the sum of the distances to the nearest design point is maximal" (maximin distance)



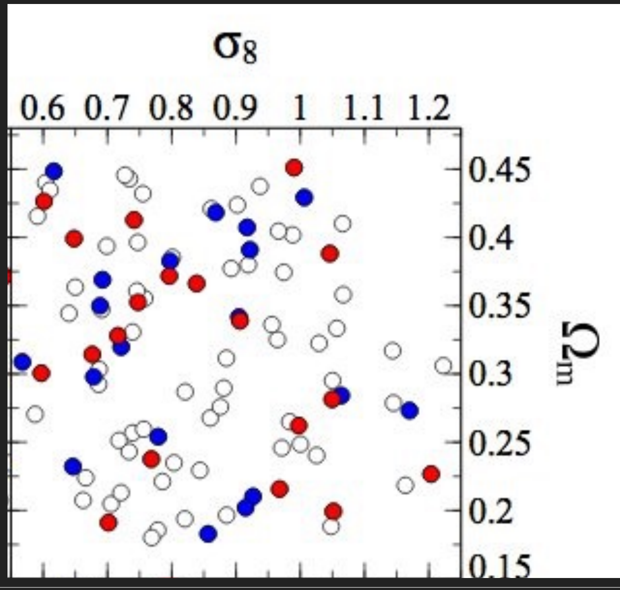


EFFICIENT SAMPLING IN MULTI DIMENSIONAL SPACE: LATIN HYPERCUBE

fiducial model

- PLANCK15 flat ΛCDM
- 24 realizations done
- assess statistical error
- check the accuracy of the emulator

$\omega_b = \Omega_b h^2$: ±5%
$\omega_c = \Omega_c h^2$: ±10%
Ω_Λ: ±20%
$\ln(10^{10} A_s)$: ±20%
n_s: ±5%
w: ±20%



84 sims are already available in total

varied cosmology

- “sliced” LH design (Ba, Brenneman & Myers ’15)
- generate 100 samples eventually
- maxi-min distance LH design for every 20 models (e.g., red/blue points)

- 2 types of sims
  - keep the initial random number seed (20 done)
  - independent seeds (40 done)
    - red: emulator
    - blue: validation

SIMULATION SPEC

- ✓ N of particles: 2048^3
- ✓ box size: $1h^{-1}$Gpc

  resolve a $10^{12}\ h^{-1}M_{solar}$ halo with ~100 particles
- ✓ 2nd-order Lagrangian PT initial condition @ z_{in}=59

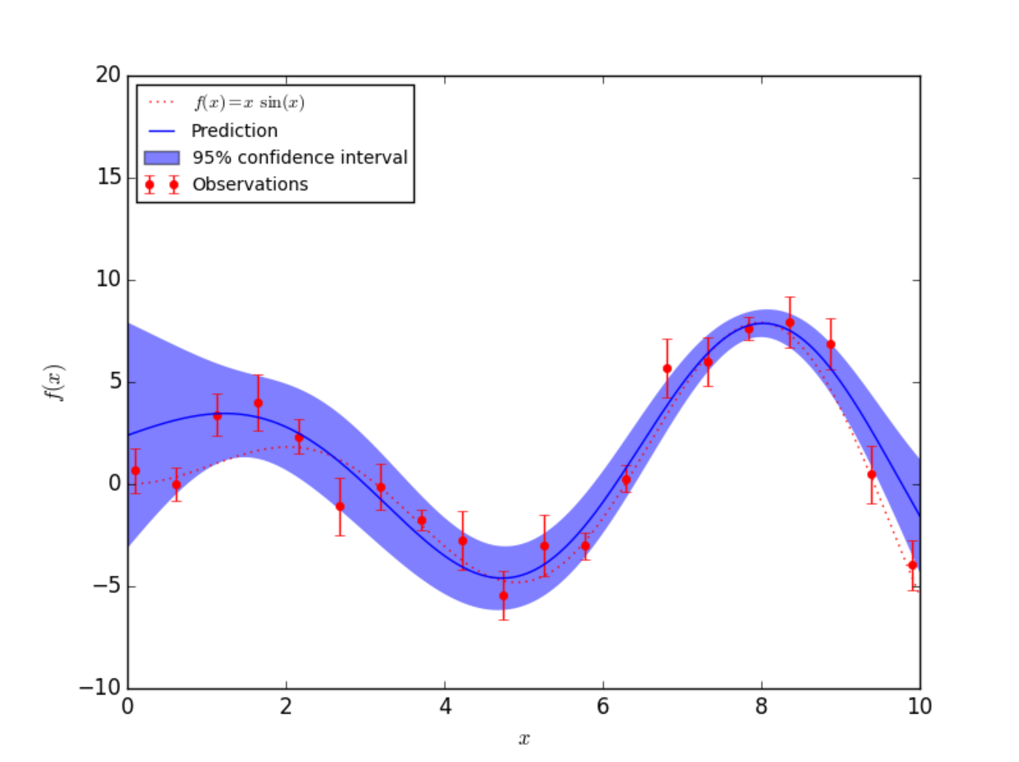
  (vary slightly for different cosmologies to keep the RMS displacement about 25% of the inter-particle separation)
- ✓ Tree-PM force by L-Gadget2 (w/ 4096^3 PM mesh)
- ✓ 21 outputs in $0 \leqq z \leqq 1.5$

  (equispaced in linear growth factor)
- ✓ Data compression (256GB -> 48GB par snapshot)
  - ✓ positions -> displacement (16 bits par dimension; accuracy ~$1h^{-1}$kpc)
  - ✓ velocity: discard after halo identification
  - ✓ ID: rearrange the order of particles by ID and then discard
  - ✓ already consuming ~200TB in half a year (~observational data)

GAUSSIAN PROCESS

- A kind of machine learning that interpolates in function space
  - non-parametric Bayesian inference
  - quick in high dimension input space
- Basic quantities (c.f., normal distribution)
  - mean function (cf. mean)
  - covariance function (cf. variance)
- mean can be anything, set zero usually
- covariance function is characterized by a simple function with several hyper parameters

ex. $C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{I}\frac{(x_i - x_i')^2}{r_i^2}\right] + \theta_2.$



✓ infer hyper parameters θ from training data (x_i, t_i)

✓ Given any input x_{N+1}, infer t_{N+1} from θ and (x_i, t_i)

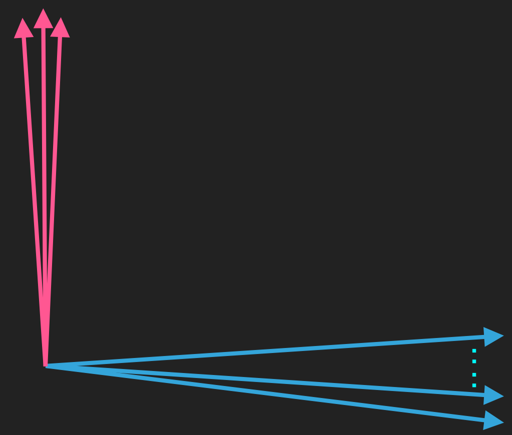
$$P(t_{N+1} \mid \mathbf{t}_N) \propto \exp\left[-\frac{1}{2}\left[\mathbf{t}_N \; t_{N+1}\right] \mathbf{C}_{N+1}^{-1} \begin{bmatrix} \mathbf{t}_N \\ t_{N+1} \end{bmatrix}\right]$$

answer:

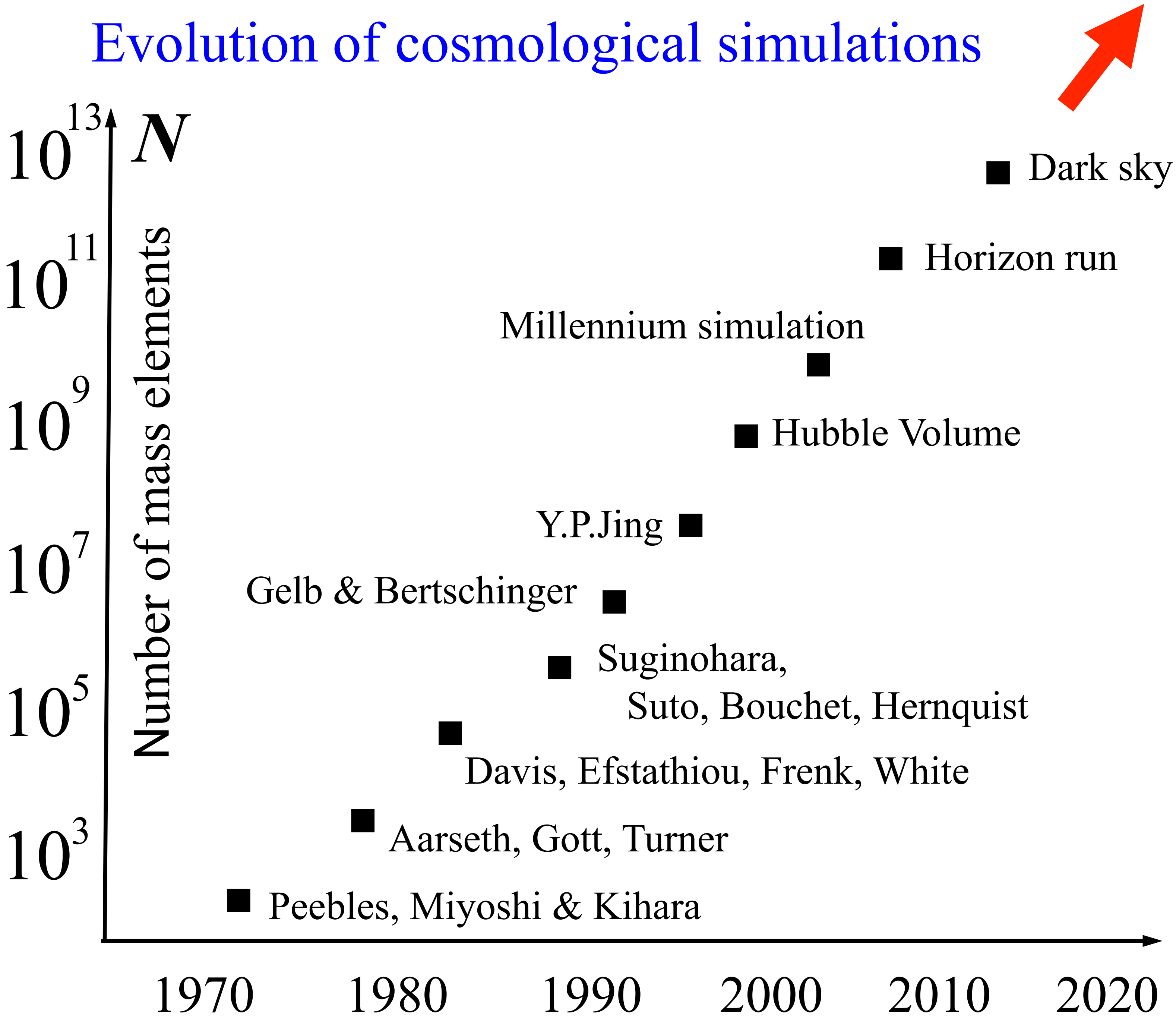
$$\begin{aligned} \hat{t}_{N+1} &= \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}_N \\ \sigma_{\hat{t}_{N+1}}^2 &= \kappa - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}. \end{aligned}$$

SUMMARY + FUTURE

- Modeling the halo mass function and galaxy-galaxy lensing signal
  - Latin hypercube design + fitting/GP/spline
  - handy emulator in python almost ready
  - accuracy test undergoing, naively expect 5% accuracy
- To come
  - scikit-learn → george
  - RSD emulator to combine g-g lensing and 3D clustering (needs bigger volume)
  - further extension under discussion
    - e.g., non-flat, w0-wa cosmologies



Evolution of cosmological simulations



“Numerical” prediction

最近 $\mathcal{N} > 10^{12}$ が達成された。

東京五輪2020までに $\mathcal{N} \sim 10^{15}$

　(この頃計算機はエクサ級へ到達)

そして、

2073年には $\mathcal{N}$ は<u>アボガドロ数</u>に達する

自己重力系におけるVlasov方程式

◆ Vlasov-Poisson方程式

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f}{\partial \boldsymbol{x}} - \nabla\phi \cdot \frac{\partial f}{\partial \boldsymbol{v}} = 0,$$

$$\nabla^2\phi = 4\pi G\rho = 4\pi G \int f d^3\boldsymbol{v}.$$

◆ 計算手法

- 位相空間を座標空間で3次元のregular meshに分割し、その中に3次元の運動量空間のregular meshで分割。
- Vlasov方程式は各成分に分割し、1次元の移流方程式に帰着させて計算。座標方向3本+運動量方向3本の計6本。
- 重力ポテンシャルはFFTを用いた畳み込み法。

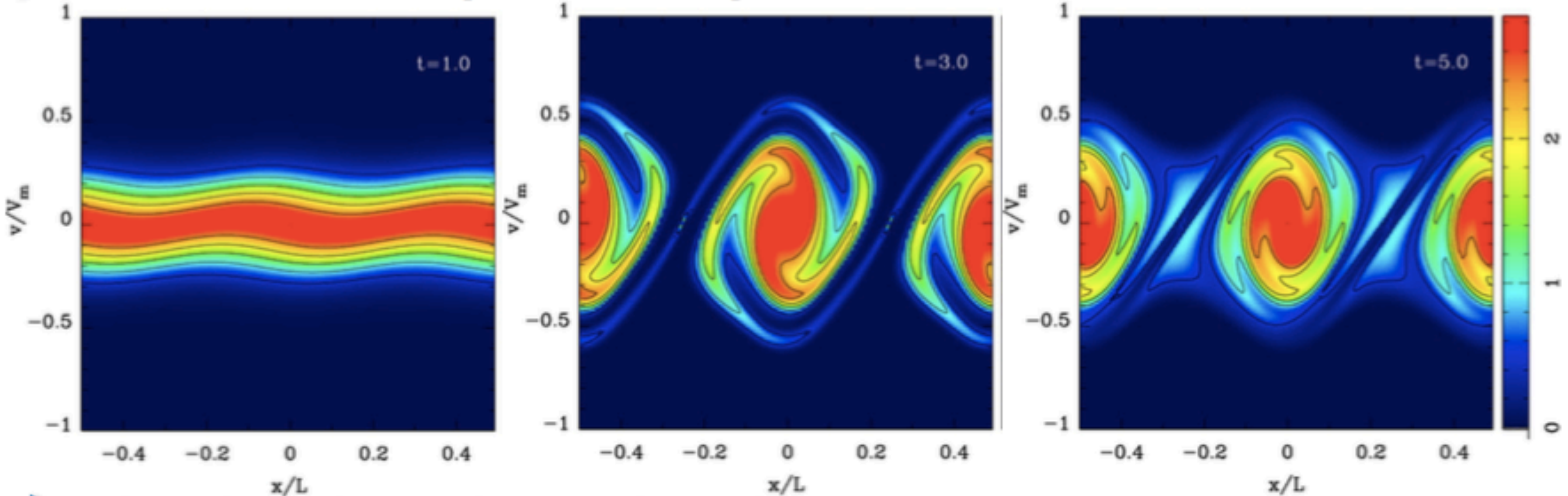
高次精度化

- 先行研究(Yoshikawa, Yoshida and Umemura 2013)
  - 6次元位相空間でのVlasov-Poissonシュミレーションを実現
  - **正値性、最大値の定理、質量保存の法則**
    を満たす数値解法である空間3次精度のPositive Flux Conservative スキーム　(Filbet et al. 2003)を使用
  - 6次元の計算なので高解像度を得る際に計算資源をつぎ込むのは困難。
    できるだけ少ないメッシュ数で高い精度が得られるスキームの実装が必要
  - PFCスキームを使用した先行研究ではCDMの宇宙論計算でN体計算と比較して大スケールのパワースペクトルでズレが見られた。
    → 現在つぎ込める計算資源で実現したい最低限の精度が足りていない
- Vlasov-Poisson方程式の精度
  - 移流方程式を解くスキームの精度に依存
  - 重力ポテンシャルを解く精度に依存

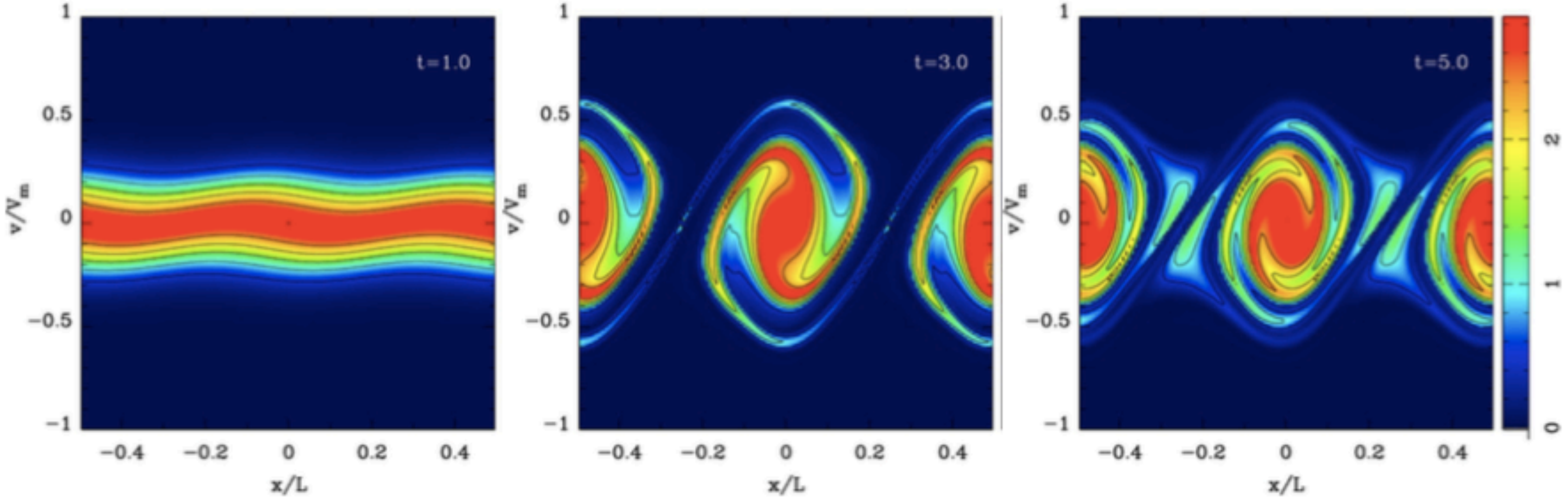
田中くん@筑波大 作成

高次精度化により位相空間で高解像度達成

▶ 3rd order scheme (PFC scheme)



▶ 5th order scheme (MP5 scheme)



Semi-Lagrange 法による時間発展

- ここまではRunge-Kutta法を使ったEuler法に基づくMPスキーム -> RKMP

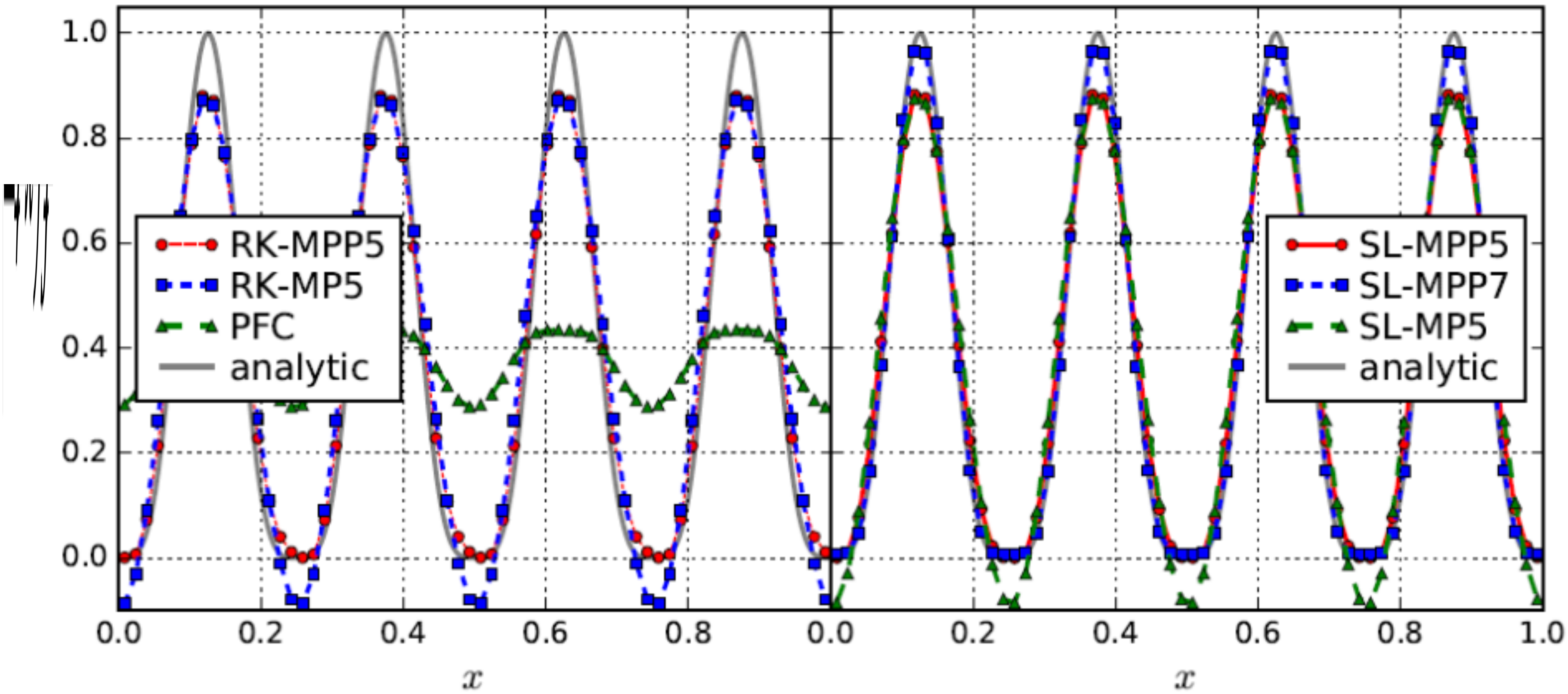
- 基本的に空間積分と時間積分の精度は安定性からあまり離れていないほうがいい。先行研究では空間精度 5 or 7 次に対して時間精度3次
  TVD Runge-Kuttaの5次や7次を使えばいいが、それに比例して計算時間が増えるのは避けたい。
  移流の計算段数と同時に境界のfluxのやり取りが必要なためノード間通信回数も増えてしまう。

- RKMP スキームでは PP Limiter を各ステップで計算する必要がある。

- 時間積分のステップ数が一回で済み、空間精度が(離れすぎた)時間精度に引きずられないようにMPスキームにSemi-Lagrange法を実装する。
  (簑島さんのアイデア)

1次元移流方程式テスト sin⁴ wave

$$f(x) = \sin^4(4\pi x).$$

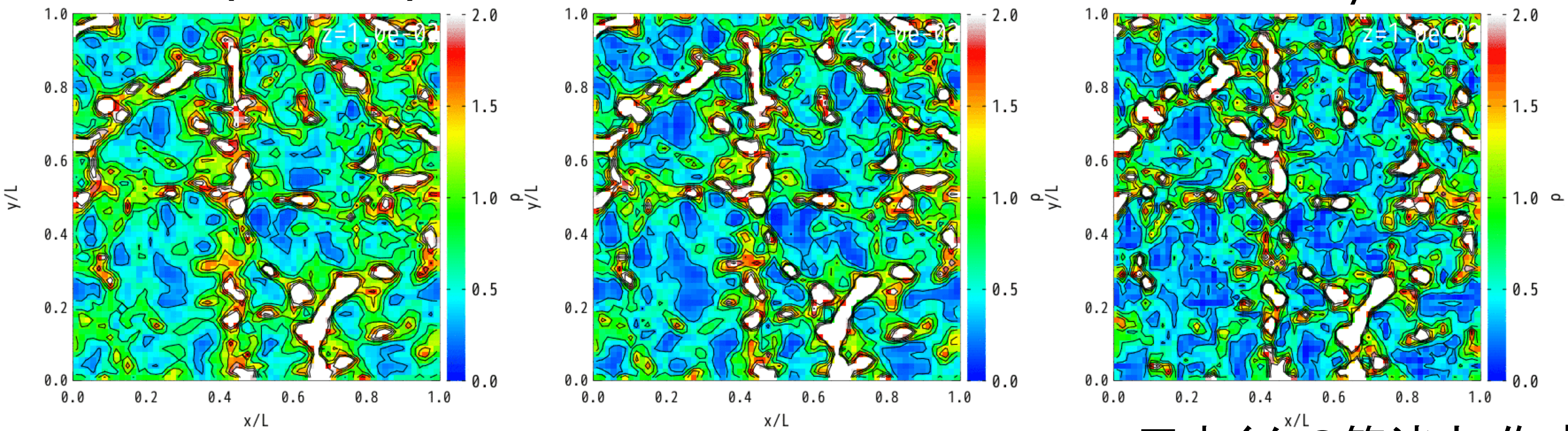


6次元宇宙論的ブラソフシミュレーション

- 6次元計算への拡張
  - 運動量空間の移流方程式に膨張項を適用
  - 膨張項を適用した方程式にも正値性を保証するPositive limiterを適用

- テスト計算：CDMでの大規模構造計算
  - ✓ PFCスキームを使用した先行研究ではN体計算と比較して大スケールのパワースペクトルでズレが見られた。
  - ✓ 今回はN体とPFC(空間3次精度)とRKMP5(空間5次精度)で比較。

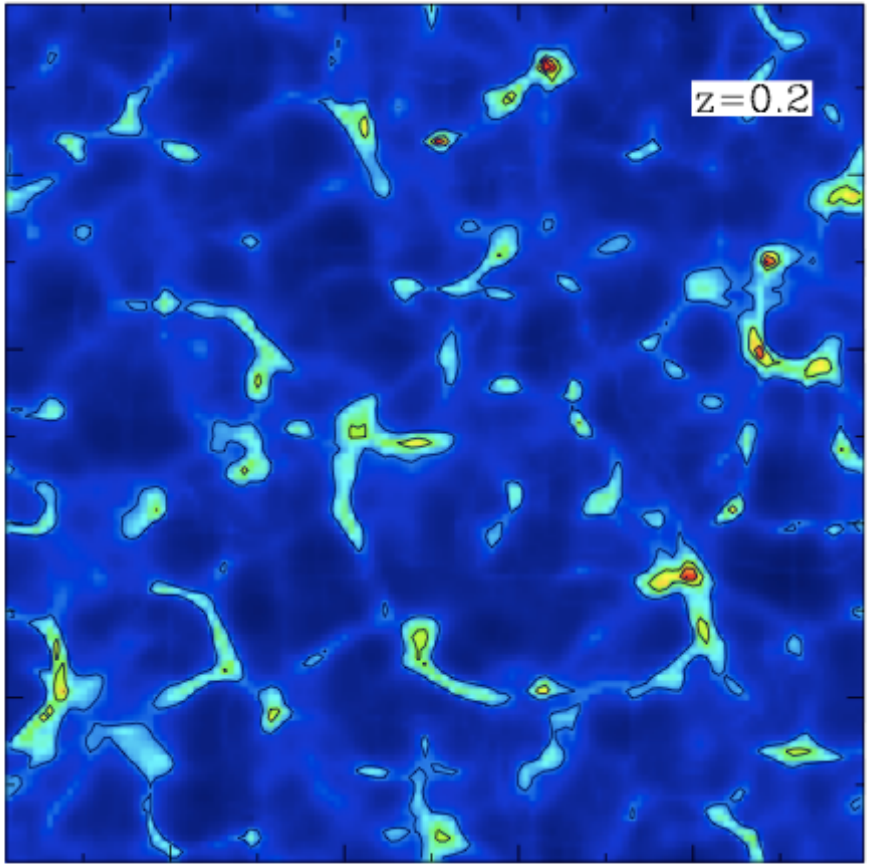
PFC(3次精度) RKMP5(5次精度) N-body



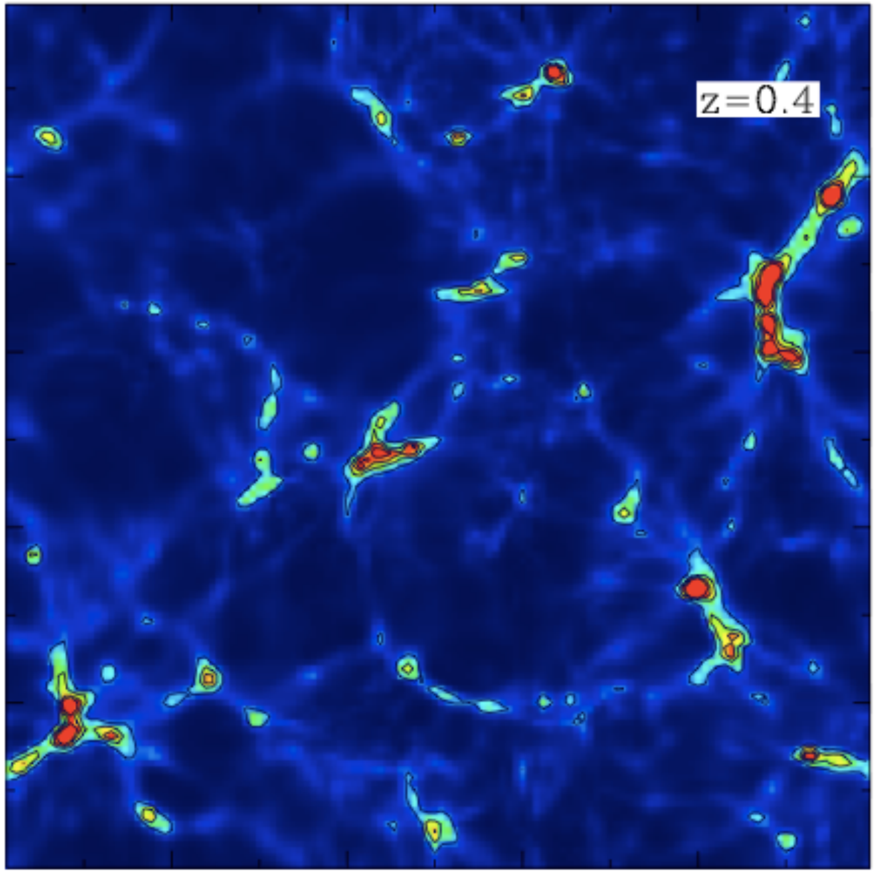
田中くん@筑波大 作成

128^6 simulation

Hot Dark Matter

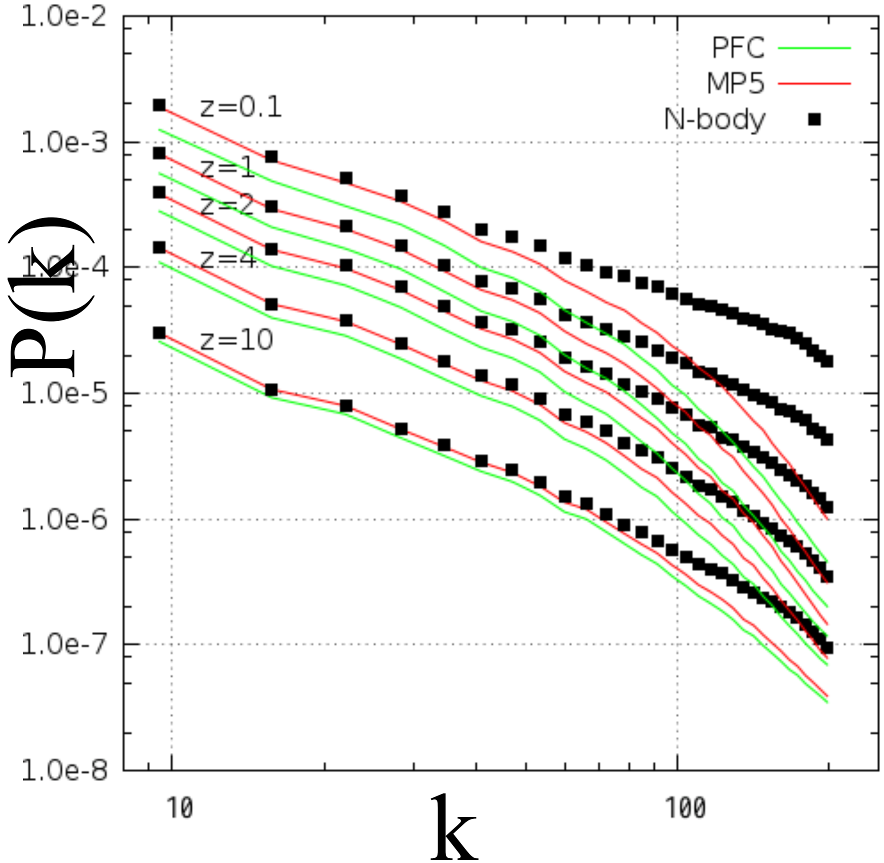


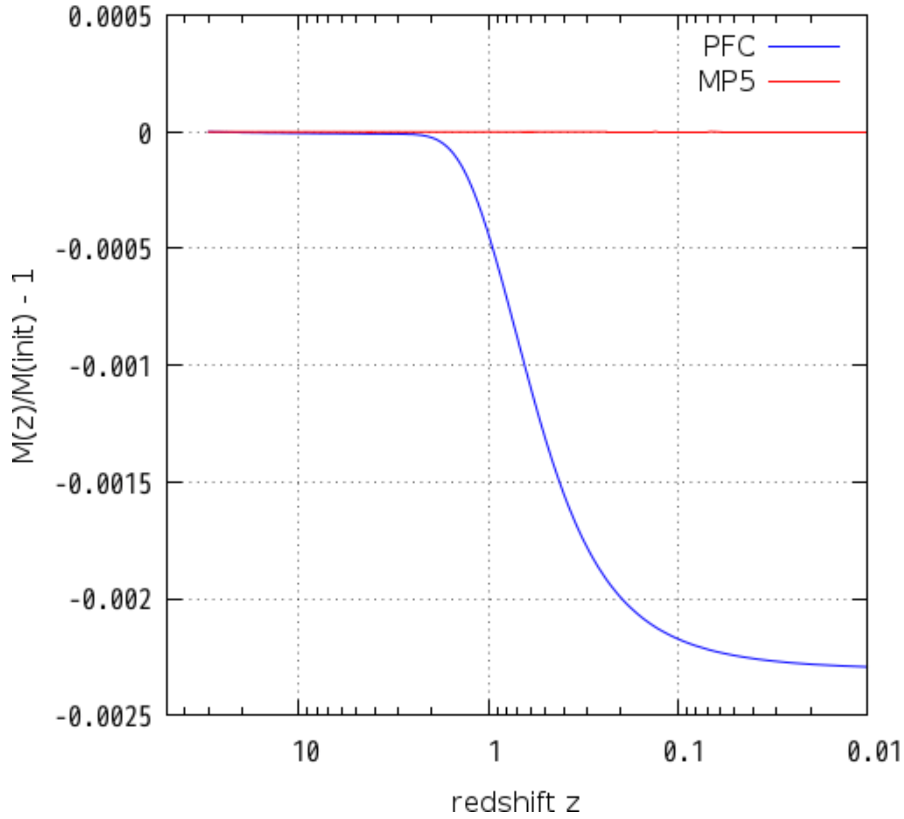
Cold Dark Matter



100 Mpc on a side

Power spectrum & Mass conservation





- ✓ zが小さくなるにつれ、3次精度のズレは大きくなるのに対し、5次精度では大スケールで一致。
- ✓ 小スケールのズレは解像度の問題。

- ✓ 3次精度では数値拡散が原因で質量が抜けていく。

田中くん@筑波大 作成

高速化

- Intel AVX, AVX2：256bit
  float x 8, double x 4
  cfca/XC30, YITP/XC40, ccs tsukuba/HA-PACS,COMA で使用

- Intel AVX512　：512bit
  float x 16, double x 8
  Oakforest-PACS の Xeon Phi (Knights Landing), skylake以降
  まだ触っていないが同様の実装をすればそれなりに速くなるはず

- Fujitsu HPC-ACE：128bit
  double x 2, float なし
  京で使用

田中くん@筑波大 作成