GPUによる 3D-RISM プログラムの高速化

[1]分子研、[2]総研大
丸山 豊[1]、平田 文男[1,2]

GPU用FFT(nufft)提供

東工大　額田彰博士

自己紹介

丸山豊 博士(理学)

分子科学研究所 平田グループ

専門研究職員（←平成23年度まで）

(次世代ナノ統合シミュレーションソフトウェア研究開発拠点)

生物物理化学、C/C++、Ruby、Lisp

目次

1.3D-RISM理論

2.GPU対応

3.性能評価

1. 3D-RISM理論

3D-RISM理論

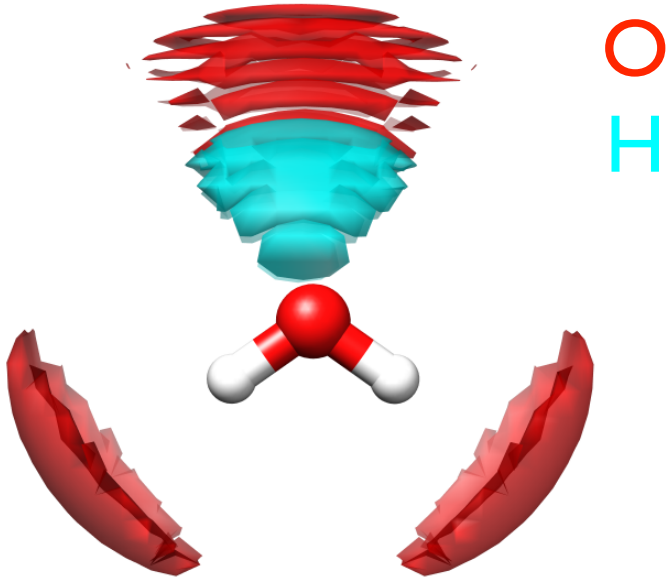
国産のプログラム（分子研平田G）

次世代スパコン中核アプリの一つ

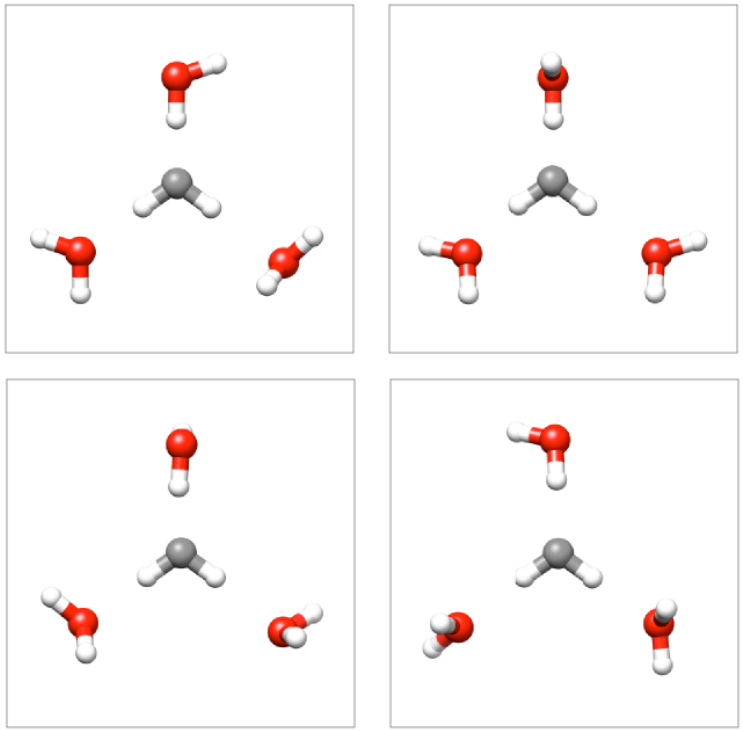
AMBER、ADFなどの商用ソフト

創薬開発への応用(JACS, 131,12430 (2009))

3次元分布



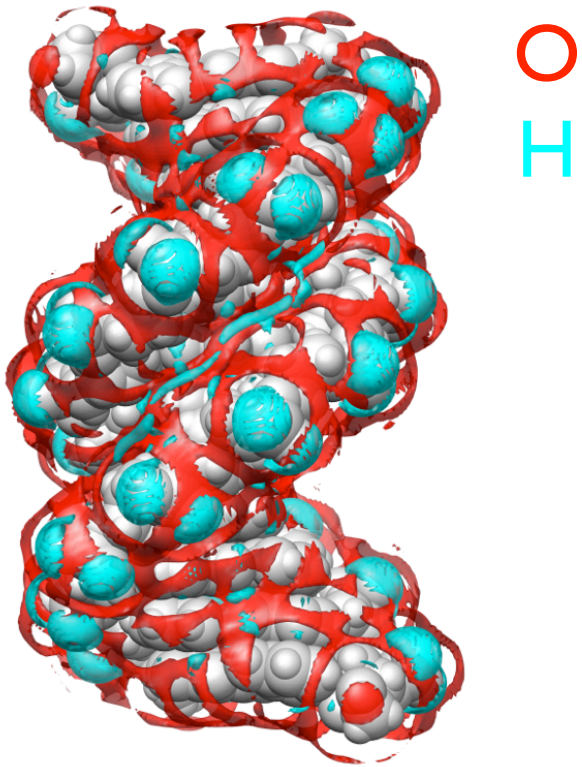
3次元分布

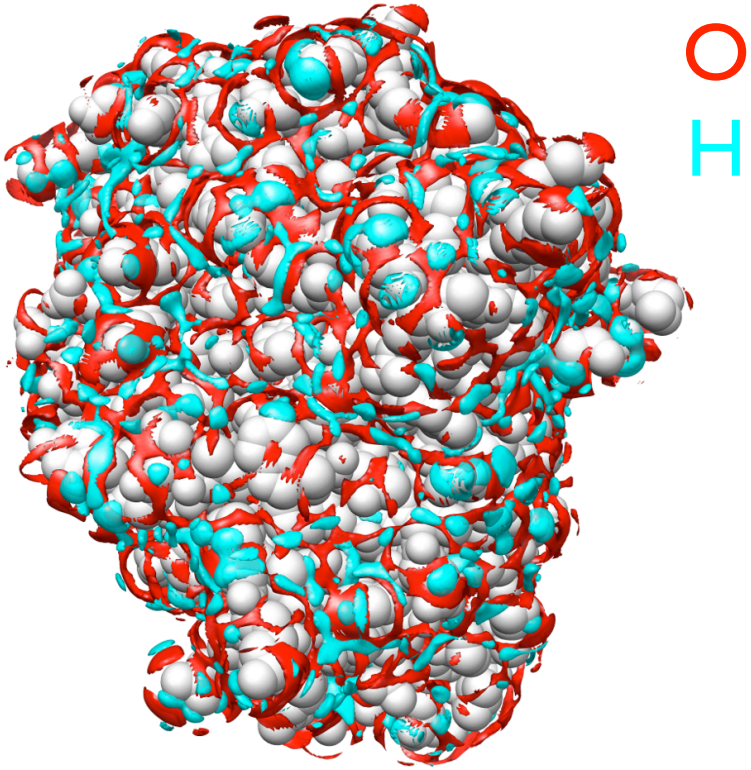


多数の配置

→ 分布から物理量(エネルギー等)の算出

応用例





DNA　　蛋白質

巨大な生体分子にも適用可能

3D-RISM方程式

3D-RISM 方程式

$$h_\gamma^{Ub}(\mathbf{r}) = c_\alpha^{Ua}(\mathbf{r}) * \left[\omega_{\alpha\gamma}^{ab}(r) + \rho^a h_{\alpha\gamma}^{ab}(r)\right]$$

フーリエ空間

3D-FFT

Kovalenko-Hirata クロージャ方程式

$$h_\gamma^{Ub}(r) = \begin{cases} \exp(\chi) - 1 & \text{for } \chi \leqslant 0 \\ \chi & \text{for } \chi > 0 \end{cases}$$

$$\chi = -\beta u_\gamma^{Ub}(r) + h_\gamma^{Ub}(r) - c_\gamma^{Ub}(r).$$

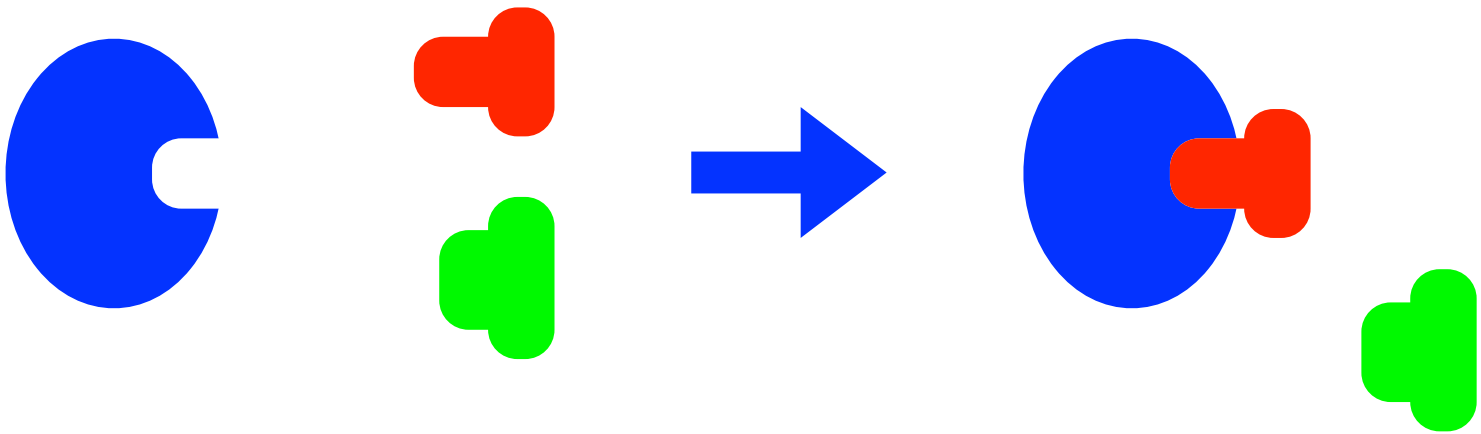
実空間

3D-FFTが必要 ← 高速化の阻害要因

3D-RISM方程式の解法

1. t(r)=h(r)-c(r)の初期値を計算する。
2. KHクロージャ方程式を解く。
3. フーリエ正変換を行う。
4. 3D-RISM方程式を解く。
5. フーリエ逆変換を行う。
6. 収束判定し、収束していたら終了する。
7. 外挿法によりt(r)を更新して2へ戻る。

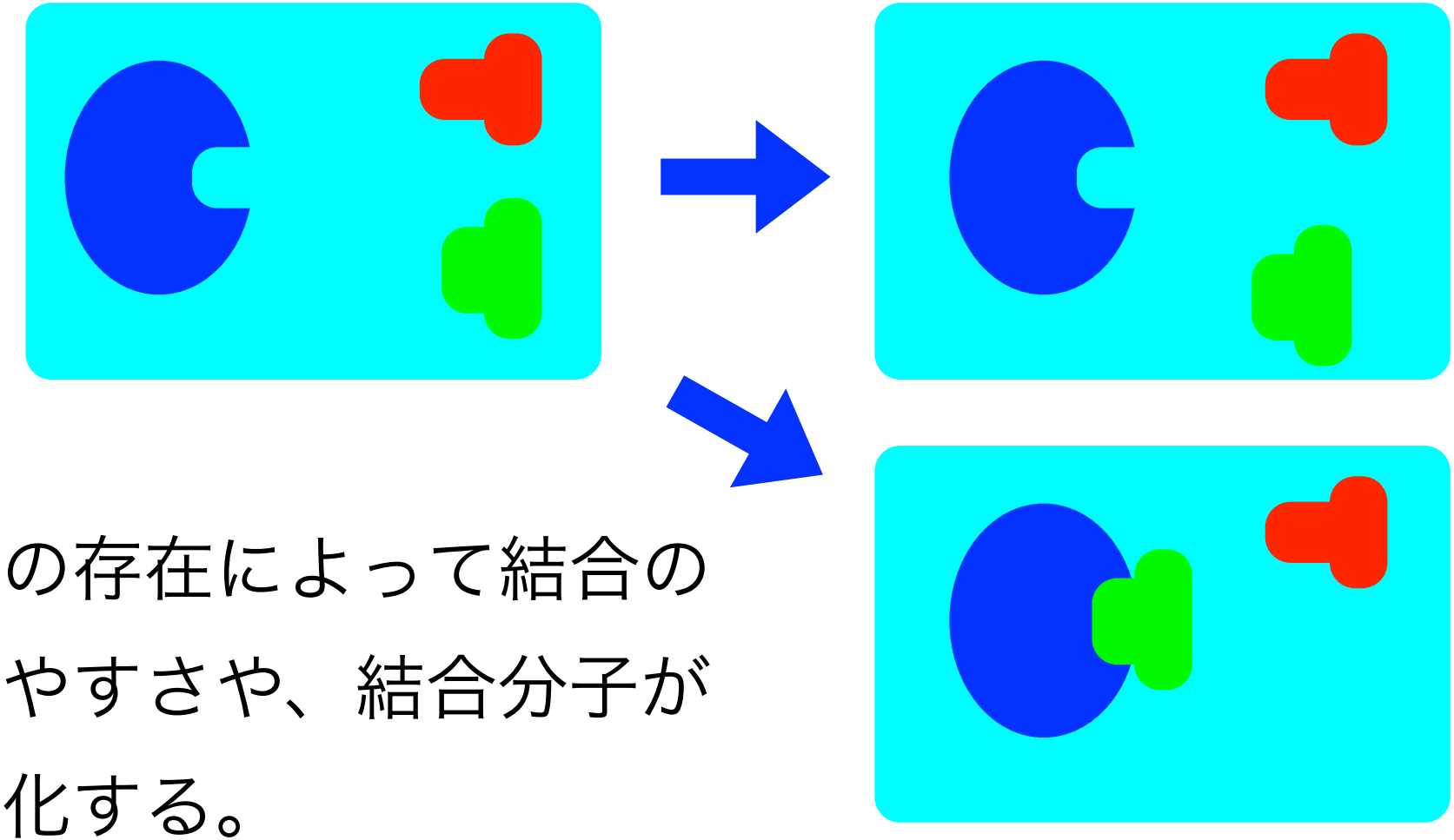
ドラッグデザイン



蛋白質　　薬

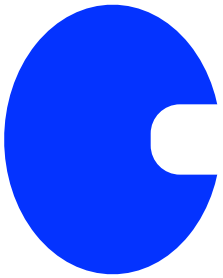
蛋白質の形にマッチする分子の探索

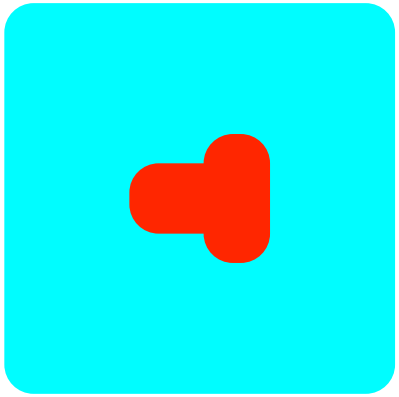
ドラッグデザイン



水の存在によって結合のしやすさや、結合分子が変化する。

ドラッグスクリーニング





M　　N (x n)

M x N (x n) 回の3D-RISM計算

→ 3D-RISMの高速化

JACS, 131,12430 (2009)

2. GPU対応

GPUの利点

メモリアクセスが速い

- GPU (Tesla C1060) 102.0GB/s (144GB/s)
- CPU (Intel Xeon) 25.6GB/s

	3DFFT	MDIIS	3D-RISM
割合(時間)	35.8%	32.8%	15.8%

3D-RISMの計算はメモリアクセスが支配的

GPUの欠点

搭載メモリが少ない。

- Tesla C1060 4GB
- 3D-RISMでは9GB必要(内MDIISで6GB)

  溶媒 水、256^3(0.25Å)

省メモリ化が必要

MDIISに換えて修正Anderson法を使用

修正Anderson法

ミキシング法の一種

過去の履歴の混ぜ合わせ率を動的に変化させる

少ない履歴で効果的に収束 → 省メモリ

修正Anderson法

t=h-c: 3D-RISM-KHで求まった量 τ:更新量

$\tau^{(n)} = u^{(n-1)}+b^{(n-1)}(v^{(n)}-u^{(n-1)})$ ← τの更新

$u^{(n-1)} = t^{(n-1)}+\Sigma(\theta_j{}^{(n-1)}+s_j)(t^{(n-j-1)}-t^{(n-1)})$

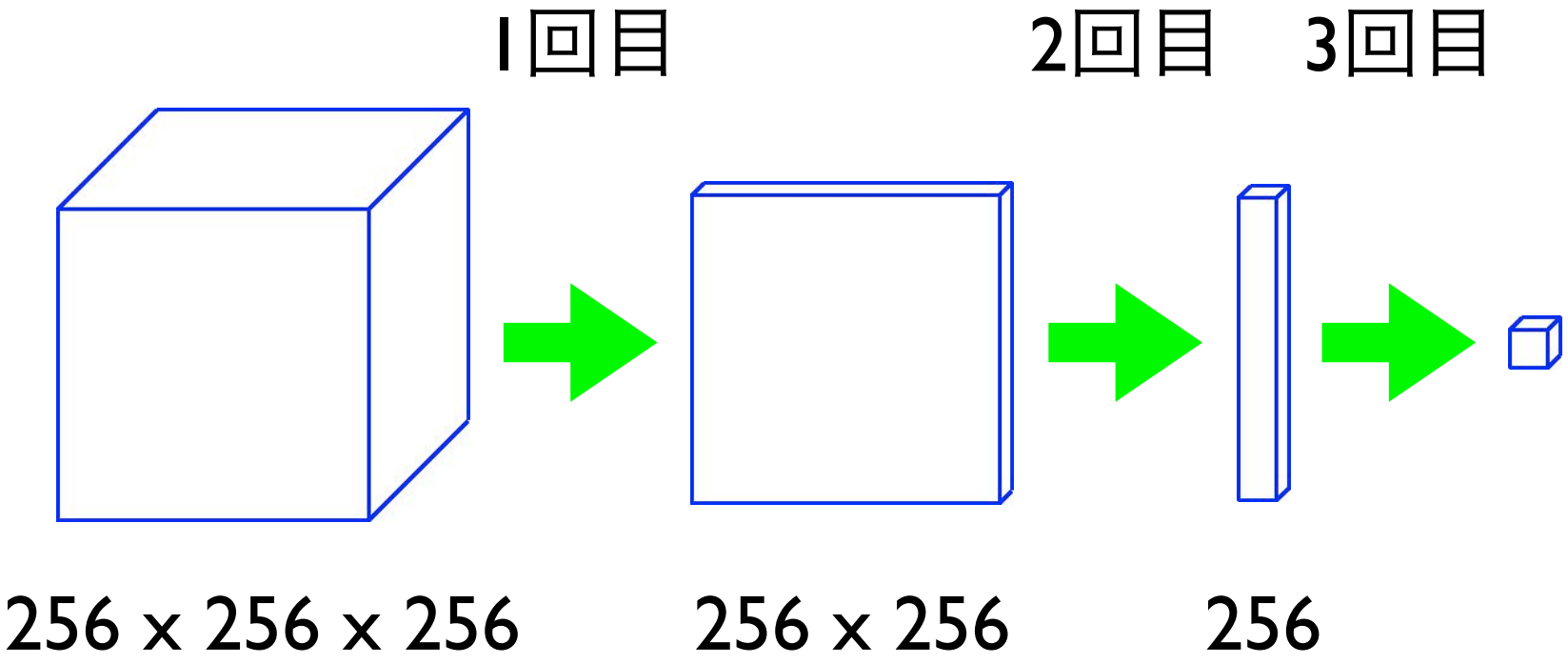
$v^{(n)} = \tau^{(n)}+\Sigma(\theta_j{}^{(n-1)}+s_j)(\tau^{(n-j)}-\tau^{(n)})$

$\Sigma(r^{(n-1)}-r^{(n-i-1)}, r^{(n-1)}-r^{(n-j-1)})\theta_j{}^{(n-1)}$ 重み付内積

$= (r^{(n-1)}, r^{(n-1)}-r^{(n-j-1)})$ $(u,v)=\Sigma u_i v_i w_i$

$r^{(n-1)} = t^{(n)}-\tau^{(n-1)}$

多段Reduction



16777216(=256^3)の要素を3回に分けてReduction

GPU化の方針

- 収束法を修正Anderson法に変更
- 収束し終わるまでGPU上で計算

CPUとGPUで比較

- CPU(シングルコア) MDIIS
- GPU(C1060) 修正Anderson法

3. 性能評価

計算条件

DNA 12塩基対(754原子)

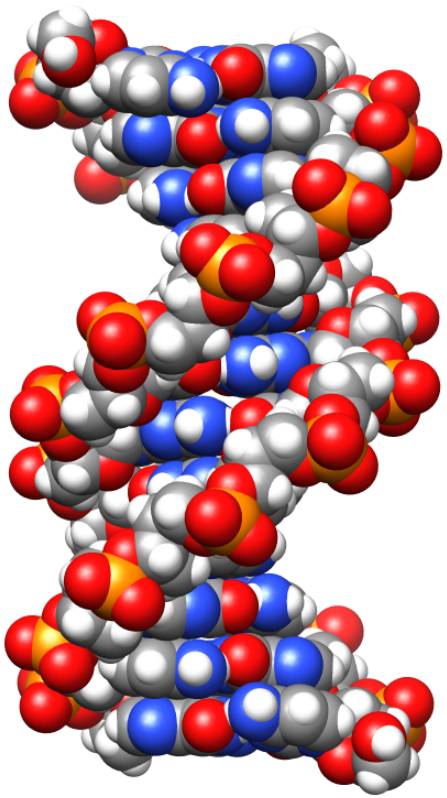
溶媒 水(2成分)

セル 256^3(0.25Å)

Intel Core i7 (2.67GHz)

NVIDIA Tesla C1060

倍精度で計算



JPC B, 2010, *114*, pp 6464

繰り返し部分の時間

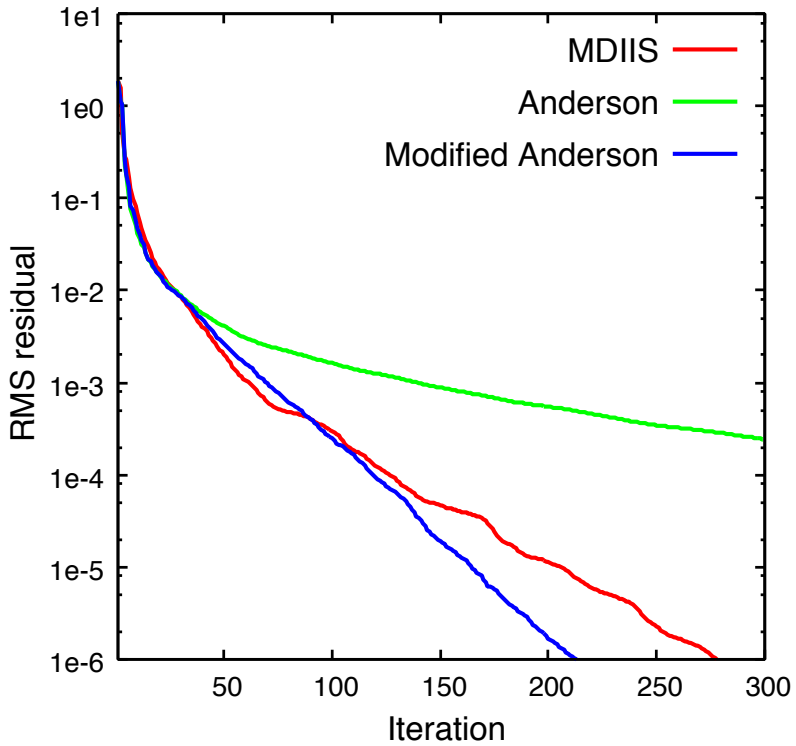
Method	MDIIS(10) (CPU)	Anderson (GPU)	MAnderson (GPU)
Time	2546.1s	857.1s (686.0)	132.7s (106.2)

cufft (nufft)

nufft is presented by A. Nukada.

修正Anderson法とnufftで約24倍の高速化

繰り返し回数と収束



MDIISよりも早く安定して収束

サブルーチン毎の割合

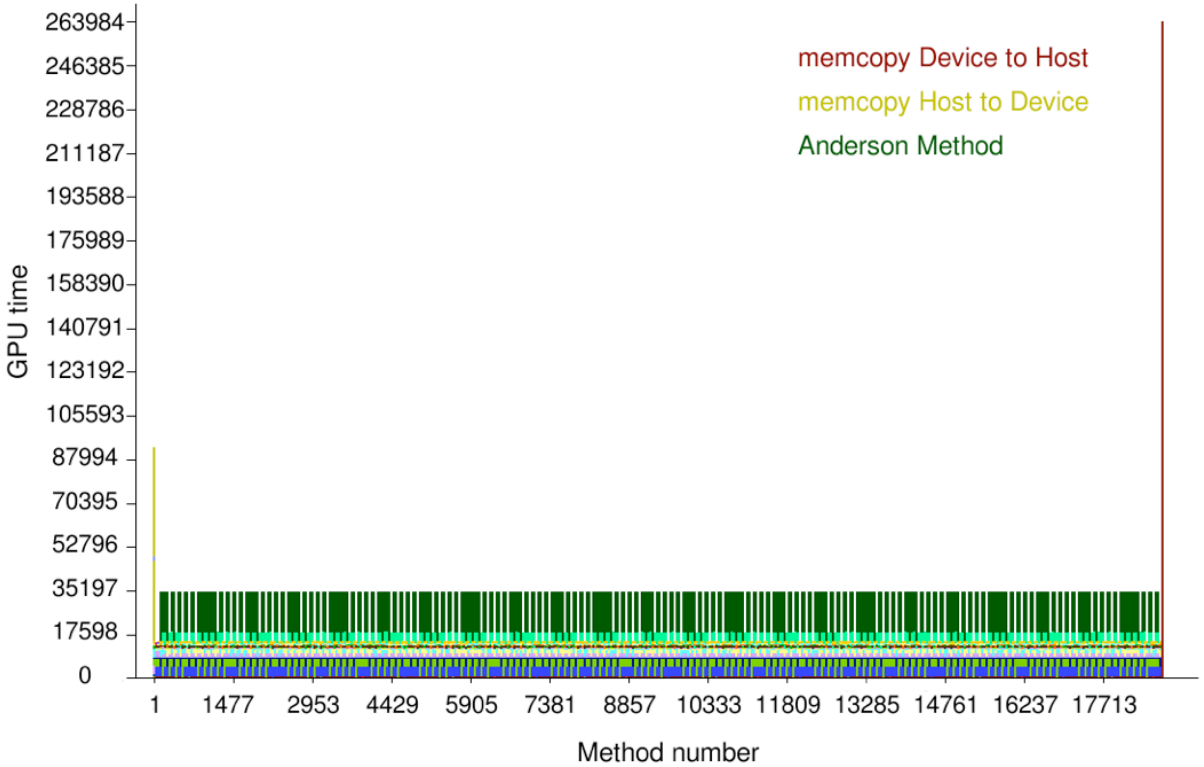
GPU

	FFT	Anderson	3D-RISM
cufft	49.1	17.6	17.6
nufft	36.2	22.1	22.1

nufftでFFT部分が改善される。

Reduction以外のメモリアクセスは順当。

サブルーチン毎の時間



セル全体のデータ転送が必要なので、CPU-GPU間のデータ転送はコストが高い。

まとめ

GPUでCPUシングルコアの24倍の高速化

CPU-GPU間のメモリ転送を考えるとマルチGPU化は難しい？

→「マルチGPUによる3D-RISMの高速化」

で科研費申請