



計算科学技術特論B

# 大規模量子化学計算(1)

小林 正人

(北海道大学理学研究院)

K-masato@mail.sci.hokudai.ac.jp

2014/07/03

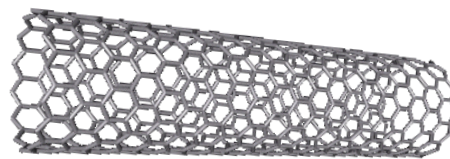
# 講義概要

- 7/3 量子化学計算の概要と構成要素、高速化
  - ◆ 量子化学計算の目的と種類
    - 量子化学計算ソフトウェア
  - ◆ 量子化学計算の手順、構成要素と高速化
    - SCF計算: 2電子積分の高速計算アルゴリズム
    - Post HF計算: MP2エネルギー計算のOpenMP並列化
- 7/10 フラグメント分割による大規模量子化学計算法
  - ◆ フラグメント分子軌道法
  - ◆ 分割統治法

# 量子化学計算とは

- 化学(物質科学)の対象 = 分子(集団)・固体

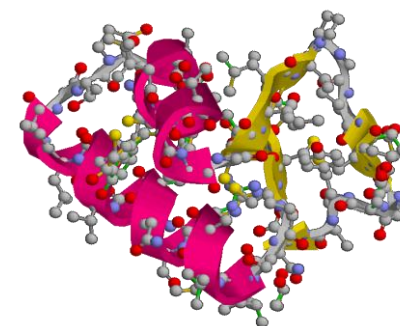
◆ 構成要素 { 原子核  
電子



(原子核と)電子の  
量子論的な振舞い



物質の  
あらゆる性質



数値的に求める =

量子化学計算

- 典型的量子化学計算

- ◆ 入力: 原子の座標、電子数、計算手法・基底関数
- ◆ 出力: 分子のエネルギー、電子分布(分子軌道)、プロパティ、エネルギー微分(場合によって)

# 量子化学計算ソフトウェア

## ■ 商用ソフトウェア

- Gaussian 09, Q-Chem, MOLPRO, Molcas, Turbomole, ADF

## ■ 無料ソフトウェア

### ◆ バイナリのみ提供

- NTChem, ORCA, Firefly, CONQUEST

### ◆ ソースコード開示 (ライセンス制限あり)

- GAMESS, DIRAC

### ◆ オープンソース (GPL, Apache license等)

- NWChem, ACES III, SMASH, OpenMX

# 入力ファイルの例

- SMASHプログラムによる水分子の計算
  - ◆ SMASH: 今夏公開予定の高並列量子化学パッケージ (by Dr. K. Ishimura, IMS)

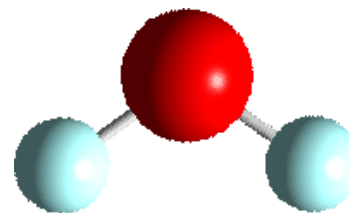
計算法の指定  
(B3LYPは密度汎関数の一種)

基底関数系の指定

```
job runtime=energy method=B3LYP basis=cc-pVDZ
```

```
geom
```

```
O  0.000    0.000    0.122
H  0.000    0.793   -0.487
H  0.000   -0.793   -0.487
```

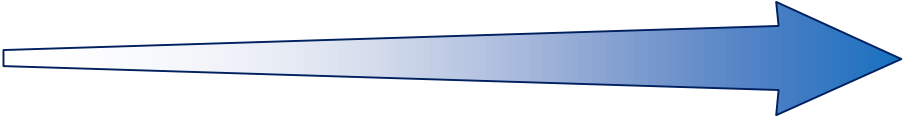


# 量子化学計算の種類

- **Hartree-Fock (HF)法**: 量子化学計算の基本
  - ◆ 非経験的な電子状態の平均場理論
  - ◆ 2電子積分計算と密対称行列対角化
- **密度汎関数理論(DFT)計算**: 現在主流の計算法
  - ◆ 形式的にはほぼHFと同じだが、高精度化が可能
  - ◆ HF + 密度汎関数の空間数値積分
- **Post-HF法**: HF計算の後に実行する高精度計算
  - ◆ 前二者とは基本的に異なる演算タイプ
  - ◆ MP法, CC法: 積分変換とテンソル縮約
  - ◆ CI法: 大規模疎行列の対角化

# 量子化学計算にかかる時間と精度

分子の大きさの3乗に比例して計算時間増大

方法	Hartree-Fock (HF)法	密度汎関数理論(DFT)	MP2法 (摂動法)	CCSD法	CCSD(T)法
計算時間	$O(N^3)$	$O(N^3)$	$O(N^5)$	$O(N^6)$	$O(N^7)$
近似レベル	← 平均場理論 →		← 電子相関理論 →		
計算精度	定性的				正確

計算時間は

- ✓ 精度の低い理論でも $O(N^3)$
- ✓ 精度が上がるにつれて莫大に

# Hartree-Fock(-Roothaan)法: 基本

- 分子中の電子のSchrödinger方程式

$$\hat{H} \Psi(r_1, r_2, \dots, r_N) = E \Psi(r_1, r_2, \dots, r_N)$$

- ◆ 直接 $N$ 電子波動関数 $\Psi$ を決定するのは難しい

- 1体近似 (Hartree-Fock近似)

$$\Psi(r_1, r_2, \dots, r_N) \approx |\psi_1(r_1) \psi_2(r_2) \dots \psi_N(r_N)|$$

$$\hat{H}_{\text{eff}} \psi_p(r_1) = \varepsilon_p \psi_p(r_1)$$

- ◆ 電子が作る平均の場の中で運動する1電子問題に帰着
- ◆  $\psi$ を所与の関数(基底関数)で展開: Roothaan法

$$\psi_p(r) = \sum_{\mu} C_{\mu p} \phi_{\mu}(r)$$



# 量子化学計算の基底関数

## ■ 原子に局在した関数(原子軌道)

◆ 「分子 = 原子の集まり」という描像に一致

■ 基底関数の数を大幅に削減

■ 行列要素の計算に必要な分子積分が複雑化

◆ スレーター型関数:  $\chi^S = N x^l y^m z^n \exp(-\zeta r)$

■ 水素原子の正確な波動関数に一致

■ 分子積分の計算が非常に面倒

◆ ガウス型関数:  $\chi^G = N x^l y^m z^n \exp(-\alpha r^2)$

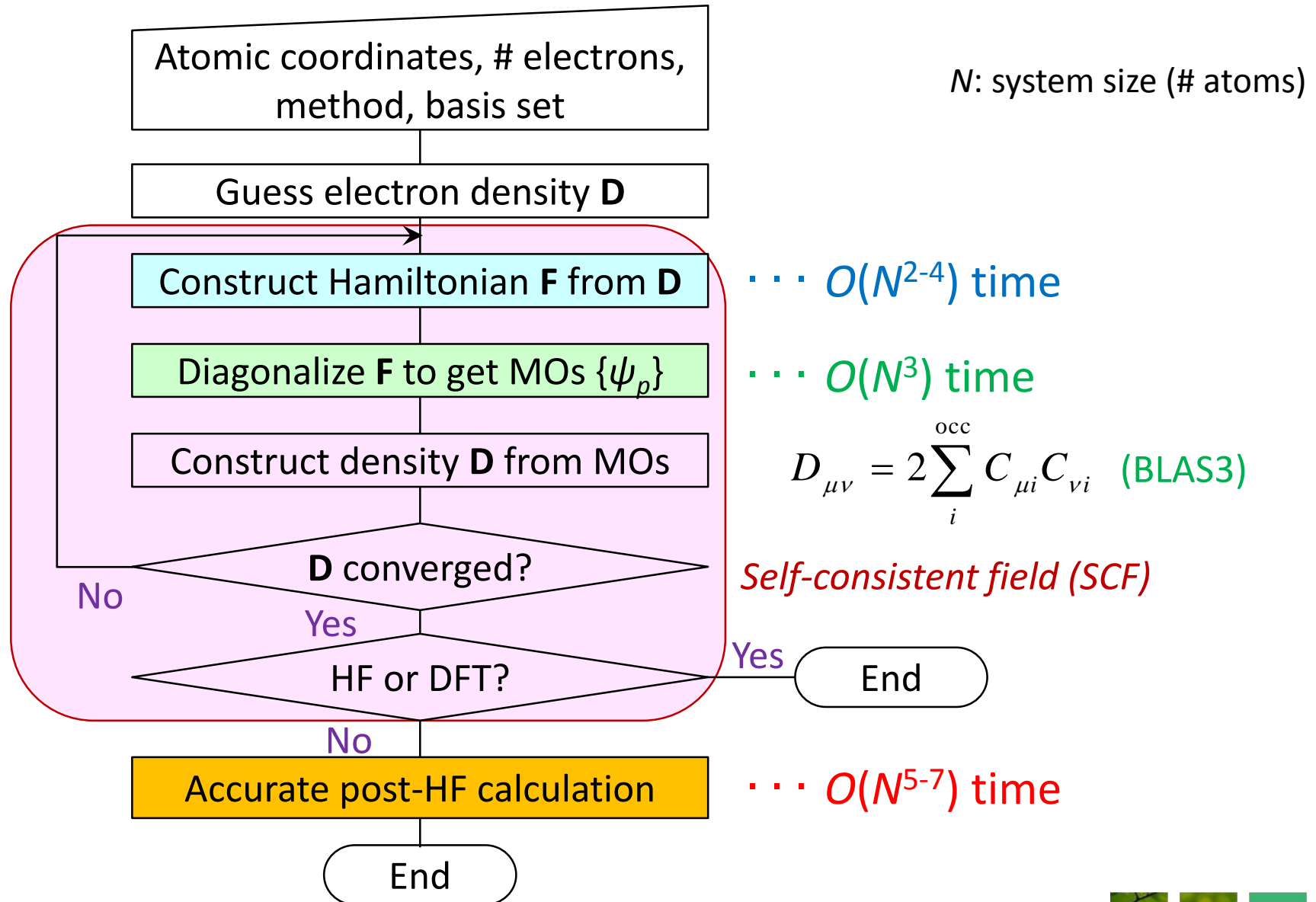
■ ガウス型関数の積がガウス型関数になるので、計算が簡略

■ 量子化学計算のスタンダード

◆ 原子様数値基底 (cf: 尾崎先生)

■ 有限要素基底, Wavelet基底, 平面波基底

# 量子化学計算の手順



# SCFの構成要素(1): Fock行列生成

## ■ Fock行列: $\mathbf{F} = \mathbf{H}^{\text{core}} + \mathbf{G}$ (対称行列)

◆  $\mathbf{H}^{\text{core}}$ : 電子密度に依存しない定数項(メモリに保存)

◆  $G_{\mu\nu} = \sum_{\lambda\sigma} D_{\lambda\sigma} [2\underline{\Gamma_{\mu\nu,\sigma\lambda}} - \underline{\Gamma_{\mu\lambda,\sigma\nu}}]$

クーロン項J(古典的な項)      交換項K(量子力学的な項)

■ 密度行列 $\mathbf{D}$ に依存 [繰り返し計算される]

■  $\Gamma_{\mu\nu,\sigma\lambda}$ : 2電子積分(後述) [4階のテンソル、メモリ保存不可]

◆  $\Gamma_{\mu\nu,\lambda\sigma} = \Gamma_{\mu\nu,\sigma\lambda} = \Gamma_{\nu\mu,\lambda\sigma} = \Gamma_{\nu\mu,\sigma\lambda} = \Gamma_{\lambda\sigma,\mu\nu} = \Gamma_{\sigma\lambda,\mu\nu} = \Gamma_{\lambda\sigma,\nu\mu} = \Gamma_{\sigma\lambda,\nu\mu}$ : 高い対称性

## 2つの方針

- ✓ Conventional SCF:  $\Gamma$ を外部記憶に保存
- ✓ Direct SCF:  $\Gamma$ を毎回計算して捨てる

# Fock行列生成(Direct SCF)の擬似コード

```
1: F ← 0
2: Loop over  $\mu\nu\lambda\sigma$  ( $\mu_{sh} \geq \nu_{sh}, \lambda_{sh} \geq \sigma_{sh}, [\mu\nu]_{sh} \geq [\lambda\sigma]_{sh}$ )
   sh: シェル(同中心のいくつかの基底関数をまとめた単位)
3:   If (MOD( $\mu\nu\lambda\sigma$ , NPROC) == MYRANK) Then
4:     If (Estimate_Magnitude( $\mu\nu\lambda\sigma$ , D) >= thresh) Then
5:       Call ConstructGamma(gamma,  $\mu\nu\lambda\sigma$ )
6:       Loop over  $\mu\nu\lambda\sigma$ 
7:         F( $\mu\nu$ ) ← F( $\mu\nu$ ) + 4*D( $\lambda\sigma$ ) * gamma( $\mu\nu\lambda\sigma$ )
8:         F( $\lambda\sigma$ ) ← F( $\lambda\sigma$ ) + 4*D( $\mu\nu$ ) * gamma( $\mu\nu\lambda\sigma$ )
9:         F( $\mu\lambda$ ) ← F( $\mu\lambda$ ) - D( $\nu\sigma$ ) * gamma( $\mu\nu\lambda\sigma$ )
10:        F( $\mu\sigma$ ) ← F( $\mu\sigma$ ) - D( $\nu\lambda$ ) * gamma( $\mu\nu\lambda\sigma$ )
11:        F( $\nu\lambda$ ) ← F( $\nu\lambda$ ) - D( $\mu\sigma$ ) * gamma( $\mu\nu\lambda\sigma$ )
12:        F( $\nu\sigma$ ) ← F( $\nu\sigma$ ) - D( $\mu\lambda$ ) * gamma( $\mu\nu\lambda\sigma$ )
13:      End Loop
14:    End If
15:  End If
16: End Loop
17: Call MPI_AllReduce(F)
18: F ← F + HCORE
```

✓ 様々な並列化パターンが検討

✓ OpenMP並列も同時に可

✓ 対称性を利用

✓ Dを用いたスクリーニング

# Fock行列計算のスクリーニング

## ■ Cauchy-Schwarzの不等式の利用

$$|x_1 y_1 + x_2 y_2 + \dots| \leq \sqrt{(x_1^2 + x_2^2 + \dots)(y_1^2 + y_2^2 + \dots)}$$



$$|\Gamma_{\mu\nu,\lambda\sigma}| \leq \sqrt{\Gamma_{\mu\nu,\mu\nu}} \cdot \sqrt{\Gamma_{\lambda\sigma,\lambda\sigma}}$$

◆  $\Gamma_{\mu\nu,\lambda\sigma}$  にかけてられる密度行列の最大絶対値:

$$D_{\max} = \text{Max}(4|D_{\mu\nu}|, 4|D_{\lambda\sigma}|, |D_{\mu\lambda}|, |D_{\mu\sigma}|, |D_{\nu\lambda}|, |D_{\nu\sigma}|)$$

$$D_{\max} \sqrt{\Gamma_{\mu\nu,\mu\nu}} \cdot \sqrt{\Gamma_{\lambda\sigma,\lambda\sigma}} \geq \varepsilon_{\text{thresh}}$$

を満たす  $\Gamma_{\mu\nu,\lambda\sigma}$  のみ計算

■ 実際にはシェルごとに見積もり

# ガウス型基底に対する2電子積分

$$\Gamma_{\mu\nu,\lambda\sigma} = \iint d\mathbf{r}_1 d\mathbf{r}_2 \phi_\mu(\mathbf{r}_1) \phi_\nu(\mathbf{r}_1) r_{12}^{-1} \phi_\lambda(\mathbf{r}_2) \phi_\sigma(\mathbf{r}_2)$$

## ■ 4重ループ( $\mu, \nu, \lambda, \sigma$ )内で計算

◆ 演算量: 形式的に $O(N^4)$  → スクリーニングで $O(N^{2-3})$

◆ MPIおよびOpenMP並列化

- 計算量が一定でなく、スクリーニングも存在するため、ロードバランシングはそれほど簡単ではない

## ■ 各2電子積分の計算については、並列化不要

## ■ 縮約の存在: 計算が複雑化

$\chi_\mu^i(\mathbf{r})$ : 原始ガウス関数

$$\phi_\mu(\mathbf{r}) = \sum_{i=1}^{K_\mu} c_\mu^i (x - R_x^\mu)^{n_x} (y - R_y^\mu)^{n_y} (z - R_z^\mu)^{n_z} \exp\left(-\alpha_\mu^i |\mathbf{r} - \mathbf{R}^\mu|^2\right)$$

# ガウス型基底に対する2電子積分

$$\begin{aligned}\Gamma_{\mu\nu,\lambda\sigma} &= \iint d\mathbf{r}_1 d\mathbf{r}_2 \phi_\mu(\mathbf{r}_1) \phi_\nu(\mathbf{r}_1) r_{12}^{-1} \phi_\lambda(\mathbf{r}_2) \phi_\sigma(\mathbf{r}_2) \\ &= \sum_{i,j,k,l}^{K_\mu, K_\nu, K_\lambda, K_\sigma} c_\mu^i c_\nu^j c_\lambda^k c_\sigma^l \iint d\mathbf{r}_1 d\mathbf{r}_2 \chi_\mu^i(\mathbf{r}_1) \chi_\nu^j(\mathbf{r}_1) r_{12}^{-1} \chi_\lambda^k(\mathbf{r}_2) \chi_\sigma^l(\mathbf{r}_2)\end{aligned}$$

## ■ 原始ガウス関数に対する積分: Rys求積法

◆  $K$ が大きい場合に計算量が大幅に[ $O(K^4)$ ]増加

## ■ 軌道角運動量量子数 $L$ に応じて、expの中身は同じで方向だけが異なる関数をいくつも計算

◆  $L=0$  ( $s$ )  $\rightarrow$  1,  $L=1$  ( $p$ )  $\rightarrow$  3,  $L=2$  ( $d$ )  $\rightarrow$  6(5),  $L=4$  ( $f$ )  $\rightarrow$  10(7)



縮約と角運動量計算の演算順序を最適化して高速化できる余地あり

# 新しい2電子積分計算アルゴリズムの開発

## ■ 2電子積分の随伴座標展開(ACE)表式

$$\Gamma_{\lambda\mu,\nu\xi} = \sum_{\{\mathbf{N}_3\}} \underbrace{C_4^{ABCD} \{\mathbf{N}_3\}}_{\text{随伴座標部 } [O(K^0)]} \underbrace{H_{4\lambda\mu\nu\xi}^{ABCD} \{\mathbf{N}_3\}}_{\text{核部 } [O(K^4)]}$$

$K$ : Gauss関数の縮約長

$$H_{4\lambda\mu\nu\xi}^{ABCD} \{\mathbf{N}_3\} = S^{AB} S^{CD} \sum_{i_1} \sum_{i_2} G_{i_1 i_2} \sigma_A^{b_A+d_A} \sigma_B^{a_B+c_B} \sigma_C^{d_C+b_C} \sigma_D^{c_D+a_D} \frac{\sigma_1^{M_A+M_B-i_1} \sigma_2^{M_C+M_D-i_2}}{(\sigma_1 + \sigma_2)^{M-i_1-i_2}} \\ \times \sum_{s_1} \sum_{s_2} (-)^{s_1+s_2} \binom{a_B + b_A}{s_1} \binom{c_D + d_C}{s_2} \left( \frac{\sigma_1}{\sigma_1 + \sigma_2} \right)^{i'+j'} \left( \frac{\sigma_2}{\sigma_1 + \sigma_2} \right)^{k'+h'}$$

## ◆ 核部に対して2つの漸化関係式(RR)を導出

- ACE水平漸化関係式(ACE-HRR)
- ACE垂直漸化関係式(ACE-VRR)

ACE-RR法

RRの計算は $K$ のループの外側で実行!



# 2電子積分の計算コスト

$$\Gamma_{\mu\nu,\lambda\sigma} = \sum_{i,j,k,l}^{K_\mu, K_\nu, K_\lambda, K_\sigma} c_\mu^i c_\nu^j c_\lambda^k c_\sigma^l \iint d\mathbf{r}_1 d\mathbf{r}_2 \chi_\mu^i(\mathbf{r}_1) \chi_\nu^j(\mathbf{r}_1) r_{12}^{-1} \chi_\lambda^k(\mathbf{r}_2) \chi_\sigma^l(\mathbf{r}_2)$$

## ■ 浮動小数点演算(FLOP)数で評価

◆  $\text{FLOP} = x K^4 + y K^2 + z$  ( $K$ 一定の場合)

例) ( $sp\ sp \mid sp\ sp$ ) タイプの2電子積分計算コスト<sup>[1]</sup>

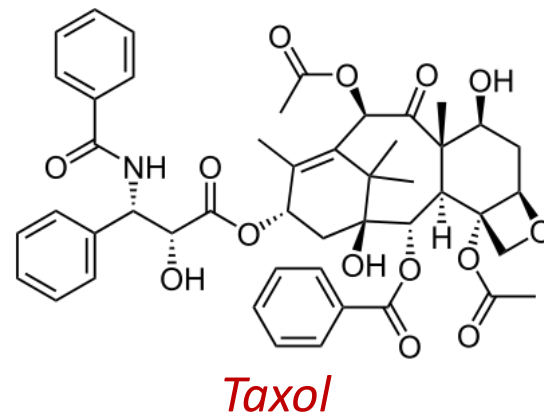
Algorithm	$x$	$y$	$z$
ACE-RR (Kobayashi & Nakai)	155	774	4 548
ACE-b3k3 (Ishida)	349	572	4 282
Ten-no	180	510	4 040
Pople & Hehre	220	2 300	4 000
Gill, Head-Gordon, & Pople	450	1 300	1 700
Dupuis, Rys, & King	1 056	30	0

昨今では、必ずしも  
FLOPが適切な指標  
とは言えない

[1] M. Kobayashi and H. Nakai, *J. Chem. Phys.* **121**, 4050 (2004).

## 2電子積分の計算時間

- タキソール分子を計算した時の2電子積分計算にかかる時間<sup>[1]</sup>
- ◆ 基底関数: STO-KG ( $K = 1-6$ )
- ◆ Direct SCF (スクリーニングあり)



Method	Time [s]					
	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$	$K = 6$
ACE-RR	11.0	42.1	91.4	187.7	360.5	631.4
Pople-Hehre	13.0	47.4	111.5	237.7	463.1	814.0
Dupuis-Rys-King	10.0	96.4	376.1	1075.7	2462.2	4855.0

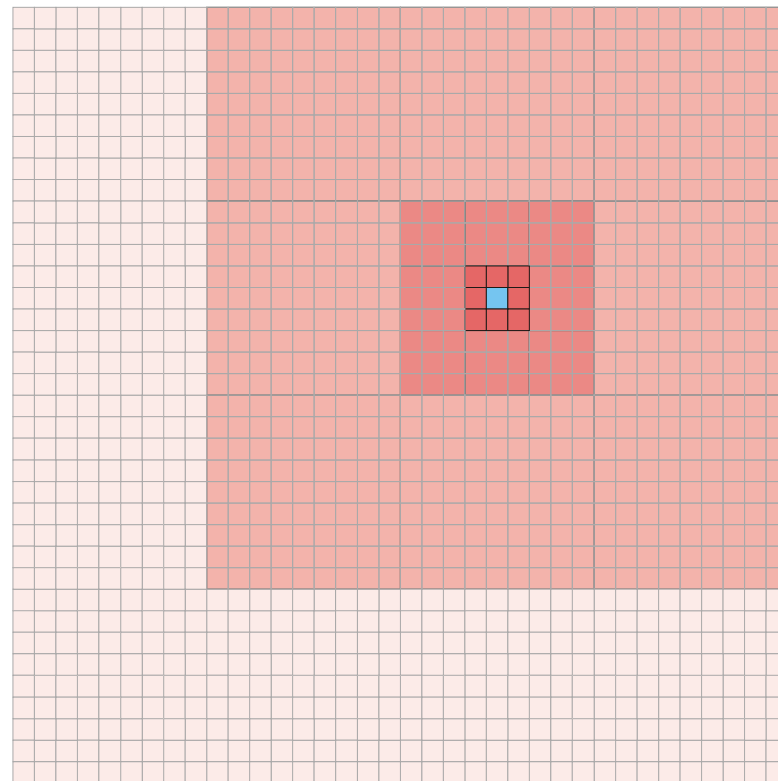
Pentium4/3.06 GHz (1 node)

**Kが大きくなるほどACE-RRにより計算時間短縮**

[1] M. Katouda, M. Kobayashi, H. Nakai, and S. Nagase, *J. Theor. Comput. Chem.* **4**, 139 (2005).

# 高速多重極展開法(FMM)の利用

- MDの講義(安藤先生)に現れたFMMは、電荷分布に対して拡張可能<sup>[1]</sup>
  - ◆ 系全体の空間を多段階のboxに分割
    - 注目するboxからの距離に従ってboxの大きさを変化
  - ◆ 分割されたboxごとに多重極展開を行い、クーロン項評価
  - ◆ 交換項には利用できない
    - が、絶縁体では指数関数的に寄与が減衰するため、打ち切りによる近似が可能<sup>[2]</sup>



[1] C.A. White and M. Head-Gordon, *J. Chem. Phys.* **101**, 6593 (1994).

[2] E. Schwegler and M. Challacombe, *J. Chem. Phys.* **105**, 2726 (1996).

# SCFの構成要素(2): 一般化固有値問題

## ■ SCF方程式: $\mathbf{FC} = \mathbf{SC}\epsilon$ (一般化固有値問題)

◆ **F**: Fock行列 (一般に密行列)

◆ **S**: 重なり行列

■ **S**に小さい固有値がある場合は、線形従属性を回避する必要あり

◆ 全根が必要

■ SCFだけでよければ、占有軌道だけでも良いが、それでも全体の1/5程度は必要

既存の線形代数ライブラリを利用

# 線形従属性を回避したコード

- SCF方程式:  $\mathbf{FC} = \mathbf{SC}\boldsymbol{\varepsilon}$  (一般化固有値問題)
  - ◆ 重なり積分 $\mathbf{S}$ を対角化:  $\mathbf{SU} = \mathbf{U}\mathbf{s}$  ( $\mathbf{s}$ : 対角行列)
  - ◆  $\mathbf{X} = \mathbf{U}\mathbf{s}^{-1/2}$ を用いて直交化
    - 固有値 $\mathbf{s}$ が小さい部分を除去 (線形従属回避)
  - ◆  $\mathbf{F}' = \mathbf{X}^T\mathbf{F}\mathbf{X}$  の通常固有値問題  $\mathbf{F}'\mathbf{C}' = \mathbf{C}'\boldsymbol{\varepsilon}$  に帰着
    - $\mathbf{C} = \mathbf{X}\mathbf{C}'$  で元の固有ベクトルに逆変換

```
1: call calc_Fprime (F, X, Fpr) ; 密行列積 (blas3)
2: call dsyev (Fpr, Cpr, eps) or dsyevd ; (lapack)
3: call dgemm (X, Cpr, C) ; 密行列積 (blas3)
```

行列積部分は簡単にMPI並列化可能

# SCFの構成要素(3): 密度汎関数の積分

## ■ Hartree-Fock法

◆ エネルギー:  $E = \text{Tr} [\mathbf{D}(\mathbf{F} + \mathbf{H}^{\text{core}})]$

◆ ハミルトニアン(=フック行列):  $\mathbf{F} = \mathbf{H}^{\text{core}} + 2\mathbf{J} - \mathbf{K}$

■  $J_{\mu\nu} = \sum_{\lambda\sigma} D_{\lambda\sigma} \Gamma_{\mu\nu,\sigma\lambda}, K_{\mu\nu} = \sum_{\lambda\sigma} D_{\lambda\sigma} \Gamma_{\mu\lambda,\nu\sigma}$

## ■ 密度汎関数理論

◆ エネルギー:  $E = \text{Tr} [\mathbf{D}(2\mathbf{H}^{\text{core}} + 2\mathbf{J})] + E_{\text{xc}}[\rho(\mathbf{r})]$

■  $E_{\text{xc}} = \int d\mathbf{r} V_{\text{xc}}[\rho(\mathbf{r}), \nabla\rho(\mathbf{r}), \dots]$

◆ ハミルトニアン:  $\mathbf{H}^{\text{KS}} = \mathbf{H}^{\text{core}} + 2\mathbf{J} + \mathbf{V}_{\text{xc}}$

■  $(V_{\text{xc}})_{\mu\nu} = \int d\mathbf{r} \phi_{\mu}(\mathbf{r}) V_{\text{xc}} \phi_{\nu}(\mathbf{r})$

■  $V_{\text{xc}} = \frac{\delta E_{\text{xc}}}{\delta\rho}$

数値積分が必要

# 密度汎関数の空間積分

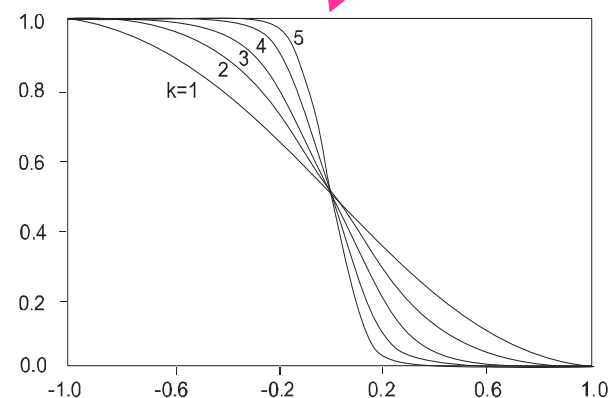
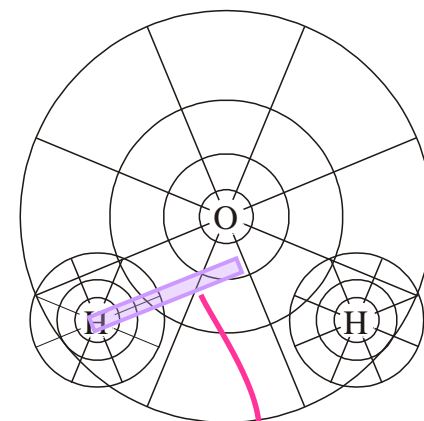
- 原子中心の極座標グリッドを用いた求積

$$\int d\mathbf{r} V_{xc}(\mathbf{r}) \approx \sum_A \sum_g^{\text{atom grid}} \omega_g p_A(\mathbf{r}_g) V_{xc}(\mathbf{r}_g)$$

- ◆  $\omega_g$ : グリッドの重み
- ◆ グリッドの重なりによる多重カウントを回避するため、分割関数  $p$  を導入<sup>[1]</sup>

$$\sum_A^{\text{atom}} p_A(\mathbf{r}_g) = 1$$

- ◆ 計算量:  $O(N_A)$  [ $N_A$ : 原子数]
- ◆ 並列化は容易



[1] A.D. Becke, *J. Chem. Phys.* **88**, 2547 (1988).

# SCFの構成要素(4): DIIS

## ■ Direct inversion in the iterative subspace (DIIS)

◆ 必須の要素ではないが、繰り返し回数の削減に有効  
(HF計算時間  $\propto$  繰り返し回数 なので重要)

◆ Fock行列を過去数回分の線形結合として予測

$$\mathbf{F}^{n,\text{DIIS}} = \sum_{i=1}^n x_i \mathbf{F}^i \quad \left( \sum_{i=1}^n x_i = 1 \right)$$

◆ 予測に用いる条件: 誤差ベクトル  $\mathbf{e}$   $\Delta \mathbf{e} = \sum_{i=1}^n x_i \mathbf{e}^i \rightarrow \mathbf{0}$

■ (1)  $\mathbf{e}^i = \mathbf{F}^i - \mathbf{F}^{i-1}$  (Anderson mixingと一致)

■ (2)  $\mathbf{e}^i = \mathbf{FDS} - \mathbf{SDF}$

◆  $B_{ij} = \mathbf{e}^{i\top} \cdot \mathbf{e}^j$  として 
$$\begin{pmatrix} 0 & -1 & -1 & \cdots & -1 \\ -1 & B_{11} & B_{12} & \cdots & B_{1n} \\ \vdots & & & \ddots & \vdots \\ -1 & B_{n1} & B_{n2} & \cdots & B_{nn} \end{pmatrix} \begin{pmatrix} -\lambda \\ x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$
 を解く (lapack)



# Post HF計算: MP2法の構成要素

## ■ もっとも簡単なpost HF計算

$$\Delta E_{\text{MP2}} = \sum_{i,j}^{\text{occ}} \sum_{a,b}^{\text{vir}} \frac{(ia | jb) [2(ia | jb) - (ib | ja)]}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b}$$

$$(ai | bj) = \sum_{\mu, \nu, \lambda, \sigma} C_{\mu a} C_{\nu i} C_{\lambda b} C_{\sigma j} \Gamma_{\mu \nu, \lambda \sigma}$$

### ◆ ボトルネック: 分子積分 $(ai | bj)$ の計算 [積分変換]

■ 演算量: 段階的に積和すれば $O(N^5)$

■ メモリ量: すべてを一度に実行するには $O(N^4)$ 必要

◆  $\mu, \nu$ : 原子軌道(AO)

◆  $i, j$ : 占有軌道(occ)

◆  $a, b$ : 仮想(非占有)軌道(vir)

軌道の数:

AO > vir > occ

# MP2計算手続きの例: GAMESSの場合

## ■ MP2計算コードの実装が4種類存在

◆ 計算量はすべて $O(N^5)$

◆ GAMESSの並列化はmessage passingのみ

$N$ : 原子軌道数

$O$ : 占有軌道数

$v$ : 仮想軌道数

コード	ローカル メモリ	グローバル メモリ	スクラッチ ディスク
SERIAL	$O(N^3)$	-	-
DDI	$O(N^2)$	$O(N^4)$	-
IMS	$O(NO^2)$	-	$O(NVO^2)$
RIMP2	(近似MP2計算)		

SERIAL, DDI, IMSをOpenMP化

# OpenMP版SERIAL MP2の擬似コード

MPI parallel

OpenMP parallel

Hybrid parallel

```
1: DO IBATCH=1, NBATCH
2:   DO  $\mu$ SH=1, NSHELL
3:     DO  $\nu$ SH=1,  $\mu$ SH
4:       DO  $\lambda$ SH=1,  $\nu$ SH
5:         IF ([ $\mu$ SH, $\nu$ SH, $\lambda$ SH].EQ.MYTASK) THEN ; (OpenMP)
6:           DO  $\sigma$ SH=1,  $\lambda$ SH
7:             calculate gamma [ $\mu$ SH, $\nu$ SH, $\lambda$ SH, $\sigma$ SH]
8:             1st(occ) transform. [ $\sigma \rightarrow I$ ] for my I -> ローカルメモリ
9:           END DO
10:        END IF
11:      END DO
12:    END DO
13:  END DO
14:  2nd&3rd&4th transform. for my I ; (BLAS3)
15:  evaluate energy for my I ; (OpenMP)
16: END DO
17: accumulate energy over MPI processes
```

Atomic演算

# MP2アルゴリズムにおけるAtomic演算

■  $\mu \geq \nu \geq \lambda \geq \sigma$ を満たすAOのセット( $\mu, \nu, \lambda, \sigma$ )に対して

◆  $\Gamma_{\mu\nu,\lambda\sigma}, \Gamma_{\mu\lambda,\nu\sigma}, \Gamma_{\mu\sigma,\nu\lambda}$ に関する演算を実行 μ, ν, λ: スレッド並列

◆  $(\mu\nu|\lambda i) += C_{\sigma i} \Gamma_{\mu\nu,\lambda\sigma}$  (足しこみ演算)

■  $\Gamma_{\mu\nu,\lambda\sigma} = \Gamma_{\mu\nu,\sigma\lambda} = \Gamma_{\lambda\sigma,\mu\nu} = \Gamma_{\lambda\sigma,\nu\mu}$

```
1: (Loop over all (μ, ν, λ, σ) quartet)
2:   Calculate gamma (μν, λσ)
3:   DO I=MyOCCb, MyOCCf
4:     !#OMP ATOMIC
5:     int1 (μν, λ, I) += gamma (μν, λσ) * C (σ, I)
6:     int1 (μν, σ, I) += gamma (μν, λσ) * C (λ, I)
7:     int1 (λσ, μ, I) += gamma (μν, λσ) * C (ν, I)
8:     int1 (λσ, ν, I) += gamma (μν, λσ) * C (μ, I)
9:   END DO
10:   Calculate gamma (μλ, νσ) ...
11:   Calculate gamma (μσ, νλ) ...
12: (End loop)
```

スレッドセーフでない  
(同時メモリアクセスの可能性)

# OpenMP版SERIALコードの改良

```
1: DO IBATCH=1, NBATCH
2:   DO  $\mu$ SH=1, NSHELL
3:     DO  $\nu$ SH=1,  $\mu$ SH
4:       IF ([ $\mu$ SH, $\nu$ SH].EQ.MYTASK) THEN ; (OpenMP)
5:         DO  $\lambda$ SH=1, NSHELL
6:           DO  $\sigma$ SH=1,  $\lambda$ SH
7:             calculate gamma [ $\mu$ SH, $\nu$ SH, $\lambda$ SH, $\sigma$ SH]
8:             1st(occ) transform. [ $\sigma \rightarrow i$ ] for my I -> ローカルメモリ
9:           END DO
10:        END IF
11:      END DO
12:    END DO
13:  END DO
14:  2nd&3rd&4th transform. for my I ; (BLAS3)
15:  evaluate energy for my I ; (OpenMP)
16: END DO
17: accumulate energy over MPI processes
```

DOとIFを交換  
ループ長を延長

MPI parallel  
OpenMP parallel  
Hybrid parallel

Atomic演算なし!

# OpenMP版IMS MP2の擬似コード

```
1: DO  $\mu$ SH=1, NSHELL
2:   IF ( $\mu$ SH.EQ.MYTASK) THEN ;(MPI parallel)
3:     DO  $\lambda$ SH=1, NSHELL
4:       DO  $\sigma$ SH=1,  $\lambda$ SH
5:         !$OMP DO
6:         DO  $\nu$ SH=1, NSHELL
7:           calculate gamma [ $\mu$ SH,  $\nu$ SH,  $\lambda$ SH,  $\sigma$ SH]
8:         END DO
9:         1st&2nd(occ) transform. [ $\nu\sigma \rightarrow IJ$ ] ;(OpenMP)
10:      END DO
11:    END DO
12:    3rd(vir) transform. [ $\lambda \rightarrow B$ ] ;(BLAS3) -> ディスク
13:  END IF
14: END DO
15: 4th(vir) transform. [ $\mu \rightarrow A$ ] for my IJ ;(BLAS3) <- ディスク
16: evaluate energy for my IJ ;(BLAS3)
17: accumulate energy over MPI processes
```

MPI parallel  
OpenMP parallel  
Hybrid parallel

# OpenMP版DDI MP2の擬似コード

```
1: DO  $\mu$ SH=1, NSHELL
2:   DO  $\nu$ SH=1,  $\mu$ SH
3:     IF ( $\mu\nu$ SH.EQ.MYTASK) THEN ;(DDI parallel)
4:       !$OMP DO
5:       DO  $\lambda$ SH=1, NSHELL
6:         DO  $\sigma$ SH=1, NSHELL
7:           calculate gamma [ $\mu$ SH,  $\nu$ SH,  $\lambda$ SH,  $\sigma$ SH]
8:           1st(occ) transform. [ $\sigma \rightarrow J$ ]
9:         END DO
10:      END DO
11:      2nd(vir)&3rd(occ) transform. [ $\nu\lambda \rightarrow IA$ ] ;(OpenMP)
12:    END IF
13:  END DO
14: END DO
15: 4th(vir) transform. for my IJ ;(BLAS3) ← グローバルメモリ
16: evaluate energy for my IJ ;(OpenMP)
17: accumulate energy over DDI processes
```

ループ長を延長して  
atomic演算回避

MPI parallel  
OpenMP parallel  
Hybrid parallel

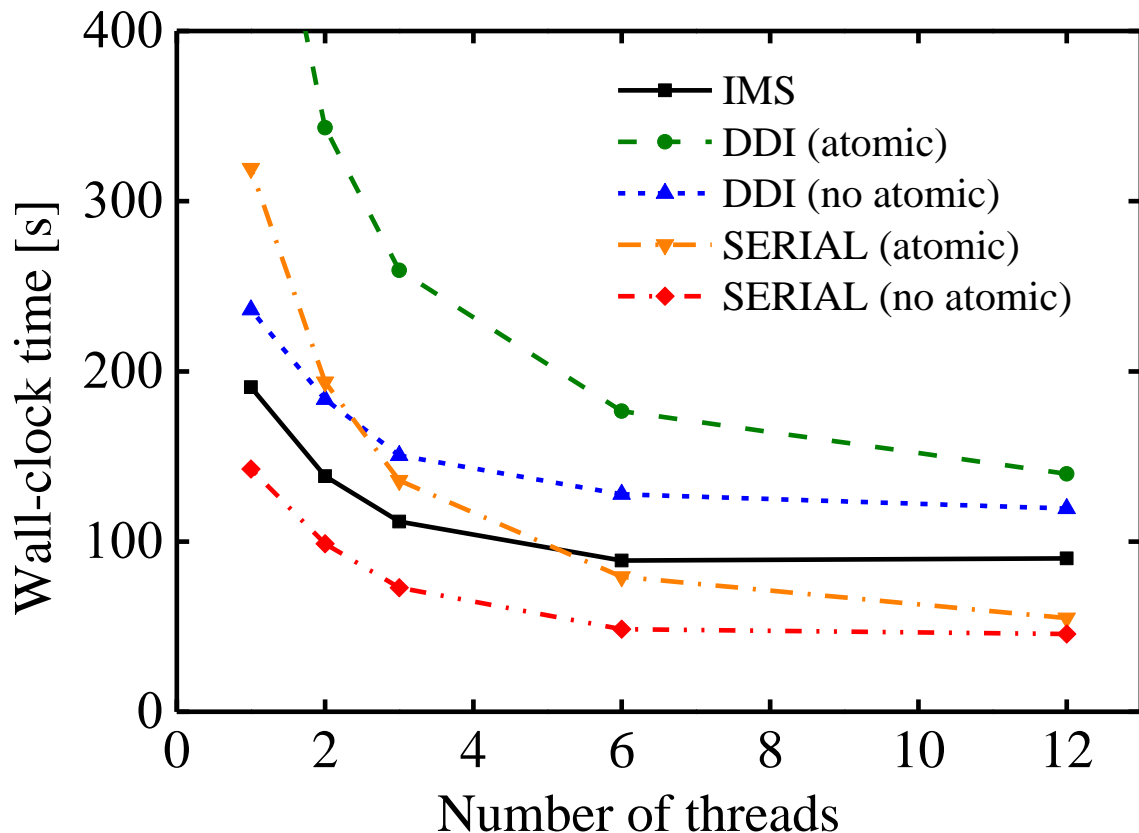
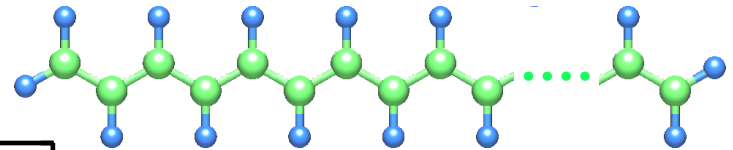
-> グローバルメモリ

# OpenMP並列MP2の効率

## ■ 計算時間のスレッド数依存性

◆ 計算対象: ポリエン $C_{30}H_{32}$  (DC-MP2/6-31G)

◆ Intel Xeon X5650 x 2 (1 node)



- ✓ 12スレッド利用時  
**SERIAL** < **IMS** < **DDI**  
(MPI並列効率は考慮なし)
- ✓ **Atomic演算を回避**することにより、大幅にパフォーマンス改善



# Post HF計算: CI法の構成要素

■ CI波動関数: 
$$\Psi_{\text{CI}} = t_0 \Phi_0 + \sum_i^{\text{occ}} \sum_a^{\text{vir}} t_i^a \Phi_i^a + \sum_{i,j}^{\text{occ}} \sum_{a,b}^{\text{vir}} t_{ij}^{ab} \Phi_{ij}^{ab} + \dots$$

◆  $\Phi_0$ : HF配置関数,  $\Phi_i^a: i \rightarrow a$ の励起配置関数, ...

◆ 変分的に $t$ を決定:  $\mathbf{Ht} = E_0 \mathbf{t}$

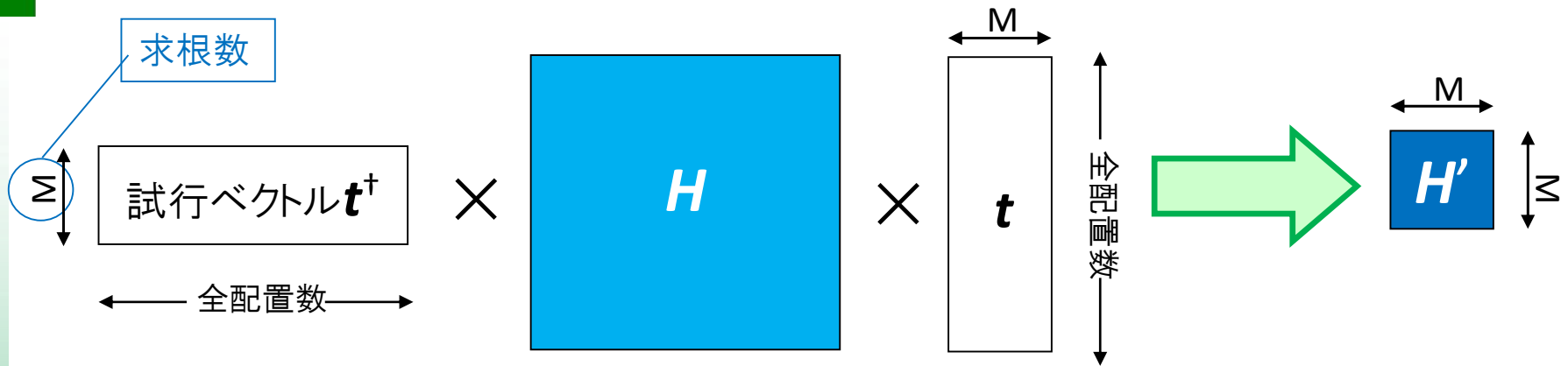
$$\mathbf{H} = \begin{pmatrix} \langle \Phi_0 | \hat{H} | \Phi_0 \rangle & 0 & \langle \Phi_0 | \hat{H} | \Phi_{kl}^{cd} \rangle & \dots \\ 0 & \langle \Phi_i^a | \hat{H} | \Phi_k^c \rangle & \langle \Phi_i^a | \hat{H} | \Phi_{kl}^{cd} \rangle & \\ \langle \Phi_{ij}^{ab} | \hat{H} | \Phi_0 \rangle & \langle \Phi_{ij}^{ab} | \hat{H} | \Phi_k^c \rangle & \langle \Phi_{ij}^{ab} | \hat{H} | \Phi_{kl}^{cd} \rangle & \\ \vdots & & & \ddots \end{pmatrix}$$

■ 大規模対角化が必要

■ 必要な根の数は数本

# 疎行列の対角化: Davidson法

## ■ 試行ベクトルを用いた繰り返し計算

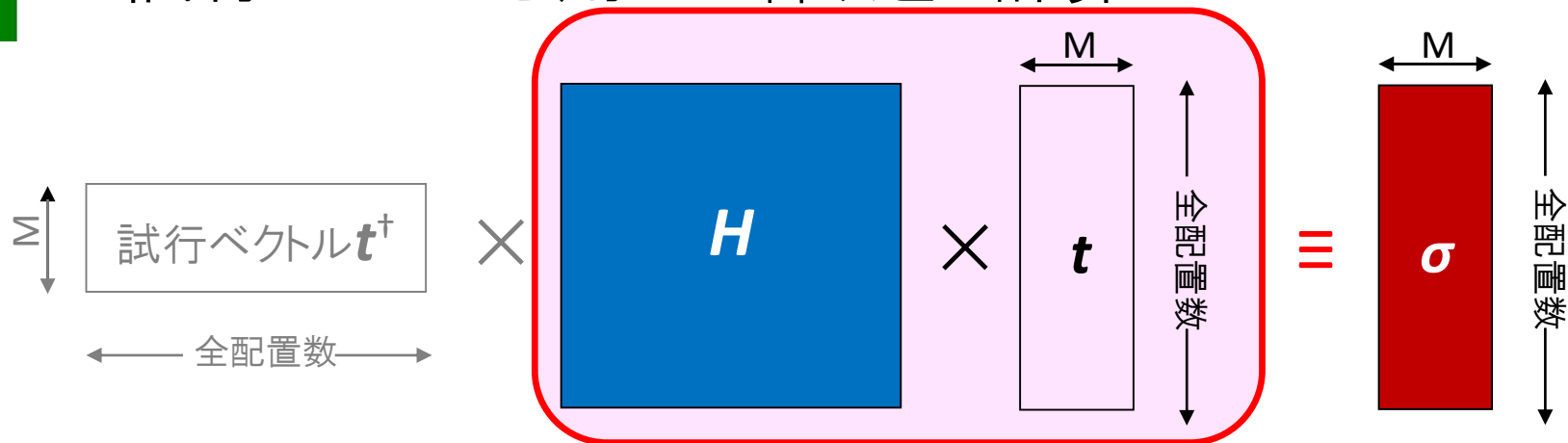


- ◆  $H$ をメモリに載せて対角化することを回避
- ◆  $H'$ の対角化  $\rightarrow t$ のアップデート
- ◆ 基底状態に対しては  $t_0 = 1$  (その他は0)が良い出発点

$H$ をスクラッチに載せる必要あり？

# CI法: Knowles-Handyアルゴリズム

## ■ 試行ベクトルを用いた繰り返し計算



◆  $H$ の行列要素: 分子積分( $pq|rs$ )の和

- ( $pq|rs$ )はスクラッチに保存[ $O(N^4)$ ]
- $H$ を保存せずに  $Ht \equiv \sigma$ を構築可能

要求メモリ量:

配置数  $\times$  配置数  $\rightarrow$  配置数  $\times M$

- ( $pq|rs$ )に対する並列化、配置に対する並列化が可能

# 本日のまとめ(1)

- 量子化学計算は手法に応じて様々な計算技術が利用される
  - ◆ HF, DFT: 2電子積分計算 [ $O(N^2-4)$ ] と密行列対角化 [ $O(N^3)$ ]
    - Gauss型基底の2電子積分計算は特に特殊な演算で、さまざまな計算アルゴリズムが開発
    - 密行列対角化は(ほぼ)全根が必要
  - ◆ MP2, CC: 積分変換 [ $O(N^5)$ ]、テンソル縮約
    - 積分変換は2電子積分計算+縮約を含む
    - DGEMM等を用いた実装が可能
  - ◆ CI: 積分変換、大規模疎行列対角化
    - Davidson法のように全根を求めない疎行列対角化

# 本日のまとめ(2)

## ■ 量子化学計算の効率化における注意点

### ◆ 演算量の削減

#### ■ 対称性の高いテンソルの再利用

##### ◆ 高並列化を考える場合には副作用も

#### ■ FLOP数による見積もりを鵜呑みにする必要はないが有用

#### ■ SIMD化、キャッシュミスの削減も重要

### ◆ 繰り返し計算の収束性の向上

#### ■ 計算時間は(1サイクルの時間) × (繰り返し回数)

### ◆ 基礎的な演算(行列積、密行列対角化等)はライブラリコール

### ◆ 並列化

#### ■ OpenMPではatomic演算の回避が性能に重要

#### ■ データ分散も考えたMPI並列化