ABINIT-MP

FMOプログラムABINIT-MPの高速化と超大規模系への対応

望月祐志[1,2] (課題責任者)

中野達也[3]、坂倉耕太[4]、渡邊啓正[5]、秋澤和輝[1]

片桐孝洋[6]、大島聡史[6]、山梨祥平[6]、森下誠[6]

1:立教大、2:東大生研、3:国立衛研、4:FOCUS

5:HPCシステムズ、6:名大

謝辞:SS研「A64FXシステムアプリ性能検討WG」の活動とも連携しています

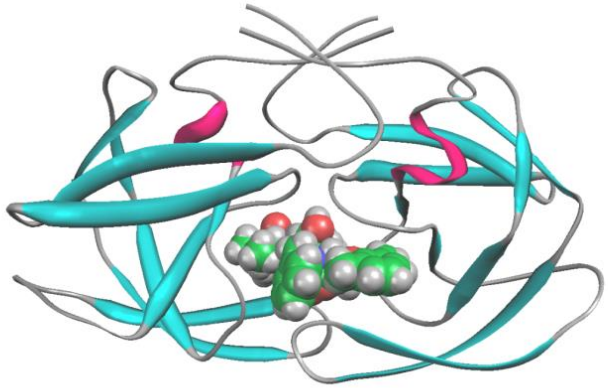
フラグメント分子軌道(FMO)法

◇巨大分子系

生体高分子や凝集系では一般的

⇒ タンパク質、DNA （水和状態）

数千～数万原子、数千～数十万軌道



【HIVプロテアーゼとロピナビル】

◇分割&統合系のアプローチの一つ

北浦らが20年前に2体展開で提案

⇒ フラグメントとその対で系のエネルギーを評価 （FMO2）

⇒ 環境静電ポテンシャル（ESP）、直接結合切断（BDA）

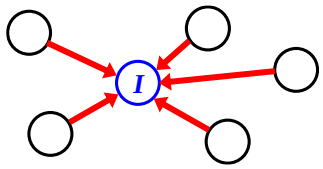
⇒ 階層的な並列処理（フラグメントリスト&内部処理）

⇒ 電子相関の導入、分子構造の最適化も可能

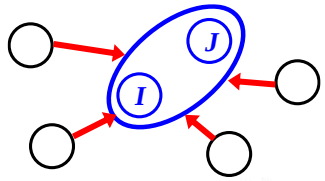
フラグメント間の相互作用エネルギー（IFIE）

⇒ 計算対象の解析ツール

⇒ 創薬設計には好適



モノマー （アミノ酸単位など）



ダイマー

FMO計算のためのプログラム

◇GAMESS-US [米国Gordonグループ]; Fedorov、北浦、中田ら

GAUSSIANに抗し得る有力なフリーソフト、世界規模 (Fortran)

⇒ 様々な計算機能をFMO化、GDDI並列

◇ABINIT-MP; 望月、中野ら

実用機能は十分、東大系PJ・CREST-PJなどで開発 (Fortran)

⇒ IOレス、MPI/OpenMP並列、4体展開FMO、専用GUI

◇PAICS; 石川

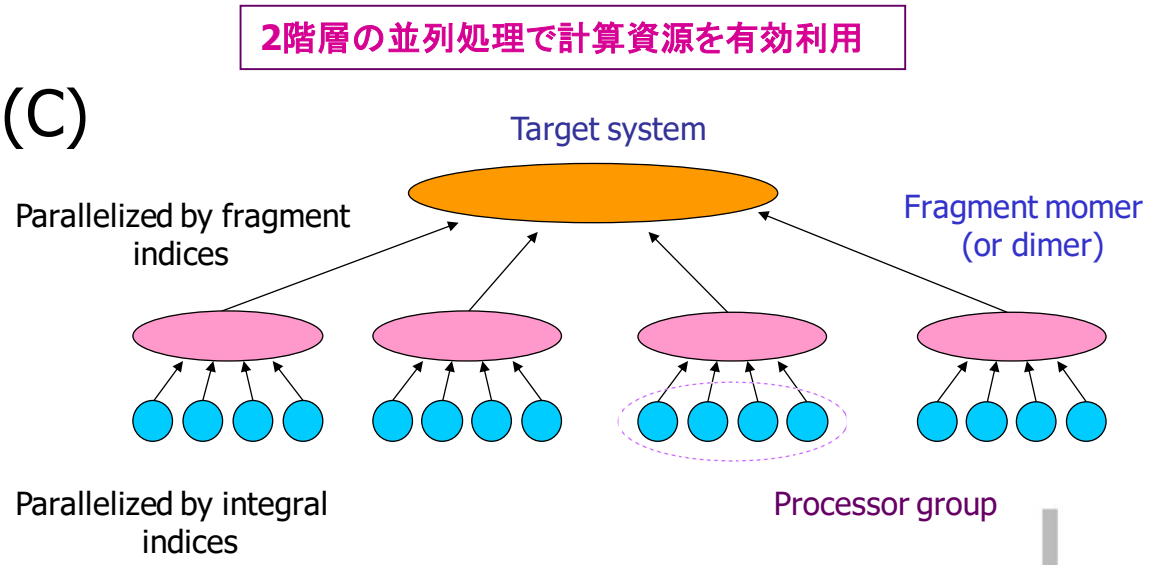
FMO-MP2(RI)に特化 (C)

⇒ MPI並列

◇OpenFMO; 稲富ら

FMO-HFのみ (C)

⇒ 超並列指向



Ref.; “The ABINIT-MP program” Y. Mochizuki et al. in “*Recent Advances of the Fragment Molecular Method*” (Springer, 2021).

ABINIT-MPの主な機能（オープンシリーズとして整備中）

- エネルギー
  - → FMO4：HF, MP2
  - → FMO2：HF～CCSD(T), LRD
  - → FMO2: CIS/CIS(D)
- エネルギー微分
  - → FMO4：HF, MP2
  - → FMO2：MP2構造最適化, MD
- その他機能
  - → SCIFIE, PB, sp2-BDA, α(ω)
  - → 電子密度生成, CAFI, FILM
- 並列化環境（PC~スパコン）
  - → MPI, OpenMP/MPI混成
  - → 最深部はBLAS処理

http://www.cenav.org/abinitmpopen1/

Open Ver. 1 （ポスト「京」のPJで整備）
- Rev. 5 (2016年12月)
- Rev. 10 (2018年2月)
- Rev. 15 (2019年3月)
- Rev. 22 (2020年6月)；当面は併存

Open Ver. 2 （「富岳」の時代に移行）
- Rev. 5 (2021年7月予定)

Refs.; T. Nakano et al., *Chem. Phys. Lett.* **318** (2000) 684. & T. Nakano et al., *Chem. Phys. Lett.* **351** (2002) 475.

基本のHF計算

$$\mathbf{F}^x\mathbf{C}^x=\mathbf{S}^x\mathbf{C}^x\boldsymbol{\varepsilon}^x \qquad \mathbf{F}^x=\mathbf{H}^x+\mathbf{G}^x \quad \Leftrightarrow \text{ HFの一般化固有値問題 （要反復計算）}$$

$$H^{\ x}_{\mu\nu}=H^{core\ x}_{\mu\nu}+V^x_{\mu\nu}+\sum_k B_k\langle\mu|\theta_k\rangle\langle\theta_k|\nu\rangle \qquad V^x_{\mu\nu}=\sum_{K\neq x}\left(u^K_{\mu\nu}+v^K_{\mu\nu}\right) \quad \Leftrightarrow \text{ 1電子部分の修飾}$$

$$u^K_{\mu\nu}=\sum_{A\in K}\left\langle\mu\left|\left(-Z_A/|\mathbf{r}-\mathbf{A}|\right)\right|\nu\right\rangle \qquad v^K_{\mu\nu}=\sum_{\lambda\sigma\in K}P^K_{\lambda\sigma}(\mu\nu|\lambda\sigma) \quad \Leftrightarrow \text{ 環境静電ポテンシャル（ESP）}$$

$$P_{\mu\nu}=2\sum_{i=1}^{occ}C^*_{\mu i}C_{\nu i} \qquad G^x_{\mu\nu}=\sum_{\lambda\sigma\in x}P^x_{\lambda\sigma}\left[(\mu\nu|\lambda\sigma)-\frac{1}{2}(\mu\sigma|\lambda\nu)\right] \quad \Leftrightarrow \text{ 2電子部分 （N}^4\text{）（並列処理）}$$

高速化のための幾つかの工夫

$$V^L_{\mu\nu}\cong\sum_{\lambda\in L}(\mathbf{P}^L\mathbf{S}^L)_{\lambda\lambda}(\mu\nu|\lambda\lambda) \quad \text{for } R_{\min}(X,L)\geq L_{\text{aoc}} \quad \Leftrightarrow \text{ ESP-AOC近似 （実用精度高し）}$$

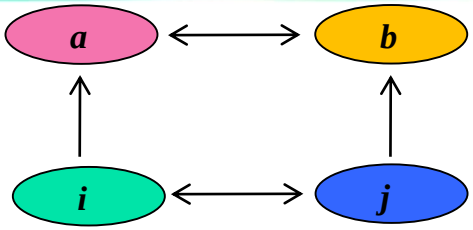
$$V^L_{\mu\nu}\cong\sum_{A\in L}\left\langle\mu\left|\left(Q_A/|\mathbf{r}-\mathbf{A}|\right)\right|\nu\right\rangle \quad \text{for } R_{\min}(X,L)\geq L_{ptc} \qquad Q_A=\sum_{\lambda\in A}(\mathbf{P}^L\mathbf{S}^L)_{\lambda\lambda} \quad \Leftrightarrow \text{ ESP-PTC近似 （速い）}$$

$$E'_{IJ}\cong E'_I+E'_J+\mathrm{Tr}\left(\mathbf{P}^I\mathbf{u}^J\right)+\mathrm{Tr}\left(\mathbf{P}^J\mathbf{u}^I\right)+\sum_{\mu\nu\in I}\sum_{\lambda\sigma\in J}\mathbf{P}^I_{\mu\nu}\mathbf{P}^J_{\lambda\sigma}(\mu\nu|\lambda\sigma) \quad \Leftrightarrow \text{ Dimer-ES近似 （HF計算無）}$$

種々の工夫により、FMO2の計算コストのシステムサイズ依存性は2乗より低い

Refs.; Y. Mochizuki et al., *Chem. Phys. Lett.* **396** (2004) 473. & Y. Mochizuki et al., *Theor. Chem. Acc.* **112** (2004) 442.
Y. Mochizuki et al. *Chem. Phys. Lett.* **457** (2008) 396.

MP2エネルギー計算（ABINIT-MPの実装）



⇔

（電子対の相互作用）

$$E^{MP2} = \sum_{ijab} \frac{(ia\,|\,jb)\big(2(ia\,|\,jb) - (ib\,|\,ja)\big)}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b}$$

$$(ia\,|\,jb) = \sum_{\mu\nu\lambda\sigma} C_{\mu i} C_{\nu a} C_{\lambda j} C_{\sigma b} (\mu\nu\,|\,\lambda\sigma)$$

（基底関数の添字から変換）

Loop over *ij*-batch ! size depending on available memory
  Loop over σ ! to be parallelized
    Loop over λ
      Preparing (μν|λσ) ! for canonical *μν*-pair
      Forming (*i*ν|λσ) ! DGEMM, fixed λσ, running over μ
      Forming (*ia*|λσ) ! DGEMM, fixed λσ, running over ν
      Forming (*ia*|*j*σ) ! DGEMM, fixed σ, direct-product for fixed λ
    End of loop over λ
    Forming (*ia*|*jb*) ! DGEMM, direct-product for fixed σ
  End of loop over σ ! all-reduce operation as barrier
  Calculate partial MP2 energy with respect to *ij*-batch
End of loop over *ij*-batch

$$(i\nu\,|\,\lambda\sigma) = \sum_{\mu} C_{\mu i} (\mu\nu\,|\,\lambda\sigma)$$

$$(ia\,|\,\lambda\sigma) = \sum_{\nu} C_{\nu a} (i\nu\,|\,\lambda\sigma)$$

$$(ia\,|\,j\sigma) = \sum_{\lambda} C_{\lambda j} (ia\,|\,\lambda\sigma)$$

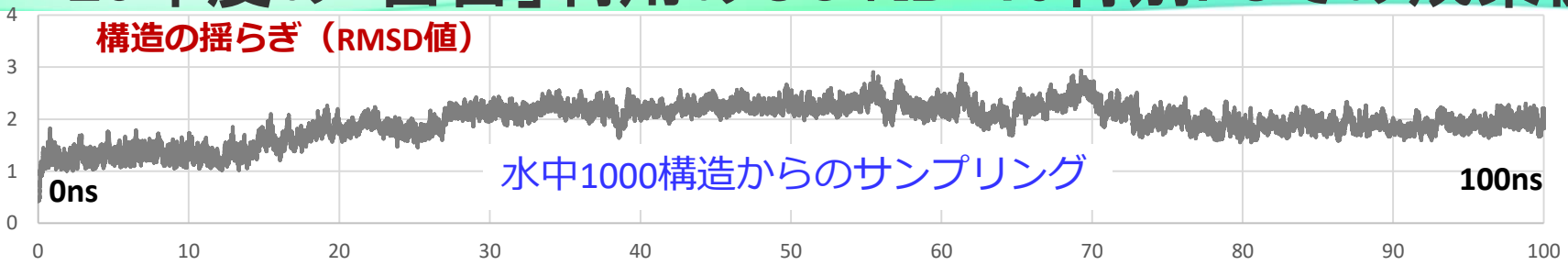
$$(ia\,|\,jb) = \sum_{\sigma} C_{\sigma b} (ia\,|\,j\sigma)$$

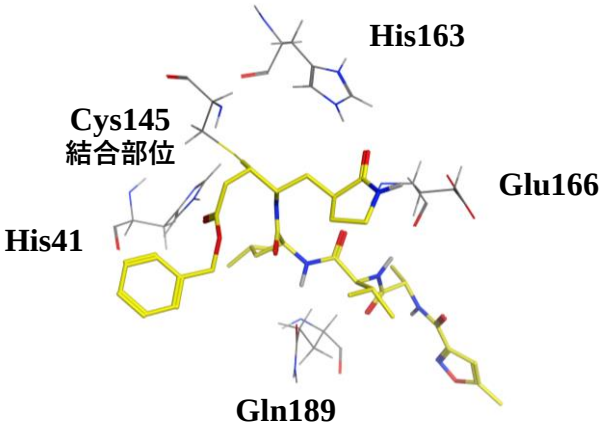
（DGEMMは行列積ルーチン）

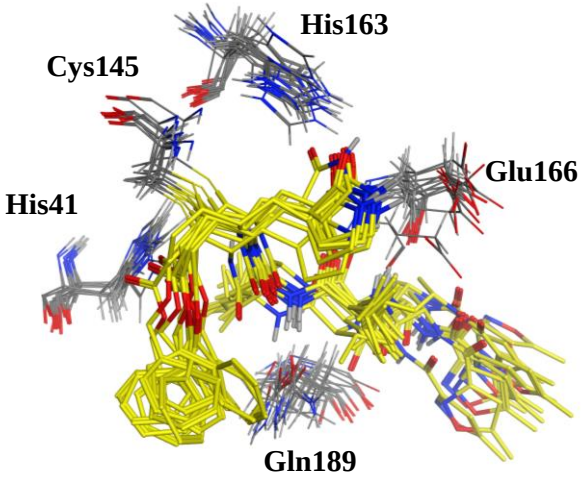
DEGMMを使わず、閾値判定でDAXPYを使うことも可能（1段目、3段目）

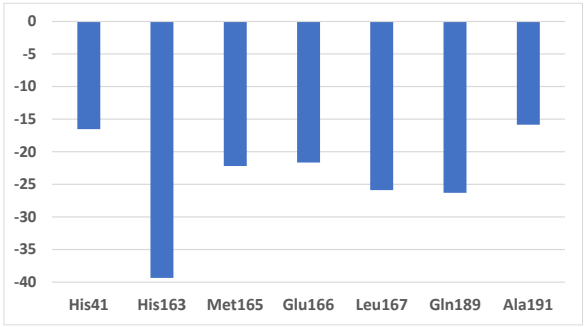
Refs.; R. Hatada et al., *J. Chem. Inform. Model.* **60** (2020) 3593 & *Appl. Phys. Express* **14** (2021) 027003.
S. Tanaka et al., *J. Phys. Chem. B* **125** (2021) 6501.

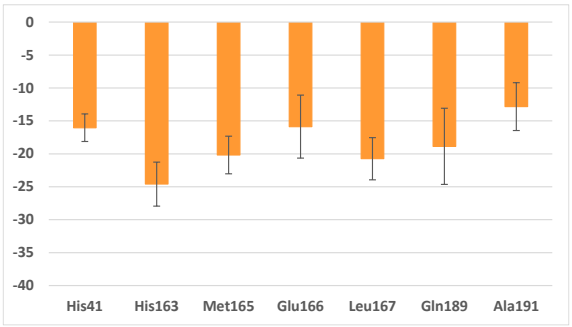
20年度の「富岳」利用のCOVID-19特別PJでの成果例











揺らぎ無しIFIE（結晶構造）

揺らぎありIFIE（生体環境）

「富岳」の威力：0.6時間/1構造、100並列で1000構造が**総計5時間**

21年度のjh210036-NAHでの実施内容と現状(6月)

■ABINIT-MP関係(「不老」Type Iサブシステム)

・超大規模系対応： 1万超フラグメントのMP2ジョブの実現

→ 「不要な配列」の削除 (BioStation Viewerとの互換性は保留)

→ 「不要なプリント」の抑制

⇒ 1.1万フラグメントのMP2/cc-pVDZ計算を達成 (1ラック)

⇒ MP2は全段DGEMM処理がベター (10%程度速い)

・高速化： 2電子積分生成が全体の1/2のコスト、HF処理が1/4

→ 積分ルーチンのSIMD化 (小原式アルゴリズムでタイプが多数)

→ Fock行列の加算処理の改良

⇒ 高速化は積分生成で17%、Fock行列構築で30%の速度向上

■機械学習関係(「不老」Type IIサブシステム)

・画像処理データの準備

→ 可視化IFIEデータの深層学習

→ タンパク質の基本構造の判定 (αヘリックス、βシートの存在)

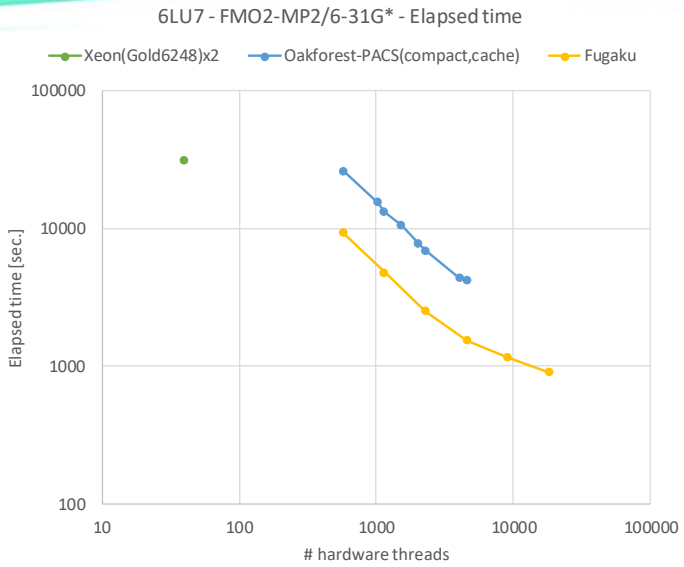
⇒ TensorFlowを使った先行事例の「再訪」に向けて準備中

【「不老」Type I】

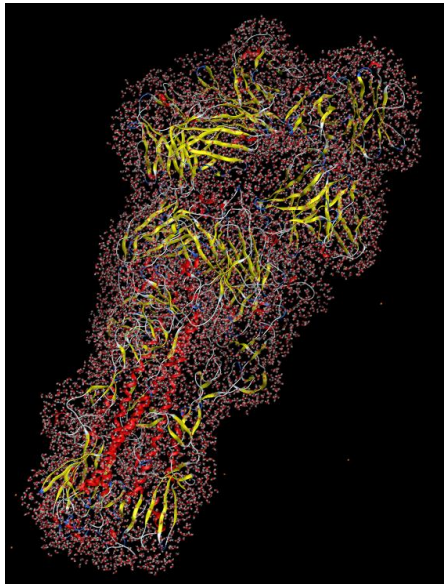
超大規模系への対応（Ver.2 Rev.2:暫定版）

- 「不老」の1ラック、1プロセス/1ノードで実行
- インフルHA+Fab抗体×2（PDB id: 1KEN）の水和モデル
- Dimer-ESの遠領域（Ldimer>5）は多重極展開で近似
- フラグメント総数は11307、MP2/cc-pVDZジョブ、9.2時間
- 基本スケーリングは良好、多ラック使用で加速の見込み

```
 ====================
    ## TIME PROFILE
 ====================

        Elapsed time: Monomer SCF      =          14546.6 seconds
        Elapsed time: Monomer MP2      =             32.5 seconds
        Elapsed time: Monomer (Total)  =          14741.5 seconds
        Elapsed time: Dimer ES         =           4021.8 seconds
        Elapsed time: Dimer SCF        =           7215.9 seconds
        Elapsed time: Dimer MP2        =           2492.4 seconds
        Elapsed time: Dimer (Total)    =          18240.6 seconds
        Elapsed time: FMO (Total)      =          32982.1 seconds

## Time profile

Number of cores (total)    =    384
Number of cores (fragment) =      1

THREADS (FRAGMENT)         =     48

Total time =          33120.9 seconds
```



スケーリング性能の例

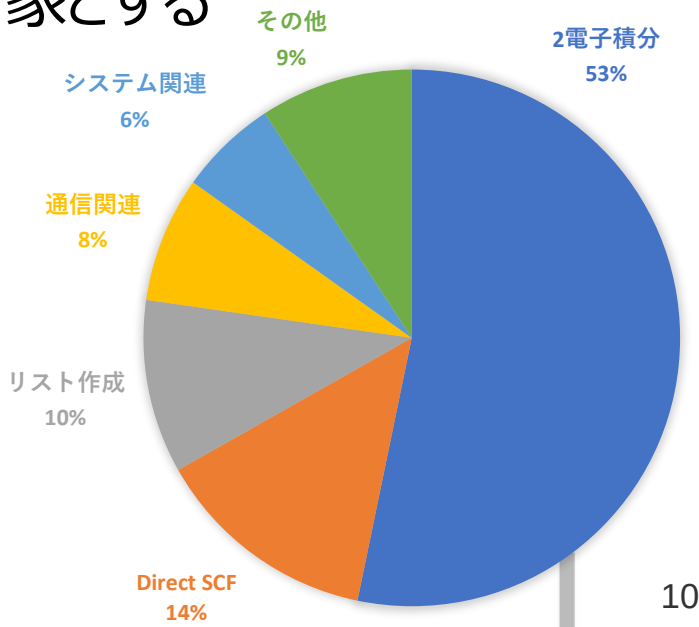


1KEN水和モデル

FMO-MP2計算のコスト分析（Ver.1 Rev.22）

・Ala_9GlyのFMO-MP2/6-31G*のテストジョブ
・12スレッド8プロセス (2ノード実行：FX1000)

- プログラム全体のコスト分布
  - 基本プロファイラによるプロセス0番、スレッド0番のコスト分布
  - 2電子積分処理が全体の約半分を占める
    ただし、81種の処理の総和であるため、
    1種あたりのコストは1%前後と非常に小さい
  - 通信に関連したコストは8%程度と小さい
  - 性能改善に向けたソース分析は以下を対象とする
    - 2電子積分
    - Direct SCF
    - リスト作成



2電子積分：81種のサブルーチン(sub_*)のコスト総和
Direct SCF：サブルーチンdirect_scf_gmatのコスト
リスト作成：3種のサブルーチン(get_tei_rs_fix, get_tei_pq_fix, get_ixijcs_to_proc_pqfix)のコスト総和
通信関連：通信に関連した処理(putofu_*, opal_*, mca_*)のコスト総和
システム関連：ライブラリやOSなどに関連した処理のコスト総和
その他：上記以外の処理の総和

改善の方向性(積分の生成)

- 改良指針(井上G@富士通の助言)
  - OCL指示詞の導入によるSIMD化の促進、一部スカラ変数化も必要
  - コンパイラオプションの変更

- リファレンス
  - オリジナルコード、Ala_9GlyのMP2ジョブ
  - コンパイラオプション:-O3 -Knosimd -Koptmsg=2 –V
  - 6-31G*//cc-pVZ: 153.0s/134.5s//337.4s/306.1s (MP2;AXPY/GEMM)

- 手動での最適化と結果
  - オリジナルコード+ssss, psss, spss, ssps, sssp, ppss, psps, pssp, spps, spsp, sspp, dsss, sdss, ssds, sssd (スカラ変数化)
  - OCL指示詞の追加
  - コンパイラオプション:-O3 -Knosimd –Kocl
  - 6-31G*//cc-pVZ:142.5s/124.5s(7.4%)//294.4s/254.0s(17.0%)

(cc-pVDZの場合、オリジナルコードと比較して全体で17.0%の高速化)
(kfastオプションの指定によってさらに高速化される可能性あり)

改善の方向性(Fock行列の構築)

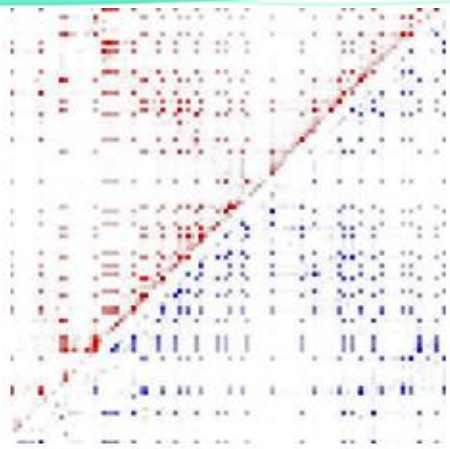
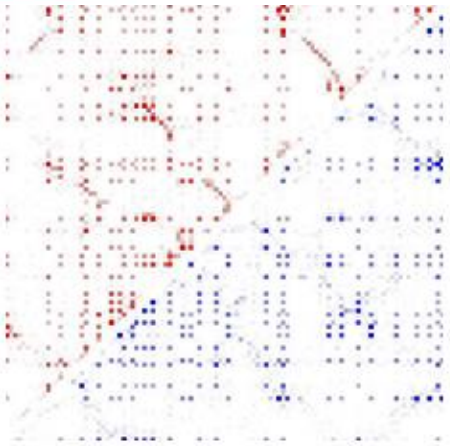
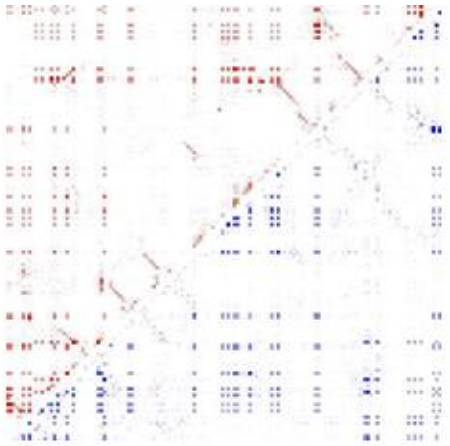
- 改良指針(井上G@富士通の助言)
  - 14個のIF分岐(添字の同値性判断)が最適化を阻害、アクセスも不連続に
- 手動での最適化と結果
  - 基底関数添字の同値性を$(1/2)^n$ (n=1,2,3)で繰り込み
  - IFは積分閾値の篩い落としのみ
  - 系にも拠るが最大で30%の加速

```
do p=ixi1,ixi2
  do q=ixj1,ixj2
    do r=ixk1,ixk2
      do s=ixl1,ixl2
        ix=ix+1
        val = sint(ix)
        if((abs(val) <= tv)) cycle
        fock(q,p)=fock(q,p)+dc(s,r)*val*2.d0! クーロン項
        fock(s,r)=fock(s,r)+dc(q,p)*val*2.d0
        fock(r,p)=fock(r,p)-dc(s,q)*val*0.5d0! 交換項
        fock(s,p)=fock(s,p)-dc(r,q)*val*0.5d0
        fock(r,q)=fock(r,q)-dc(s,p)*val*0.5d0
        fock(s,q)=fock(s,q)-dc(r,p)*val*0.5d0
      end do
    end do
  end do
end do
```

Ref.; S. Saitou et al., *Chem-Bio Inform. J.* 18 (2018) 58. / タンパク質の残基間のIFIEを上三角が安定化、下三角が不安定化でBSVで可視化、当時はPCで行ったため32x32サイズに落として学習・判断させていた。

FMO計算結果の可視化の深層学習の先行例

・IFIEの可視化（IFIE-map)の学習

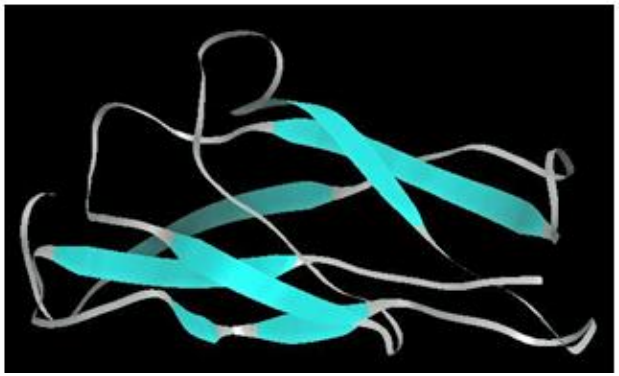




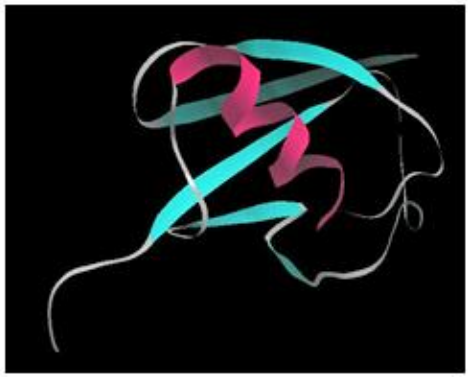
Deep Learning

TensorFlow

構造の特徴を
画像から判定


αヘリックス構造


βシート構造


両方の構造

可視化結果の再解析の準備

・既報のFMO計算結果をpython系ツールで再処理して可視化、ワークフローを確立中
・GPUスパコンの高速処理を前提に、画像をダウンサンプリングしないでそのまま利用

```python
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import os
import glob
import subprocess
import datetime

folder1 =input('csvにするフォルダ')

path1 = './'+folder1+'/*.out'
files1 = glob.glob(path1)

print(path1)
print(files1)


for i in range(len(files1)):
    subprocess.call(["python", "/home/sugiura/abmptools/getifiepieda.py","-ff","1-50","51-100","-i",files1[i]])
```

```python
foldername = input("フォルダ名入力")
path='./'+foldername+'/*-MP2total-ffmatrix.csv'

files = glob.glob(path)
print(files)
mkfoldername = folder1+'picture-'+str(datetime.datetime.now().strftime("%Y%m%d%H%M"))
os.makedirs(mkfoldername,exist_ok=True)

for i in range(len(files)):

    df_ifie_csv = pd.read_csv(files[i], header=0, index_col=0, float_precision="high")

    plt.figure()
    sns.heatmap(df_ifie_csv,vmax=50,vmin=-50,center=0, cmap='bwr_r',cbar=False) #最大値50,最小値-50　カラーバー非表示
    plt.axis('off')
    plt.savefig(mkfoldername+'/seaborn_heatmap_list'+str(i)+'.png')
    plt.close('all')
```

Refs.; R. Hatada et al., *Appl. Phys. Expr.* **14** (2021) 027003. & S. Tanaka et al., *J. Phys. Chem. B* **125** (2021) 6501.

まとめと(21年度の)関連活動

■jh210036-NAH課題@「不老」

- 超大規模系対応 （Type Iサブシステム）
  - ⇒ 1万フラグメントのFMO-MP2/cc-pVDZの完走は1Q期に達成
  - ⇒ 2万フラグメント到達に向けて改良を継続 （配列の追加整理が必要）
  - ⇒ ベンチマーク的な応用計算の実行
- 高速化 （Type Iサブシステム）
  - ⇒ 高速化は積分生成で17%、Fock行列構築で30%の速度向上
  - ⇒ HFでは積分バッファリングの併用を検討 （積分計算の総数を削減）
- 機械学習 （Type IIサブシステム）
  - ⇒ 可視化IFIEデータの深層学習の実施

■hp210026課題@「富岳」（代表者：望月）

- ウイルスタンパク質の大規模FMO計算による相互作用エネルギー解析
  - ⇒ MM-MD/FMO連携による統計的&データ科学的な評価

■HPCI拠点へのABINIT-MPのライブラリ整備 （責任者：望月）

- A64FX系スパコンへの改良版の導入
  - ⇒ 「富岳」、「不老」、「Wisteria-o」