SMASH-1.1.0 チュートリアル

石村　和也
分子科学研究所
計算分子科学研究拠点(TCCI)

第22回CMSI神戸ハンズオン
SMASHチュートリアル
2015年2月16日

目次

13:00-14:30

- SMASHの概要
- SMASHのディレクトリとファイル
- SMASHの並列化手法
- SMASHのインプットファイル

14:30-15:00

- コーヒーブレーク

15:00-17:00

- 実習及び質疑応答

SMASHの概要

SMASHプログラム

- 大規模並列量子化学計算プログラムSMASH (Scalable Molecular Analysis Solver for High performance computing systems)
- オープンソースライセンス(Apache 2.0)
- http://smash-qc.sourceforge.net/
- 2014年9月1日公開
- 最新バージョンは1.1.0 (2015年1月3日公開)
- 対象マシン：スカラ型CPUを搭載した計算機(PCクラスタから京コンピュータまで)
- エネルギー微分、構造最適化計算を重点的に整備
- 現時点で、Hartree-Fock, DFT(B3LYP), MP2計算が可能
- MPI/OpenMPハイブリッド並列を設計段階から考慮したアルゴリズム及びプログラム開発 (Module変数、サブルーチン引数の仕分け)
- 言語はFortran90/95
- 1,2電子積分など頻繁に使う計算ルーチンのライブラリ化で開発コスト削減
- 9月-2月10日までの約5か月で、ダウンロード数は178(うち国内が78%)

SMASHの開発方針

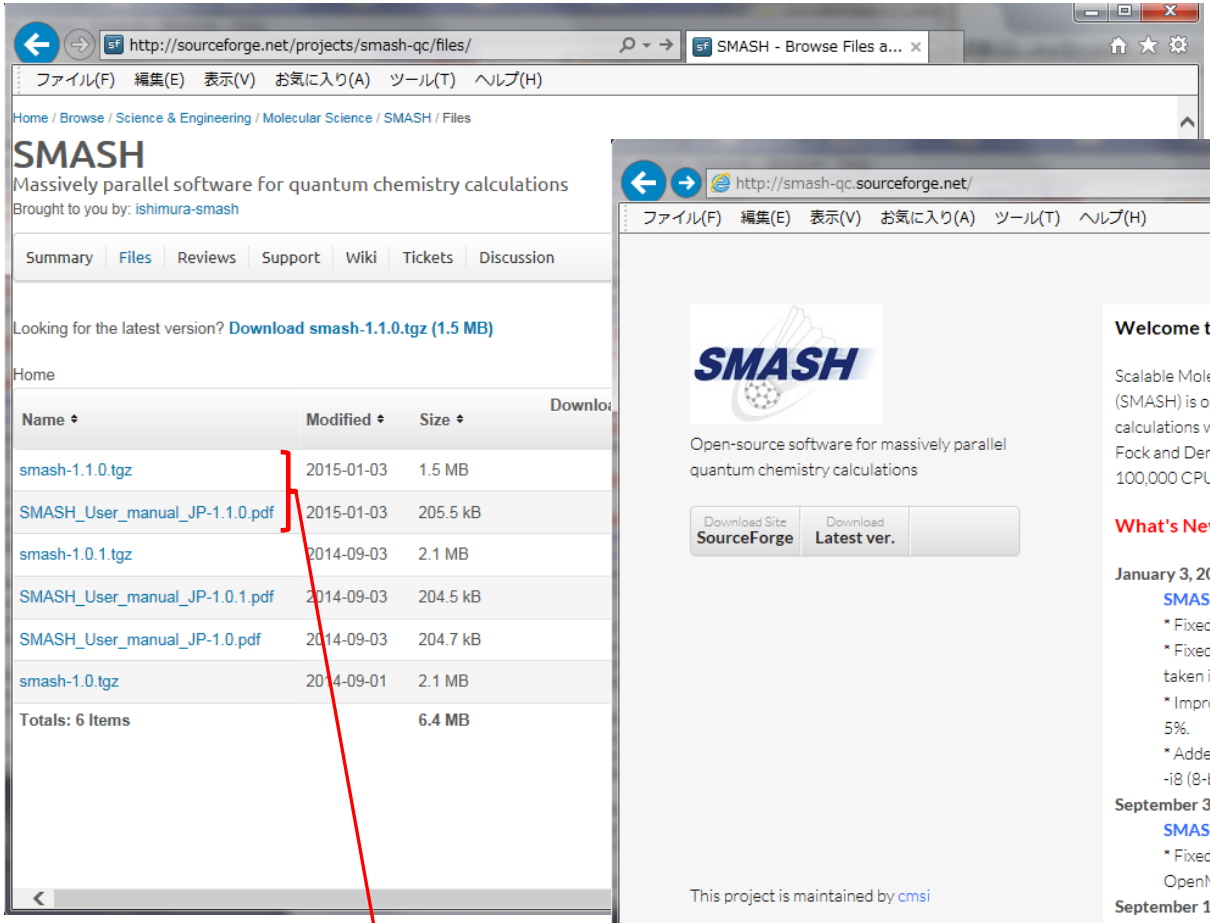
- キーワードはシンプル(実行方法、入力形式、ライセンス、開発)
- MPI/OpenMPハイブリッド並列化と高速計算アルゴリズムを組み込み、演算量削減、計算負荷とデータの均等な分散、通信の最適化を行い、一つのプログラムでスカラ型CPU搭載計算機をカバー
  - 今後ノード当たりのコア数はさらに増加
  - 京、おそらくポスト京も研究室レベルの計算機と基本的な構成はほぼ同じ
- よく用いられるルーチンのライブラリ化・モジュール化
- オープンソースライセンスで配布
  - ますます複雑になる計算機を理解した上で、最適な式、アルゴリズム、近似を開発、選択してプログラムを作る必要があり、開発コスト削減は不可欠
  - 複数の人が同じような開発・チューニングをしない仕組み作り
  - 技術・ノウハウの共有
  - 計算機科学との連携

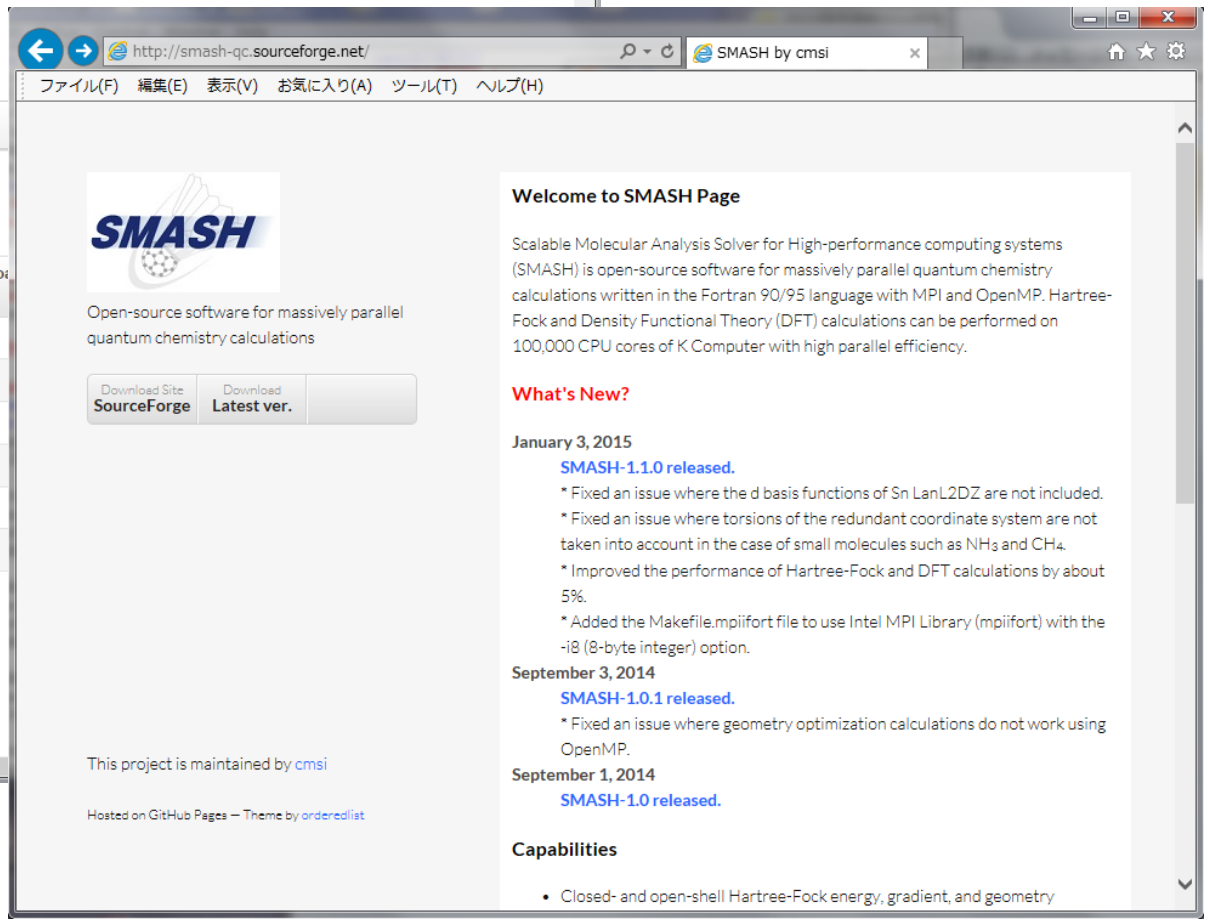
SMASHプログラミングルール

1. 言語はFortran90/95で、並列化はMPIとOpenMPを利用する。
2. コンパイラの診断オプションを使ってコードをチェックする。
   - ex) ifort -warn all -check bounds
   - gfortran -Wall -fbounds-check
   - pgf90 -Minform=inform -Mbounds
3. プロセス間のデータ通信はsrc/parallel.F90のMPIラッパーを利用する。直接MPIルーチンを呼ばない。
4. MPIを使わずにコンパイルする場合はnoMPIマクロを利用する
5. implicit noneを利用し、すべての変数を定義する。整数はi-nで始まる変数にする。
6. 配列をallocate文で確保する場合、call memsetで利用メモリ量をカウントする。開放するときは、call memunsetで開放する量を差し引く。
7. module変数は、parameter、threshold、inputデータなど計算中変更しないもののみとする。(例外：座標もmodule変数とする)
8. プログラムを止める場合、エラーメッセージを出力してcall iabortを書く。
9. inputの読み込み、checkpointファイルの読み書きを行うサブルーチンはfileio.F90に書く。
10. ディスクへの書き込みは、標準出力とcheckpointファイルの書き込みのみ。

SMASH Webページ

http://smash-qc.sourceforge.net/





最新ソースコードと
日本語マニュアル

SMASHのディレクトリとファイル

SMASHのディレクトリ構成

ディレクトリ名	内容
bin	実行ファイル
doc	ドキュメント
example	サンプルインプット・アウトプット
obj	オブジェクトファイル
src	ソースファイル
vtk	可視化用ファイル

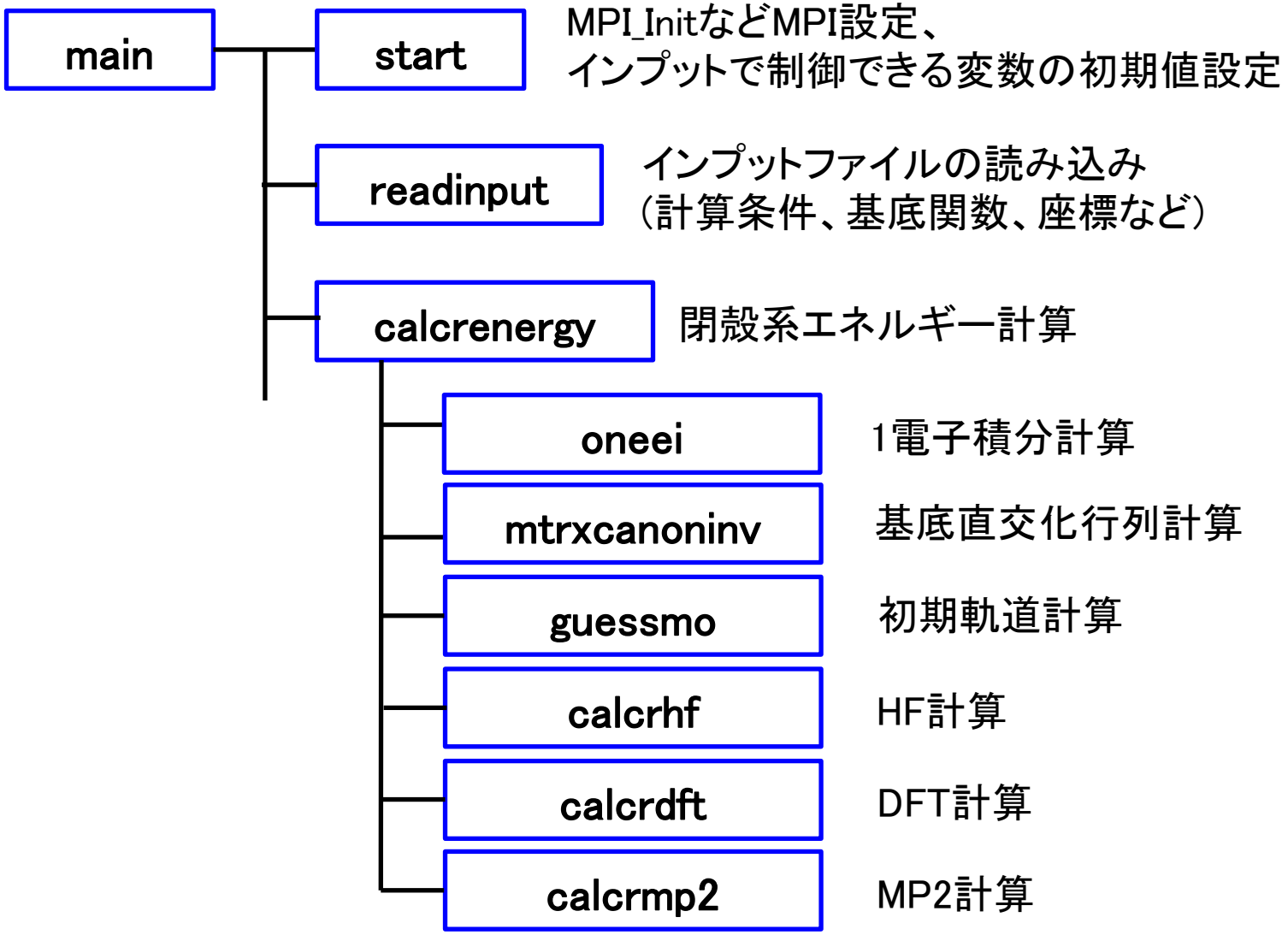
Makefile一覧	
Makefile	ifortベースのmpif90用
Makefile.fujitsu	mpifrtpx(京、FX10)用
Makefile.mpiifort	mpiifort用
Makefile.x86_64.noMPI	ifort (or gfortran)用

例)京コンピュータの場合: make -f Makefile.fujitsu

SMASHの主要なソースファイル

ファイル	内容
main.F90	メインルーチン
module.F90	モジュール変数（基底関数、座標、閾値など、座標以外は計算中変更しない変数）
parallel.F90	MPIルーチンラッパー
memory.F90	メモリ管理
fileio.F90	ファイル入出力（標準出力以外）
modulefmt.F90, modulerys.F90	1,2電子積分用データ
rysquad.F90	Rys積分のコアルーチン（f以上の角運動量の関数用、1,2電子積分で利用）
int1.F90	1電子積分計算
int2.F90, int2elec.F90, int2sp.F90, int2spd[1-4].F90	2電子積分計算
scf.F90	SCF計算

main.F90


main
start
MPI_InitなどMPI設定、
インプットで制御できる変数の初期値設定
readinput
インプットファイルの読み込み
(計算条件、基底関数、座標など)
calcrenergy
閉殻系エネルギー計算
oneei
1電子積分計算
mtrxcanoninv
基底直交化行列計算
guessmo
初期軌道計算
calcrhf
HF計算
calcrdft
DFT計算
calcrmp2
MP2計算

module.F90

モジュール	変数	内容
modjob	method runtype scftype	計算方法 (HARTREE-FOCK, B3LYPなど) 計算手順 (ENERGY, GRADIENT, OPTIMIZE) SCF波動関数の種類 (RHF, UHF)
modparallel	nproc1, myrank1 nproc2, myrank2	mpi_comm1(全体)のプロセス数とrank mpi_comm2(部分)のプロセス数とrank, 通常mpi_comm1=mpi_comm2
modmolecule	numatomic coord znuc	各原子の原子番号 各原子の座標 (単位:atomic unit) 各原子の核電荷 (ECPの補正後)
modbasis	locprim locbf locatom mprim mbf mtype	基底シェルのprimitive関数の先頭アドレス contracted基底関数の先頭アドレス 原子の番号(≠原子番号) primitive関数の数 基底関数の数 (s:1, p:3, d:5or6,...) 軌道角運動量(s:0, p:1, d:2,...)

parallel.F90

- MPIのラッパールーチンをまとめている
- use mpi、include ”mpif.h”はparallel.F90内だけで使用する
- ルーチン名は、para_＊
- マクロを使い、MPIを使わない場合でもmakeできるようにしている
- MPIを使わない場合、何もしないか、必要ならデータコピーを行う
- 整数の送受信では、整数のサイズをmoduleのmodparallelにあるinterface checkintsizeで自動判定している
- MPI_Bcastのラッパーは、para_bcastr(実数)、para_bcasti(整数)、para_bcastc(文字)、para_bcastl(ロジカル)の4つ

memory.F90

- インプットのjob memoryでノード当たりの最大使用メモリ量を設定する（デフォルト：8GB）

サブルーチン	内容
maxmemset	ノード当たりの最大使用メモリ量を8バイト変数の個数で変数memmaxに設定する
memset(msize)	allocate文で確保するメモリ量msizeをmemusedに加える もし、memusedがmemmaxを超えると計算を止める
memunset(msize)	deallocate文で開放するメモリ量msizeをmemusedから引く
memrest(msize)	利用可能なメモリ量をmsizeで返す

fileio.F90(1)

- Subroutine readinputでmpi_comm1のマスタープロセスがインプットファイルの内容を読み込み、最後にbcastですべてのプロセスに情報を送信する
  - call low2upでインプットファイルの文字をすべて(チェックポイントファイル名以外)小文字から大文字に変換する
  - 必要ならnamelist行の最初に&を追加する
  - 必要ならcall addaposで文字列のnamelist変数に' 'を追加する
  - 必要ならnamelist行の最後に/を追加する
  - 上記の内容を新たなファイルinput.dat(プロセス番号)に書き込む
  - namelistで計算方法などの情報を読み込む
  - readatomで原子の元素記号と座標を読み込む
- job basis=genの場合、readbasisで各原子の基底関数を読み込む
- job ecp=genの場合、readecpで各原子のECPを読み込む

fileio.F90(2)

subroutine readinputの冒頭でインプットファイルを右のように変換したファイル作成し、この変換ファイルの内容を読み込んでいる

インプットファイル

```
job method=mp2 basis=gen
control check=test.chk
scf dconv=1.0d-4
geom
 O   0.000     0.000     0.142
 H   0.000     0.756    -0.462
 H   0.000    -0.756    -0.462

basis
O H
6-31G(d)
****
```

input.dat(プロセス番号)

```
&JOB METHOD='MP2' BASIS='GEN' /
&CONTROL CHECK='test.chk'  /
&SCF DCONV=1.0D-4 /
GEOM
O   0.000     0.000     0.142
H   0.000     0.756    -0.462
H   0.000    -0.756    -0.462

BASIS
O H
6-31G(D)
****
```

modulefmt.F90, modulerys.F90, rysquad.F90

- 1電子、2電子積分計算ルーチンを抜き出す場合、mofulefmt.F90, modulerys.F90, rysquad.F90も必要になる
- mofulefmt.F90, modulerys.F90：1電子、2電子積分計算で必要になる内挿用テーブル
- rysquad.F90：Rys求積法で使う1電子、2電子積分共通ルーチン
- 内挿の展開次数の調整により、積分の誤差は10^{-14}以下

int1.F90(1)

サブルーチン	内容
int1cmd	1電子Coulomb積分ルーチン（S,P,D関数までの場合）
int1rys	1電子Coulomb積分ルーチン（F,G関数を含む場合）

- $\left(\chi_{\boldsymbol{i}}\middle|\sum_{a}^{\mathrm{natom}} Z_a/r_{1a}\middle|\chi_{\boldsymbol{j}}\right)$
- int1cmdとint1rysの引数は同じ
- subroutine int1cmd(cint, exij, coij, coordij, coord, znuc, natom,
  nprimij, nangij, nbfij, len1, mxprsh, threshex)
- Output: cint
- Input: その他すべて
- cint($\boldsymbol{j}$,$\boldsymbol{i}$)の順でデータが入っている
- その他の配列はexij$\left(*,\boldsymbol{1}(=\boldsymbol{i})\right)$, exij$\left(*,\boldsymbol{2}(=\boldsymbol{j})\right)$の順で入れる
- 実数はすべて倍精度

int1.F90(2)

- $(\chi_i|\sum_a^{\mathrm{natom}} Z_a/r_{1a}|\chi_j)$
- subroutine int1cmd(cint, exij, coij, coordij, coord, znuc, natom, nprimij, nangij, nbfij, len1, mxprsh, threshex)

変数	内容
cint(len1,len1)	1電子Coulomb積分
exij(mxprsh,2)	ブラ・ケットの基底関数の指数
coij(mxprsh,2)	基底関数の係数
coordij(3,2)	原子の座標
coord(3,*)	全原子の座標
znuc(*)	全原子の核電荷
natom	全原子数
nprimij(2)	基底関数のprimitive関数の数
nangij(2)	基底関数の軌道角運動量(s:0, p:1, d:2,...)
nbfij	基底関数の軌道数 (s:1, p:3, d:5or6,...)
len1	cintの次元数
mxprsh	exij, coijの第1次元数 (デフォルト 30)
threshex	exp(−x)のx閾値 (デフォルト30.0)

データの並び方

SMASH-2.0.0からcanonical orderに変更予定

p関数	d関数 (Spherical)	d関数 (Cartesian)	f関数 (Spherical)	f関数 (Cartesian)
p_x	d_0	d_{xx}	f_0	f_{xxx}
p_y	d_{+1}	d_{yy}	f_{+1}	f_{yyy}
p_z	d_{-1}	d_{zz}	f_{-1}	f_{zzz}
	d_{+2}	d_{xy}	f_{+2}	f_{xxy}
	d_{-2}	d_{xz}	f_{-2}	f_{xxz}
		d_{yz}	f_{+3}	f_{xyy}
			f_{-3}	f_{yyz}
				f_{xzz}
				f_{yzz}
				f_{xyz}

int2elec.F90, int2.F90, int2sp.F90, int2spd[1-4].F90(1)

サブルーチン	内容
int2elec	2電子Coulomb積分ルーチン（G関数まで対応）

- $(\chi_i(1)\chi_j(1)|r_{12}^{-1}|\chi_k(2)\chi_l(2))$
- Subroutine int2elec(twoeri, exijkl, coijkl, xyzijkl, nprimijkl, nangijkl, nbfijkl, maxdim, mxprsh, threshex)
- Output: twoeri
- Input:その他すべて
- twoeri($\boldsymbol{l}, \boldsymbol{k}, \boldsymbol{j}, \boldsymbol{i}$)の順でデータが入っている
- その他の配列はexijkl$(*, \mathbf{1}(=\boldsymbol{i}))$, exijkl$(*, \mathbf{2}(=\boldsymbol{j}))$, exijkl$(*, \mathbf{3}(=\boldsymbol{k}))$, exijkl$(*, \mathbf{4}(=\boldsymbol{l}))$の順で入れる
- 実数はすべて倍精度

int2elec.F90, int2.F90, int2sp.F90, int2spd[1-4].F90(2)

- $\left(\phi_i(1)\phi_j(1)|r_{12}^{-1}|\phi_k(2)\phi_l(2)\right)$
- Subroutine int2elec(twoeri, exijkl, coijkl, xyzijkl, nprimijkl, nangijkl, nbfijkl, maxdim, mxprsh, threshex)

変数	内容
twoeri(maxdim,maxdim, maxdim,maxdim)	2電子Coulomb積分
exijkl(mxprsh,4)	ブラ・ケットの基底関数の指数
coijkl(mxprsh,4)	基底関数の係数
xyzijkl(3,4)	原子の座標
nprimijkl(4)	基底関数のprimitive関数の数
nangijkl(4)	基底関数の軌道角運動量(s:0, p:1, d:2,...)
nbfijkl(4)	基底関数の軌道数 (s:1, p:3, d:5or6,...)
maxdim	twoeriの次元数
mxprsh	exijkl, coijklの第1次元数 (デフォルト 30)
threshex	exp(−x)のx閾値 (デフォルト30.0)

scf.F90


calcrhf
閉殻系HFエネルギー計算
distarray
Fock行列の履歴分散保存、行列積並列計算のための
データ分散量と先頭アドレス計算
calcdmtrx
分子軌道係数から密度行列計算
calcschwarzeri
2電子積分計算カットオフのためのデータ作成
do iter = 1, maxiter
calcrdmax
Shellごとの最大密度行列値計算 (2電子積分
計算カットオフで利用)
formrfock
Fock行列計算 (内部で2電子積分計算実行)
calcdiiserr
DIIS error行列計算(DIIS=.true.の場合)
calcrdiis
DIIS法によるFock行列内挿計算
diagfock
Fock行列対角化
ddiff
密度行列差分計算

SMASHの並列化手法

MPI/OpenMPハイブリッド並列化

- 極力均等な負荷分散
- 分散のための余分なコストは極力削減
- ノード間のデータ分散、ノード内のデータ共有、キャッシュミス削減を同時に満たすよう、多次元配列の取り方と多重ループの回し方を工夫
- MPI通信データ量と回数の最適化
- ほぼすべての演算をハイブリッド並列化（現時点では対角化以外）
- BLAS、LAPACKルーチンを利用し、スレッド並列化の開発コストを削減

SMASHの並列実行方法

- MPI(ノード間)並列

```
mpirun -np (プロセス数) bin/smash < (inputファイル名) >
(outputファイル名)
```

- OpenMP(ノード内)並列
  - bashの場合：~/.bashrcファイルに次の行を追加

```
export OMP_NUM_THREADS=(スレッド数)
ulimit -s unlimited
export OMP_STACKSIZE=1G
```

<-- おおよそ(10*基底次元*基底次元*スレッド数)バイト以上のサイズであればよい

  - csh,tcshの場合：~/.cshrcもしくは~/.tcshrcファイルに次の行を追加

```
setenv OMP_NUM_THREADS (スレッド数)
unlimit
setenv OMP_STACKSIZE 1G
```

Hartree-Fock計算手順

$$\mathbf{FC} = \boldsymbol{\varepsilon}\mathbf{SC}$$

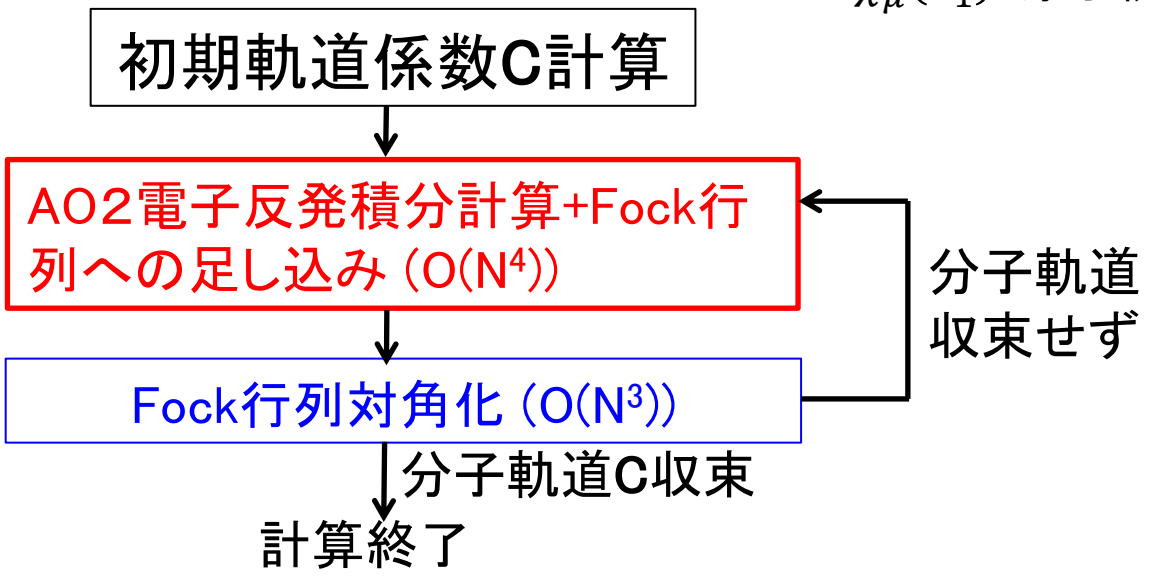
F: Fock行列, C: 分子軌道係数
S: 基底重なり行列, ε: 分子軌道エネルギー

Fock行列 $F_{\mu\nu} = H_{\mu\nu} + \sum_{i,\lambda,\sigma} C_{\lambda i} C_{\sigma i} \{2(\mu\nu|\lambda\sigma) - (\mu\lambda|\nu\sigma)\}$

原子軌道(AO)2電子積分

$$(\mu\nu|\lambda\sigma) = \int d\boldsymbol{r}_1 \int d\boldsymbol{r}_2 \chi_\mu(\boldsymbol{r}_1)\chi_\nu(\boldsymbol{r}_1)\frac{1}{r_{12}}\chi_\lambda(\boldsymbol{r}_2)\chi_\sigma(\boldsymbol{r}_2)$$

$\chi_\mu(\boldsymbol{r}_1)$: 原子軌道Gauss関数



MPI/OpenMPハイブリッド並列化

K. Ishimura, K. Kuramoto, Y. Ikuta, S. Hyodo, J. Chem. Theory Comp. 2010, 6, 1075.

scf.F90のsubroutine formrfock

Fock行列 $F_{\mu\nu} = H_{\mu\nu} + \frac{1}{2}\sum_{\lambda,\sigma} D_{\lambda\sigma}\{2(\mu\nu|\lambda\sigma) - (\mu\lambda|\nu\sigma)\}$

- OpenMPの分散を最外ループで行うことにより、スレッド生成などのオーバーヘッドを削減
- IF文を使わずにプロセス間の分散を行い、振り分けコストを大幅に削減

```
!$OMP parallel do schedule(dynamic,1) reduction(+:Fock)
do μ=n, 1, -1                          <----- OpenMPによる振り分け
  do ν=1, μ
    μν=μ*(μ+1)/2+ν
   λstart=mod(μν+mpi_rank,nproc)+1
   do λ=λstart, μ ,nproc               <----- MPIランクによる振り分け
     do σ=1, λ
       AO2電子積分(μν|λσ)計算+Fock行列に足し込み
     enddo
   enddo
  enddo
enddo
!$OMP end parallel do
call mpi_allreduce(Fock)
```

mpi_comm1とmpi_comm2

- mpi_comm1, nproc1, myrank1
  - mpi_comm1 = MPI_COMM_WORLD
  - インプットデータの読み込みと送受信、Fock行列計算の分散、MP2計算の分散など、大半の演算とプロセス間通信で利用
- mpi_comm2, nproc2, myrank2
  - DIIS、SOSCF、行列対角化、基底直交化変換の行列積など、全体で分散させるには演算量が少ない場合に、ある程度のプロセス数ごとに同じ計算をするために用意している
  - 現在は、mpi_comm2 = mpi_comm1
  - mpi_comm1とmpi_comm2を違うコミュニケータにするには、現時点ではソースを書き変える必要がある

MPI並列手順

- main.F90 subroutine start
  - call para_init, para_comm_size, para_comm_rankでMPI並列開始
  - mpi_comm1,nproc1,myrank1をmpi_comm2,nproc2,myrank2にコピー
  - mpi_comm1のマスタープロセスのみmaster=.true.にする
- fileio.F90 subroutine readinput
  - mpi_comm1のマスタープロセスがインプットの内容を読み込み、その他のプロセスにbcastで送信する
- fileio.F90 subroutine readbasis, subroutine readecp
  - mpi_comm1のマスタープロセスがインプットの内容を読み込む
  - これらのサブルーチンを出た後、その他のプロセスにbcastで送信する
- fileio.F90 subroutine readcheckinfo, subroutine readcheckguess
  - mpi_comm1のマスタープロセスがチェックポイントファイルの内容を読み込み、その他のプロセスにbcastで送信する

SMASHのインプットファイル

インプットファイル形式

- サンプルインプット・アウトプットファイルはsmash/example/を参照
- 計算方法や条件をjob, controlセクションなどで設定
- 分子座標はgeom行の次から記入、空行もしくはファイルの最後で座標読み込み終了
- 基底関数を元素ごとに指定する場合、basis行の次から記入
- 大文字と小文字の区別は無し

```
job  runtype=optimize method=b3lyp basis=gen memory=1g
control spher=.false.  cutint2=1.0d-10
geom
 O   0.0000000     0.0000000     0.1423813
 H   0.0000000     0.7568189    -0.4626257
 H   0.0000000    -0.7568189    -0.4626257

basis
O H
6-31G(d)
****
```

インプットファイル(jobセクション)

変数	内容	設定値
runtype	計算実行方法	energy: エネルギー計算 (default) gradient: エネルギー微分計算 optimize: 構造最適化計算
method	計算方法	HF: Hartree-Fock計算 (default) b3lyp: B3LYP計算 mp2: MP2計算
basis	基底関数	sto-3g (default), 6-31g, 6-31g(d), cc-pvdz, cc-pvtz, cc-pvqz, d95v, lanl2dz gen: 元素ごとに指定
memory	1ノード当たりメモリ使用量	(default 2GB) 単位:B, KB, MB, GB, TB
charge	系の電荷	(default ±0.0)
multi	スピン多重度	1: singlet 2: doublet 3,4,...: triplet, quartet, ...
scftype	波動関数種類	RHF : Closed-shell (default) UHF: Open-shell
ecp	ECP (Effective Core Potential)	lanl2dz gen: 元素ごとに指定

インプットファイル(controlセクション)

変数	内容	設定値
spher	Spherical HarmonicsもしくはCartesian基底指定	.true. : Spherical (5d, 7f,...) (default) .false. : Cartesian (6d, 10f,...)
guess	初期波動関数	huckel : 拡張Huckel計算 (default) check : チェックポイントファイル参照
check	チェックポイントファイル名	(default:空白)
cutint2	2電子積分のカットオフ	1.0d-12 (default)
bohr	距離の単位	.false. : Å(default) .true. : bohr
iprint	出力制御	1 : 最少出力 2 : 通常出力 (default) 3 : 詳細出力

インプットファイル(scfセクション)

変数	内容	設定値
diis	DIIS指定	.true. : DIIS (default) .false. : Second-order SCF
maxiter	最大SCF回数	100 (default)
dconv	SCFでの電子密度収束判定	5.0D-6 (default)
maxdiis	最大DIIS回数	20 (default)
maxsoscf	最大SOSCF回数	20 (default)

インプットファイル(opt,dftセクション)

optセクション

変数	内容	設定値
nopt	最大Opt回数	50 (default)
optconv	OptでのForce収束判定	1.0D-4 (default)
cartesian	構造最適化時の座標系	.true. : Cartesian coordinate .false. : Redundant coordinate (default)

dftセクション

変数	内容	設定値
nrad	汎関数数値積分の動径点数	96 (default)
nleb	汎関数数値積分のLebedevグリッド角度点数	302 (default)

実習内容

実習1

(ss|ss)型2電子積分計算

- 水素 STO-3G基底、4中心全て原点の場合の(ss|ss)積分値を求めよ。
- STO-3G基底は、次の指数、係数をもつs関数(軌道角運動量0、軌道数1)3つの重ね合わせである。下記の係数は規格化後の数値である。
- STO-3G基底関数：$\chi_{STO-3G-H} = \sum_{i=1}^{3} C_i \exp(-\alpha_i r^2)$
- 必要なファイルは、 modulefmt.F90 , modulerys.F90 , int2elec.F90, int2sp.F90, int2spd1.F90, int2spd2.F90, int2spd3.F90, int2spd4.F90, rysquad.F90。
- rysquad.F90の332行目のcall iabortは削除してください。
- int2elecの呼び出し方はint2.F90のsubroutine calc2eriを参照。

指数α	係数 C
3.4252509	0.27693436
0.6239137	0.26783885
0.1688554	0.08347367

実習2

(ds|ss)型2電子積分計算

- Spherical Harmonics(d軌道数：5)、Cartesian(d軌道数：6)、それぞれの場合の(ds|ss)積分値を求めよ。
- s関数：水素のSTO-3G基底、中心はすべて原点
- d関数：関数は1つ、中心座標は(1.34, 0.0, 0.0) (atomic unit)

s関数

指数α	係数 c
3.4252509	0.27693436
0.6239137	0.26783885
0.1688554	0.08347367

d関数

指数α	係数 c
0.8	1.11382493

実習の答え

- 実習1

0.7746060

- 実習2

d_0	−0.1594676
d_{+1}	0.0000000
d_{-1}	0.0000000
d_{+2}	0.2762060
d_{-2}	0.0000000

d_{xx}	0.5660562
d_{yy}	0.2471210
d_{zz}	0.2471210
d_{xy}	0.0000000
d_{xz}	0.0000000
d_{yz}	0.0000000