GAMESS-FMO 使用の手引き

ドミトリ フェドロフ、北浦和夫

平成 20 年 12 月

暫定改訂版（平成 24 年 2 月 29 日）

目次

１．GAMESS で可能な FMO 計算

FMO 計算ができるプログラムとしては、ABINIT-MP (http: //www.ciss.iis.u-tokyo.ac.jp /dl/index.html#download_2)、PAICS (http://www.paics.net/)、OpenFMO (http://www.openfmo.org/ OpenFMO/jp/index.html) と GAMESS (http://www. msg.ameslab.gov/GAMESS/) が開発されている。それぞれ特徴的な機能が実装されているので、目的に合わせてこれらを使い分けて欲しい。本稿では、GAMESS による FMO 計算の実際について述べる。

GAMESS は、アイオワ州立大学の Mark. S. Gordon 教授らのグループで開発された非経験的分子軌道法計算プログラムであり、2004年版（ver. 26 May 2004）以降から FMO 計算がサポートされている。GAMESS の入手法とコンパイルの仕方については、本稿の附録 1 を参照されたい。GAMESS はいろんなプラットフォームで動くが、Windows 版はバイナリで提供されているので手軽に計算を体験できる。また、最近開設された FMO 法のポータルサイト（http://fmo.alcf.anl.gov/fmo-intro.php）では、簡単な FMO 計算を体験できる。なお、GAMESS で通常の ab initio MO 計算をする場合の使い方については、成書（平尾公彦監修、武次徹也編集、「すぐできる量子化学計算　ビギナーズマニュアル」、講談社サイエンティフィク、2006年）を参照されたい。

FMO 計算では、通常の ab initio 計算のための入力データに加えて、分子の分割データなどを用意しなければならない。これらは、巨大分子の場合には、手作業で作成するのは大変困難な作業となる。そのため、入力データの作成を支援するためのソフトウェア FMOutil が GAMESS と一緒に配布されている。FMOutil を用いると、ポリペプチドとタンパク質については、PDB データから簡単に FMO 計算のための入力データが作成できる。通常、タンパク質の計算は構造モデルを作成することから始めなければならないので、モデリングについての知識・経験・ソフトウェアが必要となるが、FMOutil は全く始めてタンパク質を計算する人向けに、これだけで FMO 計算の入力データを作成できるように作られたもので、あくまで学習用であり実践的に用いるには機能が十分ではない。実践的な FMO 計算では、分子モデリングソフト Facio(http://www1.bbiq.jp/zzzfelis/Facio_Jp.html)を使うことを推奨する。このプログラムを用いると、ポリペプチドのみならず DNA/RNA、糖鎖やその他の巨大分子の構造モデリングと入力データの作成や計算結果の可視化を容易に行うことができる。

GAMESS にはいろいろな電子状態理論とプロパティ計算法が組み込まれているが、最新バージョン（GAMESS ver. 11 Aug 2011）で可能な FMO 計算を表 1 に示す。これらの一部の計算では、連続誘電体溶媒モデル(poralizable continuum model; PCM)、有効フラグメントポテンシャル（effective fragment potential; EFP)やモデルコアポテンシャル（model core potential; MCP)などと組み合わせて使うことができる。詳細は GAMESS のマニュアル（上記 GAMESS のホームページで閲覧できる）を参照されたい。

表１　GAMESS（FMO ver.4.1 in GAMESS ver. 11 Aug 2011）で可能な FMO 計算の一覧 [a]

波動関数/モデル	エネルギー（FMOn[b]）	エネルギー勾配 [c]	備考
RHF（閉殻系）	yes (2,3)	yes/yes	制限 Hartree-Fock
ROHF（開殻系）	Yes(2,3)	yes/no	制限 Hartree-Fock
DFT（閉殻系）	yes (2,3)	yes/no	密度汎関数理論
MP2	yes (2,3)	yes/yes	Møller-Plesset の 2 次摂動論
RI-MP2	yes(2)	no/no	Resolution of the identity-MP2
CC	yes (2,3)	no/no	カップルド・クラスター
CIS	yes(1)	no/no	1 電子励起配置間相互作用
TDDFT	yes (1,2)	yes/no	時間依存 DFT
MCSCF	yes (2)	yes/no	多配置 SCF 法

[a] その他溶媒モデルとの組み合わせ計算などについては GAMESS のマニュアルを参照

[b] 括弧内の 1、2 、3 は、それぞれ、FMO1(1 体展開)、FMO2(2 体展開)、FMO3(3 体展開)を示す

[c] 近似的/完全解析的

２．FMO 計算の default 値

始めに、FMO 計算における default 値について述べる。計算できる最大原子数は GAMESS では 2000 に設定されているが（インストール時に変更が可能）、FMO 計算では原子数は無制限である。ただし、構造最適化計算で、FMO の最適化ルーチンを用いる場合は無制限であるが、GAMESS の最適化ルーチンを用いる場合は、GAMESS の最大原子数が上限となる（GAMESS のルーチンの方が効率が良いので、2000 原子以内の場合はこちらを使うことを推奨する）。FMO 計算では、数千から数万原子からなる巨大系の計算を想定しているので、エネルギーの有効桁数を十分に確保するために、SCF 収束判定などの default 値が、低分子のための値（GAMESS の default 値）よりも厳しく設定されている。すなわち、2 電子積分のカットオフは、ICUT=12、ITOL=24、SCF の収束判定は、CONV=1e-7 である。また、FMO 特有の近似である、環境静電ポテンシャル（RESPAP と RESPPC）や離れた 2 量体の静電相互作用（RESDIM）近似の適用距離（フラグメント間の最近接原子間距離で決められるが、この距離は、原子の van der Waals 半径の和を基準として、その何倍かで指定する）は、2.0（エネルギー計算）や 2.5（エネルギー勾配計算）に設定されている。これらのパラメータは、出力ファイルにプリントされるので、いつでも確認できる。これらは、十分なテストを行って妥当な値として決められているが、変更する場合は、入力データで明示的に指定すればよい（詳細は、GAMESS のマニュアルの$FMO と$FMOPRP データグループの項を参照されたい）。

３．チュートリアルー酒精と水ー

３．１　入力データ

エタノールと水分子系を CH3、－CH2OH と H2O の 3 つのフラグメントに分割して（図 1）、FMO 計算を行う例について説明する。この計算は、共有結合の切断（CH_3…CH_2OH）と分子間相互作用（C_2H_5OH と H_2O）が含まれるので、入力データと計算結果の見方を説明するのに適した最も簡単な例である。本稿は、GAMESS の FMO チュートリアル（D.G.Fedorov, FMOtuto.inp）を、日本語に訳し、一部、修正・補筆したものである。

ここでの説明は、あくまで初心者のためのチュートリアルなので、次の段階では、FMOutil で、他の波動関数や基底関数（さらには、一部の原子に diffuse 関数を加えるなど）を使う場合の入力データを生成して、それを見て詳細を学習されたい。このソフトウェアは、FMO 計算のためのデータ作成を会話型で行うので、FMO 計算でどのようなオプションがあるかを知るのに便利である。再度の繰り返しになるが、実践的な計算を行う場合は、モデリングソフト Facio を使って入力データの作成と計算結果の可視化を行うことを強く推奨する。

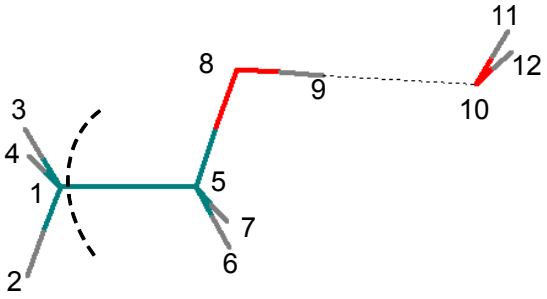


図 1　エタノールと水分子系の構造と原子の通し番号。エタノール分子の C1－C5 結合を切断し（破線アーク）、CH_3、－CH_2OH と H_2O の 3 つのフラグメントとする。

データ 1　$C_2H_5OH+H_2O$ の RHF/STO-3G 計算の入力データ

	$CONTRL SCFTYP=RHF RUNTYP=ENERGY $END				
	$SYSTEM TIMLIM=2 MEMORY=100000 $END				
	$BASIS GBASIS=STO NGAUSS=3 $END				
	$DATA				
C2H5OH+H2O					
C1					
	C	6.0	2.341068918	-0.286969289	-0.007419409
	H	1.0	3.074585965	0.377273699	0.439774414
	H	1.0	2.566531043	-0.392400032	-1.064091814
	H	1.0	2.426179456	-1.263297983	0.459562336
	C	6.0	0.916690128	0.27616509	0.183197532
	H	1.0	0.723544203	0.404142341	1.256761188
	H	1.0	0.8641657	1.27584686	-0.268509542
	O	8.0	-0.021561663	-0.620153163	-0.415679612
	H	1.0	-0.902681634	-0.194429743	-0.253432118
	O	8.0	-2.449361482	0.518010526	0.010231931
	H	1.0	-2.930984114	0.656472858	-0.839996915
	H	1.0	-3.058351768	-0.105961398	0.472645446
	$end				

まず、通常の ab initio MO 計算の入力データを、データ 1 に示す。

($CONTRL、$SYSTEM、$DATA)

GAMESS の入力データは、グループにまとめられている（本稿の附録および GAMESS のマニュアル、または、１．で述べた成書を参照)。各データグループは、$Group 名と$end で区切られる（$は必ず 2 カラム目に置く。大文字、小文字は区別されない)。$CNTRL グループで、SCFTYP=RHF で波動関数のタイプを RHF、RUNTYP=ENERGY で、エネルギーの 1 点計算であることが指定されている。$SYSTEM グループでは、TIMLIM で計算打ち切り時間（秒)、MEMERY で最大使用メモリ（byte）が指定されている。$BASIS グループでは、STO-3G 基底関数が指定されている。$DATA グループの、第 1 行目はタイトル、第 2 行目は系の対称性を点群名（今の場合、C1 なので対称性なし。FMO 計算は C1 のみ可）で指定、第 3 行目から$end の間は、ラベル、原子核電荷、原子の x 座標、y 座標、 z 座標（Å 単位）である。

($DATA)

さて、これに対応する FMO 計算の入力データをデータ 2 に示す。まず、$CONTRL、$SYSTEM と$BASIS に変更はない。FMO の原子座標は独自の$FMOXYZ グループで入力されるので、$data グループの意味が本来のものとは変わっている。ここでは、単に、基底関数を内部で生成するために必要なデータとして、系を構成する原子種の、ラベル、原子核電荷を入力する。今の例の場合、水素原子、炭素原子、酸素原子となる。また、ここでのラベルは h－1 となっているが、文字に続く－1 は、基底関数を区別するために用いられるので、単なるプリントのためのラベルではなくなっている。たとえば、分子中の同じ原 子種に対して、異なった基底関数を用いる場合は、 h －2 というラベルを与えて別の基底関数を置くことができる。この場合、ここでの指定に対応して、$FMOXYZ で入力する原子のラベルにも（default は 1 なので、この場合は省略可)、同様に文字に引き続いて“－数字”を加えることで、対応する数字の基底関数を置くことができる（GAMESS のマニュアル $FMOXYZ を参照、または、FMOutil で、一部の原子に diffuse 関数を置く場合の入力データを生成して、それを参照されたい)。

データ 2　$C_2H_5OH+H_2O$ の FMO-RHF/STO-3G（3 フラグメント）計算の入力データ

	$CONTRL SCFTYP=RHF RUNTYP=ENERGY $END				
	$SYSTEM TIMLIM=2 MEMORY=100000 $END				
	$BASIS GBASIS=STO NGAUSS=3 $END				
	$data				
C2H5OH+H2O					
c1					
	h-1	1.0			
	c-1	6.0			
	o-1	8.0			
	$end				
	$FMOXYZ				
	C	6	2.341068918	-0.286969289	-0.007419409
	H	1	3.074585965	0.377273699	0.439774414
	H	1	2.566531043	-0.392400032	-1.064091814
	H	1	2.426179456	-1.263297983	0.459562336
	C	6	0.916690128	0.27616509	0.183197532
	H	1	0.723544203	0.404142341	1.256761188
	H	1	0.8641657	1.27584686	-0.268509542
	O	8	-0.021561663	-0.620153163	-0.415679612
	H	1	-0.902681634	-0.194429743	-0.253432118
	O	8	-2.449361482	0.518010526	0.010231931
	H	1	-2.930984114	0.656472858	-0.839996915
	H	1	-3.058351768	-0.105961398	0.472645446
	$end				
	$fmo				
	nfrag=3 indat(1)=1,1,1,1, 2,2,2,2,2, 3,3,3 $end				
	$FMOBND				
	-1	5	sto-3g	mini	
	$END				

	$FMOHYB						
	sto-3g	5	5				
	1	0	-0.117784	0.542251	0	0	0.850774
	0	1	-0.117787	0.542269	0.802107	0	-0.283586
	0	1	-0.117787	0.542269	-0.401054	-0.694646	-0.283586
	0	1	-0.117787	0.542269	-0.401054	0.694646	-0.283586
	0	1	1.003621	-0.015003	0	0	0
	mini	5	5				
	1	0	-0.109772	0.515046	0	0	0.864512
	0	1	-0.109775	0.515062	0.815062	0	-0.288166
	0	1	-0.109775	0.515062	-0.407531	-0.705864	-0.288166
	0	1	-0.109775	0.515062	-0.407531	0.705864	-0.288166
	0	1	0.996474	0.01561	0	0	0
	$END						

($FMOXYZ)

以下では、FMO 特有のデータグループについて説明する。まず、$FMOXYZ グループでは、ラベル、原子番号（整数）、x 座標、y 座標、 z 座標を入力する。$DATA グループの原子核電荷とは異なって、ここでは原子番号を整数で与えることに注意されたい。

($FMO)

$fmo グループでは、フラグメントの数（nfrag）と各原子をフラグメントにアサインするためのデータ(indat)を入力する。今の例では、フラグメントの数は 3 なので、nfrag=3 である。indat は配列データなので、1 番目の要素から順次データを与えるために、indat(1)= 1 番目のデータ、2 番目のデータ、… 、n 番目のデータ（n は系中の原子の総数）の形で与える。すなわち、” indat(1)=1,1,1,1, 2,2,2,2,2, 3,3,3 ”で、原子の通し番号 1 から 4 の原子（CH3）が 1 番目のフラグメント、4 から 8 の原子（CH2OH）は 2 番目のフラグメント、10 から 13 の原子（H2O）は 3 番目のフラグメントに属することを指定している。フラグメントデータの入力は、別の形式でも行える。こちらの方は、indat(1)=0 として、引き続き、原子の通し番号を指定し、フラグメントごとに区切りとして最後に 0 を入れる。すなわち、

```
indat(1)=0
    1   －4    0
    5    –9    0
    10  －13   0
```

となる。ここで、-4 など連続した番号の終わりを与えるデータは"マイナス 4"などであり、負符号と数字の間にブランクを入れてはいけない。また、フラグメントにアサインする原子の通し番号は、連続である必要はない（GAMESS のマニュアルの$FMO データグループの項を参照。FMOutil で入力データを生成すると後者の形式で出力される）。$FMO と書いた行には、後ろに何もつけてはいけない（次の行からデータを書く）。

（$FMOBND）

次は、$FMOBND グループである。ここでは、どの結合をどのように切断するかに関するデータを与える。今の例では、C1 と C5 を切断し、その結合電子対は C5 側のフラグメントにアサインする。これを、結合が切断される原子対を原子の通し番号で指定するとともに、電子が一つ減る側の原子（bond-detached atom;BDA)に負符号をつけて“－1　5“とする。さらに、同じ行に、切断した結合の末端処理に用いる hybrid orbital のラベル（hybrid　orbital 自体は、$FMOHYB データグループで定義される。ここで使われるラベルと同じラベル）を書く。例では、”sto-3g”と書かれているが、これは基底関数の指定ではなく、$FMOHYB で定義された名前であることに注意。Hybrid orbital の名前が複数指定されているが、最初が、計算に用いる基底関数用の hybrid orbital 名で、最後は、initial guess 用の基底関数（MINI 基底関数による拡張ヒュッケル計算が初期値として使われているので）の hybrid orbital を指定

している（後者は、いつも同じでよい）。切断する結合が複数あるときは、各結合すべてについて、同様のデータを書く。分子集合系のように、共有結合切断がない場合は、このデータは不要である（したがって、$FMOHYB グループもいらない）。

マルチレイヤー FMO 法（MFMO）で、m レイヤーとする場合は、低精度から高精度に向けて m 個の名前と、最後に initial guess 用の hybrid orbital の名前を書く（この場合、当然、$FMOHYB で、m+1 個の hybrid orbital を定義しなければならない）。

この例では、”－1　5”と切断したが、同じ C1－C5 結合を切断するにしても、”－5　1”も可能である。興味ある読者は、この切断でも計算を行って、両者を比較すると面白いであろう。

($FMOHYB)

$FMOHYB データグループは、切断した結合の BDA の上に置かれる projection operator を構成する hybrid　orbitals を入力する。まず、名前を定義して（“sto-3g“や“mini“）、次の行に、その基底関数の数(“5“)と hybrid orbital の数(“5“)を入力する。これらは、引き続き与えられた hybrid orbital の MO 係数の読み込みに使われる。係数を入力する行の先頭に、”1　0”や”0　1”のように 2 個の整数のフラグがあるが、最初の整数で”1”と指定すると、この hybrid obital は切断した結合原子の電子欠損側（BDA）で変分空間から外す（project out）ことを意味する。これが、”0”であれば変分空間に加えられる。同様に、第 2 番目の整数が” 1 ” であると、この hybrid orbital は電子が余分にアサインされた原子(bond-attaced atom; BAA)の変分空間から排除され、”0”であれば排除されない（変分関数として使われる）。例では、各 hybrid orbital は、必ず一方のフラグメントでのみ排除される（使われる）ように指定されている。このようにするのが、通常の使い方である（このフラグは特殊な用途のために用意されているので変な使い方をしないのが賢明である。たとえば、モノマーやダイマーが収束しない場合、特定の hybrid orbital を変分空間から排除して収束しない原因を調査したりするのに用いることを想定している）。

これらのフラグに引き続いて、hybrid orbital の MO 係数を入力する。これらの係数は事前に求めておかなければならない。炭素原子に関しては、~/gamess/tools/fmo/LMOs.txt で提供されている。または、FMOutil で出力されるので、これをコピーして使うとよい。Facio を使って入力データを作成すれば自動的に付けてくれるので、これらを意識することなく済ませることができる。ただし、hybrid orbital は、炭素原子の限られた基底関数でのみ用意されている。使用する基底関数のものが見つからない場合は、利用者が各自で作成することになる。この際、入力では、hybrid orbital は、z-軸の正方向に張り出したものを、必ず一番最初に与えなければならないことに注意して欲しい。FMO のプログラムは、これを前提として、hybrid orbital を結合相手の原子の方向に回転させて用いている。したがって、このルールが守られていないと、計算は行われるが正しい結果を与えないので、非常に重要なことである。

Hybrid orbital は、Boys や Edmiston-Reudenberg の LMO から作成してもよいが(FMO の計算結果は、これらにあまり依存性しない)、NLMO（natural localized molecular orbital）を使うことを推奨している（これだと、空軌道の LMO も求まる)。Hybrid orbital の作り方は簡単で、sp^3 炭素原子用の場合、CH_4 分子（必ず、一つの C－H 結合を z-軸上の正の方向に向けた構造を用いる）の計算を行って NLMO を求め、これから炭素原子の基底関数の係数を取り出して、規格化（これは必ずしも要件ではない。直交化はしない）すればよい。NLMO の計算は GAUSSIAN ではすぐにできるが（ただし、MO 係数が小数点以下 4 桁しかプリントされないため、FMOutil に内蔵している hybrid orbital の MO 係数は、プログラムを修正して少数点以下 6 桁をプリントさせて規格化したものである)、GAMESS の場合は、別途、Natural Bond Order Analysis (NBO）のプログラム(http://www.chem.wisc.edu/~nbo5)を入手して、マニュアルにしたがって GAMESS と結合しなければならない。

３．２　結果の見方

まず、最初に注意しておきたいのは、ここで用いた入力データでは、FMO 計算がどのように進んでいくかを見るために、プリントオプション(nprint)を詳細プリントに設定した。これをそのままにして、巨大分子の計算を行うと出力が膨大になってしまうので、プリントレベルを適切に設定して欲しい。ちなみに、通常の ab initio 計算では$CNTRL の nprint キーワードで、出力のレベルを指定するが、FMO では、これに加えて、$FMO の nprint（同じ名前）キーワードでも、プリントレベルの指定があるので、両者を適切に 設定しなければならない(サンプルデータ ~/gamess/tools/fmo/water-16.inp で例を見るか、GAMESS のマニュアルの$CNTRL と$FMO を参照、または、FMOutil で入力データを生成すると、これら標準的な設定の入力データが生成されるので、これを参考にしてもよい)。

（FMO タイトルとオプション）

出力リストの先頭部分（リスト１の FMO のタイトルが出てくるより前）は、$DATA グループで入力した原子から構成される分子についての、通常の ab initio 計算の場合と同じ出力である（FMO では、この入力グループを利用して、基底関数のセットアップのみを行う)。FMO のタイトルに続いて、FMO 計算の様々なオプション値がプリントされる。ほとんどが、default 値を持っているので、入力データで明示的に指定しなくてすむので、これらに無頓着になりがちであるが、一度は見ておかれたい。

リスト１　FMO のタイトルとバージョン番号

```
-------------------------------------------------------------------------

          The Fragment Molecular Orbital (FMO) method.
(一部省略)
                          Version 3.0
```

（モノマー計算）

次いで、“Starting layer 1”がプリントされた以降で、3 つのモノマーに対して順次初期値（initial guess）の計算が行われる。“Running RHF energy for monomer 2”に引き続いてモノマーの SCF 計算の結果がプリントされる。モノマー計算はモノマー 1 から順番に計算されるのではなく、大きなサイズのモノマーから順次計算される（今の場合、2、1、3 の順。これは、後述するように、並列処理の効率を上げるためである。５．２参照)。モノマーは、静電ポテンシャルが自己無同着になるまで繰り返される（SCC 計算)。SCC の途中では、“Iter= 1 iFrag=　　1 EFMO=　　-24.836266728, DD= 0.277367879, DE=**********”が出力される。Iter、iFrag、EFMO、DD は、それぞれ、繰り返し回数、フラグメント（モノマー）番号、モノマーのエネルギー（単位は Hartree)、前回との密度行列の差（第 1 回目は、大きな値になって、フォーマットがあふれて“****”となることが多い)。ここでプリントされるエネルギーは、環境静電ポテンシャルを含まない内部エネルギーである（FMO の論文では E'（プライム付き）と表記されている)。モノマーの SCC 計算が完了すると、“----------Monomer SCF for layer 1 converged in 13 iterations!----------”とプリントされる（今の場合は 13 回で収束した)。

(ダイマー計算)

次いで、“Running RHF dimer　　2　　1, LL1=　26 (MP0, CC=NONE, CI=NONE).”とプリントされた以降、ダイマー計算の出力がある。ダイマーは、それぞれ 1 回きり SCF 計算される。モノマーの場合と同様、大きなサイズのダイマーから順次計算される。 SCF 計算が終わると、“iFrag=　2 jFrag=　1 EFMOu=　-152.132323873 Tr=-0.000483107”とプリントされる。EFMOu（u は uncorrelated、すなわち、HF のエネルギーを意味する）は、ダイマーの内部エネルギーである。Tr は、Tr（$\Delta\mathbf{D}^{IJ}$ ・ $\mathbf{V}^{IJ}$）項の値である。

（FMO プロパティ： One-body）

モノマーとダイマーの計算が終了すると、“FMO properties /　###############”がプリントされて、その後に全エネルギーやペア相互作用エネルギー（PIE）がプリントされる。モノマープロパティのプリント（リスト 2）では、モノマーの内部エネルギー E と DX、DY、DZ でモノマーの双極子モーメントの X、Y、Z 成分がプリントされる。一般に、分子が電

荷を持っている場合、双極子モーメントは無意味になる（一意的に決まらない）ので、注意が必要である。“Euncorr(1)”は、モノマーの内部エネルギーの和で、“Dipole moment”は、x,y,z 成分とノルム（単位は Debye）の順にプリントされている。あくまで、FMO では、特別な場合を除いて、最低でも 2 体（ダイマー）補正まで含めてプロパティを計算するのが標準であるので、ここでプリントされるモノマープロパティはすべて参考データと捉えるべきである。

リスト 2　One-body プロパティ

```
One-body FMO properties.
 ======================

                            E           DX       DY       DZ
   1(frg00001,L1)    -24.779165043  -2.15110   0.81881   0.26983
   2(frg00002,L1)   -112.578397452   1.77299   0.40971   0.39806
   3(frg00003,L1)    -74.965732238  -1.53625  -0.68387  -0.54616

 Total energy of the molecule: Euncorr(1)=     -212.323294732
 Dipole moment D(xyz),DA(1)= -1.9143535   0.5446575  0.1217335    1.9940462
```

(FMO プロパティ：Two-body)

最後にプリントされる 2 体プロパティ（リスト 3）が、FMO（今の例では FMO2）の結果である。“I、J、DL、Z、R、Q(I->J)、E、EIJ-EI-EJ、dDIJ*VIJ、tot”で、“I,J”はフラグメントダイマー IJ を示す。“DL”の“D”は計算法が dynammically correlated 法（MP2 か CI）か否かを示し、“N”（No）は RHF または DFT 計算であることを示す。これが“S”とプリントされ

リスト 3　Two-body プロパティ

```
Two-body FMO properties.
 ======================
 (一部省略)
   I    J DL  Z    R   Q(I->J)        E           EIJ-EI-EJ   dDIJ*VIJ      tot
 ------------------------------------------------------------------------------
   2    1 N1  0   0.00  0.0000   -152.132323873 -14.77476138 -0.00048311   -0.303
   3    1 N1  0   1.68  0.0000    -99.746419200  -0.00152192 -0.00000001   -0.955
   3    2 N1  0   0.66  0.0000   -187.544719002  -0.00058931 -0.00721494   -4.897

 Total energy of the molecule: Euncorr(2)=     -227.107865394

 The backbone energy EBBuncorr(2)=     -227.098539218
 Dipole moment D(xyz),DA(2)=   -2.7239511    0.7052417    0.1381139    2.8171530
```

た場合は esdimer 近似が適用されたことを示す。“L”は、マルチレイヤーのレイヤー番号を示す。“Z”は、モノマー I の電荷と J の電荷の積（これは、ペア相互用エネルギーを見ると

きに合わせてみると便利）である。“R”はモノマー間の最近接原子距離をあらわしたもので、それぞれの原子の vdW 半径の和を 1 として、その何倍離れているかを示している（この距離の計算が行われないオプションを選択した場合は、-1 とプリントされる）。したがって、共有結合が切断されたフラグメント間の距離は“0.0”となる。“Q(I->J)”は、I から J への電荷移動量（Mulliken 電荷で計算）であるが、Mulliken 電荷を計算するというオプションを指定していなければ計算できないので（今の例のように）、“0.0000”とプリントされる。

“E”はダイマーの内部エネルギー（Hartree 単位）（FMO の論文では E_{IJ}' と表記）で、“EIJ-EI-EJ”はダイマーの内部エネルギーからモノマー I と J の内部エネルギーを引いた値（Hartree）で（FMO の論文では E'_{IJ}-E'_{I}-E'_{J} と表記）、“dDIJ*VIJ”は、Tr（$\Delta\mathbf{D}^{IJ}$ ・ $\mathbf{V}^{IJ}$）項である（Hartree）。“tot”は、“EIJ-EI-EJ”と“dDIJ*VIJ”を足したエネルギー（ペア相互作用エネルギー；PIE）で、kcal/mol 単位である。ここで、共有結合しているダイマー（I=2、J=1）の場合は、“dDIJ*VIJ”項のみの値がプリントされていることに注意して欲しい。この理由は、共有結合を切断した場合には“EIJ-EI-EJ”に、BDA の“原子内相互作用”（電子の割り振りに合わせて、炭素原子の核電荷を＋5 と＋1 に分割したので）、大雑把に言えば、（+5/5e）と（+1/1e）の“相互作用エネルギー”が含まれる（I=2、J=1 の場合の-14.77Hartree の大部分）ので、物理的に無意味だからである。すなわち、PIE は、非結合相互作用の解析にのみ使うべきである（だだし、後で説明する EBB としての使い方はできる）。今の例では、モノマー 1 が CH_3、モノマー 2 が CH_2OH なので、C_2H_5OH と H_2O の分子間 PIE の和は、(-0.955)+(-4.897)=-5.852 (kcal/mol)となる。このように FMO 計算では、C_2H_5OH と H_2O 分子との相互作用に、CH_3 と CH_2OH 部分がそれぞれどの程度の寄与をしているか見積もることができる（あまり小さなフラグメントにすると、FMO の誤差が大きくなるので、このような解析の意味がなくなるので注意。どうしても、あるグループを単位として独立にそれぞれの寄与を見積もりたい場合は、FMO3 計算を行って、3-body 相互作用（ペア相互作用のカップリング）まで考察に含めるべきである）。

（FMO プロパティ：全エネルギーと双極子モーメント）

“Euncorr(2)”でプリントされるエネルギーが、FMO2 の全エネルギーで、通常の ab initio MO 計算の結果と比較できるものである。また、“Dipole moment”も同様である。ちなみに、この系の対応する ab initio MO 計算の結果は、全エネルギーが -227.1083045619
で、双極子モーメント（Dx、ｙ、ｚ、DA）は （-2.766892, 0.708072, 0.135749, 2.859280）である。FMO の全エネルギーは-227.107865394 なので、誤差は 0.28 kcal/mol である。双極子モーメント（DA）は 2.8171530 なので、誤差は 0.04 Debye（過小評価）となる。

“EBBuncorr(2)”は、バックボーンエネルギー（EBB）と定義された値で、分子中のモノマーの内部エネルギーと共有結合したダイマーとの“相互作用エネルギー”の和をとったものである。すなわち、このエネルギーと全エネルギーとの差は、全非結合エネルギーになる。このエネルギーは、配座異性体間のエネルギー差を比較するのに使われた。興味ある読者

は、ペア相互作用のエネルギー分割法（pair interaction energy decomposition analysys; PIEDA, D.G.Fedorov et al.,*J. Comp. Chem.*, 28, 222(2007))を参照されたい。

４．フラグメント分割の要点

４．１　フラグメントサイズと分割位置

(1) 一量体のサイズ：目的の精度に依る（下記参照）。蛋白質は通常一または二残基で分割する。その他の場合、10-40 原子をめどにして分割する。溶媒分子の様な結合していない分子は一ないし二分子をまとめて一フラグメントにする。

(2) 空間分割：同じ一量体に遠くにある原子を入れると、近似の適用が無くなるため、計算が遅くなる。なるべく空間的に近接する原子を一塊として分割する。

(3) 化学的考慮：多重又は非局在性を持つ結合を切ると精度が非常に悪くなる。出来れば、C-C の様な一重結合を切る。分割原子をできるだけ共役部分から離す。たとえば、O=C–C–CH-を切るとき、CH のＣを分割原子とする。

(4) 規則性がない分子の分割：蛋白質や DNA ・RNA などは規則性があるため分割は容易であるが、規則性がない巨大・複雑分子の分割は、一定の手順で機械的に分割することはできない。このような系については、分割についての一般的基準はないが、参考として次の注意事項を上げておく。

(5) C–C 結合以外で切る場合：一重結合を切れるなら、特に問題が無い。分割原子は炭素で無ければ、局在軌道を自分で作成する。そのため、~/gamess/tools/fmo/makeLMO.inp が用意してあり、これを実行すれば簡単に作れる。

例：–C=C－SiH2－を－C=C–|SiH2 で切って、分割原子は珪素になる。

(6) 共役系：–CH2=CH2|－で切っても良い。porphirine の様な系で、どうしても二重結合を切る事になれば一量体規模を大きくした方が良い。

(7) 金属性の系 :金属表面や fullerene などのメタリックな系は、本質的に分割できないので、FMO 法が使えない。

４．２　基底関数と精度

(1) 構造最適化には二体展開 FMO 法／3-21G 以上で良い。

(2) ＲＭＳ勾配収束敷居値は 10^{-3}(Hartree/Bohr)であれば一残基で良いが、10^{-4} であれば、精度を確保するために、二残基分割が必要になると思われる（特に長いα-helix が含まれる場合）。

(3) 一点計算は二体展開二残基分割か三体展開一残基分割、6-31G*以上でいい。

(4) 特に長いα-helix が含まれる場合、三体展開一残基分割の方が良い。

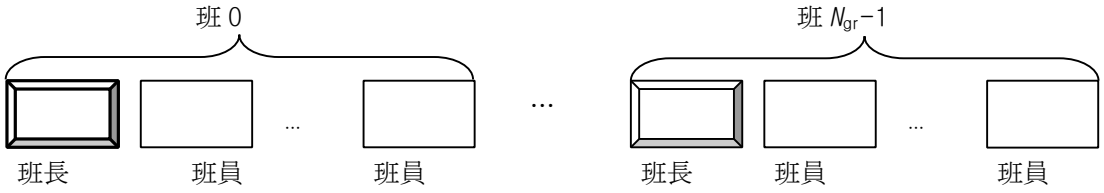
(5) 負電荷を持つグループ（ASP や GLU の側鎖 COO^-）には、diffuse 軌道を追加することを推奨する。

５．FMO の並列計算の仕組み

５．１　CPU の班分け

FMO 法では、2 段階並列化（GDDI）を使う。GDDI では、使用する CPU を班に分け、各班は独立な計算を行う。

(1) 班分け



二段階の並列化

・FMO 法では、一量体と二量体の計算を行う。それぞれの計算で班の数は次のように設定する。

一量体の班数を計算数の三から四分の一以下に設定する

二量体の班数を SCF 計算数の二分の一以下に設定する

$gddi データグループの ngroup に全 node 数を設定する（CPU 数ではない）。

例：一量体数＝10 (入力の$fmo nfrag=10)

二量体数＝40 (=4*nfrag、nfrag(nfrag-1)/2 の一部)

node 数＝16

二段階並列設定　（3 は 16/4= 4 以下、16 は 40/2=20 以下）

$gddi ngroup=16

$fmoprp　　ngrfmo(1)=3,16

(2) 並列計算と分班の要注意点は以下のとおりである。

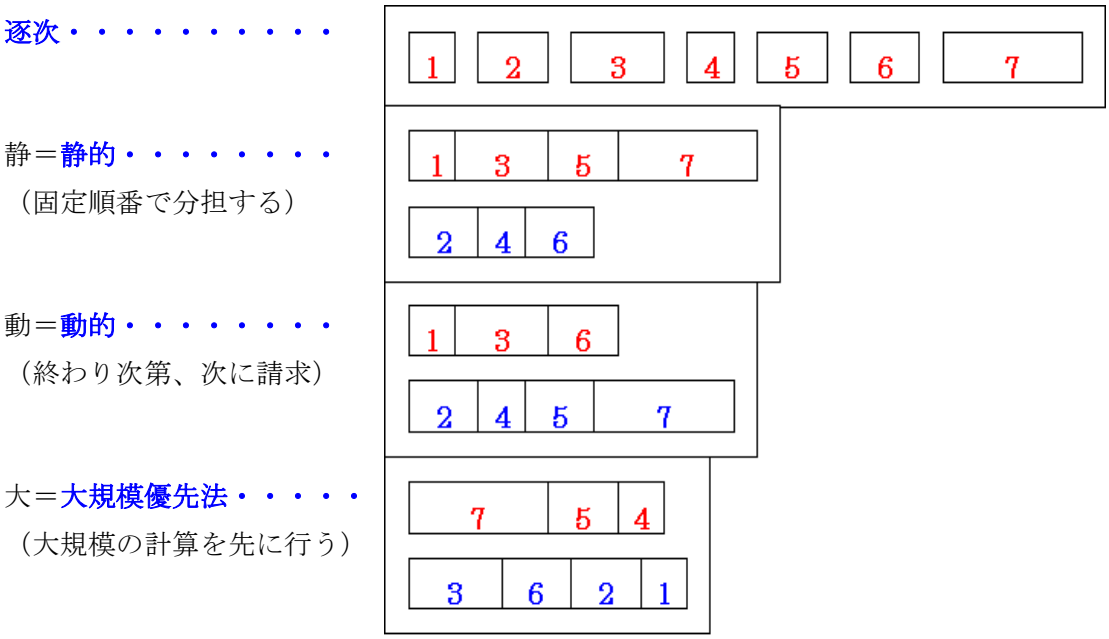
・二量体の計算は静電近似と標準 SCF、二つの種類がある。班数を決める時、SCF 二量体数だけ考慮する（通常 nfrag の 4 倍程度）。

・FMO-MP2 の場合、最低必要な分散メモリ量がある。このメモリ量を確保するため、班数を変える必要がある。例えば、最低分散メモリ量が４G であり、計算機毎に１G の RAM がある場合、一班に最低４台入れねばならない。16 台の場合、最大４班になる。最低分散メモリ量は一と二量体で変わる。

・GAMESS の現バージョンの GDDI ではＳＭＰの CPU を分割して、別々に実行することはできない。（nodefile に node:cpus=2 の代わりに node 名を二回入れるのは不可）。

５．２　並列化と分担法

２CPU で計算する例(ＣＰＵ**0** と **1**)



並列化と分担法と並列化効率の例

A. $(H_2O)_{256}$, 768 原子、6-31G*。

B. $(H_2O)_{1024}$, 3072 原子、STO-3G。

C. lysozyme tri[*N*-acetyl-*D*-glucosamine]錯体,2036 原子,STO-3G.

64 台の 1 GHz PIII, FastEthernet

台数	8			16			32			64		
分担法	動大	静大	静	動大	静大	静	動大	静大	静	動大	静大	静
A 系	1.0016	0.9534	0.9445	0.9845	0.9307	0.8548	0.9510	0.8644	0.8184	0.8885	0.5240	0.6105
B 系	0.9897	0.9753	0.9688	0.9838	0.9548	0.9198	0.9711	0.9373	0.8300	0.9520	0.8853	0.7336
C 系	0.9925	0.9552	0.8790	0.9921	0.9010	0.7074	0.9729	0.8538	0.6793	0.9417	0.7771	0.5876

動大は、ほぼすべての場合で最高並列化効率を示す。

６．FMOutil による入力データ作成　－クランビン－

６．１　プログラムの概要

FMOutil は GAMESS-FMO 計算のための入力データ作成支援プログラムである。このプログラムは、FMO オプションの設定、並列計算での CPU のグループ分け、フラグメント分割などを会話型で行えるので、入力データの作成を練習するのに役立つ。様々な FMO 計算すべてについて入力データを説明すると膨大になりすぎて現実的ではないので、利用者はこのプログラムを使って、いろいろなオプションで入力データを作成し、それらの出力を見て勉強してほしい。このプログラムは GAMESS と一緒に配布されている（~/gamess/tools/fmo/。ソースプログラムは fmoutil.f、マニュアルは fmoutil.txt である）。また、開発者のウェブサイト http://staff.aist.go.jp/　d.g.fedorov/　fmo/main.html からもダウンロードできる。

FMOutil(ver. 2.0) の機能は、

１：GAMESS-FMO の入力データの作成

２：PDB の X 線構造に水素を付加する

３：結合長、結合角、ポリペプチドの 2 面角、水素結合など構造データの計算

４：特定の距離内のアミノ酸残基の数え上げ

で、対象はポリペプチド/タンパク質に限定されるが、GAMESS-FMO 入力データ作成（および簡単な解析）に最低限必要な機能は装備されている。なお、タンパク質と有機分子の複合体の場合のように、非アミノ酸残基から構成される基質についてはフラグメントの分割、水素原子を付加する位置が一意的に確定できないので、ポストエディットが必要。詳細は FMOutil のマニュアル fmoutil.txt を参照のこと。

FMOutil は Fortran 77 で記述された比較的小さなプログラムであり、標準的な f77 コンパイラーで問題なくコンパイルできるはずである。以下のようにコンパイルして、実行ファイルを生成する。

```
f77 -o fmoutil fmoutil.f
```

こうして作成された実行ファイル‘fmoutil’を用いて、以降の作業を進める。

６．２　水素原子の付加

タンパク質の X 線構造データには水素原子の座標が含まれていないのが普通である。したがって、まず、水素原子付加を行なって完全な構造データを作る必要がある。ここでは、例として、CRAMBIN という小型のタンパク質（アミノ酸残基数 46）をとりあげ、Protein Data Bank (PDB) から入手した構造データをもとに、FMO 計算の入力データを作成する手順を説明する。もとになるタンパク質の座標は、PDB の WEB サイト、http://www.rcsb.org/pdb/ から入手する。Keyword に“crambin” と入力すると、登録されている十数種類の構造が表示されるが、ここでは“1CRN”という構造データを用いることにする。PDB ファイルをダウンロードして、“1CRN.pdb” と名付けて保存する。

FMOutil は対話型プログラムである。コマンド、

./fmoutil

を実行すると、正常なら、以下の画面が表示される。

```
 Enter Job # :
   1. Generate FMO input data for GAMESS
   2. Add hydrogen atoms to PDB data
   3. Print geometrical parameters
   4. Print Distances of Residues
   0 or Enter Key. Quit
```

まずは、水素原子を付加するので、”2“ と入力し、画面の案内に従い、入力ファイル“1CRN.pdb” と出力ファイル“1CRN-add-H.pdb”（名前は任意に指定）をそれぞれ入力する。

```
2                                          （ジョブ番号２を入力する）
  Enter Input PDB File(s) :
1CRN.pdb                                   （入力ファイル名を入力する）
  Enter Output File :
1CRN-add-H.pdb                             （出力ファイル名を入力する）
```

ここで、タンパク質の N 末端、C 末端、イオン性残基（ASP/GLU）のイオン化の状態を聞かれる。

```
 2> Protonate N-terminus (1:yes, 2:no)
 2> Protonate C-terminus (1:no, 2:yes)
 2> Protonate ASP and GLU (1:no, 2:yes)
 2> Protonate HIS (1:yes, 2:no)
```

標準的なイオン化状態を仮定するのであれば、すべて“1”と答えればよい。最後のプロンプトで、もし、“2”(no)と答えると、次の入力要求が現れる。

```
 2> Neutral HIS ... H at D1 or E2 (1:D1, 2:E2)
```

これで、出力ファイルに、水素原子を追加した座標データが PDB 形式（一部のデータは省略される）で出力される。

もし、系に水分子が含まれている場合、自動的に水素原子が付加される。非アミノ酸残

基が含まれる場合、これは自動的に水素原子付加ができないので、手動で行わなければならない（マニュアル FMOutil.txt を参照）。

水素原子付加が適切に行われたどうかを確認するため 、メインメニューで“0” を入力して、FMOutil の実行を終了し、出力ファイル “1CRN-add-H.pdb” をエディタで見る。正常に実行されていれば、以下のような座標が出力されているはずである。

リスト４　水素原子付加をした PDB ファイル（一部）

```
REMARK generated with FMOutil
REMARK natm :      642
REMARK nres :       46
REMARK nmol :        1
ATOM      1  N   THR     1      17.047  14.099   3.625
ATOM      2  CA  THR     1      16.967  12.784   4.338
ATOM      3  C   THR     1      15.685  12.755   5.133

                  （途中省略）

ATOM    641 1HD2 ASN    46      13.816   2.909  14.177
ATOM    642 2HD2 ASN    46      13.652   2.929  15.923
TER     643
END
```

PDB からダウンロードしたデータには 327 原子しか含まれていなかったが、ここでは正しく水素が付加され、合計 642 原子の座標が与えられていることを確認（水素付加は任意のモデリングソフトで可能だが、付加した水素の名前やその位置（各アミノ酸残基ごとに、heavy atom の最後に付加する）には注意すること。ここを間違うと、後でアミノ酸単位でフラグメントに分割する際に問題が生じる）。

６．３　入力データの作成

水素原子を付加して、完全な構造データができたので、次に FMO 入力データの作成に入る。再度、FMOutil を実行して、メインメニューで“１”(Generate FMO input data for GAMESS)を選択する。６．２と同様、入力ファイルと出力ファイルを聞いてくるので、水素付加した PDB ファイル“1CRN-add-H.pdb”を入力ファイルとして指定する。出力ファイルは、GAMESS での FMO 計算の入力データになるので、“1CRN-fmo.inp”という名前にしよう。

まず、コンピュータに関するデータが要求される。

```
1> How many computer nodes do you want to use ?
1> How many CPUs does each node have ?
1> How much memory does each node have (in megabytes) ?
```

それぞれ自分が使用するコンピュータのデータを入力する（これらは、適当に答えておいて、後にポストエディットで変更してもよい）。

次に、FMO 計算に関するデータが要求される。

```
1> Choose runtyp (1:energy, 2:gradient, 3:geometry optimization)
```

本練習では、“1”と答える。ここで、構造最適化を選択(番号“3”を入力)したときは、さらに、

```
1> Choose geometry optimization method
   (1:conjugate gradient, 2:simple Hessian update, 3:GMS standard optimizer)
```

と聞いてくるので、2000原子以下であれば、“3”を選択するとよい（2000原子を超える場合は“1”または“2”を選択する）。

続いて、次の入力が促される。

```
1> How many layers do you want to define (1-5) ?
```

本練習では“1”と答える。ここで、レイヤーを“*m*”と指定すると、次の質問が(*m*-1)回現れる。

```
1> Enter fragment numbers to be assigned to layer (2-nlayer).
```

次は、波動関数の選択である。

```
1> Choose wavefunction type (1:RHF, 2:DFT, 3:MP2, 4:CC, 5:MCSCFF)
   for each layer. separated by blanks.
```

本練習では“1”と答える。ここで、“2”（DFT）を選択すると暗黙としてB3RYP汎関数が選択されるので、これ以外のものを使うときは、ポストエディットで修正すること。また、“4”（CC）を選択するとCCSD(T)が選択されるので、必要なら、後で修正すること。“5”（MCSCF）を選択した場合、さらに、次の入力が促される。

```
1> Enter the MCSCF fragment number.
1> Enter the spin multiplicity of the MCSCF fragment.
1> Enter the number of core orbitals, the number of active orbitals
   and number of active electrons.
```

MCSCF計算では、計算する前に初期軌道を準備しておかなければならない。これについては、GAMESSのINPUT.DOCを参照して欲しい。

次は、基底関数である。

```
1> Enter basis set (1:STO-3G, 2:3-21G, 3:6-31G, 4:6-31G*, 5:6-311G*)
   for each layer. separated by blanks.
```

本練習では“1”と答える。STO-3Gを選択した場合、GAMESSでは第2周期の原子で、一部GAUSSIANのものと異なったexponentsが用いられている（GAMESSのREFS.DOC参照）ので、どちらを用いるかを聞いてくるので、それに答える。

“2”より大きな基底関数を指定すると、引き続いて、diffuse関数を追加するかどうかを聞いてくる。

```
1> Do you add diffuse functions on COO- groups ? (1:yes, 2:no)
```

C末、GLUやASPなどのCOO-グループには、diffuse関数を置くことを推奨する。

次は、2体展開FMO（FMO2）か3体展開FMO（FMO3）の選択である。

```
1> Enter the desired n-body expansion (2 or 3) ?
```

本練習では“2”（FMO2）と答える。“3”（FMO3）は、RHF、DFTとCCで可能である。FMO ver.3ではMP2の3体展開も可能になったが、FMOutilのバージョンアップが遅れていて、対応できていないので、ポストエディットで修正してほしい。

次いで、Mulliken電荷をプリントするかどうかを聞いてくる。

```
1> Would you like to print Mulliken charges (1:yes, 2:no)
```

これは、いつでも“1”(yes)とすることを奨める（もちろん、出力量は増える）。ただし、次の電子密度分布のcube fileの作成をするときは、Mulliken電荷のプリントは“2”（no）としなければならない（“2”と答えたときのみ、次のプロンプトがでてくる）。

```
1> Would you like to produce a cube file with the total electron
   density ? (1:no, 2:standard, 3:sparse)
1> Enter grid spacing in Angstrom.
```

ちなみに、これはEnergy計算の場合のみ可能で、構造最適化計算では選択できない（表示されない）。

次は、フラグメント分割に関する指定である。

```
1> Enter fragment size (1:1res-per-frg, 2:2res-per-frg)
1> are S-S bonded CYSs combined to one ? (1:yes, 2:no)
1> is GLY combined to the neighbor ? (1:yes, 2:no)
```

本練習では、“1”、“1”、“1”と答える。プロダクションランでは、ここで、“2”、“1”、“1”を選択することを推奨する。FMOuitl は、ポリペプチド/タンパク質のアミノ酸残基を認識して分割データを生成するが、それ以外の分子は認識できない（水分子は自動的に処理できる）。また、PDB データ中で“TER”区切りがあると、そこから異なった“分子”であると識別する。この分子が、ポリペプチドでない場合、自動的にフラグメント分割できないので、1 つのフラグメントとしてデータを作成する。これを分割したい場合は、ポストエディットしなければならない。最後に、メインメニューで “0” を入力してプログラムを終了し、出力 ファイルの中身“1CRN-fmo.inp” を確認する。すべての作業が正常に終了していれば、出力ファイル “1CRN-fmo.inp” には以下のデータが書き出されているはずである。

リスト 5　クランビンの FMO-RHF/STO-3G の入力データ

```
 $contrl runtyp=energy nprint=-5 $end
 $system mwords=20 memddi=0 $end
 $gddi ngroup=1 $end
!$intgrl nintic=-98000000 $end          （ポストエディットでコメント化）
 $scf dirscf=.t. npunch=0 $end          （ポストエディットでdirscf=.t.に変更）
 $fmoprp
   ngrfmo(1)=1,1,0,0,0,  0,0,0,0,0
   naodir=220
 $end
 $fmo
      nlayer=1                （レイヤーの数）
      nfrag=39                （フラグメントの数）
      icharg(1)= 1, 0, 0, 0, 0,   0, 0, 0, 0, 1,
                 0, 0, 0, 0, 0,   0, 1, 0, 0, 0,        （フラグメントの電荷）
                 0,-1, 0, 0, 0,   0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0,  -1, 0, 0,-1
      frgnam(1)= thr001, thr002, cys003, cys004, pro005,
                 ser006, ile007, val008, ala009, arg010,
                 ser011, asn012, phe013, asn014, val015,
                 cys016, arg017, leu018, pro019, gly020,      （フラグメントの名前）
                 pro022, glu023, ala024, ile025, ala027,
                 thr028, tyr029, thr030, ile033, ile034,
                 ile035, pro036, gly037, thr039, pro041,
                 gly042, tyr044, ala045, asn046
      indat(1)= 0
                  1      2      5    -16      0                    （フラグメントに属
                  3      4     17     18     21    -30      0     する原子の通番号）
                 19     20     31     32     35    -40
                542    543    554    555    558   -563      0
                 33     34     41     42     45    -50
                437    438    442    443    446   -451      0
                              （途中省略）
                566    567    578   -586    589   -596      0
                587    588    597    598    601   -617      0
                599    600    618    619    622   -627      0
                620    621    628   -642      0
 $end
 $fmohyb
  STO-3G        5    5
    1 0  -0.117784   0.542250   0.000000   0.000000   0.850773   （hybrid orbital
    0 1  -0.117787   0.542268   0.802106   0.000000  -0.283585    の係数）
    0 1  -0.117787   0.542268  -0.401053  -0.694646  -0.283585
    0 1  -0.117787   0.542268  -0.401053   0.694646  -0.283585
    0 1   1.003620  -0.015003   0.000000   0.000000   0.000000
```

リスト 5　前頁からの続き

```
 MINI          5   5
   1 0  -0.104883    0.308874    0.000000    0.000000    0.521806
   0 1  -0.104883    0.308874    0.491961    0.000000   -0.173934
   0 1  -0.104883    0.308876   -0.245980   -0.426050   -0.173933
   0 1  -0.104883    0.308876   -0.245980    0.426050   -0.173933
   0 1   0.988209    0.063992    0.000000    0.000000    0.000000
$end
$fmobnd
     -2     3  STO-3G       MINI
    -18    19  STO-3G       MINI
   -541   542  STO-3G       MINI
    -32    33  STO-3G       MINI
          (途中省略)
   -586   587  STO-3G       MINI
   -598   599  STO-3G       MINI
   -619   620  STO-3G       MINI
$end
$data
plant seed protein    1crn
C1
h.1-1    1
      sto 3

c.1-1    6
      sto 3

n.1-1    7
      sto 3

o.1-1    8
      sto 3

s.1-1   16
      sto 3

 $end
$fmoxyz
      1       N           17.04700089       14.09899998        3.62500000
      2       C           16.96699905       12.78400040        4.33799982
      3       C           15.68500042       12.75500011        5.13299990
      4       O           15.26799965       13.82499981        5.59399986
                    (途中省略)
    639       H           11.46599960        5.50899982       13.51599979
    640       H           11.95899963        3.91400003       12.89900017
    641       H           13.81599998        2.90899992       14.17700005
    642       H           13.65200043        2.92899990       15.92300034
 $end
```

(initial guess用 hybrid orbital の係数)

(切断された共有結合の原子対の通し番号とそこに配置する hybrid orbital の名前)

(コメント)

(対称性の記号。FMO では常に C1 (対称性なし))

(原子種の基底関数。系に含まれている全原子種について必要)

(ラベル、元素記号、原子座標 (x、y、z ; オングストローム)

出力の最後には、入力データではないが（GAMESS では無視される）、フラグメント分割に関する情報がまとめてプリントされている。この要約には、フラグメント数、各フラグメントに属する原子数、フラグメント当たりの電荷、フラグメントに与えた名前、などがまとめられている。N、C 末端の電荷、解離性アミノ酸残基の電荷、SS 結合を含むフラグメントに属する原子などが正しく処理されているかどうかをチェックするのに役立つ。これを見て、データに間違いのないことを確認しておくとよいだろう。

リスト6　フラグメント分割の要約

```
;-------------------------------------------------------------------------------
; frg#, #atm, chg,   frg names,                    res#s of the frg
;-------------------------------------------------------------------------------
;    1   14    1     thr001                          1
;    2   14    0     thr002                          2
;    3   20    0     cys003cys040                    3   40
;    4   20    0     cys004cys032                    4   32
;    5   14    0     pro005                          5
;    6   11    0     ser006                          6
           （途中省略）
;   35   14    0     pro041                         41
;   36   19   -1     gly042asp043                   42   43
;   37   21    0     tyr044                         44
;   38   10    0     ala045                         45
;   39   17   -1     asn046                         46
; imol =     1
; total # of atoms =   642
; total charge =   0
;-------------------------------------------------------------------------------
; charge of total system =    0
; s-s bond in the system =    3
; number of each residue in the system
;  gly  ala  val  phe  ile  leu  pro  met  asp  glu  lys  arg  ser  thr
;    4    5    2    1    5    1    5    0    1    1    0    2    2    6
;  tyr  cys  asn  gln  his  trp  ace  nme  hoh  non-peptide
;    2    6    3    0    0    0    0    0    0    0
; fragmentation options: nfgsiz,ifcys,ifgly     1    1    1
```

本練習の場合は問題ないが、特に非アミノ酸から構成される基質などはフラグメント化の対象とはならず、 電荷も初期値として 0 と与えられるので、データのチェックが必要である（非アミノ酸からなる基質でも、もとの PDB データで“TER”で区切られている場合には、１つの独立した分子として認識され、１フラグメントとして切り出される。そこで、後にポストエディットで、分子構造に応じて水素原子を付加し、データを修正すれば、それ程手間をかけずに入力データの作成が可能である）。

７．トラブルシューティング

７．１　メモリのトラブル

おそらく、最も頻繁に経験するのはメモリのトラブルで、メモリが足りないというメッセージがでてジョブがアボートすることだろう。これは、次のような場合によく起こる。

1) 並列 MP2 計算
2) キューブファイルの作成
3) DFT と MCSCF 計算

2)と 3)は簡単に克服できる。通常、Direct SCF を使わない場合、2 電子積分をメモリ上にプールするための容量(nintic)が、たとえば、次のように確保されるている。

```
$intgrl nintic=-98000000 $end
```

これを、たとえば、以下のように適当な量に減らせばよい。

```
$intgrl nintic=-88000000 $end
```

または、$scfデータグループで、dirscf=.t.として、Direct SCFに切り替えれば、このメモリ使用は不要になる。

1）が原因の場合が最も厄介で、FMOutilはMP2計算に必要なメモリ量を推定するが、これを正確に行うのは困難で、多くの場合は誤る。過大に見積もられた場合は、パフォーマンスが悪くなり、過小の場合はメモリ不足でアボートする。対処の方法は、まず、SCF計算の対象となる最大の2量体（距離が離れていて、静電相互作用近似が適用される2量体を除く）のサイズを調べる。この2量体の基底関数の数を計算する。もし、アボートしたジョブの出力があるときは、“Max AOs per frg:　　150”で最大のフラグメントの基底関数の数が分かるので、これを2倍すればよい。そして、このサイズのMP2計算にはどれだけのメモリが必要で、それを確保するためにGDDIでいくつのノードが必要かを推定する。たとえば、$SYSTEMデータグループが以下であるとする。

```
$system mwords=25 memddi=1498 $end
$gddi ngroup=16 $end
$fmoprp
   ngrfmo(1)=3,1,0,0,0,   0,0,0,0,0
$end
```

このデータは､16台のノードでDDIメモリは1498MW（メガワード)(1ノードあたり1498/16=94 MW)であることを意味している。3グループ（6+5+5）に分けてモノマー計算をすると、それぞれのグループは、6*94, 5*94, 5*94 MWのDDIメモリとなる。この設定で、150基底のモノマー計算でジョブがアボートした場合、出力中（マスターだけではなく、スレーブに残された出力も見る）の“Running RHF energy for monomer/dimer”を見つけて、必要なメモリ量を調べる。そして、GDDIグループの数を減らしてみる。

```
      ngrfmo(1)=2,1,0,0,0,   0,0,0,0,0
```

ngrfmo の1つ目のデータはモノマー計算時のグループ数の指定なので、この場合、２つのグ

ループとなり、それぞれのメモリ量は、8*94 MWとなる。同様にして、ダイマーについても、GDDIグループの数（ngrfmoの2つ目の値）を変える。今の場合、すでに“1”（すべてのノード）なので、これでメモリ不足が起きれば、ノードをもっと増やすしかない。

もし、経験や文献から得た知識で、MP2計算に必要なメモリが正しく推定できるとよいが、そうでない場合は、事前にいくつかの小さな分子のab initio MP2ジョブを実行して（$CNTRLデータグループでEXETYP=CHECKとすると実際計算は行われないので計算時間が節約できる）、分子サイズと必要なメモリ量の関係を学習すればよい。最後に、MP2のエネルギー勾配計算は、エネルギー計算に比べて数倍のメモリ量がいることに注意してほしい。GAMESSのMP2ルーチンは、逐次計算用と並列計算用で別々のコードになっている。もし、GDDIグループが 1 であれば、逐次計算ルーチンが実行され、この場合上記のメモリ不足は起こらない。

ある場合には、　ジョブを2つに分けることでメモリの問題を回避できる。たとえば、cube fileの作成ジョブであれば、MP2計算は必要ない（cube fileはRHF電子密度のみ可）ので、まず、RHF計算でcube fileを作成し、別のジョブでMP2計算を行えばよい（計算時間は少々無駄になるが）。

７．２　基底関数の問題

計算しようとする系が有機分子以外の場合、ある原子の基底関数が GAMESS に内臓されていないことがある。このような場合には、以下に示す例のように、$DATA データグループで明示的に基底関数を入力する必要がある。

```
$data
<Enter your job title here>
C1
U.1-1       92
s 26
1 1000000 1
...

c.1-1       6
          n21 3

$end
```

これらを入力するには、Gausian関数の指数と縮約係数を知っていなければならない。これらは、基底関数のデータベース（たとえば、https://bse.pnl.gov/bse/portal）で検索して目的の原子の目的の基底関数を入手し、入力データの形式に編集して用いればよい。

附録1　GAMESSの入手、コンパイルからテスト計算まで

１．GAMESS の入手と実行環境の設定

・GAMESS はいろいろなプラットフォームで動くが、本稿では、OS として RedHat 8.0, 9, Fedora Core を用いる場合に限定する。その他のプラットフォームについては、GAMESS のマニュアルを参照されたい。RedHat 系の OS は、http://fedora.redhat.com/download から download できる。マニュアルに従ってインストールする。

・BLAS library

BLAS は無くても GAMESS の実行はできるが、これを使うと計算が多少速くなる。BLAS は、

ftp://ftp.dulug.duke.edu/pub/redhat/linux/8.0/en/os/i386/RedHat/RPMS/blas-3.0-18.i386.rpm　から入手可能。

・compiler (C+FORTRAN)

Linux 標準の gcc と g77 を使用できるが、Intel のコンパイラー（無償）の方が二割程位速い。これは、http://www.intel.com/software/products/compilers/flin/noncom.htm　から入手可能。

・GAMESS の source

GAMESS のソースコードは、http://www.msg.ameslab.gov/GAMESS/GAMESS.html から入手する。

２．GAMESS のお膳立て

(1) その（一）root 権利で行う作業

・分散メモリ設定

root 権利で/etc/sysctl.conf に次の一行を追加する。

```
kernel.shmmax = 1073741824
```

これはＲＡＭが１Ｇ(=2^{30} byte)の場合の例であるが、ここには 1 CPU 当たりではなく、node の全メモリ量を設定する。この設定を各計算機で一回行って、再起動する。

・BLAS のインストール

```
rpm -i blas-3.0-18.i386.rpm
```

でインストールできる。

・GAMESS で使う work directory の作成

GAMESS が中間データを一時格納するための work directry を設定する。各計算機にすべて同じ設定をする。このディスクの容量は４０GB 以上あれば十分である。

```
mkdir /work/users/myname ; chown user:group /work/users/myname
```

(2) その（二）（命令形式は tcsh/csh である)

・ GAMESS の source などが格納された圧縮ファイルを展開する。

```
cd ~
tar -xzvf gamess.tar.Z
```

・ DDI の compile

games/ddi ディレクトリにある、添付のスクリプト ddicomp を一部修正してから、DDI 関係のソースプログラムをコンパイルする。

```
cd gamess/ddi
vi compddi
```

set TARGET = ibm32 を set TARGET = linux-pc に変更

```
./compddi
mv ddikick.x ..
cd ..
```

(3) その（三）

・プリプロセッサー（actvate）のコンパイル

GAMESS のソースプログラム（xxx.src）は fortran のソースプログラムではないので、そのままコンパイルはできない。プリプロセッサーで、システムに依存する部分などの処理をして、fortran のソースプログラムを生成する仕組みになっている。そのため、まず、プリプロセッサーである actvate を作成する。/games/tools　directory にある actvate.code を以下のように一部修正して、actvate.f　を作り、これをコンパイル・リンクする。

```
cd ~/games/tools
sed "s/^[*]UNX/        /g" <actvte.code   >actvte.f
f77 -o actvte.x actvte.f
cd ..
```

ここで、「　」は空白を示す

・ GAMESS の compile

コンパイルとリンクのためのスクリプトファイル（comp、compall と lked）を次のように修正する。

```
vi  comp  compall  lked
```

set TARGET=ibm32 を set TARGET=linux-pc へ変更

chdir /u1/mike/gamess を chdir ~/gamess へ変更

(4) お膳立て（四）

・ GAMESS を compile する。

./compall >& compall.log &

tail -f compall.log で完了確認の上、次に進む

・GAMESS を link する。

./lked >& lked.log

３．実行準備

・結果を格納する directory を作成する。

mkdir ~/scr

・実行用の script(~/gamess/rungms)を修正する。

ジョブスケジュールシステム（queuing system）が無いシステムを使う場合の例。

vi ~/gamess/rungms

set SCR=/scr/$USER を set SCR=/work/users/$USER へ変更

if ($os == Linux) set GMSPATH=/cu/mike/gamess を

if ($os == Linux) set GMSPATH=~/gamess へ変更

・並列計算の準備

~/gamess/rungms を修正する

vi ~/gamess/rungms

```
#
#       2. This is an example of how to run on a 8-way SMP enclosure,
```

の前に下記の if～endif を挿入する。

```
#
#  general parallel execution based on a provided host file
#
 if( -e $NCPUS ) then
    set HOSTLIST=(`cat $NCPUS`)
    set NNODES=`wc -l <$NCPUS`
    set NSMP=1
    @  NCPUS  =  $NNODES  *  $NSMP
 endif
```

４．テスト計算

・使用する計算機名を書き込んだファイル(nodefile)を作成する。

```
cd ~/gamess/tests
vi nodefile
        fmoxxx1
        fmoxxx2
        ・・・
```

の様に一行毎に１つの計算機名を書き込む。ＳＭＰの場合、fmoxxx1:cpus=2 等として入れて、~/gamess/rungms の中の set NSMP=1 を set NSMP=2 に変更する。

実行する前、nodefile に記述した計算機で rsh が実行できるように設定しておかないと計算が出来ない。

・テスト計算の実行

```
foreach i ( `ls *.inp` )
~/gamess/rungms $i:r 00 nodefile >&$i:r.log
end
```

＊よくあるトラブル

・実行する時、…log が既に存在すると云う警告が出て止まってしまう。

GAMESS の実行スクリプト（rungms）は、ジョブ名（入力ファイル xxx.inp の xxx）として、不用意に出力ファイルの上書きされるのを防ぐために、重要なファイルが存在するかどうかチェックしている。そのひとつが、ジョブ名.log ファイルで、これがあると実行しない設定になっている。

解決：実行する前、ジョブ名.log ファイルを削除する

・実行したら、…dat 又は…irc が既に存在すると云う警告が出て止まってしまう。

解決：計算する前、~/scr/ジョブ名.dat または~/scr/ジョブ名.irc を削除する

５．入力データ形式

GAMESS の入力データは、次のようにグループ化されている。

（例）

```
 $CONTRL  SCFTYP=RHF  RUNTYP=ENERGY  $END
 $SYSTEM  TIMLIM=2  MEMORY=100000  $END
 $BASIS  GBASIS=STO  NGAUSS=3  $END
 $DATA
C2H5OH+H2O
C1
 O           8.0  -2.4493614824   0.5180105259   0.0102319306
```

```
H           1.0   -2.9309841137    0.6564728575   -0.8399969145
H           1.0   -3.0583517680   -0.1059613981    0.4726454459
$end
```

- ✓ 入力を項目（group）に分ける。
- ✓ 各項目は＄CONTRL　・・・　$END で構成される。項目名の$の前には必ず一つの空白を入れる。
- ✓ 項目名は 6 桁。不足分は空白を加える。（「$BASIS」→「$BASIS　」）
- ✓ 項目の中で変数を設定する（例：SCFTYP=RHF）。
- ✓ 行列要素を$GUESS IORDER(20)=1,10 の様に入れる。この例では、iorder(20)=1、iorder(21)=10 だけを設定している。他の要素は default 値になる。
- ✓ 入力の行長は８０カラムまで。
- ✓ 長い項目を 2 行等に分ける。例：
  $guess iorder(11)=1,10,11,12,13,
  14,15,16 norder(1)=1 $end
- ✓ 大文字と小文字は同一視される。
- ✓ たいていの変数には初期値が設定されている。$DATA の様に初期値が設定されていない項目は、必ず入力しなければならない。
- ✓ 存在しない項目があれば無視される。例えば、間違えて、$SYSTEM の代わりに $SYSTEN を入力すると、$SYSTEN の中身を全て無視される。
- ✓ 項目は必要な時に読み込まれる。そのため、MP2 の項目を間違ったら、RHF が終わるまで、間違いが見付からない。実際に計算を行う前、入力を確かめるため、$CONTRL EXETYP=CHECK　$END で実行することを薦める。
- ✓ 変数形式は一般的には自由。浮動小数点変数を 0.001 又は 1.0e-3 で、論理変数を.true. 又は.t.で入力可。但し、幾つか不自由な項目もあり(例：$VEC)、決められた書式（固定フォーマット）に従わないと実行が停まってしまう。