Amber 利用の手引き

TSUBAME Computing Services, Center for Information Infrastructure

2025-01-31

Table of contents

1. はじめに

本書は、AmberをTSUBAME4.0で利用する方法について説明しています。 また、TSUBAME4.0 を利用するにあたっては、「TSUBAME4.0利用の手引き」もご覧下さい。 利用環境や注意事項などが詳細に記述されております。
Amberの開発元ではAmberに関するWebページを公開しています。次のアドレスを参照してください。

https://ambermd.org/

また、コンフレックス株式会社のAmberのページは次の通りです。

https://www.conflex.co.jp/prod_amber.html

Info

TSUBAME4.0にインストール済みのAmberは、学内利用者および学術機関に所属する利用者に限り使用可能です。Amberの利用には別途アプリケーション利用料が必要になります。詳細は利用料の概略のアプリケーション (**TSUBAME4.0**で一部有償化)をご覧下さい。
Amberのライセンス改定(2023.4.15)に伴い非営利、政府機関、学術機関のライセンス費用は0ドルとなっており、各自でAmber公式サイトのGetting AmberXX for non-commerical useからダウンロード & コンパイルしてご利用いただくことも可能です。
一方で、TSUBAME4.0のAmberにおいてはスパコンライセンスのため無償ではなく、サポート契約も行っております。そのため、TSUBAME4.0においてコンパイル済みのバイナリやサポートの利用を希望する場合はTSUBAME4.0ポータルのアプリケーション利用設定よりAmberの利用権を個別にご購入ください。

1.1. 利用できるバージョン

TSUBAME4で利用可能な最新バージョンについてはTSUBAME計算サービスWebサイトの サポートされているアプリケーション ページをご確認下さい。
研究に支障がない限り、バグ修正の入っている最新版をご利用下さい。

1.2. 概要

Amberは本来タンパク質・核酸の分子動力学計算のために開発されたプログラムですが、最近では糖用のパラメータも開発され、化学・生物系の研究のために益々有用なツールとなってきました。ご自分の研究で利用する場合は、マニュアルや関連する論文等の使用例をよく調べて、Amberが採用しているモデルや理論の限界、応用範囲等を把握しておくことが必要です。現在、Amberはソースコードを無制限にコピーすることはできませんが、東京科学大内部で利用することは可能なので、これを基にさらに発展した手法を取り込むことも可能です。

Amberの主なプログラムを以下に示します。

1.2.1. モデル作成

LEaP
Amberの入力データの作成を簡便にするためのプログラムで、以下のprep、link、edit、parmの機能を備えています。

PREP
Amberのデータベースには20個のアミノ酸残基および核酸の構成ユニットに対するトポロジーファイルが予め用意されています。 このデータベースにユーザが作成したトポロジーファイルを読み込み、それを解釈し追加することが可能です。

LINK
AmberデータベースやPREPによって用意された残基トポロジーを使用して、分子の一次構造を決定するモジュールです。 たとえ分子がただ1つの残基で構成されている場合でも、このステップを通過する必要があります。

EDIT
EDITには以下に示す3つの機能があります。LINKで作成したトポロジーの残基-残基間の二面角を、PDB(Protein Data Bank)フォーマットに変換します。

LINKで作成されたファイルをCartesian座標などに変換します。 溶質分子の周りに水分子を配置することにより水溶液のシミュレーションの初期データを作成します。

PARM
分子力学パラメータ(平衡結合長、平衡結合角、力の定数、ファン・デル・ワールス力およびクーロン力など)を各自由度および原子対に割り当てるモジュールです。 自分の扱う分子のパラメータがAmberのオリジナルデータベースにない場合は、新規にパラメータを作成して(あるいはどこか別の力場から捜し出して)プログラムに教えてあげる必要があります。

1.2.2. 計算処理

SANDER

エネルギー極小化や分子動力学計算を行うモジュールです。後者では、周期境界条件、拘束動力学(SHAKE法)などが使えるのはもちろんのこと、アンサンブル(NVE, NVT, NPT)の指定も行えます。また、Amber 4.1からは、Ewald法を用いた計算が可能です。(水分子は電気的な永久双極子を持っているため、分子間の静電的相互作用は遠距離まで及びます。基本セルのサイズは、この相互作用をカットオフするにはあまりにも小さいため、何らかの工夫が必要となります。その一つがEwald法です。周期境界条件の下では、系全体を見ると基本セルの電荷分布を単位に、イオン結晶のように電荷が配列しています。Ewald法では、ポテンシャルエネルギーを基本セル内での誤差関数の和とフーリエ級数(逆格子空間での和)の二種類の項で表現し、イオン結晶でのポテンシャルエネルギーの和を効率よく計算しています)。 SANDERではこの他にNMR-NOEデータをリファインする機能がオプションとして用意されています。

PMEMD

sanderの計算速度を大幅に改良したものです。並列化効率も大きく向上しています。 Xeon PhiやGPUがサポートされています。

GIBBS

2つの状態間の自由エネルギー差を計算するモジュールです。自由エネルギー摂動法(Free Energy Perturbation)を初めとする5つのオプションが用意されています。

NMODE

エネルギーの核座標に関する一次、二次微分の計算から遷移状態の探索や基準振動解析を行うモジュールです。

ROAR

機能を拡張した"Penn State"版のsanderです。主な拡張は、システムの一部を量子力学的に定義することができる点です。その他、Nose-Hooverの連鎖MD積分法、Ewald法、multiple-time-scale積分法を導入しています。

その他
データ解析のためのモジュールANAL、CARNAL、RDPARM、NMANAL/LMANAL等があります。Amberの主なモジュールは上記の通りですが、各モジュールを実行するためには様々な計算条件パラメータ(例えばセルの大きさ、クーロン力のカットオフ距離、シミュレーション時間等)や計算ルートを決めるオプションを指定しなければなりません。 これらについてはマニュアルや、Amberに関する研究論文を参照して下さい。

1.3. マニュアル

Amber Reference Manuals　ambermd.org

2. Amberの使用方法

Info

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

2.1. Amberの実行

2.1.1. インタラクティブ実行

ログイン方法を参考にログインノードにログイン後、インタラクティブノードを利用したX転送を参考にノードをX転送付きで確保するか、Open OnDemandを利用して計算ノードにログインしてください。
以下以降の例では、全て計算ノードにログインした状態で行います。

2.1.1.1. CUI実行

以下はあくまでもコマンドサンプルです。実際の計算にはmdin,prmtop,inpcrdなどのインプットファイルや初期パラメータを記載したファイルが必要となります。

(1) インタラクティブでの逐次処理の場合の利用手順を以下に示します。

```
[rNnN]$ cd <利用したいディレクトリ>
[rNnN]$ module load amber
[rNnN]$ sander [-O|A] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

(2) インタラクティブでの並列処理(sander.MPI)の場合の利用手順を以下に示します。

```
[rNnN]$ cd <利用したいディレクトリ>
[rNnN]$ module load amber
[rNnN]$ mpirun -np [並列数] -x LD_LIBRARY_PATH sander.MPI [-O|A] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

(3) インタラクティブでのGPU逐次処理(pmemd.cuda)の場合の利用手順を以下に示します。

```
[rNnN]$ cd <利用したいディレクトリ>
[rNnN]$ module load amber
[rNnN]$ pmemd.cuda [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

(4) インタラクティブでのGPU並列処理(pmemd.cuda.MPI)の場合の利用手順を以下に示します。

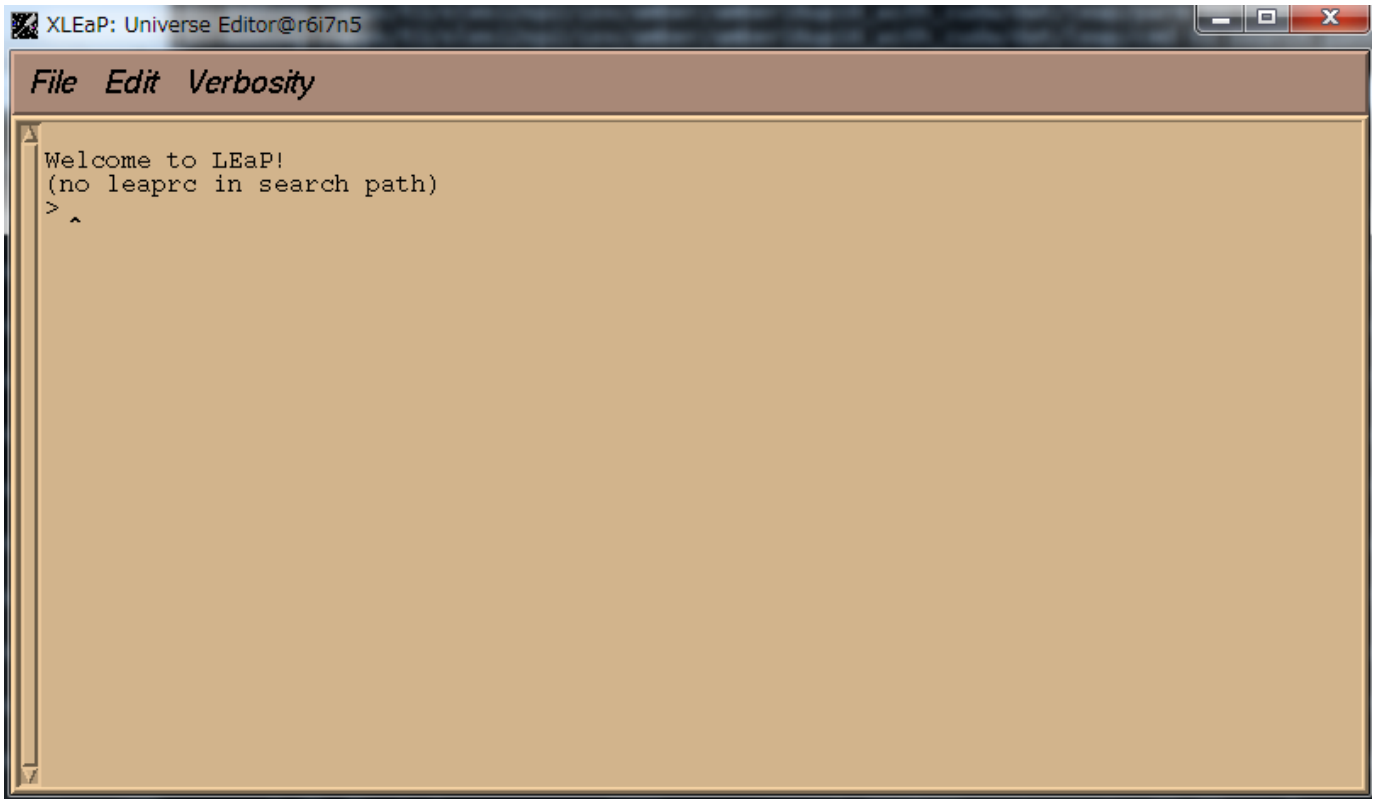
```
[rNnN]$ cd <利用したいディレクトリ>
[rNnN]$ module load amber
[rNnN]$ mpirun -np [並列数] -x LD_LIBRARY_PATH pmemd.cuda.MPI [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

2.1.1.2. GUI実行

例では2時間接続で、割り当てノードとしてr1n11が割り当てられた場合を想定しております。
割り当てノードはコマンド実行時に空いているノードですので、明示的にノードを指定することはできません。

qrshの実行

```
[login]$ qrsh -g [TSUBAMEグループ] -l s_core=1 -l h_rt=2:00:00
[rNnN]$ module load amber
[rNnN]$ xleap
-I: Adding /apps/t4/rhel9/isv/amber/22up05_ambertools23up06/dat/leap/prep to search path.
-I: Adding /apps/t4/rhel9/isv/amber/22up05_ambertools23up06/dat/leap/lib to search path.
-I: Adding /apps/t4/rhel9/isv/amber/22up05_ambertools23up06/dat/leap/parm to search path.
-I: Adding /apps/t4/rhel9/isv/amber/22up05_ambertools23up06/dat/leap/cmd to search path.
```



xleapの画面

2.1.2. バッチジョブスケジューラーAGEによる実行

以下はあくまでもコマンドサンプルです。 実際の計算にはprmtop,inpcrdなどの初期パラメータを記載したファイルが必要となります。
バッチキューシステムの場合の利用手順を以下に示します。

```
[login]$ cd <利用したいディレクトリ>
[login]$ qsub parallel.sh        # parallel.shを利用する場合
```

スクリプト例　CPU並列処理

```
#!/bin/bash
#プライオリティ
#$ -p -5
#実行ディレクトリ：カレントディレクトリ
#$ -cwd
#$ -N amber_parallel_test_job #job名
#送信先メールアドレス
#$ -M ambertest[at]o.cc.titech.ac.jp
#エラーメッセージファイル名、設定なしだとスクリプト.e.JOBID
#$ -e uge.err
#標準出力ファイル名、設定なしだとスクリプト.o.JOBID
#$ -o uge.out
#*必須：資源タイプの指定
#$ -l node_h=2
#*必須：時間指定
#$ -l h_rt=0:10:00
#$ -V

#CPU数の設定と設定の出力
export NSLOTS=28
echo Running on host `hostname`
echo "UGE job id: $JOB_ID"
echo Time is `date`
```

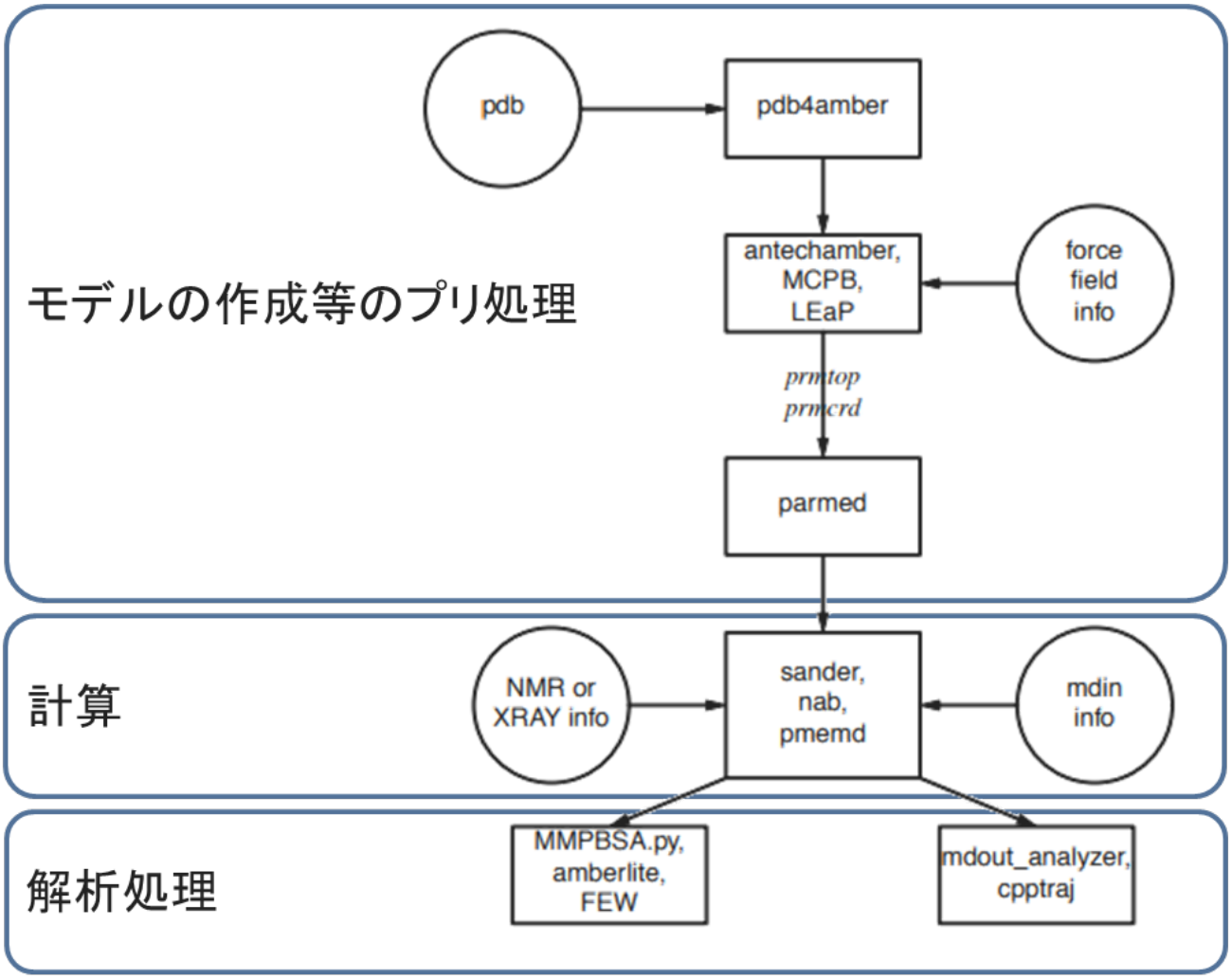
```
echo Directory is `pwd`
echo This job runs on the following processors:
echo This job has allocated $NSLOTS processors


#利用ファイルの指定
in=./mdin
out=./mdout_para
inpcrd=./inpcrd
top=./top

#インプットファイルをスクリプト内に書き込む場合の処理
cat <<eof > $in
 Relaxtion of trip cage using
&cntrl
  imin=1,maxcyc=5000,irest=0, ntx=1,
  nstlim=10, dt=0.001,
  ntc=1, ntf=1, ioutfm=1
  ntt=9, tautp=0.5,
  tempi=298.0, temp0=298.0,
  ntpr=1, ntwx=20,
  ntb=0, igb=8,
  nkija=3, gamma_ln=0.01,
  cut=999.0,rgbmax=999.0,
  idistr=0
 /
eof

#モジュールの呼び出し
module load amber

#sander.mpiの実行
mpirun -np $NSLOTS -x LD_LIBRARY_PATH \
sander.MPI -O -i $in -c $inpcrd -p $top -o $out < /dev/null

#不必要なファイルの削除
/bin/rm -f $in restrt
```

スクリプト例　GPU並列処理

```
#!/bin/bash
#$ -p -5
#$ -cwd
#$ -N amber_cuda_parallel_test_job
#$ -m e
#$ -e uge.err
#$ -o uge.out
#$ -l node_h=2
#$ -l h_rt=0:30:0
#$ -V

export NSLOTS=8
echo Running on host `hostname`
echo "UGE job id: $JOB_ID"
echo Time is `date`
echo Directory is `pwd`
echo This job runs on the following GPUs:
echo This job has allocated $NSLOTS GPUs

in=./mdin
out=./mdout
inpcrd=./inpcrd
top=./top

cat <<eof > $in
FIX (active) full dynamics ( constraint dynamics: constant volume)
&cntrl
   ntx = 7,       irest = 1,
   ntpr = 100,     ntwx = 0,     ntwr = 0,
   ntf = 2,       ntc = 2,       tol = 0.000001,
   cut = 8.0,
   nstlim = 500,  dt = 0.00150,
   nscm = 250,
   ntt = 0,
   lastist = 4000000,
   lastrst = 6000000,
 /
eof

module load amber

mpirun -np $NSLOTS -x LD_LIBRARY_PATH \
pmemd.cuda.MPI -O -i $in -c $inpcrd -p $top -o $out < /dev/null

/bin/rm -f $in restrt
```

3. Amberの計算の流れ

Amberの計算を行う場合の、基本的な作業の流れを以下に示します。 インタラクティブノード上でモデルの作成等のプリ処理を行い、計算、解析処理はAGEでのジョブ投入や、インタラクティブノードで行うことができます。



4. 分子の構築方法

Info

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

4.1. Leap

LEaPはAmber分子動力学プログラムのインプットファイルを準備するモジュールです。 LEaPの名前の由来は、AmberのモジュールのPREP-LINK-EDIT-PARMの頭文字(LINK、EDIT and PARM)からきています。
LEaPにはtLEaPとxLEaPの二種類があり、そのうちxLEaPはX-Windowsのグラフィカル・ユーザインタフェースを利用したもので、大変簡単に分子の構築が行えます。

4.2. 入力データの例

4.2.1. コマンド

LEaPの核はコマンドラインインタフェースであり、コマンドによってオブジェクトを操作します。 LEaPコマンドのフォーマットは以下の　種類があります。

```
command argument1 argument2 argument3 ...
variable = command argument1 argument2 ...
```

ここでargumentはNUMBERs、STRINGs、LISTsなどのObjectsです。commandについては4.5節で説明します。

4.2.2. Objects

ObjectsとはLEaPの基本的な要素です。
ObjectsにはNUMBERsやSTRINGsのSimple ObjectsとUNITs、RESIDUEs、ATOMsのComplex Objectsがあります。 Complex Objectsとは複数のSimple Objectsを含んだObjectです。 例えば、RESIDUEsはATOMsとresidue name、connect atom、residue typeを含んだComplex Objectsです。 一番大きなObjectsはUNITsであり、Molecular DynamicsのInputFileの基となります。
つまりLEaPは、いくつかの小さなObjectsを組み合わせ目的のUNITsを作っていくオブジェクト指向の分子構築ソフトです。 以下にObjectsの簡単な説明を載せます。(詳細はマニュアルを参照して下さい。)

Simple Objects

項目	説明
Numbers	倍精度の数値。
STRINGs	文字列。スペースが間に入るときはダブルクォーテーションでくくる。
LISTs	一連のObjectsで次のように使用する。{ 1 2 3 }
PARAMETSETs	Amber力場のパラメータのセット(bond、angle、torsion、nonbonded parameter)

Complex Objects

項目	説明
ATOMS	name(STRING)、type(STRING)、charge(NUMBER)、position(NUMBER)などのObjectsを含んだComplex Objects。
RESIDUEs	複数のATOMs Objectsとconnect1 sim 5(結合の情報)、restype、nameのObjectsを含むComplex Objects
UNITs	RESIDUEsにPARAMSETsを加えたComplex Objectsで最も重要なObjects。

4.2.3. Complex ObjectsからSub objectsへのアクセス方法

あるObjectsからその中に含まれるObjectsにアクセスする時にはdot'.'使います。
例えばdipeptideというファイル名の付いたジペプチドALA-PHEにおいて、ALAをアクセスし、descコマンドで情報を引き出す場合には、以下の2通りのアクセス方法があります。

```
desc dipeptide.ALA (dipeptideのALAという名前のREDIDUEs)
又は
desc dipeptide.1(dipeptideの１番目のREDIDUEs)
```

同様に、ALAの3番目の原子のCAへアクセスする時には以下のように行います。

```
> desc dipeptide.ALA.CA
又は
> desc dipeptide.ALA.3
又は
> desc dipeptide.1.CA
又は
> desc dipeptide.1.3
```

4.2.4. Variables

Objectsを操作するために付けた名前をVariablesといいます。 Variablesはlistコマンドで表示できます。

4.3. xLEaPの使用方法

xLEaPはLEaPのX-Windowsバージョンです。 ユーザインタフェースを通じてモデルの構築が行えます。 本項ではDNAを例に説明します。 moduleコマンドにてAmber環境を読み込んでいる、という前提条件にて説明いたします。

4.3.1. データの準備

ディレクトリを作成します。

```
[login/rNnN]$ mkdir amber
[login/rNnN]$ cd amber
```

pdbファイルを作成するためのnabファイルを作成します。

```
[login/rNnN]$ vi nuc.nab
#以下の内容を記載
molecule m;
m = fd_helix( "abdna", "aaaaaaaaaa", "dna" );
putpdb( "nuc.pdb", m, "-wwpdb");
```

pdbファイルを作成します。

```
[rNnN]$ nab nuc.nab
[rNnN]$./a.out
[rNnN]$ ls -la
total 17736
drwxr-xr-x 2 XXXXXXX tsubame-users      512 Sep  1 09:27 .
drwxr-xr-x 9 XXXXXXX tsubame-users     4096 Sep  5 09:25 ..
-rwxr-xr-x 1 XXXXXXX tsubame-users 18000464 Sep  5 09:26 a.out
-rw-r--r-- 1 XXXXXXX tsubame-users      573 Sep  5 09:26 nuc.c
-rw-r--r-- 1 XXXXXXX tsubame-users       93 Sep  5 09:26 nuc.nab
```

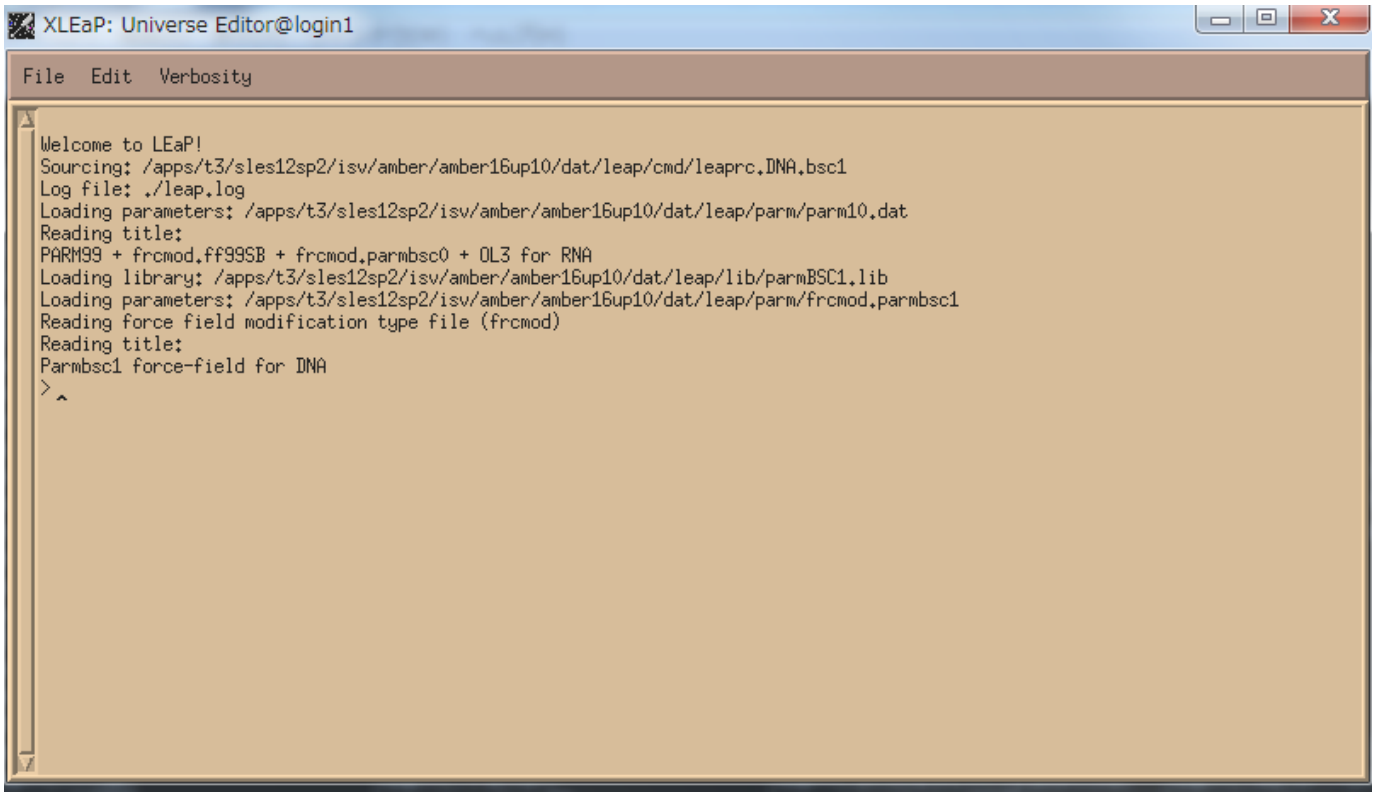
```
-rw-r--r-- 1 XXXXXXX tsubame-users    51686 Sep  5 09:27 nuc.pdb
-rw-r--r-- 1 XXXXXXX tsubame-users     2165 Sep  5 09:27 tleap.out
```

4.3.2. xLEaPの起動

xLEaPを起動するためにはサーバ上でxleapコマンドを実行します。

```
[rNnN]$ xleap -s -f <filename>
<filename>は、$AMBERHOME/dat/leap/cmd/下のファイル等を指定してください。
[rNnN]$ xleap -s -f /apps/t4/rhel9/isv/amber/22up05_ambertools23up06/dat/leap/cmd/leaprc.DNA.bsc1
```

xleapコマンドを実行するとUniverse Editorウィンドウが表示されます。



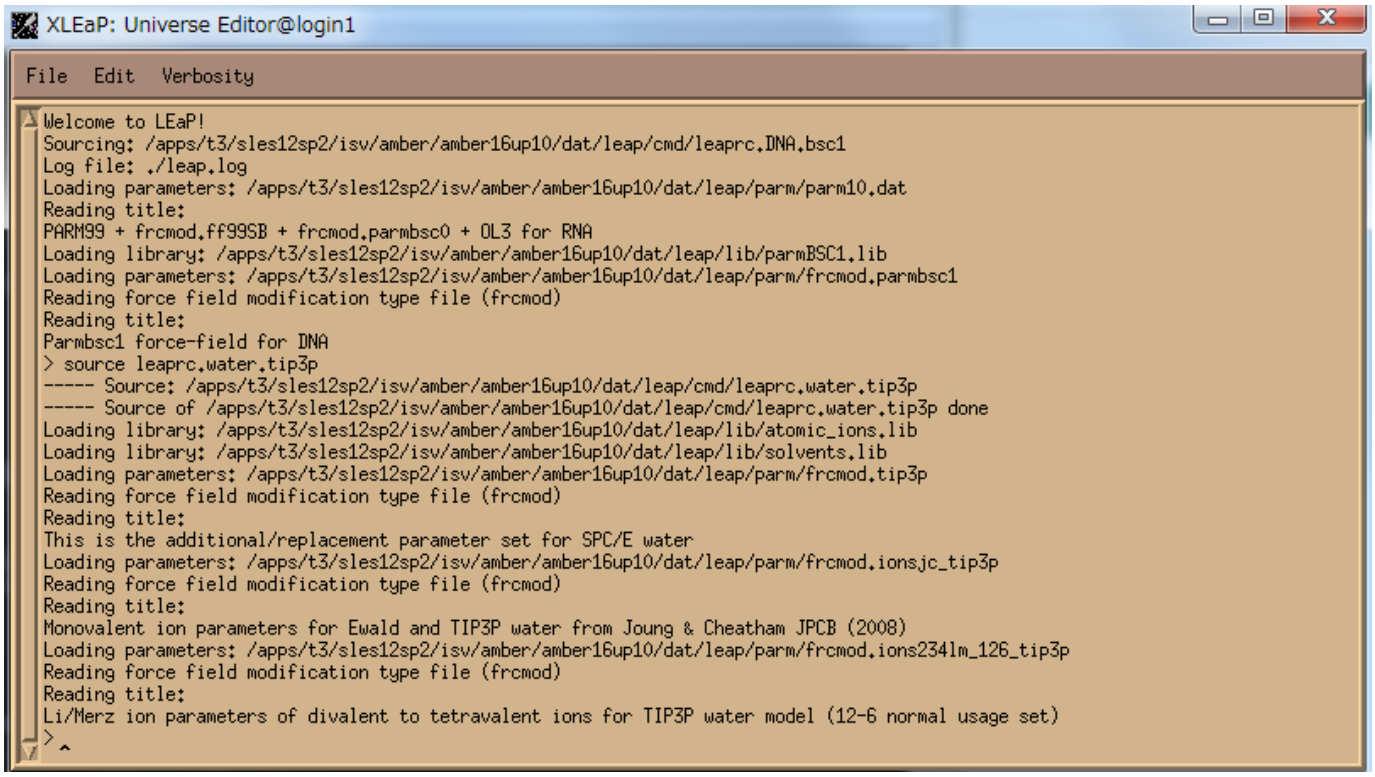
xLEaPの起動時のオプションを以下に示します。

```
-h start-up optionsのリストの表示
-I {dir} サーチするディレクトリの追加
-f {file} スタート時に実行するコマンドの入ったファイルの読み込み
-s スタート時に`leaprc'を読み込まない
```

4.3.3. パラメータの付加

追加で力場を読み込むため、ウィンドウ内で以下のように実行します。

```
> seource leaprc.water.tip3p
```



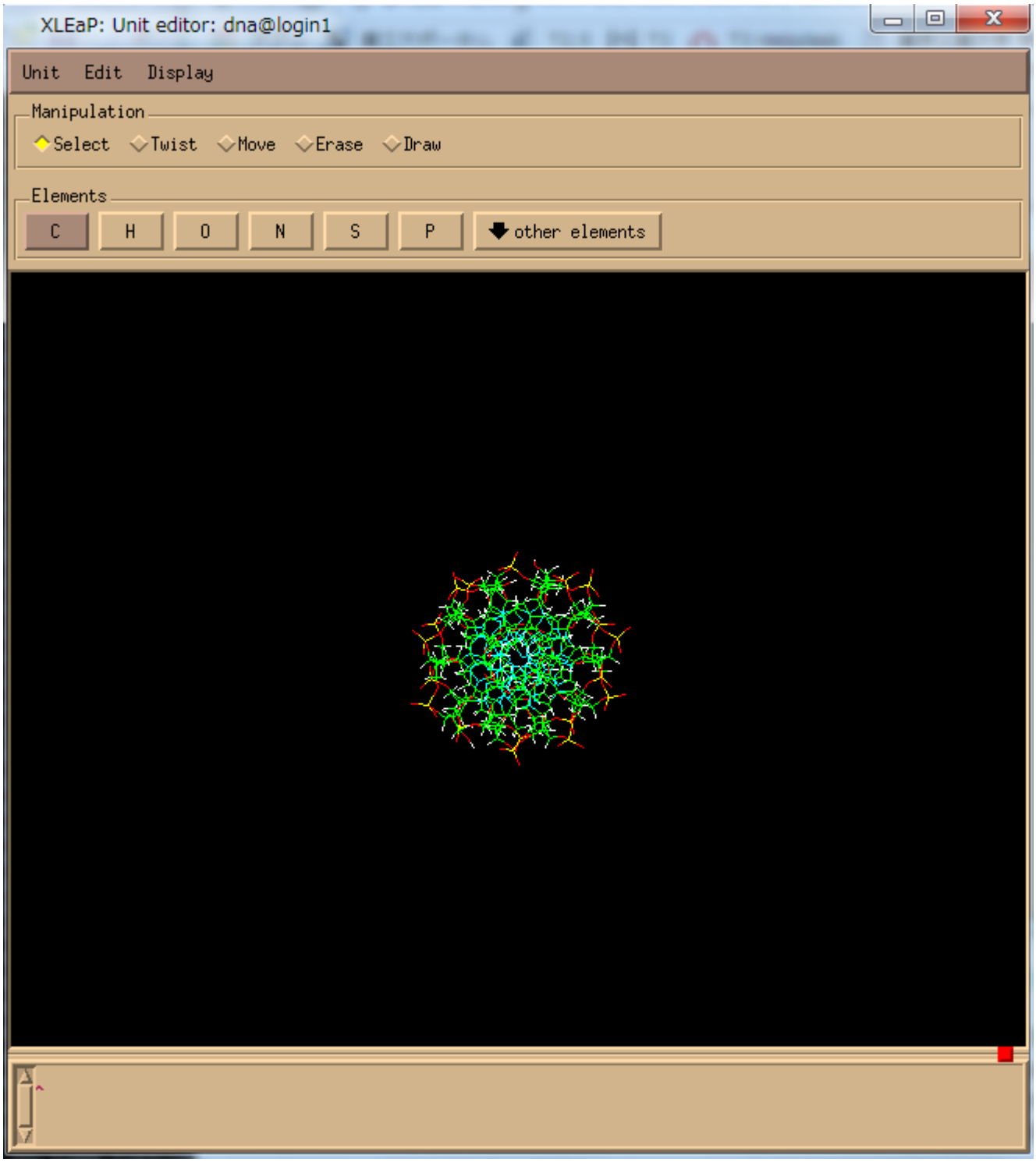
力場を読み込んだのでPDBを読み込むため、以下のように実行します。
下記はユーザが作成したディレクトリを指定してください

```
> dna = loadpdb "<PathToDirectory>/amber/dna/nuc.pdb"
total atoms in file: 638
```

PDBを表示、編集するため、以下のように実行します。

```
> edit dna
```

実行後、DNAが表示されたUnit editorウィンドウが表示されます。



以下のコマンドで真空条件のDNAのパラメータを保存できます。

```
> saveamberparm dna dna_vac.prmtop dna_vac.inpcrd
```

このまま保存すると以下の電荷に関するワーニングが出ますが、prmtopおよびinpcrdファイルが作成されます。 prmtopおよびinpcrdファイルは5章で説明します。詳細はマニュアルをご確認ください。

```
> saveamberparm dna dna_vac.prmtop dna_vac.inpcrd
Checking Unit.
WARNING: The unperturbed charge of the unit: -18.000000 is not zero.

 -- ignoring the warning.

Building topology.
```

```
Building atom parameters.
Building bond parameters.
Building angle parameters.
Building proper torsion parameters.
Building improper torsion parameters.
 total 110 improper torsions applied
Building H-Bond parameters.
Incorporating Non-Bonded adjustments.
Not Marking per-residue atom chain types.
Marking per-residue atom chain types.
 (no restraints)
```

4.3.4. チャージの付加

ワーニングに対応するため、以下のコマンドでイオンの付加を行います。 0は電荷をノーマルにするというオプションです。

```
> addions dna Na+ 0
```

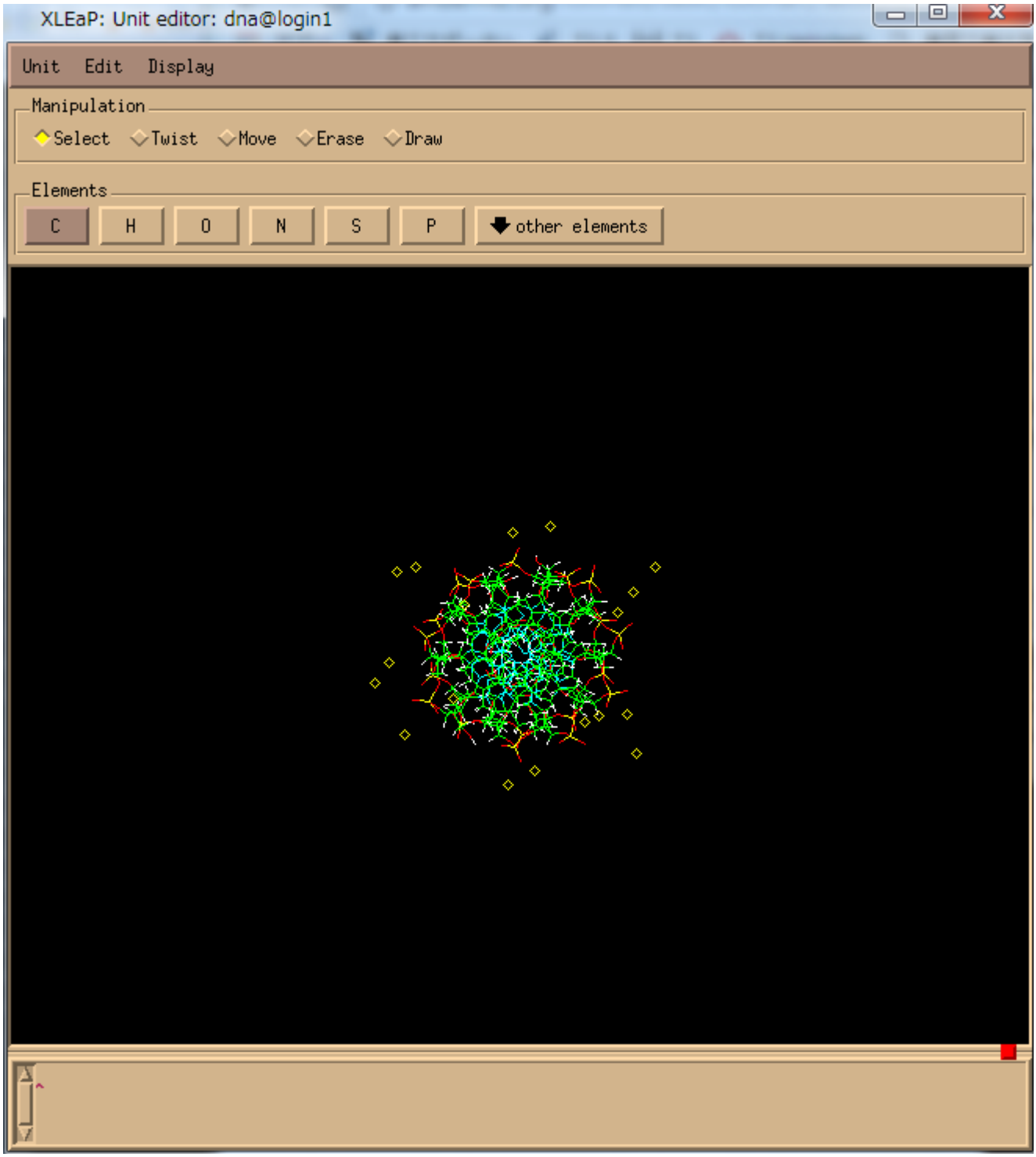
-18の電化に対して18個ナトリウムイオンが追加されました。

```
> addions dna Na+ 0
18 Na+ ions required to neutralize.
Adding 18 counter ions to "dna" using 1A grid
Grid extends from solute vdw + 3.65  to  9.75
Resolution:      1.00 Angstrom.
grid build: 0 sec
 (no solvent present)
Calculating grid charges
charges: 0 sec
Placed Na+ in dna at (6.44, 3.95, 17.79).
Placed Na+ in dna at (5.44, -5.05, 10.79).
Placed Na+ in dna at (-10.56, 5.95, 13.79).
Placed Na+ in dna at (-10.56, -6.05, 19.79).
Placed Na+ in dna at (-1.56, 11.95, 9.79).
Placed Na+ in dna at (-10.56, -4.05, 6.79).
Placed Na+ in dna at (-6.56, 4.95, 27.79).
Placed Na+ in dna at (11.44, -8.05, 22.79).
Placed Na+ in dna at (0.44, -12.05, 13.79).
Placed Na+ in dna at (11.44, 7.95, 10.79).
Placed Na+ in dna at (1.44, 11.95, 19.79).
Placed Na+ in dna at (10.44, -9.05, 4.79).
Placed Na+ in dna at (-7.56, 7.95, -0.21).
Placed Na+ in dna at (-11.56, -8.05, 27.79).
Placed Na+ in dna at (13.44, 1.95, 24.79).
Placed Na+ in dna at (-2.56, -12.05, 23.79).
Placed Na+ in dna at (-10.56, 8.95, 21.79).
Placed Na+ in dna at (13.44, 0.95, 3.79).

Done adding ions.
```

カウンターイオンがどのように入ったか確認するため、以下を実行します。

```
> edit dna
```

DNAの周りにカウンターイオンが存在することが確認できました。



4.3.5. MDの入力ファイルの作成

電荷がノーマルになったprmtopとinpcrdファイルを保存します。

```
> saveamberparm dna dna_cio.prmtop dna_cio.inpcrd
```

エラーが起こっていないことを確認してください。

```
> saveamberparm dna dna_cio.prmtop dna_cio.inpcrd
Checking Unit.
Building topology.
Building atom parameters.
Building bond parameters.
```

```
Building angle parameters.
Building proper torsion parameters.
Building improper torsion parameters.
 total 110 improper torsions applied
Building H-Bond parameters.
Incorporating Non-Bonded adjustments.
Not Marking per-residue atom chain types.
Marking per-residue atom chain types.
 (no restraints)
```

4.3.6. 水の付加

溶媒などを入れるため、コピーを作成しておきます。

```
> dna_cio = copy dna
```

水のボックスを作成します。 TIP3Pモデルを利用し、DNAの周りに8オングストロームのボックスを作成します。

```
> solvatebox dna TIP3PBOX 8.0
```

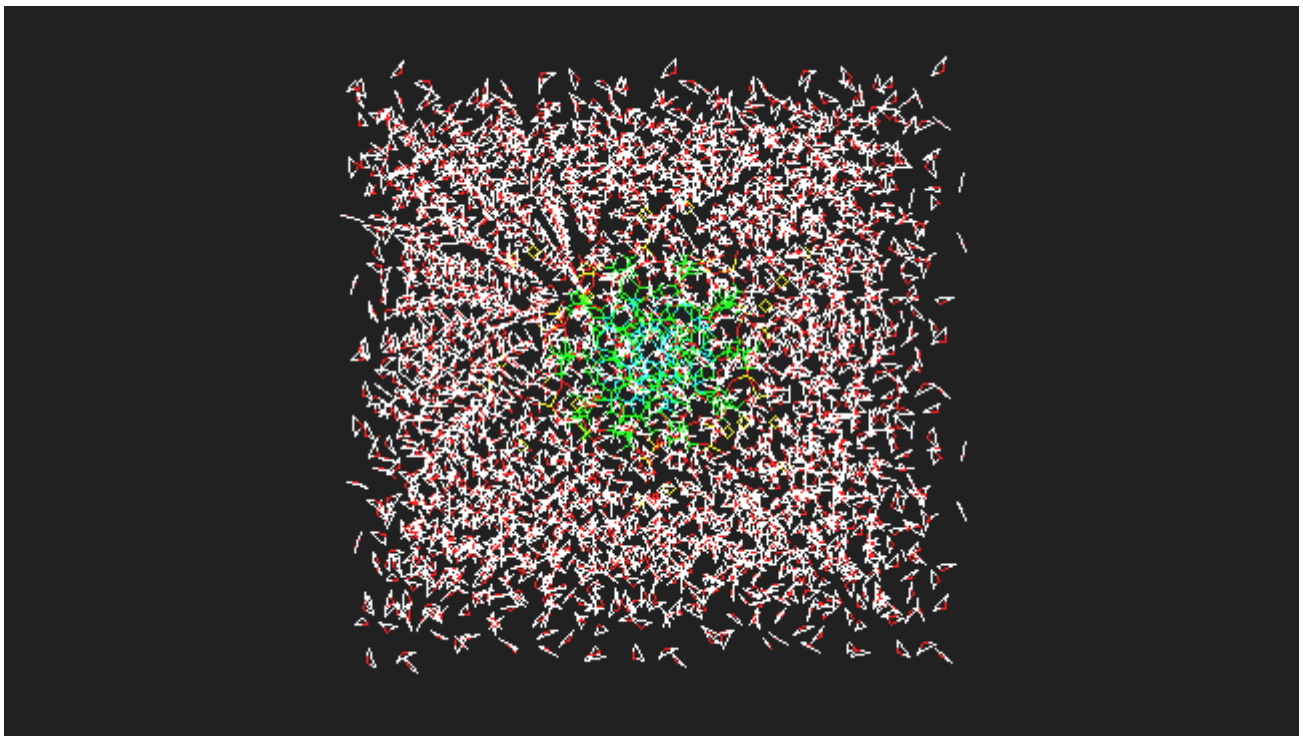
エラーが起こっていないことを確認してください。
2767個の水分子で満たされた約47x46x59のボックスが作成されました。

```
> solvatebox dna TIP3PBOX 8.0
  Solute vdw bounding box:              27.738 26.738 40.099
  Total bounding box for atom centers:  43.738 42.738 56.099
  Solvent unit box:                     18.774 18.774 18.774
  Total vdw box size:                   46.743 45.963 58.910 angstroms.
  Volume: 126564.801 A^3
  Total mass 56295.944 amu,  Density 0.739 g/cc
  Added 2767 residues.
```

水がどのように入ったか確認するため、以下を実行します。

```
> edit dna
```

DNAの周りに水が入ったことが確認できました。



実際の計算で利用するために八面体ボックスで設定するために下記のコマンドを実行します。

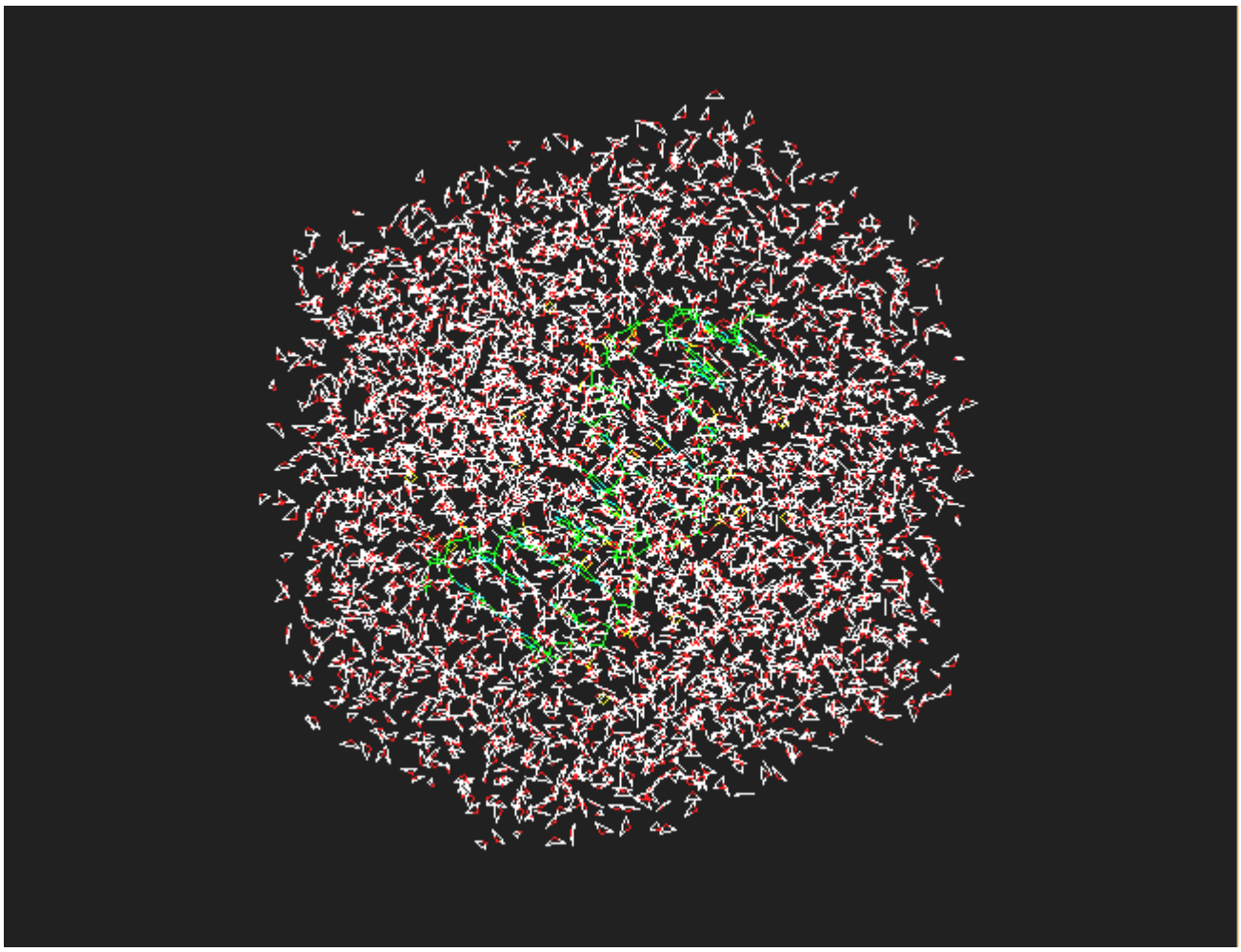
```
> solvateoct dna_cio TIP3PBOX 8.0
```

以下の出力が表示されます。

```
> solvateoct dna_cio TIP3PBOX 8.0
Scaling up box by a factor of 1.368620 to meet diagonal cut criterion
  Solute vdw bounding box:              27.987 26.927 38.921
  Total bounding box for atom centers:  60.819 60.819 60.819
      (box expansion for 'iso' is  51.9%)
  Solvent unit box:                     18.774 18.774 18.774
  Volume: 118123.162 A^3 (oct)
  Total mass 61286.376 amu,  Density 0.862 g/cc
  Added 3044 residues.
```

可視化して確認します。

```
> edit dna_cio
```



セルが八面体となっているprmtopとinpcrdファイルを保存します。

```
> saveamberparm dna_cio dna_wat.prmtop dna_wat.inpcrd
```

水で埋めたDNAをpdbファイルに出力するには以下のコマンドを実行します。

```
> savepdb dna dna.pdb
> savepdb dna_cio dna_cio.pdb
```

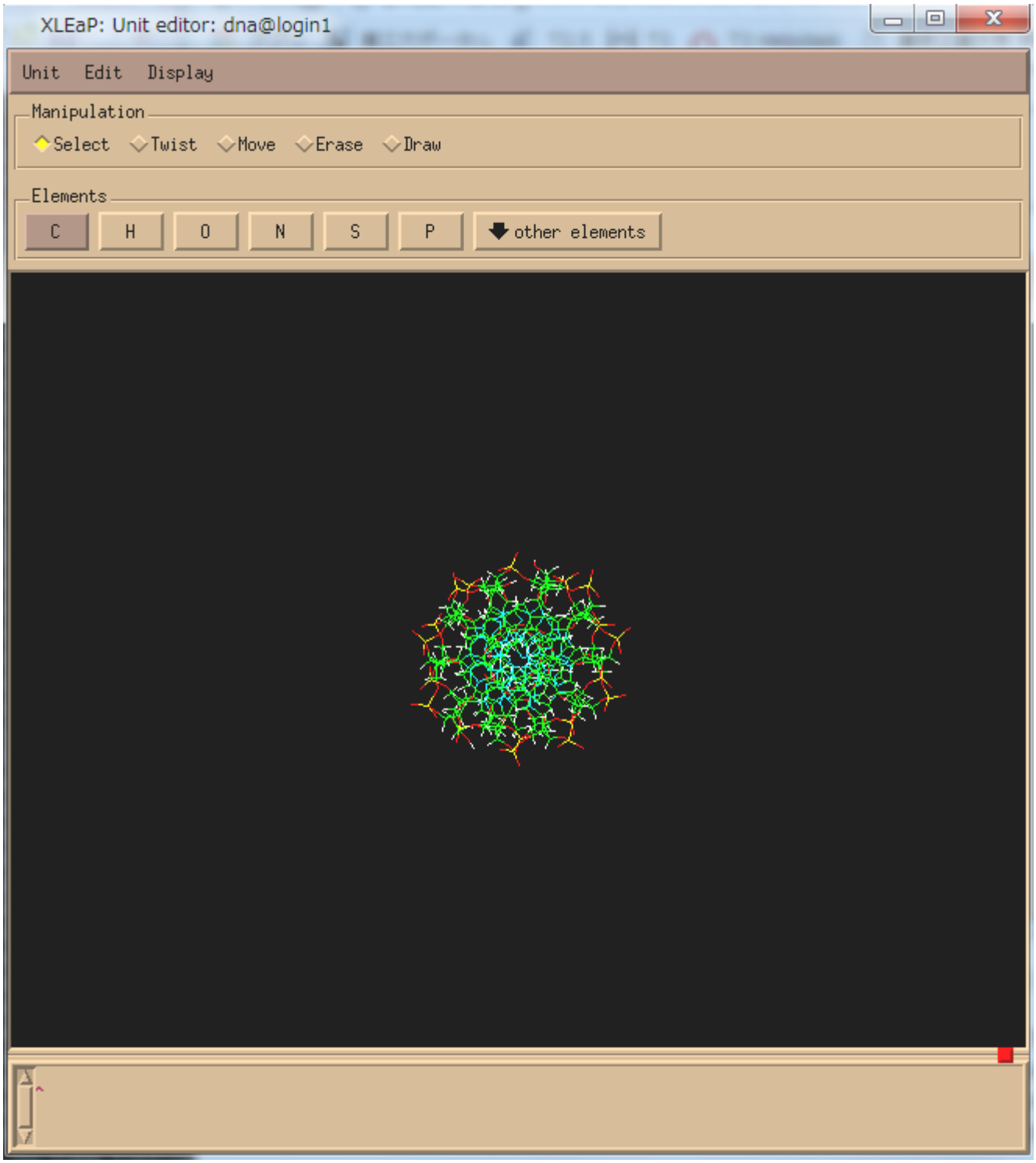
問題がない場合は以下のように表示されます。

```
> savepdb dna dna.pdb
Writing pdb file: dna.pdb
   printing CRYST1 record to PDB file with box info
> savepdb dna_cio dna_cio.pdb
Writing pdb file: dna_cio.pdb
   printing CRYST1 record to PDB file with box info
```

4.3.7. 分子の変更

Unit editerによる分子の変更手順について、先程読み込んだDNAを例にとって説明します。
PDBを表示、編集するため、以下のように実行します。

```
> edit dna
```



Unit editerが起動します。

Drawモード
Unit editorウィンドウ内のDRAWボタンをクリックし、マウスカーソルをViewing Window(右図の黒い部分)内に入れるとマウスカーソルが鉛筆の形になります
このモードをDRAWモードと呼びます。 このモードでは以下の処理が行えます。

・原子の生成

必要な原子を選択し、Viewing Window上でクリックします。 原子はElementsエリアのボタン、もしくはother elementsより選択ください。

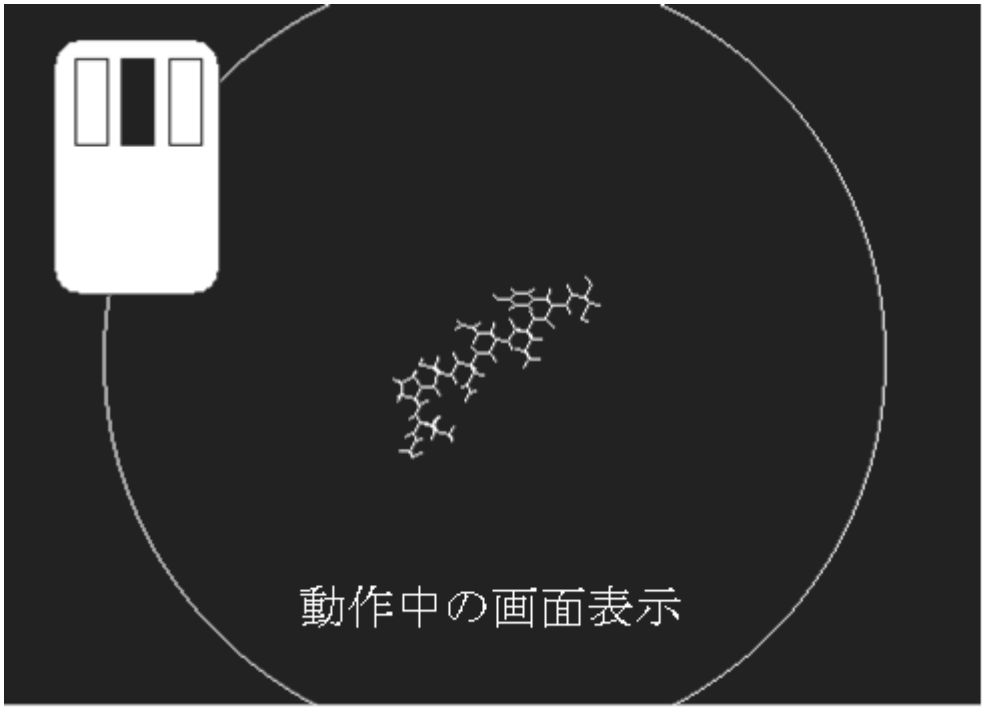
・結合の作成

結合の始点から終点までドラッグします。 二重結合の場合は2回、三重結合の場合は3回行います。
下図の左側が操作中の画面表示、右図が操作後の結合表示となります。

モデルの回転、移動、ズーミング
モデルの回転、移動、ズーミングはDRAWモード、ERASEモード、SELECTモード、TWISTモード、MOVEモードのいずれのモードにおいても以下の操作で行えます。

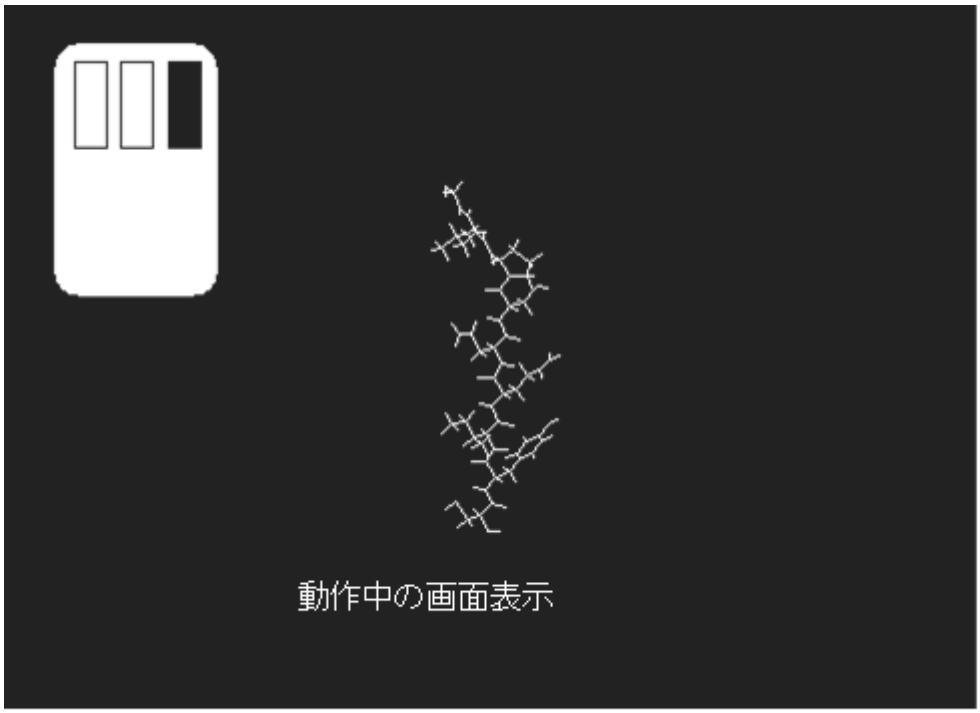
・モデルの回転

マウスの中ボタンでドラッグします。
(マウスによっては中ボタンがないものやホイールクリックに割り当てられているものがございます)



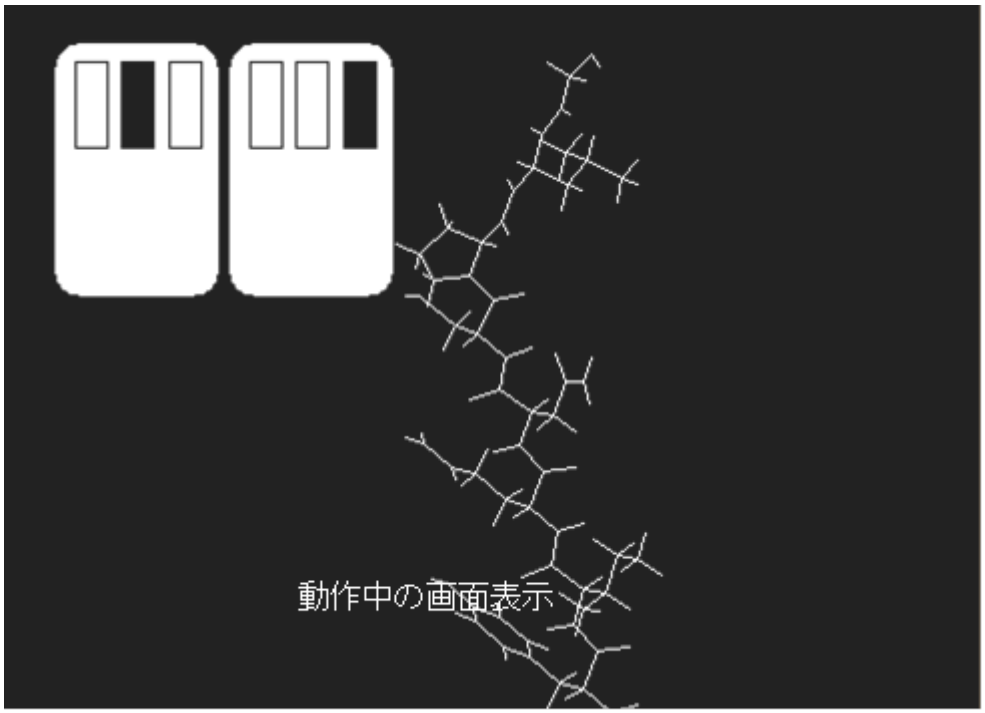
・モデルの平行移動

マウスの右ボタンでドラッグします。



・モデルの拡大・縮小

中クリックを押しながら右クリックを押し、ドラッグ
もしくは、Ctrlキーを押しながらマウスの右ボタンでドラッグします。
(注)マシンの設定により異なる場合があります。



ERASEモード
ERASEボタンをクリックし、ERASEモードにします。 このモードで原子または結合部分をクリックすると原子や結合が消去できます。

SELECTモード
SELECTボタンをクリックし、SELECTモードにします。 マウスの左ボタンでドラッグすると選択範囲の原子が選択され、違う色で表示されます。

選択範囲の解除は、シフトキーを押下しながらマウス左ボタンでドラッグします。 選択された原子はMOVEモードやTWISTモードで操作が行えます。

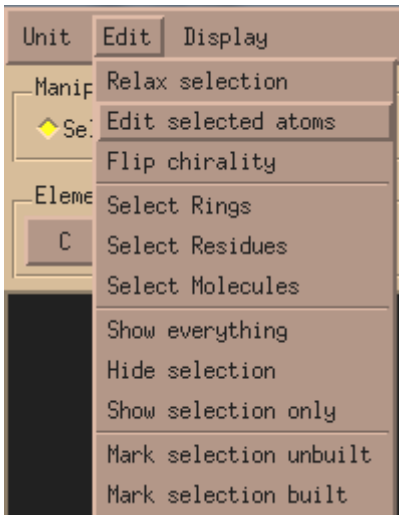
TWISTモード

4つの接続されている原子をSELECTモードで選択した後、TWISTボタンをクリックし、TWISTモードにします。 マウス左ボタンをドラッグすると、1つの原子をもとにして接続ボンドが回転します。

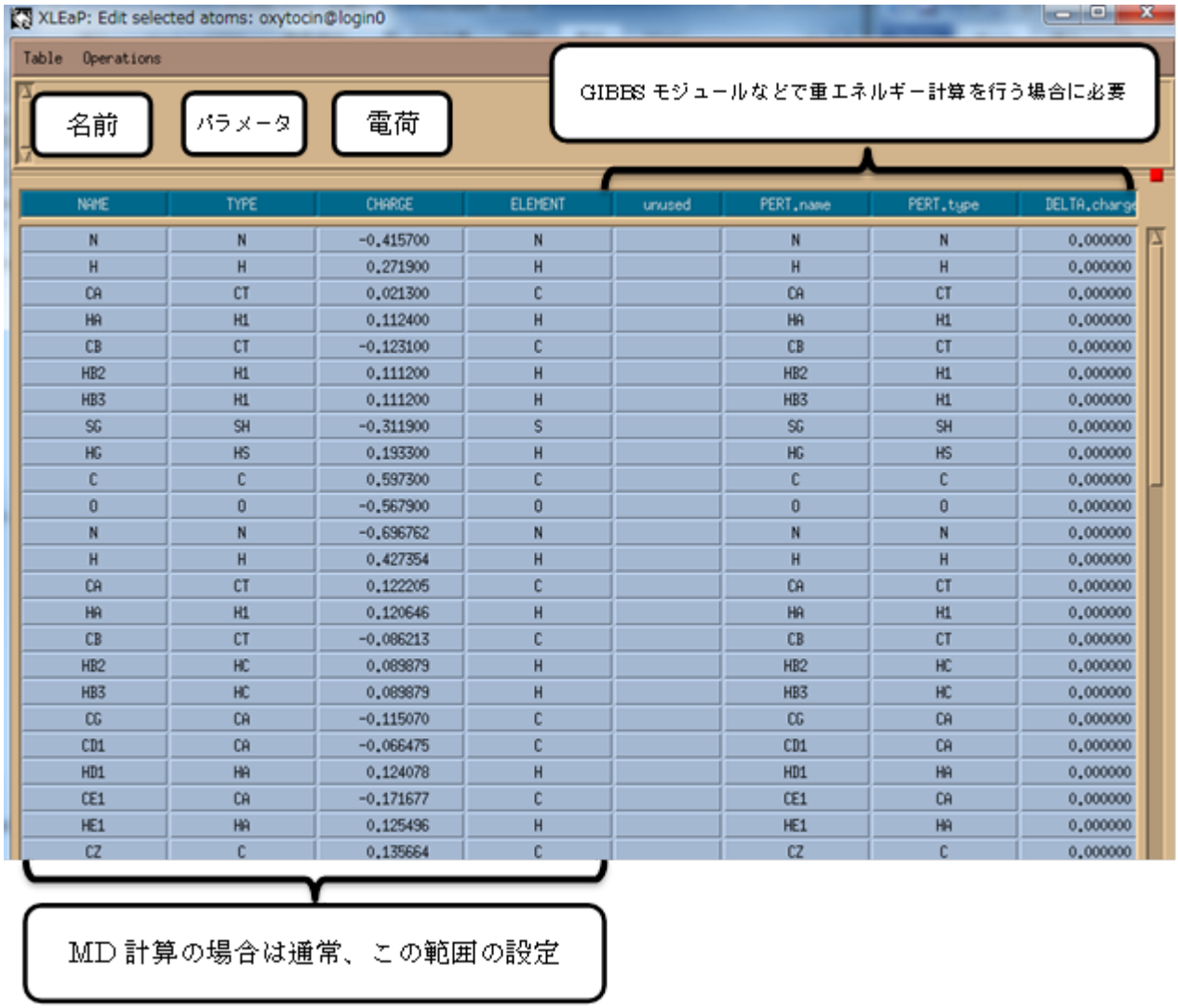
MOVEモード
MOVEボタンをクリックし、MOVEモードにします。 マウス左ボタンをドラッグするとSELECTモードで選択された原子だけが移動します。

自分の扱う分子のパラメータがAMBERのオリジナルデータベースにない場合は、新規にパラメータを作成する必要があります。 すでに設定されている場合はこのセクションをとばして下さい。

SELECT機能で編集したい原子を選択し、Unit editorウィンドウのDisplayメニューからNamesとTypesを選択し、表示させます。
Editメニューから Edit selected atomsを選択します。



Edit selected atomsウィンドウが表示されるので、NAME TYPE, CHARGE等を編集します。



編集が終わったらEdit selected atomsウィンドウのTableメニューよりSave and quitを選択し、終了します。

4.3.8. チェック

Check unitは以下の4点についてチェックを行います。

- bonds length
- non-integral total charge of the UNIT
- missing force field atom type
- close contacts between nonbonded ATOMs

Unit editorウィンドウでSELECTモードにし、ペプチド全体を選択します。

Unit menuからCheck unitを選択し、types、chargeに問題がないことを確認します。

4.3.9. 構造最適化

Edit menuからRelax selectionを選択します。 これは選択された原子のbonds、angles、torsionsについて最適化を行います。

4.4. LEaPのヘルプコマンド

Universe Editorウィンドウ内で、helpコマンドを実行すると使用可能なコマンドの一覧が表示されます。 また、helpコマンドの引き数に詳細を知りたいコマンドをつけて実行すると、そのコマンドの詳細なマニュアルが表示されます。

```
> > help
Help is available on the following subjects:

_cmd_options_       _types_             add                 addAtomTypes
addH                addIons             addIons2            addIonsRand
addPath             addPdbAtomMap       addPdbResMap        alias
alignAxes           bond                bondByDistance      center
charge              check               clearPdbAtomMap     clearPdbResMap
clearVariables      combine             copy                createAtom
createParmset       createResidue       createUnit          crossLink
debugOff            debugOn             debugStatus         deleteBond
deleteOffLibEntry   deleteRestraint     desc                deSelect
displayPdbAtomMap   displayPdbResMap    edit                flip
groupSelectedAtoms  help                impose              list
listOff             loadAmberParams     loadAmberPrep       loadMol2
loadMol3            loadOff             loadPdb             loadPdbUsingSeq
logFile             matchVariables      measureGeom         quit
relax               remove              restrainAngle       restrainBond
restrainTorsion     saveAmberParm       saveAmberParmNetcdf saveAmberParmPert
saveAmberParmPol    saveAmberParmPolPertsaveAmberPrep       saveMol2
saveMol3            saveOff             saveOffParm         savePdb
scaleCharges        select              sequence            set
set_default         setBox              showdefault         solvateBox
solvateCap          solvateDontClip     solvateOct          solvateShell
source              transform           translate           verbosity
zMatrix

For a list of the current aliases, type "alias".
```

4.5. LEaPのコマンド

ここでは、LEaPで使用される主なコマンドの使用法を示します。 他のコマンドについてはhelpコマンド、あるいはマニュアルを参照して下さい。

4.5.1. addPath

addPath path path: STRING
ファイルをサーチするPassの追加。

```
> addPath /home0/procon/xkibuse
```

4.5.2. alias

alias [string1 [string2]] string1: STRING string2: STRING
エイリアスの作成。

```
> alias q quit
```

4.5.3. charge

charge container
container: UNIT/RESIDUE/ATOM

Total chargeの計算。

```
> charge ALA
```

4.5.4. check

alias unit [params]
unit: UNIT params: PARMSET

UNITのチェック。bonds length、total charge、missing force field、close contactsについて調べる。

```
> check ALA
```

4.5.5. combine

variable = combine list variable: objest
list: LIST

list中のUNITsを一つのUNITに繋げる。 原子の結合は行わない。(sequence commandは結合も行う。)

```
> tripeptide= combine ALA GLY PRO
```

4.5.6. copy

newvariable = copy variable newvariable: objest variable: objest
コピーを作る。

```
> ala = copy ALA
```

```
備考
> ala = ALA
は違う意味で、これはALAにalaという別名を付けたことになります。つまり、alaを編集するとALAも同時に編集されます
```

4.5.7. desc

desc variable variable: objest
objectの内容の表示。

```
> desc ALA
> desc ALA.1
```

4.5.8. edit

edit unit unit: UNIT
エディターの起動(xleapのみで動作)。

```
> edit insulin_monomer
```

4.5.9. help

help string string: STRING
ヘルプ。

```
> help quit
```

4.5.10. impose

impose unit seqlist internals unit: UNIT
seqlist: LIST internals: LIST

UNITのinternal coordinatesをインポーズする。 下の例では、UNIT中のsequence numbers 1、2、3のRESIDUE部をαヘリックスコンフォメーションにしています。

```
> impose peptide { 1 2 3 } { { $N $CA $C $N -40.0 } { $C $N $CA $C -60.0 } }
```

4.5.11. list

list
現在定義されている全てのvariablesを表示

```
> list
```

4.5.12. listOff

listOff library library: STRING
libraryに保存されているUNITs/PARMSETsの表示。

```
> listOFF amino4.lib
```

4.5.13. loadAmberParams

variables = loadAbmerParams filename variable: PARMSET
filename: STRING

AMBER format papameter set fileをロードし、variableに置く。

```
> parm91 = loadAmberParams parm91X.dat
```

4.5.14. loadAmberPrep

loadAbmerPrep filename filename: STRING
ファイル名filenameのAMBER PREP input fileをロードし、filenameと同じ名前のUNITを生成する。

```
> loadAmberPrep cra.in
```

4.5.15. loadOff

loadOff filename filename: STRING
ファイル名filenameのOFF libraryをロードする。そのlibraryのUNITsやPARAMSETsの全てがロードされる。

```
> loadAmberPrep cra.in
```

4.5.16. loadPdb

variables = loadPdb filename variable: object
filename: STRING

ファイル名filenameのProtein Databank format fileをロードする。 このCommandの使用の際は注意が必要です。マニュアルを参照して下さい。

```
> crambin = loadPdb 1crn
```

4.5.17. mesureGeom

mesureGeom atom1 atom2 [atom3 [atom4]] atom1: ATOM
atom2: ATOM atom3: ATOM atom4: ATOM

distance、angle、torsionの計測。

```
> measure ALA.ALA.1 ALA.ALA.3 ALA.ALA.5
```

4.5.18. quit

quit
LEaPの終了。

```
> quit
```

4.5.19. saveAmberParm

saveAmberParm unit topologyfilename coordinatefilename unit: UNIT
topologyfilename: STRING coordinatefilename: STRING

そのUNITに対するトポロジーファイルとcoordinateファイルを保存する。 これらのfilesがLEaPの最終生成物で、この後に続くMDの入力ファイルになる。

```
> saveAmberParm ALA ala.top ala.crd
```

4.5.20. saveOff

saveOff object filename object: object
filename: STRING

UNITsやPARAMSETsをfilenameでObject File Format(OFF)で保存。 もしfilenameが既存するならそのfileに追加される。

```
> saveOff BGLU bglu.lib
```

4.5.21. savePdb

savePdb unit filename unit: UNIT
filename: STRING

UNITsやPARAMSETsをfilenameでObject File Format(OFF)で保存。 もしfilenameが既存するならそのfileに追加される。

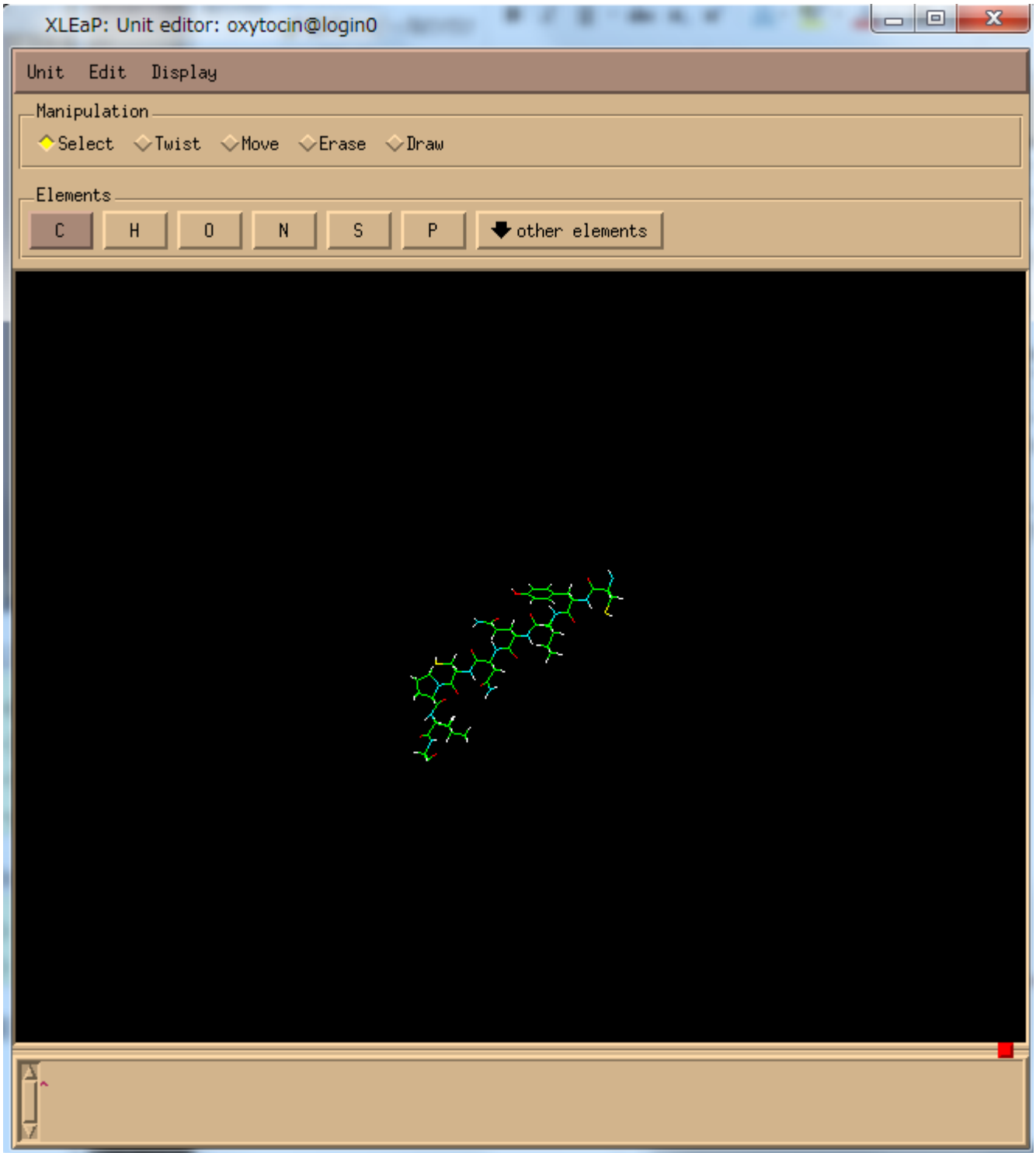
```
> saveOff BGLU bglu.lib
```

4.5.22. sequence

variables = sequence list variable: UNIT
list: LIST

list中のUNITsを一つのUNITに繋げる。

```
> oxytocin = sequence { CYS TYR ILE GLN ASN CYS PRO LEU GLY }
> edit oxytocin
```



4.5.23. solvateBox

solvateBox solute solvent buffer [closeness] solute: UNIT
solvent: UNIT buffer: object closeness: NUMBER

soluteの周りに溶媒を用意する。 bufferはsolute ATOMから溶媒のBOXの壁までの距離で、単一の数値の時はx、y、z方向全てに対して同じ値が使われる。 これは下の例のようにLISTで3方向別々に指定できる。closenessは溶媒とsoluteの距離です。単位はA

```
> solvateBox sol WATBOX216 8 2.0
> solvateBox sol WATBOX216 { 8.2 7.4 9.0 } 2.0
```

4.5.24. solvateShell

solvateShell solute solvent thickness [closeness] solute: UNIT
solvent: UNIT thickness: NUMBER closeness: NUMBER

soluteの周りに溶媒のシェルを作る。

```
> solvateShell shell WATBOX216 8.0 2.0
```

4.5.25. source

source filename
filename: STRING

テキストファイルの中のCommandを実行。

```
> source file.x
```

4.5.26. zMatrix

zMatrix object zmatrix
マニュアルをご確認下さい。

5. 分子動力学計算

分子動力学計算は以下の手順で行います。
Amberがインストールされているディレクトリはロードしたバージョンによって異なりますが、環境変数としてはすべて、$AMBERHOMEに設定されます。
このディレクトリ下の主なディレクトリを以下に示します。

$AMBERHOME/内の主なディレクトリ

ディレクトリ	説明
dat/	Amberのデータベース
bin/	Amber実行モジュール
test/	各モジュールを実行するためのスクリプト例

Minimization、equilibration、Molecular Dynamicsは、binディレクトリ内にある sander というモジュールで行います。sander の使用方法は下のようになっています。

```
usage: sander  [-O|A] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
                  [-ref refc -x mdcrd -v mdvel -e mden -frc mdfrc -idip inpdip -rdip rstdip -mdip mddip
                  -inf mdinfo -radii radii -y inptraj -amd amd.log -scaledMD scaledMD.log] -cph-data -ce-data <file>-host ipi_hostname -port ipi_port
```

それぞれのファイルは以下の通りです。

sanderによる計算に必要なファイル

ファイル	説明
mdin	min/mdにおけるコントロールデータ(input)。
mdout	アウトプットファイル(output)。
prmtop	分子のトポロジー、パラメータの情報のデータ(input)。
inpcrd	初期構造と(オプションで)初期速度のデータ(input)。
restrt	最終の座標、速度、ボックスサイズ(for Constant Pressure)のデータ(output)。
refc	ポジションを束縛するときのデータ(input)。
mdcrd	座標値のトラジェクトリファイル(output)。
mdvel	速度のトラジェクトリファイル(output)。
mden	エネルギーのトラジェクトリファイル(output)。
mdinfo	mdoutの一番最新の部分(output)。 \

インプットファイルのオプション指定は&cntrlから/の間に記載し、オプションが複数ある場合は「,」で区切り、&cntrl から/までの行は必ず行頭に空白を入れて下さい。
インプットファイルで使用できる全オプションの詳細説明はマニュアルを参照して下さい。

インプットファイルのオプション

オプション	説明
imin	0の場合はMD計算、1の場合はエネルギー極小化計算、5の場合は振動解析を行います。
nmropt	0の場合はNMRを考慮しない計算、1,2の場合はNMR計算を行います。
ntx	初期座標、速度および基本セルの読み込み方法の指定フラグです。1がデフォルトとなっており、初期座標のみが呼び出されます。MD計算のリスタートでは5を利用して、初期座標、速度を呼び出します。基本セルはntb>0のときに行われます。なお、irestが1のときのみ、速度が呼び出されます。
irest	リスタートフラグです。0の場合はリスタートを行わず、1の場合は座標や速度をリスタートファイルから読み込みます。
ntpr	mdoutおよびmdinfoを更新するステップ数、デフォルトは50
ntb	周期的境界条件の設定
cut	カットオフ値、単位はオングストローム
maxcyc	極小化計算の最大サイクル数
ntr	0より大きい場合に位置束縛条件の設定を行います。デフォルトは0
ntc	SHAKE法による分子固定設定、ntfと合わせて利用します。
ntf	力の評価設定、ntcと合わせて利用します。
tempi	初期温度設定
temp0	定常温度の設定
gamma_ln	衝突頻度因子
ntwx	mdcrd
ntwr	リスタートファイルを更新するステップ数、デフォルトはnstlim
nstlim	MD計算のステップ数
pres0	初期圧力設定、デフォルトは1.0
taup	圧力緩和時間、デフォルトは1.0
ntt	定温シミュレーションの条件
dt	トラジェクトリー計算における時間ステップ

6. GPUによる高速化

6.1. 概要

このページでは、GPU計算についての説明を行います。
TSUBAME4では計算ノードの各ノードにNvidia tesla H100が4基内蔵されており、ユーザはGPUによる高速化の恩恵を受けることができます。
近年、GPU計算に対応するアプリケーションも増えており、Amberもその一つです。
PMEMDの新機能としてNVIDIA製GPUを用いたNVE、NVT、NPTカノニカルアンサンブルでの溶媒分子を露わに扱うPMEシミュレーションおよび連続誘電体モデルであるGeneralized Bornシミュレーションの高速化が挙げられます。 また、マルチGPUでの計算にも対応しています。

なお、この章にある情報はほとんどが次のURLからの引用、和訳となっています。 より詳しく調べたい方、または最新の内容を確認したい方はambermd.orgのページをご参照ください。

http://ambermd.org/GPUSupport.php

GPUによる高速化は恩恵が大きいですが、歴史は浅いためユーザは注意深くなる必要があるとambermd.org では述べられています。もし、問題に遭遇したら同等のシミュレーションをCPUで実施して、シミュレーション設定の問題かどうかの切り分けを行うようにしてください。

6.2. pmemd.cudaとpmemdとの違い

6.2.1. 機能の違い

NVE、NVT、NPTカノニカルアンサンブルでのExplicit solvent(陽溶媒)PMEシミュレーションおよびImplicit solvent(暗溶媒)Generalized Bornシミュレーションは、標準のGPUを使用しないpmemdとほぼ同等になるようにデザインされています。
しかしいくつかの制約があり、制約内容は概要で示したambermd.orgのページをご参照ください。 また、念のためCPUで短いシミュレーションを実行して Ewald error estimateを確認し、妥当な数値となっているかどうか確認することをおすすめします。

6.2.2. 出力の違い

出力ファイルのフォーマットにいくつかの差異がありますGPU計算に対応したpmemd.cudaで実行した際、pmemdとの実行結果の主な違いは次の4点です。

- GPU の著作情報が現れること
- GPUのdevice 情報が出力されること
- Conditional Compilation Defines UsedとしてCUDAが含まれていること
- Ewald error estimateの出力がなされないこと

なお、同一の入力ファイルを用いてGPU, CPUそれぞれで計算した結果をsdiffコマンドで差分をとって確認すると次のようになります。左がGPU,右がCPUとなります。

```
 Amber 16 PMEMD                              2016                              Amber 16 PMEMD                              2016
          ------------------------------------------------------                            ------------------------------------------------------
| PMEMD implementation of SANDER, Release 16                                                          | PMEMD implementation of SANDER, Release 16
| Run on 09/04/2017 at 12:41:23                                                                                                 |                           | Run on
09/04/2017 at 13:37:44
|   Executable path: pmemd.cuda.MPI                                                                                             |                           |   Executable
path: pmemd.MPI
| Working directory: /home/7/A2901692/amber/dna_wat3                                  |             | Working directory: /home/7/A2901692/amber/dna_wat2
|          Hostname: r2i5n7                                                                                                     |                           |
Hostname: r5i4n3
  [-O]verwriting
output
[-O]verwriting output
File
Assignments:
File Assignments:
|   MDIN: dna_water_md.in                                                                                                       |   MDIN: dna_water_md.in
|  MDOUT: dna_water_md.out                                                                                                      |  MDOUT: dna_water_md.out
| INPCRD: dna_water_md_fixd.rst                                                                                                 |    INPCRD: dna_water_md_fixd.rst
|   PARM: prmtop                                                                                                                |   PARM: prmtop
| RESTRT: dna_water_md.rst                                                                                                      |   RESTRT: dna_water_md.rst
|   REFC: refc                                                                                                                  |   REFC: refc
```

```
|  MDVEL: mdvel                                                    |  MDVEL: mdvel
|   MDEN: mden                                                     |   MDEN: mden
|  MDCRD: dna_water_md.mdcrd                                       |  MDCRD: dna_water_md.mdcrd
| MDINFO: mdinfo                                                   | MDINFO: mdinfo
|LOGFILE: logfile                                                  |LOGFILE: logfile
|  MDFRC: mdfrc                                                    |  MDFRC: mdfrc
 Here is the input file:                                                                                  Here is the input file:
略
Note: ig = -1. Setting random seed to   234768 based on wallc |    Note: ig = -1. Setting random seed to   938027 based on wallc
      microseconds and disabling the synchronization of rando          microseconds and disabling the synchronization of rando
      between tasks to improve performance.                                                                between tasks to improve performance.
|--------------------- INFORMATION ----------------------     <
| GPU (CUDA) Version of PMEMD in use: NVIDIA GPU IN USE.      <
|                    Version 16.0.0                                                                  <
|                      02/25/2016                                                                  <
| Implementation by:                                                                                                  <
|                    Ross C. Walker     (SDSC)                                                <
|                    Scott Le Grand     (nVIDIA)                          <
| Precision model in use:                                                                                             <
|      [SPFP] - Single Precision Forces, 64-bit Fixed Point   <
|               Accumulation. (Default)                                                                  <
|------------------------------------------------------------     <
|----------------- CITATION INFORMATION ------------------     <
|    When publishing work that utilized the CUDA version      <
|    of AMBER, please cite the following in addition to                          <
|    the regular AMBER citations:                                                                                 <
|  - Romelia Salomon-Ferrer; Andreas W. Goetz; Duncan                        <
|    Poole; Scott Le Grand; Ross C. Walker "Routine                             <
|    microsecond molecular dynamics simulations with                          <
|    AMBER - Part II: Particle Mesh Ewald", J. Chem.                           <
|    Theory Comput., 2013, 9 (9), pp3878-3888,                                      <
|    DOI: 10.1021/ct400314y.                                                                                      <
|  - Andreas W. Goetz; Mark J. Williamson; Dong Xu;                             <
|    Duncan Poole; Scott Le Grand; Ross C. Walker                               <
|    "Routine microsecond molecular dynamics simulations      <
|    with AMBER - Part I: Generalized Born", J. Chem.                           <
|    Theory Comput., 2012, 8 (5), pp1542-1555.                                      <
|  - Scott Le Grand; Andreas W. Goetz; Ross C. Walker                           <
|    "SPFP: Speed without compromise - a mixed precision      <
|    model for GPU accelerated molecular dynamics                               <
|    simulations.", Comp. Phys. Comm., 2013, 184                                  <
|    pp374-380, DOI: 10.1016/j.cpc.2012.09.022                                      <
|------------------------------------------------------------     <

<
|------------------- GPU DEVICE INFO --------------------     <
|                        Task ID:      0                                  <
|            CUDA_VISIBLE_DEVICES: not set                                <
|   CUDA Capable Devices Detected:      4                                    <
|           CUDA Device ID in use:      0                                      <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                  <
| CUDA Device Num Multiprocessors:     56                                     <
|           CUDA Device Core Freq:   1.48 GHz                                                 <
|                        Task ID:      1                                  <
|            CUDA_VISIBLE_DEVICES: not set                                <
|   CUDA Capable Devices Detected:      4                                    <
|           CUDA Device ID in use:      1                                      <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                  <
| CUDA Device Num Multiprocessors:     56                                     <
|           CUDA Device Core Freq:   1.48 GHz                                                 <
|                        Task ID:      2                                  <
|            CUDA_VISIBLE_DEVICES: not set                                <
|   CUDA Capable Devices Detected:      4                                    <
|           CUDA Device ID in use:      2                                      <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                  <
| CUDA Device Num Multiprocessors:     56                                     <
|           CUDA Device Core Freq:   1.48 GHz                                                 <
|                        Task ID:      3                                  <
|            CUDA_VISIBLE_DEVICES: not set                                <
|   CUDA Capable Devices Detected:      4                                    <
|           CUDA Device ID in use:      3                                      <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                  <
| CUDA Device Num Multiprocessors:     56                                     <
|           CUDA Device Core Freq:   1.48 GHz                                                 <
|                        Task ID:      4                                  <
|            CUDA_VISIBLE_DEVICES: not set                                <
|   CUDA Capable Devices Detected:      4                                    <
|           CUDA Device ID in use:      0                                      <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                  <
| CUDA Device Num Multiprocessors:     56                                     <
|           CUDA Device Core Freq:   1.48 GHz                                                 <
|                        Task ID:      5                                  <
|            CUDA_VISIBLE_DEVICES: not set                                <
|   CUDA Capable Devices Detected:      4                                    <
|           CUDA Device ID in use:      1                                      <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                  <
| CUDA Device Num Multiprocessors:     56                                     <
|           CUDA Device Core Freq:   1.48 GHz                                                 <
```

```
|                          Task ID:      6                                       <
|            CUDA_VISIBLE_DEVICES: not set                                      <
|   CUDA Capable Devices Detected:      4                                          <
|           CUDA Device ID in use:      2                                           <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                              <
| CUDA Device Num Multiprocessors:     56                                          <
|           CUDA Device Core Freq:   1.48 GHz                                                             <
|
<
|
<
|                          Task ID:      7                                       <
|            CUDA_VISIBLE_DEVICES: not set                                      <
|   CUDA Capable Devices Detected:      4                                          <
|           CUDA Device ID in use:      3                                           <
|                CUDA Device Name: Tesla P100-SXM2-16GB        <
|     CUDA Device Global Mem Size:  16276 MB                                                              <
| CUDA Device Num Multiprocessors:     56                                          <
|           CUDA Device Core Freq:   1.48 GHz                                                             <
|
<
|--------------------------------------------------------     <
|---------------- GPU PEER TO PEER INFO -----------------     <
||   Peer to Peer support: ENABLED                                                          <
|--------------------------------------------------------     <
略
|  Final Performance Info:                                                                                             |
Final Performance Info:
|     -----------------------------------------------------                       |     -----------------------------------------------------
|     Average timings for last   15900 steps:                                       |               |     Average timings for last   23900 steps:
|     Elapsed(s) =      27.98 Per Step(ms) =       1.76                    |               |     Elapsed(s) =      55.95 Per Step(ms) =       2.34
|         ns/day =      98.18   seconds/ns =     879.98              |               |         ns/day =      73.82   seconds/ns =    1170.49
|
|
|     Average timings for all steps:                                                                                |     Average
timings for all steps:
|     Elapsed(s) =      87.83 Per Step(ms) =       1.76                    |               |     Elapsed(s) =     116.06 Per Step(ms) =       2.32
|         ns/day =      98.37   seconds/ns =     878.34              |               |         ns/day =      74.44   seconds/ns =    1160.64
|     -----------------------------------------------------                       |     -----------------------------------------------------

|  Master Setup CPU time:            3.32 seconds                           |               |  Master Setup CPU time:            0.16 seconds
|  Master NonSetup CPU time:        87.67 seconds                           |               |  Master NonSetup CPU time:       115.83 seconds
|  Master Total CPU time:           90.99 seconds     0.03 ho |                          |  Master Total CPU time:          115.99 seconds     0.03 ho
|
|  Master Setup wall time:          21    seconds                            |               |  Master Setup wall time:           1    seconds
|  Master NonSetup wall time:       88    seconds                            |               |  Master NonSetup wall time:      116    seconds
|  Master Total wall time:         109    seconds     0.03 ho |                          |  Master Total wall time:         117    seconds     0.03 ho
```