

2020年6月18日(木) CCMSハンズオン



MateriApps LIVE!講習

LAMMPS を利用した古典動力学計算 の簡単な実習

高度情報科学技術研究機構(RIST)

利用支援部 吉澤香奈子

Outline

- LAMMPSについて
 - 分子動力学計算について
 - 力場(ポテンシャル)について
 - 実行時エラー
- LAMMPSのexamplesの実行
 - melt, micelle, colloid を実行する
 - 計算結果をVMDで可視化する
 - micelle を3次元化する
 - サイズを大きくした計算

LAMMPSについて

- Large-scale Atomic/Molecular Massively Parallel Simulator
<http://lammps.sandia.gov/>
- 並列計算機のために設計された分子動力学シミュレータ
 - 物質系を構成する原子や分子の1つ1つに対する運動方程式を数値的に計算している
 - 高並列計算のため2007年にC++でオブジェクト指向で完全に、rewriteされている
オブジェクト指向 → コード拡張・公開 → 多分野・多スケールに渡る計算
 - 現在、粒子系統合シミュレータとして高い支持
- ソースコードダウンロード
<http://lammps.sandia.gov/download.html>
 - 最新のStable version (3 Mar 2020) : **2020年6月14日現在**
ソースファイル名 : **lammps-stable.tar.gz**
 - Stable以外の最新のコードもある
過去のメジャーなコード : <http://lammps.sandia.gov/tars/>
- ドキュメント
<http://lammps.sandia.gov/doc/Manual.html>

何か不明なことがあつたらこの
ページを見る

分子動力学計算について

- ニュートンの運動方程式

$$F=ma$$

- 複数の粒子の運動を記述する運動方程式を時間に関して積分して粒子の軌跡を求める。
- 力は経験的に決められたポテンシャル（力場）の空間座標に対する微分である。

- モデリング（平衡化）の流れ

- 平衡化計算
エネルギー極小化
- 平衡化計算
温度一定MD
- 平衡化計算
温度・圧力一定MD
- 本計算
温度・圧力一定MD

力場(ポテンシャル)について(1)



- LAMMPS Force Fields
http://lammps.sandia.gov/doc/99/force_fields.html
- まずは、パッケージにある examples や potentials を参考にする。
(例)LAMMPSのパッケージ(lammps-stable.tar.gz)の
lammps-3Mar20/potentials
を参考にする。
- 目安としては、計算対象の物性が正しいものを選ぶ。
(例)常温常圧の鉄は、体心立方格子構造(bcc構造)
lammps-3Mar20/potentials/Fe_mm.eam.fs を用いる

```
# Interatomic potential – Embedded Atom Method
pair style eam/fs
# interaction pairs , filename , Element parameters
pair coeff * * Fe mm.eam. fs Fe
```

入力ファイルの書式例

- 自分でカスタマイズできるが、初心者には難しい。
(→ 専門家に相談)
- 力場(ポテンシャル)にこだわっても、結局は古典計算?
経験的に決める
→ 過去の成功例、論文になっているポテンシャルを用いる。
(非経験的に決めるなら第一原理計算)

物性が正しいように決める

力場(ポテンシャル)について(2)



- ポテンシャルのデータベース
Interatomic Potentials Repository Project
<https://www.ctcms.nist.gov/potentials/>
(例) Al-Ni: <https://www.ctcms.nist.gov/potentials/Al-Ni.html>
Mishin-Ni-Al-2009.eam.alloy を用いる
- 分子モデリングソフトウェアで設定する
下記のURLなどを参考にする
Winmostar: https://winmostar.com/jp/manual_jp.html
Molby: <http://molby.osdn.jp/doc/ja/index.html>

実行時エラー(1)

- 実行時にエラーが出た場合

ホームページ

<https://lammps.sandia.gov/doc/Errors.html>

13.3. Error messages

1-3 bond count is inconsistent

An inconsistency was detected when computing the number of 1-3 neighbors for each atom. This likely means something is wrong with the bond topologies you have defined.

1-4 bond count is inconsistent

An inconsistency was detected when computing the number of 1-4 neighbors for each atom. This likely means something is wrong with the bond topologies you have defined.

Accelerator sharing is not currently supported on system

Multiple MPI processes cannot share the accelerator on your system. For NVIDIA GPUs, see the nvidia-smi command to change this setting.

All angle coeffs are not set

All angle coefficients must be set in the data file or by the angle_coeff command before running a simulation.

...

を参考にする。

実行時エラー(2)

- 経験を積むとエラーの原因がわかりやすくなる。
- (例) colloidのパッケージが入っていない場合
examples の colloid を実行しようとすると、
ERROR: Unknown pair style colloid (../force.cpp:246)
Last command: pair_style colloid 12.5
のエラーが出る。

入力ファイル(in.colloid)において、
“pair_style colloid 12.5” という指定が許されていない

https://lammps.sandia.gov/doc/Errors_messages.html

13.3. Error messages

のページには、

Unknown pair style

The choice of pair style is unknown.

→ make yes-colloid によって、colloidのパッケージを入れれば解決する。

LAMMPSのexamplesについて

➤ LAMMPS の examples

LAMMPSパッケージ: `lammmps-3Mar20/examples`

(MateriApps LIVE!: `/usr/share/lammmps/examples`)

➤ examples の解説

<https://lammmps.sandia.gov/doc/Examples.html>

- 2次元系(ビジュアル重視)

- `colloid`, `crack`, `flow`, `friction`,
`micelle`, `nemd`, `obstacle`, `shear`

- 3次元系(簡単なもの)

- `melt`, `peptide`

- 力場

- `dreiding`, `kim`, `reax`

- 高速化機能

- `cuda`, `gpu`, `intel`

高分子物理向けのExamples

例	説明
<code>melt</code>	rapid melt of 3d LJ system
<code>micelle</code>	self-assembly of small lipid-like molecules into 2d bilayers
<code>colloid</code>	big colloid particles in a small particle solvent, 2d system
<code>peptide</code>	dynamics of a small solvated peptide chain (5-mer)

examples/melt(1)

melt: FCC格子状に並んだ LJ 粒子が溶けて拡散

入力ファイル: in.melt

```
# 3d Lennard-Jones melt

units          lj
atom_style     atomic

lattice        fcc 0.8442
region         box block 0 10 0 10 0 10
create_box     1 box
create_atoms   1 box
mass           1 1.0

velocity       all create 3.0 87287

pair_style     lj/cut 2.5
pair_coeff     1 1 1.0 1.0 2.5

neighbor       0.3 bin
neigh_modify   every 20 delay 0 check no
```

neigh_modify では、どのくらいの頻度で neighbor list を更新するかの指定。
ここでは、neighbor list の最後の構成から 0 ステップまで再構成をせず、0 ステップから 20 ステップ後に neighbor list の再構成を行う。

Lennard-Jones (LJ)
古典粒子

格子定数: FCC 0.8442
シミュレーションボックスの指定
→ 今は、4,000 原子の計算
create_atoms で、シミュレーションボックスの中に原子を配置する。

初期温度の設定

速度の設定

力場パラメータの設定 (lj ポテンシャルを指定)

neighbor では、ポテンシャル cutoff よりどれだけ外側まで隣接原子として探すかを指定 (距離内にある粒子で neighbor list を構成する)。ここでは (cutoff + 0.3) Ang の範囲を隣接とし、bin で O(N) の探索手法を用いる

fix

1 all nve

fix で系の制御

NVE アンサンブル

examples/melt(2)

入力ファイル: in.melt (続き)

```
#dump id all atom 50 dump.melt

#dump 2 all image 25 image.*.jpg type type &
#      axes yes 0.8 0.02 view 60 -30
#dump_modify 2 pad 3                                # は、コメントアウト
                                                    # dump:出力形式の指定

#dump 3 all movie 25 movie.mpg type type &
#      axes yes 0.8 0.02 view 60 -30
#dump_modify 3 pad 3                                何ステップ毎に温度情報
                                                    # を出力するかの指定

thermo 50                                         計算実行(step)数
run 250
```

実行準備

- ホームディレクトリに作業ディレクトリを作る。(例: \$HOME/lammps)
\$ mkdir lammps
- 作成した作業ディレクトリへ移動する。(\$HOME/lammps)
\$ cd lammps
- exmapleファイルのコピー(melt, micelle, colloid)
\$ cp -r /usr/share/lammps/examples/melt ./
\$ cp -r /usr/share/lammps/examples/examples/micelle ./
\$ cp -r /usr/share/lammps/examples/examples/colloid ./

meltの実行(1)

- melt へ移動

```
$ cd melt
```

- melt の中のファイルの確認

```
$ ls
```

```
in.melt  log.27Nov18.melt.g++.1  log.27Nov18.melt.g++.4
```

- in.melt の実行

```
$ lammmps < in.melt
```

```
LAMMPS (4 Feb 2020)
```

```
OMP_NUM_THREADS environment is not set. Defaulting to 1 thread.
```

```
(../comm.cpp:90)
```

```
using 1 OpenMP thread(s) per MPI task
```

```
...
```

```
Total # of neighbors = 151513
```

```
Ave neighs/atom = 37.8783
```

```
Neighbor list builds = 12
```

```
Dangerous builds not checked
```

```
Total wall time: 0:00:00
```

エラーメッセージなしで
Total wall time が表示されれば
正常終了

- log.lammmps というログファイルも出力される。

meltの実行(2)

- MPIプロセス数=4 での実行

```
$ mpirun -n 4 lammps < in.melt  
LAMMPS (4 Feb 2020)
```

...

1 by 2 by 2 MPI processor grid

...

Performance: 90089.282 tau/day, 208.540 timesteps/s

22.1% CPU use with 4 MPI tasks x 1 OpenMP threads

...

マシンに4コア以上ある場合

meltの実行(3)



- スレッド数=4での実行

```
$ export OMP_NUM_THREADS=4
```

```
$ echo $OMP_NUM_THREADS
```

```
4
```

```
$ lammmps < in.melt
```

```
LAMMPS (4 Feb 2020)
```

```
using 4 OpenMP thread(s) per MPI task
```

```
...
```

```
Performance: 115561.777 tau/day, 267.504 timesteps/s
```

```
97.6% CPU use with 1 MPI tasks x 4 OpenMP threads
```

```
...
```

マシンに4コア以上ある場合

(補足) OMP_NUM_THREADS指定の削除

```
$ unset OMP_NUM_THREADS
```

```
$ echo $OMP_NUM_THREADS
```

micelle と colloid の実行

- micelle へ移動 → melt と同様の操作

```
$ cd $HOME/lammps/micelle
```

```
$ gzip -d data.micelle.gz
```

```
$ lammps < in.micelle
```

(注) MateriApps LIVE!では、ファイルが gz になっているものがあるので、その時は解凍してから使う。

- colloid へ移動 → melt, micelle と同様の操作

```
$ cd $HOME/lammps/colloid
```

```
$ lammps < in.colloid
```

VMDで可視化する(1)

- LAMMPSでdumpファイルの出力 → 入力ファイルの編集
- \$HOME/lammpas/melt へ移動
\$ cd \$HOME/lammpas/melt
- in.melt のバックアップを取る
\$ cp in.melt in.melt.org : (例)
- melt の入力ファイルをエディタ(emacs)で編集
\$ emacs in.melt &

VMDで可視化する(2)

- emacs の起動

emacs@malive.local

File Edit Options Buffers Tools Help

3d Lennard-Jones melt

```

units      lj
atom_style atomic

lattice    fcc 0.8442
region    box block 0 10 0 10 0 10
create_box 1 box
create_atoms 1 box
mass       1 1.0

velocity   all create 3.0 87287

pair_style lj/cut 2.5
pair_coeff 1 1 1.0 1.0 2.5

neighbor   0.3 bin

```

in.melt Top L1 (Fundamental)

Welcome to GNU Emacs, one component of the GNU/Linux operating system.

[Emacs Tutorial](#) Learn basic keystroke commands
[Emacs Guided Tour](#) Overview of Emacs features at gnu.org
[View Emacs Manual](#) View the Emacs manual using Info
[Absence of Warranty](#) GNU Emacs comes with ABSOLUTELY NO WARRANTY
[Copying Conditions](#) Conditions for redistributing and changing Emacs
[Ordering Manuals](#) Purchasing printed copies of manuals

To quit a partially entered command, type Control-g.

This is GNU Emacs 23.4.1 (i486-pc-linux-gnu, GTK+ Version 2.24.10) of 2012-09-10 on murphy, modified by Debian
Copyright (C) 2012 Free Software Foundation, Inc.

[Dismiss this startup screen](#) Never show it again.

-U:%%- *GNU Emacs* All L3 (Fundamental)

For information about GNU Emacs and the GNU system, type C-h C-a.

①

Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial
Emacs Guided Tour
View Emacs Manual
Absence of Warranty
Copying Conditions
Ordering Manuals

Learn basic keystroke commands
Overview of Emacs features at gnu.org
View the Emacs manual using Info
GNU Emacs comes with ABSOLUTELY NO WARRANTY
Conditions for redistributing and changing Emacs
Purchasing printed copies of manuals

To quit a partially entered command, type Control-g.

This is GNU Emacs 23.4.1 (i486-pc-linux-gnu, GTK+ Version 2.24.10) of 2012-09-10 on murphy, modified by Debian
Copyright (C) 2012 Free Software Foundation, Inc.

[Dismiss this startup screen](#) Never show it again.

② ここ(青字)をクリック

① チェックを入れる

②

emacs@malive.local

File Edit Options Buffers Tools Help

3d Lennard-Jones melt

```

units      lj
atom_style atomic

lattice    fcc 0.8442
region    box block 0 10 0 10 0 10
create_box 1 box
create_atoms 1 box
mass       1 1.0

velocity   all create 3.0 87287

pair_style lj/cut 2.5
pair_coeff 1 1 1.0 1.0 2.5

neighbor   0.3 bin
neigh_modify every 20 delay 0 check no

fix        1 all nve

#dump      id all atom 50 dump.melt

#dump      2 all image 25 image.*.jpg type type &
#           axes yes 0.8 0.02 view 60 -30
#           #dump_modify 2 pad 3

#dump      3 all movie 25 movie.mpg type type &
#           axes yes 0.8 0.02 view 60 -30
#           #dump_modify 3 pad 3

thermo    50
run       250

```

in.melt All L1 (Fundamental)

Wrote /home/user/.emacs

③

VMDで可視化する(3)

- melt の入力ファイル(in.melt)の編集

```
# 3d Lennard-Jones melt

units          lj
atom_style    atomic

lattice        fcc 0.8442
region         box block 0 10 0 10 0 10
create_box     1 box
create_atoms   1 box
mass           1 1.0

velocity       all create 3.0 87287

pair_style     lj/cut 2.5
pair_coeff    1 1 1.0 1.0 2.5

neighbor       0.3 bin
neigh_modify   every 20 delay 0 check no

fix            1 all nve

dump          id all atom 50 dump.melt ← #を取る
#dump          2 all image 25 image.*.jpg type type &
#               axes yes 0.8 0.02 view 60 -30
#dump_modify  2 pad 3

#dump          3 all movie 25 movie.mpg type type &
#               axes yes 0.8 0.02 view 60 -30
#dump_modify  3 pad 3

thermo        50
run           250
```

VMDで可視化する(4)

- 保存して、emacs を閉じる

emacs@malive.local

File Edit Options Buffers Tools Help

保存

```

# 3d Lennard-Jones melt

units          lj
atom_style     atomic

lattice        fcc 0.8442
region         box block 0 10 0 10 0 10
create_box     1 box
create_atoms   1 box
mass           1 1.0

velocity       all create 3.0 87287

pair_style     lj/cut 2.5
pair_coeff    1 1 1.0 1.0 2.5

neighbor       0.3 bin
neigh_modify   every 20 delay 0 check no

fix            1 all nve

#dump          id all atom 50 dump.melt

#dump          2 all image 25 image.*.jpg type type &
#               axes yes 0.8 0.02 view 60 -30
#dump_modify   2 pad 3

#dump          3 all movie 25 movie.mpg type type &
#               axes yes 0.8 0.02 view 60 -30
#dump_modify   3 pad 3

thermo        50
run           250
...:... in.melt All L1 (Fundamental)...

```

Wrote /home/user/.emacs

VMDで可視化する(5)

- 編集した in.melt の実行

```
$ lammps < in.melt
```

エラーメッセージなしで
Total wall time が表示されれば
正常終了

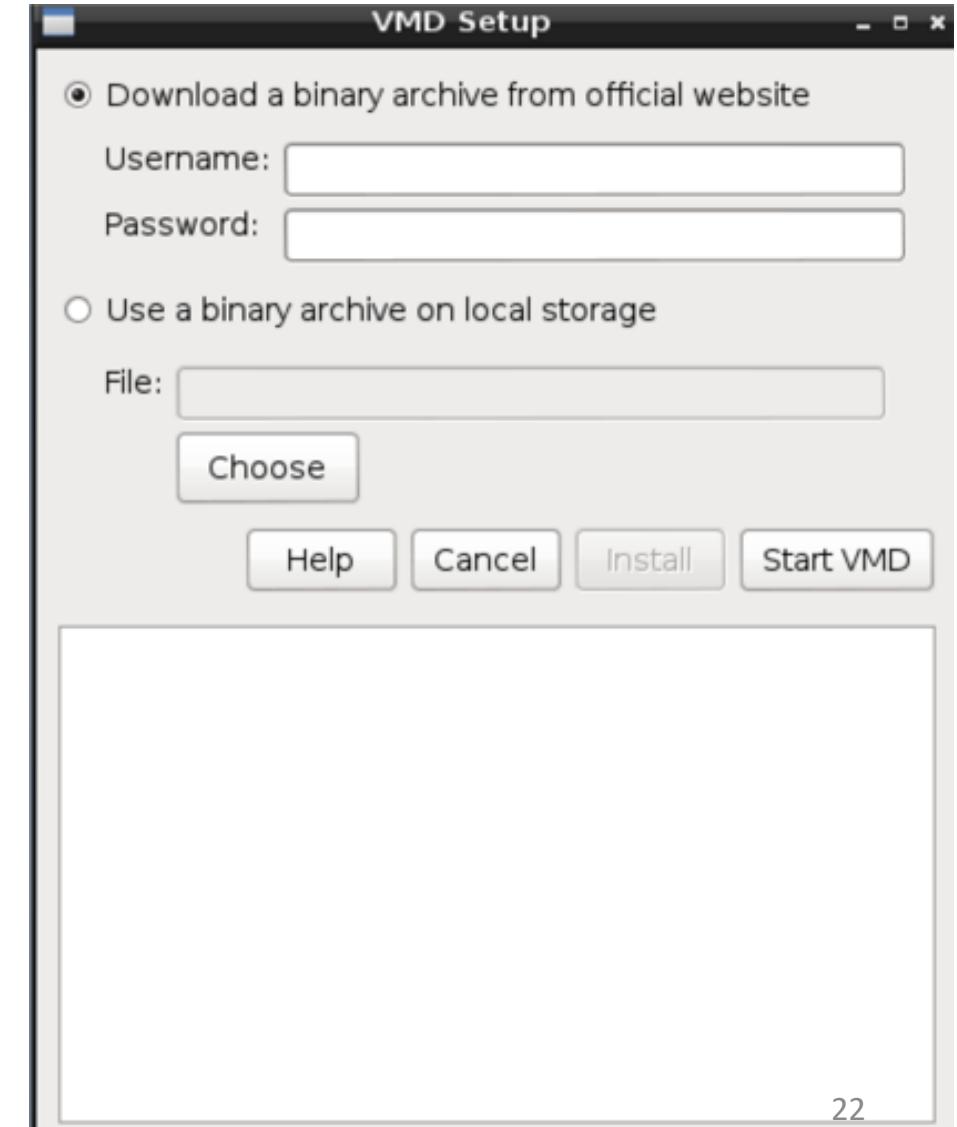
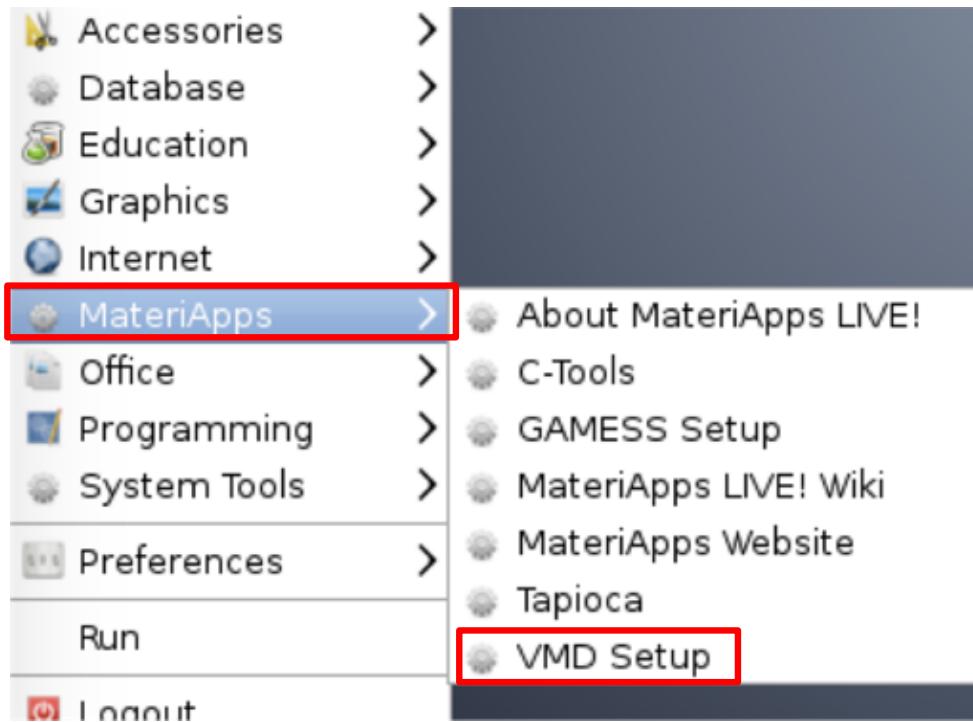
- dump.melt というファイルができているか確認する。

```
$ ls
```

dump.melt	in.melt.org	log.5Oct16.melt.g++.4
in.melt	log.5Oct16.melt.g++.1	log.lammps

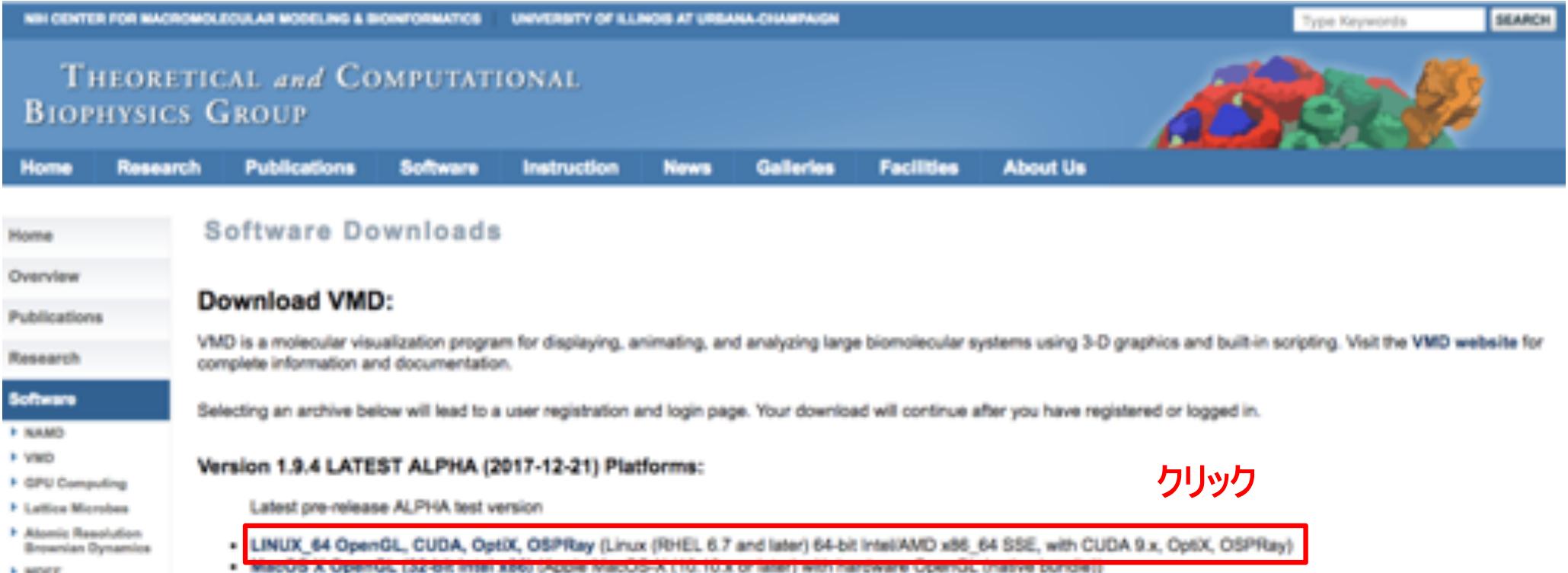
VMDで可視化する(6)

- VMDの起動
「MateriApps」の「VMD Setup」を選択



VMDで可視化する(7)

- VMDのユーザ登録
(すでにユーザIDとパスワードをお持ちの方は必要ありません。)
- VMDのウェブサイトへアクセスする。
「Download VMD」でgoogle検索する。
Download VMD - Theoretical and Computational Biophysics Group
<https://www.ks.uiuc.edu/Development/Download/download.cgi?PackageName=VMD>
のサイトへアクセス



The screenshot shows the homepage of the Theoretical and Computational Biophysics Group. The top navigation bar includes links for Home, Research, Publications, Software, Instruction, News, Galleries, Facilities, and About Us. The Software menu item is highlighted. The main content area features a banner with a molecular visualization image and the text "THEORETICAL and COMPUTATIONAL BIOPHYSICS GROUP". Below the banner, the "Software Downloads" section is visible, with a sub-section titled "Download VMD:". It describes VMD as a molecular visualization program and provides a link to the VMD website for more information. A sidebar on the left lists software categories: NAMD, VMD, GPU Computing, Lattice Molecules, Atomic Resolution Brownian Dynamics, and MDFF. The "Software Downloads" section also lists "Version 1.9.4 LATEST ALPHA (2017-12-21) Platforms:" with options for Linux (OpenGL, CUDA, OptiX, OSPRay) and Mac OS X (OpenGL, Intel Xeon (Apple Mac OS X), Intel iMac (Intel Mac OS X), Intel Mac mini (Intel Mac OS X)). The "LINUX_64 OpenGL, CUDA, OptiX, OSPRay (Linux (RHEL 6.7 and later) 64-bit Intel/AMD x86_64 SSE, with CUDA 9.x, OptiX, OSPRay)" option is highlighted with a red box.

クリック

VMDで可視化する(8)



- usernameとpasswordを入力
(新しく自分で決める。)

すでにユーザIDとパスワードをお持ちの方は、この作業は必要ありません。

Registration/Login

You will need a username and password to download software.

If this is your first download, please choose a username and password to register.

Current NAMD or VMD users, please enter your existing username and password.

Username:

① UsernameとPasswordを入力
(注) rist2018は使えません

Password:

② ボタンを押す

Your download will continue after you have registered or logged in.

VMDで可視化する(9)

- ユーザ登録をする

New User Registration

New User Registration for 'rist2018':

First and Last Name:

Kanako Yoshizawa

Email Address:

yoshizawa@rist.or.jp

Affiliation:

Academic Government Industrial Other (specify)

The number of people using TCBG software at my site is:

1 2-4 5-10 11-20 21 or more

I use TCBG software primarily for:

Research Teaching Commerce Personal

The work I do with TCBG software is funded (at least partially) by NIH:

Yes No

Re-enter password for confirmation:

Register

すでにユーザIDとパスワードをお持ちの方は、この作業は必要ありません。

① ご自身の情報を入力

② ボタンを押す

VMDで可視化する(10)



- 承諾をする

Software Downloads

Welcome! Account created for 'rist2018'.

Please remember your password for future downloads.

You may avoid logins for 6 months by saving a cookie on your browser:

すでにユーザIDとパスワードをお持ちの方は、この作業は必要ありません。

VMD 1.9.4 for Linux (RHEL 6.x) 64-bit x86_64 w/ SSE, CUD

To download this software you must agree to abide by the terms of the following license:

UNIVERSITY OF ILLINOIS VISUAL MOLECULAR DYNAMICS SOFTWARE LICENSE AGREEMENT

Upon execution of this Agreement by the party identified below ("Licensee"),
The Board of Trustees of the University of Illinois ("Illinois"), on behalf
of The Theoretical and Computational Biophysics Group ("TCBG") in the
Beckman Institute, will provide the Visual Molecular Dynamics ("VMD") software in
Executable Code and/or Source Code form ("Software") to Licensee, subject to the
following
terms and conditions. For purposes of this Agreement, Executable Code is the
compiled code, which is ready to run on Licensee's computer. Source code
consists of a set of files which contain the actual program commands that are
compiled to form the Executable Cod

ボタンを押す

I am Kanako Yoshizawa and I agree to the terms of this License

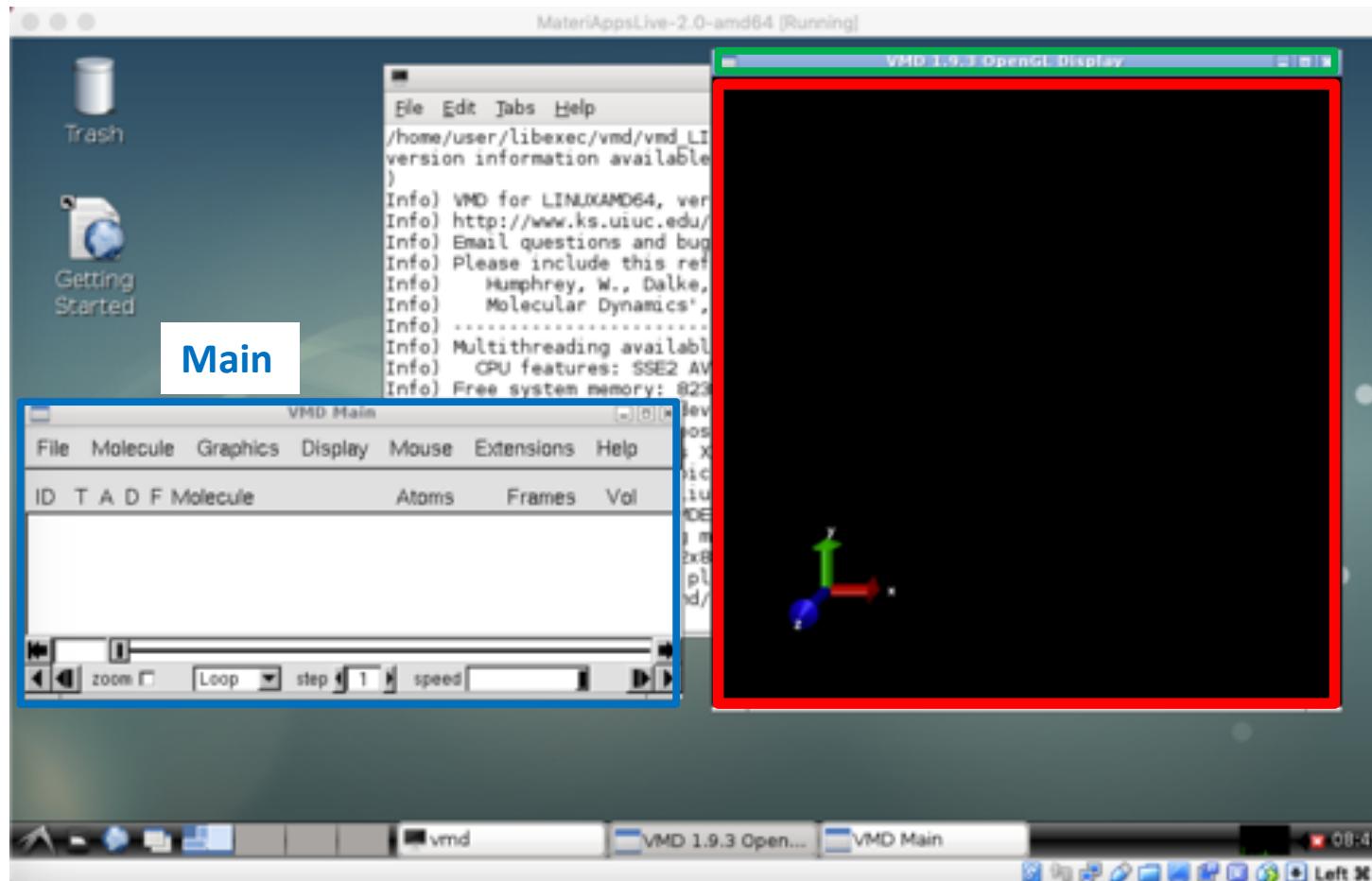
VMDで可視化する(11)

- 「VMD Setup」において、usernameとpasswordを入力



VMDで可視化する(12)

- VMDのウィンドウ

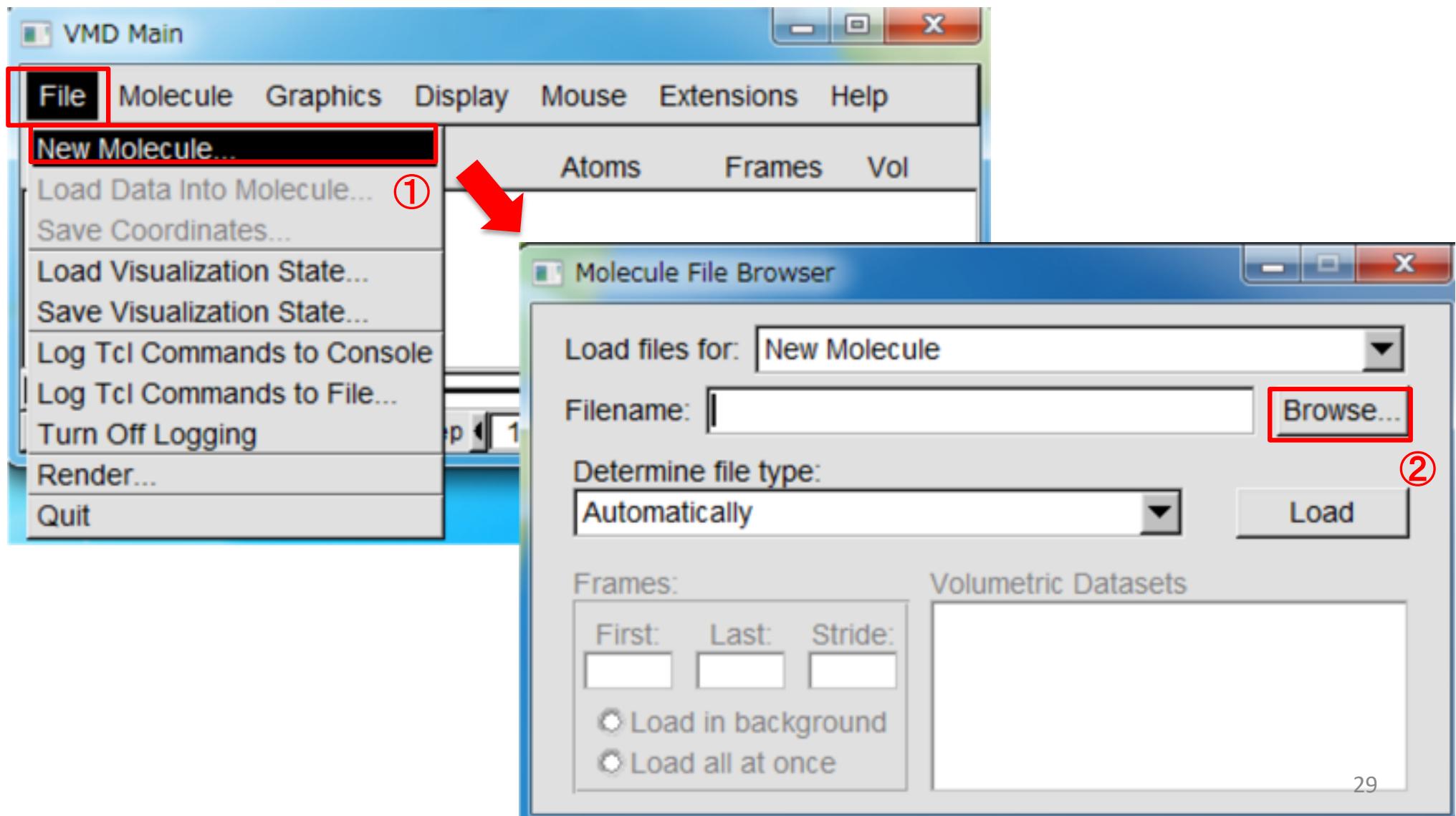


(注) 緑の部分が隠れてしまい、ウィンドウが移動できないときは、

- Mac の場合
option を押しながら ドラッグ
- Windows の場合 ALT を押しながら ドラッグ
すると、赤で囲まれたウィンドウを移動できる。

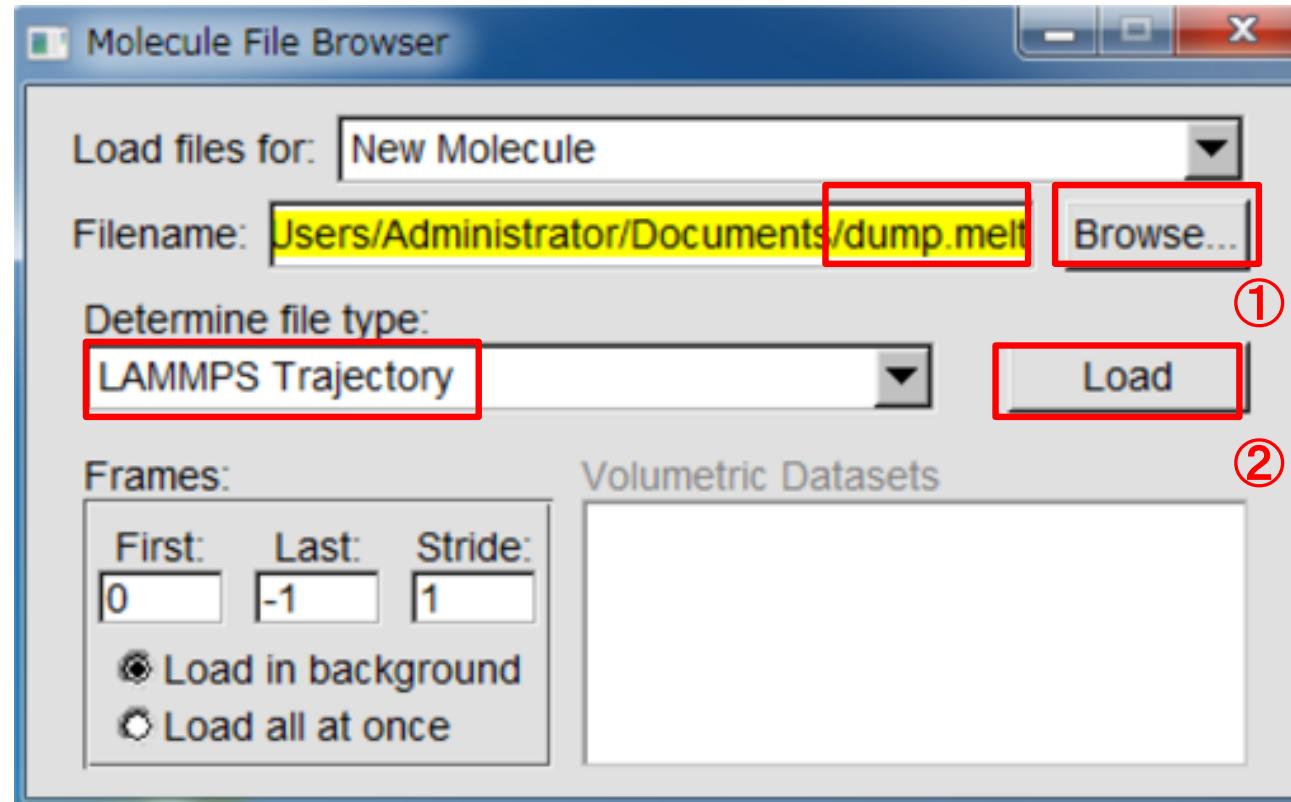
VMDで可視化する(13)

- dump.melt を開く



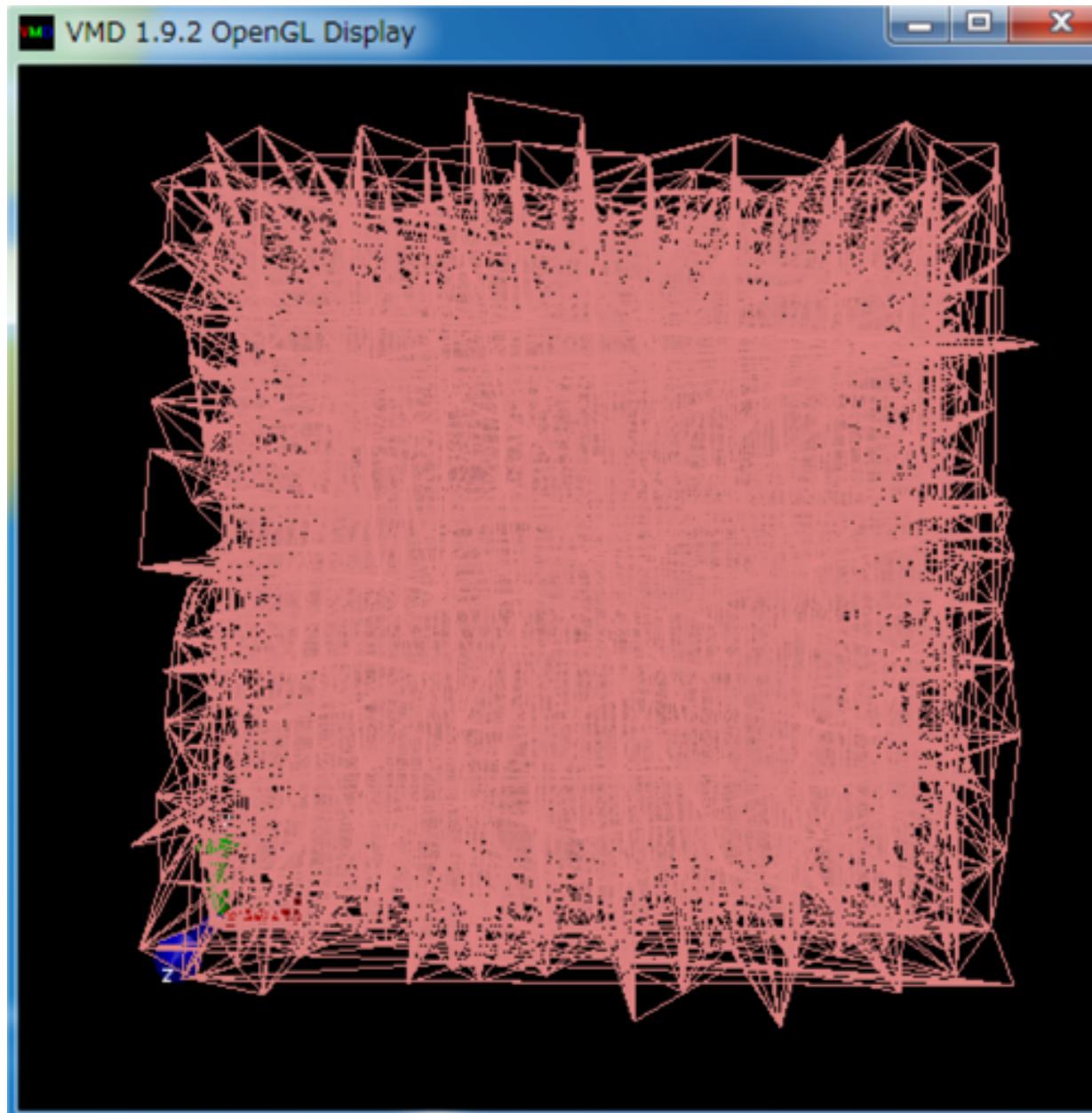
VMDで可視化する(14)

- ファイルタイプの選択 → 「LAMMPS Trajectory」



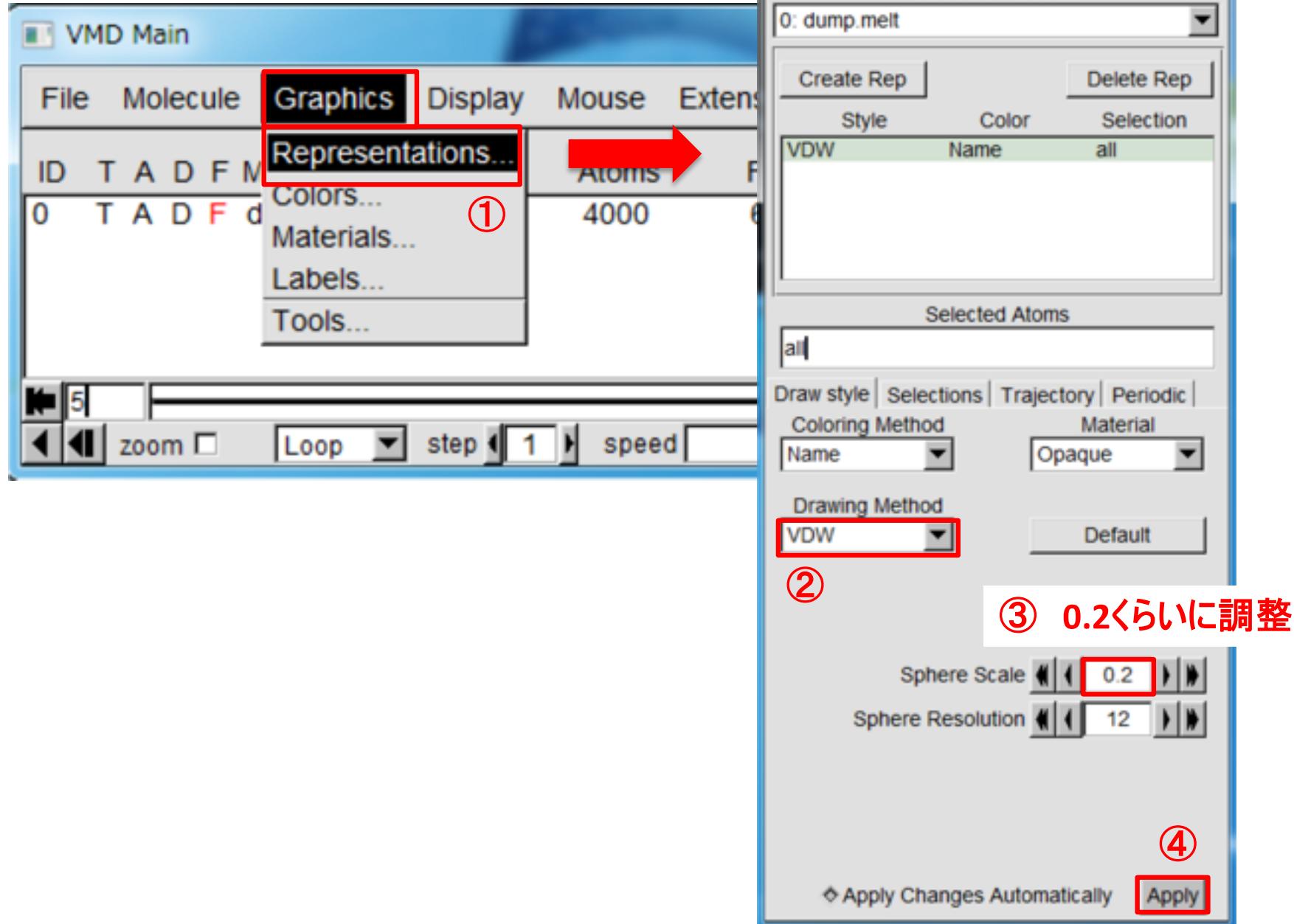
VMDで可視化する(15)

- ディスプレイの起動(Linesの表示)



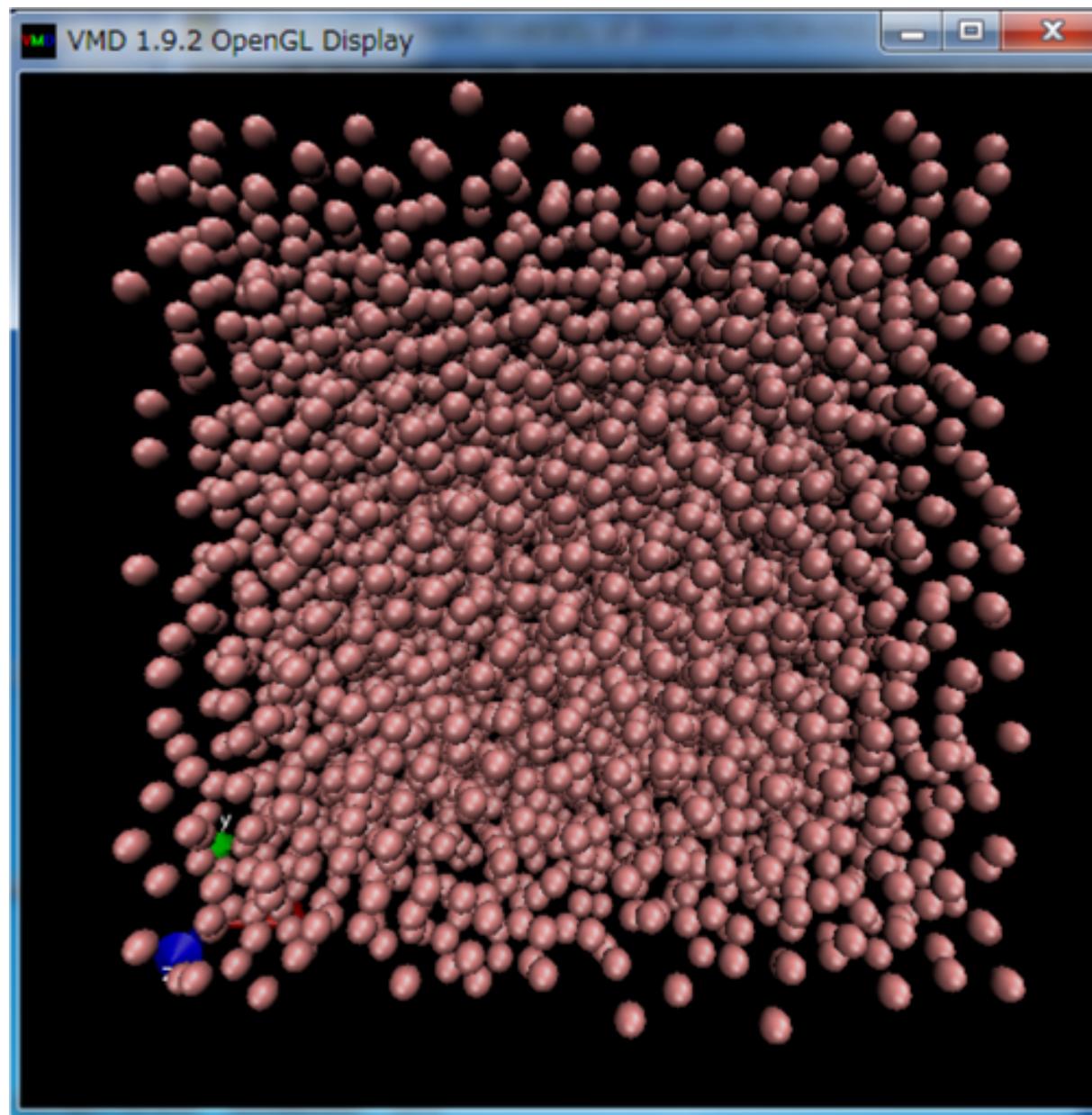
VMDで可視化する(16)

- 表示形式の調整



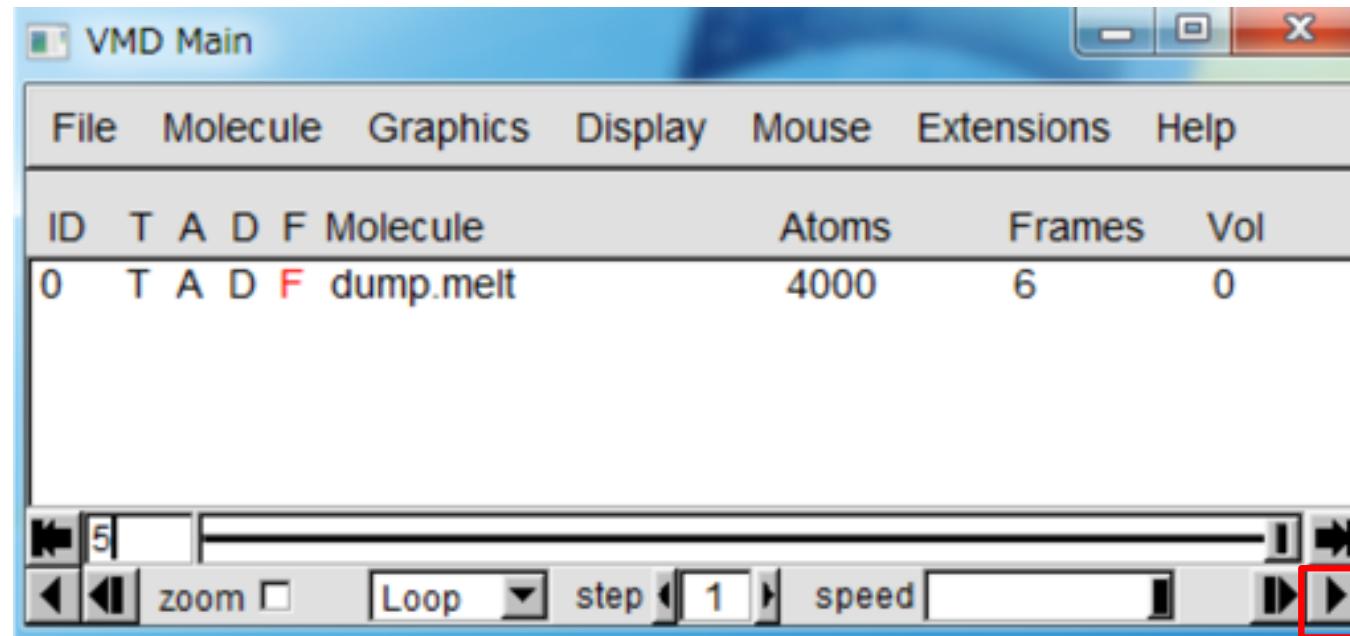
VMDで可視化する(17)

- ディスプレイ(VDWの表示)



VMDで可視化する(18)

- MDスタート



ここをクリック

micelle と colloid の可視化の課題

1. micelle と colloid に対しても dumpファイルを出力
2. micelle と colloid の dumpファイルをVMDで可視化

micelle の可視化(1)

```
# 2d micelle simulation

dimension 2      2次元の計算

neighbor 0.3 bin
neigh_modify delay 5

atom_style bond

# Soft potential push-off

read_data data.micelle      データファイル data.micelle を読み込む
special_bonds fene

...
fix 1 all nve      NVEアンサンブル
...
thermo 1000

dump 1 all atom 2000 dump.micelle ← # を取る
...

```

micelle の可視化(2)



- 編集した in.micelle の実行

```
$ lammps < in.micelle
```

エラーメッセージなしで

Total wall time が表示されれば
正常終了

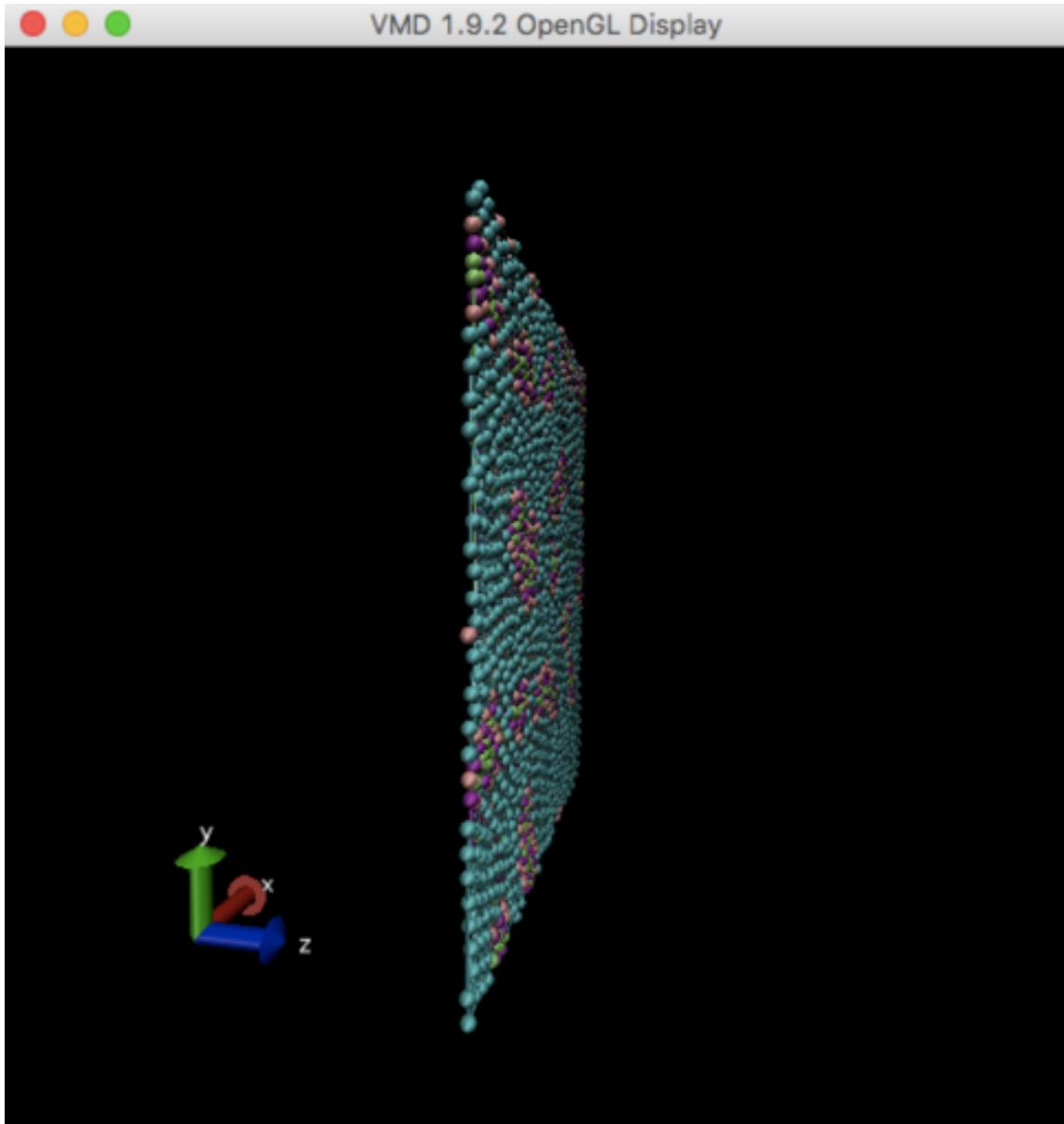
- dump.micelle というファイルができているか確認する。

```
$ ls
```

```
data.micelle  dump.micelle  log.5Oct16.micelle.g++.1.gz  log.lammps  
def.micelle   in.micelle   log.5Oct16.micelle.g++.4.gz
```

micelle の可視化(3)

- dump.micelle を開く



可視化をすると
計算している系が
わかりやすい。

micelle の3次元化の課題

1. 2次元の micelle が、3次元になるように入力ファイルを編集する
 2. これらの結果をVMDで可視化する
-
- 作業ディレクトリを作る。(例: \$HOME/lammps/micelle3d など)
\$ cd \$HOME/lammps
\$ mkdir micelle3d
 - micelle(colloid)ディレクトリから in.micell(in.colloid) をコピーする
\$ cp ./micelle/in.micelle ./micelle3d
\$ cp ./micelle/data.micelle ./micelle3d

micelleの3D化について(1)

1. 変更点(inファイル)

- Dimension文コメントアウト
- Replicateコマンド(read_dataの下)
replicate 1 1 36

– NPT計算に変更

```
fix 1 all npt temp 1.0 1.0 1.0 iso 3.0 3.0 10.0
```

(計算時間がかかるようなら、run 60000 の回数を減らす)

2. 変更点(dataファイル)

– PBC-boxの大きさ

0.0000000E+00 0.996023889 zlo zhi

ここで、 $0.996023889 = 35.85686/36$

micelleの3D化について(2)

in.micelle(入力ファイル)

```
# 3d micelle simulation
# dimension 2
neighbor 0.3 bin
neigh_modify delay 5
atom_style bond
# Soft potential push-off
read_data data.micelle
replicate 1 1 36
special_bonds fene
pair_style soft 1.12246
pair_coeff * * 0.0 1.12246
bond_style harmonic
bond_coeff 1 50.0 0.75
velocity all create 0.45 2349852
variable prefactor equal ramp(1.0,20.0)
fix 1 all npt temp 1.0 1.0 1.0 iso 3.0 3.0 10.0
fix 2 all temp/rescale 100 0.45 0.45 0.02 1.0
fix 3 all adapt 1 pair soft a * * v_prefactor
# fix 4 all enforce2d
thermo 50
run 1000
unfix 3
```

→ 続く

dimension 2
2次元の設定のコメントアウト
→ 3次元

read_data data.micelle
data.micelle のデータファイルが読み込まれる

replicate 1 1 36
z方向に36個複製する

fix 1 all npt temp 1.0 1.0 1.0 iso 3.0 3.0 10.0
1 (=ID)という名称で、全て (=all) の粒子に対して、NPT 積分を行う。温度 (=temp) は、計算開始時は 1.0、終了時は 1.0、時間単位 1.0 で damp する。iso は、外圧を指定する。

micelleの3D化について(3)

in.micelle(入力ファイル)

```
→ 続く          pair_coeff    1 3 1.0 1.0 1.12246
                pair_coeff    1 4 1.0 1.0 1.12246

# Main run          thermo      1000

pair_style    lj/cut 2.5          dump        1 all atom 2000 dump.micelle

# solvent/head - full-size and long-range          #dump        2 all image 2000 image.*.jpg type type
                                                    zoom 1.6
                                                    #dump_modify  2 pad 5 adiam 1 0.5 adiam 2 1.5
                                                    adiam 3 1.0 adiam 4 0.75

pair_coeff    1 1 1.0 1.0 2.5
pair_coeff    2 2 1.0 1.0 2.5
pair_coeff    1 2 1.0 1.0 2.5

# tail/tail - size-averaged and long-range          #dump        3 all movie 2000 movie.mpg type type
                                                    zoom 1.6
                                                    #dump_modify  3 pad 5 adiam 1 0.5 adiam 2 1.5
                                                    adiam 3 1.0 adiam 4 0.75

pair_coeff    3 3 1.0 0.75 2.5
pair_coeff    4 4 1.0 0.50 2.5
pair_coeff    3 4 1.0 0.67 2.5          reset_timestep 0
                                         run        60000 計算時間がかかるようなら
                                         # solvent/tail - full-size and repulsive          減らす
```

micelleの3D化について(5)



data.micelle(データファイル)

LAMMPS 3d micelle data file

1200 atoms

1 1.000000

300 bonds

2 1.000000

0 angles

3 1.000000

0 dihedrals

4 1.000000

0 impropers

Atoms

4 atom types

1 139 2 0.000 0.000 0.000

1 bond types

2 0 1 1.195 0.000 0.000

0 angle types

3 0 1 2.390 0.000 0.000

0 dihedral types

4 0 1 3.586 0.000 0.000

0 improper types

5 0 1 4.781 0.000 0.000

...

0.0000000E+00 35.85686 xlo xhi

0.0000000E+00 35.85686 ylo yhi

0.0000000E+00 0.996023889 zlo zhi

Masses

43

データファイルのフォーマット

データファイル

LAMMPS Description

5 atoms
4 bonds
1 atom types
1 bond types

-10.0 10.0 xlo xhi
-10.0 10.0 ylo yhi
-10.0 10.0 zlo zhi

Masses

1 1

Atoms

1 1 1 0.0 0.0 0.0
2 1 1 0.0 0.0 1.0
3 1 1 0.0 0.0 2.0
4 1 1 0.0 0.0 3.0
5 1 1 0.0 0.0 4.0

Bonds

1 1 1 2
2 1 2 3
3 1 3 4
4 1 4 5

※ 速度データを含む場合
(AtomsとBondsの間)

Velocities

1 0.5 0.5 0.5

データフォーマット

Masses

atom type番号, 質量

周期境界条件
のwrap数

Atoms

粒子番号, Mol-ID, atom type, x, y, z, wx, wy, wz

Velocities

粒子番号, vx, vy, vz

Bonds

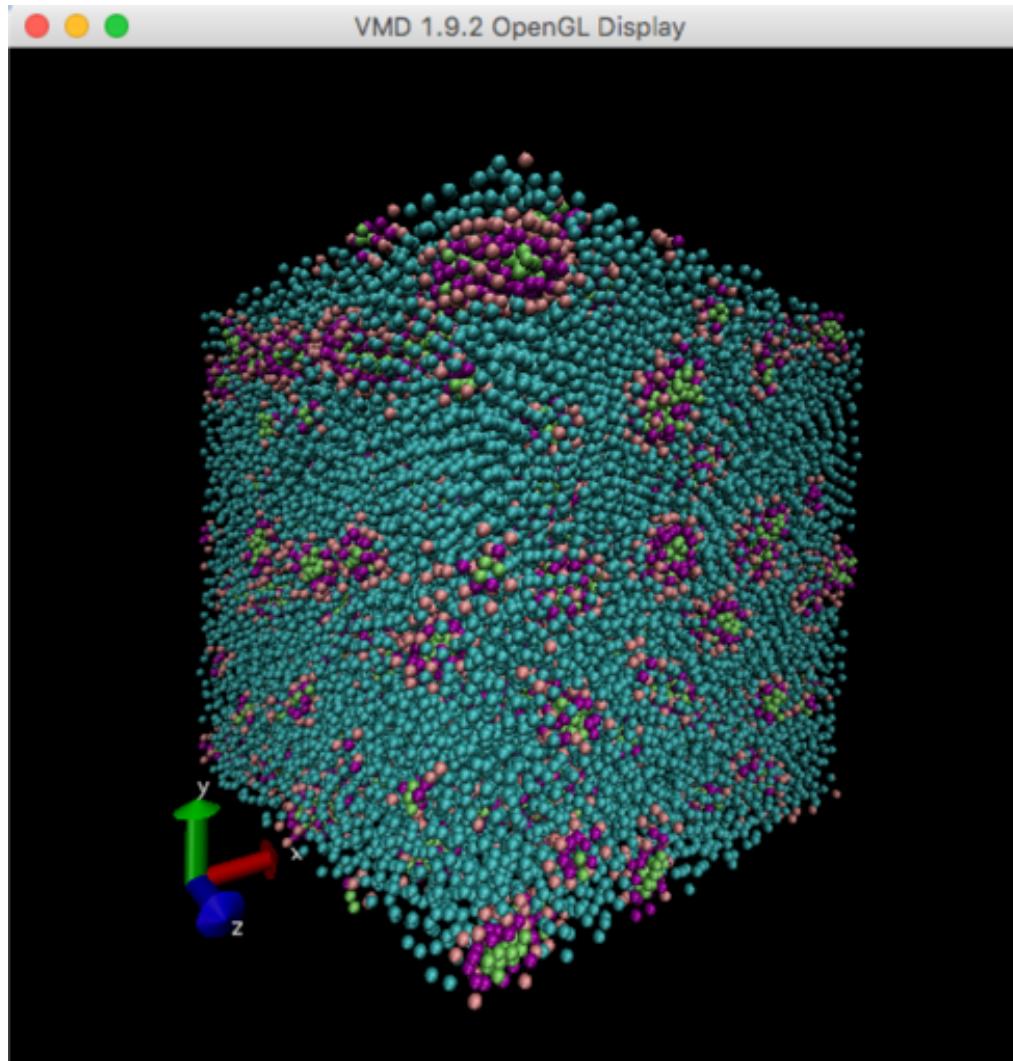
ボンド番号, bond type, bond1, bond2

※ MassesとAtomsの間に、力場 パラメータを記録する場合もある。

※ Bondsの後に、Angles, Dihedrals, Impropersを記録する場合もある。

micelleの3D化について(4)

- in.micelle と data.micelle の編集後、実行
\$ lammps < in.micelle
- dump.micelle を開く



VMDで可視化すると、3D になつたことが確認出来る

サイズを大きくした計算(1)

- 作業ディレクトリを作る。(例: \$HOME/lammps/melt2)
\$ cd \$HOME/lammps
\$ mkdir melt2
- 作成した作業ディレクトリへ移動する。(\$HOME/lammps/melt2)
\$ cd melt2
- meltディレクトリから in.melt をコピーする
\$ cp ../melt/in.melt ./
- melt の入力ファイルを開く
\$ emacs in.melt

サイズを大きくした計算(2)

```
# 3d Lennard-Jones melt

units          lj
atom_style     atomic

lattice        fcc 0.8442
region         box block 0 40 0 40 0 40      256,000原子になる
create_box     1 box
create_atoms   1 box
mass           1 1.0

velocity       all create 3.0 87287

pair_style     lj/cut 2.5
pair_coeff    1 1 1.0 1.0 2.5

neighbor       0.3 bin
neigh_modify   every 20 delay 0 check no

fix            1 all nve
```

サイズを大きくした計算(3)

- 編集した in.melt の実行

```
$ lammmps < in.melt
```

- ログファイルの確認

```
$ cat log.lammps
```

...

```
Created 256000 atoms
```

...

サイズを大きくした計算(4)

examples/melt

実行時間(時:分:秒) iMac:
2.8 GHz Intel Core i7

	手持ちのPC(iMac)	
プロセス数	1	4
4000原子1,000ステップ	0:00:02	0:00:01
4000原子10,000ステップ	0:00:29	0:00:11
256,000原子1,000ステップ	0:03:06	0:01:13
256,000原子10,000ステップ	0:31:15	0:12:48
2,048,000原子1,000ステップ	0:27:04	0:10:06
2,048,000原子10,000ステップ		1:46:38

フラットMPI

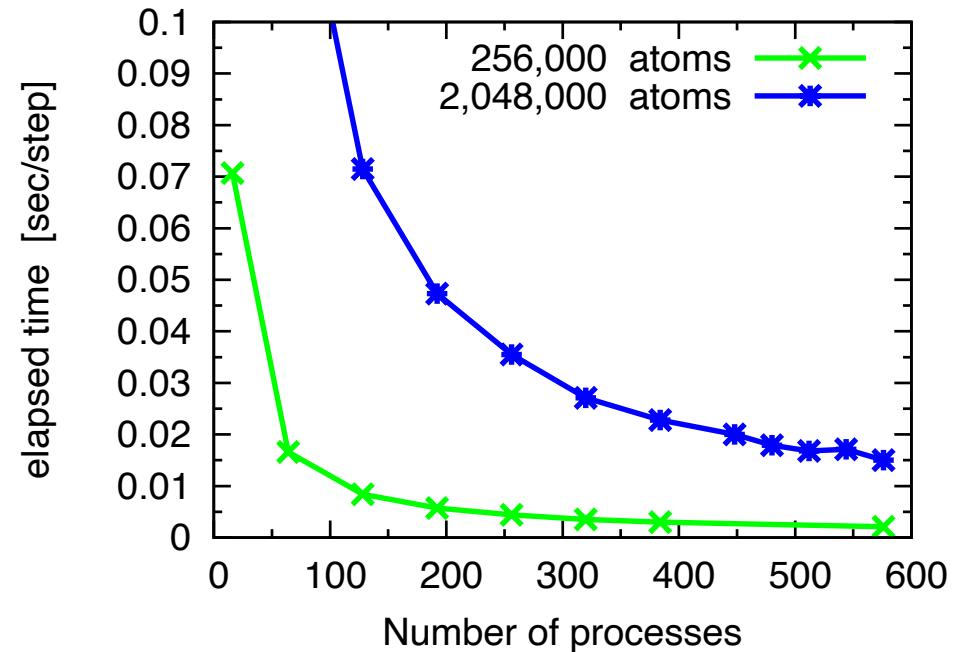
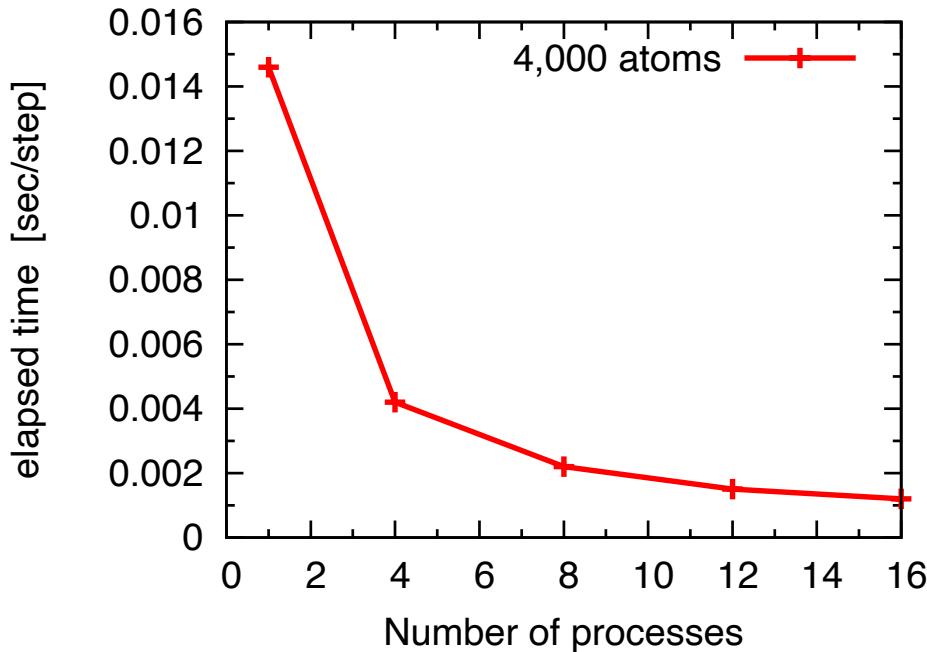
FX10		
16	256	576
0:00:01	0:00:05	
0:00:12	0:00:03	
0:01:11	0:00:04	0:00:02
0:11:47	0:00:44	0:00:21
0:08:56	0:00:36	0:00:15
1:28:35	0:05:55	0:02:30

- 主にプロセス数当たりの粒子数減少に伴う実行時間の減少
→ 計算規模(原子数など)が小さくて、プロセス数を増やすと、通信等により実行時間が増加する場合がある。
- 計算規模が大きくなる → [スパコン\(HPCI\)の利用](#)へ

サイズを大きくした計算(5)

examples/melt

フラットMPI(原子数:4,000、256,000、2,048,000)



4,000原子: ~16プロセス
 256,000原子: ~600プロセス
 2,048,000原子: 600プロセス以上