

2020年8月6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)

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(演習)
 - melt を実行する
 - ✓ 計算結果を可視化する
 - ✓ エネルギーをプロットする
 - micelle を実行する
 - ✓ 計算結果を可視化する
 - rerun を実行する
 - ✓ 動径分布関数(RDF)をプロットする
 - サイズを大きくした計算
- HPCIについて

LAMMPSについて

- Large-scale **A**tomic/**M**olecular **M**assively **P**arallel **S**imulator
<https://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
<https://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
<https://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。
- 例：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
https://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のパッケージ内にある potentials の
/usr/share/lammps/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)
/usr/share/lammps/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: <https://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

MateriApps LIVE!: `/usr/share/lammps/examples`

- examples の解説

<https://lammps.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

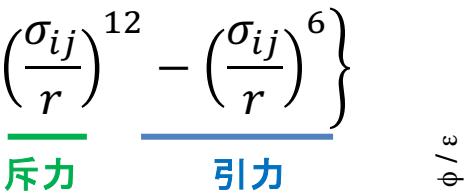
高分子物理向けのExamples

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

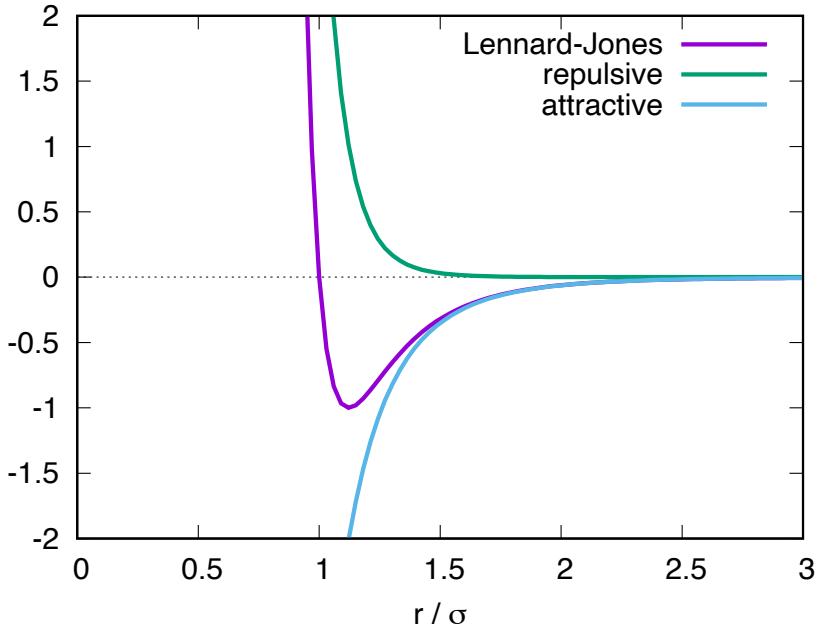
meltについて

- melt: FCC 格子状に並んだ Lennard-Jones (LJ) 粒子が溶けて拡散
- Lennard-Jones ポテンシャル

$$\phi_{ij}(r) = 4\epsilon_{ij} \left\{ \left(\frac{\sigma_{ij}}{r} \right)^{12} - \left(\frac{\sigma_{ij}}{r} \right)^6 \right\}$$


斥力 引力

- ポテンシャルの傾きが力として働く。
- 分子同士が十分離れているとき($r \rightarrow$ 大)、力は働くかない。
- 分子同士が近づくと、引力が働くようになり、 $r \sim 1.12$ 付近でエネルギーが最小となり、分子が近づきすぎると斥力(反発力)が働き、エネルギーは急激に増大する。



- ◆ 引力と斥力を持ち、希ガスの性質(ファンデルワールス力)を良く表す粒子モデルとして知られている。
- ◆ 古典MD計算のモデルポテンシャルとしてよく用いられる。

examples/melt(1)

入力ファイル: in.melt

```
# 3d Lennard-Jones melt
```

units lj
atom_style atomic

Lennard-Jones (LJ)
古典粒子

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

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。

velocity all create 3.0 87287

速度の設定

pair_style lj/cut 2.5

pair_coeff 1 1 1.0 1.0 2.5

力場パラメータの設定 (ljポテンシャルを指定)

neighbor 0.3 bin
neigh_modify every 20 delay 0 check no

neighborでは、ポテンシャルcutoffよりどれだけ外側までを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 &      # は、コメントアウト
#       axes yes 0.8 0.02 view 60 -30           dump:出力形式の指定
#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 何ステップ毎に温度情報を出力するかの指定
run    250 計算実行(step)数
```

詳しくは、

Commands

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

を参照

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)

```
$ cd $HOME
```

```
$ mkdir lammps
```

- 作成した作業ディレクトリへ移動する。(\$HOME/lammps)

```
$ cd lammps
```

- examplesファイルのコピー (melt, micelle, colloid)

```
$ cp -r /usr/share/lammps/examples/melt ./
```

```
$ cp -r /usr/share/lammps/examples/micelle ./
```

```
$ cp -r /usr/share/lammps/examples/rerun/ ./
```

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 neights/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.lammps 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

マシンに4コア以上ある場合

```
$ mpirun -oversubscribe -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

...

meltの実行(3)

- スレッド数=4 での実行

```
$ OMP_NUM_THREADS=4 lammps < 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

melt: 計算結果を可視化する(1)

- melt の入力ファイル(in.melt)を編集する。
- 可視化用の入力ファイル in.melt.viz を作成する。
 \$ cp in.melt in.melt.viz
- エディタ(emacs)を起動する。
 \$ emacs in.melt.viz &

melt: 計算結果を可視化する(2)

- emacs の設定

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](#) 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.

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
#           pad 3

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

thermo    50
run       250

```

in.melt All L1 (Fundamental)-----

Wrote /home/user/.emacs

③

melt: 計算結果を可視化する(3)

- melt の入力ファイル(in.melt.viz)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 5 delay 0 check no
fix            1 all nve

dump          id all atom 5 dump.melt
...
```

20 から 5 に変更する

#を取り、四角で囲まれた内容
に修正する

melt: 計算結果を可視化する(4)

- 編集した in.melt.viz の実行

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

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。

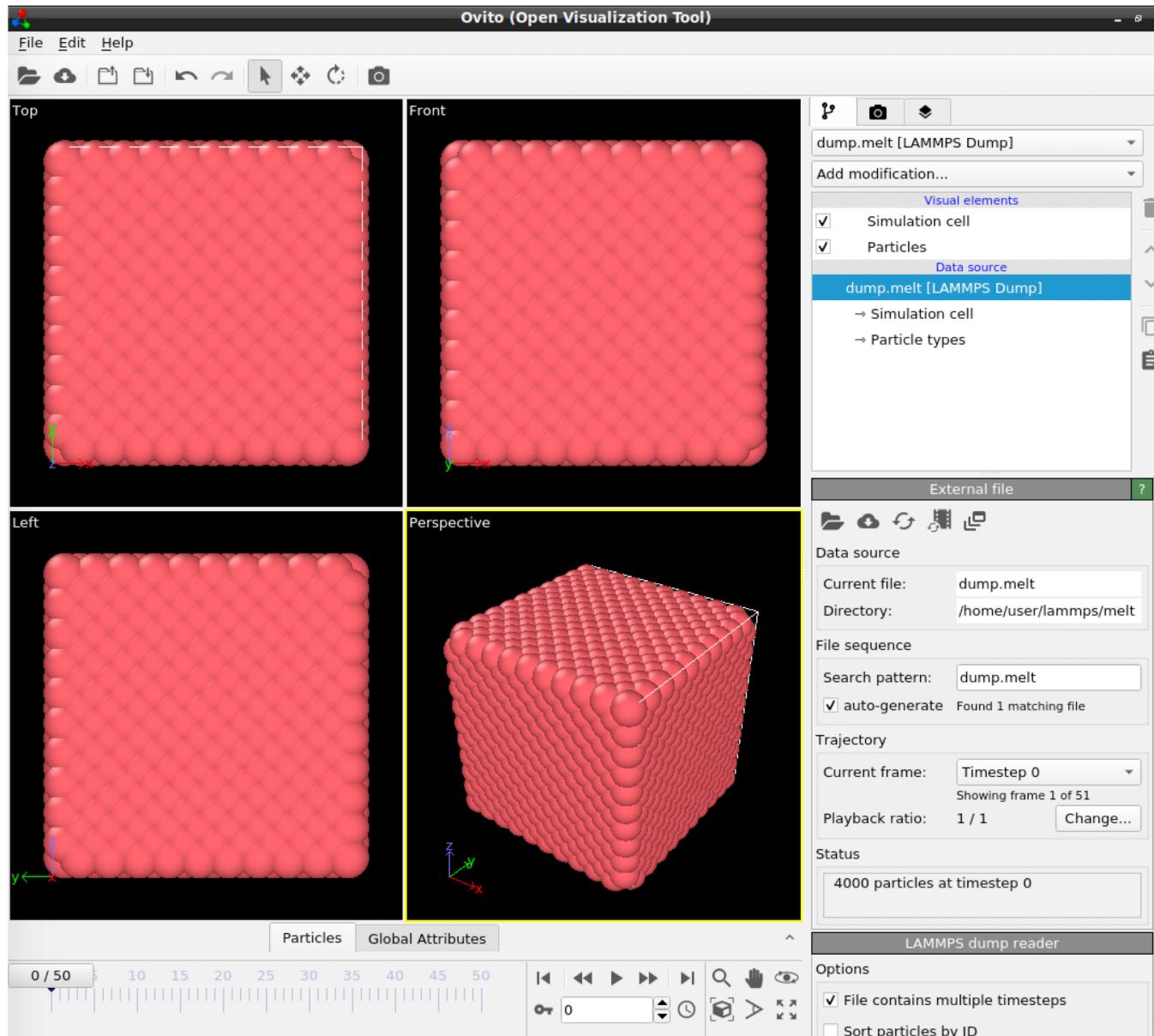
```
$ ls  
dump.melt    in.melt.viz      log.lammps
```

- MateriApps LIVE! (仮装マシン) 上で、OVITO で可視化する。

```
$ ovito dump.melt
```

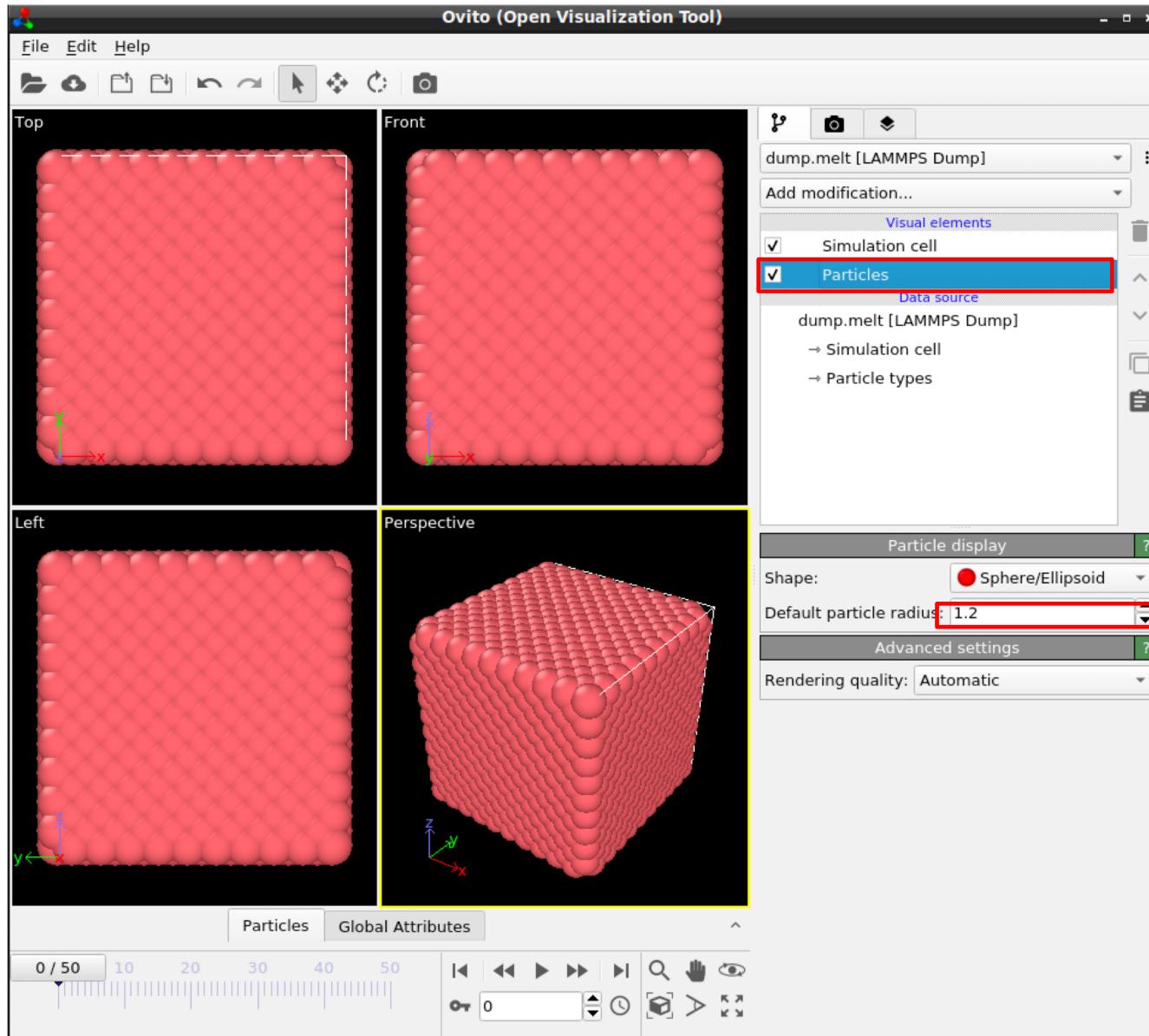
melt: 計算結果を可視化する(5)

- OVITO の起動



melt: 計算結果を可視化する(6)

- 粒子のサイズの変更

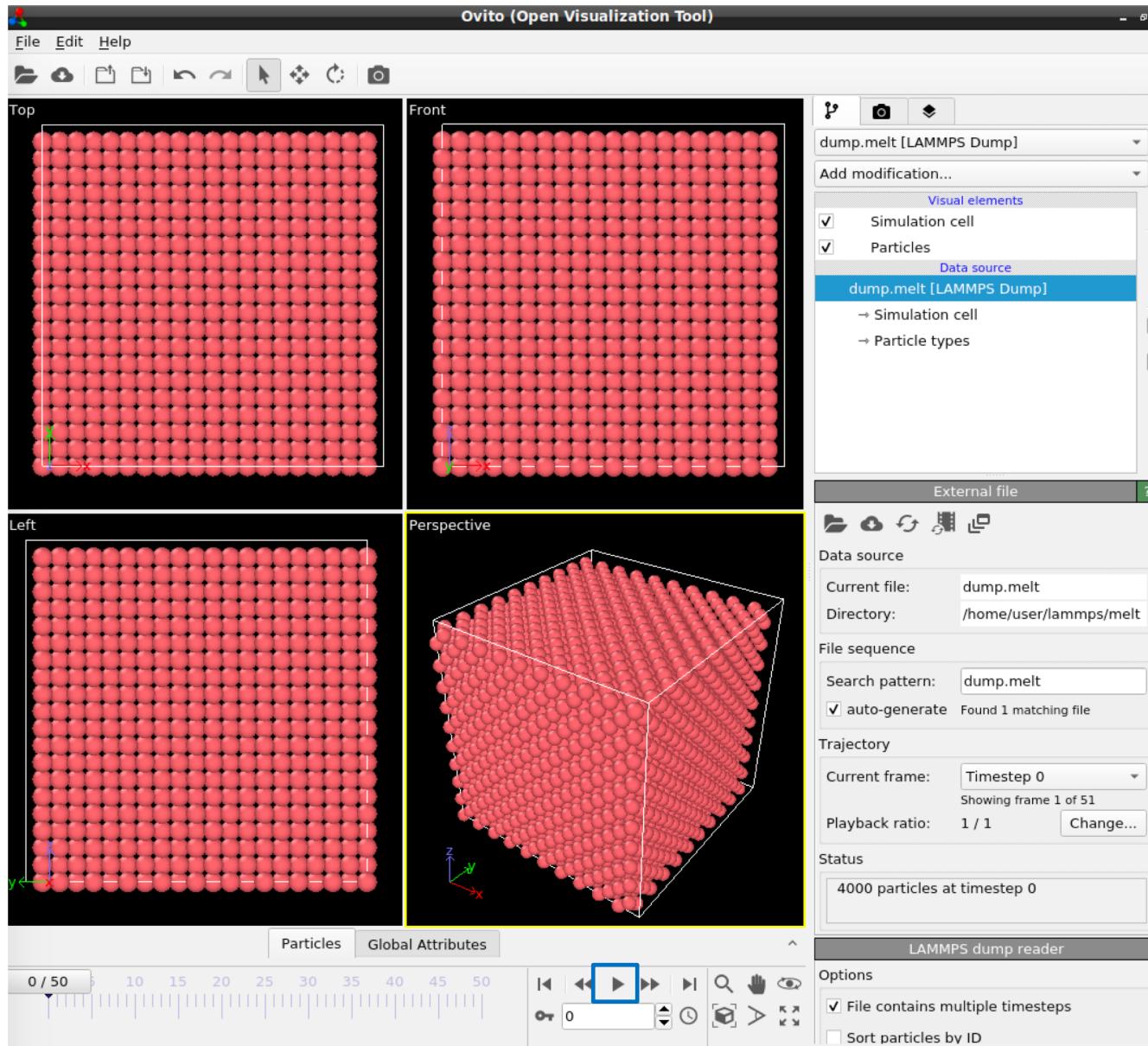


①「Particle」を選択する。

② 0.5 程度に設定する。

melt: 計算結果を可視化する(7)

- 動画再生



スタートボタン
「▶」を押す

melt: 計算結果を可視化する(8)

- ホストマシンの OVITOで可視化する。
- ホストマシンに dump.melt をコピーする。
(「Shared」というファイルが共有化されているとする。)
\$ cp dump.melt /media/sf_Shared

melt: エネルギーをプロットする(1)

- エネルギーを出力するの入力ファイル in.melt.erg を作成する。
\$ cp in.melt.viz in.melt.erg
- エディタ(emacs)を起動する。
\$ emacs in.melt.erg &

melt: エネルギーをプロットする(2)

- melt の入力ファイル(in.melt.erg)の編集

```
# 3d Lennard-Jones melt

units          lj
atom_style     atomic

...
neighbor      0.3 bin
neigh_modify   every 5 delay 0 check no

fix           1 all nve                                         追記する

variable ST equal step          ST にステップ数を割り当てる
variable ETOT equal etotal     ETOT にトータルエネルギーを割り当てる

fix    energy all print 2 "${ST} ${ETOT}" file out.erg screen no

dump        id all atom 5 dump.melt
...

```

melt: エネルギーをプロットする(3)

- 編集した in.melt.erg の実行

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

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

- out.erg というファイルができているか確認する。

```
$ ls
```

```
out.erg      dump.melt      in.melt.erg      log.lammps
```

- gnuplot を用いてプロットする。

```
$ gnuplot
```

```
...
```

GNUPLOT

Versin 5.2 patchlevel 6

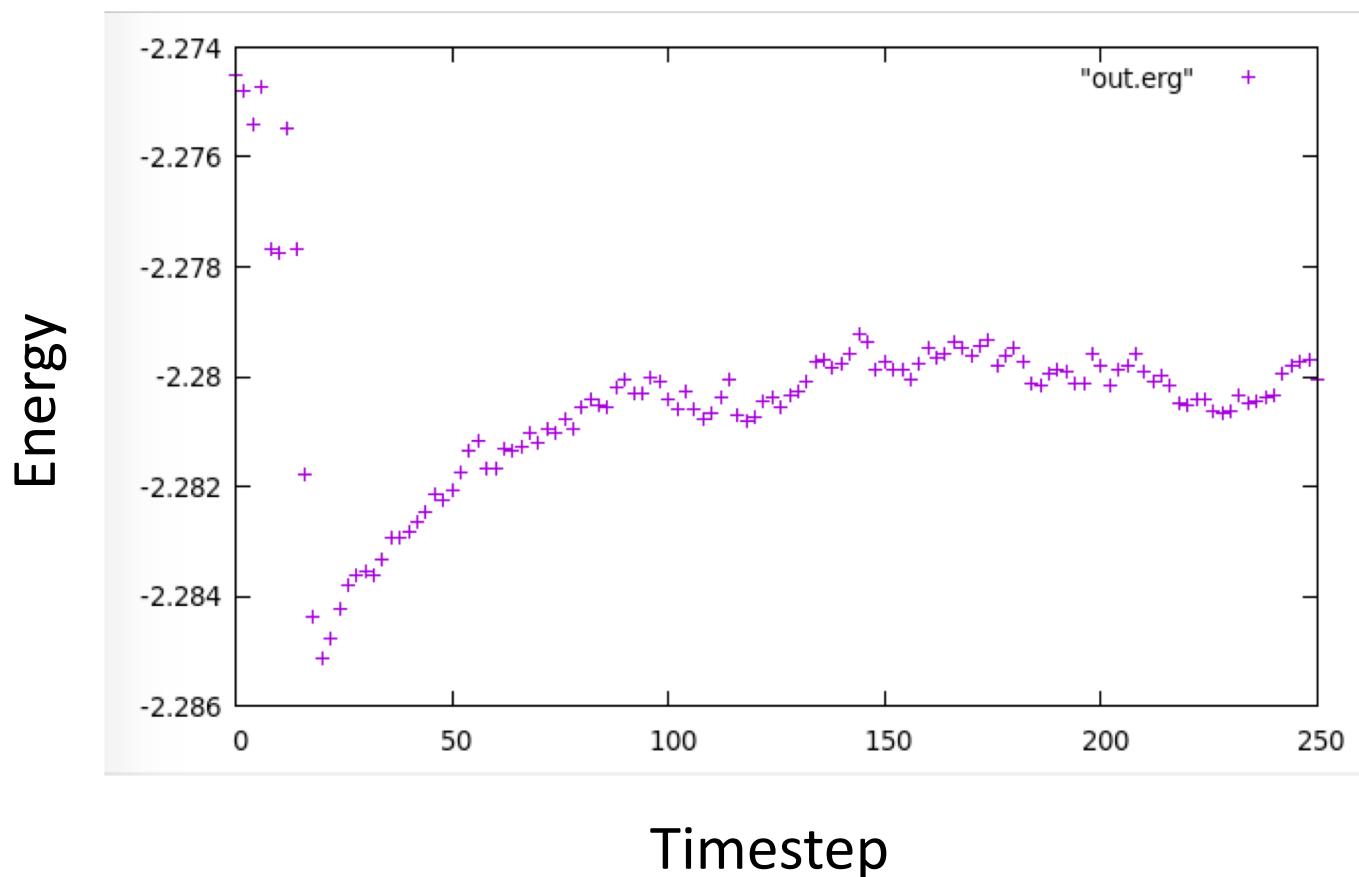
last modified 2019-01-01

```
...
```

Terminal type is now 'wxt'

```
gnuplot> plot "out.erg"
```

melt: エネルギーをプロットする(4)



micelle の実行

- micelle: 会合する分子の2次元の計算
micelle へ移動 → melt と同様の操作

\$ cd \$HOME/lammps/micelle

\$ lammps < in.micelle

(注) 4 Feb 2020 の実行では、

WARNING: Communication cutoff 1.42246 is shorter than a bond length based estimate of 1.425. This may lead to errors. (../comm.cpp:686)

の警告ができるが(カットオフを大きくすると回避できる)、この演習においては無視してください。
(examples の中にあるログファイル 29 Mar 2020 の実行においては、この警告は出ない。)

→ 自分で入力ファイルを作成する時は警告は出ないようにした方が良い。

micelle の可視化(1)

- micell の入力ファイル(in.micell)の編集

```
# 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 50

dump 1 all atom 2000 dump.micelle
```

← #を取る

micelle の可視化(2)

```
#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

#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

reset timestep 0
run    100000
```

ステップ数を増やす

micelle の可視化(2)

- 編集した in.micelle の実行

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

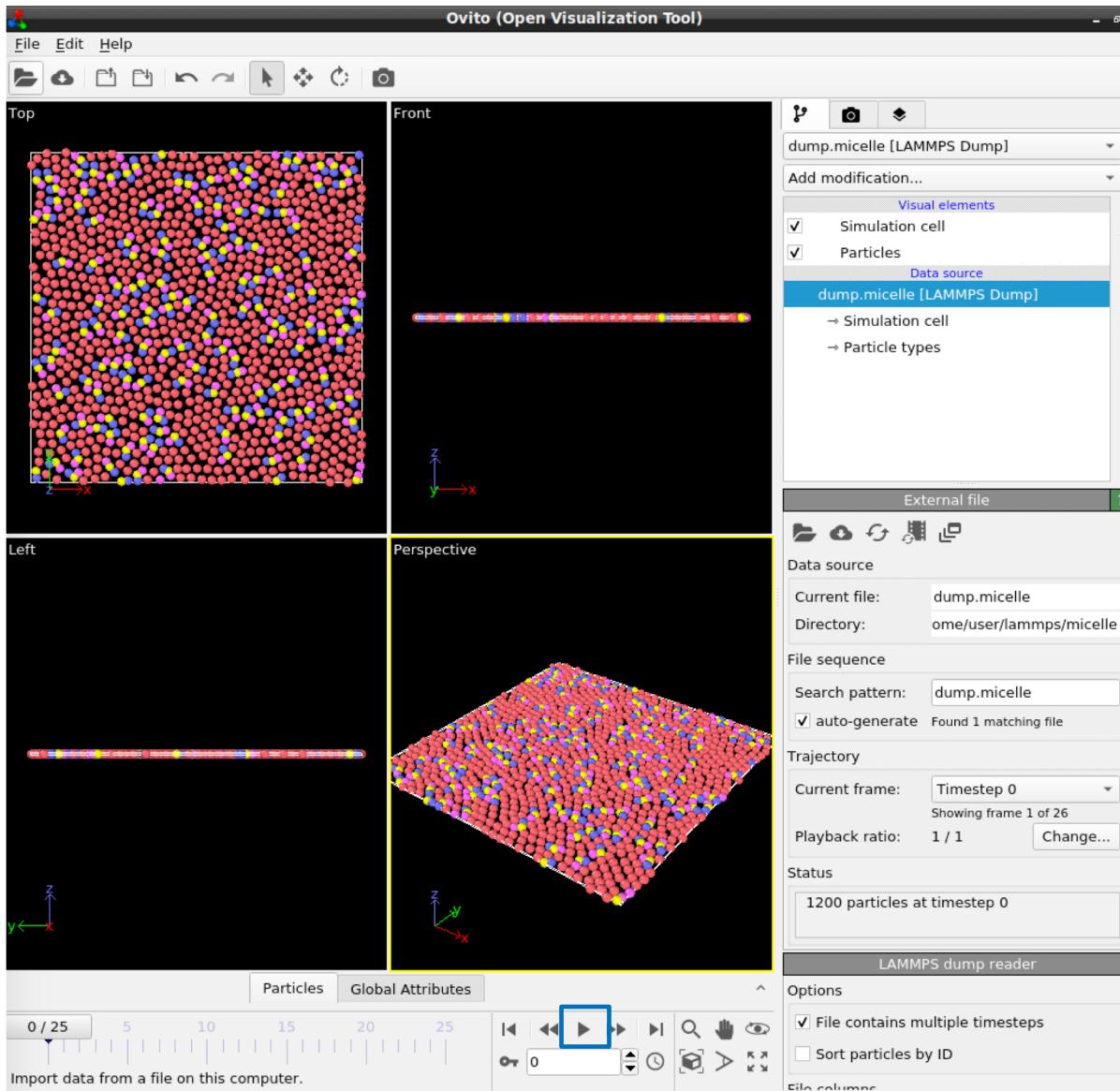
エラーメッセージなしで
Total wall time が表示されれば
正常終了

- 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.lammps  
def.micelle in.micelle
```

micelle の可視化(3)

- dump.micelle を開く
\$ ovito dump.micelle



スタートボタン
「▶」を押す

rerun: 動径分布関数(RDF)をプロットする(1)



- rerun へ移動
\$ cd \$HOME/lammps/rerun
- 入力ファイル in.rdf.first の内容の確認
\$ less in.rdf.first

```
# 3d Lennard-Jones melt

variable      x index 1
variable      y index 1
variable      z index 1
...
fix          1 all nve

dump         1 all custom 100 lj.dump id type x y z

compute     myRDF all rdf 50 cutoff 2.5      半径 2.5 の距離において、50 点計算
fix          2 all ave/time 100 10 1000 c_myRDF[*] file rdf.first mode vector
thermo      100                                タイムステップ 100、110、120、130、140、150、160、170、180、190、
run         1000                               1000 の平均値を、1000 ステップ目における値とする。
```

rerun: 動径分布関数(RDF)をプロットする(2)



- in.rdf.first の実行

```
$ lammmps < in.rdf.first
```

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

- rdf.first というファイルができているか確認する。

```
$ ls  
rdf.first    lj.dump    in.rdf.first    log.lammps
```

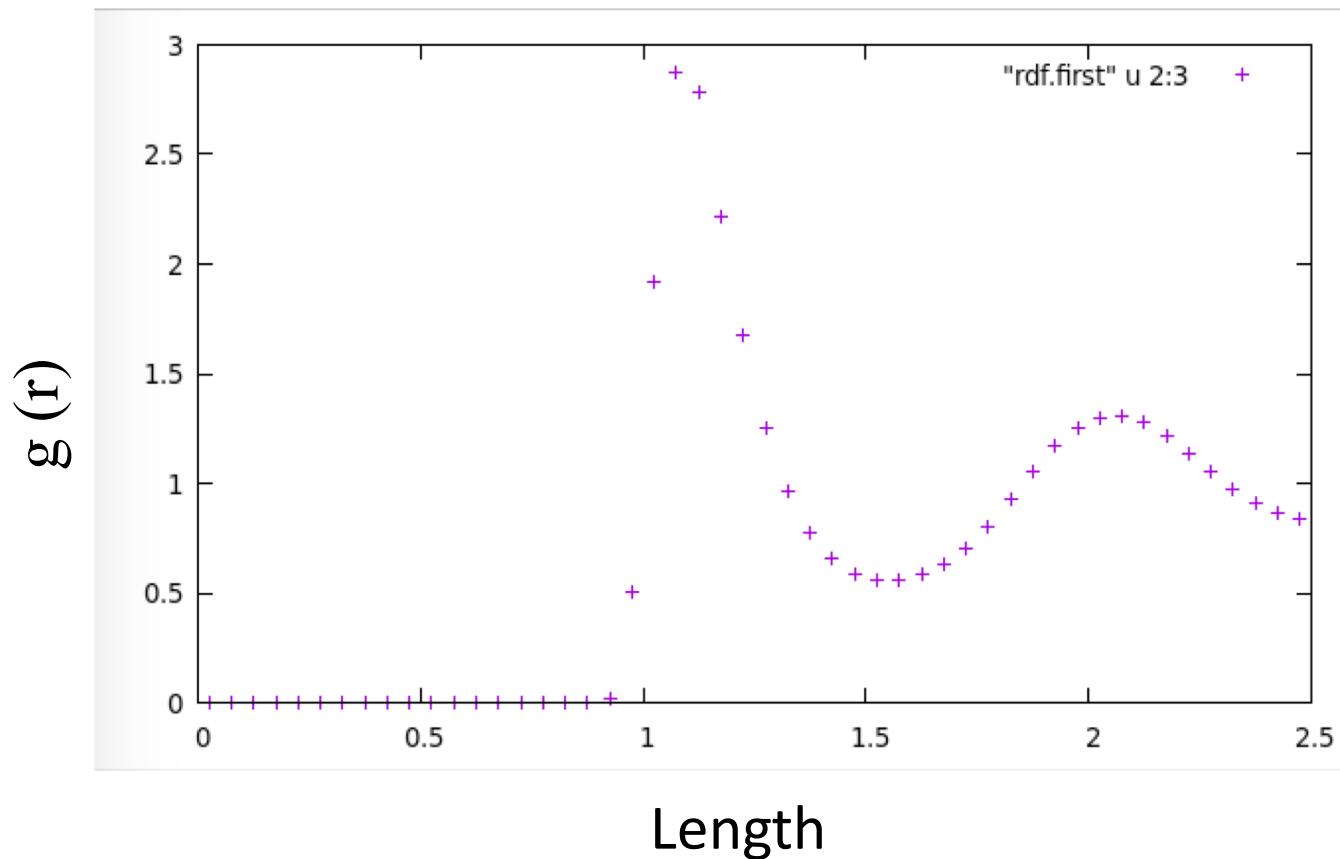
- gnuplot を用いてプロットする。

```
$ gnuplot
```

...

```
gnuplot> plot "rdf.first" u 2:3
```

rerun: 動径分布関数(RDF)をプロットする(3)



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

```
$ lammps < 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

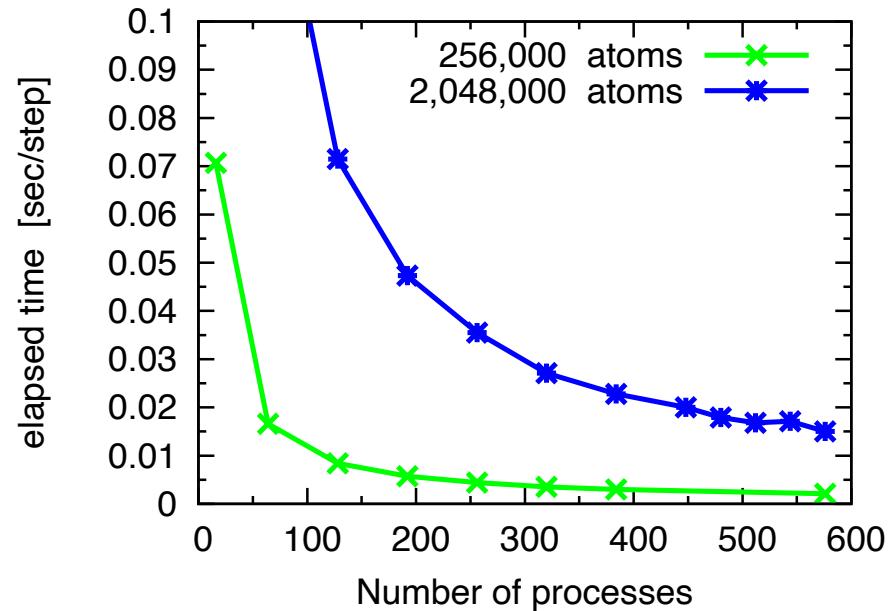
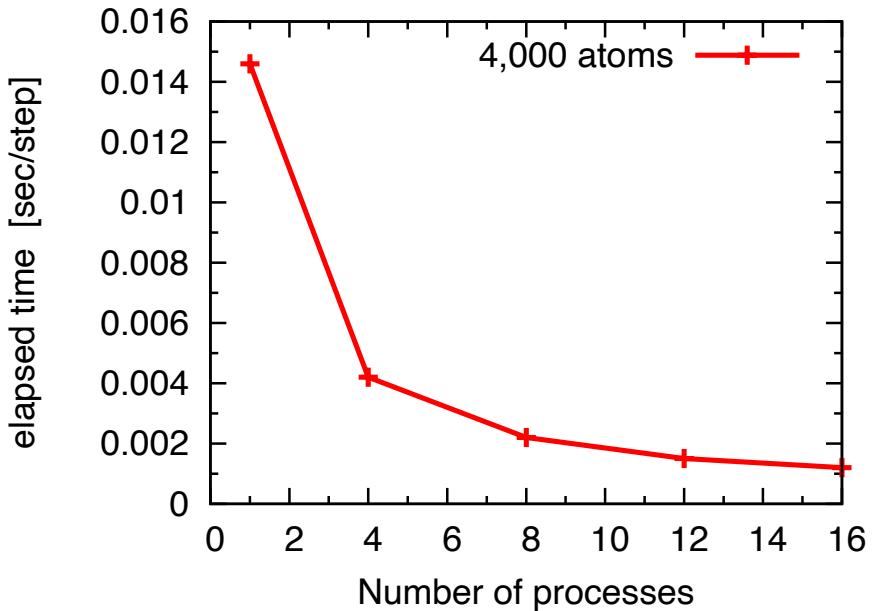
FX10		
16	256	576
0:00:01	0:00:05	
0:00:12	0:00:03	
0:01:11	0:00:04	0:00:02
0:11:47	0:00:44	0:00:21
0:08:56	0:00:36	0:00:15
1:28:35	0:05:55	0:02:30

- 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