物性研究・電子版 Vol. 5, No. 2, 052601（2016年5月号）

平成27年度 修士論文

短距離古典分子動力学法における加速化手法の系統的構築：

近接リスト-セル分割ハイブリッド法とリスト半径・セルサイズの最適化

新潟大学大学院自然科学研究科 博士前期課程 2 年
数理物質科学専攻　物理学コース

F14A001B

淡路　大輔

概要

短距離古典分子動力学 (Molecular Dynamics: MD) 法における計算時間短縮の手法として，相互作用する近傍粒子を効率よく探索する方法である Verlet の近接リスト法 (Verlet Neighbor List: VNL 法) とセル分割法 (Cell Linked List: CLL 法) は従来からよく用いられている．本論文では両者を組み合わせたハイブリッドな手法 (VNL-CLL 法) を提案し，その最適化の方法について述べている．VNL 法はカットオフ半径より少し大きなリスト半径を設けておき，その範囲内に存在する近傍粒子をリストに登録，各 MD ステップごとにそのリストに含まれる近傍粒子との相互作用を計算する方法である．そこでは粒子配置が時間発展によって変化するため数ステップ毎にリストを更新しなければならない．一般にはこの更新頻度は 10 ステップ程度に固定することが多いが，リストを動的に更新する手法を採用することにより計算時間の短縮を図ることができる．この動的更新型の VNL 法では，計算システム (粒子数，温度，密度，カットオフ半径) に対して計算コストを最適とするリスト半径がある事が従来までに分かっているが，さらなる高速化のためには各パラメータ依存性の検討が必要である．一方，CLL 法では計算の対称としている系全体を一辺がカットオフ半径よりもわずかに大きい立方体セルに分割する．これにより粒子間相互作用は，同じセルに属する粒子間，もしくは隣接セルに属する粒子間に限定される．隣接するセルの数は限られており (2 次元で 8 個，3 次元で 26 個)，相互作用の数を大幅に減らすことが出来る．さらにセル分割数を増加させ，より細分化したセルを用いることによって参照体積を減らす．これにより従来型の CLL 法と比べて，計算の加速化を図ることができる．この手法は排他的セル分割法と呼ばれ，相互作用 (カットオフ半径) が非常に短距離的である粉体のような系に対して大きな効果がある．VNL 法において通常は全粒子探索によりリストを再構築するが，これには $\mathcal{O}(N^2)$ の計算コストを要する．しかしながら，リスト更新に CLL 法を用いて近傍粒子探索をすることによりその計算コストを $\mathcal{O}(N^2)$ から $\mathcal{O}(N)$ まで低減することができる．この手法を VNL-CLL 法と呼ぶ．本研究では VNL-CLL 法において Verlet の近接リスト更新の際に排他的セル分割法を用い，計算時間のセルサイズ (あるいはセル分割数) 依存性を系統的に明らかにした．また，近接リストに対する動的更新の手法を導入し，VNL 法の結果に基づき，計算時間のリスト半径依存性を計算システム (粒子数，温度，密度，カットオフ半径) を変化させ，網羅的に検討した．その結果，計算時間を最小化する最適なリスト半径，セルサイズの組み合わせが計算システムに依存すること見出し，さらなる加速化のためにリスト半径，セルサイズの最適化の手法を構築した．

目次

概要 i

第 1 章 序論 1
1.1 分子間相互作用 1
1.2 運動方程式の数値積分 3
1.3 周期境界条件 4
1.4 古典統計力学の基礎理論 5
1.5 本論文の目的と構成 7

第 2 章 ベルレの近接リスト法 9
2.1 概要 9
2.2 SKIN(リスト半径) の最適化 12

第 3 章 セル分割法 17
3.1 概要 17
3.2 排他的セル分割法 20
3.3 セルサイズの最適化 23

第 4 章 VNL-CLL 法 26
4.1 概要 27
4.2 リスト半径・セルサイズの最適化 29

第 5 章 結論 34

謝辞 36

参考文献 37

第 1 章

序論

分子動力学法 (Molecular Dynamics: MD 法) はモンテカルロ法 (Monte Carlo: MC 法) と並ぶ分子シミュレーションの手法の一つである．MC 法はハミルトニアンから，系の状態を実現するボルツマン分布に従う粒子配置を乱数を用いて求め，平衡状態における熱力学量のカノニカル平均を計算する手法である．これに対し，MD 法は運動方程式を数値積分して粒子運動の時間発展を求める．そのため熱平衡状態における熱力学量ばかりではなく，原子・分子の動的挙動についても有用な情報を得る事が可能である．さらに，外場があるような非平衡状態における粒子系についてもシミュレーションを行うことができる利点がある．本章ではまず，MD 法の概要について紹介する [1–4].

1.1 分子間相互作用

MD 法と MC 法，どちらを用いて分子シミュレーションを行うにしても，粒子間に働く相互作用を設定する．希ガス原子間に働く力は 2 体相互作用 (対ポテンシャル， pair potential) によってよく記述され，希ガス以外に単純な金属やイオン系に対しても用いられる．もし，構造をもつ分子を考えれば 3 体力，4 体力が無視できなくなる．

質量が m である単成分球状 N 粒子系のポテンシャルエネルギーを $U_N(\{\boldsymbol{r}_i\})$ と書く事にする．$\boldsymbol{r}_i$ は i 番目の粒子の座標ベクトルであり，$\{\boldsymbol{r}_i\}$ はすべての粒子の座標を与えて初めて決まる関数である事を示

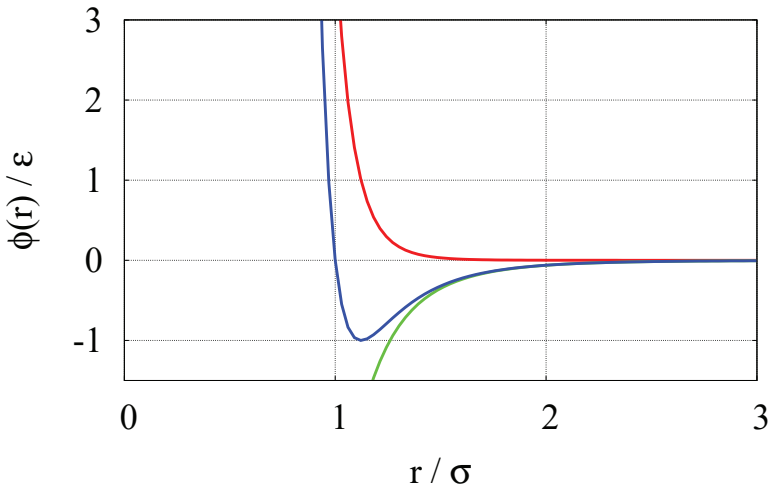


図 1.1 Lennard-Jones(LJ) ポテンシャル：青線が LJ ポテンシャル，赤線は斥力項，緑線は引力項を表す．

す．ここで，粒子間距離 r にのみ依存する古典的 2 体相互作用ポテンシャル (pair potential) を $\phi(r)$ とすれば，全ポテンシャルエネルギーは

$$U_N(\{\boldsymbol{r}_i\}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{i\neq j}^{N}\phi(|\boldsymbol{r}_i - \boldsymbol{r}_j|) \tag{1.1}$$

である．代表例としてクーロン力や分散力などが挙げられる．係数 1/2 は $(i,\ j)$ ペアを 2 重に数える事を防ぐためのものである．ポテンシャルエネルギー U_N から，i 番目粒子に働く力を

$$\begin{aligned}\boldsymbol{F}_i &= -\frac{\partial U_N}{\partial \boldsymbol{r}_i} \\ &= -\frac{1}{2}\sum_{j\neq i}^{N}\frac{\partial\phi(r_{ij})}{\partial \boldsymbol{r}_i} \\ &= \frac{1}{2}\sum_{j\neq i}^{N}\left[\frac{\partial\phi(r_{ij})}{\partial \boldsymbol{r}_i} + \frac{\partial\phi(r_{ji})}{\partial \boldsymbol{r}_i}\right] \\ &= -\sum_{j\neq i}^{N}\frac{\mathrm{d}\phi(r_{ij})}{\mathrm{d}r_{ij}}\frac{\partial r_{ij}}{\partial \boldsymbol{r}_i} \\ &= -\sum_{j\neq i}^{N}\frac{\mathrm{d}\phi(r_{ij})}{\mathrm{d}r_{ij}}\frac{\boldsymbol{r}_i - \boldsymbol{r}_j}{r_{ij}}\end{aligned} \tag{1.2}$$

と与えることができる．ここで，$(\boldsymbol{r}_i - \boldsymbol{r}_j)/r_{ij}$ は，j 粒子から i 粒子に向かう単位ベクトルを表している．

2 体ポテンシャルの代表例として Lennard-Jones(LJ) ポテンシャルを図 1.1 に示す．LJ ポテンシャルは，アルゴンやキセノンなどの希ガスのファンデルワールス力に対するポテンシャルとして用いられ

$$\phi(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{1.3}$$

と与えられる．ϵ はポテンシャル深さであり，σ は粒子直径である．第 1 項は斥力項，第 2 項は引力項であるが，$r < \sigma$ ではポテンシャルの勾配が急峻で強い斥力が働き，r^{-1} で減衰するクーロンポテンシャルに比べて非常に短距離的な相互作用である．粒子間に働く力は，

$$\boldsymbol{F} = -\boldsymbol{\nabla}\phi(r) = -\frac{\mathrm{d}\phi(r)}{\mathrm{d}r}\frac{\boldsymbol{r}}{r} = \frac{24\epsilon}{\sigma}\left[2\left(\frac{\sigma}{r}\right)^{13} - \left(\frac{\sigma}{r}\right)^{7}\right]\frac{\boldsymbol{r}}{r} \tag{1.4}$$

となる．

LJ ポテンシャルなどの短距離相互作用では、一般に粒子間距離が粒子直径の 3 倍程度になるとその相互作用は急激に小さくなる。そこで、シミュレーションの都合上カットオフする半径を設定し、それより遠方にある粒子との間に働く力をゼロにする。これを相互作用のカットオフという。これにより、カットオフ半径内に含まれない遠方の粒子との相互作用を計算しなくても良くなるが、そもそもカットオフ半径内に含まれるか否かを判定するために粒子間距離の計算は必須である。つまり依然として粒子間距離 (あるいは粒子ペア) の計算に $\mathcal{O}(N^2)$ の計算コストを要してしまう。この計算コストを $\mathcal{O}(N^2)$ から $\mathcal{O}(N)$ に低減するアルゴリズムとして，近接粒子の探索手法がいくつか提案されている．本研究においてはこの近接粒子探索手法に焦点を当てている．

1.2 運動方程式の数値積分

相互作用ポテンシャル $\phi(r)$ を LJ ポテンシャルによって与えられた粒子系などで，ニュートンの運動方程式

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \frac{\boldsymbol{F}(t)}{m} \tag{1.5}$$

を数値的に解く方法が MD 法である．実用的な数値計算においては，比較的簡単な数値積分法が使われることが多い．その中のひとつが Verlet の差分式 [1–5] と呼ばれる，式 (1.5) を時間刻み Δt に対して 2 次精度を持つ差分式である．時刻 $t+\Delta t$ と $t-\Delta t$ における粒子の位置 $\boldsymbol{r}(t\pm\Delta t)$ をテイラー展開すると，

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t\boldsymbol{v}(t) + \frac{(\Delta t)^2}{2}\frac{\boldsymbol{F}(t)}{m} + \mathcal{O}((\Delta t)^3) \tag{1.6}$$

$$\boldsymbol{r}(t-\Delta t) = \boldsymbol{r}(t) - \Delta t\boldsymbol{v}(t) + \frac{(\Delta t)^2}{2}\frac{\boldsymbol{F}(t)}{m} + \mathcal{O}((\Delta t)^3) \tag{1.7}$$

を得る．両辺の和と差をとると

$$\boldsymbol{r}(t+\Delta t) + \boldsymbol{r}(t-\Delta t) = 2\boldsymbol{r}(t) + (\Delta t)^2\frac{\boldsymbol{F}(t)}{m} + \mathcal{O}((\Delta t)^4) \tag{1.8}$$

$$\boldsymbol{r}(t+\Delta t) - \boldsymbol{r}(t-\Delta t) = 2\Delta t\boldsymbol{v}(t) + \mathcal{O}((\Delta t)^3) \tag{1.9}$$

これより，時刻 $t+\Delta t$ における位置と t における速度は

$$\boldsymbol{r}(t+\Delta t) = 2\boldsymbol{r}(t) - \boldsymbol{r}(t-\Delta t) + (\Delta t)^2\frac{\boldsymbol{F}(t)}{m} + \mathcal{O}((\Delta t)^4) \tag{1.10}$$

$$\boldsymbol{v}(t) = \frac{1}{2\Delta t}\{\boldsymbol{r}(t+\Delta t) - \boldsymbol{r}(t-\Delta t)\} + \mathcal{O}((\Delta t)^2) \tag{1.11}$$

で与えられる．これが Verlet の差分式である．時刻 $t+\Delta t$ における位置を求めるには，２つの時刻 t と $t-\Delta t$ での位置が必要である．初期条件を位置と速度で与えると，$t=\Delta t$ における位置 $\boldsymbol{r}(\Delta t)$ は式 1.7 から求まる．これと $\boldsymbol{r}(0)$ から $\boldsymbol{r}(2\Delta t)$ を計算し，式 (1.11) より，速度 $\boldsymbol{v}(\Delta t)$ が得られる．この差分方程式は時間反転 ($\Delta t \to -\Delta t$) に対して対称であり，速度の符号を変えると系は描いた軌跡を逆向きに辿って行く．式 (1.11) の代わりに速度を

$$\boldsymbol{v}\left(t+\frac{\Delta t}{2}\right) = \frac{1}{\Delta t}\left[\boldsymbol{r}(t+\Delta t) - \boldsymbol{r}(t)\right] \tag{1.12}$$

で与えると，式 (1.10) は

$$\boldsymbol{v}\left(t+\frac{\Delta t}{2}\right) = \boldsymbol{v}\left(t-\frac{\Delta t}{2}\right) + \Delta t\frac{\boldsymbol{F}(t)}{m} \tag{1.13}$$

と書き換えられる．これと，式 (1.12) を $\boldsymbol{r}(t+\Delta t)$ について解いた

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t\boldsymbol{v}\left(t+\frac{\mathrm{d}t}{2}\right) \tag{1.14}$$

を組み合わせたのが，よく使われる Verlet の蛙跳び (Verlet's leap frog) 法である [1–5]．式 (1.13) で最初に速度を修正し, 次に式 (1.14) で位置の更新を行う．

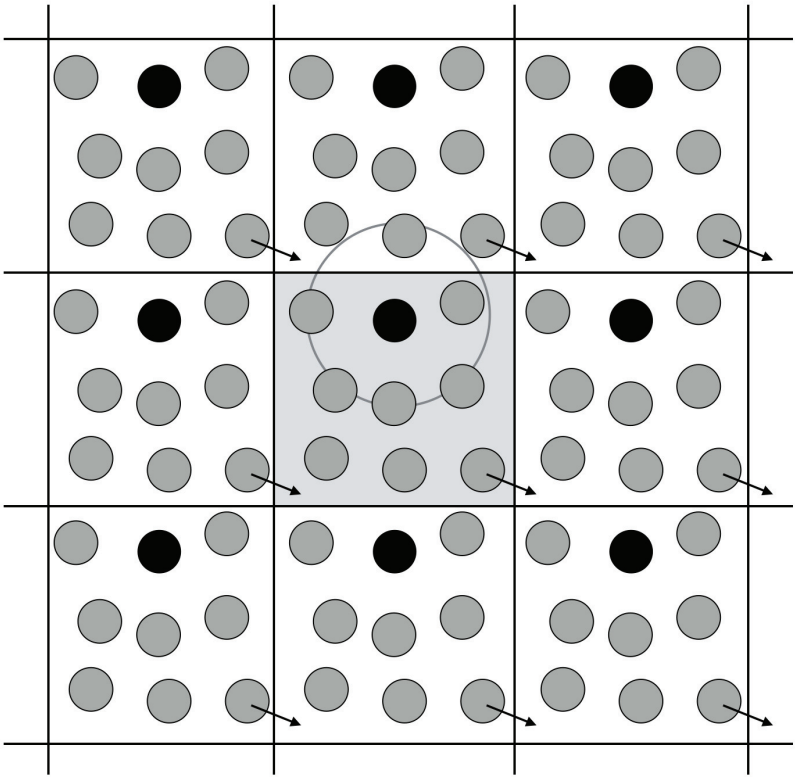


図 1.2　周期境界条件の概念図：中央の灰色のセルが基本セル，周辺のセルがイメージセルを表している．基本セルは一辺 $L = V^{1/3}$ で決まる．V は計算系の全体積である．

1.3　周期境界条件

シミュレーションをおこなう系の物性を調べる際，物質の表面の性質なのか，表面から十分離れた内部 (バルク) の状態なのか，あるいは 2 つの相の共存状態なのかなど，対象とする系のおかれている状況に応じて境界条件を設定しなければならない．

現実系のような 10^{23} 個の粒子を取り扱うことは，現在の最新鋭のコンピュータをもってしてもシリアル CPU で $10^4 \sim 10^6$，並列計算により $\sim 10^9$ 個程度が限界であり，事実上不可能である．$10^4 \sim 10^6$ 個程度の粒子数のシミュレーションでは界面による影響は顕著であり，限られた粒子数でバルクの性質を抽出することがシミュレーションにおいては必須となる．そこで，バルクのシミュレーションに用いる周期境界条件 (Periodic Boundary Condition) について説明する．図 1.2 のように，計算システムを x, y, z 方向にそれぞれに繰り返し並べ，以下の条件を考慮することが周期境界条件である．中央のセルが一辺が $L = V^{1/3}$ の基本セル，周辺のセルをイメージセルと呼ぶ．ここで V は全体積である．周期境界条件では以下の点が考慮されている．

- 有限個の粒子が有限サイズの計算セルに閉じ込められていると考える．計算セルの形状は，周期的に並べる事が出来るよう，立方体，直方体，または，平行六面体とする．
- ある粒子が時間発展により運動し，位置を変え，右側のイメージセルに移ったとすれば (図 1.2 の矢印)，左隣のイメージセルから同一の粒子が補充される．
- 粒子の相互作用に関しても周期境界を考えなければならない．イメージセルを導入したとしても，実質的に同じペアの相互作用を重ねて数えることがないように，実粒子・shadow 粒子を含めて最

も近いペアの相互作用のみを考慮する必要がある．これは相互作用を計算セルサイズの半分のところでカットする事に相当し，minimum image convention と呼ばれている．

以上３つの条件を考慮し，周期境界条件のもとに計算を実行する．これにより，少ない粒子数のシミュレーションでもバルクの性質を抽出することが可能となる．他の境界条件としては，反射境界，自由境界などがある．

1.4 古典統計力学の基礎理論

MD シミュレーションから，各時刻における粒子の位置 $\boldsymbol{r}_i(t)$ および速度 $\boldsymbol{v}_i(t)$ が得られる．以下では古典統計力学に基づいて粒子の位置・速度から得られる代表的な熱力学量の計算方法に関して概説する [6–9].

(a) **エネルギー**

運動方程式を数値的に解くと，全エネルギー

$$E = \left\langle \frac{1}{2}\sum_{i=1}^{N}\sum_{i\neq j}^{N}\phi(|\boldsymbol{r}_i - \boldsymbol{r}_j|)\right\rangle + \left\langle \sum_{i=1}^{N}\frac{1}{2}m\boldsymbol{v}_i^2\right\rangle \tag{1.15}$$

は保存される．つまり，小正準集合 (NVE アンサンブル) を計算していることになる．ここで $\langle\cdots\rangle$ はアンサンブル平均または時間平均を表している．

(b) **温度**

ある時刻における瞬間的な温度 $T(t)$ は，運動エネルギーから

$$T(t) = \frac{1}{3Nk_B}\sum_{i=1}^{N} m\boldsymbol{v}_i^2 \tag{1.16}$$

と決める事ができる．つまり，エネルギー等分配則の結果である．さらに充分長い時間をかけて計算し系が熱平衡状態に到達したとき系の温度は瞬間的な温度 $T(t)$ の時間平均から

$$T = \langle T(t)\rangle_t = \lim_{\tau\to\infty}\frac{1}{\tau}\int_0^{\tau} T(t)\mathrm{d}t \tag{1.17}$$

のように求められる．ここで $\langle\cdots\rangle_t$ は特に時間に対する平均値をとることを意味している．

(c) **圧力**

圧力は Virial の定理に基づいて計算される．まず，i 番目粒子が容器の壁から受ける力を $\boldsymbol{W}_i$，他の粒子からの力の総和を $\boldsymbol{F}_i$ とすれば

$$m\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2} = \boldsymbol{F}_i + \boldsymbol{W}_i \tag{1.18}$$

の運動方程式が成り立つ．この両辺の各項と i 番目粒子の位置ベクトル $\boldsymbol{r}_i$ との内積をとり，時間積分をとる．$\boldsymbol{r}_i\cdot\boldsymbol{F}_i$ の項は

$$\int_0^{\tau}\boldsymbol{r}_i\cdot\frac{\mathrm{d}^2\boldsymbol{r}_i}{\mathrm{d}t^2}\mathrm{d}t = \left[\boldsymbol{r}_i\cdot\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}\right]_0^{\tau} - \int_0^{\tau}\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}\cdot\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}\mathrm{d}t \tag{1.19}$$

のように部分積分を行うと，右辺第 1 項の $\boldsymbol{r}_i\cdot\boldsymbol{v}_i$ は任意の時刻で有限の値を持つ．したがって τ で割って $\tau\to\infty$ とすれば消える．全ての粒子に対して和をとれば，

$$\left\langle\sum_{i=1}^{N}\frac{1}{2}m\left(\frac{\mathrm{d}\boldsymbol{r}_i}{\mathrm{d}t}\right)^2\right\rangle_t = -\frac{1}{2}\left\langle\sum_{i=1}^{N}\boldsymbol{r}_i\cdot(\boldsymbol{F}_i+\boldsymbol{W}_i)\right\rangle_t \tag{1.20}$$

となる．ここで左辺は運動エネルギーの長時間平均である．一方で，右辺の量は Clausiu によって名付けられた Virial である．系の運動エネルギーの長時間平均が Virial に等しいことを示すこの関係式は Virial の定理と呼ばれる．さらに，エネルギー等分配則から

$$\left\langle \sum_{i=1}^{N} \boldsymbol{r}_i \cdot \boldsymbol{W}_i \right\rangle_t = -3Nk_BT - \left\langle \sum_{i=1}^{N} \boldsymbol{r}_i \cdot \boldsymbol{F}_i \right\rangle_t \tag{1.21}$$

とすることができ，左辺は壁から受ける力の Virial である．これは圧力 P と関係づけられる．つまり，i 番目粒子が容器の壁に及ぼす力は $-\boldsymbol{W}_i$ であるが，すべての粒子が壁面の単位面積あたりに及ぼす力の平均が圧力である．壁に位置ベクトル $\boldsymbol{r}$ の面積要素 $\mathrm{d}S$ をとり，そこから外向き単位法線ベクトル $\boldsymbol{n}$ とすれば，単位面積 $\mathrm{d}S$ が受ける平均の力は $P\boldsymbol{n}\mathrm{d}S$ である．したがって Gauss の定理を用いれば

$$\left\langle \sum_{i=1}^{N} \boldsymbol{r}_i \cdot \boldsymbol{W}_i \right\rangle_t = \int_S P\boldsymbol{n} \cdot \boldsymbol{r}\mathrm{d}S = -3PV \tag{1.22}$$

である．よって圧力は

$$P = \frac{Nk_BT}{V} + \frac{1}{3V}\left\langle \sum_{i=1}^{N} \boldsymbol{r}_i \cdot \boldsymbol{F}_i \right\rangle_t \tag{1.23}$$

$$= \frac{Nk_BT}{V} - \frac{1}{3V}\left\langle \sum_{i=1}^{N} \boldsymbol{r}_i \cdot \nabla_i U_N(\{\boldsymbol{r}_i\}) \right\rangle_t \tag{1.24}$$

として計算する事ができる．

(d) **動径分布関数**

1 つの粒子を中心とした他の粒子の位置の統計的分布を表す関数として動径分布関数がある．原点 $(r = 0)$ に粒子があるとき半径 $(r, r + dr)$ の球殻内にある粒子の平均個数を $\mathrm{d}n(r)$ とおくと，平均個数と動径分布関数とは

$$\mathrm{d}n(r) = \rho g(r) 4\pi r^2 \mathrm{d}r, \quad \rho = \frac{N}{V} \tag{1.25}$$

の関係で結ばれている．ここで ρ は数密度である．この動径分布関数は液体構造の静的性質を調べる指標となる．$g(r)$ は $r \to \infty$ で $g(r) = 1$ となるように規格化され，また排除体積により $r \to 0$ で $g(r) = 0$ となる．また，この動径分布関数を用いて系のエネルギー，圧力を評価することが可能である．内部エネルギーを

$$E = E^{\mathrm{id}} + E^{\mathrm{ex}} \tag{1.26}$$

のように理想項と過剰項に分ける．ここで理想項は

$$E^{\mathrm{id}} = \frac{3}{2}Nk_BT \tag{1.27}$$

過剰項は

$$E^{\mathrm{ex}} = \langle U_N(\{\boldsymbol{r}_i\})\rangle = \frac{1}{Z_N}\int U_N(\{\boldsymbol{r}_i\}) \exp(-U_N(\{\boldsymbol{r}_i\})/k_BT)\mathrm{d}\boldsymbol{r}^N \tag{1.28}$$

である．また

$$Z_N = \int \exp(-U_N(\{\boldsymbol{r}_i\})/k_BT)\mathrm{d}\boldsymbol{r}^N \tag{1.29}$$

は配置積分と呼ばれる．内部エネルギーの過剰項は動径分布関数 $g(r)$ を用いて

$$\frac{E^{\mathrm{ex}}}{N} = 2\pi\rho\int_0^\infty \phi(r)g(r)r^2\mathrm{d}r \tag{1.30}$$

としてあらわさる．圧力は式 (1.24) より，式 (1.28) と同様の考察から

$$\frac{P}{\rho k_B T} = 1 - \frac{2\pi\rho}{3k_B T}\int_0^\infty \phi'(r)g(r)r^3 \mathrm{d}r \tag{1.31}$$

とあらわすことができる．

1.5 本論文の目的と構成

分子シミュレーションにおいて最も時間がかかるのは，粒子間相互作用 (ポテンシャルエネルギーや力の計算) の部分である．LJ ポテンシャルのような 2 体相互作用ですら，粒子間距離の計算に $\mathcal{O}(N^2)$ の計算コストを要してしまう．これは位置の更新，速度の更新などの $\mathcal{O}(N)$ の計算コストに対し顕著である．大きなシステムサイズの系に対して，純粋に $\mathcal{O}(N^2)$ の計算を行うことは現代のコンピュータパフォーマンスを持ってしても現実的でない．本論文の目的は，計算効率を向上するための近接粒子探索リストアルゴリズムを提案すると共に，それぞれの手法を最適化する方法を構築する事である．

本論文では 2 つの粒子探索手法を組み合わせた VNL-CLL 法を提案している．その要素はリストの動的更新の手法を用いたベルレの近接リスト法 (Verlet Neighbor List: VNL 法) とセル分割法 (Cell Linked List: CLL 法) からなる．さらにこの CLL 法にはセルサイズを拡張した排他的セル分割法 (Exclusive CLL 法) を採用した．VNL 法・CLL 法にはそれぞれ利点・欠点がある．VNL 法は基本的に計算コストを $\mathcal{O}(N)$ に低減することができる利点があるが，数ステップに 1 度 $\mathcal{O}(N^2)$ の計算コストを要する更新を伴う欠点がある．CLL 法はリスト更新の必要がなく常に $\mathcal{O}(N)$ の計算コストに低減することができる一方で，VNL 法と比較しリストに保存しておく粒子数が多い事が欠点である．本研究において新規に開発した VNL-CLL 法は VNL 法・CLL 法両手法の利点を組み合わせることによって，それぞれの欠点を克服している大きな特徴がある．

本論文の構成は以下の通りである．第 2 章では VNL 法とそれをさらに拡張したリストの動的更新の手法を紹介する．第 3 章では VNL 法と同様に良く用いられる手法である CLL 法と，より高度な排他的セル分割法に対し，アルゴリズムと最適化の方法を紹介する．また，対応するパラメータ依存性の検討，計算性能の比較を行った．第 4 章では VNL 法，CLL 法の両者の利点を組み合わせた VNL-CLL 法を提案し，その最適化手法のための計算時間に対する評価式を解析的に与える．第 5 章では結論を述べる．また，最適化された各種法 (VNL 法，リスト動的更新型 VNL 法，CLL 法，Exclusive CLL 法，VNL-CLL 法) に関して計算パフォーマンスの比較を行った．

なお，本研究では CPU1 コア計算を実施し，各近接粒子探索手法の効率性について検討している．具体的には LJ ポテンシャル，周期境界条件を用いて NVE アンサンブルのもとで計算した．各手法ともに 100,000 ステップのレコードをとり計算時間を評価した．自然科学研究機構計算科学研究センターの計算機を使用した．詳細は以下の通りである．

Fujitsu PRIMERGY RX300 S7
CPU: Intel Xeon E5-2690
Memory: 8GB
Compiler: インテル (R) 64 対応インテル (R) C コンパイラー XE (インテル (R) 64 対応アプリケーション用)、バージョン 15.0.2.164 ビルド 20150121 (C) 1985-2015 Intel Corporation.

また，本論文の結果は特に断らない限り，長さ σ，質量 m，時間 $\tau = \sqrt{m\sigma^2/k_B T}$，エネルギー ϵ，温度

ϵ/k_B として無次元化してあらわす．時間刻み Δt は，0.005τ として計算を実行した．

第 2 章

ベルレの近接リスト法

2.1 概要

カットオフ半径 $r_{\rm cut}$ により，それぞれの粒子は半径 $r \leq r_{\rm cut}$ に存在する粒子としか相互作用をしない．通常の固体や液体であれば，その数は

$$4\pi\rho \int_0^{r_{cut}} {\rm d}r r^2 g(r) \sim \frac{4}{3}\pi\rho r_{\rm cut}^3 \tag{2.1}$$

で評価される．ここで密度 $\rho = 1.0$ の系を考える．LJ ポテンシャルで典型的に用いるカットオフ半径 $r_{\rm cut} = 2.5$ とすれば相互作用する粒子数は高々 200 個程度であり，系全体の総粒子数 (1000 粒子以上と考える) と比べると極めて少ない．そこで，各粒子についてあらかじめ相互作用をする粒子ペア，あるいは粒子リストを設けておけば計算コストを抑えることができると期待される．実際には図 2.1 で表されるように，カットオフ半径の少し外側に SKIN $= r_{\rm list} - r_{\rm cut}$ を設けておき，その中に含まれる粒子のインデックスを配列に記録しておく．この手法をベルレの近接リスト法 (Verlet Neighbor List: VNL 法) と呼ぶ [10,11]．もちろんそのリストの作成の計算コストは $\mathcal{O}(N^2)$ であるが，その更新頻度を 10 ステップに 1 回，あるいは 100 ステップに 1 回などのように少なくできれば良い．実際には時間発展により粒子

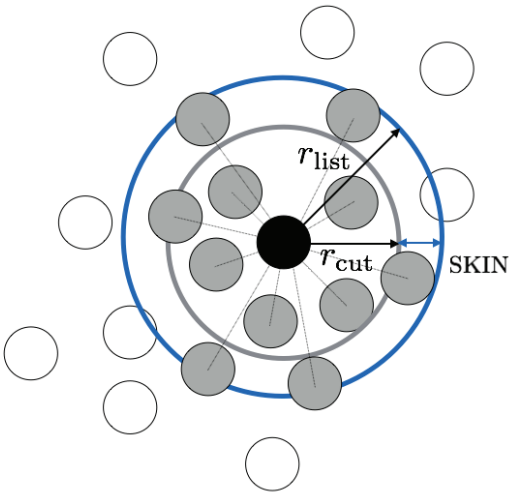


図 2.1　ベルレの近接リスト法 (VNL) の概念図：SKIN $= r_{\rm list} - r_{\rm cut}$，$r_{\rm cut}$ はカットオフ半径，$r_{\rm list}$ はリスト半径を表す．灰色粒子は黒色粒子の近接リストに記憶されている粒子であり，毎ステップリストを用いて相互作用を計算することが出来る．白色粒子はリストの外にでており，計算の必要の無い粒子である．

は次第に移動していくので，ある程度計算が進んだ後に，そのリストを更新する必要がある．つまり粒子がリストの範囲から出てしまう前にリスト更新を行わなければならない．リストの更新を除けば，相互作用の計算コストは $\mathcal{O}(N^2)$ から $\mathcal{O}(N)$ に低減することが可能である．固体の様に粒子がほとんど移動しないような系に対してはリスト更新の頻度を少なくする事が可能であり VNL 法は非常に効果的である．逆に言えば高温で粒子が激しく運動しているような液体系に対しては頻繁にリストを更新しなければならず，あまり効果的ではない．

VNL 法における近接リストの作成に関する該当部分を **Algorithm1** に示す．まず始めにリストを構成する時点の粒子の配置を保存する．これは後にリストの更新のタイミングを決定するために必要となる．具体的には，毎ステップ，各粒子の変位を計算しそれが SKIN/2 を超えてしまうとリストを更新するということを繰り返す (詳細に関しては後に説明する)．次に，カットオフ半径 $r_{\rm cut}$ よりも少し大きい半径 $r_{\rm list} = r_{\rm cut} + {\rm SKIN}$ を設けておき，その中に含まれる粒子をリストに保存していく．point[i] は i 番目の粒子と相互作用する j 粒子のインデックスがリスト配列の何番目から読み取ればよいかを示している．したがって，i 番目の粒子の VNL に属している相手方 j 粒子は list[i] の i = point[i] から point[i+1] - 1 までに収められていることになる．そして，リストを使用して力を計算する際には **Algorithm2** の様にすれば良い．最後に相手方 j 粒子のインデックスを配列 list[ ] から参照し相互作用を計算する．

リストの更新間隔は 10 ステップなどに固定しても，次の更新までに粒子がリストから飛び出した場合には相互作用する粒子の誤参照が起こってしまい計算が破綻する．そこで毎ステップ，リストを構築した時点からの粒子の変位を計算し，それがある閾値を超えてしまったらリストを更新すれば，これにより相互作用する粒子の誤参照を防ぐ事ができ，かつその更新間隔を最大限に取る事が可能となる．つまり，リストの更新間隔を事前に固定する必要がなくなる．この行程をリストの動的更新と呼ぶ [12–14]．リスト動的更新の構造は **Algorithm3** に示した．

ここで，閾値に関して考察する．図 2.2(a) に記すように黒い 2 つの粒子の運動に注目する．これら 2 つの粒子は互いの近接リストに含まれ合っている．ここで，これら 2 つの粒子が互いに対角方向に速度 v

Algorithm 1 Make the Verlet Neighbor List

```
 1: =====Save current configuration.=====
 2: for i=0 to NumParticle do
 3:     rx0[i] ← rx[i]
 4:     ry0[i] ← ry[i]
 5:     rz0[i] ← rz[i]
 6: end for
 7: =====Make the Verlet Neighbor List.=====
 8: (int)nlist ← 0
 9: for i=0 to NumParticle - 1 do
10:     point[i] ← nlist + 1
11:     for j=i+1 to NumParticle do
12:         calculate r_ij
13:         if r_ij < r_list then
14:             nlist ← nlist + 1
15:             list[nlist] ← j
16:             if r_ij < r_cut then
17:                 calculate force
18:             end if
19:         end if
20:     end for
21: end for
22: point[N-1] ← nlist + 1
```

Algorithm 2 Use the Verlet Neighbor List

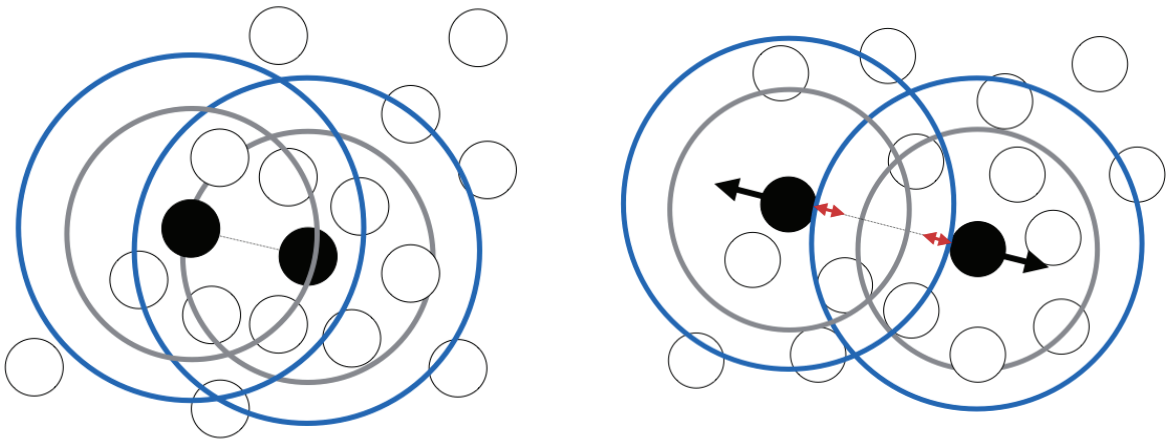
```
1: for i=0 to NumParticle - 1 do
2:    j_begin ← point[i]
3:    j_end ← point[i+1] - 1
4:    nlist ← point[i]
5:    if j_begin <= j_end then
6:       for j_nab = j_begin to j_end do
7:          j ← list[j_nab]
8:          calculate r_ij
9:          if r_ij < r_cut then
10:            calculate force
11:         end if
12:      end for
13:   end if
14: end for
```

で等速直線運動したと仮定すると，n ステップ後の位置は右図の様になる．図 2.2(b) では各粒子は互いの近接リストの範囲から出てしまっている．この時各粒子がそれぞれ移動した距離は

$$v \times n\mathrm{dt} = \mathrm{SKIN}/2 \tag{2.2}$$

と計算する事が出来る．つまり，粒子変位が SKIN/2 になったときリストを更新すれば良い．この仮定は実際の多粒子系の MD 計算においてはほぼ起こりえない．なぜなら粒子は周囲の粒子と互いに衝突し合い，対角方向へ等速直線運動など起こりえないからである．したがって，粒子の変位が閾値 SKIN/2 を超えてしまったらリストを更新するという設定が妥当である事を意味している．

以上の事から，VNL 法の構造は **Algorithm4** のようになる．毎ステップ，リストを更新するか否かチェックし，更新するならばリストを再構築すると共に力を計算する．更新しないならリストを用いて力を計算する事ができる．



(a) $t = t_0$ における粒子配置.　(b) $t = t_0 + ndt$ における粒子配置.

図 2.2　Verlet Neighbor List criterion：(a)$t = t_0$ において，黒色粒子はそれぞれ近接リストに粒子を記録してあり，互いにリストに含まれ合っている．(b)n ステップ経過した後，黒色粒子は互いの近接リストから飛び出す．

Algorithm 3 Check update of the Verlet Neighbor List

```
dispmx ← 0
for i=0 to NumParticle do
    dispmx ← max ( | rx[i] - rx0[i] | , dispmx)
    dispmx ← max ( | ry[i] - ry0[i] | , dispmx)
    dispmx ← max ( | rz[i] - rz0[i] | , dispmx)
    dispmx ← √3.0 dispmx
end for
if  dispmx > SKIN / 2 then
    Make the Verlet Neighbor List
end if
```

Algorithm 4 the Verlet Neighbor List

```
==== MD loop start =====
for step = 0 to nstep do
    move r(t) → r(t+dt) and v(t) → v(t+dt/2)
    Check update of the VNL
    if Need update of the VNL then
        Make the VNL and calculate force O(N^2)
    else
        Use the VNL and calculate force O(N)
    end if
    move v(t+dt/2) → v(t+dt)
end for
```

2.2 SKIN(リスト半径) の最適化

式 (2.2) より，リスト更新の条件は

$$\max_i |\boldsymbol{r}(t_0 + n\mathrm{d}t) - \boldsymbol{r}(t_0)| > \frac{\mathrm{SKIN}}{2} \tag{2.3}$$

と書き換えられる．ここで n は更新間隔，$\mathrm{d}t$ はシミュレーションの時間刻みを示している．$t = t_0$ の粒子位置からの変位を全粒子に関して毎ステップ計算しておき，そのうちのどれか 1 粒子でも変位が $SKIN/2$ を越えてしまうとリストを更新することになる．前述のように，更新間隔 n は粒子の運動に従い，つまり動的に変化する．よって，更新間隔 n は温度 T，密度 ρ，粒子数 N，カットオフ半径 r_{cut} に依存することになる．

いま $n+1$ ステップの計算を実行することを考える．このとき，n ステップの間はリストを使用 ($\mathcal{O}(N)$) し，$n+1$ ステップ目にリストを更新する為に全粒子探索 ($\mathcal{O}(N^2)$) をするので，その間，相互作用の計算に要する時間は解析的に

$$T_{\mathrm{VNL}} = \frac{1}{2} N \tau_f \left[\frac{N-1}{n+1} + \frac{n}{n+1} \left(4\pi\rho \int_0^{r_{\mathrm{list}}} dr r^2 g(r) - 1 \right) \right] \tag{2.4}$$

と，あらわすことができる．ここで τ_f は各粒子ペアの計算に要する単位時間量である．第 1 項はリスト更新に要する時間，第 2 項はリスト使用に要する時間を表している．

VNL 法に対してパラメータ (リスト半径) を最適化するために，リスト半径 r_{list} をシステムを設定しているパラメータ $(N, T, \rho, r_{\mathrm{cut}})$ から見積もる近似を考える．式 (2.3) から，SKIN は更新間隔 n と粒子変位から見積もる事ができるので，

$$\mathrm{SKIN} = 2n\langle x \rangle \tag{2.5}$$

とすればよい．ここで，$\langle x \rangle$ は最も長い距離移動し $SKIN/2$ を超える粒子の最大変位をあらわしている．ここで $\langle x \rangle$ をどのように評価するかが重要となる．まず始めに考えられることは，拡散係数 D から長さスケール l_D を見積もることである．つまり，$\langle x \rangle \sim l_D$ とし，ここで $l_D = \sqrt{D\delta t}$ という近似である．しかしながらこの l_D は，拡散過程から特徴付けられる長さスケールであり，リスト半径 $r_{\rm list} = 2.5 \sim 5.0$ に比べると非常に長く不適切である．そこで，短い長さスケールを特徴付けるための平均自由行程 $l_{\rm th}$ を考える．系の温度 T から，エネルギー等分配則 $\frac{1}{2}mv^2 = \frac{3}{2}k_BT$(3 次元) を用いて平均速度 $\langle v \rangle$ を評価する事で見積もる事ができる．これにより，より適切な近似 $\langle x \rangle \sim l_{\rm th}$ が可能となり，$l_{\rm th} = \langle v \rangle \delta t$ とすればよい．しかしながらまだ十分ではない．平均速度 $\langle v \rangle$ から粒子の “平均変位” を求めるのではなく，全粒子の中から最も長い距離移動した粒子の “最大変位” を求めなければならない．そこで粒子の “最大変位” を

$$\langle x \rangle \sim c\delta t\sqrt{\frac{3k_BT}{m}} \tag{2.6}$$

とし，定数 c は Maxwell 分布の中で最も速い粒子を抽出するため $c = 3.0$ とする．図 2.3 は粒子数 1,000, 10,000, 100,000 の 3 つのシステムに対する速度分布の結果を示している．

次により簡単化のため，粒子の分布を平均密度で近似する．つまり，$g(r) = 1$ とし，

$$\int_0^{r_{\rm list}} dr r^2 g(r) \sim \frac{r_{\rm list}^3}{3} \tag{2.7}$$

となる近似である．これにより，1 粒子が $n+1$ ステップに要する計算時間は

$$\begin{aligned} f_{\rm VNL} = \frac{T_{\rm VNL}}{2\tau_f N(N-1)} &= \frac{1}{n+1} + \frac{n}{n+1}\frac{1}{N-1}\left[\frac{4\pi}{3}\rho r_{\rm list}^3 - 1\right] \\ &= \frac{1}{n+1} + \frac{n}{n+1}\frac{1}{N-1}\left[\frac{4\pi}{3}\rho(r_{\rm cut} + 2n\langle x \rangle)^3 - 1\right] \end{aligned} \tag{2.8}$$

で評価する事が可能となる [15]．右辺第 1 項は $n+1$ ステップ目に一度リストを更新する事を反映しており，第 2 項は n ステップの間，半径 $r_{\rm list}$ の球形の VNL を使用する事に対応している．リストの動的更新の手法を採用することで，式 (2.2) から明らかなように更新間隔 n はリスト半径 $r_{\rm list}$ あるいは SKIN に

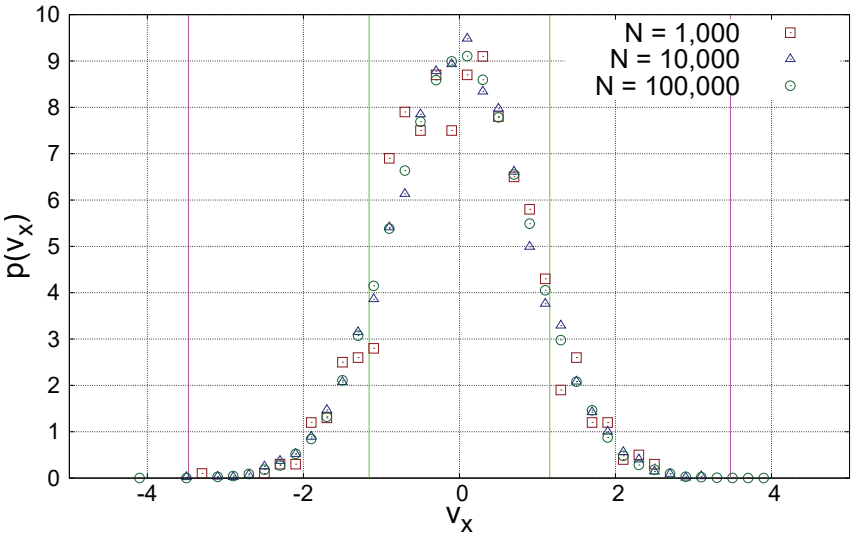
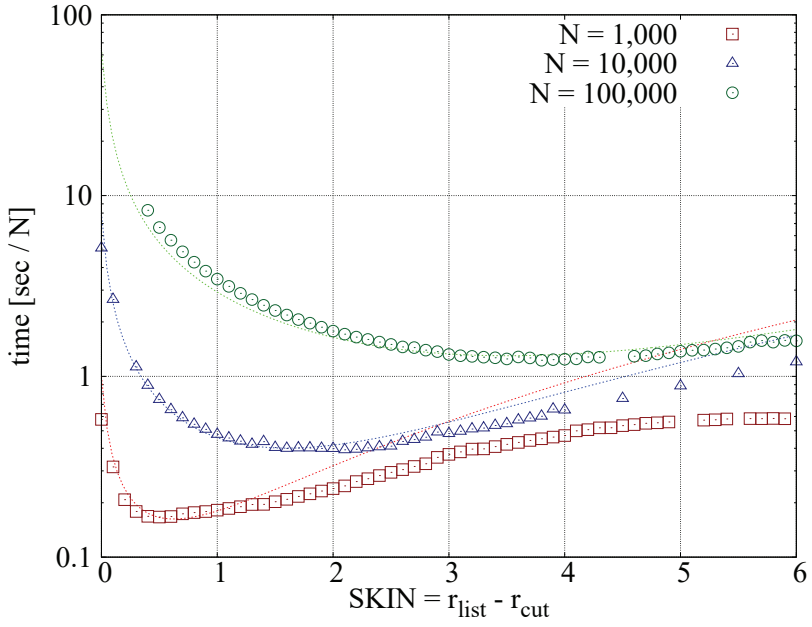


図 2.3　粒子速度の x 成分の分布：$T = 0.772$，$\rho = 0.8$ として $N = 1,000, 10,000, 100,000$ の 10 ステップ間の平均速度分布の結果をあらわしている．緑線はエネルギー等分配則から見積もった速度 $\sqrt{3k_BT/m}$，紫線は $3\sqrt{3k_BT/m}$ をあらわす．

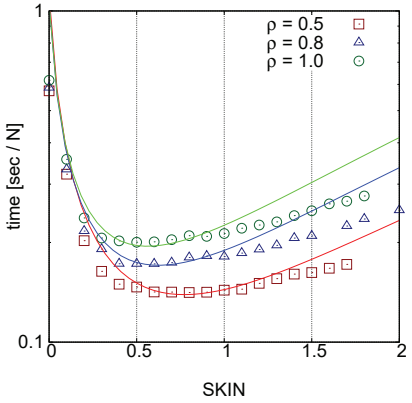
依存する．SKIN が大きければ，それだけ更新間隔を長く取る事ができるが，一方で，そもそも SKIN が大きい事でその中に含まれる粒子数

$$4\pi\rho\int_0^{r_{\rm list}} {\rm d}r r^2 g(r) \sim \frac{4}{3}\pi\rho r_{\rm list}^3 \tag{2.9}$$

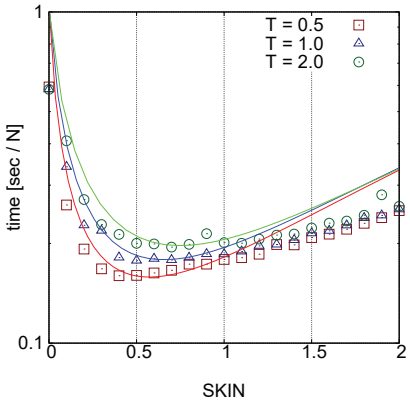
も増えてしまう．したがって，$\mathcal{O}(N^2)$ のリスト更新はほとんどしなくてもよくなるが，リストを使用して毎ステップ相互作用を計算する際に参照する粒子数が多くなるので，$\mathcal{O}(N)$ (正確には $\mathcal{O}(N\times$ リストに含まれている隣接粒子)) のリスト使用に計算時間がかかってしまう．逆に SKIN が小さければ頻繁にリスト更新を行ってしまい，$\mathcal{O}(N^2)$ の計算コストが大半を占めてしまう．極端な例である SKIN $= 0$



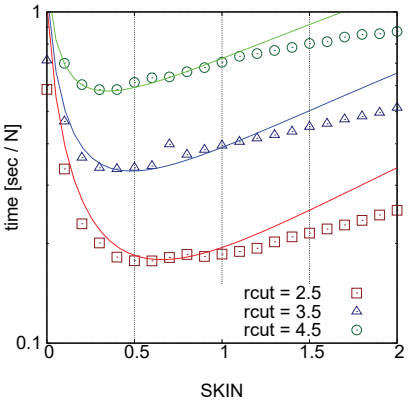
(a) システムサイズ (粒子数) 依存性



(b) 密度依存性



(c) 温度依存性



(d) カットオフ半径依存性

図 2.4　計算時間の SKIN サイズ依存性：点はシミュレーションの結果を，実線は式 (2.8) をそれぞれ表している．(a) システムサイズ依存性，(b) 密度依存性，(c) 温度依存性，(d) カットオフ半径依存性を検証した．計算時間を最小化する最適リスト半径 (SKIN サイズ) はシステムサイズ，密度，温度，カットオフ半径に依存する．SKIN サイズを小さくとるとリスト更新 ($\mathcal{O}(N^2)$) の頻度が増え，計算コストを要する．SKIN サイズを大きくすると参照体積が増え，隣接粒子数のループに計算コストを要する．

の時は毎ステップ毎にリストを更新する事になる．また，SKIN $\to L$ の極限ではシステム全体の粒子がリストに含まれることになり，リストを使用したとしても $\mathcal{O}(N^2)$ の計算コストを要してしまう．このように，SKIN と更新頻度は互いにトレードオフの関係にあり，計算時間を最適とする SKIN の特定が必要とされる．

式 (2.8) の結果とシミュレーションによる結果を図 2.4(a)∼(d) に示す．シミュレーションは LJ ポテンシャルで NVE アンサンブルを 100,000 ステップ行った．縦軸は 1 粒子あたりに要する計算時間，横軸は SKIN である．各パラメータに対する依存性は以下のようにまとめられる．

(a) **粒子数依存性**

温度 $T = 0.772$，密度 $\rho = 0.8$，カットオフ半径 $r_{\rm cut} = 2.5$ とし，システムサイズ (粒子数)$N = 1,000, 10,000, 100,000$ の 3 つの系に対しシミュレーションを行った．シミュレーションの結果，計算時間を最適化する SKIN サイズはそれぞれのシステムサイズに対して，SKIN $= 0.5\sigma(N = 1,000), 1.1\sigma(N = 10,000), 6.3\sigma(N = 100,000)$ である．$n+1$ ステップに一度行う更新のステップでは VNL 法は $\mathcal{O}(N^2)$ の計算コストを伴う．更新までは $\mathcal{O}(N)$ を維持することができるため，粒子数が増えるとその効果は顕著になる．従って，粒子数が増えるに伴い，なるべくリスト更新をしない方が (更新間隔 n が大きい方が) 計算コストを落とすことができると考えられる．つまり，粒子数の増加に対して更新間隔 n を延ばすために，最適 SKIN は大きくなる．

(b) **密度依存性**

粒子数 $N = 1,000$，温度 $T = 0.772$，カットオフ半径 $r_{\rm cut} = 2.5$ とし，密度 $\rho = 0.5, 0.8, 1.0$ の 3 つの系に対しシミュレーションを行った．計算時間を最適とする SKIN サイズはそれぞれ，SKIN $= 0.8\sigma(\rho = 0.5), 0.6\sigma(\rho = 0.8), 0.5\sigma(\rho = 1.0)$ である．密度はリストに含まれる粒子数に関係する．当然，低密度であればリストに記憶する相互作用ペアの数は小さくなり，計算コストも下がる．また，低密度になるにつれ粒子が移動する平均自由行程が長くなる．これにより，その分だけ粒子はリストの範囲から出やすくなり，頻繁に $\mathcal{O}(N^2)$ の更新を行ってしまう．したがって，更新間隔を長くした方が計算効率は向上する．そのため，低密度なほど最適 SKIN は大きい．

(c) **温度依存性**

粒子数 $N = 1,000$，密度 $\rho = 0.8$，カットオフ半径 $r_{\rm cut} = 2.5$ とし，温度 $T = 0.500, 1.000, 2.000$ の 3 つの系に対しシミュレーションを行った．計算時間を最適とする SKIN サイズはそれぞれ，SKIN $= 0.4\sigma(T = 0.5), 0.5\sigma(T = 1.0), 0.7\sigma(T = 2.0)$ である．温度は粒子の速度に依存する．システムの温度が高いほど粒子の速度は増加し，それに伴い粒子がリストの範囲から出る頻度も増加する．そのため温度が高い系では，なるべく更新間隔 n を大きくし，リストをより長く使い続けた方が計算効率が良くなるため，SKIN を大きく取った方が良い．

(d) **カットオフ半径依存性**

粒子数 $N = 1,000$，温度 $T = 1.000$，密度 $\rho = 0.8$ とし，カットオフ半径 $r_{\rm cut} = 2.5, 3.5, 4.5$ の 3 つの系に対しシミュレーションを行った．計算時間を最適とする SKIN サイズはそれぞれ，SKIN $= 0.6\sigma(r_{\rm cut} = 2.5), 0.4\sigma(r_{\rm cut} = 3.5), 0.3\sigma(r_{\rm cut} = 4.5)$ である．カットオフ半径が大きいほど，カットオフ球体積内に含まれる相互作用を計算するペアの数は多くなる．つまり，リストに含まれる粒子ペアも多くなる ($\mathcal{O}(N \times$ リストに含まれている隣接粒子数))．極端な例をあげれば，カットオフ半径がシステムの一辺の長さと等しい場合，リスト使用による計算コストが $\mathcal{O}(N^2)$ となってしまう．つまり，カットオフ半径が大きくなるにつれ，リスト更新による計算コストの寄与は相対的には小さくなる．逆に，カットオフ半径が小さければシステム全体に比べてリスト体積は

非常に小さいので，なるべくリスト更新をしないほうが効率が良い．そのため，カットオフ半径が小さいほど最適 SKIN は大きい．

図 2.4(a)∼(d) において，SKIN が大きくなるにつれ，式 (2.8) の結果とシミュレーションの結果に差が生じていることがわかる．これは式 (2.6) において $c = 3.0$ と仮定したことに起因する．この $c = 3.0$ は系に含まれている粒子の最大変位を見積もるための指標として採用した値であった．粒子数が増加するに伴い，あるステップにおける Maxwell 分布において $3\sqrt{3k_BT/m}$ を上回る速度を持つ粒子が存在する確率は，粒子数が多い方が大きい (例えば 100,000 粒子のうち $3\sqrt{3k_BT/m}$ を上回る速度をもつ粒子を見出す確率が 1/100,000 であるとすると，100,000 粒子系では毎ステップ 10,000 粒子系では 10 ステップに 1 度，そのような速度を持つ粒子を観測することになる)．c が小さければ最大速度をもつ粒子が実際よりも高い確率で観測されると仮定する事になる．つまり，実際よりも更新頻度が上がると仮定してしまうことになり，したがって，SKIN が大きい領域で評価式はシミュレーション結果より大きな値をとる．しかしながら，式 (2.8) において最も興味があるのは，計算時間を最小化する最適な SKIN の特定であるから，$c = 3.0$ としても $N \in [1000, 100, 000]$ のシステムサイズに対しては十分に最適値を特定する事ができる．なお，VNL 法においてはリスト更新の際の全粒子探索の計算コスト $\mathcal{O}(N^2)$ がボトルネックとなるため，巨大なシステムサイズ ($N > 100,000$) に対しては用いられることはない．

第 3 章

セル分割法

3.1 概要

高速化のためのもうひとつの有益な方法として，計算系を一辺が $L_{\rm cell}$ の小さな立方体セルに分割する事を考える．長さ $L_{\rm cell}$ を $r_{\rm cut} \leq L_{\rm cell}$ を満たすように選ぶことで，粒子間の相互作用は同じセルに属する粒子間，もしくは隣接セルに属する粒子間に限定される．図 3.1 に示すように，隣接するセルの数は限られている (2 次元なら 8 つ，3 次元なら 26 個) ことから，大きな系であれば相互作用の数を大幅に減らすことができる．この方法をセル分割法 (Cell Linked List: CLL 法) と呼ぶ [10, 11]．それぞれのセルに属する粒子の数は有限 (高々 100 粒子程度) であるから，相互作用に要する計算時間は $\mathcal{O}(N)$ に抑える事ができる．分割するシステムが直方体であれば一般に分割セルも直方体となるが，それぞれのセルの一辺の長さが $r_{\rm cut}$ よりも長くすればよい．

3 次元系における CLL 法では，参照すべき隣接セルを特定するには **Algorithm5** のようにおこなう．隣接セルの特定作業をマッピングと呼ぶ．ここで用いている関数 icell(ix, iy, iz) の詳細は **Algorithm6** に示している． ここで 3 × 3 のセルの中心のセルインデックスを imap とする (図 3.2)．さらにこ

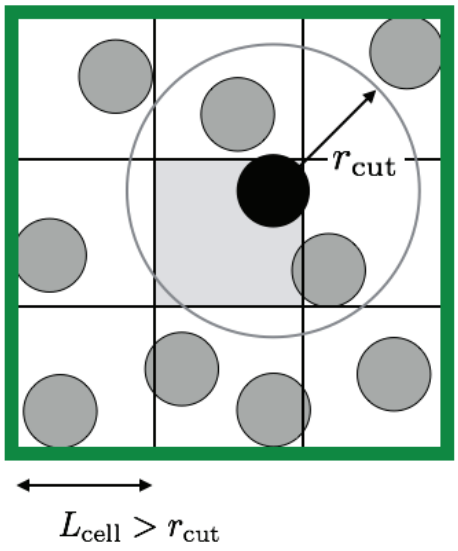


図 3.1 セル分割法 (CLL) の概念図：$L_{\rm cell}$ はセルの一辺の長さ，$r_{\rm cut}$ はカットオフ半径をそれぞれ表している．黒色粒子は中央の灰色セルに属しており，相互作用の計算の為に参照する粒子は隣接している白色セルに含まれている．

Algorithm 5 Create the maps of the cell list (Cell Index)

```
 1: int ix, iy, iz
 2: int imap, mapsize, maps[mapsize]
 3: int M (システムボックス一辺の分割数)
 4: mapsize = 13 * M * M * M (13 は周囲の参照セル数，M*M*M はシステムの全セル数)
 5: ===== Initialize the array of the map =====
 6: for  imap = 0 to mapsize do
 7:    map[imap] ← 0
 8: end for
 9: ===== create the maps of the cell list =====
10: for iz=0 to M do
11:    for iy=0 to M do
12:       for ix=0 to M do
13:          imap ← ( icellNo(ix, iy, iz) ) * 13;
14:          map[ imap + 1  ] ← icellNo( ix+1, iy   , iz   );
15:          map[ imap + 2  ] ← icellNo( ix+1, iy+1, iz   );
16:          map[ imap + 3  ] ← icellNo( ix   , iy+1, iz   );
17:          map[ imap + 4  ] ← icellNo( ix-1, iy+1, iz   );
18:          map[ imap + 5  ] ← icellNo( ix+1, iy   , iz-1);
19:          map[ imap + 6  ] ← icellNo( ix+1, iy+1, iz-1);
20:          map[ imap + 7  ] ← icellNo( ix   , iy+1, iz-1);
21:          map[ imap + 8  ] ← icellNo( ix-1, iy+1, iz-1);
22:          map[ imap + 9  ] ← icellNo( ix+1, iy   , iz+1);
23:          map[ imap + 10 ] ← icellNo( ix+1, iy+1, iz+1);
24:          map[ imap + 11 ] ← icellNo( ix   , iy+1, iz+1);
25:          map[ imap + 12 ] ← icellNo( ix-1, iy+1, iz+1);
26:          map[ imap + 13 ] ← icellNo( ix   , iy   , iz+1);
27:       end for
28:    end for
29: end for
```

Algorithm 6 function icellNo(ix, iy, iz): Return the Index of the Cell

```
1: int ix, iy, iz
2: int icellNo
3: int M (システムボックス一辺の分割数)
4: icellNo ← mod(ix + M, M)
                + mod(iy + M, M) * M
                + mod(iz + M, M) * M * M     (mod(a, b) は a を b で割った余りを返す)
5: return(icellNo)
```

の imap から周囲に隣接した参照セルのインデックスを imap+1 から imap+13 まで割り振る．周囲に隣接したセルは 26 個あるが，粒子間相互作用は作用・反作用の法則から図 3.2 に示すように，$N_{\mathrm{cell}} = (3 \times 3 \times 3 - 1)/2 = 13$ 個のみ参照できればよい．ここで N_{cell} は参照セル数と呼ぶ．つまり，配列 map[　] に隣接セルのインデックスを埋め込んで行く．このマッピングにはもちろん計算コストを要するが，空間を分割するだけであるから粒子がどのセルに属するかは考える必要はないため，MD を開始する前にあらかじめ行っておけばよい．

次に，各ステップにおいて計算される粒子の位置とそれに対応したセルインデックス imap をリンクさせ，隣接したセルから粒子ペアを特定する (**Algorithm7**)．ここで便宜上，システムの一辺を $[-L/2, L/2]$ から $[0, 1]$ に変数変換して i 粒子の位置を格納し，そこから対応するセルインデックス icell を決めていることに注意する．セルインデックス icell に含まれる i 粒子を配列 head[icell](はじめは 0 に初期化されている) から list[i] として保存する．その次に i 粒子のインデックスを head[icell] に上書き保存して行く．これを繰り返すことで CLL 法におけるリスト構造を実現する．すでに述べたように，この作業に

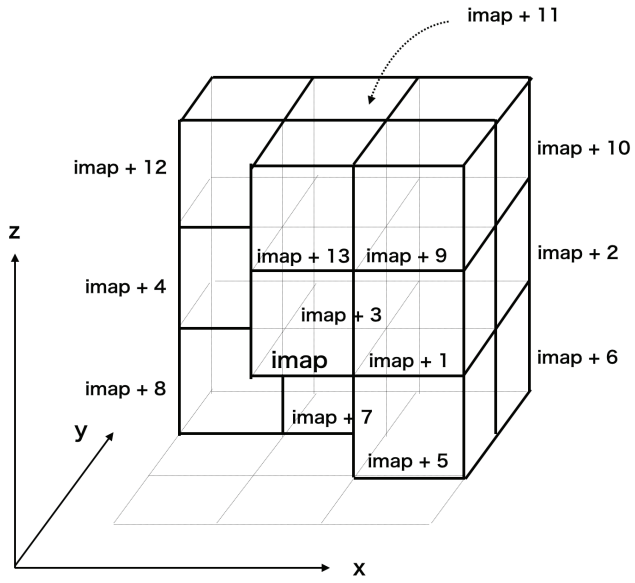


図 3.2　隣接セルのマッピング：中央のセル imap が参照する隣接セルのインデックスを imap+1 から imap+13 まで割り振る．作用反作用の法則から参照セルは対角方向のみを割り当てれば良い．

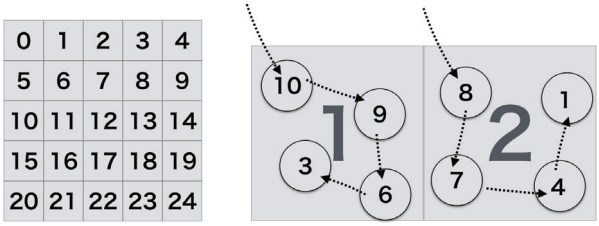


図 3.3　左：セルインデックス icell　右：粒子インデックス i のリンクを表している．配列 head[icell] を用いてセルに含まれている粒子インデックス i の先頭を参照し，粒子インデックス i から配列 list[i] を用いて次に内包している粒子インデックスを参照していく．

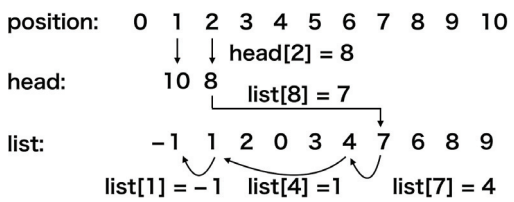


図 3.4　head 配列，list 配列のリンク構造の概念図：position2 のセルには head[2] = 8，つまり 8 番の粒子が先頭に入っている．さらに list[8] = 7，つまり 8 番粒子の次は 7 番粒子が入っている．list 配列が-1 を返すまでこれを繰り返す．

は各ステップ $\mathcal{O}(N)$ の計算コストを要する．リンク作業の具体的な様子を図 3.3，3.4 に示す．第 1 に，セルインデックスから配列 head[ ] を用いて，セルに含まれる最も大きいインデックスを持つ粒子を参照する．第 2 に次に小さいインデックスを持つ粒子を配列 list[ ] を用いて参照する．これを繰り返すことにより，最終的に-1 を終了判定として次のセルへとこの作業を移行する．例えば，図 3.4 では，セルインデックス 2 から head[2] = 8，つまり 2 のセルに含まれる最も大きなインデックスを持つ粒子は 8 だとわかる．ここから，list[8] = 7 であることから，8 の粒子の次は 7 を参照し，その次は 4，1，-1 といった具合である．リスト参照の具体的なアルゴリズムを **Algorithm8** に示している．

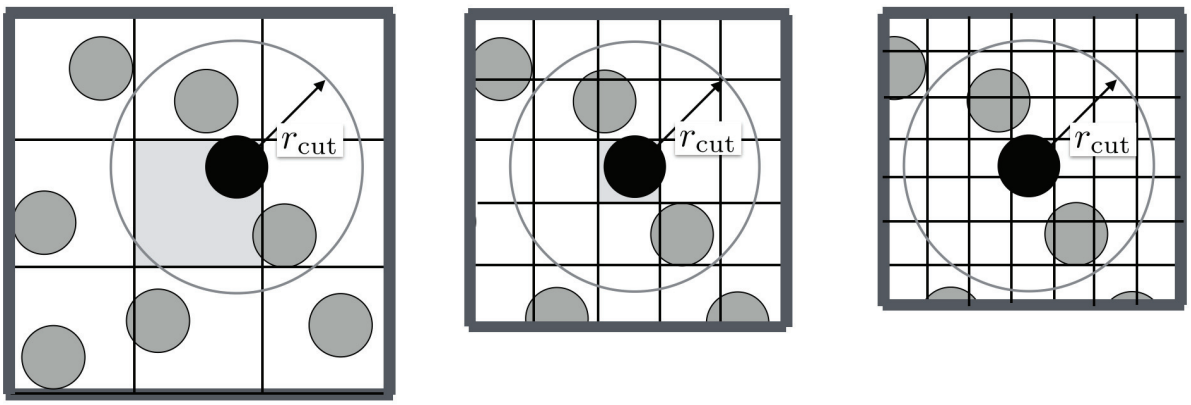
Algorithm5 から **Algorithm8** に基づき，全体のアルゴリズムをまとめると **Algorithm9** のようになる．つまり各ステップで相互作用を計算する前に隣接セルのマッピングを行い粒子の位置とセルの住所をリンクしセル参照を行う．

Algorithm 7 Link the Particle Index to the Cell Index

```
int icell, i, j
int M (システムボックス一辺の分割数)
double Systemsize (システムの一辺の長さ)
double cellsize (セルの一辺の長さ)
cellsize ← Systemsize / (double)M
if  cellsize < rcut  then
   Error: Cell size is too small for cutoff radius.
end if
for i=0 to NumParticle do
   dmyrxi ← rx[i] - floor( rx[i] + 0.5)
   dmyryi ← ry[i] - floor( ry[i] + 0.5)
   dmyrzi ← rz[i] - floor( rz[i] + 0.5)
   icell ← (int)( ( dmyrxi + 0.5 ) / (double)M )
                  + (int)( ( dmyryi + 0.5 ) / (double)M ) * M
                  + (int)( ( dmyrzi + 0.5 ) / (double)M ) * M * M
   list[i] ← head[icell]
   head[icell] ← i
end for
```

3.2 排他的セル分割法

CLL 法は実装が比較的簡単なことから良く用いられる手法であり，システムサイズに関係なく計算効率を $\mathcal{O}(N)$ で抑えることが出来る．しかしながら，少ない粒子数の系 ($\sim 1,000$ 粒子数以下) では VNL



(a) (b) (c)

図 3.5 排他的セル分割法: (a) 第 1 隣接セル参照法 ($f_c = 1, N_{\mathrm{cell}} = 3^3$ 個)，(b) 第 2 隣接セル参照法 ($f_c = 2, N_{\mathrm{cell}} = 5^3$ 個)，(c) 第 3 隣接セル参照法 ($f_c = 3, N_{\mathrm{cell}} = 7^3$ 個)．括弧の f_c は x, y, z 方向の隣接参照セル数，N_{cell} は参照セル数をあらわしている．太枠で囲った領域が参照体積 V_{ref}，円で囲まれた領域がカットオフ体積 V_{cutoff} である．参照セル数 N_{cell} は第 1 隣接セル参照なら 3^3 個あるが，粒子間相互作用は作用・反作用の法則から $(3^3 - 1)/2 = 13$ 個，第 2 隣接セル参照なら $(5^3 - 1)/2 = 62$ 個，第 3 隣接セル参照なら $(7^3 - 1)/2 = 171$ 個というように，対角方向に属する半分のセルのみを参照できればよい．

Algorithm 8 Use the cell linked List and calculate force

```
for icell=0 to Ncell do
   ======= select i particle contain in each cells ======
   i ← head[icell]
   while i >= 0 do
      ======= j particle contain in the current cell ======
      j ← list[i]
      while j >= 0 do
         Caluculate r_ij
         if r_ij < r_cut then
            Calculate force
         end if
      end while
      j ← list[j]
      ======= j particle contain in the neighbor cells ======
      jcell0 ← reference cell × icell
      for neighbor = 0 to reference cell do
         jcell ← map[ jcell0 + neighbor ]
         j ← head[ jcell ]
         while j >= 0 do
            Caluculate r_ij
            if r_ij < r_cut then
               Calculate force
            end if
            j ← list[j]
         end while
      end for
      i ← list[i]
   end while
end for
```

Algorithm 9 Cell Linked List

```
Create the map of the cell list.
====== MD Loop Start =====
for step = 0 to nstep do
   move r(t) → r(t+dt) and v(t) → v(t+dt/2)
   Link the particle index to the cell index.
   Use the cell linked list and calculate force.
   move v(t+dt/2) → v(t+dt)
end for
```

法に比べ計算効率が劣る．本質的に計算に必要とする領域は，カットオフ半径 $r_{\rm cut}$ 内に含まれる領域 (以下，カットオフ体積を $V_{\rm cutoff} = 4\pi r_{\rm cut}^3/3$ とする) であり，それに比べて参照しなければならない体積 (以下，参照体積 $V_{\rm ref} = 3^3 \times L_{\rm cell}^3$) が大きいことが要因として挙げられる．具体的に 3 次元系において $L_{\rm cell} = r_{\rm cut} = 2.5$ とすれば，$V_{\rm ref}/V_{\rm cutoff}$ は 6.4 倍である．つまり，カットオフ半径に含まれないが参照しなければならない領域が 85 % も存在していることを意味している．少ない粒子数の系に対して VNL 法の方が計算効率が優れているのは，VNL の参照体積は $V_{\rm ref} = 4\pi r_{\rm list}^3/3$ であり，$r_{\rm cut} = 2.5$, $r_{\rm list} = r_{\rm cut} + 0.5$ に取ったとしてもカットオフ球の外側にある参照領域はおおよそ 42 % である．粒子数が少なければ，VNL 法のボトルネックであったリスト更新の際の $\mathcal{O}(N^2)$ はそれほど全体の計算時間に対し甚大な寄与にはならない．したがって参照体積を小さく取ることが計算効率の向上に直結している．そ

Algorithm 10 Create the maps of the exclusive cell list (Cell Index)

```
int imap, mapsize, maps[mapsize], count
int M (システムボックス一辺の分割数)
int reference cell (参照セル数), adjacent cell (各方向の隣接セル数)
refernce cell ← (2.0∗adjacent cell+1)^3 − 1 / 2.0
=====Initialize the array of the map=====
for  imap = 0 to mapsize do
   map[imap] ← 0
end for
=====Create the array of the map=====
for iz=0 to M do
   for iy=0 to M do
      for ix=0 to M do
         imap ← icell(ix, iy, iz) * reference cell
         count ← 0
         for jx= -adjacent cell to adjacent cell do
            for jy= -adjacent cell to adjacent cell do
               for jz= -adjacent cell to adjacent cell do
                  if jy > 0 then
                     count ++
                     map[ imap + count] ← icell(ix + jx, iy + jy, iz + jz)
                  else if jy == 0 then
                     if jx >= 0 AND jx > 0 then
                        count ++
                        map[ imap + count] ← icell(ix + jx, iy + jy, iz + jz)
                     else if  jx >= 0 AND jz == 0 then
                        count ++
                        map[ imap + count] ← icell(ix + jx, iy + jy, iz + jz)
                     else if  jx == 0 AND jz < 0 then
                        count ++
                        map[ imap + count] ← icell(ix + jx, iy + jy, iz + jz)
                     else
                        ...nothing to do
                     end if
                  end if
               end for
            end for
         end for
         =====End of the adjacent cell loop=====
      end for
   end for
end for
=====End of the M loop=====
```

こで，セルをより細分化し，さらに外側のセルまで参照することを考える．つまり，セルサイズの条件を

$$L_{\mathrm{cell}} > r_{\mathrm{cut}} \qquad (f_c = 1) \tag{3.1}$$

$$\frac{r_{\mathrm{cut}}}{f_c - 1} > L_{\mathrm{cell}} > \frac{r_{\mathrm{cut}}}{f_c} \qquad (f_c > 1) \tag{3.2}$$

とすることで参照体積を小さくする事が可能となる (図 3.5)．ここで，f_c は x, y, z 方向の隣接参照セル数をあらわしている．$f_c = 1$ の場合は，前節に述べた従来型の CLL 法である．このようにセルの細分化を行う方法を排他的セル分割 (Exclusive cell linked list) 法と呼ぶ [16,17]．図 3.5 に排他的セル分割法の概念図を示した．セルをより細分化することで 1 つのセルに 1 つ以上の粒子が入ることができなくなることに注意する．

前節で説明され，また図 3.5(a) で示されている CLL 法は，第 1 隣接のセルまで参照する方法 (以下，第一隣接セル参照) である．この方法ではセルの一辺の長さ (セルサイズ $L_{\rm cell}$) はカットオフ半径 $r_{\rm cut}$ よりも大きいものとしていたが，第 2 隣接セル法では注目している粒子が属しているセルから 2 つ外側までセル参照を行う．この場合セルサイズは $r_{\rm cut} > L_{\rm cell} > r_{\rm cut}/2$ を満たさなければならない図 3.5(b)．セルサイズは

$$L_{\rm cell} = \frac{L}{M} \tag{3.3}$$

のようにシステム一辺を均等に分割して設定する．ここで一辺のセル分割数は M である．例えば，$N = 1,000$，$\rho = 0.8$，$T = 1.0$，$r_{\rm cut} = 2.5$ の系においてはシステム一辺の長さは $L \sim 10.772$ であり，

- $L_{\rm cell}/r_{\rm cut} \in [1.0772(M = 4), 1.436(M = 3)]$：第 1 隣接セル参照 ($f_c = 1$, 図 3.5(a))
- $L_{\rm cell}/r_{\rm cut} \in [0.478(M = 9), 1.0772(M = 4)]$ ：第 2 隣接セル参照 ($f_c = 2$, 図 3.5(b))
- $L_{\rm cell}/r_{\rm cut} \in [0.331(M = 13), 0.478(M = 9)]$ ：第 3 隣接セル参照 ($f_c = 3$, 図 3.5(c))

のようにセル分割を行うことができる．それぞれ，セル分割はシステムの一辺の長さを等分割しなければいけないことからセルサイズは離散値を取る．分割数 $M = 4$ とすれば

$$L_{\rm cell} = \frac{L}{M} = \frac{10.772}{4} \sim 2.693 \tag{3.4}$$

となり，さらに $M = 5$ とすれば

$$L_{\rm cell} = \frac{L}{M} = \frac{10.772}{5} \sim 2.154 \tag{3.5}$$

となるが，従来型の CLL 法 ($f_c = 1$) では $L_{\rm cell} \geq r_{\rm cut} = 2.5$ でなければならないので 5 分割はできない．しかし，第 2 隣接セル参照 ($f_c = 2$) の場合，2 つ隣りまで参照数を増やすことで $r_{\rm cut} > L_{\rm cell} \geq r_{\rm cut}/2$ のようにセルサイズを設定することが出来るようになる．第 2 隣接セル参照 ($f_c = 2$) を取れば 5 分割が可能になる．**Algorithm10** でアルゴリズムとして示しているように隣接セルのマッピングを行うことで，参照セル数を増やしセルサイズを小さくすることが可能となる．

3.3 セルサイズの最適化

排他的セル分割法を用いることで，従来型 CLL 法からさらなる計算効率の向上が期待される．セルを細かくすることで参照体積が小さくなり，計算に関係のない参照体積は減少する．しかし，その一方で参照しなければならないセル数が $3^3, 5^3, 7^3 \cdots$ と増加する．これによりセルインデックスのループに計算時間を要するようになる．従って，計算時間に対するセルサイズには最適値が存在することになる．そこでセル分割法に関しても，VNL 法と同様の解析的評価式を提案する．セル分割法のアルゴリズムをまとめると，

1. システムをセルで領域分割し，隣接セルのマッピング (**Algorithm9**)
2. 各粒子がそれぞれどのセルに属するかリンクしリストを作成 (**Algorithm7**)
3. セルに関するループ (**Algorithm8**)
4. その中に含まれる粒子で距離の計算 (**Algorithm8**)
5. さらにカットオフ半径内に含まれる粒子のみ相互作用の計算 (**Algorithm8**)

の 5 つのフェーズから構成される．この中で計算時間がセル数に依存するのは 1, 3, 4, 5 のフェーズである．フェーズ 2 は粒子のインデックスに関してループを行うだけであるから，計算時間はセル数には依存

しない．さらに，フェーズ 1 は相互作用計算を開始する直前に 1 回だけ行うため，全体の計算時間にはほとんど寄与しない．従って，計算時間のセルサイズ依存性が顕著にあらわれてくるのは 3, 4, 5 のフェーズとなる．以上の考察から，1 粒子が各ステップに要する計算時間の解析的評価式を

$$\begin{aligned} T_{\mathrm{CLL}} =& \frac{1}{2}(N_{\mathrm{cell}}+1)\times\tau_\alpha \\ &+\frac{1}{2}\left(\rho L_{\mathrm{cell}}^3(N_{\mathrm{cell}}+1)-1\right)\times\tau_\beta \\ &+\frac{4}{3}\frac{\pi r_{\mathrm{cut}}^3}{N_{\mathrm{cell}}L_{\mathrm{cell}}^3}\times\tau_\gamma \end{aligned} \tag{3.6}$$

のようにあらわすことができる [15]．右辺第 1 項はセルインデックスのループに要する時間，第 2 項はセル内に含まれる粒子インデックスのループに要する時間，第 3 項はカットオフ半径体積に含まれる粒子ペアに対して相互作用を計算する時間に対応している．ここで，τ_α,τ_β,τ_γ は計算機に依存し，あらかじめ 1 ステップでの，各フェーズに要する時間を計測することによって与えられる．式 (3.6) から明らかなように，CLL 法の 1 粒子あたりの計算時間は密度，カットオフ半径に依存するが温度には依存しない．つまり各ステップにおいて粒子の “位置” とセルのインデックスとをリンクし，隣り合ったセルから粒子のインデックスを参照してくるだけの手法であるから，この手順には一切粒子の “速度” に依存する要素は含まれない．言い換えれば，CLL 法はシステムの動的要素に依存しないリストアルゴリズムである．

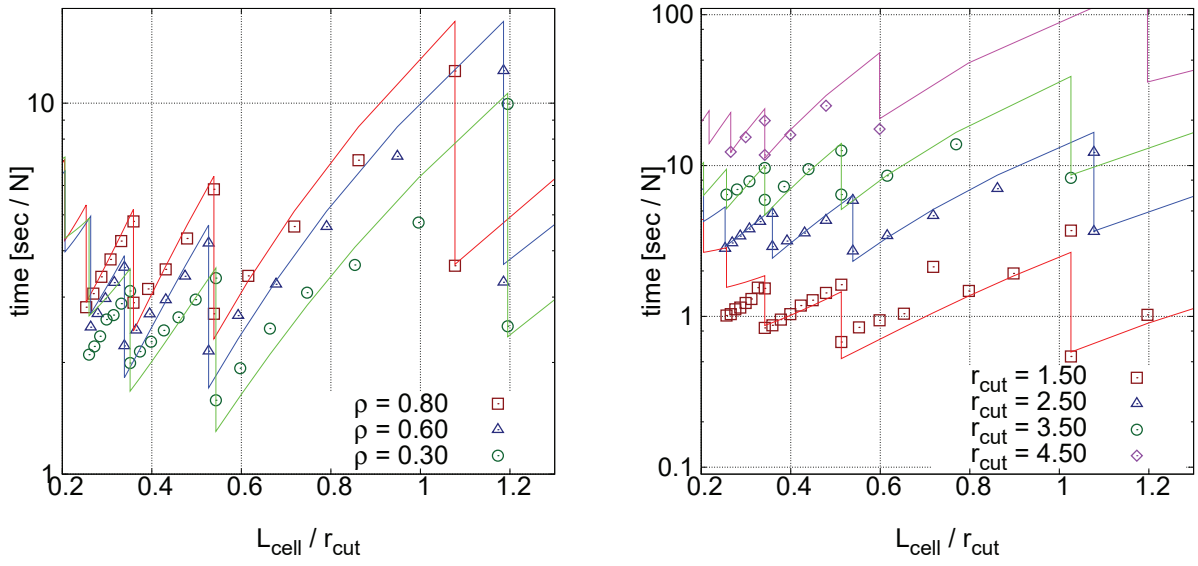
式 (3.6) による計算時間のセルサイズ依存性と，シミュレーションの結果を図 3.6(a),(b) に示す．VNL 法と同様，LJ ポテンシャルで NVE アンサンブルを 100,000 ステップ計算し，粒子数あたりの計算時間として評価している．例えば図 3.6(a) の赤線にあらわされるように，この系では第 2 隣接セル参照の手法で $L_{\mathrm{cell}}/r_{\mathrm{cut}} \sim 0.478$(9 分割) のセルサイズが最も効率が良い．各パラメータに対する依存性は以下のようにまとめられる．

(a) **密度依存性**

システムの体積 V は密度 ρ，粒子数 N によって決まり，システムの一辺の長さは $L=(N/\rho)^{1/3}$ で与えられる．このことからセルはシステムの一辺を均等に分割して作らなければならないので，取りうるセルサイズ L_{cell} はそれぞれの密度に応じて変わってくる．また，当然のことながら，高密度なほどカットオフ球体積に含まれる粒子数が増えるので計算コストがかさむ．これは式 (3.6) の第 2 項の寄与であるが，ただし，セル内に含まれる粒子ペア数のカウントに要する時間に対応しているので，カットオフ半径 $r_{\mathrm{cut}}=2.5$ ではどの密度においても第 2 隣接セル参照で最も細分化したセルサイズが最適であることがわかる．

(b) **カットオフ半径依存性**

カットオフ半径 $r_{\mathrm{cut}}=4.5$ では第 3 隣接セル参照 ($f_c=3$) が最適であり，$r_{\mathrm{cut}}=3.5, 2.5$ では第 2 隣接セル参照 ($f_c=2$) が最適，$r_{\mathrm{cut}}=1.5$ では第 1 隣接セル参照 ($f_c=1$) が最適であることがわかる．これは式 (3.6) の第 3 項に着目すると理解できる．カットオフ半径 r_{cut} が大きければ，対応して大きなセルでシステムを分割しなければならない．つまり，セル分割には式 (3.1) の条件を満たさなければならない．粒子数 $N=1,000$，密度 $\rho=0.8$ の系では一辺が $L_{\mathrm{cell}}=10.772$ となり，一辺を 3 等分することを考えると，セルサイズは $L_{\mathrm{cell}}=L/M=10.772/3=3.5906$ となり，カットオフ半径が $r_{\mathrm{cut}}=4.5$ であったとすると上記の条件を満たさない．つまり，カットオフ半径 $r_{\mathrm{cut}}=4.5$ の系ではそもそも第 1 隣接セル参照はできない．これらのことから，カットオフ半径が大きくなるにしたがい，セル分割数，隣接参照セル数を増やした方が計算効率が良くなる．



(a) 密度依存性　　(b) カットオフ半径依存性

図 3.6　計算時間のセルサイズ依存性：点はシミュレーションの結果を，実線は式 (3.6) をそれぞれ表している．(a) 密度依存性，(b) カットオフ半径依存性を検証した．計算時間を最小化する最適リスト半径 (SKIN) サイズはシステムサイズ，密度に依存する．セルサイズを小さいくとるとセル数が増え，セルインデックスのループに計算コストを要する．セルサイズを大きくすると参照体積 (粒子数) が増え，粒子数のループに計算コストを要する．図 (a) において，いずれの密度に対しても第 2 隣接セル参照法が最適である．図 (b) において，$r_{\rm cut} = 1.5$ では第 3 隣接セル参照 ($f_c = 3$) が最適，$r_{\rm cut} = 2.5$, $r_{\rm cut} = 3.5$ では第 2 隣接セル参照 ($f_c = 2$) が最適，$r_{\rm cut} = 4.5$ では第 1 隣接セル参照 ($f_c = 1$) が最適である．

第 4 章

VNL-CLL 法

第 2 章で紹介したベルレの近接リスト法 (Verlet Neighbor List: VNL 法)，第 3 章で紹介したセル分割法 (Cell Linked List: CLL 法) および，排他的セル分割法の長所・短所を以下にまとめる．

- **VNL 法**
  - リスト更新に $\mathcal{O}(N^2)$ の計算コストを要するため，10,000 粒子数以上の系に対してはそれがボトルネックとなり現実的でない．一方で，少ない粒子数の系に対しては高いパフォーマンスを発揮する．
  - 最も速く移動する粒子によって更新間隔 n が決定されてしまう．これは気液共存系などの場合に適さないが，運動性の乏しい結晶固体系などには n は長くなるメリットがある．
  - リスト半径 r_{list} を最適に選ぶためには試行錯誤が必要であったが，式 (2.8) によって決定することが可能．
  - CLL 法に比べ参照体積が小さく，リスト上に保っておく粒子数も少なく (メモリ量が少ない)，その分だけ計算速度は速い．
  - 並列計算との相性が悪い．
- **CLL 法**
  - 全体として $\mathcal{O}(N)$ の計算量であるため，10,000 粒子数以上の系に対しても高いパフォーマンスを実現することが可能．
  - 粒子の運動性に影響しないため，気液共存系などの不均一系の場合に対してもデメリットはない．
  - 排他的セル分割法では，最適化なセルサイズは自明ではなかったが，式 (3.6) によって決定することが可能．
  - VNL 法と比べ参照体積は大きく，リスト上に保っておく粒子数は多い．さらにリスト配列も複数ある (メモリ量が多い)．その分だけ計算速度は遅い．
  - 並列計算との相性が良い．

これらより，参照体積の点で VNL 法では計算効率が良いが，数ステップに一度リスト更新 $\mathcal{O}(N^2)$ を必要とするためそれがボトルネックとなっている．そこで，このリスト更新によるボトルネックを CLL 法を用いて補うことが出来ればさらなる高速化が期待される．そこで VNL-CLL 法を提案する．

4.1 概要

VNL-CLL 法は VNL 法の動的更新を基本としている．つまり，更新するまでの間 n ステップはベルレの近接リストを用いて相互作用を計算する．つまり，各ステップに近接リストを更新するか否かチェックを行い，動的リスト更新の手法を採用する．$n+1$ ステップ目に全粒子探索 ($\mathcal{O}(N^2)$) を行うことで，VNL を更新するのではなく，その代わりに CLL($\mathcal{O}(N)$) を用いて粒子探索を行う．これにより全体で $\mathcal{O}(N)$ の計算を実現することが可能となる．VNL を基本的に用いており，さらに参照体積を最小化することができることから高速化が見込める．VNL-CLL 法の詳細なアルゴリズムは **Algorithm11** であり，**Algorithm12**, **Algorithm13** から構成されている．

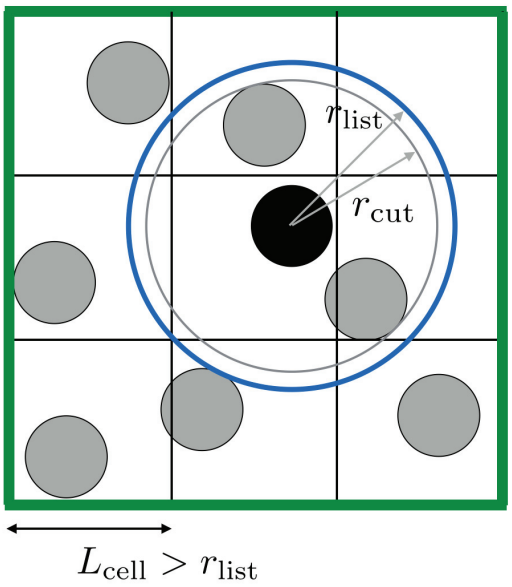


図 4.1 VNL-CLL 法の概念図：青色円は VNL 法における近接リスト半径をもった円，CLL 法において緑四角は参照する隣接セル領域を表している．セルの中に Verlet の近接リストを内包させるため，セルサイズの取りうる大きさの下限が r_{list} で決まっていることに注意する．

Algorithm 11 VNL-CLL

```
1: Create the map of the cell list.
2: Make the VNL and calculate force.
3: ====== MD loop start ======
4: for  step = 0 to nstep do
5:    move r(t) → r(t+dt) and v(t) → v(t+dt/2)
6:    Check update of the VNL.
7:    if  Need update of the VNL  then
8:       ======Make the VNL using CLL======
9:       Link the particle index to the cell index.
10:      Use the CLL and make the VNL.
11:      Calculate force.
12:   else
13:      ======Use the VNL======
14:      Use the verlet neighbor list and calculate force.
15:   end if
16:   move v(t+dt/2) → v(t+dt)
17: end for
```

Algorithm 12 VNL-CLL: Make the VNL using CLL

```
for icell = 0 to Ncell do
  ======select i particle contain in each cell======
  i ← head[icell]
  while i >= 0 do
    ======j particle contain in the current cell======
    j ← cell list[i]
    while j >= 0 do
      calculate r_ij
      if r_ij < r_list then
        neighbor list[i][point[i]] ← j
        point[i] ++
        if r_ij < r_cut then
          calculate force
        end if
      end if
      j ← cell list[j]
    end while
    ======j particle contain in the neighbor cell======
    jcell0 ← reference cell * icell
    while neighbor = 1 to reference cell do
      jcell ← map[ jcell0 + neighbor ]
      j ← head[jcell]
      while j >= 0 do
        calculate r_ij
        if r_ij < r_list then
          neighbor list[i][point[i]] ← j
          point[i] ++
          if r_ij < r_cut then
            calculate force
          end if
        end if
        j ← cell list[j]
      end while
    end while
    i ← cell list[i]
  end while
end for
for i=0 to NumParticle do
  end point[i] ← point[i]
end for
```

第 1 に，CLL 法と同様に相互作用を計算する前に隣接セルインデックスのマッピングを行う．第 2 に，初期配置をもとに Verlet の近接リストを作成し相互作用を計算する．位置の更新，速度の更新をした後，VNL を更新するか否かチェックする．更新するのであれば CLL 法を用いて VNL を再構成する (**Algorithm12**)．更新しないのであれば，そのまま VNL を用いて力を計算する (**Algorithm13**)．ここで，VNL-CLL 法ではセルサイズの設定条件が CLL 法とは異なることに注意する．CLL 法ではカットオフ半径より大きな一辺を持ったセルでシステムを均等に分割していたが，VNL-CLL 法では参照セルの内部にカットオフ半径 r_{cut} だけでなく VNL 法のリスト半径 r_{list} も含まれる．したがって，セルサイズ L_{cell} の取りうる条件は以下のようになる．

$$L_{\mathrm{cell}} > r_{\mathrm{list}} \qquad (f_c = 1) \tag{4.1}$$

$$\frac{r_{\mathrm{list}}}{f_c - 1} > L_{\mathrm{cell}} > \frac{r_{\mathrm{list}}}{f_c} \quad (r_{\mathrm{list}} = r_{\mathrm{cut}} + \mathrm{SKIN}) \qquad (f_c > 1) \tag{4.2}$$

Algorithm 13 VNL-CLL: Use the VNL

```
 1: for i=0 to NumParticle do
 2:   j_begin ← 0
 3:   j_end ← end point[i] - 1
 4:   if j_begin <= j_end then
 5:     for j_nab = j_begin to j_end do
 6:       j ← neighbor list[i][jnab]
 7:       if j > -1 then
 8:         calculate r_ij
 9:         if r_ij < r_cut then
10:           calculate force
11:         end if
12:       end if
13:     end for
14:   end if
15: end for
```

となる．セルサイズはリスト半径によって下限が決められていることに注意が必要である．

4.2 リスト半径・セルサイズの最適化

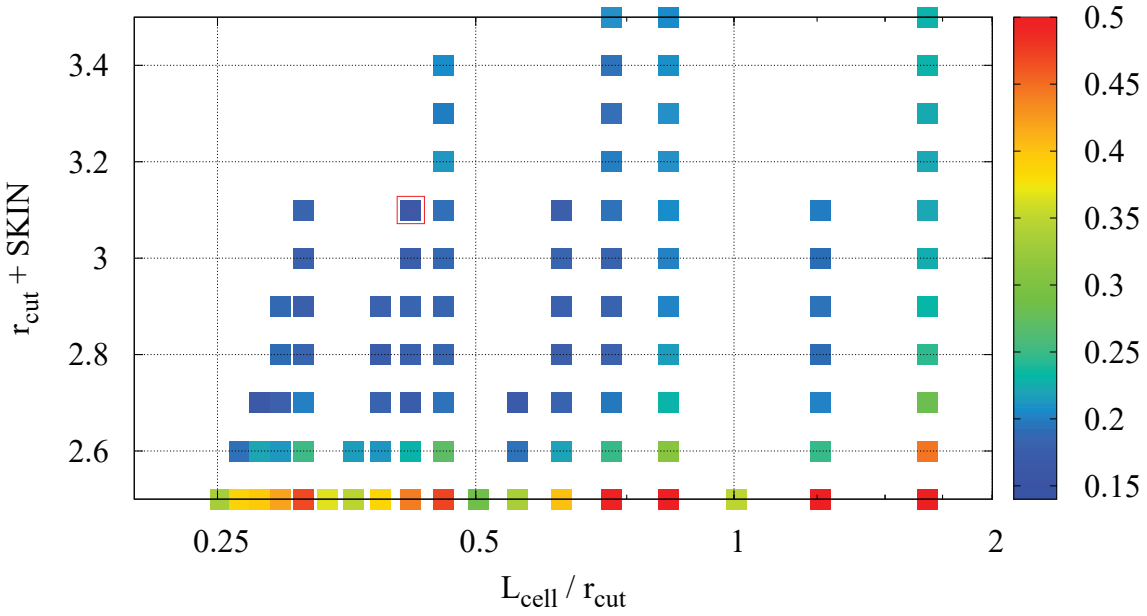
VNL 法，CLL 法の結果に基づいて VNL-CLL 法に対する 1 粒子が 1 ステップに要する計算時間に対する解析的評価式は

$$
\begin{aligned}
f_{\mathrm{VNL-CLL}} &= \frac{1}{n+1}\frac{T_{\mathrm{CLL}}}{N}(\tau_{\mathrm{CLL}}/\tau_{\mathrm{VNL}}) + \frac{n}{n+1}\frac{1}{N-1}\left(\frac{4}{3}\pi\rho(r_{\mathrm{cut}}+\mathrm{SKIN})^3 - 1\right) \\
&= \frac{1}{n+1}\frac{T_{\mathrm{CLL}}}{N}(V_{\mathrm{ref}}^{\mathrm{CLL}}/V_{\mathrm{ref}}^{\mathrm{VNL}}) + \frac{n}{n+1}\frac{1}{N-1}\left(\frac{4}{3}\pi\rho(r_{\mathrm{cut}}+2n\langle x\rangle)^3 - 1\right)
\end{aligned}
\tag{4.3}
$$

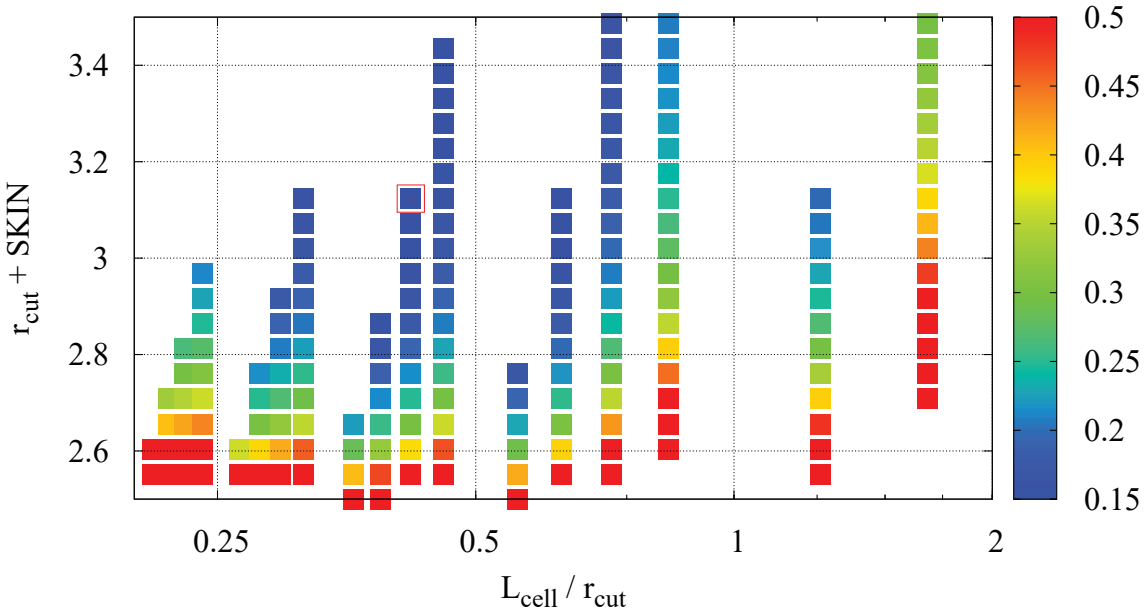
のようになる．VNL 法を基にベルレの近接リストを更新する際に CLL 法を用いるのであるから，右辺第 1 項のリスト更新に対応する項に CLL 法に要する時間 T_{CLL} が含まれる．τ_{CLL}, τ_{VNL} はそれぞれ，1 ステップに CLL，VNL を使用し計算する時間であるが，ともに参照体積に比例する．実際には，半径 r_{list} の球である VNL の参照体積と，CLL 法での参照体積は一辺 $2r_{\mathrm{list}}$ の立方体との体積比であるから，

$$
\frac{\tau_{\mathrm{CLL}}}{\tau_{\mathrm{VNL}}} = \frac{V_{\mathrm{ref}}^{\mathrm{CLL}}}{V_{\mathrm{ref}}^{\mathrm{VNL}}} = \frac{(2r_{\mathrm{list}})^3}{4\pi r_{\mathrm{list}}^3/3} \simeq 1.91 \tag{4.4}
$$

として考えれば良い．したがって，VNL-CLL 法の計算時間は温度 T，密度 ρ，粒子数 N，カットオフ半径 r_{cut} に依存する．計算時間の SKIN 依存性，セルサイズ依存性として図 4.2(b) に，シミュレーションの結果を，図 4.2(a) に式 (4.3) の結果を示す．カラーバーは粒子数あたりの計算時間を表しており，横軸はセルサイズ，縦軸は SKIN である．セルサイズ L_{cell} が大きいと，その参照体積が大きく，計算コストが大きい．セルサイズが小さすぎるとセル数が増え計算コストがかかってしまう．また，セルサイズはシステムの一辺を均等に分割しなければならないため離散値を取る．SKIN は大きすぎると，VNL の更新頻度を低減することができるが，参照体積が大きく計算コストが大きい．また，SKIN の上限はセルサイズによって決まっている．SKIN が小さいと更新頻度が上がる．更新には CLL 法を用いるため $\mathcal{O}(N^2)$ ではないにせよ，参照体積は CLL 法の方が大きいため計算コストを要する．したがって，計算コストを最適とする両者の組み合わせが存在する．各パラメータに対する依存性は以下のようにまとめられる．



(a) シミュレーションの結果.



(b) 式 (4.3) の結果.

図 4.2　計算時間に対するリスト半径 (SKIN サイズ)，セルサイズ依存性．$N = 1,000$，$T = 1.0$，$\rho = 0.5$, $r_{\rm cut} = 2.5$ で 1 粒子あたりの計算時間を計測した．この系において最適リスト半径 $r_{\rm list} = r_{\rm cut} + {\rm SKIN}$，最適セルサイズの組み合わせはそれぞれ $r_{\rm list} = 3.1$, $L_{\rm cell}/r_{\rm cut} = 0.42 (f_c = 3)$((a) シミュレーション)，$r_{\rm list} = 3.15$, $L_{\rm cell}/r_{\rm cut} = 0.42 (f_c = 3)$((b) 式 4.3) である．セルサイズが大きいと参照体積が増え，計算コストを要する．セルサイズが小さいとセル数が増え，セルインデックスのループに計算コストを要する．SKIN サイズが大きいと参照体積が増え，計算コストを要する．SKIN サイズが小さいと更新頻度が増え，リストの再構成に計算コストを要する．

(a) **粒子数依存性 (図 4.3)**

温度 $T = 1.0$，密度 $\rho = 0.8$，カットオフ半径 $r_{\rm cut} = 2.5$ とし，システムサイズ (粒子数)$N = 500, 1,000, 10,000, 100,000$ の 4 つの系に対し計算時間の SKIN，セルサイズ依存性をプロットした．粒子数 N が大きいとシステムの体積 $L_{\rm box}$ も大きくなる．セルサイズはシステムの一辺を均等に分割するため，それだけシステムが大きい方が分割するセルの間隔を細かくとることができる．$n+1$ ステップに 1 度行う更新では VNL-CLL 法は CLL 法をベースとするため $\mathcal{O}(N^2)$ ではなく，$\mathcal{O}(N)$ に計算コストを低減するが，それでもなお VNL 法の方が計算効率は良い．VNL のでは，粒子数が増えると計算コストがリスト更新のため $\mathcal{O}(N^2)$ のボトルネックを残していたが，CLL 法とハイブリッド化することで更新のボトルネックは CLL の計算コストに置き換わる．

(b) **温度依存性 (図 4.4)**

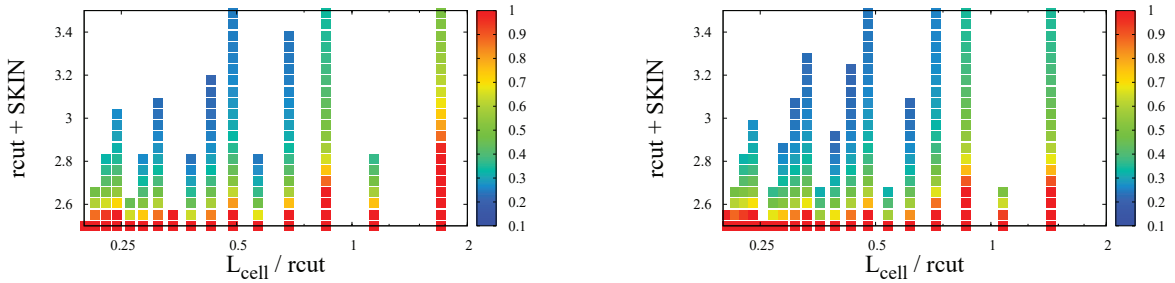
粒子数 $N = 1,000$，密度 $\rho = 0.8$，カットオフ半径 $r_{\rm cut} = 2.5$ とし，温度 $T = 0.2, 0.5, 1.0, 2.0$ の 4 つの系に対して計算時間の SKIN，セルサイズ依存性をプロットした．システムの温度が高いほど粒子速度は平均的に増加することから，それに伴い粒子がリストの範囲から出る頻度も増加する．そのため温度が高い系では，なるべく更新間隔 n を大きくしリストをより長く使い続けた方が計算効率が良くなる．したがって SKIN は大きく取った方が良い．VNL 法のみの場合は低温にするほど，ただ SKIN を長く取ればよいだけであったが，VNL-CLL 法の場合，SKIN はセルサイズによって上限が決まっているため，それほど長く取る事はできない．セルサイズを大きく設定し，SKIN を長く取ろうとしても，そもそもセルサイズが大きいと参照体積が大きくなり，計算コストがかかってしまう．これは VNL-CLL 法にすることで更新によるボトルネックが支配的でなくなった結果であると言える．

(c) **密度依存性 (図 4.5)**

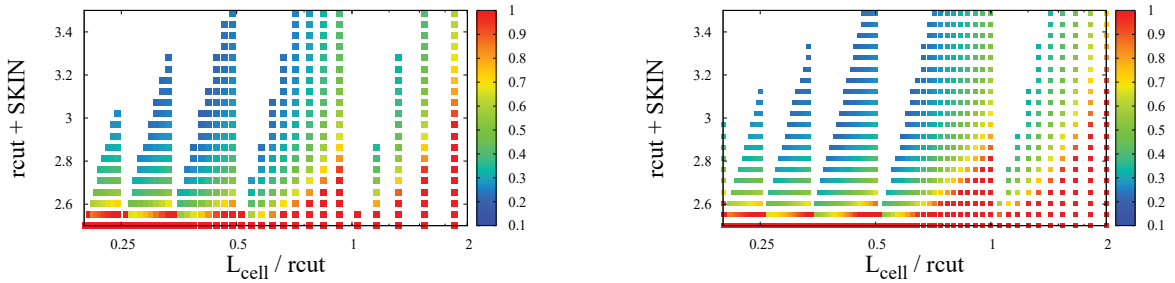
粒子数 $N = 1,000$，温度 $T = 1.0$，カットオフ半径 $r_{\rm cut} = 2.5$ とし，密度 $\rho = 0.3, 0.5, 0.8, 1.0$ の 4 つの系に対して計算時間の SKIN，セルサイズ依存性をプロットした．密度によりシステムの体積 $L_{\rm box} = V/\rho$ は変化する．したがって，低密度なほどシステムの一辺の長さは大きくなり，　その分だけセル分割を細かくする事ができる．また，密度はリストに含まれる粒子数に関係する．明らかに，低密度であればリストに記憶する相互作用ペアの数は小さくなり，計算コストも下がる．また，低密度になるにつれ粒子が移動する平均自由行程が長くなる．これにより，その分だけ粒子はリストの範囲から出やすくなり頻繁に $\mathcal{O}(N^2)$ の更新を行ってしまう．したがって，更新間隔を長くした方が計算効率は向上する．そのため，低密度なほど最適 SKIN は大きくなるが，温度依存性と同様に SKIN はセルサイズで上限が決まる．

(d) **カットオフ半径依存性 (図 4.6)**

粒子数 $N = 1,000$，温度 $T = 1.0$，密度 $\rho = 0.8$ とし，カットオフ半径 $r_{\rm cut} = 1.5, 2.0, 2.5, 3.0$ の 4 つの系に対し計算時間の SKIN，セルサイズ依存性をプロットした．カットオフ半径が大きいほど，カットオフ球体積内に含まれる相互作用を計算するペアの数は多くなる．対応して，SKIN，セルサイズもその分だけ大きく取らなければならない．つまり，リストに含まれる粒子ペアも多くなる．カットオフ半径 $r_{\rm cut}$ が大きくなるにつれ，リスト更新による計算コストの寄与は相対的に小さくなる．

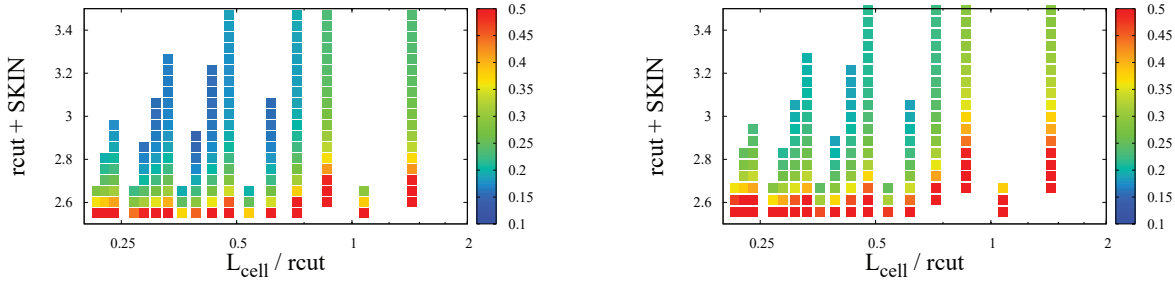


$N = 500$ $N = 1,000$

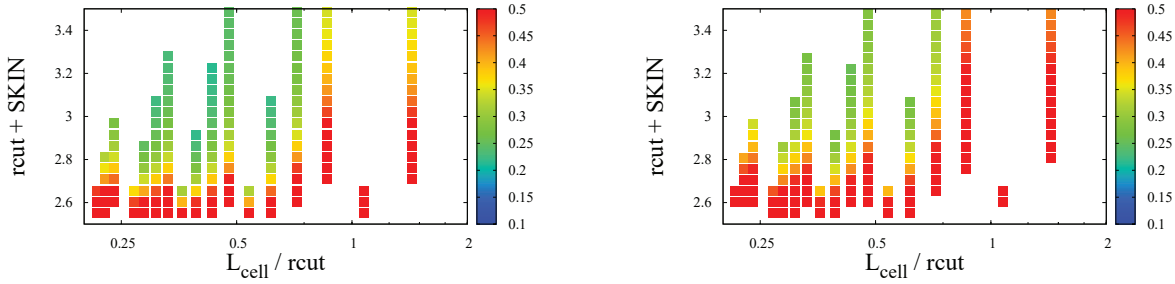


$N = 10,000$ $N = 100,000$

図 4.3　式 (4.3) の結果：計算時間の SKIN，セルサイズ依存性．$T = 1.0$，$\rho = 0.8$，$r_{\mathrm{cut}} = 2.5$ としてそれぞれ粒子数を変化させている．

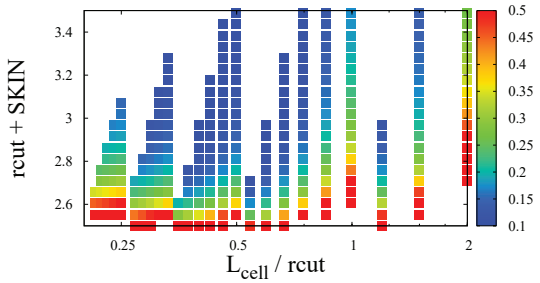


$T = 0.2$ $T = 0.5$

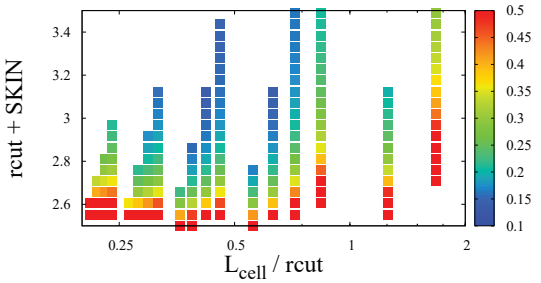


$T = 1.0$ $T = 2.0$

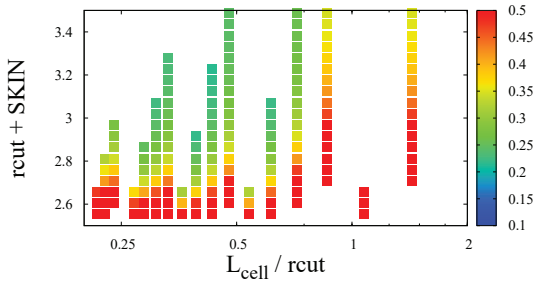
図 4.4　式 (4.3) の結果：計算時間の SKIN，セルサイズ依存性．$N = 1,000$，$\rho = 0.8$，$r_{\mathrm{cut}} = 2.5$ としてそれぞれ温度を変化させている．



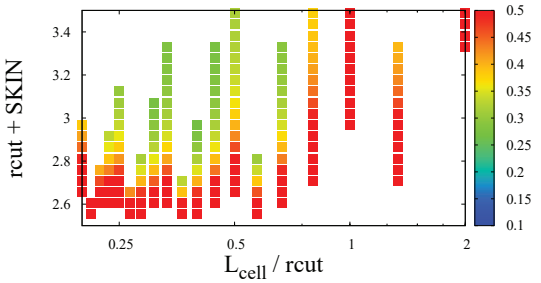
$\rho = 0.3$



$\rho = 0.5$

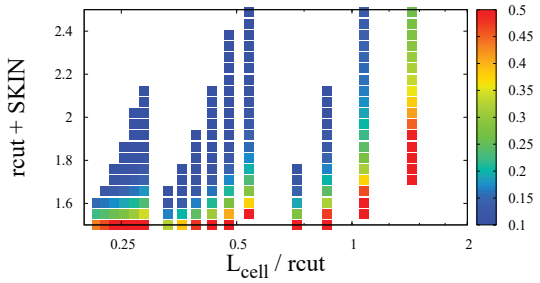


$\rho = 0.8$

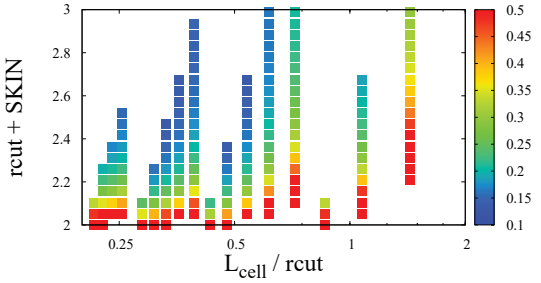


$\rho = 1.0$

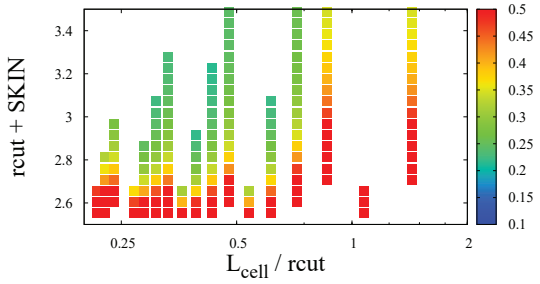
図 4.5　式 (4.3) の結果：計算時間の SKIN，セルサイズ依存性．$N = 1,000$，$T = 1.0$，$r_{\mathrm{cut}} = 2.5$ としてそれぞれ密度を変化させている．



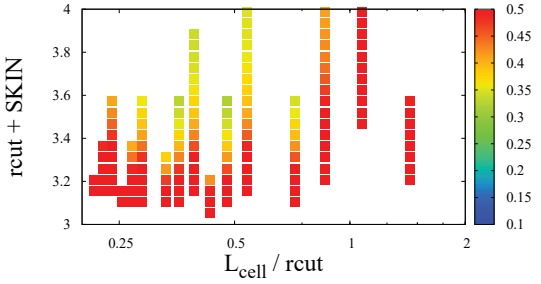
$r_{\mathrm{cut}} = 1.5$



$r_{\mathrm{cut}} = 2.0$



$r_{\mathrm{cut}} = 2.5$



$r_{\mathrm{cut}} = 3.0$

図 4.6　式 (4.3) の結果：計算時間の SKIN，セルサイズ依存性．$N = 1,000$，$T = 1.0$，$\rho = 0.8$ としてそれぞれ粒子数を変化させている．

第 5 章

結論

本論文では，LJ ポテンシャルなどの短距離相互作用をモデルポテンシャルとする短距離古典分子動力学法の計算高速化アルゴリズムに対して最適化技法を構築した．Verlet の近接リスト法 (Verlet Neighbor List: VNL 法) では，動的リスト更新とそれに伴う計算時間のリスト半径 (SKIN) 依存性，リスト半径の最適化に関して述べた．セル分割法 (Cell Linked List: CLL 法) では，より高度な手法である排他的セル分割法 (Exclusive CLL 法) に対して，計算時間のセルサイズ依存性，セルサイズの最適化を行った．これらに基づき，本論文では新たに VNL 法，CLL 法の両者を組み合わせた手法である VNL-CLL 法の最適化手法を構築した．この最適化手法により，計算システムを決めるパラメータ (粒子数 N, 温度 T, 密度 ρ, カットオフ半径 r_{cut}) の組み合わせに対し，計算効率を最大化する最適リスト半径 (SKIN)，最適セルサイズの組み合わせが一意に決定される．SKIN= 0.5 に固定した VNL 法、SKIN を最適化した VNL 法、CLL 法、SKIN= 0.5 に固定した VNL-CLL 法、SKIN、セルサイズ共に最適化した VNL-CLL 法に関して計算効率の比較を行った。図 5.1 にそれぞれの手法における計算パフォーマンスについて結果をまとめた．パフォーマンスは単位時間あたりの粒子数 × ステップ数として定義した。よって，この値が高いほど計算性能が良いことになる．

粒子数が小さい系 ($N < 1000$) に対しては、全粒子探索 $\mathcal{O}(N^2)$ の効果が少ないため CLL 法に比べ、SKIN を最適化した VNL 法が圧倒的に優れている。粒子数が大きい系に対しては VNL 法は $\mathcal{O}(N^2)$ のため急激に計算効率が低下してしまう。一方で CLL 法は $\mathcal{O}(N)$ を維持する。

VNL 法、CLL 法の両者の利点を上手く組み合わせることが出来る VNL-CLL 法では、今回、SKIN とセルサイズの最適化を行う事によってさらなる計算効率の向上に成功した。CLL 法と比較して 2 ∼ 3 倍の計算性能を実現する。粒子数の小さな系に対しては VNL 法の恩恵を顕著に受けることができ大きな系に対してもシステムサイズによらず $\mathcal{O}(N)$ を維持することが可能である。

本研究では，シングルコアプロセッサにおける VNL-CLL 法の計算効率最適化を行った．これは並列化によるさらなる大規模計算を見据えた上での最適化パラメータを提供している．並列計算においては一般に，VNL 法に比べ CLL 法が並列化効率が良い事が知られており，領域 (セル) ごとにスレッドを割り当てることができるためである [18–20]．したがって，本研究における VNL-CLL 法に関しても，同様に高い並列化効率が期待される．並列計算における高速化技法としては，CLL 法で用いる配列のインデックスを並び替える手法 (Pairwise Cell List) [21] や，セルの領域内で粒子のインデックスをソートする手法 (空間ソーティング) [22–26]，さらに GPU を用いた並列計算 [27, 28] など様々な視点からのアプローチが盛んに行われている．今後，それらの要素を複合的に取り入れた VNL-CLL 法を実装し，パラメータ依存する計算時間の最適化が必須の課題となる．

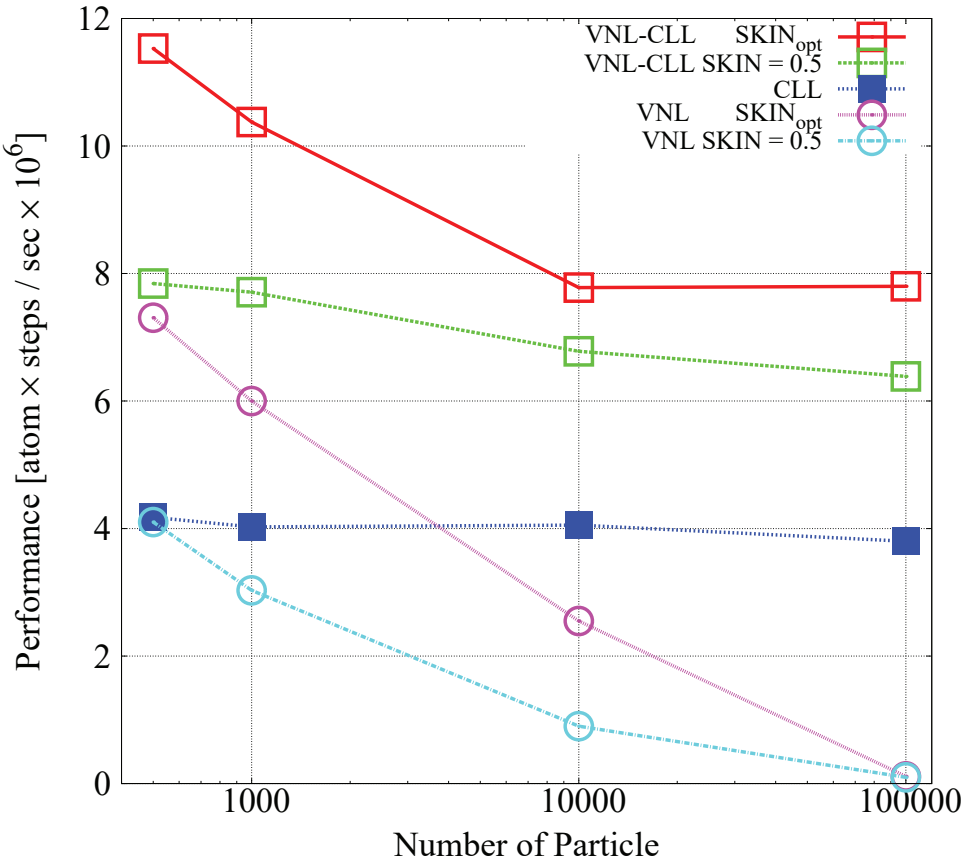


図 5.1　各種法におけるパフォーマンス：(赤) 最適化 VNL-CLL 法，(緑)SKIN=0.5 とした VNL-CLL 法，(青)CLL 法，(紫) 最適化 VNL 法，(水色)SKIN=0.5 とした VNL 法に対する計算性能を表している．

謝辞

本研究成果は自然科学研究機構計算科学研究センターの計算機を利用して得られたものです.

本論文をまとめるにあたり、多くの方々の御協力を頂きました。皆様に心より感謝申し上げます。

指導教官である金鋼准教授には、本研究を進めるにあたり非常に多くの助言を頂きました。本研究がこの様に一応の決着を迎えられたのは、ひとえに金先生のご尽力のおかげです。この場を借りて深く御礼申し上げます。

東京大学物性研究所の渡辺宙志先生，名古屋工業大学の礒部雅晴先生，東北大学金属材料研究所の芝隼人先生には研究会において，丁寧かつ有益な助言を頂きました.

新潟大学理学部物理学科物性理論研究室の先生方、大野義章教授、吉森明教授、奥西巧一准教授、柳瀬陽一准教授にも研究内容へのご意見や、研究室での活動など様々なご支援、ご協力を頂いております。

物性理論研究室の先輩方にも本研究を進めるにあたり、多くのご助言を頂きました。特に山田武見さん、関孝一さん、中村康晴さんにはプログラミングに関する応用の議論や数値計算の高速化、プレゼンテーションの流れなど、細かな部分にまで数々の助言を頂きました。ありがとうございます。

同研究室である、樋口沙希さん、池田光佑くんには研究内容の議論から日々のゼミまで、多くの知識を得る事ができました。

同期、後輩のみなさんには、研究意外の場面でも大変お世話になりました。皆様のおかげで博士前期課程をとても楽しく過ごす事ができました。

最後に、私を支えてくれた家族と友人に感謝します。

参考文献

[1] 岡崎 進,「コンピュータ・シミュレーションの基礎」, 化学同人 (2000).
[2] 上田 顕,「計算物理入門 分子シミュレーションを中心に」, サイエンス社 (2001).
[3] 上田 顕,「分子シミュレーション–古典系から量子系手法まで–, 裳華房 (2003).
[4] 田中 實, 山本 良一,「計算物理学と計算化学」, 海文堂 (1988).
[5] 佐藤 明,「HOW TO 分子シミュレーション –分子動力学法, モンテカルロ法, ブラウン動力学法, 散逸粒子動力学法–」, 共立出版 (2004).
[6] J. P. Hansen, I. R. Mcdonald, "Theory of Simple Liquids, Fourth Edition", Elsevier (2013).
[7] D. J. Evans, Gary. Morriss, "Statistical Mechanics of Nonequilibrium Liquids", Cambridge University Press (2008).
[8] M. E. Tuckerman, "Statistical Mechanics: Theory and Molecular Simulation", Oxford University Press (2010).
[9] D. Frenkel, B. Smit, "Understanding Molecular Simulation", Academic Press (2002).
[10] M. P. Allen, D. J. Tildesley, "Computer Simulation of Liquids", Oxford University Press (2009).
[11] D. C. Rapaport, "The Art of Molecular Dynamics Simulation", Cambridge University Press (1995).
[12] A. A. Chialvo and P. G. Debenedetti, "On the use of the Verlet neighbor list in molecular dynamics", Computer Physics Communications **60** (1990) 215-224.
[13] A. A. Chialvo and P. G. Debenedetti, "On the performance of an automated Verlet neighbor list alogritm for large systems on a vector processer", Computer Physics Communications **64** (1991) 15-18.
[14] A. A. Chialvo and P. G. Debenedetti, "An automated Verlet neighbor list algorithm with a multiple time-step approach for the simulation of large systems", Computer Physics Communications **70** (1992) 467-477.
[15] G. Sutmann, V. Stegailov, "Optimization of neighbor list techniques in liquid matter simulations", Journal of Molecular Liquids **125** (2006) 197-203.
[16] M. Isobe, "Simple and efficient algorithm for large scale molecular dynamics simulation in hard disk sistem" International Journal of Modern Physics C **10** (1999) 1281-1293.
[17] W. Mattson, B. M. Rice, "Near-neighbor calculations using a modified cell-linked method", Computer Physics Communications **119** (1999) 135-148.
[18] 牛島 省,「OpenMP によるプログラミングと数値計算表」, 丸善株式会社 (2006).
[19] 菅原 清文,「C/C++ プログラマーのための OpenMP 並列プログラミング」, 株式会社シナノ (2009).

[20] 金田 康正,「並列数値処理 –高速化と性能向上のために–」, コロナ社 (2010).

[21] P. Gonnet, "Pairwise Verlet Lists: Combining Cell Lists and Verlet Lists to Improve Memory Locality and Parallelism", Journal of Computational Chemistry **33** (2012) 76-81.

[22] Z. Yao, J. S. Wang, G. R. Liu and M. Cheng, "Improved neighbor list algorithm in molocular simulations using cell decomposition and data sorting method", Computer Physics Communications **161** (2004) 27-35.

[23] S. Meloni, M. Rosati, "Efficient particle labeling in atomistic simulations", Journal of Computational Chemistry **126** (2007) 121102

[24] U. Welliing, G. Germano, "Efficiency of linked cell algorithms", Computer Physics Communications **182** (2011) 611-615.

[25] H. Watanabe, M. Suzuki and N. Ito, "Efficient Implementations of Molecular Dynamics Simulations for Lennard-Jones Systems", Progress of Theoretical Physics, **126** (2011) 203-235.

[26] H. Watanabe, M. Suzuki and N. Ito, "Huge-scale Molecular Dynamics Simulation of Multibubble Nuclei", Computer Physics Communicaitons, **184** (2013) 2775-2784.

[27] J. A. Anderson, C. D. Lorenz, A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units", Journal of Computational Physics, **227** (2008) 5342-5359.

[28] A. J. Proctor, C. A. Stevens and S. S. Cho, "GPU-Optimized Hybrid Neighbor/Cell List Algorithm for Coarse-Grained MD simulations of Protein and RNA Folding and Assembly", Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine, (2013) 633-640.