GENESIS 向け
区間多項式近似短距離力カーネル最適化報告書

12/2018

コデザイン推進チーム

要約

本文書は，分子動力学シミュレーションパッケージソフト「GENESIS」の近距離相互作用計算カーネルのポスト「京」で採用されるプロセッサ「A64FX」に向けた最適化に関する報告書である．分子動力学シミュレーションソフトでは，その計算時間の大部分を占める粒子間の相互作用計算には，一般に Particle Mesh Ewald(PME) 法や Fast Multipole 法のように，遠くの粒子との相互作用を効率良く計算する方法が用いられることが多い．GENESIS で採用されている PME 法では粒子間相互作用の成分を距離に応じて急速に減衰する近距離成分と遠距離成分に分けて計算しており，本文書で議論するのは前者の計算速度である．現在 GENESIS で採用されている近距離相互作用計算カーネルでは，テーブルから参照した値を線形補間して計算を行っている．この方式は，カーネル内での演算量を劇的に減らせる一方，大きなテーブルへの間接アクセスがボトルネックになっている．A64FX は，単精度で 16 並列の SIMD ユニットを持つが，間接アクセスのスループットは理想的な場合でも 1 サイクル 2 語である．したがって，同一キャッシュライン上に複数語ある場合を除いては 16 語をロードするのに 8 サイクルかかることになる．さらに，データが L1/L2 キャッシュにのっていない場合には大きなペナルティがある (表 2.1 のメモリレイテンシを参照) ため，参照範囲が広い間接アクセスが大量に発生するような実装は好ましいとは言えない．したがって，本文書では大きなテーブルへの間接アドレスアクセスを排除するため，SIMD レジスタ数本にのる大きさのテーブルを用いた区間多項式近似を用いた近距離相互作用計算カーネルの最適化を行った結果を報告する．区間多項式近似カーネルを用いた場合，2018 年 9 月 13 日時点での GENESIS の Nonb15F カーネルの 1 ステップあたりの実機での計算時間 12.2 ms に対して，理研シミュレータにおいて 7.87 ms(1.55 倍) 相当の計算速度を達成した．精度については，GENESIS においてテーブル密度を 20 とした場合と同程度の精度を保っている．現在，コンパイラのバグや不備，シミュレータと実機の差異によって，性能が低く見積もられていると予測される部分が有り，実機ではさらに 1 割程度性能が改善する見込みがあると考えている．

表 1　Nonbond_June1 と多項式近似カーネルの比較．ネイバーサーチのマージンを 1.5 Å と想定．

カーネル名	Nonbond_June1	区間多項式近似カーネル
測定方法	実機	理研シミュレータ
コアあたり単位時間あたりの相互作用数	56.7 M 相互作用/s	87.94 M 相互作用/s
1 チップ 48 コア換算の 1 ステップの実行時間	12.2 ms	7.87 ms
倍率	1.00	1.55

目次

第 1 章

はじめに

1.1　この文書の目的

この文書では，ポスト京で採用される A64FX プロセッサ向けに，理研杉田チームで開発されている分子動力学シミュレーションパッケージ「GENESIS」の短距離相互作用の計算カーネルを高速化する手法を検討する．この章では，カーネルの高速化のために必要なプロセッサやシミュレータに関する事前知識について触れる．次章では，現在の GENESIS の短距離力カーネルの概要や実装，ベンチマーク結果を紹介し，4 章では本文書で提案する区間多項式近似手法の概要，最適化手順，ベンチマークの条件，その結果について述べる．

第 2 章

測定環境など

2.1 A64FX

A64FX はポスト「京」向けに富士通が開発を進めているプロセッサである．A64FX のスペックを表 2.1 に示す．アーキテクチャに ARMv8-A を採用し，ARM Scalable Vector Extension (SVE) の SIMD 命令が使用できる．4 つの Core Memory Group(CMG) からなり，各 CMG は 12 個の演算コアと 1 つのアシスタントコアで構成されている．メモリは 4 つの High Bandwidth Memory 2(HBM2) があり，各 CMG に一つずつつながっている．そのため，各 CMG からちがう CMG のメモリへアクセスしようとするとアクセス速度が変わる．全体での最大バンド幅は 1 TB/s 程度である．512 bit 幅の SIMD 命令が実行でき，理論ピーク性能は倍精度で 2.7 TFlops，単精度ではその倍である．メモリ階層は L2 までの階層キャッシュで，各キャッシュやメモリへのアクセスレイテンシなどは表の通りである．

以降では，分子動力学シミュレーションコードを最適化するにあたって重要な特徴について述べる．命令はアウトオブオーダー (OoO) 実行される．加算や乗算，Multiply and Add (MAD) 命令のレイテンシは 9 と若干大きく，分子動力学シミュレーションなどのレイテンシネックになりやすいアプリケーションではこの長いレイテンシをうまく隠蔽する必要があるが，Simulataneous Multi Thread (SMT) などの演算レイテンシを隠蔽するような機能はハードウェア的には OoO ぐらいしか採用されていない．そのため，演算順序に依存性がある場合，ソフトウェアパイプライニング (SWPL) 段数がレイテンシの数を上回らなければ，基本的に理論上到達可能な速度になることはない．演算同士の順序依存性を減らしつつ，レジスタの使用数を抑えながら実装することで，SWPL の段数を増やしてレイテンシが隠蔽される様にする必要がある．レイテンシが 9 の命令がある場合，Arithmetic Logic Unit(ALU) が 2 つあるので SWPL の段数が少なくとも 18 以上でループ長がループ開始部分の SWPL が効いていない部分が隠蔽される程度長くないと理想的な性能は得られない．これらに関してはロードストアの回数は増えるが，ループ分割を行うなどの対策が有効である．SWPL を有効にするためには，最内側ループにある程度の回転数が必要となるため，分子動力学の様に，ボトルネック部分に 2 重ループがでてくるアプリケーションでは，可能であれば最内側ループを SIMD 化するのは避けるべきであると考えられる．

表 2.1　A64FX の仕様およびコンパイラのバージョン等.

項目	説明
コア数 / CMG	12
CMG 数	4
ALU / コア	2
SIMD 幅	512 bit
動作周波数	1.8 GHz
理論ピーク性能	倍精度 2.7 TFlops
SIMD レジスタ数 / コア	32 本
メモリ (容量，ピークバンド幅)	4 × HBM2(32 GB, 256 GB/s)
メモリ階層 (レイテンシ)	L1 (5)
	L2 (42)
	HBM (CMG 内 255 / CMG 外 420)
コンパイラ	FCCpx (FCC) 1.2.0 20180907 simulating gcc version 4.1.2

2.2　理研シミュレータ

この節では，カーネルの最適化を行う際に使用した A64FX の理研シミュレータと実機との差異，使用したツールについて述べる．理研シミュレータは，富士通側で使用している A64FX のシミュレータを理研側では使用できないために理研側で開発されている A64FX シミュレータである．Gem5 を利用して作られている．現状，富士通のコンパイラから吐かれたアセンブラを利用して，.axf ファイルを作成し，シミュレーションを行う．動作は速いが実際の実行時間を取ることができない atomic というモードと，実行サイクル数から実機での実行時間を予測できる o3 というモードがある．以降では，基本的に o3 モードを利用した際の話を述べる．

2.2.1　実機とシミュレータの差異

Arm SVE 命令は，3 オペランドの命令である．そのため，MAD 命令を実行する際には，一つのオペランドが入力と出力の両方を担当することになる．そのため，各オペランドの値全てを取っておく必要がある場合，あらかじめ別のレジスタに値を移すことが必要となる場合がある．このときに，movprfx という命令が吐かれる．A64FX の実機ではこの movprfx と MAD 命令は一つの命令としてデコード・実行されるため movprfx のレイテンシを気にする必要はない．しかしながら，理研シミュレータでは実装の関係で，movprfx と MAD 命令は別々にデコードされ，movprfx も 1 サイクルのレイテンシがある命令として実行される．そのため，movprfx が頻繁に吐かれるようなコードの場合，MAD 命令は実質レイテンシが 1 サイクル増えることとなる

し，movprfx の分も OoO 資源などが割かれることになる．movprfx 命令はインアクティブなプレディケートを 0 で埋める命令の際にも必ず発効される．SWPL の段数が十分に大きいループでは，すべての命令は実質 1 サイクルのレイテンシで実行される．理研シミュレータで時間測定を行う場合，本来は隠蔽されるはずの movprfx 命令の割合が大きくループの性能評価に影響を与える可能性があるので注意が必要である．

2.2.2　シミュレーション結果の可視化

本文書で行った最適化では，理研シミュレータで行われたシミュレーションの結果をサイクルレベルでどのようにデコードから実行まで行われているかを可視化するために，フリーウェアの Gem5 の実行結果可視化ソフト Konata [1] を利用した．本文書で述べている演算のレイテンシなどは，シミュレーション結果を可視化した値 (つまり，理研シミュレータで設定されている値) を用いている場合が多い．

第 3 章

GENESIS

この章では，GENESIS 向け短距離力最適化カーネルを作成するに当たって必要な現状の GENESIS の実装やテスト問題に関する情報を示す．

3.1 短距離相互作用

GENESIS の短距離相互作用は主に 2 つの要素に分けられる．一つは，スイッチング関数がかかった Lennad-Jones 相互作用である．もう一つは，Ewald 法を用いて静電相互作用を計算するときの短距離成分である．本報告書で使用したカットオフ距離 r_c，スイッチング開始距離 r_s，Ewald 法のパラメータ α の設定値を表 3.1 に示す．

3.1.1 スイッチング関数を乗じた Lennard-Jones 相互作用

GENESIS で採用されている charmm の Lennard-Jones ポテンシャル $U_{\mathrm{LJSW}}(r)$ は，一般に用いられる 12-6 型の Lennard-Jones ポテンシャル U_{LJ} とスイッチング関数 $S(r)$ の積で書けて，

$$U_{\mathrm{LJSW}}(r) = U_{\mathrm{LJ}}(r)S(r). \tag{3.1.1}$$

ただし，

$$U_{\mathrm{LJ}}(r) = 4\epsilon\left\{\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right\}, \tag{3.1.2}$$

表 3.1　短距離力計算時のパラメータ．

r_c	12 Å
r_s	10 Å
α	0.34

$$S(r) = \begin{cases} 1 & (r < r_s) \\ \dfrac{(r_c^2 - r^2)^2(r_c^2 + 2r^2 - 3r_s^2)}{(r_c^2 - r_s^2)^3} & (r_s \leq r < r_c) \\ 0 & (r_c \leq r) \end{cases} \tag{3.1.3}$$

である．ここで，σ と ϵ は粒子の直径とポテンシャルの深さを表すパラメータであり，原子種ごとにその値が決まっている．違う種類の原子同士が相互作用をする場合は，原子 i と原子 j のそれぞれの σ と ϵ を用いて，Lorentz-Berthelot 則によって以下の様に決まる．

$$\sigma = \frac{\sigma_i + \sigma_j}{2}, \tag{3.1.4}$$

$$\epsilon = \sqrt{\epsilon_i \epsilon_j}. \tag{3.1.5}$$

スイッチング関数を乗じた LJ 相互作用による力 $\boldsymbol{F}_{\mathrm{LJSW}}$ は，

$$\boldsymbol{F}_{\mathrm{LJSW}}(r) = -\frac{dU_{\mathrm{LJSW}}}{d\boldsymbol{r}} = \begin{cases} \boldsymbol{F}_{\mathrm{LJ}} & (r < r_s) \\ \boldsymbol{F}_{\mathrm{LJ}} S(r) - U_{\mathrm{LJ}}(r) \dfrac{dS(r)}{d\boldsymbol{r}} & (r_s \leq r < r_c) \\ 0 & (r_c \leq r) \end{cases} \tag{3.1.6}$$

となる．ただし，$\boldsymbol{F}_{\mathrm{LJ}}(r)$ と $\frac{dS(r)}{d\boldsymbol{r}}$ は，

$$\boldsymbol{F}_{\mathrm{LJ}} = -\frac{dU_{\mathrm{LJ}}}{d\boldsymbol{r}} = 24\epsilon \left\{ 2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right\} \frac{\boldsymbol{r}}{r^2}, \tag{3.1.7}$$

$$\frac{dS(r)}{d\boldsymbol{r}} = \begin{cases} 0 & (r < r_s) \\ \dfrac{12(r_c^2 - r^2)(r_s^2 - r^2)}{(r_c^2 - r_s^2)^3} \boldsymbol{r} & (r_s \leq r < r_c) \\ 0 & (r_c \leq r) \end{cases} \tag{3.1.8}$$

である．

3.1.2　Ewald 法を用いた静電相互作用の短距離成分

粒子 i と j の電荷をそれぞれ Q，q としたとき，粒子間の静電ポテンシャルは以下の様に表される．

$$U_{\mathrm{CLMB}}(r) = \frac{Qq}{4\pi\epsilon_0 r}. \tag{3.1.9}$$

ここで，ϵ_0 は真空の誘電率である．静電ポテンシャルは，LJ ポテンシャルと違い r に対する減衰が遅い関数であり，周期境界条件下ではシミュレーションセルの周りに存在するイメージセルの寄与を考慮する必要があるが，本報告書では長距離力を考慮せず短距離力についてのみ議論をするため，粒子 i と j の間の距離は，シミュレーションセル及びイメージセルの中の j 粒子の中で i 粒子と最も近いものとの距離とする．

周期境界条件で静電相互作用を高速に計算できる Ewald 法では，この相互作用を 0 への収束の早い近距離成分と収束の遅い遠距離成分と自己相関項に分解される．

$$U_{\mathrm{CLMB}}(r) = U_{\mathrm{short}}(r) + U_{\mathrm{long}}(r) = \mathrm{erfc}(\alpha r) U_{\mathrm{CLMB}}(r) + \mathrm{erf}(\alpha r) U_{\mathrm{CLMB}}(r). \tag{3.1.10}$$

ここで，α は Ewald 法におけるパラメータであり，計算の精度と速度を考慮して決定される．実際のプログラムにおいては，α は実行時にユーザが決定する．また，erf(αr) と erfc(αr) は誤差関数と補誤差関数と呼ばれる．

$$\mathrm{erfc}(x) = 1 - \mathrm{erf}(x) = \int_0^x e^{-t^2} dt. \tag{3.1.11}$$

したがって，近距離成分のポテンシャル U_{short} と力 $\boldsymbol{F}_{\mathrm{short}}$ は，補誤差関数と指数関数を用いて以下のように表される．

$$U_{\mathrm{short}}(r) = \frac{Qq}{4\pi\epsilon_0} \frac{\mathrm{erfc}(\alpha r)}{r}, \tag{3.1.12}$$

$$\boldsymbol{F}_{\mathrm{short}}(r) = -\frac{dU_{\mathrm{short}}}{d\boldsymbol{r}} = \frac{Qq}{4\pi\epsilon_0} \left\{ \frac{\mathrm{erfc}(\alpha r)}{r^2} + \frac{2\alpha}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r^2)}{r} \right\} \frac{\boldsymbol{r}}{r}. \tag{3.1.13}$$

3.2　短距離力カーネルの実装

現状の GENESIS では，事前に上記 2 つの相互作用 (LJ 相互作用を $1/r$ の引力項と斥力項に分けるので実際には 3 つの項) を計算して 2 粒子間の距離 r の関数としてテーブル化する．カーネルの最内ループでは，計算された r の近傍点から線形補間して計算を行っている．テーブルの大きさはテーブルの密度の設定にもよるが数千点から数万点程度とっており，LJ 相互作用用に 2 つ，静電相互作用用に 1 つのテーブルを保持している．これらのテーブルは，L1 キャッシュに載りきる大きさではなく，カーネルの実行時間が頻繁に起こるテーブルのメモリもしくはキャッシュへのアクセス速度で律速すると予想される．実際，Kernel_June1 では，演算待ちよりもメモリアクセス待ちの時間が大半を占めている (図 3.1).

3.3　Kernel_June1 のシミュレータおよび実機での測定状況

GENESIS のベンチマークには，Apoa1 と呼ばれるテスト系が用いられている．この系は，シミュレーションセルのサイズが 108.861198 Å×108.861198 Å×77.758003 Å で，そのなかに 92227 個の原子が入っている．表 3.1 に示した通りカットオフ半径が 12 Å で系の密度が一様だと仮定すると，カットオフ半径の球体内には約 724 個の原子が存在することになる．よって，1 つの粒子は，平均して中心の粒子を除く 723 個の粒子と相互作用することになると考えられる．Apoa1 を入力に GENESIS v1.3.0 を実際に実行したときには，1 ステップあたりの作用反作用を考慮した相互作用数の数は，33.2 M 相互作用であった．上記の一様な密度の仮定でも，$92227 \times 723/2 = 33.4$ M 相互作用であり，概ね一致している．実行時の相互作用数が若干少ないのは，一様密度の仮定では分子内の隣接原子を相互作用対象から取り除いていないためであると考えている．9 月 13 日時点の GENESIS の実機での計測では，Nonbond_June1 カーネルでは 1 ステップあたり 12.2 ms かかっている．この実行時間はシミュレータ比 1.08 倍である．実

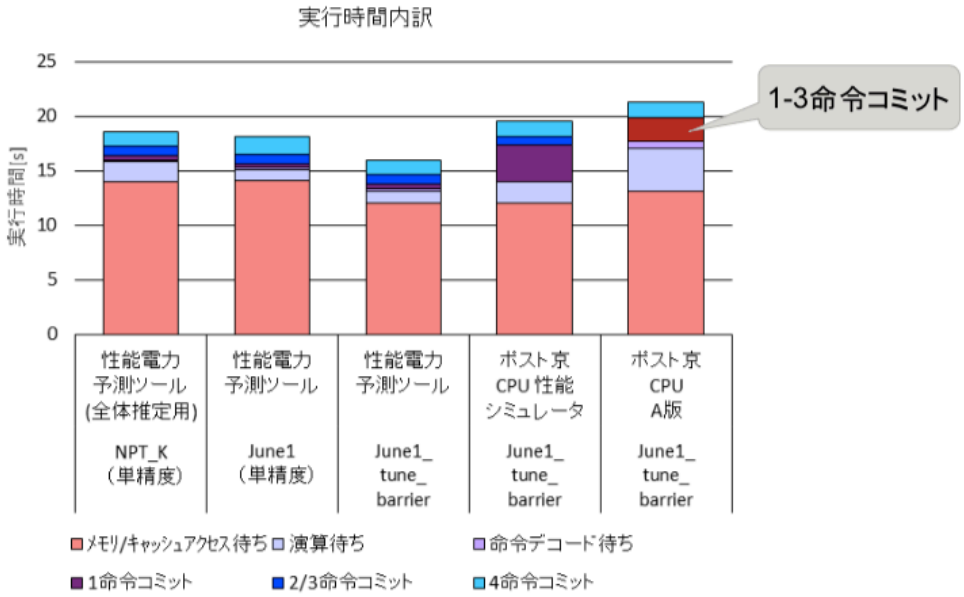

図 3.1　カーネル WG で提示された GENESIS の Nonb15F カーネルの予測ツール，シミュレータ，実機での実行時間内訳．ピンク色のメモリ/キャッシュアクセス待ちが演算時間の大半を占める．実機では演算待ちも少し多くなっている．

機測定は 48 コア実行想定の 1 ステップあたりの時間であるので，1 コアあたりの計算性能は，33.2 M 相互作用 / (12.2 ms × 48 core) = 56.7 M 相互作用/s となる．本報告書では，GENESIS の短距離力カーネルの現状の性能値を，1 コアあたり 56.7 M 相互作用/s(作用反作用を考慮)，もしくは Apoa1 入力時の 1 チップ換算の 1 ステップの実行時間が 12.2 ms/step として，最適化カーネルとの比較を行う．

第 4 章

区間多項式近似カーネル

4.1 MD シミュレーションパッケージでの短距離力カーネルの実装

GENESIS の実装のようなテーブルを利用した相互作用の計算方法は，さまざまなポテンシャル形に対応しやすいなどのメリットもあり様々なパッケージでも用いられていることはあるが，静電相互作用に限って言えばよく用いられている分子動力学シミュレーションパッケージでは素直に，もしくは erfc(x) や exp(x) の項を近似するなど工夫して近距離相互作用を SIMD 化した実装が一般的である．512-bit SIMD のカーネルを持つ MD シミュレーションパッケージで最も計算速度が速いものの一つと思われる GROMACS を例にとっても，ミニマックス多項式近似を用いて，$\alpha^2 r^2$ を引数に

$$f(\alpha^2 r^2) = \frac{2}{\sqrt{\pi}} \frac{\exp(-\alpha^2 r^2)}{\alpha^2 r^2} - \frac{\mathrm{erf}(\alpha r)}{\alpha^3 r^3} \tag{4.1.1}$$

を高速に求める関数を実装している．しかしながら，この近似は次数が多く途中で逆数の計算も一度入るため，依然として計算量は少なくない．さらに計算量を減らすための手法として，本報告書ではさらに効率的な近似方法として区間多項式近似を行う手法を提案する．

4.2 512 bit SIMD に適応した多項式近似の方法

ARM SVE 命令には 512 bit のレジスタから，インデックスを用いてレジスタの要素を並び替える tbl 命令が存在する．本手法ではその tbl 命令に着目し，計算量の多い式を区間多項式で近似し，その係数をレジスタ上に保持しながら計算を行うことによって高速化を図る．

一定の距離で十分に小さくなる短距離力では，一般にカットオフ距離で計算を打ち切る手法を用いることが多い．そのため，その相互作用を計算する領域を m 個に分割し，それぞれの区間で n 次多項式で近似することを考える．ここで，単精度で計算することを考えると，$m = 16$ のときには $n + 1$ 本の 512 bit レジスタに区間多項式の係数全てをレジスタに載せることができる．

ここで問題となるのが，関数をどう多項式近似するかと計算領域の分割法である．まず，前者

について述べる．Ewald 法の静電相互作用短距離成分は，計算領域内での変化が 10 桁以上の広範な桁にわたる関数である．特に，粒子間の距離 r が小さいときには大きく変化する．そのため，区間によっては近似の次数 n を上げなくては十分な精度が得られない．もう一度静電相互作用の短距離成分の式を示す．

$$\boldsymbol{F}_{ij}(r) = \frac{Qq}{4\pi\epsilon_0}\left\{\frac{\mathrm{erfc}(\alpha r)}{r^2} + \frac{2\alpha}{\sqrt{\pi}}\frac{\exp(-\alpha^2 r^2)}{r}\right\}\frac{\boldsymbol{r}}{r} \tag{4.2.1}$$

これを $R = \alpha r$ の関数だと考えると，

$$\boldsymbol{F}_{ij}(R) = \frac{Qq\alpha^3}{4\pi\epsilon_0}\left\{\frac{\mathrm{erfc}(R)}{R^2} + \frac{2}{\sqrt{\pi}}\frac{\exp(-R^2)}{R}\right\}\frac{\boldsymbol{r}}{R} \tag{4.2.2}$$

となる．このようにすることで，係数を除けば，プログラム実行時に決定する α に依存しない関数となる．つまり，

$$f(R) = \frac{\mathrm{erfc}(R)}{R^3} + \frac{2}{\sqrt{\pi}}\frac{\exp(-R^2)}{R^2} \tag{4.2.3}$$

を考えれば良い．この関数は，$R = 0$ 付近で発散し，R が小さい時に大きく値が変化する関数である．実際のプログラムの中では，LJ 相互作用はナイーブに計算することを考えると，力の計算を行う際に必ず $1/r$(もしくは $1/r^2$) を計算している．そのため，この関数に R の累乗をかけた関数から目的の値を得ることは乗算数回で得ることができる．さらに，目的の関数自体の N 乗根を近似多項式から得て，その値を N 乗すれば，演算回数を大きく増やさずに近似する関数型を変えることができる．図 4.1 に検討した様々な関数形を示す．ここで，$f(x)$ から $f(x) * x^4$(紫，緑，水色，橙色，黄色) までを見ると，$f(x) * x^3$(橙色) が最も r が小さい時に値が変動しないことがわかる．さらに，$\sqrt{f(x) * x^3}$ は $f(x) * x^3$ に比べて r が大きい時にも大きく値が変化していないことがわかる．本手法では，R が小さい時と大きい時共に変化が少なく，緩やかに減少していく $\sqrt{f(R)R^3}$ を近似対象とした．この関数を使うと，$F_{ij}(R)$ の係数部分に含まれる α^3 と R^3 に含まれる α^3 が打ち消しあい，近似値を 2 乗した値に $1/r^3$ を乗じると $f(x)$ を得ることができる．実際の計算では，αr よりも $(\alpha r)^2$ の方が演算数が少なく求まるため，$(\alpha r)^2$ から近似を行う．

次に，近似する領域の分割方法について述べる．近似する区間の分割方法が決まると，ある程度の精度を保つために近似の次数が決まる．

本カーネルでは，A64FX の SIMD 幅に合わせて 16 点の区間に区切ることを考える．静電相互作用の短距離成分は，r が大きくなるにつれて値が小さくなるため，2 粒子間の距離が近いほど近似の精度を上げる必要がある．しかしながら，複雑な分割区間の設定は計算負荷上好ましくない．本手法では，$(\alpha r)^2$ の i 番目の区間を $[2^{i-7}, 2^{i-6})$ として多項式近似を行った (近似可能な範囲を $0.0078125 \le (\alpha r)^2 < 512$，および $0.08838834764/\alpha \le r < 22.627416998/\alpha$ とするため)．このような区間の設定を行うと，最初の区間の視点を決めれば R の指数部の該当部分 $\log_2 m$ bit を取り出すことで，どの区間の近似式を計算するのか i を決定できる．また，倍精度で計算した静電相互作用の短距離成分との絶対誤差が GENESIS の線形補間を用いた計算とほぼ同程度もしくはそれ以下にになるように近似の次数 n を 5 とした．

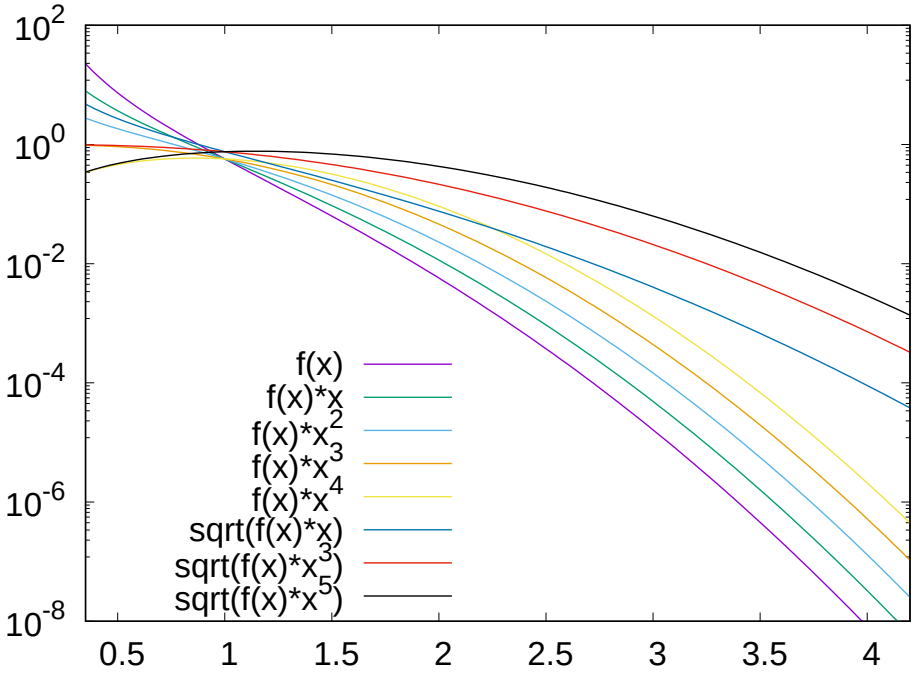

図 4.1　静電相互作用の短距離成分の関数形の検討.

$n = 5$，$m = 16$ の時の $\sqrt{f(R)R^3}$ を近似して求めた静電相互作用の短距離成分と Kernel_June1 で採用されている線形補間から求めた値 (GENESIS でのパラメータ，table density は 20) を比較する (図 4.2)．図 4.2 は倍精度でナイーブに静電相互作用を計算した場合との誤差を示しており，上段が絶対誤差 (差の絶対値)，下段が相対誤差 (絶対誤差を静電相互作用の値で割ったもの) である．区分多項式 (e4bit,m0bit,5th) は概ね全ての領域で線形補完 (genesis) を下回っており精度に問題がないことがわかる．r の小さい区間で精度が落ちている部分もあるが，相対誤差としては 10^{-7} 程度であり，単精度の範囲では問題無いと考える．また，原子同士が 1 Å 程度まで近づくことはまれである．

区間多項式近似の最適な係数を求める際には Sollya [2] というプログラムを用いた．本手法で用いた係数を出力するためのスクリプトは付録の Algorithm3 に示す．Sollya を用いて求めた 6×16 個 (n 次近似多項式では $n + 1$ 個の定数が必要となる) の定数を SIMD レジスタに載せて演算を行うことを考える．計算手順は Algorithm1 の様になる．Sollya から求めたテーブルは t に入っている．実際には 6 本の SIMD レジスタにそれぞれ $t[0]$ から $t[5]$ までが載っている．まず，$x = (\alpha r)^2$ を計算する．次に，x を 23 ビット右シフトしてから下位 4 ビットを取得することで，x が m 個の区間の何番目に含まれているかを計算する (i)．この際，浮動小数点数では，指数部に下駄履き表現を使っているので，定数を足している．i 番目の区間の始まり x_m は i から求めることができる (ここでは $x_m = 2^{i-7}$) ので $dh = x - x_m$ を求めることができる．あとは，dh とテーブル t から n 次多項式の計算を行うだけである．n 次多項式の k 番目の係数 $t[k][i]$ は，ARM SVE や Intel AVX-512 などでは，それぞれ tbl や permutexvar などの SIMD 命令を使え

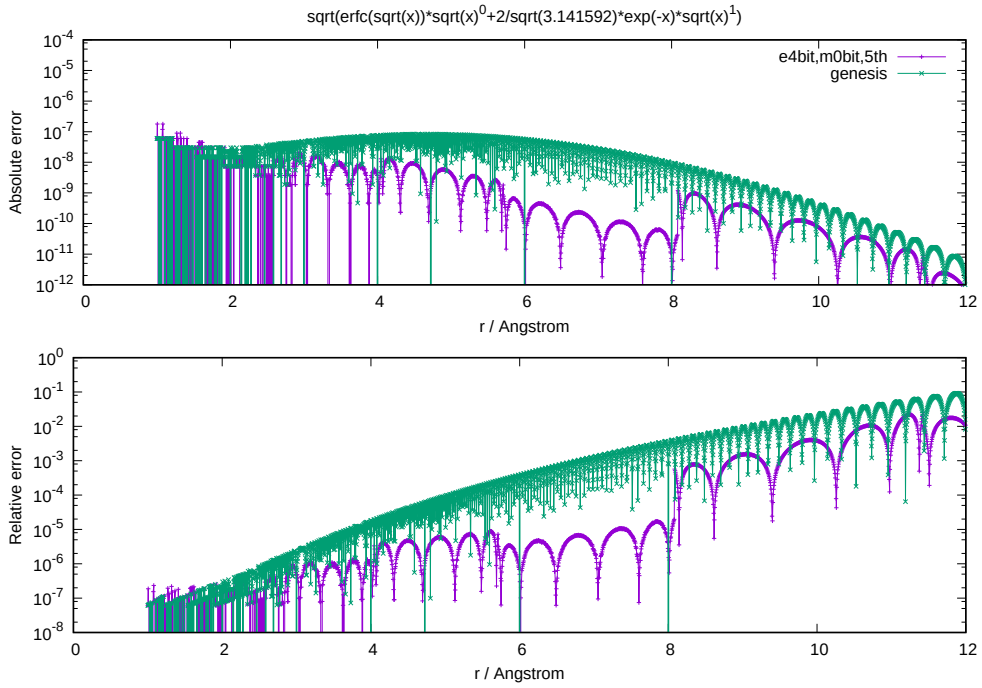

図 4.2　区分多項式近似と線形補間の精度比較．比較対象は倍精度でナイーブに F(R) を計算した値．

ば，SIMD の各要素の x から求めたそれぞれの i をインデックスとして，並べ替えた値を得ることができるため，SIMD 化は容易である．静電相互作用の近距離成分の演算量を減らせる一方，テーブルを $N+1$ 本のレジスタで保持しなくてはならないため，A64FX では SWPL の段数を上げることが難しいことが欠点である．

Algorithm 1 区間多項式近似の計算方法

Require:

t[6][16] : 16×6 のテーブル

(to_float) や (to_int) は 32bit 整数や 32bit 浮動小数の各ビット変化させずに float や int に変換するキャスト

function PolynomialApproximation(r, α)

1 $x \leftarrow (\alpha r)^2$

2 $i \leftarrow$ (((to_int)R+0x44000000) >> 23) & 0x0f

3 $dh \leftarrow x-$(to_float)(i << 23 − 0x44000000)

4 **return** $t[0][i] + dh * (t[1][i] + dh * (t[2][i] + dh * (t[3][i] + dh * (t[4][i] + dh * t[5][i]))))$

end function

4.3　最適化カーネルでの前提条件

A64FX 上で理想的な速度を達成するには，SWPL の効果を活用するために最内側カーネルの回転数がある程度大きいことが求められる．そのため，本カーネルでは相互作用計算の外側ループ (相互作用を受ける粒子のループ，i ループ) を SIMD 化を行うこととした．内側ループ (相互作用を与える粒子のループ，j のループ) では，メモリ上に連続な SIMD 幅 (16 個) 分の粒子に共通のネイバーリストを作成し，j 粒子の情報を SIMD レジスタ内で放送して，相互作用の計算を行っている．理想的な性能に近づけるには最内側ループの SWPL の段数を上げる必要があるため，最内側ループはそれぞれ，r^2 計算ループ，$1/r$ 計算ループ，LJ 相互作用計算ループ，スイッチング関数計算ループ，区間多項式近似の区間計算ループ，多項式計算ループ，静電相互作用計算ループ，アキュムレート及び反作用計算ループの 8 つのループに分割を行った．

4.3.1　入力データ

シミュレータ実行時の入力データは以下の様に作成した．

1. GENESIS(v1.3.0) を実際に実行し，Apoa1 の力の計算を実行．
2. その際に生成された実際のセル情報および粒子データとセル-セル間のペアリストをファイルに出力．
3. 必要なサイズに切り出して (セル-セルペアリスト情報から相互作用計算を行うセルとその周りのセルのみを取り出す) 入力ファイルを生成．

実際の入力データとしては，相互作用を受けるセルの数を 2 つとし，2 つのセルおよび相互作用を及ぼす周辺のセルを取り出した．1 つのセルに対しては，$5^3 = 125$ のセル (自セル含む) が相互作用を及ぼす．

4.3.2　入力データの性質

セル内に含まれる粒子数の累積分布関数を図 4.3 に示す．セル内の粒子数は $0 < N < 80$ の範囲に存在し，そのほとんどが 50 前後に分布していることがわかる．そのため，極端に小さいセルを大量に含む部分でカーネルの性能を測定しない限りは妥当な見積であると考える．

4.3.3　粒子データの並べ方

テストコードでは，長めの 1 次元配列を確保し，セル番号の小さい方から順番に，入力データから得た粒子のデータを連続に詰めている．その際，粒子データは 128 bit アライメントされており，座標の 3 つの浮動小数点数 (x, y, z) と下位 24 bit がその粒子が所属する分子の ID を，上位 8 bit が原子の種類を表す 32bit 符号なし整数 (w) が入っている．これは，相互作用計算の時

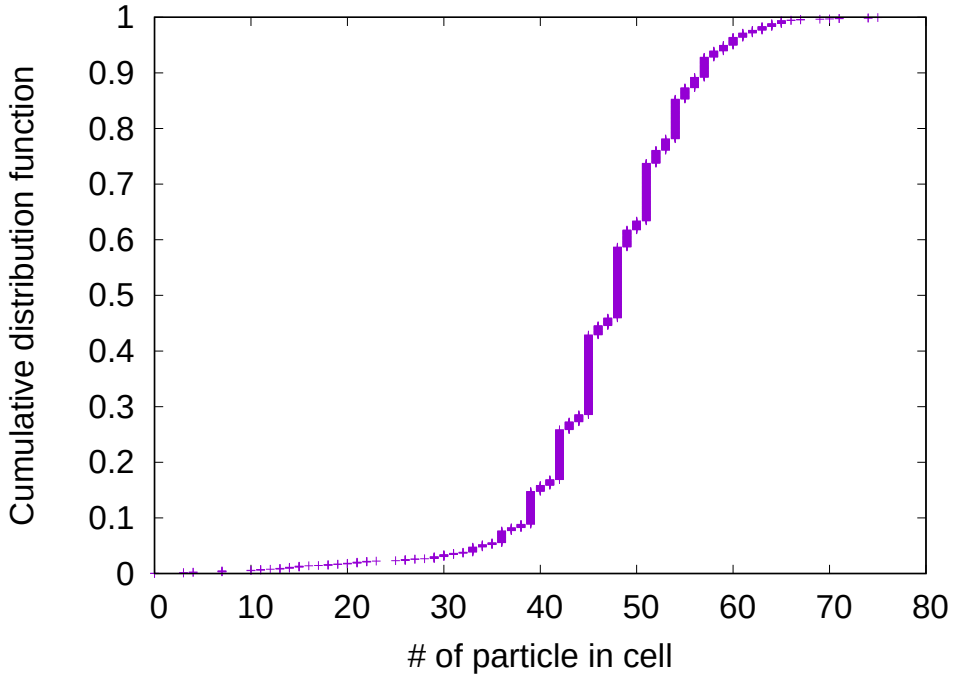

図 4.3　セル内に含まれる粒子数の累積分布関数.

に i 粒子のデータを SVE のストラクチャーロード命令 (ST4) を用いて x，y，z，w それぞれの方向に並べ替えるためである．同様に，力の配列も x,y,z,w の 4 要素ベクトルが並んでいる形になっている．GENESIS 本体の配列構造とは異なるが，カーネル計算時の座標配列は作業用の一時的な配列であるため，比較的簡単に対応可能だと考えている．

また，座標データとは別に，各粒子の電荷 q は連続な一次元配列上に，LJ パラメータ σ，ϵ は，w の上位 4 bit の情報からアクセスできる 2 次元のテーブルに納められている．

4.3.4　排除リストの取り扱い

GENESIS では，相互作用リストを作成する際に分子内の近距離原子に対して排除リストを反映させている．SIMD 環境で排除リストを考慮しながら計算を行うことは実装上難しいため，本カーネルでは同一分子内の原子同士では相互作用の計算はマスクして行わないこととした．そのため，分子内の相互作用に関しては別途計算を行うことが必要となるので注意されたい．本文書では分子内相互作用の計算速度には触れない．同一分子かどうかの判定には w の下位 24 bit を用いている．

4.4　シミュレータを用いた最適化

ここでは例として 2 粒子間の距離を計算するループについて考えてみる (Algorithm 2)．SWPL が十分に効いていれば，二つの ALU はそれぞれ自分が担当する別個のループを処理してるともみなせるので，ここでは一つの ALU に演算を詰めていくことを考える．dx と dy，dz の演算はそれぞれに依存関係がないため，直前の命令が終わった後，すぐに演算を実行することができる．r^2 の最初の MAD 命令は dx と dy の演算が終わるまで待つ必要がある．同様にそれ以降の MAD 命令は全て依存関係のある演算なので，直前の演算が終わるまで実行を待つ必要がある．また，MAD 命令の前には movprfx 命令が発効されている．j 粒子のロード命令のレイテンシは 6，ストアは命令の発行のみを行えば良いのでレイテンシは 1 とする．j 粒子のロード命令の後には，SIMD レジスタへブロードキャストを行う dup 命令が発行される．この命令のレイテンシは 4 である．浮動小数演算命令のレイテンシは 9 なので，依存関係を考慮した場合の SWPL が効いていない場合の 1 回転あたりのサイクル数は 49 である．SWPL の段数が十分に大きいとき，全ての命令のレイテンシは隠蔽されるため，このループは発効される命令の数である 16 サイクルで実行でき，そのときの性能は 16 SIMD × 2 ALU × 1.8 GHz × 9 flops / 16 cycle = 32.4 GFlops となる．このループの実際の SWPL 段数とシミュレータの実行結果から得られた演算性能は，それぞれは 9 段と 21.29 GFlops であった．SWPL の段数は命令のレイテンシを隠蔽するのには十分であるが，実際にはロード・ストア時のアドレス計算などがあるため，このような演算数の少ない短いループの場合，この程度の差は許容範囲であると考える．9 段の SWPL のループでは，r^2 計算に関係ある命令が 151，それ以外の命令が 82 命令あった．$32.4 \times 151 / (151 + 82) = 21.0$ GFlops となり，概ね測定結果と一致する．分割した他ループについても同様の検討を行い，そのループでボトルネックを考慮した上では理想的に近い性能が出ていることを確認している．SWPL 非適用時に各ループの浮動小数演算が理想的に実行できた場合に各演算がどのような依存関係で実行され，何サイクルかかるのかは別資料を参照されたい．

4.5　各種測定

本節では，区間多項式近似を用いたカーネルについて，2 つのケースの性能について議論する．以下では，理研シミュレータを用いて，1 コアでの性能を測定しているものとする．シミュレータを用いる都合上，計算時間の都合で Apoa1 の全ての粒子の相互作用を計算することは難しいため，i 粒子 80 個分についてのみ相互作用を計算する．表 4.1 に計算条件を示す．コンパイルオプションには「-O3 -Kfast -Kocl -Kswp_strong -Knosch_pre_ra,nosch_post_ra」を用いた．

Algorithm 2 粒子間距離の計算のループ

Require:

x_i, y_i, z_i : i 粒子の x, y, z 座標

x_j, y_j, z_j : j 粒子の x, y, z 座標

function CalcSquareDistance($x_i, y_i, z_i, x_j, y_j, z_j$)

load x_j, y_j, z_j

broadcast x_j, y_j, z_j to SIMD register

$dx \leftarrow x_i - x_j$

$dy \leftarrow y_i - y_j$

$dz \leftarrow z_i - z_j$

$r^2 \leftarrow dx * dx$

$r^2 \leftarrow r^2 + dy * dy$

$r^2 \leftarrow r^2 + dz * dz$

store dx, dy, dz, r^2

end function

表 4.1　パフォーマンス測定における i 粒子の数と各 SIMD ループにおける j ループの長さ．

i 粒子数		80
j 粒子数	ループ 0	1294
	ループ 1	1244
	ループ 2	1641
	ループ 3	1577
	ループ 4	1388
	計	7144

4.5.1　N^2 計算の場合のパフォーマンス

まず，ARM SVE を用いて実装したナイーブな LJ 相互作用及び区間多項式近似を用いた静電相互作用計算カーネルが A64FX において理想的な場合にどの程度の性能を達成できているかを確認するために，カットオフや分子内相互作用計算を排除しない場合の計算性能について見ていく．ここでは，静電相互作用や LJ 相互作用の係数は定数とみなす．また，本来はネイバーリストを介して不連続にアクセスされる j 粒子は，メモリ上に連続に並んでいるものとする．また，ここでは周期境界条件や作用反作用は考慮しない．実際にはカットオフされたり，排除されたりする粒子についても相互作用計算を行うので，相互作用の値は実際のものとは異なっている場合があるが，ここでは純粋にカーネルの速度を測ることを目的とする．

1 ループ 16 粒子に対して，合計で 7144 個の j 粒子が存在するため，相互作用数は $16 \times 7144 =$

114304 である．このときの計算時間は 0.276 ms であったので，実効性能は 414.14 M 相互作用/s である．本カーネルの総浮動小数点演算命令の約半分が MAD 命令のため，コアあたりの到達可能なピーク性能は 115.2 ∗ 3/4 = 86.4 GFlops である．1 相互作用あたりの演算数を 73 とすると，N^2 計算カーネルの性能は，30.23 GFlops となり，ピーク性能の約 35 % の性能がでている．

4.5.2　SIMD 幅ネイバーリストを用いたベンチマーク系 (Apoa1) の場合のパフォーマンス

次に，実際の相互作用を計算する場合を考える．まず，ネイバーリストのマージンがない (カットオフ距離がネイバーリストのサーチ距離の) 場合に，カットオフや分子内原子の排除を行った際のプレディケートのアクティブな数ごとのカウントと割合を図 4.4 に示す．プレディケートのアクティブな数ごとのカウントは，SIMD 幅の全てのレーンがアクティブな 16 が最も高く，それ以外ではアクティブなレーンが少ないほど比率が高くなる傾向にある．プレディケートのアクティブ率が 3 の倍数付近でピークを持っている理由は，本ベンチマークで使用される系ではほとんどの分子が 3 原子から構成される水分子であるためだと考えられる．トータルのアクティブなプレディケートの数は 53473 で，これが実際に計算された相互作用の数にあたる．また，全体に占めるアクティブなプレディケートの割合は 47.85 % であった．したがって，本カーネルはこのベンチマークにおいて，最大でも前節で見た理想的な条件での性能の 47.58 %，つまり，197.05 M 相互作用/s までしか到達できない．また，本節の計算では反作用の足し込みが追加されているため，前節の場合に比べて計算量が増えていることも述べておく．

ネイバーリストのマージンを 0 Å から 3 Å まで変化させた場合の実行時間と実効性能を表 4.2 に示す．ネイバーサーチ距離が大きくなるにしたがって，余分な計算が増えるために実行時間が長くなっている．本カーネルでは，r^2 の計算を行った後に SIMD 幅の全てのレーンがインアクティブな時に，その際の j 粒子を相互作用計算から除外するような実装は行っていない．そのため，実行時間の増分はネイバーサーチ距離が大きくなった際のネイバーリストの長さに比例する．示したネイバーサーチ距離の範囲では，本カーネルを使用することにより，GENESIS のテーブル参照実装に比べて 2.20 倍から 1.32 倍の高速化が期待される．

理想的な性能からの乖離については，j 粒子のデータへのランダムアクセス化，相互作用パラメータの非定数化，周期境界条件の適用の順に影響が大きかった．j 粒子のデータへのランダムアクセス化については，プリフェッチ命令が有効な場合も考えられる．

最後に，3.3 で示した現行 GENESIS の実機での測定結果と本カーネルの比較を表 4.3 に示す．

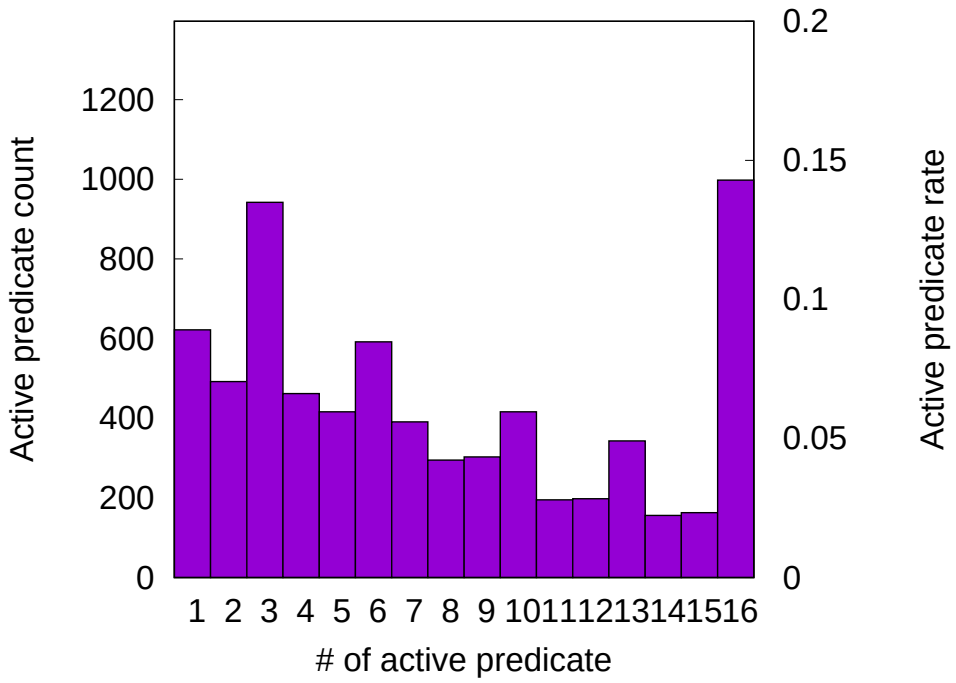

図 4.4　SIMD 幅中のアクティブなプレディケートの数及びその全体に占める割合.

表 4.2　ネイバーサーチ距離とカーネル実行時間，実効性能.

ネイバーサーチ距離 (Å)	実行時間 (ms)	実行性能 (M 相互作用/s)
0.0	0.42927	124.57
0.5	0.45797	116.76
1.0	0.51975	102.88
1.5	0.56768	94.20
2.0	0.60805	87.94
2.5	0.64565	82.82
3.0	0.71439	74.85

表 4.3　Nonbond_June1 と多項式近似カーネルの比較．ネイバーサーチのマージンを 1.5 Å と想定.

カーネル名	Nonbond_June1	区間多項式近似カーネル
測定方法	実機	理研シミュレータ
コアあたり単位時間あたりの相互作用数	56.7 M 相互作用/s	87.94 M 相互作用/s
1 チップ 48 コアでの 1 ステップの実行時間	12.2 ms	7.87 ms
倍率	×1.0	×1.55

第 5 章

おわりに

この文書では，分子動力学シミュレーションパッケージ「GENESIS」の LJ 相互作用および静電相互作用の短距離力計算カーネルを A64FX 向けに区間多項式近似を用いて最適化する手法について検討した．

現状の GENESIS Kernel_June1 はテーブルを用いて線形補間することで短距離力の計算を行っているため，性能のボトルネックはメモリおよびキャッシュへのアクセス待ちとなっている．これは，大きなテーブルを間接参照するために起こっている．そのため，短距離力の高速化にはメモリ上でのテーブルへの間接参照を排除し，間接アクセスを行わない高速な計算手法を開発する必要があった．

本報告書では，SIMD レジスタに収まる程度の大きさのテーブルを用いる区間多項式近似手法によるカーネルを作成し，最適化を行うことにより，実機および富士通シミュレータを用いて測定された Kernel_June1 の性能に対して，テーブルを用いた線形近似の既存手法と同等の精度を保ちつつ 1.55 倍の速度向上を達成できる見込みである．

さらなる計算速度の向上の可能性としては，現状計算時間の 20% 程度を占めている反作用を計算するループに関するコンパイラのバグがとれてさらなる最適化がかかること，カーネル全体のインラインアセンブラ化によるアドレス計算最適化などをすることなどが考えられる．

参考文献

[1] “Konata,” https://github.com/shioyadan/Konata.

[2] “Sollya,” http://sollya.gforge.inria.fr.

付録

5.1 Skylake Xeon(実機) と A64FX(理研シミュレータ) の比較

ARM SVE での実装を行う前に，多項式近似カーネルの性能評価として実装した区間多項式近似 (PPA) カーネルと，一般に使用される MD シミュレーションパッケージで最速のコードのひとつである GROMACS で用いられている方式の近距離力計算カーネルとの比較を行った．Skylake Xeon(仕様については表 5.1 を参照) 上で上記 2 つのカーネルを用いて計算を行った場合と，A64FX(理研シミュレータ) 上で PPA カーネルを実行した場合の比較を図 5.1，5.2 に示す．GROMACS に関しては，Skylake Xeon と A64FX のクロック周波数の比でノーマライズして A64FX にあわせたデータも載せている (GROMACS normalized)．この比較はテスト実装で行われており，本文書で解説している実際の実装での結果とは多少異なる可能性に注意されたい．具体的には，この比較で用いたカーネルでは，相互作用を及ぼす粒子 (j 粒子) が SIMD 化されていること，周期境界条件を適用しない理想的な N^2 の計算 (512 粒子) での計算であることが違いとしてあげられる．また，A64FX のカーネルは実際に用いられたカーネルほどは最適化されてはいない．A64FX では GROMACS 方式は実装されていないため省略する．GROMACS 方式の詳細については `http://manual.gromacs.org/documentation/current/doxygen/html-lib/simd__math_8h.xhtml` の pmeForceCorrectionSingleAcculacy の項を参照のこと．

5.2 使用した Sollya スクリプト

区間多項式近似の係数テーブルを作る際に用いた Sollya スクリプトを algorithm3 に示す．

表 5.1 比較に用いた Skylake Xeon の仕様．

Name	Xeon Gold 6140
Frequency	2.30 GHz
# of cores	18 (比較では 1 コアのみ使用)

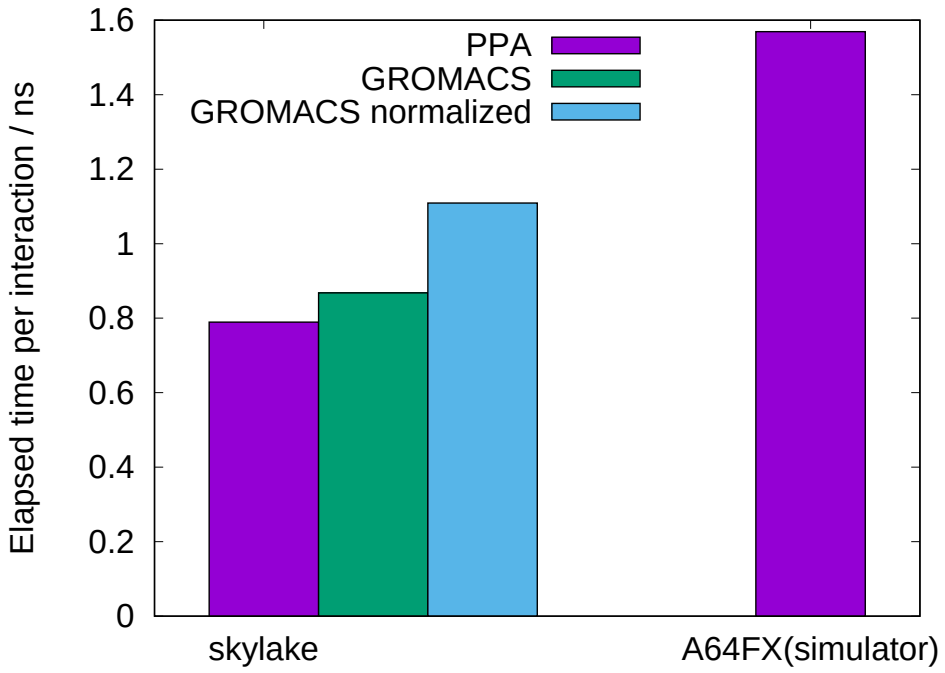

図 5.1　区間多項式近似 (PPA) と GROMACS で採用されている方式の Skylake Xeon と A64FX での計算時間の比較.

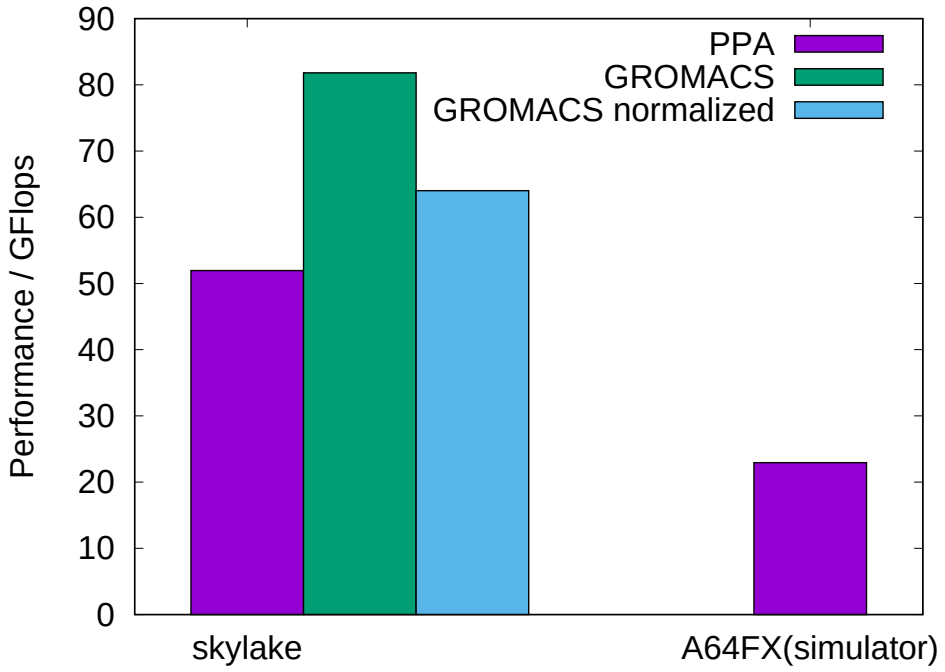

図 5.2　区間多項式近似 (PPA) と GROMACS で採用されている方式の Skylake Xeon と A64FX での計算速度の比較.

Algorithm 3 区間多項式近似カーネル用の係数を出力する Sollya スクリプト

```
prec = 60;
f = sqrt(erfc(sqrt(x))/sqrt(x)^3+2/sqrt(3.141592)*exp(-x)/sqrt(x)^2);
for n from 0 to 2^4-1 do {
 p = remez(f(x+((0.0078125)*2^n)), 5, [0;((0.0078125)*2^(n+1))-((0.0078125)*2^n)]);
 write("{ ");
 for i from 0 to 5 do {
  write(coeff(p,i)," , ");
 };
 write(" },\n");
};
```

5.3 最適化による演算順序の変更によるコンパイラのバグ

2018 年 9 月以前の富士通コンパイラで，本カーネルを最適化レベル O1 以上でコンパイルを行うと，SIMD の比較命令 cmpXX(XX は ge や lt) が適切な順番で配置されないコンパイラのバグに起因して計算結果が変わってしまう．該当箇所のソースコードをアルゴリズム 4 に示す．このコンパイラの最適化時のバグに関しては Redmine の Flagship2020-A にてチケット (#577) を切って富士通側に報告を行い，着手済みで未だチケットとしては解決にははなっていないが，オプション「-Knosch_pre_ra,nosch_post_ra」をつけるとバグを回避できることがわかっている．もしくは，空のアセンブラを該当ループに入れるとそのループだけ最適化が効かないためバグを回避できる．アルゴリズム 4 のループの性能としては，前者の方が 3 割ほど良くなる．

現状，筆者が使っている理研シミュレータを実行するサーバー (plum) にインストールされている富士通コンパイラが新しくなり (2018 年 10 月版) その場合このバグは確認されなかった．本書で報告されているパフォーマンスや計測時間の値は旧コンパイラでバグ回避用のオプションをつけた場合で報告をしているが，新コンパイラでオプションをつける場合とつけずに最適化を行った場合の性能差は 0.4% ほどであり，新しいコンパイラで計測をし直しても大きな性能差は生じないと結論づけたためである．

Algorithm 4 バグのあるループのコード．NO_OPTIMIZATION マクロを有効にすると最適化が行われず，正しく計算が行われる．バグの詳細としては，最適化によって for 文終了の比較命令と svcmpge 命令が入れ替わり，終了条件によっては for 文が早期に終了したり，長くなったりする．

```
#pragma loop norecurrence
#ifdef NO_OPTIMIZATION
for(int l=0,offset=0;l<ldnl[is];l++,offset+=16){
#else
const int n = ldnl[is];
for(int l=0,offset=0;l<n;l++,offset+=16){
#endif
  svfloat32_t flj = svld1_f32(pg,f_tmp+offset);
  svfloat32_t fcl = svld1_f32(pg,fc_tmp+offset);
  flj = svadd_f32_z(pg,flj,fcl);
  svfloat32_t dx = svld1_f32(pg,dx_tmp+offset);
  svfloat32_t dy = svld1_f32(pg,dy_tmp+offset);
  svfloat32_t dz = svld1_f32(pg,dz_tmp+offset);
  svfloat32_t fx = svmul_f32_z(pg,flj,dx);
  svfloat32_t fy = svmul_f32_z(pg,flj,dy);
  svfloat32_t fz = svmul_f32_z(pg,flj,dz);
  fxi = svadd_f32_z(pg,fx,fxi);
  fyi = svadd_f32_z(pg,fy,fyi);
  fzi = svadd_f32_z(pg,fz,fzi);
  int j = dnl[is][l];
  svbool_t excl = svcmpge_n_s32(pg,svdup_n_s32(j),i+16);
  f[j].x -= svaddv_f32(excl,fx);
  f[j].y -= svaddv_f32(excl,fy);
  f[j].z -= svaddv_f32(excl,fz);
}
```