千葉工業大学

修士学位論文

CUDAを用いたMPS法のための疎行列格納形式動的選択による行列ベクトル積の評価

2022年3月

所属専攻　　：情報科学専攻

学生番号・氏名　：2081027番

塙　翔登

指導教員　　：前川 仁孝　教授

修士論文要旨

専　攻	学生番号	氏　　名
情報科学	2081027	塙　翔登

論文題目

CUDA を用いた MPS 法のための
疎行列格納形式動的選択による
行列ベクトル積の評価

キーワード　CUDA, MPS 法，疎行列格納形式，疎行列ベクトル積

論文要旨

本論文では，MPS 法を高速化するために CUDA を用いた MPS 法における疎行列格納形式の動的選択で選択時間を隠蔽する手法を提案する．MPS 法は，流体の動きを解析するために流体を粒子の集まりとして粒子の動きを解析する．MPS 法の圧力計算は，疎行列を係数に持つ行列ベクトル積を繰り返し計算する．係数行列は問題規模に応じて変化するため，問題規模が大きい実問題では疎行列ベクトル積に時間がかかる．このため，並列計算による高速化が求められており，同一命令が多い疎行列ベクトル積では同一命令を大量のコアで処理できる CUDA が有効である．CUDA を用いた疎行列ベクトル積では，疎な係数行列を疎行列格納形式で格納することで無駄な演算やメモリアクセス回数を削減する．また，従来の疎行列ベクトル積では行列形状に合わせて適する格納形式を用いることの有効性が示されている．CUDA を用いた MPS 法における疎行列格納形式動的選択は，次ステップで用いる格納形式の決定と CUDA を用いる圧力計算に処理の依存がない．そこで，CUDA による圧力計算の実行中に，CPU 側で次ステップの格納形式を決定することで，格納形式を選択する時間を隠蔽し，MPS 法を高速化する．また，MPS 法でより適切な格納形式を決定するために非零要素のばらつきや非零要素率を確認し，選択条件を調整する．評価の結果，選択時間を隠蔽する MPS 法の実行時間は従来手法と比較して最大 1.02 倍の高速化率が得られた．また，選択条件を調整した MPS 法の実行時間は最大 1.089 倍の高速化率が得られた．

千葉工業大学大学院情報科学研究科

◎要旨は 600 字程度

Summary of Master's Thesis

Course	Student No.	SURNAME, Firstname
Information and Computer Science	2081027	HANAWA Shoto

Title

Evaluation of Matrix-Vector Multiplication by Dynamic Selecting Storage Schemes of Sparse Matrix for MPS method on CUDA

Keywords CUDA, MPS method, Sparse matrix storage schemes, SpMV

Summary

This paper proposes a speedup MPS method with hiding the selecting time in the auto-tuning algorithm of SpMV by selecting storage schemes of the MPS method on CUDA. MPS method models a fluid as a collection of particles and analyzes the movement of the particles. Pressure calculation in the MPS method is an iterative SpMV with a sparse matrix. Since the coefficient matrix varies with the problem size, the SpMV is time-consuming for real problems. For this reason, SpMV on CUDA is effective because it can process the same instruction on many cores. SpMV on CUDA stores the coefficient matrices in sparse matrix storage schemes to reduce the number of unnecessary operations and memory accesses. Conventional SpMV on CUDA shows that switching storage schemes by the matrix is effective. The auto-tuning algorithm of SpMV by selecting storage schemes of the MPS method on CUDA does not depend on the next steps of the storage schemes determination and pressure calculation process. Therefore, the storage schemes of the next step are determined during the pressure calculation by CUDA to hide the time for selecting the storage schemes. Also, the selection conditions using the variation of non-zero elements and element rates are adjusted to determine the appropriate storage schemes. The performance evaluation shows that the MPS method with hidden selection time up to 1.02 times and the MPS method with adjusted selection conditions is up to 1.089 times compared with a conventional method.

Graduate School Information and Computer Science, Chiba Institute of Technology

The contents of English summary should be written in about 200 words.

目 次

図 目 次

表 目 次

第1章
はじめに

MPS 法は，連続体である解析対象を粒子の集まりとしてモデル化し，粘性や重力，圧力などを基に，粒子の動きを解析する手法である [1]．本手法は，水，空気などの非圧縮性流体を解析する手法であり，自由表面や大変形などが伴う動きの激しい物理現象の解析に優れている [2]．このことから，これまでに津波解析や土石流解析，地震応答解析などの様々な問題に適用，実用化される [3][4]．特に津波や地震など災害問題における解析では，実際に起こる可能性のある災害を解析し，災害対策に役立てることが可能である．このような問題規模が大きい実問題に対しては，1 つの問題の解析に膨大な時間がかかり，実用化が難しいため，MPS 法の高速化が求められている．

MPS 法の計算処理では，時間ステップごとに粘性項，重力項を陽的に，圧力を陰的に繰り返し計算するため，圧力計算に最も時間がかかる．MPS 法の圧力計算部分で生成される連立方程式の係数行列は，各粒子とその近傍にある粒子の相互作用に基づいて生成される．連立方程式の係数行列は疎行列であるため，圧力計算には主に CG 法が用いられる．MPS 法ではこの圧力に関する連立方程式求解を並列化し高速化する研究が行われている．圧力計算の高速化における研究として，計算領域を空間分割しそれぞれの領域を複数 PC で並列計算する手法 [5] やスレッド並列化を行う手法 [6]，大規模解析におけるメモリ使用量を考慮した分散共有メモリ並列計算上でプロセス並列とスレッド並列を組み合わせた手法 [7]，大量のコアにより同一命令を複数実行できる CUDA を用いた高速化が行われている [8]．

CUDA を用いた MPS 法の連立方程式求解において，疎な係数行列を密行列として格納すると，無駄な零要素が多く非零要素の格納場所もばらばらである．このため，演算処理回数やメモリアクセス回数が増え，処理時間が大きくなる．このことから，CUDA を用いた MPS 法における疎行列ベクトル積では，無駄な計算の排除やメモリアクセス回数の削減が行われている [9]．疎行列ベクトル積では，無駄な計算や無駄なメモリアクセス回数の削減を目的とした疎行列格納形式がある [10]．疎行列格納形式は，疎行列の零要素を省き，非零要素をメモリに格納する方法で様々な格納形式がある．また，格納形式によってメモリアクセス削減を優先する格納形式

や零要素削減を優先する格納形式など方向性が異なることから，対象とする行列によって適する格納形式が違う可能性がある．CUDA を用いた疎行列計算における研究では，CRS 形式，ELL 形式，JDS 形式など複数の疎行列格納形式を用いて，行列に適する疎行列格納形式を用いることの有効性が示されている [11]．

MPS 法では問題に合わせて CRS 形式，ELL 形式，JDS 形式が用いられている [2]．CRS 形式は零要素を全て削減するが, メモリアクセス回数が ELL 形式，JDS 形式と比べて多い．ELL 形式は零要素を残すことでメモリアクセス回数を削減するが，パディングによる無駄な演算が発生する．JDS 形式は零要素を全て削減し，メモリアクセス回数も削減できるが，スレッドごとの処理数が偏り，処理時間が遅くなる可能性がある [12]．MPS 法では対象問題に合わせてこれらの疎行列格納形式を 1 つに決定して用いる．このため，圧力計算に使用する係数行列の非零要素位置に合わせて疎行列格納形式を動的に選択することにより高速化ができると考えられる．そこで，本研究では，CUDA を用いた MPS 法の圧力計算で疎行列格納形式の動的選択を行い，圧力計算時間を高速化する．提案手法では，MPS 法の係数行列において時間ステップごとの流体粒子位置によって行列の非零要素配置が変化することを利用し，前ステップの動的選択後に次ステップの格納形式を決定する．また，次ステップの格納形式決定と CUDA による圧力計算は，用いるアーキテクチャが異なることから，処理の依存がない．このため，本手法では，CUDA による圧力計算中に次ステップの格納形式を決定することで格納形式選択時間を隠蔽する．また，本手法の格納形式選択パラメータは，係数行列の規模や形状に大きく左右される．このため，CUDA を用いた MPS 法でより適切な格納形式が選ばれるように選択条件のパラメータを調整する．

本研究では，最初に予備評価として，疎行列ベクトル積における短縮時間の評価，動的選択におけるパラメータの設定，選択時間隠蔽による実行時間について評価する．その後，疎行列格納形式の動的選択を実装した MPS 法における全体実行時間を評価する．疎行列ベクトル積における短縮時間の評価では，MPS 法が生成する疎行列においてそれぞれ CRS 形式，ELL 形式，JDS 形式の CUDA を用いた行列ベクトル積の実行時間を測定し，求解の実行時間から格納形式を切り替えることによって短縮できる時間があるのかを確認する．動的選択におけるパラメータの設定では，MPS 法においてより適切な格納形式を選ぶために，本研究で用いる粒子配置より，ばらつきや非零要素率の推移を調査し，動的選択のパラメータを設定する．また，選択時間隠蔽による実行時間の評価では，本手法で提案する選択時間の隠蔽が有効であるかどうかを確認するために，格納形式選択時間の隠蔽をする MPS 法と隠蔽を

しない MPS 法とで実行時間を比較する．これらの予備評価を踏まえ，本評価では，MPS 法全体の処理時間と圧力計算時間を測定し，CUDA を用いた MPS 法に疎行列格納形式の動的選択を用いた行列ベクトル積の有効性を確認する．

以下の章では，まず，第 2 章，第 3 章で本手法の基となる MPS 法の理論と CUDA を用いた MPS 法の圧力計算について述べる．次に，提案手法を説明するため，第 4 章で MPS 法における疎行列格納形式の動的選択や提案手法である選択時間の隠蔽について述べる．第 5 章では本手法が MPS 法に有効であるかを確認するため，予備評価を含めて動的選択を用いた MPS 法の実行時間について評価し，最後に第 6 章でまとめる．

第2章
MPS法

粒子法は，連続体である解析対象を粒子の集まりとしてモデル化し，粘性や重力，圧力に基づいて，粒子の動きを解析する手法である[(1)(13)]．粒子法は，計算点の粒子を流れに沿って移動させるため，有限差分法，有限体積法，有限要素法などの格子法に用いる計算格子を必要としない．図 2–1 に格子法のモデル，図 2–2 に粒子法のモデルをそれぞれ示す．図 2–1 の格子法では，問題領域を格子状に分割し，格子が隣接している部分を格子点とする．格子法における速度などの変数は格子点上に格納し，格子点を用いて離散式を構築する．これに対し，図 2–2 の粒子法は，粒子ひとつひとつが計算点となり，粒子ごとに変数をもつ．このため，変数の格納としては格子法の格子点と粒子法の粒子は同じ意味を持つ．粒子法では，粒子相互作用モデルによって離散式を構築する．粒子相互作用モデルは，対象の粒子に対し，その近傍にある粒子が影響するため，格子法のように近接関係をあらかじめ明示する必要は無く，格子法より細かい物質の動作を解析することができる．このため，粒子法は格子法に比べて，自由表面や大変形などが伴う動きの激しい物理現象を解くことに優れている．

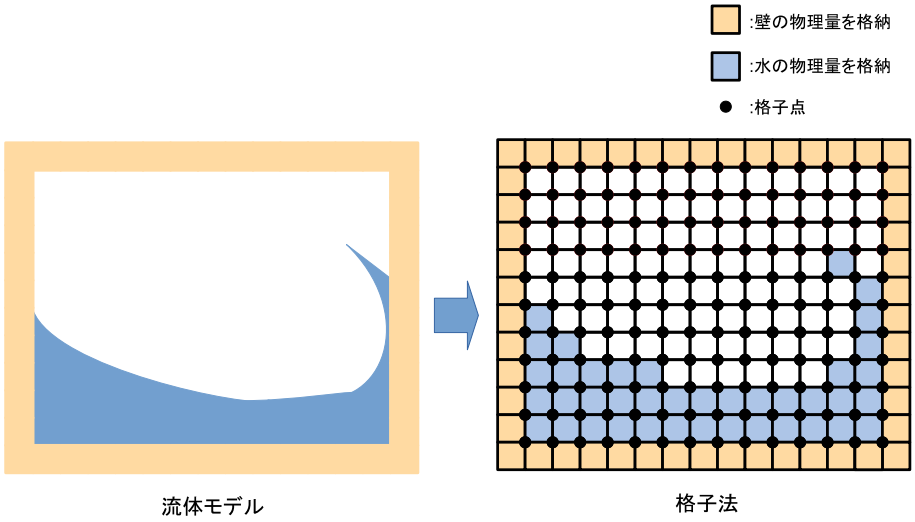


図 2–1 ： 格子法のモデル化

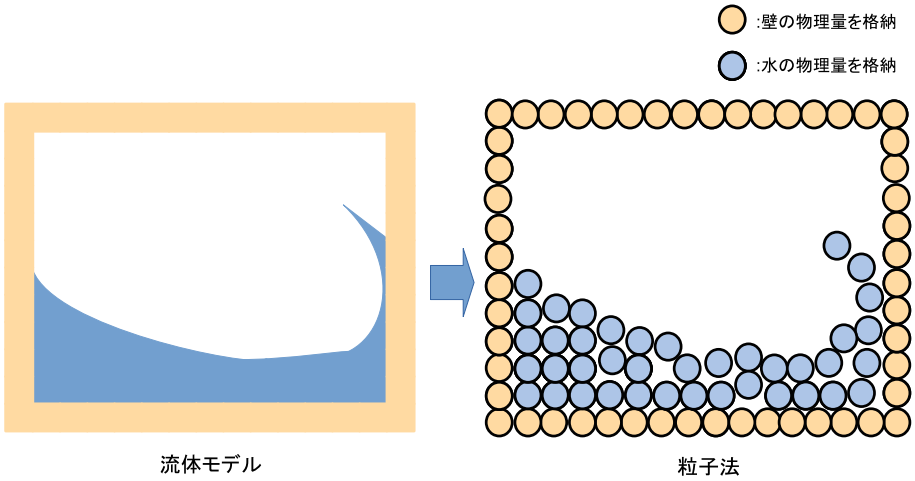


図 2–2 ： 粒子法のモデル化

MPS 法は，非圧縮性流体を解析する粒子法のひとつである．粒子法では，時間だけが独立変数となり，粒子の位置，速度，加速度は時間の関数として，流体を構成している粒子を運動方程式を解いて粒子を追跡することで解析する．粒子法の連続体運動の記述法としてラグランジュ法を使用する．MPS 法は，粒子間距離などの物理量計算に粒子間相互作用モデルを導入する．粒子間相互作用モデルでは，ある一定距離内に含まれる近傍粒子との相互作用を時間ステップごとに計算し，粒子の過剰接近による団粒化を抑止する．近傍粒子を計算する際に，すべての粒子 N 個との物理量を計算した場合，ひとつの粒子に対し $N-1$ 回の計算が必要となり，計算のオーダーは N^2 に比例する．MPS 法では，近くの粒子ほど相互作用が大きく，遠くの粒子との相互作用は無視できるほど小さい．このため，近くにある粒子を効率よく探索すれば，近傍粒子探索にかかる時間を削減でき，計算時間の短縮が可能である．図 2–3 に粒子間の相互作用を示す．図 2–3 の粒子間相互作用モデルは，対象とする粒子 i に対し，その粒子 i から半径 r_e 以内の近傍に存在する複数の粒子 j と相互作用するものとする．この相互作用の範囲を決める半径 r_e を影響半径と呼ぶ．ここで用いる粒子間相互作用モデルは，あくまで計算上のモデルであり，実際の粒子間相互作用ではない．

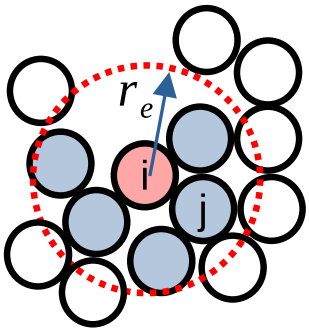


図 2–3 ： 粒子間の相互作用

2.1　MPS 法の基礎理論

MPS 法において液体や気体などの流体の現象を解析するためには，流体の現象を表す基本となる方程式の運動量保存則 (ナビエ – ストークス方程式) と質量保存則 (連続の式) に従って．流体を移動し，質量や密度を計算する．式 (2–1) にナビエ – ストークス方程式，式 (2–2) に連続の式を示す．ここで，P は圧力，ρ は密度，ν は動粘性係数，u は速度，g は重力加速度である．また，式 (2–1) の右辺の第 1 項は，圧力による加速度の項であり，式 (2–1) の右辺の第 2 項は，粘性力による加速度の項である．式 (2–2) の連続の式は，ある単位体積の領域から流体が流出するとき，その領域内の流体の密度がどのように変化するかを表す．MPS 法は，この 2 式に従って，流体粒子を移動させて流体を解析する．

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{1}{\rho}\nabla P + \nu\nabla^2\boldsymbol{u} + \boldsymbol{g} \tag{2–1}$$

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\boldsymbol{u} = 0 \tag{2–2}$$

式 (2–1)，式 (2–2) では，密度 ρ が用いられることから，流体の計算をする際には流体の密度が重要となる．流体現象を表すには，流体の密度を計算の中で適切に評価する必要がある．その評価方法として，MPS 法では粒子数密度を用いる．粒子数密度は，注目している粒子のまわりに粒子がどの程度の密度で分布しているかを示す．図 2–4 にある粒子 i に位置における粒子数密度計算の概念図を示す．図 2–4 では，粒子間距離 l_0 で粒子を均等に配置する初期配置を想定している．赤色の粒子が対象粒子であり，その粒子に対し r_e の影響範囲がある．影響範囲内にある緑色の粒子は，色の濃さにより，重みの大きさを表し，対象となる粒子に近いほど重みが大

きいことを示している．粒子数密度は，緑色の粒子の重みを足し合わせたものとなるため，式 (2–3) で表す．ここで，n_i は粒子 i の粒子数密度，$\boldsymbol{r}_i$ は粒子 i の位置ベクトル，$\boldsymbol{r}_j$ は粒子 j の位置ベクトルである．式 (2–3) 中の $\omega\left(|\boldsymbol{r}_j - \boldsymbol{r}_i|\right)$ は重み関数を表し，式 (2–4) で計算する．遠くの粒子に対する重みは無視できるほど小さいため，近傍粒子に限定して総和をとることで計算時間が短縮できる．このため，式 (2–4) では，対象粒子からの距離が r_e 以上離れると重み関数の値を 0 とする．また，粒子数密度の基準値を定めるために，図 2–4 のように粒子が等間隔に配置する初期配置において，粒子数密度を求める．式 (2–5) に粒子数密度の基準値を示す．式 (2–5) より，粒子数密度の基準値は n^0 で表され，全粒子において共通の値とし，時間によっても変化しない．また，n^0 は重み関数の総和の基準でもあり，重み平均の際に定数として用いる．

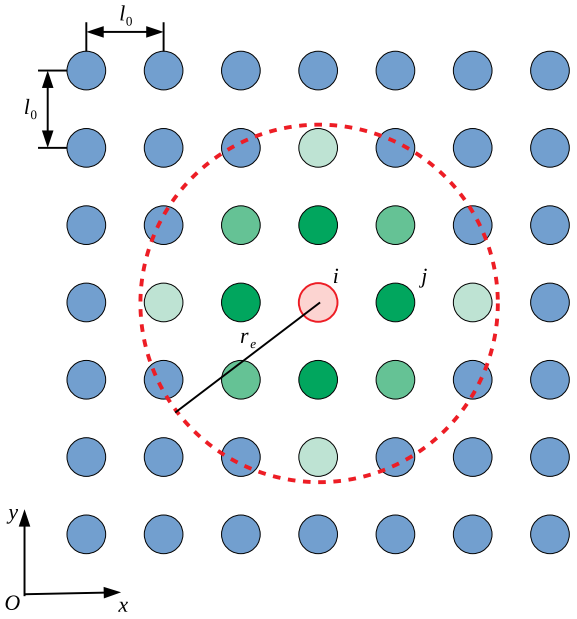


図 2–4 ： 粒子重み関数

$$n_i = \sum_{j \neq i} \omega\left(|\boldsymbol{r}_j - \boldsymbol{r}_i|\right) \tag{2–3}$$

$$\omega(r) = \begin{cases} \left(\dfrac{r_e}{r}\right) - 1 & (r < r_e) \\ 0 & (r \geq r_e) \end{cases} \tag{2–4}$$

$$n^0 = \sum_{j \neq i'} \omega \left(|\boldsymbol{r}_j^0 - \boldsymbol{r}_{i'}^0| \right) \tag{2–5}$$

2.1.1 勾配モデル

流体の運動方程式である式 (2–1) のナビエ – ストークス方程式の右辺の加速度成分の計算には，偏微分演算子のナブラとラプラシアンが含まれる．コンピュータの演算が可能であるのは，四則演算であり，偏微分という演算は基本的にコンピュータで行うことができない．このため，離散化をすることで偏微分をコンピュータに演算できる形に変換する．ナブラはスカラー変数に作用してベクトルが得られる演算子である．式 (2–6) にナブラの演算式を示す．式 (2–6) より，ナブラは x，y，z に関する各方向の 1 階の偏微分の演算で表される．これをスカラー量 ϕ に作用させれば，その位置における ϕ の勾配ベクトルを求めることができる．式 (2–7) に ϕ が $\phi(x, y, z)$ のときの勾配ベクトルを示す．式 (2–7) の勾配ベクトルの大きさは傾きの大きさを示し，勾配ベクトルをベクトルの大きさで割ったものが，ϕ が高い方向を表す単位方向ベクトルとなる．MPS 法では，ϕ に対する勾配ベクトルを勾配モデルと呼ぶ．MPS 法における ϕ には圧力 P，速度 $\boldsymbol{u}$ が該当する．勾配モデルは，圧力計算で用いられるため，ϕ は圧力 P となり，ある粒子に対してその近傍にある粒子との間の圧力の受け渡しにより表される．このため，勾配ベクトルに近傍の粒子を考慮するような重み関数を利用した重み平均を取るようにする．よって，MPS 法では粒子 i の位置における勾配ベクトルに対して式 (2–8) に示す計算モデルを用いる．式 (2–8) において，d は空間の次元数である．記号 $\langle\rangle_i$ は，粒子 i の位置において，粒子のモデルを用いて $\langle\rangle$ の中の値や関数を近似していることを示す．このため，偏微分の記号で表す項と粒子モデルで近似した項を $\simeq$ でなく $=$ で表すことができる．図 2–5 に，式 (2–8) における ϕ を圧力 P とした際の勾配モデルの概念図を示す．図 2–5 では，i の対象粒子と複数の近傍粒子のうち j の近傍粒子に着目する．図 2–5 において，粒子 i と粒子 j の上にある緑色の棒の高さは圧力の高さを示す．式 (2–8) の ϕ を P とすると，$P_j - P_i/|\boldsymbol{r}_j - \boldsymbol{r}_i|$ は粒子 i とその近傍粒子 j の圧力の差を距離で割った値であり，粒子 i から粒子 j へ向かう方向における圧力勾配の大きさを示す．次の係数である $(\boldsymbol{r}_j - \boldsymbol{r}_i)/|\boldsymbol{r}_j - \boldsymbol{r}_i|$ は，粒子 i から粒子 j へ向かう単位方向ベクトルを示す．3 番目の係数は重み関数である．

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \tag{2–6}$$

$$\nabla\phi = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \phi = \begin{pmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \\ \frac{\partial\phi}{\partial z} \end{pmatrix} \tag{2–7}$$

$$\langle\nabla\phi\rangle_i = \frac{d}{n^0}\sum_{j\neq i}\left[\frac{\phi_j - \phi_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}(\boldsymbol{r}_j - \boldsymbol{r}_i)\omega(|\boldsymbol{r}_j - \boldsymbol{r}_i|)\right] \tag{2–8}$$

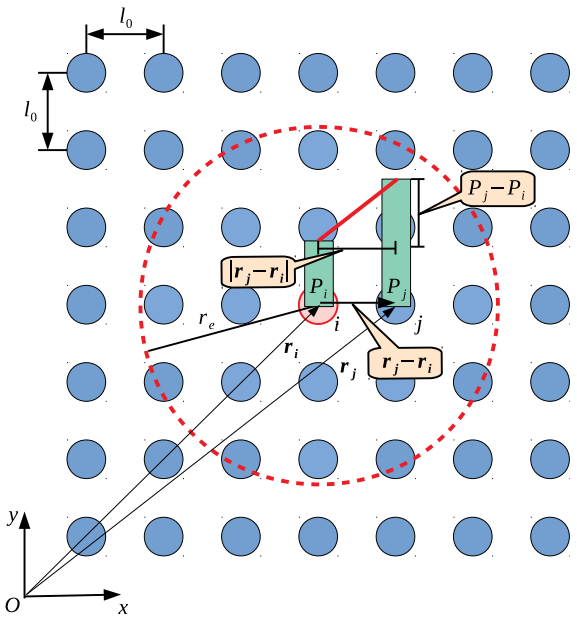


図 2–5 ： 勾配モデル

2.1.2　ラプラシアンモデル

式 (2–9) にラプラシアンの演算式を示す．式 (2–9) より，あるスカラー量 ϕ のラプラシアンをとる値は，x，y，z 方向の偏微分をそれぞれ 2 回ずつ足し合わせたものである．ラプラシアンは，粘性項の計算や圧力のポアソン方程式で計算する必要がある．このため，MPS 法のラプラシアンモデルにおける ϕ には，圧力 P，速度 $\boldsymbol{u}$ が該当する．式 (2–10) に粒子 i の ϕ に対するラプラシアンモデルを示す．ここで，λn^0 は，式 (2–11) で表される．式 (2–8) の勾配モデルと同じく，d は次元数，ω は重み関

数である．n^0 は式 (2–5) で求める粒子数密度であり，重み関数の総和を規格化するために用いる．また，λ^0 は影響半径内にある近傍粒子との距離の 2 乗における重み平均値を意味する．ラプラシアンモデルは，粒子 i と粒子 j の粒子間で対称性があり，圧力ポアソン方程式などに適用する場合，係数行列が対象になる．図 2–6 にラプラシアンモデルの概念図を示す．図 2–6 では，粒子 i が持つ物理量を近傍粒子 j に拡散する様子を示す．図 2–6 中の粒子 i にある棒の高さは粒子 i が持つ ϕ の大きさを表し，棒の白い領域は，粒子 i の ϕ の減少を示す．また，図 2–6 中の粒子 j にある棒の高さは粒子 j が粒子 i から受け取る ϕ の大きさを示す．ϕ を渡す際，式 (2–10) で表されるように，粒子 i がもつ ϕ の大きさである ϕ_i に比例して ϕ を近傍粒子 j に渡す．また，重み関数を用いて，近くの粒子には比較的多くの ϕ を与え，遠くの粒子には少しの ϕ を渡す．

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} \tag{2–9}$$

$$\langle\nabla^2\phi\rangle_i = \frac{2d}{\lambda^0 n^0}\sum_{j\neq i}(\phi_j - \phi_i)\omega(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \tag{2–10}$$

$$\lambda^0 = \frac{\sum_{j\neq i'}\left(|\boldsymbol{r}_j^0 - \boldsymbol{r}_{i'}^0|^2\omega(|\boldsymbol{r}_j^0 - \boldsymbol{r}_{i'}^0|\right)}{\sum_{j\neq i'}\omega(|\boldsymbol{r}_j^0 - \boldsymbol{r}_{i'}^0|)} \tag{2–11}$$

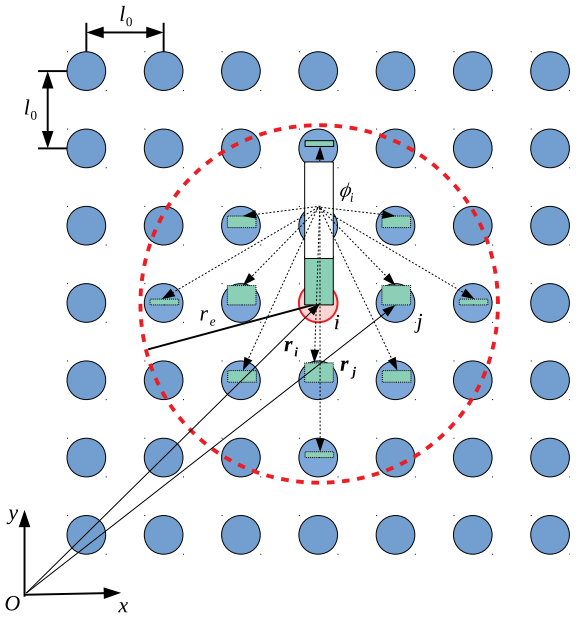


図 2–6 ： ラプラシアンモデル

2.2　MPS 法の処理

第 2.1 章の基礎理論を元に MPS 法の動作を説明する．MPS 法は，時刻 t^k における各粒子の位置 $\boldsymbol{r}_j^k$，速度 $\boldsymbol{u}_j^k$，圧力 P_j^k を元に，次の時刻の値である $\boldsymbol{r}_j^{k+1}$，$\boldsymbol{u}_j^{k+1}$，P_j^{k+1} を計算する．また，MPS 法は圧力のみを陰的，圧力以外を陽的に解くことから半陰解法である．図 2–7 に MPS 法のフローチャートを示す．図 2–7 より，粘性項，重力項を陽的に計算し，仮の粒子位置と粒子速度を求める．

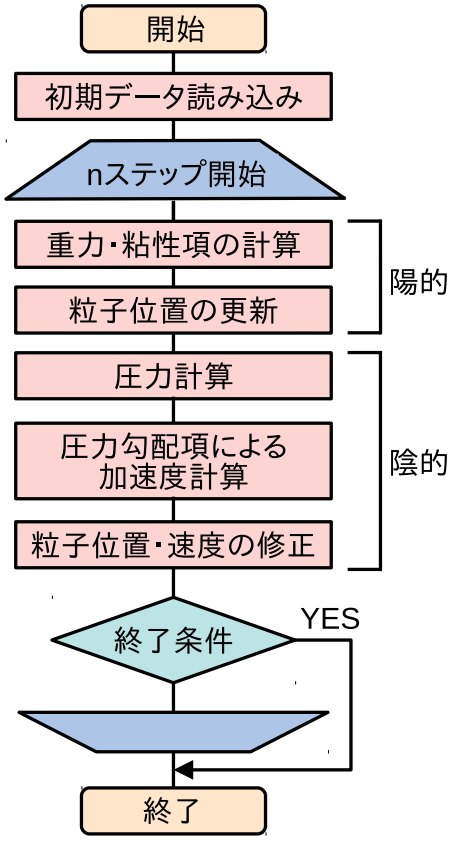


図 2–7 ： MPS 法のフローチャート

MPS 法で解くべきナビエ – ストークス方程式は，式 (2–12) のようになる．式 (2–12) の ρ^0 は密度を表し，非圧縮性流体では常に一定である．式 (2–12) を半陰解法アルゴリズムに沿って計算する．

$$\frac{\boldsymbol{u}_i^{k+1}-\boldsymbol{u}_i^k}{\Delta t}=-\frac{1}{\rho^0}\langle\nabla P\rangle_i^{k+1}+\nu\langle\nabla^2 u\rangle_i^k+\boldsymbol{g}^k \tag{2–12}$$

$$\frac{\boldsymbol{u}_i^* - \boldsymbol{u}_i^k}{\Delta t} = \nu \langle \nabla^2 u \rangle_i^k + \boldsymbol{g}^k \tag{2–13}$$

$$\frac{\boldsymbol{u}_i^{k+1} - \boldsymbol{u}_i^*}{\Delta t} = -\frac{1}{\rho^0} \langle \nabla P \rangle_i^{k+1} \tag{2–14}$$

粘性項と重力項は，式 (2–12) の右辺における第 2 項と第 3 項である．粘性項は，流体摩擦の効果であり，流体の速度をそのまわりの流体の速度に近づけようとする力である．粘性力の大きさは速度の 2 階微分に比例する．また，水や油など解析対象とする流体によっても値が変化する．重力項は，外力であり，主に重力加速度を与える．MPS 法では，最初に式 (2–13) の粘性項と重力項を陽的に計算し，式 (2–15) を用いて粒子の仮の速度 $\boldsymbol{u}_i^*$ を求める．

$$\boldsymbol{u}_i^* = \boldsymbol{u}_i^k + \Delta t \left[\nu \langle \nabla^2 u \rangle_i^k + \boldsymbol{g}^k \right] \tag{2–15}$$

式 (2–15) の右辺には時刻 t^k の値が用いられているため，各数値を代入するだけで仮の速度が求められる．式 (2–15) の粘性項には，ラプラシアンが含まれているため，式 (2–11) のラプラシアンモデルを適用した式 (2–17) を計算する．式 (2–17) は，すべての粒子 i について行う．図 2–8(a) に仮速度の計算が完了した粒子の状態を示す．図 2–8(a) の青色の粒子は流体粒子，茶色の粒子は壁粒子，緑色の粒子はダミー粒子を示す．

$$\langle \nabla^2 \boldsymbol{u} \rangle_i^k = \frac{2d}{\lambda^0 n^0} \sum_{j \neq i} \left(\boldsymbol{u}_j^k - \boldsymbol{u}_i^k \right) \omega(|\boldsymbol{r}_j^k - \boldsymbol{r}_i^k|) \tag{2–16}$$

$$\boldsymbol{u}_i^* = \boldsymbol{u}_i^k + \Delta t \left[\nu \frac{2d}{\lambda^0 n^0} \sum_{j \neq i} \left(\boldsymbol{u}_j^k - \boldsymbol{u}_i^k \right) \omega(|\boldsymbol{r}_j^k - \boldsymbol{r}_i^k|) + \boldsymbol{g}^k \right] \tag{2–17}$$

次に式 (2–17) で計算した仮速度 $\boldsymbol{u}_i^*$ を用いて式 (2–18) により仮の位置を求める．図 2–8(b) に仮の位置に粒子が移動した状態を示す．図 2–8(b) では，非圧縮条件が満たされていないため，密度が高く，粒子同士が接近しすぎてしまう．

$$\boldsymbol{r}_i^* = \boldsymbol{r}_i^k + \boldsymbol{u}_i^* \Delta t \tag{2–18}$$

次に仮の粒子位置と粒子速度を用いて圧力を解くことによって求め，粒子位置，粒子速度を修正する．圧力は式 (2–12) の右辺における第 1 項である．粘性項と重力項の計算が終了した時点の $\boldsymbol{r}_i^*$ における粒子数密度 n_i^* は，非圧縮性を考慮した計算を行っていないため，粒子数密度 n^0 が一定でない．このため，陰的な計算で粒子数密

度を修正する必要がある．粒子数密度を修正することで，粒子の位置，速度，圧力が修正され，次の時刻 t^{k+1} の値が確定する．粒子数密度の修正を式 (2–19)，速度の修正を式 (2–20) に示す．式 (2–19) の $\boldsymbol{n}_i'$ は粒子数密度の修正量，式 (2–20) の $\boldsymbol{u}_i'$ は速度の修正量である．仮の速度 $\boldsymbol{u}_i^*$ は，非圧縮性を満たしていないため，$\boldsymbol{u}_i'$ により速度を修正する．速度の修正量が陰的な圧力項の計算によって生じるとすると，式 (2–14) に式 (2–20) を代入して式 (2–21) のようになる．式 (2–21) の圧力 $\langle \nabla P \rangle_i^{k+1}$ の計算には，式 (2–8) の勾配モデルを用いる．

$$n_i^{k+1} = n^0 = n_i^* + n_i' \tag{2–19}$$

$$\boldsymbol{u}_i^{k+1} = \boldsymbol{u}_i^* + \boldsymbol{u}_i' \tag{2–20}$$

$$\boldsymbol{u}_i' = -\frac{\Delta t}{\rho^0} \langle \nabla P \rangle_i^{k+1} \tag{2–21}$$

ここで，式 (2–2) の流体の質量保存則において，左辺の第 2 項を ρ^0 で近似すると，式 (2–22) となる．また，流体の密度は粒子数密度に比例するため，$\rho^i = n_i$，$\rho^0 = n_0$ となり，式の両辺を ρ^0 で割ると式 (2–23) となる．式 (2–23) を時間に対して離散化すると式 (2–24) となる．

$$\frac{D\rho_i}{Dt} + \rho^0 \nabla \cdot n_i' = 0 \tag{2–22}$$

$$\frac{1}{n^0} \frac{Dn_i}{Dt} + \nabla \cdot n_i' = 0 \tag{2–23}$$

$$\frac{1}{n^0} \frac{\boldsymbol{u}_i^* - \boldsymbol{u}_i'}{\Delta t} + \nabla \cdot n_i' = 0 \tag{2–24}$$

式 (2–21) の両辺に ∇ をかけて，式 (2–24) を代入し，式 (2–19) を用いて書き換えると圧力のポアソン方程式を得ることができる．式 (2–25) に圧力のポアソン方程式を示す．式 (2–25) の左辺に式 (2–11) のラプラシアンモデルを適用すると式 (2–26) となり，P_i^{k+1} に対する連立一次方程式を得ることができる．この時，重み関数の計算に用いる粒子の座標は，陽的に計算した仮の粒子座標 $\boldsymbol{r}_j^*$ を用いる．式 (2–26) を解くことで，次の時刻 $k+1$ の圧力が求まる．図 2–8(c) に圧力を計算するときの粒子の状態を示す．図 2–8(c) では，赤色の粒子が圧力がかかる粒子であり，色が濃いほど圧力が強いことを示す．

$$\langle \nabla^2 P \rangle_i^{k+1} = -\frac{\rho^0}{\Delta t^2} \frac{n_i^* - n^0}{n^0} \tag{2–25}$$

$$\langle \nabla^2 P \rangle_i^{k+1} = \frac{2d}{\lambda^0 n^0} \sum_{j \neq i} \left(P_j - P_i \right) \omega(|\boldsymbol{r}_j^* - \boldsymbol{r}_i^*|) \tag{2–26}$$

式 (2–26) で求めた圧力 P_i^{k+1} を式 (2–21) に代入すると速度の修正量 $\boldsymbol{u}_i'$ を求めることができ，更にその値を式 (2–20) に代入すると次の時刻の粒子速度 $\boldsymbol{u}_i^{k+1}$ が得られる．最後に，求めた $\boldsymbol{u}_i^{k+1}$ を用いて粒子位置を修正する．次の時刻の粒子位置 $\boldsymbol{r}_i^{k+1}$ は，式 (2–27) により求める．図 2–8(d) に粒子速度と粒子位置を修正した粒子の状態を示す．図 2–8(d) では，粒子速度と粒子位置を修正することにより，密度が高く粒子が重なっていた部分が圧力に従って移動する．

$$\boldsymbol{r}_i^{k+1} = \boldsymbol{r}_i^* + (\boldsymbol{u}_i^{k+1} - \boldsymbol{u}_i^*) \Delta t \tag{2–27}$$

これらの処理を時間ステップで繰り返すことにより，粒子を解析する．半陰解法の MPS 法は，圧力場の解に相当激しい擾乱を伴う．このため，流体を評価の対象として粒子法を使用する場合は，圧力擾乱の低減が必要である．また，MPS 法における表面張力の計算では，ディリクレ境界条件を用いる．ナビエ – ストークス方程式では，圧力勾配項で圧力を相対的な量として扱えることから，自由表面の粒子の圧力を 0[Pa] と固定して計算する．また，MPS 法では，各項の計算時に近傍粒子探索を行うことにより，計算回数を削減する．近傍粒子探索の手法としては，主に相互作用計算する可能性がある近傍粒子のインデックスを各粒子が近傍リストとして記憶する粒子登録法や，計算領域をセル分割してハッシュ配列を用意することで各粒子の所属するセル番号を記憶するハッシュ法がある [14]．また，メモリ使用量を抑えるために，粒子登録法とハッシュ法のハイブリット手法 [15] や，粒子登録法と連結リスト法のハイブリット手法 [16] がある．他手法の研究では，スライスグリッドと言われるデータ構造とソートを組み合わせてデータの空間局所性を高めることにより，キャッシュ効率を向上し，近傍粒子探索を効率化した手法がある [17][18]．

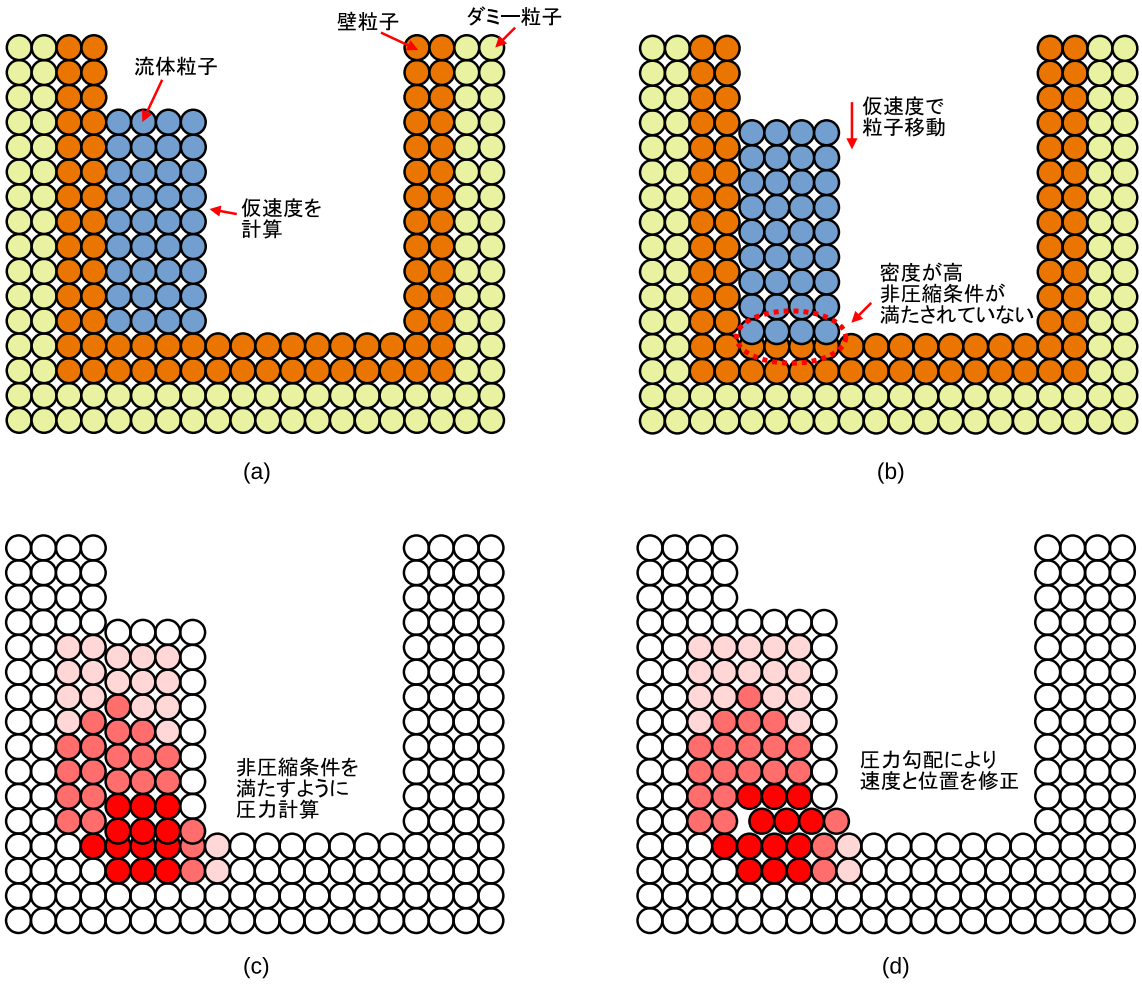


図 2–8 ： 半陰解法を用いた粒子の移動方法

2.3　MPS 法の圧力計算における係数行列形状

MPS 法の圧力計算は，流体粒子とその影響範囲にある粒子との相互作用力に基づいて生成された行列を係数とする式 (2–26) の連立一次方程式を解く．図 2–9 に圧力計算で生成する係数行列の例を示す．図中の粒子番号①～⑥は流体の粒子を表し，それ以外の○は壁の粒子を表す．①の粒子の周りにある点線の赤い円は粒子の影響範囲を表す．図 2–9 では，壁粒子から作用する力を定数として扱う．図 2–9 より，係数行列の大きさは，流体粒子数の 2 乗で，影響範囲内に粒子がある場合，その粒子との相互作用により，係数行列に相互作用に基づく値が格納される．例として，3 番の粒子に着目した場合，近傍粒子は，4 番の粒子である．これにより，係数行列の 3

行目には，対角要素と近傍粒子の番号である列に相互作用に基づく値が格納される．この操作を全ての流体粒子に対して相互作用の値を計算することにより，係数行列を生成する．MPS 法で生成される係数行列は，対称疎行列であることから，圧力計算では計算処理部分に主に CG 法などが使用される [19]．このため，後藤らによる GPU を用いた MPS 法では，対角スケーリング前処理を行う SCG 法を用いている [8]．また，2 次元だけではなく，3 次元の MPS 法を対象にした研究がある [20][2][9][21]．

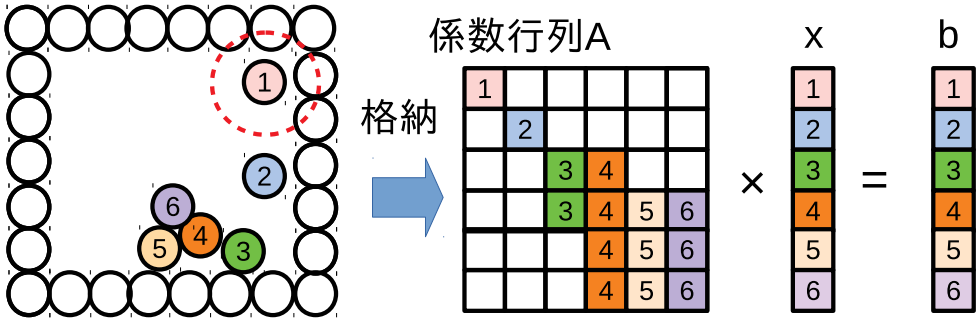


図 2–9 ： 係数行列生成の例

2.4　CG 法

CG 法 (共役勾配法) は，対称正定値行列を係数とする連立一次方程式を解くためのアルゴリズムである．行列 A が正定値対象の連立 1 次方程式を解くことは，次の式 (2–28) のような 2 次関数最小化問題を解くことと同値となる．

$$f(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} \tag{2–28}$$

図 2–10 に CG 法のフローチャートを示す．図 2–10 より，CG 法は近似解に対する残差を用いた方向ベクトルの計算を収束するまで時間ステップごとに繰り返す．収束が難しい条件の悪い問題は，行列に前処理の適用により，非常に速く収束することができる．CG 法の前処理行列に望まれる性質は，元の行列の疎性の保持とその近似度の高さである．加えて，反復計算の高い並列性を実現するためには，前処理行列の形状が重要である．代表的な前処理手法に，連立 1 次方程式の係数行列を三角行列どうしの積の形に不完全分解する不完全コレスキー分解などがある．

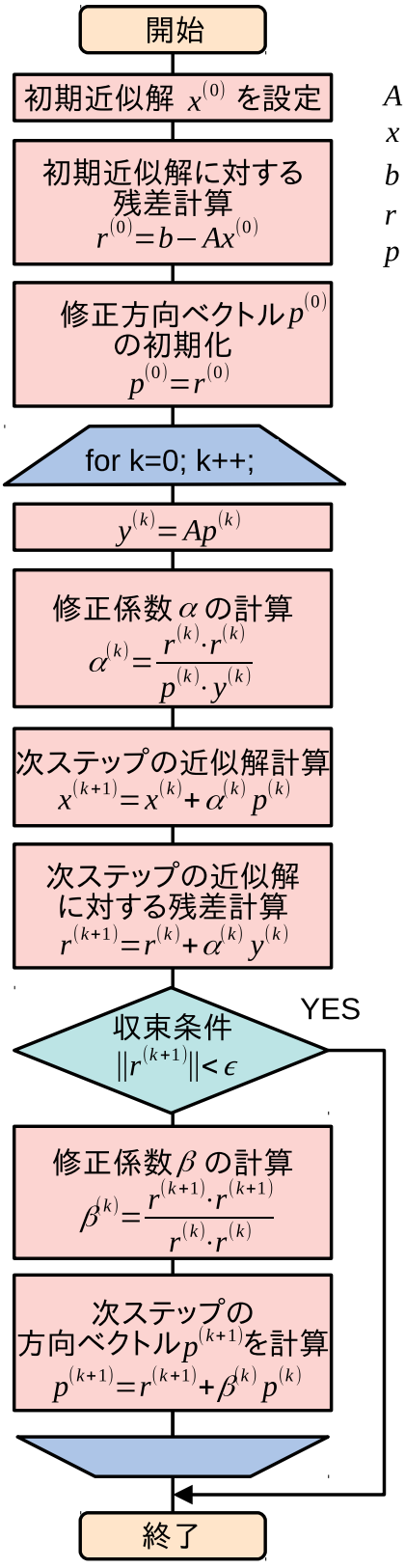


図 2–10 ： CG 法のフローチャート

第3章
CUDAを用いたMPS法の圧力計算

CUDAを用いたMPS法は，圧力における計算をGPUを用いて実行する．図3–1にCUDAを用いたMPS法のフローチャートを示す．グローバルメモリに格納される物理量，圧力，速度，位置のデータが，スレッドからのアクセスが連続なコアレスアクセスになるように配置し，1スレッドに1粒子の計算を行う．圧力計算部分では，疎行列ベクトル積にCG法が用いられている．メモリアクセス遅延を改善するために，圧力方程式の係数行列に対して，疎行列格納形式を用いる．CUDAを用いたMPS法の圧力計算に対する疎行列格納形式には主にCRS形式とELL形式が用いられており，グローバルメモリに対するコアレスアクセスを行うことができ，メモリアクセスによる遅延を改善することができる[10]．

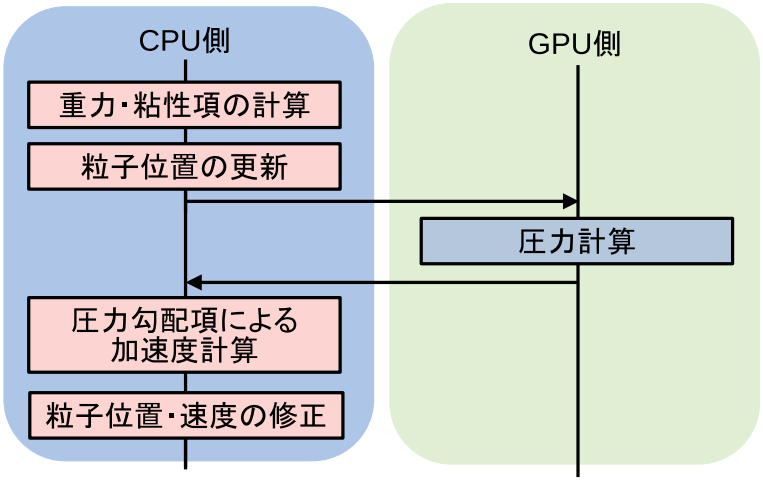


図 3–1 ： CUDAを用いたMPS法のフローチャート

3.1 CUDA

CUDAは，NVIDIA社が提供するGPU向けのC言語の統合開発環境であり，コンパイラやライブラリで構成される[22]．CUDAを用いることで，GPUの内部構造を意識しなくてもプログラミングが可能であるため，汎目的計算であるGPGPUを

容易に実現することが可能である．CUDA プログラムは，CPU 側で計算するコードと GPU 側で計算するコードの二つで構成される．一般的に，CPU 側のことをホスト側，GPU 側のことをデバイス側と呼ぶ．また，GPU デバイスで実行されるコードのことをカーネルと呼び，カーネルは GPU スレッドとして GPU でスケジュールされて実行される．CUDA プログラムの処理流れとしては，ホスト側からデバイス側に用いるデータをコピーし，カーネルを呼び出して必要な処理をデバイス側で行い，処理したデータをデバイス側からホスト側にコピーする．CUDA におけるほとんどの動作では，ホストはデバイスから独立した状態で動作でき，カーネルが起動するとすぐに制御がホストに戻るため，デバイス上でカーネルが実行されている間に CPU が他のタスクを実行することができる．また，CUDA プログラミングモデルは主に非同期であるため，GPU で実行される計算がホスト-デバイス間におけるデータ転送とオーバーラップする可能性がある．本節では，3.1.1 節，3.1.2 節で CUDA におけるメモリ階層とスレッド階層を述べる．次に，3.1.3 節，3.1.4 節で本研究で使用するアーキテクチャについて述べる．最後に，3.1.5 節，3.1.6 節で CUDA で高速化する上で必要な理論について述べる．

3.1.1 CUDA におけるメモリ管理

CUDA では，システムがホストとデバイスで構成されるため，それぞれ独立したメモリが必要である．CUDA は，GPU アーキテクチャのメモリ階層という抽象概念がある．図 3–2 に CUDA におけるメモリ階層を示す．図 3–2 より，各 GPU デバイスには，さまざまな目的に使用されるメモリが複数存在する．主によく使用されるメモリとしてグローバルメモリとシェアードメモリがある．グローバルメモリは，GPU において最も容量が大きく，データ転送遅延が大きいメモリである．シェアードメモリは，SM 内に存在し，グローバルメモリより容量は小さいが，データ転送遅延が小さい．多くのデバイスのデータアクセスは，グローバルメモリから開始され，多くの GPU アプリケーションは，メモリの帯域幅によって制限される．このため，グローバルメモリの帯域幅を最大限利用することが重要である．また，他のメモリとして GPU において最も高速なレジスタや，コンスタントメモリ，テクスチャメモリが存在する．

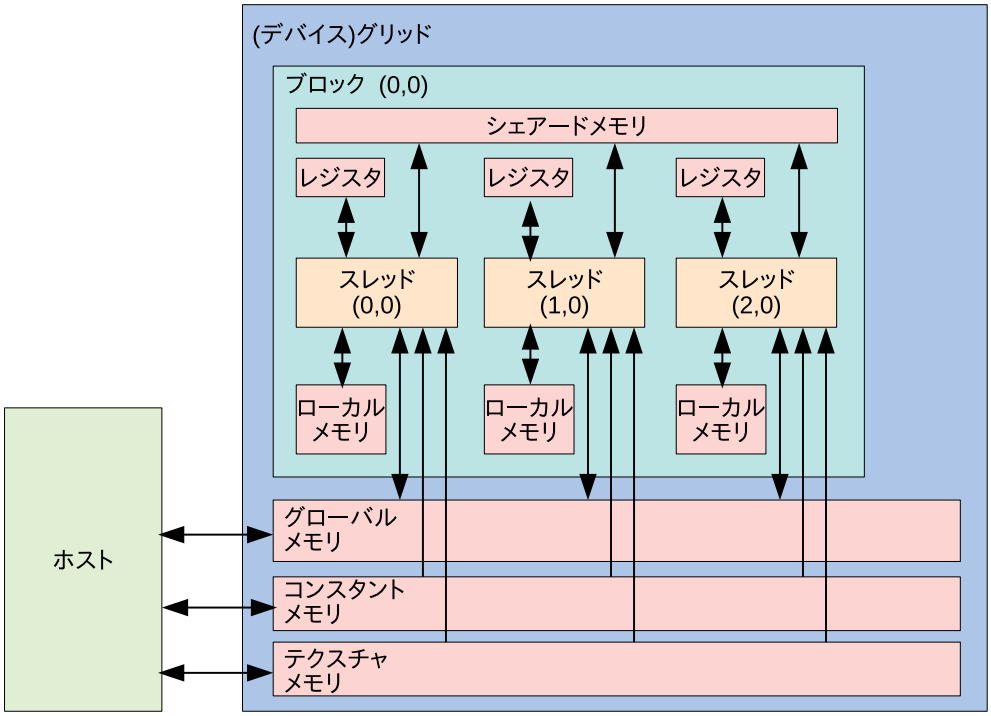


図 3–2 ： メモリ階層

3.1.2　CUDA におけるスレッド構成

CUDA では，ホスト側からカーネル関数を呼び出すと，実行制御がデバイスへ移動する．デバイスでは，大量のスレッドが生成され，カーネル関数によって規定されたステートメントが各スレッドによって実行される．CUDA では，スレッドの構成がスレッド階層として抽象化されている．スレッド階層は，複数のスレッドからなるブロックと複数のブロックからなるグリッドので構成される．図 3–3 にスレッド階層を示す．図 3–3 のように，1 つのカーネルによって生成されたすべてのスレッドをまとめてグリッドと呼ぶ．グリッド内のスレッドは，すべて同じグローバルメモリ空間を共有する．また，グリッドは，互いに協調して動作可能なスレッドグループであるスレッドブロックで構成される．スレッドブロックは，ブロックに属するスレッド間の同期が可能で，ブロック内で共有されるシェアードメモリが存在する．グリッドとブロックのサイズには，いくつか制限があり，主な制限要因として，レジスタやシェアードメモリのような利用可能な演算リソースがある．また，グリッドとブロックは，カーネル関数のスレッド階層を論理的に表すもので，演算リソースやメモリリソースはデバイスごとに異なる可能性があるが，スレッドを図 3–3 の

ような構成にすることで，同じコードを様々なデバイスで効率よく実行できる．

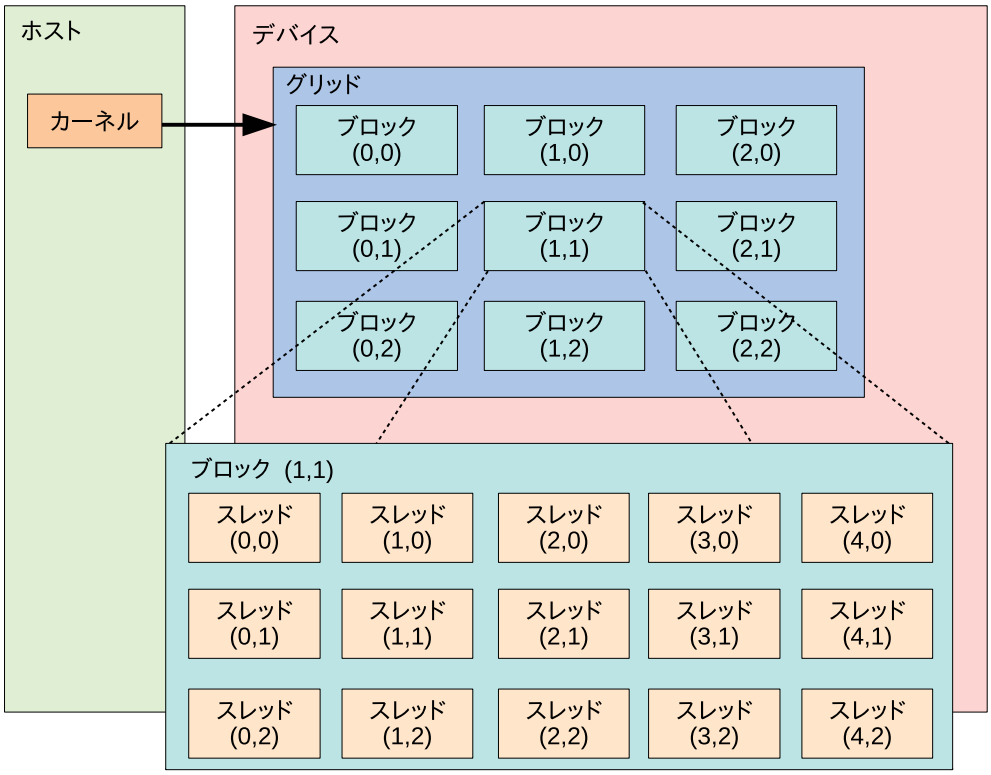


図 3–3 ： スレッド階層

3.1.3　GPU アーキテクチャ

GPU アーキテクチャは，SM の構成要素の配列を中心に構成されている．GPU の SM は，それぞれ何百ものスレッドを同時実行する．GPU には，それぞれ複数の SM が搭載されているため，1 つの GPU で大量のスレッドの同時実行が可能である．カーネルグリッドが起動すると，そのカーネルグリッドのスレッドブロックが利用可能な SM に分配された状態で実行される．スレッドブロック内のスレッドは，一度 SM に割り当てられると，その SM でのみ並列に実行する．また，1 つの SM に複数のスレッドブロックが割り当てられることもあり，それらのスレッドブロックは，利用可能な SM リソースに基づいてスケジュールされる．CUDA は，SIMT アーキテクチャを採用しており，スレッドが 32 個ずつ，ワープと呼ばれるグループにまとめられた上で実行する．ワープ内のスレッドは全て同じ命令を同時に実行し，スレッドは，それぞれ命令実行パスとレジスタ状態を持ち，独自のデータで現在の命

令を実行する．各 SM は，割り当てられたスレッドブロックを 32 個のスレッドからなるワープに分解し，利用可能なハードウェアリソースにスケジュールする．SIMT アーキテクチャは，同じ命令を複数の実行ユニットにブロードキャストすることで並列処理を行う．ワープ内のスレッドは，すべて同じプログラムアドレスで同時に開始されるが，個々のスレッドの振る舞いは異なっている可能性がある．スレッドブロックは，1 つの SM 上でスケジュールされ，実行が完了するまで SM 上に留まる．また，SM は同時に複数のスレッドブロックを実行できる．シェアードメモリとレジスタは，SM の貴重なリソースであり，シェアードメモリは，SM に割り当てられたスレッドブロックの間で分配され，レジスタはスレッドの間で分配される．スレッドブロック内のスレッドは，これらのリソースを通じて協調的に動作し，データをやり取りする．スレッドブロック内でワープをスケジュールする順番は特に無く，アクティブなワープの数は，SM のリソースによって制限される．例として，デバイスメモリから値の読み込みを待っているなどの何らかの理由でワープがアイドル状態になっている場合は，同じ SM に割り当てられているスレッドブロックから別の実行可能なワープをスケジュールできる．

3.1.4　Pascal アーキテクチャ

Pascal アーキテクチャは，NVIDIA 社が 2016 年に発表した GPU アーキテクチャである．一般的な Pascal アーキテクチャは，CUDA コア 32 個をワープという単位でまとめ，2 命令発行の命令バッファ，及び超越関数ユニット (SFU)8 個と合わせて，プロセッシングブロックと呼ばれる実行単位を設けている．プロセッシングブロックは，4 つを 1 組として，SM を構成する．各 SM には，テクスチャユニットが設置され，SM に PolyMorph エンジンを併設したものを TPC と呼ぶ．この TPC を 5 つまとめたものに Raster エンジンを加えたものを GPC とする．本研究で使用する NVIDIA TITAN X は，2016 年 8 月に発売された Pascal アーキテクチャが搭載された GPU である．NVIDIA TITAN X の SM は 28 個，各 SM が 128 個の CUDA コアを持ち，GPU 全体で 3,584 個の CUDA コアを持つ．テクスチャユニットは 224 個となる．GP102 のダイのトランジスタ数は 120 億個であり，各 SM 毎にジオメトリユニットを持つため，ジオメトリユニット数は 28 となる．ラスタライザなどを共有する GPC は 6 クラスタで，各クラスタ毎に最大 5 個の SM を備える．基本構成は，下位の GP104 と同じで，gp104 を拡大した構成となっている．NVIDIA TITAN X のメモリ種類は，GDDR5X であり，メモリ帯域は 480GB/s である．

NVIDIA TITAN X の GPU ダイは，「GP102」である．図 3–4 に全体構成図を示

す．図 3–4 より，GP102 のダイは，プロセッサクラスタである SM が 30 個で構成されている．また，メモリの DRAM インターフェイスは x32 が 12 個の 384-bit 幅である．NVIDIA TITAN X では，歩留まりを上げるために SM は 30 個のうち 2 個が無効にされており，28 個が有効である．このため，30 個の SM のうち 2 個まで不良があっても製品として出荷できる．最近の GPU を含むメニイコアプロセッサでは，物理的に搭載しているコア数よりも少ない数を有効にして製品化することが一般化している．

GP102 の SM の構成は，グラフィックス向けの GP10X シリーズに共通のものである．GP102 の SM 構成は，世代的には Pascal だが，Maxwell アーキテクチャの拡張となっている．SM の内部には，32 レーンのベクタユニットが 4 基あり，合計で 128 個の単精度積和演算の並列実行が可能である．NVIDIA 用語では，1 個の SM に 128 個の CUDA ユニットが搭載されている．また，個々の 32 個の CUDA コアに，8-way の Special Function Unit(SFU) とロード/ストアユニットが付属しており，SM 全体で 4-way のテクスチャユニットが 2 基搭載されている．

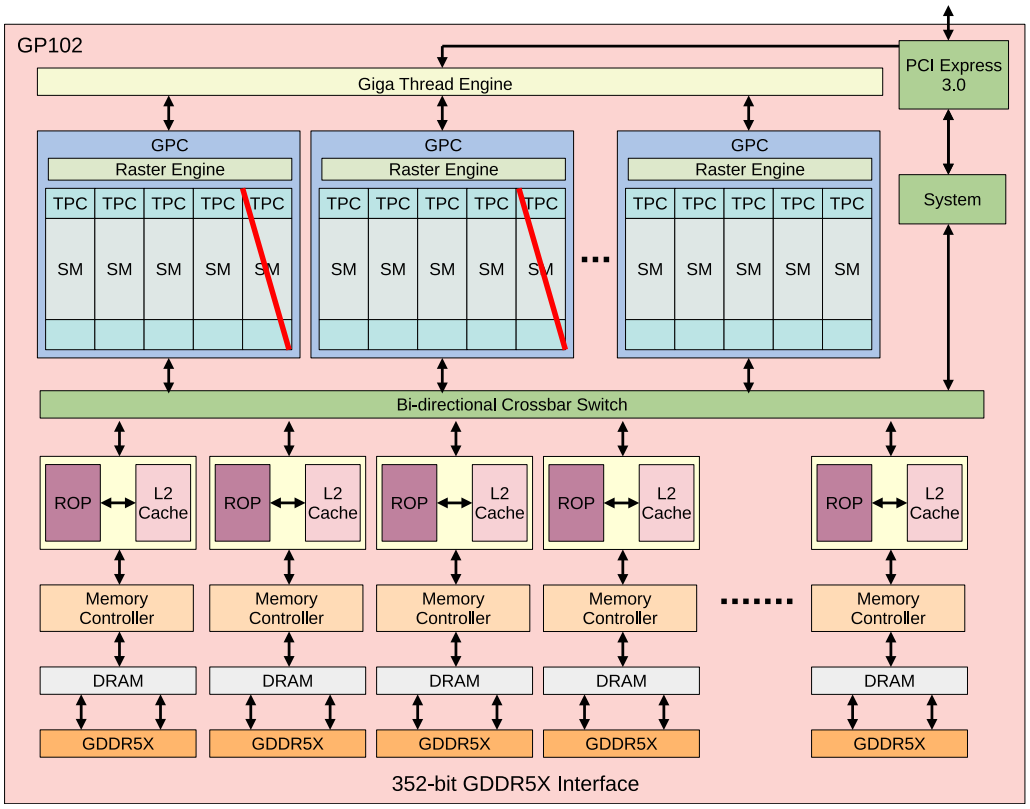


図 3–4 ： NVIDIA TITAN X の全体構成

3.1.5　ワープダイバージェンス

ワープは SM 内での基本的な実行単位であり，グリッドを起動すると，グリッドに含まれているスレッドブロックが SM に分配される．また，スレッドブロックが SM にスケジュールされると，スレッドブロックがさらにワープに分解される．ワープは，32 個の連続するスレッドで構成され，ワープ内のスレッドは全て同じ命令を実行し，その処理にはスレッド独自のデータが使用される．また，ハードウェアがスレッドブロックに割り当てるワープの数はそのつど異なり，1 つのワープが異なるスレッドブロックにまたがることはない．スレッドブロックのサイズがワープサイズの倍数ではない場合，最後のワープに含まれているスレッドの一部はアクティブ化されず，アクティブ化されないスレッドは，使用されていなくてもレジスタなどの SM リソースを消費する．

CPU における条件分岐などの制御では，分岐予測を実行するための複雑なハードウェアが存在するが，GPU は比較的シンプルなデバイスであり，複雑なハードウェアは組み込まれていない．ワープ内のスレッドは，全て同じサイクルでまったく同じ命令を実行しなければならない．このため，命令を実行するスレッドが 1 つでも存在する場合は，そのワープ内のすべてのスレッドがその命令を実行しなければならないが，条件分岐などでスレッドごとの評価が違う場合など，同じワープ内のスレッドが異なる命令を実行することをワープダイバージェンスと呼ぶ．図 3–5 にワープダイバージェンスの例を示す．図 3–5 のように，if 文などの制御命令が必要となるプログラムを実行すると CUDA は，2 ステップに渡って命令を実行する．まず，制御命令の判定が真となる場合に実行するべき命令を実行し，次のステップで制御命令の判定が偽となる場合に実行するべき命令を実行する．最後に，必要な命令の実行結果のみをマスクを実行して取り出す．このように，ワープダイバージェンスは制御命令の複雑性によって不要な命令を実行し，パフォーマンスを大幅に低下させる可能性がある．このため，パフォーマンスを最適化するためには，できるだけ同じワープ内で複数の実行パスを使用することを避けるようなアルゴリズムの設計が必要である．

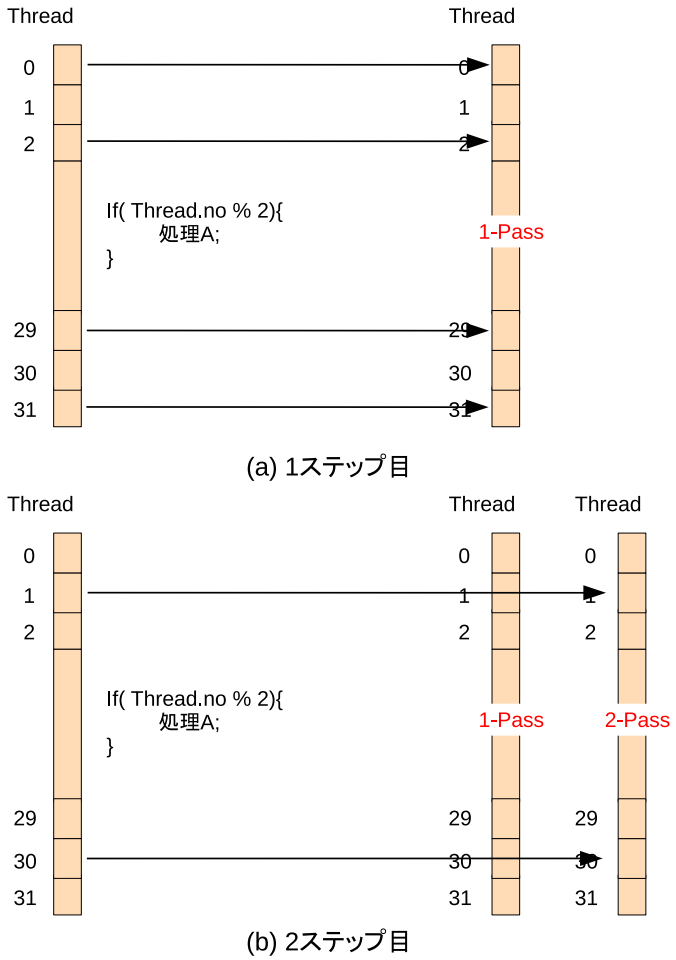


図 3–5 ： ワープダイバージェンス

3.1.6　アライン/コアレスアクセス

データの読み取りと書き込みにおいて，最適化するためには，メモリアクセス操作が特定の条件を満たす必要がある．CUDA におけるメモリ操作は，ワープごとに発行されるため，メモリに関連する命令を実行する際，ワープ内のスレッドはそれぞれロードとストアに使用するメモリアドレスを提供する．その後，ワープにおける 32 個のスレッドが協調し，リクエストされたアドレスを一つのメモリアクセスにまとめて処理される．このため，メモリ操作の最適化には，デバイスメモリのアクセスに関して，アラインメモリアクセスとコアレスメモリアクセスを満たす必要がある．アラインメモリアクセスは，デバイスメモリトランザクションの最初のアドレスが，トランザクションの処理に使用されているキャッシュ粒度の倍数である場

合に発生する．コアレスメモリアクセスは，ワープ内の 32 個のスレッドがすべて隣り合ったメモリチャンクにアクセスする場合に発生する．アライン/コアレスメモリアクセスは，理想的なアクセスであり，アラインされたメモリアクセスから始まるメモリチャンクにワープがアクセスする．図 3–6 にアライン/コアレスメモリアクセスを示す．図 3–6 のように本例のアクセスは，CUDA スレッドの実行単位は，ワープ単位であるため，ワープ内の 32 スレッドが連続した領域をそれぞれロードすることでコアレスアクセスとなる．また，全てのスレッドが 128 バイトの範囲内のデータにアクセスしていることから一度のアクセスでメモリアクセスが完了できる．このように，データの読み書きにおいてメモリ操作をアラインとコアレスの両方になるような構成にすることが重要である．

一方，ワープ内のスレッドが順にアクセスしないような場合はコアレスアクセスとならずにメモリアクセスの先頭アドレスから個別にアクセスする．このような場合は，メモリトランザクションが複数回必要になる．加えて，メモリアクセス単位のアクセス範囲を超えるような場合はアラインアクセスにならず，各メモリアクセスの先頭となるアドレスから個別にアクセスする．この場合も，アンコアレスアクセスと同時に複数回のメモリトランザクションが必要となる．図 3–7 にミスアライン/アンコアレスメモリアクセスを示す．図 3–7 のアクセスは，ワープ内の 32 スレッドが連続しない領域を読み込むため，複数回のアクセスが行われることにより効率の悪いアクセスとなる．

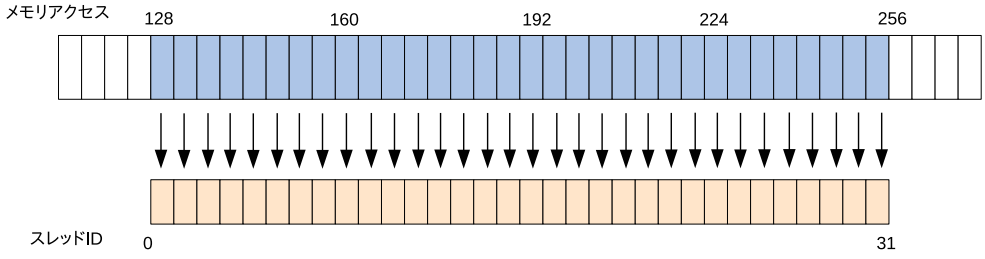


図 3–6 ： アライン/コアレスメモリアクセス

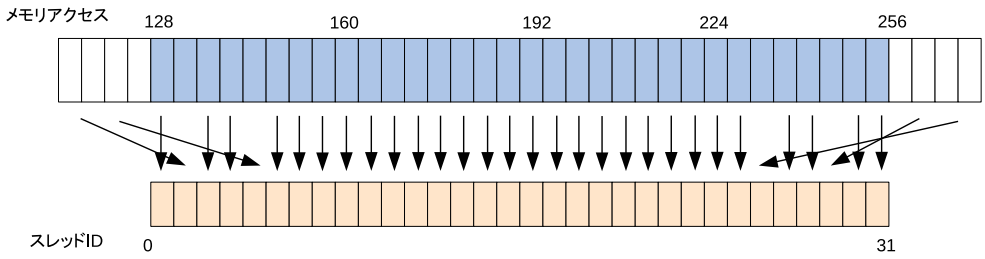


図 3–7 ： ミスアライン/アンコアレスメモリアクセス

3.2　疎行列格納形式

疎行列を密行列としてメモリに格納し，計算処理を行う場合，疎行列は零要素が多いため，無駄な計算処理やメモリ使用が発生する．これらの無駄な部分を改善するため，疎行列から零要素は省き，非零要素だけをメモリに格納する疎行列格納形式がある．疎行列格納形式はこれまでに様々な形式が提案されており，代表的な形式として COO 形式，CRS 形式，ELL 形式，JDS 形式，HYB 形式, SELL 形式 [23] などがある．CUDA は 3.1.6 節より，メモリアクセス回数によって実行時間が変化する．それぞれの格納形式は零要素の削減率や配列数が異なり，零要素を完全に削減すると配列数やメモリアクセス回数が多くなり，逆に配列数やメモリアクセス回数を少なくすると零要素が完全に削減できない場合がある．このように，疎行列格納形式には無駄な演算とメモリアクセスのトレードオフが生じるため，多くの MPS 法では，問題の特性に応じた形式を採用する．

3.2.1　COO 形式

COO 形式は，疎行列の非零要素とその非零要素がある場所の行番号と列番号をそれぞれ格納する形式である．図 3–8 に COO 形式の格納例を示す．図 3–8 の配列 val は非零要素の値，配列 col は非零要素の列番号を格納する配列である．また，配列 row は，非零要素の行番号を格納している．COO 形式は，零要素を全て削減できるため，零要素による無駄な演算が発生しない．また，配列 col と配列 row を参照して非零要素を検出できるため，密行列格納に比べるとメモリアクセス回数は少ない．COO 形式は，行番号と列番号をそのまま格納しているため，他の形式とメモリアクセス回数を比較すると，1 行ごとの非零要素にアクセスしづらいという点がある．

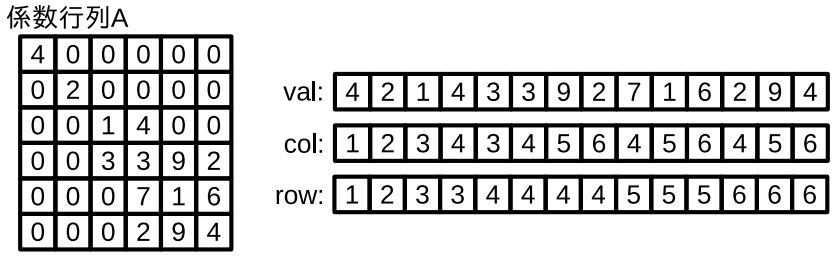


図 3–8 ： COO 形式

3.2.2　CRS 形式

CRS 形式は，疎行列を行方向に走査し，零要素を省いて非零要素を格納する格納形式である [24]．図 3–9 に CRS 形式の格納例を示す．図 3–9 の配列 val は非零要素の値，配列 col は非零要素の列番号を格納する配列である．また，配列 row は，val と col における各行先頭の非零要素の添字を格納している．CRS 形式は，零要素に対する無駄な演算が発生しないことが利点である．これに対する欠点としては，非零要素の分布によって 1 行ごとの非零要素数に差があると，スレッドごとの処理時間が異なることである．また，ELL 形式に比べると配列数が多いため，ELL 形式よりメモリアクセス回数が多くなる [25][26]．

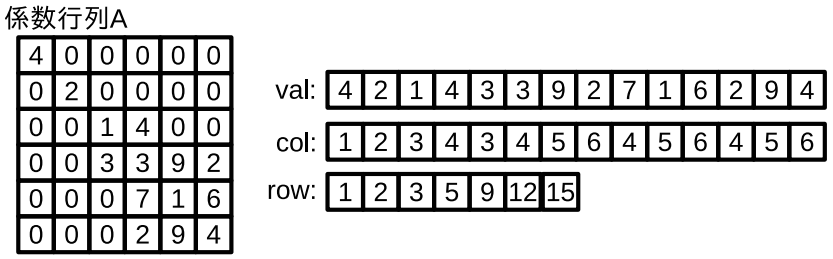


図 3–9 ： CRS 形式

3.2.3　ELL 形式

ELL 形式は，疎行列を n × k の密行列に格納する [27]．ここで n は元の疎行列の一辺のサイズ，k は疎行列 1 行あたりの非零要素数の最大値である．また，1 行あたりの非零要素が k に満たない場合，その行には 0 をパディングする．図 3–10 に ELL

形式の格納例を示す．図 3–10 の配列 val は非零要素の値，配列 col は非零要素の列番号を格納する配列である．ELL 形式は，欠点としてパディングにより計算処理に不要な演算が発生する．これに対する利点は，列数がすべて揃っており，並列化する際に行ごとの要素にコアレスアクセスができ，配列数も少ないため，CRS 形式や JDS 形式と比べてメモリアクセス回数が少ないことである．

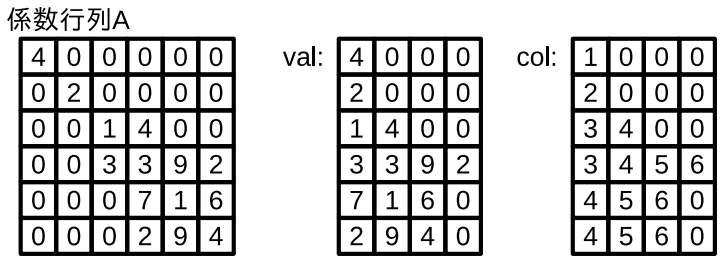


図 3–10 ： ELL 形式

3.2.4 JDS 形式

JDS 形式は，行列を行方向に詰め，1 行あたりの非零要素が多い順に並び替える．その後，列方向に走査し 4 つの配列と 1 つの変数に格納する．図 3–11 に JDS 形式の格納例を示す．図 3–11 の配列 val は非零要素の値，配列 col は非零要素の列番号，perm は非零要素数でソートした行番号，ptr は列のオフセット，max は，最も非零要素数が多い行の非零要素数である．JDS 形式は，配列 val と配列 col にアクセスする際，連続する番号のスレッドが連続するメモリ領域にアクセスしているため，1 回のメモリアクセスで複数のスレッドが同時に参照でき，コアレスアクセスとなり，メモリアクセス回数を削減できる [12]．欠点としては，行ごとの非零要素が多い順にソートするため，行の最大要素数と最小要素数の差が大きいと，スレッドごとの処理に差ができ，ワープダイバージェンスによるオーバーヘッドが発生する．このオーバーヘッドは，非零要素数の差に比例する．

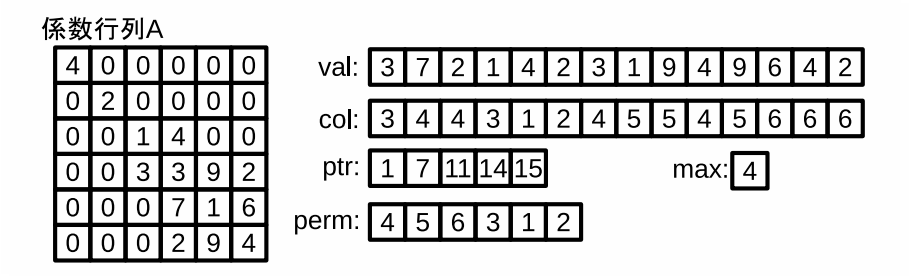


図 3–11 ： JDS 形式

3.2.5　HYB 形式

HYB 形式は，基本的に ELL 形式で格納し，ELL 形式で埋まらない場所についてのみその非零要素を COO 形式で持つような形式である．HYB 形式は，特定の行や列に非零要素が固まっている場合，少し散らばっているような場合に有効である．どの列までを ELL 形式で格納するのかは任意であり，使用する行列に合わせて適切な列で区切る必要がある．図に 5 列までを ELL 形式として格納した HYB 形式を示す．図のように，ELL 形式でパディングが多い部分を COO 形式で格納することにより，無駄なパディングを削減することが出来る．欠点として，列の区切りは使用者が決定しなければならないため，区切る列として最適な列を検討しなければならないことが挙げられる．

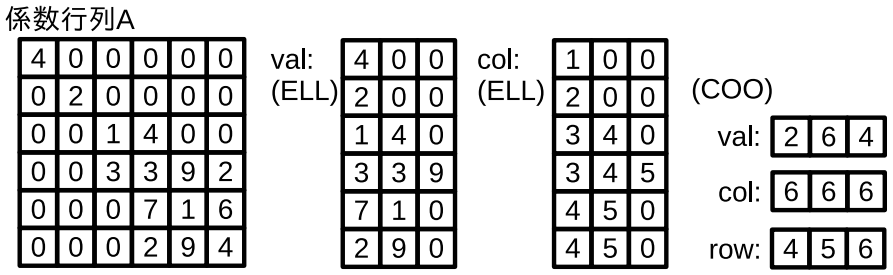


図 3–12 ： HYB 形式

3.3　CUDA ライブラリ

CUDA ライブラリは，NVIDIA 社やサードパーティから提供されているディープラーニングや画像処理など特定分野での利用を想定したライブラリである．これらのライブラリは，使い勝手の良い高度な API の提供を目的として設計されており，複雑なアプリケーションの構成要素として使用できる．また，データフォーマットも標準化されているため，既存のアプリケーションへの組み込みが容易である．本研究では，線形代数計算ができる cuBLAS ライブラリ，疎行列線型代数計算ができる cuSPARSE ライブラリを圧力計算における CG 法と行列ベクトル積で使用している．本節では，3.3.1 節で CG 法で主に用いている cuBLAS ライブラリについて述べる．その後，3.3.2 節で疎行列ベクトル積に用いている cuSPARSE ライブラリについて述べる．

3.3.1　cuBLAS ライブラリ

cuBLAS ライブラリは，線形代数関数を集めたライブラリである．cuBLAS ライブラリは，既存の線形代数ライブラリである BLAS に基づいて設計されている．cuBLAS の関数は，密ベクトルと密行列の演算のみに最適化されており，BLAS と同様に演算対象となるデータの種類に基づいて複数のカテゴリに分類される．表 3–1 に cuBLAS の分類を示す．表 3–1 より，本研究では MPS 法の圧力計算中にある CG 法を Level1，Level2 の cuBLAS 関数で実装している．

表 3–1 ： cuBLAS の分類

Level1	ベクトル加算など
Level2	行列とベクトルの演算
Level3	行列同士の演算

3.3.2　cuSPARSE ライブラリ

cuSPARSE ライブラリは，汎用の疎行列線形代数関数が幅広く実装されているライブラリである．cuBLAS ライブラリは，密行列のみに最適化されているが，cuSPARSE ライブラリは疎行列のデータフォーマットにも対応している．また，cuSPARSE ライブラリも cuBLAS ライブラリと同様に複数のカテゴリに分類されている．表 3–2

に cuSPARSE の分類を示す．表 3–2 より，本研究で扱う行列ベクトル積は Level2 に存在する関数を用いる．cuSPARSE ライブラリの疎行列におけるデータ格納フォーマットは，3.2 節の疎行列格納形式である．このため，cuSPARSE ライブラリの演算関数を使用する場合には，計算に用いる疎行列を適用する疎行列格納形式で格納する．このことから，cuSPARSE ライブラリには，疎行列格納形式を変換できる関数が実装されている．

表 3–2 ： cuSPARSE の分類

Level1	密ベクトルと疎ベクトルに対する操作
Level2	疎行列と密ベクトルに対する操作
Level3	疎行列と密行列に対する操作

第4章
CUDAを用いたMPS法における疎行列格納形式の動的選択

MPS法における圧力計算部分で生成される対称疎行列は，CRS形式，ELL形式，JDS形式など様々な疎行列格納形式が利用される[8]．MPS法の圧力計算における係数行列は，時間ステップごとに流動する流体粒子とその影響範囲にある粒子との相互作用力に基づいて生成され，非零要素の位置も粒子の移動に合わせて時間ステップごとに変化する．このため，MPS法の係数行列における格納形式は，適する形式と適さない形式が存在する可能性がある．図4–1，図4–2に2つの粒子配置を示す．行列中の赤い部分は非零要素，緑の部分はパディングである．図4–1は，係数行列において1行ごとの非零要素数にあまり偏りが無く，ELL形式によるパディングも少ない．このため，図4–1は，CRS形式よりパディングが少なく，コアレスアクセスによる恩恵が得られるELL形式の方が適するといえる．対して，図4–2は，係数行列における1行ごとの非零要素数の差があり，ELL形式におけるパディングが大きいため，コアレスアクセスによる恩恵より，パディングによる無駄な計算が実行時間に悪影響を与える可能性がある．このため，図4–1は,無駄な計算が削減できるCRS形式が適するといえる．格納する形式が適切でないと無駄な計算や無駄なメモリ確保が発生するため，MPS法の係数行列に対しても様々な格納形式の中から適切な格納形式を決める必要がある．従来のMPS法では，問題の特性に応じて主にELL形式，CRS形式，JDS形式から一つの形式を決定して用いる[8][9]．

これに対して，CUDAを用いた疎行列ベクトル積において行列形状に合わせて適する格納形式を用いることが有効である[11]．このことから，MPS法の圧力計算に使用する係数行列の非零要素位置に合わせて疎行列格納形式を動的に選択することにより，無駄な計算や無駄なメモリアクセス回数の削減による計算時間の高速化が期待できる．そこで，本研究では，CUDAを用いたMPS法の圧力計算で疎行列格納形式を動的に選択し，MPS法の実行時間を高速化する．提案手法では，MPS法の粒子配置及び行列形状は1ステップで大きく変化しないということから，動的選択後に次ステップで用いる格納形式を決定する．また，次ステップの格納形式決定

と CUDA による圧力計算の行列ベクトル積は処理するアーキテクチャが異なり，処理に依存がない．このため，本手法では，CUDA による圧力計算中に CPU 側で次ステップの格納形式を決定する．これにより格納形式の選択時間を隠蔽する．また，従来の格納形式選択に用いる選択条件パラメータは，係数行列の規模や形状に大きく左右される．このため，従来のパラメータのままでは，適する格納形式が選ばれない可能性がある．本手法では，より最適な格納形式が選ばれるように，格納形式選択の条件パラメータを調整する．

本章では，4.1 節で従来の CUDA を用いた疎行列ベクトル積における疎行列格納形式の動的選択手法について述べる．次に，4.2 節で提案手法である CUDA を用いた MPS 法における格納形式の動的選択手法について述べる．最後に 4.3 節で本手法で実装した疎行列格納形式の変換について述べる．

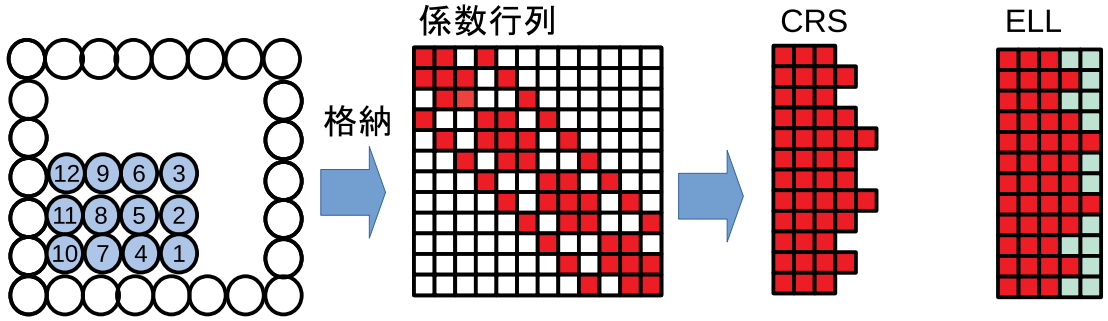


図 4–1 ： 初期配置による行列形状における格納形式

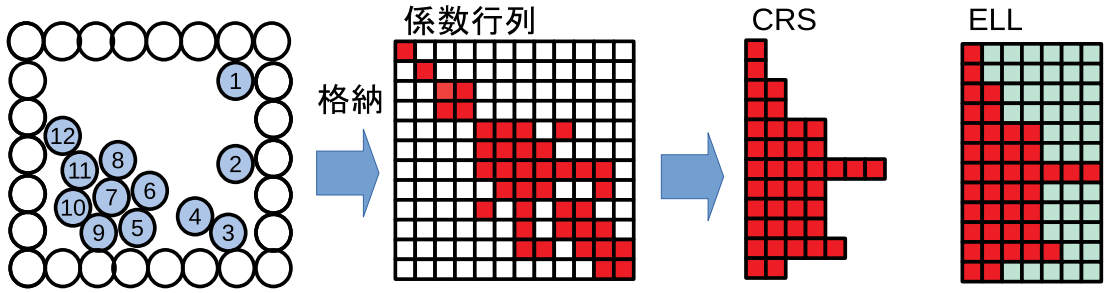


図 4–2 ： 数ステップ後の行列形状における格納形式

4.1 CUDA を用いた疎行列ベクトル積の疎行列格納形式動的選択

格納形式を動的選択するためには，各ステップの行列形状から適切な格納形式を判別する条件が必要である．従来の CUDA を用いた疎行列ベクトル積では，疎行列格納形式である CRS 形式，ELL 形式，JDS 形式を動的に選択する [11]．従来の格納形式の判別では，係数行列の 1 行あたりの非零要素のばらつき B と係数行列全体の非零要素率 R を用いる．式 (4–1)，式 (4–2) に非零要素のばらつき B および非零要素率 R を算出する式を示す．式 (4–1) より，非零要素のばらつきは，1 行あたりの非零要素数を用いて算出する．このため，1 行における非零要素数の多い行と少ない行に極端な差がある場合，1 行あたりの非零要素数の平均が小さくなり，ばらつきが大きくなりやすい．また，式 (4–2) より，非零要素数は，一つの粒子に対する近傍の粒子が増加すると，それに伴い非零要素数も増加する．このことから，粒子が圧縮された配置ではばらつきが小さく非零要素率が大きくなり，粒子が拡散する配置ではばらつきが大きく非零要素率は小さくなる．また，ばらつきと非零要素率は係数行列における非零要素数から算出されるため，係数行列の規模や形状に大きく左右する．式 (4–3) の久保田らの提案した従来の格納形式選択条件パラメータは，一辺が 10 万以上の行列サイズが大きい行列を元に決定する [11]．これに対して，本研究の粒子配置は従来研究に比べ行列サイズが小さいため，算出するばらつきと非零要素率の値が全体的に小さく，元の選択条件パラメータでは適切な格納形式が選択されない可能性がある．このことから，動的選択手法では，より適した選択パラメータに調整する必要がある．このことから，本研究では，格納形式の選択条件のパラメータをより適したパラメータに調整する．本手法では，評価で用いる粒子配置から全ステップのばらつきと非零要素率を算出し，最大値，最小値，平均値や値の推移から最適と考えられる選択条件パラメータを決定する．

$$B = \frac{\text{1 行あたりの非零要素数の最大値}}{\text{1 行あたりの非零要素数の平均}} \tag{4–1}$$

$$R = \frac{\text{非零要素数}}{\text{係数行列の全要素数}} \tag{4–2}$$

$$\begin{cases} \text{ELL 形式} : (B < 2.0 \text{ かつ } R < 0.048) \\ \text{CRS 形式} : (B < 8.0 \text{ または } R \geq 0.048) \\ \text{JDS 形式} : (\text{上記以外}) \end{cases} \tag{4–3}$$

4.2 CUDA を用いた MPS 法の格納形式動的選択手法

本研究では，無駄な計算とメモリアクセス回数のトレードオフの関係から，動的選択を行うことでそれぞれの利点をあわせもった実装が可能であると考えられ，従来の研究でも用いられている CRS 形式，ELL 形式，JDS 形式を選択する格納形式とする．適切な格納形式を判別する条件は，従来の動的選択 [11] におけるパラメータを参考にし，式 (4–1)，式 (4–2) のばらつきと非零要素率を用いて格納形式を判別する．これらのパラメータは，非零要素の係数行列から算出するパラメータを取得することができる．また，MPS 法の係数行列は, 密行列で格納すると無駄な零要素が多く格納される．このため, 行列における非零要素の値と列, 行番号を格納する実装が簡単な COO 形式で係数行列を格納する．図 4–3 に格納形式選択のフローチャートを示す．図 4–3 のより，格納形式の選択の処理は，まず COO 形式で係数行列を作成し，COO 形式から条件パラメータより選択されるそれぞれの格納形式に変換する．このため，格納形式の判別にかかる時間や格納形式を変換する時間のコストが発生する．本手法では，粒子配置が 1 ステップで大きく変動しないことを利用して，係数行列に適する格納形式を次のステップで生成する．図 4–4 に時間ステップごとの動的選択におけるフローチャートを示す．図 4–4 において，前ステップの動的選択後に，次ステップの格納形式を決定しておき，次ステップで前ステップで決定した格納形式を用いる．また，次ステップの格納形式を決定する処理と CUDA による圧力計算は，処理に使用するアーキテクチャが異なるため依存性がない．そこで，本手法では，CUDA による圧力計算中に次ステップの格納形式を決定することで，格納形式選択にかかる時間を隠蔽する．図 4–5 に選択時間を隠蔽するフローチャートを示す．図 4–5 では，格納形式選択にかかる時間より CUDA による圧力計算における処理時間のほうが多くかかるため，格納形式を選択する時間が隠蔽できると考えられる．本研究は，MPS 法における格納形式の動的選択による高速化が目的である．このため，MPS 法における格納形式の変動と格納形式切り替えによる時間短縮を踏まえた上で，格納形式動的選択の評価をする必要がある．

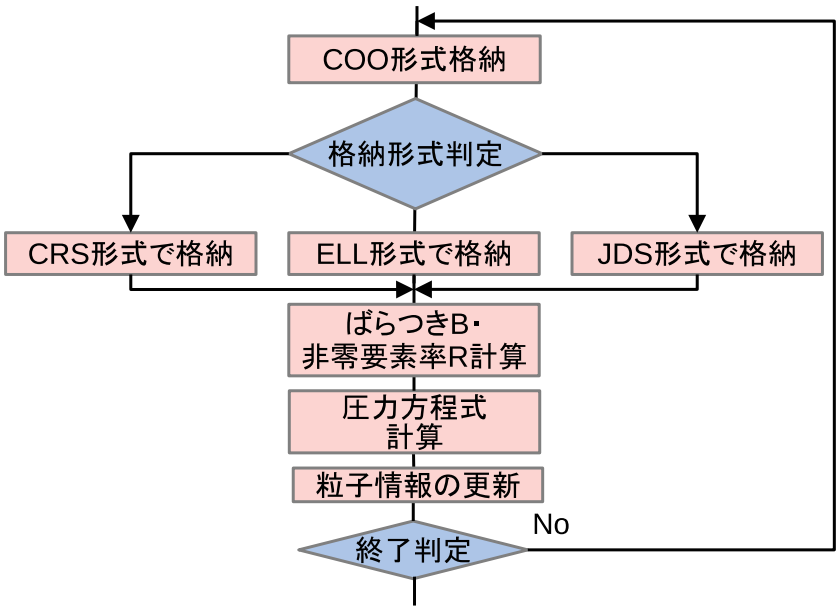


図 4–3 ： 格納形式選択のフローチャート

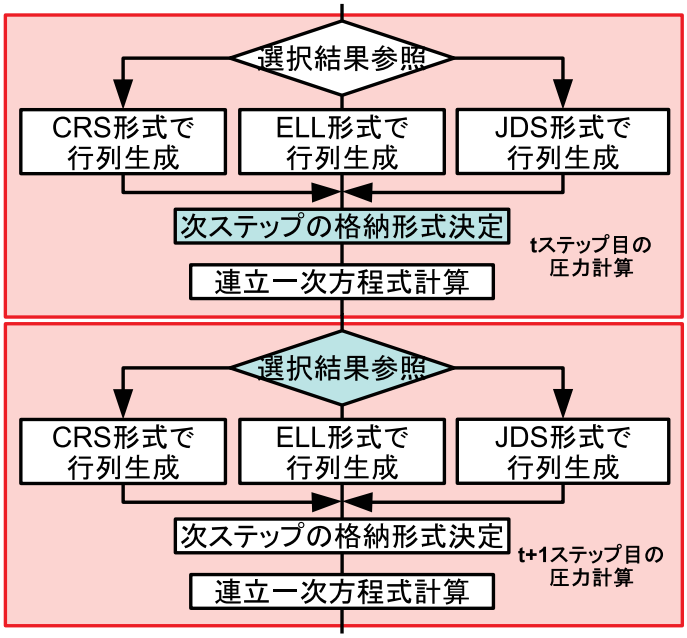


図 4–4 ： 時間ステップごとの格納形式選択のフローチャート

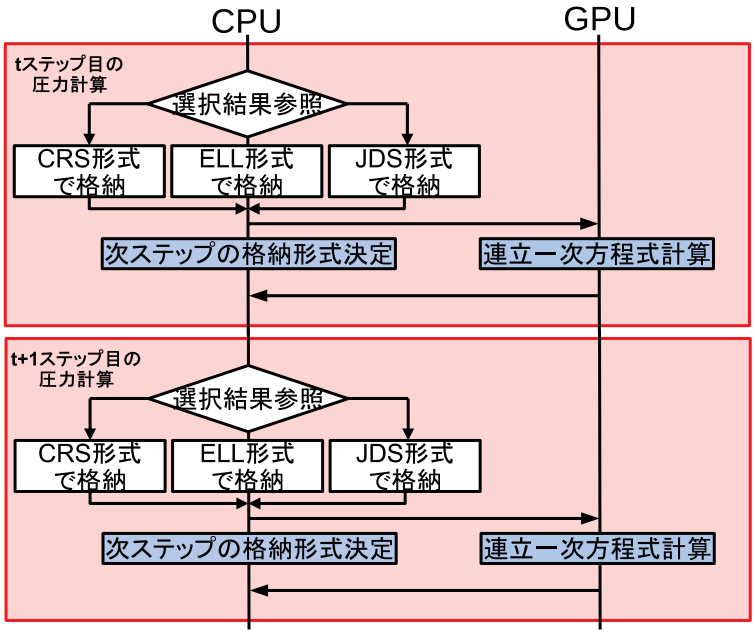


図 4–5 ： 格納形式選択時間の隠蔽

4.3　疎行列格納形式の変換

本研究では，ELL 形式，JDS 形式における要素格納の実装方法として，CRS 形式からの変換を採用する．図 4–6 にそれぞれ COO 形式から ELL 形式，JDS 形式へ変換するフローチャートを示す．JDS 形式は，ELL 形式と比較して格納する行列の行要素を 1 行あたりの非零要素が多い順に並び替えるため，ソート前の行番号を配列として保存する．

4.3.1　COO 形式から ELL 形式への変換

COO 形式から ELL 形式の変換では，COO 形式から変換した CRS 形式で確保した配列 row を参照して各行列の非零要素数を行ごとにカウントした配列 ellnnz を用意する．図 4–7 に CRS 形式から配列 ellnnz と配列 ellptr の生成例を示す．図 4–7 より，配列 ellnnz は配列 row の要素ごとの差分をとることで取得できる．また，列方向に走査して列のオフセットである配列 ellptr を取得する．ELL 形式は，行ごとの最大非零要素数が最大列数となり，行ごとの非零要素数が，最大要素数より少ない場合は，パディングが挿入される．本研究で用いる ELL 形式におけるパディングの

数値は 0 としている．このため，ELL 形式でおける要素えを格納する配列 ellval と列番号を格納する配列 ellcol は，あらかじめ 0 で初期化する．その後，最大列数と粒子数を参照し，列のオフセットを取得し，それを参照しながら COO 形式の要素と列番号を ELL 形式の配列に格納する．図 4–8 に ELL 形式への要素格納例を示す．図 4–8 では，配列 row を参照して COO 形式の配列 val から格納する要素を決定する．格納する場所は配列 ellptr を参照して決定し，配列 val から要素を格納する．その後，ループ変数を利用し，全ての要素分繰り返すことで格納形式の変換を行う．

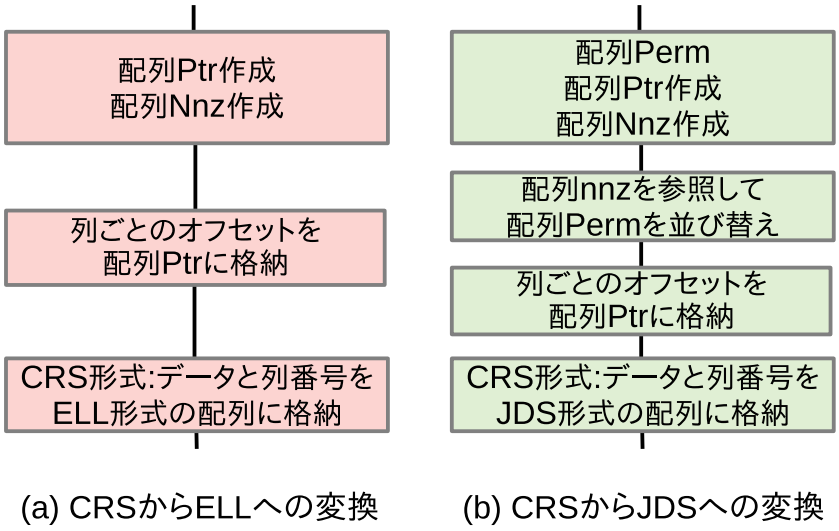


図 4–6 ： ELL 形式と JDS 形式の変換フローチャート

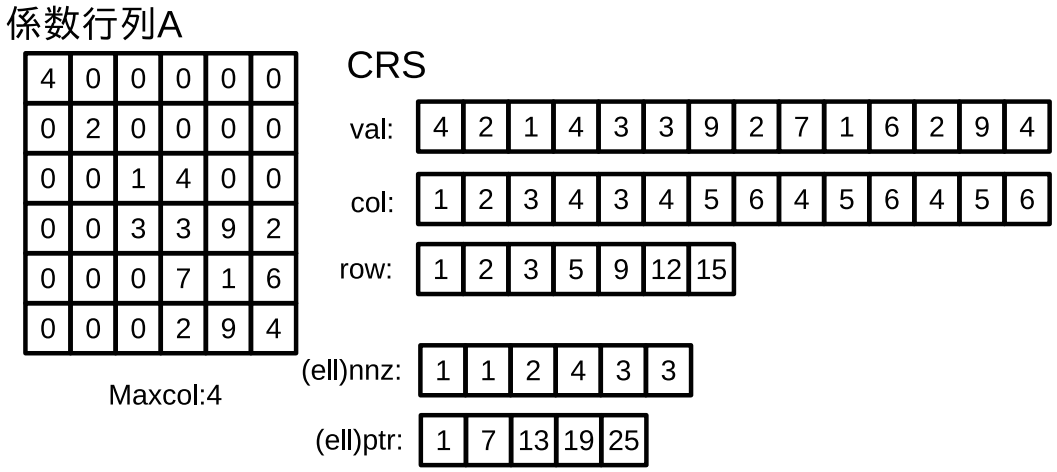


図 4–7 ： ELL 形式における配列 ellnnz と配列 ellptr の生成例

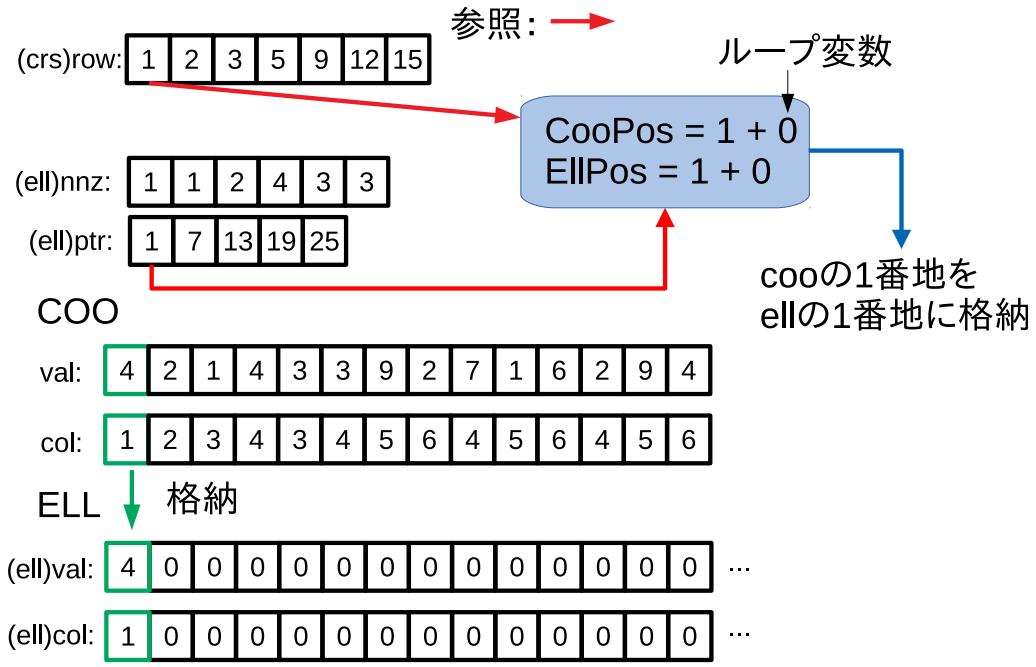


図 4–8 ： ELL 形式への要素格納例

4.3.2　COO 形式から JDS 形式への変換

図 4–6(b) より，COO 形式から JDS 形式へは，JDS 形式に変換することで必要になるソート前の行番号を保存する配列 jdsperm を作成する．また，ELL 形式と同様に，CRS 形式で確保した配列 row を参照して各行列の非零要素数を行ごとにカウントした配列 jdsnnz を用意する．図 4–9 に CRS 形式から配列 jdsperm と配列 jdsnnz の生成例を示す．図 4–9 より，CRS 形式で確保した配列 row は行ごとのオフセットを格納する配列であるため，連続した要素の差を求める事で行ごとの非零要素数を算出することが出来る．その後，配列 jdsnnz に格納されている行ごとの非零要素数を参照し，配列 jdsperm を行ごとの非零要素が多い順に並び替える．並び替えの際，非零要素数が同数である場合には，行番号の小さい順に格納する．本研究では，ソートプログラムとして qsort ライブラリを使用する．並び替え後は，配列 nnz の値を参照することで，列のオフセットを取得し，それを参照しながら COO 形式の要素と列番号を JDS 形式の配列に格納する．図 4–10 に JDS 形式への要素格納例を示す．図 4–10 では，配列 perm と配列 row を参照して COO 形式の配列 val から格納する要素を決定する．格納する場所は配列 ellptr を参照して決定し，配列 val から要

素を格納する．その後，ループ変数を利用し，全ての要素分繰り返すことで格納形式の変換を行う．

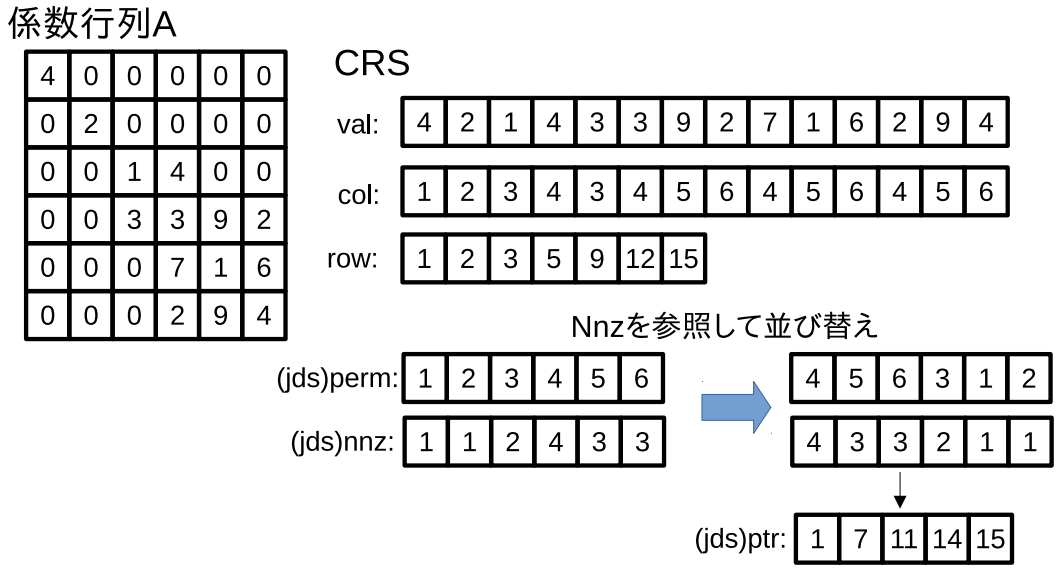


図 4–9 ： CRS 形式から JDS 形式への変換例

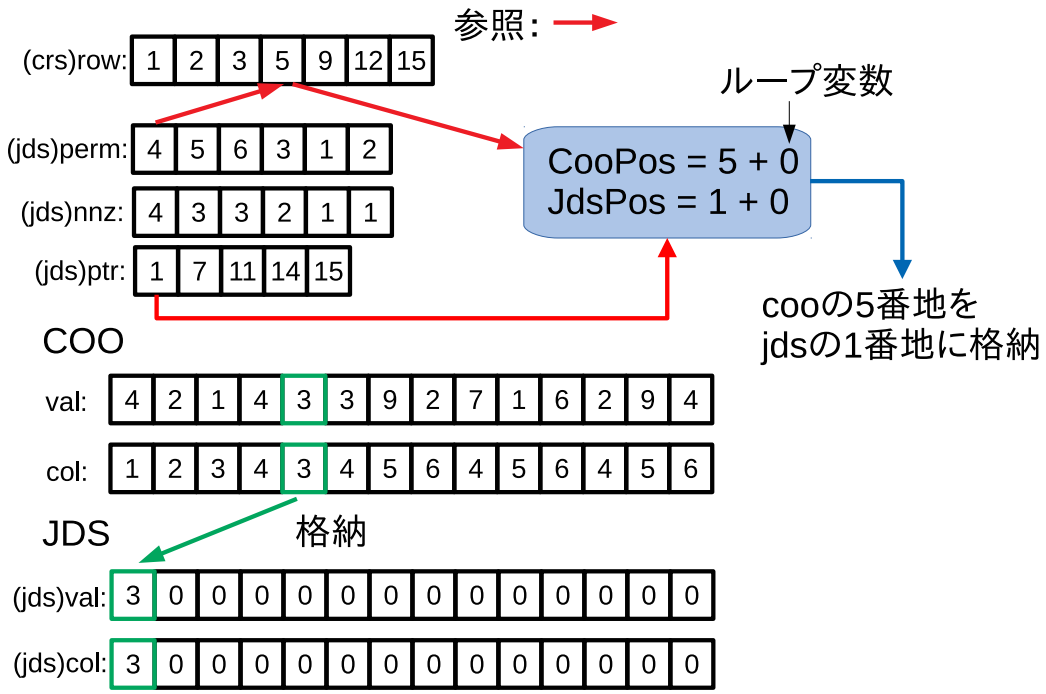


図 4–10 ： JDS 形式への要素格納例

第5章
評価

本研究では，CUDA を用いた MPS 法の圧力計算中における行列ベクトル積に対して疎行列格納形式の動的選択の有効性を確認するために，疎行列格納形式の動的選択が CUDA を用いた MPS 法の実行時間に与える影響を評価する．本研究における予備評価では，提案手法が MPS 法の高速化に有効であるかを確認するため，疎行列ベクトル積における短縮時間の評価，動的選択におけるパラメータの設定，選択時間隠蔽による実行時間の評価を行う．これらの予備評価を踏まえ，本評価では，MPS 法全体の処理時間を測定し，CUDA を用いた MPS 法に疎行列格納形式の動的選択を用いた行列ベクトル積の有効性を確認する．本章では，5.1 節で本研究で使用する評価環境を述べる．次に，5.2 節に予備評価について述べる．それぞれの予備評価においては，5.2.1 節で疎行列ベクトル積における短縮時間，5.2.2 節で動的選択におけるパラメータの設定，5.2.3 節で選択時間隠蔽による実行時間の評価について述べる．最後に 5.3 節で疎行列格納形式の動的選択を実装した CUDA を用いた MPS 法における実行時間の評価について述べる．

5.1 評価環境

表 5–1 に本研究で使用した評価環境を示す．表 5–1 より，本研究で用いる GPU は，NVIDIA TITAN X である．NVIDIA TITAN X は，3.1.4 節に示す通り，Pascal アーキテクチャが搭載されている．また，SM 数は 28，各 SM が 128 個の CUDA コアを持ち，総 CUDA コア数は 3584 である．この GPU 単体のメモリ容量は 12GB である．

表 5–1 ： 評価環境

CPU	Xeon E5-2667v2
GPU	NVIDIA TITAN X
CUDA version	10.1

5.2　予備評価

本研究における予備評価では，提案手法が MPS 法の高速化に有効であるかを確認するため，疎行列ベクトル積における短縮時間の評価，動的選択におけるパラメータの設定，選択時間隠蔽による実行時間の評価を行う．疎行列ベクトル積における短縮時間の評価では，MPS 法が生成する疎行列においてそれぞれ CRS 形式，ELL 形式，JDS 形式の CUDA を用いた行列ベクトル積の実行時間を測定し，求解の実行時間から格納形式ごとの実行時間に差があるのかを確認する．動的選択におけるパラメータの設定では，本研究における粒子配置から，より適切な格納形式を選択するために，ばらつきと非零要素率の推移を調査し，本手法における動的選択のパラメータを設定する．選択時間隠蔽による実行時間の評価では，本手法で実装する格納形式選択時間の隠蔽について有効性を確認する．また，本評価では測定に用いる粒子の初期配置を設定する必要がある．このため，図 5–1 に壁粒子の初期配置，図 5–2 に予備評価で用いる水粒子を含む粒子初期配置をそれぞれ示す．壁粒子は，粒子間距離を 0.025 として図 5–1 のように x 軸と y 軸それぞれ 0 から 3.0 まで，厚さが 0.4 で流体がある領域を囲むように配置する．水粒子を含む図 5–2(a) は，ダム崩壊問題をモデルに水粒子を配置したものである．対して，図 5–2(b) は，流体を x 軸の中心と y 軸をできる限り上部に配置したものである．本評価の水粒子には，ある方向から任意の加速度を与えず，重力加速度のみを与えて解析する．このため，図 5–2(a) は，左に寄せられた水粒子が，重力に従い崩れる．また，図 5–2(b) は，上部にある粒子が重力に従い落下する．それぞれの水粒子は，x 軸に 1.2，y 軸に 2.0 の範囲に水粒子を配置する．流体落下問題は，同じ範囲の水粒子を横向きにして設置する．このため，図 5–2 の粒子配置の総粒子数は 5824 個であり，そのうちの水粒子は 3840 個である．表 5–2 に本研究の解析で用いるパラメータを示す．表 5–2 の粒子数密度，勾配モデル，ラプラシアンモデルの影響半径 r_e は，それぞれの重み関数で用いられる．影響半径は，大きな数値を用いると解析の安定性が向上するが，計算量も大きく増える．本研究で用いる影響半径は，初期粒子配置における粒子間距離 l_0 で規格化されているため，実際には $2.1l_0$ のように l_0 をかけた値が解析では用いられる．また，本研究では，流体を水と想定しているため，流体密度は，1000[kg/m] とする．

表 5–2 ： 解析で用いるパラメータ

パラメータ名	変数名	数値
粒子数密度用の影響半径	r_e	2.1
勾配モデル用の影響半径	r_e	2.1
ラプラシアンモデル用の影響半径	r_e	3.1
次元数	d	2
重力	g	-9.8
流体密度	n^0	1000
グリッドサイズ	gs	3.5

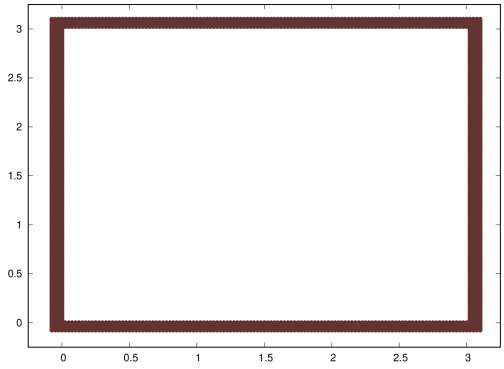
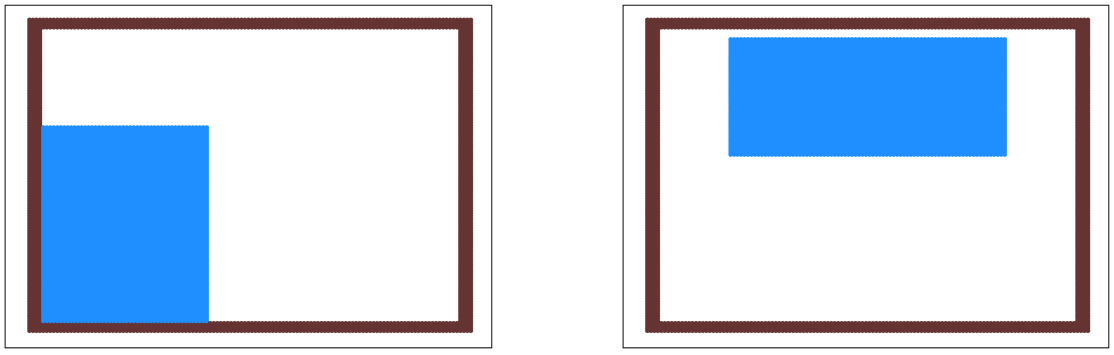


図 5–1 ： 壁粒子初期配置



(a)ダム崩壊問題　　(b)流体落下問題

図 5–2 ： 予備評価における粒子初期配置

5.2.1　疎行列ベクトル積における短縮時間の評価

本評価では，実際に MPS 法が生成する疎行列を ELL 形式，CRS 形式，JDS 形式で格納し，それぞれ疎行列ベクトル積を行うことによる実行時間を踏まえた短縮時間を評価する．これにより，MPS 法で生成された行列形状に対して適切な格納形式を用いることの有効性を確認する．本測定では，図 5–2 の粒子配置を用いて時間ステップごとに ELL 形式，CRS 形式，JDS 形式の疎行列ベクトル積を含む CG 法における実行時間を測定し，それぞれの形式における実行時間の差を算出する．表 5–3 と表 5–5 にそれぞれの形式における 1 ステップの CG 法にかかる時間を示す．表 5–3 と表 5–5 における最大時間は，選択された格納形式に適さない行列を計算した時間であり，最小時間は格納形式に適した行列を計算した時間である．このため，表 5–3 における時間差は，選択された格納形式が適する行列を計算した場合に短縮できる時間を示している．このことから，適切と判断された格納形式による計算時間は，他の格納形式より短くなることが確認できる．また，格納形式の選択にかかる時間は，1 ステップあたり平均で 0.11[μs] であるため，一つの格納形式を使い続けることより，格納形式を動的選択することで得られる効果のほうが大きいことがわかる．

表 5–3 ： ダム崩壊問題における 1 ステップの CG 法にかかる時間 [ms]

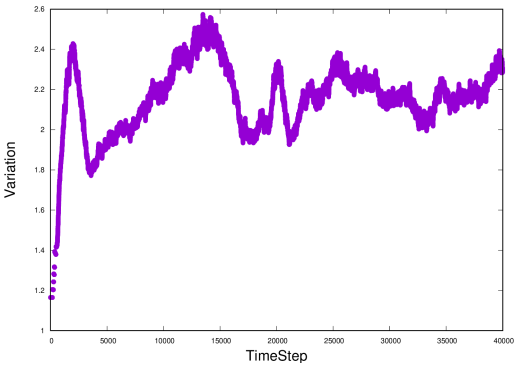
選択された格納形式	最大	最小	時間差
CRS 形式	4.672	2.468	2.204
ELL 形式	4.653	3.479	1.174
JDS 形式	4.652	3.301	1.351

表 5–4 ： 流体落下問題における 1 ステップの CG 法にかかる時間 [ms]

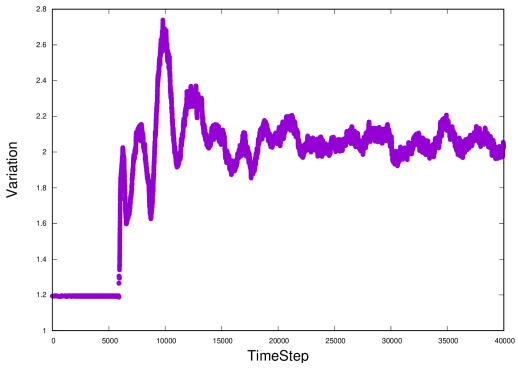
選択された格納形式	最大	最小	時間差
CRS 形式	4.640	2.127	2.513
ELL 形式	4.694	3.279	1.415
JDS 形式	4.815	3.178	1.637

5.2.2　動的選択におけるパラメータの設定

5.2.1 節より，適切な格納形式を用いることにより短縮できる時間があることを確認した．本評価では，MPS 法においてより適切な格納形式を選ぶために，動的選択のパラメータを設定する．格納形式を選択するためのパラメータは，従来研究における式 (4–1)，式 (4–2) のばらつきと非零要素率を用いた式 (4–3) を参考に決定する．式 (4–3) において，ELL 形式はばらつきが小さく非零要素率が小さい行列，CRS 形式はばらつきが大きく非零要素率が大きい行列，JDS 形式は ELL 形式と CRS 形式のどちらにも満たない行列が選択される．また，ばらつきと非零要素率は係数行列における非零要素数から算出されるため，係数行列の規模や形状に大きく左右する．しかしながら，従来研究において決定されたパラメータは，行列サイズが非常に大きい行列を元に決定している．本研究で用いる行列は，従来研究に比べ行列サイズが小さいため，適切な格納形式が選択されない可能性がある．このことから，動的選択手法では，より適した選択パラメータに調整する必要がある．図 5–3 と図 5–4 に本研究で使用する粒子配置においてばらつきと非零要素率の変動を示す．図 5–3 では横軸がタイプステップ数で縦軸がばらつきである．図 5–3 においてばらつきは，最大で 2.8，最小で 1.2 の幅で変化があり，平均は 2.3 である．また，ばらつきはタイムステップが進むにつれて大きくなることが確認できる．図 5–4 では横軸がタイプステップ数で縦軸が非零要素数である．図 5–4 において，非零要素率は最大で 0.0078，最小で 0.0053 の変化があり，平均は 0.005455 である．また，タイムステップが進むにつれて値が小さくなることが確認できる．図 5–3 と図 5–4 をそれぞれ問題ごとに見ると，ダム崩壊問題に比べて流体落下問題は，急激な値の変化が確認できる．よって，本研究で使用するばらつきのパラメータは，ELL 形式において平均の 2.3 とする．また，CRS 形式のばらつきの設定値として，最大値の 2.8 に近づくことが数ステップであることから，最大の 2.8 より 0.1 低い 2.7 とする．非零要素率のパラメータは，平均の 0.005455 を基準として設定する．これらを踏まえ，本手法における格納形式選択パラメータは，式 (5–1) のように設定する．

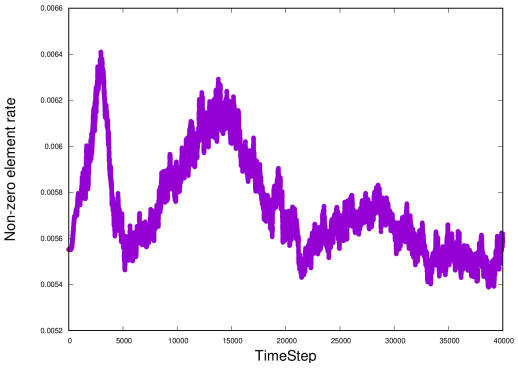


(a)ダム崩壊問題

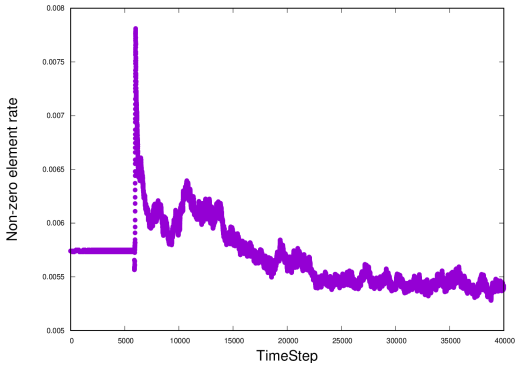


(b)流体落下問題

図 5–3 ： 本粒子配置におけるばらつきの変動



(a)ダム崩壊問題



(b)流体落下問題

図 5–4 ： 本粒子配置における非零要素率の変動

$$\begin{cases} \text{ELL 形式} : (B < 2.3 \text{ かつ } R < 0.005455) \\ \text{CRS 形式} : (B < 2.7 \text{ または } R \geq 0.005455) \\ \text{JDS 形式} : (\text{上記以外}) \end{cases} \tag{5–1}$$

式 (5–1) における選択条件を踏まえ，提案手法における選択条件パラメータの有効性を確認するために，従来パラメータを用いた MPS 法と実行時間の比較を行う．測定対象は，ダム崩壊問題，流体落下問題とする．この測定では，時間ステップの幅を 0.0001[s] とし，計測範囲を 0〜4.0[s] の 40000 ステップまで解析する．比較する手法は，式 (4–3) の久保田らの条件パラメータを用いた動的選択による MPS 法と

式 (5–1) の提案手法に適用する条件パラメータを用いた動的選択による MPS 法である．表 5–5 に従来のパラメータによる MPS 法の実行時間と提案のパラメータによる MPS 法の実行時間を示す．表 5–5 より，ダム崩壊問題と流体落下問題のどちらの問題に対しても提案するパラメータを使用したほうが高速であることを確認できる．また，高速化率は，最大で 1.089 倍が得られた．このことから，対象の行列によって選択条件パラメータの調整が必要であることを確認できる．

表 5–5 ： 選択条件パラメータを用いた MPS 法の実行時間 [s]

対象問題	従来のパラメータ [s]	提案のパラメータ [s]	高速化率 [倍]
ダム崩壊問題	708.20	651.81	1.086
流体落下問題	699.83	642.37	1.089

5.2.3 選択時間隠蔽による実行時間の評価

本手法は，CUDA を用いた圧力計算中に次ステップの格納形式を決定することで格納形式の選択時間を隠蔽する．このことから，選択時間の隠蔽による有効性を確認するために，選択時間を隠蔽する MPS 法と隠蔽しない MPS 法の実行時間を比較する．測定対象は，ダム崩壊問題，流体落下問題とする．また，時間ステップの幅を 0.0001[s] とし，計測範囲を 0〜4.0[s] の 40000 ステップまで解析する．表 5–6 に選択時間を隠蔽する MPS 法の実行時間と選択時間を隠蔽しない MPS 法の実行時間を示す．表 5–6 より，ダム崩壊問題と流体落下問題のどちらの問題に対しても選択時間を隠蔽したほうが高速であることを確認できる．また，高速化率は，最大で 1.02 倍が得られた．このことから，選択時間の隠蔽が有効であることを確認できる．

表 5–6 ： 選択時間の隠蔽による MPS 法の実行時間 [s]

対象問題	隠蔽しない手法 [s]	隠蔽する手法 [s]	高速化率 [倍]
ダム崩壊問題	653.32	651.81	1.002
流体落下問題	655.32	642.37	1.020

5.3　動的選択における MPS 法の全体実行時間の評価

予備評価において，格納する格納形式によって疎行列ベクトル積の時間が変化することを確認し，選択条件パラメータ調整や選択時間隠蔽の有効性を確認した．これらの予備評価を踏まえ，本評価では，疎行列格納形式の動的選択を行う手法とそれぞれの格納形式単体を用いる手法とで全体実行時間を比較し，MPS 法における疎行列格納形式動的選択の有効性を確認する．粒子配置は，予備評価で用いた図 5–2 の 2 つの問題に加え，水に落下する配置と底面配置を用いる．図 5–5 に本評価で用いる 4 つの粒子配置，図 5–6 に解析中の 4 つの粒子配置を示す．図 5–5 の粒子配置は，図 5–6 のように，水に落下する配置では上から水に落下するような動作をし，底面配置は底面に配置した水が重力に従い，波打つような動作をする．水に落下する配置の総粒子数は 8224 個で，底面配置の総粒子数は 5344 個である．解析で用いるパラメータは，表 5–2 と同様である．また，時間ステップの幅を 0.0001[s] とし，計測範囲を 0〜4.0[s] の 40000 ステップまで解析する．表 5–7 にそれぞれの粒子配置による MPS 法の処理全体における実行時間を示す．表 5–7 において，動的選択手法は，ダム崩壊問題，流体落下問題，底面配置問題で一番高速であることが確認できる．これにより，動的選択手法が CUDA を用いた MPS 法において有効であることが確認できる．また，CRS 形式のみ，ELL 形式のみ，JDS 形式のみとの実行時間を比べると CRS 形式が高速であることがわかる．このため，図 5–7 に CRS 形式のみの処理時間に対する速度向上率を示す．図 5–7 より，CRS 形式のみと動的選択手法は，ELL 形式のみ，JDS 形式のみに比べて高速であることが確認できる．しかしながら，動的選択手法と CRS 形式のみを比較すると動的選択手法が高速ではあるが，全体実行時間の時間差は 2.25[s] であり，高速化率が低い．この原因を調査するために，表 5–8，表 5–9，表 5–10，表 5–11 に表 5–7 の全体実行時間における計算処理や変換処理の内訳を示す．また，表 5–12，表 5–13，表 5–14，表 5–15 に表 5–8，表 5–9，表 5–10，表 5–11 のそれぞれの値から 1 ステップにかかる時間を示す．表 5–8，表 5–9，表 5–10，表 5–11 より，それぞれの格納形式において変換時間に差があることが確認できる．

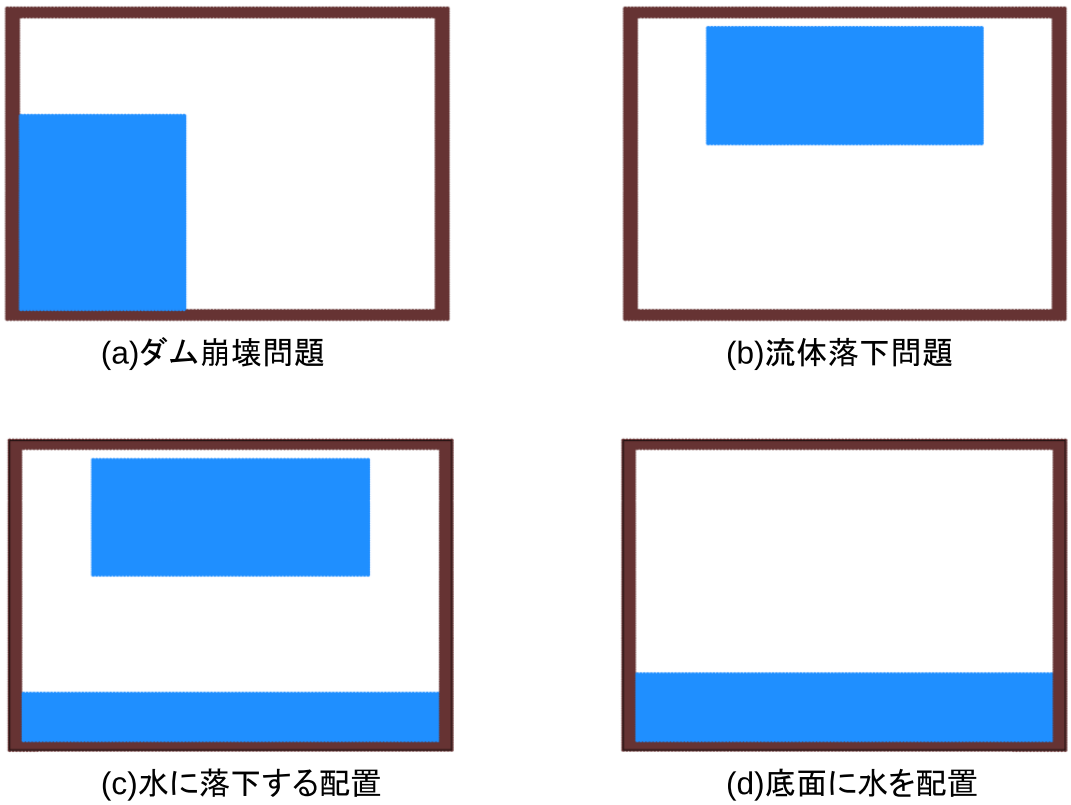


図 5–5 ： 本評価における粒子初期配置

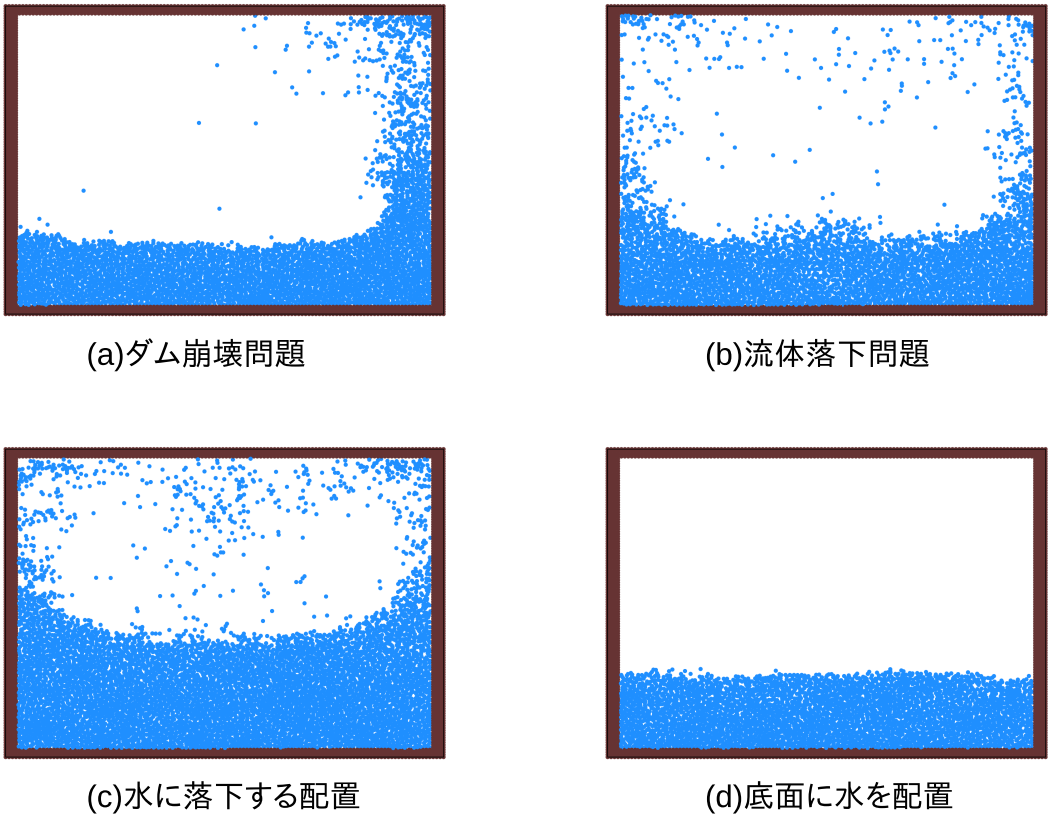


図 5–6 ： 解析中の粒子配置

表 5–7 ： MPS 法の全体実行時間 [s]

粒子配置	CRS 形式のみ	ELL 形式のみ	JDS 形式のみ	動的選択
ダム崩壊	654.06	706.17	708.11	651.81
流体落下	643.54	685.83	700.73	642.37
水に落下	863.28	904.11	931.31	904.99
底面配置	581.74	623.63	631.79	580.23

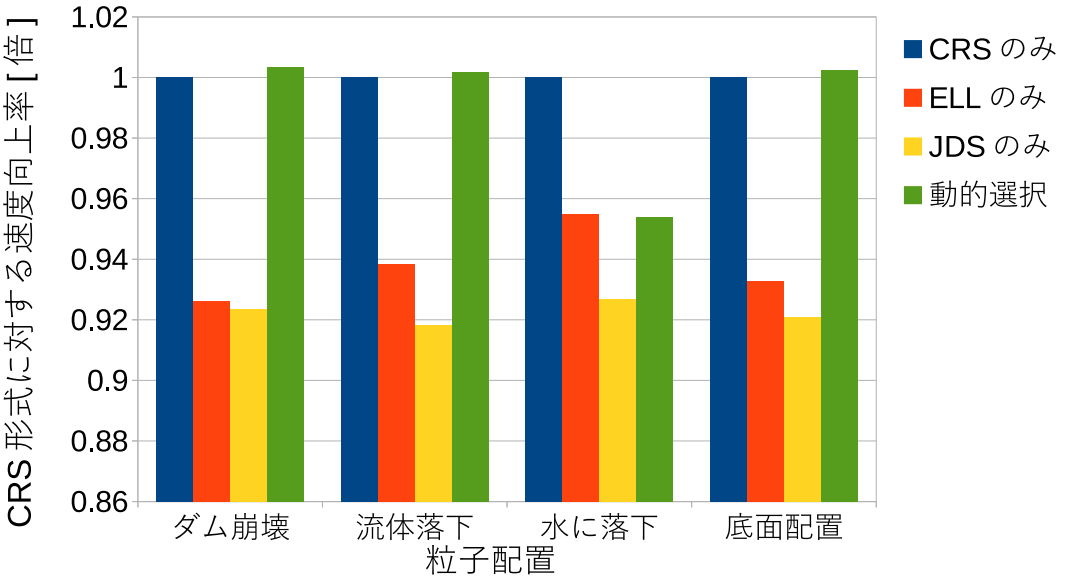


図 5–7 ： CRS 形式に対する速度向上率

表 5–8 ： ダム崩壊問題における MPS 法の圧力計算時間と格納形式変換時間 [s]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	654.06	120.16	137.47
ELL 形式のみ	706.17	127.95	168.85
JDS 形式のみ	708.11	145.48	167.67
動的選択	651.81	118.11	137.17

表 5–9 ： 流体落下問題における MPS 法の圧力計算時間と格納形式変換時間 [s]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	643.54	119.29	151.18
ELL 形式のみ	685.83	128.25	179.19
JDS 形式のみ	700.73	144.37	183.09
動的選択	642.37	121.99	150.43

表 5–10 ： 水の落下配置における MPS 法の圧力計算時間と格納形式変換時間 [s]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	863.28	170.00	172.22
ELL 形式のみ	904.11	176.84	201.71
JDS 形式のみ	931.31	204.44	207.74
動的選択	904.99	178.51	202.99

表 5–11 ： 底面配置における MPS 法の圧力計算時間と格納形式変換時間 [s]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	581.74	108.46	123.62
ELL 形式のみ	623.63	115.79	151.10
JDS 形式のみ	631.79	130.26	150.71
動的選択	580.23	107.79	123.87

表 5–12 ： ダム崩壊問題における MPS 法の圧力計算時間と格納形式変換時間 (1 ステップ)[ms]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	16.35	3.00	3.43
ELL 形式のみ	17.65	3.19	4.22
JDS 形式のみ	17.70	3.63	4.19
動的選択	16.29	2.95	3.42

表 5–13 ： 流体落下問題における MPS 法の圧力計算時間と格納形式変換時間 (1 ステップ)[ms]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	16.08	2.98	3.77
ELL 形式のみ	17.14	3.20	4.47
JDS 形式のみ	17.51	3.60	4.57
動的選択	16.05	3.04	3.76

表 5–14 ： 水の落下配置における MPS 法の圧力計算時間と格納形式変換時間 (1 ステップ)[ms]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	21.58	4.25	4.30
ELL 形式のみ	22.60	4.42	5.04
JDS 形式のみ	23.28	5.11	5.19
動的選択	22.62	4.46	5.07

表 5–15 ： 底面配置における MPS 法の圧力計算時間と格納形式変換時間 (1 ステップ)[ms]

手法	全体時間	変換時間	圧力計算時間
CRS 形式のみ	14.54	2.71	3.09
ELL 形式のみ	15.59	2.89	3.77
JDS 形式のみ	15.79	3.25	3.76
動的選択	14.50	2.69	3.09

5.3.1 格納形式における変換時間の評価

5.3 章より，動的選択を用いた MPS 法において，格納形式の変換にかかる時間的コストが大きいことがわかった．このため，本評価では，動的選択を用いた MPS 法によるより高速化率の高い高速化に向けて，それぞれの格納形式にかかる変換時間

の比較，検討を行う．表 5–16 に 1 ステップにおける COO 形式から CRS 形式・ELL 形式・JDS 形式への変換時間を示す．表 5–16 より，変換時間を比較すると，ELL 形式と JDS 形式に時間が多くかかっていることがわかる．このため，変換時間と 5.2.2 章の予備評価における短縮時間を比較する．表 5–17，図 5–8 にダム崩壊問題における短縮時間と格納形式変換の時間差を示す．表 5–17 における時間差は，変換時間から短縮時間を引いた値であるため，数値が小さいほど 1 ステップにおける短縮時間が大きいことを示している．表 5–17，図 5–8 より，CRS 形式，ELL 形式は短縮時間が変換時間を上回っているが，JDS 形式は短縮時間より変換時間に時間がかかっていることが確認できる．このことから，格納形式の変換時間に時間がかかっていることから，MPS 法による全体実行時間による高速化率が高くならない原因だと考えられる．格納形式の変換時間がかかってしまう理由としては，本研究の格納形式変換において ELL 形式と JDS 形式は，CRS 形式からの変換を行っているため，ソートなどの処理行程数が多く，無視できない時間がかかってしまっていると考えられる．このため，MPS 法の全体実行時間の高速化のためには，格納形式への格納や変換などに対する最適化や格納形式の変換時間を考慮した格納形式選択をすることが考えられる．

表 5–16 ： COO 形式からの変換時間 [ms]

格納形式	変換時間
CRS 形式	0.329
ELL 形式	1.085
JDS 形式	1.874

表 5–17 ： 1 ステップにおけるダム崩壊の短縮時間と格納形式変換時間 [ms]

格納形式	変換時間	短縮時間	時間差
CRS 形式	0.329	2.204	-1.875
ELL 形式	1.085	1.174	-0.089
JDS 形式	1.874	1.351	0.496

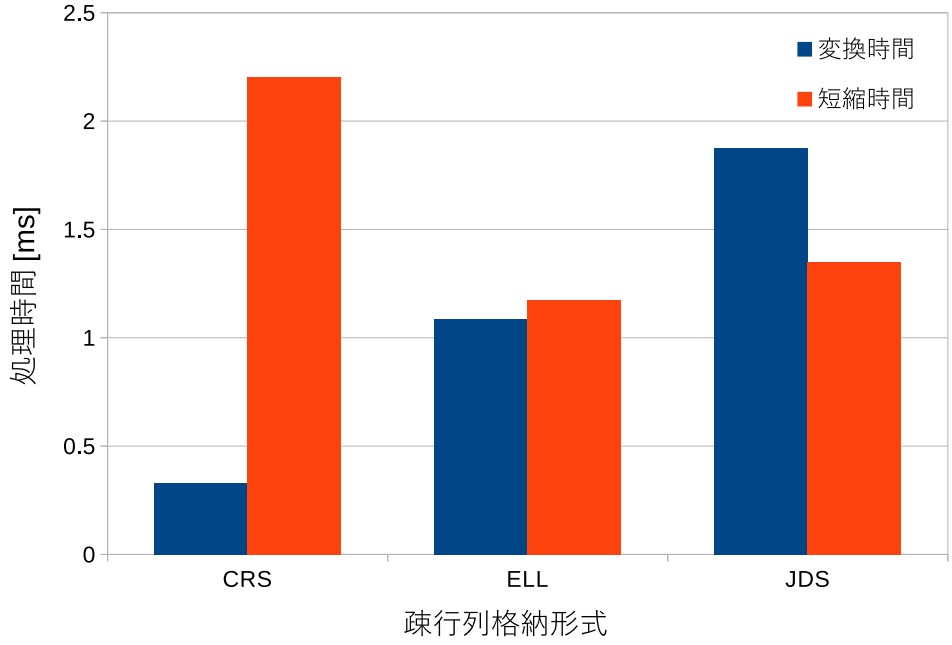


図 5–8 ： COO 形式からの変換時間と短縮時間

5.3.2　格納形式切り替えの評価

本評価では，図 5–5 に示す 4 つの粒子配置を用いて実行時間の評価を行った．実行時間の評価の結果，複数の粒子配置において有効であることが確認できたが，実際に用いた粒子配置が動的選択手法にとって有利であるか，不利であるかによっても結果に変化があると考えられる．このため，本評価の粒子配置が動的選択に向いていたかを検討する．格納形式選択の切り替えが多い粒子配置は，一つの格納形式を使い続けるより，最適な格納形式による高速な行列ベクトル積が期待でき，動的選択における処理時間への恩恵が大きいと考えられる．このため，本研究で用いた粒子配置について選択された格納形式の傾向と格納形式の選択回数，切り替え回数を調査する．表 5–18 に格納形式の切り替え回数と選択回数の推移を示す．表 5–18 より，ダム崩壊問題と流体落下問題において切り替え回数が多く，それぞれの形式における格納形式の選択も多いことがわかる．これらを踏まえ，図 5–9 に，それぞれの粒子配置で時間ステップごとに選択された格納形式を示す．図 5–9 より，切り替え数が比較的多いダム崩壊問題と流体落下問題は，0 から 20000 ステップまでは，格納形式が変わらず，20000 から 40000 ステップにかけて選択形式の切り替えが起こっている．このことから，0 から 20000 ステップまでの壁に到達するなどして粒

子が拡散するまでは粒子の動きが小さいため，格納形式が変化せず，全ステップを通して切り替え回数が少ないと考えられる．このため，粒子が常に流動するような実問題などの粒子配置に動的選択手法を実行できれば，格納形式の切り替えが多く起こり，格納形式の動的選択をより活かすことができると考えられる．

表 5–18 ： 格納形式切り替えの推移

粒子配置	CRS の回数	ELL の回数	JDS の回数	切り替え回数
ダム崩壊	38942	814	244	314
流体落下	29110	10209	681	1113
水に落下	1951	37891	128	25
底面配置	39672	266	62	12

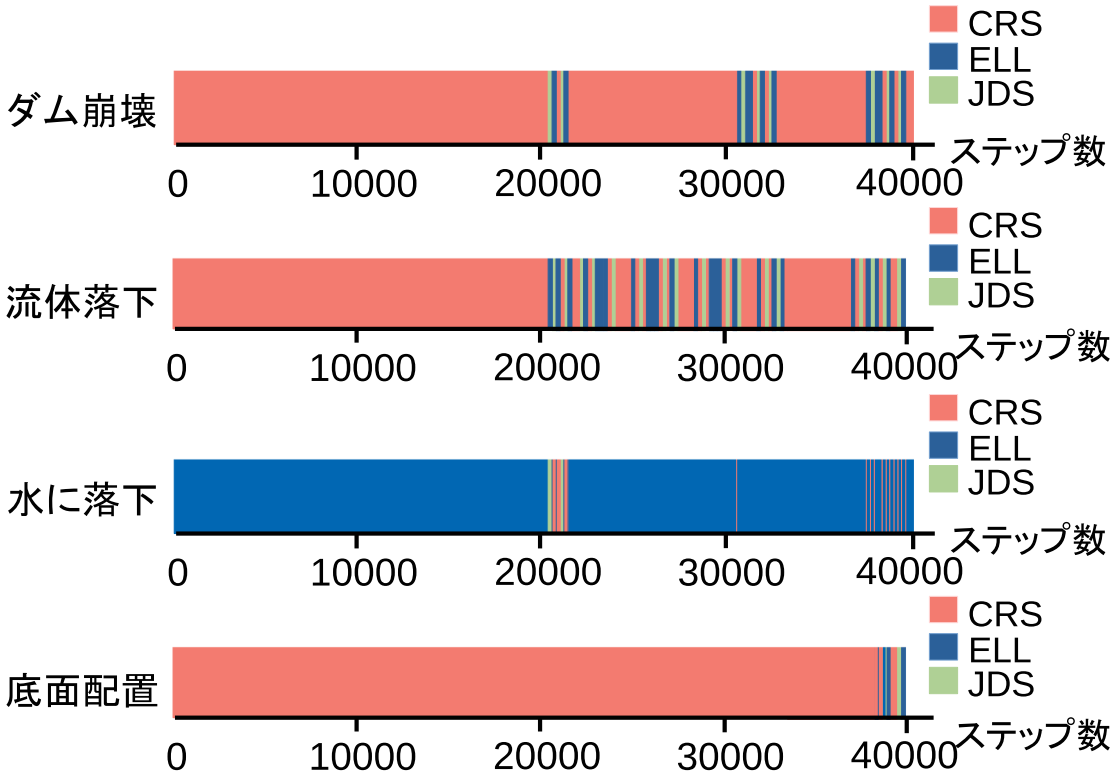


図 5–9 ： 時間ステップごとの選択された格納形式

第6章
おわりに

本論文では，CUDA を用いた MPS 法における実行時間の高速化を目的として，CUDA を用いた MPS 法に疎行列格納形式の動的選択を用いた行列ベクトル積の有効性を評価した．CUDA を用いた MPS 法の圧力方程式求解における行列ベクトル積は，生成される係数行列が対象疎行列であり，零要素による無駄な計算の排除による高速化やメモリアクセスを連続にするメモリアクセス回数の削減による高速化が有効と考えられる．疎行列ベクトル積においては，無駄な計算や無駄なアクセス回数を削減することを目的とした格納方法として疎行列格納形式がある．疎行列格納形式は，削減できる演算数やメモリアクセスが格納形式ごとに異なる．このため，それぞれの格納形式で行ごとの非零要素におけるばらつきや行列全体の非零要素率によって得意な粒子配置がある．このことから，従来の MPS 法では対象問題に合わせて疎行列格納形式を 1 つに決定して用いている．このため，圧力計算に使用する係数行列の非零要素位置に合わせて疎行列格納形式を動的に選択することにより高速化ができると考えられる．そこで，本研究では，CUDA を用いた MPS 法の圧力計算で疎行列格納形式の動的を選択する手法を提案し，MPS 法の全体実行時間における有効性を評価した．提案手法では，MPS 法の係数行列において時間ステップごとの流体粒子位置によって行列の非零要素配置が変化することを利用し，前ステップの動的選択後に次ステップの格納形式を決定する．また，次ステップの格納形式決定と CUDA による圧力計算は用いるアーキテクチャが異なることから，処理の依存がない．このため，本手法では CUDA による圧力計算中に次ステップの格納形式を決定することで格納形式選択時間を隠蔽する．また，本手法の格納形式選択パラメータは，係数行列の規模や形状に大きく左右される．このため，CUDA を用いた MPS 法でより適切な格納形式が選ばれるように選択条件のパラメータを調整する．

予備評価では，疎行列ベクトル積における短縮時間の評価，動的選択におけるパラメータの設定，選択時間隠蔽による実行時間の評価を行った．疎行列ベクトル積における短縮時間の評価では，MPS 法が生成する疎行列においてそれぞれ CRS 形式，ELL 形式，JDS 形式の CUDA を用いた行列ベクトル積の実行時間を測定し，求解の実行時間から格納形式を切り替えることによって短縮できる時間があるのかを確

認した．実行時間における評価の結果，粒子配置に対して最適な格納形式を利用することで，短縮できる時間が 1 ステップで最大 2.513[ms] あることを確認した．動的選択におけるパラメータの設定では，MPS 法においてより適切な格納形式を選ぶために，動的選択のパラメータを設定した．本手法において提案したパラメータでは，従来のパラメータを用いた MPS 法より，最大 1.089 倍の高速化率が得られた．このことから，対象の行列によって選択条件パラメータの調整が必要であることを確認した．選択時間隠蔽による実行時間の評価では，本手法で提案した選択時間の隠蔽が有効であるかどうかを確認した．評価の結果，最大 1.02 倍の高速化率が得られ，選択時間の隠蔽が有効であることを確認した．これらの予備評価を踏まえ，また，CUDA を用いた MPS 法の全体実行時間における評価では，CRS 形式・ELL 形式・JDS 形式それぞれの格納形式のみによる実行時間と動的選択手法による実行時間を比較した．評価の結果，CRS 形式のみと動的選択手法は，ELL 形式のみ，JDS 形式のみに比べて高速であることを確認した．しかしながら，動的選択手法と CRS 形式のみを比較すると動的選択手法が高速ではあるが，全体実行時間の時間差は 2.25[s] であり，高速化率が低いことも確認した．このため，高速化率が低い場合がある原因の調査として，格納形式の切り替えにおける格納形式の変換時間を測定した．その結果，1 ステップにおける格納形式の変換に最大 1.874[ms] かかっており，行列ベクトル積における短縮時間の効果が薄くなっていることを確認できた．このことから，CUDA を用いた MPS 法の疎行列格納形式動的選択手法における全体実行時間の高速化において，行列ベクトル積だけではなく，格納形式変換の最適化や格納形式変換を考慮した格納形式の選択が必要であることが明らかになった．

今後の展望としては，行列ベクトル積部分の最適化や格納形式変換の処理時間におけるチューニングを行った全体実行時間の評価をすることが挙げられる．また，本研究で使用した粒子配置は単純な配置であり，実問題を想定した大規模な粒子配置を用いて実行時間を測定することでより実用を目的とした評価ができると考えられる．

謝辞

本研究の機会及び素晴らしい実験環境を与えて下さり，貴重な時間を割いて研究の方向性を御指導頂きました前川 仁孝教授に深く感謝致します．研究の方向性をはじめ研究の細部に至るまで数々の有意義な御意見，御助言を賜わりました富永 浩文氏に感謝致します．本研究を進めるにあたり，日頃から惜しみなく御指導して頂きました中村 あすか氏に心から感謝致します．研究への様々な御提案や研究の進め方を丁寧に御指導下さった清水　達哉氏に感謝致します．特に，本研究のきっかけを与えて下さり，研究の進め方を丁寧に御指導下さった牧野氏にはこの場を借りて心から深く感謝致します．貴重な御意見，様々な御提案を頂いた前川研究室の皆様に御礼申し上げます．最後に，私をここまで育てて下さった家族に深く感謝します．

2021 年 12 月 24 日

参考文献

(1) 越塚誠一，柴田和也，室谷浩平： 粒子法入門，丸善出版 (2014).

(2) 堀智恵実，後藤仁志，五十里洋行，KHAYYER, A.：数値波動水槽のための 3-DMPS 法の GPU による高速化，土木学会論文集 B2(海岸工学)，Vol. 66, No. 1, pp. 56–60 (2010).

(3) 吉田郁政， 石丸真：MPS 法を用いた地震応答解析のための基礎検討，土木学会論文集 A， Vol. 66, No. 2, pp. 206–218 (2010).

(4) 別府万寿博，井上隆太，石川信隆，長谷川祐治，水山高久：修正 MPS 法による土石流段波モデルのシミュレーション解析，砂防学会誌， Vol. 63, No. 6, pp. 32–42 (2011).

(5) 入部綱清，藤澤智光，越塚誠一：粒子法による大規模解析におけるノード間通信の低減，日本計算工学会論文集，Vol. 2008, No. 20080020 (2008).

(6) 松谷浩明，武田一朗，橋本雅弘，平野啓之：粒子法を用いた熱可塑性スタンパブルシートの流動シミュレーション，日本複合材料学会誌，Vol. 40, No. 5, pp. 227–237 (2014).

(7) 渡邉忠尚，入部綱清，仲座栄三：ハイブリット型並列計算による MPS 法の大規模流体解析，土木学会論文集 B2，Vol. 68, No. 1, pp. 6–16 (2012).

(8) 後藤仁志，堀智恵実，五十里洋行，KHAYYER, A.：GPU による粒子法半陰解法アルゴリズムの高速化，土木学会論文集 B， Vol. 66, No. 2, pp. 217–222 (2010).

(9) Chen, X. and Wan, D.: GPU accelerated MPS method for large-scale 3-D violent surface Flows, Ocean Engineering, Vol. 171, pp. 677–694 (2019).

(10) Bell, N. and Garland, M.: Efficient Sparse Matrix-Vector Multiplication on CUDA，NVIDIA Technical Report NVR-2008-004，p. 32 (2008).

(11) 久保田悠司，高橋大介：GPU における格納形式自動選択による疎行列ベクトル積の高速化，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. 128, No. 19, pp. 1–7 (2010).

(12) 長坂一生，富永浩文，中村あすか，前川仁孝：CUDA におけるダイナミックパラレリズムを用いた JDS 形式疎行列ベクトル積の評価，情報処理学会第 80 回全国大会 (2018).

(13) Hu, X. Y. and Adams, N. A.: An incompressible multi-phase SPH method, J.Comput.Phys, Vol. 202, No. 1, pp. 65–93 (2005).

(14) 渡辺勢也，青木尊之，都築怜理，下川辺隆：接触による粒子相互作用の GPU 計算での近傍探索手法，情報処理学会論文誌，コンピューティングシステム，Vol. 8, No. 4, pp. 50–60 (2015).

(15) 西浦泰介， 阪口秀： GPU を用いた DEM の高速化アルゴリズム，日本計算工学会論文集，Vol. 2010, No. 20100007 (2010).

(16) Viccione, G., Bovolin, V. and Carratelli, E. P.: Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations, International Journal for Numerical Methods in Fluids, Vol. 58, No. 6, pp. 625–638 (2008).

(17) 原田隆宏，越塚誠一，河口洋一郎：GPU を用いた粒子法シミュレーションのためのスライスデータ構造，日本計算工学会論文集，Vol. 2007, No. 20070028 (2007).

(18) 原田隆宏，政家一誠，越塚誠一，河口洋一郎:GPU 上での粒子法シミュレーションの空間局所性を用いた高速化，日本計算工学会論文集，Vol. 2008, No. 20080016 (2008).

(19) Cevahir, A., Nukada, A. and Matsuoka, S.: CG on GPU-enhanced Clusters, 情報処理学会研究報告， Vol. 123, No. 15, pp. 1–8 (2009).

(20) Li, H., Zhang, Y. and Wan, D.: GPU Based Acceleration of MPS for 3D Free Surface Flows，Proceedings of the 9th international Workshop on Ship and Marine Hydrodynamics，Glasgow，UK((2015).

(21) 曽田康秀，渡邊明英， 小島崇：動的領域分割および複数 GPU を用いた MPS 粒子法の高速化，土木学会論文集 A2(応用力学)，Vol. 69, No. 2, pp. 95–105 (2013).

(22) Cheng, J., Grossman, M. and Mckercher, T.: CUDA C プロフェッショナルプログラミング，インプレス (2015).

(23) 佐藤駿一，高橋大介：GPU における SELL 形式疎行列ベクトル積の実装と性能評価，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. 164, No. 3, pp. 1–6 (2018).

(24) 櫻井隆雄，　直野健，片桐孝洋，中島研吾，黒田久泰，猪貝光祥：自動チューニングインターフェース OpenATLib における疎行列ベクトル積アルゴリズム，情報処理学会研究報告，Vol. 125, No. 2, pp. 1–8 (2010).

(25) 椋木大地，高橋大介：GPU における高速な CRS 形式疎行列ベクトル積の実装，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. 138, No. 5, pp. 1–7 (2013).

(26) 吉澤大樹，高橋大介：GPU における CRS 形式疎行列ベクトル積の自動チューニング，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC)，Vol. 135, No. 31, pp. 1–6 (2012).

(27) Kincaid, D. R., Oppe, T. C. and Young, D. M.: ITPACKV 2D User ' s Guide, Tech-nical report, University of Texas (1989).