博 士 論 文

適応的細分化格子の動的負荷分散による流れ場の並列計算

指導教官　松澤 照男 教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

古山 彰一

1999 年 1 月 14 日

要 旨

適応的細分化格子を各 PE に対して動的に分散することで、効率の良い並列計算を行なった。適応格子法は圧縮性流体解析を行なう場合、衝撃波などの数値的な不連続面をシャープに捕えるために、そのような領域に対して局所的に細分化格子を与えるものである。この手法を用いることで数値的な不連続面で、精度を保った解を得ることが可能になる。また、局所的な格子の細分化手法であり、計算領域全体を細分化格子で覆う必要がないため、計算コストの点で非常に有利な手法である。さらにこの手法は、物理量の勾配をみることで、必要な場所に自動的に細分化格子を与えるため、物理的な振舞いが予めわからない非定常の問題に対して非常に有効である。

適応格子法の並列計算として領域分割法を適用する場合、格子の多い領域を受け持つ Processing Element(以下 PE) と格子の少ない領域を受け持つ PE が存在し、各 PE の負荷量にかなりのばらつきが発生することを考慮しなければならない。この影響を考慮せずに並列計算を行なった場合、パフォーマンスの向上はほとんど見られない。また、適応格子法は非定常の問題に用いられるため、計算ステップが進むにつれ細分化格子が生成される領域が移動する。そのため、各 PE の受け持つ領域内の格子数は常に変動し続ける。このような理由から、領域分割法を用いて負荷の分散を行なうためには、各 PE の受け持つ領域を動的に変動させることで各 PE の負荷量を均等にしながら並列計算を行なう必要がある。

しかしながら、動的領域分割法は一般的に非常に複雑な手法である。特に分散メモリ計算機環境では、 PE 間で必要データを交換する必要があり、効率的な動的負荷分散を行なうためには、ユーザがデータ交換に関する手続きを明示的に示すことが必要である。また、このデータ交換にかかるコストも計算時間に比べ一般的に大きいため、これを小さく保ち、なおかつ各 PE の持つ負荷量がある程度均等化されている必要がある。

本論文では、以上のような背景を踏まえ、適応格子法に最適な動的領域分割法について高パフォーマンスが得られる計算手法を検討する。本法を用いることにより、負荷分散が非常に難しいとされる適応格子法の並列計算で 8PE 使用時で最大 6.8 倍の速度向上比が得られた。これは領域の大きさを等面積に固定して並列計算を行なった静的領域分割法に比べ 1.6 倍程度の速度向上の結果である。

目 次

第 1 章

はじめに

1.1 研究の背景

近年の計算機の著しい性能向上の恩恵を受け、数値流体力学 (CFD: Computational Fluid Dynamics) を始めとするシミュレーション分野での研究成果の発展は著しい。大規模計算機の性能向上により、格子点数を増加させ、精度の高い計算が可能になってきたのがその理由である。特に CFD の分野では 3 次元計算が一般的に用いられるようになり、その成果はシミュレーションのみにとどまらず、広く社会的に貢献できる応用技術の確立にもつながっている。

シミュレーションを行うにあたり、大規模計算に使われる計算機として、並列型計算機とベクトル型計算機が上げられる。特にベクトル型計算機は、 CFD 分野で頻繁に用いられる。ベクトル長を長くできる問題が多いという理由もあるが、ベクトル計算がしにくい代表的な問題であるポアソン方程式などの楕円型の方程式の解法などについても、これまで多くの研究がなされており、実用的な範囲で利用できる手法が確立されているのがその理由である。

一方、もう一つの大規模計算手法の代表的な手法である並列計算は、今なお、発展途上の部分である。並列計算は、複数のプロセッサを用いて文字どおり並列にそれらを実行させ計算する手法である。ベクトル計算機と違い計算機の大規模化が比較的容易であるため、シミュレーション分野を始めとする大規模計算が必要な分野で、並列計算技法の確立は非常に重要な意味を持つ。

並列計算手法として現在 CFD で用いられる代表的な手法として以下のものが上げられ

る。

- データを各 PE に分散するが、同タイミングでインストラクションの処理を行う SIMD 型。
- 分散メモリ型を強く意図し、データを分散し、さらにインストラクションも各 PE に任せる分散型の並列計算手法である MIMD 型。

SIMD 型の手法においてユーザが行う事は、並列計算の対象となる配列を各 PE に分散する事を明示的にプログラム中に書き込む事である。そのため、コード内容を詳細に知らない場合でも比較的容易に並列化を行う事が可能であり、基本的にはコンパイラ任せの感が強い。実際に自動並列化コンパイラに関する研究では、この手法を用いて並列化を行うのがこれまでの主流である。自動並列化に関する研究は実用化に向け、ユーザインタフェースが使い易くなってきている。また、パフォーマンスについても計算対象によっては容易に並列化効率を上げていく事が可能になってきた。しかし一般的な問題に対して、メモリを有効に用いる事に限界がある。また、メッセージパッシング等の複雑な処理は、ユーザから隠蔽する形になっており、ある程度以上のパフォーマンスを求めるにはユーザが何等かの形で並列処理に関する部分に積極的に関わる必要がある。特に、負荷量が局所的に異なる場合、またさらに、それらの負荷量が時間が進むにつれて動的に変動する場合に高いパフォーマンスを得るためには、ユーザが明示的に処理を与える必要がある。

このような負荷分散問題の一つの解決法として、計算領域を複数の小領域に分割しそれらの小領域を各 PE に担当させ並列計算を行う手法がある。この手法は領域分割法 (DDM : Domain Decomposition Method) と呼ばれ、並列化手法の 2 つ目でのべた MIMD 型の代表的な手法であり、 CFD の分野で数年前からよく聞かれるようになった。 CFD の分野で、局所的に負荷量の変動が起こる問題として、適応格子法があげられる。この手法は圧縮性流体解析を行なう場合、衝撃波などの数値的な不連続面をシャープに捕えるために、 Berger[11, 12] らによって提唱された手法である。この手法は、数値的不連続面に対して局所的に細分化格子を与えるものである。この手法を用いることで数値的な不連続面で高い精度を保ち解を得ることが可能になる。また、局所的な格子の細分化手法であり、計算領域全体を細分化格子で覆う必要がないため、計算コストの点で非常に有利な手法である。さらにこの手法は、物理量の勾配をみることで、必要な場所に自動的に細分化格子を与えるため、物理的な振舞いが予めわからない非定常の問題に対して非常に有効

である。適応格子法の並列計算として領域分割法を適用する場合、格子の多い領域を受け持つ PE と格子の少ない領域を受け持つ PE が存在し、各 PE の負荷量にかなりのばらつきが発生することを考慮しなければならない。この影響を考慮せずに並列計算を行なった場合、パフォーマンスの向上はほとんど見られない。また、適応格子法は非定常の問題に用いられるため、計算ステップが進むにつれ細分化格子が生成される領域が移動する。そのため、各 PE の受け持つ領域内の格子数は常に変動し続ける。このような背景から、領域分割法を用いて負荷の分散を行なうためには、各 PE の受け持つ負荷量を動的に変動させ、各 PE の負荷量を均等にしながら並列計算を行なう必要がある。とはいうものの動的領域分割法は一般的に非常に複雑な手法である。特に分散メモリ計算機環境では、 PE 間で必要データを交換する必要があり、効率的な動的負荷分散を行なうためには、ユーザがデータ交換に関する手続きを明示的に示すことが必要である。また、このデータ交換にかかるコストも計算時間に比べ非常に大きいため、これを小さく保ち、なおかつ各 PE の持つ負荷量がある程度均等化されている必要である。

1.2 研究の目的

本論文では、以上のような背景を踏まえ、適応格子法に最適な動的領域分割法について高パフォーマンスが得られる計算手法を検討する。この手法は前述の通り、各 PE の負荷量の問題、通信コストの問題を考慮する必要がある。そのため、適応格子法のみではなく、局所的に計算量が増大するような問題で、動的領域分割法を用いて行なう並列計算全般に対して効果的に応用可能であると考える。

これまで適応格子法に関してさまざまな研究がなされている。特に近年では、非構造格子に対する適応格子法の研究が盛んに行なわれている。非構造格子は複雑形状周りの流れ解析などで広く用いられているが、構造格子を用いた場合と比べて、一般的に消費するメモリ量、計算時間が大きい。この手法に対する並列計算については、これまでいくつかの並列計算アルゴリズムが検討されている。非構造格子の適応格子法に対して、総合的なパフォーマンスの評価を行なったものとしては、 A. Sohn ら [13] の研究と、木下ら [17] の研究があげられる。 A.Sohn らは、 JOVE と呼ばれるロードバランサーを導入して並列計算を行なった。この手法は、ある時間ステップでの各 PE の計算時間を計ることで負荷バランスの情報を得て、負荷バランスが悪い場合に再領域分割を行なう。この際、すでに分割された領域で得られる負荷量のバランスと、新たに領域分割を行なうことで得られる

負荷量のバランスを考慮に入れ、再領域分割を行なう最適タイミングを得ている。また、領域分割法としては、非構造格子での並列計算用に開発された手法である、HARP [14], JOSTLE [15], MRSB [16] などが用いられる。これらの手法は、領域の形状に対して制限が緩い分、データを交換をするPEが増えるといった問題が生じる。また、これらの領域分割手法は計算コストが大きいため、総合的な計算時間の中で、大きな時間を占めないようにする必要がある。木下らは、計算領域全体をPE数よりも多い数の領域に分割し、各プロセッサがこれらの領域を動的に受け持つアルゴリズムを提唱している。この手法は、負荷分散の単純な手法として用いられるサイクリックな手法の静的な部分を拡張し、各PEが動的に分担領域を変更するものである。しかしながら、物理的に隣接している領域が、並列計算機のネットワーク上でも隣接しているとは限らないため、通信コストの問題がある。

本論文では特に正方格子に対する適応格子法について並列計算アルゴリズムの提唱を行なう。正方格子は非構造格子に比べ格子を張る事が容易である事から最近見直されつつある [18][19]。これらの手法では格子形状が正方であるため、精度の高い計算を行うためには細かい格子が必要である。そのため、格子数が非常に多くなり並列計算が必要とされる。

このような背景を踏まえ、動的領域分割法を用いて、正方格子に対する適応格子法の並列計算アルゴリズムについて検討を行なう。この手法は、計算領域を分割し、それぞれの領域を各PEに割り当て、負荷分散は、その領域を伸縮させることで行なう。この方法は以下の利点がある。

- 各PEの受け持つ領域の境界の値のみを通信し合えば良いので、データを交換する通信先PEが両隣と明示的に示すことが可能で、かつ、データ通信を少なくできる。
- 通信を行なうPEが、領域の変動に関係なく同じである。
- コーディングが容易である。

また、各PEの受け持つ領域を再分割するタイミングも並列計算の際に重要となる。この問題では、各PEの受け持つ領域を再分割することで、どの程度の利益が得られるかを予測する必要がある。また再分割を行なうには、どの程度のコストがかかるのかも予測する必要がある。これらのトレードオフの問題を考慮に入れ、領域の再配分のタイミングについても自動的に各PEの領域を決定する手法が必要である。そこで本論文では、これら

の点を考慮に入れた適応格子法に最適な領域分割法を提供する。特に 8PE 程度を使用した場合の分散メモリ型並列計算機を使用した場合についてパフォーマンス評価を行い、一般的な問題に柔軟に対応できる手法を提供する。

1.3 本論文の構成

本論文は以下の構成からなる。第 2 章では、今回並列計算の対象として取り上げた数値計算法である、 TVD 法、適応格子法について述べる。第 3 章では、並列計算手法として領域分割法を取り上げ、本論文の主題である動的領域分割法を用いた負荷分散法について述べる。第 4 章では、第 3 章で提案した手法に関する結果を述べる。第 5 章では、本論文で提案する手法のパフォーマンス評価について議論を行ない、一般的な問題への適用性などについて議論を行う。最後に第 5 章では全体のまとめを行いこの手法の今後のどのような場面で利用されていくか将来の発展性について言及する。

第 2 章

流体の方程式について

この章では、本研究で使用した、非粘性圧縮性流体の基礎方程式と、その数値解析法について述べる。数値解析法については、 TVD 法、適応格子法について言及する。

2.1 基礎方程式について

本論文で計算対象とする流体は、圧縮性流体とする。また、粘性の寄与が物体近傍に限られると仮定し、その効果を無視した非粘性の方程式を用いる [1]。

非粘性流における質量保存の式は、 ρ を密度、 u を速度とすると、微分形、積分形でそれぞれ以下のように書く事ができる。

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0 \tag{2.1}$$

$$\oint (\rho \, dx - \rho u \, dt) = 0 \tag{2.2}$$

これらの式は、一次元的な検査面を考え、そこから流入する質量流量と流出する質量流量の差が、その検査面内の密度の時間変化となることから導かれる。

運動量 ρu については、検査空間内の運動量は両端から出入りする運動量流量の差だけではなく、両端の圧力差によって起こる流れによっても変化する。従って、

$$\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(p + \rho u^2) = 0 \tag{2.3}$$

$$\oint (\rho u \, dx - (p + \rho u^2) dt) = 0 \tag{2.4}$$

と書く事ができる。

次にエネルギーに関する式は、E を単位質量あたりの全エネルギーとすると、エネルギーは $e = \rho E$(単位体積あたりの全エネルギー) と書ける。 検査空間内でエネルギーの変化は両端でのエネルギー流束 eu の差と圧力が行なう仕事の差によって起こるので、次のようにエネルギーの式を書く事ができる。

$$\frac{\partial}{\partial t}e + \frac{\partial}{\partial x}((e+p)u) = 0 \tag{2.5}$$

$$\oint (e\ dx - (e+p)u\ dt) = 0 \tag{2.6}$$

これらの式の中でエネルギー e は全エネルギーであり、内部エネルギーと運動エネルギーの和である。 $\bar{E}$ を単位質量あたりの内部エネルギーとすると、

$$e = \rho(\bar{E} + \frac{u^2}{2}) \tag{2.7}$$

と書く事ができる。

式 (2.5) に現れる $(e+p)$ は単位体積あたりの全エンタルピー h で、

$$h = e + p = \rho H \tag{2.8}$$

である。ここで、 H は単位質量あたりの全エンタルピーである。

以上で、質量、運動量、エネルギーに関する 3 つの方程式が導かれたが、4 つの未知数 (ρ, u, p, e) がある。そこで、気体の状態方程式を導入する。状態方程式は一般的に、

$$p = p(\rho, \bar{E}) \tag{2.9}$$

であるが、本法では、理想気体を扱うため γ を比熱比とすると、

$$p = (\gamma - 1)\rho\bar{E} \tag{2.10}$$

と書くことができる。

時間方向に解くべき保存変数ベクトル Q とこれに対応する流束のベクトル表記をおこなう。

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ e \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \rho u \\ p + \rho u^2 \\ (e+p)u \end{bmatrix}. \tag{2.11}$$

まとめると以下のように書く事ができる。

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} = 0, \tag{2.12}$$

$$\oint (\mathbf{Q}dx - \mathbf{E}dt) = 0. \tag{2.13}$$

2 次元のオイラー方程式の場合、質量保存の式は積分形で以下のように表示できる。

$$\frac{\partial}{\partial t}\int\int\int_{\Omega} \rho \, dV + \int\int_{d\Omega} \rho \vec{u} \cdot d\vec{A} = 0 \tag{2.14}$$

ガウスの発散定理を利用して第 2 項を書き換えると、

$$\int\int\int_{\Omega} (\frac{\partial}{\partial t}\rho + \nabla \cdot (\rho \vec{u}))dV = 0 \tag{2.15}$$

となる。この式の第 2 項の表面積分をデカルト座標形の矩形領域について考え、その領域を極点的に小さくしていくと、保存形で書かれた微分形の 2 次元オイラー方程式が得られる。

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}\rho u + \frac{\partial}{\partial y}\rho v = 0 \tag{2.16}$$

2 次元方程式では運動量の式における運動量流量は x,y の 2 つの方向がある。 x, y それぞれの方向の単位ベクトルを $\vec{j}, \vec{k}$ で表すとする。 x 方向の運動量流量とこの検査体積に働く力の項、すなわち圧力項の 2 つをまとめて表面成分で表示すると、

$$p \cdot \vec{j} + \rho u \cdot \vec{u} \tag{2.17}$$

と書けるから、ある検査体積を出入りする x 方向の運動量のバランスは

$$\frac{\partial}{\partial t}\int\int\int_{\Omega} \rho u dV + \int\int_{\partial\Omega} (p \cdot \vec{j} + \rho u \cdot \vec{u}) d\vec{A} = 0 \tag{2.18}$$

質量保存と同じくガウスの発散定理により、

$$\int\int\int_{Omega}\{\frac{\partial}{\partial t}(\rho u)+\nabla\cdot(p\cdot\vec{j}+\rho u\cdot\vec{u})\}dV=0 \tag{2.19}$$

となる。微分形は、

$$\frac{\partial}{\partial t}(\rho u)+\frac{\partial}{\partial x}(p+\rho u^2)+\frac{\partial}{\partial y}(\rho uv)=0 \tag{2.20}$$

と得られる。 y 方向の方程式も同様にして、まず積分形が、

$$\frac{\partial}{\partial t}\int\int\int_{\Omega}\rho v dV+\int\int_{\partial\Omega}(p\cdot\vec{k}+\rho v\cdot\vec{u})d\vec{A}=0 \tag{2.21}$$

微分形が

$$\frac{\partial}{\partial t}(\rho v)+\frac{\partial}{\partial x}(\rho uv)+\frac{\partial}{\partial y}(p+\rho v^2)=0 \tag{2.22}$$

エネルギーの保存の式も一次元と全く同様に、積分形が、

$$\frac{\partial}{\partial t}\int\int\int_{\Omega}e\ dV+\int\int_{\partial\Omega}(e+p)\vec{u}\cdot d\vec{A}=0 \tag{2.23}$$

と得られる。微分形は、

$$\frac{\partial e}{\partial t}+\frac{\partial}{\partial x}((e+p)u)+\frac{\partial}{\partial y}((e+p)v)=0 \tag{2.24}$$

となる。一次元の場合と同様に、数値計算で良く見るベクトル形式に書き換えると、

$$\frac{\partial\mathbf{Q}}{\partial t}+\frac{\partial\mathbf{E}}{\partial x}+\frac{\partial\mathbf{F}}{\partial y}=0 \tag{2.25}$$

ただし、

$$\mathbf{Q}=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ e\end{bmatrix},\quad \mathbf{E}=\begin{bmatrix}\rho u\\ p+\rho u^2\\ \rho uv\\ (e+p)u\end{bmatrix},\quad \mathbf{F}=\begin{bmatrix}\rho v\\ \rho vu\\ p+\rho v^2\\ (e+p)v\end{bmatrix} \tag{2.26}$$

2.2 数値解析法

この数式を解くための手法として TVD 法を用いた。 TVD 法は圧縮性流体解析で一般的に利用されている手法である。この手法は、「単調であるスキーム」という条件を緩

め、全変動 (total variation) が増加しないという概念から、 Harten によって提案された。この概念から、数値的に不連続な領域付近で起こる、数値の振動の問題が解決される。このようにして開発された、流れのスムーズなところで高次精度を維持し不連続な部分を鮮明に表すことのできるスキームを高解像度スキームと呼ぶ。 高次精度を満足するために、 Harten の流束修正法、 Harten-Yee の風上型 TVD、 Yee の対象型 TVD、 MUSCL 法などがあげられるが [1]、本研究では、局所的な精度の維持に対して後述する適応格子法を用いるため、ここでは高解像度スキームのオリジナルとなる Harten の流束修正法を用いた。

また、局所的に精度を維持するために、局所格子細分化法を使用した [2]。計算の精度は計算法自身の精度と共に格子点密度に大きく依存する。一方、格子点総数は計算機環境により制限を受けるため格子形成の際には流れ場をある程度予想して格子点配置を考えることが必要である。局所格子細分化法は流れ場の計算と連動させて衝撃波などの流れの変化の激しいところに格子点を集中して計算精度を改善する方法である。一般的に基本格子として構造格子を用いた場合、格子点を移動させる方法と、格子点を局所的に追加・削除する方法が用いられる。前者の手法は、ある程度の面積を持った細分化格子で覆われた計算領域を移動させるため、流れ場に対して効率の良い細分化格子を与えることが実質不可能である。そのため格子点数の変動や格子のトポロジー変化はないものの、計算資源を効率良く使うと言う点では、少々不満が残る。それに対して後者の手法は、基本格子を構造格子とした場合、格子点数の変化、トポロジーの変化が生じるため非常に複雑な手法である。非構造格子を用いた場合には、格子点の追加・削除はそれほど複雑な過程ではないため、頻繁に利用される手法である。しかしながら、正方格子で精度を高く保つことができ、また、初期格子が張り易いと言う点から、構造格子に対して格子点を追加・削除する手法は非常に魅力的である。本研究では、物理量（密度など）の勾配が大きい場所に数値的な不連続面があると仮定し、そのような場所で格子を追加細分化し精度を維持する手法を用いる。この手法を用いることで、計算資源を有効に活用しながら、精度の高い計算が可能になる。

2.2.1 TVD 法

計算空間内の離散点上である物理量を u とした場合、以下のような変動量が定義できる。

$$|u_{j+1}^n - u_j^n| \tag{2.27}$$

この変動量を計算空間内で積分すると、以下のように全変動量が定義できる。

$$TV(u^n) = \sum_j |u_{j+1}^n - u_j^n| \tag{2.28}$$

この全変動量が時間ステップの進行と共に増加しないと考えたもの、すなわち、

$$TV(u^{n+1}) \leq TV(u^n) \tag{2.29}$$

を Total Variation Diminishing といい、この条件を満足するスキームを TVD スキームと呼ぶ

また、流速修正法としては、 Harten の流束修正法を用いた [4, 6]。スカラー方程式に対する一次精度の風上差分法の数値流速は、

$$\tilde{f}_{j+1/2} = \frac{1}{2}[(f_j + 1 + f_j) - |c|(u_{j+1} - u_j)] \tag{2.30}$$

の様に書ける。ここで Harten は f を $\tilde{f}$ で置き換えることで大局的高次精度を維持しようと考えた。

$$\tilde{f}_{j+1/2} = \frac{1}{2}[(\tilde{f}_{j+1} + \tilde{f}_j) - |\bar{c}|_{j+1/2}(u_{j+1} - u_j)] \tag{2.31}$$

この式が Harten の流束修正法を表す。ここで、 $\bar{c} = c + \gamma$ は修正された特性速度であり、γ は流束制限関数 g の関数である。それらは次のように定義される。

$$\gamma_{j+1/2} = \begin{cases} (g_{j+1} - g_j)/\Delta_{j+1/2} & \Delta_{j+1/2} \neq 0 \\ 0 & \Delta_{j+1/2} = 0 \end{cases} \tag{2.32}$$

$$r_j = minmod(\sigma_{j+1/2}\Delta_{j+1/2},\ \ \sigma_{j-1/2}\Delta_{j-1/2}) \tag{2.33}$$

ここで、 $\Delta_{j+1/2} = u_{j+1} - u_j$ である。また、 $\sigma_{j+1/2}$ は $\sigma(c_{j+1/2})$ と定義され、 $\sigma(z) = \frac{1}{2}[\Psi(z) - \frac{\Delta t}{\Delta x}z^2]$ を用いて計算される。ここに現れる Ψ は z の絶対値に相当するが、非物

理的な解を避けるためのエントロピー補正量 (entropy fix) と呼ばれるものである。これは以下の式が用いられる。

$$\Psi = \begin{cases} |z| & |z| \geq \delta \\ (z^2 + \delta^2)/2\delta & |z| < \delta \end{cases} \tag{2.34}$$

以上の理論から具体的な式の離散化は以下のように書く事が出来る [6]。以下に、$\mathbf{A}_x(\mathbf{q})$ の固有値、左右固有ベクトルを載せる。

$$a_1 = v_x - c \tag{2.35}$$

$$a_2 = v_x \tag{2.36}$$

$$a_3 = v_x \tag{2.37}$$

$$a_4 = v_x \tag{2.38}$$

$$a_5 = v_x + c \tag{2.39}$$

$$c = (\gamma p/\rho)^{1/2} \tag{2.40}$$

右固有ベクトルは以下のように書く事ができる。

$$\mathbf{R_1} = \begin{pmatrix} 1 \\ v_x - c \\ v_y \\ v_z \\ H - v_x c \end{pmatrix}, \mathbf{R_2} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ v_y \end{pmatrix}, \mathbf{R_3} = \begin{pmatrix} 1 \\ v_x \\ v_y \\ v_z \\ \theta/2 \end{pmatrix}, \mathbf{R_4} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ v_z \end{pmatrix}, \mathbf{R_5} = \begin{pmatrix} 1 \\ v_x + c \\ v_y \\ v_z \\ H + v_x c \end{pmatrix}$$

ここで、$H = (E + p)/\rho$であり、$\theta = v_x^2 + v_y^2 + v_z^2$ である。左固有ベクトルは以下のように書く事ができる。

$$\mathbf{L_1} = [\frac{(\gamma-1)\theta/2 + cv_x}{2c^2}, -\frac{(\gamma-1)v_x + c}{2c^2}, -\frac{(\gamma-1)v_y}{2c^2}, -\frac{(\gamma-1)v_z}{2c^2}, \frac{(\gamma-1)}{2c^2}] \tag{2.41}$$

$$\mathbf{L_2} = [-v_y, 0, 1, 0, 0] \tag{2.42}$$

$$\mathbf{L_3} = [1 - \frac{(\gamma-1)\theta}{2c^2}, \frac{(\gamma-1)v_x}{c^2}, \frac{(\gamma-1)v_y}{c^2}, \frac{(\gamma-1)v_z}{c^2}, -\frac{\gamma-1}{c^2}] \tag{2.43}$$

$$\mathbf{L_4} = [-v_z, 0, 0, 1, 0] \tag{2.44}$$

$$\mathbf{L_5} = [\frac{(\gamma-1)\theta/2 - cv_x}{2c^2}, -\frac{(\gamma-1)v_x - c}{2c^2}, -\frac{(\gamma-1)v_y}{2c^2}, -\frac{(\gamma-1)v_z}{2c^2}, \frac{(\gamma-1)}{2c^2}] \tag{2.45}$$

ここで、 Roe のリーマン近似解法を用いる際の Roe 平均についてまとめておく。

$$v_{x,i+1/2} = \frac{\sqrt{\rho_i}v_{x,i} + \sqrt{\rho_{i+1}}v_{x,i+1}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} \tag{2.46}$$

$$v_{y,i+1/2} = \frac{\sqrt{\rho_i}v_{y,i} + \sqrt{\rho_{i+1}}v_{y,i+1}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} \tag{2.47}$$

$$v_{z,i+1/2} = \frac{\sqrt{\rho_i}v_{z,i} + \sqrt{\rho_{i+1}}v_{z,i+1}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} \tag{2.48}$$

$$H_{i+1/2} = \frac{\sqrt{\rho_i}H_i + \sqrt{\rho_{i+1}}H_{i+1}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} \tag{2.49}$$

ここで、

$$c_{i+1/2} = \{(\gamma-1)[H_{i+1/2} - \frac{1}{2}(v^2_{x,i+1/2} + v^2_{y,i+1/2} + v^2_{z,i+1/2})]\}^{1/2} \tag{2.50}$$

以下に、 x 方向における計算過程を載せる。

$$L_x\mathbf{q}^n_i = \mathbf{q}^n_i - \frac{\Delta t^n}{\Delta x}(\bar{f}_{x,i+1/2} - \bar{f}_{x,i-1/2}) \tag{2.51}$$

$$\bar{f}_{x,i+1/2} = \frac{1}{2}[\mathbf{F}_x(\mathbf{q}^n_i) + \mathbf{F}_x(\mathbf{q}^n_{i+1})] - \frac{\Delta x}{2\Delta t^n}\sum_{k=1}^{5}\beta_{k,i+1/2}\mathbf{R}^n_{k,i+1/2} \tag{2.52}$$

$$\beta_{k,i+1/2} = Q_k(\frac{\Delta t^n}{\Delta x}a^n_{k,i+1/2} + \gamma_{k,i+1/2})\alpha_{k,i+1/2} - (g_{k,i} + g_{k,i+1}) \tag{2.53}$$

$$\alpha_{k,i+1/2} = L^n_{k,i+1/2} \cdot (bfq^n_{i+1} - bfq^n_i) \tag{2.54}$$

$$\gamma_{k,i+1/2} = \begin{cases} (g_{k,i+1} - g_{k,i})/\alpha_{k,j+1/2}, & for \alpha_{k,j+1/2} \neq 0 \\ 0, & for \alpha_{k,j+1/2} = 0 \end{cases} \tag{2.55}$$

$$g_{k,i} = sign(\tilde{g}_{k,i+1/2}) max\{0, \ \ min[|\tilde{g}_{k,j+1/2}|, \ \ \tilde{g}_{k,i+1/2} sign(\tilde{g}_{k,i+1/2})]\} \tag{2.56}$$

$$\tilde{g}_{k,i+1/2} = \frac{1}{2}[Q_k(\frac{\Delta t^n}{\Delta x} a^n_{k,i+1/2}) - (\frac{\Delta t^n}{\Delta x} a^n_{k,i+1/2})^2]\alpha_{k,i+1/2} \tag{2.57}$$

$$Q_k(\chi) = \begin{cases} [\chi^2/(4\epsilon)] + \epsilon, & for |\chi| < 2\epsilon \\ |\chi|, & for |\chi| \geq 2\epsilon, \end{cases} \quad \epsilon = \begin{cases} 0.03, & for k = 1 \ \ and \ \ 5 \\ 0, & for k = 2, \ \ 3, \ \ and \ \ 4. \end{cases} \tag{2.58}$$

時間刻み Δt^n は、 CFL 条件 $[\Delta t^n < \Delta x/(|v^n_{x,i+1/2}| + c^n_{i+1/2})]$ によって制限される。また、多次元化は、以下の方法で行なう。

$$\mathbf{q}^{n+1} = L_x L_y L_y L_x \mathbf{q}^n \tag{2.59}$$

2.2.2 適応格子法

適応格子法 (AMR: Adaptive Mesh Refinement method) は、 Berger と Colella[11] によって研究された手法である。これは、物理的な変動の激しい領域や物体近傍などで、局所的に格子を細かくすることで、その領域の物理状態を詳細に示すことができる手法である。この手法は、物理状態に応じて局所的に格子を細分化する手法であることから、非定常の問題に対して非常に有効であり、また、不必要に格子を増やすことをしないために計算コストの点で非常に優れた手法であると言える。本論文では、圧縮性流体を解析する場合に衝撃波等の不連続面を数値的にシャープに捉えるためにこの手法を用いる。この手法を用いることで、衝撃波が現れる領域が予めわからない場合でも、衝撃波が発生する部分に細分化格子が張れることになり、精度が高い効率の良い計算が可能となる。

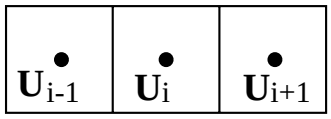


図 2.1: レルナーの指針値

分割指針について

適合的細分化格子法は、任意の時間に物理量の状態を見る事によって、格子の配置を局所的に変更していく。この場合格子の配置の変更とは、初期条件で使用していた格子を局所的に細分化する事である。格子の細分化の基準はさまざまな方法が考えられるが、本研究では圧縮性流体を扱っている為、この流体の性質に合っている分割指針を用いて、格子配置の変更を行なう。圧縮性流体の性質として、衝撃波などの発生が上げられる。衝撃波の領域は、物理量が不連続にジャンプしている所で、この様子を数値的に精度良く捉えることは困難である。そこで、このような領域をあらかじめ予測しておき、 TVD 法を適用する前に、この領域について格子を細分化する事を考える。このような領域を特定する為に、以下のような分割指針を用いる。

$$E_i = \frac{|U_{i-1} - 2U_i + U_{i+1}|}{|U_{i-1} - U_i| + |U_i - U_{i+1}| + c} \tag{2.60}$$

$$c = \delta(|U_{i-1}| + 2|U_i| + |U_{i+1}|) \tag{2.61}$$

この式は Löhner の指針値 [5] と呼ばれるものである。ここで U の値は物理量で、本研究では密度を用いた。δ は任意の数値で、 0.05 を用いた。また、下付き文字は、格子番号をあらわしている。この指針値は物理量の勾配が大きな領域で大きな値を取る事になる。つまり、この値がある値よりも大きな領域において、格子の細分化を行なえば良い。実際に物理量の不連続面にこの指針だてを行なう。図 2.2は、この式によって分割指標を立てられた位置を示すものである。横軸が格子が配置されている位置を表し、縦軸がある物理量を表している。ここでは、 x が 5.0、 5.1 のところに物理量が不連続である部分を想定している。図中で $\times$、 $*$ とマーキングされている部分が、この指標の立った位置を示すものである。初期格子の細分化レベルを 0、細分化が進むにつれてレベル 1、レベル 2 と言うようにレベル数が増えるとし、この場合特に、$\times$はこの位置にレベル 1 の格子を

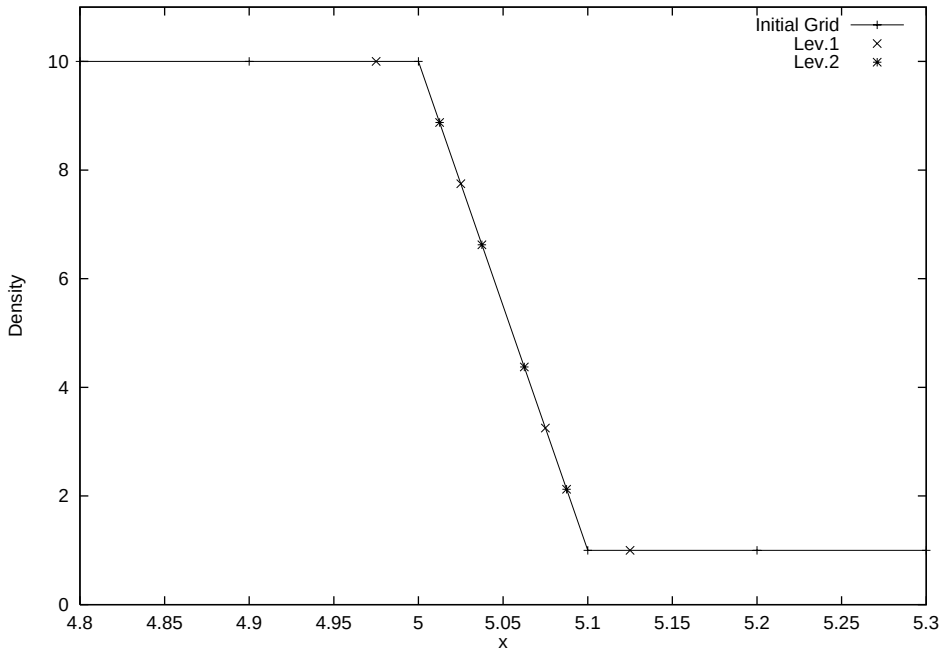


図 2.2: レルナーの指針値の適用位置

生成することを表しており、＊についても同様にレベル 2 の格子を生成することを表している。この図を見ると、不連続面に対してマーキングが行われ、細分化の必要な部分が細分化格子が生成される対象領域になっていることがわかる。

細かい格子のデータについて

適合的細分化格子法は、初期サイズの格子を細分化していく事から、細分化される事で生成された格子のデータを何らかの形で得ることが必要である。本論文では、格子の細分化指針として Löhner の指針値を用いており、これは、物理量が不連続になっている領域で格子を細分化する指針だてを行なうものであった。このことから、細分化によって生成された格子のデータは、物理量の不連続な領域で適切な値を与える事が必要である。しかし実際には、連続領域でありながらも物理量の勾配が大きい領域において Löhner の指針値は格子細分化位置として認識する可能性がある。そのため、より粗い格子のデータをを一次で内挿し細かい格子のデータとする。具体的には、最も近い、左右のより粗い格子を検索しその距離により空間的な一次内挿を行った。

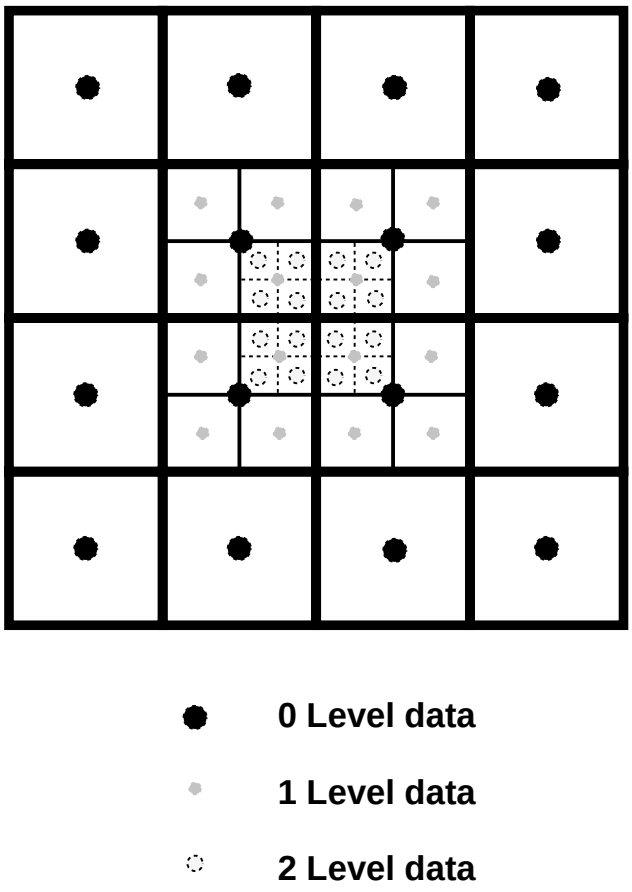


図 2.3: 格子の配置図（例）

タイムスケジューリングについて

格子の分割指針によって分割された格子により、計算領域には大小の格子が混在する事になる（図 2.3）。これをどの様に解くかが次なる課題となる。TVD スキームを安定に解く為に、初期格子においてクーラン数を 0.7 にとっている。（クーラン数とは、計算領域での情報伝播速度と物理領域での情報伝播速度の比であり、ここで利用している陽的スキームにおいては、クーラン数を 1 以下にする必要がある）。細分化された格子においても、安定にスキームを解く為に、クーラン数を 0.7 にする必要がある。そのことから、レベル 0 における時間刻みを Δt とした場合に、レベル 1 の格子の時間刻みは $1/2\Delta t$、レベル 3 での時間刻みは $1/4\Delta t$ で解く事になる。これらの事から、図 2.4のようなスケージューリングで、初期格子における 1 ステップ、つまり Δt を進める。この表か

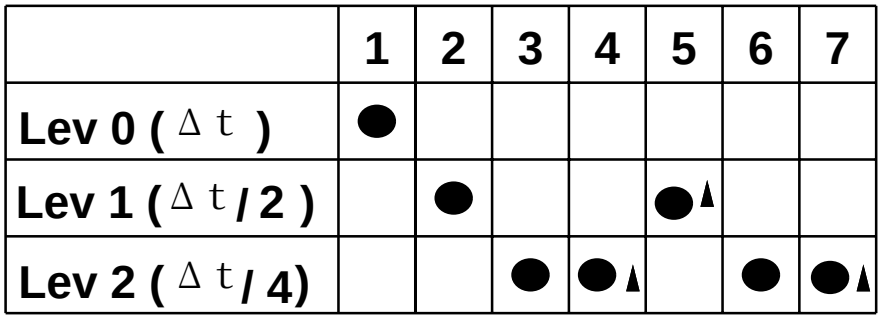


図 2.4: タイムスケジューリング

ら、 2 levelでの適応格子法は 7 ステップを踏むことで Δt 進むことがわかる。また、図の中で矢印が表しているものは異なる格子間でのデータのやりとりを行なうタイミングを表したものであるが、これについては次節で述べる事にする。

このようなスケジューリングで、サイズの異なる格子上で流れ場を解く。細分化格子での物理量は Δt 進める間に複数回の計算を行う必要があるので、計算バッファを適切に与える必要がある。そのバッファについては、同レベルの細分化格子が隣接している場合はお互いにその値を利用する。一方、粗い格子と接しているような場合は、 level 0 の格子を一番始めに Δt 進めるので、 t_0 と $t_0 + \Delta t$ の値を用いて、時間について一時の内挿を行い必要データとする。

細かい格子から粗い格子へのデータ転送

より細かい格子からなっている格子の体積は $V = 2^N v$ で求められる。ここで v は $finer\ cell\ volume$、 N は $spatial\ dimension$ である。また境界の長さは、 $A = 2^{N-1}a$ で表すことができる。結局セルのエッジの長さは、 $D = \frac{V}{A} = 2\frac{v}{a} = 2d$ となる。またタイムステップは $T = 2t$ とする。細かいグリッドで覆われているような粗い格子の物理量は、細かい格子での物理量 (u) の平均値を用いて、与える方法を用いる。

$$U \leftarrow \frac{1}{V}\sum_{i=1}^{2^N} u_i v = \frac{1}{2^N}\sum_{i=1}^{2^N} u_i \equiv \overline{u} \tag{2.62}$$

格子は、細分化レベル毎にそれぞれ独立に解かれるため、レベルの違う格子が接しているようなところで、流束の不一致が起こることが考えられる。そこで、以下のような補正を加えることによってこの問題を解決する。

$$
\begin{aligned}
U \leftarrow U - \frac{1}{V}(TAF(U) - \sum_{n=1}^{2}\sum_{i=1}^{2^N-1} taf_i^n(u)) \\
= U - \frac{T}{D}(F(U) - \frac{1}{2^N}\sum_{n=1}^{2}\sum_{i=1}^{2^N-1} f_i^n(u)) \\
= U - \frac{T}{D}(F - \overline{f}) \qquad (2.63)
\end{aligned}
$$

この流束の補正によって影響を受ける格子は、図 2.5で、• で示した部分である。これは、より細かい格子に接しているようなより粗い格子である。より細かい方の格子は、粗い格子よりも精度が良いものとして、物理量の更新は行なわないものとする。

2.2.3 衝撃波管問題について

TVD 法と 1 次元 1 細分化レベルにおける適合的細分化格子法を用いて、その精度評価を行なった。 計算モデルとして衝撃波管問題を取り上げる。この問題は、予め解析解がわかっている為に、スキームの精度評価の際に利用される。初期条件としては、図 2.6のように与える。時間 $t = 0$ の時に、領域の中間を仕切っていた仕切りをはずし、その後の経過を見るものである。図 2.7がその結果で、横軸は位置を表し、縦軸は密度を表している。この実験は x 方向に 100 の格子点をとっている。図の中で、実線はレベル 1 の細分化格子で計算領域全体を覆った場合の計算結果を示す。点線はレベル 0 の初期格子のみで計算を行った場合の結果である。＋が 1 レベルの AMR を用いた場合の結果である。また、この時点での物理領域における実時間は $t = 14.154$ である。

下図は x が 0.6 から 0.8 での拡大図であり、特に衝撃波近傍で初期格子のみで解いた時よりも、よりシャープな解が得られている事がわかる。

また表 2.1に計算コストに関しての結果を載せた。これは上記実験の場合の、計算時間について触れたものである。計算機は Pentium II 233MHz のパーソナルコンピュータである。この表で、 Coarse grids は計算領域全体を初期格子のみを用いて計算を行なった場合の結果を示した。 Fine grids は、計算領域全体をレベル 1 の細分化格子で覆い計算した場合の結果である。 1 Lev. AMR は 1 レベルの適応格子を用いた場合の計算結果である。 1 Lev. AMR は、計算ステップを進める毎に格子の細分化、粗化を判定しているため、 Coarse grid の場合と比較して付加された時間が大きくなっている。この手続きを適切なタイミングで行なえば、この時間はもっと削減することが可能であると考える。使

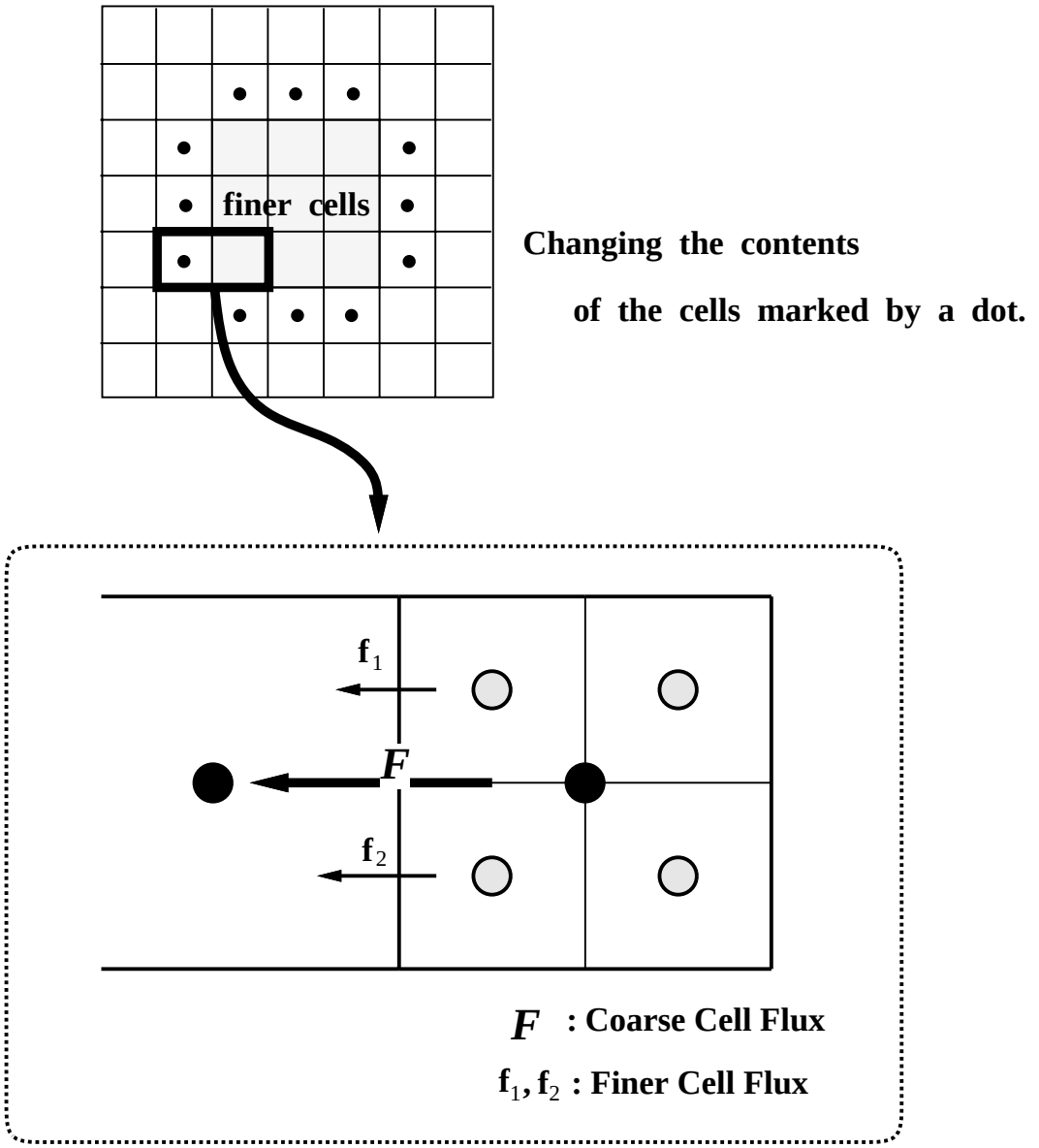


図 2.5: 流束補正について

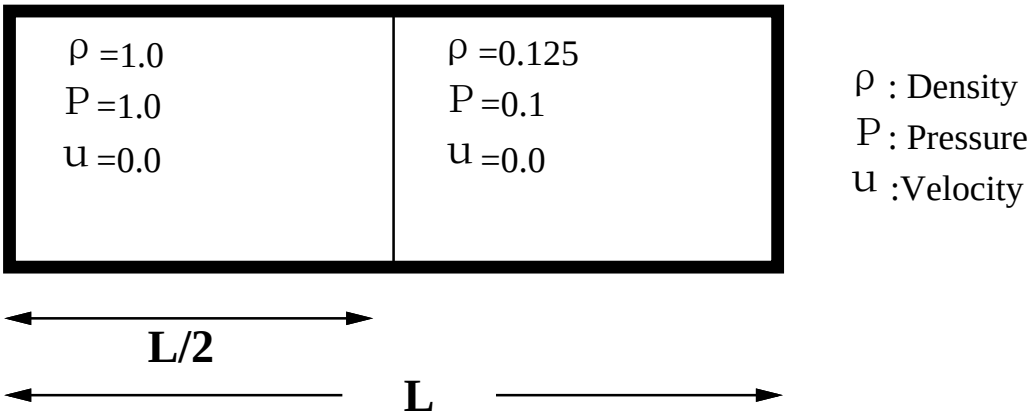


図 2.6: 衝撃波管問題について

Grid	time (sec)
Coarse grids	0.44
Fine grids	1.75
1 Lev. AMR	1.40

表 2.1: AMR 法の計算時間

用メモリについては、現在メモリ利用を動的には行なっておらずどの手法でも同じメモリ量を使用しているが、 malloc() などの関数を用いることで、使用メモリ量が削減可能であると考えられる [7]。

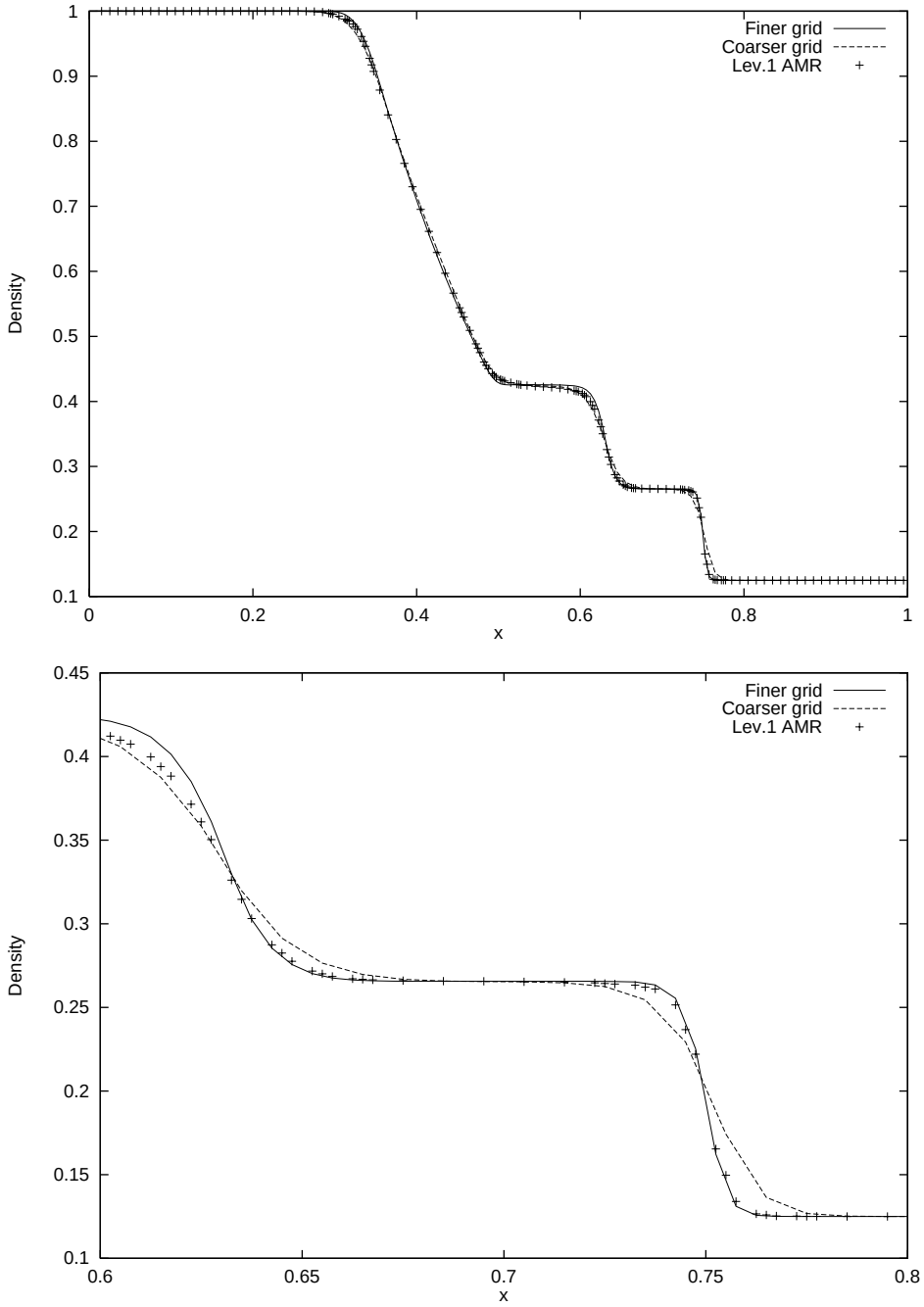


図 2.7: 衝撃波管問題の結果

第 3 章

適応格子法に関する並列計算

この章では、静的領域分割法、動的領域分割法について述べる。静的領域分割法では、単純な並列化手法として従来行われてきたものを取り上げ、その手法について具体的に述べる。また、本論文の主題となる動的領域分割法を用いた並列計算アルゴリズムについて言及する。

3.1 静的な領域分割法

3.1.1 領域分割形状

一般的に領域分割法を用いた並列計算といった場合、各 PE は、時間の変動、流れ場の変化には関係なく常に定常の計算領域を受け持つ。この手法は「静的領域分割法」と呼ばれ、各 PE が常に同じ領域を計算するだけではなく、メッセージ交換をする相手 PE と交換するメッセージ量が常に同じであるため、並列計算を容易に実行できる。 CFD の分野では、各 PE の負荷量が予めわかる問題が多いため、負荷量が均等になるように計算領 域を分割し、各 PE に分散させることで効率のよい並列計算が実現できる。この手法は、各 PE の負荷量が時間が進むにつれて変動しない問題に対して、高いパフォーマンスが期待できる。特に本論文で使用している双曲型の方程式を、差分法で数値解析を行なう場合は、ある離散点での物理量が影響を及ぼす領域は、 CFL(Courant) 条件から制限される。従って、非常に単純に並列化を施すことが可能である。具体的には、各 PE が受け持つ領域の境界に数メッシュ分の適切なバッファを加えてやるだけで、それぞれの PE が

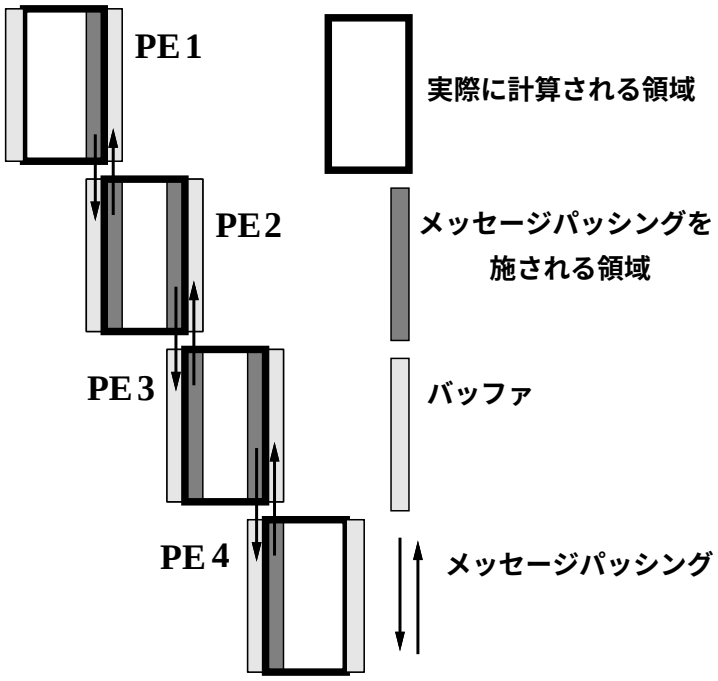
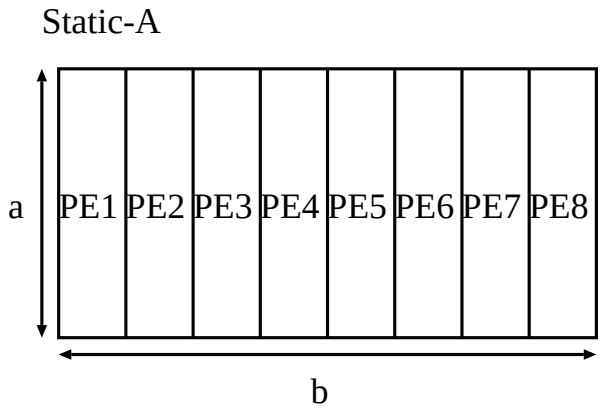


図 3.1: 静的領域分割法での通信対象領域

独立に計算を進めることが可能になる (図 3.1)。このバッファにおける物理量を各タイムステップ毎に隣の PE とメッセージパッシングすることで、計算を進めることが可能である。

ここでは、この「静的領域分割法」について 3 例を取り上げる。図 3.3で、この手法の一つ目の手法を示した。この図は、ステップ付管内流れを表しており、各 PE の受け持つ格子数がほぼ同じになるように横方向に計算領域を分割している。この手法を、 Static-A と呼ぶことにする。管内流れのように計算領域形状が比較的長い問題については、ある一つの PE に注目した場合、この手法ではメッセージ交換を行う相手 PE が多くても 2 つであるため、通信コストを考えた場合に非常に有効であると考えられる。

計算領域の形状がある程度正方に近い場合は、図 3.4のような分割が一般に行われる。この手法を Static-B とする。一つ目の手法と比べてある PE が通信を行う相手先 PE 数が増えるが、通信量自体は少なくすることができる。具体的には図 3.2で示した。この図で計算領域全体のサイズを縦が a、横が b とした場合、例えば Static-A での PE3 は $a + a = 2a$ の長さの境界についてデータ交換を行う必要がある。同じ条件で Static-B の場合では、 $a/2 + a/2 + b/4 = a + b/4$ になる。よって $a \approx b$ の場合では、 Static-B の分割が



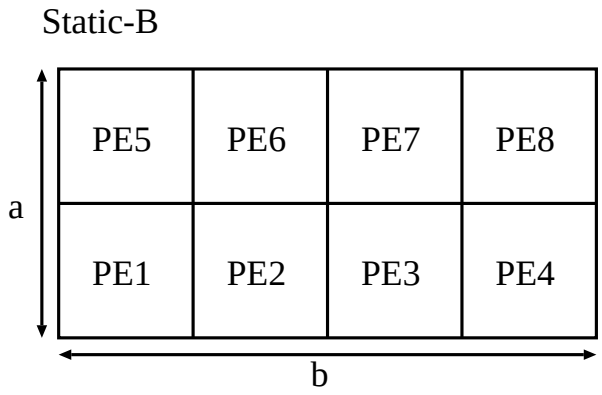


図 3.2: 通信量

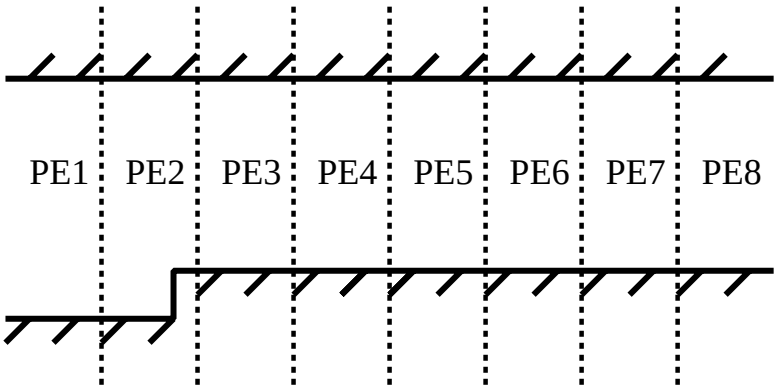


図 3.3: Static-A での各 PE の受け持つ領域 (8PE 使用時)

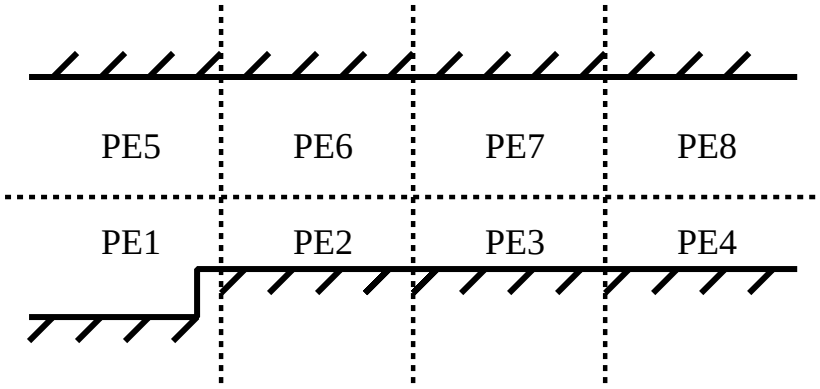


図 3.4: Static-B での各 PE の受け持つ領域 (8PE 使用時)

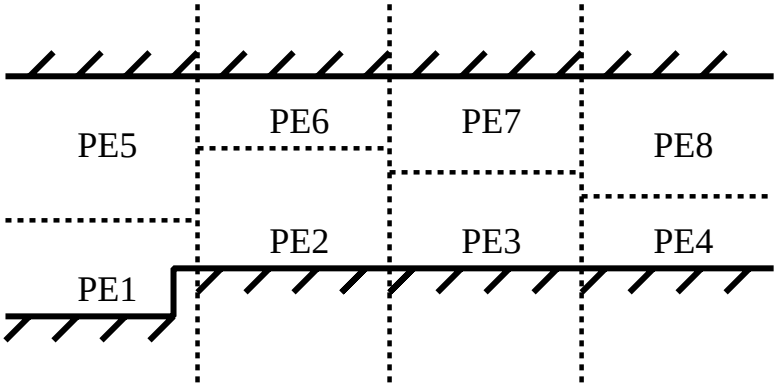


図 3.5: Static-C での各 PE の受け持つ領域 (8PE 使用時)

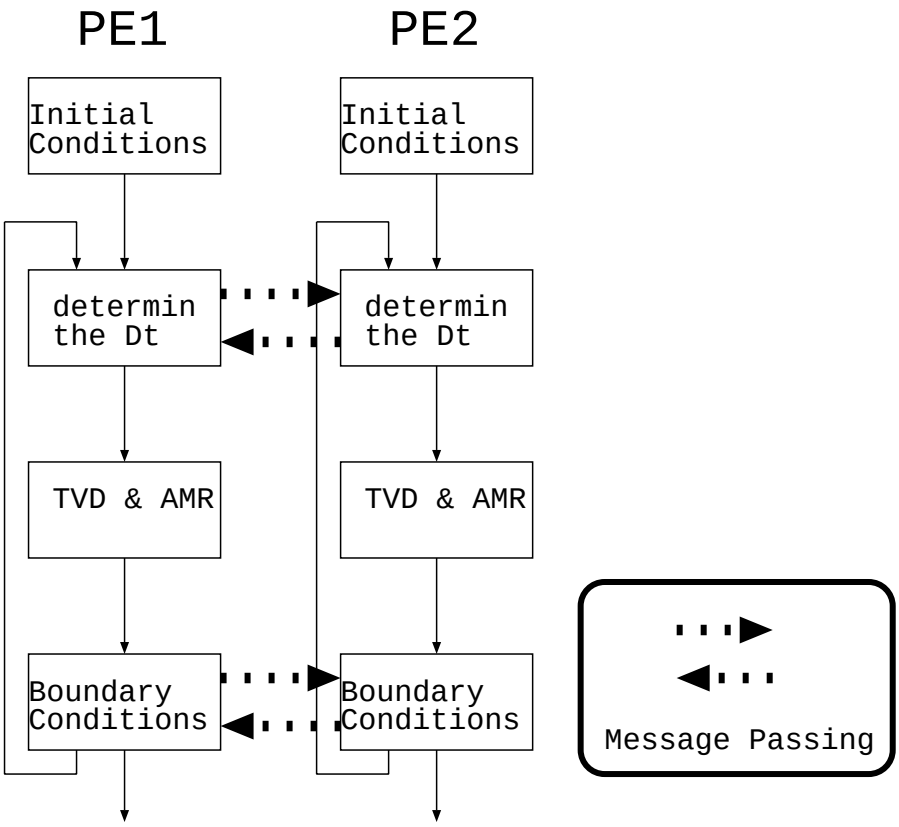


図 3.6: 静的領域分割法の流れ図

優位であると考えられる。

計算領域内で局所的に負荷量が増減している場合には、図 3.5のような分割形状を用いる。これを、 Static-C と呼ぶ。この図では、 PE1,PE6 が受け持つ領域付近に大きな負荷がある事を仮定して、これらの PE の受け持つ領域のサイズを小さくし、負荷量が極端に大きくなることを避ける。通信の複雑さや、それに伴う通信時間の増加が考えられるが、負荷分散という観点からこの手法は優位であると思われる。

3.1.2 計算の流れ

以上の手法について実際の計算の流れを示す。通信先 PE と通信が行われる領域の記述はそれぞれの手法で異なるが、計算の流れとしては、３つとも同様に図 3.6で示したものとなる。この図では 2 つの PE を用いた場合のものについて記述している。この図の中で実線はプロセスの流れを示しており、点線はメッセージパッシングを行なうタイミングを表している。簡単に流れを説明すると、最初に初期条件の設定を行なう (図中の Initial Condisions)。次に、 Δt の決定を行なう。この際、使用 PE 中で計算されるもっとも小さい値を Δt として設定するので、 PE 間でメッセージパッシングが必要である。その後 TVD 法と AMR 法を解く。これは、各 PE で完全に独立に計算を進める。最後に PE 間でオーバーラップしている部分の物理量を交換しあい、境界条件を設定する。以上が静的領域分割法の流れである。また、それぞれの PE で交換するデータは初期格子のみとした。これは、並列計算を行なう場合に問題となる通信量をできるだけ少なくするためにこのような事を行なった。

3.2 動的領域分割法

計算領域内で局所的に負荷量が増減する場合、予め割り当てられた領域のみを担当する静的領域分割法では、各 PE の負荷量の不均一化が起こることが考えられる。 Static-C はある程度負荷分散という点で考慮を行っているが、動的に負荷量が変動する問題に対しては適応しきれない事が想像できる。特に流れ場の状況に応じて計算格子を局所的に細分化する適応格子法などでは、この負荷量の不均一化が大きな問題となり、これをできるだけ無くすことがパフォーマンスを向上させるために必要な手法となる。前書きでも述べたが、この問題を解消するためにこれまで数例の手法が研究されている。しかしなが

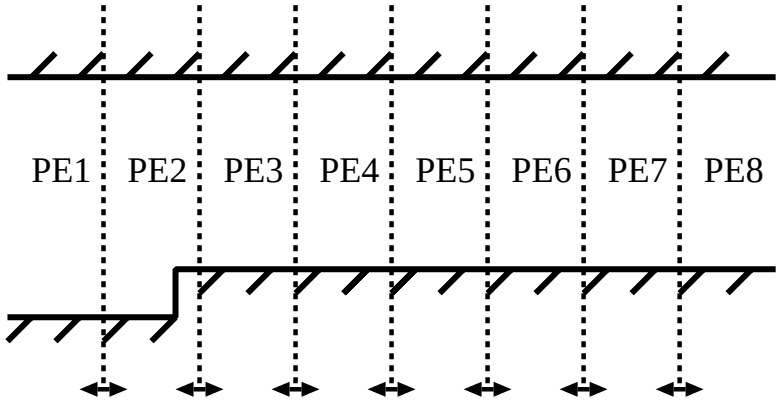


図 3.7: Dynamic-A での各 PE の受け持つ領域 (8PE 使用時)

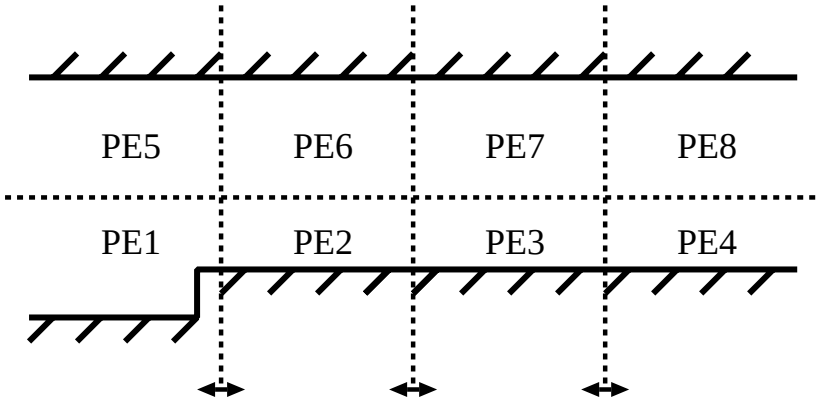


図 3.8: Dynamic-B での各 PE の受け持つ領域 (8PE 使用時)

ら、その中で通信量を考慮に入れた手法が少ないことと、通信量を考慮したとしても非常に複雑な過程を要するものであるため、本論文では、通信量を考慮し、比較的容易に実装可能な手法を提案する。

3.2.1 領域分割形状

図 3.7は領域の分割形状を Static-A と同様に行い、動的に領域形状を時間ステップが進むにつれて変動させる手法の領域分割図である。この手法を Dynamic-A と呼ぶこととする。ここで、領域の変動についてはある程度の制限を加え一次元的とする。この制限により各 PE は、受け持つ計算領域の形状が変動したとしても、常に同じ PE とデータを交換することから通信の手続きを簡略化できる。またさらに、この手法は各 PE の受け持つ領域を動的に伸縮させることから、負荷分散が動的に行うことになり、局所的な負荷量の増大が抑制されると期待できる。

また、図 3.8で計算領域形状が比較的正方に近い場合を考慮した領域分割図を載せた。この分割手法を Dynamic-B とする。この手法も、動的に各 PE の計算領域を変動させるが、 Dynamic-A と同様にその変動方向を一次元的な方向にのみ制限する。そのため計算が進むにつれて複雑な形状になっていくと考えられる計算領域を、長方形に保つ事が可能となる。また、ある PE に着目した場合、通信を行う相手 PE が流れ場の変動（時間の進行）に影響せず常に同じであるため、通信の複雑さの点で優位であると考えられる。また、負荷分散についても、 Dynamic-A と同様に有効な手法であると考えられる。

3.2.2 負荷分散手続き

前節で負荷量により領域形状を伸縮させる手法についてその領域の分割形状について言及した。ここではその領域を決定するための負荷量の定量化について言及する。適応格子法を並列計算する場合、細分化格子が生成される領域は流れ場の状態に依存するため、ある PE が担当している領域に細分化格子が集中し、その PE の負荷が非常に大きくなることが一般的に考えられる。そのため、各 PE の負荷量を正確に把握しておく必要がある。負荷量を定量化するにあたり考慮に入れるべき要素は以下の 2 つである。

1. 各 PE の受け持つ領域内にある格子数。

2. 格子の細分化レベル。

(1) については、各 PE の持つ負荷量は格子数と密接な関係があることから必要な要素である。(2) は、格子の細分化が進むにつれ計算量が増えるために必要な要素である。Ryu らの手法 [6] についてタイムスケジューリングをみると (図 2.4)、細分化格子は計算回数が増えていることがわかる。具体的には、レベル 0 の格子は 1 回、レベル 1 の格子は 2 回、レベル 2 の格子は 4 回の計算を施すことで、1 タイムステップ、つまり Δt の計算が行なわれる。また、一つの格子が細分化された場合には２次元の場合４つの細分化格子が生成されるため、一般的には細かい格子つまり計算回数の多い格子が数多く生成される。よって (2) の要素も考慮に入れる必要がある。

以上の２つの要素を考慮に入れ、以下の式 (LB: Load Balancing indicator) を用いることで、各 PE における負荷の定量化をおこなった。

$$LB_{PE} = \sum_{level=0}^{LEV} 2^{level} N_{level}. \tag{3.1}$$

ここで、LB_{PE} は、プロセッサ番号 PE の LB の値、LEV は適応格子法における最大細分化レベル、N_{level} は格子の細分化レベルが $level$ である格子の数を表している。具体的に最大細分化レベルが 2 レベルの場合には、以下の式で LB の値を求める。

$$LB_{PE} = N_0 + 2 \times N_1 + 4 \times N_2. \tag{3.2}$$

この値が各プロセッサで平均化されれば、負荷量が平均化されパフォーマンスの向上につながると考えられる。実際にこの負荷量の平均化は、各 PE の受け持つ領域を大小させることで行なう。図 3.9は Dynamic-A を用いた場合の概念図である。この図では、PE2 が非常に大きな LB の値を持っていると仮定しているため、PE2 の新しい領域はかなり小さくされている。また逆に、PE4 は比較的 LB の値が小さいと仮定されていて、新しい領域は古い領域よりも大きな面積を持つように制御された。この操作を行なうことで、各 PE の持つ負荷量 LB の値を平均化する。

3.2.3 再領域分割手法

前節において、各 PE における負荷を定量化し、その平均化の方法を示した。しかしながら、本研究では非定常流を対象としており、それには衝撃波などの数値的な不連続面が新たにできたり、移動したり、消滅したりという性質がある。また、そのような振舞いにともない、細分化が施された格子が増えたり減ったり、または、移動したりする。そのよ

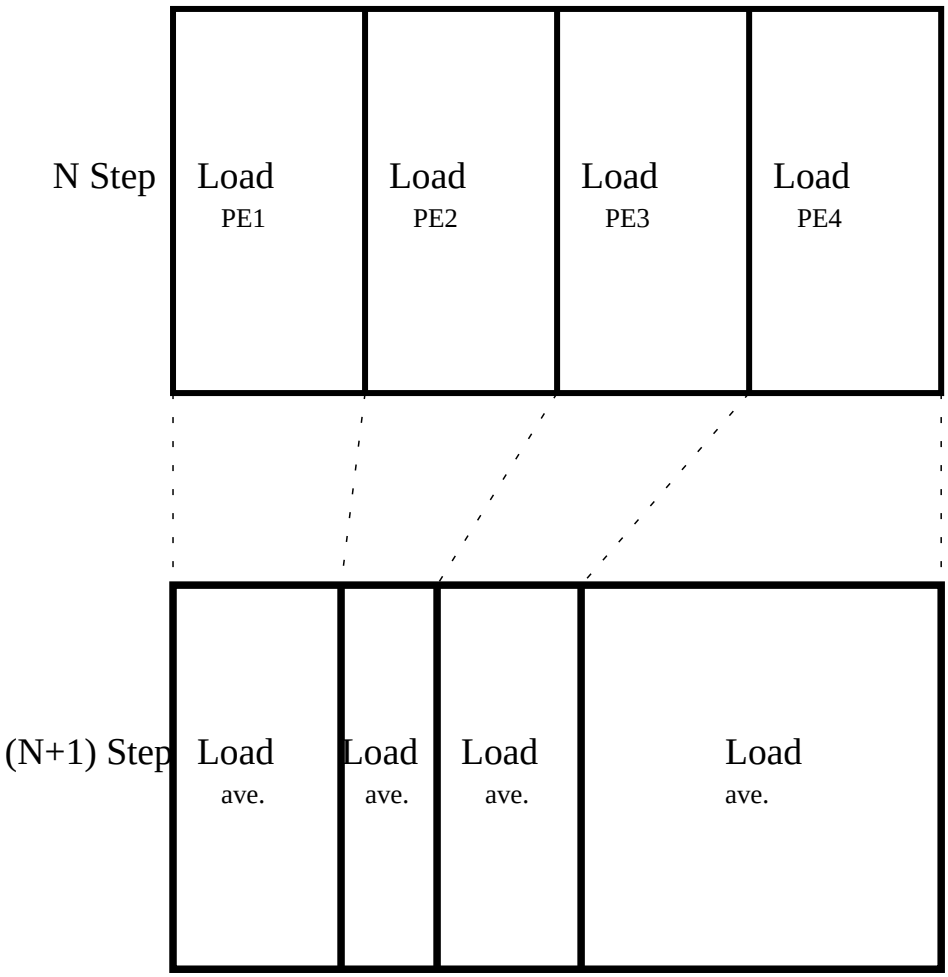


図 3.9: 各 PE の計算する領域

うな状況が考えられることから、各 PE の負荷量が時間ステップが進むにつれ変化することが容易に予想できる。そこで、時間ステップに対して、動的に各 PE のもつ領域を大小させることで、負荷量 LB を平均化することを考える。この部分の手続きとしては次のステップを踏む。

1. 各 PE 内で LB の値を計算する。
2. 各 PE は、 LB の値をある PE に送る。
3. 各 PE の LB 値を渡されたある PE は、 LB の平均値を求める。
4. この PE の中で、各 PE の現在受け持っている領域と、 LB を平均化するために求めた新しい領域の差分の情報を得る。
5. (4) で求まった、古い領域と新しい領域の差分の情報から、新しい領域に対して物理量のデータをメッセージパッシングする。

このステップの中で、 (5) のステップが少々複雑である。ここで行なうメッセージパッシングのパターンとして、以下のパターンが考えられる。

1. メッセージパッシングを全く行なわない PE。
2. 隣接する片側の PE からデータを受け、もう片側の PE にデータを送る PE。
3. 隣接する両隣の PE からデータを受ける PE。
4. 隣接する両隣の PE にデータを送る PE。

(1) については、古い領域と新しい領域の始点と終点の x 方向における位置（つまり、左端と右端の物理空間全体における格子の座標）が同じである場合はメッセージパッシングは行なわない。 (2) については、左側からデータを受けるか、右側からデータを受けるかと言う 2 つのパターンがある。 (3)、 (4) に関しては、隣接する PE に対して、データを受けるだけ、もしくは、データを送るだけといった、同じ操作を施すパターンである。この部分は、 PE1 から順番に、新しい領域に対応させるような手法を用いている。

ここまで、時間ステップにおいて動的に各 PE の受け持っている領域を大小させる手法について示した。この付加的な部分を静的領域分割法の中に組み込むことで、前節で定

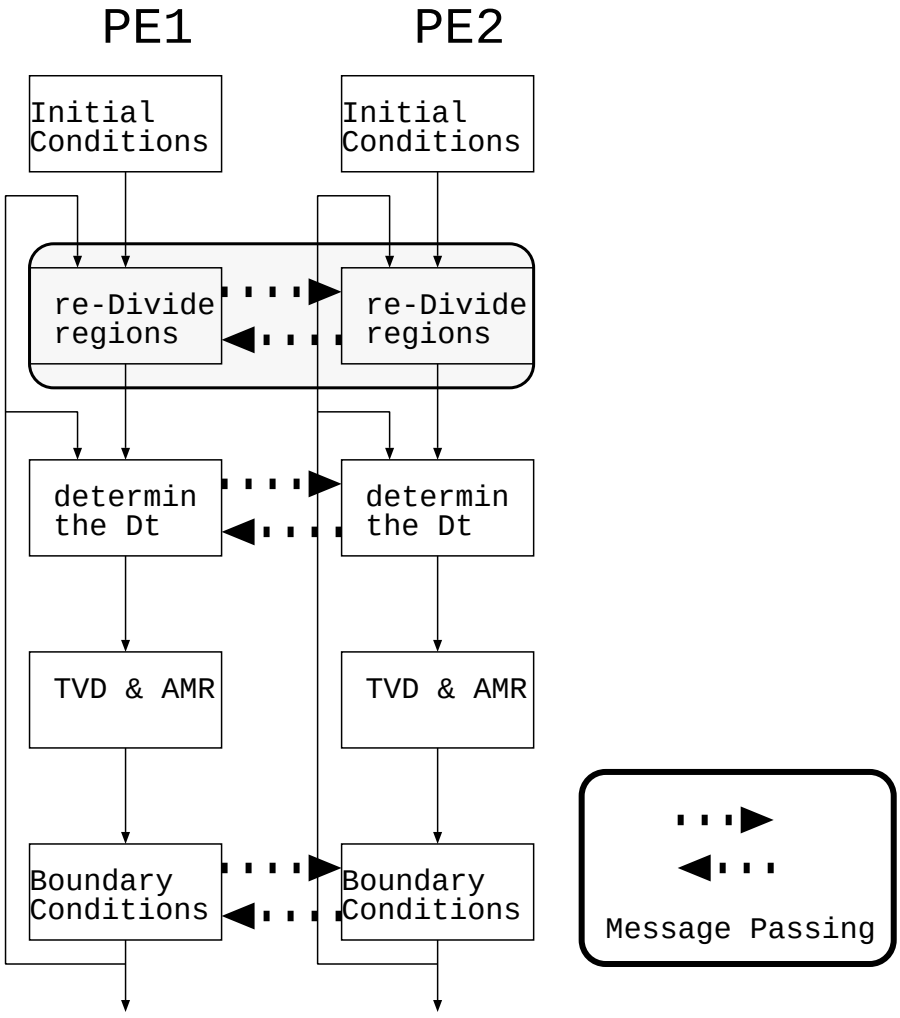


図 3.10: 動的領域分割法の流れ図

量化した各 PE における負荷量を平均化することが可能になる。この手法を組み込んだフローチャートを図 3.10に示した。これは図 3.6にここで示した新しい手法を付加したものである。付加した部分は、マスクのかかっている部分である。

また静的な手法と同様、通信の対象となる格子を初期格子のみとして、通信量を増加させない手法を取った。

3.2.4 再領域分割のタイミング

動的領域分割法を用いる際、領域の再決定のタイミングは非常に重要である。現状の領域の分割状況が良いものであるにもか変わらず、変更を行なったり、または、再領域分割を行なうコストが再領域分割後のパフォーマンス向上に結びつかない場合などは、再領域分割を行なう適切なタイミングではないことを示すものである。そのため、このタイミングを何かしらの条件から導き出す必要がある。

本研究では、近年の並列計算機のハードウエア性能の向上から、メッセージパッシング量をそれほど重要視しないで、簡易的にこのタイミングを知る手法を見い出した。これは、各 PE の負荷量をモニターし、最大負荷量と最小負荷量の PE の負荷量比が、ある程度以上になった場合に再領域分割をするというものである。ここでこの負荷量は、前述した LB という値で評価を行なった。

$$\frac{LB_{max.}}{LB_{min.}} = LoadRatio \tag{3.3}$$

ここで、$LoadRatio$ が、ある閾値以上になった場合、再領域分割を行なう。本論文ではこの手法を「タイミング最適化」と呼ぶ。

第 4 章

結果

この章では、これまで述べてきた各種並列化手法について実験を行ない、その結果を示す。実験はステップ付パイプ内流れの問題を取り上げる。また、並列計算機についても簡単に触れる。

4.1 並列計算機

使用する並列計算機として SGI 社の CRAY T3E を用いた。これは、ノード間が 3 次元トーラス型インターコネクト・ネットワークで結合されている分散メモリ型の並列計算機である。1 ノード 1PE の構成となっており、本学では 128 ノードが結合されている。CPU はは DECchip 21164a(300MHz, 600MFLOPS) を用いており、1 ノードあたりのメモリは 64MB となっている。

本研究では、1 〜 8PE を用いた場合について検討を行なった。また、メッセージパッシングライブラリとして、MPI(Message Passing Interface) を用いた。

4.2 初期条件

本研究では、段差のある風洞にマッハ 3 の風を流し込んだ場合の物理状態を解析する問題について実験を行なった。この問題は、強い衝撃波の解析問題として、Woodward [10] らの実験が有名である。彼らの実験では、強い衝撃波が徐々に成長し、さらに固体壁に衝撃波がぶつかり反射などを繰り返す状況が見られた。

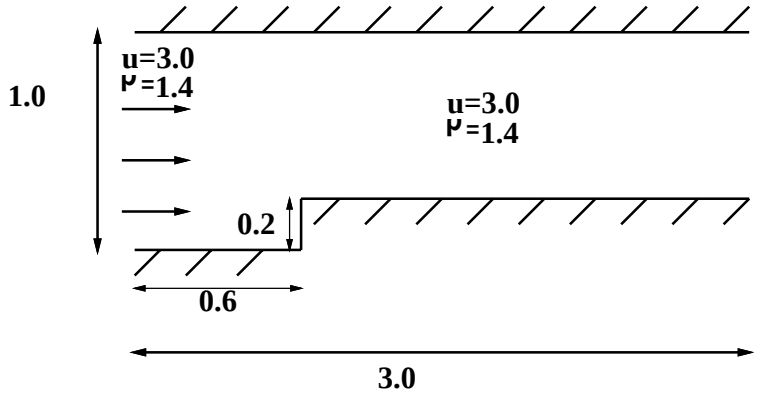


図 4.1: 初期条件

本研究の実験の中では、具体的に次のような条件を与えた (図 4.1)。風洞は、入口の高さが 1、長さが 3 の風洞に入口から 0.6 の位置に 0.2 の高さの段差をもうけた。実験では、この風洞に入口から速度 3 の風を流し込む。入口の境界条件は流し込みの条件で、出口は物理量勾配 0 の条件を用いた。また、上下に関しては、固体壁の条件で計算を行なった。初期条件は、密度 1.4、圧力 1.0、横方向速度 3.0、縦方向速度 0.0、比熱比は 1.4 を用いた。 また、この計算空間に対して横方向に 120、縦方向に 40 個のレベル 0 格子（初期格子）を配置した。

図 4.2は、以上の条件の元で、初期格子のみで解いた結果である。等高線、色は密度を表しており、赤くなるほど密度が高くなっていることを示す。上からそれぞれ、 100 step、 300 step、 500 step、 700 step の状態で、実時間で表すとそれぞれ、 t=0.411, 1.223, 2.055, 2.926 となる。流れ場の様子としては、風洞の中で段差を設けた近辺で強い衝撃波が発生し、時間が進むにつれ衝撃波が成長しているのがわかる。また、$t = 1.2$ で衝撃波が上側の固体壁とぶつかることで反射していることがわかる。さらに時間を進めると (t=2.0)、下側の固体壁においても衝撃波が反射していることを確認できる。これらの結果は、 Woodward らの結果と非常に良く一致している。

4.3 静的領域分割法について

はじめに、一次元的静的領域分割法を用いた場合について結果を載せる。計算条件は、段差つき風洞モデルで、 100 ステップまでの計算を行なった。このステップ数に対応す

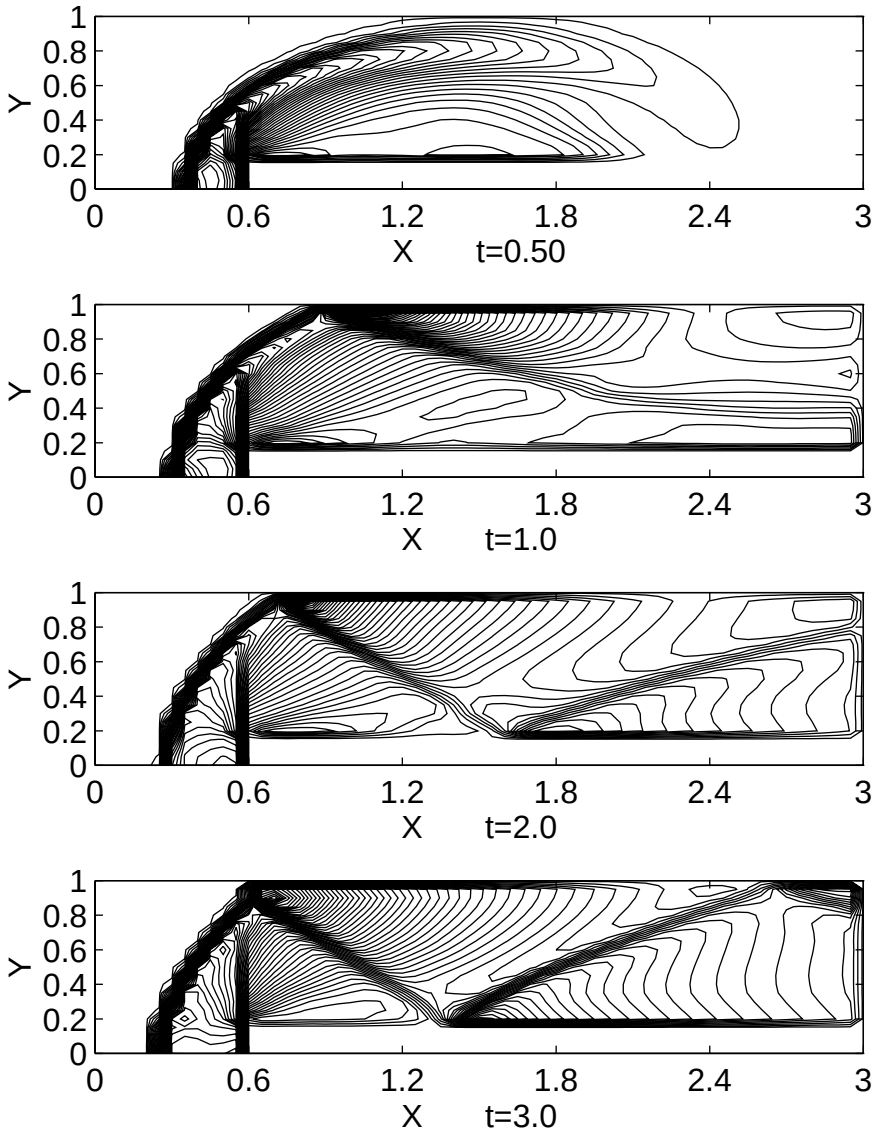


図 4.2: 流れ場

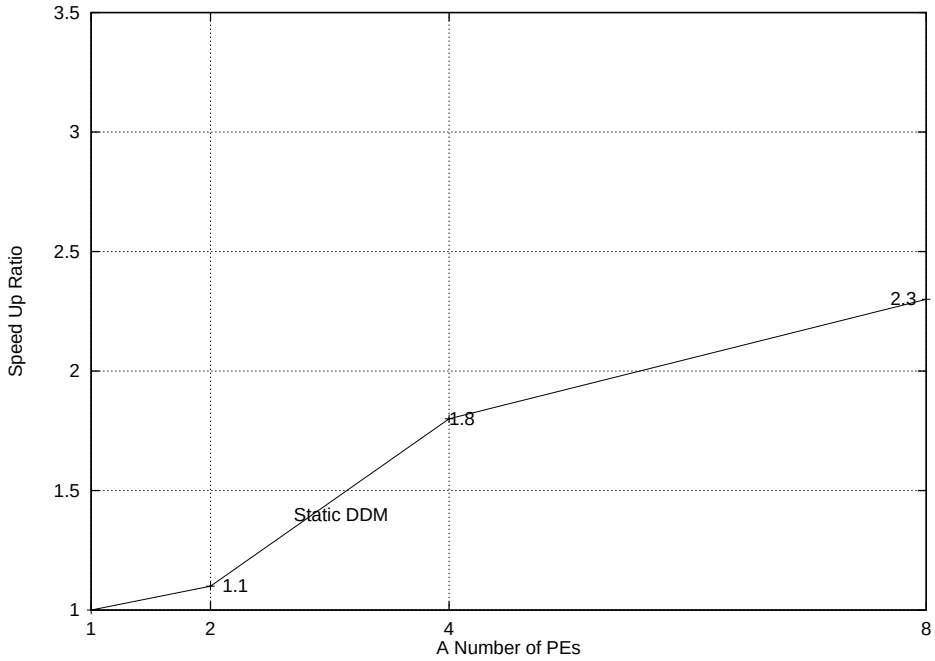


図 4.3: Static-A における速度向上比 (0-100Steps)

る実時間は約 t=0.4 である。これは図 4.2の最上位の図の状態に対応しており、流れ場の変動が激しい状態である。

4.3.1 t=0.4 までの計算 (100 ステップ)

Static-A を用いて $t = 0.4$ までの速度向上比に関する結果を図 4.3に示した。この図で、横軸は使用 PE 数、縦軸は速度向上比を表している。2, 4, 8 個の PE を用いて並列計算を行ない、これらの PE 数に対応する速度向上比はそれぞれ、1.1, 1.8, 2.3 倍となった。さらに、各 PE の実行時間についての実験を行なった。この実行時間は、全計算時間から、通信に関する時間や同期待ちの時間などの通信コストを省いたものである。図 4.4がその結果である。この図は、8PE を用いて並列計算を行なった場合の結果である。この図で、縦軸は実行時間、横軸は PE の番号を表し、20 ステップ毎の実行時間のスプリットタイムを示した。この図をみると、20 〜 40 ステップの辺りから、PE2 の実行時間が増え始めていることがわかる。また、40 ステップを越える辺りから、PE1 と 3 の実行時間が増え始め、それ以上のステップ数になると、PE1, 2, 3 の実行時間が、非常に増えていることがわかる。しかしながら、PE 番号が 6 以上のものについては、100 ステップの

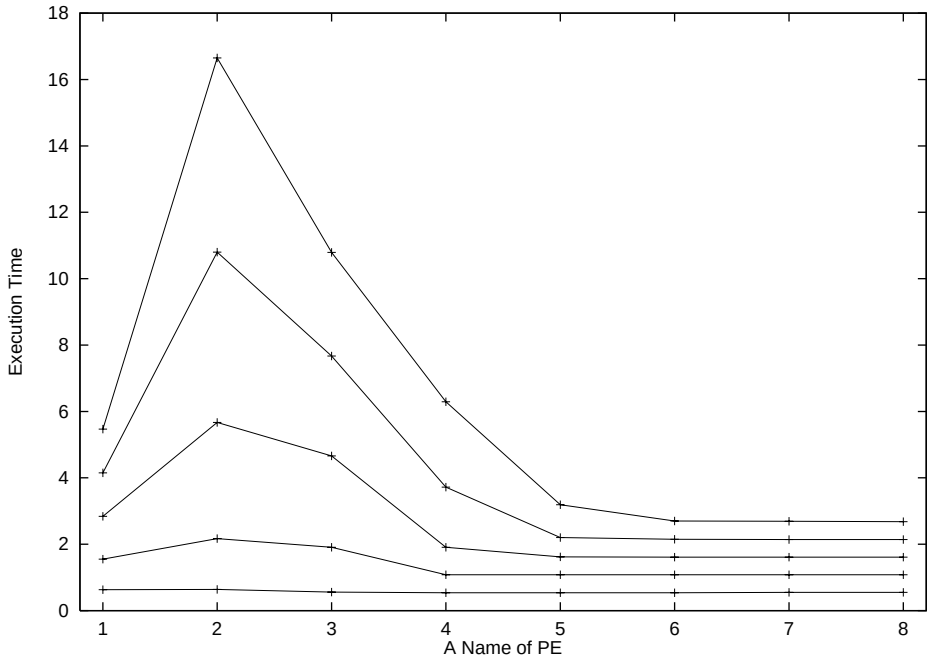


図 4.4: Static-A を用いた場合の各 PE の実行時間 (0-100steps)

計算を実行している間は、実行時間の増減具合に変化が見られないことがわかる。 100 ステップを終了したところで、 PE1 は、 PE6 以上のものと比べ 2.0 倍、 PE2 については 6.2 倍、 PE3 については 4.1 倍程度の差があることがわかる。

また、 Static-B, Static-C についても同様に、実行時間について図を載せる。図 4.5は Static-B に関する結果である。この手法を用いた場合は、 PE1 と PE6 で大きな負荷がかかっていることがわかる。 100step を終了した時点でもっとも負荷がかかっている PE1 と、もっとも負荷が少なかった PE4 の間で、約 7 倍程度の差が出ていることがわかる。

図 4.6では Static-C の結果を示したが、この場合には、 PE1, 2, 5 に大きな負荷がかかっている様子がみられる。しかし、 Static-A,B と比較した場合、最小負荷 PE と最大負荷 PE の差が緩和されている。具体的には、最大負荷量を持つ PE5 と最小負荷量をもつ PE8 では、約 5 倍程度の差となっている。

次に、並列計算を行なった時の付加時間について触れる。データの通信時間や、同期をとる際の待ち時間などがこの時間に含まれる。今後この時間を「付加時間」と呼ぶ事にする。この付加時間は、全実行時間を T_{all} とし、各 PE の実行時間（メッセージ通信時間等

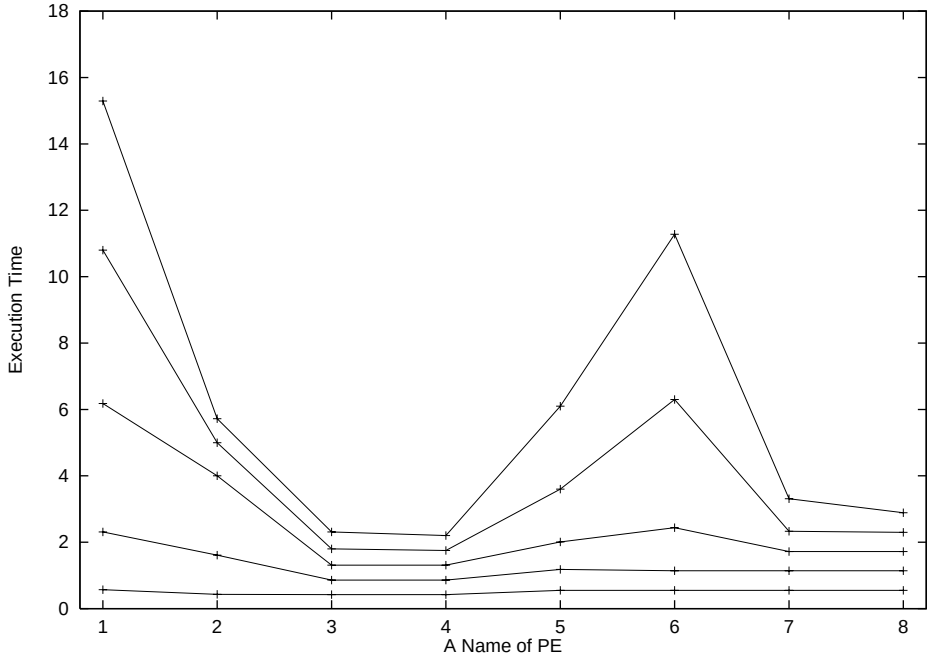


図 4.5: Static-B を用いた場合の各 PE の実行時間 (0-100steps)

を除く）を T_{exe} とした場合、

$$\text{付加時間} = \frac{T_{all}}{PE\text{数}} - T_{exe} \tag{4.1}$$

の様に定義する。図 4.7は Static-B で、図 4.8は Static-C を用いた場合の各 PE でのこの時間を示している。これらの図で横軸は PE 番号を示し、縦軸がこの付加時間を示している。また、付加時間で図 4.7では 16.3、図 4.8では 13.3 の位置に線が引かれているがこれは、各 PE での総実行時間の平均値を示している。 Static-B の場合 (図 4.7)PE3, 4, 7, 8 は比較的付加時間が多く、 CPU が実際に計算を行なっている時間よりも多い。それに比べ、 PE1 ではほとんどの時間が実際の計算に使われていることがわかる。 Static-C の場合では、 PE3, 4, 5 が比較的付加時間が多いことがわかる。表 4.1ではこれらの結果を定量的に示した。この表で Max. は、付加時間が総計算時間に対してもっとも大きい割合をもった PE とその割合を示している。 Min. は、それがもっとも小さい割合の PE とその割合である。 Static-B では、付加時間の割合がもっとも多いのは PE4 で 87% を占める。また、最も少ないのは PE1 で 6% の結果が得られた。 Static-C では、付加時間の割合がもっとも多いのが PE8 で 83%、少ないのが PE5 で 17% という結果が得られた。

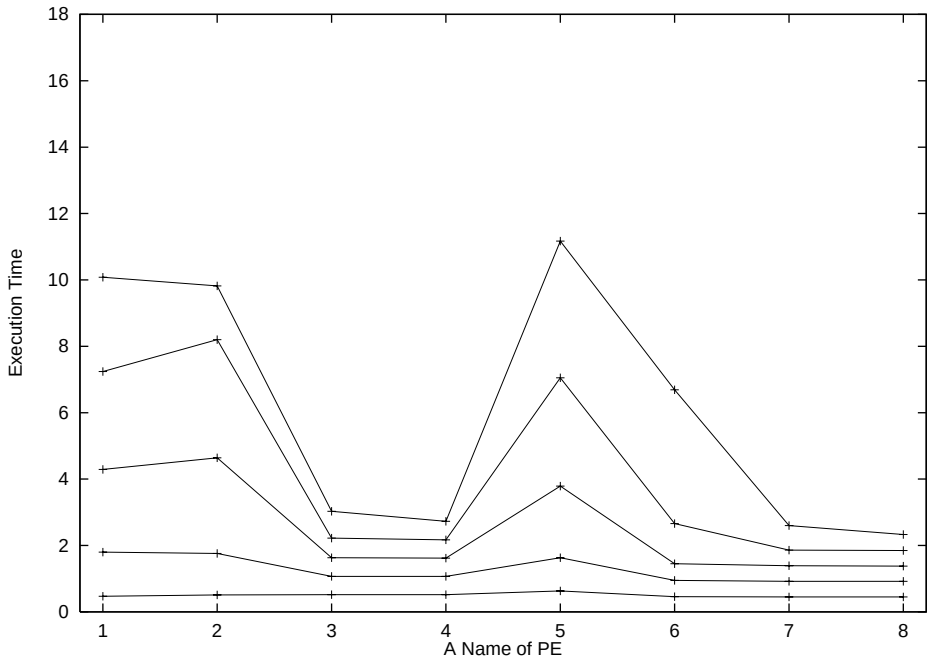


図 4.6: Static-C を用いた場合の各 PE の実行時間 (0-100steps)

	Max.	Min.	S.U.R.
Static B	87% (PE4)	6% (PE1)	2.7
Static C	83% (PE8)	17% (PE5)	3.3

表 4.1: 付加時間の割合 (0-100steps)

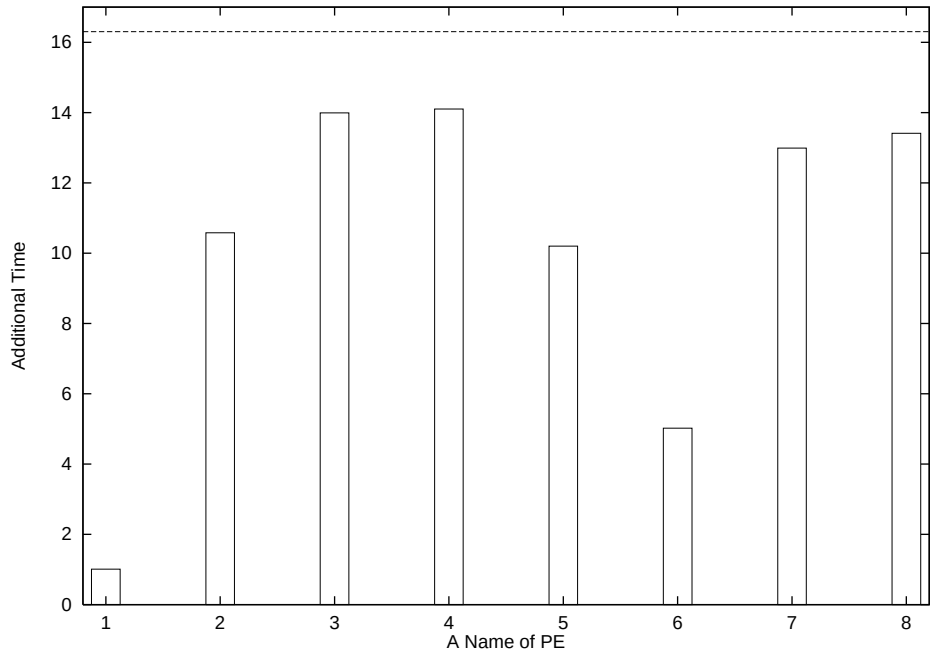


図 4.7: Static-B を用いた場合の各 PE の付加時間 (0-100steps)

また、 Static-B, Static-C については、 8 個の PE を用いた場合のみ、速度向上比を計測した。表 4.1で、 S.U.R. で示したところがこの結果であり、それぞれ 2.7 倍、 3.3 倍の結果が得られた。

4.4 動的領域分割法について

静的な領域分割法と同様に、 100 ステップまでの計算で速度向上比等の計測を行った。領域再分割のタイミングについては定期的な手法と、タイミング最適化を使ったもので行なった。

4.4.1 t=0.4 までの計算（100 ステップ）

まずはじめに、動的領域分割法を実際に用いて計算を行なうために定義した、 LB について述べる。図 4.9がその結果である。この図は、 8PE を用いて並列計算を行なった場合の結果を表している。この図で、縦軸は LB の積算値、横軸は PE の番号を表して

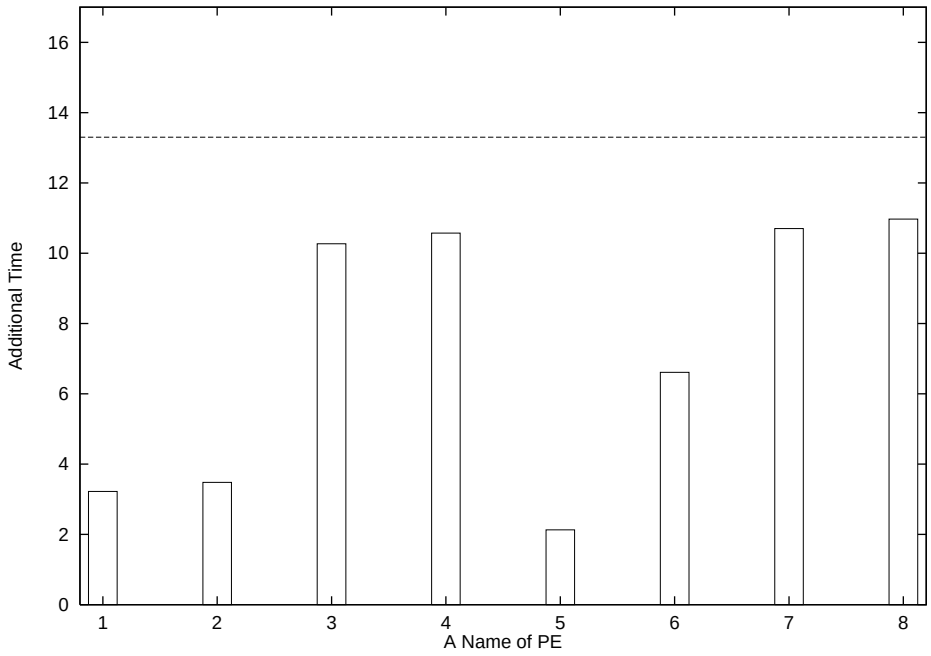


図 4.8: Static-C を用いた場合の各 PE の付加時間時間 (0-100steps)

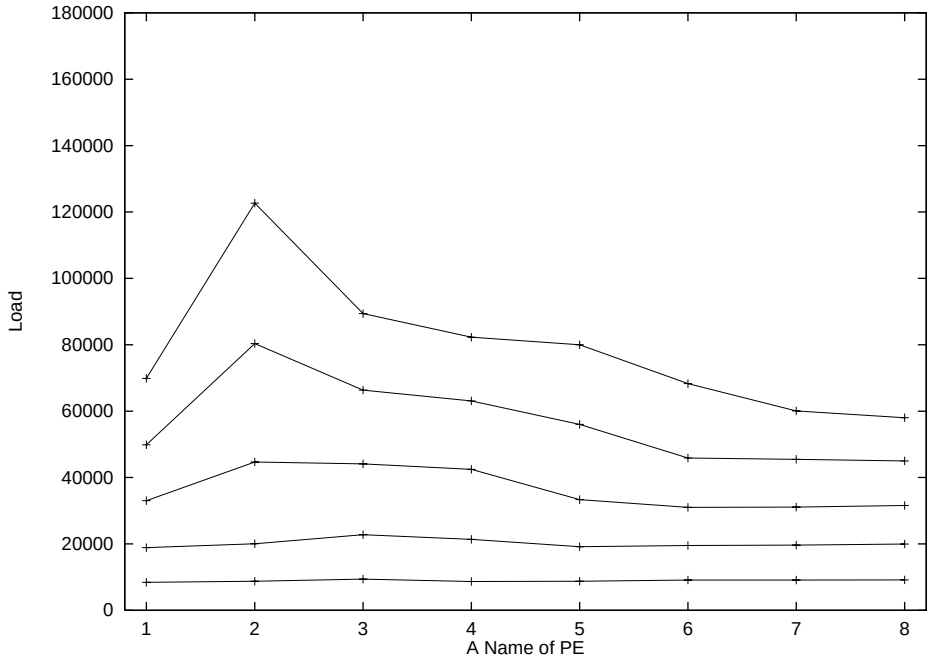


図 4.9: Dynamic-A を用いた場合の各 PE の負荷量 (0-100Step)

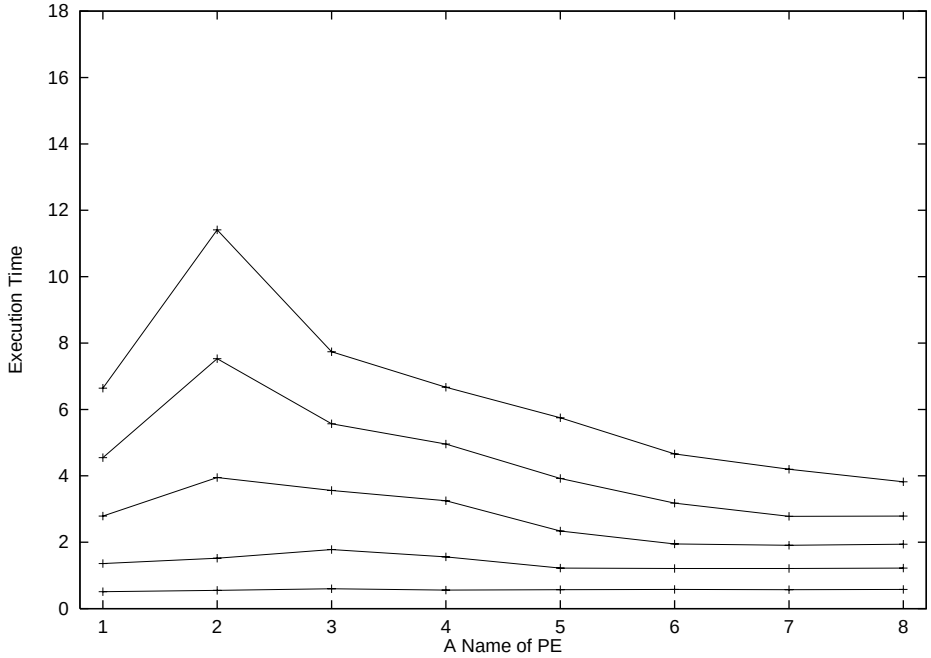


図 4.10: Dynamic-A を用いた場合の各 PE の実行時間 (0-100Step)

おり、 20 ステップ毎の LB の積算値を示した。この図から、 60 ステップを越える辺りから、 PE2 の負荷量が顕著に増え始めていることがわかる。 100 ステップを終了したところでは、 PE2 は、 PE8 などと比べ 2.2 倍程度の差があることがわかる。

次に実際の PE の実行時間についての結果を載せる。図 4.10がその結果である。この図で横軸は PE 番号、縦軸は実行時間を示している。このグラフで示した値は、静的なものと同様、メッセージパッシングの時間や、このプロセスで発生する「待ち」の時間を除いており、純粋に PE が計算を行なっている時間を示している。各線はそれぞれ 20 ステップ毎の積算時間を示しており、 100 ステップまで示している。 100 step を終ったところで、 PE2 が最大負荷量を持つが、最小負荷量を持つ PE8 と比べ約 2.8 倍程度の差がある。

次に、 Dynamic-A を用いた場合の速度向上比を述べる。図 4.11は、定期的なタイミングで再領域分割を行なった場合の結果である。この図で横軸は使用 PE 数、縦軸は PE を一つ用いた場合の速度を 1 とした場合の速度比である。この図の中で実線が 3 本と、点線が 1 本描いてあるが、実線は、 100 step 中にそれぞれ、 2,25,49 回の再領域分割を行なった場合の結果である。点線は、 0 回、すなわち Static-A の結果で、パフォーマン

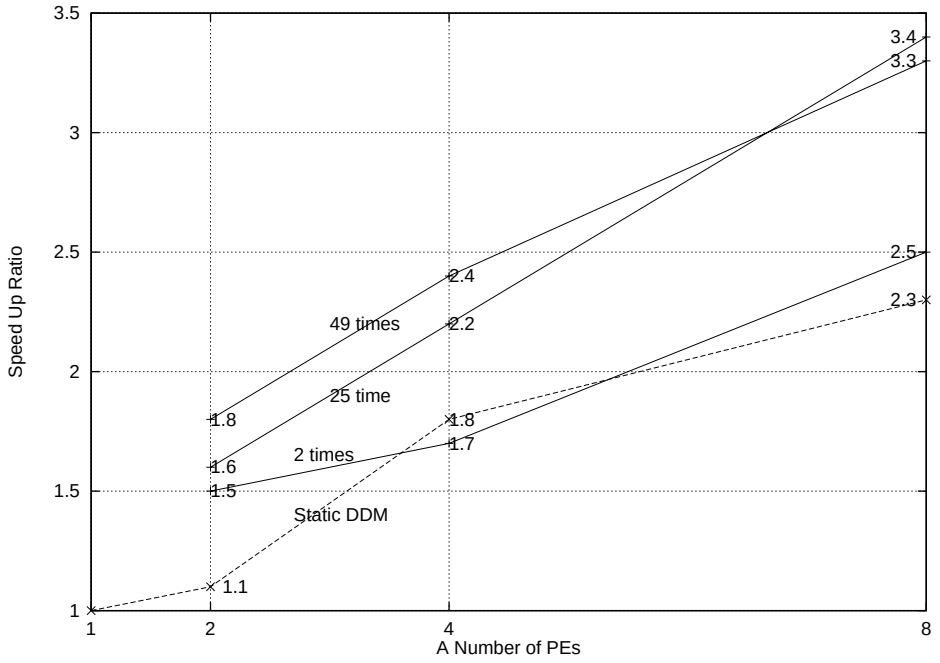


図 4.11: Dynamic-A での速度向上比 (0-100Step)

ス評価の参考のため記述した。この結果より、 8PE 用いた場合、最大で 3.4 倍程度のパフォーマンスが得られていることがわかる。

また、 Dynamic-A の手法に対して、タイミング最適化の手法を付け加えた。 8PE 使用した時の結果を表 4.2で示す。この表で、「負荷量の差」は最大の負荷量を持つ PE と最小の負荷量を持つ PE の負荷量の比で、それぞれ、 1.1, 1.2, 1.3, 1.4, 1.5, 2.0, 3.0, 4.0 倍以上になった時に領域の再分割を行なう手法をとったことを示す。また、 S.Up Rat. は、差がそれぞれの場合の速度向上比を示しており、差が 1.1 倍〜 1.5 倍の場合は 3.3 〜 3.4 倍の速度向上比が得られたことを示している。差が 2.0, 3.0, 4.0 の場合ではそれぞれ 3.1, 3.0, 2.8 倍の速度向上比が得られた。

次に、 Dynamic-B に関する結果を述べる。図 4.12は Dynamic-B 用いた場合の各 PE の実行時間の結果である。横軸は PE 番号、縦軸は実行時間を示している。 PE1, 7 で大きな負荷がかかっている様子がわかる。 100 step 終了後、最も負荷が小さい PE4 と、最も負荷が大きい PE1 との間には、約 2.7 倍程度の開きがある。また、 Static-B, Static-C と同様にして、付加時間に関する結果を載せる。図 4.13がその結果である。図 4.7, 4.8と同様、横軸が PE 番号、縦軸が付加時間、付加時間が 11.9 のところに引かれている線は、

負荷量の差	S.Up Rat.(8PEs)
1.1 倍	3.4
1.2 倍	3.3
1.3 倍	3.4
1.4 倍	3.3
1.5 倍	3.3
2.0 倍	3.1
3.0 倍	3.0
4.0 倍	2.8

表 4.2: 最大負荷量と最小負荷量の差と速度向上比

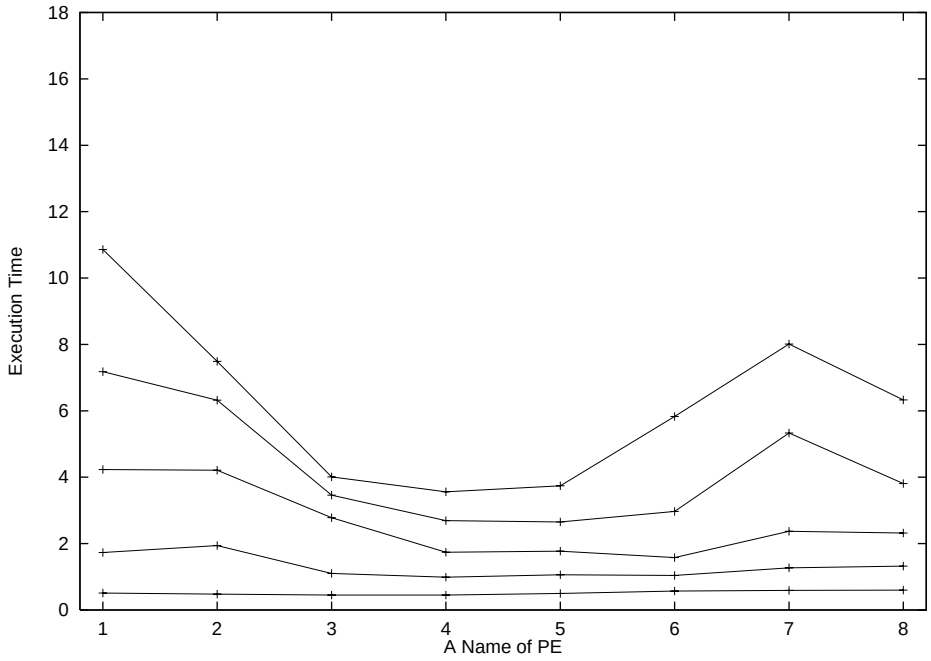


図 4.12: Dynamic-B を用いた場合の実行時間 (0-100steps)

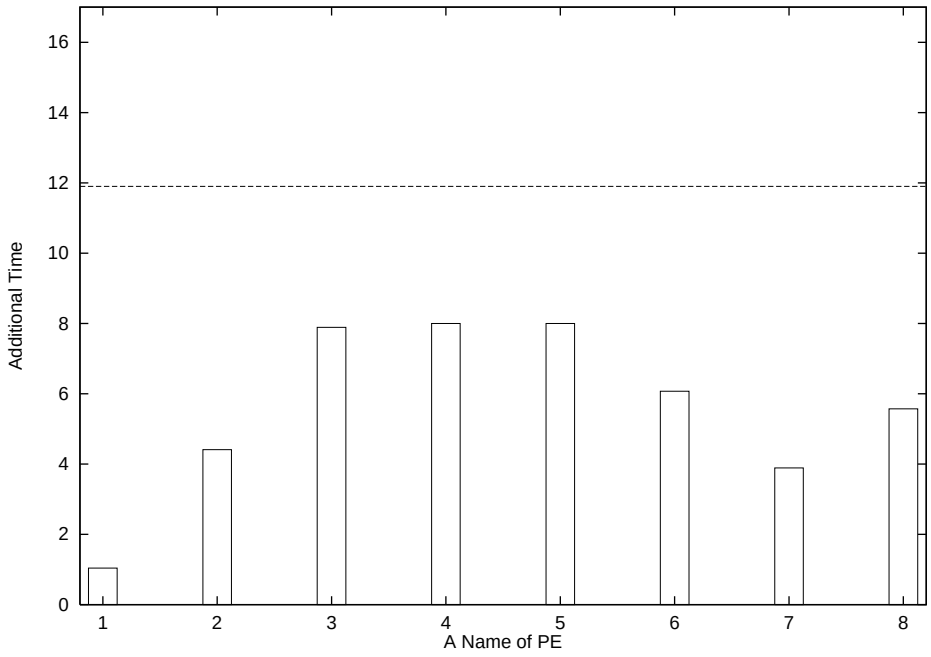


図 4.13: Dynamic-B を用いた場合の付加時間 (0-100steps)

	Max.	Min.
Dynamic-B	71% (PE4)	12% (PE1)

表 4.3: Dynamic-B を用いた場合の付加時間と速度向上比 (0-100steps)

各 PE の総実行時間である。この図を見ると、 PE1 が最も付加時間が少なく、 PE3,4,5 が付加時間を多く費やしていることがわかる。定量的には、表 4.3に掲げた通りである。 Max. が最も付加時間が多かった PE とその全体時間に占める割合で、 Min. は最も付加時間が少なかった PE とその全体に占める割合である。この表から、 PE4 がもっとも付加時間が多く 71% の割合を占めたことがわかる。一方、 PE1 は最も付加時間の割合が少なく 12% であった。さらに表 4.4では、 8 個の PE を用いた場合の Dynamic-B の手法の速度向上比についても触れた。これより、 8PE を使用した際に 100 step 中に 50 回の再領域分割を施すことで、 3.7 倍程度の速度向上比が得られていることがわかる。

再分割回数	速度向上比
100	3.60
50	3.70
20	3.71
10	3.68
0	2.70

表 4.4: Dynamic-B を用いた場合の再領域分割のタイミングと速度向上比 (0-100steps)

4.5 流れ場の変動が比較的緩い状態 (t=3.0 までの計算,700 ステップ)

前節で提示した結果は、流れ場の変動が比較的激しい状態でのものであった。ここでは、流れ場の変動が比較的安定している状態での結果を述べる。図 4.14に 700 ステップ (t=3.0) まで計算を行った際の速度向上比を載せた。この図で、横軸は使用 PE 数、縦軸は速度向上比を表している。実線が Static-A の結果で、濃い点線が Dynamic-A の結果である。また、薄い点線は線形速度向上比で、パフォーマンスの理想線を示している。 Static-A では 2, 4, 8 個の PE を用いる事で、 1.3, 2.0, 3.9 倍の結果を得た。 Dynamic-A ではそれぞれ、 1.9, 3.8, 6.2 倍の結果が得られた。

図 4.15には同条件での、各 PE の実際の計算時間を示した。この図の中で点線が Static-A の結果を示している。この結果を見ると、 PE2, 3, 4 付近で負荷量が多くなっている事がわかる。 PE2 は、 PE8 と比べ、 700 ステップの計算の中で 2.5 倍程度の差がついている事が分かる。

また、領域再配分のタイミングにタイミング最適化の手法を用いた場合の、 Dynamic-A に関する結果も同時に載せた。この結果を見ると、 PE2 で若干負荷量が多くなっている様子が見られるが全体的には負荷量がほぼ均等に分散されている様子が分かる。

最後に、 Dynamic-B を用いた場合の速度向上比とともに、表 4.5に速度向上比の結果をまとめた。 Dynamic-B について、この条件の元では、 6.4 倍の速度向上比が得られていることがわかる。

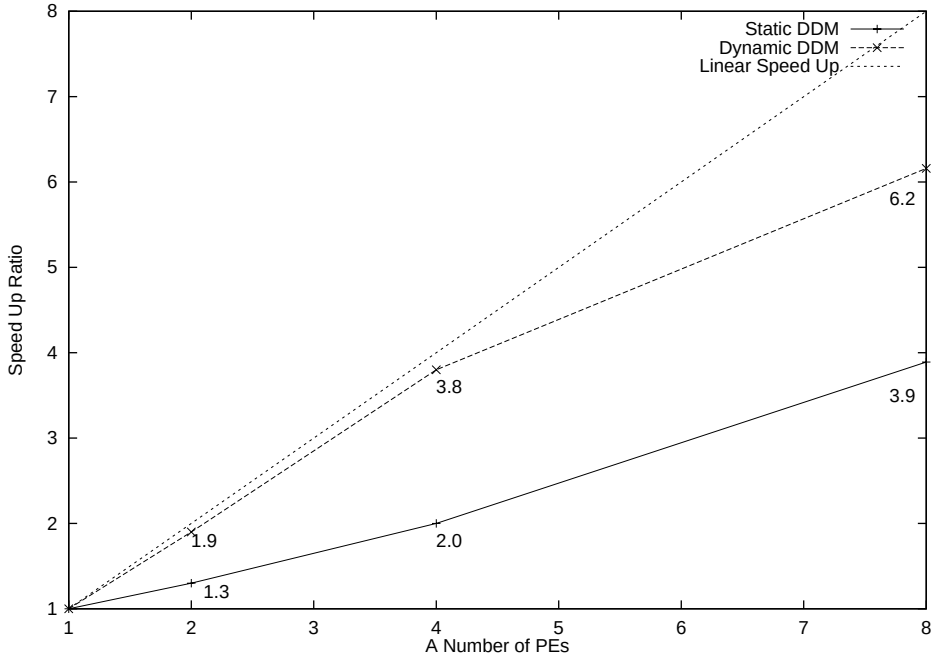


図 4.14: Static-A, Dynamic-A を用いた場合の速度向上比 (0-700Steps)

	0-100 steps	0-700 steps
Static-A	2.3 倍	3.9 倍
Dynamic-A	3.4 倍	6.2 倍
Dynamic-B	3.7 倍	6.4 倍

表 4.5: 8PE を用いた場合の速度向上比 (0-700)

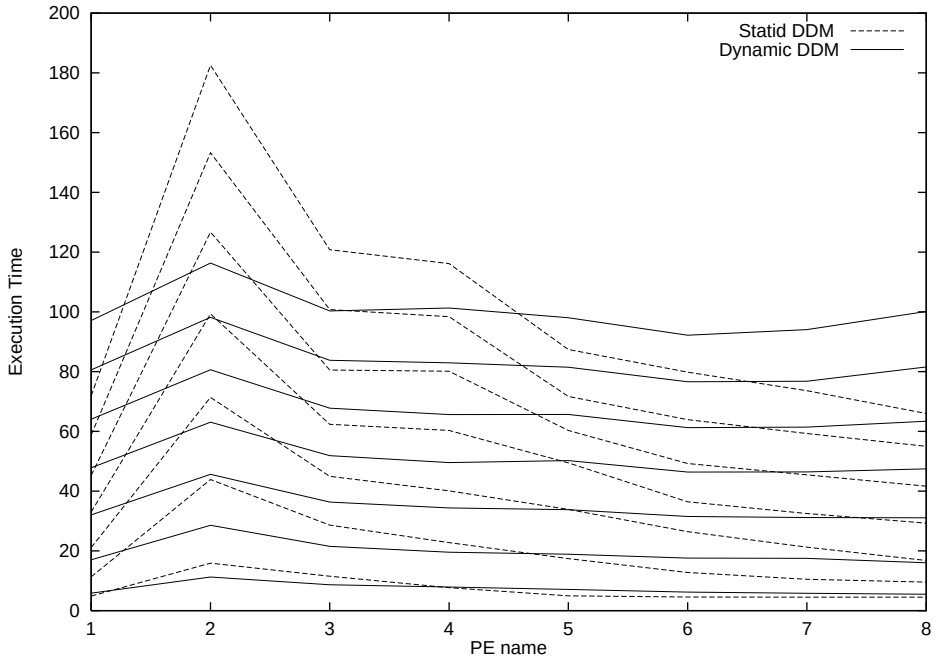


図 4.15: Static-A, Dynamic-A を用いた場合の各 PE の実行時間 (0-700Steps)

4.6 計算空間が大きい場合の結果

計算空間をやや大きめにした場合のパフォーマンス評価を行なうために、 Static-A, Dynamic-A, Dynamic-B を用いて、実験を行なった。具体的には、図 4.16のように、これまでの計算対象から下流部に対して管を伸ばし、格子数も 2 倍とした。計測対象は、これまでの実験と同様、 100steps 目と 700steps 目までの計算とし、結果を表 4.6に載せた。

	0-100 steps	0-700 steps
Static-A	2.0 倍	3.7 倍
Dynamic-A	4.07 倍	6.7 倍
Dynamic-B	4.50 倍	6.8 倍

表 4.6: 計算空間を大きくした場合の速度向上比

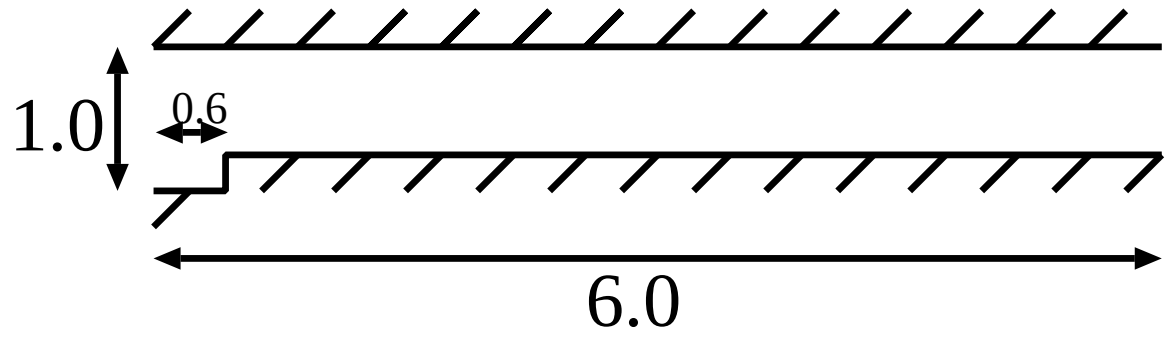


図 4.16: 大きい計算空間

この表から、 100 step 目までの計算では Dynamic-A を用いた場合 4.07 倍、 Dynamic-B を用いた場合では 4.50 倍、 700 step 目までの計算では Dynamic-A で 6.7 倍、 Dynamic-B で 6.8 倍のの速度向上比が得られた。

4.7 CM5 での結果

本論文では、 CRAY-T3E の結果を載せてきたが、過去に同様の計算を TMC の CM-5 で行なったので、この結果も載せる。図 4.17は、各 PE での実行時間を載せており、横軸が PE 番号、縦軸が負荷時間を除いた実行時間を示す。また、点線が Static-A を用いた場合の各 PE での実行時間で、実線が、 Dynamic-A を用いた場合の各 PE での実行時間である。 100 step までの計算で Static-A を用いた場合、 PE2 は 700 秒近くの実行時間がかかっており、 PE8 などと比較して 7 倍近くの差がある。一方 Dynamic-A では、もっとも負荷の多い PE2 ともっとも負荷の少ない PE8 で 3 倍程度の差がある。

速度向上比については図 4.18で載せた。横軸が PE 数、縦軸が速度向上比である。また、点線は Static-A の結果、実線は Dynamic-A の結果を示した。 PE を、 2、 4、 8 個使用した場合に、 Static-A では、 1.1, 1.7, 2.3 倍の結果で、 Dynamic-A の場合では、 1.8, 2.3, 3.4 倍の結果であった。

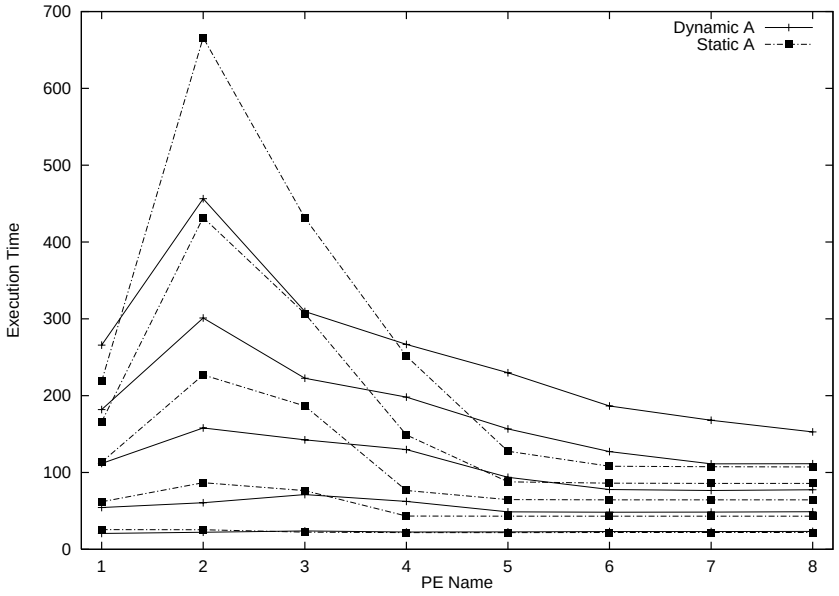


図 4.17: CM-5 での実行時間、 Static-A, Dynamic-A

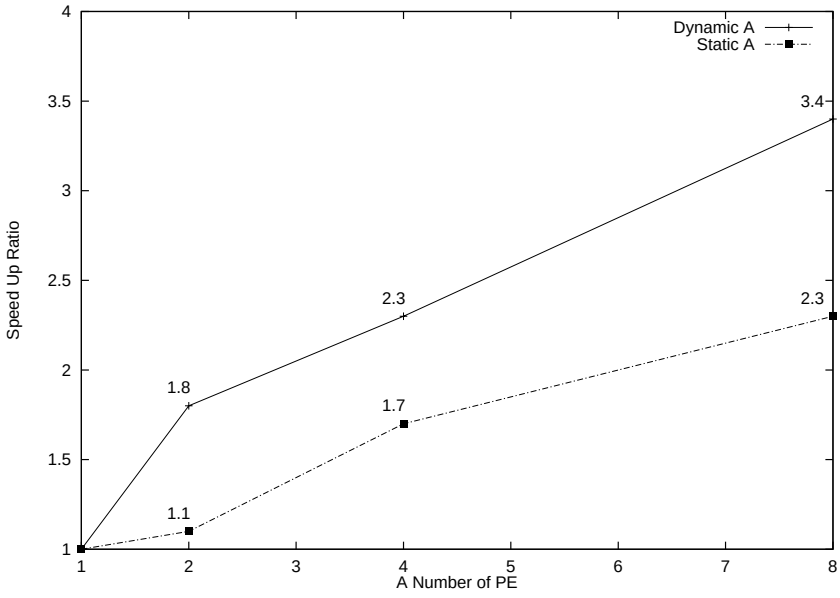


図 4.18: CM-5 での速度向上比、 Static-A, Dynamic-A

第 5 章

議論

適応格子法は計算領域内で局所的に細分化格子を与えることから、単純に領域分割法を用いて並列計算を行った場合には、局所的な部分で格子数が増大し、並列計算の際、その領域を受け持っている PE は非常に大きな負荷を持つことが考えられる。このような場合、動的に領域を大小させ、各 PE の負荷のバランスをとる手法である動的領域分割法が有効であると考えられる。

本章では前章で得られた結果を元に適応格子法を並列計算する場合の最適化手法について議論を行う。

5.1 負荷分散

静的な領域分割法、動的な領域分割法について負荷分散について検討を行う。まずはじめに、 100 step までの負荷分散について検討する。 Static-A(図 4.4) を用いた結果では、適応格子法を並列計算するにあたり予想された各ＰＥの負荷量の不釣り合いの問題が顕著に出た。この手法は領域分割の仕方として横方向に単純に等間隔で分割を行い、また静的であるため、簡単に並列計算機に実装できる。しかしながら、適応格子法のように、流れ場の変動に応じて局所的に負荷量が変動する問題に対してこの手法を用いた場合、負荷量のバランスをとることが非常に難しい。今回計算を行った 100 step までの状況は、図 4.2の最上図までの計算に相当する。この時間帯で、ステップ近傍で等高線が集中していることからこの領域で細分化格子が多く生成されていることがわかる。そのため、そのステップ近傍の領域を受け持つＰＥの負荷量が増大する。この様子は図 4.4での PE2, 3

に大きな負荷がかかっている様子から説明できる。100 step 終了後で、PE2 と最も負荷量が少なかった PE8 の間には、約 6 倍程度の開きがある。同様な手法で、分割する方向を 2 次元的にした Static-B の手法では、PE1 がこのステップ領域を担当しており、高い負荷がかかっていることがわかる (図 4.5)。Static-A と違う点としては、最も負荷量が多かった PE1 の実行時間が、Static-A での最大負荷量を持った PE2 と比べ若干ではあるが減少しており、領域の分割の仕方によって負荷分散が行われた。これらの実験を行った計算条件は、横長のパイプ形状であるため、2 次元的に領域分割を行った場合と一次元的に領域分割を行った場合ではそれほど顕著にはパフォーマンスに差はないが、一般的な形状の場合では、領域分割の仕方によってパフォーマンスに重大な影響を及ぼす可能性がある事を示している。Static-C の結果 (図 4.6) はそのことをより明確にさせる。この手法は、ステップのある領域では細分化格子が多く生成されると仮定し、その近辺を受け持つ PE の担当領域を小さくする手法である。相変わらず PE1, 2, 5 で大きな負荷量を持つものの、それらの負荷量の大きさは、Static-A, B と比べ大幅に小さくなっており、領域分割の仕方により負荷の分散具合に大きな影響を及ぼすことがわかる。

次に動的な手法について検討する。まずはじめに、負荷量として定義した LB について言及する。図 4.9が各 PE での積算の LB 量を表しており、図 4.10が Dynamic-A を用いた場合の各 PE の実際の実行時間を示す。細かい部分では若干形が異なるが、PE2 が多くの負荷量を持つ様子や、PE8 が 100 step 近くまで計算が進まないと負荷量の変動があまり起きないことなどの点で一致している。これらの点で、LB を負荷量と定義して再領域分割の際の各 PE の受け持つ領域を決定することがほぼ妥当であると言える。

次に、負荷分散について述べる。Dynamic-A は、ある PE で増大すると考えられる負荷量を、各 PE が担当する領域を伸縮することにより、積極的に分散する手法である。図 4.10で示したグラフではこの積極的な負荷分散の状況が見られる。PE2 で突出した負荷量が現れるが、Static-A と比べると全 PE に負荷量が分散されていることがわかる。特に、Static-A では PE6, 7, 8 にはほとんど負荷量の分散が行われなかったが、動的な手法を用いることでこの状況が改善されている様子がわかる。具体的には、100 step 終了後の PE2 と PE8 との負荷量の差は 3 倍弱となっており、Static-A では 6 倍程度の差があった負荷量の問題が改善されている。Dynamic-B も動的に領域の形状を伸縮させることで負荷分散を行う手法である。この手法の負荷分散の状況を図 4.12で示した。計算開始時は Static-B と同じであるため、同様に PE1, 6 の受け持つ領域付近で大きい負荷が

かかっている様子が見られる。ところが、 Dynamic-A の手法と同様に動的に負荷分散を行うこの手法は、負荷量の分散を積極的に行うために、 Dynamic-A も含めた他の手法と比べ、負荷の平均化という点で非常に良い結果が得られている。 Dynamic-A では、 100 step 終了後に 3 倍程度あった最大負荷量と最小負荷量の差は、 Dynamic-B を用いることで、 2.6 倍にまで抑えられている。 Dynamic-B は Dynamic-A と比較して、分割方向が増えただけの違いであるが、このような結果が得られた。原因として次のようなことが上げられる。 Dynamic-A の負荷量を見た場合、ある程度の負荷量の分散は行われているが、まだ少数の PE に負荷量が集中している状態である。この原因は、領域の大小の自由度が小さいためであると考えられる。具体的には、計算領域として横方向の格子数を 120 としている。これを各 PE に対して分け与えるが、計算バッファや、適応格子法で生成される細分化格子のデータを内挿により求めるために、各 PE の担当する計算領域は最低数個の格子を横方向に持たなければならない。具体的に本研究では、この数を 7 としている。 8 個の PE を用いた場合、計算領域の横方向の格子数が 120 であるので、 1 つの PE は横方向に対して平均 15 の格子を持つことになる。また、 1 つの PE が最低持たなければならない格子数が 7 であるから、この領域分割法では半分程度の負荷の分散しかできないことになる。実際に、各 PE の受け持ち範囲を見ると、 PE2、 PE3、 PE4、 PE5、 PE6 はかなり早い段階から最低のメッシュ数 7 に行きついており、この手法ではそれ以上の負荷分散ができない。それに対して、 Static-B の場合、動的に領域を伸縮させる方向で 4 個の PE によって計算領域が分割される。各 PE の受け持つ平均の横方向の領域の長さは 30 となり、領域伸縮の自由度が Static-A と比較して大幅に増えていることになる。このことから、 Static-B の手法は、局所的に負荷量が増大する問題に対して負荷分散の点で非常に有効である。さらに、現段階では比較的計算領域が小さい問題で、領域伸縮の自由度の問題が非常に厳しいものとなっているが、もっと一般的な大規模計算になった場合、この制限が緩くなることからさらなるパフォーマンスの向上が期待できる。

5.2 付加時間

付加時間には、通信を行う際の通信時間はもちろん、並列計算の際にどうしても行う必要がある同期の際の待ち時間も含まれる。 Static-B の手法では、図 4.7で見られるように、この付加時間が計算全体の中で非常に多くの割合を占めている。特に PE3, 4, 7, 8 は全体の計算時間の 85% 以上をこの付加時間で占める。それとは逆に PE1 については、

ほとんどが実行時間で占められていることがわかる。これは、 PE1 での負荷量が非常に大きいために生じる。他の PE、特に PE3, 4, 7, 8 などは、負荷量がそれほど大きくないにもかかわらず同期の過程で PE1 が処理を終えるまでに待つ形になり、結果としてこの付加時間が増大している。また、 PE1 のこの付加時間は殆ど通信に使用された時間と考えられる。これは、 PE1 の負荷量は計算初期から常に大きく、この計算を通して PE1 が他の PE の処理が終了するのを待つ事が殆どないことから言える。そこから、通信に費やされている時間は全実行時間の 6% 程度であると考えられ、付加時間でそれ以外のものは同期の際の待ち時間で占められていることが予想できる。これらの事から、この手法は通信自体に必要とされるコストは小さいが、負荷分散の状態が良くなく、結果として付加時間が増える手法であると言える。

Static-C に関する結果を図 4.8に示した。 Static-B と比べて、全体的に付加時間の占める割合が小さい。 Static-B の場合には、 PE1 だけが極端に付加時間が小さく、負荷がこの PE だけに集中している様子が見られた。しかしながら、 Static-C では PE1, 2, 5 で比較的この時間が小さく負荷の分散が比較的良好に行われていることを表す。図 4.8をみると、 Static-C ではこの付加時間が PE8 で最大になり、全計算時間の 83% を占める。これは、 Static-B を用いた場合の結果である 87% (PE4) より小さく、この結果からも負荷分散が行われていることがわかる。また、付加時間が最小の PE に関してであるが、 Static-B を用いた場合に PE1 で 6% を占めていたが、 Static-C では PE5 で 17% を占めるようになり、この割合が大きくなっていることが分かる。 Static-B と異なり、ある一つの PE のみが計算全体を通して常に最大の負荷を持っているわけではないので、この 17% という値が全て通信に用いられた時間であるとはいえないが、負荷の均等化が進んでいる中でこの値が大きくなるということは、少なからず通信量が増えているからだと考えられる。実際に、 PE2 などでは、通信先 PE が多いためこのようなことが起こると考えられる。そのためこの手法は、負荷分散は比較的良好に行われているが、通信のコストが大きく、また通信が複雑なことから結果として付加時間が増える手法であると言える。

次に動的な手法である Dynamic-B での付加時間について考察を行う。図 4.13を見ると、上記 2 つの手法と比べて付加時間がだいぶ均等になっていることがわかる。付加時間が最大である PE4 でさえ、その占める割合が 71% に抑えられており、効率の良い計算が行われている。またこの手法では Static-B と同様に PE1 が最も負荷量が多く他の PE に対して待ち時間を与えている。図 4.12をみると計算全体を通して PE1 での負荷が最も

高く、表 4.3で 12% となっている。この量は Static-B と同様、 PE1 が常に最大負荷を持っていることから、殆どが通信時間で占められていると考えられる。 Static-B でも、 PE1 の付加時間はほとんど通信時間で占められていると述べたが、それと比較して通信自体の時間がそれほど増加していないことがわかる。通信量自体は動的な手法を用いることによって増えてはいるが、通信時間が激増しているわけではない。この結果から、 Dynamic-B は通信コストが非常に小さくさらに、負荷分散が良好に行われる手法であると言える。

5.3 速度向上比

並列計算に関して総合的な評価は速度向上比によって与えられる。 8PE を使用した場合の速度向上比を改めてまとめ直す。

	S.U.Ratio(8PE)
Static-A	2.3
Static-B	2.7
Static-C	3.3
Dynamic-A	3.4
Dynamic-B	3.7

これらの結果からわかるように、動的な領域分割法を用いた場合は静的な領域分割法を用いた場合よりも速度向上比の結果が良い。静的な手法で Static-C は速度向上比の点で Dynamic-A に匹敵する結果を出しているが、この手法を一般的な問題に使うのは困難である。今回、 Static-C を計算するにあたり数例のパターンを実際に計算を行ったが、この結果が最も良いものであった。領域形状を変更した場合にパフォーマンスに影響を及ぼし易く、今後一般問題を考えた場合にどのように最適な計算領域に分割するかを事前に考えることが必要であり、それは非常に困難なプロセスである。そのため一般問題には向いていないと考える。

それと比較して、比較的簡略化された手法で高いパフォーマンスが得られているのが動的な領域分割法である、 Dynamic-A、 Dynamic-B である。特に Dynamic-B はこの中でも突出している。負荷分散、付加時間、速度向上比の点で常に他の手法よりも高いパフォーマンスを示す結果となった。

5.4　一般的な問題への拡張

本論文の中で提案した動的な領域分割を用いて並列計算を行う手法の一般問題への拡張について考察を行う。

Dynamic-A は通信コスト、コーディングの容易さの点で非常に良い手法であった。負荷分散、速度向上比に関しても良好な結果を導き出しており十分実用的な範囲で用いる事が可能であると考える。動的に領域を変動させる方向が十分大きい場合、特にこの手法は有効である。表 4.6での結果がそれを示している。並列計算が行われる状況としては、この結果が導き出されたような大規模な計算が一般的であるため、大規模問題に適した手法であると言える。しかし、この手法では、 PE 数を増やすことによって各 PE の担当する領域の伸縮の条件が厳しくなり、これが原因でパフォーマンスが伸びない状況が見られた。計算空間が正方に近い場合などは、大規模計算の際でもこのような問題が起こると考えられる。 Dynamic-B はそのような問題を解消した手法である。はじめにコーディングに関する点であるが、これは Dynamic-A と比較してもそれほど大きな労力ではない。領域の分割自体は 2 次元的であるが、動的に領域を伸縮させるのを一方向と限定しているためにコーディングの複雑さに関して問題はない。またこれに関連して、ある任意の PE が通信を行う相手 PE が領域の変動とは関係なく一定であるため、コーディングの簡素化とともに、通信の複雑さの解消が行われている。また、領域の分割方向を増やしたことにより、通信時間の増加が懸念されるが、本章の付加時間の部分で述べたとおり、大幅な通信量増加はみられず、並列効率に関してこの影響はあまり出ていない。そのため、Dynamic-B は一般的な計算問題に対して良好な結果を出し得る手法であると考える。

これまで述べてきた計算条件は、流れ場の状態としてドラスティックに変動する時間帯であり、局所的に細分化格子が増大するという非常に条件が厳しいものであった。ある程度流れ場が落ち着き、計算領域全体に細分化格子が生成されている場合での結果を図 4.15で示した。この結果は Static-A,Dynamic-A の各 PE での実行時間を表しているが、特に Dynamic-A の手法では負荷のバランスがきれいに取れており、速度向上比でも 8PE 使用時に 6.2 倍という高い結果が得られている。また、 Dynamic-B を用いた場合はさらに高いパフォーマンスである 6.4 倍とう結果が得られた。一方、 Static-A では 3.9 倍程度のパフォーマンスしか得られなかった。このことから、負荷量が計算領域全体で比較的分散されているような状況の元でも、動的な手法である Dynamic-A,Dynamic-B は有効な手段であることがわかる。

次に、計算領域が比較的大きい場合での評価を述べる。計算領域で特に動的に領域を変動させる方向が大きい場合、 Dynamic-A,Dynamic-B の手法は高いパフォーマンスが期待できることを述べたが、表 4.6でこのような場合の速度向上比について言及している。予想されたとおり Dynamic-A,Dynamic-B ともに高いパフォーマンスを示している。これは計算領域サイズの自由度が高くなったため、各 PE の受け持つ計算領域を伸縮させることで負荷分散を行なう本手法に対して有利に働いたためであると言える。

さらにここでは、これらの手法の一般化として、他の計算機でも同様のパフォーマンスが得られるかどうかの検証を行う。使用計算機は TMC の CM5 で、これはファットツリー型のノード構成をもつ分散メモリ型の並列計算機である。 CPU は Sparc IU で、単体性能として、 33MHz, 4.2Mflops を持つ。また、 CM5 には 16Node 毎にベクトルプロセッシングユニットが付いている。

計算を行った手法は Static-A,Dynamic-A である。負荷分散についての結果を図 4.17に示したが、 T3E での同じ条件での結果である図 4.4, 4.10と比較するとほぼ同様な負荷分散状況が見られる。計算時間に着目すると、 T3E の実行時間は CM5 と比較して約 40 倍の速度向上となっている。理論的には、 T3E で使用されている DECchip 21164 は単体性能で 600Mflops であるので、 140 倍程度の速度向上が見られるはずである。しかしこの程度の速度向上しか見られないのは、 CM5 のベクトルユニットでの効果が働いていると考えられる。プログラム自体はベクトル計算を意識していないため、ベクトル化率は小さいと思われるが、手法の基本は差分法であるため、ベクトル化されている部分があると思われる。そのためベクトルユニットを持つ CM5 の結果が良くなっており、 40 倍程度の速度向上になっていると考えられる。速度向上比については、図 4.18で示した。 T3E を用いた場合の結果である図 4.11, 4.3とほぼ同じ結果が得られた。このような点で本手法は一般的な計算対象はもとより、計算機を選ばない手法であると言える。このことは、どのような状況の元でも安定したパフォーマンスが供給できることを示している。

これまでの議論で、適応格子法を並列計算機に実装する手法に関して、動的領域分割を用いた手法、特に Dynamic-B の手法が優れていて、また、一般的な問題へ拡張可能な手法であることを議論してきた。最後に今後の課題について言及する。精度の問題を考慮した場合、現状の初期格子のみの通信では問題がある。細分化格子も含めて通信しようとした場合、通信コストが大きくなると考えられる。特に最領域分割を行う際には、非常に多くのデータを通信する必要が出てくるために、再領域分割を行う最適タイミングについて

今後議論すべき問題である。本論文では簡易的なタイミング最適化ということで、最小負荷量と最大負荷量の比を負荷量のばらつき具合とし再領域分割のタイミングを見つけ出していた。しかし、高い精度を保つために細分化格子を通信する場合、この通信量も考慮に入れたタイミング最適化を行っていく必要があると考える。この考察に関しては、PC クラスタなどの通信速度が遅い計算機環境の元では切実な問題であり、今後検討すべき部分である。

第 6 章

結言

適応格子法を CFD で用いた場合の並列計算アルゴリズムに関する研究を行なった。本研究では、LB という値で各 PE における負荷量の定量化を行ない、この値を各 PE に対して均等に配分するために、各 PE の受け持つ領域を大小させる事で負荷量の分散を行なった。この手法を用いることにより、8PE 使用時で最大 6.8 倍の速度向上比が見られ、また、領域の大きさを等面積に固定して並列計算を行なった静的領域分割法に比べ 1.8 倍程度の速度向上が見られた。また、流れ場の変動が激しい段階でのパフォーマンスの評価を行なった。この評価は、負荷分散が非常にしにくい問題であるが、領域分割を 2 次元的なものとし、さらに動的に行なうことで、8PE 使用時に 3.7 倍の結果を得ることが出来た。また、この結果は領域の大きさを等面積に固定した静的領域分割法に比べ 1.4 倍程度の速度向上である。

近年、適応格子法は非構造格子系で局所的に格子を細分化する手法として試みがなされている。非構造格子は、格子点の位置に関して制限が無いことから比較的容易に格子を張ることが可能である反面、計算効率、精度などの点を含めると構造格子程は困難ではないにしても格子生成が容易ではなくなる。特に三次元の問題に対して格子を張る事は非常に困難になる。そのため正方格子系で適応的な手法を用いることが現在見直されつつあることは、最初にも述べた。しかしながら、精度の高い計算を行なうには、大規模な計算は避けて通れない。特にハードウエア的に大規模化を行ない易い並列計算機を用いた計算は、今後さらに需要が多くなると考えられる。そのため、このような計算機上で正方格子を用いた数値計算が今後一般的に行なわれることは必至であると考える。このような背景から、本論文で示した手法が今後需要の高い手法になると考えられ、工学的に重要な意味を

持つことが期待される。

謝辞

本研究を行なうにあたり, 終始御指導を賜わった松澤 照男 教授に深謝致すとともに御礼申し上げます。

本論文をまとめるにあたり、堀口 進 教授、國藤 進 教授、阿部 亨助教授には有益な御助言と御審査をいただき、深く感謝致します。

文部省宇宙科学研究所の藤井 孝藏 教授には、特別共同利用研究員として受け入れていただき深謝いたします。宇宙科学研究所での一年間は、研究者として勉強することが多く、私にとって非常に有意義でした。このよう場を設けて頂いたことに心から感謝致します。

東京理科大学理工学部の河村　洋　教授には、計算力学講演会、数値流体力学会などで、本研究に関する御助言を度々頂きました。ここに心から感謝致します。

最後に本論文をまとめるに当たって御協力いただいた松澤研究室の諸兄に厚く御礼申し上げます.

参考文献

[1] 藤井孝蔵, “流体力学の数値計算法”, 東京大学出版会

[2] 数値流体力学編集委員会 編,
“数値流体力学シリーズ 6 格子形成法とコンピュータグラフィックス”,
東京大学出版会

[3] Y.Kallinderis and A. Vidwans,
“Generic Parallel Adaptive-Grid Navier-Stokes Algorithm.”
AIAA Journal 32, No.1, January 1994

[4] Ami Harten,
“High Resolution Schemes for Hyperbolic Conservation Laws.”
Journal of Comput. Phys. 49,357-393(1983)

[5] Löhner R,
“An Adaptive Finite Element Schemes for Transient Problems in CFD.”
Compt. Meths. Appl. Mech. Engrg 61, 323-338(1987)

[6] Dongsu Ryu, Jeremaiah, P.Ostriker, Hyesung Kang and Renyue Cen,
“A Cosmological Hydrodynamics Code Based on the Total Variation Diminishing Scheme.”
The Astrophysical Journal 414, 1-19(1993)

[7] 増永ら,
“適応格子を用いた非定常衝撃波の WS 数値計算 第 7 回数値流体力学シンポジウム講演論文集, 455-458(1993)

[8] van Leer, B. "Flux-Vector Splitting for the Euler Equations."
in Lecture Notes in Physics, 70, 509, Springer.

[9] Mönchmeyer, R., Müller, E. 1979,A&A,78,167.

[10] P.Woodward, P.Colella "Thie Numerical Simulation of
Two-Dimensional Fluid with Strong Shocks."
Journal of Compt. Phys. 54, 115-173(1984)

[11] M.J.Berger and P.Collela, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations" Journal of Comp. Phys. 53,484-512(1984)

[12] M.J.Berger and P.Collela, "Local Adaptive Mesh Refinement for Shock Hydrodynamics." Journal of Comp. Phys. 82,64-84(1989)

[13] A.Sohn, R.Biswas, H.D. Simon, "Impact of Load Balancing on Unstructured Adaptive Grid Computations for Distributed-Memory Multiprocessors", Proc. the Eighth IEEE Symposium on Parallel and Distributed Processing, New Orleans, Louisiana, October 1996, pp.26-33.

[14] H.D.Simon, A.Sohn, R.Biswas, "HARP: A Fast Spectral Partitioner" ,the Ninth ACM Symposium on Parallel Algorithms and Architectures, Newport, Rhode Island, June 1997.

[15] C.Walshaw, M.Cross, M.Everett, "Mesh partitioning and load-balancing for distributed memory parallel systems", Proc. Parallel & Distributed Computing for Computational Mechanics, Lochinver, Scotland, 1997.

[16] C.Walshaw, M.Cross, M.Everett, "A Localised Algorithm for Optimising Unstructured Mesh Partitions", Int.J.Supercomputer Appl., 9(4):280-295, 1995

[17] 木下 利博, 井上 督," 解適合非構造格子を用いた流体解析の並列処理", 第 10 回数値流体力学シンポジウム講演論文集,pp.260-261(1996)

[18] 姫野 龍太郎, " 自動車空力の数値シミュレーション", 第 11 回数値流体力学シンポジウム講演論文集,pp.31-32(1997)

[19] 寺本 進, 藤井 孝藏, " 直行格子計算における物体形状表現誤差の影響について", 第11回数値流体力学シンポジウム講演論文集,pp.45-46(1997)

[20] 古山彰一, 松澤照男," 動的負荷分散を用いた適応格子法の並列計算", 情報処理学会論文誌、第38巻第9号、平成9年9月, pp1869-1877

[21] Kozo Fujii, "Unified Zonal Method Based on the Fortified Solution Algorithm", *J. of Comp. Phys.*, Vol.118, pp.92-108 (1995).

[22] 古山彰一, 松澤照男, 藤井孝藏, 國藤進 "Zonal 法を用いた分岐部位流れの解析", 第12回数値流体力学シンポジウム講演論文集（印刷中）

本研究に関する発表論文

査読付き論文

- 古山彰一, 松澤照男,
  「2 次元的な領域分割法を用いた適応格子法の並列計算」, 情報処理学会論文誌 (投稿中)

- 古山彰一松澤照男,
  「動的負荷分散を用いた適応格子法の並列計算」, 情報処理学会論文誌 38 巻 9 号, 平成 9 年 9 月号,pp.1869-1877

査読付き国際会議

- Shouichi Furuyama, Teruo Matsuzawa,
  "Parallel AMR using Dynamic and Automatic Load Balancing", JSME Centennial Grand Congress. Proceedings of International Conference on Fluid Engineering, Vol.2, pp645-650, Jul.,1997

口頭発表等

- 古山彰一, 松澤照男, 藤井孝藏, 國藤進,
  ”Zonal 法を用いた分岐部位流れの解析”, 第 12 回数値流体力学シンポジウム講演論文集（印刷中）

- 古山彰一, 松澤照男,
  「多次元的な自動領域分割法を用いた適応格子法の並列計算」, 日本原子力研究所計算科学技術推進センター平成 9 年度計算科学技術ソフトウエア開発提案中間報告書,1998 年 3 月

- 古山彰一, 松澤照男,
  「2 次元的な動的領域分割法を用いた適応格子法の並列計算」, 第 11 回数値流体力学シンポジウム講演論文集、 pp393-394, 1997 年 12 月

- 古山彰一, 松澤照男,
  「多次元的領域分割法を用いた並列 AMR について」, 第 10 回数値流体力学シンポジウム講演論文集,pp262-263, 1996 年 12 月

- 古山彰一, 松澤照男,
  「自動領域分割法を伴う適応格子法の並列計算」, 第 9 回計算力学講演会講演論文集,pp473-474,1996 年 11 月

- 古山彰一, 松澤照男,
  「動的負荷分散を用いた適応格子法の並列計算」, 日本流体力学年会’96 講演論文集,pp267-268,1996 年 7 月

- 古山彰一, 松澤照男.
  「適合的細分化格子法に適した並列アルゴリズムの開発」, 第 8 回計算力学講演会講演論文集,pp303-304,1995 年 11 月

- 古山彰一, 松澤照男,
  「適応格子法に適した MIMD 型並列アルゴリズムの開発」, 第 9 回数値流体力学シンポジウム講演論文集,pp75-76,1995 年 12 月