ワークステーション・クラスタによる
大規模有限要素解析の並列化に
関する研究

倉前 宏行

要 旨

近年、有限要素解析に要求される問題の規模が大規模なものとなっている。しかし、現在の高速計算環境においては、単一プロセッサの処理能力は半導体素子の限界に近づいてきており、さらなる大規模解析への要求には、多数のプロセッサを同時に用いる並列処理方式の採用が不可避となっている。したがって、大規模解析を実現するための高速計算環境が並列処理方式へと移行するのに伴い、有限要素法の並列処理に適した解析アルゴリズムの開発が急務となっている。

一方で、高性能なUNIXワークステーションが急速に普及し、ネットワークに多数接続された環境が増えてきた。こうした計算機環境を一種の巨大な並列計算機とみなして、分散・並列処理に利用することへの期待が高まっている。しかし、ワークステーション・クラスタによる並列解析においては、ネットワークの通信容量の制約が非常に大きいため、クラスタ環境に適した並列解析法の開発のほか、並列処理を行なう際のさまざまなパラメータの最適化が重要となる。

そこで、本研究においては、ワークステーション・クラスタによる大規模並列有限要素解析を実現するために、

(1) ワークステーション・クラスタとの適合性が良い並列解析法の開発

(2) 並列性能の定量的な評価に基づいた、並列パラメータの最適化方法の確立

(3) メッシュレス的手法を取り入れた高性能領域分割法の開発

(4) 電磁構造連成問題の並列解析法の提案

を行ない、並列有限要素解析システムを開発・構築した。

本論文では、第2章において、有限要素法を並列化する方法として、並列ガウス消去法および並列共役勾配法を提案した。さらに、領域分割法を加えた3つの並列解析法ついて、並列解析時間の評価式を導出して並列性能の定量的な評価を行なった結果、

(1) 最適な並列パラメータの決定法を示した。

(2) 3 つの方法とも最適なパラメータを設定することにより、性能が大きく向上することを示した。

(3) いずれの方法においても、ネットワークの通信容量の制約が厳しいことが明らかとなった。

(4) 通信容量の制約を緩和させる 1 つの方法として、サブネットを利用すると大きな効果があることを示した。

(5) 35 万元規模の大規模解析が 20 台のワークステーションを用いて十分実用的な時間で行なえることを確認した。

第 3 章においては、通信性能が高い Fast Ethernet をワークステーション・クラスタ・システムに使用した場合の並列性能の定量的な評価および比較を、領域分割法、領域型共役勾配法および並列ガウス消去法の 3 種類の並列有限要素解析法に対して行った。その結果、

(1) 高速ネットワークの利用は、ワークステーション・クラスタによる並列有限要素解析において極めて有効である。

(2) 高速ネットワーク環境の性能を最大限に引き出すには、並列パラメータの最適な設定が重要である。

(3) ネットワークの飽和パラメータを定義し、これを用いて Ethernet 環境と比較した結果、高速ネットワークの利用は通信の制約を大きく緩和させることができる。

(4) 高速ネットワークの導入により、35 万元規模の解析を 10 台のワークステーションを用いて短時間に行える。

ことを確認した。

第 4 章においては、これまで全く研究開発が行なわれていない電磁構造連成問題の並列解析法について、領域分割法と領域型共役勾配法を組み合わせた並列解析法を提案した。本並列解析法の並列性能の評価を行なった結果、

(1) 領域分割法と領域型共役勾配法を組み合わせて用いることにより、高い並列性能が得られることを確認した。

(2) 核融合装置真空容器の実機モデルの大規模連成解析に適用し、本並列解析法の実用性を検証した。

第 5 章においては、従来の領域分割法にメッシュレス的手法を取り入れ、並列処理との適合性が良好な新しい並列解析法を提案した。これは、領域間境界上にメッシュレス仮想節点を配置し、領域間を移動最小自乗法により接続する方法である。本解析法の有効性を検証するため、いくつかの 2 次元平板問題を解析した結果、

(1) 仮想節点を境界節点よりも大幅に減らしても、精度良く並列解析することができる。

(2) 領域間で要素分割が不連続となる非適合問題の解析が実用上十分な精度で可能である。

(3) 境界節点数よりも少ない仮想節点を用いて並列解析を行なうことにより、通常の領域分割法に比べて、共役勾配法の収束性および並列性能が向上する。

ことを確認した。

第 6 章では、結論として本研究の成果をまとめた。以上の結果、ワークステーション・クラスタを用いた並列有限要素解析を実現し、大規模問題の解析に非常に有効な手段であることを示した。今後、本研究の成果が非線形問題あるいは非構造問題などといった工学的緒問題へ応用され、一般に広く実用化されて大規模問題の解析に利用されることが期待される。

Abstract

Recently, models of finite element analyses have become extremely large. Since computation performance of a single processor has reached the limitation of semiconductor device, parallel processing technique is required to solve the large-scale problems. On the other hand, the computing environment with many high-performance workstations on a network has recently become available. These workstations can be regarded as a virtual parallel computer, or a workstation cluster to perform the large-scale finite element analysis. However, there are some problems such as limited communication capacity and low parallel efficiency in the workstation cluster system. Since the parallel performance depends strongly on the parameters of parallel computing, evaluation of parallel computation time and optimization of the parallel parameters are needed.

To realize the high-performance parallel finite element analysis using workstation clusters,

(1) a parallel analysis algorithm with coarse granularity suitable for workstation clusters is developed,

(2) an optimization technique of the parallel parameters of finite element analyses is developed based on evaluation of parallel computation time,

(3) a high-performance domain decomposition method using a meshless technique is proposed, and

(4) a parallel analysis method for an electromagnetic-mechanical coupled problem is also proposed.

In chapter 2, parallel algorithms are developed on the cluster system for the parallel Gaussian elimination method and the parallel conjugate gradient method. According

to the performance evaluation of these methods and the domain decomposition method,

(1) methods to determine the optimum parallel parameters are developed,

(2) significant improvement is obtained using the optimum parameters,

(3) data transmission through the network is the largest limitation,

(4) subnetworking is effective to expand the data transmission limit,

(5) three-dimensional large-scale problems with up to 350,000 degrees of freedom are solved using twenty workstations.

In chapter 3, the effect of Fast Ethernet is evaluated and compared on high-performance cluster systems for the domain decomposition method, the parallel conjugate gradient method and the parallel Gaussian elimination method. The results show that

(1) the high-speed network is efficient for high-performance parallel finite element analysis using workstation clusters,

(2) optimization of the parallel parameters is important to achieve the maximum performance of high-speed network,

(3) a relaxation of network limitation in Fast Ethernet environment is observed using network saturation parameter and compared to Ethernet,

(4) a large-scale problem with more than 350,000 degrees of freedom can be solved in about one hour using ten workstation with high-speed network.

In chapter 4, a parallel algorithm is proposed for the electromagnetic-mechanical coupled analysis using the domain decomposition method in combination with the domain partitioned conjugate gradient algorithm. The method to evaluate the parallel performance is discussed for the coupled problem. The results show that

(1) high parallel performance can be obtained by the combined domain decomposition method,

(2) this method is efficient for a large-scale coupled problem of a magnetic fusion device component.

In chapter 5, a high-performance domain decomposition method using a meshless technique for the large-scale parallel finite element analysis is proposed. The subdomains are connected by meshless virtual nodes on the domain interface based on the moving least square method. The results of some two-dimensional plate analyses show that

(1) almost the same accuracy is obtained with much fewer virtual nodes than the interface nodes,

(2) the problem with non-conforming interface among subdomains can be solved practically,

(3) high parallel efficiency and improvement of the convergence of the conjugate gradient method can be obtained by the decrease of the virtual nodes compared to the conventional domain decomposition method.

In chapter 6, conclusions of this study are summarized. Parallel finite element analysis for the large-scale problem is realized using workstation cluster. It will be expected that results of this study will be widely utilized by applied to general engineering field such as nonlinear and non-structural problems.

目 次

第 1 章

序 論

1.1 大規模解析の並列化の必要性

有限要素法は、コンピュータの飛躍的な進歩とともに発展し、現在では構造解析のみならず、流体解析、電磁場解析などさまざまな物理現象の解析に適用されている。実際の物理現象のシミュレーションにおいては、流体構造、電磁構造、熱流体などといった、いくつかの系が連成する複雑系・複合系の超多自由度の解析が必要となり、さらに、さまざまな非線形問題を取り扱わなければならないため、解析の規模は超巨大化する。

設計現場においても、設計の高度化によって、旧来の公式や試作・実験に基づいた設計 (いわゆる Design by Formula 方式) から、有限要素法をはじめとする数値シミュレーションに基づいた解析による設計 (Design by Analysis) へと移行している。こうした CAE (Computer-Aided Engineering: 計算機援用工学) の普及にともない、より大規模な解析モデルを用いた数値シミュレーションが要求されている。すなわち、より複雑で高精度な解析の要求から、これまで解析モデルを簡便な 2 次元モデルやシェル要素に置き換えて解析していたものを、実物をそのまま 3 次元のソリッドモデルを用いて解析を行なうといったことや、より詳細な要素分割モデルを用いた高精度な解析が必要となっている。また、解析対象物をそれぞれの部品に分けて部分的な解析を行なっていたものを、巨大で複雑なコンポーネントを丸ごと解析するといった要求もある。こうした大規模な問題をいかに効率的に解くかということが、現在、大きな課題となっている。

これまでは、パイプライン方式を採用したベクトル型スーパーコンピュータをはじめとする高性能・高速計算機の登場により、大規模問題の解析を可能なものにしてきた。また、こうした計算機環境の飛躍的な進歩は、有限要素法をはじめとする計算力学・計

算工学を発展させる原動力の役割を果たしてきた。しかし、現在、単一プロセッサの処理能力は半導体素子の限界に近付いてきており、さらなる大規模解析への要求を満たすのは困難となっている。したがって、今後、これ以上の演算性能の向上には、多数のプロセッサを同時に用いる並列処理方式の採用が不可避となっている。

一方で、かつての汎用大型計算機なみの処理能力を有する UNIX ワークステーションが急速に普及していることにより、計算機環境が大きく変化している。すなわち、1 人 1 台の個人計算機環境において、3 次元ソリッドモデラーなどを用いた解析モデルの作成から、解析計算、コンピュータ・グラフィックス (CG) を利用した解析結果の可視化表示までを手軽に行なえるようになった。さらに、こうしたワークステーションはローカル・エリア・ネットワーク (LAN) にインターネット接続され、1 人で多数のワークステーションを同時に利用することも可能である。

こうした 1 本のネットワークに接続された多数のワークステーションは、一種の巨大な並列計算機とみなすことができる。したがって、こうしたネットワーク環境を利用することにより、1 台のワークステーションでは実行不可能な大規模解析を手軽に並列解析することが可能となる。このようなネットワーク環境の利用方法は、ネットワーク・コンピューティングあるいはワークステーション・クラスタと呼ばれ、新しい計算機利用環境として注目と期待を集めている。

しかし、従来の有限要素法は、すべて単一プロセッサの計算機を用いた逐次処理を前提として解析アルゴリズムが開発されてきた。しかし、大規模解析を実現するための計算機環境が並列処理方式へと移行するのにともない、有限要素法も並列処理に適したアルゴリズムの開発を行なわなければならない。すなわち、逐次処理用の計算アルゴリズムを並列処理環境にそのまま適用しても、効率よく並列処理が行なえないため、高速化をはかることができない。したがって、独立に演算可能な処理部分を拡大して複数のプロセッサ上で並列に演算できるようにするとともに、プロセッサ間のデータ交換の頻度およびデータ転送量が少なくなるような解析アルゴリズムの開発が必要となる。

そこで、本研究においては、ワークステーション・クラスタを用いて大規模問題の解析を高速に実現するための有限要素法の並列解析アルゴリズムを開発することを目的とする。本章においては、まず、これまでのさまざまな研究例をもとに、有限要素法を並列化する際の問題点および考慮しなければならない点を整理し、ワークステーション・クラスタに適した並列解析法を開発するにあたっての留意点を摘出する。

1.2 並列計算環境

1.2.1 ベクトル型スーパーコンピュータ

1980 年代、配列演算を高速化するためのパイプライン方式を採用したベクトル型スーパーコンピュータが開発され、計算力学の分野でも有効に利用されてきた。ベクトル処理は、演算処理が連続する配列データに対して個々の演算処理を細かいステージに分解してパイプラインに流し込み、各ステージを独立に動作させて演算処理を次々と行なうことにより、演算の高速化をはかる。したがって、全処理のうちベクトル処理が可能となる割合、すなわちベクトル化率が高速化の決め手となり、ベクトル長を長く取ることやスカラー演算をベクトル化が可能な計算に変更することが必要となる。

1.2.2 並列計算機

並列処理は、1 つの処理を多数の独立に演算可能な処理に分割し、それらを複数のプロセッサ上で同時に実行することにより高速化する。このため、処理を分割してプロセッサに割り当てる際、ほかのプロセッサとのデータの交換や同期を行なわずに実行できる計算量の大きさ、すなわち並列処理の粒度 (granularity) が大きいほど並列計算が効率的に行なわれ高速化を図ることができる。こうした並列処理の性能を表す指標としては、通常、スピード・アップ (speed-up) や並列効率 (parallel efficiency) といったものが用いられる。スピード・アップは、並列処理によって得られる計算時間の加速率に相当し、1 台のプロセッサを用いた場合の計算時間と n 台のプロセッサによる並列計算時間との比により表される。また、並列効率は、1 台のプロセッサによる計算時間と n 台のプロセッサによる並列計算時間の n 倍との比により表され、n 台のプロセッサを用いたときに計算時間が $1/n$、すなわち、スピード・アップが n となるとき、並列効率は 100 %となる。

並列計算機は、一般的に、数プロセッサを備えたものをマルチプロセッサ、プロセッサ数が 1000 を越えるものを超並列計算機と呼ばれている。さらに、プロセッサのメモリへのアクセス方式や演算命令の流れなどの違いによって、いくつかの形態に分類される。

メモリへのアクセス方式としては、全てのプロセッサが同一のアドレス空間を共有し、すべてのプロセッサが 1 つのメモリをアクセスする共有メモリ方式と、それぞれのプロセッサが独立した専用のメモリを持つ分散メモリ方式に分類される。共有メモリ方式は、プロセッサからのメモリアクセスが容易であるため、プログラミングも比較的容易であるが、プロセッサ数が多くなるにしたがいメモリへのデータアクセスが頻繁になること

から、多数のプロセッサを有する並列計算機においては分散メモリ方式が採用される。分散メモリ方式の場合には、他のプロセッサが所有するメモリ上のデータが必要になると、プロセッサ間を接続するネットワークを通じてデータの送受信を行なわなければならない。こうしたデータ交換はメッセージ・パッシングと呼ばれ、プログラミングの際には各プロセッサにおけるデータ (メッセージ) の受渡しを明示的に記述しなければならない。

また、各プロセッサの動作命令の流れの違いにより、SIMD (Single Instruction stream, Multiple Data stream: 単一命令, 多重データ) 方式と、MIMD (Multiple Instruction stream, Multiple Data stream: 多重命令, 多重データ) 方式に分けられる。SIMD 方式は、1 台のコントロール・ユニットからプロードキャストされた命令により、全てのプロセッサがそれぞれのデータに対して同じ処理を行なう。このため、各プロセッサの動作は常に同じ状態であり、適用する問題によっては、比較的容易に並列化することができる。これに対して MIMD 方式は、各プロセッサ・エレメントにそれぞれコントロール・ユニットが内蔵され、それぞれのプロセッサは完全に独立して動作する。このため、汎用性および柔軟性が高い並列処理を実現することができるが、プログラミングは各プロセッサの動作を全て記述しなければならないため、複雑となる。

マルチプロセッサなど小規模な並列計算機においては、共有メモリ型 SIMD 方式が一般的であるのに対し、プロセッサ数が多い超並列計算機では量産されている RISC プロセッサを用いた分散メモリ型 MIMD 方式のものが一般的である。

ベクトル並列計算機と呼ばれる、パイプラインをベクトルレジスタを備えたベクトル・ユニットを複数並べて並列に動作させる並列計算機が登場している。これは、プロセッサのベクトル化および並列アーキテクチャの採用という両面から高速化をはかることにより、適用範囲の広い並列処理環境として注目されている。

1.2.3 ワークステーション・クラスタ

ワークステーション・クラスタは、それぞれのワークステーションをプロセッサとローカル・メモリとして、これがネットワークによって接続されているとみなすことができるため、分散メモリ型 MIMD 方式の並列処理環境と等価である。最近の高性能ワークステーションには、スーパースカラーを採用した 64 bit RISC プロセッサが搭載されており、これらを利用したクラスタ・システムは、1 プロセッサあたりの演算処理能力が大きい。さらに、ネットワーク環境の普及により、ワークステーション・クラスタ・システムは身近に利用可能であるとともに、大規模な並列システムを容易に構築すること

ができ、既存資源の有効利用という点からも非常に有効な方法である。

また、PVM (Parallel Virtual Machine)[1.1]、p4 (Portable Programs for Parallel Processors)[1.2]、TCGMSG (Theoretical Chemistry Group Message Passing System)[1.3]といったワークステーション・クラスタを用いた並列アプリケーション・プログラムの構築を支援するシステムが開発されている。これらはフリーソフトウェアとして公開されていることから、ネットワーク環境における分散・並列システムを比較的容易に構築できるようになっている。

しかし、現在広く普及しているネットワークは、Ethernet をはじめとする 10 Mbps クラスの比較的低速なネットワークであるため、ワークステーション・クラスタを用いた並列処理においては、ネットワークの通信容量の制約が大きな問題となる。ネットワークに関しては、最近、100 Mbps クラスの Fast Ethernet や ATM (Asynchronous Transfer Mode: 非同期モード) スイッチを用いた高速ネットワークが普及しはじめており、ますますワークステーション・クラスタを用いた並列処理への期待が高まってきている。

1.3 有限要素法の並列化

有限要素解析の一般的な手順は

1. 解析データの入力
2. 要素マトリックスの作成と全体系への組込み
3. 連立一次方程式の解法
4. 要素ごとの物理量の算出

という流れとなる。ここで、2. および 4. の要素ごとに関する計算は、それぞれの要素ごとに独立に行なうため、容易に並列化が可能である。しかし、これらの計算量は全解析計算のうちのごくわずかであり、大規模解析を行う際に最も計算時間を要するのは、連立一次方程式解法である。また、この係数マトリックスを記憶するための膨大な記憶容量が必要となる。したがって、大規模解析においては、いかに大次元の連立一次方程式を高速に解くかが問題となる。

連立一次方程式の解法には、大別して、ガウスの消去法に代表される直接解法と共役勾配法に代表される共役勾配法がある。直接解法は、有限回の演算により安定した解が得られるものの、必要とされる記憶容量が膨大であるのに対し、反復解法は、問題の性質によって収束性が変化するものの、記憶容量の面で大規模解析に適している。

1.3.1 直接解法の並列化

有限要素解析を高速化するため、これまで並列化に関する研究が盛んに行なわれてきている。直接解法の高速化を目指した方法としては、ベクトル計算機において内積形式のコレスキー分解[1.4]を用いると、最内側ループがベクトル化され高速化される。しかし、有限要素法の係数マトリックスのようにバンド状でスパースなマトリックスでは、ベクトル長が長くとれないことからリストベクトルを用いてマトリックスの要素を一次元配列に並べる手法がとられる。

SIMD 型の並列計算機を用いた有限要素法の並列化としては、データ並列処理のために拡張されたプログラミング言語を用いて解析コードを記述することにより、要素レベルの演算やさまざまな繰り返しのループを容易に並列化することができる[1.5, 1.6]。

MIMD 型の並列計算機を用いた直接解法の並列化に関する研究としては、Melosh ら[1.7]や Zois[1.8]による全体マトリックスをブロック分割してプロセッサに割り当て、ガウスの消去法の演算を並列化する方法がある。しかし、これらの方法はマトリックスの分散および演算に対してマトリックスの対称性やスパース性が考慮されていないため、演算量および記憶容量が増大している。一方、Allik[1.9]らは、バンドマトリックスに対するガウスの消去法を並列化している。しかし、この方法は、演算が全てマトリックスの行単位に分割されているため、並列処理の粒度が非常に小さい。また、Farhat らは、スカイライン法を並列化し Parallel Active Column Equation Solver[1.10]を開発した。これは、内積形式のコレスキー分解において、分解行とのベクトル内積演算を並列化するもので、さまざまな並列計算機上で良好な並列性能が得られている[1.11]。

ベクトル並列計算機を用いた直接解法の並列化として、三好ら[1.12]は、Farhat らの Parallel Active Column Equation Solver において、さらに 3 行 3 列のマトリックス成分を同時に分解するパラレルスカイライン法を開発し、汎用有限要素解析コード SUPER SOLVE において実用化されている。また、コレスキー分解のベクトル並列処理としては、演算を外積形式に変更することにより、最内側ループをベクトル化し、更に外側ループを並列化する方法が開発されており[1.13–1.15]、さまざまな大規模問題の解析において、その有効性が示されている。[1.16–1.18]。しかし、これらの研究例は、ベクトル並列計算機というある特定のハードウェアを対象としたものであり、超並列計算機やワークステーション・クラスタといった並列計算環境においてその成果を利用することができない。

こうした直接解法の並列化は、記憶容量の制約が厳しいため解析可能な規模に限界がある。また共有メモリ型あるいは密結合のプロセッサによる並列環境においては、メモリアクセスが高速に行なえるため高い並列性能が得られるが、ワークステーション・ク

ラスタのような疎結合・分散メモリ型の並列環境においては、並列処理の粒度が低いためメッセージパッシングによるオーバーヘッドの増大により、高い性能を得ることは困難である。

1.3.2 反復解法の並列化

反復解法である共役勾配法の高速化に関する研究としては、係数マトリックスに対して前処理を行なうことにより解の収束性を向上させる PCG (Preconditioned Conjugate Gradient) 法[1.19]に関する研究がある。三好らは、係数マトリックスを不完全コレスキー分解して前処理に用いる ICCG (Imcomplete Cholesky Conjugate Gradient) 法[1.20]に関して、コレスキー分解の演算をベクトル化することにより高速化した[1.21]。また、直接解法である 3 行 3 列同時分解のパラレルスカイライン法と比較して、問題の種類によってはベクトル化された ICCG 法の方が高速であることを示した。ICCG 法にはベクトル化が不可能な後退代入演算が含まれるため、Hayami ら[1.22]は、共役勾配法の前処理法として、係数マトリックスの対角項を用いたスケーリングを行なう SCG 法 (Scaled Conjugate Gradient) によりベクトル化率を向上させ、ICCG 法よりも高速化が実現されている。

一方、共役勾配法の並列化方法としては、Hughes ら[1.23]による EBE (Element-By-Element) 法がある。これは、要素マトリックスを全体系に組み込まずに共役勾配法の演算を要素レベルで行ない、その演算結果を全体化して解く方法である。したがって、非常に少ない記憶容量で解析することができ、さまざまな並列環境に適用されている[1.24, 1.25]ものの、並列計算が全て要素単位で行なわれ粒度が非常に小さいことから、ワークステーション・クラスタのようなプロセッサが疎結合された並列処理環境においては、効率の良い並列化は困難である。

Element-By-Element 法は要素レベルにおいて並列化されるため粒度は極めて細かいが、これを大きくする方法として、領域型共役勾配法 (Superelement-By-Superelement)[1.26–1.31]がある。これは、解析領域を複数の領域に分けてプロセッサに割り当て、それぞれの領域ごとに係数マトリックスを作成した後、共役勾配法の演算を領域ごとに行なうことで並列化するものである。さらに、Papadrakakis ら[1.32]は、係数マトリックスの逆マトリックスをノイマン多項式を用いて近似し、これを用いて領域型共役勾配法に前処理を行ない、収束性の向上をはかっている。

1.3.3 領域分割法による並列化

並列計算を行なうための演算およびデータの分割方法として、直接解法の並列化のように、全体化されたマトリックスを行あるいは列方向のブロックに分割することにより並列性を導く方法に対して、あらかじめ解析領域を分割しておき、領域ごとに解析を行なうことにより並列化する方法がある。こうした方法は領域分割法 (Domain Decomposition Method) と呼ばれるが、前述した領域型共役勾配法も広義の領域分割法といえる。しかし、領域型共役勾配法の場合には、解析は全体領域として行なわれ、領域ごとには共役勾配法のマトリックス演算を行なっているのみである。よって、ここで扱う領域分割法としては、領域ごとに独立に解析を行なって解を求める方法に限定し、領域型共役勾配法などとは区別することとする。

領域を分割して有限要素解析を行なう方法としては、従来からサブストラクチャー法[1.33]と呼ばれる方法があった。これを利用して並列化する方法として、Farhat[1.34]らや Noor ら[1.35]による並列サブストラクチャー法がある。この方法は、各領域の内部自由度を並列に消去して、境界自由度の方程式を作成して解く。境界自由度の方程式は、各領域の境界自由度に関するマトリックスを 1ヶ所のプロセッサに集めて全体化し、これを逐次処理あるいは Parallel Active Column Equation Solver などを用いた直接解法によって解く。したがって、この処理部分において並列性能が低下することになる。

Glowinski ら[1.36]は、領域分割を行なった際の領域間境界における解の連続条件をラグランジュの未定乗数法を用いて付帯条件として与えて満足させ、共役勾配法などの反復解法を適用する方法を提案した。しかし、定式化の過程においてさまざまな制約があるため、一般性・汎用性という点に問題があった。Yagawa ら[1.37]は、Glowinski らの方法をベースにして、並列サブストラクチャー法における境界自由度の解法に共役勾配法を適用することにより、境界マトリックスを全体化することなく並列解析する方法を開発した。さらに、この領域分割法をトランスピュータ、ワークステーション・クラスタ、超並列計算機などさまざまな並列計算環境に適用しているほか[1.37–1.39]、動的問題、非線形弾塑性問題に応用している[1.40, 1.41]。一方で、領域分割法に関する数学的な裏付けも行なわれている[1.42]。これにより、領域間境界における境界条件としての取り扱い方として、いくつかの方法が示されるとともに、構造問題のみならずさまざまな問題へ適用されている[1.43]。

領域分割法に分類される他の並列化方法としては、Farhat らによる FETI 法 (Finite Element Tearing and Interconnecting)[1.11, 1.44]およびその変形版である RFETI 法 (Regularized FETI)[1.45]、モルタル有限要素法[1.46, 1.47]などがある。これらは、いずれも領域

間境界における解の連続性をラグランジュ未定乗数法により求めるものである。

しかし、領域分割法は、領域の分割方法によって並列性能が大きく変化することから、性能を最大限に引き出すためには、これらの最適化が必要とされる。

1.3.4 ワークステーション・クラスタによる有限要素法の並列化

一般に、これまでの研究から、効率が良くかつ実用的な並列解析法を開発するためには、以下の点を考慮しなければならない。

1. 並列計算の高粒度化

   逐次計算しなければならない演算量を限りなく減らすとともに、他のプロセッサとのデータ交換をできる限り行なわないようにする。すなわち、並列処理が可能な演算量を大きくし、通信量を減少させて並列処理の粒度を大きくすることにより、高い並列効率を得ることができる。

2. 計算負荷の均等化

   各プロセッサに割り当てる計算量に差があると、割り当てられた計算量が少ないプロセッサに待ち時間が生じることとなり、並列性能が低下する。したがって、プロセッサへ割り当てる計算量を均等化しなければならない。

3. 容易なプログラミング

   一般に並列アルゴリズムは複雑になり、プログラミングやデバッグが困難となる。したがって、できるだけ簡便な解析アルゴリズムを用いて汎用性や移植性のよい並列解析法を開発しなければならない。

ワークステーション・クラスタを用いた並列解析においては、1.2 節においても述べたように、ネットワークの通信容量の制約が非常に厳しいため、通信のオーバーヘッドが大変大きい。このため、クラスタ環境に適した並列解析法としては、特に、並列計算の粒度を大きくし、プロセッサ間で行なわれる通信の頻度を少なくするとともに、通信量そのものを小さくしなければならない。

こうした点を考慮した上で、本研究においては、ワークステーション・クラスタとの適合性が良い有限要素法の並列解析法を提案する。

1.4 本研究の目的

ワークステーション・クラスタは、プロセッサが疎結合された分散メモリ方式の並列環境であるが、現在広く普及している 10 Mbps クラスの Ethernet の通信性能は、並列計算機におけるプロセッサ間を接続するネットワーク性能に比べ、極めて低いという問題がある。したがって、前節で述べたさまざまな並列有限要素解析法のうち、直接解法の並列化や Element-by-Element 法においては、並列計算がマトリックスの行単位あるいは要素単位で行なわれるため粒度が小さく、そのままクラスタ環境に適用しても効率の良い並列処理を行なうことができない。一方、領域分割法は、並列計算が領域単位に行なわれるため比較的粒度が大きい並列解析法であるが、領域への分割方法によってプロセッサへ割り当てられる演算量のほか通信量が大きく変化するため、並列性能が領域分割数に大きく依存する。したがって、並列性能を最大限に発揮させるためには、並列処理を行なう際にこうした各種のパラメータを最適な値に設定しなければならない。

そこで、本研究においては、ワークステーション・クラスタとの適合性が良い有限要素法の並列解析法を提案するとともに、開発した並列解析法の並列性能を定量的に評価し、これに基づいて最大の並列性能を得るための並列パラメータの最適化方法を明らかにする。並列性能の定量的な評価を行なうため、並列解析時間を予測する評価式を、問題の大きさや並列台数といった基本量を用いて導出し、これを用いた性能の最適化方法を提案する。

解析問題の対象としては、3 次元弾性問題のほか、これまで並列解析法の研究開発が全く行なわれていない電磁構造連成問題について、並列解析法を提案するとともに、実際に大規模問題の解析を行ない実用性を検証する。さらに、こうした研究成果をもとに、より高性能でさまざまな制約を取り払うことが可能なメッシュレス的手法を取り入れた新しい領域分割法を提案し、その有効性を検証する。

本論文では、まず第 2 章において、ワークステーション・クラスタを用いて並列処理を実現するための技術について述べたのち、有限要素法を並列化する方法としてマトリックスのブロック分割に基づく並列ガウス消去法および並列共役勾配法を提案する。並列ガウス消去法は、前進消去の演算部分をバンド・マトリックスの複数行をまとめたブロック単位で並列化する方法を開発ことにより、並列計算の粒度を大きくとることが可能となる。また、並列共役勾配法は、マトリックス・ベクトル演算部分を並列化するが、プロセッサへの演算の分散方法の違いにより、ブロック型および領域型の 2 つの方法を開発する。さらに、領域分割法を加えた 3 つの方法について、それぞれ並列性能を定量的に評価するための並列解析時間を表す評価式を導出し、並列パラメータの最適化を行な

う方法を提案する。これにより、各方法の性能の限界を明らかにするとともに、提案した並列解析法および並列パラメータの最適化方法によって、35 万自由度を越える大規模な 3 次元問題を並列解析し、本手法の有効性を確認する。

第 3 章においては、第 2 章で述べた 3 つの並列解析法について、演算性能および通信性能が異なるさまざまなクラスタ・システムにおいて並列性能の評価を行なう。特に、従来の 10 倍の通信性能を有する高速ネットワークを用いた場合の並列性能について検討する。この結果に基づいて、高速ネットワーク環境に適した並列パラメータの設定方法を明らかにする。

第 4 章においては、電磁場と構造体を同時に解析する電磁構造連成解析を並列化する方法を提案する。この連成解析には領域分割法をそのまま適用することが困難であるため、領域型共役勾配法を併用した並列解析法を開発するとともに、例題の並列解析および性能の定量的な評価を行ない有効性を検証する。さらに、核融合装置真空容器に生じる電磁構造連成現象を実機モデルを用いて並列解析し、本手法の実用性を検証する。

第 5 章においては、以上の研究成果をもとに領域分割法におけるさまざまな制約を取り払うことを目的として、メッシュレス法に用いられる手法を導入した新しい領域分割法を提案する。この新しい領域分割法の開発によって、並列性能および共役勾配法の解の収束性が向上することや領域分割に関連するさまざまな制約が取り払われることについてまとめ、本手法の有効性を示す。さらに、この解析法を用いて例題の解析を行ない、解析方法の妥当性を検証するとともに、並列性能の比較検討を行ない、本手法の実用性を明らかにする。

参考文献

(1.1) A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, "PVM3 User's Guide and Reference Manual", Reserch Report ORNL/TM-12187, Oark Ridge National Laboratory, (1993).

(1.2) R. Butler and E. Lusk, "User's Guide to the p4 Programing System", Technical Report ANL-92/17, Argonne National Laboratory, (1992).

(1.3) R. J. Harrison, "TCGMSG Send/receive subroutines —version 4.03 user's manual", Battelle Pacific Northwest Laboratory, (1993).

(1.4) 村田, 小国, 唐木, "スーパーコンピュータ 科学技術計算への適用", 丸善, (1985).

(1.5) R. R. Namburu, "Enthalpy-Based Explicit Finite Element Data Parallel Formulations for Thermal Analysis of Structures", *Comput. Sys. Eng.*, **4**, (1993), *pp.* 445–452.

(1.6) R. C. Shieh, "Massively Parallel Computational Methods for Finite Element Analysis of Transient Structural Responses", *Comput. Sys. Eng.*, **4**, (1993), *pp.* 421–433.

(1.7) R. J. Melosh, S. Utku and M. Salama, "Direct Finite Element Equation Solving Algorithms", *Compt. Struct.*, **20**, (1985), *pp.* 99–105.

(1.8) D. Zois, "Parallel Processing Techniques for FE Analysis: System Solution", *Compt. Struct.*, **28**, (1988), *pp.* 261–274.

(1.9) H. Allik, S. Moore, E. O'Neil and E. Tenenbaum, "Finite Element Analysis on the BBN Butterfly Multiprocessor", *Compt. Struct.*, **27**, (1987), *pp.* 13–21.

(1.10) C. Farhat and E. Wilson, "A Parallel Active Column Equation Solver", *Compt. Struct.*, **28**, (1988), *pp.* 289–304.

(1.11) C. Farhat and F. X. Roux, "Implicit Parallel Processing in Structural Mechanics", *Computational Mechanics Advances*, **2**, (1994), *pp.* 1–124.

(1.12) 三好, 吉田, "スーパーコンピュータによる三次元表面き裂の有限要素解析", 日本機械学会論文集 A 編, **53**-486, (1987), *pp.* 255–260.

(1.13) C. Farhat, “Redesigning the Skyline Solver for Parallel/Vector Supercomputers”, *International Journal of High Speed Computing*, **2**-3, (1990), *pp.* 223–238.

(1.14) J. Qin and D. T. Nguyen, “A New Parallel-Vector Finite Element Analysis Software on Distributed-Memory Computers”, *AIAA-93-1307-CP*, **93**-1307, (1993), *pp.* 98–102.

(1.15) M. A. Baddourah, O. O. Storaasli and S. W. Bostic, “Linear Static Structural and Vibration Analysis on High-Performance Computers”, *Comput. Sys. Eng.*, **4**, (1993), *pp.* 363–371.

(1.16) A. C. Damhang, K. M. Mathisen and K. M. Okstad, “The Use of Sparse Matrix Methods in Finite-Element Codes for Structural Mechanics Application”, *Comput. Sys. Eng.*, **4**, (1993), *pp.* 355–362.

(1.17) V. Gupta, J. Newell, O. Storaasli, M. Baddourah and S. Bostic, “Space Station Static and Dynamic Analyses using Parallel Methods”, *Comput. Sys. Eng.*, **4**, (1993), *pp.* 387–398.

(1.18) E. Poole, J. Bauer, T. Strattor and T. Weidner, “Large-Scale Nonlinear Structural Analysis Simulation on CRAY Parallel/Vector Supercomputers”, *Comput. Sys. Eng.*, **4**, (1993), *pp.* 405–414.

(1.19) 例えば、島崎, スーパーコンピュータとプログラミング, 共立出版, (1989).

(1.20) J. A. Meijerink and H. A. Van der Vorst, “An Iterative Solution Method for Liniear Systems of which the Cofficient Matrix is a Symmetric M-matrix”, *Math. Comp.*, **37**, (1977), *pp.* 148–162.

(1.21) 三好, 高野, 吉田, “スーパーコンピュータによる三次元有限要素解析”, 構造工学における数値解析法シンポジウム 論文集, **10**, (1986), *pp.* 287–292.

(1.22) K. Hayami and N. Harada, “The Scaled Conjugate Gradient Method and Vector Processors” *Proc. of the First Internaional Conference on Supercomputing System, IEEE Computer Society*, (1985), *pp.* 213–221.

(1.23) T. J. R. Hughes, I. Levit and J. Winget, “An Element-by-Element Solution Algorithm for Problems of Structural and Solid Mechanics” *Comput. Meths. Appl. Mech. Engng.*, **36**, (1983), *pp.* 241–254.

(1.24) H. Adeli and S. Kumar, "Distributed Finite-Element Analysis on a Network of Workstations—Algorithms", *J. Struct. Eng.*, **121**, (1995), *pp.* 1448–1455.

(1.25) S. Kumar and H. Adeli, "Distributed Finite Element Analysis on a Network of Workstations—Implementation and Applications", *J. Struct. Eng.*, **121**, (1995), *pp.* 1456–1462.

(1.26) W. T. Carter, T.-L. Sham and K. H. Law, "A Parallel Finite Element Method and its Prototype Implementation on a Hypercube", *Compt. Struct.*, **31**, (1989), *pp.* 921–934.

(1.27) M. Papadrakakis, "Solving Large-Scale Linear Problems in Solid and Structural Mechanics", In: M. Papadrakakis (ed): Solving large-scale problems in mechanics, John Wiley & Sons Ltd, (1993), *pp.* 1–37.

(1.28) J. F. Abel, B. H. Aubert and S. H. Hsieh, "Towards Parallel Solutions of Nonlinear Structural Dynamics in a Networked Workstation Environment", *Proc. of International Conference on Computational Engineering Science*, (1991), *pp.* 1–6.

(1.29) S.-H. Hsieh and J. F. Abel, "Use of Networked Workstations for Parallel Nonlinear Structual Dynamic Simulations of Rotating Bladed-Disk Assemblies", *Compt. Sys. Eng.*, **4**, (1993), *pp.* 521–530.

(1.30) K. Iwano, V. Čingoski, K. Kaneda and H. Yamashita, "A Parallel Processing Method in Finite Element Analysis using Domain Division", *IEEE Trans. Mag.*, **30**, (1994), . *pp.* 3598–3601.

(1.31) A. Chatterjee, J. L. Volakis, "Parallel Computation of 3D Electromagnetic Scattering using Finite Elements and Conformal ABCs", *IEEE Trans. Mag.*, **30**, (1994), . *pp.* 3608–3609

(1.32) M. Papadrakakis, N. Bitoulas and K. Hatjkonstantinou, "An Effective Superelement-by-Superelement Solution Method for Large Finite Element Computations", *Compt. Systems. Engng.*, **2**, (1991), *pp.* 535–540.

(1.33) 例えば、P. Tong and J. N. Rossettos, (矢川 訳) "エンジニアのための有限要素法", 共立出版, (1983).

(1.34) C. Farhat and E. Wilson, “A New Finite Element Concurrent Computer Program Architecture”, *Int. J. Num. Meth. Engng.*, **24**, (1987), *pp.* 1771–1792

(1.35) A. K. Noor and J. M. Peters, “A Partitioning Strategy for Efficient Nonliner Finite Element Dynamic Analysis on Multiprocessor Computers”, *Compt. Struct.*, **31**, (1989), *pp.* 795–810.

(1.36) R. Glowinski, Q. V. Dinh and J. Periaux, “Domain Decomposition Methods for Nonlinear Problems in Fluid Dynamics”, *Comput. Meths. App. Mech. Engng.*, **40**, (1983), *pp.* 27–109.

(1.37) G. Yagawa, N. Soneda and S. Yoshimura, “A Large Scale Finite Element Analysis using Domain Decomposition Method on a Parallel Computer”, *Comput. and Struct.*, **38**, (1991), *pp.* 615–625.

(1.38) 吉岡, 矢川, 吉村, 曽根田, “大規模・超高速計算力学のためのネットワーク・コンピューティング手法の開発”, 日本機械学会論文集 A 編, **57**-541, (1991), *pp.* 1964–1972.

(1.39) G. Yagawa and R. Shioya, “Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition”, *Comput. Sys. Eng.*, **4**, (1993), *pp.* 495–504.

(1.40) 大石, 山田, 吉村, 矢川, “領域分割法の動的有限要素解析への適用”, 日本機械学会論文集 A 編, **58**-552, (1992), *pp.* 1445–1452.

(1.41) 矢川, 内山, “EWS ネットワークを用いた弾塑性有限要素法”, 日本機械学会論文集 A 編, **60**-580, (1994), *pp.* 2737–2742.

(1.42) P. L. Tallec, “Domain Decomposition Methods in Computational Mechanics”, *Computational Mechanics Advances*, **1**, (1994), *pp.* 121–220.

(1.43) A. Suzuki, “Implementation of Domain Decomposition Methods on Parallel Computer ADENART”, *Parallel Computational Fluid Dynamics: Newalgorithms and Apllications*, N. Satofuka, J. Periaux and A. Ecer (eds.), Evlsevier, (1995).

(1.44) C. Farhat and F.-X. Roux, “A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm”, *Int. J. Num. Meth. Engng.*, **32**, (1991), *pp.* 1205–1227.

(1.45) C. Farhat, “Fast Structural Design and Analysis via Hybrid Domain Decomposition on Massively Parallel Processors” *Compt. Sys. Eng.*, **4**, (1993), *pp.* 453–472.

(1.46) C. Bernardi, Y. Maday and A. Patera, “A New Nonconforming Approach to Domain Decomposition: The Mortar Element Method”, *Nonliner partial differntial equations and their applications*, H. Brezis and J. L. Lions (eds.), Pitman, (1989), *pp.* 13–51.

(1.47) 藤間, “モルタル有限要素法による流れ問題の並列計算”, 計算工学講演会論文集, **1**, (1996), *pp.* 59–62.

第 2 章

有限要素解析の並列化および性能評価

2.1 はじめに

大規模有限要素解析においては、連立一次方程式の解法に多大な計算時間と記憶容量が必要とするが、この解法には大別して、直接解法と反復解法がある。ガウスの消去法に代表される直接解法は、未知数を順次消去していくため有限回の演算で安定した解が得られ、また演算手法やプログラミングも容易で利用しやすいことから、一般に広く利用されている。しかし、この並列化は、必要とする記憶容量や多いことや並列処理の粒度の面で問題が多いとされている。一方、共役勾配法に代表される反復解法は、近似解を収束計算により正解に近づけていく方法であるため、問題の性質や収束判定値によって計算時間や解の精度が変化する。しかし、有限要素法のように係数マトリックスが疎行列 (スパース・マトリックス) の場合には非零成分のみを記憶すれば良いため、大規模解析においては記憶容量の面で有利である。

有限要素解析を並列化する方法としては、領域分割法と呼ばれる全体の解析領域を分割して並列解析する方法が有力であるとされる[2.1–2.3]。しかし、領域の切り方によって演算量や通信量が大きく変化するとともに、領域間境界に対して適用される共役勾配法の収束性が変化するため、並列性能が大きく変化する。したがって、並列性能を最大限に引き出すためには、領域の分割方法を最適なものに設定することが重要となる。

本章では、ワークステーション・クラスタを用いた並列有限要素解析法として、ガウスの消去法および共役勾配法を並列化する方法を開発し、さらに領域分割法を加えた 3 つの方法について、それぞれの並列化方法および並列解析システムの実装方法について述べる。さらに、これらの並列性能の定量的な評価および比較を行なうため、並列解析

時間を演算量と通信量をもとに予測する評価式を導出するとともに、この評価式に基づいてそれぞれの方法における並列パラメータの最適化を行なう。これにより、各並列化方法の性能の限界を明らかにするとともに、実際に 35 万自由度を越える 3 次元解析に適用して、これらの方法の有効性を確認する。

2.2 並列化方法

2.2.1 ワークステーション・クラスタによる並列処理

(a) マスター/スレーブモデルに基づいたシステムの実装

ネットワークに接続された多数のワークステーションを、1 つの分散メモリ MIMD 型の並列計算機とみなして並列計算を行なう。並列解析システムの実装は、マスター/スレーブモデルに基づいて、Fig. 2.1 に示されるように、1 台のワークステーションを全体の処理を管理するマスター・マシン、残りを並列計算を行なうスレーブ・マシンとに分ける。スレーブ・マシンには、独立に計算可能な処理を割り当てておく。

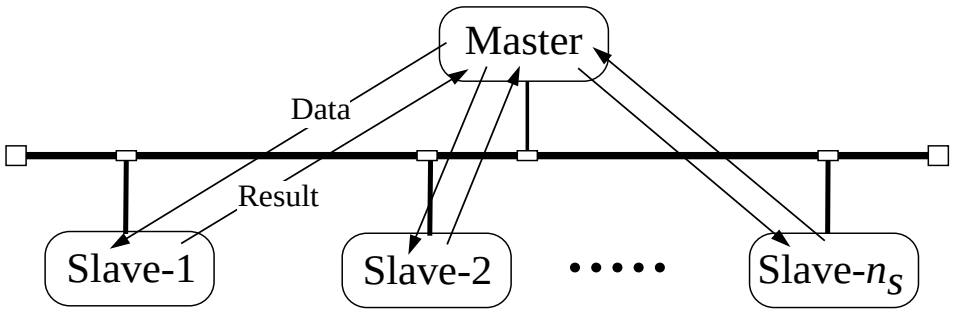


Fig. 2.1 Schematic diagram of a workstation cluster system

並列解析の全体の流れは、Fig. 2.2 に示すように、マスター・マシンとスレーブ・マシンとの間でデータ転送を行ないながら進められる。具体的には、マスター・マシンでは、

1. 初期化

   使用するワークステーションを指定してスレーブ・プロセスを起動し、通信の初期化を行なう。

2. データの送信

   並列計算を行なうのに必要なデータを各スレーブ・マシンに送信する。

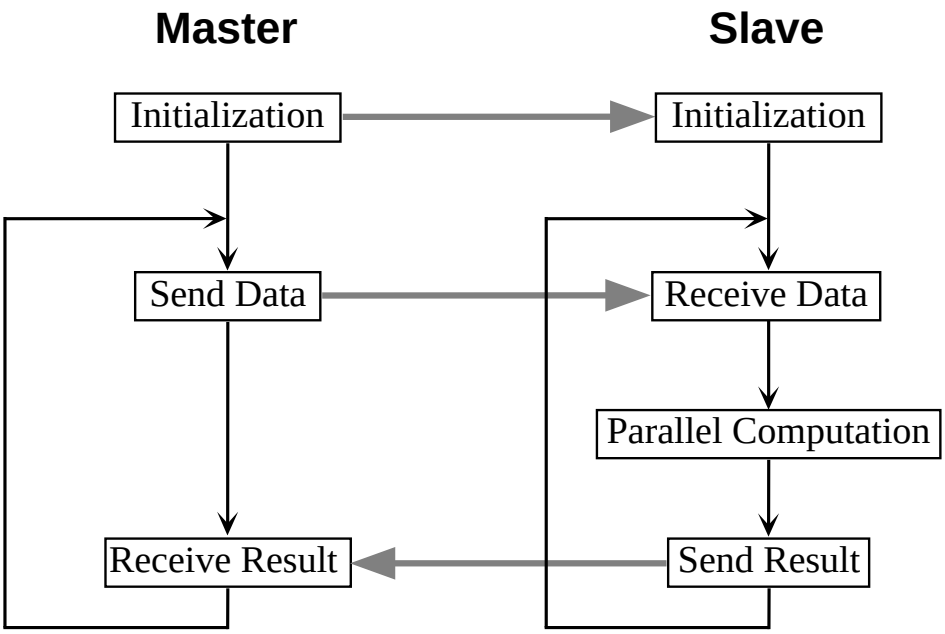


Fig. 2.2 Message passing

3. 計算結果の受信

   各スレーブ・マシンで行なわれた計算の結果を受信し、これを用いてマスター・マシンが担当する処理を行なう。

4. 上記の 2. 3. を全体の処理が終了するまで繰り返す。

また、スレーブ・マシンでは、

1. 初期化

   マスター・マシンからプロセスが起動され、通信の初期化を行なう。

2. データの受信

   担当している計算を行なうのに必要なデータを受信する。

3. 並列計算

   割り当てられた計算を実行する。

4. 計算結果の送信

   計算結果を送信する。

5. 上記の 2. から 4. を繰り返す。

このように、マスター・マシンとスレーブ・マシンとの間では、データの受渡しや各処理の同期を取る必要があるため、メッセージ・パッシングを用いてデータ (メッセージ) の送受信を行なう。

(b) ソケットを用いたメッセージ・パッシング

本研究では、ワークステーション・クラスタにおいてメッセージ・パッシングを実現するため、UNIX に実装されているソケット[2.4]を用いてデータ通信を行なう。

ソケットは、プロセス間の通信チャンネルを提供する機能として、BSD UNIX に実装されている。この機能は、Fig. 2.3 に示すように、ISO/OSI (International Organization for Standardization on Open System Interconnection) の 7 階層参照モデルのうち、トランスポート層に対するアプリケーション・プログラム・インターフェース (API) を提供する。この通信プロトコルは、通信方式が異なる型 (type) と通信範囲が異なる定義域 (address family) の組合せにより異なる。

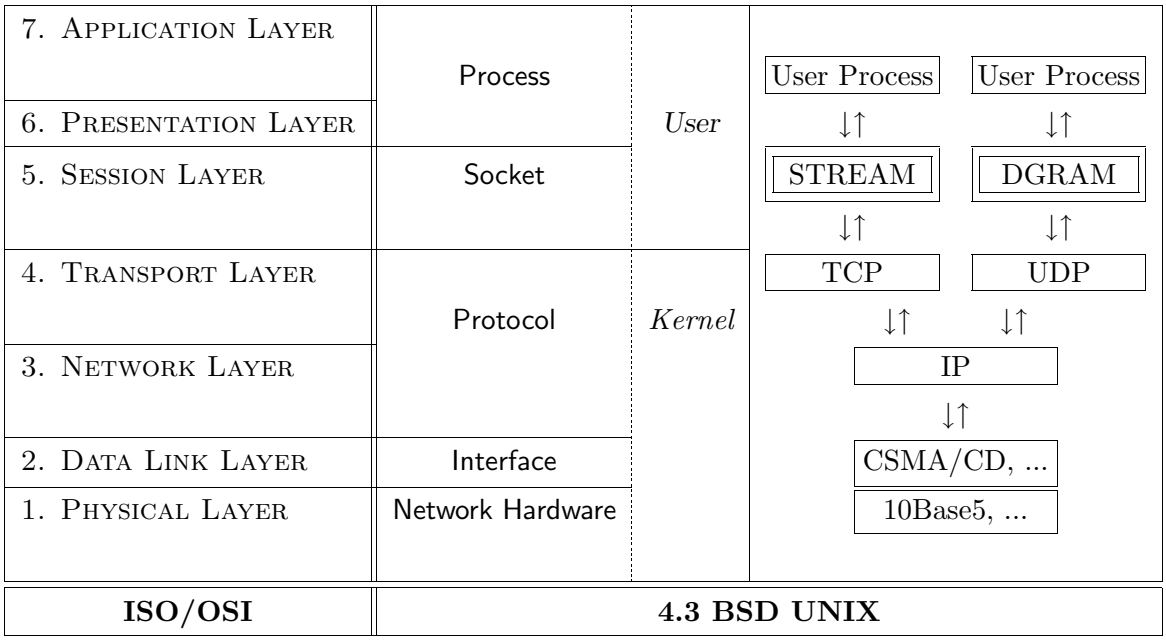


Fig. 2.3 Network structure

ネットワークを介したインターネット・ドメインのソケット・インターフェースには、Fig. 2.3 に示すように、TCP (Transmission Control Protocol) によって実装されている

ストリーム (STREAM)・ソケットと、UDP (User Datagram Protocol) によって実装されているデータグラム (DGRAM)・ソケットがある。ストリーム・ソケットは、Fig. 2.4 に示すように、通信を行なうプロセス間にバーチャル・サーキットと呼ばれる仮想的な全二重通信路を設定し、ソケット同士を接続して通信を行なう。したがって、双方向通信の信頼性が高く、順序正しいデータ通信が提供される。これに対しデータグラム・ソケットは、ソケット同士の接続を一切行なわず、Fig. 2.4 に示すように、データパケットを用いてそれぞれのネットワーク層の機能により通信が提供される。このため、ストリーム・ソケットに比べて、ソケットの接続といった通信オーバヘッドの低減が期待できるが、複数のデータグラムが発送された場合などには、発信された順序で受信される保証はなく、データの到着そのものすら保証されない。

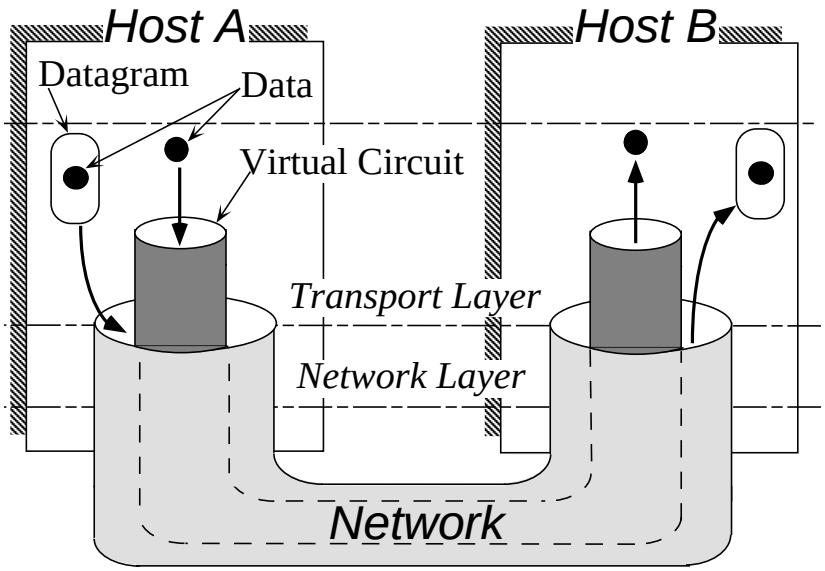


Fig. 2.4 BSD socket interface

そこで本研究では、ストリーム・ソケットを用いてメッセージ・パッシングを行なうこととする。また、ワークステーションとして、Sun SPARCstation 2 (28.5 MIPS, 4.2 MFLOPS, 主メモリ 48 MB)、ネットワークには Ethernet (10Base-5, 10 Mbps) を使用する。

2.2.2 並列ガウス消去法

大規模有限要素解析の連立一次方程式解法にガウスの消去法を適用すると、前進消去の部分にCPU時間のほとんどを費やす。そこで、前進消去の処理を分散・並列化し、大規模解析の高速化を実現する。

有限要素法の剛性マトリックスは、通常、対称で対角付近の比較的近い部分に非零成分が並んだバンド・マトリックスである。このため、バンド・マトリックスにガウスの消去法を適用する場合、上三角 (あるいは、下三角) 部分の非零成分が充満したバンド部分に対して演算を行なうことで、演算量を節約することができる。

マトリックス a_{ij} に対する前進消去は、ピボット行 k を用いて、それ以降の行 i および列 j の成分に対して

$$a_{ij} = a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj} \tag{2.1}$$

の演算を行なうため、プログラム上において kij の3重ループとなる。これらの反復ループのうち、内側の2つのループ i および j に関してはデータ依存性がないことから並列化することができる[2.5–2.7]。特に、ベクトル・プロセッサを複数備えた共有メモリ型のベクトル並列計算機においては、最内側ループ i をベクトル化し、さらにループ j を並列化することにより、高効率な並列化を実現することができる[2.8–2.10]。しかし、この並列化手法をそのままワークステーション・クラスタのような分散メモリ環境に適用しようとすると、ピボット行 k を1行ずつ転送して演算を行なわなければならないため、並列処理の粒度が非常に細かくなることから、高い並列効率を得ることができない。

そこで、Fig. 2.5 に示すように、上三角部分のバンド・マトリックスを一定行ごとのブロックに分割し、このブロック単位に前進消去の演算を並列に行なう。まず、第1ブロック内を前進消去すると、これをピボット・ブロックとして第2ブロック以降の対応する列の前進消去が可能となる。次に、第2ブロック内の前進消去を行なうと、これを用いて第3ブロック以降の前進消去を同様に行なうことができる。したがって、各ブロックを各スレーブ・マシンに割り当てておけば、前進消去が終了したピボットブロックを用いて、並列に前進消去を行なうことができる。

まず、剛性マトリックスの担当ブロックを各スレーブ・マシン上で分散作成する。第1ブロックを担当するスレーブ・マシンは、ブロック内の前進消去を行なう。マスター・マシンはこの最初のピボットブロックをスレーブ・マシンから受信し、他のスレーブ・マシンに転送する。各スレーブ・マシンは受信したピボットブロックを用いて、担当ブロックにおける対応する列の前進消去を行なう。この処理を全てのピボットブロックに

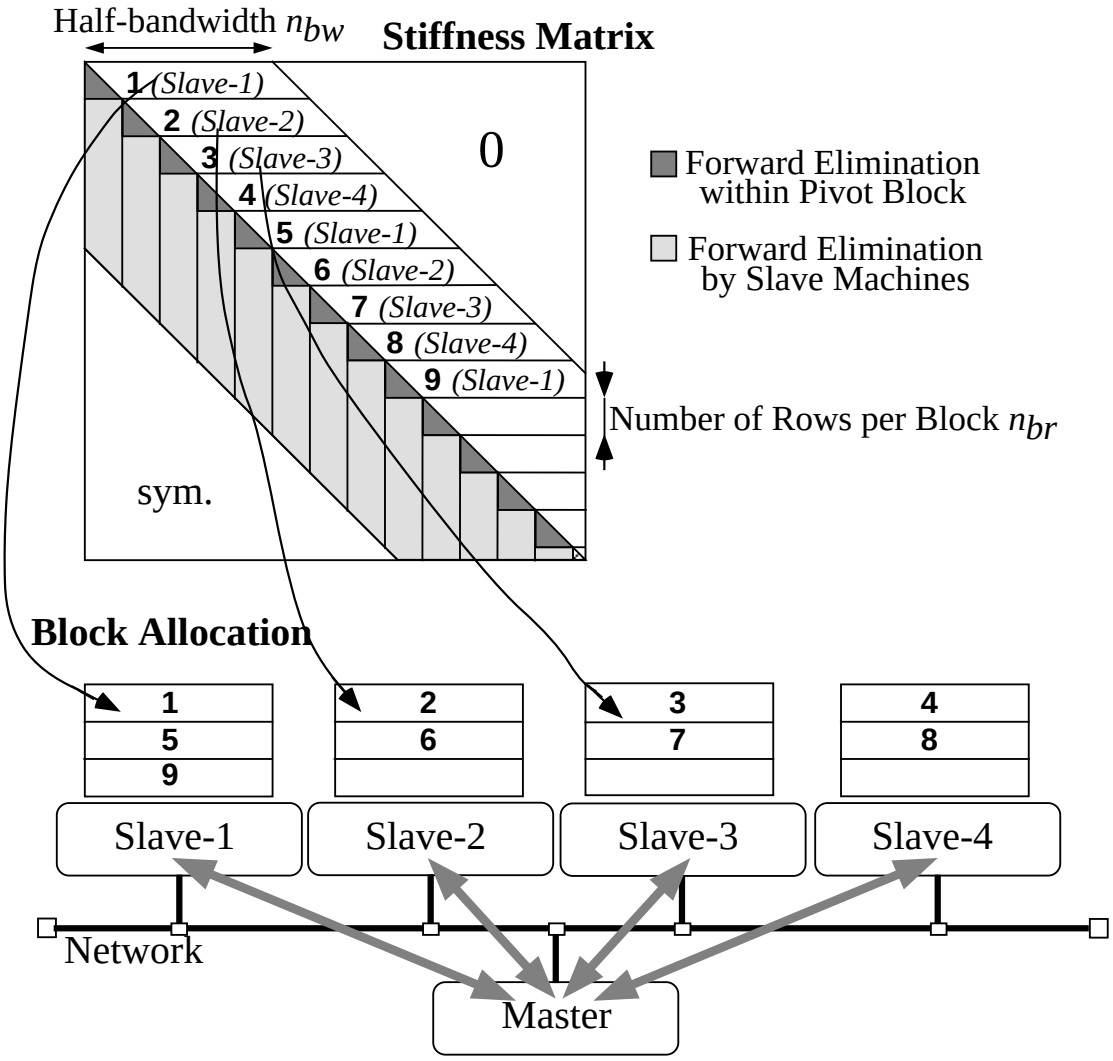


Fig. 2.5 Allocation of the stiffness matrix to the slave machines

対して行なうことにより、前進消去が完了する。なお、後退代入についても、代入する解ベクトルを転送しながら各スレーブ・マシン上で行うことができる。また、各ブロック内の前進消去は並列化されないため、ブロックの行数により並列性能が変化することになる。

2.2.3 並列共役勾配法

正値対称なマトリックス [**K**] を係数とする連立一次方程式

$$[\mathbf{K}]\{\mathrm{u}\} = \{\mathrm{f}\} \tag{2.2}$$

の解法は、汎関数

$$\Pi\left(\{\mathrm{u}\}\right) = \frac{1}{2}\{\mathrm{u}\}^T[\mathbf{K}]\{\mathrm{u}\} - \{\mathrm{f}\}^T\{\mathrm{u}\} \tag{2.3}$$

の最小化問題と等価である。共役勾配法 (Conjugate Gradient Method, CG 法) は、探索方向ベクトル {p} と残差ベクトル {r} により、この最小化問題を次の反復計算により解く[(2.11)]。

Step 0: 初期化

$$\{\mathrm{r}\}_0 = \{\mathrm{f}\} - [\mathbf{K}]\{\mathrm{u}\}_0 \qquad (\{\mathrm{u}\}_0 \text{は初期値}) \tag{2.4}$$

$$\{\mathrm{p}\}_0 = \{\mathrm{r}\}_0 \tag{2.5}$$

$$n \leftarrow 0$$

Step 1: 解と残差の修正

$$\alpha_n = \frac{\{\mathrm{r}\}_{n+1}^T\{\mathrm{r}\}_{n+1}}{\{\mathrm{p}\}_n^T[\mathbf{K}]\{\mathrm{p}\}_n} \tag{2.6}$$

$$\{\mathrm{x}\}_{n+1} = \{\mathrm{x}\}_n + \alpha_n\{\mathrm{p}\}_n \tag{2.7}$$

$$\{\mathrm{r}\}_{n+1} = \{\mathrm{r}\}_n - \alpha_n[\mathbf{K}]\{\mathrm{p}\}_n \tag{2.8}$$

Step 2: 収束判定

$$\frac{||\{\mathrm{r}\}_{n+1}||_2}{||\{\mathrm{f}\}||_2} \leq \varepsilon \quad \text{なら終了} \tag{2.9}$$

Step 3: 降下方向の更新

$$\beta_n = \frac{\{\mathrm{r}\}_{n+1}^T\{\mathrm{r}\}_{n+1}}{\{\mathrm{r}\}_n^T\{\mathrm{r}\}_n} \tag{2.10}$$

$$\{\mathrm{p}\}_{n+1} = \{\mathrm{r}\}_{n+1} + \beta_n\{\mathrm{p}\}_n \tag{2.11}$$

$n \leftarrow n+1$ として **Step 1** に戻る

ここで、α^n, β^n はそれぞれ、n 回目の反復ステップにおける解の探索方向への修正量と次ステップにおける探索方向の修正量である。

共役勾配法の反復過程においては、係数マトリックス $[\mathbf{K}]$ と探索方向ベクトル $\{\mathrm{p}\}_n$ の積の演算部分 (式 (2.6) および式 (2.8)) に CPU 時間のほとんどを費やす。そこで、剛性マトリックスをスレーブ・マシン上に分散し、この演算を並列化して高速化をはかる。なお、剛性マトリックスの分散方法の違いにより、ブロック型共役勾配法と領域型共役勾配法の 2 つの方法を提案する。

(a) ブロック型共役勾配法

Fig. 2.6 に示すように、剛性マトリックス [**K**] を並列台数分に行方向にブロック分割することとし、それぞれのブロックを各スレーブ・マシン上で分散作成する。このとき剛性マトリックス内に多数含まれる零成分は演算に必要ないため、非零成分のみを記憶すると記憶容量および演算量が削減できる。探索方向ベクトル {p} を各スレーブ・マシンに転送し、割り当てられたブロックとの積 $\{\mathrm{b}_k\}$ を計算する。全てのスレーブ・マシンにおける計算結果をマスター・マシンに集めれば、剛性マトリックスと探索方向ベクトルとの積 {b} が得られたことになり、解の修正や次の反復ステップで必要となる残差や探索方向ベクトルといったパラメータの演算が行なえる。このとき、各スレーブ・マシンへの探索方向ベクトルを転送する際にマトリックスの非零成分に対応する成分のみとすることにより、データ転送量を減らすことができる。

$$\{\mathrm{b}\} = [\mathbf{K}]\{\mathrm{p}\}_n$$

$$\Downarrow$$

$$\begin{Bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_{n_s} \end{Bmatrix} = \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \\ \vdots \\ \mathbf{K}_{n_s} \end{bmatrix} \begin{Bmatrix} \\ \mathrm{p} \\ \\ \end{Bmatrix}_n$$

$$\Downarrow$$

$$\underbrace{\{\mathrm{b}_1\} = [\mathbf{K}_1]\{\mathrm{p}\}_n}_{\text{Slave 1}} \quad \underbrace{\{\mathrm{b}_2\} = [\mathbf{K}_2]\{\mathrm{p}\}_n}_{\text{Slave 2}} \quad \cdots \quad \underbrace{\{\mathrm{b}_{n_s}\} = [\mathbf{K}_{n_s}]\{\mathrm{p}\}_n}_{\text{Slave } n_s}$$

Fig. 2.6 Parallel submatrix-vector product operation

(b) 領域型共役勾配法

Fig. 2.7 のように、全体の解析領域を分割してスレーブ・マシンに割り当て、領域ごとの部分剛性マトリックス $\left[\mathbf{K}^{(k)}\right]$ を分散作成する。探索方向ベクトルを各スレーブ・マシンに転送し、部分領域単位に剛性マトリックスと探索方向ベクトルとの積を行なう。この結果をマスター・マシンに集め、全領域について領域間境界の自由度成分の総和をとると、マトリックス・ベクトル演算の結果を得ることができる。この場合にも、部分

剛性マトリックス $\left[\mathbf{K}^{(k)}\right]$ は非零成分のみを記憶するとともに、探索方向ベクトルの転送は演算を行なう部分領域の自由度成分のみとすることにより、演算量およびデータ転送量を減らすことができる。

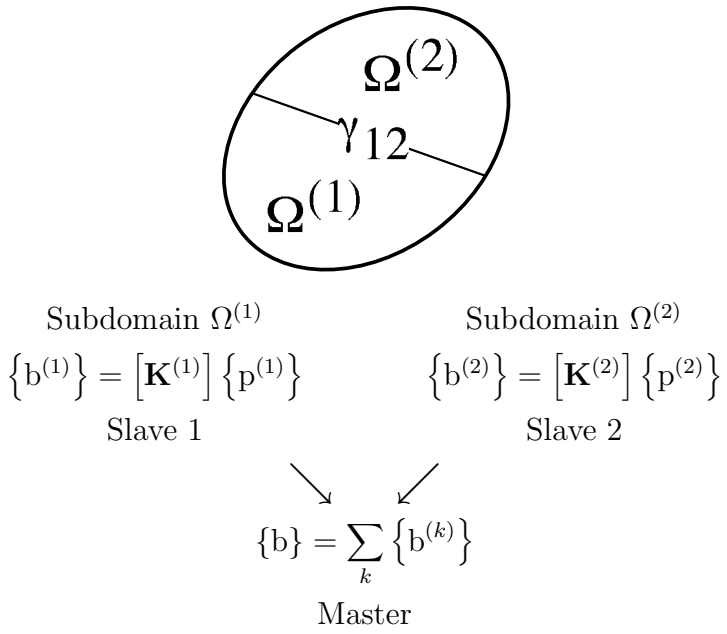


Fig. 2.7 Domain partitioned CG method

2.2.4 領域分割法

(a) 方法の概要

領域分割法は解析領域を部分領域に分割して解析を行い、部分領域間の解の連続性は共役勾配法などの収束計算で満足させる方法である[2.1–2.3]。

部分領域 (k) における剛性方程式を内部自由度 $\left\{\mathrm{u}^{(k)}\right\}$ と領域間境界自由度 $\{\mu\}$ に分けて

$$\begin{bmatrix} \mathbf{K}_{11}^{(k)} & \mathbf{K}_{12}^{(k)} \\ \mathbf{K}_{12}^{(k)^T} & \mathbf{K}_{22}^{(k)} \end{bmatrix} \begin{Bmatrix} \mathrm{u}^{(k)} \\ \mu \end{Bmatrix} = \begin{Bmatrix} \mathrm{F}^{(k)} \\ \tau^{(k)} \end{Bmatrix} \tag{2.12}$$

と書くことにする。ここで、$\left[K_{ij}^{(k)}\right]$ は領域の剛性マトリックス、$\left\{F^{(k)}\right\}, \left\{\tau^{(k)}\right\}$ は荷重ベクトルである。式 (2.12) は、上側と下側の 2 つの部分に分けて

$$\left[\mathbf{K}_{11}^{(k)}\right]\left\{\mathrm{u}^{(k)}\right\} = \left\{\mathrm{F}^{(k)}\right\} - \left[\mathbf{K}_{12}^{(k)}\right]\{\mu\} \tag{2.13}$$

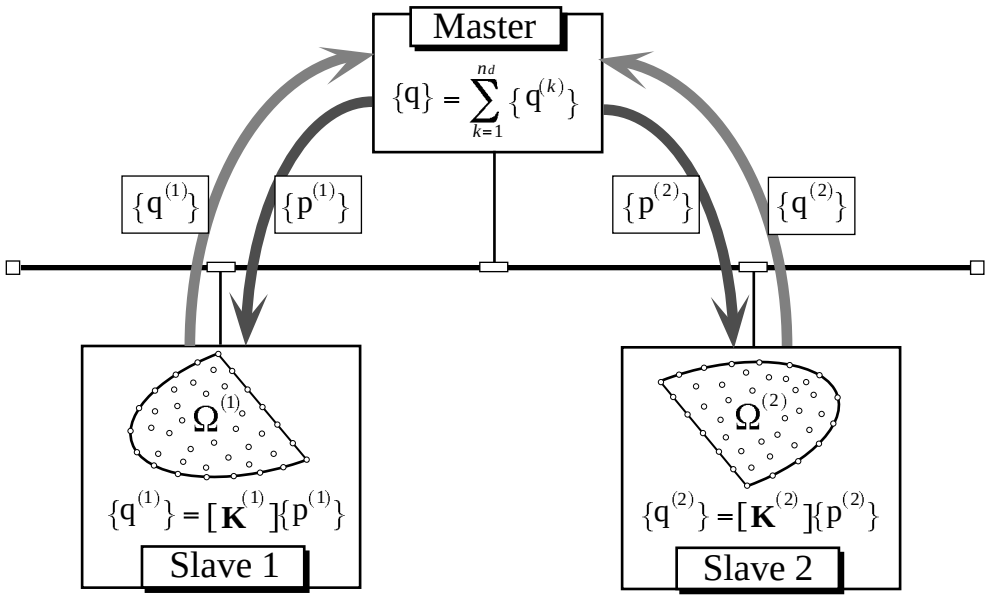


Fig. 2.8 Parallel implementation of the domain partitioned CG method

$$\left[\mathbf{K}_{12}^{(k)}\right]^T \left\{\mathrm{u}^{(k)}\right\} + \left[\mathbf{K}_{22}^{(k)}\right] \{\mu\} = \left\{\tau^{(k)}\right\} \tag{2.14}$$

と表される。式 (2.13) より得られる $\left\{\mathrm{u}^{(k)}\right\}$ を式 (2.14) に代入すると、

$$\left[\left[\mathbf{K}_{22}^{(k)}\right] - \left[\mathbf{K}_{12}^{(k)}\right]^T \left[\mathbf{K}_{11}^{(k)}\right]^{-1} \left[\mathbf{K}_{12}^{(k)}\right]\right] \{\mu\} = \left\{\tau^{(k)}\right\} - \left[\mathbf{K}_{12}^{(k)}\right]^T \left[\mathbf{K}_{11}^{(k)}\right]^{-1} \left\{\mathrm{F}^{(k)}\right\} \tag{2.15}$$

のように、内部自由度が消去される。

通常のサブストラクチャー法においては、式 (2.15) を全領域について

$$\begin{aligned} &\sum_k \left[\left[\mathbf{K}_{22}^{(k)}\right] - \left[\mathbf{K}_{12}^{(k)}\right]^T \left[\mathbf{K}_{11}^{(k)}\right]^{-1} \left[\mathbf{K}_{12}^{(k)}\right]\right] \{\mu\} \\ &\qquad = \sum_k \left\{\left\{\tau^{(k)}\right\} - \left[\mathbf{K}_{12}^{(k)}\right]^T \left[\mathbf{K}_{11}^{(k)}\right]^{-1} \left\{\mathrm{F}^{(k)}\right\}\right\} \end{aligned} \tag{2.16}$$

と全体化したのち、これを解いて $\{\mu\}$ を求める。$\{\mu\}$ が求まると、各領域ごとに、式 (2.13) より内部自由度の変位 $\left\{\mathrm{u}^{(k)}\right\}$ が得られる。しかし、領域分割法では、式 (2.16) の解法に共役勾配法などの反復解法を適用することにより、式 (2.15) を全体系に組み込むことなく解くことができる。式 (2.16) を共役勾配法によって解く際には、左辺の係数マトリックスと探索方向ベクトルとの積

$$\left(\sum_k \left[\left[\mathbf{K}_{22}^{(k)}\right] - \left[\mathbf{K}_{12}^{(k)}\right]^T \left[\mathbf{K}_{11}^{(k)}\right]^{-1} \left[\mathbf{K}_{12}^{(k)}\right]\right]\right) \{\mathrm{p}\}_n \tag{2.17}$$

の演算を行なうが、これは、

$$\sum_k \left(\left[\left[\mathbf{K}_{22}^{(k)} \right] - \left[\mathbf{K}_{12}^{(k)} \right]^T \left[\mathbf{K}_{11}^{(k)} \right]^{-1} \left[\mathbf{K}_{12}^{(k)} \right] \right] \{\mathrm{p}\}_n \right) \tag{2.18}$$

のように、領域ごとにマトリックス・ベクトル演算を行ない、その結果を全体化することにより、等価な演算結果を得ることができる。

領域ごとの演算は、式 (2.12) の $\{\mu\}$ の部分に探索方向ベクトル $\{\mathrm{p}\}_n$ をいわば境界条件として与え、これに対応する反力 $\left\{\tau^{(k)}\right\}$ 計算する。この時、式 (2.12) 中で、$\left\{\mathrm{F}^{(k)}\right\}$ は 0 とするため、式 (2.15) において

$$\left[\left[\mathbf{K}_{22}^{(k)} \right] - \left[\mathbf{K}_{12}^{(k)} \right]^T \left[\mathbf{K}_{11}^{(k)} \right]^{-1} \left[\mathbf{K}_{12}^{(k)} \right] \right] \{\mathrm{p}\}_n \tag{2.19}$$

を計算したことになる。この演算結果を全体系のベクトルにまとめ、これを用いて共役勾配法のパラメータ演算を行なう。

(b) 本方法の特徴

領域分割法は、従来からあるスレーブ・ストラクチャー法に共役勾配法を適用したものと等価な方法であるが、前節で述べたように、領域ごとに $\left\{\tau^{(k)}\right\}$ を求める方が並列化される演算部分が多く、またプロセッサ間のデータ交換量も領域間境界自由度分のベクトルへと小さくなることから、高効率の並列化が実現できる。

領域型共役勾配法と比較すると、領域を分割することにより並列化を実現するという点では共通するが、各領域の内部自由度が縮小され共役勾配法の反復計算が適用される自由度数が境界自由度数分へと小さくなっていることから、データ転送量が小さくなるとともに収束性も向上する。また、領域内の解析には直接解法、領域間には反復解法をそれぞれ適用し、2 つの解法のそれぞれの利点を生かしてうまく組み合わせることにより、高性能な並列化が期待できる。しかし領域分割数を変えると、この 2 つの解法のバランスが変わることによって、収束性のほか並列性能も大きく変化する。極端な場合、領域分割数を減らして全体領域を 1 領域として解析すると、直接解法による解析と等価になり、逆に、領域分割数を増して 1 要素を 1 領域に設定して解析すると、共役勾配法による解析と等価になる。したがって、並列性能が最大となる最適な領域分割数が存在する。

(c) ワークステーション・クラスタへのシステムの実装

ワークステーション・クラスタを用いた並列システムでは、部分領域の計算をスレーブ・マシンに割り当てて並列に解析を行ない、領域間境界自由度に関する共役勾配法の

パラメータ演算はマスター・マシンで行う。すなわち、マスター・マシン上で計算された探索方向ベクトルを全スレーブ・マシンに転送する。各スレーブ・マシンは担当する部分領域について、残差計算などに必要となるベクトルを計算し、マスター・マシンへ返す。なお、高速化をはかるため、各部分領域の剛性マトリックスの前進消去は反復の 1 回目にのみ行い、消去後のマトリックスをそのままスレーブ・マシン上に保存して反復の 2 回目以降は後退代入のみで済ませることにする。

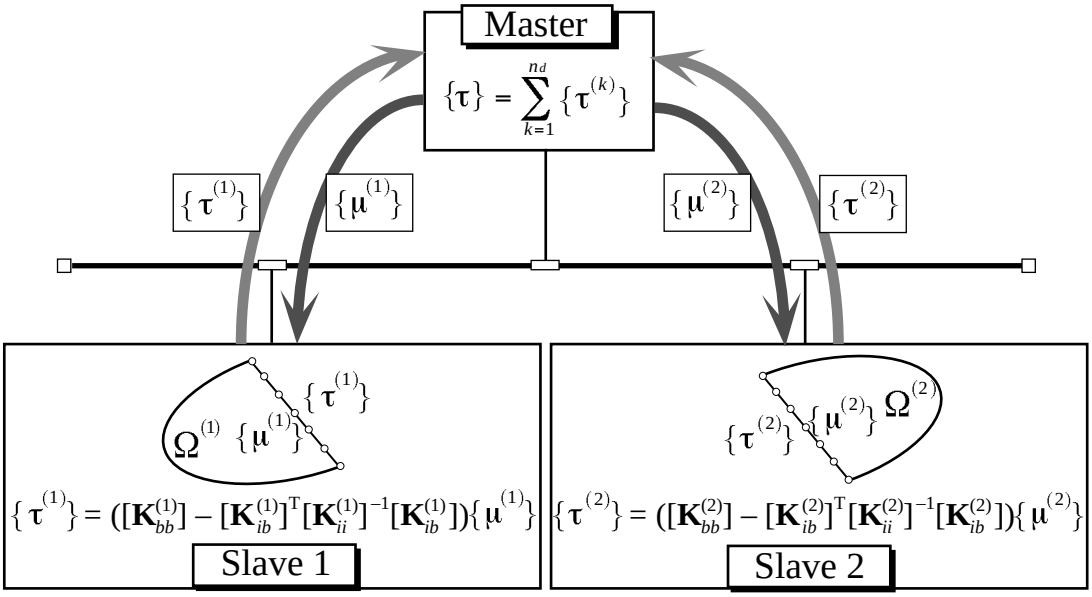


Fig. 2.9 Parallel implementation of the domain decomposition method

2.3 並列性能の定量化

2.3.1 解析時間の評価方法

並列時の解析時間、加速率および並列効率といった並列性能は、問題の規模や並列台数、計算の分散方法などに大きく依存する。そこで、並列化の基本パラメータを用いて解析時間を定量化する。

たとえば、並列ガウス消去法の場合、Fig. 2.10 (a) のように各スレーブ・マシンはマスター・マシンからピボットブロックを受け取ると、担当ブロックに対する前進消去を開始する。消去が終了すると次のピボットブロックを担当しているスレーブ・マシンは、ブロック内消去を行なった後マスター・マシンにこのブロックを送信する。この操作を

全ブロックに対して繰り返し、前進消去が終了することになる。このタイムチャートに基づき各処理の演算量や通信量を積算すると、並列解析時間が基本パラメータの関数として求まる。演算速度や通信速度に関する係数は、小規模な例題の実測により求めることにする。

並列台数を増すと、1 台当たりの演算量は減少するが、データ転送量は増加するためネットワークの通信容量は限界に達し、ネットワークの飽和が起こる。この時、スレーブ・マシン側に通信待ちが生じ処理時間が増大するが、このような場合にも Fig. 2.10 (b) に基づき解析時間が求まる。

なお、この並列性能の定量化方法は、並列共役勾配法や領域分割法に対しても適用できる。

2.3.2 並列ガウス消去法

並列ガウス消去法の各処理に要する時間は次のように表される。

- 1 回のピボットブロックの転送

$$T_c = \alpha_c(n_{bw}+1)n_{br} + \beta \tag{2.20}$$

- 1 台のスレーブ・マシンが担当するブロック消去

$$T_f = \alpha_f(n_{br}n_s + n_{br} + n_{bw} - 1)\frac{n_{br}n_{bw}}{n_s} \tag{2.21}$$

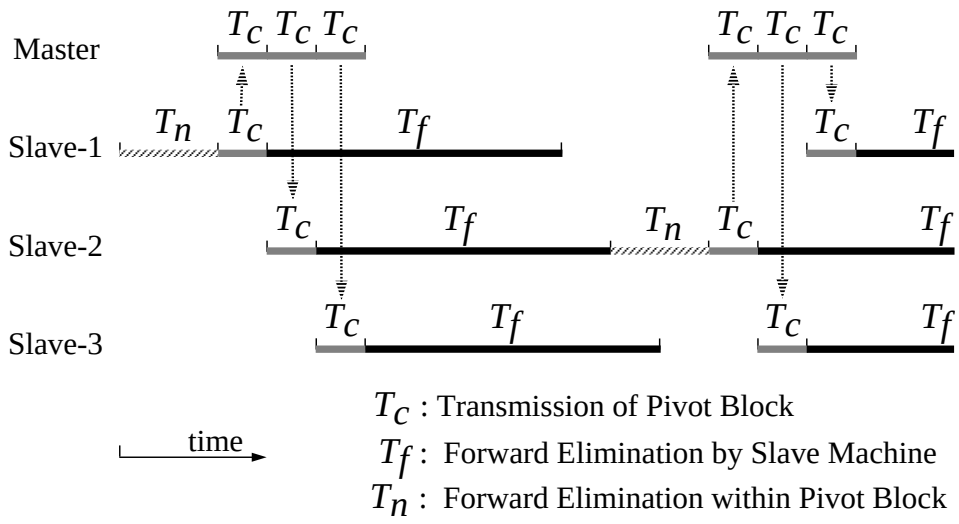
- ブロック内前進消去

$$T_n = \alpha_n n_{br}^2 n_{bw} \tag{2.22}$$

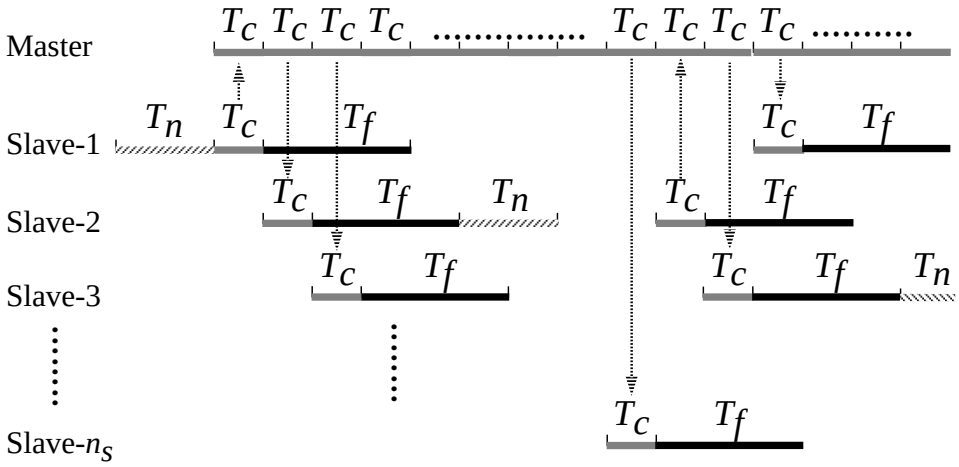
- 後退代入

$$T_b = \alpha_c(n_s n_{eq} - n_s n_{br} + n_{br}) + \beta\left(2\frac{n_{eq}}{n_{br}} - 1\right) + \alpha_b n_{bw} n_{eq} \tag{2.23}$$

ここで、n_s は並列マシン台数、n_{br} はブロックの行数、n_{eq} は全自由度数、n_{bw} は剛性マトリックスのバンド幅であり、$\alpha_c, \alpha_f, \alpha_n, \alpha_b, \beta$ はこれらのパラメータに依存しない定数である。これらの定数を小規模例題の解析により実測したところ、Table 2.1 に示す値となった。なお、この定数は、演算精度が単精度 (Single Precision) および倍精度 (Double Precision) のそれぞれの場合について測定した。また、式 (2.20)〜(2.23) はいずれも並列台数 n_s とブロック行数 n_{br} の関数であり、この二つのパラメータの設定が解析時間に影響する。



(a) normal condition



(b) saturated condition

Fig. 2.10 Time chart of parallel Gaussian elimination method

式 (2.20)〜(2.22) はいずれも 1 ブロック当たりのものであるため、解析時間は Fig. 2.10 (a) に基づき

$$\begin{aligned} T &= (2T_c + T_f + T_n)\frac{n_{eq}}{n_{br}} + T_b \\ &= C_1\frac{1}{n_s} + C_2\frac{n_{br}}{n_s} + C_3 n_{br} + C_4\frac{1}{n_{br}} \\ &\quad + C_5 n_s n_{br} + C_6 n_s + C_7 \end{aligned} \tag{2.24}$$

Table 2.1 Coefficients of operation and communication speed (parallel Gaussian elimination method)

	Single Precision	Double Precision
α_f	2.34420×10^{-7}	3.18092×10^{-7}
α_n	1.83096×10^{-7}	2.76550×10^{-7}
α_b	5.04579×10^{-7}	1.05641×10^{-6}
α_c	6.13110×10^{-6}	1.22622×10^{-5}
β	8.34818×10^{-3}	

となる。ここで C_1〜C_7 は並列パラメータに依存しない係数であるが、支配的な項は第1〜4 項であるため、並列台数 n_s に対して反比例して解析時間が減少する。ブロック行数 n_{br} については第 2、3 項は比例、第 4 項は反比例の関係であるため、解析時間が最小となる n_{br} の最適値が存在することになる。

一方、ネットワークの飽和が生じた場合には Fig. 2.10 (b) に基づき

$$\begin{aligned} T &= T_c \frac{n_{eq}}{n_{br}} n_s + T_b \\ &= C_8 n_s + C_9 n_s n_{br} + C_{10} \frac{n_s}{n_{br}} + C_{11} n_{br} \\ &\quad + C_{12} \frac{1}{n_{br}} + C_{13} \end{aligned} \tag{2.25}$$

と表され、並列台数 n_s に比例して解析時間が増加することになる。

2.3.3 並列共役勾配法

並列共役勾配法の場合も同様にして、以下のように解析時間を定量化することができる。

(a) ブロック型共役勾配法

剛性マトリックスを大きさが等しい n_s 個のブロックに分割し、n_s 台で並列解析を行なう時の各処理に要する時間は次のように表される。

- スレーブ・マシンへの探索方向ベクトルの転送

$$T_s = \alpha_c \left(2n_{bw} + \frac{n_{eq}}{n_s} \right) + \beta \tag{2.26}$$

- スレーブ・マシンにおけるマトリックス・ベクトル演算

$$T_a = \alpha_a \frac{n_{eq}}{n_s} \tag{2.27}$$

- スレーブ・マシンの演算結果の転送

$$T_r = \alpha_c \frac{n_{eq}}{n_s} + \beta \tag{2.28}$$

- マスター・マシンにおける修正演算

$$T_p = \alpha_p n_{eq} \tag{2.29}$$

ここで、α_a, α_p は定数であるため、解析時間に影響する並列パラメータは並列台数 n_s のみである。これらの定数を測定した結果を、Table 2.2 に示す。なお、反復計算における数値丸め誤差を抑えるため、演算精度は倍精度とする。

Table 2.2 Coefficients of operation and communication speed (Block CG method)

	Double Precision
α_a	4.780×10^{-4}
α_p	3.360×10^{-5}
α_c	1.226×10^{-5}
β	8.348×10^{-3}

全解析時間は収束までの反復回数を I_t として

$$\begin{aligned} T &= (T_s + T_r n_s + T_a + T_p) I_t \\ &= C_{14} \frac{1}{n_s} + C_{15} n_s + C_{16} \end{aligned} \tag{2.30}$$

となる。ここで、支配的な項は第 1 項であり、並列台数 n_s に反比例して解析時間が減少する。また、ネットワークの飽和が生じた場合には

$$\begin{aligned} T &= \{(T_s + T_r) n_s + T_p\} I_t \\ &= C_{17} n_s + C_{18} \end{aligned} \tag{2.31}$$

となるため、並列台数に比例して解析時間が増加することになる。

(b) 領域型共役勾配法

各領域の自由度数 n'_{eq} が等しくなるように n_d 個の領域に分割した場合、反復 1 回当たりの各処理時間は次のように表される。

- スレーブ・マシンへの探索方向ベクトルの転送

$$T_p = \alpha_c n'_{eq} \frac{n_d}{n_s} + \beta \tag{2.32}$$

- スレーブ・マシンにおけるマトリックス・ベクトル演算

$$T_d = \alpha_d n'_{eq} \frac{n_d}{n_s} \tag{2.33}$$

- スレーブ・マシンの演算結果の転送

$$T_g = \alpha_c n'_{eq} \frac{n_d}{n_s} + \beta \tag{2.34}$$

- マスター・マシンにおける修正演算

$$T_m = \alpha_p n_{eq} \tag{2.35}$$

ここで、α_d, α_m は定数であるため、解析時間は総領域数 n_d と並列台数 n_s によって変化する。これらの定数を測定した結果を、Table 2.3 に示す。

Table 2.3 Coefficients of operation and communication speed (Domain CG method)

	Double Precision
α_d	7.024×10^{-5}
α_m	8.486×10^{-6}
α_c	1.226×10^{-5}
β	8.348×10^{-3}

全解析時間は、反復回数 I_t より

$$T = (T_d + T_p + T_g n_s + T_m)\, I_t \tag{2.36}$$

と表される。ここで、支配的な項は第 1 項目であるから、並列台数 n_s に反比例して解析時間が減少する。また、ネットワークの飽和が生じた場合には、

$$T = (T_p + T_g)\, n_s I_t + T_m I_t \tag{2.37}$$

と表される。

2.3.4 領域分割法

領域分割法で、各部分領域の自由度数 n'_{eq} およびバンド幅 n'_{bw} を一定とすると、反復 1 回当たりの各処理時間は次のように表される。

- 領域間自由度データの送信

$$T_{co} = \alpha_c n_{bf} + \beta \tag{2.38}$$

- 領域内の解析 (反復 1 回目)

$$T_{d1} = \alpha_{d1} n'_{eq} n'^2_{bw} \frac{n_d}{n_s} \tag{2.39}$$

- 領域内の解析 (反復 2 回目以降)

$$T_{d2} = \alpha_{d2} n'_{eq} n'_{bw} \frac{n_d}{n_s} \tag{2.40}$$

- 共役勾配法のパラメータ修正演算

$$T_{cg} = \alpha_{cg} n_{bf} \tag{2.41}$$

ここで、n_d は領域分割数、n_{bf} は領域間境界自由度数、$\alpha_{d1}, \alpha_{d2}, \alpha_{cg}$ は定数である。これらの定数の実測値を Table 2.4 に示す。

解析時間は反復回数を I_t として

$$T = T_{co}(n_s + 1)I_t + T_{d1} + T_{d2}(I_t - 1) + T_{cg} I_t \tag{2.42}$$

となる。また、ネットワークに飽和が生じた時も

$$T = T_{co}(2n_s I_t - n_s + 1) + T_{d1} + T_{cg} I_t \tag{2.43}$$

と表される。

ここで、領域分割数 n_d によって部分領域内の自由度数 n'_{eq}、バンド幅 n'_{bw} および領域間境界自由度 n_{bf} が変化し、反復回数 I_t も変化する。したがって、領域分割数 n_d と並列台数 n_s が並列のパラメータである。

Table 2.4 Coefficients of operation and communication speed (domain decomposition method)

	Double Precision
α_{d1}	2.32544×10^{-6}
α_{d2}	2.70810×10^{-6}
α_{cg}	8.04954×10^{-6}
α_{co}	8.89320×10^{-6}
β	8.34818×10^{-3}

2.3.5 反復回数の予測

並列共役勾配法と領域分割法の解析時間を評価するには、共役勾配法の収束までの反復回数を予測する必要がある。自由度数との関係を求めるため、Fig. 2.11 に示す立方体の一様引張り問題を考え、各辺を N 等分の部分領域に分割、さらに各部分領域は 8 節点アイソパラメトリック要素によって各辺 M 等分の要素に分割し、N と M の組合せを変えて解析した。

このときの並列共役勾配法 ($N = 1$) による全自由度数と反復回数の関係を Fig. 2.12 (a) に、領域分割法 ($N > 1$) による領域間境界自由度数と反復回数の関係を Fig. 2.12 (b) に示す。最小 2 乗近似すると

- 並列共役勾配法

$$I_t = 12.667 n_{eq}^{0.359} \tag{2.44}$$

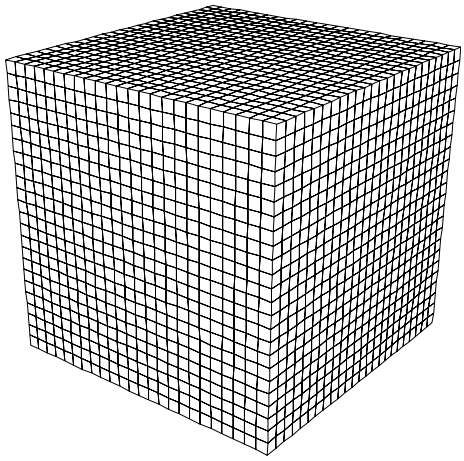
- 領域分割法

$$I_t = 12.648 n_{bf}^{0.293} \tag{2.45}$$

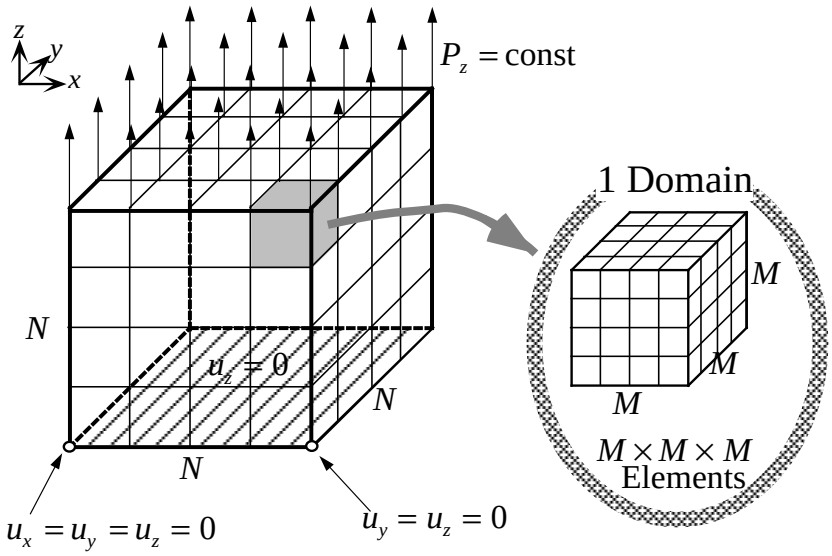
が得られ、これを用いて解析時間の予測を行なう。

2.3.6 評価式の検証

導出した評価式の妥当性を検証するため、並列ガウス消去法、並列共役勾配法および領域分割法のそれぞれに対して、並列台数を変化させ、実際の解析時間と予測時間の比較を行なった。Fig. 2.13 に示すように、いずれの方法においても通常時およびネット



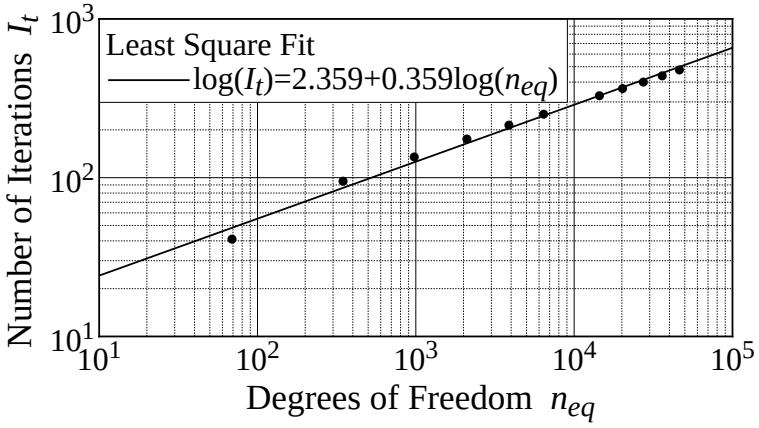
(a) finite element mesh



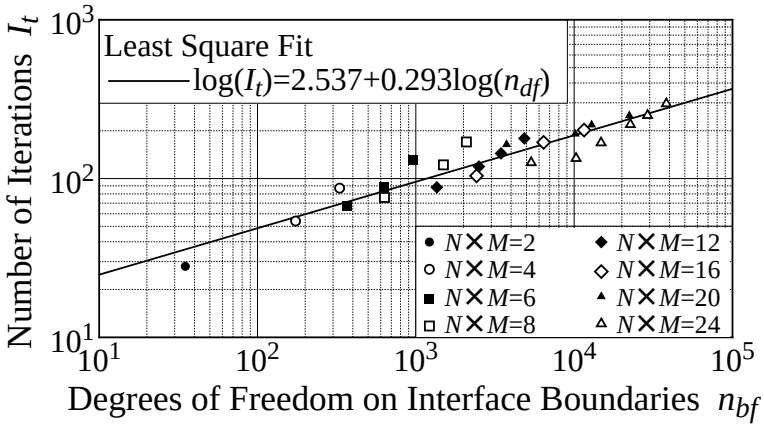
(b) domain decomposition

Fig. 2.11 Three-dimensional model for performance evaluation

ワーク飽和時に対し良い一致が見られる。なお、スレーブ台数が大きい範囲において、実際の解析時間とネットワーク飽和時の評価時間との間に若干の誤差が見られるが、これは、演算速度および通信速度を表す定数の実測を少ないスレーブ台数を設定して行なったためであると考えられる。しかし、本研究においては、導出した評価式より並列解析時間を最小にする最適な並列パラメータを求めるため、ネットワークが通常の状態および飽和状態における 2 つの曲線の交点付近の評価時間が最も重要であることから、この



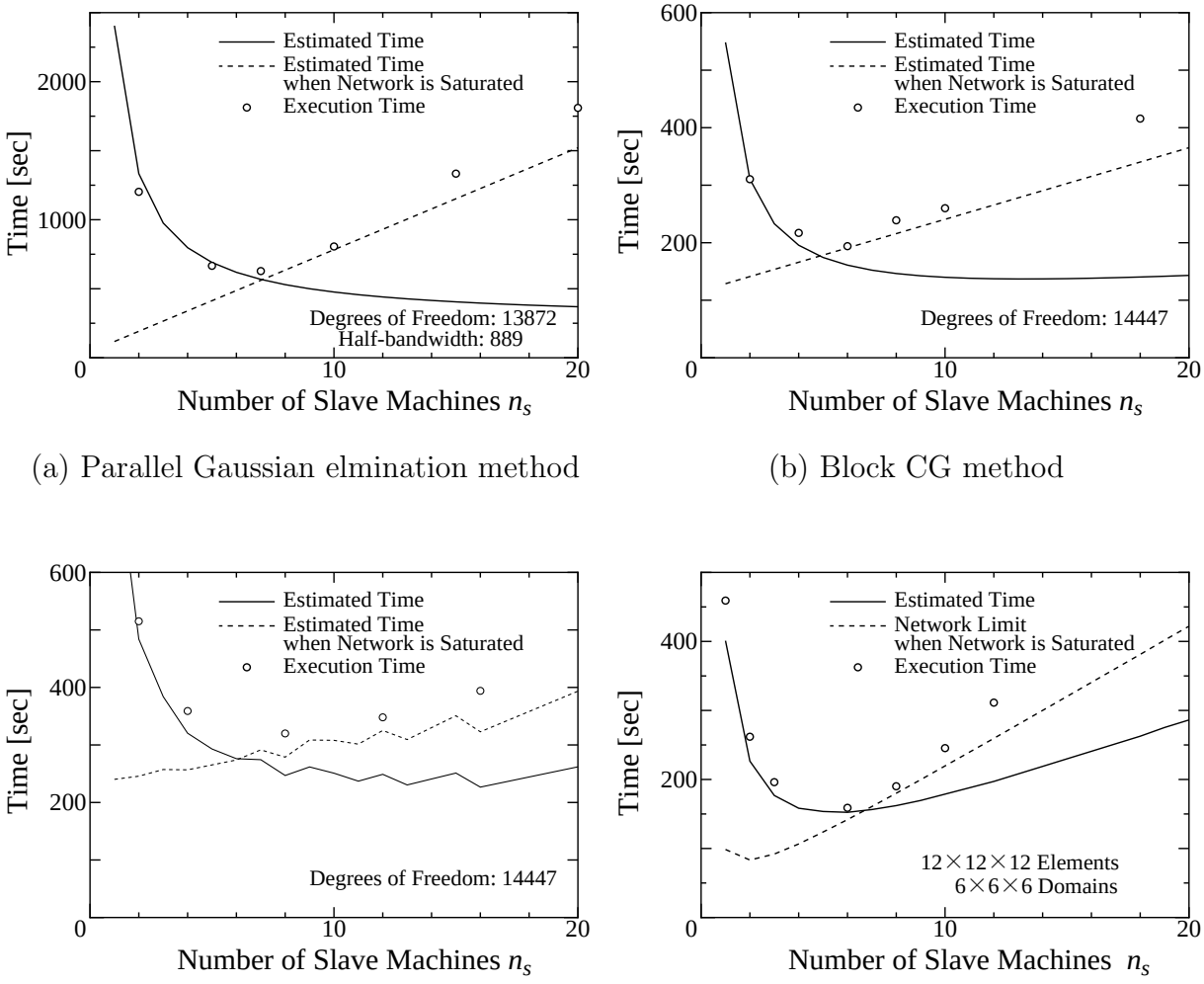
(a) CG method



(b) DDM

Fig. 2.12 Number of iterations for solution convergence

誤差については実用上問題ないといえる。以上のことから、導出した評価式の妥当性が確認され、これらの式を用いて解析時間の評価や並列パラメータの最適化を行なうことができる。



(a) Parallel Gaussian elmination method

(b) Block CG method

(c) Domain CG method

(d) DDM

Fig. 2.13 Comparison between estimated and executed time

2.4 システムの最適化

2.4.1 並列ガウス消去法の最適化

(a) 並列パラメータの最適化

並列ガウス消去法は、ブロック行数と並列台数により解析時間が変化する。ブロック行数が少ないと、ピボットブロックの転送回数が増えて通信時のオーバヘッドが増大するが、逆にブロック行数を多くすると、並列に処理されないブロック内消去時間が増大

する。このため、ブロック行数には最適な値が存在し、これは式 (2.24) において

$$\begin{aligned}\frac{\partial T}{\partial n_{br}} &= -C_1\frac{1}{n_{br}^2} + C_2 + C_3\frac{1}{n_s} + C_5 n_s \\ &= 0 \end{aligned} \tag{2.46}$$

を解くと、

$$n_{br} = \sqrt{\frac{C_1 \cdot n_s}{C_2 \cdot n_s + C_3 + C_5 \cdot n_s^2}} \tag{2.47}$$

となって、解析時間が最小となる最適なブロック行数を得ることができる。こうして得られた n_{br} に対して Fig. 2.13 (a) に相当する図を作成し、並列台数 n_s の最適値を求めることができる。

ブロック行数と並列台数に対しては、Fig. 2.14 に示すような 4 種類の制約が存在する。

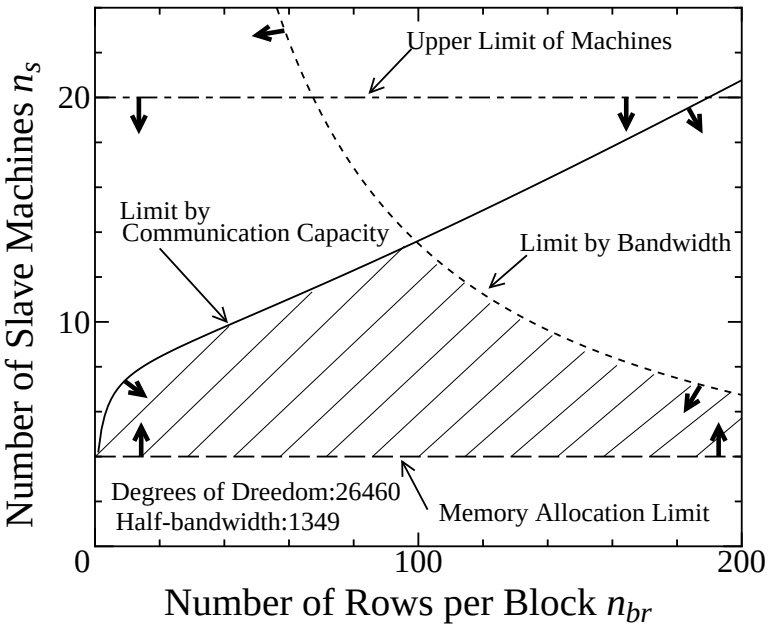


Fig. 2.14 Limitation to the parameters of parallel Gaussian elimination method

1. ネットワークを飽和させないための制約

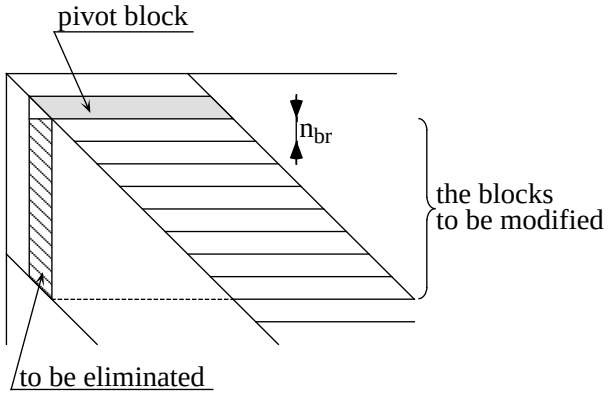
   飽和が起こらない限界は、ブロック行数と並列台数の関係により Fig. 2.14 の右上がりの曲線で表される。

2. バンド幅によるブロック行数への制約

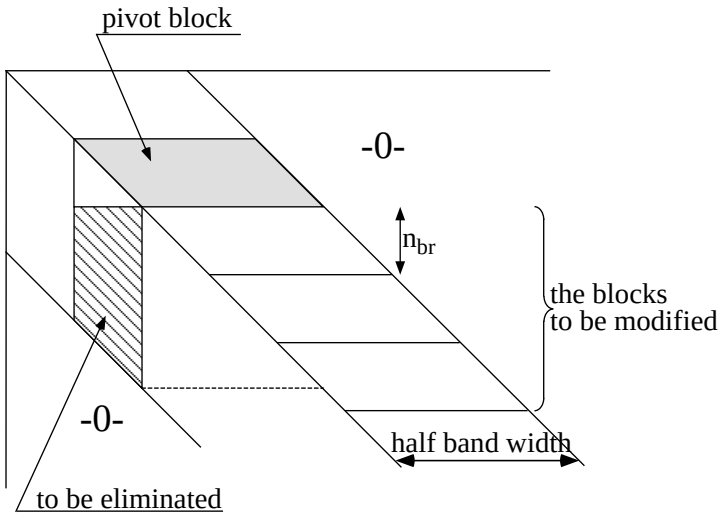
バンド法では半バンド幅を越えて消去を行なう必要がないため、Fig. 2.15 のようにブロックが大きく設定すると、消去の対象となるブロック数が減少し、並列計算を行なえないスレーブ・マシンが出てくる。したがって、ブロック行数にはバンド幅による制約が存在し、これは

$$n_{br} \leq \frac{n_{bw}}{n_s} \tag{2.48}$$

という条件式で表され、Fig. 2.14 の右下がりの曲線となる。



(a) small block size



(b) large block size

Fig. 2.15 Limit by band width

3. 剛性マトリックスを各マシンに分散記憶するのに必要なメモリ容量からくる制約

4. 利用可能なマシン台数の制約

3., 4. は使用するクラスタ環境による制約である。

このため、上記の最適値がこれらの制約を満足しない場合も生じる。したがって、これらの条件を満足しかつ効率よく計算できるように並列パラメータを設定しなければならないが、最初の 3 つの制約に関しては多少越えていたとしても、並列性能は低下するものの、解析の実行は可能である。

(b) サブ・ネットワークを用いた通信容量の拡張

並列パラメータに対する 4 つの制約のうち、現状の技術レベルで最も問題となるのはネットワークの通信容量による制約である。そこで、複数のサブネットから構成されるネットワークを利用することによってデータ通信を高速化し、並列効率の向上をはかる。

Fig. 2.16 に示すように各サブネット上にマスター・マシンを起動し、スレーブ・マシンへのピボット・ブロックの転送を各サブネット内で並列に行うこととする。これにより、通信容量を実質的にサブネット数倍に増大させることができる。

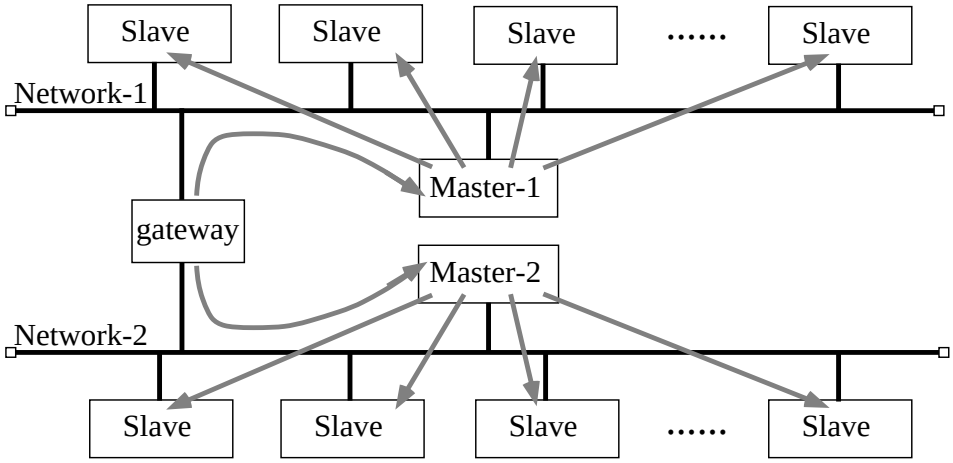


Fig. 2.16 Improvement of data transmission by subnetworking

この方法で実際に解析を行うと、Fig. 2.17 に示すようにネットワークの飽和が起こりにくく、また、飽和が生じた場合にも高い性能を維持することができた。

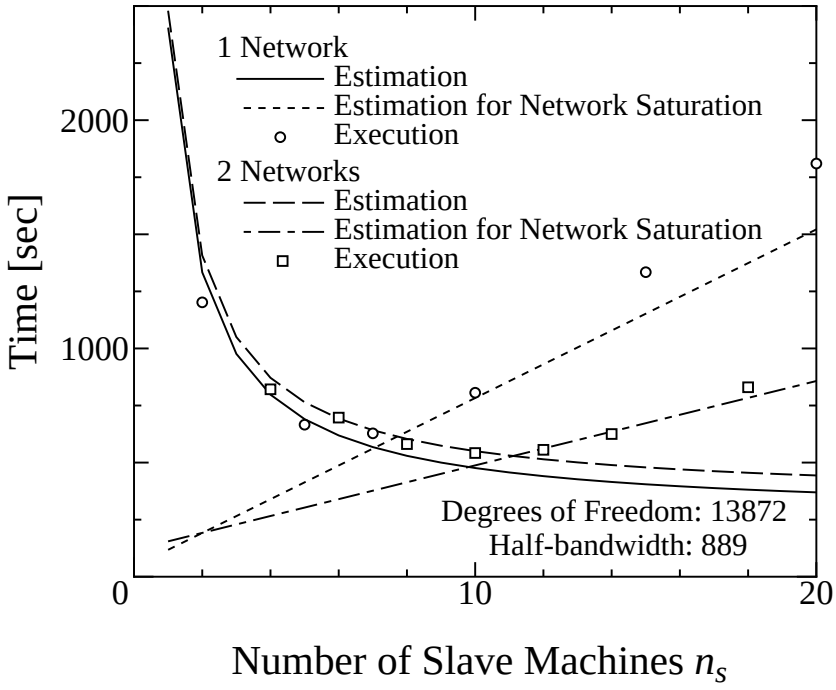


Fig. 2.17 Effect of subnetwork system

2.4.2 並列共役勾配法の最適化

(a) ブロック型共役勾配法

並列共役勾配法の並列パラメータは並列台数のみであるため、これを式 (2.30)、(2.31) および (2.44) を用いて、ネットワークが飽和を起こさず最短時間で解析が行えるように設定する。例えば、Fig. 2.13 (b) に示した問題の場合、並列台数が 5 台のときが最短時間で解析できることがわかる。

(b) 領域型共役勾配法

領域型共役勾配法の場合、解析時間の評価式 (2.36) および (2.37) には、並列パラメータとして並列台数のほか領域分割数が含まれている。しかし本方法の場合には、Fig. 2.11 に示した立方体の一辺を 24 要素に分割した問題 (総自由度数 46,247) を、一辺の領域分割数 N と領域内要素分割数 M の組合せを種々に設定 ($N \times M = 24$) しても、評価式によって得られる解析時間は Fig. 2.18 に示すように極端な設定をしない限り大きな変化はない。これは本方法が部分領域の設定が性能に依存しない方法であり、領域分割数を変化させても演算量、および通信量の変化が小さいためである。

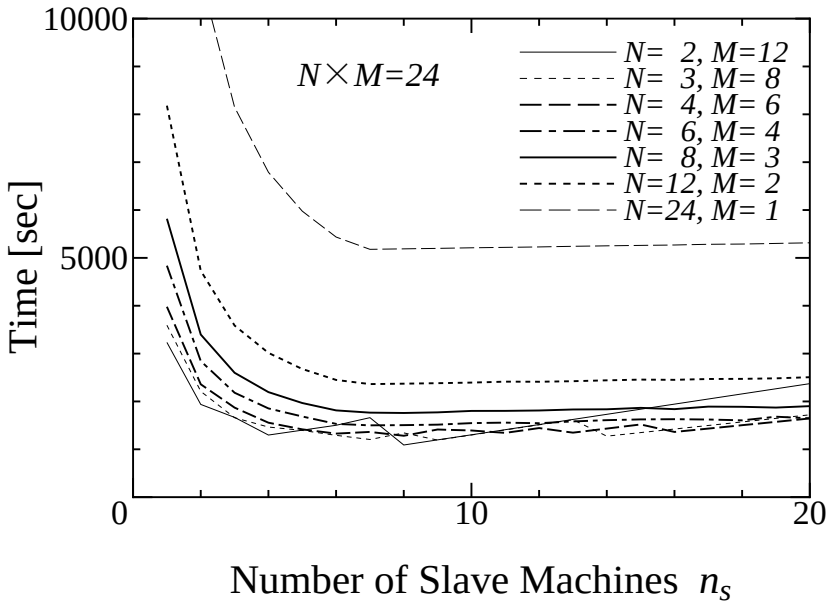


Fig. 2.18 Dependence of computation time (Domain type)

このように領域数の依存性が小さいため、領域数の設定は考慮せずに並列台数を最適な値に設定すれば良い。

2.4.3 領域分割法の最適化

領域分割法は、領域分割数と並列台数によって並列性能が変化する。例えば、Fig. 2.11 の立方体の一辺を 24 要素に分割した問題の場合、一辺の領域分割数 N と領域内要素分割数 M の組み合わせを種々に設定すると $(N \times M = 24)$、並列台数の変化に対して Fig. 2.19 のように式 (2.42) および (2.43) で得られる評価時間が変化する。すなわち、各領域の自由度数が大きいと領域内の解析には時間がかかるが、領域間境界自由度数は小さいため収束は速い。逆に、各領域の自由度数を小さくすると領域内の解析は速くなるが、領域間境界自由度数が大きくなるため収束が遅くなり、同時に通信量も増加する。

このように、N と M の各組合せごとに解析時間が最小となる並列台数が評価式 (2.42), (2.43) を用いて求まる。しかし、並列台数に対してはメモリの分散確保に必要となる最小台数、利用可能な最大台数といった制約も存在する。そこで、最短時間で解析を行うためには、並列パラメータを以下の手順で決定する。

1. N と M の各組合せに対し、評価式 (2.42), (2.43) により解析時間が最短となる台

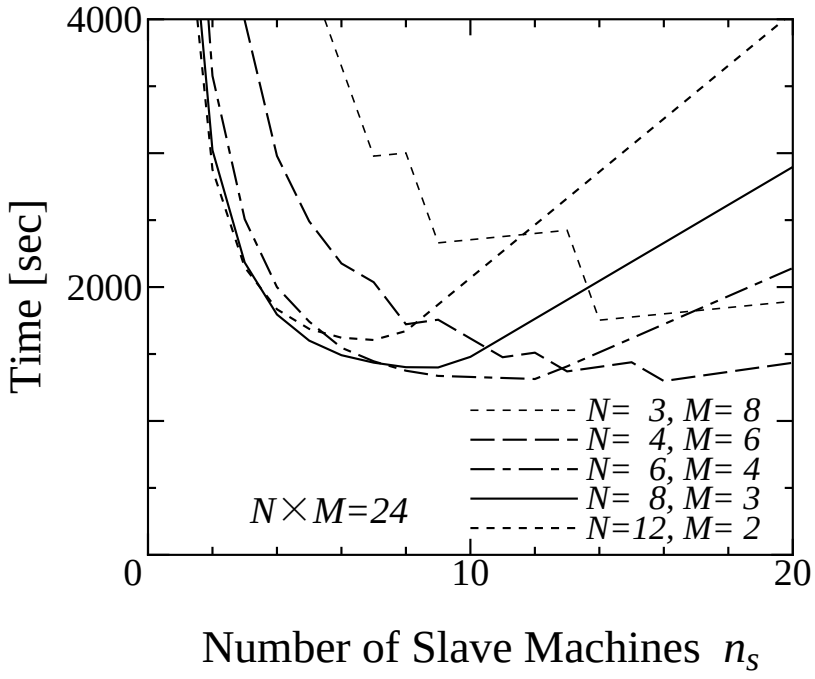


Fig. 2.19 Variation of computation time for different domain decompositions

数とその時間を求める。

2. メモリの分散確保に必要な台数に達していないケースに対しては、確保できるように台数を変更し、その時の解析時間を求める

3. 利用可能台数を越えているケースに対しては、それ以下となるように台数を変更し、その時の解析時間を求める

4. 上記の N と M の組合せの中から、解析時間が最小となるものを選択する

この並列パラメータの最適化の効果を確認するため、上記の立方体 (24×24×24 要素分割) の問題に対し、並列台数は 12 台に固定して領域分割数を変えて解析を行なった。通信や領域内計算などの各処理に要した時間を Fig. 2.20 に示す。Fig. 2.19 で予測されたように 6×6×6 領域に分割した場合が最も高速であり、これよりも領域数が少ないとスレーブ・マシンにおける領域内計算時間が増している。逆に、領域数を多くするとマスター・マシンにおける通信時間が増大し、これにともなってスレーブ・マシンでは待ち時間が増大している。

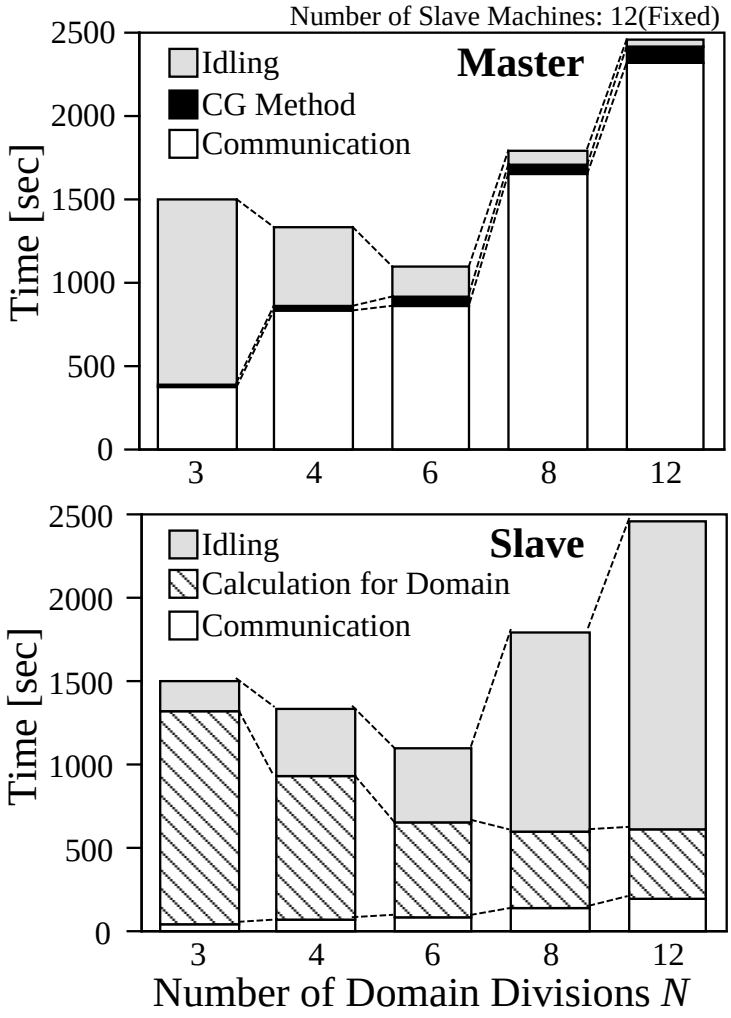


Fig. 2.20 Effect of optimum domain decomposition parameters

このように並列パラメータの最適化の効果が大きいことがわかると同時に、この方法の妥当性も確認される。

2.5 3 つの方法の性能比較

立方体 (24×24×24 要素分割) の問題に対し、評価式を用いて得られた並列パラメータと解析時間の関係を Fig. 2.21 に示す。いずれの方法も、解析時間が最小となる最適な並列パラメータが存在している。並列ガウス消去法は、全体的に時間がかかっているが、これは他の 2 つの方法に比べ演算量、通信量ともに多いためである。しかし、ガウスの消去法は、一般に反復法とは異なり、問題の性質によらず一定の演算の後に解が得られ

るという利点がある。領域分割法は最適な領域分割数に設定すれば最も速いが、場合によっては並列共役勾配法より遅く、極端な場合には並列ガウス消去法より遅い場合さえあるため、注意が必要である。

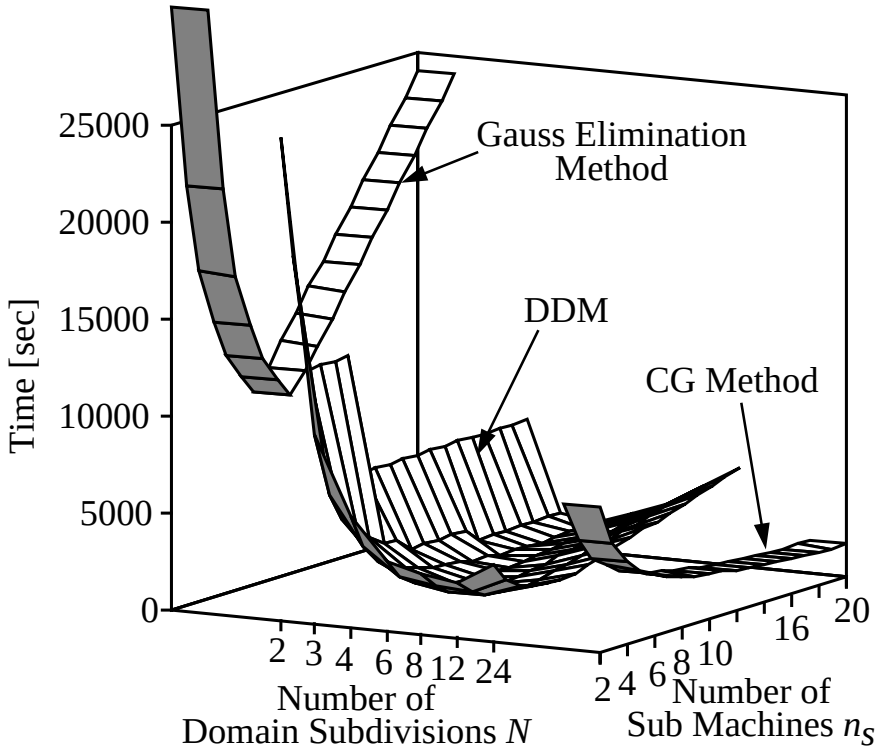


Fig. 2.21 Comparison of estimated parallel performance

このように、領域分割型のほうが高速であることがわかる。並列台数を増すと同程度の解析時間となっているが、これはブロック分割型との通信量の差が縮まるためである。

次に、こうして得られた最適な並列パラメータを用いて、実際に解析を行なった結果を Table 2.5 に示す。ここで、並列ガウス消去法の最適な並列台数が 8 台と他の方法に比べて少ないが、これは通信容量による制約が大きいためである。また、スレーブ・マシン 1 台当りに必要なメモリ量は並列共役勾配法が最も少ないが、これは剛性マトリックスの非零成分のみを記憶していること、および領域分割法の領域内解析にガウスの消去法を用いていることによる。解析時間は、評価式による予測通り領域分割法が最も高速であった。評価式により求めた並列効率は、いずれの方法とも 50%程度であるが、ここで行なった最適化は解析時間を最小にするものであって、並列効率は必ずしも最適化されていないためである。

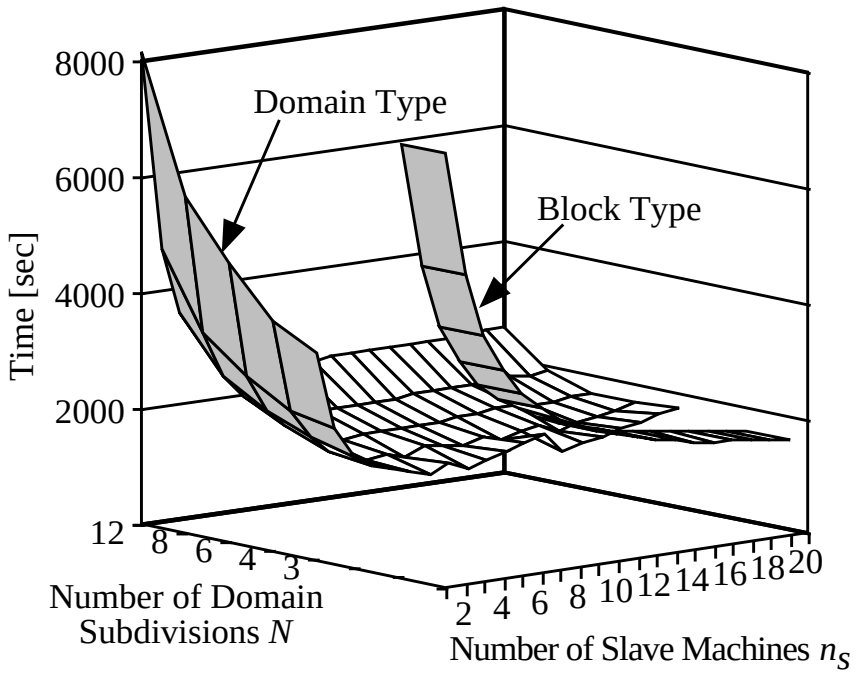


Fig. 2.22 Comparison of parallel performance for two CG method

Table 2.5 Comparison of parallel performance by execution

	Gauss	CG		DDM
		Block	Domain	
Number of Workstations	8	20	10	12
Required Memory [MB]	90.4	1.5	1.5	5.2
Execution Time [min]	292.2	23.9	25.8	18.2
Parallel Efficiency*1 [%]	47.4	46.6	31.3	52.0

*1Estimated

2.6 大規模問題への適用

大規模解析の例として、夜間の計算機利用者のいない時間で行なえる範囲内 (ただし、ワークステーション台数は最大 20 台) の最大規模の解析を行った。問題設定と解析結果を Table 2.6 に示す。

Table 2.6 Results of a large scale problem

	Gauss		CG		DDM
			Block	Domain	
Number of Elements	32×32×32		48×48×48[*1]		
Number of Nodes	35937		117649		
Degrees of Freedom	104544		350544		
Total Matrix Size [MB]	1383[*2]		230	230	1046
Number of Workstations	18		20		
Total RAM Size [MB]	864		960		
Estimated Time [h]	10.8[*3]	6.6[*4]	6.5	5.4	7.0
Execution Time [h]	12.9	6.9	6.8	5.2	10.8
Parallel Efficiency[*5]	39.4	64.5	50.0	14.1	46.9

[*1] 4×4×4 Domains for Domain CG Method, 6×6×6 Domains for DDM Method

[*2] Single Precision

[*3] 1 Network

[*4] 2 Networks

[*5] Estimated

このようにガウスの消去法によっても、並列パラメータなどを最適化することで 10 万元規模の解析が可能であること、並列共役勾配法、領域分割法では 35 万元を越える大規模解析が十分実用的に解析できることを確認した。

2.7 まとめ

ワークステーション・クラスタを用いて、ガウスの消去法、共役勾配法および領域分割法による並列有限要素解析システムを開発し、さらに定量的な性能評価を行なった結果、

1. 最適な並列パラメータの決定法を示した。
2. 3 つの方法とも最適なパラメータを設定することにより、性能が大きく向上することを示した。
3. いずれの方法においても通信容量の制約が厳しいことがわかった。

4. 通信容量の制約を緩和させる 1 つの方法として、サブネットを利用すると大きな効果があることを示した。

5. 35 万元規模の大規模解析が 20 台のワークステーションを用いて十分実用的な時間で行なえることを確認した。

参考文献

(2.1) G. Yagawa, N. Soneda and S. Yoshimura, “A Large Scale Finite Element Analysis using Domain Decomposition Method on a Parallel Computer”, *Comput. and Struct.*, **38**, (1991), *pp.* 615–625.

(2.2) M. Papadrakakis, “Solving Large-Scale Linear Problems in Solid and Structural Mechanics”, In: M. Papadrakakis (ed): Solving large-scale problems in mechanics, John Wiley & Sons Ltd, (1993), *pp.* 1–37.

(2.3) C. Farhat and F. X. Roux, “Implicit Parallel Processing in Structural Mechanics”, *Computational Mechanics Advances*, **2**, (1994), *pp.* 1–124.

(2.4) 例えば、W. R. Stevens, (篠田 訳), “UNIX ネットワークプログラミング”, トッパン, (1992), *pp.* 309–406.

(2.5) R. J. Melosh, S. Utku and M. Salama, “Direct Finite Element Equation Solving Algorithms”, *Compt. Struct.*, **20**, (1985), *pp.* 99–105.

(2.6) D. Zois, “Parallel Processing Techniques for FE Analysis: System Solution”, *Compt. Struct.*, **28**, (1988), *pp.* 261–274.

(2.7) H. Allik, S. Moore, E. O’Neil and E. Tenenbaum, “Finite Element Analysis on the BBN Butterfly Multiprocessor”, *Compt. Struct.*, **27**, (1987), *pp.* 13–21.

(2.8) C. Farhat, “Redesigning the Skyline Solver for Parallel/Vector Supercomputers”, *International Journal of High Speed Computing*, **2**-3, (1990), *pp.* 223–238.

(2.9) J. Qin and D. T. Nguyen, “A New Parallel-Vector Finite Element Analysis Software on Distributed-Memory Computers”, *AIAA-93-1307-CP*, **93**-1307, (1993), *pp.* 98–102.

(2.10) M. A. Baddourah, O. O. Storaasli and S. W. Bostic, “Linear Static Structural and Vibration Analysis on High-Performance Computers”, *Comput. Sys. Eng.*, **4**, (1993), *pp.* 363–371.

(2.11) 例えば、戸川, “共役勾配法”, 教育出版, (1977).

第 3 章

高速ネットワークを用いたクラスタ・システムにおける並列性能の評価

3.1 はじめに

前章で述べたように、ワークステーション・クラスタを用いた並列処理においては、ネットワークの通信容量による制約が大きな問題となる。こうした中、現在広く普及している Ethernet (10 Mbps) に変わり、100 Mbps クラスの Fast Ethernet や ATM スイッチといった高速ネットワークの普及が始まっており、ワークステーション・クラスタを用いた並列処理の性能向上が期待される[3.1]。

そこで、通信性能および演算性能が異なるさまざまなクラスタ・システムを用いて並列性能の評価と比較を行なう。まず、Ethernet 環境と Fast Ethernet 環境のそれぞれにおいて、さまざまなメッセージ・パッシング・ライブラリを用いて通信性能の評価および比較を行なう。次に、Fast Ethernet の高速ネットワークを用いたクラスタ・システムおよび演算性能が異なるクラスタ・システムにおいて、前章で述べた領域分割法、領域型共役勾配法、並列ガウス消去法の並列性能の評価と比較を行なう。

以上のことから、クラスタ・システムの演算性能および通信性能が向上することによって、並列性能と並列パラメータへの依存性が大幅に変化することについて議論する。さらに、高速ネットワークを用いたクラスタ・システムの性能を最大限に引き出すための最適な並列パラメータの設定方法を明らかにする。

3.2 高速ネットワークの通信性能

3.2.1 クラスタ・システムの構成

本章における並列性能の評価と比較には、Table 3.1 に示す、演算性能が異なる 4 種類のクラスタ・システムを用いる。これらのうち、NWS–5000 は Fast Ethernet (100Base-TX, 100 Mbps)、それ以外の SPARCstation 2, SPARCstation 10 および HP 9000 Series 700 Model 735 (以下、HP 9000/735 と略記) は Ethernet (10Base-2, 10Base-T および 10Base-5) により接続されている。

Table 3.1 Computation performance of workstation cluster systems

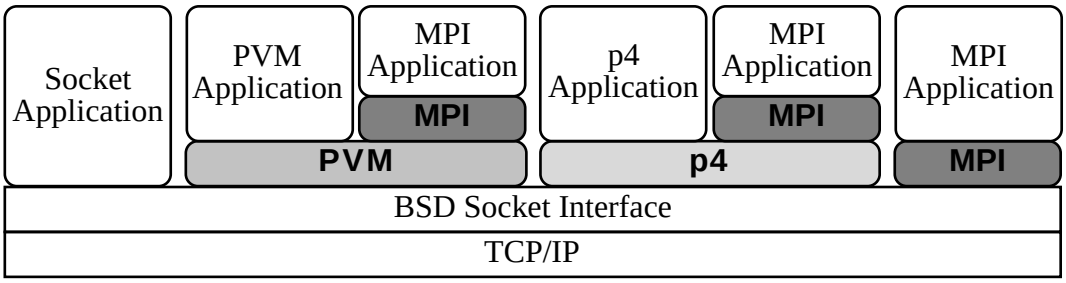
Workstation *(Network)*	SPECfp92	SPECint92
SPARCstation 2 *(Ethernet 10Base-2, 10 Mbps)*	22.8	21.8
SPARCstation 10 *(Ethernet 10Base-T, 10 Mbps)*	60.2	50.2
HP 9000 Series 700 Model 735 *(Ethernet 10Base-5, 10 Mbps)*	150.0	80.0
NWS–5000 *(Fast Ethernet 100Base-TX, 100 Mbps)*	125.0	117.0

3.2.2 メッセージ・パッシング・ライブラリの性能比較

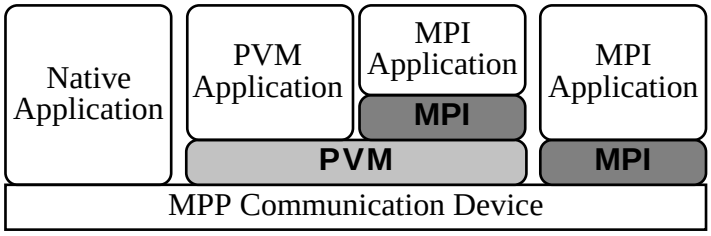
(a) メッセージ・パッシング・ライブラリ

ワークステーション・クラスタや分散メモリ MIMD 型の並列計算機においては、プロセッサ間でデータを交換するためメッセージ・パッシングが用いられる。ワークステーション・クラスタにおいてメッセージ・パッシングを行なう方法としては、Fig. 3.1 (a) に示すように、本研究で用いているソケット・インターフェースを用いるほかに、PVM (Parallel Virtual Machine)[3.2]、p4 (Portable Programs for Parallel Processors)[3.3]、TCGMSG (Theoretical Chemistry Group Message passing system)[3.4] といった、無償で提供されているメッセージ・パッシング・ライブラリを用いる方法もある。これらは、ソケットを用いて開発されているが、ユーザが作成するアプリケーション・プログラムからは、繁雑な UNIX システムコールを直接呼び出すことなく、簡単な関数 (サブルーチン) に

適当な引数を与えて呼び出すだけで通信を行なうことができる。



(a) Message passing interface for workstation cluster



(b) Message passing interface for parallel computer

Fig. 3.1 Message passing library

PVM は、Fig. 3.2 のように、各ワークステーション上に通信やプロセスを管理するためのプロセス (デーモン) を起動し、データの転送はこのデーモンを介して行なわれる。これにより、通信形態として、同期/非同期通信、ブロック/非ブロック通信、1 対 1/1 対多通信といったさまざまな方式を選択することができる。また、PVM はワークステーション・クラスタのみならず、(超) 並列計算機にも専用の通信デバイスを用いて移植されており、多くの機種で利用できる。このため、PVM を用いて作成されたアプリケーション・プログラムは、Fig. 3.1 (b) に示すように、ワークステーション・クラスタや PVM が動作するさまざまな並列計算機において互換性を保つことができる。

こうした並列プログラムの他機種への移植性を向上させるため、メッセージ・パッシングにおけるインターフェース部分の標準化・統一化を目指して、MPI (Message Passing Interface)[3.5]により 130 を越える関数 (サブルーチン) の仕様が定義されている。この規約に基づいて実装されたメッセージ・パッシング・ライブラリとしては、MPICH、LAM などがある。

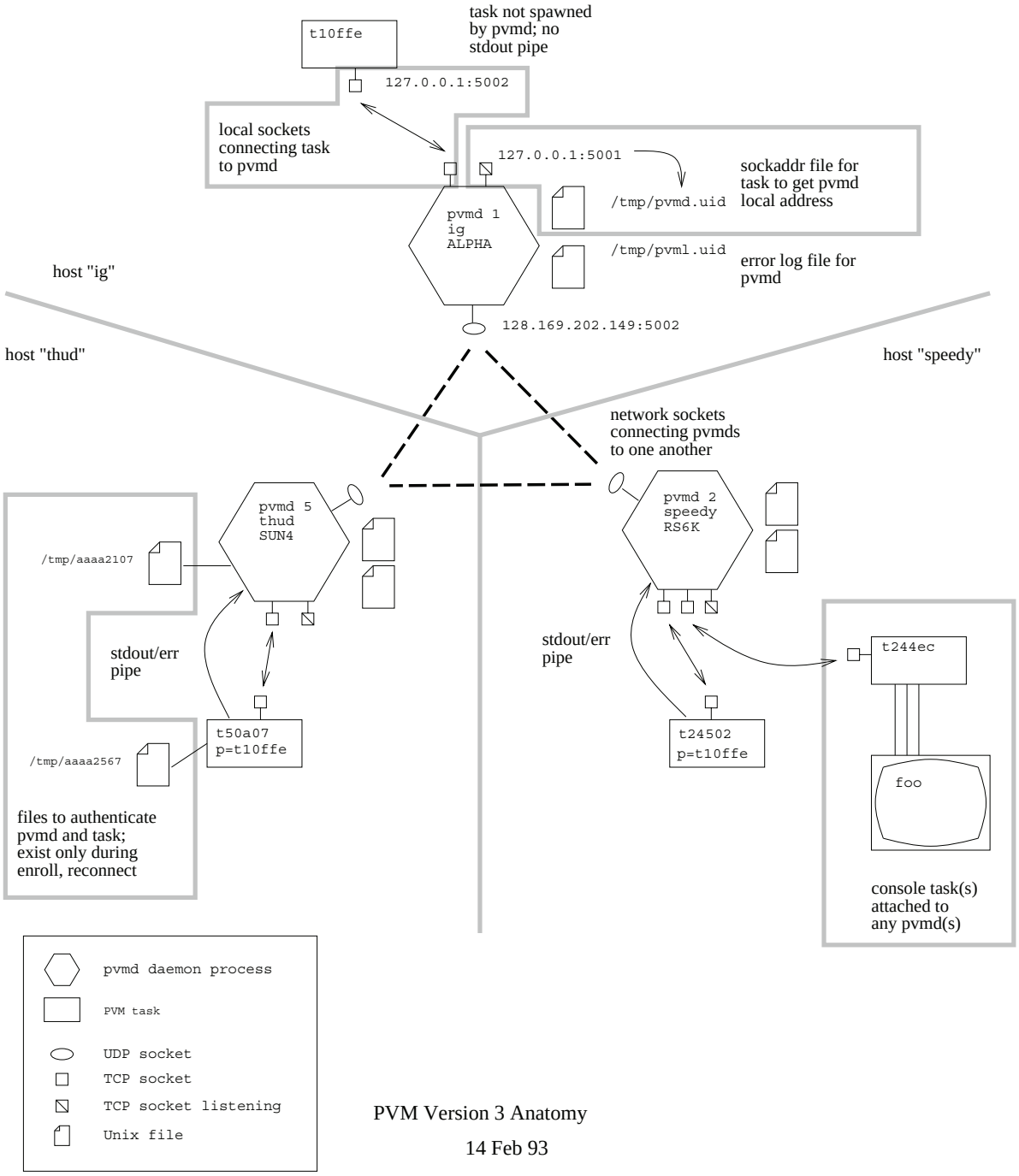
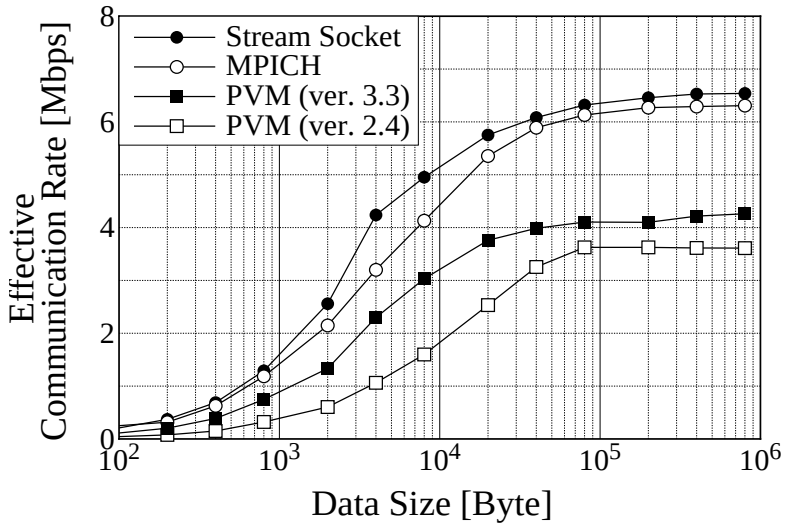


Fig. 3.2 PVM version 3 system overview[3.2]

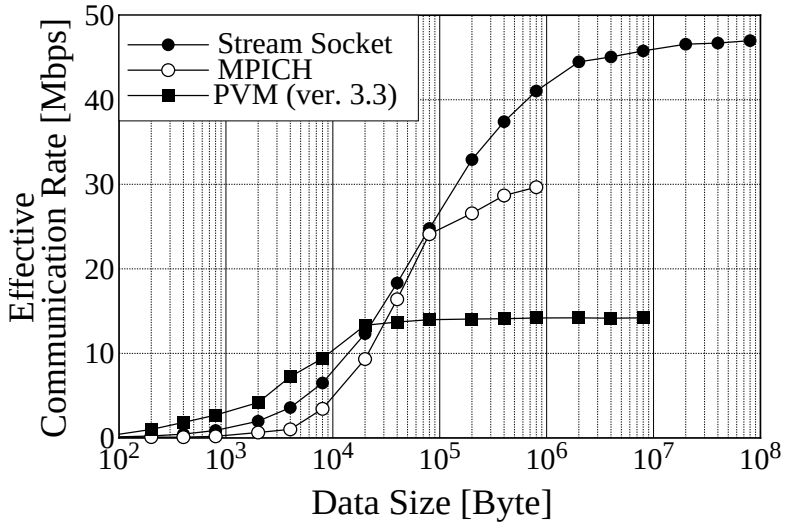
(b) 通信性能の比較

並列効率やスピード・アップといった並列処理性能は、プロセッサ間のデータ転送時間に大きく依存する。すなわち、ワークステーション・クラスタ・システムにおいて同

じ性能のネットワークを用いる場合にも、メッセージ・パッシング・ライブラリの性能により、並列性能が変化する。



(a) Ethernet



(b) Fast Ethernet

Fig. 3.3 Communication performance of message passing libraries

そこで、前節で述べたメッセージ・パッシング・ライブラリのうち、一般に広く用いられている PVM、および、MPI のワークステーション・クラスタへの実装である MPICH の通信性能を、本研究で用いているソケットと比較する。Fig. 3.3 に、Ethernet と Fast Ethernet のそれぞれにおいて、1 度に送受信するデータの大きさを変化させて測定した

実効通信性能を示す。

いずれのネットワークにおいても、ソケット・インターフェースを直接用いた場合に比べ、MPICH、PVM のメッセージ・パッシング・ライブラリを用いると通信性能が低下している。これは、ライブラリの内部で余分なオーバヘッドが付加されているためであると考えられる。特に、PVM の通信性能が低い結果となっているが、これは、通信を行なう際に送信/受信側のそれぞれのデーモン・プロセスを経由しているためであると考えられる。

3.2.3 通信性能の定量化

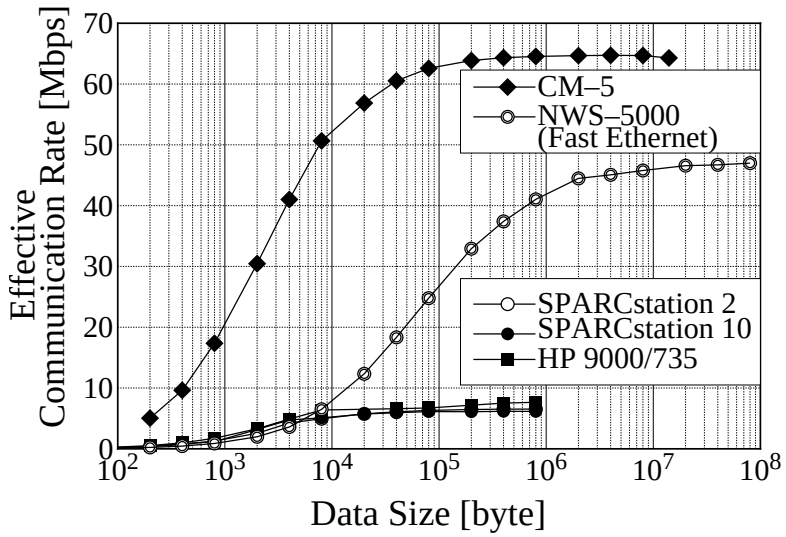
性能が異なる 4 種類のクラスタ・システムにおいて、ソケット・インターフェースを用いる場合の実効通信速度と通信時間を、1 度に送受信するデータサイズを変化させて測定した。この結果を、Fig. 3.4 に示す。いずれのシステムにおいても、通信の初期化などにより、データ転送量が小さいほど、通信性能が低下する。Fast Ethernet は、一度に送受信するデータサイズが小さいと、通常の Ethernet とほぼ同等、あるいはそれ以下の性能しか得られず、データサイズを大きくとることにより、その性能が生かされてくることがわかる。

同図 (b) より、データ転送に要する時間 T_c [sec] を、一度に送受信するデータサイズ n [byte] を用いて

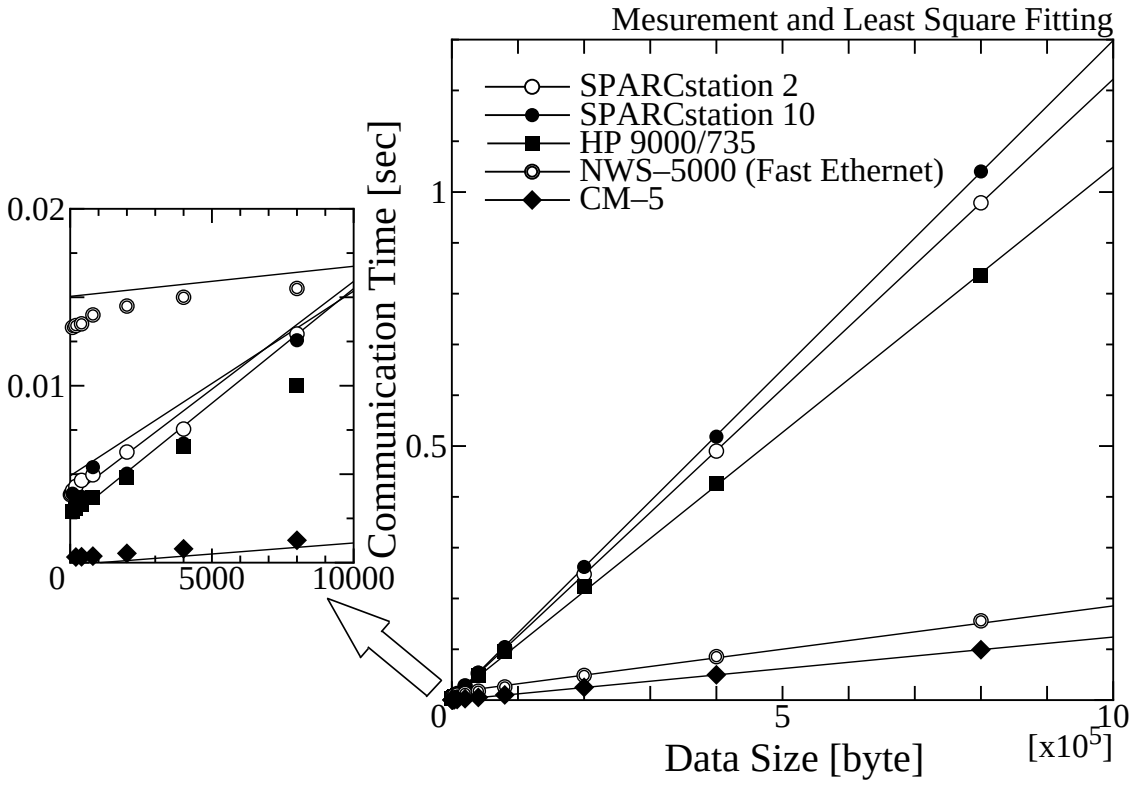
$$T_c = \alpha n + \beta \tag{3.1}$$

のように 1 次関数に最小 2 乗近似することができる。ここで、α は通信速度を表す係数、β は通信の初期化などに要するオーバヘッド時間を表し、これらはネットワークの通信性能により変化する。各システムにおける値を、Table 3.2 に示し、これらの値は、後述の並列解析時間の評価に用いる。

なお、Fig. 3.4 および Table 3.1 には、参考までに、超並列計算機 Connection Machine CM–5 (Fat Tree) の通信性能もプロットした。Fig. 3.4 (a) のように、Fast Ethernet (NWS–5000) の最大通信速度は、超並列計算機 (CM–5) とほぼ同等であるが、データサイズが小さいときは、通信初期化などのオーバヘッドが大きいため、通信性能が低下している。したがって、Fast Ethernet を用いたクラスタ・システムでは、一度に送受信するデータサイズを大きくとることができれば、超並列計算機なみの性能が期待できる。



(a) effective communication rate



(b) communication time

Fig. 3.4 Communication performance of workstation clusters

Table 3.2 Communication performance of workstation cluster systems

Workstation *(Network)*	α [sec/byte]	β [sec]
SPARCstation 2 *(Ethernet 10Base-2, 10 Mbps)*	1.218×10^{-6}	0.0037
SPARCstation 10 *(Ethernet 10Base-T, 10 Mbps)*	1.296×10^{-6}	0.0025
HP 9000/735 *(Ethernet 10Base-5, 10 Mbps)*	1.044×10^{-6}	0.0049
NWS–5000 *(Fast Ethernet 100Base-TX, 100 Mbps)*	1.703×10^{-7}	0.0151
Connection Machine CM–5 *(Fat Tree)*	1.236×10^{-7}	0.0003

3.3 領域分割法

3.3.1 各クラスタ・システムにおける性能の定量化

領域分割法の並列解析時間は、2.3.4 節で述べたように、

$$\begin{aligned} T &= T_{co}(n_s+1)I_t + T_{d1} + T_{d2}(I_t-1) + T_{cg}I_t \\ &= \left\{\alpha_{d1}n'_{eq}n'^2_{bw} + \alpha_{d2}n'_{eq}n'_{bw}(I_t-1)\right\}\frac{n_d}{n_s} + (\alpha_c n_{bf}+\beta)I_t n_s \\ &\quad +(\alpha_c n_{bf}+\beta+\alpha_{cg}n_{bf})I_t \end{aligned} \tag{3.2}$$

と表される。また、ネットワークが飽和した場合には

$$\begin{aligned} T &= T_{co}(2n_sI_t - n_s + 1) + T_{d1} + T_{cg}I_t \\ &= (\alpha_c n_{bf}+\beta)(2I_t-1)n_s + \alpha_{d1}n'_{eq}n'^2_{bw}\frac{n_d}{n_s} + \alpha_c n_{bf} + \beta + \alpha_{cg}n_{bf}I_t \end{aligned} \tag{3.3}$$

となる。

ここで、演算速度を表す定数 $\alpha_{d1}, \alpha_{d2}, \alpha_{cg}$ は、クラスタ・システムの性能により異なる。4 種類のクラスタ・システムにおいて、それぞれ小規模な例題を解析し、これらの定数を実測した結果、Table 3.3 に示す値となった。また、通信速度を表す定数 α_c, β については、前節で述べた通信性能の定量化の結果より、α [sec/bite] を α_c [sec/word] に換算することにより、等価なものとなる。

Table 3.3 Coefficients of computation speed (domain decomposition method)

Workstation	α_{d1}	α_{d2}	α_{cg}
SPARCstation 2	2.32544×10^{-6}	2.70810×10^{-6}	8.04954×10^{-6}
SPARCstation 10	9.69298×10^{-7}	1.05016×10^{-6}	3.51717×10^{-6}
HP 9000/735	2.79842×10^{-7}	3.99980×10^{-7}	2.02626×10^{-6}
NWS–5000	5.02752×10^{-7}	5.58571×10^{-7}	2.84646×10^{-6}

したがって、式 (3.2) および (3.3) に、それぞれのクラスタ・システムにおける各定数を代入することにより、並列解析間を定量化することができる。

3.3.2 領域数が等しい場合の性能比較

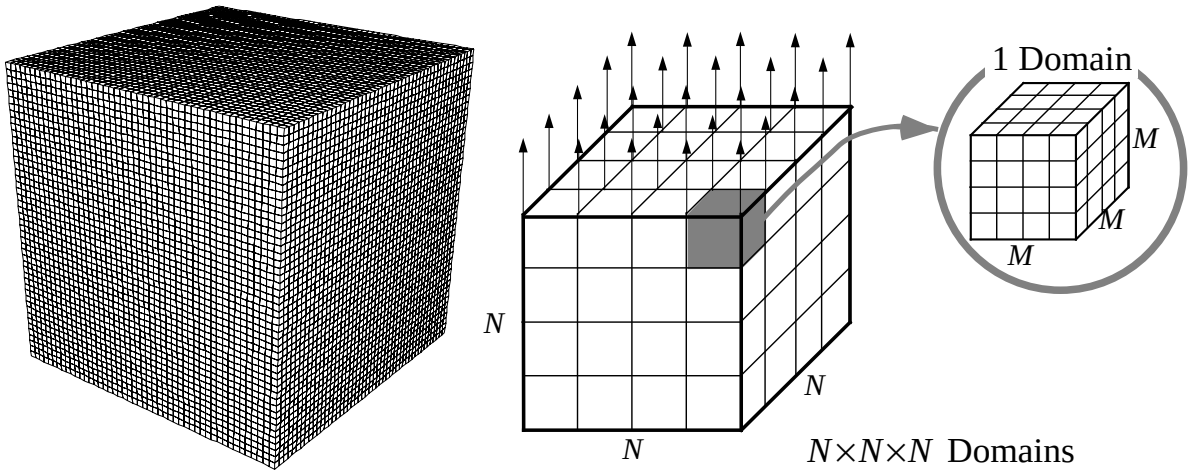
(a) 評価時間の比較

Fig. 3.5 に示す一様引張りを受ける立方体を、8 節点アイソパラメトリック要素により 48×48×48 に分割した問題 (350544 自由度) を設定する。この解析モデルを 6×6×6 領域 ($n_d = 216$) に分割した場合、4 種類のクラスタ・システムにおけるそれぞれのスレーブ台数に対する評価時間の変化を、Fig. 3.6 に示す。Ethernet を用いた 3 種類のシステム (SPARCstation2, SPARCstation 10, HP 9000/735) を比較すると、演算性能が高いシステムほど計算時間は短縮される。しかし、これに伴ってネットワークの通信容量の制約が相対的に厳しくなるため、HP 9000/735 や SPARCstation 10 においては、少ないスレーブ台数でネットワークの飽和が起きている。

一方、Fast Ethernet のシステム (NWS–5000) においては、通信性能が高いため通信時間が短縮されることにより、ネットワークの飽和が起きていない。この結果、スレーブ台数を増しても効率的に並列処理を行なうことができ、解析が高速化されることがわかる。

(b) ネットワークの飽和パラメータの比較

ネットワークの飽和の起こりやすさは、ネットワークの通信性能およびワークステーションの演算性能によって大きく変化する。例えば、同じ演算性能のワークステーションを用いても、ネットワークの通信性能が向上して通信時間が短縮されると、飽和は起



(a) finite element mesh　　(b) decomposition into domains

Fig. 3.5 Finite element model for performance evaluation

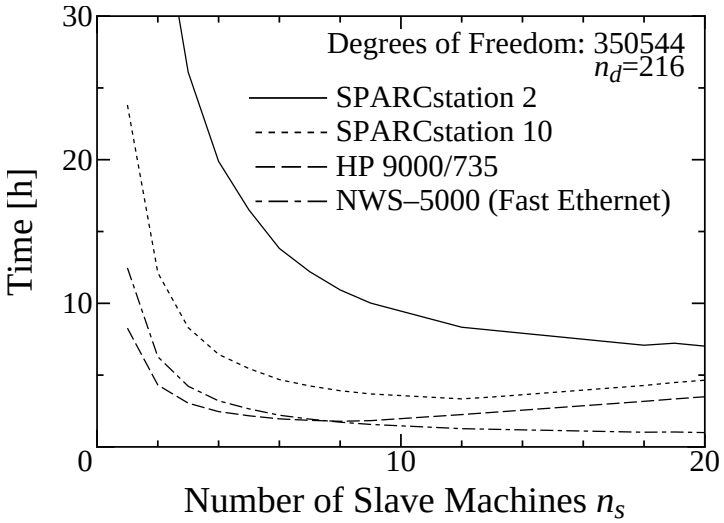


Fig. 3.6 Parallel execution time of domain decomposition method by different workstation clusters

こりにくくなる。逆に、同じ通信性能のネットワークを用いても、ワークステーションの演算性能が向上して演算時間が短縮されると、飽和は起こりやすくなる。

このネットワークが飽和を起こす度合は、スレーブ 1 台当たりの演算時間と、マスターからスレーブへの通信時間との比によって表すことができる。ここでは、この比を

飽和パラメータ S_p として定義する。領域分割法の場合には、領域間境界データの転送時間 T_{co} と、共役勾配法の反復 2 回目以降におけるスレーブ 1 台当たりの演算時間 $T_{d2}\frac{n_d}{n_s}$ の比

$$S_p = \frac{T_{co}(n_s - 1)}{T_{d2}\frac{n_d}{n_s}} \tag{3.4}$$

として表わされ、$S_p > 1$ のときネットワークが飽和状態となる。

スレーブ台数に対する飽和パラメータの変化は、Fig. 3.7 のようになり、曲線の傾きが大きいほどネットワークの飽和が起こりやすいことを示している。また、それぞれの曲線と $S_p = 1$ の直線との交点は、ネットワークの飽和が起こりはじめる臨界点を表し、この点において並列解析時間が最小となる。

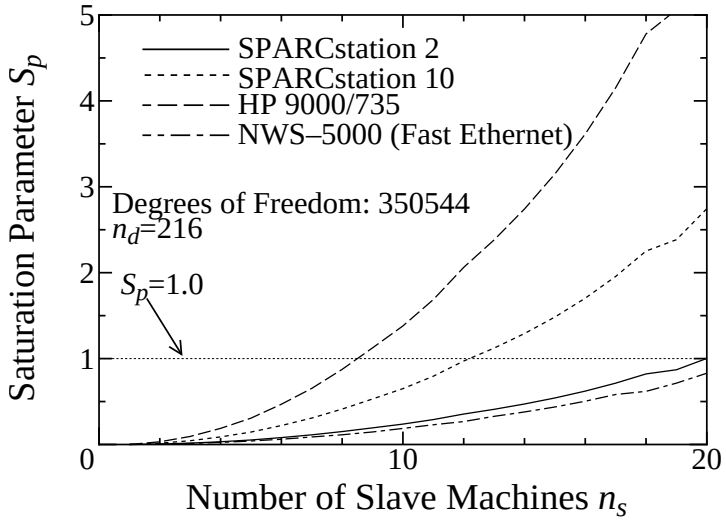


Fig. 3.7 Saturation parameter for different workstation clusters (DDM)

ここで、NWS–5000 (Fast Ethernet) と SPARCstation 2 (Ethernet) の飽和パラメータの変化は、良く似た傾向を示しているが、これは、2 つのシステムの演算性能と通信性能の比率が、ほぼ等しいためである。よって、Ethernet に接続された SPARCstation 2 と、Fast Ethernet に接続された NWS–5000 のシステムの飽和の起こりやすさは、同程度であるといえる。

3.3.3 領域数に対する依存性および最適化

演算性能がほぼ等しい NWS–5000 (Fast Ethernet) と HP 9000/735 (Ethernet) のシステムにおいて、立方体を 48×48×48 要素に分割した問題に対して、領域数を 3×3×3 か

ら 12×12×12 まで変化させたときの解析時間を比較すると、Fig. 3.8 のようになる。ここでスレーブ台数は、HP 9000/735 (Ethernet) においては、それぞれの領域数に対して解析時間が最小となる最適な台数に設定した。また、NWS–5000 (Fast Ethernet) においては、HP 9000/735 (Ethernet) の最適な台数ををそのまま設定した場合 (図中には、NWS–5000 (non-optimized) と表記) と、このシステムにおいて最適な台数をそれぞれ設定した場合 (図中には、NWS–5000 (optimized) と表記) について、比較している。

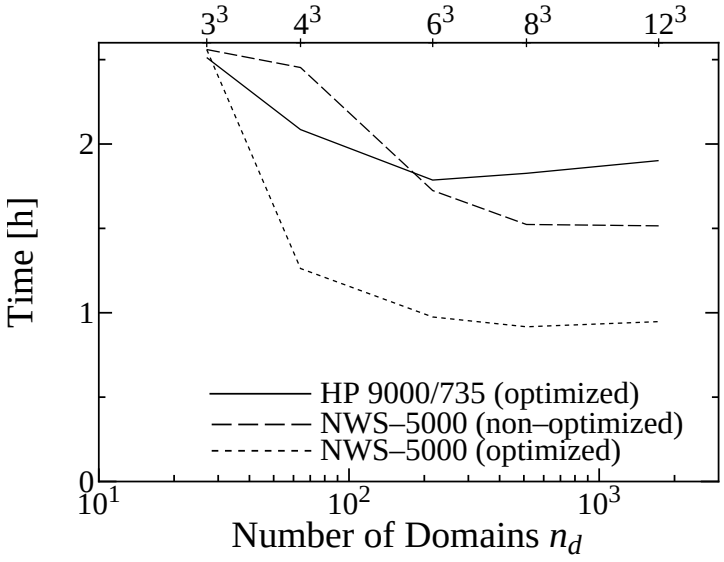


Fig. 3.8 Dependence of parallel execution time on number of domains

ネットワークの性能が高い Fast Ethernet においては飽和が起こりにくいため、スレーブ台数を Ethernet の場合よりも多く設定することができることから、並列処理の効果によって高速化されている。特に、領域数を増してネットワークへの負荷が大きくなると、この差が大きくなる。このように、並列パラメータである領域数とスレーブ台数を、それぞれ Fast Ethernet の環境に適した値に設定することにより、高速化されることがわかる。

最適な領域数は、Ethernet においては 6×6×6 であるのに対し、Fast Ethernet の場合には、8×8×8 と大きな値となっている。これは、領域数を増すことにより領域ごとの解析時間が短縮されるとともに、共役勾配法が適用される領域間境界自由度が増大し、通信 1 回当たりのデータサイズが大きくなることによって、Fast Ethernet の通信性能が Fig. 3.9 のように向上しているためである。

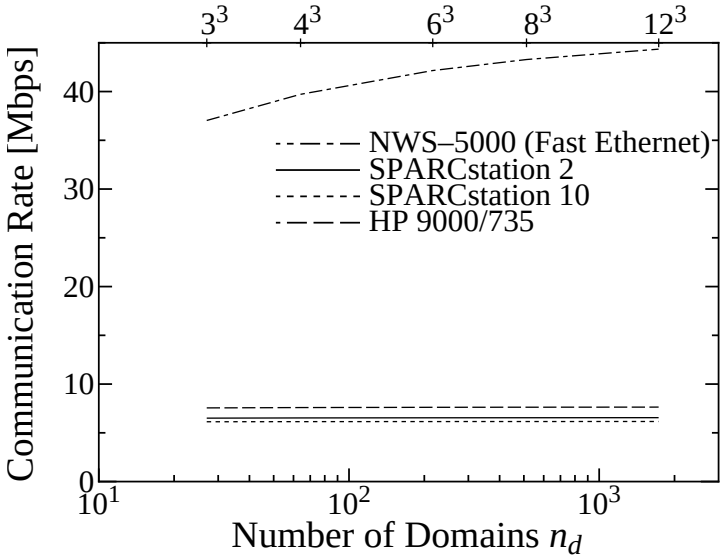


Fig. 3.9 Relation between communication performance and number of domains

3.3.4 実際の解析における並列性能の比較

立方体を 48×48×48 要素に分割した大規模問題を、SPARCstation 2 (Ethernet) と NWS–5000 (Fast Ethernet) の 2 つのシステム上で実際に解析を行ったところ、Table 3.4 の結果が得られた。並列パラメータはそれぞれのシステムにおいて、解析時間が最小となる値に設定している。NWS–5000 は SPARCstation 2 に比べ演算性能が高いため解析時間が短縮され、さらに、Fast Ethernet の効果によって、ネットワークが飽和を起こさず、高い並列効率を得ることができた。

Table 3.4 Execution time and parallel efficiency (DDM)

	SPARCstation 2 (Ethernet)	NWS–5000 (Fast Ethernet)
Number of Slave Machines	20	10
Number of Domains	216	512
Time [h]	10.8	1.2
Parallel Efficiency [%]	46.9	75.7

3.4 領域型共役勾配法

3.4.1 各クラスタ・システムにおける性能の定量化

領域型共役勾配法の並列解析時間は、2.3.3(b) 節で述べたように、

$$\begin{aligned} T &= (T_d + T_p + T_g n_s + T_m)\, I_t \\ &= (\alpha_d + \alpha_c)\, I_t n'_{eq} \frac{n_d}{n_s} + \beta I_t n_s + \alpha_c n'_{eq} I_t n_d + (\beta + \alpha_m n_{eq})\, I_t \end{aligned} \tag{3.5}$$

と表される。また、ネットワークが飽和した場合には、

$$\begin{aligned} T &= (T_p + T_g)\, n_s I_t + T_m I_t \\ &= 2\beta I_t n_s + 2\alpha_c n'_{eq} I_t n_d + \alpha_m n_{eq} I_t \end{aligned} \tag{3.6}$$

となる。

ここで、演算速度を表す定数 α_d, α_m を 4 種類のクラスタ・システムにおいてそれぞれ実測した結果、Table 3.5 に示す値となった。したがって、領域分割法の場合と同様に、式 (3.5)、(3.6) にそれぞれのクラスタ・システムにおける定数を代入することにより、解析時間を評価することができる。

Table 3.5 Coefficients of computation speed (Domain CG method)

Workstation	α_d	α_m
SPARCstation 2	7.024×10^{-5}	8.486×10^{-6}
SPARCstation 10	3.055×10^{-5}	3.095×10^{-6}
HP 9000/735	1.258×10^{-5}	2.094×10^{-6}
NWS–5000	1.694×10^{-5}	1.774×10^{-6}

3.4.2 評価時間の比較

領域分割法の場合と同じ、立方体を 48×48×48 要素に分割した問題に対する評価時間を、Fig. 3.10 に示す。Fast Ethernet を用いると、通信の高速化によりネットワークの飽和が生じにくいため、演算性能がほぼ等しい HP 9000/735 (Ethernet) に比べて、解析時間が短縮されている。

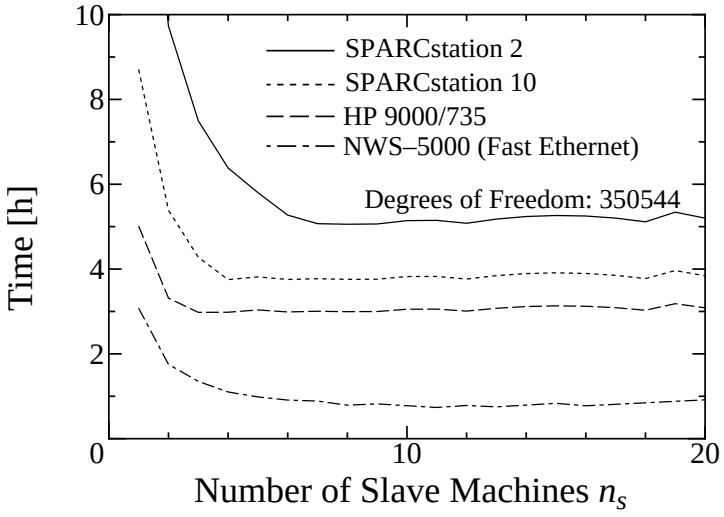


Fig. 3.10 Parallel execution time of domain partitioned CG method by different workstation clusters

3.4.3 飽和パラメータの比較

領域型共役勾配法の飽和パラメータは、領域分割法の場合と同様の定義に基づき、探索方向ベクトルの転送時間とスレーブ 1 台当たりの演算時間との比

$$S_p = \frac{T_p(n_s - 1)}{T_d \dfrac{n_d}{n_s}} \tag{3.7}$$

として表される。スレーブ台数に対する飽和パラメータの変化を Fig. 3.11 に示す。通信量が多いため領域分割法に比べて全般的に飽和しやすい傾向にあるが、この場合も Fast Ethernet における飽和パラメータの傾きが最も小さく、ネットワークの飽和が起こりにくいことがわかる。

3.4.4 実際の解析における並列性能の比較

この大規模問題を、SPARCstation 2 (Ethernet) と NWS–5000 (Fast Ethernet) の 2 つのシステムにおいて、実際に解析した結果を Table 3.6 に示す。NWS–5000 のシステムにおいては、領域分割法の場合と同様に、演算性能と通信性能の向上により高速に解析することができた。なお、並列効率が低い値となっているのは、通信量が多いことと、解析時間が最小となるように並列パラメータを設定しているためである。

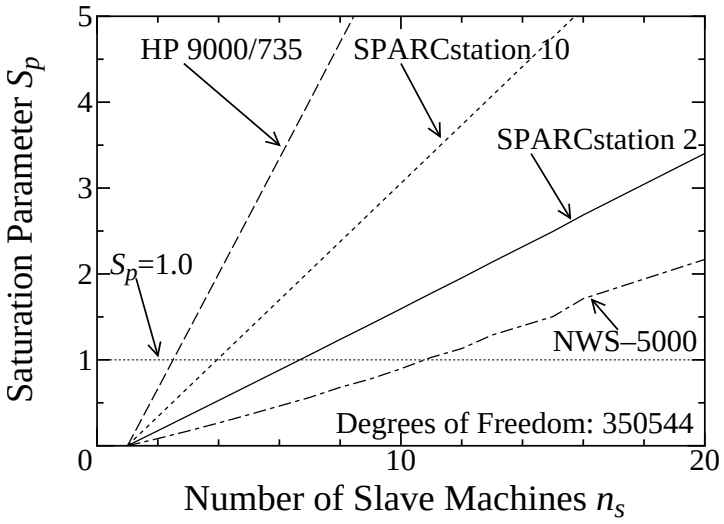


Fig. 3.11 Saturation parameter for different workstation clusters (Domain CG)

Table 3.6 Execution time and parallel efficiency (Domain CG)

	SPARCstation 2 (Ethernet)	NWS–5000 (Fast Ethernet)
Number of Slave Machines	20	10
Number of Domains	64	64
Time [h]	5.2	1.0
Parallel Efficiency [%]	14.1	32.9

3.5 並列ガウス消去法

3.5.1 各クラスタ・システムにおける性能の定量化

並列ガウス消去法の解析時間は、2.3.2 節で述べたように、

$$
\begin{aligned}
T &= (2T_c + T_f + T_n)\frac{n_{eq}}{n_{br}} + T_b \\
&= 4\beta\frac{n_{eq}}{n_b r} + (\alpha_f + \alpha_n)\, n_{bw} n_{eq} n_{br} + \left(\alpha_c + \alpha_f n_{bw}\frac{n_{eq}}{n_s}\right) n_{br} + \alpha_f\,(n_{bw}-1)\, n_{bw}\frac{n_{eq}}{n_s} \\
&\quad + \alpha_c\,(n_{eq} - n_{br})\, n_s + \{2\alpha_c\,(n_{bw}+1) + \alpha_b n_b w\}\, n_{eq} - \beta
\end{aligned}
\tag{3.8}
$$

と表される。また、ネットワークが飽和した場合には、

$$T = T_c \frac{n_{eq}}{n_{br}} n_s + T_b$$
$$= \alpha_c n_{eq}(n_{bw}+2)n_s + \alpha_c(1-n_s)n_{br} + \beta(n_s+2)\frac{n_{eq}}{n_{br}} + \alpha_b n_{bw} n_{eq} - \beta \qquad (3.9)$$

となる。

Table 3.7 Coefficients of computation speed (parallel Gaussian elimination method)

Workstation	α_f	α_n	α_b
SPARCstation 2	2.34420×10^{-7}	1.83096×10^{-7}	5.04579×10^{-7}
SPARCstation 10	7.21425×10^{-8}	9.05435×10^{-8}	3.83333×10^{-7}
HP 9000/735	2.71279×10^{-8}	4.72637×10^{-8}	1.73632×10^{-7}
NWS–5000	3.35564×10^{-8}	4.14677×10^{-8}	2.63308×10^{-7}

3.5.2 評価時間の比較

立方体を 16×16×16 要素に分割した問題 (13872 自由度) に対する評価時間を、Fig. 3.12 に示す。ここで、ブロック行数は、それぞれのシステムにおいて最適な値を設定した。NWS–5000 (Fast Ethernet) の場合には、ネットワークの飽和が起こりにくいため、演算性能が同等の HP 9000/735 (Ethernet) よりも高速に解析を行うことができる。

3.5.3 ブロック行数に対する依存性および最適化

スレーブ台数を 5 台に固定した時の、ブロック行数に対する評価時間の変化を、Fig. 3.13 に示す。ここで、HP 9000/735 (Ethernet) のシステムにおいては、スレーブ台数 5 台においてネットワーク飽和状態にあるため、ほかのシステムと依存性が異なっている。

Fast Ethernet における最適なブロック行数は、Ethernet の場合に比べて大きな値となっている。これは、ブロック行数によって通信 1 回当たりの通信量が変化するため、Fig. 3.14 に示すように、ブロック行数を大きく設定することにより、Fast Ethernet の通信性能を引き出すことができる。したがって、Fast Ethernet を利用する場合には、ブロック行数を Ethernet の場合よりも大きく設定し、高い通信性能が得られるようにしなければならない。

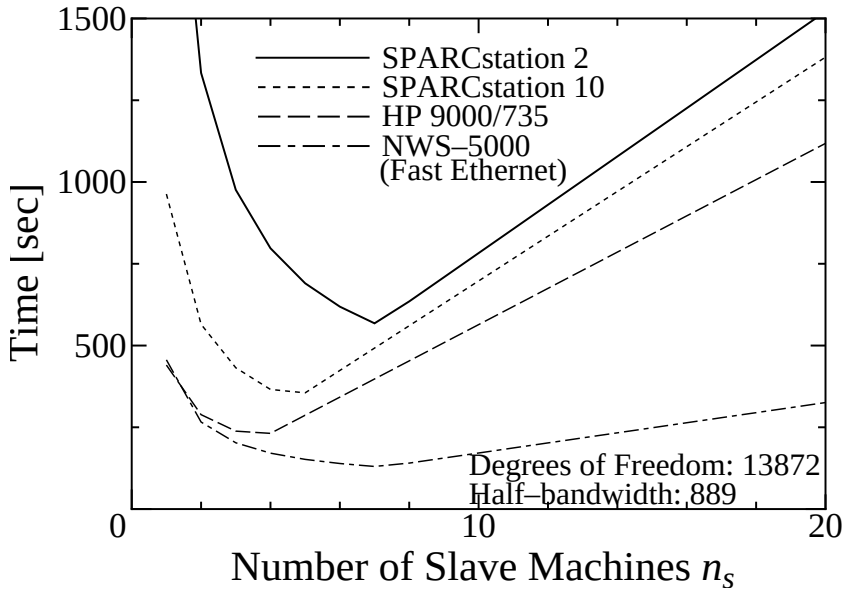


Fig. 3.12 Parallel execution time of parallel Gaussian elimination method by different workstation clusters

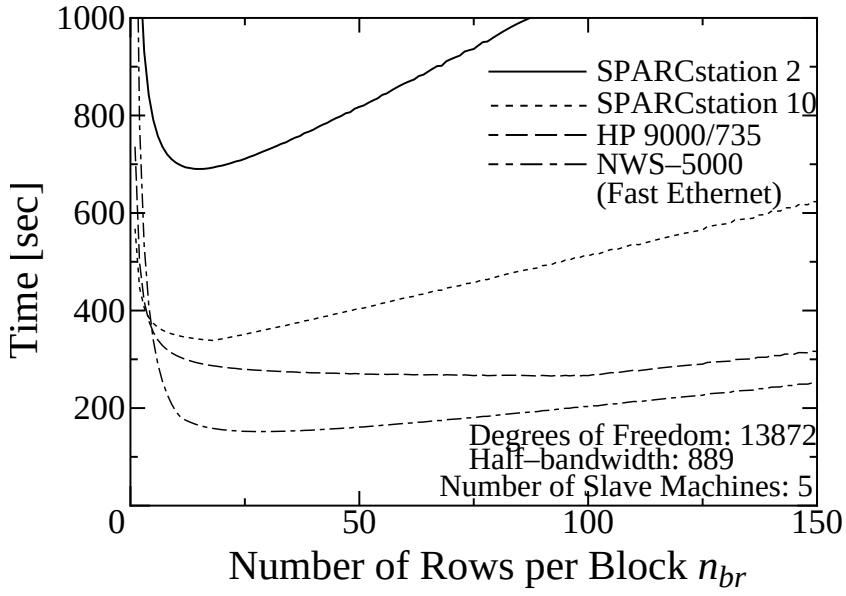


Fig. 3.13 Dependence of parallel execution time on block size of parallel Gaussian elimination method

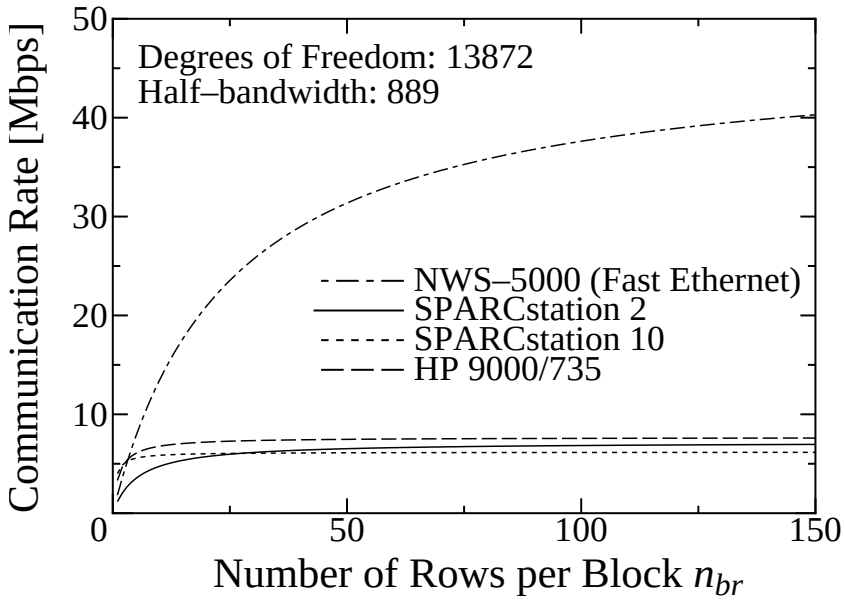


Fig. 3.14 Relation between communication performance and block size

3.5.4 並列パラメータへの制約の比較

ブロック行数とスレーブ台数の設定に対して、図 3.15 に示すような制約が存在する。まず、第一に、ネットワークを飽和させないための通信容量による制約であり、これによってスレーブ台数の最大値が設定される。第二は、バンドマトリックスに対する前進消去が、全てのスレーブマシンにおいて並列に行えるための、バンド幅により決定されるブロック行数の最大値であり、これはクラスタ・システムの性能に依存しない制約である。これらの制約を満たすようにブロック行数とスレーブ台数を設定しなければならないが、Fast Ethernet を用いたシステムにおいては、通信容量の制約が大幅に緩和されるため、スレーブ台数を大きく設定することができ、解析を高速に行うことができる。

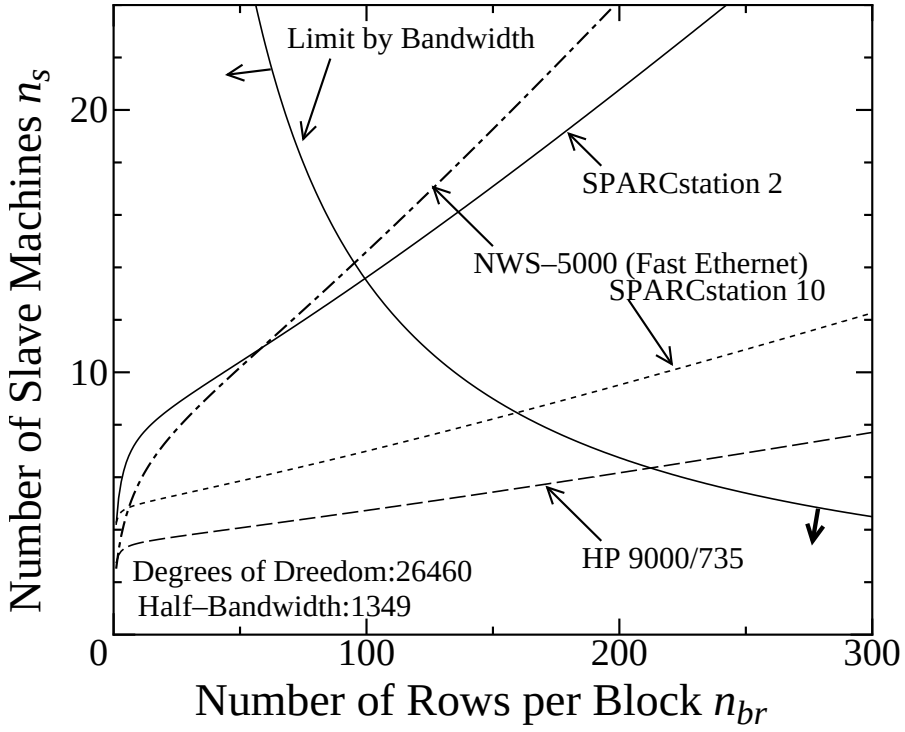


Fig. 3.15 Limitation to the parallel parameters of parallel Gaussian elimination method

3.5.5 実際の解析における並列性能の比較

立方体を 32×32×32 要素に分割した問題 (104544 自由度) を、SPARCstation 2 (Ethernet) と NWS–5000 (Fast Ethernet) の 2 つのシステムにおいてそれぞれ解析を行ったところ、Table 3.8 に示す高い並列効率が得られた。

Table 3.8 Execution time and parallel efficiency (parallel Gaussian elimination method)

	SPARCstation 2 (Ethernet)	NWS–5000 (Fast Ethernet)
Number of Slave Machines	18	10
Time [h]	12.9	1.8
Parallel Efficiency [%]	39.4	61.3

3.6 まとめ

通信性能が高い Fast Ethernet を用いたワークステーション・クラスタ・システムにおいて、領域分割法、領域型共役勾配法および並列ガウス消去法の 3 種類の並列有限要素解析手法について、並列性能の評価および比較を行った。その結果、以下の結論を得た。

1. 高速ネットワークの利用は、ワークステーション・クラスタによる並列有限要素解析において、並列処理が効率的に行なえるようになることから、解析の高速化を図る上で極めて有効である。

2. 高速ネットワーク環境の性能を十分に引き出すには、通信 1 回あたりのデータ転送量を大きくとることにより通信時のオーバヘッドを小さくする、並列パラメータの最適な設定が重要である。この結果として、領域分割法においては、Ethernet 環境の場合よりも領域分割数は小さ目、また、並列ガウス消去法においては、ブロックサイズは大き目に設定されることになる。

3. ネットワークの通信容量の制約を定量的に表す飽和パラメータを定義するとともに、これを用いて Ethernet 環境と比較した結果、高速ネットワークの利用が通信の制約を大きく緩和させることを確認した。

4. 高速ネットワークの導入により、35 万元規模の解析を、10 台のワークステーションを用いて、短時間に行えることを確認した。

参考文献

(3.1) 松田, 冨山, 松田, “大規模熱流動解析への分散 Network Computing の応用に関する研究 (1) ATM と Ether の比較”, 日本機械学会通常総会講演会講演論文集 (I), No. 96-1, (1996), *pp.* 195–196.

(3.2) A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, “PVM3 User’s Guide and Reference Manual”, Reserch Report ORNL/TM-12187, Oark Ridge National Laboratory, (1993).

(3.3) R. Butler and E. Lusk, “User’s Guide to the p4 Programing System”, Technical Report ANL-92/17, Argonne National Laboratory, (1992).

(3.4) R. J. Harrison, “TCGMSG Send/receive subroutines —version 4.03 user’s manual”, Battelle Pacific Northwest Laboratory, (1993).

(3.5) W. Gropp, E. Lusk, “A Test Implementation of the MPI Draft Message-Passing Standard”, Technical Report ANL-92/47, Argonne National Laboratory, (1992).

第 4 章

電磁構造連成問題の並列有限要素解析

4.1 はじめに

核融合炉や磁気浮上列車などの強磁場中におかれた機器の設計においては、磁場と渦電流による電磁力および磁場と構造物の変形による速度起電力の相互作用を考慮した、電磁構造連成解析が必要となる。この連成現象は、一般に、速度起電力によって発生する渦電流が構造物の変形を抑えるような電磁力として作用することから、磁気減衰効果とも呼ばれている[4.1–4.3]。したがって、こうした機器の設計においては、連成現象を考慮することが大変重要となる。

このような電磁場と構造体の変形が連成する複合現象の解析は、互いに影響を及ぼし合う渦電流と変位を同時に解析しなければならないため、多自由度の大次元問題となる。さらに、渦電流解析における係数マトリックスがフル・マトリックスとなることから、多大な解析時間が必要となる。

こうした複雑系の問題においては、並列解析法に関する研究はほかに例がなく、また、領域分割法をこの問題の解析にそのまま適用しても並列化を行なうことが不可能である。したがって、Fig. 4.1 に示すような核融合装置真空容器[4.4]のような複雑で超大規模な連成解析を行なうのは困難な状況にある。

そこで、電磁構造連成解析に適した並列化方法として、領域分割法と領域型共役勾配法を組み合わせた新しい方法を提案する。本章においては、まず連成解析の方法を示して並列化を行なう際の問題点を摘出し、新しく提案する並列解析法を示す。さらに、平板のたわみ連成問題において並列性能の定量的な評価を行ない、本並列解析法の有効性を検証する。また、本方法の実用性を確認するため、核融合装置真空容器に生じる電磁

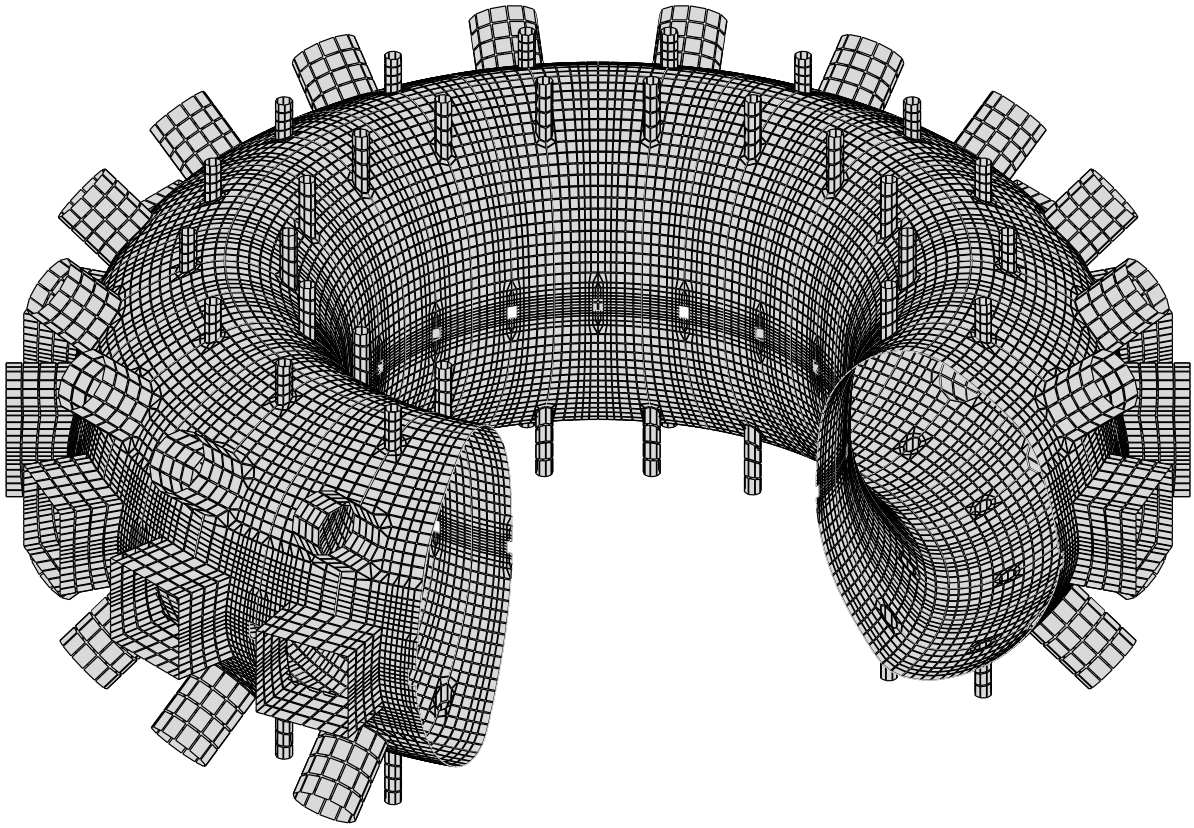


Fig. 4.1 Vacuum vessel of magnetic fusion device JT–60U[4.4]

構造連成現象を実機モデル (1/36 セクタモデル) を用いた大規模並列解析を行ない、並列性能を確認するとともに、大規模解析への有効性を検証する。

4.2 電磁構造連成解析方法

4.2.1 電磁構造連成問題 (磁気減衰効果)

Fig. 4.2 に示すように、定常磁場 $\boldsymbol{B}$ と平行に一端を固定した導体平板を置く。この平板と垂直な方向に時間的に変化する磁場 $\dot{\boldsymbol{B}}^{ex}$ を印加すると、平板に渦電流 $\boldsymbol{J}$ が発生し、Fig. 4.3 のように、定常磁場 $\boldsymbol{B}$ によって $\boldsymbol{J} \times \boldsymbol{B}$ 電磁力が発生する。電磁力により平板が変形すると、この時の変形速度 $\dot{\boldsymbol{u}}$ と定常磁場 $\boldsymbol{B}$ によって $\dot{\boldsymbol{u}} \times \boldsymbol{B}$ 速度起電力が発生する。この速度起電力は、Fig. 4.3 のように発生している渦電流を減少させる方向に電流が流れるため、平板の変形を抑える減衰力として作用する。

この現象は、一般に構造物の変形を抑えるように作用することから、磁気減衰効果と

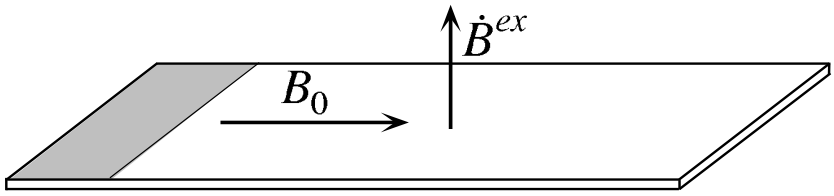
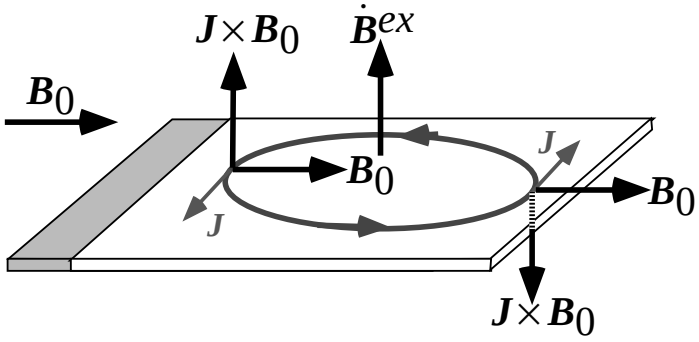
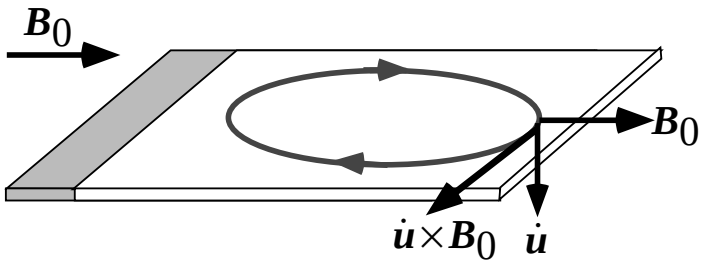


Fig. 4.2 Schematic view of the cantilever plate



(a) electromagnetic force



(b) electromotive force

Fig. 4.3 Electromagnetic structural coupling effect

呼ばれている。このように、電磁構造連成現象は、$\boldsymbol{J} \times \boldsymbol{B}$ 電磁力による変形が $\dot{\boldsymbol{u}} \times \boldsymbol{B}$ 速度起電力を誘起し、また、速度起電力によって電流が発生するという複雑な現象である。

4.2.2 速度起電力を考慮した渦電流解析

電流密度 $\boldsymbol{j}$ を電流ベクトルポテンシャル $\boldsymbol{T}$ を用いて

$$\boldsymbol{j} = \nabla \times \boldsymbol{T} \tag{4.1}$$

と表す。これを用いると、速度起電力を考慮した渦電流の支配方程式は、

$$\frac{1}{\kappa}\nabla \times \nabla \times \boldsymbol{T} = -\dot{\boldsymbol{B}} + \nabla \times (\dot{\boldsymbol{u}} \times \boldsymbol{B}) \tag{4.2}$$

となる。

ここで、解析対象を薄肉シェル構造物に限定すると、電流は構造物の面内方向成分のみを考慮すれば良い。そこで、局面シェル上の任意の点で、局面に沿って互いに直行する局面シェル座標系 $(\bar{x}, \bar{y}, \bar{z})$ を導入すると、電流ベクトルポテンシャル $\boldsymbol{T}$ は、局面に垂直な成分 T のみの電流スカラーポテンシャルとなる。

よって、電流スカラーポテンシャル T を用いて渦電流の支配方程式 (4.2) を書き改めると、

$$\frac{\partial B^{in}_{\bar{z}}}{\partial t} - \frac{1}{\kappa}\left(\frac{\partial^2 T}{\partial \bar{x}^2} + \frac{\partial^2 T}{\partial \bar{y}^2}\right) = -\frac{\partial B^{ex}_{\bar{z}}}{\partial t} \tag{4.3}$$

となり、これが薄肉シェル構造物に生じる渦電流の支配方程式となる。ここで、左辺第 1 項は渦電流が作る磁場、第 2 項は電気抵抗、第 3 項は構造物の変形による速度起電力、右辺は外部磁場を表す。

渦電流の解析法には、上述した電流ベクトルポテンシャル $\boldsymbol{T}$ に基礎をおく方法のほか、磁気ベクトルポテンシャル $\boldsymbol{A}$ に基礎をおく方法がある。磁気ベクトルポテンシャル $\boldsymbol{A}$ を用いる場合には、渦電流が流れる導体領域のほか、その周りの無限に広がる空気領域も解析領域として取り扱わなければならない。これに対して、本研究で採用する電流ベクトルポテンシャル $\boldsymbol{T}$ を用いる場合には、導体領域のみを取り扱えば良いので、少ない自由度数で解析を行なうことができる。

ガラーキン法を用いて支配方程式 (4.3) を離散化すると、速度起電力を考慮した渦電流解析の有限要素式

$$[\mathbf{U}]\{\dot{\mathrm{T}}\} + [\mathbf{R}]\{\mathrm{T}\} + [\mathbf{C}_e]\{\dot{\mathrm{u}}\} = \{\dot{\mathrm{B}}^{ex}\} \tag{4.4}$$

が得られる。ここで、$[\mathbf{U}]$ はインダクタンスマトリックス、$[\mathbf{R}]$ はレジスタンスマトリックス、$[\mathbf{C}_e]\{\dot{\mathrm{u}}\}$ は速度起電力項であり、$[\mathbf{C}_e]$ は磁気減衰に関する連成サブマトリックス、$\{\dot{\mathrm{B}}^{ex}\}$ は外部磁場ベクトルである。

4.2.3 電磁力を考慮した構造解析

(a) アイソパラメトリックシェル要素

曲面シェル構造物の応力解析を、8 節点アイソパラメトリックシェル要素[4.5]を用いて行なう。8 節点アイソパラメトリックシェル要素による変位場は、2 次元の 8 節点アイソパラメトリック要素の形状関数 $N_i(\xi,\eta)$ を用いて

$$\begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix} = \sum_{i=1}^{8} N_i(\xi,\eta) \begin{Bmatrix} u_{xi} \\ u_{yi} \\ u_{zi} \end{Bmatrix} + \sum_{i=1}^{8} N_i(\xi,\eta)\frac{\zeta}{2} \begin{bmatrix} \mathbf{v}_{1i} & -\mathbf{v}_{2i} \end{bmatrix} \begin{Bmatrix} \alpha_i \\ \beta_i \end{Bmatrix} \tag{4.5}$$

と表される。ここで、ξ,η,ζ は要素内において正規化された局所座標系、$\mathbf{v}_{1i}$ および $\mathbf{v}_{2i}$ は、Fig. 4.4 に示すように、節点 i における板厚方向のベクトル $\mathbf{V}_{3i}$ と直交する 2 軸方向の単位ベクトルであり、これらの軸まわりの回転角を β_i,α_i とする。

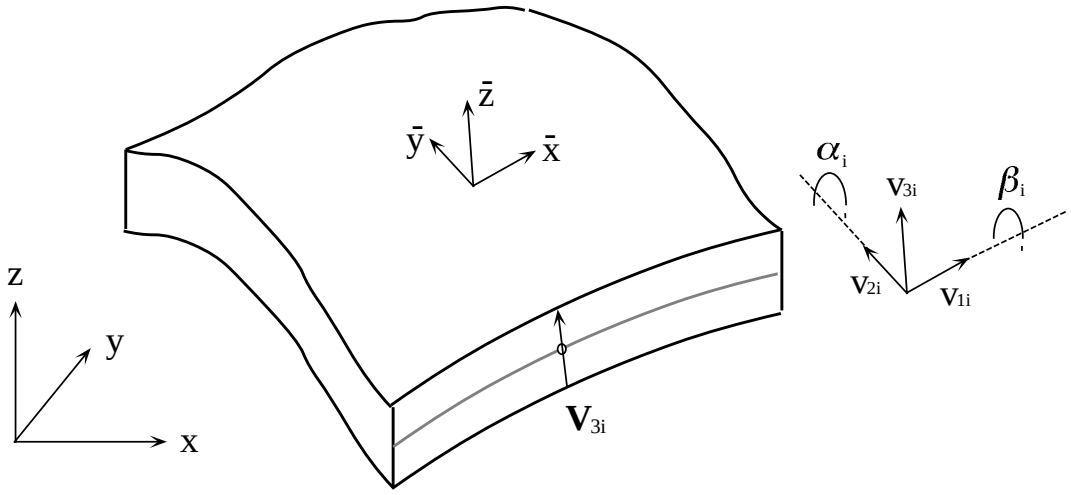
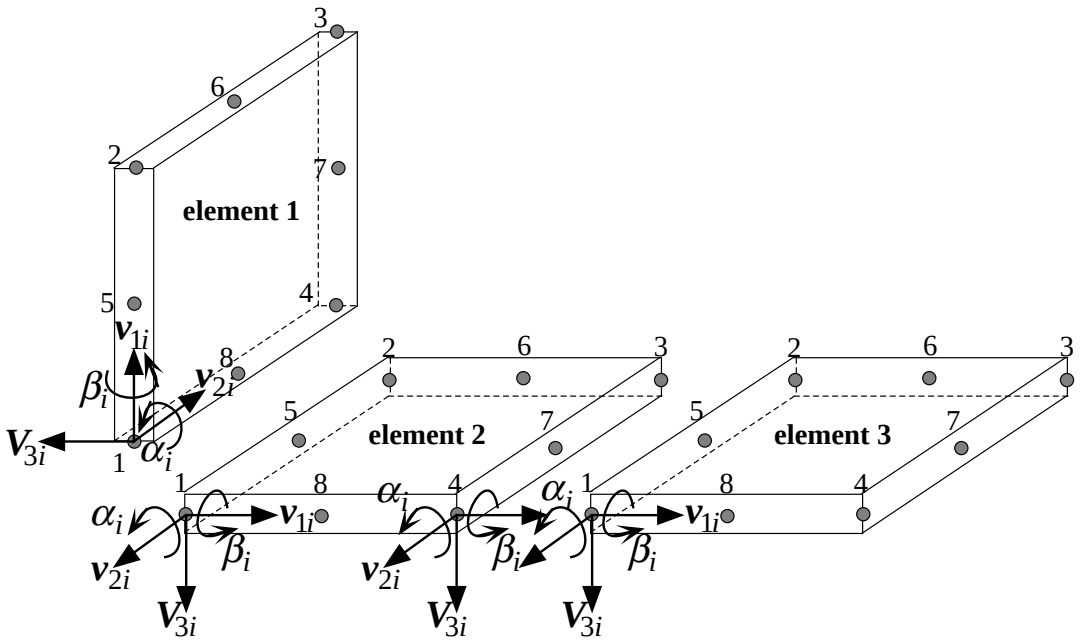


Fig. 4.4 Isoparametric shell element

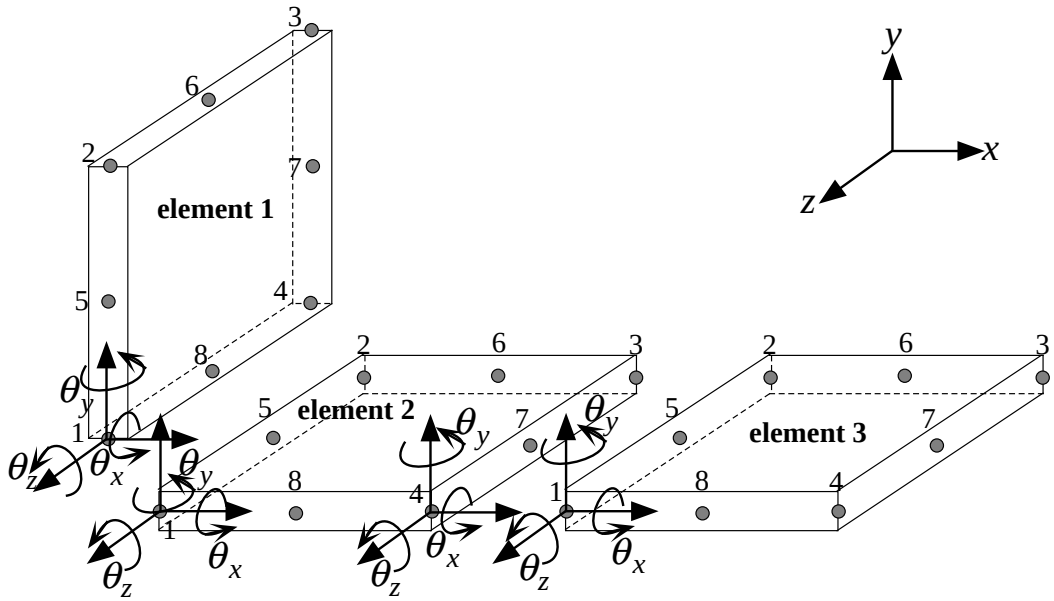
このように、8 節点アイソパラメトリックシェル要素では、各節点についてそれぞれ、3 方向の変位 u_{xi},u_{yi},u_{zi} のほか、回転角 α_i,β_i を含めた計 5 自由度となる。回転角 α_i,β_i については、要素ごとの局所座標系で定義されるが、Fig. (a) に示す要素 2 の節点 4 と要素 4 の節点 1 のように、要素の面外方向のベクトル $\mathbf{V}_{3i}$ が一致するため、要素間で適合する。

しかし、同図 (a) に示す要素 1 と要素 2 にように要素間で折れ曲がりがある場合には、それぞれの要素で定義される面外方向のベクトル $\mathbf{V}_{3i}$ が異なるため、$\mathbf{v}_{1i},\mathbf{v}_{2i}$ まわりの

回転角 β_i, α_i が非適合となる。したがって、折れ曲がり部分を含むシェル構造物を解析する場合には、同図 (b) のように、回転角を全体座標系 (x, y, z) で定義し、要素間における回転角の適合性を満足させなければならない。このとき、要素 2 の節点 4 と要素 3 の節点 1 のように、1 節点に集まる全ての要素が同一平面内にある場合には、面外方向まわりの回転剛性が 0 となるため、全体剛性マトリックスが特異となる。これを避けるため、仮想回転剛性と呼ばれる任意の値の剛性係数を、剛性マトリックスの面外方向まわりの回転角自由度に対して挿入する。以上のように、回転角を全体座標系で定義する場合には、各節点について、3 方向の変位と回転角 $\theta_x, \theta_y, \theta_z$ を含めた計 6 自由度となる。



(a) local coordinate system



(b) global coordinate system

Fig. 4.5 Definition of rotation angle

(b) 電磁力を考慮した有限要素式

電磁力を考慮して、8 節点アイソパラメトリック要素を用いたシェル構造物の有限要素式は、

$$[\mathbf{M}]\{\ddot{\mathrm{u}}\} + [\mathbf{K}]\{\mathrm{u}\} + [\mathbf{C}_s]\{\mathrm{T}\} = \{\mathrm{F}^{ex}\} \tag{4.6}$$

と表される。ここで、$[\mathbf{M}]$ は質量マトリックス、$[\mathbf{K}]$ は剛性マトリックス、$[\mathbf{C}_s]\{\mathrm{T}\}$ は電磁力項であり、$[\mathbf{C}_s]$ は電磁力に関するマトリックス、$\{\mathrm{F}^{ex}\}$ は外力ベクトルである。

4.2.4 連成解析方法

速度起電力を考慮した渦電流解析の有限要素式 (4.4) と、電磁力を考慮した構造解析の有限要素式 (4.6) を組合せ、連立させることにより電磁構造連成系の有限要素式

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathrm{u}} \\ \ddot{\mathrm{T}} \end{Bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{C}_e & \mathbf{U} \end{bmatrix} \begin{Bmatrix} \dot{\mathrm{u}} \\ \dot{\mathrm{T}} \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & \mathbf{C}_s \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \begin{Bmatrix} \mathrm{u} \\ \mathrm{T} \end{Bmatrix} = \begin{Bmatrix} \mathrm{F}^{ex} \\ \dot{\mathrm{B}}^{ex} \end{Bmatrix} \tag{4.7}$$

が得られる[4.3]。

この式に、ニューマークの β 法およびクランク・ニコルソン法の直接時間積分法を適用すると、

$$\begin{bmatrix} \mathbf{K} + \dfrac{\mathbf{M}}{\beta(\Delta t)^2} & \mathbf{C}_s \\ -\mathbf{C}_e & -(\mathbf{U} + \alpha\Delta t\mathbf{R}) \end{bmatrix} \begin{Bmatrix} \mathrm{u} \\ \mathrm{T} \end{Bmatrix}_{n+1} = \begin{Bmatrix} \mathrm{F}_1 \\ -\mathrm{F}_2 \end{Bmatrix} \tag{4.8}$$

という連立一次方程式が得られる。ここで、Δt は時間きざみ幅、β はニューマークの β 法における定数で $\beta = \frac{1}{4}$、α はクランク・ニコルソン法における定数で $\alpha = \frac{1}{2}$ である。また、右辺ベクトル $\{\mathrm{F}_1\}, \{\mathrm{F}_2\}$ はそれぞれ

$$\{\mathrm{F}_1\} = \{\mathrm{F}^{ex}\}_{n+1} - [\mathbf{M}]\left\{\left(1 - \frac{1}{2\beta}\right)\{\ddot{\mathrm{u}}\}_n - \frac{1}{\beta\Delta t}\{\dot{\mathrm{u}}\}_n - \frac{1}{\beta(\Delta t)^2}\{\mathrm{u}\}_n\right\} \tag{4.9}$$

$$\{\mathrm{F}_2\} = -\alpha\Delta t\{\dot{\mathrm{B}}^{ex}\}_{n+1} - (1-\alpha)\Delta t\{\dot{\mathrm{B}}^{ex}\}_n + [\mathbf{C}_e]\{\mathrm{u}\}_n + \Big([\mathbf{U}] - (1-\alpha)\Delta t[\mathbf{R}]\Big)\{\mathrm{T}\}_n \tag{4.10}$$

である。

この連立一次方程式を時間ステップごとに解くことにより、変位 $\{\mathrm{u}\}$ および電流スカラーポテンシャル $\{\mathrm{T}\}$ を同時に求めることができる。なお、マトックス $[\mathbf{M}], [\mathbf{K}], [\mathbf{U}], [\mathbf{R}]$ はそれぞれ対称であり、また、

$$-[\mathbf{C}_e] = [\mathbf{C}_s]^T \tag{4.11}$$

であることから、式 (4.8) の係数マトリックスは対称である。

式 (4.8) の未知数には、シェル要素による変位 {u} および電流スカラーポテンシャル {T} の両方を含むため、大次元の連立一次方程式となる。さらに、インダクタンスマトリックス [**U**] はフルマトックスであるため、この連立一次方程式の解法には、多大な計算時間と記憶容量が必要となる。

4.3 連成問題の並列化方法

4.3.1 電磁構造連成解析への領域分割法の適用性

2.2.4 節で述べた領域分割法の手順にしたがって電磁構造連成解析の並列化を行なおうとした場合、領域を分割すると、Fig. 4.6 に示すように、相互インダクタンスにより領域内を流れる渦電流 $\boldsymbol{j}$ によって誘起される磁場 $\dot{\boldsymbol{B}}^{in}$ が空気中を伝わり、領域間で互いに影響を及ぼし合う。したがって、領域 (k) の渦電流自由度のマトリックス方程式を、領域の内部自由度 (添字 (1)) と領域間境界自由度 (添字 (2)) に分けて表すと

$$\begin{bmatrix} \mathbf{A}_{11}^{(k)} & \mathbf{A}_{12}^{(k)} & \mathbf{A}_{13} \\ \mathbf{A}_{21}^{(k)} & \mathbf{A}_{22}^{(k)} & \mathbf{A}_{23} \end{bmatrix} \begin{Bmatrix} \mathrm{T}_1^{(k)} \\ \mathrm{T}_2^{(k)} \\ \mathrm{T}_3 \end{Bmatrix} = \begin{Bmatrix} \mathrm{B}_1^{(k)} \\ \mathrm{B}_2^{(k)} \end{Bmatrix} \tag{4.12}$$

のようになる。この方程式の係数マトリックスは正方ではなく、未知数には、領域内と境界上の電流スカラーポテンシャル $\left\{\mathrm{T}_1^{(k)}\right\}, \left\{\mathrm{T}_2^{(k)}\right\}$ のほかに、他の全領域の電流スカラーポテンシャル $\{\mathrm{T}_3\}$ が含まれている。このため、領域分割法の計算方法にしたがって、領域間境界上の電流スカラーポテンシャル $\left\{\mathrm{T}_2^{(k)}\right\}$ の部分に境界条件を設定しても、領域ごとに独立に解析を行なうことができない。よって、電磁構造連成解析に領域分割法をそのまま適用して並列化を行なうことは、困難である。

4.3.2 領域型共役勾配法による並列化

電磁構造連成解析に領域型共役勾配法を適用する場合には、変位および電流スカラーポテンシャルの全自由度に対して共役勾配法が適用されるため、解析領域を分割しても、領域間におけるインダクタンスの影響を考慮することができる。ただし、各領域における演算には、全自由度の電流スカラーポテンシャルが必要となる。

解析領域を 2 つの領域に分割した場合、時間ステップごとに解くべき連立一次方程式

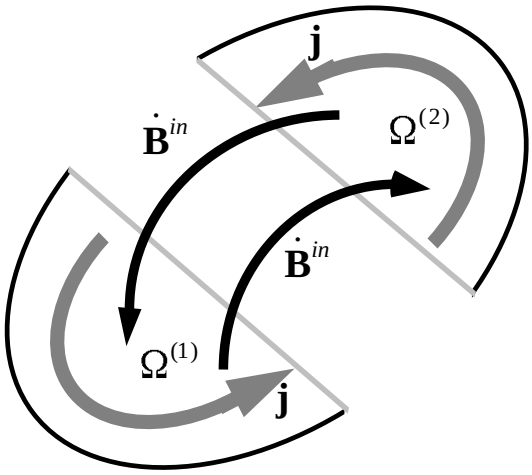


Fig. 4.6 Mutual inductance between domains

(4.8) は

$$\begin{bmatrix} \mathbf{K}^{(1)}+\dfrac{\mathbf{M}^{(1)}}{\beta(\Delta t)^2} & & \mathbf{C}_s^{(1)} & \\ & \mathbf{K}^{(2)}+\dfrac{\mathbf{M}^{(2)}}{\beta(\Delta t)^2} & & \mathbf{C}_s^{(2)} \\ -\mathbf{C}_e^{(1)} & & -\left(\mathbf{U}^{(1)}+\alpha\Delta t\mathbf{R}^{(1)}\right) & \\ & -\mathbf{C}_e^{(1)} & & -\left(\mathbf{U}^{(2)}+\alpha\Delta t\mathbf{R}^{(2)}\right) \end{bmatrix} \begin{Bmatrix} \mathrm{u}^{(1)} \\ \mathrm{u}^{(2)} \\ \mathrm{T} \end{Bmatrix}_{n+1} = \begin{Bmatrix} \mathrm{F}_1^{(1)} \\ \mathrm{F}_1^{(2)} \\ -\mathrm{F}_2 \end{Bmatrix} \tag{4.13}$$

となる。したがって、領域ごとに 6 つのマトリックス $\left[\mathbf{M}^{(k)}\right]$, $\left[\mathbf{K}^{(k)}\right]$, $\left[\mathbf{U}^{(k)}\right]$, $\left[\mathbf{R}^{(k)}\right]$, $\left[\mathbf{C}_s^{(k)}\right]$, $\left[\mathbf{C}_e^{(k)}\right]$ を領域内の要素マトリックスからそれぞれ作成すると、共役勾配法における係数マトリックスと探索方向ベクトルとの積を、領域ごとに構造自由度および渦電流自由度についてそれぞれ

$$\left\{\mathrm{q}_s^{(k)}\right\} = \left[\mathbf{K}^{(k)}+\frac{\mathbf{M}^{(k)}}{\beta(\Delta t)^2}\right]\left\{\mathrm{u}^{(k)}\right\} + \left[\mathbf{C}_s^{(k)}\right]\left\{\mathrm{T}^{(k)}\right\} \tag{4.14}$$

$$\left\{\mathrm{q}_e^{(k)}\right\} = \left[-\mathbf{C}_e^{(k)}\right]\left\{\mathrm{u}^{(k)}\right\} + \left[-\left(\mathbf{U}^{(k)} + \alpha\Delta t\mathbf{R}^{(k)}\right)\right]\{\mathrm{T}\} \tag{4.15}$$

のように行なうことができる。ここで、渦電流自由度の演算には、全領域の電流スカラーポテンシャル {T} が必要となる。

領域ごとに得られた演算結果を全領域について

$$\{\mathrm{b}\} = \sum_k \left\{ \begin{array}{c} \mathrm{q}_s^{(k)} \\ \mathrm{q}_e^{(k)} \end{array} \right\} \tag{4.16}$$

のように全体化することにより、マトリックス・ベクトル演算の結果を得ることができる。

式 (4.13) の係数マトリックスは、条件数が大きい性質の悪いマトリックスであるため、対角項を用いたスケーリングによる前処理を行ない、共役勾配法の収束性を向上させる。また、非定常解析における第 2 時間ステップ目以降においては、共役勾配法の初期解として 1 つ前の時間ステップにおける解を用いて、高速化をはかる。

4.3.3 領域分割法と領域型共役勾配法を組み合わせた並列化方法

電磁構造連成問題において、渦電流解析には領域分割法を適用することができないが、構造自由度に関しては領域ごとに独立に解析を行なうことが可能である。そこで、渦電流自由度は前節で述べた領域型共役勾配法を用いて並列解析を行ない、構造自由度には領域分割法を適用して並列解析を行なう。ただし、連成解析には連成サブマトリックスを含んでおり、さらに領域分割法と領域型共役勾配法を組み合わせて同時に反復計算を行なうため、以下に示す計算法により並列化を行なう。

式 (4.8) における構造自由度を、領域 (k) について、内部自由度 (添字 i) と境界自由度 (添字 b) に分けて表すと、

$$\begin{bmatrix} \mathbf{K}_{ii}^{(k)} + \dfrac{\mathbf{M}_{ii}^{(k)}}{\beta(\Delta t)^2} & \mathbf{K}_{ib}^{(k)} + \dfrac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2} \\ \mathbf{K}_{ib}^{(k)^T} + \dfrac{\mathbf{M}_{ib}^{(k)^T}}{\beta(\Delta t)^2} & \mathbf{K}_{bb}^{(k)} + \dfrac{\mathbf{M}_{bb}^{(k)}}{\beta(\Delta t)^2} \end{bmatrix} \begin{Bmatrix} \mathrm{u}^{(k)} \\ \mu^{(k)} \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{s_i}^{(k)} \\ \mathbf{C}_{s_b}^{(k)} \end{bmatrix} \left\{\mathrm{T}^{(k)}\right\} = \begin{Bmatrix} \mathrm{f}^{(k)} \\ \tau^{(k)} \end{Bmatrix} \tag{4.17}$$

となる。これより、領域の内部自由度の変位 $\{\mathrm{u}^{(k)}\}$ は、境界自由度の変位 $\{\mu^{(k)}\}$ と、その領域内の電流スカラーポテンシャル $\{\mathrm{T}^{(k)}\}$ を用いて

$$\left\{\mathrm{u}^{(k)}\right\} = \left\{\mathrm{f}^{(k)}\right\} - \left[\mathbf{C}_{s_i}^{(k)}\right]\left\{\mathrm{T}^{(k)}\right\} - \left[\mathbf{K}_{ii}^{(k)} + \frac{\mathbf{M}_{ii}^{(k)}}{\beta(\Delta t)^2}\right]^{-1}\left[\mathbf{K}_{ib}^{(k)} + \frac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2}\right]\left\{\mu^{(k)}\right\} \tag{4.18}$$

のように求めることができる。これを用いて、領域分割法の境界自由度に対する演算と、領域型共役勾配法の演算を行なう。境界自由度に対する共役勾配法の係数マトリックス

とベクトルとの積演算は、各領域ごとに

$$\begin{aligned}\left\{\tau^{(k)}\right\} &= \left[\mathbf{K}_{ib}^{(k)} + \frac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2}\right]^T \left\{\mathrm{u}^{(k)}\right\} + \left[\mathbf{K}_{bb}^{(k)} + \frac{\mathbf{M}_{bb}^{(k)}}{\beta(\Delta t)^2}\right] \left\{\mu^{(k)}\right\} + \left[\mathbf{C}_{s_b}^{(k)}\right] \left\{\mathrm{T}^{(k)}\right\} \\ &= \left(\left[\mathbf{K}_{bb}^{(k)} + \frac{\mathbf{M}_{bb}^{(k)}}{\beta(\Delta t)^2}\right] \right. \\ &\qquad \left. - \left[\mathbf{K}_{ib}^{(k)} + \frac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2}\right]^T \left[\mathbf{K}_{ii}^{(k)} + \frac{\mathbf{M}_{ii}^{(k)}}{\beta(\Delta t)^2}\right]^{-1} \left[\mathbf{K}_{ib}^{(k)} + \frac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2}\right] \right) \left\{\mu^{(k)}\right\} \\ &\quad + \left(\left[\mathbf{C}_{s_b}^{(k)}\right] - \left[\mathbf{K}_{ib}^{(k)} + \frac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2}\right]^T \left[\mathbf{C}_{s_i}^{(k)}\right] \right) \left\{\mathrm{T}^{(k)}\right\} \\ &\quad + \left[\mathbf{K}_{ib}^{(k)} + \frac{\mathbf{M}_{ib}^{(k)}}{\beta(\Delta t)^2}\right]^T \left\{\mathrm{f}^{(k)}\right\} \end{aligned} \tag{4.19}$$

と行なうことができる。

渦電流自由度についても、領域 (k) について

$$\left[-\mathbf{C}_e^{(k)}\right] \begin{Bmatrix} \mathrm{u}^{(k)} \\ \mu^{(k)} \end{Bmatrix} + \left[-\left(\mathbf{U}^{(k)} + \alpha\Delta t\mathbf{R}^{(k)}\right)\right] \left\{\mathrm{T}\right\} = \left\{\mathrm{q}_e^{(k)}\right\} \tag{4.20}$$

と表される。したがって、境界自由度の変位 $\left\{\mu^{(k)}\right\}$ と全自由度の電流スカラーポテンシャル $\{\mathrm{T}\}$、および、式 (4.18) により求められた領域の内部自由度の変位 $\{\mathrm{u}^{(k)}\}$ を用いて、領域型共役勾配法における領域ごとの係数マトリックスとベクトルとの積演算を行なうことができる。

こうして各領域ごとに得られた演算結果 $\left\{\tau^{(k)}\right\}, \left\{\mathrm{q}_e^{(k)}\right\}$ を集めた後、全領域について

$$\begin{Bmatrix} \tau \\ \mathrm{q}_e \end{Bmatrix} = \sum_k \begin{Bmatrix} \tau^{(k)} \\ \mathrm{q}_e^{(k)} \end{Bmatrix} \tag{4.21}$$

のように総和を取ることにより、構造境界自由度と渦電流自由度の両方に対して、同時に共役勾配法の演算を行なうことができる。ここで、共役勾配法の反復計算は、境界自由度の変位および電流スカラーポテンシャルに対して行なわれるため、式 (4.8) に対して直接 CG 法を適用した場合に比べ、反復回数およびデータ転送量を削減することができる。

以上をまとめると、実際の計算手順は以下のようになる。

Step 1: 初期化

領域間境界に初期変位 $\{\mu\}$、全領域の電流スカラーポテンシャルに初期値 $\{\mathrm{T}\}$ を設定する。

Step 2: 各領域での計算

1. 領域の内部自由度の変位 $\left\{\mathrm{u}^{(k)}\right\}$ を、電流スカラーポテンシャル $\left\{\mathrm{T}^{(k)}\right\}$ により領域に作用する電磁力 $\left[\mathbf{C}_{s_i}^{(k)}\right]\left\{\mathrm{T}^{(k)}\right\}$、および、境界に設定された変位 $\left\{\mu^{(k)}\right\}$ を用いて、式 (4.18) を解いて求める。
2. 構造境界自由度に対する領域分割法の計算 (式 (4.19)) を行なう。
3. 領域の内部自由度の変位 $\left\{\mathrm{u}^{(k)}\right\}$ と全領域の電流スカラーポテンシャル $\{\mathrm{T}\}$ を用いて、渦電流自由度に対する領域型共役勾配法の計算を行なう。

Step 3: 全体化と共役勾配法のパラメータ更新

1. 各領域での構造自由度の計算結果 $\left\{\tau^{(k)}\right\}$ と、渦電流自由度の計算結果 $\left\{\mathrm{q}_e^{(k)}\right\}$ を集めて全体化し、解ベクトル $\{\mu\}$ および $\{\mathrm{T}\}$ を更新する。
2. 収束判定を行ない、解が収束していなければ、残差ベクトルおよび探索方向ベクトルを更新して **Step 2** に戻る。
3. 解が収束していれば、非定常解析の時間ステップを更新し **Step 1** に戻る。

4.3.4 クラスタ環境へのシステムの実装

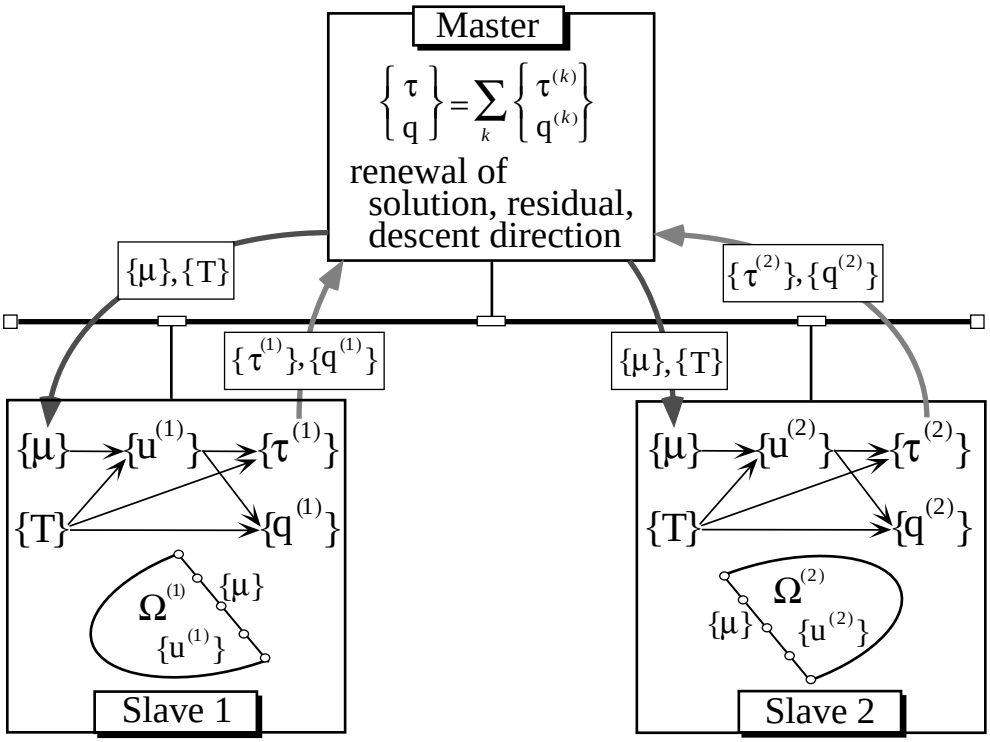
ワークステーション・クラスタへのシステムの実装は、Fig. 4.7 のように、スレーブ・マシン上に領域ごとの演算を割り当てて並列計算を行ない、マスターマシン上で各領域の計算結果を全体化し共役勾配法のパラメータ更新演算を行なう。

マスターからスレーブへ転送されるデータは、Table 4.1 に示すように、領域型共役勾配法の場合には、

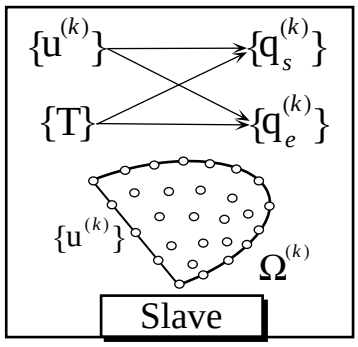
- スレーブが担当する領域 (k) の構造自由度 $\left\{\mathrm{u}^{(k)}\right\}$
- 全領域の渦電流自由度 $\{\mathrm{T}\}$

に対応した探索方向ベクトルであるのに対して、領域分割法と領域型共役勾配法を組み合わせた方法の場合には、

- スレーブが担当する領域 (k) の境界上の構造自由度 $\left\{\mu^{(k)}\right\}$



(a) combined domain decomposition method



(b) domain CG method

Fig. 4.7 Schematic diagram of a workstation cluster system

- 全領域の渦電流自由度 {T}

となり、データ転送量が構造自由度に関して境界自由度数分へと小さくなる。また、スレーブからマスターへの演算結果の転送は、領域型共役勾配法の場合には、

- 領域の構造自由度の演算結果 $\left\{q_s^{(k)}\right\}$

- 渦電流自由度の演算結果 $\left\{\mathrm{q}_e^{(k)}\right\}$

であるのに対して、領域分割法と領域型共役勾配法を組み合わせた方法の場合には、

- 境界上の構造自由度の演算結果$\left\{\tau^{(k)}\right\}$
- 渦電流自由度の演算結果 $\left\{\mathrm{q}_e^{(k)}\right\}$

となる。

スレーブ・マシン上で行なわれる領域ごとの計算は、領域型共役勾配法の場合には、式 (4.14) および (4.15) における係数マトリックスと探索方向ベクトルとの積演算であるのに対し、領域分割法と領域型共役勾配法を組み合わせた方法の場合には、前節で述べたように、式 (4.18)〜(4.20) の演算を行なう。共役勾配法の反復の対象となる自由度は、領域型共役勾配法の場合には変位 {u} および電流スカラーポテンシャル {T} の全ての未知数であるのに対し、領域分割法と領域型共役勾配法を組み合わせた方法の場合には、領域間境界の変位 $\{\mu\}$ および電流スカラーポテンシャル {T} と、構造自由度に関して小さくなる。

Table 4.1 Comparison between domain partitioned CG method and combined domain decomposition method

	Domain CG method	Combined DDM method
Data transmission (Master→Slave)	$\left\{\mathrm{u}^{(k)}\right\}$, {T}	$\left\{\mu^{(k)}\right\}$, {T}
Parallel computation (Slave)	Eqn. (4.14), (4.15)	Eqn. (4.18)〜(4.20)
Data transmission (Slave→Master)	$\left\{\mathrm{q}_s^{(k)}\right\}$, $\left\{\mathrm{q}_e^{(k)}\right\}$	$\left\{\tau^{(k)}\right\}$, $\left\{\mathrm{q}_e^{(k)}\right\}$
CG iteration (Master)	{u}, {T}	$\{\mu\}$, {T}

4.4 並列性能の評価

4.4.1 並列性能の定量化

(a) 領域分割法と領域型共役勾配法を組み合わせた方法の並列性能の定量化

各領域における構造自由度数 n'_t、スカイラインの平均カラム高さ n'_{bw}、境界上の構造自由度数 n'_{bf}、渦電流自由度 n'_e を一定とすると、反復 1 回当たりの各処理時間は次のように表される。

- スレーブ・マシンへの探索方向ベクトルの転送

$$T_s = \alpha_c \left(n'_{bf} \frac{n_d}{n_s} + n_e \right) + \beta \tag{4.22}$$

- 1 領域分の演算時間 (反復 1 回目)

$$T_{d1} = \alpha_{d1} n'^2_{bw} n'_t + \alpha_{es} n'_e n'_t + \alpha_{ed} n_e n'_e \tag{4.23}$$

- 1 領域分の演算時間 (反復 2 回目)

$$T_{d2} = \alpha_{d2} n'_{bw} n'_t + \alpha_{es} n'_e n'_t + \alpha_{ed} n_e n'_e \tag{4.24}$$

- スレーブ・マシンにおける演算結果の転送

$$T_g = \alpha_c \left(n'_{bf} + n'_e \right) \frac{n_d}{n_s} + \beta \tag{4.25}$$

- 共役勾配法のパラメータ修正演算

$$T_m = \alpha_{cg} \left(n_{bf} + n_e \right) \tag{4.26}$$

ここで、n_e は全領域の渦電流自由度数、n'_{bf} は領域の境界上の構造自由度数、n_{bf} は全境界の構造自由度数、n_d は総領域数、n_s はスレーブ台数であり、α_c, β, α_{d1}, α_{es}, α_{ed}, α_{d2}, α_{cg} は定数である。

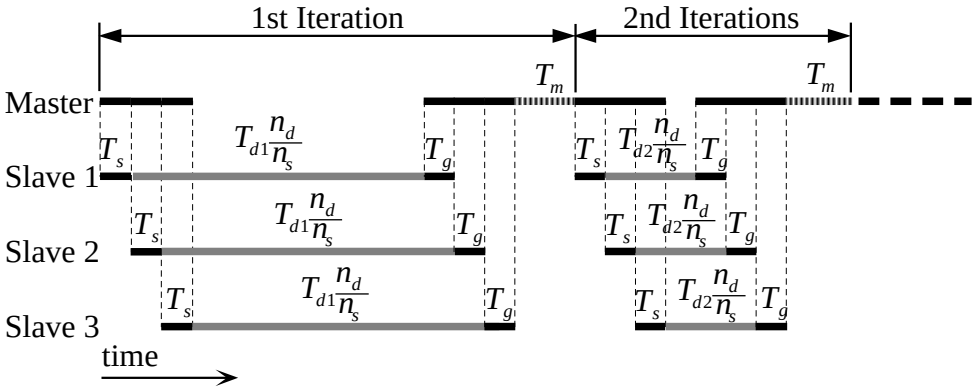
これらそれぞれの処理は、Fig. 4.8 に示すように繰り返される。このタイムチャートに基づいて、反復回数を I_t とすると、解析時間は

$$T = T_{d1} \frac{n_d}{n_s} + T_{d2} \frac{n_d}{n_s} (I_t - 1) + (T_s + T_g n_s + T_m) I_t \tag{4.27}$$

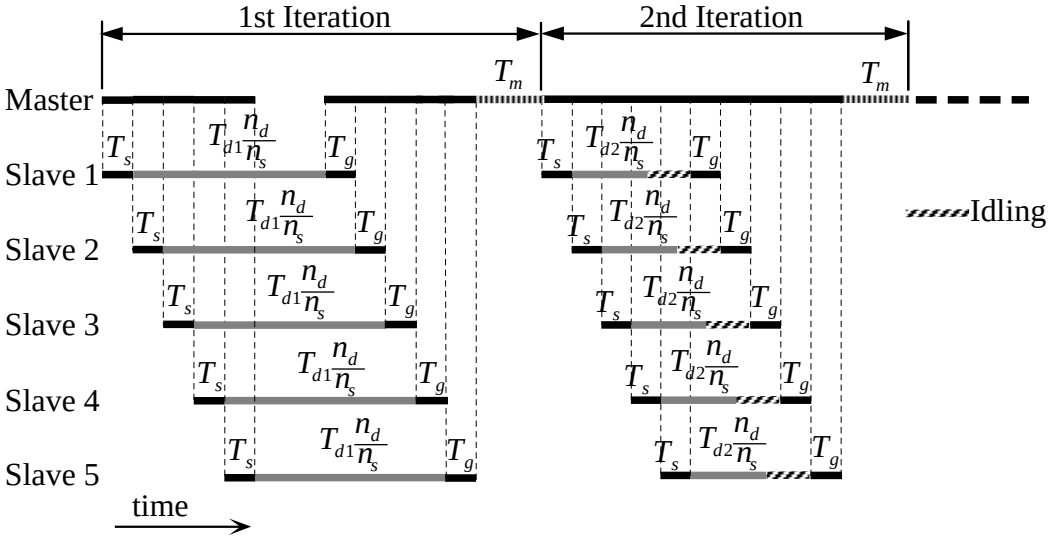
と表される。また、ネットワークの飽和が生じた場合には、Fig. 4.8 (b) にもとづき、

$$T = T_{d1} \frac{n_d}{n_s} + T_s (n_s I_t - n_s + 1) + T_g n_s (I_t - 1) + T_m I_t \tag{4.28}$$

と表される。



(a) normal condition



(b) network saturated condition

Fig. 4.8 Time chart of combined domain decomposition method

(b) 領域型共役勾配法の並列性能の定量化

領域型共役勾配法による並列化の場合には、各処理時間は次のようになる

- スレーブ・マシンへの探索方向ベクトルの転送

$$T_s = \alpha_c \left(n_t' \frac{n_d}{n_s} + n_e \right) + \beta \tag{4.29}$$

- 1 領域分の演算時間

$$T_d = \alpha_1 n_t' + 2\alpha_2 n_e' n_t' + \alpha_3 n_e n_e' \tag{4.30}$$

- スレーブ・マシンにおける演算結果の転送

$$T_g = \alpha_c \left(n'_t + 2n'_e \right) \frac{n_d}{n_s} + \beta \tag{4.31}$$

- 共役勾配法のパラメータ修正演算

$$T_m = \alpha_{cg} \left(n_t + n_e \right) \tag{4.32}$$

ここで、α_1, α_2, α_3 は定数である。

反復回数を I_t とすると、領域分割法を組み合わせた方法の場合と同様にタイムチャートに基づいて、解析時間は

$$T = \left(T_s + T_d \frac{n_d}{n_s} + T_g n_s + T_m \right) I_t \tag{4.33}$$

と表される。また、ネットワークの飽和が生じた場合には、

$$T + \left(T_s + T_r \right) n_s I_t + T_m I_t \tag{4.34}$$

と表される。

4.4.2 並列性能の評価および比較

(a) 平板たわみの連成問題

並列性能を評価するため、Fig. 4.9 に示す磁場中におかれた平板のたわみ連成問題を並列解析する。この問題は、平板と垂直な方向に

$$B_z(t) = 5.5 \times 10^{-2} \exp\left(-\frac{t}{6.6 \times 10^{-3}} \right) \qquad [\mathrm{T}] \tag{4.35}$$

のように時間的に変化する磁場を印加し、渦電流を発生させる。発生した渦電流と、平板の長手方向の定常磁場 $\boldsymbol{B}_x$ により電磁力が作用するため、平板が振動するとともに速度起電力が発生する。定常磁場 $\boldsymbol{B}_x$ の大きさを変えると連成の強さが変わるため、平板の自由端におけるたわみは Fig. 4.10 のように変化する。

この問題を解析するため、Fig. 4.11 に示すように、平板を 8 節点アイソパラメトリックシェル要素により要素分割し、さらにこれを各領域の自由度数が等しくなるように 2, 4, 8 領域に分割する。なお、本解析においては、定常磁場 $\boldsymbol{B}_x$ を 0.5 [T]、時間きざみ幅 Δt を 0.1 [ms] とする。

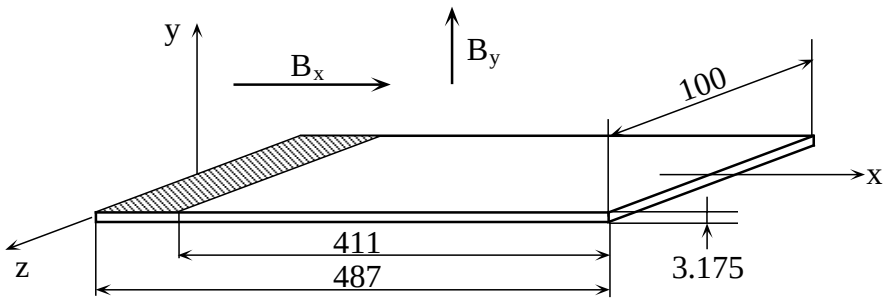


Fig. 4.9 Schematic diagram of cantilever plate in crossed magnetic field

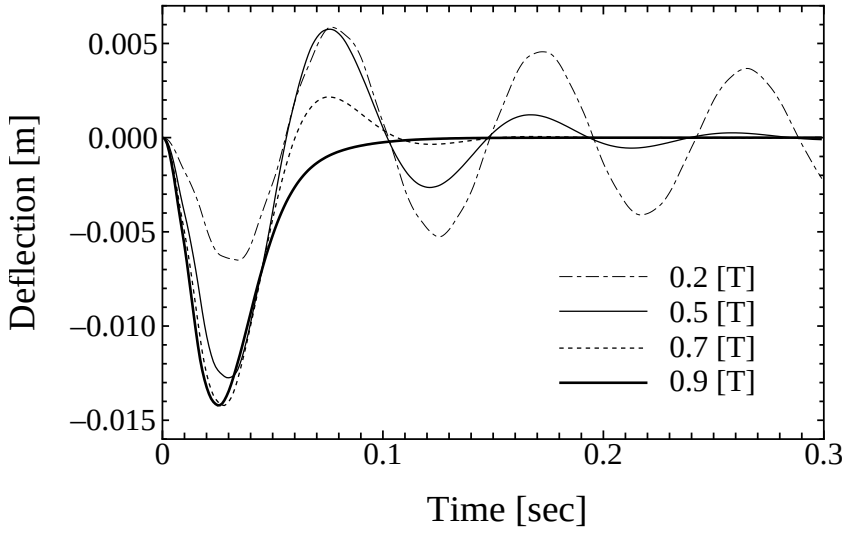


Fig. 4.10 Deflection at free end

(b) 評価式の妥当性の検証

Fig. 4.11 における 8 領域に分割した場合について、クラスタ・システムに SPARCstation 10 (60.2 SPECfp92, 50.2 SPECint92) および Ethernet (10Base-T, 10 Mbps) を用いたときの評価時間と実際の解析時間を比較して、Fig. 4.12 に示す。

この問題は規模が小さいため、いずれの方法においてもスレーブ台数が 3 台を越えるとネットワークが飽和状態となっている。実際の解析時間と評価時間は、いずれの方法においても、通常の場合およびネットワークが飽和した場合ともに良く一致しており、評価式の妥当性が検証される。

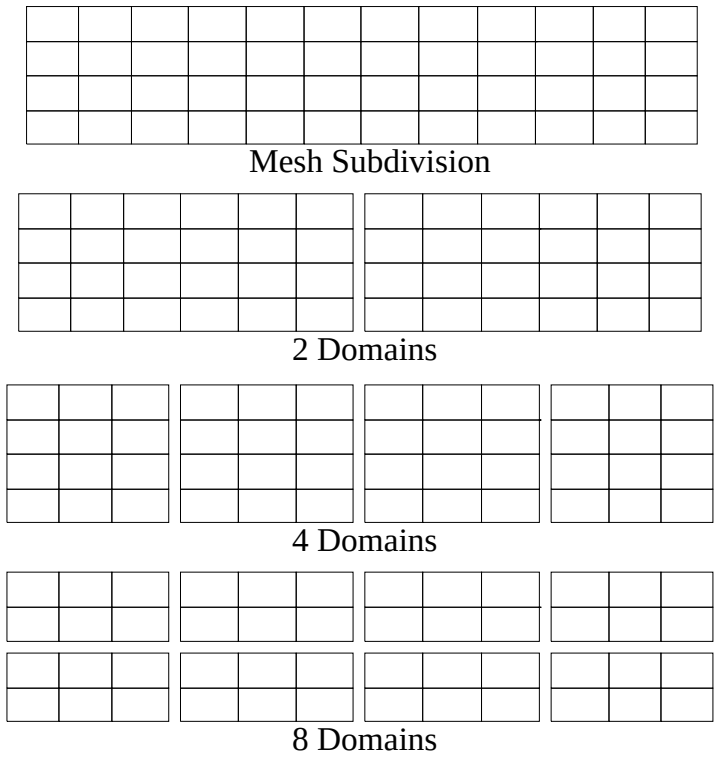


Fig. 4.11 Finite element mesh and domains of the plate

(c) 共役勾配法の解の収束性の比較

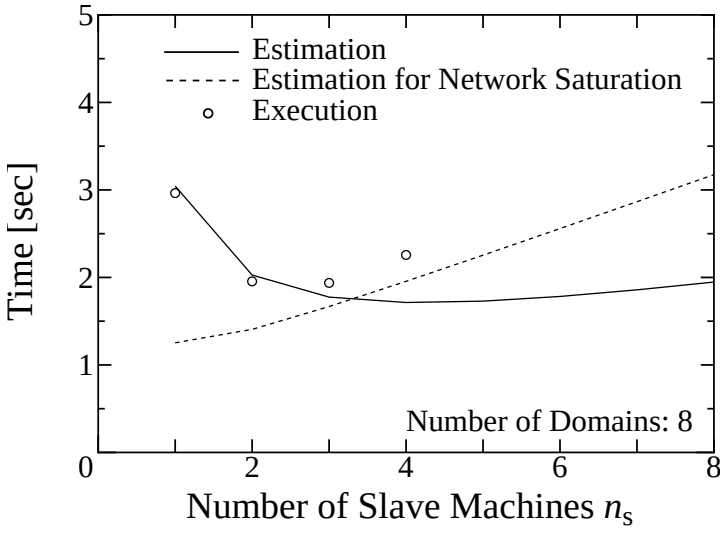
Fig. 4.11 に示した 8 領域の問題について、2 つの方法の共役勾配法の収束の様子を比較して Fig. 4.13 (a) に示す。領域分割法と領域型共役勾配法を組み合わせた方法の場合には、反復計算の対象となる構造自由度が境界自由度数分へと小さくなることから、収束性が向上している。

しかし、領域数を 2, 4, 8 と変化させると、境界自由度数の変化にともなって収束性が Fig. 4.13 (b) のように変化する。

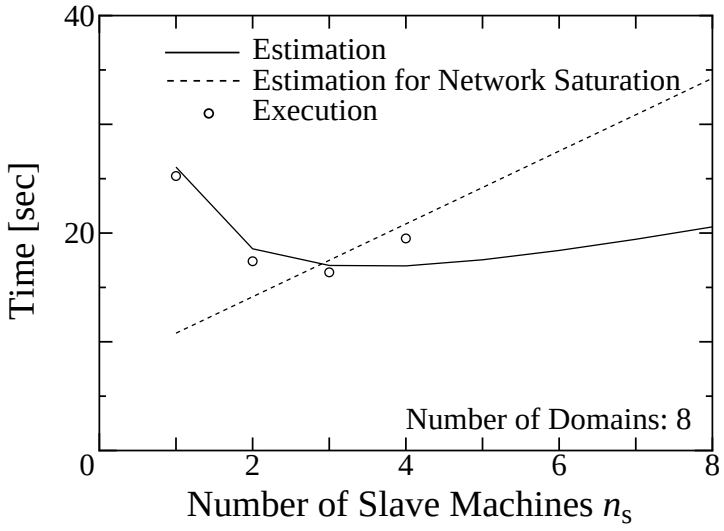
(d) 並列性能の比較

2 つの方法において、スレーブ数は 2 台に固定し、領域数を Fig. 4.11 のように 2, 4, 8 と変化させたときの解析時間を Table 4.2 に示す。領域型共役勾配法の場合には、領域分割数を変えても演算量および通信量ともに大きく変化しないため、解析時間はほぼ一定である。

領域分割法と領域型共役勾配法を組み合わせた方法の場合には、領域分割数により領



(a) combined DDM method

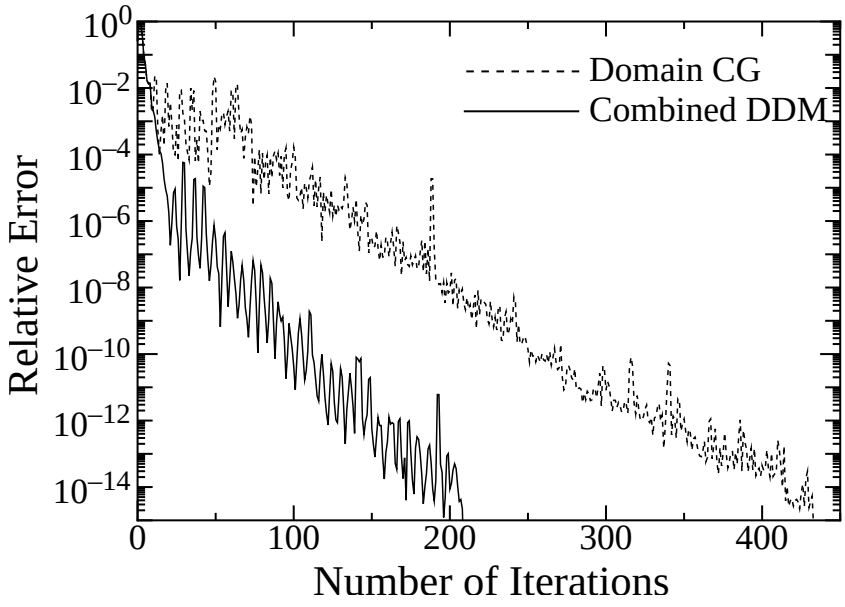


(b) domain CG method

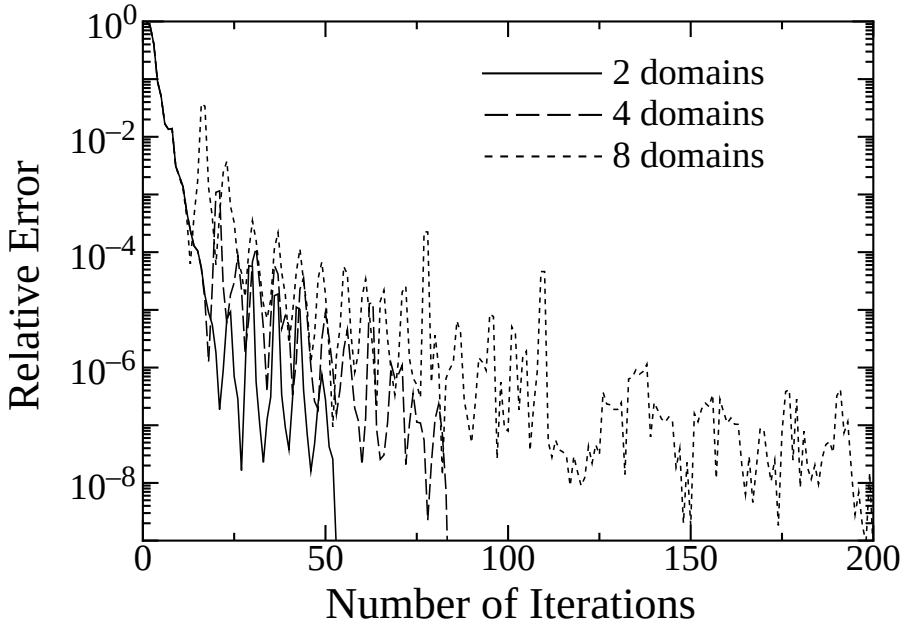
Fig. 4.12 Comparison between estimation and execution time

域ごとの演算量が変化するとともに、共役勾配法の収束性も変化するため、解析時間が変化している。しかし、データ転送量および解が収束するための反復回数は領域型共役勾配法に比べて小さいため、短時間で解析できることがわかる。

平板の要素分割を Fig. 4.14 に示すように、問題の規模を大きくした場合の並列性能を、Fig. 4.15 に示す。領域分割法と領域型共役勾配法を組み合わせた方法においては、構造自由度に関するデータ転送量が小さくなることから、領域型共役勾配法に比べて高



(a) comparison between domain CG and combined DDM



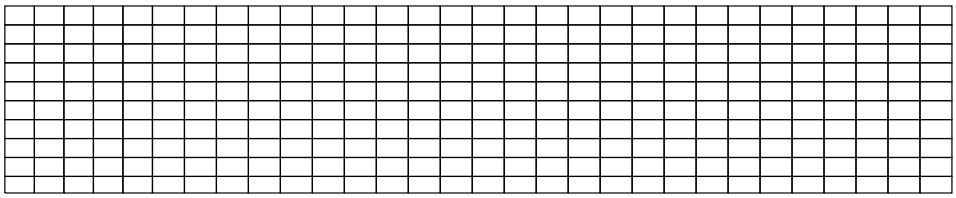
(b) dependence on number of domains

Fig. 4.13 Convergence of CG method

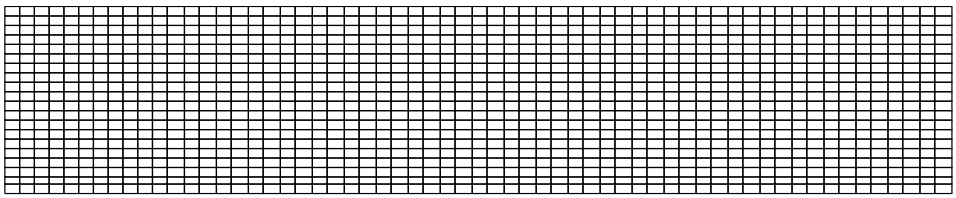
い並列性能が得られる。

Table 4.2 Comparison of computation time

Number of Domains	Combined DDM method		Domain CG method	
	Estimation [sec]	Execution [sec]	Estimation [sec]	Execution [sec]
2	0.84	0.89	19.1	17.4
4	1.35	1.33	18.4	16.4
8	2.03	1.96	18.6	17.4



(a) 300 elements, 3221 dof



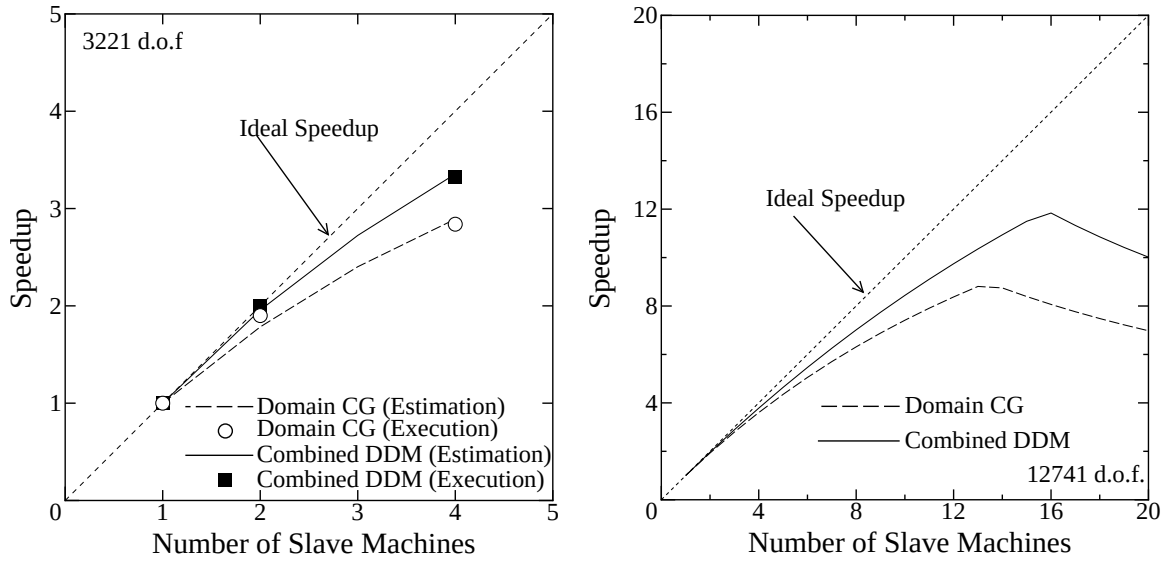
(b) 1200 elements, 12741 dof

Fig. 4.14 Finite element mesh for large-scale problem

4.5 核融合装置真空容器への適用

4.5.1 解析条件

大規模連成問題の例として、Fig. 4.1 に示した臨界プラズマ試験装置 JT–60U の真空容器に生じる電磁構造連成現象を並列解析する。この真空容器は、プラズマを閉じ込めるためのトロイダル磁場およびポロイダル磁場の強磁場中に置かれ、プラズマを生成する領域を高真空に保つ機器である。トカマク装置ではプラズマが不安定となり、プラズマが移動しながら消滅する、ディスラプションと呼ばれる現象がしばしば発生する。プ



(a) execution (b) estimation

Fig. 4.15 Comparison of parallel performance

ラズマ電流が消滅すると、真空容器には誘導電流が生じ非常に大きな電磁力が作用することにより、複雑な電磁構造連成現象が発生する。

解析条件はプラズマのディスラプション時を想定し、設計条件と等しく、定常なトロイダル磁場中においてプラズマ電流が消滅する状態を模擬する。プラズマを閉じ込めるためのトロイダル磁場は、Fig. 4.16 に示されるように、ドーナッツ方向に沿う中心軸に対称な磁場であり、その強さは中心軸からの距離に反比例する。この磁場を中心軸に沿って流れる線電流が作る磁場として与えることとし、各軸方向の大きさを

$$B_x = -14.4\frac{y}{x^2+y^2} \quad [\mathrm{T}] \tag{4.36}$$

$$B_y = 14.4\frac{x}{x^2+y^2} \quad [\mathrm{T}] \tag{4.37}$$

$$B_z = 0 \quad [\mathrm{T}] \tag{4.38}$$

とする。プラズマのディスラプションは、ドーナッツ方向に流れている 6 [MA] のプラズマ電流が、Fig. 4.17 に示されるように、10 [msec] 間に消滅するものとし、非定常解析の時間きざみ幅は $\Delta t = 0.01$ [ms] とする。

解析には、対称性を考慮して Fig. 4.18 (a) に示す 1/36 セクターモデルを使用し、8 節点アイソパラメトリックシェル要素を用いて要素分割する。ここで、ポート部分などに

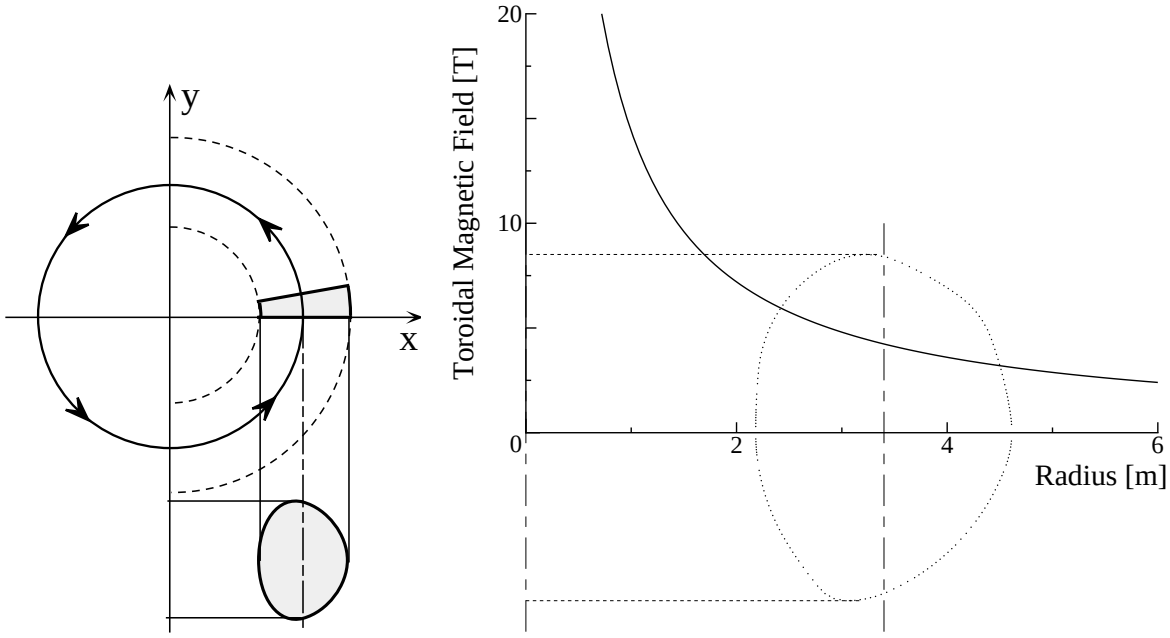


Fig. 4.16 Toroidal magnetic field

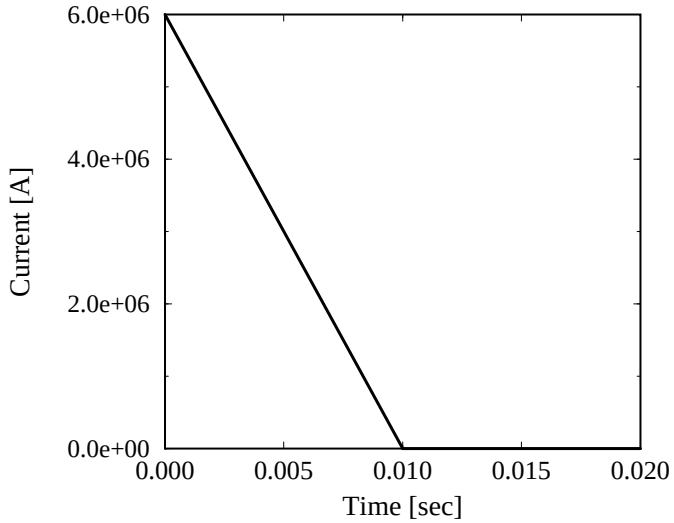
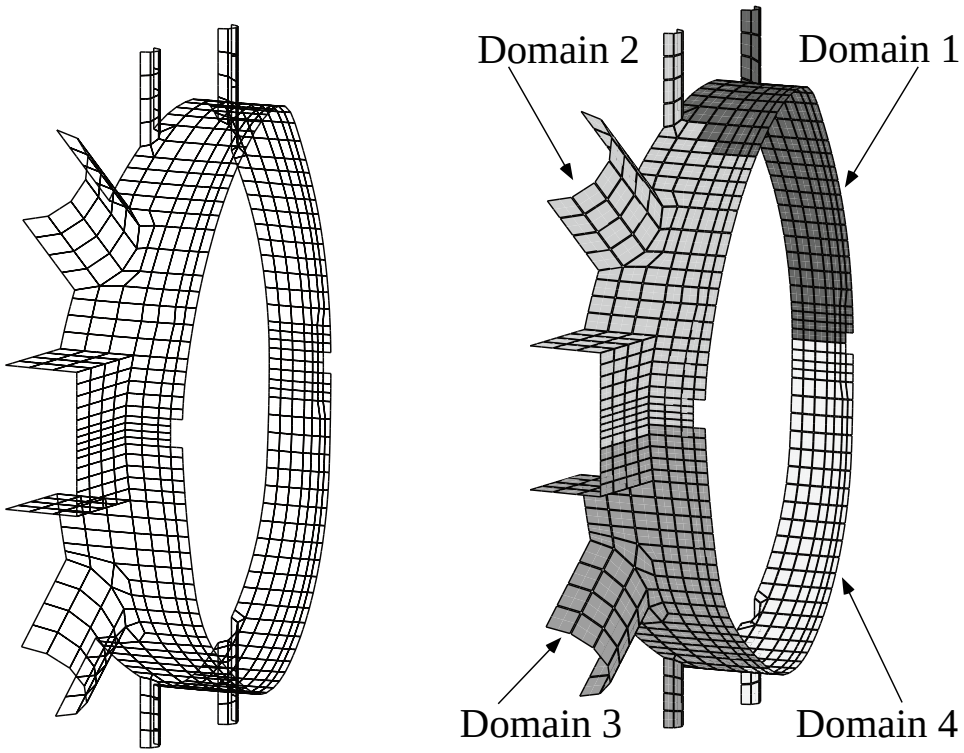


Fig. 4.17 Plasma current

折れ曲がりを含んでいるため、4.2.3 (a) 節で述べたように、構造解析の回転角自由度を全体座標系で定義することとし、1 節点あたりの構造自由度を 6 自由度とする。

並列計算のための領域分割は、Fig. 4.18 (b) のように 4 領域に分割した。このとき、各領域の計算負荷を均等化するため、Table 4.3 に示すように、各領域の要素数が等しくなるように分割した。これにより、1 領域あたりの解析に必要となるメモリ量は、いずれも 80 [MB] 以下となる。また、この問題の総自由度数は 17,673 (構造自由度数 15,098、渦電流自由度数 2,575) であるが、領域分割法により各領域の内部自由度が縮小されるため、CG 法の反復計算の対象となる自由度数は 2,907 (構造境界自由度数 332、渦電流自由度数 2,575) となる。



(a) mesh subdivision (b) 4 domains

Fig. 4.18 Finite element mesh and domains of JT–60U vacuum vessel

4.5.2 並列性能の評価

並列解析の過程における、CG 法の反復 1 回目および 2 回目以降の計算時間と並列性能を、Table 4.4 に示す。ここで、CG 法の反復 1 回目においては、各部分領域の内部自由度を消去するための演算が含まれている。同表には、式 (4.27) に基づいたスレーブ数が 1〜4 台の場合の評価時間、および、実際に 4 台のスレーブを用いて解析した際の結

Table 4.3 Characteristics of JT–60U vacuum vessel model

Number of elements	809			
Number of nodes	2716			
Number of domains	4			
Number of nodes on interface boundary	57			
Structural degrees of freedom	15098			
Interface boundary degrees of freedom	332			
Eddy current degrees of freedom	2575			
Domain number	1	2	3	4
Number of elements	202	203	202	202
Number of nodes	689	702	695	687
Structural degrees of freedom	3779	3975	3915	3761
Interface boundary degrees of freedom	134	234	198	98
Eddy current degrees of freedom	672	702	695	687
Matrix size per domain [MB]	75.6	79.4	78.5	75.1

果を示した。ここで、本解析は 1 台のスレーブによる解析がメモリ容量の制約により実行できないことから、並列効率の算出については評価式を用いて行なうこととする。いずれの場合も並列効率が 90 [%] を越えており、高い並列性能を得ることができた。

なお、時間積分の 1 スッテプあたりには 1500 回程度の共役勾配法の反復が行なわれ、10 ステップの時間積分には、4 台のスレーブを用いて 11.8 時間要した。

4.6 まとめ

電磁構造連成解析のための並列解析法として、領域分割法と領域型共役勾配法を組み合わせた新しい方法を開発した。平板のたわみ連成問題および核融合装置真空容器の実機モデルの解析において並列性能の評価を行なった結果

- 領域分割法と領域型共役勾配法を組み合わせて用いることにより、並列性能および共役勾配法の解の収束性が向上することを確認した。

Table 4.4 Summary of parallel performance

	Number of slaves	Execution time [sec]	Speedup	Parallel efficiency [%]
first CG iteration including matrix condensation				
Estimation	1	94.0	1.000	100.0
	2	47.0	2.000	100.0
	4	23.5	3.999	100.0
Execution	4	24.0	3.917	97.9
second or later CG iteration				
Estimation	1	13.7	1.000	100.0
	2	6.6	1.988	99.4
	4	3.5	3.914	97.9
Execution	4	3.7	3.703	92.6

- 核融合装置真空容器の実機モデルの大規模連成解析に適用し、本並列解析法の実用性を検証した。

参考文献

(4.1) L. R. Turner and T. Q. Hua, “Results for Cantilever Beam Moving Crossed Magnetic Field”, *COMPEL*, **9**, (1990), *pp.* 205–216.

(4.2) T. Takagi, S. Kawamura and J. Tani, “New TEAM Problem 16 Magnetic Damping in Torsional Mode,” *Proc. Third TEAM Workshop*, EUR 14173 EN, (1992), *pp.* 275–279.

(4.3) T. Horie and T. Niho, “Electromagnetic and Mechanical Interaction Analysis of a Thin Shell Structure Vibrating in an Electromagnetic Field,” *Int. J. of Applied Electromagnetics in Materials*, **4**, (1994), *pp.* 363–368.

(4.4) H. Ninomiya, T. Ando, T. Horie, H. Horiike, K. Koizumi el al., “JT–60 Upgrade Device for Confinement and Steady State Studies,” *Plasma Devices and Operations*, **1**, (1990), *pp.* 43–65.

(4.5) O. C. Zienkiewicz, (吉識, 山田 訳), “マトリックス有限要素法”, 培風館, (1990), *pp.* 391–402.

第 5 章

メッシュレス境界節点を用いた高性能領域分割法の開発

5.1 はじめに

前章までの研究成果から、領域分割法は有限要素法の並列化方法として、高効率な並列化を実現できる非常に有効な方法であるといえる。しかし領域分割法は、並列計算のため解析領域を分割する際、領域間において要素分割の連続性を保たなければならないため、要素分割および領域分割が大きな課題となる。また、領域間境界の解を共役勾配法による反復計算で求めることから、大規模問題に適用した場合、境界自由度数の増加にともなって解の収束性が悪化し反復回数が増大するという問題がある。

領域間境界において要素分割が非適合となる問題の解析法としては、Farhat らによって提案された FETI 法 (Finite Element Tearing and Interconnecting)[5.1, 5.2]やモルタル有限要素法[5.3, 5.4]などの方法により並列解析を行なうことができる。しかし、これらの方法はいずれも、領域間境界上における解の連続条件を満足させるため、ラクランジュ未定乗数法を用いて付帯条件を導入している。このため、解析自由度が増加するとともに、解かれる方程式が特異となるといった問題点がある。

一方で、近年、要素を必要としない解析法としてフリーメッシュ法、メッシュレス法の研究が精力的に行なわれている[5.5–5.8]。これらの方法は、要素分割などのさまざまな制約を取り払う有効な方法であるとともに、要素および節点の接続情報を必要としないことから、分散・並列処理との適合性が高いと考えられる。

そこで本章においては、領域分割法において領域間の境界部分にメッシュレスの手法を適用して領域間における要素および節点の接続情報を取り払うことにより、領域分割

や反復回数の増加といった問題に対して、ブレークスルーをもたらす高性能な領域分割法を提案する。これにより、領域間境界において要素分割が不連続となる問題の並列解析が可能となることや、共役勾配法の収束性が向上するといった本手法の有効性を示す。

5.2 高性能領域分割法の概要

5.2.1 解法の概要

問題をいくつかの領域に分割して解く場合、得られた解が正解であるためには、隣接する領域間において

- 応力 (表面力) のつり合い
- 変位の連続性

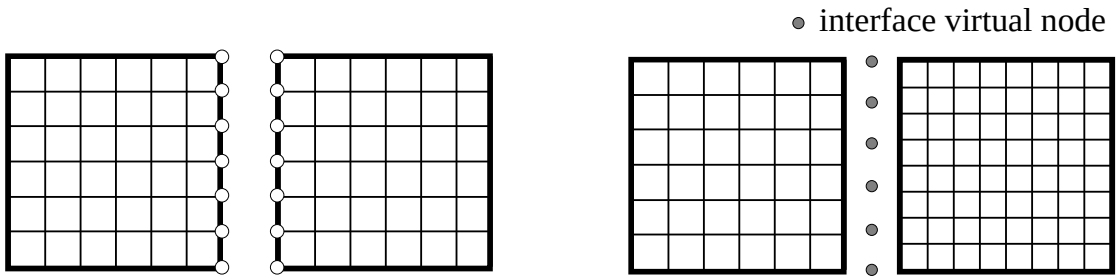
がともに満足されている必要がある。領域分割法においては、隣接する領域間の境界に共通の変位を規定し、これをいわば境界条件として各領域ごとに解析を行ない反力を求め、これが領域間でつり合うように共役勾配法により境界上の変位を修正する。このため、隣接する領域間においては、Fig. 5.1 (a) に示すように、互いに節点を共有し要素分割が連続になるようにしなければならない。

そこで、Fig. 5.1 (b) に示すように、領域間境界に領域の節点とは異なる別の仮想境界節点を配置し、これを用いて領域間境界における表面力および変位の連続条件を満足させる。領域分割法の計算手順と同様に、

1. 仮想節点上に変位を規定する
2. 仮想節点上の変位を各領域の境界節点に変換する
3. 領域ごとに解析を行ない境界節点反力を求める
4. 領域ごとの境界節点における反力を仮想節点に変換する
5. 仮想節点における反力の不つり合い量を求め、これがゼロになるように仮想節点変位を修正する

を繰り返す。

仮想節点と境界節点との間の変換には、エレメント・フリー・ガラーキン法[5.6, 5.8]に用いられている移動最小自乗法[5.9]による内挿関数を用いる。すなわち、仮想境界節点上に規定された変位を領域の境界節点値に変換する場合には、Fig. 5.2 に示すように、境界



(a) conventional DDM　　(b) present method

Fig. 5.1 Domain subdivision

節点 A を評価点 (evaluation point) としてその近傍にある仮想節点から内挿する。これにより、仮想節点郡はその節点間および領域の境界節点との間に接続情報 (connectivity) を一切持つ必要が無い。さらに、各領域の境界節点は領域ごとに完全に独立しているため、隣接する領域の間で要素および節点の配置に関して、領域分割法における要素の適合条件の制約を完全に取り払うことができる。

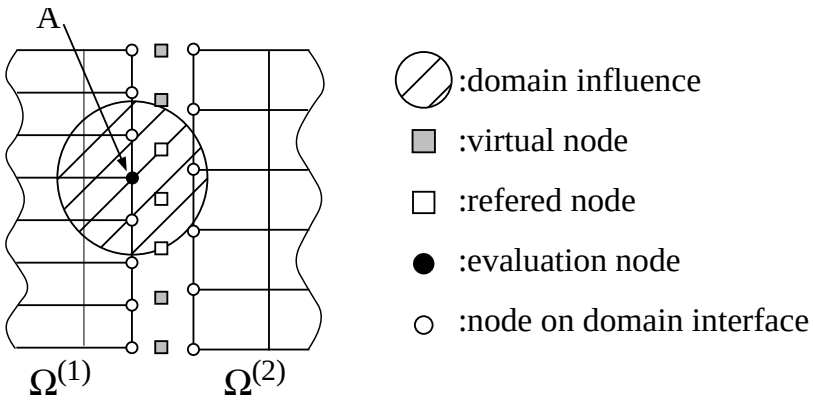


Fig. 5.2 Interpolation method using moving least square method

5.2.2 本手法の利点

本研究で提案する境界仮想節点を用いる方法は、通常の領域分割法に比べて

- 境界節点数よりも仮想節点数を減らすことにより、共役勾配法の収束性および並列性能を向上させることができる

- 仮想節点を用いて領域間を接続するため、領域間境界における要素分割の適合条件を満たす必要がない

といった特徴がある。これにより、本解析法は以下のような利点が得られると考えられる。

(a) 共役勾配法の収束性の向上

境界仮想節点は、節点間における接続情報を持たず、境界への配置やその数に関して一切の制約を受けない。このため、解の精度が低下しない範囲で仮想節点数を領域間境界の節点数よりも少なく配置すると、共役勾配法における反復計算の対象となる自由度数が減少するため、解の収束性が飛躍的に向上する。

(b) 共役勾配法の収束度に応じた仮想節点の追加

共役勾配法の反復計算の過程において、反復の初期段階においては非常に少ない仮想節点を用いて計算を行ない、解の収束度に応じて徐々に仮想節点数を増していくといった方法が可能となる。これは、反復計算の途中で新たに仮想節点を生成し、この節点におけるパラメータを領域間の内挿と同様に移動最小自乗法によって設定するとともに、反復計算の自由度に追加する。こうした方法は、要素のような節点間の接続が規定される場合には適合条件を満足させるといったことが不可能であるが、本手法のようにメッシュレス手法を取り入れることに実現可能となる。

(c) 並列性能の向上

仮想節点と各領域の境界節点との内挿演算は、領域ごとに完全に独立に行なえるため、本手法において新たに追加される演算は、全て並列に計算することができる。さらに、仮想節点を領域境界節点数よりも少なく配置することにより、並列計算における境界データの転送量が減少することから、並列処理性能を向上させることができる。

(d) 領域ごとに独立な要素分割の実現

仮想節点上で領域間のつり合いを満足させるため、領域境界の節点配置および要素分割は領域間で適合し連続したものである必要はない。このため、通常の領域分割法で要素分割および領域分割を行なう際には、Fig. 5.3 に示すように、まず解析領域全体を要素分割し、この要素分割にしたがって領域への分割を行なわなければならない。

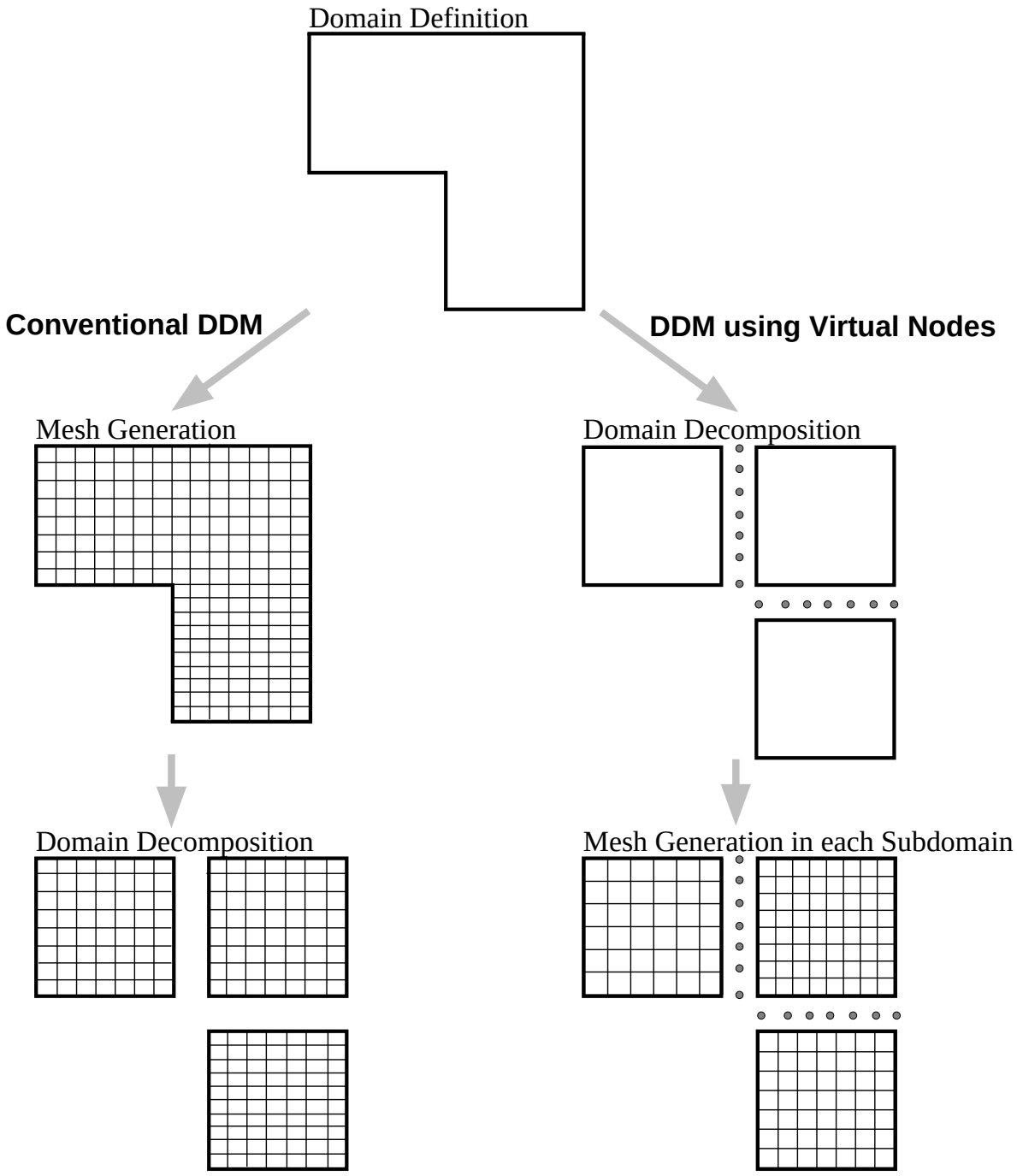


Fig. 5.3 Mesh generation for domain decomposition method

しかし、本解析法においては、全体の解析領域を部分領域に分割し、領域ごとに独立に要素分割を行なうことができる。したがって、領域分割法のためのメッシュ生成および領域分割における

- 領域の自由度数を等しくする
- 領域間の境界自由度数が最小となるように領域分割を行なう

といった厳しい制約条件を取り払うことができる。

(e) 並列アダプティブ有限要素法への応用

有限要素法によって得られる解の精度は要素分割の形態に大きく依存するが、得られた有限要素解から誤差を事後推定し、要素分割を自動的に改善するアダプティブ有限要素法が開発されている。アダブティブ法には、大別して

r 法 全体の要素数は不変で、節点位置を移動させる

h 法 誤差が大きい部分の要素を細分化する

p 法 誤差が大きい部分の要素の内挿関数の次数を上げる

という 3 つの方法がある[5.10–5.12]。

領域分割法にこのアダプティブ法を適用する場合、領域ごとに節点配置や要素分割を変更すると、領域間の境界上でメッシュの不連続性が生じることとなり、解析が不可能となる。したがって、要素の細分化や節点の移動は全体領域に対して行なったのち、領域の分割をやり直さなければならないため、領域分割法にアダプティブ法を適用しても並列アダプティブ有限要素法の実現は困難である。

しかし、本解析法の場合には、領域間におけるメッシュの連続性は全く考慮する必要がないため、Fig. 5.4 に示すように、領域ごとに独立にアダプティブ法を実行しながら解析を行なうことができる。さらに、Fig. 5.5 に示すように、仮想節点に対する共役勾配法の反復ループ中にアダプティブ法の手続きを挿入し、反復の初期段階においては、各領域の解析を粗い要素分割を用いて行ない解の収束を加速させ、収束が進むにつれて、徐々にアダプティブ法にしたがってメッシュ分割を細分化する。これにより、共役勾配法の収束性が向上するとともに、メッシュ分割も解析と同時に並列化されることから、メッシュ分割および領域分割を含めたトータルの解析時間を大幅に短縮することができる。

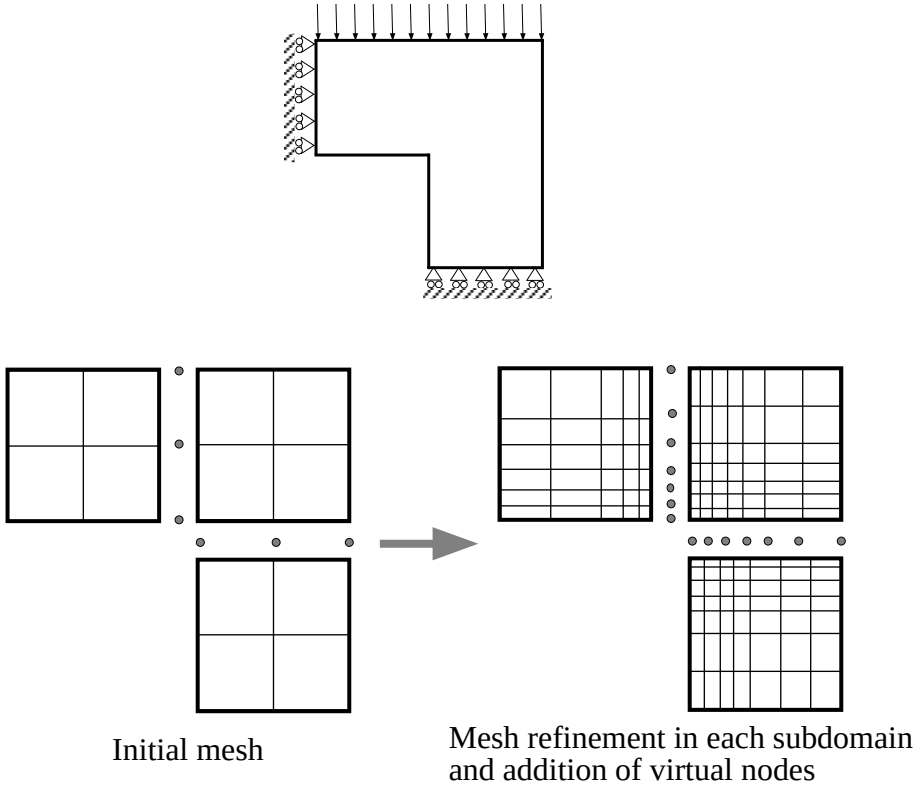


Fig. 5.4 Adaptive mesh refinement for domain decomposition method

5.3 移動最小自乗法による領域間の接続方法

5.3.1 各領域における内部自由度の消去

領域分割法で、領域 (k) における剛性方程式を内部自由度 (添字 i) と境界自由度 (添字 b) に分けて表すと

$$\begin{bmatrix} \mathbf{K}_{ii}^{(k)} & \mathbf{K}_{ib}^{(k)} \\ \mathbf{K}_{ib}^{T(k)} & \mathbf{K}_{bb}^{(k)} \end{bmatrix} \begin{Bmatrix} \mathrm{u}^{(k)} \\ \mu^{(k)} \end{Bmatrix} = \begin{Bmatrix} \mathrm{f}^{(k)} \\ \tau^{(k)} \end{Bmatrix} \tag{5.1}$$

のようになる。ここで、$\left\{\mathrm{u}^{(k)}\right\}$、$\left\{\mu^{(k)}\right\}$ はそれぞれ内部自由度および境界自由度の変位である。

2.2.4 節で述べたように、領域分割法の手順にしたがって内部自由度の変位 $\left\{\mathrm{u}^{(k)}\right\}$ は、

$$\left\{\mathrm{u}^{(k)}\right\} = \left[\mathbf{K}_{ii}^{(k)}\right]^{-1} \left(\left\{\mathrm{f}^{(k)}\right\} - \left[\mathbf{K}_{ib}^{(k)}\right] \left\{\mu^{(k)}\right\}\right) \tag{5.2}$$

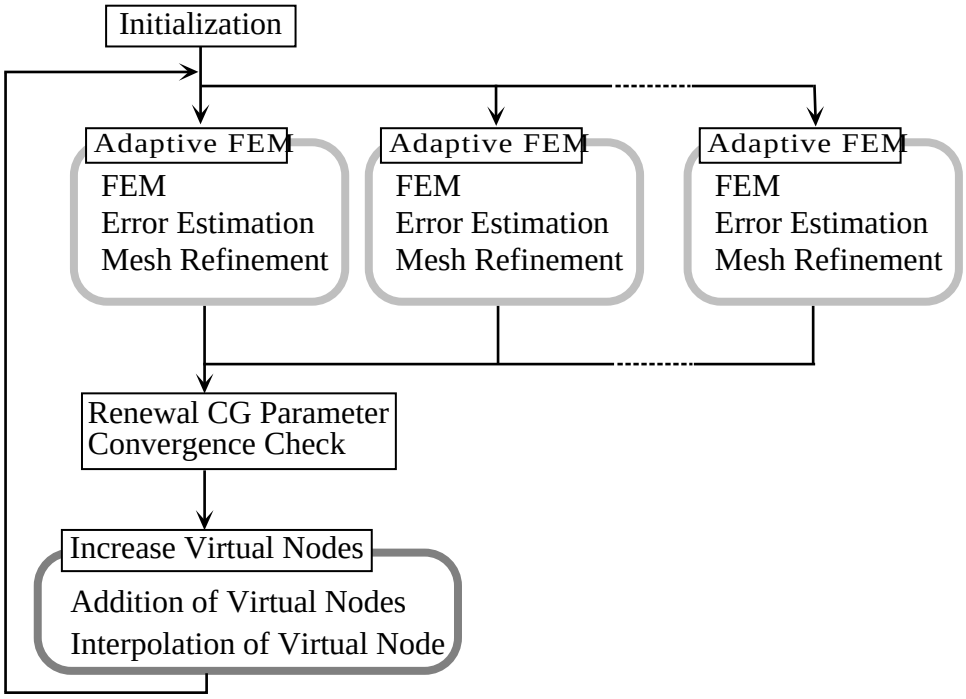


Fig. 5.5 Flow chart of domain decomposition method with adaptive mesh refinement

と表されるから、これを用いて境界自由度に関する式

$$\left(\left[\mathbf{K}_{bb}^{(k)}\right]-\left[\mathbf{K}_{ib}^{(k)}\right]^{T}\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left[\mathbf{K}_{ib}^{(k)}\right]\right)\left\{\mu^{(k)}\right\}=\left\{\tau^{(k)}\right\}-\left[\mathbf{K}_{ib}^{(k)}\right]^{T}\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left\{\mathrm{f}^{(k)}\right\} \quad (5.3)$$

が得られる。通常の領域分割法では、全領域についての式 (5.3) に対して共役勾配法を適用して境界自由度の変位 $\left\{\mu^{(k)}\right\}$ を求める。

本解析法においては、境界仮想節点に変位 $\left\{\mu_v^{(k)}\right\}$ を設定し、これを式 (5.3) における境界節点変位 $\left\{\mu^{(k)}\right\}$ に内挿して領域分割法と同様の演算を行なったのち、その結果を再び仮想節点に内挿して共役勾配法の反復計算を行なう。

5.3.2 移動最小自乗法に基づく境界節点の内挿

ここでは、移動最小自乗法を用いて仮想節点に規定された変位 $\{\mu_v\}$ を境界節点に内挿する方法を示す。

Fig. 5.2 に示した境界節点 A (x, y) における変位を、1 次あるいは 2 次といった多項式を用いて

$$\mu^h=\sum_{j=1}^{m}p_j(x,y)a_j(x,y)$$

$$= \{\mathrm{p}(x,y)\}^T \{\mathrm{a}(x,y)\} \tag{5.4}$$

のように近似する。ここで、p_j は空間座標 (x,y) を含む多項式、a_j は未定係数である。また、m は多項式展開に用いた項数であり、二次元問題において 1 次多項式を用いて近似する場合、多項式 $\{\mathrm{p}(x,y)\}$ は

$$\{\mathrm{p}(x,y)\} = \begin{Bmatrix} 1 \\ x \\ y \end{Bmatrix} \tag{5.5}$$

となり、$m = 3$ となる。また、2 次多項式を用いる場合、$\{\mathrm{p}(x,y)\}$ は

$$\{\mathrm{p}(x,y)\} = \begin{Bmatrix} 1 \\ x \\ y \\ x^2 \\ y^2 \\ xy \end{Bmatrix} \tag{5.6}$$

となり、$m = 6$ となる。

未定係数 $\{\mathrm{a}(x,y)\}$ については、次の関数

$$\begin{aligned} J &= \sum_{I=1}^{n} w\left(\{\mathrm{x}\} - \left\{\mathrm{x}_I^{(k)}\right\}\right)\left(\mu_A^{(k)}(\{\mathrm{x}\}) - \mu_I^{(k)}\right)^2 \\ &= \sum_{I=1}^{n} w\left(\{\mathrm{x}\} - \left\{\mathrm{x}_I^{(k)}\right\}\right)\left(\{\mathrm{p}(x,y)\}^T \{\mathrm{a}(x,y)\} - \mu_I^{(k)}\right)^2 \end{aligned} \tag{5.7}$$

を最小化させるように決定する。ここで、$\{\mathrm{x}\}^T = [x,y]$ であり、n は境界節点近傍にある内挿に用いる仮想節点数、$\left\{\mathrm{x}_I^{(k)}\right\}$、$\mu_I^{(k)}$ はそれぞれ仮想節点 I の節点座標ベクトル $[x_I, y_I]^T$ および節点変位値である。

また、$w(\mathbf{r})$ は内挿に用いる仮想節点に対してそれぞれ重みづけを行なうための重み関数であり、$w(\mathbf{r})$ は $||\mathbf{r}||_2 = 0$ で 1 を取り、$||\mathbf{r}||_2$ が増加するにつれて滑らかに減少する連続関数である。ここで、$||\mathbf{r}||$ は仮想節点との距離 r

$$r = ||\mathbf{r}||_2 = \sqrt{(x - x_I)^2 + (y - y_I)^2} \tag{5.8}$$

である。エレメント・フリー・ガラーキン法における移動最小自乗法を用いた内挿関数において、Belytschko らは重み関数として Fig. 5.6 に示す以下のような関数を用いている[(5.6)]。

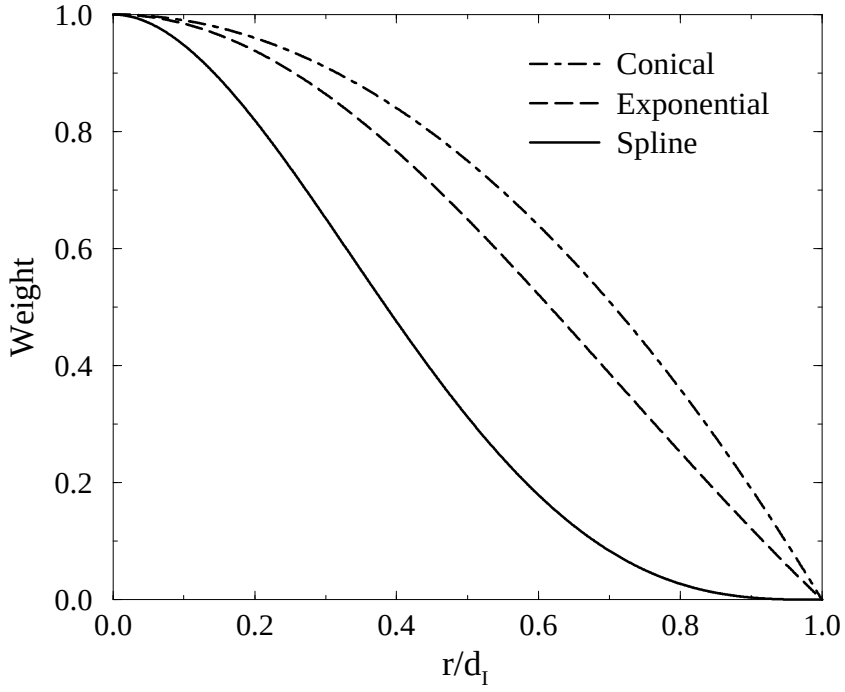


Fig. 5.6 Weight functions

- Conical weight function

$$w_I(r) = 1.0 - \left(\frac{r}{d_I}\right)^2 \qquad (0 \le r \le d_I) \tag{5.9}$$

- Exponential weight function

$$w_I(r) = \frac{e^{-(r/c)^2} - e^{-(d_I/c)^2}}{1 - e^{-(d_I/c)^2}} \tag{5.10}$$

- Quartic Spline function

$$w_I(r) = 1.0 - 6.0\left(\frac{r}{d_I}\right)^2 + 8.0\left(\frac{r}{d_I}\right)^3 - 3.0\left(\frac{r}{d_I}\right)^4 \qquad (0 \le r \le d_I) \tag{5.11}$$

移動最小自乗法の内挿精度は、重み関数の選択のほか、内挿を行なう際に用いる仮想節点を決定する影響半径 (domain influence) d_I によって大きく変化する。

式 (5.7) は

$$J = \left\{[\mathbf{D}]^T \{\mathrm{a}\} - \left\{\mu_{Av}^{(k)}\right\}\right\}^T [\mathbf{W}] \left\{[\mathbf{D}]^T \{\mathrm{a}\} - \left\{\mu_{Av}^{(k)}\right\}\right\} \tag{5.12}$$

と表される。ここに、

$$[\mathbf{D}] = [\{\mathrm{p}(x_1, y_1)\}, \{\mathrm{p}(x_2, y_2)\}, \cdots, \{\mathrm{p}(x_n, y_n)\}] \tag{5.13}$$

$$[\mathbf{W}] = \mathrm{diag}\left[w(\{\mathrm{x}\}-\{\mathrm{x}_1\}), w(\{\mathrm{x}\}-\{\mathrm{x}_2\}), \cdots, w(\{\mathrm{x}\}-\{\mathrm{x}_n\})\right]$$

$$= \begin{bmatrix} w(\{\mathrm{x}\}-\{\mathrm{x}_1\}) & 0 & \cdots & 0 \\ 0 & w(\{\mathrm{x}\}-\{\mathrm{x}_2\}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w(\{\mathrm{x}\}-\{\mathrm{x}_n\}) \end{bmatrix} \tag{5.14}$$

であり、$\left\{\mu_{Av}^{(k)}\right\}$ は内挿に用いる仮想節点の変位ベクトル

$$\left\{\mu_{Av}^{(k)}\right\} = \begin{Bmatrix} \mu_{A1}^{(k)} \\ \mu_{A2}^{(k)} \\ \vdots \\ \mu_{An}^{(k)} \end{Bmatrix} \tag{5.15}$$

である。式 (5.12) の未定係数 $\{\mathrm{a}\}$ についての停留条件 (最小化条件) は、

$$\delta J = 2\left\{\delta\mathrm{a}\right\}^T \left([\mathbf{D}][\mathbf{W}][\mathbf{D}]^T\{\mathrm{a}\} - [\mathbf{D}][\mathbf{W}]\left\{\mu_{Av}^{(k)}\right\}\right) = 0 \tag{5.16}$$

のように表されるから、これを $\{\mathrm{a}\}$ について解くと、

$$\{\mathrm{a}\} = \left([\mathbf{D}][\mathbf{W}][\mathbf{D}]^T\right)^{-1}[\mathbf{D}][\mathbf{W}]\left\{\mu_v^{(k)}\right\}$$

$$= [\mathbf{A}]^{-1}[\mathbf{B}]\left\{\mu_{Av}^{(k)}\right\} \tag{5.17}$$

が得られる。ここで、$[\mathbf{A}]$ および $[\mathbf{B}]$ は

$$[\mathbf{A}] = [\mathbf{D}][\mathbf{W}][\mathbf{D}]^T \tag{5.18}$$

$$[\mathbf{B}] = [\mathbf{D}][\mathbf{W}] \tag{5.19}$$

とおいたが、それぞれ $m \times m$ および $m \times n$ のマトリックスである。

こうして得られた未定係数 $\{a\}$ を式 (5.4) に代入すると、境界節点 A における変位 $\mu_A^{(k)}$ は、その近傍に位置する仮想節点の変位 $\left\{\mu_{Av}^{(k)}\right\}$ を用いて

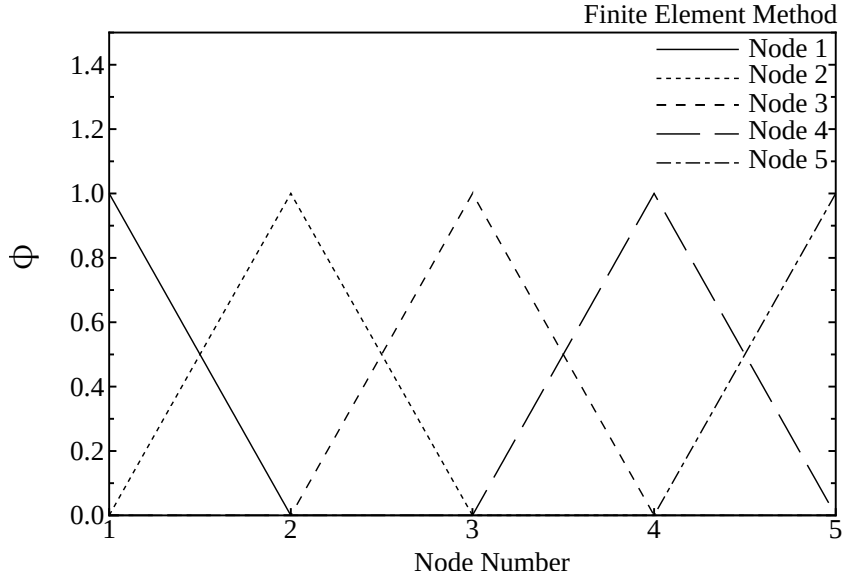
$$\mu_A^{(k)} = \left\{\mathrm{p}(x,y)\right\}^T[\mathbf{A}]^{-1}[\mathbf{B}]\left\{\mu_{Av}^{(k)}\right\}$$

$$= \left\{\phi_A(x,y)\right\}^T\left\{\mu_{Av}^{(k)}\right\} \tag{5.20}$$

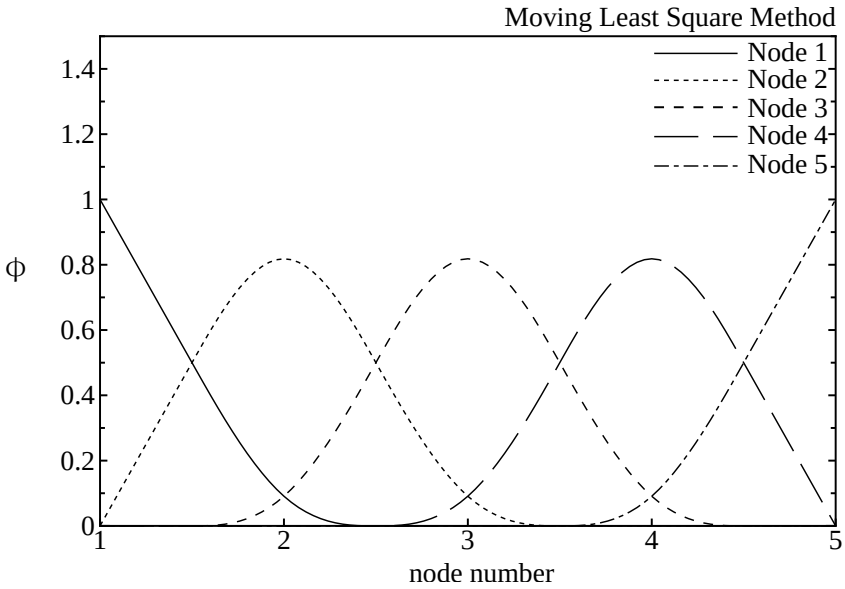
と内挿される。ここで、$\left\{\phi_A^{(k)}(x,y)\right\}$ は移動最小自乗法による内挿関数であり、

$$\left\{\phi_A^{(k)}(x,y)\right\}^T = \left\{\mathrm{p}(x,y)\right\}^T[\mathbf{A}]^{-1}[\mathbf{B}] \tag{5.21}$$

と表される。この内挿関数は、1 次多項式を用いて近似した場合においても、Fig. 5.7 に示すように、有限要素法で用いられる内挿関数よりも高精度な内挿を行なうことができる。



(a) Finite element method



(b) Moving least square method

Fig. 5.7 Comparison of interpolation function

5.3.3 境界仮想節点への共役勾配法の適用

前節で述べた内挿を、各領域において境界上の全ての節点について行なうと、領域 (k) における境界変位ベクトル $\left\{\mu^{(k)}\right\}$ は

$$\left\{\mu^{(k)}\right\} = \left[\mathbf{T}^{(k)}\right]\left\{\mu_v^{(k)}\right\} \tag{5.22}$$

と表される。ここで、マトリックス $\left[\mathbf{T}^{(k)}\right]$ は内挿関数 $\left\{\phi^{(k)}\right\}$ を用いて

$$\left[\mathbf{T}^{(k)}\right]^T = \begin{bmatrix} \left\{\phi_1^{(k)}\right\} & \left\{\phi_2^{(k)}\right\} & \cdots \end{bmatrix} \tag{5.23}$$

と表される。この変換マトリックスは、境界節点数と仮想節点数は異なるため正方マトリックスでないことに注意しなければならない。

式 (5.22) を各領域における内部自由度の式 (5.2) に代入すると、

$$\begin{aligned} \left\{\mathrm{u}^{(k)}\right\} &= \left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left(\left\{\mathrm{f}^{(k)}\right\} - \left[\mathbf{K}_{ib}^{(k)}\right]\left\{\mu^{(k)}\right\}\right) \\ &= \left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left(\left\{\mathrm{f}^{(k)}\right\} - \left[\mathbf{K}_{ib}^{(k)}\right]\left[\mathbf{T}^{(k)}\right]\left\{\mu_v^{(k)}\right\}\right) \end{aligned} \tag{5.24}$$

が得られ、仮想節点の変位を用いて領域ごとに解析を行なうことができる。これを用いて、各領域における境界自由度の式 (5.3) は

$$\begin{aligned} &\left(\left[\mathbf{K}_{bb}^{(k)}\right] - \left[\mathbf{K}_{ib}^{(k)}\right]^T\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left[\mathbf{K}_{ib}^{(k)}\right]\right)\left[\mathbf{T}^{(k)}\right]\left\{\mu_v^{(k)}\right\} \\ &= \left\{\tau^{(k)}\right\} - \left[\mathbf{K}_{ib}^{(k)}\right]\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left\{\mathrm{f}^{(k)}\right\} \end{aligned} \tag{5.25}$$

と表され、仮想節点の変位に対する領域ごとの境界における節点反力を導くことができる。式 (5.25) の両辺に左から $\left[\mathbf{T}^{(k)}\right]^T$ を乗じて、境界節点反力を仮想節点に内挿すると、

$$\begin{aligned} &\left[\mathbf{T}^{(k)}\right]^T\left(\left[\mathbf{K}_{bb}^{(k)}\right] - \left[\mathbf{K}_{ib}^{(k)}\right]^T\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left[\mathbf{K}_{ib}^{(k)}\right]\right)\left[\mathbf{T}^{(k)}\right]\left\{\mu_v^{(k)}\right\} \\ &= \left[\mathbf{T}^{(k)}\right]^T\left(\left\{\tau^{(k)}\right\} - \left[\mathbf{K}_{ib}^{(k)}\right]^T\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left\{\mathrm{f}^{(k)}\right\}\right) \\ &= \left\{\tau_v^{(k)}\right\} \end{aligned} \tag{5.26}$$

となり、領域 (k) における仮想節点反力を求めることができる。

領域ごとに得られた式 (5.26) を全領域について全体化

$$\sum_k \left[\mathbf{T}^{(k)}\right]^T\left(\left[\mathbf{K}_{bb}^{(k)}\right] - \left[\mathbf{K}_{ib}^{(k)}\right]^T\left[\mathbf{K}_{ii}^{(k)}\right]^{-1}\left[\mathbf{K}_{ib}^{(k)}\right]\right)\left[\mathbf{T}^{(k)}\right]\{\mu_v\} = \sum_k \left\{\tau_v^{(k)}\right\} \tag{5.27}$$

し、これに対して共役勾配法を適用して $\{\mu_v\}$ を求める。ここで、係数マトリックス $\sum_k \left[\mathbf{T}^{(k)}\right]^T \left(\left[\mathbf{K}_{bb}^{(k)}\right] - \left[\mathbf{K}_{ib}^{(k)}\right]^T \left[\mathbf{K}_{ii}^{(k)}\right]^{-1} \left[\mathbf{K}_{ib}^{(k)}\right]\right) \left[\mathbf{T}^{(k)}\right]$ は仮想節点の総自由度数サイズの正値・対称マトリックスである。

実際の並列解析においては、式 (5.27) のように係数マトリックスを全体系に足し合わせることは行なわず、式 (5.26) により各領域ごとにマトリックス・ベクトル演算を行なったものを集めて全体系のベクトルにまとめ、これを用いて共役勾配法のパラメータ演算を行なう。

5.3.4 クラスタ・システムへの実装

仮想節点を用いた領域分割法は、Fig. 5.8 に示すように、

1. 仮想節点から境界節点への変位の内挿
2. 領域内の解析
3. 境界節点反力の仮想節点への内挿

の演算は、全て領域ごとに独立に行なうことができる。また、領域ごとに計算された仮想節点の反力を集め、全体系のベクトルにまとめると、共役勾配法のパラメータ演算を行なうことができる。

クラスタ・システムへの実装は、Fig. 5.9 に示すように、通常の領域分割法と同様に領域ごとの計算をスレーブ・マシンに割り当てて並列に解析を行ない、仮想節点に対する共役勾配法の演算はマスター・マシン上で行なう。すなわち、マスター・マシン上で計算された仮想節点の変位に対する探索方向ベクトルを全てのスレーブ・マシンに転送する。各スレーブ・マシンは、演算を担当する領域について仮想節点のデータを領域境界に内挿して解析を行ない、その結果を再び仮想節点に内挿したのち、マスター・マシンへ送り返す。なお、並列解析の高速化をはかるため、各領域の剛性マトリックスに対する前進消去および境界データの内挿を行なうための変換マトリックスの作成は反復の 1 回目のみに行ない、これらのマトリックスをそのままスレーブ・マシン上に保存して、反復 2 回目以降はこれらを用いて演算を行なうこととする。

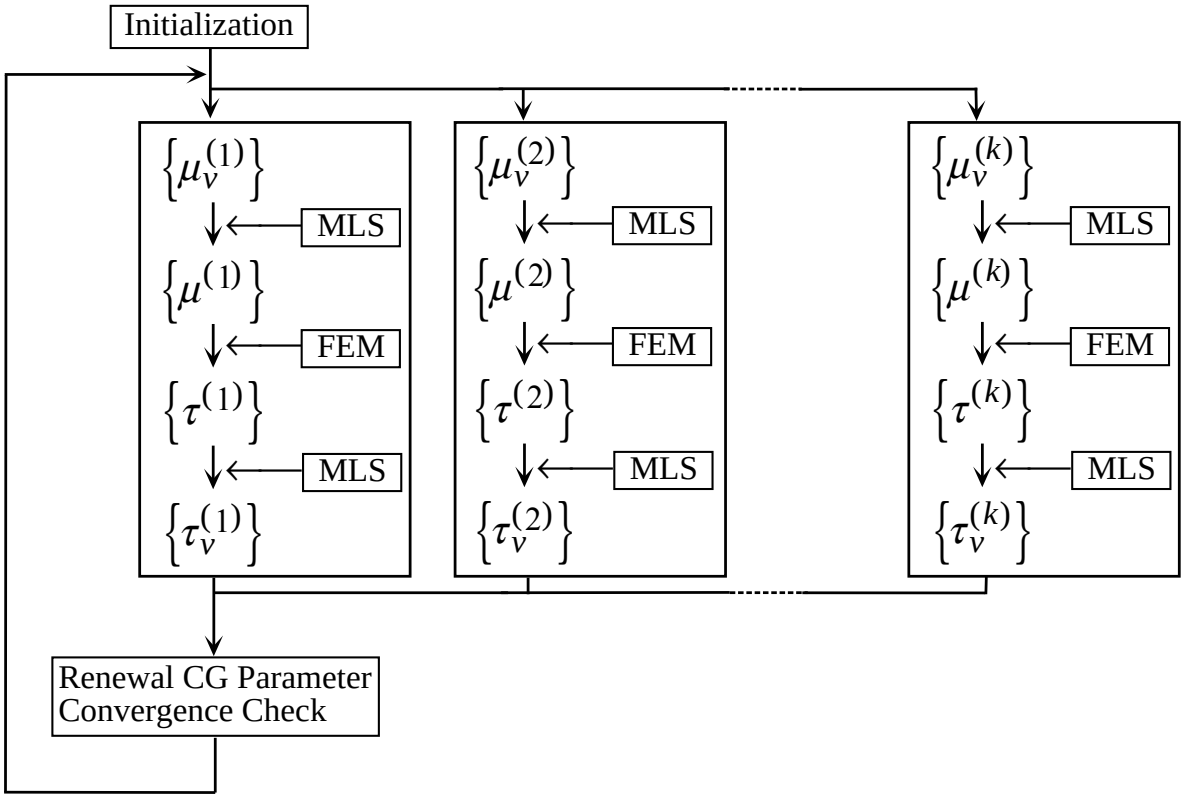


Fig. 5.8 Flowchart of domain decomposition method using virtual nodes

5.4 本解析法の検証

5.4.1 領域間境界において要素分割が適合する問題の解析

本解析法および解析コードの妥当性を検証するため、まず領域間境界において要素分割が連続している適合領域問題の解析を行ない、解の収束性および精度を通常の領域分割法と比較する。

(a) 一様引張り荷重を受ける平板の解析

Fig. 5.10 に示す一様引張り荷重を受ける正方形板を 2 つの領域に分割して解析する。各領域は 4 節点アイソパラメトリック要素により 50×100 要素に等分割し、領域間境界上に境界節点と一致するように 101 個の仮想節点を配置した場合と、仮想節点数を 51 個に減らして等間隔に配置した 2 つのケースについて、通常の領域分割法と比較する。

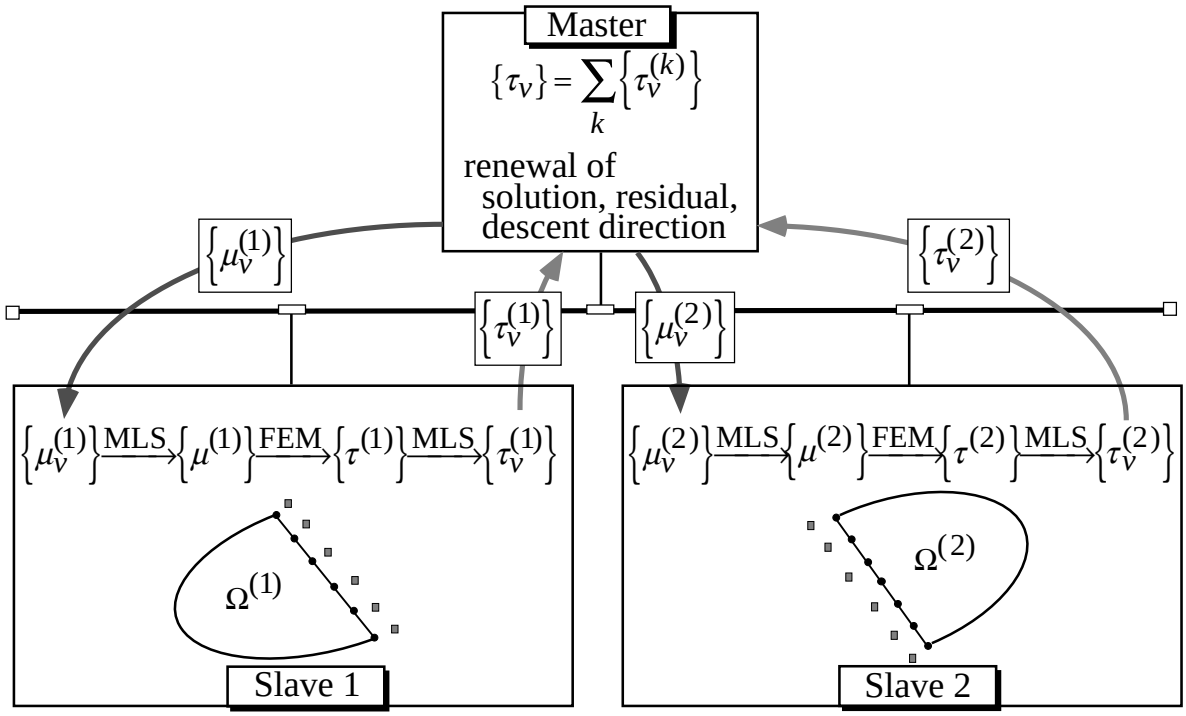


Fig. 5.9 Analysis system on the workstation cluster

なお、移動最小自乗法の近似次数は 1 次関数とし、重み関数には 4 次スプライン関数

$$w(r) = 1.0 - 6.0\left(\frac{r}{d_I}\right)^2 + 8.0\left(\frac{r}{d_I}\right)^3 - 3.0\left(\frac{r}{d_I}\right)^4 \tag{5.28}$$

を用いる。

Fig. 5.11 に共役勾配法の反復計算おける解の収束の様子を領域分割法と比較して示す。同図 (a) より、仮想節点数が異なるいずれのケースにおいても反復が進むにつれ残差が減少し、領域分割法に比べて良好な収束性が見られる。また、仮想節点数が少ない場合の方が反復計算の自由度数が減少するため、収束性は速い。反復過程における Fig. 5.10 A 点の仮想節点変位を厳密解により正規化した値 (反復解が正解と一致すると 1.0 となる値) は、Fig. 5.11 (b) に示すように反復が進むにつれ正解に収束している。

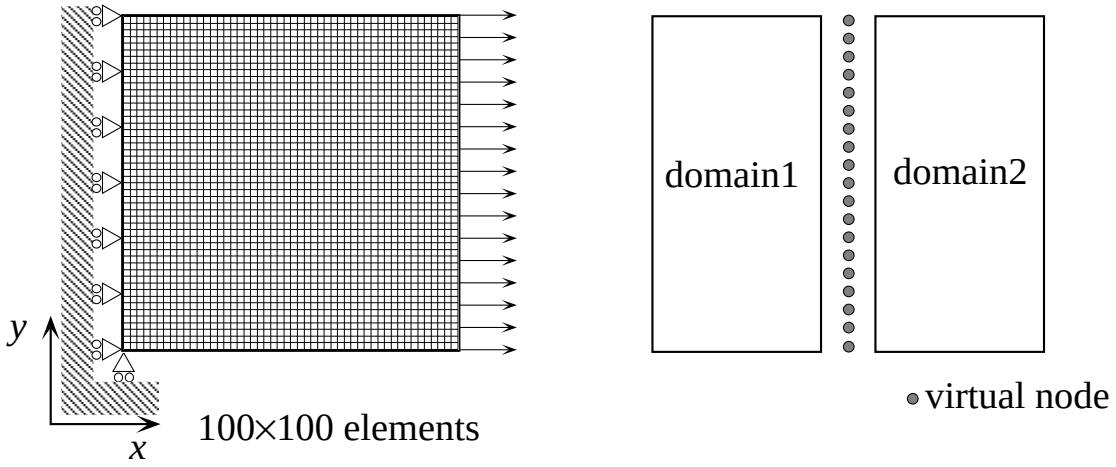
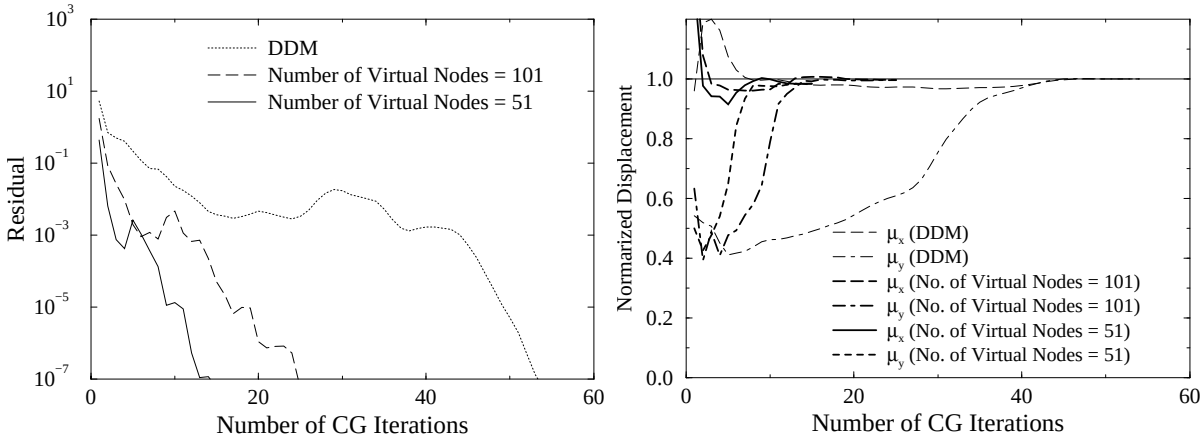


Fig. 5.10 A square plate subjected to a uniform tensile load



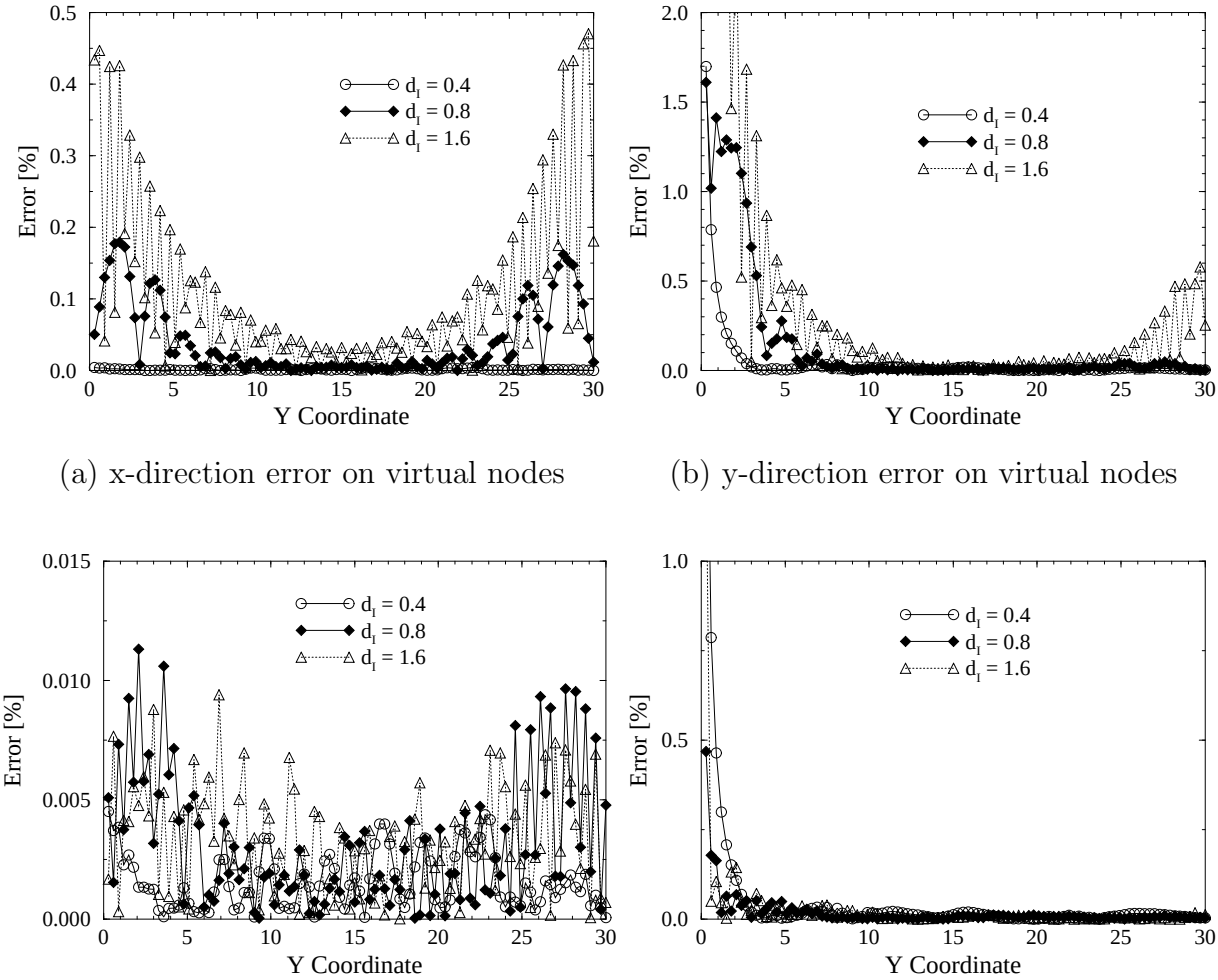
(a) residual (b) displacement

Fig. 5.11 Condition of CG convergence

収束後の境界上における解の精度を、仮想節点を 101 個配置した場合を Fig. 5.12 に、51 個配置した場合を Fig. 5.13 にそれぞれ示す。いずれのグラフにおいても、横軸は Fig. 5.10 に示した A 点を y 軸原点として B 点に向かって取った距離に対応し、縦軸にそれぞれの節点において得られた変位 μ を厳密解 μ_{exact} により誤差を

$$\text{Error } [\%] = \left| \frac{\mu_{\text{exact}} - \mu}{\mu_{\text{exact}}} \right| \times 100 \ [\%] \tag{5.29}$$

としてプロットした。



(a) x-direction error on virtual nodes

(b) y-direction error on virtual nodes

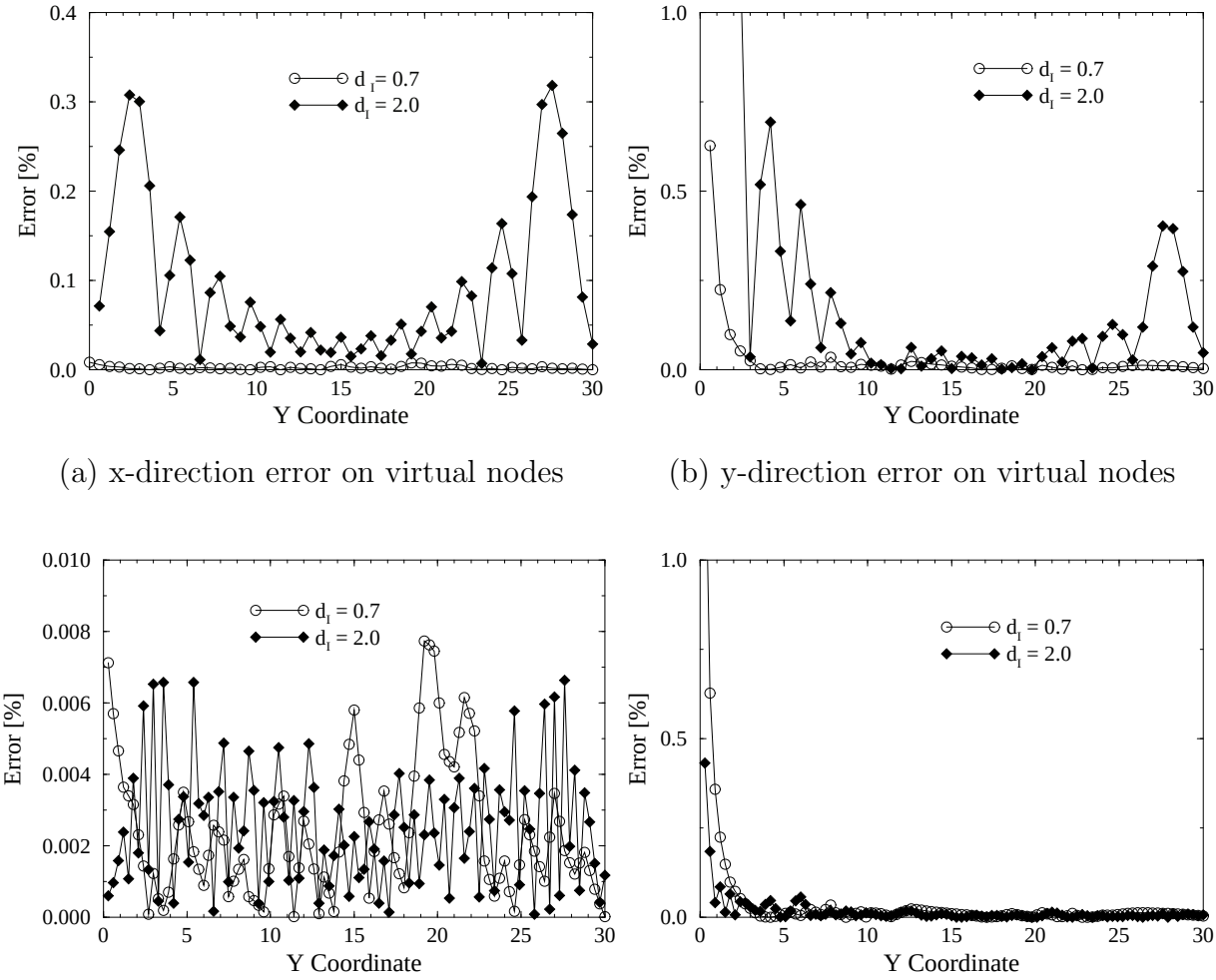
(c) x-direction error on boundary nodes

(d) y-direction error on boundary nodes

Fig. 5.12 Displacement error for 2-domain problem using 101 virtual nodes

収束後の仮想節点における X 方向および Y 方向の変位誤差を Fig. 5.12 (a), (b) に、境界節点における X 方向および Y 方向の変位誤差を同図 (c), (d) に示す。ここで、移動最小自乗法における影響半径 d_I を、仮想節点が 101 個の場合には 0.4, 0.8, 1.6 の 3 ケース、仮想節点が 51 個の場合には 0.7, 2.0 の 2 ケース設定した。影響半径を変えると、1 つの境界節点の内挿に選択される仮想節点数が Table 5.1 に示すように変化し、これを大きく取るほど 1 つの境界節点の内挿に用いる仮想節点数が増加する。

影響半径を大きく設定すると、Fig. 5.12 (a), (b) に示すように、仮想節点における変位誤差は大きくなるが、これを用いて内挿される境界節点の変位誤差は同図 (c), (d) のようにいずれも非常に小さい。なお、同図 (b), (d) に示した Y 方向の変位誤差において、



(a) x-direction error on virtual nodes　(b) y-direction error on virtual nodes

(c) x-direction error on boundary nodes　(b) y-direction error on boundary nodes

Fig. 5.13 Displacement error for 2-domain problem using 51 virtual nodes

A 点近傍の誤差は大きい値となっているが、これは変位誤差の算出において式 (5.29) の厳密解 μ_{exact} が 0 となるため、相対的に大きくなるためである。

境界上に配置する仮想節点数を 51 個に減らしても Fig. 5.13 のように良好な精度で解析できることが確認される。

次に、同じ正方形板を Fig. 5.14 に示すように 10×10 の 100 領域に格子上に分割し、辺 AB 上に一様引張り荷重を与える。仮想節点は、境界節点と等しく 1727 個配置した場合と、297 個に減らして配置した場合の 2 ケース設定する。なお、移動最小自乗法の影響半径は、仮想節点数が 1727 の場合には $d_I = 0.4$ とし 1 つの境界節点の内挿に近傍の 3 個の仮想節点を用い、仮想節点数を 297 に減らした場合には $d_I = 3.0$ として 4〜5

Table 5.1 Number of virtual nodes for interpolation

Total number of virtual nodes	Size of domain influence d_I	Selected number of virtual nodes
101	0.4	2〜3
	0.8	3〜5
	1.6	5〜11
51	0.7	2〜3
	2.0	3〜6

個の仮想節点を用いて内挿する。

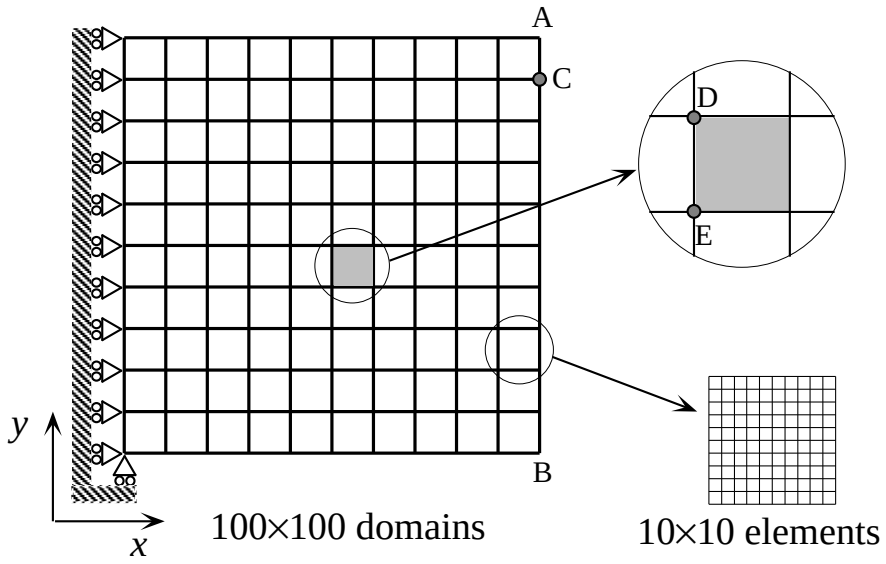
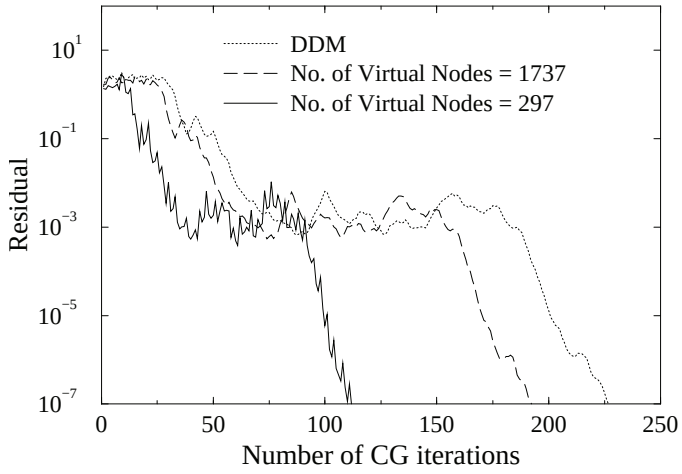
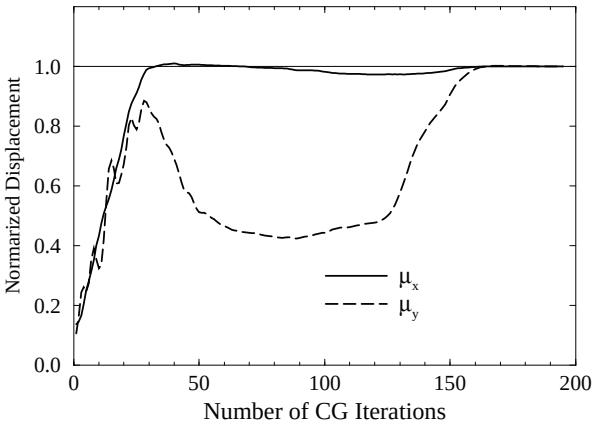


Fig. 5.14 A square plate subjected to a uniform tensile load

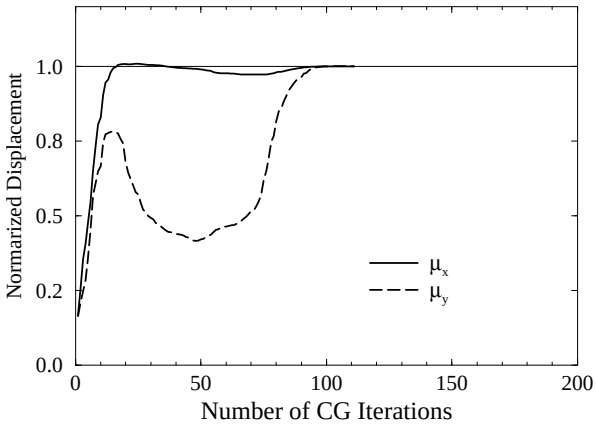
共役勾配法の収束状況は、Fig. 5.15 に示すように、領域分割法に比べ良好な収束性となっており、この場合も仮想節点数を減らすことにより収束は速くなる。



(a) residual



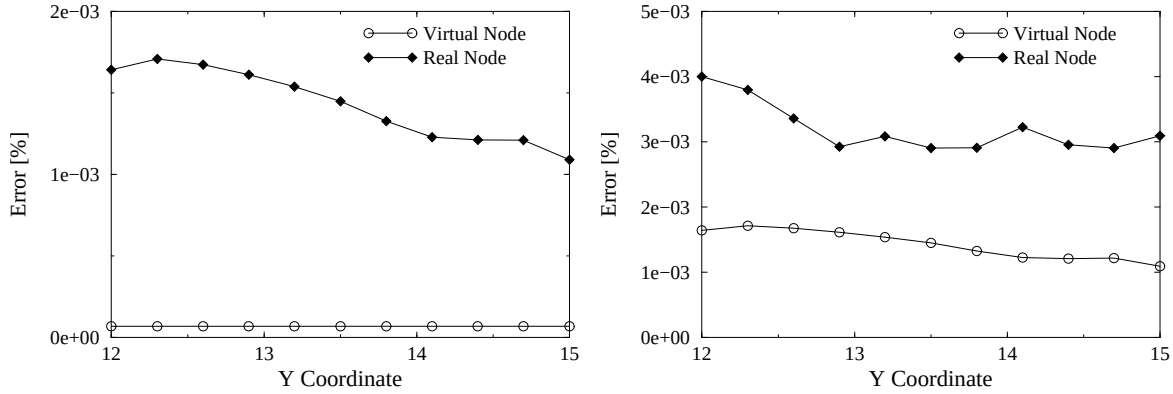
(b) displacement (1737 virtual nodes)



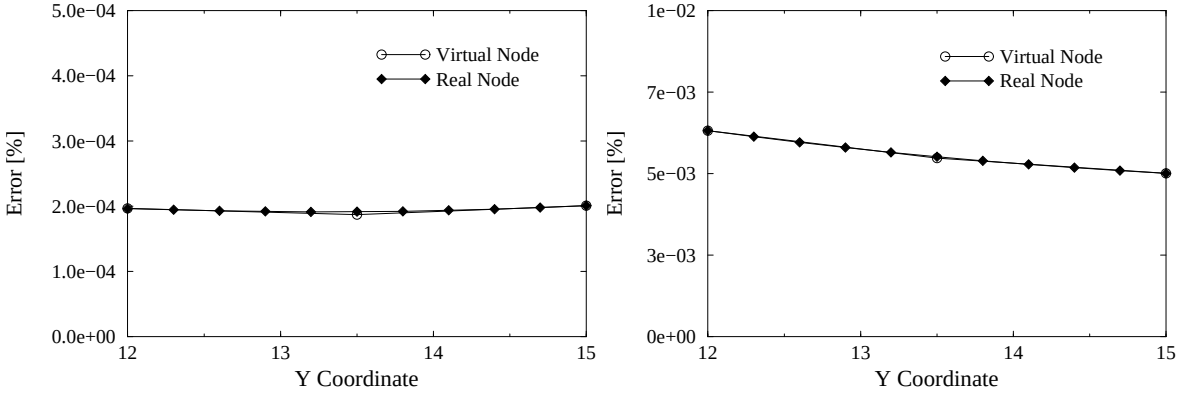
(c) displacement (297 virtual nodes)

Fig. 5.15 Condition of CG convergence for square domain problem

収束後の平板中心付近の領域境界 (Fig. 5.14 に示す境界 DE 上) における変位誤差は、Fig. 5.16 に示すように、非常に小さい値となっており、精度良く解析されていることが確認される。



(a) x-direction error (VN=1737)　(b) y-direction error (VN=1737)



(c) x-direction error (VN=297)　(b) y-direction error (VN=297)

Fig. 5.16 Displacement error of square domain problem

(b) 曲げ荷重を受ける平板の解析

Fig. 5.17 に示す正方形板に 1 点に集中荷重を与えた曲げ問題を、全体を 100×100 要素に等分割し、これを 2 つの領域に分割して解析する。仮想節点は境界節点と等しく 101 点配置した場合と、51 点に減らした 2 ケース設定し、それぞれの影響半径の大きさは Table 5.1 に示したものを設定する。

領域間境界における変位誤差は、Fig. 5.18 に示すように、非常に小さく精度良く解析できることが確認される。なお、境界の中心付近において X 方向の変位誤差が大きい値を示しているが、これはこの付近において変位が 0 となるため、相対的に誤差が大きくなるためである。

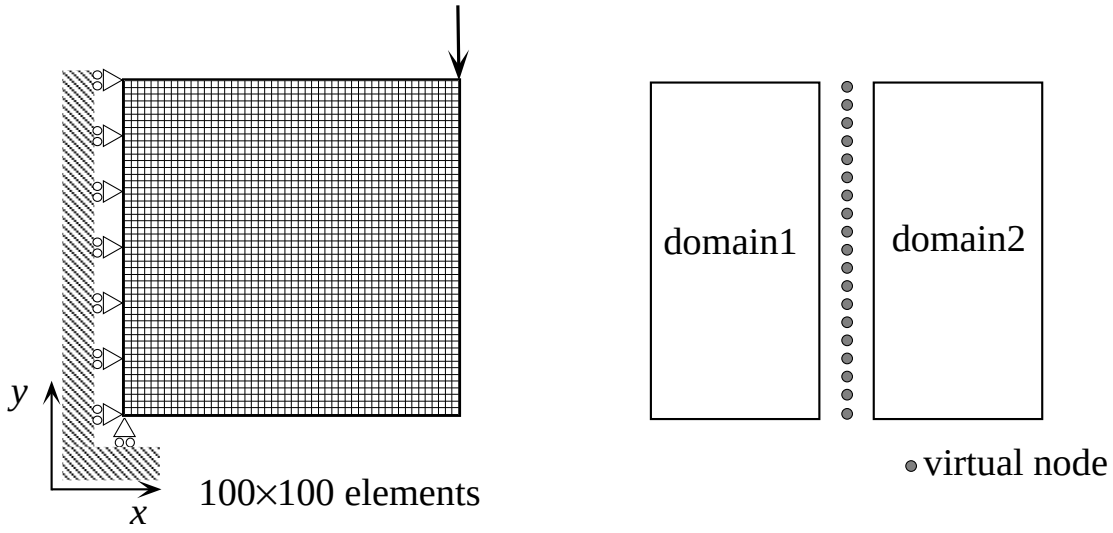


Fig. 5.17 A square plate subjected to bending load

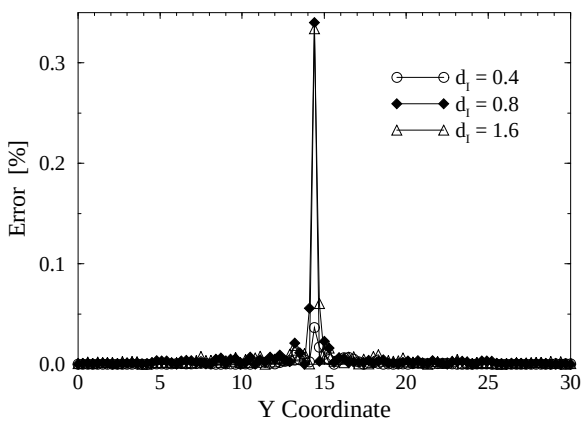
以上のことから、領域間境界において要素分割が適合する通常の問題に対しては、仮想節点数によらず良好な精度で解析できており、本解析法の妥当性が検証される。

5.4.2 領域間境界において要素分割が非適合となる問題の解析

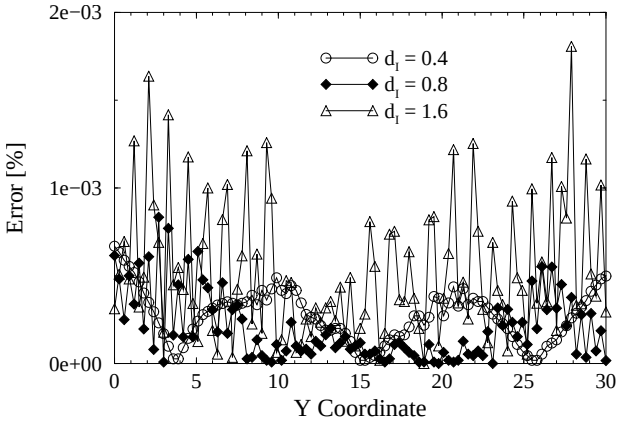
一様引張り荷重を受ける平板を、Fig. 5.19 に示すように 2 つの領域に分割し、それぞれの領域の要素分割数を変えることにより、領域間境界において要素分割が非適合となる問題を解析する。非適合の度合を変えるため、領域 (2) の要素分割数は 50×50 に固定し、領域 (1) の要素分割を 35×35 および 25×25 の 2 種類の問題を設定した。仮想節点は、それぞれにおいて 51, 38, 26 点を境界上に等間隔に配置し、要素分割数と仮想節点数が異なる合計 6 ケースについて比較する。

Fig. 5.20 に共役勾配法による仮想節点変位の収束の様子を示す。いずれのケースにおいても、解は発散することなく安定した収束性がみられるが、収束に向かう変位の値については、仮想節点数の設定によっては正解の値と異なるところに収束している。

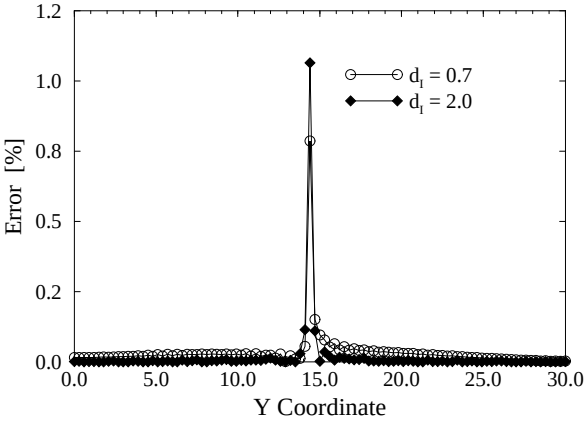
解が収束した後の領域間境界における変位誤差を、領域 (2) を 35×35 分割した場合を Fig. 5.21 に、25×25 分割の場合を Fig. 5.22 に示す。35×35 分割においては、仮想節点を 38 点および 26 点配置した場合には、Fig. 5.21 (c)〜(f) に示すように、X, Y いずれの方向の変位誤差も 1 %以下と小さいが、51 点配置した場合には、数%の誤差となっており解の精度が低下している。



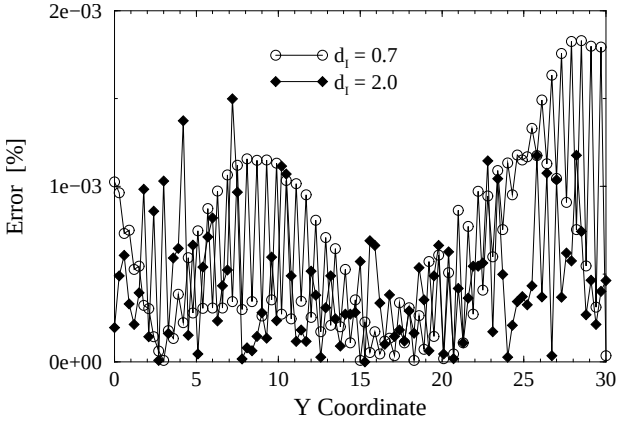
(a) x-direction error (VN=101)



(b) y-direction error (VN=101)



(c) x-direction error (VN=51)



(b) y-direction error (VN=51)

Fig. 5.18 Displacement error on boundary nodes for bending problem

一方、25×25 分割においては、仮想節点を 26 点配置した場合には、Fig. 5.22 (e), (f) に示すように、変位誤差が非常に小さい値となっているが、51 点および 38 点配置した場合には誤差は大きくなっている。以上のことから、領域間境界において要素分割が非適合となる問題においては、解の精度は仮想節点数に大きく依存し、また、非適合の度合によって最適な仮想節点数は異なることがわかる。

そこで、非適合の度合をさらに変えて解析を行ない、仮想節点数に対する変位誤差の変化を調べる。Fig. 5.19 に示した領域 (2) の要素分割数は 50×50 に固定し、領域 (1) の要素分割を 45×45 から 5×5 まで 9 段階変え、このときの領域間境界における変位誤差の最大値を Fig. 5.23 に示す。ここで、仮想節点数は

- 領域 (1) の境界節点数と等しく配置する場合

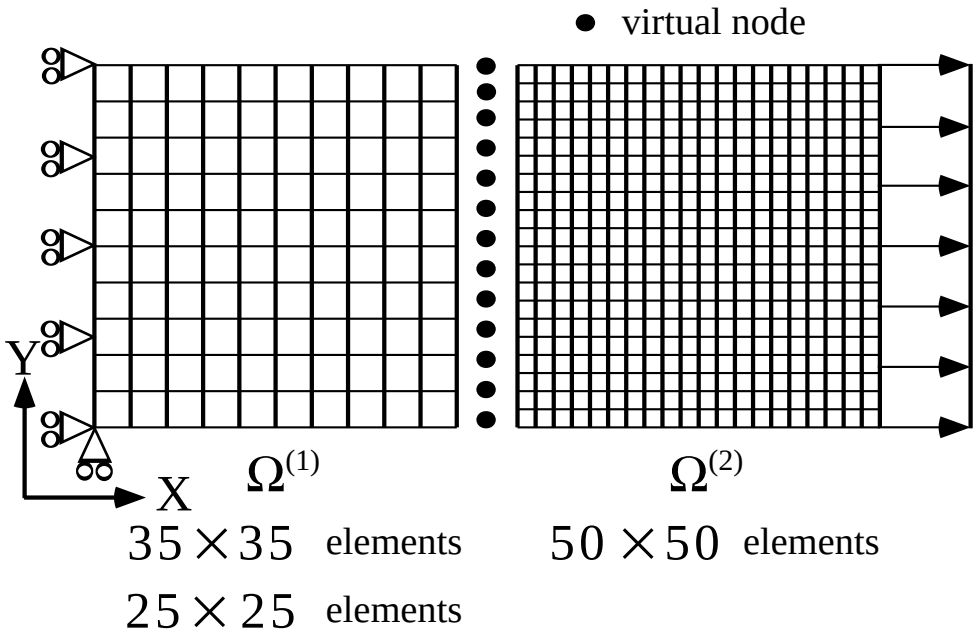


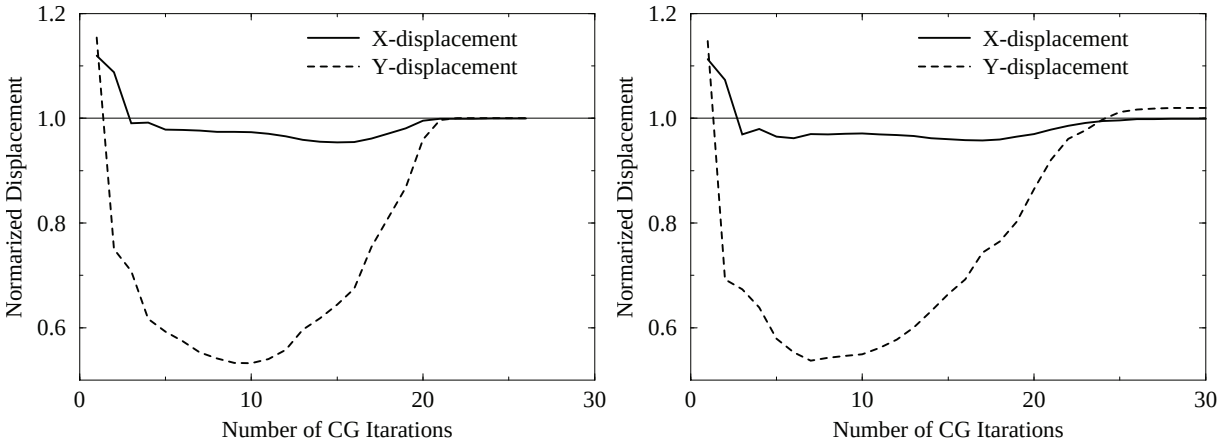
Fig. 5.19 Test problem with non-conforming interface between subdomains

- 領域 (2) の境界節点数と等しく 51 点配置する場合
- 領域 (1) の境界節点数よりも、さらに 8 割程度に減らして配置する場合
- 2 つの領域の境界節点数の平均値に設定する場合

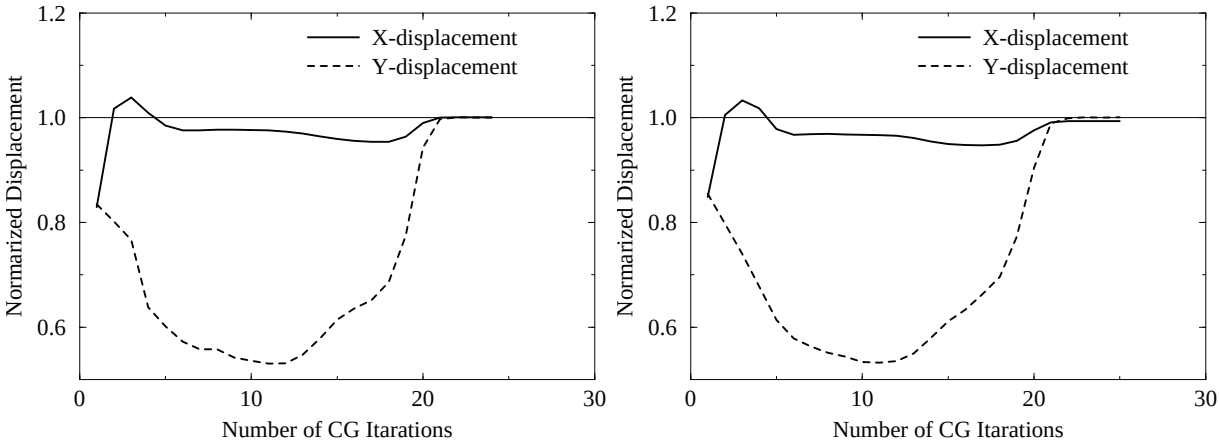
の 4 種類をそれぞれ設定した。

領域 (1) の要素分割を粗くして非適合の度合が大きくなるにしたがい、誤差は大きくなる。特に、仮想節点数を 51 点に固定した場合および 2 つの境界節点数の平均値に設定した場合には、誤差の増加が著しい。しかし、領域 (2) の境界節点数に合わせて仮想節点を配置すると、いずれのケースにおいても最大誤差は 1%程度もしくはそれ以下となり、精度良く解析できることがわかる。

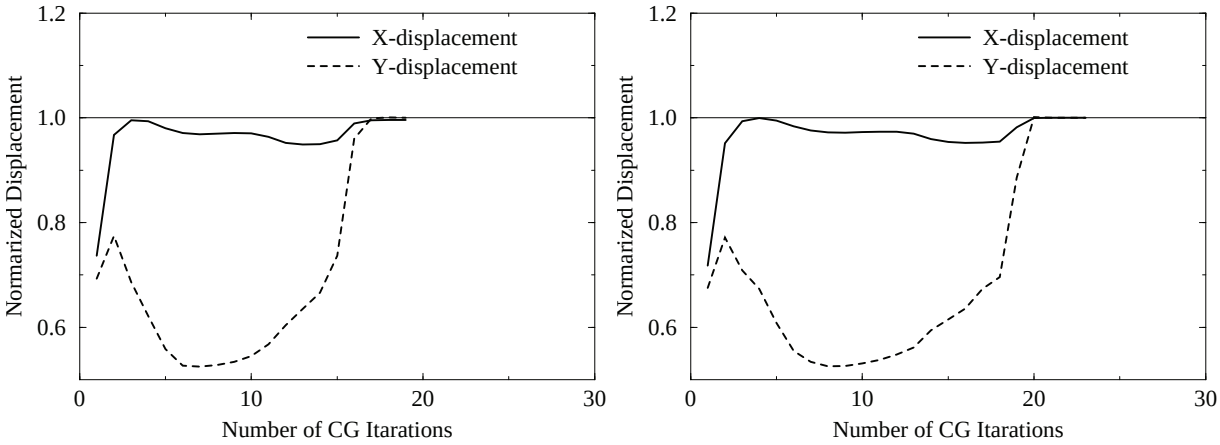
以上のことから、仮想節点数の設定に注意が必要であるが、本解析法によって非適合領域問題の解析が可能であることが確認される。



(a) 35×35 mesh, VN=51　　(b) 25×25 mesh, VN=51

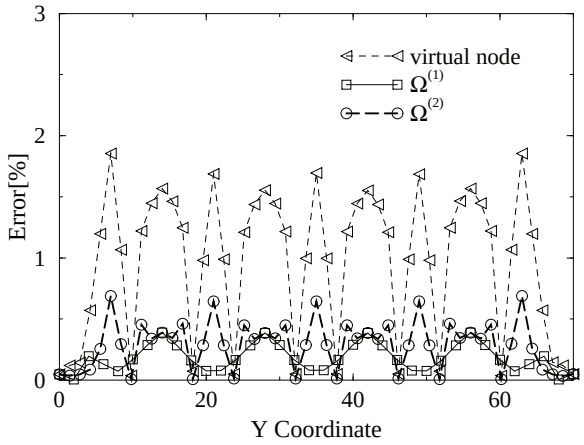


(c) 35×35 mesh, VN=38　　(d) 25×25 mesh, VN=38

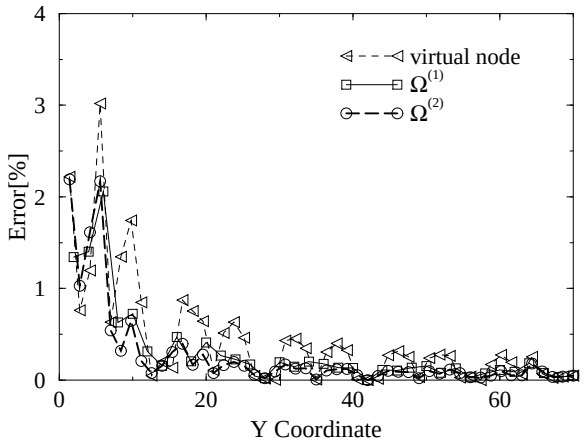


(e) 35×35 mesh, VN=26　　(f) 25×25 mesh, VN=26

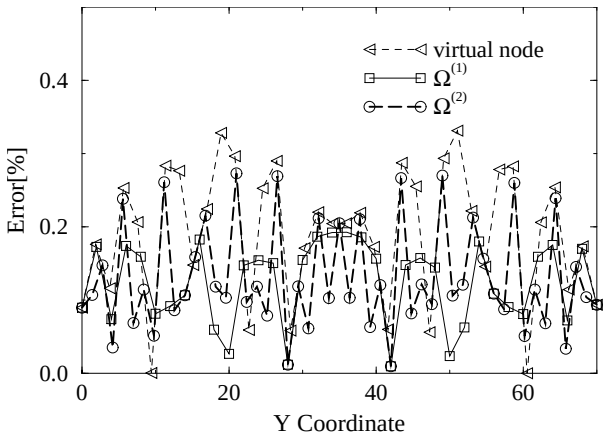
Fig. 5.20 Condition of CG convergence for non-conforming problems



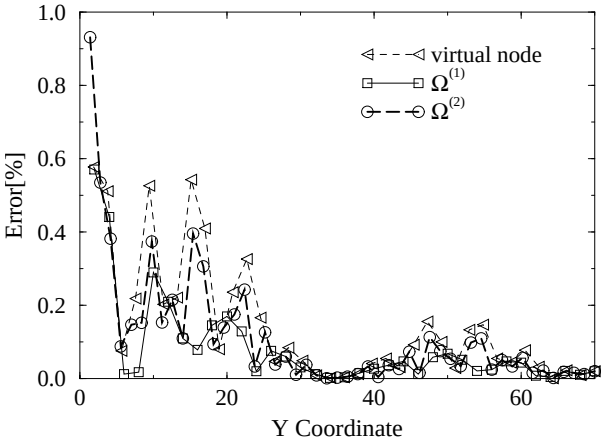
(a) x-direction error (VN=51)



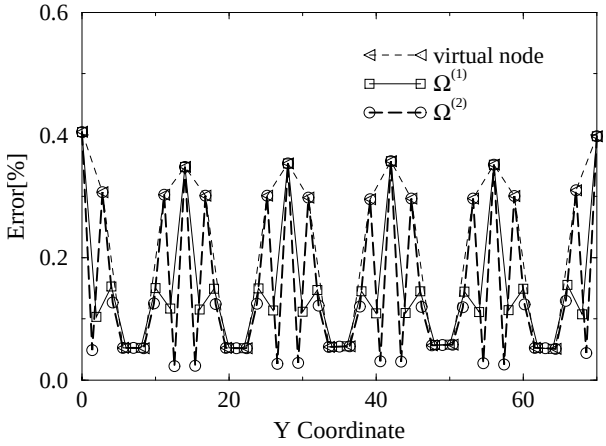
(b) y-direction error (VN=51)



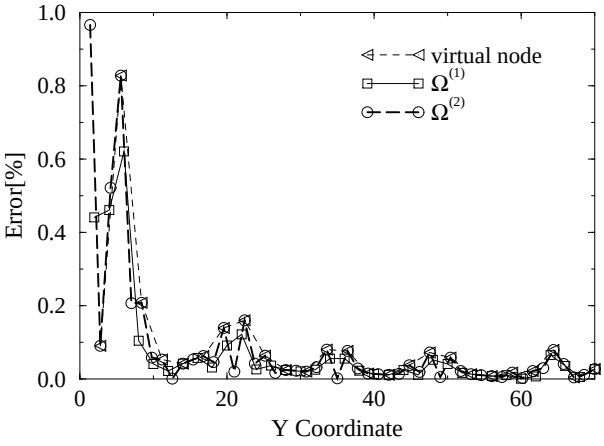
(c) x-direction error (VN=38)



(d) y-direction error (VN=38)

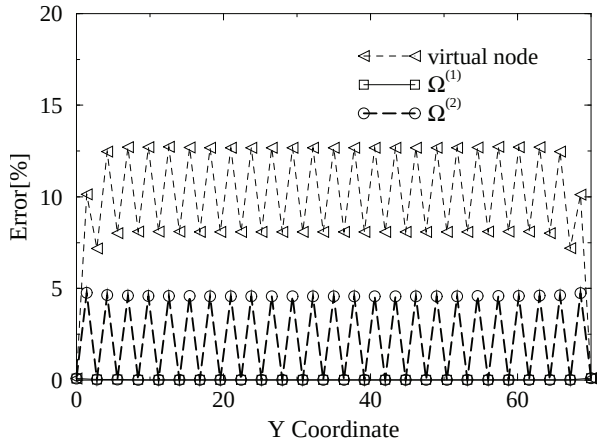


(e) x-direction error (VN=26)

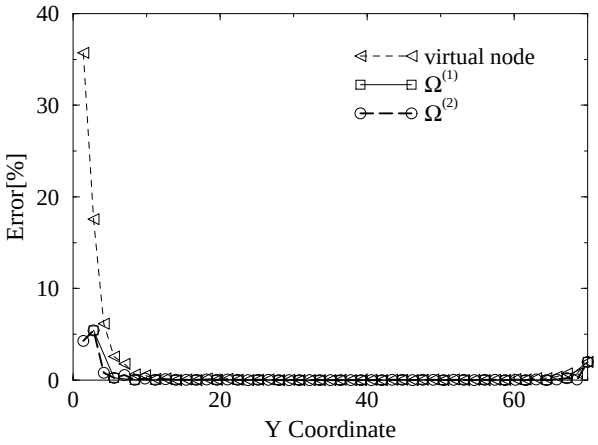


(f) y-direction error (VN=26)

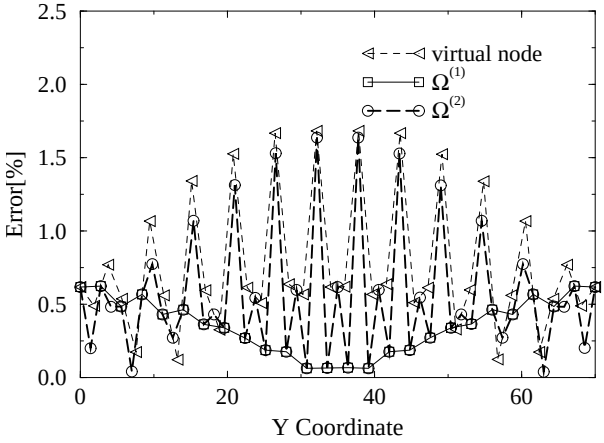
Fig. 5.21 Displacement error for non-conforming problem (35×35 mesh)



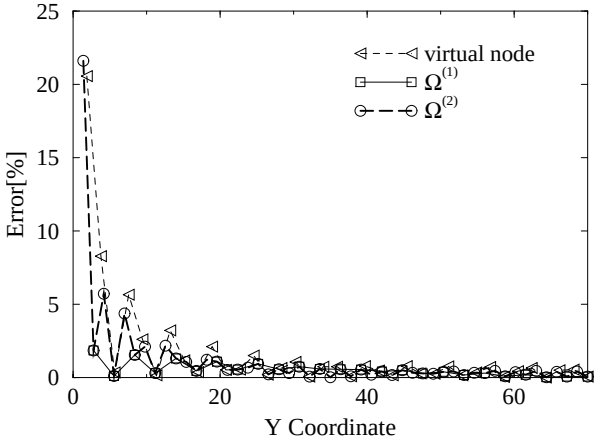
(a) x-direction error (VN=51)



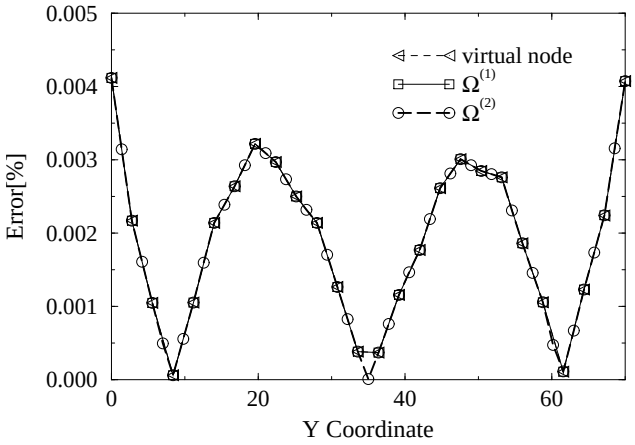
(b) y-direction error (VN=51)



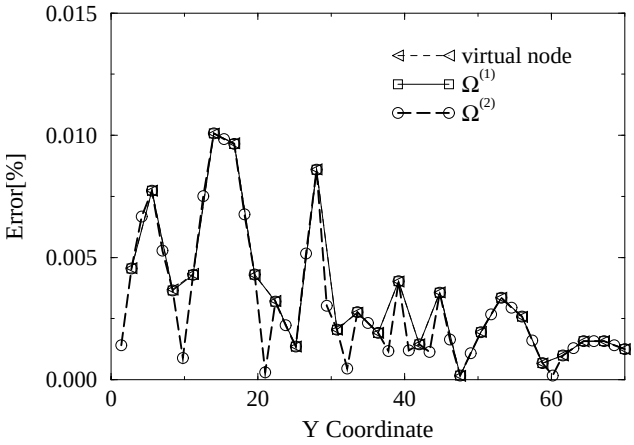
(c) x-direction error (VN=38)



(d) y-direction error (VN=38)

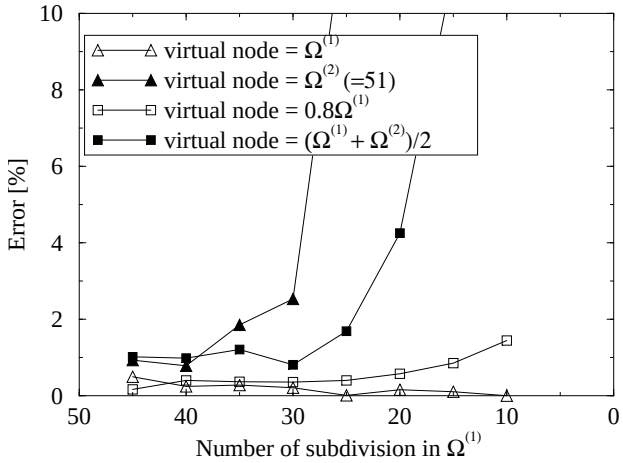


(e) x-direction error (VN=26)

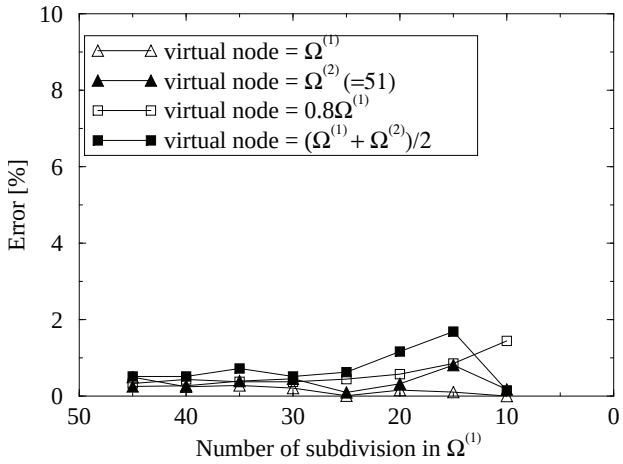


(f) y-direction error (VN=26)

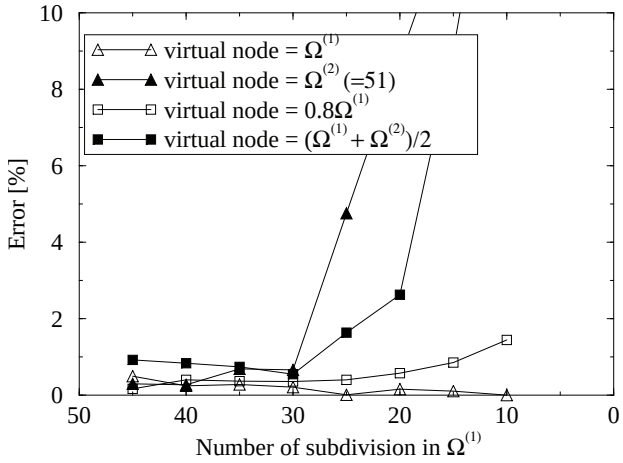
Fig. 5.22 Displacement error for non-conforming problem (25×25 mesh)



(a) virtual nodes



(b) domain (1) boundary nodes



(c) domain (2) boundary nodes

Fig. 5.23 Displacement error for non-conforming problem

5.5 並列性能の評価

5.5.1 解析時間の定量化

本解析法の並列性能を定量的に評価するため、解析時間の定量化を行なう。解析する問題を領域間において要素分割が適合している問題に限定し、このときの各領域の自由度数 n'_{eq} およびバンド幅 n'_{bw} を一定とする。総領域数を n_d、スレーブ台数を n_s とすると、共役勾配法の反復 1 回当たりの各処理時間は以下のように表すことができる。

- 仮想節点データの送信

$$T_{co} = \alpha_c n'_v \frac{n_d}{n_s} + \beta \tag{5.30}$$

- 1 台のスレーブ・マシンが担当する領域の解析時間 (反復 1 回目)
  - 移動最小自乗法による仮想節点パラメータの内挿

$$T_{mls1} = \alpha_{t1} n'_v n'_{bf} \frac{n_d}{n_s} \tag{5.31}$$

  - 領域内の解析

$$T_{d1} = \alpha_{d1} n'_{eq} n'^2_{bw} \frac{n_d}{n_s} \tag{5.32}$$

- 1 台のスレーブ・マシンが担当する領域の解析時間 (反復 2 回目以降)
  - 移動最小自乗法による仮想節点パラメータの内挿

$$T_{mls2} = \alpha_{t2} n'_v n'_{bf} \frac{n_d}{n_s} \tag{5.33}$$

  - 領域内の解析

$$T_{d2} = \alpha_{d2} n'_{eq} n'_{bw} \frac{n_d}{n_s} \tag{5.34}$$

- 共役勾配法のパラメータ修正演算

$$T_{cg} = \alpha_{cg} n_v \tag{5.35}$$

ここで、n_v は全仮想節点の総自由度数、n'_v は 1 つの領域の境界に配置されている仮想節点の自由度数、n'_{bf} は 1 つの領域の境界自由度数であり、α_c, β, α_{d1}, α_{d2}, α_{t1}, α_{t2}, α_{cg} は定数である。これらの定数の NWS–5000 (Fast Ethernet) のクラスタ・システムにおける実測結果を、Table 5.2 および Table 5.3 に示す。

Table 5.2 Coefficients of communication speed

	NWS–5000 (Fast Ethernet)
α_c [sec/word]	1.3624×10^{-6}
β [sec]	0.0151

Table 5.3 Coefficients of computation speed

	NWS–5000
α_{d1}	5.02752×10^{-7}
α_{d2}	5.58571×10^{-7}
α_{cg}	2.84646×10^{-6}
α_{t1}	1.90482×10^{-6}
α_{t2}	4.93009×10^{-7}

反復回数を I_t とすると、解析時間は

$$T = T_{co}\,(n_s + 1) + T_{d1} + T_{mls1} + (T_{d1} + T_{mls2})\,(I_t - 1) + T_{cg} I_t \tag{5.36}$$

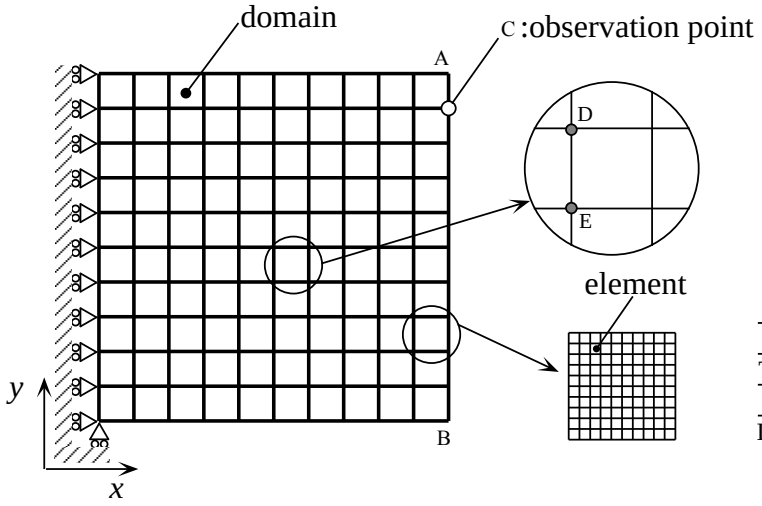
と表される。また、ネットワークに飽和が生じた場合には、

$$T = T_{co}\,(2 n_s I_t - n_s - 1) + T_{d1} + T_{mls1} + T_{cg} I_t \tag{5.37}$$

と表される。ここで、スレーブ台数 n_s、領域分割数 n_d のほか、境界上に配置する仮想節点数により n_v および n'_v のほか反復回数 I_t も変化することから、この 3 つが本解析法における並列パラメータとなる。

5.5.2 評価式の妥当性の検証

導出した評価式の妥当性を確認するため、Fig. 5.24 に示すような一様引張りを受ける正方形板を 100×100 要素に分割し 10×10 領域に分割した問題に対して、実際の解析時間と評価時間の比較を行なった。このとき、仮想節点は境界節点と同一の位置に配置した。Fig. 5.25 に示すように、通常時およびネットワーク飽和時において良い一致が見られ、導出した評価式の妥当性が確認される。



DOFS	20300
Total Number of Nodes	10201
Total Number of Elements	10000
Number of Domains	100
DOFs of Domain Interface	3465

Fig. 5.24 Test problem for the parallel performance evaluation

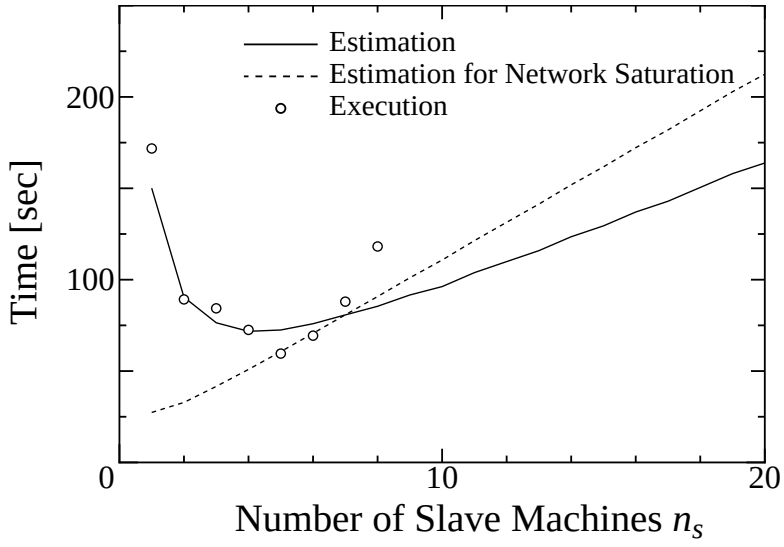


Fig. 5.25 Comparison between estimation and execution time

5.5.3 領域分割法との並列性能の比較

Fig. 5.24 に示した問題は境界節点数は 3465 であるが、境界上に配置する仮想節点数をこれよりも少なく設定すると、共役勾配法の計算の対象となる自由度は通常の領域分割法に比べ減少する。このため、仮想節点数に対して、解が収束に至るまでの反復回数

は、Fig. 5.26 に示すように変化する。このように、本解析法は境界節点数よりも少ない

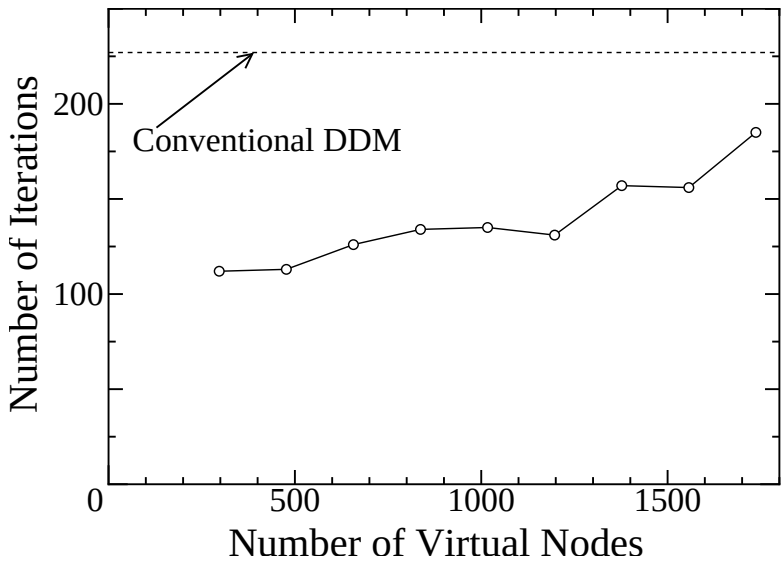


Fig. 5.26 Comparison of number of CG iterations

仮想節点数を用いて解析することにより、通常の領域分割法に比べて収束性が向上する。

仮想節点数を変化させると、データ転送量や移動最小自乗法による内挿演算の時間が変化するため、スレーブ台数に対する解析時間は Fig. 5.27 のように変化する。本解析法は、通常の領域分割法に比べて、領域ごとに行なわれる演算量は内挿演算分多くなり、逆にデータ転送量は仮想節点自由度数分へと小さくなることから、ネットワークの飽和は起こりにくいことがわかる。

スレーブ台数を 2 および 4 台に固定したとき、マスターおよびスレーブにおいてそれぞれ行なわれる演算と通信の各処理時間を Fig. 5.28 に示す。仮想節点数を境界節点数と等しい 3465 に設定すると、スレーブ・マシン上で行なわれる領域ごとの内挿演算に時間がかかるため、解析時間は領域分割法より増大する。しかし、仮想節点数を減少させると、内挿の演算時間および通信時間が減少するとともに反復回数も Fig. 5.26 に示したように減少するため、Table 5.4 に示すように、領域分割法よりも高速に解析することができる。

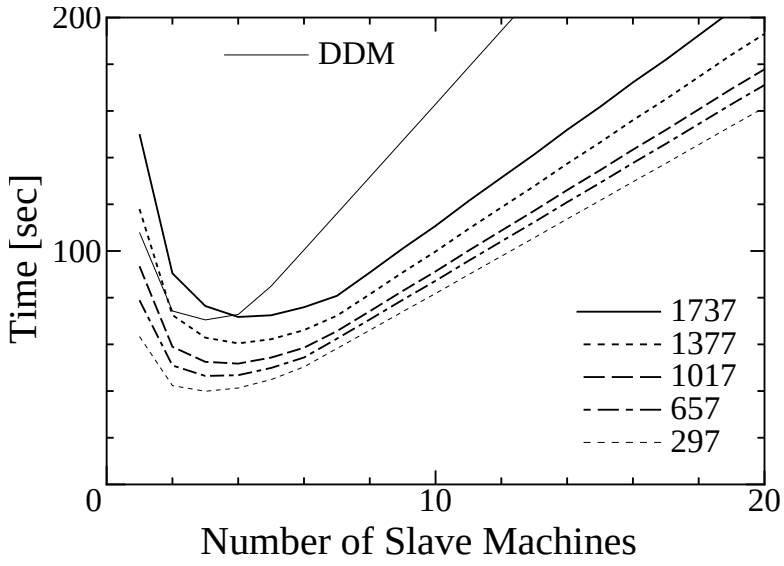


Fig. 5.27 Variation of estimation time for different number of virtual nodes

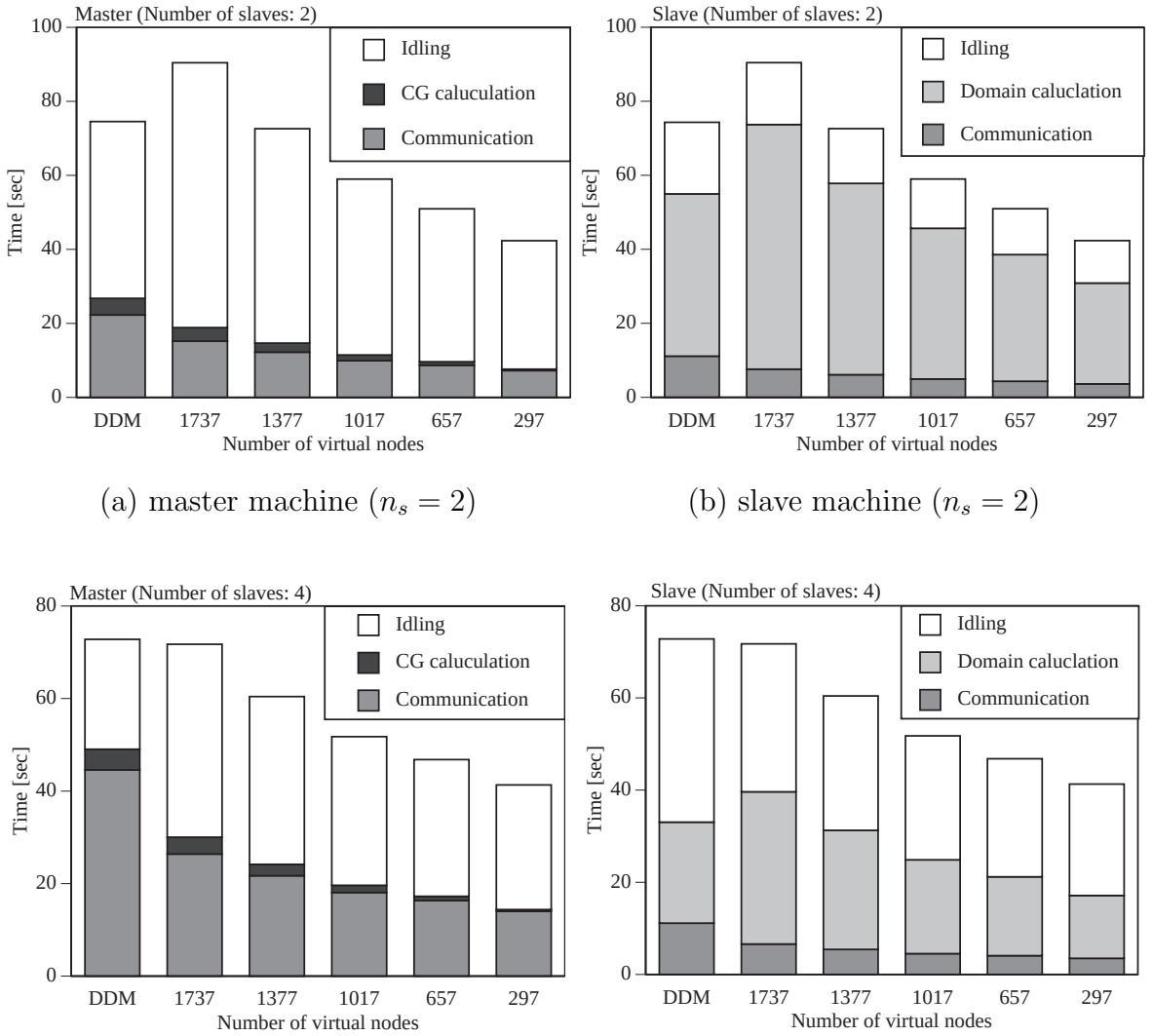
Table 5.4 Comparison of parallel performance between DDM and presented method

Number of virtual nodes	Nubmber of iterations	Time [sec]	Speed-up vs. DDM
DDM	227	72.8	1.0
1737	185	71.3	1.01
1377	157	60.4	1.20
1017	135	51.8	1.41
657	126	46.8	1.56
297	112	41.3	1.76

Number of Slave Machines: 4

5.6 今後の検討課題

本解析法は、領域間を内挿精度が高い移動最小自乗法を適用して接続することにより、境界上に配置する仮想節点数を境界節点数よりも減少させることが可能であるとともに、共役勾配法の解の収束性および並列性能を向上させることを確認した。以上のように、



(a) master machine ($n_s = 2$)　(b) slave machine ($n_s = 2$)

(c) master machine ($n_s = 4$)　(d) slave machine ($n_s = 4$)

Fig. 5.28 Processing time on master and slave machine

本解析法は大規模有限要素解析の並列化手法として非常に有力な可能性を持っていることを明らかにしたものの、本研究においては、基本的な 2 次元問題の解析に適用するに留まった。したがって、今後、本解析法が広く実用化されるためには、

- 3 次元大規模問題、領域分割が複雑な問題への適用

  本解析法の有効性がもっとも発揮される、大規模な 3 次元問題や複雑に領域分割が行なわれた問題に適用し、並列性能および共役勾配法の収束性が通常の領域分割法に比べて向上することを確認する。

- 領域ごとにアダプティブ有限要素法を行なう非適合問題の並列解析

  それぞれの領域ごとにアダプティブ有限要素法を適用し、その際、領域間境界において要素分割が不連続となる問題の並列解析を行なう。これにより、並列アダプティブ有限要素法の有効性を検証する。

- 共役勾配法の収束度に応じた仮想節点数の動的な変更による収束性の向上

  仮想節点のメッシュレス性を生かし、共役勾配法による解の収束度合に応じて仮想節点数を徐々に増加させ、収束性のさらなる向上を図る方法を開発する。

といった応用研究が必要であり、こうした研究開発が進展し本解析法が広く実用化されることを期待する。

5.7 まとめ

領域分割法における境界上の適合条件を取り払うとともに並列性能をさらに向上させるため、領域間境界上にメッシュレス仮想節点を配置し領域間を移動最小自乗法により精度良く接続する新しい領域分割法を提案した。本解析法の妥当性を検証するため、2次元の平板問題について荷重条件および領域分割を各種設定して解析を行なった結果、以下の結論を得た。

- 領域間において要素分割が適合する問題においては、仮想節点を境界節点よりも大幅に減らしても精度良く解析することができる。

- 領域間で非適合となる問題においては、非適合の度合が大きくなると、仮想節点を配置する数によっては解の精度が低下することから、仮想節点の配置に十分な注意が必要である。

- 境界節点数よりも少ない仮想節点を用いて並列解析を行なうことにより、通常の領域分割法に比べて、共役勾配法の収束性および並列性能が向上する。

なお、本研究においては、解析法の提案および例題の解析により本解析法の有効性が確認されたが、実用化のためには今後の応用研究が行なわれることに期待する。

参考文献

(5.1) C. Farhat and F.-X. Roux, “A Method of Finite Element Tearing and Interconnecting and its Parallel Solution Algorithm”, *Int. J. Num. Meth. Engng.*, **32**, (1991), *pp.* 1205–1227.

(5.2) C. Farhat, “Fast Structural Design and Analysis via Hybrid Domain Decomposition on Massively Parallel Processors” *Compt. Systems. Engng.*, **4**, (1993), *pp.* 453–472.

(5.3) C. Bernardi, Y. Maday and A. Patera, “A New Nonconforming Approach to Domain Decomposition: The Mortar Element Method”, *Nonliner partial differntial equations and their applications*, H. Brezis and J. L. Lions (eds.), Pitman, (1989), *pp.* 13–51.

(5.4) 藤間, “モルタル有限要素法による流れ問題の並列計算”, 計算工学講演会論文集, **1**, (1996), *pp.* 59–62.

(5.5) B. Nayroles, G. Touzot and P. Villon, “Generalizing the Finite Element Method: Diffuse Approximation and Diffuse Elements”, *Comput. Mech.*, **10**, (1992), *pp.* 307–318.

(5.6) T. Belytschko, Y. Y. Lu and L. Gu, “Element-Free Galerkin Method”, *Int. J. Num. Meth. Engng.*, **37**, (1994), *pp.* 229–256.

(5.7) G. Yagawa, T. Yamada and H. Kawai, “Some Remarks on Free Mesh Method: A Kind of Meshless Finite Element Method”, *Proc. of ICES'95*, (1995), *pp.* 1572–1577

(5.8) 奥田, 長嶋, 矢川, “エレメントフリーガラーキン法に関する基礎的検討 (第 1 報, 常微分方程式への適用)”, 日本機械学会論文集 A 編, **61**-590, (1995), *pp.* 2302–2308.

(5.9) P. Lancaster and K. Salkauskas, “Surfaces Generated by Moving Least Squares Methods”, *Math. Comput.*, **37**, (1981), *pp.* 141–158.

(5.10) A. R. Diaz, N. Kikuchi and J. E. Taylor, “A Method of Grid Optimization for Finite Element Methods”, *Compt. Meths. Appl. Mech. Eng.*, **41**, (1983), *pp.* 29–45.

(5.11) K. Kashiyama and M. Kawahara, “Adaptive Finite Element Method for Linear Water Wave Problems”, *Proc. JSCE*, No. 387-8, (1987), *pp.* 115–124.

(5.12) O. C. Zienkiewicz and J. Z. Zhu, “Adaptivity and Mesh Generation”, *Int. J. Num. Meths. Eng.*, **32**, (1991), *pp.* 783–810.

第 6 章

結 論

本論文で述べた、ワークステーション・クラスタによる大規模有限要素解析に関する研究で得られた著者の見解は、以下の通りである。

ワークステーション・クラスタを用いた並列処理においては、ネットワークの通信容量による制約が非常に大きいため、データ通信量および頻度を減少させることにより、大きな粒度の並列処理を行なわなければならない。こうした留意点をもとに、大規模問題の並列解析法を開発するとともに、並列性能の定量的な評価を行なった。

1. ワークステーション・クラスタを用いて、ガウスの消去法、共役勾配法および領域分割法による並列有限要素解析システムを開発し、さらに定量的な性能評価を行なった結果、
   - 最適な並列パラメータの決定法を示すとともに、3 つの方法とも最適なパラメータを設定することにより、性能が大きく向上することを示した。
   - いずれの方法においても、ネットワークの通信容量の制約が厳しいことが明らかとなり、通信容量の制約を緩和させる 1 つの方法として、サブネットを利用すると大きな効果があることを示した。
   - 35 万元規模の大規模解析が 20 台のワークステーションを用いて十分実用的な時間で行なえることを確認した。
2. 通信性能が高い Fast Ethernet を用いたワークステーション・クラスタ・システムにおいて、領域分割法、領域型共役勾配法および並列ガウス消去法の 3 種類の並列有限要素解析手法について、並列性能の評価および比較を行った結果、

- 高速ネットワークの利用は、ワークステーション・クラスタによる並列有限要素解析において、極めて有効である。
- 高速ネットワーク環境の性能を十分に引き出すには、並列パラメータの最適な設定が重要である。
- ネットワークの飽和パラメータを定義し、これを用いてEthernet環境と比較した結果、高速ネットワークの利用が通信の制約を大きく緩和させることを確認した。
- 高速ネットワークの導入により、35万元規模の解析を、10台のワークステーションを用いて、短時間に行えることを確認した。

3. 電磁構造連成問題の並列解析法として、領域分割法と領域型共役勾配法を組み合わせた並列解析法を開発するとともに、平板のたわみ連成問題および核融合装置真空容器の実機モデルの解析において並列性能の評価を行なった結果、
   - 領域分割法と領域型共役勾配法を組み合わせて用いることにより、並列性能および共役勾配法の解の収束性が向上することを確認した。
   - 核融合装置真空容器の実機モデルの大規模連成解析に適用し、本並列解析法の実用性を検証した。
4. 領域間境界上にメッシュレス仮想節点を配置して領域間を移動最小自乗法により精度良く接続する新しい領域分割法を提案し、いくつかの2次元平板問題を解析した結果、
   - 仮想節点を境界節点よりも大幅に減らしても、精度良く並列解析することができる。
   - 領域間で要素分割が不連続となる非適合問題の解析が可能であることを確認した。ただし、非適合の度合が大きくなると仮想節点を配置する数によっては解の精度が低下することから、仮想節点の配置に十分な注意が必要である。
   - 境界節点数よりも少ない仮想節点を用いて並列解析を行なうことにより、通常の領域分割法に比べて、共役勾配法の収束性および並列性能が向上する。

以上の研究成果が実用化され、大規模問題の解析に少しでも役立つことができれば幸いである。

謝 辞

本研究を行なうにあたり、卒業研究から通算して 6 年間に渡り、終始、熱心な御指導を賜わりました九州工業大学情報工学部助教授 堀江知義 先生に深く感謝致します。また、九州工業大学情報工学部教授 中垣通彦 先生、九州工業大学情報工学部助手 萩原世也 先生には、研究を進める過程で貴重な御助言と励ましの御言葉をいただきました。深く感謝致します。

同じ研究室の同期生として 6 年間一緒に過ごさせていただいた九州工業大学大学院情報工学研究科博士後期課程の二保知也 氏には、電磁構造連成解析の並列化に関する研究を行なうにあたり、システム構築のベースとした解析コード EMSHELL を提供していただいたほか、研究全般にわたって数多くの御助言あるいは御討論をいただきました。深く感謝致します。また、メッシュレス仮想節点を用いた領域分割法を開発するにあたり、九州工業大学大学院博士前期課程の宮垣英司 氏には、システム構築や解析計算の実行に関して多大な御協力をいただきました。感謝致します。

本研究を行なった間、九州工業大学情報工学部 堀江研究室に所属された中山征一朗 氏、西河智雅 氏、田中伸宏 氏、石田一久 氏、堀池豊 氏には、研究を進める上で御協力をいただきました。ここに感謝致します。また、同研究室の学部生および大学院生には、御協力や激励をいただきました。感謝致します。

最後に、学生生活を経済的に支え長い目で見守り続けてくれていた両親と、電子メイルを使っていつも精神的に励ましてくれた親友 武井祐司 君に、心より感謝致します。