分散メモリ環境における数値計算の並列化

博士論文

分散メモリ環境における数値計算の並列化

髙田　雅美

2004年1月9日

奈良女子大学大学院人間文化研究科
博士後期課程　複合領域科学専攻

博士論文

分散メモリ環境における数値計算の並列化

高田　雅美

2004年1月9日

奈良女子大学大学院人間文化研究科
博士後期課程　複合領域科学専攻

本論文は奈良女子大学大学院人間文化研究科に
博士（理学）授与の要件として提出した博士論文である

髙田　雅美

指導教官：　城　和貴　教授

分散メモリ環境における数値計算の並列化[1]

高田　雅美

概要

物理現象や自然現象などの解明のために，数値シミュレーションが使われる．この数値解析は，多くの場合，大規模な計算とメモリを伴う．計算機性能が飛躍的に向上し，逐次プログラム用の最適化コンパイラが一般的になった今日でさえ，数値解析は，非常に長い計算時間を必要とする．また，大規模な解析をおこなうためのメモリが不足することもある．

仮に，精密な数値解析を短時間でおこなうことができるならば，ターンアラウンドタイムが短くなり，より多くのパラメータチョイスをおこなうことが可能となり，幅広い研究成果を得ることが期待される．

実行時間の短縮とメモリ不足の解消のための手段として，並列計算機を使用することがあげられる．現在，多くの大規模並列計算機は，分散メモリ環境である．そのため，並列プログラムを開発する場合，プログラム分割の最適化と同時に，データ分割も最適化されていなければならない．そのためには，専門的な知識が必要となり，並列プログラムの開発は非常に困難なものとなる．特に，既存の大規模数値アプリケーションとしての逐次プログラムを並列プログラムに変換することは，一段と困難を極めることとなる．

また，分散メモリ環境として，Grid システムが着目されている．Grid システムとは，異なる計算機間の通信方法やプログラミング方法等を定義することによって，様々な計算機を用いて並列化するための仕組みである．将来的に，私達の身の回りにある携帯端末，PDA，家電製品等に組み込まれている計算機も利用した Grid システムの構築が考えられる．

[1]奈良女子大学大学院人間文化研究科博士論文，2004 年 1 月 9 日

本研究の研究対象は，数値計算をおこなうための分散メモリ環境である．この研究対象に対して，並列プログラムの開発と新たな計算機としての携帯端末の可能性を検証する．

本論文の構成は，V部である．

第II部では，擬似スペクトル法を用いた乱流場の直接数値シミュレーションの並列化について述べた．

第II部の第2章では，並列プログラムの開発対象とするアプリケーションである自由表面乱流場における熱物質輸送の数値シミュレーションについて説明した．数値解析の精度をあげるためには，しばしば擬似スペクトル法が用いられる．擬似スペクトル法は，流体の時間発展方程式を解く際，周波数空間で計算を行う．そのために，問題空間は，フーリエ変換を用いて周波数空間に変換される．

第II部の第3章において，計算時に発生するプロセッサ間の通信オーバーヘッドにまで着目した並列化手法を提案した．通常，乱流シミュレーションの並列化において，配列データは，strip mining によって実空間を均等に分割される．しかし，擬似スペクトル法を用いた乱流場の直接数値シミュレーションに strip mining を適用した場合，通信時間と通信オーバーヘッドが膨大になる可能性がある．そこで，通信時間と通信オーバーヘッドを考慮し，複数の配列を処理する際に，一つの配列がなるべく少数のプロセッサに割り当てられるように分割し，なおかつ，複数の配列の演算が同時に処理できるような並列化手法を提案した．

第II部の第4章では，提案した並列化手法の有効性を検討するために，並列プログラムを Message Passing Interface を用いて実装し，安価な分散メモリ環境において性能評価をおこなった．その結果，現実的な実行時間で解が得られることを確認した．

第III部では，NTT Docomo が提供するiアプリ対応の携帯端末を研究対象とし，数値計算の実行が可能であるかどうか検証した．

第III部の第2章では，iアプリについて説明した．iアプリ対応の携帯端末は，非常に貧弱ではあるが，計算リソースを持っており，2003 年 12 月現在，その市場稼動台数は 2,100 万台を超えている．このうち，大部分の携帯端末はアイドル状態であるため，余剰計算リソースが存在している．そこで，アイドル状態にある携帯端末において，iアプリを用いた数値計算を行うための分散処理システムによって，計算リソースを有効活用する方法を提案した．

第III部の第3章では，プロトタイプとして用いる遺伝的アルゴリズムについて説明

した.

第III部の第4章では，遺伝的アルゴリズムをiアプリを用いた分散処理システムとして実装するために，島モデルの遺伝的アルゴリズムと呼ばれるモデルをMaster-Slave型に改良した.

第III部の第5章では，提案したアルゴリズムの性能を評価するために，トイプロブレムであるナップサック問題を解き，性能評価をおこなった.

第IV部では，関連研究についてまとめる.

第IV部の第2章では，並列化された数値計算ライブラリを紹介する.

第IV部の第3章において，乱流シミュレーションの並列化に関する研究をまとめる.

第IV部の第4章において，余剰計算リソースに関する研究について述べる.

第IV部の第5章において，Gridシステムについて説明する.

キーワード

乱流シミュレーション，並列プログラム，分散メモリシステム，性能評価，
直接数値シミュレーション，iアプリ，余剰計算リソース，遺伝的アルゴリズム，
島モデル，Master-Slave

Parallelizations of Numerical Calculations on Distributed Memory Systems

Masami Takata

Abstract

When analyzing physical and natural phenomena, numerical simulations are used. Large-scale computational quantities and memory are often needed for the numerical calculations. Even when computer abilities are improved rapidly and optimizing compilers for sequential programs are used in general, the numerical calculation takes very long computational time. And, the lack of memory is often caused by Large-scale computational quantities.

If the calculation times decrease, the turn-around times shorten. Therefore, it expects that various research results would be obtained since more parameters can be chosen.

To decrease calculation time and to get the most of memory, it is the most realistic method to use parallel computers. Many parallel computers adopt distributed memory systems presently. Therefore, data partitioning and assignment should be optimized as well as parallelization, when developing parallelized programs. Hence, it is very difficult to develop parallelized programs, since it requires special knowledge of the parallelization. Particularly, it is more difficult to transform sequential programs as Large-scale numerical applications into parallelized program.

Grid systems are cynosures as distributed memory systems. Grid systems are plots, that are methods of communication systems and programs in different computer systems, for parallelizations of various computers. In a future, a grid system consists of mobile phones, PDA, and electrical appliances with computational resources.

The object research is a development of distributed memory systems for numerical calculations. In this research, I develop a parallel program and validate whether mobile phones have computational resources for numerical calculations.

This paper consists of V parts.

In the II part, I explain a parallel method that a directed numerical simulation using pseudo-spectral method is transformed to a parallel program.

In the 2 section of the II part, I describe a direct numerical simulation for a turbulent flow. To increase the accuracy of this simulation, some times pseudo-spectral method is applied. Applying pseudo-spectral method means that the calculation is carried out in the frequency domain, that is transformed by Fourier transform.

In the 3 section of the II part, I propose an improving parallel method that regards to overhead of communication between processors. The conventional parallelization method for the directed numerical simulation divides a loop structure in the calculation homogeneously, that is called strip mining, in spatial domain. However, it may require a lot of communication time and communication overhead, in the case that the strip mining refers to the direct numerical simulation using pseudo-spectral method. To solve the problem, I propose an improving method for parallelization: In the case of carrying out plural processes with several arrays, a process for an array is assigned to as few processors as possible, and plural processes are carried out as simultaneous as possible.

In the 4 section of the II part, I implemented the parallel program with Message Passing Interface. As a result, I obtained the solution of the direct numerical simulation in practical time with an entry level distributed computing system.

In the III part, I validate whether the i-APPLI, provided by NTT Docomo Co.ltd, can calculate numerical calculations.

In the 2 section of the III part, I explain the i-APPLI. Although a mobile phone has just poor calculation resource, it provides data communication facility as well as calculation resource. Currently, over 21 millions mobile phones with i-APPLI have been shipped to the market, and most of them are considered as idle states. A computational resource consisting of many idle mobile phones is attractive for a distributed processing system. Therefore, I propose a distributed processing system using mobile phones for numerical

calculations.

In the 3 section of the III part, I describe a genetic algorithm that is used as a prototype.

In the 4 section of the III part, an island genetic algorithm using i-APPLI is improved to Master-Slave model.

In the 5 section of the III part, I implemented a toy-problem, that solve the knapsack problem, and I show the performance by experimental results.

In the IV part, I organize some relational researches.

In the 2 section of the IV part, I introduce some parallelized numerical calculation libraries.

In the 3 section of the IV part, I show some parallel methods for a flow simulation.

In the 4 section of the IV part, I explain two relational researches in surplus computational resources.

In the 5 section of the IV part, I explain Grid systems.

Keyword

turbulent flow simulation, parallelized program, distributed memory system,
performance evaluation, direct numerical simulation, i-APPLI,
surplus calculation resource, genetic algorithm, island model, Master-Slave

目 次

図目次

表目次

第I部

はじめに

大気流・天気予測等の地球規模の物理・自然現象などの解明のために，それぞれの現象は数値化され，数式によって解析される．この数値解析を用いた場合，対象となる現象のスケールによっては，実行時間が非常に長くなるため現実的な時間で実行することが難しい．この問題は，近年，計算機性能が飛躍的に向上し，逐次プログラム用の最適化コンパイラを使用可能な環境が一般的になったことによって，多少なりとも解決された．しかし，現代の最新技術を用いたとしても，解析対象が大き過ぎたり，精密な解析結果を得るためには，膨大な実行時間が必要となる．また，数値解析をおこなうためには，各現象を数値化しなければならないが，その情報を保持するために，数値解析の実行中，大規模なメモリを必要とすることが多い．このため，メモリ不足といった問題が生じることもある．

仮に，精密な数値解析を短時間でおこなえるならば，ターンアラウンドタイムが短くなり，より多くのパラメータチョイスをおこなうことや，さらに大規模な，あるいは詳細な解析が可能となる．その結果，様々な分野で新たな研究成果を得ることが期待される．

実行時間の短縮とメモリ不足の解消のための手段として，現状で最も現実的な方法として，Grid システム [29] や原子力研究所の地球シミュレータに代表される並列計算機を使用することがあげられる．現在，多くの大規模並列計算機は，分散メモリ環境である．このアーキテクチャのための並列プログラムを開発する場合，互いに関連し合うプログラム分割とデータ分割を，それぞれ同時に考慮しつつ最適化・並列化する必要がある．なぜならば，データ分割もプログラム分割も NP 完全問題であるからだ．もし，プログラム分割のみ最適化されているとするならば，データを必要とするプロセッサ以外のプロセッサにそのデータが配置されている可能性がある．この場合，通信オーバーヘッドが生じ，全体の計算時間が増加する場合がある．一方，データ分割のみを考慮すると，プログラム分割が効果的におこなわれない危険性がある．ゆえに，データ分割もプログラム分割も同時に最適化される事が好ましい．そのためには，専門的な知識が必要となり，並列プログラムの開発は非常に困難なものとなる．特に，既存の大規模数値アプリケーションとしての逐次プログラムを並列プログラムに変換することは，一段と困難を極めることとなる．

分散メモリ環境として現在着目されている Grid システムとは，異なる計算機間の通信方法やプログラミング方法等を決定することによって，多種多様な計算機を並列化し有効活用するための仕組みである．この Grid システムが研究対象とする計算機は，地球シミュレータのような大規模並列計算機が大部分を占める．しかし，将来的には，台数確保

という観点から考えると，私達の身の回りにある携帯端末，PDA，家電製品等に組み込まれている計算機も利用したGridシステムを構築することが好ましい．

本研究の研究対象は，数値計算をおこなうための分散メモリ環境である．この研究対象に対して，並列プログラムの開発と新たな計算機としての携帯端末の可能性を検証する．

本論文の第II部から第V部までの構成は，以下の通りである．

第II部では，擬似スペクトル法を用いた乱流場の直接数値シミュレーション [79] の並列化について述べる．この並列化では，シミュレーション計算を複数の計算機に割り振り，総実行時間が最も短くなるようなプログラムとデータの分割方法を研究する．

第II部の第2章では，並列プログラムの開発対象とするアプリケーションである自由表面乱流場における熱物質輸送の数値シミュレーションについて説明する．乱流シミュレーションでは，場のスケール指数乗に比例した計算時間が必要となるため，スーパコンピュータを用いても精密な結果を現実的な時間で得ることが困難である [56]．しかし，乱流シミュレーションを必要とする分野において，粗雑な結果では実用化が難しいため，短時間での精密結果の取得方法の確立が望まれている．ゆえに，計算時間の短縮は非常に重要な課題である．また，今回対象とした乱流シミュレーションは，擬似スペクトル法と呼ばれる手法を用いたもので，これは空間周波数領域において時間発展を計算し，非線形項を実空間領域において計算するものである．

第II部の第3章において，計算時に発生するプロセッサ間の通信オーバーヘッドにまで着目した並列化手法を提案する．通常，乱流シミュレーションは，対象空間をメッシュ状に表し，その格子点上の情報を配列に保持することによっておこなわれる．隣接する格子点間では，計算のためにデータのやりとりが発生するため，並列化の際には，空間をまとまったブロックに細分化し，それぞれを各プロセッサに割り当てるということがおこなわれる．これは，物理現象の空間的局所性に基づいた分割である [43] [80]．擬似スペクトル法を用いた乱流場の解析では，周波数空間で計算を行い，畳み込み演算に関する部分は実空間で計算するため，フーリエ変換・逆変換が必要となる．そのため，通常の並列化手法を適用すると，分割されたある配列のデータを周波数変換するたびに，分割された配列データが全てのプロセッサで必要となる．このため，大量の通信オーバーヘッドが生じる．これを回避するために，一つの配列データを少数のプロセッサに割り当てる手法を提案する．このことは，データを分割した際に考慮しなければならない通信オーバーヘッドを最小にするという基準で考えた時に導かれるものである．

第II部の第4章では，提案した並列化手法を直接数値シミュレーションに適用し，実際に実行させて有効性を示す．

第III部では，数値計算をおこなうための新たな計算機として，NTT Docomo [47] が提供する Java 環境である i アプリ対応の携帯端末 [20] を研究対象とし，その可能性を検証する．

第III部の第2章では，i アプリについて説明する．i アプリ対応の携帯端末は，非常に貧弱ではあるが，計算リソースを持っており，その数は 2,100 万台を超えている [23] ．しかし，実際に常時 i アプリを実行している携帯端末は少なく，余剰計算リソースとして存在している状態である．そこで，この余剰計算リソースを用いて，数値計算を分散処理させる．ただし，現在の携帯端末の Java の実装では，データ量や処理速度に大きなボトルネックがあるため，実装可能な計算手法はそれほど多くない．

第III部の第3章では，プロトタイプとして用いる遺伝的アルゴリズム [18] について説明する．遺伝的アルゴリズムは，生物進化の過程を模倣した状態空間の探索手法で，組み合わせ最適化において，良好な最適化を示すことが知られている．

第III部の第4章では，遺伝的アルゴリズムを i アプリを用いた分散処理システムとして実装するために，島モデルの遺伝的アルゴリズム [60] と呼ばれるモデルを適用する．島モデルの遺伝的アルゴリズムでは，島である携帯端末同士のデータ交換が必要となる．そのための手段として，i アプリでは，サーバを介する通信が存在する．そこで，既存の島モデルの遺伝的アルゴリズムを Master-Slave 型に改良する．

第III部の第5章では，前章で提案した Master-Slave 型の島モデルの遺伝的アルゴリズムの性能を評価するために，トイプロブレムであるナップサック問題を解く．

第IV部では，関連研究についてまとめる．

第IV部の第2章では，並列化された数値計算ライブラリを 4 種類説明する．

第IV部の第3章において，乱流シミュレーションの並列化に関する研究として，陽解法，陰解法，擬似スペクトル法について，それぞれ説明する．

第IV部の第4章において，余剰計算リソースを用いた分散処理として SETI@home [63] と Folding@home [12] について述べる．

第IV部の第5章において，Grid システムについて説明する．

第II部

乱流シミュレーションの並列化手法

第1章　はじめに

原子炉，核融合炉・化学プラント等の工業装置内，さらには海洋・河川等の自然環境内において，特定の条件下で，自由表面を有する乱流場が頻繁に出現する．この乱流構造およびそれに付随する熱物質輸送を解析することは，工学上，重要である．これらの乱流場を解析する手法として，理論に基づく手法，実験に基づく手法，流体運動を記述する方程式の数値解を計算機上で求める直接数値シミュレーション（ *DNS* ： Direct Numerical Simulation ）がある [78] [79] ．これらの手法のうち，*DNS* は，近年，大型計算機の発達とともに，有力な解析手法として注目されている．

DNS で扱う乱流場は，非定常の 3 次元上のベクトル場で表現されているため，計算すべき格子点の数はスケールの 3 乗に比例して大きくなる．また，工学分野で対象となる乱流場は，高レイノルズ数・高プラントル数であることが多い．このため，大規模で精密なシミュレーションを高速におこなわなければならない．ゆえに，乱流場の *DNS* 計算のためには，大規模計算可能な高速演算ユニットと大規模メモリが必要となる．この問題を解決するために，乱流場の *DNS* 計算のための逐次プログラムを並列化することによって，演算の高速化と大規模メモリの確保を実現するというアプローチが検討されている．

現在，数値計算を実行する並列計算機環境としては 2 種類の環境が考えられる．一つは複数のプロセッサがメモリを共有する共有メモリ環境と呼ばれるアーキテクチャであり，もう一つは複数のプロセッサがそれぞれ固有のメモリを持つ分散メモリ環境と呼ばれるアーキテクチャである．大規模計算の計算時間のみを分散化させることが目的であるならば，計算データの通信をする必要がなく並列化に伴うオーバーヘッドが生じにくい共有メモリ環境を対象とした並列プログラムが有効である．しかし，本研究のように，大規模なメモリも必要とする場合，共有メモリ環境を用いると，必要なデータ量が多くなり過ぎ，メモリ不足のため，実行不可能となる可能性がある．この問題を回避するためには，計算時間のみならず計算データも分散化させる必要がある．そこで，計算データの分散が

可能な分散メモリ型並列計算機に対応した並列プログラムの開発が必要となる．

計算データを分散化させる場合，各メモリが保有可能なデータ量のみならず，データ交換のための通信コストとそれによって生じる通信オーバーヘッドを考慮しなければならない．本研究で取り上げた乱流場の *DNS* 計算は数値不安定性の回避と高精度な解を取得するために，時間の経過にともなう各格子点の発展のための計算は，擬似スペクトル法（ *PSM* ： Pseudo Spectral Method ）を用いて周波数空間上でおこなっている [79] ．しかしながら，この流体の方程式には非線形演算が含まれるため，更新をおこなう際には，データの一部をいったん実空間へ変換する必要が生じる．データを実空間と周波数空間とでやり取りするためには，離散フーリエ変換が必要となる．そのため，乱流場のシミュレーションにおける時間経過の基本単位を step とすると，1 step ごとに，フーリエ変換・逆変換が必要となる．通常の並列化技法として適用される手法は，空間を等分割するタイプの並列化手法（ strip mining ） [43] [80] である．この並列化手法を今回対象とした *PSM* を用いた乱流場の *DNS* 計算に適用した場合，各 step ごとに，全プロセッサ間で，分散された全データをやり取りする必要が生じ，通信コストと通信オーバーヘッドが膨大になることが予想される．そこで，乱流場の *DNS* 計算を並列化するために，通信コストと通信オーバーヘッドを抑えた形で分散メモリ環境に効率よく対応させるための手法の提案が望まれている．

第 II 部の第 2 章において，熱物質輸送の方程式から導かれる自由表面乱流場における *DNS* に関して定式化をおこなう．第 3 章では，乱流場の *DNS* 計算を分散メモリ型並列計算機環境へ適用するためのデータ通信方式の選択とデータとプログラムの分割手法について論じる．第 4 章では，第 2 章で説明した *PSM* を用いた *DNS* に対して，第 3 章で提案した並列化手法を適用し，逐次プログラムと並列プログラムの実行時間を比較する．

第2章　自由表面乱流場における熱物質輸送の直接数値シミュレーション

自由界面を有する乱流場は，広範囲の工学分野で頻繁に出現する．この際，自由表面上での熱や物質移動が界面での乱流構造を通しておこなわれる．仮に，自由表面が穏やかで変形が無視できるような状態であるならば，熱物質輸送の主因と考えられる乱流は，自由表面とは直接関係のない壁面等によるものであり，その生成には自由表面は寄与しない．しかし，乱流構造は，複雑な自由表面が存在することにより，変形を受け特長的な変化を示す [19] [33] ．この自由表面を有する乱流は，単なるランダムな流れを形成するのではなく，時空間に対して極めて高い自由度を持ち，大小様々なスケールを有する．そのため，適切なスケールで 3 次元非定常計算をおこなう必要がある [56] ．

3 次元非定常計算をおこなうためには，大規模計算可能な高速演算ユニットと大規模メモリが必要となる．この問題を解決するための方法として，分散メモリ環境に対応した並列化の開発があげられる．この方法を適用すると，並列化によって，演算負荷を分散させるだけでなく，分散メモリ環境の利用によって，大規模メモリの確保を容易にすることが可能である．

以下，第2.1節では，対象とする乱流場の説明をおこなう．第2.2節では，乱流場の *DNS* 計算における問題点を説明し，その解決方法について述べる．第2.3節では，基本方程式を説明する．第2.4節では，*DNS* 計算の解の精度を上げるために，第2.3節で説明した基本方程式を基に，格子点密度と依存関係を持たない *PSM* で用いられる周波数空間での方程式を表す．

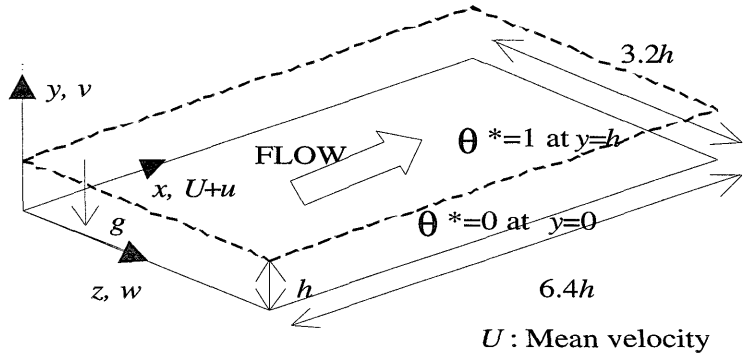

図 2.1: 解析対象と座標系

2.1　自由表面乱流場

本研究で対象とした流動場は，無限に広い平板上を一定の外力により駆動される十分に発展した低フルード数の回水路乱流場である．図 2.1 は，2 次元開水路乱流場の 3 次元座標系を模式的に表したものである．計算条件は，流れ場の平均断面流速 U_m と水深 h において，レイノルズ数を約 2,270 ，プラントル係数を 1 とし，（ x ，y ，z ）方向にそれぞれ（ 64 ，82 ，64 ）の格子点を用いた．山本らは，このスケールの格子点で乱流場解析をおこない，十分な精度の結果を得ている [78] [79] ．境界条件として，壁面で no-slip 条件，水面で free-slip 条件，主流およびスパン方向には周期境界条件を適用する．また，この計算結果の例として，上流側から仮想的な粒子を注入して流れを可視化したものを，図 2.2 に示す．流れは，左から右に向かっている．壁面付近の低速流体が乱流の組織運動により，水面付近まで達し，表面更新渦となり，大規模な水平渦を構成している様子が観察できる．

2.2　直接数値シミュレーションにおける問題点と解決方法

広範囲な工学分野において対象とされる流れ場を解析する際，*DNS* 計算を用いて精密な解を得るためには，高レイノルズ数・高プラントル数を必要とする場合が多い．

レイノルズ数 $Re = UL/\nu$ （ U ： 代表速度，L ： 代表長さ，ν ： 流体の動粘性係

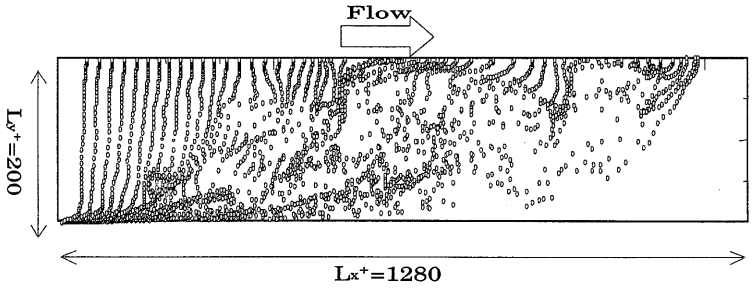

図 2.2: 上流側から仮想的な粒子を注入した場合の流れ

数）とは，流れの状態を記述する無次元の数で，対象としている流体中の慣性力と粘性力の比を表すものである．乱流の完全数値シミュレーション（ *FTS* ： Full Turbulence Simulation）において，レイノルズ数は重要な指標となる．流れが乱流である場合，レイノルズ数は大きな値（ 10^3 以上 ）となるが，これを数値解として表現するためには，少なくともレイノルズ数の 9/4 乗以上の格子点を要した非定常の 3 次元計算をおこなわなければならない [56] ．そのため，仮に $10G$ FLOPS のスーパコンピュータを用いて比較的小さいレイノルズ数 1,000 の乱流を計算する場合でさえも，数 step の時間発展をさせるために，30 分以上の計算時間を必要とする．

流体を記述するもう一つの定数であるプラントル数 Pr は，流体の拡散係数 ν と熱拡散係数 α の比（ $Pr = \nu/\alpha$ ）によって表される．この際，流体の拡散係数 ν よりも熱の拡散係数 α が小さい高プラントル流体における乱流場の熱輸送の解析をおこなう場合，3 次元の各方向にプラントル数の $-1/2$ 乗に比例した格子解像度を必要とする．ゆえに，高プラントル自由表面乱流場の *DNS* 計算を考える場合においても，大規模なメモリと高速演算に対処した計算環境が必要となる．

工学分野において *DNS* 計算を必要とするような乱流場では，高いレイノルズ数とプラントル数を用いて十分な精度の解析をおこなうために，格子点密度を高くする必要がある．そのため，大規模メモリで高速演算ユニットを有する計算機資源を長時間にわたって使用しなければならない．

大規模計算可能な高速演算ユニットの利用は，限られた機関の関係者に限定されている．また，仮に利用可能であったとしても，長時間，計算機資源を独占することは，好ま

しくない状況である．このため，一般的な計算機で計算できるように *DNS* 計算のレイノルズ数やプラントル数を低くするか，高速演算ユニットの代用となる方法を考案する必要がある．先に述べた方法は，乱流場の解析結果の精度の低下をもたらす．これに対し，後述の方法では，*DNS* 計算のアルゴリズムは変更されないため，十分な精度の解析が可能である．そこで，後述の方法を用いて，高速演算ユニットの長時間の確保が困難であるという問題を解決するために，乱流場の *DNS* 計算を並列化することが望まれている．

並列化に際し，大規模メモリの問題を解決しなければならない．第1章で述べたように，並列化のためのアーキテクチャとしては，共有メモリ環境と分散メモリ環境が存在する．共有メモリ型並列計算機の場合，メモリ容量は，計算機開発の際に決定される．そのため，メモリ容量の増加は，アーキテクチャレベルからの再開発が必要となるため，非常に困難である．これに対し，分散メモリ型並列計算機の場合，各プロセッサが固有のメモリを保有するため，必要なメモリ容量を満たすために，プロセッサを増加させることで対処可能である．

以上より，自由表面乱流場における熱物質輸送の *DNS* 計算を実行するためには，分散メモリ環境での並列化が必要となる．

2.3　乱流場の直接数値シミュレーションで用いる基本方程式

数値解析手法を用いるにあたって，乱流場を表現する基本方程式は非圧縮性流体の運動を計算するために，以下のような Navier-Stokes 方程式 [39] を用いた．

$$\begin{aligned}
\frac{\partial u_i^*}{\partial t} + u_j^* \frac{\partial u_i^*}{\partial x_j} &= F_i - \frac{\partial}{\partial x_i}\left(\frac{p^*}{\rho}\right) + \nu \frac{\partial^2 u_i^*}{\partial {x_j}^2}, \\
\frac{\partial u_i^*}{\partial x_i} &= 0, \\
\frac{\partial \theta^*}{\partial t} + u_j^* \frac{\partial \theta^*}{\partial x_j} &= \alpha \frac{\partial^2 \theta^*}{\partial {x_j}^2}
\end{aligned} \tag{2.1}$$

ただし，u_i^* は瞬間的な流速の i （$i=1$，2，3）方向の成分を表し，x_1 （x）は流下方向，x_2 （y）は鉛直方向，x_3 （z）は横断方向を表す．p^* は瞬間圧力，ρ は流体の密度である．F_i は，i 方向の重力加速度を表す．θ^* は，$(T^* - T_b)/(T_s - T_b)$ で表される水面および壁面における温度差により規格化した温度，T^* は温度，T_s，T_b は水面および壁

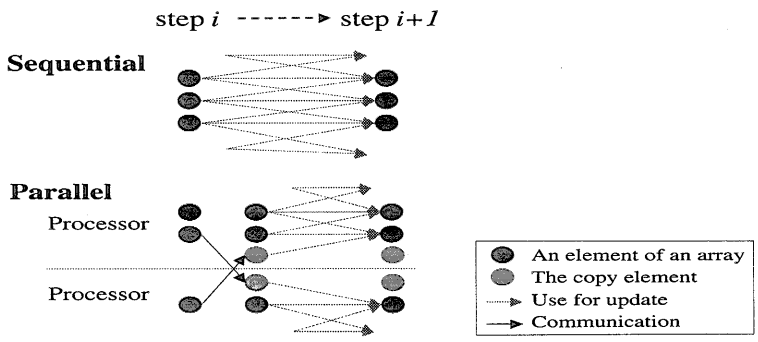

図 2.3: *FDM* を用いた際の 1 step 分の要素の関係

面における温度をそれぞれ表すものとする．ただし，$T_s > T_b$ を満たす．繰り返し添え字は縮約規約に従うものとする．

通常，流体の *DNS* 計算では，式 (2.1) で表現される微分方程式を差分化し，有限差分方程式（ *FDM* ： Finite Difference Method ）の形で解く．図 2.3 は，*FDM* の手法の一つである陽解法を用いた場合の各配列要素の依存関係を表す．図中，横軸は，step の経過を表し，上部は逐次プログラムを用いた場合の依存関係，下部は並列プログラムを用いることによってデータの通信を必要とする場合の依存関係を表す．*FDM* の場合，図 2.3 に表されるように，空間的に近接する格子点の情報のみによって次の step の状態が決定される．そのため，分散メモリ環境を対象とする並列プログラムを開発する際，空間方向に対して分割をおこない，分割された境界付近の情報のみを隣接部分の計算をおこなうプロセッサと送受信すれば良いと考えられる．

2.4 擬似スペクトル法を用いた乱流場の直接数値シミュレーションで用いられる方程式

FDM の解の精度を上げるためには，格子点密度を高くする必要がある．陽解法では，Courant 条件 [54] より，格子点密度の 2 乗に比例させて 1 step に割り当てる時間幅を小さくしなければならない．そのため，非常に短い時間経過をシミュレーションするためにも，大量の step 数が必要となり，計算時間のほうがシミュレーション上の経過時間より

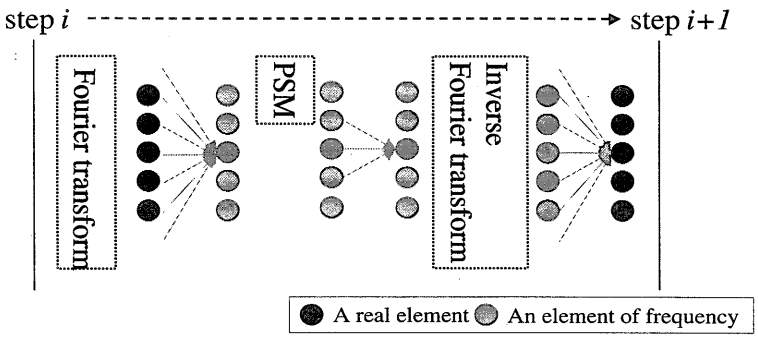

図 2.4: *PSM* を用いた乱流場の *DNS* 計算の 1 step における計算の流れ

も長くなるという問題が生じる可能性がある．それに対して，陰解法スキームや *PSM* のような周波数空間での解法では，時間幅と格子点密度は依存し合わない．ゆえに，実行時間を考慮した実装をおこなうためには，陰解法スキームもしくは *PSM* を用いるほうが現実的である．本研究では，*PSM* を用いて *DNS* 計算をおこなった．

式 (2.1) の周波数空間での表現は以下のように表される．

$$\begin{aligned} \frac{\partial U_i^*}{\partial t} + \mathrm{j}k_j U_j^* \circ U_i^* &= \mathcal{F}[F_i] - \mathrm{j}k_i \mathcal{F}\left[\frac{p^*}{\rho}\right] - \nu k_j^2 U_i, \\ \mathrm{j}k_i U_i &= 0, \\ \frac{\partial \Theta^*}{\partial t} + \mathrm{j}k_j U_j^* \circ \Theta^* &= -\alpha k_j^2 \Theta^* \end{aligned} \tag{2.2}$$

ただし，U_j^*，Θ^* は，それぞれ u_j^*，θ^* の空間周波数表現とし，演算子 $\mathcal{F}$ はフーリエ変換を表す．また，演算子 $\circ$ は，畳み込み計算（ $K \circ F(\omega) = \int_{-\infty}^{\infty} K(\omega')F(\omega-\omega')d\omega'$ ）を表すものとする．

式 (2.2) に関して，周波数空間では微分に関する演算は簡単に表せるようになるが，実空間での積に関する演算部分は畳み込み表現となる．そこで，式 (2.2) の畳み込み計算に関する部分は，実空間で演算をおこなったほうが効率的である．そのためには，時間発展の各 step ごとに，フーリエ変換とその逆変換が必要となる．つまり，各 step における計算の流れは，図 2.4 のようになる．

ここで考えなければならないのは分散メモリ環境での並列化の分割方法である．フーリエ変換は，実空間から周波数空間への変換であり，ある一つの周波数成分を得るためだけでも，図 2.4 に表されるように，全実空間のデータを必要とする．並列化の際には，通

常，対象空間をまとまったブロックに細分化し，1 つのブロックを 1 つのプロセッサに割り当てるということがおこなわれる．これは，strip mining と呼ばれる並列化手法で，物理現象の空間局所性に基づいた分割である．しかし，*PSM* を用いた乱流場の *DNS* 計算に対して，strip mining を適用すると，各 step においてフーリエ変換・逆変換をおこなう都度，各プロセッサ間で，境界だけでなく全データを送受信する必要が生じるために，通信コストと通信オーバーヘッドが膨大になる．したがって，分割方法を工夫する必要がある．この分割手法に関しては，以下の第 3 章で提案し，第 4 章において適用する．

第3章　並列化の際に考慮すべきことと並列化手法の提案

大規模データを分散メモリ型並列計算機で計算させる場合，プログラム分割のみならず，通信コストと通信オーバーヘッドを考慮したデータ分割をおこなう必要がある．

そこで，第3.1節において，分散メモリ型並列計算機を対象とする並列化ライブラリ *MPI*（Message Passing Interface）[40] の環境下での同期式通信と非同期式通信のメリットおよびデメリットに関して述べ，第3.2節において，提案する並列化手法に関して簡単な例を用いて論じる．

3.1　プロセッサ間の通信方式

並列プログラムを分散メモリ環境において実装した場合，各プロセッサに対して効果的にデータが分割されていたとしても，最小限のデータ通信が必要となる．このデータ通信をおこなうための関数として，並列化ライブラリ *MPI* には，同期式通信と非同期式通信の2種類の通信方式がある．

以下，第3.1.1項，第3.1.2項において，それぞれ，同期式通信，非同期式通信について説明する．第3.1.3項において，乱流場の *DNS* 計算をする際に必要となる通信方式の選択基準について述べる．

3.1.1　同期式通信

同期式通信の特徴は，送信側プロセッサによる送信用関数（*MPI_SEND*）の呼び出しと，対になる受信側プロセッサによる受信用関数（*MPI_RECV*）の呼び出しによって，データ通信が同期的におこなわれることである．

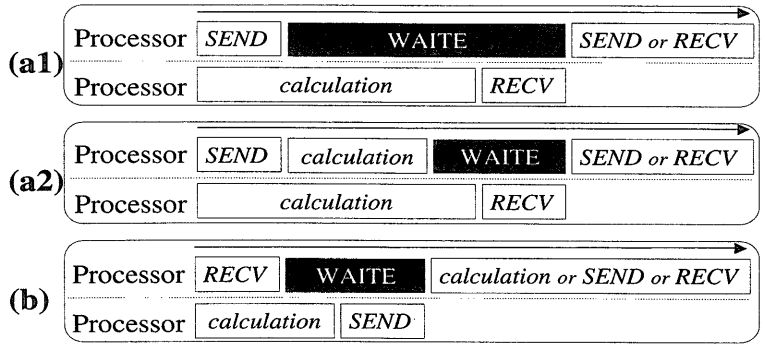

図 3.1: 同期式通信の模式図．(a) 送信用関数のほうが対になる受信用関数より先に呼び出される場合 ＜ (a1) 送信用関数の呼出し後，通信用関数の呼び出しがある場合． (a2) 送信用関数の呼出し後，計算の実行がある場合 ＞．(b) 受信用関数のほうが対になる送信用関数より先に呼び出される場合

図 3.1 は，同期式通信を模式的に表したものである．横向きの矢印は，時系列の向きを表す．計算時間の異なる処理をおこなうプロセッサ間で通信をおこなう場合，通信用関数の呼び出しのタイミングがずれる可能性がある．送信側プロセッサに割り当てられた計算量が少なく，先に送信用関数を呼び出した場合，図 3.1 の (a1) , (a2) に表されるような状況になる．送信用関数の呼び出し後，対になる受信用関数が呼び出されるまでの間，次の通信用関数を呼び出すことは不可能である．しかし，(a2) に表されるように，計算処理は，実行することが可能である．一方，受信側のプロセッサのほうが計算時間が短く，先に受信用関数を呼び出した場合，図 3.1 の (b) に表されように，対になる送信用関数が呼び出されるまでの間，次の通信用関数の呼び出しのみならず計算処理も実行することが不可能である．

つまり，図 3.1 の全ての場合において，通信用関数を呼び出すタイミングによっては，実行を一時中断して，対になる通信用関数を待たなければならない可能性があり，このことにより，通信オーバーヘッドが生じる．そのため，同期式通信をおこなう際には，通信に関わる全てのプロセッサの計算時間が均等になるようにプログラムを分割しなければならない．

また，通信用関数の呼び出し後，対となる通信用関数が呼び出されるまで，送信側プロセッサも受信側プロセッサも次の通信用関数を呼び出せないため，実行途中のある時点で

同一プロセッサから呼び出されている通信用関数は常に一つである．そのため，実行時の環境や状況に関係なく，通信の順番は同じものとなる．

3.1.2 非同期式通信

非同期式通信の特徴として，送信用関数（ *MPI_ISEND* ）の呼び出しと，受信用関数（ *MPI_IREV* ）の呼び出しは，別々のタイミングにおこなうことが可能であるという点があげられる．そのため，各プロセッサは，対になる通信用関数の呼び出しを待つ必要がない．ゆえに，計算を中断することなく，実行を続けることが可能となる．よって，通信に関するデータが，通信可能な状態になった時点で，対になるプロセッサの状況に関係なく，通信用関数を呼び出しておくことができる．

また，対になる通信用関数の呼び出しを待つ間，異なる通信用関数を複数呼び出すことも可能であるという特徴を持つ．このことは，通信用関数の呼び出しのタイミングによっては，同一のプロセッサから複数の通信用関数が呼び出されているという状況を作ることが可能であることを意味する．これは，計算時間がブロッキングされることはないものの，データ通信のタイミングによっては，間違えたタイミングにデータを書き換えてしまう危険性があることを意味する．

この問題を回避するためには，プログラム設計の段階で，データの依存性に対して十分に考慮する必要がある．通信用関数を呼び出した後，通信に関するデータの内容を更新する場合，通信が終了していなければならない．また，受信用関数を呼び出した後，通信に関するデータを利用する場合も，通信が終了していなければならない．そこで，並列化ライブラリ *MPI* には，実行を一時中断して通信の終了を待つための関数 *MPI_WAIT* が用意されている．この関数を利用することによって，間違えたタイミングにデータを書き換える可能性は減るものの，*MPI_WAIT* を呼び出すタイミングによっては，同期式通信同様，待ち時間が生じ，通信オーバーヘッドとなる．

すなわち，非同期式通信を用いた場合，通信関数を呼び出すタイミングを気にする必要がなく，複数の通信を呼び出すことが可能であるが，通信の終了と通信に関するデータの利用・更新のタイミングによっては，実行結果が間違えたものとなる可能性がある．そのため，非同期式通信を用いた並列プログラムを設計するためには，プログラマが専門的な知識と経験をつむ必要がある．ゆえに，一般的なプログラマにとって，非同期式通信を用

いた並列プログラムの開発は困難であり，プログラマへの負担が大きくなる．

3.1.3 通信方式の選択基準

乱流場の *DNS* 計算のような時間発展方程式の並列化を取り扱う場合には，空間方向に関するループを strip mining によって分割することによりデータ分割をおこなう方法が一般的である．この場合，空間が均等に分割されるため，各計算部分がホモジニアスな関係となり，それらの計算が割り当てられた各プロセッサの計算は，ほぼ同時に終了するものと考えられる．したがって，この場合，同期式通信を用いてもブロッキングの影響はほとんど生じない．

逆に，空間が非均等に分割された場合や，各プロセッサに異なる計算が割り当てられた場合，最も計算時間が長くなるプロセッサに同期しなければならなくなる．また，プロセッサの処理速度が非均等な環境においても，同期式通信のブロッキングの影響は大きくなる．このような場合，非同期式通信を用いたほうが，通信オーバーヘッドを回避しやすい．ただし，並列化ライブラリ *MPI* の関数 *MPI_WAIT* の呼び出しによるオーバーヘッドを最低限に抑えるために，計算の順番を変更する等の工夫が必要になる場合がある．

3.2 通信コストの削減を目的とする並列化手法

do 文に関して，各イタレーション間に依存関係が存在しない場合，並列実行文の *doall* 文に変換することができる．一般的に，*doall* 文は，全プロセッサに対して strip mining と呼ばれる手法で等分割されることが理想的とされている．

通常，流体のシミュレーションを *FDM* を用いておこなう場合，各配列データを更新する際考慮しなければならない作用は，図 2.3 に示されるような，空間的に隣接もしくは非常に近い位置からの作用のみである．そのため，並列化のために必要となるデータ通信は，境界部分のみであり，通信コストは非常に小さいといえる．したがって，乱流場の *DNS* 計算の並列化は，strip mining のような手法を用いることが一般的である．

しかしながら，本研究では，計算精度をあげるために *PSM* を適用し，周波数空間での *DNS* 計算をおこなっている．式 (2.2) には，畳み込み演算による非線形成分が含まれて

いることから，時間発展をおこなう各 step において，この非線形成分を計算するために，いったん実空間に戻した形で非線形項の計算をおこない，もう一度周波数空間に戻すという方法を適用している．

フーリエ変換の性質上，ある配列の一要素を更新する時でさえ，図 2.4 に表されるように，その配列の全データが必要となる．これは，データの依存性が著しく増大することを意味する．すなわち，空間方向に対して strip mining を適用した場合，フーリエ変換・逆変換をおこなう都度必要となる全配列要素のブロードキャストによる通信コストと通信オーバーヘッドは，極めて大きいものとなることが予想される．

そこで本研究では，strip mining を用いて配列を空間方向に等分割するのではなく，配列データ間の依存関係から推定される通信コストに応じて分割する手法を提案する．

本分割手法の効果を検証するために，第 3.2.1 項に述べる簡単な例題を用いた．従来の方法である strip mining を適用した一般的な並列化手法 A について，第 3.2.2 項で述べる．提案する並列化手法 B を適用した場合について，第 3.2.3 項で説明する．第 3.2.4 項において，並列化に関する考察をおこない，実際のプログラムの実行時間を評価する．

3.2.1 予備実験で用いるプログラム例

分割手法の効果を検証するために，次のような計算を行う例 1 を作成した．

1. *do* 10 で，配列 x に値を代入
2. *do* 20 で，配列 y 中の最小値を探索
3. *do* 30 で，配列 z に対する操作を実行
4. *do* 40 で，手順 2 で求めた最小値 min と手順 1 で計算された配列 x の M(i) 番目の要素との積を計算

この例 1 をプログラム化すると次のようになる．

＜例1＞

```
    do 999 loop=1, 1000
       do 10 i=1, 100000
10        x(i)=i
       min=y(1)
       do 20 j=2, 100000
20        if(min .gt. y(j)) min=y(j)
       do 30 k=1, 100000
30        z(k)=z(k)*k
       do 40 i=1, 100000
40        w(i)=x(M(i))*min
999 continue
```

なお M(i) は添え字をランダムに並べたものとする．これによって，並列化の際，手順4に，手順1で計算された配列 x の全てのデータが，全てのプロセッサにおいて必要となる状況を作り出した．

3.2.2 並列化手法 A

並列化手法 A は，全プロセッサに対して *doall* 文を strip mining によって均等に分割する一般的な並列化手法である．すなわち，複数の配列を処理していく場合，各配列を逐次的に処理し，配列の処理ごとに複数のプロセッサに均等にデータと演算を分割していく手法である．この場合，第3.1節で述べたように，各プロセッサ間に計算時間の差がなくなるので，同期式通信によるデータ通信をおこなうことを考える．

図3.2は，例1に対して，4プロセッサを用い手法 A を適用して実装した場合の実行順序を表す．図中，横軸は時間経過を表し，上部と下部はそれぞれ逐次プログラムおよび並列プログラムの実行順序を表す．

この分割手法は，配列の要素をプロセッサの数に応じて分割する方式である．よって，

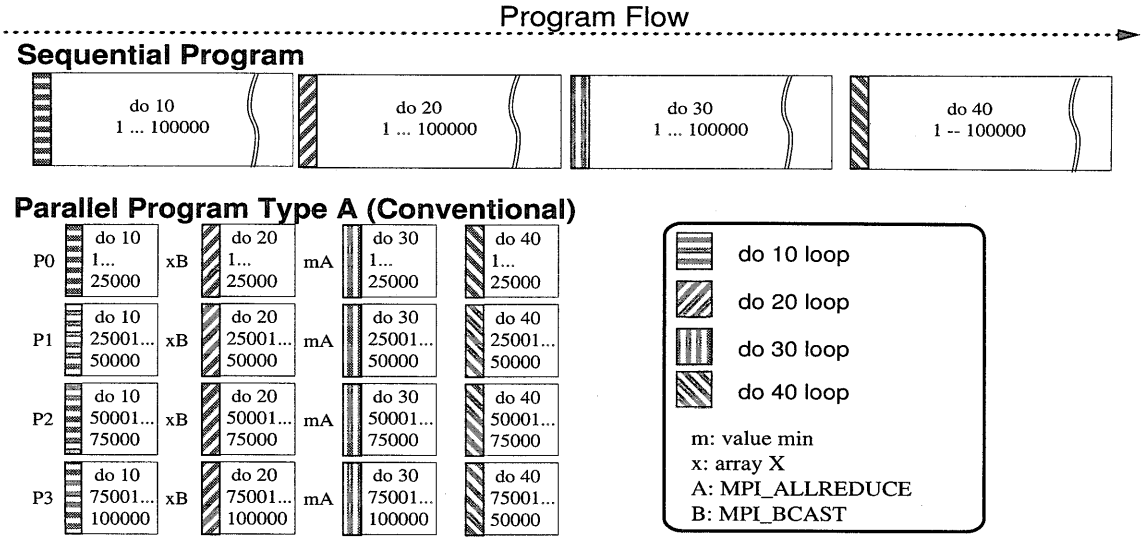

図 3.2: 各 *do* 文に対し，全プロセッサを対象とする strip mining による分割を実装した場合の並列プログラムの概念図（並列化手法 *A*）

各ループ番号に関して配列要素を 25,000 個ずつに分割し，そのデータを各プロセッサで処理させる．処理後，計算方法に応じて，計算結果を通信によって交換する．並列化手法 *A* において，*do* 20 に strip mining を施すことによって，各プロセッサは分割された配列要素の中から極小値となるものを検出する．検出後，各プロセッサが検出した変数 *min* のうち最小のものを選択するために，同期式通信 *MPI_ALLREDUCE* が必要となる．また，*do* 10 において初期化される配列 x は *do* 40 においてランダムにアクセスされる．そのため，各プロセッサは自分の持っていないデータを取得する必要が生じ，ブロードキャストをおこなう同期式通信 *MPI_BCAST* によってデータを取得しなければならない．

3.2.3 並列化手法 *B*

我々が提案する並列化手法 *B* は，複数の配列の処理をおこなう際に，配列の計算におけるデータ依存を考慮し，1 個の配列がなるべく少数のプロセッサに割り当てられるように分割し，複数の配列に関するループをなるべく同時に処理する手法である．

図 3.3 は，例 1 に対して手法 *B* を適用した場合の実行順序を示した図である．例 1 に

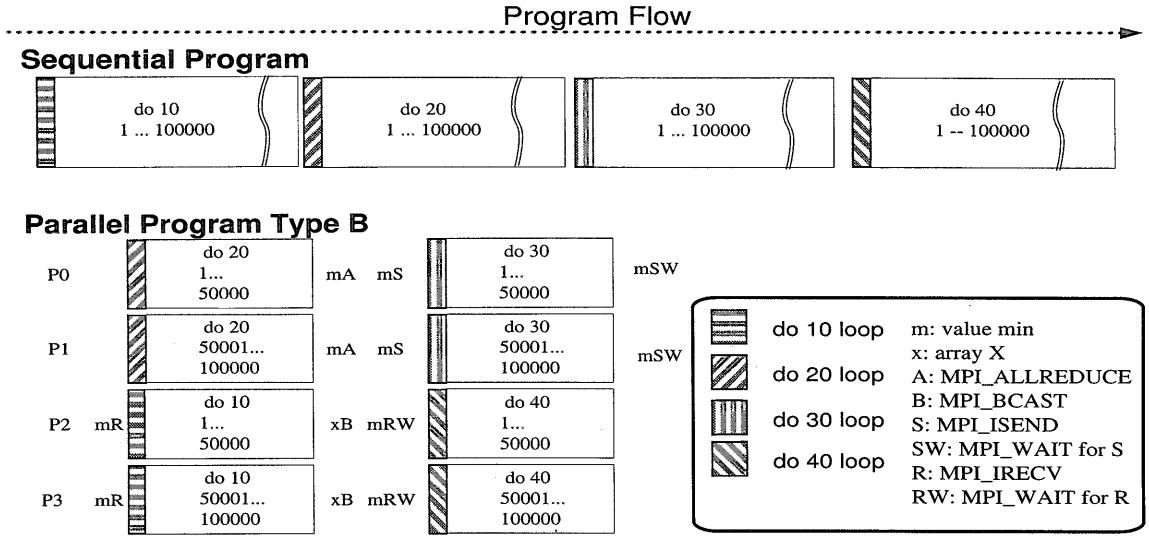

図 3.3: 各 *do* 文に対し，いくつかのプロセッサを対象とする strip mining による分割を実装した場合の並列プログラムの概念図（並列化手法 *B* ）

おける配列 x のデータ依存関係は， *do* 10 と *do* 40 でのみ生じる．これらの配列を分割することによって生じる通信コストを減少させるためには，*do* 10 と *do* 40 を計算するプロセッサ数を減少させればよい．*do* 20 と *do* 40 では，変数 *min* に関してデータ依存関係がある．また，*do* 30 は他のループとデータ依存関係を持たず，*do* 10 と *do* 20 は依存関係を持たない．以上のことを踏まえると，*do* 40 を計算する前に実行されるべきループは，*do* 10 , *do* 20 である．ゆえに，並列化手法 *B* として，全プロセッサのうち，半数ずつのプロセッサを用いて，同時に *do* 10 と *do* 20 を計算させることとした．

例 1 において，*do* 20 を strip mining によって 2 つに分割し，各々をプロセッサ id 0 （ $P0$ ）と id 1 （ $P1$ ）とに割り当てた．$P1$ は，処理後，計算結果である *min* を $P0$ に送信する．同タイミングに，プロセッサ id 2 （ $P2$ ），id 3 （ $P3$ ）において，*do* 10 を処理させ，処理後，それぞれ，配列 x を送受信する．$P0$ において，$P1$ から送信された *min* を受信後，$P0$ 固有の *min* と比較し，小さいほうを $P2$, $P3$ に送信する．最後に，各通信終了後，$P0$ と $P1$ において，*do* 30 を処理し，$P2$ と $P3$ において，*do* 40 を処理する．$P0$ と $P1$ に関して，*do* 20 を strip mining を用いて分割しているため，計算時間の差は少ないと考えられる．また，$P2$ と $P3$ に関しても同様である．ゆえに，$P0$ と $P1$

表 3.1: 例 1 の実行時間

	user time (s)	system time(s)
逐次プログラム	123	0
並列化手法 A	55	73
並列化手法 B	43	28

間，または，$P2$ と $P3$ 間における通信は，同期式通信とした．しかし，$P0$ と $P2$ ，$P3$ 間における通信は，do 20 と do 10 の計算時間が異なるため，非同期式通信を適用するものとした．

3.2.4 例 1 を用いた予備実験

例 1 の逐次プログラムと並列化手法 A および B をそれぞれ用いて実装した並列プログラムを実行した場合の性能を比較する．

表 3.1 は，それぞれのプログラムにおける例 1 の実行時間である．user 時間は各プロセッサにおける演算時間および関数呼び出しのための時間の合計を表し，system 時間は通信時間および通信オーバーヘッドを表す指標とする．

今回提案した並列化手法 B は，並列化手法 A と比較して，プログラムの設計段階で，データ依存関係と計算順序を考慮せねばならず，実装をおこなう際のポータビリティという面においては劣ると考えられる．しかし，例 1 のプログラムを用いた予備実験の結果より，並列化手法 B を用いた並列プログラムのほうが，並列化手法 A を適用するよりも，通信コストが多いにもかかわらず，関数呼び出しによる user 時間の増加が少なく，また，system 時間が減少していることから，通信オーバーヘッドが抑えられていることがわかる．したがって，性能面において，並列化手法 B は並列化手法 A よりも優れていると考えられる．

第4章　並列化手法の直接数値シミュレーションへの適用と実行結果

第3.2節で提案した並列化手法 B を，本研究で取り扱う *PSM* を用いた乱流場の *DNS* 計算のためのプログラムに実装し，実際の有効性を検証する．

以下，第4.1節では，本研究で対象とする *DNS* 計算に対する並列化手法 B の適用を説明する．第4.2節では，より多くのプロセッサを用いた並列化が可能となるように，第4.1節で説明した並列化手法の適用を拡張する．第4.3節では，*PSM* を用いた *DNS* 計算の逐次プログラムと並列プログラムを用いて，実験によって有効性を検証する．

4.1　直接数値シミュレーションへの提案手法の実装

本研究で対象とした *PSM* を用いた乱流場の *DNS* 計算のための逐次プログラムは，初期化部分と計算部分に分割することができる．この逐次プログラムの実行時間の大部分は，時間発展の計算部分の繰り返しによって消費される．ゆえに，時間発展計算部を最適に並列化することが重要となる．

乱流場の *DNS* 計算で使われる主な配列を表4.1に示す．乱流場の *DNS* 計算の時間発展計算部において，次の処理がなされる．

1. フーリエ変換後の各流速 u，v，w，及び熱 t を用いてそれぞれの対流を計算
2. Courant 条件 [54] の計算
3. 手順1によって得られた全対流を用いて，流速及び熱の全エネルギーを計算
4. 手順1によって得られた全対流を用いて，各流速 u，v，w，及び熱 t に対する擬スペクトル法 [51] とエイリアシング誤差の除去 [9] を施す

表 4.1: 乱流場 *DNS* 計算で用いる主要配列

x y z	Grid intervals and coordinates in the three dimension
dist_x dist_y dist_z	The temperature at the water surface or wall
u v w	Flow velocities in the x, y, z direction
fu, fuo fv, fvo fw, fwo	(a convective term) + (a viscous term)
t	The temperature
ft fto	(a convective term) + (a viscous term) in the energy equation of the temperature
p	(pressure) / (density)

5. 2 次精度の Adams-Bashforth 法 [54] による流速 u , v , w の統合

6. step 5 によってえられた統合値を用いて，圧力項 p のポアソン解法をおこなうために，フーリエ変換と 3 重対角行列の連立一次代数方程式の直接解法 *TDMA* （3 重対角行列法）を使った直接解法の計算

7. 圧力項 p を用いて，流速 u , v , w , 及び熱 t の更新

手順 1 , 4 , 7 において，配列 t , u , v , w に関する計算時間は全て等しい．手順 2 と手順 3 の計算時間は，手順 5 と手順 6 の計算時間の 1/3 である．これらの計算時間の関係より，配列 t の計算ならびに手順 2 , 3 の計算を 1 個のプロセッサでおこない，同じタイミングに，3 個のプロセッサを用いて，各配列 u , v , w をそれぞれ計算し，手順 5 による統合後，手順 6 の計算を strip mining することにより並列実行する．

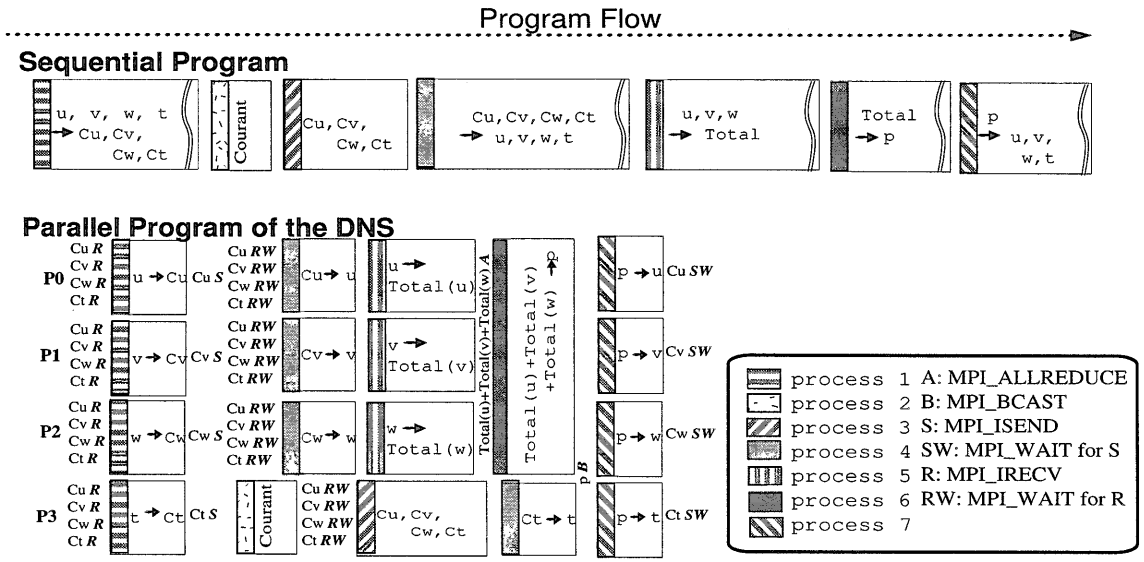

図 4.1: 乱流場の *DNS* 計算に対し，並列化手法 *B* を適用した場合の並列プログラムの概念図

手順 1，4，7 において，各配列要素のデータを処理するにあたり，必要なデータは，その隣接する配列要素であるため，他の配列を計算するプロセッサからデータ通信をする必要性はない．手順 3，4 において，手順 1 の計算によって得られた各配列の対流値が全てのプロセッサにおいて必要となる．そこで，手順 1 の計算終了後，各プロセッサは，計算された各対流値を非同期式に送信し，手順 2 の計算終了後，非同期式に受信することとする．手順 5 に関して，各流速 u，v，w それぞれにおける統合後，全流速の統合を並列化ライブラリ *MPI* の同期式通信関数 *MPI_ALLREDUCE* を用いておこなう．手順 7 において必要な圧力項 p は，手順 6 において計算される．そこで，手順 7 における各配列更新の計算時間は等しいので，同期式ブロードキャスト通信 *MPI_BCAST* によって得ることとする．これらのことを踏まえて乱流場の *DNS* 計算を並列化すると，図 4.1 のようになる．

以上より，並列化手法 *B* を *DNS* に適用する場合，プロセッサ数が 4 の場合，最も有効になると考えられる．

4.2　多数のプロセッサを用いた乱流場の直接数値シミュレーションの並列化

第4.1節において述べた適用方法では，4プロセッサを用いることによって，並列化手法 B を有効に活用することができる．しかし，乱流場の *DNS* 計算を並列化する際，大規模計算と大規模データをさらに分散させるためには，プロセッサ数4というのは，少な過ぎる．そこで，より多くのプロセッサを用いて並列化可能となるように，第4.1節で説明した適用方法を拡張する必要がある．

乱流場の *DNS* 計算に関して，時間発展計算部分の各手順において，各配列計算は，多重ループを用いておこなわれ，多重ループの最外郭の *do* 文は，イタレーション間に依存関係を持たない．そのため，strip mining によって等分に分散させた場合，並列化による有効性を失うことはない．ゆえに，プロセッサ数が $4n$（n ： 自然数）であった場合，プロセッサ数4用に分散された計算をそれぞれ n 個の部分に strip mining することによって，並列化手法 B の有効性を失うことなく拡張することが可能である．

プロセッサ数が $4n$ の場合，各配列要素のデータを処理するにあたり，その隣接する配列要素が必要となるため，プロセッサ間での通信が必要となる．この場合，配列要素は strip mining によって等分割されているため，同期式通信を用いても支障はないように思われる．しかし，実際には，プロセッサにおいて，隣接する配列要素を計算するプロセッサとの間で，送信と受信の両方の通信が同時に呼び出されるため，同期式通信を用いた場合，図4.2のような状態になる．図4.2では，下向きが時間経過の方向を表す．図4.2の (a) の場合，通信ブロッキングが生じ，実行継続が不可能となる．これに対して，図4.2の (b) の場合，送信関数と受信関数の呼び出しを交互にすることによって，隣接する配列要素を交換することが可能となるが，通信オーバーヘッドが生じる可能性がある．そこで，隣接する配列要素を計算するプロセッサ間での通信は，非同期式通信を用いるものとする．

4.3　計算機実験と性能評価

PSM を用いた乱流場の *DNS* 計算のためのプログラムとして，逐次プログラムと，並

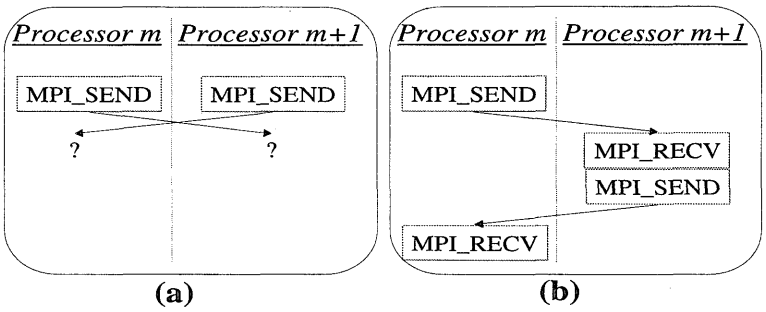

図 4.2: プロセッサ m $(1 \leq m < n)$ とその隣接する配列要素を計算するプロセッサ $m+1$ との間での通信のタイミング例. (a) 互いに同期式送信 *MPI_SEND* を呼び出した場合. (b) 同期式通信のみで, お互いの隣接する配列要素を交換した場合

列化手法 B を用いた並列プログラムを実装し, 実際の有効性を検証した.

実験環境として, 1G Hz の Pentium III の CPU を持つ計算機を 52 台用意し, OS として Linux 2.4.2 (Redhat Linux 7.1), 並列化ライブラリとして MPICH 1.2.4 を用いた. 通信には, 100base-TX を用いた LAN 上の TCP/IP を利用した.

実験に用いたプログラムにおいて, 安定した乱流場のシミュレーション結果を得るためには, 時間発展部の計算部分の繰り返し step 数を 100,000 回程度に設定しなければならず, このための計算時間は, 約 1 週間かかる. しかしながら, 各 step における計算手順が変化することはないので, 並列化による性能評価をおこなうためには, 1 step の計算時間が計測できればよいこととなる. ここでは, 計測時間の揺らぎを考慮して, step 数を 100 回とした場合の計算時間を評価することとした.

乱流場の *DNS* 計算をするための逐次プログラムの user 時間と system 時間は, それぞれ 2,632 秒, 2 秒であった. 逐次プログラムを実行した場合でも, 並列プログラムを実行した場合同様, 若干の system 時間が生じている. この system 時間は, ファイルの読み書きによるものであると思われる.

並列プログラム (プロセッサ数 4 , 8 , 16 , 32 , 52) の user 時間と system 時間を図 4.3 に示す. プロセッサ数 4 , 8 , 16 , 32 , 52 において, user 時間と system 時間を合わせた時間のうち最大のものは, それぞれ 747 秒, 462 秒, 306 秒, 241 秒, 230 秒であった. これは, 逐次プログラムの時間に対する比で表すと, 0.28 , 0.17 , 0.11 , 0.09 , 0.08 である. 比率がリニアにならなかった理由として, system 時間に該当する通信オー

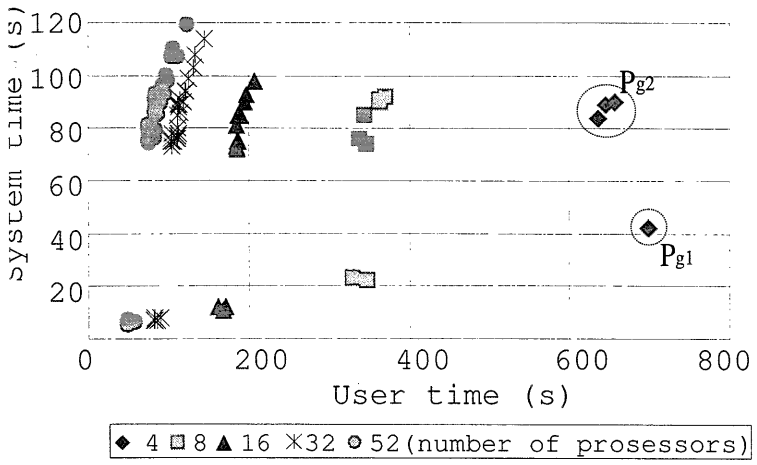

図 4.3: 乱流場 *DNS* の計算を並列プログラムでおこなった場合の user 時間と system 時間の関係

バーヘッドの増加があげられる．user 時間のみに着目すれば，理想時間（逐次プログラムの実行時間 / プロセッサ使用数）とほぼ等しい．

4 個のプロセッサを用いた並列プログラムに関して，図 4.3 に表されている一番右下のプロセッサ P_{g1} の user 時間が最も長い．この原因として考えられることは，乱流場の *DNS* 計算における結果の標準出力をプロセッサ P_{g1} に集中させたことである．また，残り 3 つのプロセッサ P_{g2} に関して，system 時間が長い．これは，第 4.1 節で説明した計算部分に関して，手順 5，6 で用いたデータ通信後に計算を必要とするタイプの通信 *MPI_ALLREDUCE* と多数のブロードキャスト通信 *MPI_BCAST* によるものであると思われる．

8 個以上のプロセッサを用いた並列プログラムに関して，図 4.3 より，4 個のプロセッサを用いた並列プログラムの実行時間が等分に分割されていることがわかる．プロセッサ P_{g1} に対応するプロセッサに関しては，system 時間が減少している．これは，プロセッサ数が増加したことによって，一度に通信されるデータサイズが小さくなったためであると考えられる．一方，3 個のプロセッサ P_{g2} に対応するプロセッサに関して，system 時間があまり変化しなかった原因としては，3 個のプロセッサ P_{g2} によって計算されていたデータを細分化することにより，*MPI_ALLREDUCE* の実行時間が長くかかることと，*MPI_BCAST* の回数が増加したことがあげられる．

第5章　まとめ

本研究において，自由表面乱流場における熱物質輸送を解析するために，逐次プログラムで開発された擬似スペクトル法（*PSM*）を用いた直接数値シミュレーション（*DNS*）の特徴を考慮し，分散メモリ型並列計算機用の並列化ライブラリ Message Passing Interface を利用して，プロセッサ数 $4n$ （n ： 自然数）用の並列プログラムを開発した．

計算機物理学で扱う微分方程式の作用が，格子点密度での解法である陽解法や陰解法のように，時間・空間的に近傍部分の影響ですむという前提条件を持つ場合，従来の全てのプロセッサに対して strip mining によってループ文を等分割するという並列化手法 *A* は有効である．しかしながら，計算技法として，本研究が対象とするような周波数空間での解法である *PSM* を用いた乱流場の *DNS* 計算をする場合や，配列をランダムにアクセスしなければならないような計算をおこなう場合，この前提条件が崩れるため，本研究で提案した通信プロトコルの種類を必要に応じて選択し複数のループを同タイミングに実行させることが可能な分割手法 *B* が有効になると考えられる．

PSM を用いた *DNS* 計算に対する並列化手法 *B* の性能を評価するために，データ通信コストを考慮して，並列プログラムを実装した．実験結果から，提案された並列化手法 *B* を用いた場合，user 時間に関して，ほぼリニアな結果が得られた．また，system 時間に関して，並列実行時間として現実的な環境で使用しうる結果が得られた．

本研究で用いた実験環境は，輻輳が大きく信頼性がそれほど高くない TCP/IP の下で実行をおこなわざるをえなかった．それにもかかわらず，このような環境下においても有効な結果が得られたことの意味は大きい．本研究の成果は，エントリレベルのネットワークと PC ワークステーションの組み合わせという比較的性能の劣った環境においても並列計算が有効であることを示したものと考えられる．

第III部

i アプリを用いた並列化の可能性と島モデルを用いた遺伝的アルゴリズムの実装

第1章　はじめに

情報化社会の高度化が進むにつれ，エンドユーザが用いる情報端末はダウンサイジングされる傾向にある．現在，主に使われている情報端末としては，計算および通信を行うためのリソースを有したパソコン・ 携帯情報端末（ PDA ： Personal Digital Assistance ）・携帯端末などが挙げられる．

これらの情報端末のうち，パソコンや PDA は，比較的複雑な計算が可能な反面，典型的なエンドユーザにとって，キーボード等の入力を必要とするアプリケーションの入手・設定・操作・管理は複雑なものとなっている．そのため，エンドユーザが，自立的に，新規のアプリケーションを利用することは少ない．

それに比べて，携帯端末は，処理性能が劣るもののアプリケーションの入手方法等が単純化されている．また，アプリケーション等の個人設定においても，操作が限定されているため，壊れる心配をすることなく，手軽に使用できるという利点を持つ．そのため，幅広い世代の人々が，個人の趣味にそった設定等を独自に行い，情報端末として利用している．ゆえに，携帯端末の普及は，今後，世界規模に発達していくものと考えられる．

携帯電話会社 NTT Docomo [47] が提供する携帯端末では，通信機能として，通常の音声通話を行う電話としての機能と，ネットワーク接続を行う i モード [26] との 2 種類がある．第 III 部で研究対象とした i アプリ [20] は，この i モードを用いてサーバから実行可能な Java^{TM} プログラムをダウンロードし，携帯端末本体で実行させるという形をとる．i アプリ対応の携帯端末は，2001 年の *503i* シリーズ発表以来，急速に市民権を得て，すでに 2004 年 12 月では 2,100 万台以上が市場に流通している [23] ．

i アプリ対応の携帯端末は，計算機としては貧弱ではあるが，ネットワーク機能とプログラミングツールを用いてユーザが自由に使える計算機としてのリソースを有している．この計算機としてのリソースを利用する様々な i アプリが開発されているが，実際に利用されているものは，画像・音声の再生，ゲーム，占いなどのエンターテイメント性の強い

ものと，時刻表，スケジュール帳，株価情報，地図・渋滞情報の表示などの社会的実用性の強いものが主である．これらのiアプリは，どちらも計算を行う主体としてのアプリケーションではない．また，待ち受け画面用のiアプリ以外は，常時実行を必要としないため，現在流通しているほとんど全ての携帯端末でiアプリ用の計算リソースが余っている状態であると考えられる．そこで，本研究では，iアプリを用いた分散処理を行うことによって，アイドル状態にある圧倒的多数の携帯端末の余剰計算リソースを利用した大規模数値計算の可能性を検討することを目的とする．

携帯端末で大規模数値計算を行うためには，分散処理を行う方法が考えられる．この際，十分な数の携帯端末を確保しなければならず，場合によっては数万台のオーダの携帯端末が必要となることもある．これらのユーザを確保するためには，分散処理を待ち受けiアプリのバックグラウンド処理として行うコンテンツ等の提供や換金システムの確立等を行うことで対処可能であると考えられるが，第III部の研究対象範囲外である．

本研究では，数値計算の第一歩として，遺伝的アルゴリズム（GA：Genetic Algorithm）[18] の手法の一つである分散GAのうち島モデル [60] を用いたものを *503i*, *504i* シリーズに実装し，携帯端末の計算可能性と性能評価を行う．

以下，第III部の第2章において，iアプリの特徴について簡単に述べる．第3章において，GAの仕組みについて説明する．第4章において，iアプリを用いた数値計算の可能性を検証するための一つのアプローチとして，島モデルのGAの諸手順をiアプリ用に変更する．iアプリを用いて島モデルのGAが実行可能であるかどうか第5章において，ナップサック問題を解くことにより性能評価をおこなう．

第2章　i アプリ

携帯端末上で利用可能なデータ通信やアプリケーションの実装には，Java^{TM} 言語が用いられている．そのため，一般的なプログラマにとって，多様なアプリケーションを容易に開発することが可能である．ただし，省電力およびメモリ等の制約条件が厳しい機器向けの J2ME 規格 [65] に従ったプログラムを記述しなければならないので，これらの制約条件には留意する必要がある．

現在，Java 仮想計算機が搭載された携帯端末は，携帯電話会社 NTT Docomo が提供しているｉアプリ対応端末以外に，ボーダフォン株式会社 [74] のフェーズ 1，2 [15] の対応端末，KDDI 株式会社 [30] の携帯端末 AU [4] の ezplus 対応端末がある．これらの Java 仮想計算機は，それぞれ異なるの API （ Application Program Interface ）が定義されている．ｉアプリは，DoJa [20] と呼ばれる仕様が採用されており，ｉモードの機能を最大限に発揮することが可能である．それに対し，フェーズ 1，2，ezplus は，MIDP （ Mobile Information Device Profile ）と呼ばれる仕様が採用されている．MIDP は，PDA の一種である Palm [52] でも採用されている Java^{TM} である．そのため，将来的に携帯端末のみならず PDA も含めた分散処理を考慮すると，MIDP 採用端末を研究対象としたほうが望ましい．しかし，携帯端末のみに着目すると，現在の携帯端末の市場流通台数を考慮すると，ボーダフォンもしくは Docomo が提供する携帯端末を研究対象とすることが好ましい．また，Java アプリケーションのダウンロードができるサーバは，Docomo の場合自由に設定できるのに対して，ボーダフォンの場合コンテンツアグリゲータと呼ばれるボーダフォン専用の Java アプリケーションを収集・審査・管理するサイト運営サーバに限定されている．そのため，より容易に研究開発が可能な Docomo が提供するｉアプリを本研究の対象とする．

以下，第 2.1 節では，ｉアプリの種類について説明する．第 2.2 節では，ｉアプリの機種性能について述べる．分散処理による数値計算の可能性については，第 2.3 節で説明する．

2.1 iアプリの種類

iアプリは，2種類に分類することができる [3]．

一つは，非接続型であるスタンドアロンタイプである．このタイプのiアプリは，iアプリ実行中，ネットワーク通信を必要としない．つまり，このアプリケーションは，携帯端末のみで動作可能である．ただし，最初のiアプリダウンロード時だけは，ネットワークに接続し，パケット通信がおこなわれる．

もう一つは，データ更新にネットワーク接続を必要とするサーバ連動タイプである．このタイプは，サーバとの通信を介して，他の携帯端末やサーバとのデータ通信が可能である．つまり，サーバを経由することで，複数の携帯端末間で，データ共有が可能となる．この際の接続方法には，ユーザの手動による方法と，iアプリ自体が一定時間間隔で自動的におこなう方法がある．ただし，データを通信する場合以外，ネットワークに携帯端末が接続している必要性はない．また，将来的には，Bluetooth^{TM} などが普及すれば，サーバを介さずに携帯端末同士のみで情報交換をおこなえる可能性があるが，大規模計算を携帯端末のような非常に小さな計算リソース上で実装するためには，データはどこかで一元管理されているシステムのほうが望ましいと考えられる．そのため，サーバを介してデータ共有をおこなうほうが現実的である．

2.2 iアプリ搭載の携帯電話の性能

携帯端末の種類によって，プログラムサイズや使用可能なファイルタイプの制限があるものの，シリーズが進むにつれ，表2.1に表されているように，標準的なスペックは向上している．

表2.2は，市販されている数種類の携帯端末上で，付録Aのiアプリを実行し，MIPS値を測定した結果である．得られたMIPS値は，各社ばらつきがあるものの，*504i* シリーズ以降の機種は数MIPSの処理速度を持つと言えるだろう．これは，1980年代に使われたDEC社のVAX-11/780シリーズやSun Microsystems社の初期のSPARCプロセッサに相当するものである．

プログラムの実装において，記憶領域としてヒープが必要となる．表2.3は，各携帯端末シリーズのヒープ容量を表す．ただし，ヒープ容量が，Javaのオブジェクトなどを格

表 2.1: *503i*, *504i*, *FOMA*, *505i* の標準スペック (KB)

	503i	504i	FOMA	505i
通信サイズ (上り)	5	5	5	10
(下り)	10	10	10	20
.Jar 容量	10	30	30	30
Scratch Pad 容量	10	100	200	200

表 2.2: 各携帯端末における MIPS 値

	503i	503is	504i	504is	505i
Fujitsu (F)	0.11		2.56	2.56	
Panasonic (P)			0.92	0.93	
NEC (N)	0.21	0.21	10.0	10.0	
MITSUBISHI (D)			0.47	-	7.35
Sony (SO)	0.60	1.11	1.11	-	1.48

納する Java ヒープと，メディアデータや画面表示のためのリソースを格納するネイティブデータヒープの区別がある場合は「java ヒープ / ネイティブヒープ」の順序で記載し，区別がない場合は Java ヒープ容量を記載している．

2.3 携帯端末を用いた分散処理の可能性

一般的な分散処理システムは，単体の計算機で処理するには過大な計算量やデータ量を持つアプリケーションをネットワークに接続された複数の計算機を利用して実行する．ただし，この場合の計算機とは，一般的に使用されているパソコン，ワークステーション，もしくはクラスタ型コンピュータを構成するユニットであり，携帯端末よりもはるかに高い計算能力を持つ反面，その構成台数は数 100 台程度のオーダで考えられている．

一方，i アプリ対応の携帯端末は，低性能ながらも，普及台数は 2,100 万台以上と非常に多く，それらのうち，一部しか参加者がいなかったとしても，数万台の携帯端末を分散処理のユニットとして確保できる可能性がある．また，携帯端末ユーザは，普段使用して

表 2.3: ヒープ容量 (KB)

	Fujitsu (F)	Panasonic (P)	NEC (N)
503i	600	288/150	96/286
503is	600	288/150	96/286
504i	1,000	1,500	544/600
504is	1,500	1,500	544/1,030
2051	600/700	-	760/1,228
2102V	1,536/1,024	760/1,228	760/1,488
505i	2,000/1,500	3,500	1,494/1,472
	MITSUBISHI (D)	Sony (SO)	Sharp (SH)
503i	1,110/220	350	-
503is	1,110/220	350	-
504i	1,024/524	1,000	-
505i	1,536/2,048	1,800	1,700

いない余剰計算リソースのみを提供するだけなので，比較的負担が少ないと考えられる．さらに，ボーダフォンや AU においても，Java^{TM} のプログラムを実行可能な機種が開発されている．そのため，今後，分散処理のユニットとして利用可能な携帯端末数は，増加の一途をたどるものと考えられる．ゆえに，一つ一つの携帯端末の計算リソースが小さいとはいえ，その余剰計算リソースを利用し携帯端末間で分散処理することによって，様々な研究のための数値計算をおこなうことが可能であると考えられ，かなりの成果が期待できる．

すでに，パソコン上では，この余剰計算リソースを用いて解探索をおこなうというコンセプトは，実現されており，電波望遠鏡で得られたデータをスクリーンセーバのバックグラウンド処理で解析するための SETI@home [63] や，タンパク質構造解析に用いられる Folding@home [12] などがよく知られている．

本研究では，貧弱な計算能力しか有さない携帯端末を利用して実用的な数値計算を可能とするための分散処理の枠組みを提案する．ただし，データを一元管理するためにサーバを介した通信が必要となるため，Master-Slave 型の分散処理システムを構築するものとす

る．このプロトタイプとして，GA の一種である島モデルを i アプリ用に変更し，その有効性を検証する．

第3章　遺伝的アルゴリズム

i アプリを用いて数値計算が可能かどうか検証するために，遺伝的アルゴリズム（ GA）を用いる．そこで，本章では，遺伝的アルゴリズムについて簡単に説明する．

第3.1節では，一台の計算機用に開発された GA について説明する．第3.2節では，複数の計算機用に開発された分散 GA のうち，島モデルと呼ばれる GA について述べる．

3.1　遺伝的アルゴリズム

GA の手順として，一般的に，次のような手順がとられる．

1. 母集団の初期生成
2. 収束条件を満たしているならば，実行終了
3. 母集団から個体を選択
4. 交叉
5. 突然変異
6. 各個体の評価および子集団の決定
7. 手順 2 に戻る

この手順は図3.1のように表される．これらの手順のうち，最も重要となってくるのが，オペレータ（選択・交叉・突然変異）である．

以下，第3.1.1項において，個体の設定について述べる．選択に関する各種の説明は，第3.1.2項で述べる．第3.1.3項では，交叉について説明する．第3.1.4項では，突然変異に関して述べる．

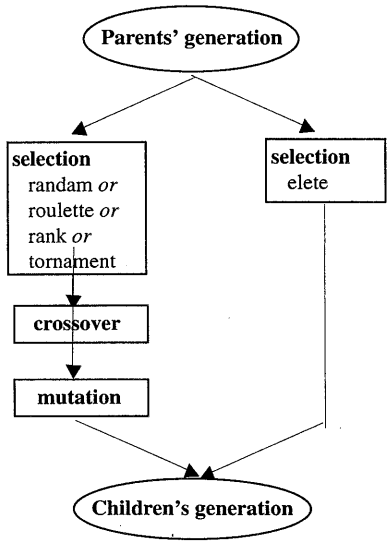

図 3.1: 親世代から子世代にいたるオペレーション方法

3.1.1 個体の諸設定

対象とする問題を GA として扱う場合，問題空間の変数は，遺伝的操作に適した形でコード化され，それに合わせて適応度の式を決定する必要がある．

コード化において，もっとも扱いやすく，初心者でも利用可能なものとして，バイナリ・コーディング [37] と呼ばれる方法がある．バイナリ・コーディングにおいて，問題空間の変数は，固定長のビット列から成る 2 進数に変換される．現在の GA の理論の多くは，固定長で固定オーダのバイナリ・コーディングの仮定に基づいている．そのため，適切なパラメータの値の推定の際によく使われるコード化である．

しかし，多くのアプリケーションにおいて，問題空間の変数の基の表記方法は，アルファベットや実数である．そのため，グラフ生成方法に対する文字表現を使ったコード化 [31]，条件設定に対する実数値表現を使ったコード化 [35]，ニューラルネットワークの重みに対する実数値表現 [38]，タンパク質のねじれ角に対する実数値表現 [62] 等が存在する．

コード化のための各種表現は，問題空間の性質や GA の細部に大きく依存している．そのため，どのコード化が最良であるか示す指針はない．

コード化に合わせて決定しなければならないのが，適応度関数である．適応度関数は，

問題空間を数式で表すため，複雑な多変数関数となることが多い．また，適応度関数は，コード化された個体情報を利用するため，コード化のための各種表現と密接な関係を持つ．この適応度関数を使って適応度を求める．適応度とは，各個体が問題空間において，最適解への近さを調べる際に用いられる指標である．

3.1.2 遺伝的アルゴリズムにおける個体の選択方法

次世代の個体を生成するために，交叉・突然変異される個体を選択しなければならない．母集団から個体を選択する手法は様々であるが，一般的な手法としては，ランダム選択，ルーレット選択，ランク選択，トーナメント選択がある．また，優良解を優先的に次世代に残すための手法として，エリート選択がある．

以下，各選択手法について述べる．

ランダム選択

個体の性質・適応度に関係なく，個体をランダムに選択する手法である．そのため，通常の生物世界では，優秀な個体がより選ばれる状況であるのに対して，個体数 M とすると，式 (3.1) の確率 P_j のように全ての個体が平等に選ばれる．

$$P_j = \frac{1}{M} \tag{3.1}$$

この手法では，問題空間の広域探索は可能となるが，解に到達できない可能性が高い．そのため，他の選択方法と同時に適用されるべきである．

ルーレット選択

GA を実装する際，最も一般的に用いられている手法である．この手法は，適応度 F_j が高い個体 j ほど選択される確率 P_j が高くなるように設定された手法である．一番単純に確率 P_j を得るためには，総個体数 M の集団に対して，以下の重み付けの式 (3.2) が計算される [18] .

$$P_j = \frac{F_j}{\sum_{k=1}^{M} F_k} \tag{3.2}$$

この手法を用いることによって，生物進化と同様の選択がなされる．しかし，同じ遺伝子情報を持つ個体が複数含まれていた場合，その個体が選択される確率が異常に高くなり，他の個体が選ばれにくくなるという問題点もある．

ランク選択

各個体の適応度に対して，ランク付けをおこない，そのランクに対して一次減少関数を計算し，その結果を整数化することによって，選択確率を決定する方法である [5]．この方法によって，似通った適応度を持つ個体同士であっても，より適応度の高い個体が確実に選ばれることとなり，収束が速くなることが期待される．ただし，この手法は，初期設定に対する依存性が高く，局所解に陥る可能性が高い．

トーナメント選択

母集団から，ある S 個の個体を選択し，それらのうち，最も適応度の高い個体を選択する手法である．S 個の個体は，ランダムに選ばれているため，ルーレット選択やランク選択に比べ，多様性は失われにくい．しかし，ランダムに選択された S 個の集団から最も適応度の高い個体を選択するため，近似した適応度を持つ個体が計算の初期段階において母集団の大部分を占める可能性がある．また，1 つの個体を選択するためだけでさえ，S 個の個体をランダムに選択し，適応度の比較をする必要が生じるため，他の選択手法と比べ，計算時間は長くなる．

エリート選択

この選択手法を用いる場合，母集団から，適応度の高いほうから T 個の個体を無条件に次世代に残すことを決定し，残りの個体は，エリート選択以外の選択手法によって選択し，世代交代していく．このことによって，適応度の高い個体が交叉や突然変異によって破壊されることを防ぐことが可能となる．そのため，探索能力が優れているといえるが，親世代において適応度の順に並べ替える必要があることや，探索が局所解に陥る場合がある等の問題がある．

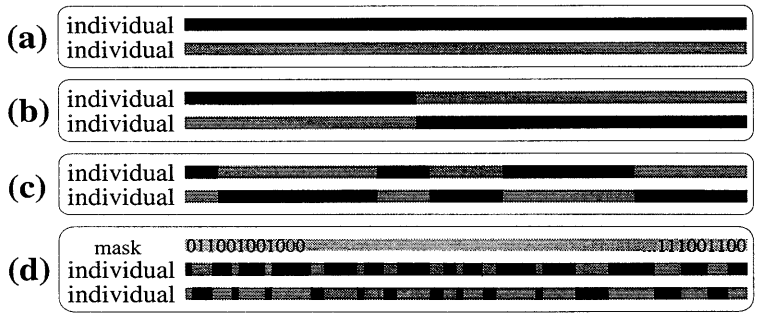

図 3.2: 選択された 2 個体に対する交叉の適用．(a) 選択された個体 (b) 一点交叉 (c) 多点交叉 (d) 一様交叉

3.1.3 交叉について

交叉オペレータは，よい形質をもつ染色体の一部を交換する操作であり，この操作によって，より高い適応度を持つ個体を生成する効果が期待される．通常，2 つの個体を選択し，0 から 1 の実数の乱数値が交叉率 P_c (%) を下回った場合のみ交叉をおこなう．

交叉方法には，一点交叉，多点交叉，一様交叉の 3 種類がある．図 3.2 は，各交叉方法の模式図である．(b) に表されるように，一点交叉は，個体遺伝子のある一点から後ろの遺伝子を互いに入れ替える手法である．多点交叉は，選択された地点のうち 2 点によってはさまれた遺伝子を入れ替える手法であり，(c) のようになる．一方，一様交叉では，染色体長と同じ長さのランダムにラベルを割り当てたマスクを作成し，このマスクに基づいて (d) のように交叉をおこなう [21] .

これら 3 種類の交叉手法のうち，一様交叉が最も探索能力に優れているといわれている．しかし，親個体の性質をできるだけ継承するためには，一点交叉を用いるほうが好ましい．

3.1.4 突然変異について

突然変異は，母集団の多様性を維持するために用いられる．ただし，突然変異率 P_m(%) が高すぎる場合，適応度の高い個体までも破壊され，最適解を得ることが困難となることがある．突然変異率を高くする場合，第 3.1.2 項で説明したように，適応度の高い個体を

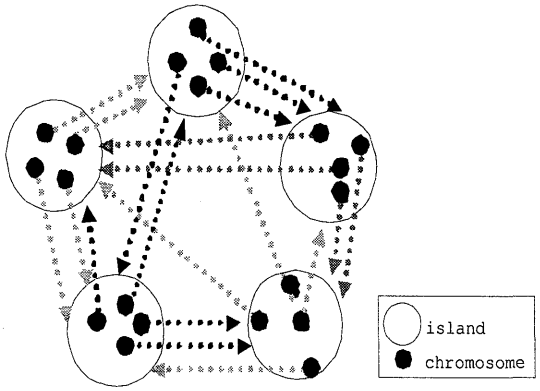

図 3.3: 各個体の移住例

無条件に次世代に残すことが可能なエリート選択を同時に適用することが好ましいと考えられる.

また, 母集団に含まれる個体が多様な場合個体, 突然変異率を高くする必要性はないものと考えられる.

3.2 島モデルの遺伝的アルゴリズム

一般に, GA がその真価を発揮するのは, 遺伝子情報が非常に多い場合で, 他の探索手法では適切な遺伝子を決定することが困難な場合が多い. この場合, 遺伝子長が長くなり, 優良解を得るためには GA の各世代における個体数を多くすることが必要である. 母集団に含まれる個体数が多ければ, 探索範囲を広くとり, 多種多様な個体を保持することが可能である. しかし, 母集団が大きくなり過ぎると, 計算時間の増加やメモリ使用量の増大といった問題が生じる. この問題を回避し, 探索空間を広くとるための方法として, 島モデルと呼ばれる GA がある [32] .

島モデルの GA は, 母集団を複数の部分集団 (島) に分割し, 島ごとに遺伝的操作を適用する手法である. また, 一定の世代更新が終了するごとに, 他の島と個体を交換する. この操作は, 移住 (migration) と呼ばれる.

島モデルの GA は, いくつかの島でそれぞれ独自に進化させ, ある島において個体が全て局所解に陥ったとしても, 移住を繰り返すことによって, 局所解への収束を避けるこ

とを可能とする．そのため，通常の単一母集団として実装された GA と比較して，より適応度の高い優良解を発見することが可能であると報告されている．

実装にあたって，各島を各計算ユニットに割り当てることによって，分散処理が可能となる．このことによって，移住の際，異なる計算ユニット間で通信が必要となる．

図 3.3 は，一般的な移住の仕組みである．移住は，数世代ごとにおこなわれる．その際，移住する個体は，ランダムに選択され，移住先もランダムに選択される．

第4章　iアプリを用いた遺伝的アルゴリズム

iアプリとしてGAを実装する場合，機種固有のマルチメディア系APIを用いる必要がないため，機種別にiアプリを実装しなくてもよい．また，選択・交叉・突然変異といったGAの操作[18]は，一般的な計算機用に実装する場合同様，iアプリ用として実装することが可能である．ただし，iアプリは，小型デバイス用のJavaであるJ2ME規格に従っているため，浮動小数点演算を用いることができない．すなわち，整数演算のみを用いてアルゴリズムを構築しなければならない．

以下，第4.1節において，iアプリを用いた島モデルのGAを実装する際の注意点と概念図について説明する．第4.2節において，第3.2節で説明した一般的な計算機用の島モデルのGAをiアプリ用に変更する．

4.1　iアプリに実装するための注意点

iアプリに対する実装において，各個体の遺伝子情報は携帯端末上のヒープ領域に保持される．しかしながら，表2.3に示されているように，十分な個体数を持つ母集団を保持することは難しい．また，携帯端末の場合，一般的な計算機と比較して，演算速度も著しく劣るため，GAの実行時に取り得る個体数に関しては，大きな制限を受ける．このような制限のために，一つの携帯端末で完結したGAをおこなうよりは，複数の携帯端末で島モデルのGA[60]と呼ばれる分散協調型のGAを実装したほうが，実装上有利であると考えられる[69]．

島モデルのGAを実装する場合，移住をおこなう必要があるが，各島単位である携帯端末間では，現在のところ，直接通信をおこなうことは難しい．IrDAなどの赤外線通信

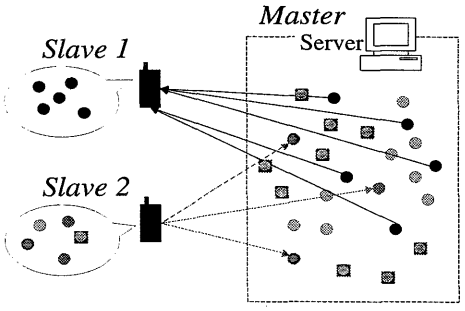

図 4.1: iアプリ用の島モデルの GA の概念図

や Bluetooth^{TM} などを使用することも考えられるが，これらの機能が存在しない機種もあるため，あまり現実的ではない．より遠くの携帯端末と情報交換をおこなうためには，何らかの通信サーバを介するほうが現実的である．そこで，我々のiアプリを用いた島モデルの GA の実装においては，Master-Slave 型の分散処理を用いることが可能となるように，島モデルの GA を変更することとした．

我々の用いた Master-Slave 型の島モデルの GA は，図4.1のように表される．この際，母集団は Master であるサーバ内におかれる．Slave である各携帯端末では，母集団から取り出された部分集団を用いて，世代更新をおこなう．各携帯端末で得られた更新結果は，サーバに送信され，サーバでは，更新結果をもとに，母集団の更新をおこなう．サーバと携帯端末間においては，通信をおこなう必要があるが，この通信は HTTP プロトコルを用いた．iアプリ対応の携帯端末は，DoJa [20] と呼ばれる J2ME から派生した通信機器用の Java が実装されており，DoJa には，HTTP 通信のサブセットがライブラリとして提供されている．Web サーバをデータ共有のインターフェースとする利点は，iモードとの親和性だけでなく，新たにクラスライブラリを構築する必要がなく，実行時に少しでも多くのヒープを利用できることもあげられる．本実装方式では，通信の核となる部分は HTTP プロトコルにおける GET 要求および POST 要求によって実装する．

4.2 i アプリのための Master-Slave 型への拡張

本研究では，改良方法として，サーバに母集団を置き，携帯端末では，母集団から取り出した部分集団を用いて世代更新をおこなう．

このiアプリのために改良されたアルゴリズムは，図4.2のようになる．

1. （サーバ） N 個からなる母集団を生成し，ログファイルに保存
2. （携帯端末） i アプリを起動
3. （携帯端末→サーバ） GET 要求による個体取得リクエストの発行
4. （サーバ） 母集団から K （ $1 < K \leq N$ ）個の個体をランダムに選択
5. （サーバ→携帯端末）サーバから携帯端末に，選択された K 個の個体情報を通信
6. （携帯端末） GA のオペレーション（選択，交叉，突然変異）を G 世代目が生成されるまで繰り返す
7. （携帯端末）優良個体を J （ $1 \leq J \leq K$ ）個選択
8. （携帯端末→サーバ） POST 要求を用いて，携帯端末からサーバに，選択された J 個体を通信
9. （サーバ）受信した J 個体と母集団の N 個体をソート
10. （サーバ）ソートされた $N+J$ 個体から，適応度の高い N 個体を選択し，ログファイルに上書き保存
11. （携帯端末） i アプリを終了しない場合，手順 (3) に戻る

手順 3 から 5 は，各携帯端末からの GET 要求の呼び出しに応じて実行される．同様に，手順 8 から 10 は，各携帯端末からの POST 要求の呼び出しに応じて実行される．

今回の実験で用いた実装方式では，手順 8 で用いられる POST 要求は，GA の操作終了後，ユーザが手動で携帯端末モニター上の送信メニューを選択することによって実行さ

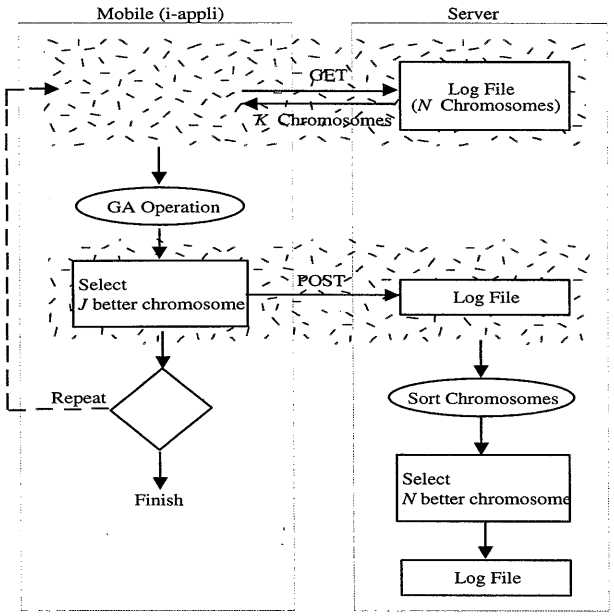

図 4.2: iアプリで GA を実行するためのアルゴリズム

れるように設定した．これは，電波圏外でiアプリが終了した場合，サーバに情報を送信できないという問題を回避し，確実に実験結果を送信するための措置である．今後の実用化においては，この POST 要求の呼び出しは，自動的におこなわれるようにされるべきである．

一般に，従来の島モデルの GA [60] では，ある島において個体が全てある特定の局所解に陥ったとしても，他の島からの移住によって，局所解から脱する可能性があるという特徴を持っている．改良された島モデルの GA において，同様の性質を持たせるために，サーバ側で母集団を管理し，携帯端末ではその一部分を用いて世代更新をおこなうこととする．つまり，ある携帯端末において得られた結果である J 個の個体は，局所解を示すものであったとしても，サーバ側の母集団において，一部分の個体として扱われる．さらに，次のiアプリ起動時に携帯端末に送信される部分集団は，母集団内からランダムに選択されるため，局所解のみでなく，他の携帯端末から母集団に送信された個体も含まれる可能性が高い．このため，母集団全体が局所解に陥る傾向があったとしても，そのスピードはそれほど速くなく，局所解からの脱出は十分に可能である．ただし，手順 10 において，適応度の高い個体を優先的に保存しているため，母集団の更新回数が増すほど，各個

体の多様性がなくなり，局所的な解に陥る危険性がなくなる訳ではない．

また，島モデルの性質より，携帯端末の結果が局所解であっても支障がないため，1 つの携帯端末上の GA 操作において，個体の多様性の維持を重視する必要はそれほどない．ここでは，K 個の部分集団を携帯端末に割り当てているが，K を数十個体程度にすれば，十分実用的な長さの遺伝子を割り当てることが可能である．世代更新回数 G に関しては，携帯端末の計算資源である演算能力や消費される電力等を考慮した場合，長時間世代交代させることはそれほど得策ではない．また，終了世代において，母集団に返される個体が未成熟であったとしても，それほど問題は起こらないと考えられる．ゆえに，世代更新回数 G を小さく設定しても支障はない．

第5章　携帯端末における実行結果

GA を携帯端末上で i アプリとして実装したものを用いて，実行結果が妥当なものであるか検証した．

使用した機種は，*SO504i*，*SO503is*，*N503i*，*P504i* である．携帯端末のシリーズが異なる場合，古いシリーズに合わせた i アプリを開発するか，各シリーズの性能を存分に利用するために，各シリーズ専用の i アプリを開発しなければならない．本研究において，実用的な数値計算が実行可能であることを検証するにあたり，各シリーズの性能を考慮しなくてすむように，古いシリーズである *503i* にあわせた開発をおこなった．

第5.1節では，ナップサック問題を解くための GA について定義する．第5.2節では，一台の携帯電話を用いた場合の実験結果について述べる．第5.3節では，複数の携帯電話を使って島モデルと呼ばれる GA を実行した場合の結果について述べる．

5.1　ナップサック問題を解くための遺伝的アルゴリズムの諸定義

本研究では，数値計算の実行可能性を検証するために，ナップサック問題を解くための GA を具体例として採用した．

以下，第5.1.1項において，ナップサック問題について簡単に説明する．第5.1.2項において，ナップサック問題を GA で解くための諸定義をおこなう．第5.1.3項において，ランダム探索よりも GA のほうが有効であることを示すための予備実験として，ランダム探索の一種であるマルコフ連鎖モンテカルロ法（MCMC：Markov Chain Monte Carlo）をおこなう．

5.1.1 ナップサック問題

ナップサック問題は，NP 完全問題であることが知られており，総当り方式で真の解を得るために必要な時間は，荷物の個数の指数オーダに比例した時間となる．そのため，荷物の個数が増えると，実用的な時間で解をみつけることが困難となる．

ナップサック問題とは，ナップサックの耐荷重量 W の制限内で，どの荷物をナップサックに詰めれば，最も価値を高くすることができるかということを判定する問題である．荷物の個数を L 個とし，それぞれに重さ w_j （ $1 \leq j \leq L$ ） と価値 c_j とが定義されているような状況において，ナップサック問題は，以下のように定式化される．

$$\begin{aligned} \max_{\{X_j\}} \quad & \sum_{j=1}^{L} c_j X_j, \\ subject \quad to \quad & \sum_{j=1}^{L} w_j X_j \leq W, \\ & X_j \in \{0,1\} \quad (j = 1, ..., L) \end{aligned} \tag{5.1}$$

ただし，変数 X_j は，荷物 j がナップサックに詰められている場合 1 とし，詰められていない場合 0 とする．

5.1.2 ナップサック問題用の遺伝的アルゴリズムの定義

GA を用いて実装するために，集団個体数を M とし，この中の i 番目の個体の遺伝子を $\mathbf{s}_i$ で表すものとする（ $1 \leq i \leq M$ ）．荷物をナップサックに入れるか入れないかは，1 bit で表現可能なので，i 番目の遺伝子 $\mathbf{s}_i$ の j 番目の要素を $s_{i,j}$ で表し，0 または 1 の 2 値を取るものとする．この場合，j 番目の荷物を入れる場合は $s_{i,j} = 1$，入れない場合は $s_{i,j} = 0$ として表すものとする．また，L 個の荷物を考えているので，遺伝子の長さは L となる（ $1 \leq j \leq L$ ）．

i 番目の個体の総重量 W_i は，

$$W_i = \sum_{j=1}^{L} s_{i,j} w_j \tag{5.2}$$

とする．このときの総価値 C_i は，

$$C_i = \sum_{j=1}^{L} s_{i,j} c_j \tag{5.3}$$

で表される．個体の適応度 F_i は，総価値とナップサックの制約条件

$$W \geq W_i \tag{5.4}$$

を考え，

$$F_i = \max\{0, C_i - \alpha(\max\{0, W_i - W\})\} \tag{5.5}$$

とする [60]．ただし，α は，ペナルティパラメータである．仮に，i 番目の個体の総重量がナップサックの耐荷重量を超過している場合，$W_i - W$ が正となり，ペナルティとして，超過重量と α の積を総価値から減じたものが適応度となる．ペナルティを強くする場合には，α を大きな値として適用すればよい．

GA の世代交代モデルには，単純 GA を用いた．単純 GA とは，親集団からの交配選択方法にルーレット選択を用い，生存選択には生成された子集団が次世代の親集団となるモデルである．したがって，携帯端末上では，世代交代によって優良解が破壊される可能性もある．しかし，各携帯端末の最終的な個体情報は，サーバ側に順序付けられた上で保持され，なおかつ，携帯端末が返す個体数 J とサーバ側の母集団の個体数 N には $N \gg J$ という関係が成り立つため，1 つの携帯端末における 1 試行の結果が悪くとも，サーバ側の母集団に与える影響はそれほど大きくないと考えられる．

親世代から子世代への移行にあたって，ルーレット選択による交叉と突然変異によって，新たに M 個体生成する．

交叉方法は，計算負荷を少なくし簡単におこなうために，一点交叉とする．この際，交叉地点は，ランダムに選択されるものとする．

各携帯端末においては，GA の最終世代の解を収束させる必要性はそれほどなく，むしろ，より幅広い探索が必要となると考えられる．そのため，ここでは，突然変異率を通常の単純 GA でおこなう場合よりも高めの 5% に設定した．

表 5.1: 各荷物の価値と重量（ $L = 16$ ）

価値 c_i	8	9	10	8	9	10	7	7	6	9	8	7	6	9	10	7
重量 w_i	10	6	7	7	8	9	6	9	9	10	8	7	8	7	10	9

5.1.3 ランダム探索による予備実験

突然変異率を高めに設定した場合，GA 的な要因よりもランダム探索的な要因が強くなることも考えられる．そこで，予備実験としてパソコン上において，上記ナップサック問題のランダム探索をおこなった．

この予備実験では，ランダム探索として，その一手法である MCMC を採用した．ナップサックの荷物の個数は $L = 16$ とし，500 ステップを 1 試行として実験をおこなった．これは，GA において 50 個体で 10 世代の探索に双頭すると考えられる．

MCMC 法による解探索を 100 試行おこなった結果，最適解が 104 であるのに対して，準最適解である 90 以上の個体は生成するものの，最適解を持つ個体を生成することはなかった．また，ほぼ全ての試行において，約 10 ステップで収束し，それ以降，準最適解から抜け出すことは無かった．

5.2 単純遺伝的アルゴリズムの実験結果

実行にあたって，ナップサック問題における荷物の個数 L は 16 個とした．各荷物の価値 c_i と重量 w_i は表5.1を用いた．また，ナップサックの耐重量は，$W = 100$ とした．この問題における最適解は，6 種類存在し，その総価値および適応度は 104 となる．

単純 GA であるため，母集団を携帯端末に保持させなければならない．そこで，母集団の個体数は $N = M = 50$ とした．携帯端末からサーバに返される個体数は，$J = 5$ とした．ペナルティパラメータは，遺伝子長と同じ $\alpha = 16$ とした．また，GA の終了世代数は，長時間の使用を避けるために，$G = 5$ とした．

J2SE を用いて単純 GA を通常のパソコン上で実行する Java アプリケーションと i アプリをそれぞれ 10 回実行した結果が，表5.2である．表5.2より，J2SE の場合も i アプリの場合も，最終世代における最大適応度の範囲は，ほぼ一致することが分かる．また，同じ最大適応度であっても，異なる遺伝子情報を持つ個体が結果として得られた．同一機

表 5.2: 単純 GA を用いた場合の J2SE と i アプリの 10 回分の実行結果

J2SE	87	91	97	98	97	93	95	96	96	98
i-APPLI	98	89	95	91	91	83	85	97	97	92

種で実行した場合も，実行ごとに異なる最大適応度と遺伝子情報が得られた．つまり，実行ごとに異なる個体が生成されているものと考えられる．

i アプリの携帯端末上での実行時間を計測するために，*SO503is* を用いた．単純 GA を i アプリを用いて実行した場合，平均実行時間は，サーバから携帯端末への初期データ通信と携帯端末用への受信データのデコード等の前処理に関して 14 秒，GA の実行に関して 3 秒，最終世代の結果の送信に関して 3 秒であった．前処理に関する時間のほうが，最終世代の結果の送信時間より長い．これは、最終世代の結果として送信される個体数が $J = 5$ であるのに対して，前処理では個体数 $N = M = 50$ が受信されるためであると考えられる．

以上より，i アプリを用いた場合も，J2SE を用いて GA をおこなった場合同様の有効性があるものと考えられる．ただし，一台の携帯端末のみで GA を実行した場合，問題の規模や探索空間の広さに応じて，ヒープ容量や安定した通信を可能とするデータ量等を考慮して，遺伝子長や終了世代を変更する必要がある．つまり，現在のヒープ容量等の携帯端末性能を考慮すると，1 台の携帯端末では，小規模な GA の探索しかおこなえないという問題が残る．

5.3 島モデルの遺伝的アルゴリズムの実験結果

実行にあたって，ナップサック問題と GA に関する定数は，第 5.2 節のものと同値を用いた．ただし，島モデルの GA において，母集団はサーバで保持されるため，大規模な $N = 200$ とし，携帯端末で実行する小集団の個体数は $K = M = 50$ とした．また，島モデルの GA では，未発達な状態で終了しても支障がないため，終了世代数は，$G = 5$ とした．

図 5.1 は，i アプリの連続 5 回の繰り返しを 10 セットおこなった結果として得られた母集団の最大適応度の推移を表す．図中，横軸は i アプリの繰り返し回数を表し，縦軸は

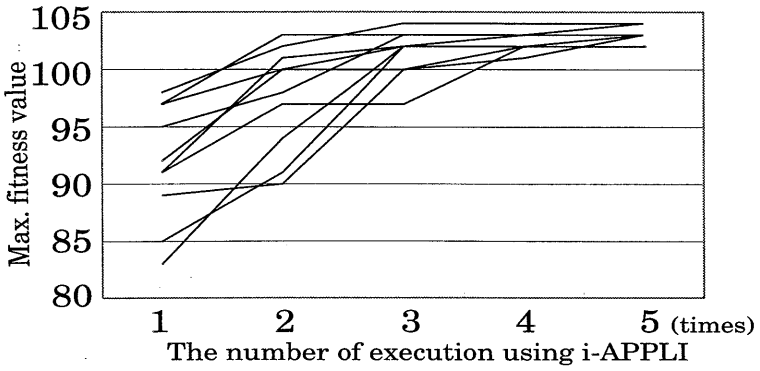

図 5.1: i アプリの連続実行結果（ $L = 16$ ）

母集団における最大適応度の推移を表す．

各回終了後，母集団に携帯端末から $J = 5$ 個の優良個体が移住する．このため，図 5.1 に表されるように，i アプリの実行回数にともなって，最大適応度が大きくなっていることがわかる．さらに，最良解である適応度 104 で遺伝子情報の異なる個体が生成される場合もあった．つまり，i アプリのために改良したアルゴリズムの手順 8 から 10 で実行される移住によって，より優良な解を生成し，最良解を得ることが可能である．

次に，より大規模な遺伝子長を対象とする GA の実行が可能であるかどうか検証するために，以下の実験をおこなった．実行にあたって，ナップサック問題における荷物の個数 L は 50 個とし，各荷物の価値 c_i と重量 w_i は 6 から 10 の自然数を与えた．また，ナップサック問題の耐荷重量は，$W = 300$ とした．GA で用いられるパラメータは，先におこなった実験と同じものを用いた．ただし，ペナルティパラメータは，遺伝子長と同じ $\alpha = 50$ とした．

図 5.2 は，i アプリの連続 10 回の繰り返しを 10 セットおこなった結果得られた母集団の最大適応度の推移を表す．図 5.2 に表されるように，$L = 16$ の場合同様，i アプリの実行回数の増加にともなって，最大適応度が大きくなっていることがわかる．また，i アプリの実行回数が 5 回目までは，実行回数にともなって，最大適応度が大きくなり，6 回目以降は，ほぼ横ばい状態ではあるものの，若干最大適応度が大きくなっている．これは，移住の効果が効いているものと思われる．

以上より，島モデルの GA を i アプリに実装する際，改良されたアルゴリズムは，遜

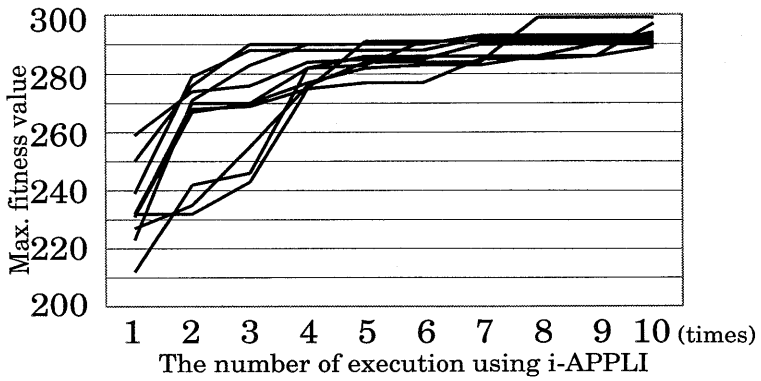

図 5.2: i アプリの連続実行結果（ $L = 50$ ）

色のない解を探索しうることが判明した．また，最終的な解を得るためには，何度も実行しなければならないが，母集団を大きく取ることが可能であるため，探索空間を広げることは容易である．つまり，現在のヒープ容量等の携帯端末性能を考慮すると，大規模探索をするためには，実行回数は多くなるが，改良されたアルゴリズムを用いることが有効であると考えられる．

第6章　まとめ

現在，携帯通信の市場には 2,100 万台以上の i アプリ対応の携帯端末が普及しており，この携帯端末は，ネットワーク機能と計算リソースを有した小型計算機とみなすことができる．第 III 部では，i アプリ対応の携帯端末の余剰計算リソースを利用した数値計算の可能性について述べた．その一例として，i アプリを用いて島モデルの遺伝的アルゴリズム（ GA ）を実装した．実装にあたって，Master-Slave 型に島モデルの GA を改良した．この改良されたアルゴリズムを検証するために，ナップサック問題を解いた．その結果，ナップサック問題のスケールに関して問題があるものの，一般的な島モデルの GA の実行と比較して，ほぼ遜色のない性能結果を示せた．

2003 年 6 月より発売が開始された *505i* シリーズでは，各機種とも，ヒープ容量，Scratch-Pad 容量ともに最大容量が 2 倍以上に増加されたため，より大規模な数値計算が可能になるものと思われる．

第IV部

関連研究

第1章　はじめに

様々な物理・自然現象の仕組みを解明するためには，いくつかの定義と仮定をおこない，それを実証しなければならない．初歩的な物理・自然現象は，実験によって実証することが可能である．しかし，近年，マクロな世界やミクロな世界を対象とする現象の解明が望まれるようになり，実験によって実証することが困難となってきた．さらに，複雑な前提条件を満たす環境を再現することは，非常に困難である．そのため，物理・自然現象を数式であらわす [39] ことによって，シミュレーションをおこない，その結果と現実との落差を少なくすることによって，現象解明をする方法がとられるようになった．

数式を用いた解明方法は，多くの場合，膨大な計算を必要とする．近年，計算機の発達により，その膨大な計算を数値解析としておこなうことが可能となった．しかし，計算機を用いた数値解析では，離散的な値しか扱うことができず，数式ではなく数値として扱わなければ成らない．そのため，数式を用いて実際に計算する場合と比べ，誤差が生じる可能性がある．そこで，この誤差を微小にするための数学的モデルの作成・数値計算が必要となる．

数値解析で用いる数学的モデルの作成・数値計算に関する諸定義を確定するためには，多くのパラメータを的確に指定しなければならない．パラメータチョイスのためには，シミュレーションを膨大な回数おこなう必要がある．しかし，このシミュレーションは，多くの場合，膨大な計算量を必要とするため，現実的な時間で多種多様なパラメータを用いて実行することは不可能である．

この問題を解決するために，複数計算機を用いた並列化が望まれている．

数値計算を並列実行するために，第II部，第III部において，並列プログラムの開発と携帯端末を用いた数値処理の可能性について説明した．第IV部では，それらに関連する研究について述べる．

以下，第2章において，いくつか数値解析について説明し，その並列化に関する研究に

ついて述べる．第3章において，第II部で対象とした乱流シミュレーションに対する並列化の関連研究についてまとめる．第4章において，携帯端末を用いて数値処理を行う際に利用する余剰計算リソースに関する研究について述べる．第5章において，性能の異なる計算機間での分散処理システムである Grid システム [29] についてまとめる．

第2章　数値解析に関する研究

数値解析をおこなうために，物理・自然現象に関する方程式をアルゴリズム化し，ライブラリとして利用することができる．同様に，数値解析の並列化ライブラリが存在する．

第2.1節では，*ScaLAPACK*（Scalable LAPACK）[6] [61] について説明する．第2.2節では，NAG 並列計算ライブラリ [42] について説明する．*SSL II/VPP*（Scientific Subroutine Library II for Vector Parallel Processors）について，第2.3節において説明する．第2.4節では，*PARCEL*（Parallel Computing Elements）[64] [77] について説明する．

2.1　ScaLAPACK

逐次計算機用のライブラリとして，数学ソフトウェア・スーパコンピュータの性能評価データなどの情報を提供する Netlib [44] で公開されている無償のライブラリとして *BLAS*（Basic Linear Algebra Subprograms）[8] がある．このライブラリでは，ベクトルと行列の積などの基本線形計算をおこなうためのルーチンが用意されている．

線形計算ライブラリ *LAPACK*（Linear Algebra PACKage）[34] は，*BLAS* を拡張したものである．この *LAPACK* には，約 300 種類のルーチンが用意されており，それらの主な機能は，連立一次方程式，線形最小二乗問題，固有値問題，特異値分解である．

並列化線形計算ライブラリ *ScaLAPACK* は，*LAPACK* を並列化したものであり，その主な機能は，連立一次方程式，逆行列，固有値問題，特異値分解である．この並列線形計算ライブラリは，分散メモリ環境を対象とする．そのため，行列・部分行列などの転送処理をおこなうためにライブラリ *BLACS*（Basic Linear Algebra Communication Subprograms）[7] とライブラリ *PBLAS*（Parallel BLAS）[53] が適用される．*BLACS* には，並列化ライブラリ *MPI*（Message Passing Interface）[40] を用いるものと，並

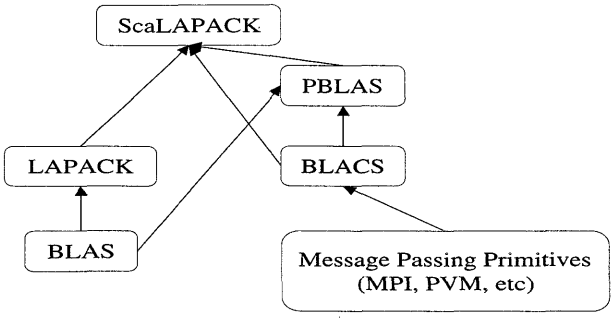

図 2.1: Netlib に公開されているライブラリ間の関係

列化ライブラリ *PVM*（Parallel Virtual Machine）[55] を用いるものが，それぞれ用意されている．*PBLAS* は，*BLAS* を並列化したもので，基本線形計算を並列に行うことができる．

以上のライブラリの関係を表すと，図 2.1 のようになる．

2.2 NAG 並列計算ライブラリ

イギリスに本社を持つ NAG（The Numerical Algorithms Group）は，高度な数学問題を解くための各種ソフトウェア開発をおこなっており，開発された数値計算・統計解析ライブラリは，世界的に高い評価を得ている．

NAG は，Fortran 言語を用いた並列計算ライブラリを 2 種類開発している．

そのうちの一つが，分散メモリ環境を対象とする NAG Parallel Library である．このライブラリは，並列化ライブラリ *MPI* を用いて開発されているため，Linux クラスタ・UNIX クラスタのみならず，共有メモリ型並列計算機やクラスタ，スーパコンピュータでの利用が可能である．また，並列化ライブラリ *MPI* に関する個所は全てルーチン内部で扱われるため，利用者が並列化を意識する必要はない．

もう一つの並列計算ライブラリとして，共有メモリ型並列計算機を対象とする並列化ライブラリ NAG Fortran SMP Library がある．このライブラリは，プログラムにリンクさせるだけで使用することができる．また，並列化には，Open MP が用いられている．

2.3 SSL II/VPP

SSL II/VPP は，逐次用に富士通株式会社 [14] が開発した数値計算サブルーチンライブラリ *SSL II* を基に，富士通株式会社とオーストラリア国立大学の数値計算グループが共同開発した並列化ライブラリである．

SSL II/VPP の主な計算機能は，行列積，連立一次方程式，固有値問題，逆行列，実・複素数のフーリエ変換，線形最小二乗解，一様乱数・正規乱数の生成である．連立一次方程式で扱うことが可能な行列として，実行列，複素行列，正定値対称行列，実バンド・スパース行列がある．また，固有値問題で扱うことができるものとして，3 重対角，実対称，Hermite，実対称スパースがある．

流体力学・分子化学・核融合などの現象をシミュレーションする際に利用される数値解析において，連立一次方程式・固有値計算・フーリエ変換などの計算は，膨大なコストを必要とする．そのため，*SSL II/VPP* を適用することによって，実行時間の短縮が期待され，格段の性能向上を得られる．

2.4 PARCEL

日本原子力研究所 [28] では，大規模な科学技術計算がおこなわれている．これら多くの数値計算において，連立一次方程式，固有値問題，フーリエ変換などが，全計算時間の大部分を占める．そのため，並列計算能力を最大限引き出すための並列数値計算ライブラリが必要となる．そこで，日本原子力研究所では，並列数値計算ライブラリとして，*PARCEL* を開発した．

PARCEL の主な計算機能は，連立一次方程式，固有値問題，2 次元・3 次元高速フーリエ変換，擬似一様乱数の生成である．これらのうち，連立一次方程式は，並列化線形計算ライブラリ *ScaLAPACK* にも存在するが，反復解法をおこなえるルーチンを含むものは *PARCEL* のみである．

また，*PARCEL* は，分散メモリ環境を対象に開発されているため，内部では，並列化ライブラリ *MPI* が用いられる．そのため，機種依存ライブラリではないため，多種多様な計算機を用いた並列化が可能である．

第3章　流体に関する研究

並列プログラムの開発として，第II部において，自由表面乱流場における熱物質輸送の直接数値シミュレーション（ *DNS* ： Direct Numerical Simulation ）を対象とした．*DNS* では，Navier-Stokes 方程式 [39] が用いられる．Navier-Stokes 方程式を数値解析するために，微分方程式を差分化し，有限差分方程式（ *FDM* ： Finite Difference Method ）の形で解く．この際，用いられる解法として，陽解法，陰解法がある．また，微分方程式に関する部分を周波数空間で扱うことが可能な擬似スペクトル法（ *PSM*： Pseudo Spectral Method ）を用いて，Navier-Stokes 方程式を数値解析する方法もある．

以下，第3.1節において，陽解法の並列化について述べる．第3.2節において，陰解法の関連研究を挙げる．第3.3節において，*PSM* を用いた乱流場の *DNS* の並列化に関する研究について述べる．

3.1　陽解法の並列化

陽解法を用いた *DNS* 計算の並列化としては，姫路工業大学の坂上らの 3 次元流体コード *IMPACT-3D* に関する研究 [57]，中央大学の樫山らの浅水長波流れ解析を対象とする研究 [58]，東京大学の矢川らの圧縮性流体に関する研究 [13] 等がある．

坂上らは，並列プログラミング形式の一つである HPF （ High Performance Fortran）を用いて実装している．この HPF は，並列プログラムを開発する際に必要な専門知識を極力削除して実装できるよう開発されているため，多くの自然科学者に利用されている．坂上らは，プラズマシミュレーションコード *IMPACT-3D* の並列化に関する研究をおこなっている．並列プログラムにおいて，問題空間である 3 次元シミュレーション空間は，小さな部分空間に分割され，各プロセッサに割り当てられる．この際，問題空間の分割方法の違いによって，並列実行効率が異なる．また，ネットワークの構造と通信速度の関係

も，並列実行効率に大きな影響を与える．そこで，坂上らは，問題空間の分割方法の違いによる並列実行効率の差について調査し，問題空間の適切な分割サイズと各プロセッサへの割り当てを提案している．その有効性を検証するために，地球シミュレータの Node を 512 個用いて実行し，14.9T FLOPS （ピーク性能比： 45% ）という成果を得ている．この成果によって，2002 年 SC において，Gordon Bell 賞を受賞した．

樫山らの研究では，大規模な大気汚染や水環境における流体解析のための並列化手法の構築を目的としている．この目的のために，離散化手法として，複雑形状を有する領域の流れを解析可能な有限要素法（非構造格子）を用い，並列化手法として，領域分割法を基盤としている．樫山らは，台風通過による高潮の状況をシミュレーションするための並列プログラムを開発し，512 個のプロセッサを用いて実験をおこなっている．実験の結果，演算速度倍率が理想倍率と等しくなった場合の並列化効率を 100% とすると，有限要素法によって格子化された問題空間の格子点数が 76,497 の場合の並列化効率は 47% ，格子点数が 133,546 の場合は 66% ，格子点数 206,977 の場合は 76% を得ている．また，樫山らは，Arbitrary Lagrangian-Eulerian 表記による Navier-Stokes 方程式を基本方程式として用いた界面追跡法 [46] [50] のための並列化手法を提案している [59] [68] .

矢川らは，動的陽解法を用いて，衝突問題など比較的短時間の瞬間的な現象の解析を研究対象としている．この解法は，時間幅を小さく取らなければならないため，計算時間が長くなるという問題を持っているが，高いデータ並列性も持ち合わせているため，並列処理に適した解法である．並列化に際し，問題空間は部分領域単位で扱われるため，大規模な並列解析が可能である．そのため，3 次元解析が実用化の段階である [11] ．また，矢川らは，領域分割をおこなう際，一旦大きな部分領域に分割し，その領域をさらに小さく分割する方法をとっている [41] [75] [76] ．このことによって，動的陽解法の並列実行を階層的に実装することが可能となる [48] [49] .

3.2 陰解法の並列化

陰解法を用いた *DNS* 計算の並列化としては，東京大学の荒川らの 3 次元非定常非圧縮性流体 LES （ Large Eddy Simulation ）解析を対象とする研究 [1] と，京都大学の牛島らの *QSI* （ Quintic Spline Interpolation ）スキームを対称とする研究 [72] がある．

荒川らの研究目的の一つとして，ヴァナキュラー風車の開発がある [2] ．ヴァナキュラー風車とは，地域密着型の風力発電用風車のことであり，風力発電能力を最大限得ることができる風車の形状を解析することは，重要である．また，エネルギー経済システムのシミュレーションを一定の仮定を基におこなった結果，2050 年には国内で $1M$ W の大型風車が 10 万台も普及する可能性がある事を示唆した．仮に 10 万台も普及するならば，一律の規格の風車のみでは，景観を損なう可能性がある．ゆえに，ヴァナキュラー風車の開発は，必要不可欠である．ヴァナキュラー風車を開発する際，その風車の周りの流体を解析しなければならない．そのために，荒川らは，問題空間の時系列の幅を大きくとることが可能な陰解法を適用し，その並列化に力を注いでいる．また，粒度の小さい並列計算機でも高い効率を維持できるような流体プログラムの開発をおこないながら，その解析を専門におこなうための超並列シミュレータの開発を手がけている．

牛島らの研究では，3 次元一般座標系で表示された非圧縮性流体の基本式をコロケート格子上で有限差分法により離散化する手法に基づく乱流場を対象としている．問題空間はコロケート格子として扱われるが，このコロケート格子は，圧力勾配を格子境界で評価される必要がある [73] ．非圧縮性流体の解法において，内部流動の精度を向上させるためには，基本方程式中の移流に関する項の取り扱いが重要となる．そこで，高精度のスキームを必要とする *DNS* を視野に入れた開発をおこなうために，予測段階に 5 次精度相当の *QSI* スキームを用いた陰解法を用いる [71] ．牛島らは，*QSI* スキームを用いた陰解法で必要な連立一次方程式の計算を並列化するにあたり，Bi-CGSTAB 法を利用している．

3.3 擬似スペクトル法の並列化

PSM を用いた乱流場の *DNS* 計算の並列化に関する研究は，本研究 [67] 以外では，東京大学の古村らの研究 [16] [17] があるだけである．しかし，逐次実行において，*PSM* を用いた乱流場の *DNS* では，陽解法や陰解法と比べ，精度の高い結果を短時間で得ることが可能であるため，並列化による高速化が可能となれば，よりよい乱流解析が可能となるものと考えられる．

古村らは，不均質な媒質中を伝播する地震波動伝播の特徴と並列シミュレーションによる強震動の生成メカニズムを解明するための研究をおこなっている．地震波動計算に

PSM を適用することによって，*FDM* を適用する場合に比べ，計算に必要なメモリと時間を大幅に減少させることが可能となった．しかし，並列化に際し，本研究 [67] と同様，分割領域をまたがる フーリエ変換のために大規模なプロセッサ間通信が必要となる．この問題を解決するために，*PSM* と *FDM* を結合することにより，両者の特色を活かしたハイブリッド型並列計算法を開発している．

第4章　余剰計算リソースの利用に関する研究

携帯端末の余剰計算リソースを用いた数値処理に関する研究は，新規分野であるため，関連研究を見出すことができない．しかし，一般的な計算機において，余剰計算リソースを利用した分散処理はすで存在する．

以下，第4.1節において，SETI@home [63] について説明する．第4.2節において，Folding@home [12] について述べる．

4.1　SETI@home

一般的な計算機の余剰計算リソースを利用した分散処理として，電波望遠鏡で得られたデータをスクリーンセーバのバックグラウンド処理で解析する SETI@home がある．

SETI （ the Search for Extra-Terrestrial Intelligence ）とは，宇宙に存在する可能性のある生命を見出すためのプロジェクトであり，宇宙に飛び交う電波を解析することによって，生命体の存在可能性を調査することである．これは，仮に生命体が存在するならば，地球人同様，通信をおこなうために電波を使うという考えに基づいている．電波を使う場合，電波の強さにもよるが，いくらか漏れてしまう．SETI プロジェクトでは，この漏れ出した電波をみつけることによって，宇宙生命の可能性を見出すことを試みている．

微弱な電波を受信するために，巨大な電波望遠鏡の利用が必要である．そのため，アメリカ合衆国のバークレー大学の SETI 探査で使われているアレシボ電波望遠鏡が用いられている．アレシボ電波望遠鏡のパラボラアンテナの皿部分は固定されているため，旋回できない．しかし，受信アンテナは弓型の線路にすえつけられているため，天頂から 20 度までの範囲であれば観測することができる．

SETIの調査では，膨大な量のデータを解析する必要があり，探索空間も，宇宙という非常に広大な範囲を対象としている．そのためには，大規模な高速演算ユニットを搭載した大型計算機が必要となるが，SETI@homeのスポンサーである惑星協会（The Planetary Society）は，大型計算機を維持管理する能力がない．そこで，対象空間を細分化したくさんの小さな計算機を用いて解析することを考え出した．

SETI@homeでは，各周波数における電波音を正確にとらえ，その解析を各ユーザが所有する計算機のバックグラウンドでおこなう．この際，1台の計算機が解析する電波音の周波数の幅は0.07 Hzとなる．周波数幅を小さくすることによって，より正確に電波音をとらえることが可能となる．また，現在，国際的な協定で使用禁止とされている$1420M$ Hzから$1427M$ Hzの間の周波数に関する情報は，宇宙の未知なる生物が発した可能性が高いため，重要である．

4.2 Folding@home

SETI@home以外に，余剰計算リソースを使ったプロジェクトとして，タンパク質の構造解析に用いられるFolding@homeがある．この目標は，タンパク質の折り畳みをシミュレートすることである．

タンパク質は，アミノ酸のネックレスのようなものである．そのネックレスのつながり方によって，酵素や抗体としての機能を持つ．酵素とは，生物を動かす生化学反応の原動力であり，抗体とは，侵入物質を見つけ，免疫システムに望ましくない場合は排除する役割をなす．タンパク質は，アミノ酸のつながり方によって，全く異なる性質を持つ．そのため，様々なつながり方をしたタンパク質とその性質を解明することが，生命の仕組みを解明することを意味する．

実在のタンパク質は，酵素あるいは抗体として機能するために，折り畳み（fold）と呼ばれる方法を使って形状を瞬時に形成する．この形成方法を解析することが可能となれば，様々な病気に対する抗体や健康のために必要な酵素を作ることが可能となり，医療機関に大きな影響をもたらす．

その具体例として，アルツハイマー病，嚢胞性線維症，CJD（狂牛病），遺伝型肺気腫などの病気や多くの癌などの治療があげられる．これらの病気は，タンパク質が誤って折

り畳まれた結果，組織が凝集することによって生じる．そこで，誤って折り畳まれているタンパク質を一度広げ，再度正しく折り畳むことができるならば，この病気を治療することが可能であると考えられている．

現在，Folding@home を用いて分散処理させることによって，5 から 10 μ 秒の時間でタンパク質を折り畳むことが可能なシミュレーション法がある．このシミュレーションによって得られた解析結果は，Folding@home の運営管理をしているスタンフォード大学の化学学科に所属する Pande Group が保持する．ただし，非営利団体であるため，科学的研究と教育が目的であるならば，誰でも利用可能である．

第5章　Grid システムに関する研究

分散処理システムとして，現在，Grid システム [29] が着目されている．Grid システムは，広範囲に存在する様々な計算機資源を結びつけて，様々なサービスの構築やアプリケーションの実行を可能とするものである．特に，ハイパフォーマンスコンピューティングの分野では，複数の計算機資源を統合的に利用することによる処理の高速化に関して注目度が高い．

以下，第5.1節において，Grid システムについて説明する．第5.2節において，Grid システムの将来像について述べる．

5.1　Grid システムについて

Grid の語源は，電気を伝える高圧送電網（パワーグリッド）に由来する．この意味は，コンセントにさすだけで必要な電力がえられたように，情報コンセントに計算機を接続するだけで，ネットワーク接続された様々な計算機を利用して，シミュレーション，数値解析，データベースの分析，蓄積された情報の参照等を享受することが可能となることの期待を表す．

アメリカ合衆国にあるアルゴンヌ国立研究所兼シカゴ大学に所属する Ian Foster 氏は，Grid システムとして機能しているかどうかを判定するために以下のチェック項目を作成した．

一つ目は，集中管理されていない分散した資源を，必要に応じてコーディネート可能であるかどうかという点である．従来の並列化システムは，計算機の選択・設定・管理の全てを集中的におこなっている．そのため，計算負荷が密になる時間と疎になる時間が存在する．Grid システムでは，エージェントシステム等を利用することによって，計算負荷が最も疎な計算機に仕事を割り当てることによって，処理時間の短縮をおこなう．この

エージェントシステムとしては，独立行政法人産業技術総合研究所 [10] において開発された Ninf [45] が有名である．

二つ目は，オープンスタンダードなプロトコルやインターフェースの利用である．多くの人が Grid システムを利用するためには，WWW 同様，無償で使えることが望ましい．そのため，異なる計算機間で利用可能なプロトコルやインターフェースを公開することは重要である．プロトコルのひとつとして，日本原子力研究所 [28] で開発された異機種並列計算機間通信ライブラリ Stampi [24] [25] がある．Stampi は，直接外部と通信できないプロセッサからの通信を外部の通信可能なプロセッサに代行させることによっておこなわれる．この際用いられる通信ライブラリは，並列化ライブラリ *MPI* を基盤とする．

三つ目は，単純には得られない質の高いサービスの提供である．もし，単純な問題や並列計算機で十分解決できる問題を対象とするのであれば，Grid システムを利用する必要はない．

Grid システムには，高信頼ウェブサービスのためのビジネスグリッド，超大規模データ処理のためのデータグリッド，高速計算サービスのためのコンピューティンググリッド，余剰計算リソースを利用するための PC グリッドがある．これらの Grid システムの境界は曖昧であるため，実現しようとするサービスに応じて，複数の Grid システムを組み合わせる必要がある．

5.2 Grid システムの将来

Grid システムを用いて，分散化させる場合，計算機台数が多ければ多いほど，能力を上昇させることが可能である．将来的には，携帯端末のみならず，PDA や家電製品においても，J2ME 規格 [65] の Java 仮想計算機が組み込まれることが想定される．仮に，Grid システムの一要素として身の回りの J2ME 規格の Java 仮想計算機を用いることが可能となれば，大規模な Grid システムを構築することができる．この際，既存の研究 [22] を応用することによって，膨大な計算能力を得ることが可能となる．

また，携帯端末，PDA，家電製品に組み込まれている Java 仮想計算機の主な用途は，エンターテイメント性の強いものや社会的に実用性の高いものである．そのため，常時実行していたとしても，Java 仮想計算機の能力を最大限利用する状況が継続するわけで

はない．ゆえに，余剰計算リソースが発生し，Grid システムとして用いることが可能である．

本研究でプロトタイプとして実装した遺伝的アルゴリズムは，すでに一般的な計算機を用いた Grid システムにおいて検証済みである [70]．ただし，Master-Slave 型の島モデルの遺伝的アルゴリズム [66] は，Grid システムとして検証されていない．

また，Grid を用いた分散処理として，陽解法を用いた乱流場の *DNS* 計算の分散処理をおこなう i アプリの開発は，すでに着手されている [27] ．

第Ⅴ部

むすび

大気流の解析や天気予測等の地球規模の物理現象や自然現象などの数値解析を実行する場合，大規模な数値計算を伴う．計算機の性能が飛躍的に向上し，逐次プログラム用の最適化コンパイラが一般的となった今日でさえ，地球シミュレーションには非常に長い計算時間が必要となる．この問題を解決するために，最も現実的な方法として，Grid システム [29] のような並列計算機を仕様する事が考えられる．そのために，本研究では，既存の逐次プログラムの並列化と，新たな Grid システムとして着目される携帯端末での数値計算について検証した．

本研究で用いた既存の逐次プログラムは，擬似スペクトル法（ *PSM* ： Pseudo Spectral Method ）を用いた乱流場の直接数値シミュレーション（ *DNS* ： Direct Numerical Simulation ） [79] である．通常の *DNS* の場合，物理現象の空間的局所性に基づいた分割手法である strip mining [43] [80] を用いて，適用される空間を等分割することによって，並列化される．今回対象とした *PSM* を用いた乱流場の *DNS* 計算では，数値不安定性を回避し高精度な解を得るために，時間発展を周波数空間上で扱い，非線形演算を実空間で扱っている．データを実空間と周波数空間とでやり取りするためには，離散フーリエ変換が必要となる．この際，strip mining を適用すると，全プロセッサ間で，分散された全データをやり取りする必要が生じ，通信コストがと通信オーバーヘッドが膨大になる可能性がある．そこで，*PSM* を用いた乱流場の *DNS* 計算を並列化するために，1 つの配列データを少数のプロセッサに割り当てる並列化手法 B を提案し，第 II 部において有効性を示した．この並列化手法 B は，データを分割した際に考慮しなければならない通信オーバーヘッドを最小にするという基準で考えた時に導かれるものである．

新たな Grid システムとして着目される携帯端末での数値計算の可能性を検証するために，NTT Docomo [47] が提供する i アプリ [20] を対象に研究をおこなった．現在，i アプリ搭載の携帯端末は 2,100 万台以上市場に普及している．しかし，常時 i アプリを実行している端末は極一部であるため，多くの余剰計算リソースが存在する．そこで，i アプリを用いて数値計算を実行するために，この余剰計算リソースを利用して分散処理を行うことを考えた．そのためのプロトタイプとして，遺伝的アルゴリズム（ GA ： Genetic Algorithm ） [18] を実装した．実装にあたり，i アプリを用いて分散処理するために，島モデルと呼ばれる GA [32] を Master-Slave 型に改良した．トイプロブレムであるナップサック問題を用いて，改良した島モデルの GA の性能を検証し，十分な結果を得た．

謝辞

本研究を進めるにあたり，終始温かい御指導を賜りました奈良女子大学大学院人間文化研究科複合領域科学専攻複合情報科学講座の城和貴教授に深く感謝します．

和歌山大学工学部情報通信システム学科國枝義敏教授には，本研究全般に渡って御支援ならびに御指導を賜りましたことをここに述べ，感謝の意を表します．

また，本論文をまとめるにあたり，京都大学大学院工学研究科の功刀資彰助教授には，貴重なる御助言と御指導を頂き，また，本研究の議論に貴重な時間を割いていただきましたことを深く感謝いたします．

本研究に関して的確な御助言を頂いた京都大学大学院工学研究科の諸氏ならびに修了生の山本義暢氏に感謝いたします．

山口大学工学部の庄野逸助教授には，本論文をまとめるにあたり，的確な御助言を頂きましたことを心より感謝いたします．

奈良女子大学理学部情報科学科数理情報学講座の落合豊行教授，自然情報学講座の林田佐智子教授，ならびに奈良女子大学大学院人間文化研究科複合領域科学専攻の高橋智助教授には，本論文の副査をして頂き，貴重な御意見を賜りましたことを感謝いたします．

奈良女子大学大学院専任助手の渡辺知恵美先生には，著書の良き相談相手として助けて頂きましたことを感謝します．

最後に，今日まで筆者の研究活動に対する理解と協力をいただいた複合情報科学講座の諸氏，家族，そして友人に感謝します．

参考文献

[1] Arakawa, C., Okano, M., and Itahara, H.: *Numerical Analysis of Supersonic Mixing Layer Using LES and Its Application, ASME/JSME Fluid Engineering Conference,* FEDSM99-7315, pp.1–8 (1999).

[2] 荒川忠一，有賀誠一，飯田誠：ヴァナキュラー風車の提案，**環境芸術学会論文集**，1-1，pp.13–19 (2001).

[3] アスキー書籍編集部：*i* モード *Java* プログラミング，*ASCII,* (2001).

[4] *AU* 公式サイト，*http://www.au.kddi.com/*

[5] Baker, J. E.: *Adaptive Selection Methods for Genetic Algorithms, Proc. 1st Int. Joint Conf. on Genetic Algorithms (ICGA85),* (1985).

[6] Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., and Walker, D.: *ScaLAPACK User's Guide, SIAM,* (1997).

[7] *Basic Linear Algebra Communication Subprograms,* *http://www.netlib.org/blacs/*

[8] *Basic Linear Algebra Subprograms, http://www.netlib.org/blas/*

[9] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T.A.: *Spectral Methods in Fluid Dynamics, Springer Verlag*(1988).

[10] 独立行政法人産業技術総合研究所グリッド研究センター，*http://www.gtrc.aist.go.jp/jp/*

[11] *Free Mesh Method 3 D*, *http://dolphin.eng.ynu.ac.jp/fmm/*

[12] *Folding@home* スタンフォード大学, *http://folding.stanford.edu/*

[13] 藤澤智光, 矢川元基：圧縮性流体に関する並列フリーメッシュ法（節点処理有限要素法）, **理論応用力学講演会講演論文集**, Vol.50, pp.151–152 (2001).

[14] 富士通株式会社, *http://jp.fujitsu.com/*

[15] 布留川英一：*JBuilder Mobileset* 携帯 *Java* プリケーション開発ガイド（ *au* , *J-PHONE* 対応）**毎日コミュニケーションズ**

[16] 古村孝志, 纐纈一起, 竹中博士：*PSM/FDM* ハイブリッド型並列計算による地震波動の *3* 次元数値シミュレーション *(P002)*, **日本地震学会講演予稿集秋季大会（ポスターセッション）**, pp.132 (1999).

[17] 古村孝志, 纐纈一起, 竹中博士：大規模 *3* 次元地震波動場（音響場）モデリングのための *PSM/FDM* ハイブリッド型並列計算, **物理探査**, 第 53 巻, 第 4 号, pp.294–308 (2000).

[18] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley* (1989).

[19] Handler, R. A., Swean Jr., T. F., Leighton, R. I., Swearingen, J. D.: *Length scales and the energy balance for turbulence near a free surface, AIAA J.31*, pp.1998-2007 (1993).

[20] *i* アプリコンテンツの作成について, *http://www.nttdocomo.co.jp/p_s/imode/java/*

[21] 伊庭斉志：遺伝的アルゴリズムの基礎 - *GA* の謎を解く -, **オーム社**, (2002).

[22] *Proceedings of the Third IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE* (2001).

[23] *i* モード契約数,
http://www.nttdocomo.co.jp/corporate/report/contract/index.html

[24] 今村俊幸，小出洋，武宮博：異機種並列計算機間通信ライブラリ： *Stampi* - 利用手引書 第二版，*JAERI-Data/Code*，2000-002 (2000).

[25] Imamura, T., Tsujita, Y., Koide, H., and Takemiya, H.: *An Architecture of Stampi : MPI Library on a Cluster of Parallel Computers, Proceedings of EuroPVM/MPI 2000*, LNCS 1908, pp.200–207 (2000).

[26] *i* モード製品サービス，*http://www.nttdocomo.co.jp/p_s/imode/*

[27] 石内寿子，程暁紅，高田雅美，渡辺知恵美，城和貴：*NCC@i-APPLI*：*i* アプリによる数値計算コンポーネントの構想，情報処理学会 *MPS* シンポジウム，Vol.2003, No.14，pp.45–52 (2003).

[28] *JAERI* 日本原子力研究所，*http://www.jaeri.go.jp*

[29] Kesselman, F.: *The GRID Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, Inc.*, (1999).

[30] *KDDI* 公式サイト，*http://www.kddi.com/corporate/index.html*

[31] Kitano, H.: *Designing neural networks using genetic algorithms with graph generation system, Complex Systems 4*, pp.461–476 (1990).

[32] 北野宏明：遺伝的アルゴリズム *2*，産業図書 (1997).

[33] Komori, S., Ueda, H., Ogino, F., Mizushina, T.: *Turbulence Structure and Transport Mechanism at the Free Surface in an Open Channel Flow, Int. J. Heat and Mass Transfer*, Vol.25, No.4, pp.513–521 (1982).

[34] *Linear Algebra PACKage, http://www.netlib.org/lapack*

[35] Meyer, T. P., Packard, N. H.: *Local forecasting of high-dimensional chaotic dynamics, Nonlinear Modeling and Forecasting, Addison-Wesley* (1992).

[36] *MIPS* 値計算用 ベンチマーク，*http://alice.ics.nara-wu.ac.jp/ takata/i*

[37] Mitchell, M. （監訳：伊庭斉志）：情報科学セミナー　遺伝的アルゴリズムの方法，**電気大出版局** (1997).

[38] Montana, D. J., L. D. Davis: *Training feed forward networks using genetic algorithms, In Proceeding of the International Joint Conference on Artificial Intelligence*, Morgan Kaufmann (1989)

[39] 森平勇三：別冊・数理科学　方程式と自然，**サイエンス社.** (1993).

[40] *Message Passing Interface, http://www-unix.mcs.anl.gov/mpi/index.html*

[41] 宮村倫司，野口裕久，塩谷隆二，吉村忍，矢川元基：階層型領域分割法による超並列弾塑性有限要素解析，**日本機械学会論文集**，A 編，65-534，pp.1201–1208 (1999).

[42] *NAG* 並列計算ライブラリ，*http://www.nag-j.co.jp/Library/paraLib.htm*

[43] 中田育男: コンパイラの構成と最適化，**朝倉書店**，(1999).

[44] *Netlib Repository at UTK and ORNL, http://www.netlib.org*

[45] *Ninf: A Global Computing Infrastructure,*
http://ninf.apgrid.org/papers/papers.j.shtml

[46] Nomura, N.: *ALE finite element computations of fluid structure interaction problems, Computer Methods in Applied Mechanics and Engineering*, 112, pp.291–308 (1994).

[47] *NTT Docomo* 公式サイト，*http://www.nttdocomo.co.jp/*

[48] 大石篤哉，山田勝稔，吉村忍，矢川元基：領域分割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，pp.1445–1452 (1992).

[49] 大石篤哉，山田勝稔，吉村忍，矢川元基：*EWS* ネットワークによる動的問題の並列有限要素解析，**日本機械学会論文集**，A 編，62-593，pp.253–260 (1996).

[50] Okamoto, T., and Kawahara, M.: *Two-dimensional sloshing analysis by the Arbitrary Lagrangian-Eulerian finite element methods*, *Proceedings of JSCE*, 441, I-18, pp.39–48 (1992).

[51] Orszag, S.A.: *a Numerical simulation of in compressible flows within simple boundaries*, Galerkin (Spectral) representations, Stud. Appl. Math. 50, pp.293–327 (1971).

[52] *Palm* 公式サイト，*http://www.palm-japan.com/home.html*

[53] *Parallel BLAS*, *http://www.netlib.org/pblas*

[54] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T.: *Numerical Recipes in C*, *Cambridge University Press* (1988).

[55] *Parallel Virtual Machine*, *http://www.csm.ornl.gov/pvm/pvm_home.html*

[56] ランダウ・リフシッツ: 流体力学（*1*），**東京図書**（1990）.

[57] 坂上仁志，高橋豊：*3* 次元流体コードの領域分割法における並列効率，**電子情報通信学会論文誌**，Vol.80-D-I，pp.683–691 (1997).

[58] 桜庭雅明，樫山和男，菅野諭：移動境界を考慮した *Space-Time* 有限要素法による浅水長波流れ解析**応用力学論文集**，Vol.3，pp.255–262 (1999).

[59] 桜庭雅明，田中聖三，玉城宏幸，樫山和男：大規模自由表面流れ解析のための *ALE* 並列有限要素法，**応用力学論文集**，2，pp.233–240 (1999).

[60] 三宮信夫，喜多一，玉置久，岩本貴司：遺伝アルゴリズムと最適化，**朝倉書店** (1998).

[61] *Scalable LAPACK*, *http://www.netlib.org/scalapack*

[62] Schulze-Kremer, S.: *Genetic algorithms for protein tertiary structure prediction*, *Parallel Problem Solving from Nature 2. North-Holland* (1992).

[63] *The Search for Extraterrestrial Intelligence*, *http://www.planetary.or.jp/setiathome/home_japanese.html*

[64] 清水大志，岸田則生，市原潔，鈴木惣一郎，佐藤滋，田中靖久，蕪木英雄：原研における並列数値計算ライブラリの開発，**第 *2* 回計算工学会講演論文集**，2，105 (1997).

[65] *Sun J2ME* 規格，*http://java.sun.com/j2me*

[66] 髙田雅美，柴山智子，渡辺知恵美，庄野逸，城和貴：*i* アプリを用いた分散処理の可能性，**情報処理学会論文誌数理モデル化と応用**，採録予定.

[67] 髙田雅美，山本義暢，庄野逸，功刀資彰，城和貴：擬似スペクトル法を用いた乱流場の直接数値シミュレーションの並列化と性能評価，**情報処理学会論文誌コンピューティングシステム**，Vol.44，No.SIG6(ACS1)，pp45–54 (2003).

[68] 田中聖三，桜庭雅明，樫山和男：*ALE* 並列有限要素法による自由表面流れの非線形解析，**第 *14* 回数値流体力学シンポジウム**，*http://wwwsoc.nii.ac.jp/jscfd/cfds14/pdf/c07-3.pdf*，C07-3 (200).

[69] Tanese, R.: *Distributed Genetic Algorithms, Proc. 3rd Intl. Conf. on Genetic Algorithms* , pp434–439, 1989.

[70] Tanimura, Y., Hiroyasu, T., Miki, M., and Aoi, K.: *The System for Evolutionary Computing on the Computational Grid, Proc. of the 14th IASTED International Conference PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS (PDCS 2002)*, pp.39–44 (2002).

[71] 牛島省，禰津家久，山上路生，坂根由季子：コロケート格子を利用した自由液面を有する流動場の解析手法，**海岸工学論文集**，第 48 巻，pp.21–25 (2001).

[72] 牛島省，禰津家久，奥山洋平：領域分割法を用いた *QSI* スキームによる移流拡散方程式の並列計算法，**水工学論文集**，第 46 巻，pp.415–420 (2002).

[73] 牛島省，禰津家久：陰的 *SMAC* 法に基づく自由水面流れの高次精度・高速解法，**海岸工学論文集**，第 49 巻，pp.36–40 (2002).

[74] ボーダフォン 公式サイト，*http://www.vodafone.jp/scripts/japanese/top.jsp*

[75] Yagawa, G., and Shioya, R.: *Parallel finite elements on a massively parallel computer with domain decomposition*, *Computing Systems in Engineering*, 4, pp.495–503 (1993).

[76] 矢川元基，塩谷隆二：超並列有限要素解析，**朝倉書店**，(1989).

[77] Yamada, S., Shimizu, F., Kobayashi, K., Kaburaki, H., and Kishida, N.: *Performance Analysis of Parallel Mathematical Subroutine Library PARCEL*, *Proceedings of SNA2000*, CD-ROM Proceeding (2000).

[78] Yamamoto, Y., Kunugi, T., and Serizawa, A.: *Turbulence Statistic and Scalar Transport in an Open-Channel Flow*, *Advances in Turbulence VIII edited by C. Dopazo*, pp.231–234 (2000).

[79] 山本義暢：熱輸送を伴う自由表面混相乱流場の直接数値シミュレーション，**京都大学博士論文**，(2002).

[80] Zima, H., and Chapman, B.: *Supercompilers for Parallel and Vector Computers*, *Addison-Wesley Publishing* (1991).

付 録A　MIPS 値を計るための i アプリ用プログラム

MIPS 値を測定するにあたり，[36] を作成した．作成にあたり，[3] のスタンドアロン・アプリケーションで紹介されたユーティリティ iBench を利用することとした．利用クラスは，i アプリ起動時の設定に関するクラス（ iBench.java ）と時間測定に関するクラス（ TimeSpan.java ）である．

終了画面には，1,000,000 ループ内において，足し算を 1 回実行した場合の実行時間を 1 行目に，足し算を 2 回実行した場合の実行時間を 2 行目に表示する．この際，実行時間の単位は，ミリ秒とする．また，3 行目には，2 行目の実行時間から 1 行目の実行時間を引いた値を出力する．この実行は以下のような MIPS 用のクラスを作成することによって行う．

```
// MIPS 用プログラム
import com.nttdocomo.ui.*;

public class MIPS extends Canvas{
 boolean endFlag = false ;
 TimeSpan spn1,spn2;
 int width, height;

 MIPS(){
   spn1 = new TimeSpan();
   spn2 = new TimeSpan();
```

```
    width = Display.getWidth();
    height = Display.getHeight();
  }

  public void paint(Graphics g){
    long time,time1, time2;
    String str;
    if(endFlag == false){
      g.lock();
      g.clearRect(0,0,width,height);
      g.drawString("MIPS Start", 0, 20);
      g.unlock(true);
// ループ内の足し算 1 回の測定
      spn1.start(); // 測定開始
      mips_loop1(1000000); // 実行
      spn1.stop(); // 測定終了
// ループ内の足し算 2 回の測定
      spn2.start();
      mips_loop2(1000000);
      spn2.stop();
      endFlag = true;
      repaint();
    } else { // 結果表示
      g.lock();
      g.clearRect(0, 0, width, height);
      g.drawString("MIPS End", 5,20);
// 実行時間の取得
      time1 = spn1.getResult();
      time2 = spn2.getResult();
      time = time2 - time1;
```

```
      str = time+"msec";
      g.drawString(str,5,80);
      g.drawString(spn1.toString(),5,40);
      g.drawString(spn2.toString(),5,60);
      g.unlock(true);
      endFlag = false;
    }
  }

// ループ内の足し算 1 回の測定
  public void mips_loop1(int size){
    int i, itmp;
    for(i = 0; i <= size; i++) itmp = 1+1;
  }

// ループ内の足し算 2 回の測定
  public void mips_loop2(int size){
    int i, itmp;
    for(i = 0; i <= size; i++){
      itmp = 1+1;
      itmp = 1+1;
    }
  }
}
```

付 録B　研究業績

B.1　本論文に関連する研究業績

論文

高田雅美，柴山智子，渡辺知恵美，庄野逸，城和貴：i アプリを用いた分散処理の可能性，情報処理学会論文誌数理モデル化と応用，採録決定.

高田雅美，山本義暢，庄野逸，功刀資彰，城和貴：擬似スペクトル法を用いた乱流場の直接数値シミュレーションの並列化と性能評価，情報処理学会論文誌コンピューティングシステム，Vol.44，No.SIG6(ACS1)，pp45–54（2003 年 5 月）.

国内シンポジウム（査読あり）

石内寿子，程暁紅，高田雅美，渡辺知恵美，城和貴：*NCC@i-APPLI*：i アプリによる数値計算コンポーネントの構想，情報処理学会 *MPS* シンポジウム，Vol.2003，No.14，pp.45–52（2003 年 10 月）.

柴山智子，高田雅美，庄野逸，城和貴：i アプリを用いた分散 *GA* の実装，情報処理学会 *MPS* シンポジウム，Vol.2003，No.2，pp.251–258（2003 年 1 月）.

国際会議（査読あり）

Masami Takata, Yoshinobu Yamamoto, Hayaru Shouno, Tomoaki Kunugi and Kazuki Joe: *Effective Parallelization of a Turbulent Flow Simulation*, *The Proceedings of the* 5^{th} *International Conference and Exhibition on High-Performance Computing in Asia-Pacific Region*, CD-ROM Proceeding (Sep. 2001).

研究会（査読なし）

高田雅美，山本義暢，功刀資彰，城和貴：乱流シミュレーションの並列化と評価，**第** *133* **回情報処理学会計算機アーキテクチャ研究会**，2001-ARC-141，Vol.2001, No. 10，pp.1–6 (2001年1月)．

B.2 その他の研究業績

国内シンポジウム（査読あり）

高田雅美，福田京子，庄野逸，城和貴：*GA* を用いたレイリー波分散曲線に基づく水平成層構造の推定，情報処理学会 *MPS* シンポジウム，Vol.2003，No.2，pp.229–231（2003年1月）．

国際会議（査読あり）

Masami Takata, Hayaru Shouno and Kazuki Joe: *An Improvement of Program Partitioning Based Genetic Algorithm, the proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '02)*, Vol.I, pp.215–221 (Jun. 2002).

Mayumi Oto, Masami Takata, Hiroaki Yoshida and Kazuki Joe: *A Quantum Algorithm for Searching Web Communities, the proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '02)*, Vol.I, pp.260–266 (Jun. 2002).

Masami Takata, Yoshitoshi Kunieda and Kazuki Joe: *A heuristic approach to improve a branch and bound based program partitioning algorithm, the proceedings of the 1999 International Workshop on Innovative Architecture (IWIA '99)*, pp.105–114 (Nov. 2000).

Masami Takata, Yoshitoshi Kunieda and Kazuki Joe: *Accelerated Program partitioning Algorithm - An Improvement of Girkar's Algorithm -, the proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '00)*, Vol.II, pp.699–705 (Jun. 2000).

研究会（査読なし）

髙田雅美，庄野逸，岡田真人，城和貴：誤り訂正符合のナイーブ平均場近似，**電子情報通信学会技術研究報告** *NC 2002-172-202*，Vol.102，No.730，pp143–148（2003年3月）．

齊藤哲哉，髙田雅美，城和貴，國枝義敏，福田晃：ニューラルネットワークによるプログラム分割アルゴリズムの改善，**第 *28* 回　情報処理学会数理モデル化と問題解決研究会**，MPS-28-6，Vol.2000，pp21–24（2000年2月）．

齊藤哲哉，髙田雅美，城和貴，國枝義敏，福田晃：実用的な近似解を与えるプログラム分割アルゴリズム，**第 *27* 回　情報処理学会数理モデル化と問題解決研究会**，MPS-27-8，Vol.99，pp29–32（1999年11月）．

髙田雅美，齊藤哲哉，中西恒夫，城和貴：分枝限定法を用いたプログラム分割の下界に対する考察，**情報処理学会計算機アーキテクチャ研究会**，ARC-134-14，Vol.67，pp.79–84（1999年8月）．