九州大学学術情報リポジトリ
Kyushu University Institutional Repository

大規模並列計算機上における反復解法の実行性能予測モデルに関する研究と反復解法の高効率化法

阿部, 譲司

https://doi.org/10.15017/1654914

出版情報：九州大学, 2015, 博士（工学）, 課程博士
バージョン：
権利関係：全文ファイル公表済

概要

偏微分方程式の高速な求解は，多くの理工学の分野において必要とされる．近年の HPC システムの進展により，複数の方程式を連成(マルチフィジクス）させたり，多階層に渡る方程式を連立（マルチスケール）させたりすることが可能となりつつある．これらの問題を実際の形状を考慮して解析を行う場合には，有限要素法，有限体積法，有限差分法等を用いて方程式を離散化する必要がある．したがって，大規模なマルチフィジクス・マルチスケール解析を行うことは，それを離散化する際に生成される大規模行列を線形方程式として求解することと等価となる．通常このような大規模問題を解く場合，メモリ消費量および計算時間の観点から直接法ではなく反復法が用いられる．

反復法によって大規模な問題をより高速に解こうとした時，従来のスイッチを基幹とした多段の動的ネットワークを使った並列計算機アーキテクチャでは，多数の PE(Processing Element)を用いた場合に，速度上の障壁の存在が予想される．このため，Exascale 時代を迎えるのに先立って，この障壁を少しでも緩和させるために富士通 京，IBM BG/Q，CRAY XC30 などのマシンでは，PE 間を静的なダイレクトネットワークで直結するアーキテクチャを採用している．

静的ネットワークを用いてダイレクトネットワークタイプの分散並列計算機を構成する場合，ネットワークトポロジーが計算機の性能を左右する．よって，ネットワークトポロジーの性質を表す性能指標を求め，それを用いてトポロジー間の特性比較を行うことは，アプリケーションを定めず簡易的に評価できる点で，非常に有用である．近年，nVIDIA GPU，Intel Xeon Phi をはじめとする HPC 向けのプロセッシングシステムでは，低いクロック周波数で駆動される低機能なプロセッシングエレメントを多数配置することで，エネルギー消費量を抑制しつつ全体として高性能を達成している．その際，LSI Chip 内に，いままで並列計算機で使われて来たインターコネクトを埋め込む必要が生じている．その結果，通常の並列計算機を構成する大規模なインターコネクトと Chip 内部のインターコネクトの両方を合わせた，より大規模ネットワークをベースとして並列計算機のアーキテクチャ設計を行う必要性が出てきている．

このような背景のもと，本論文では，Exsascale 時代以降の並列計算機システム上で反復

法を実行する場合の実行性能予測法を新たに提案する．また，反復法を並列計算機上で高効率に実行させる場合の方法について，行列の並列計算機上へマッピング法による方法およびPreconditionerに着目する方法で実現した．以下この三点について本論文の要約を述べる．

第一に，大規模なダイレクトネットワークによって結合された多数のPEよって構成された科学技術計算向け並列計算機上で実行されるアルゴリズムの特性を評価する方法を新たに提案した．インターコネクトネットワークトポロジーの評価に関しては，平均PE間距離を用いた．いままで近似値しかなったが，新たな手法を提案し，非対称構造も含むトポロジー毎の完全な評価式を新たに求めた．これによって CPU 内部のインターコネクトネットワークの新たな設計指標を提案した．並列計算機上の性能評価モデルは，インターコネクトネットワークトポロジー，ネットワーク転送速度，ネットワークリンク遅延時間，PEの浮動小数点演算性能，PE数により構成される．並列計算機上でのアルゴリズム実行特性の予測に関しては，PE間のpeer to peerの伝送特性およびPE当たりの演算性能を計測もしくは仮定すれば，今後想定される メニコア CPU，もしくはオフロードタイプの GPU より成る大規模並列計算機上での性能予測も可能となる．提案法を 3 次元有限差分法より生成された行列を並列計算機上で共役勾配法によって求解する場合に適用した．共役勾配法は，内積等のグローバルな通信を伴う演算，近接PEとの通信を伴う行列ベクトル積演算，通信を伴わないベクトル演算より成る．これらの演算に対して，評価モデルを適用し，インターコネクトネットワークの形状，ネットワーク転送速度，ネットワークリンク遅延時間，PEの浮動小数点演算性能，PE数の関数として，実行時間予測式を新たに求めた．これらの評価モデルと新規に提案した通信遅延隠蔽状態に関する指標を用いて，3次元有限差分法で離散化した行列をCG法で解く場合の処理速度特性などを求めた．次に，CG法を実行した場合の並列計算機の限界実行性能をもたらす最適PE数の解析的モデルについて提示し，その結果に基づいてインターコネクトトポロジー毎の限界実行性能を解析的に示した．

第二に，反復法等で必要となる行列ベクトル積を並列計算機上で高効率に実行する方法として，ブロック交換法によって行列の非零要素を均等に並列計算機上のスレッドに割り当てる方法，BLEX法を新たに提案した．周知のように，行列ベクトル積は線形方程式を解く時によく用いられる，そしてその効率的な計算というのは重要である．特に，行列ベクト

ル積は並列計算機上で線形方程式を解く時の計算において大きな部分を占める．計算の速度は，ロードバランス即ち，それぞれのプロセスにおいて行列の非零行列成分をいかに均等に分割するかに大きく依存する．一般的に Block Cyclic (BC)分割は非零行列成分を均等に分割する点においては効果的な手段である．しかし，BC 分割における並列性能は，扱うブロックの数が変わることで大きく変化することは有名である．それゆえ，BC 分割においては，最適なブロックの数を決めることが難しい．

本論文では，ブロック交換法によって，行列の非零要素を均等に並列計算機上のスレッドに割り当てる方法を提案した．本論文では，通常の BC 分割に基づいて，BLEX 法を実現するにあたり，2 種の戦略を採用した．第一の戦略は，行列の非零要素をスレッド毎にほぼ均等に分割することである．第二の戦略は，ブロック交換法を用いることで，さらに均等な分割を行うことである．結果として，BLEX 法は BC 分割と伴に行列の非零要素を分散したスレッドに非常に均等に割り当てることを可能とした．

第三に，Inverse-based ドロッピング手法の考え方を対称行列に適用し，さらに原論文では十分明らかにされなかった Inverse-based ドロッピング手法が持つ収束特性をより明確にした．クラウト(Crout)版 ILU 前処理には，上三角行列Uの逆行列U^{-1}の列毎のノルムの大きさを見積もる Inverse-based ドロッピング技法を採用することができるという利点がある．本論文では，対称正定値行列を係数に持つ線形方程式に対して，上記の逆行列をベースにしたドロッピング技法をどのように拡張するかについて記述した.分解過程で発生するフィルインを棄却するドロッピング操作が前処理行列の逆行列U^{-1}に与える影響を見積もり，すなわち，U^{-T}の行のノルム(U^{-1}の列のノルム)をできるだけ正確に見積もり，それに基づきドロッピングする対称行列用の ib_IC 分解を導出した．数値実験を通して，ib_IC 分解つき CG 法の収束は，従来のドロッピング手法に基づく IC 分解つき CG 法の収束と比較して，閾値の変化に対して安定であることがわかった．

大規模並列計算機上における反復解法の実行性能予測モデルに関する研究と反復解法の高効率化法

阿部譲司

２０１６年３月

九州大学大学院
システム情報科学府
情報工学専攻

目次

第 1 章　緒言 1
1.1　研究の背景 1
1.1.1　CPU の基本技術の形成とその変遷 1
1.1.2　並列計算機アーキテクチャ技術の形成と発展 2
1.1.3　今後のスーパーコンピュータに求められるブレークスルー 3
1.2　学術的背景 5
1.2.1　並列計算機上でのアルゴリズム実行性能予測 5
1.2.2　行列ベクトル積の計算の高速算法 5
1.2.3　逆行列ベースの要素棄却型の IC 分解前処理の性能評価 6
1.3　本研究の目的と意義 7
1.4　本研究の実施方法 8
1.5　本論文の構成 9
第 2 章　並列計算機上の行列解法の実行速度特性 2
2.1　並列計算機モデルの定義 2
2.2　メニコア CPU の Chip 内インターコネクトトポロジー評価指標の提案 4
2.2.1　Linear Array 構造 7
2.2.2　2 次元 Mesh 構造 8
2.2.3　3 次元 Mesh 構造 11
2.2.4　Ring 構造 14
2.2.5　2 次元 Torus 構造 15
2.2.6　3 次元 Torus 構造 18
2.2.7　Hypercube 構造 22
2.2.8　インターコネクトトポロジーの特性比較 24
2.3　対象とする問題の定義 30
2.4　並列環境で反復法を実行する場合の基本演算要素 35
2.4.1　CG 法アルゴリズムと演算 36
2.4.2　行列ベクトル積に伴う通信 37
2.4.3　内積演算に伴う通信 40
2.5　並列計算機上での実行時間予測モデル 43
2.5.1　処理時間の定式化 43
2.5.2　通信遅延隠蔽閾値 PE 数 46

2.5.3 PE 数と処理速度の関係 50
2.5.4 Strong scaling 51
2.6 並列処理システムの限界実行速度 54
2.7 第 2 章のまとめ 62
第 3 章 逆行列ベースの要素棄却型の IC 分解前処理の性能評価 63
3.1 は じ め に 63
3.2 従来の IC 分解とフィルインの棄却 64
3.3 IC 分解で生じる誤差に関する考察 65
3.4 Inverse-based ドロッピング手法に基づく IC 分解 67
3.5 Ib_IC 分解の算法 70
3.6 数値実験 72
3.6.1 計算機環境と計算条件 72
3.6.2 実験結果 74
3.6.3 考察 84
3.6.4 2 種類の IC 分解の収束速度と残差について 85
3.7 第 3 章のまとめと今後の課題 85
第 4 章 行列ベクトル積の並列計算機用のブロック交換法 86
4.1 はじめに 86
4.2 従来の分割方法 87
4.2.1 ブロック分割 87
4.2.2 ブロック巡回分割 88
4.3 BLEX 法 89
4.3.1 ブロック数の自動決定 89
4.3.2 ブロック交換技術による均一な非零行列成分の割り当て 90
4.4 数値実験 91
4.4.1 BLEX 法における計算コスト 96
4.4.2 ブロック巡回分割ならびに BLEX 法における行列ベクトル積の並列計算性能推定 97
4.5 第 4 章のまとめ 103
第 5 章 結言 105
5.1 本研究の結論 105
5.2 今後の課題と展望 106
参考文献 108

第1章　緒言

1.1 研究の背景

1.1.1 CPU の基本技術の形成とその変遷

1971 年，Intel 4004 という IC(integrated circuit)チップが発表された．この IC は小型の 16pin 汎用パッケージングでありながら，2300 個のトランジスタからなる論理回路により 4bit 整数演算部，アキュムレータ，レジスタなどのコンピュータとしての最低限の必須機能を有していた．全てはここから始まった．

その後，データ幅は 8bit 化され 1972 年に Intel 8008, 1974 年に Intel 8080, 1976 年に Zilog Z80 などが相次いで発表された．しかしながらこれらの 1 チップマイコンは整数演算のみが実行可能であり，科学技術計算用としての需要に応え得るものではなかった．一方では科学技術計算分野では依然としてディスクリート素子による汎用計算機の使用がほとんどであった．この頃の汎用計算機の演算性能は IBM system/370 [60]で 1MIPS である．すなわち，当時のコンピュータは 1 チップマイコンによる組み込み・個人用途と，汎用計算機による科学技術計算用途およびデータ処理に二極分化していた．

さて，1980 年に発表された Intel 8086+8087 のチップセットでは浮動小数点演算が実現され，コンピュータ，あるいはプロセッサとしての一定の完成度に到達したと考えられる．Intel 8086+8087 プロセッサの基本性能としては，32bit 浮動小数点演算で 50 kFLOPS 程度であった [62]．後で述べるが，Intel は Intel 8086+8087 の後継 Intel 80286+80287 を用いて，汎用 CPU を用いた分散メモリタイプの商用並列計算機 iPSC/1 [66] [4]を開発販売した．

その後，1999 年には，dual core の IBM Power4 が発表された [59]．そしてこれをひとつ

の端緒として，ここにシングルプロセッサでクロック周波数向上による高性能化競争は終止符が打たれ，core 数を増加させるマルチコア時代へ突入し，今に至っている．2015 年 9 月現在では，2cpu で 1 共有メモリノードを構成する場合，最大 36core 構成が可能となっている．2016 年には Intel は Xeon Phi knights Landing で 72core cpu を実現させるとアナウンスしている [43].

1.1.2 並列計算機アーキテクチャ技術の形成と発展

科学技術向け計算機は，1976 年に CRAY-1 [48]から始まった．CRAY-1 は，パイプライン演算器とレジスタ群で構成されたベクトル計算機である，ベクトル計算機は例えば流体計算のように演算器に対するデータロードが多い時に非常に有効な計算機である．この種の計算機は，後継の CRAY-2，CRAY XMP，CRAY YMP，富士通 VP シリーズ， 日立 S810/S820，CDC ETA10，NEC SX シリーズ/地球シミュレータなどに引き継がれ，1990 年代を中心に一世を風靡した．ベクトル計算機は，ベクトル演算主体で構成されるアプリケーションについては，得られる速度はその当時の最高速度であった．しかし，例えばベクトル演算が 90%程度以下となるアプリケーションに関しては，アムダールの法則に従って，速度は限りなくスカラー演算に近くなってしまうこと，価格的に高価であること，CPU の性能が著しかったことなどにより，世界的には 2000 年代になると姿を消して行った．

この動きと同時期に，すなわち，1980 年代後半により，カスタム LSI もしくは汎用 CPU をネットワークもしくはバスによって結合した並列計算機が多数開発された．バス結合によるもとしては，TITAN/Stardent，Alliant，Sequent，Encore，KSR などがあるが，結合の性質上速度的には限られた性能しか得ることができなかった．

他方，ネットワークによる結合によるアーキテクチャとしては，Thinking Machines，Intel，IBM，nCUBE，FPS，BBN などがある．これらの内で Thinking Machines CM-1 は集合演算をネットワークルータ上で行うなど「京」でも用いられた方法を採用し，基本アーキテクチャおよび命令実行方式は後に Grape-DR [8]に強い影響を与えた．また，Intel については，CPU とネットワークについては，汎用製品を用い，OS についても Unix サブセットを用いていたことより考えると，現行の並列計算機の原型であったと考えることがで

きる．

結果として，CM1 以降のスーパーコンピュータは，PE(processing unit)を接続して全体として一台のコンピュータとして動作させる方式，すなわち並列アーキテクチャ(Parallel Architecture)に収斂して行った．多数の PE をインターコネクトで結合し MPI(Message Passing Interface)によって全体を制御することによって構成された MIMD(Multiple Instruction Multiple Data Stream)型の並列計算機構成する方式は現在のスーパーコンピュータまで踏襲されている．

その結果，2000 年代後半以降は，CPU のマルチコア化の加速，nVIDIA Tesla GPU，Intel Xeon Phi などのアクセラレータの登場が特徴的であった．Top500 [57]によると 2015 年 6 月現在の性能第 1 位は，アクセラレータに Xeon Phi を用いた中国の天河 2 号である．浮動小数点演算性能は 55PFLOPS であり，300 万 core より構成されている．また，同サイトでの公式予測によると 2020 年には 1e18 FLOPS すなわち 1 Exa FLOPS に到達すると見込まれている(Fig. 1-1 参照)．

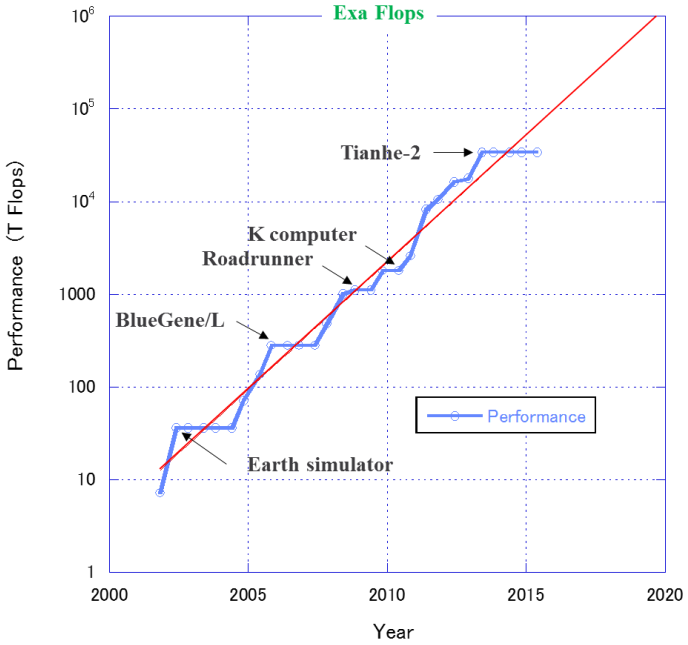


Fig. 1-1 HPC Performance Development History

1.1.3 今後のスーパーコンピュータに求められるブレ

ークスルー

2015 年時点，CPU では Intel が 2.3GHz 18 core の Haswell をリリースし，ピーク性能としては，660GFLOPS に達している．通常 2 個の CPU で SMP(Symmetric Multiprocessing)を構成するので，倍の 1.32TFLOPS となる．nVIDIA Tesla GPU，Intel Xeon Phi はともにピーク性能が 3TFLOP となっている．ともに PCI-e を介して CPU よりデータとのデータのやり取りを行う必要があり，通信による速度低下は大きなものがある．

TOP500 の性能測定は HPL(High Performance Linpack)で行なわれ，通信の比重はとても低く問題にはならないが，実アプリケーションではこれが，大きな問題となる．次世代の Xeon Phi KNL(Knights Landing)では，PCI-e を介して CPU と接続するのではなく，自らが 240 core 超のメニコア CPU として動作可能となる．そして，自身がクラスタの PE(Processing Element)としてネットワークに接続することが可能となる．Xeon Phi KNL はバンド幅が 400GB/秒となる高速な 3 次元メモリを有するため，分散メモリタイプのベクトル計算機に近い存在となると思われる．

振り返ってみると，これらの現在のスーパーコンピュータのプロセッシングエレメント(PE: processing unit)の原型は 1982 年にリリースされた Cray X-MP にあるといえる．Cray X-MP は，世界初の共有メモリ型並列ベクトル計算機であり，共有メモリ+SIMD(Single Instruction Multiple Data Stream)プロセッサタイプの現在のマルチコア CPU，Xeon Phi KNL，GPU と同じ方式と考えることができる．Exascale 時代以降以降もメニコア化による高速化は最も重要となるが，可伸性の点では限界があり，メニコア化に際しさらなるブレークスルーが必要となる．

したがって，Xeon Phi KNL では，大規模並列を想定して，ネットワークファブリック自体を CPU die に取り込む方式を採用する．今後は，Cray XC30 [19]，京コンピュータ [7]，IBM BG/Q [18]などで採用されている静的ネットワークで PE 間を直接結合する方式に向かう方向とされている．よって，Exascale を目指すにあたり，分散メモリ型アーキテクチャ，プロセッサのメニコア化と SIMD 化，静的インターコネクトを用いた PE 間ダイレクトネットワークが必要となることは明白である．よって，本論文では，この前提を基に議論を以降の展開する．

1.2 学術的背景

本節では，前節の研究の背景のもとに展開される本論文の 2 章，3 章，4 章の研究の学術的背景を以下に述べる

1.2.1 並列計算機上でのアルゴリズム実行性能予測

10 年以内に到来が予想される Exascale [10] [20] [25] [32]時代では，1 つの JOB で数万以上の PE 数が使用可能となる．その結果，大規模な単一並列 JOB の処理速度の向上が常に求められる Capability Computing 色がより強くなることが確実視されている．並列計算機のハードウェアとしての特性を決める主要な要素は PE の単体性能，PE 数，インターコネクト送受信性能，インターコネクトトポロジーである．新規に超並列計算機を設計する際にこれらの要素は重要な設計パラメータであり，これらのパラメータを基に大規模な問題に関して，単一並列アルゴリズムを想定して，システム全体の設計および最適化を行う必要があるが，現在はコンポーネント毎の H/W の最適化に終始しており，理想とかけ離れた状態にある．もし，実行アルゴリズム，計算機アーキテクチャ，PE 数で性能が予測可能となれば，並列計算機設計時のみならず，プログラム実行時にも最適な PE 数を選択できるため，システム全体のスループットの向上も期待される．本研究では，実行アルゴリズムを幅広い領域で利用されている共役勾配法にフォーカスして論を進めるが，方法自体は一般的なアルゴリズムに拡張可能な方法である．

1.2.2 行列ベクトル積の計算の高速算法

行列ベクトル積は並列計算機上で線形方程式を解く際の計算において大きな部分を占める [50] [58]. そのため，多数の行列ベクトル積の高速計算アルゴリズムが報告されている [14] [31] [34] [64] [65]．しかし，そのようなアルゴリズムに対する並列計算機上でのトータルな計算速度は，行列の非零行列成分を各スレッドにどのように割り当てるかに大きく依存する．

一般的に Block Cyclic (BC)分割 [37]は非零行列成分を均等に分割する点においては効果

的な手段とされている．しかし，BC 分割における並列性能は，扱うブロック数に応じて大きく変化することが知られており，それゆえ，BC 分割における最適なブロック数を決定することは困難とされていた．このブロック数の決定が可能となれば行列ベクトル積の並列計算機上での実行速度改善が期待される．

1.2.3 逆行列ベースの要素棄却型の IC 分解前処理の性能評価

連立一次方程式を反復法で解く場合，前処理として不完全コレスキー分解(Incomplete Cholesky factorization，以下 IC 分解と表記する)が広く用いられている．IC 分解では，係数行列を近似する前処理行列を生成し，その近似逆行列を係数行列に作用させて変換し，変換された後の方程式に対して共役勾配法(Conjugate Gradient，以下 CG 法と表記する)などの反復法を適用する．さらに，ドロッピングつき IC 分解では，元の係数行列では値が零の要素が，前処理行列の分解過程で零でない値を持った要素(以下，フィルインと呼ぶ）に対して，予め定めた閾値より小さい場合はそれを棄却するという不完全分解を行う．その際，閾値の大きさによっては，反復法の収束性に大きなばらつきが生じることがある．そのばらつきの判定のためには与行列の近似逆行列の誤差行列を評価する必要があるにも関わらず，その適切な評価がされていなかった [47] [46] [15] [16].

それに対し，Li は任意の正則行列用の前処理として Crout 版 ILU 分解(以下，ILUC 分解と表記する）を提案した [36]．文献 [36] では，従来の ILUT(dual Threshold ILU）分解 [49] [50]との収束性の違いが明らかにされ，さらに逆行列の誤差行列 X の具体的な評価方法について Inverse-based dropping 手法（以下，IB ドロッピング手法と表記する）と呼ばれる方法が提案された．

さて，行列の次元数 n に対して，特異値分解の計算量は$O(n^5)$で評価されることが知られている．一方，反復法の計算量はc $O(n^2)$で評価される．ここで，計算量の係数 c は収束までの反復回数によって決まるため，反復法の研究では，少ない反復回数で近似解が得られることが重要とされる．また，係数 c の大きさは通常は，次元数 n と比べて相当に小さく，したがってオーダーも$O(n^3)$を超えることはない．この考え方を基に並列計算機を志向して

対称行列に限定したアルゴリズムを構成すれば近似解の高速求解が可能となる．

1.3 本研究の目的と意義

1.2.1 小節でも述べたように Exascale 時代を迎えようとしている現在, 1 つのジョブにかけられる PE 数は拡大の一途をたどっている．一般的には，多段に接続されたスイッチで Fat tree ネットワークを用いて並列計算機を構成し計算を行うこととなるが, 多数の PE を用いる場合には実行効率が低下し，ハードウェアの性能が生かせなくなることが予想される．

本研究の着想の発端はここにある. 例えば, 現行の富士通 京コンピュータ, Cray XC30, IBM BG/Q ではすでに 3D torus，Dragonfly network [23]，Hypercube などのインターコネクトトポロジーが活用され，より大きな PE 数で実行効率を上げ，全体の計算速度を引き上げようという動きがある．しかし，そのようなインターコネクトを用いた場合，ターゲットとするアルゴリズムに関して PE 数を増加させた際に, どの程度のパフォーマンスが得られるのか，もしくは，CPU の演算性能，ネットワーク伝送特性，ネットワークトポロジーを変化させた場合の性能予測，HPC アーキテクチャ設計時において一大関心事ともなると考えられる．またそうして設計された超並列計算機上での最適なアルゴリズム構成はいかにあるべきなのか，という超並列アーキテクチャ向きのアルゴリズム最適化問題も重要な解決すべき課題である．

したがって，このような着想経緯とモチベーションに基づいて本研究の目的を次のように定める. すなわち, 第一に静的ダイレクトネットワークを用いた理想的な超並列計算機の性能予測, 第二にと実計算機上での反復法アルゴリズムの最適化である. 特にターゲットとする計算分野は，行列演算とする．これは HPC が要求される科学技術計算では線形モデルを扱うことが多く，行列とベクトルで表現された連立線形方程式の求解問題に帰着されること，また，多変量の連立高階偏微分方程式でモデルかされる問題にあっても有限要素法(Finite Element Method: FEM)あるいは有限差分解析法(FDM)などによって連立線形方程式に近似可能であることという理由に基づいている．

また本研究の意義として, 本研究の成果は超並列計算機, そしてソフトウェアの設計開発

の現場での利活用が挙げられる．また，学術的には HPC 分野ひいては計算理工学，情報科学，電子工学の関連分野での貢献も期待できる．特に超並列アーキテクチャの新たな方式，インターコネクトの新たなトポロジー，HPC 向けのアルゴリズムといった種々の性能評価に関連する学術研究の一道標となることが期待される．

1.4 本研究の実施方法

本研究では超並列アーキテクチャ上での最適化された行列演算アルゴリズムと性能事前評価指標を与えるための 3 種の項目を定める．すなわち，

第一課題: Chip 内インターコネクト評価法と並列計算機上の行列解法の実行速度特性予測法の構築

第二課題: 対称行列向け ib_IC 分解の導出とその CG 法への適用

第三課題: BC 分割に基づいた BLEX 法の実現

である．

さて，本研究の実施にあたっては次のような理想並列計算機を想定，対象とする．すなわち，分散メモリを用いる SPMD(Single Program Multiple Data）型であり，処理を実行する PE と，通信および同期に用いる CL より構成される．PE 間は静的なダイレクトネットワークより結合され，そのインターコネクトトポロジーとインターコネクト伝送特性の多様性により，種々の計算機アーキテクチャの表現が可能となる．また，システム全体は，統一的なクロックにより同期がとれており，同期ずれに伴う時間的ロスはない．

以下では上述の各課題での研究実施方法を示す．

第一の課題では，はじめに Chip 内インターコネクトの評価指標として平均 PE 間距離を採用する．新たに平均 PE 間距離の導出法を提案することにより，正確な平均 PE 間距離を用いて Chip 内インターコネクトの評価を行う．次に，理想並列計算機上での実行性能を予測する方法を提案する．新たに通信遅延隠蔽閾値 PE 数を定義し，それを用いて実行時間を予測する．さらに，提案法を用いて，FDM による離散化の際に生成される行列に，理想並列計算機上で CG 法を実行する場合の性能を予測する数値的な方法を導出する．限界実行

速度と限界実行速度をもたらす PE 数について, 理想並列計算機を記述する際に用いたパラメータの関数として解析的に求める方法を新たに提案する. そして, 与えられた問題のサイズと限界実行速度の関係性について考察する.

第二の課題では IC 分解で生じる誤差, 前処理行列の逆行列のノルムの評価方法とそれに基づきフィルインに対するドロッピング方法を考察する. 次いで, ib-IC 分解 のアルゴリズムを構築して, 数値実験を通じて, 閾値を変化させ, 従来の IC 分解の収束不安定性と ib-IC 分解の収束安定性の相違を抽出する.

第三の課題では, 通常の BC 分割に基づいて BLEX 法を実現するにあたり, 2 種の戦略を採用する. 戦略(1)としては, 行列の非零要素をスレッド毎にほぼ均等に分割することである. 戦略(2)では, ブロック交換法を用いることで, さらに均等な分割を行うものである. 結果として, BLEX 法は BC 分割と伴に行列の非零要素を分散したスレッドに可能な限り均等に割り当てることを目指す.

さらに, 以上の各課題の結果を踏まえた総括的な論考を与えて本研究の結論とする.

1.5 本論文の構成

本論文は全 5 の章で構成さる.

第 1 章では, 超並列アーキテクチャ上での行列演算算法と性能評価に関する本論文の緒言としてその導入を図る. 1.1 節では本研究の背景としてプロセッサの基本技術の形成と需要の変遷, 並列計算機アーキテクチャ技術の形成発展について述べ, 今後のスーパーコンピュータに求められるブレークスルーまでの過程を概観する. 1.2 節では本研究の直接的に関与する学術的背景について概説する. 以上の研究背景を着想の経緯として 1.3 節では本研究の目的を, 超並列アーキテクチャを志向した行列演算算法と理想コンピュータの性能および性能限界の予測と定めるとともにその産業上および学術上の意義を示す. 1.4 節では本研究の実施方法について 3 つの検討課題のそれぞれについて実験あるいは解析的手法につい

て説明する．1.5 節では全 6 の章からなる本論文の構成を示す.

第ニ章では，メニコア CPU 内の Chip 内インターコネクトトポロジー評価法と，理想並列計算機上のアルゴリズム実行性能を予測するための，通信遅延隠蔽閾値プロセッシングエレメント数を用いた方法を提案する．2.1 節ではここでの検討課題の導入を行う．2.2 節ではメニコア CPU の Chip 内インターコネクトトポロジー評価指標の提案について述べる．2.3 節では対象とする問題の定義について述べ，2.4 節では並列環境で反復法を実行する場合の基本演算要素について触れる．2.5 節では並列計算機上での実行時間予測モデルついて提案し，本手法を用いて並列計算機上での CG 法実行時の諸性を示す．2.6 節では，提案法を用いて, 並列計算機上で行列に CG 法を適用する場合の限界実行速度とその際の PE 数を解析的に示し，さらに，与えられた問題のサイズと限界実行速度の関係を示す．2.7 節では本章の検討結果を要約してまとめる.

第 3 章では, 逆行列ベースの要素棄却型の IC 分解前処理の性能評価について述べる．3.1 節では本章の導入を企図して目的と意義について概観する．3.2 節では, IC 分解で生じる誤差に関する考察を行う．3.3 節では，前処理行列の逆行列のノルムの評価方法およびそれに基づきフィルインに対するドロッピング方法について述べる．3.4 節では IB ドロッピング手法に基づく IC 分解のアルゴリズムを記述する．3.5 節では，数値実験を通じて，閾値を変化させ，従来の IC 分解の収束不安定性と提案法の収束安定性の違いについて明らかにする．3.6 節では，本章の内容をまとめる.

第 4 章では，行列ベクトル積の並列アーキテクチャ用のブロック交換法 BLEX を提案する．4.1 節では本章の検討課題の導入を行う．4.2 節では，BLEX の詳細について述べる．提案法は，並列計算機上の疎行列ベクトル積の高速化を行うために 2 種類の方法を用いている．次いで 4.3 節では，数値実験を通して提案法の効果について評価する．4.4 節では本章の検討結果をまとめる.

第五章では本論文の結言として本研究を総括する．5.1 節では本研究の成果を要約して結論とする．5.2 節では本研究での問題点を指摘して今後の課題を導く．5.3 節では本研究の成果に立脚して次のフェーズとして超並列アーキテクチャに関する学術上の展望を与えて本論文の結語とする

第2章　並列計算機上の行列解法の実行速度特性

本章では，メニコア CPU 内の Chip 内インターコネクトトポロジー評価法 [52]と，理想並列計算機上でのアルゴリズム実行性能を予測するための，通信遅延隠蔽閾値プロセッシングエレメント数を用いた方法を提案する．2.1 節ではここでの検討課題の導入を行う．2.2 節ではメニコア CPU の Chip 内インターコネクトトポロジー評価指標の提案について述べる．2.3 節では対象とする問題の定義について述べ，2.4 節では並列環境で反復法を実行する場合の基本演算要素について触れる．2.5 節では並列計算機上での実行時間予測モデルついて提案し，本手法を用いて並列計算機上での CG 法実行時の諸性を示す．2.6 節では，提案法を用いて，並列計算機上で行列に CG 法を適用する場合の限界実行速度とその際の PE 数を解析的に示し，さらに，与えられた問題のサイズと限界実行速度の関係を導出する．2.7 節では本章の検討結果を要約してまとめる．

2.1 並列計算機モデルの定義

本章で対象とする並列計算機は，分散メモリを用いる SPMD(Single Program Multiple Data）型の計算機である．したがって，並列計算機は処理を実行する PE(Processing Element)と，通信および同期に用いる CL(Communication Link)より構成される．PE 間は静的なダイレクトネットワーク [24] [23]より結合され，そのインターコネクトトポロジーの多様性により，種々の計算機アーキテクチャの表現が可能となる．また，システム全体は，統一的なクロックにより同期がとれており，同期ずれに伴う時間的ロスがない理想的な計算機を想定する．

Fig. 2-1 (a)-(g) に，本章における対象インターコネクトトポロジーを示す．ここで，黒い球は PE を表し，線分は CL を表す．本章では，Linear Array，Ring，2D Mesh，2D Torus，

3D Mesh，3D Torus，Hypercube の 7 構造を対象とする．

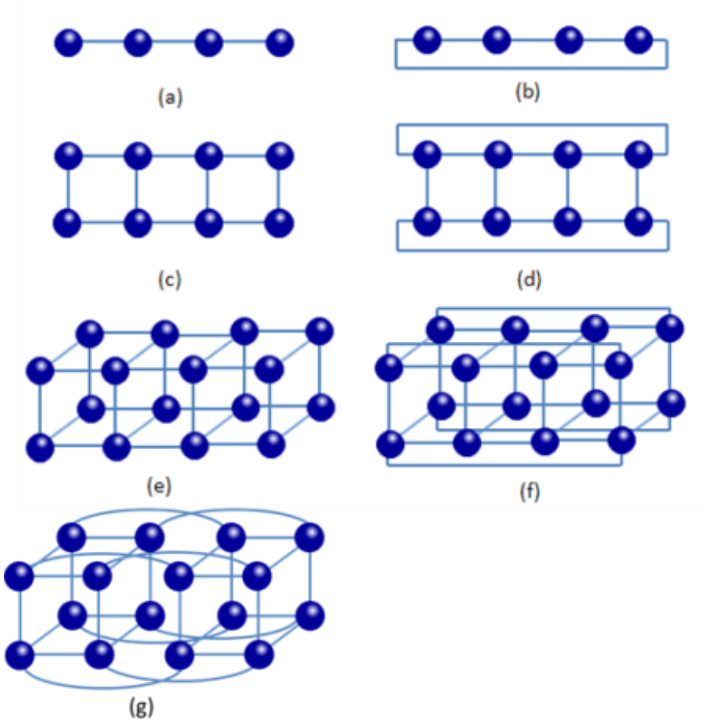


Fig. 2-1 Network Topologies：(a) Linear Array (b) Ring (c) 2D Mesh (d) 2D Torus (e) 3D Mesh (f) 3D Torus (g) Hypercube.

次に，PE を中心とした，より詳細な通信 model を Fig. 2-2 (a),(b)に示す．ここで，一つの port（図中の黒四角）は，同時に「送信」または「受信」動作を行うことができる．port は隣接 PE の port と Link を介して直接結合する．

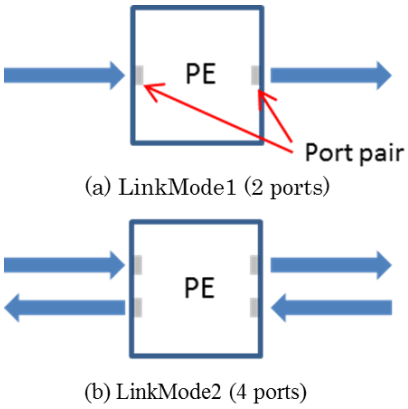


Fig. 2-2 Communication model of a PE.

各 PE は独自の PE 番号を持っており，各次元方向において，基本的に，自分より番号が若い PE（Predecessor）と，自分より番号が大きな PE（Successor）と複数の port ペアで結合する．Port ペアは独立して動作可能である．隣接 PE と結合する Port ペアの数によって，PE のプロパティーを区別することができる．Fig. 2-2 に LinkMode と呼ぶモデル例を示す．例えば，Fig. 2-2 (a) LinkMode1 の port ペアでは，自 PE へのデータの「受信」と他 PE へのデータの「送信」を同時に処理できるものとする．また，Fig. 2-1 (b) LinkMode2 では，二つの port ペアの同時動作が可能であるとする．

次に，CL の伝送特性のモデルを式(2-1)に示す．

$$T_c(n) = \frac{n}{c} + \tau_c \tag{2-1}$$

ここで，T_cは転送時間[s]，nは転送 word 数，cは転送速度[word/s]，τ_cはセットアップ時間[s]とする．非常に長いパケットを送信する場合でも，このモデルが適用することができるとする．

2.2 メニコア CPU の Chip 内インターコネクトトポロジー評価指標の提案

静的ネットワークを用いてダイレクトネットワークタイプの分散並列計算機を構成する

場合，ネットワークトポロジーが計算機の性能を左右する．よって，ネットワークトポロジーの性質を表す性能指標を求め，それを用いてトポロジー間の特性比較を行うことは，アプリケーションを定めず簡易的に評価できる点で，非常に有用である．近年，nVIDIA GPU，Intel Xeon Phi [8]をはじめとする HPC 向けのプロセッシングシステムでは，低いクロック周波数で駆動される低機能なプロセッシングエレメントを多数配置することで，エネルギー消費量を抑制しつつ全体として高性能を達成している．その際，LSI Chip 内に，今まで並列計算機で使われて来たインターコネクトを埋め込む必要が生じている．その結果．通常の並列計算機より小規模かつ形状が非対称なインターコネクトの評価を行う必要性が出てきている．

あるネットワークトポロジーを用いた並列計算機中内でランダムに通信が発生する場合を考える．これは 1 つの JOB もしくは複数の JOB を並列計算機上で実行する場合に起こる通信パターンに相当する．Fig. 2-3 に示すように，PE_iより最短経路上にある中継 PE を介して目的とするPE_jにメッセージを送信する時，メッセージを送出する PE を除いて介在する PE 数を距離と定義する．

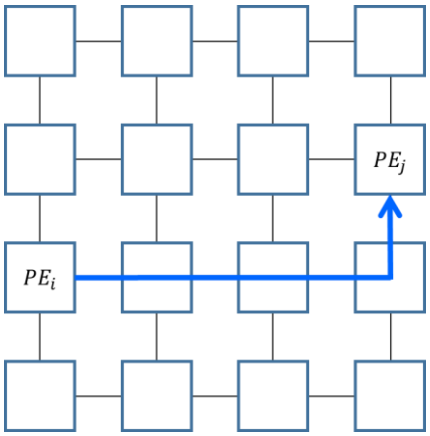


Fig. 2-3 Definition of distance between PE_i and PE_j.

ある PE より他の全ての PE へメッセージを送信する組み合わせの距離の総和を PE ペアの組み合わせ総数で除したものをシステムの平均距離とし，本章ではこれを用いて議論を進める．この PE 間の平均距離は，システムが JOB を処理する能力に直接影響を及ぼす指

標となる．平均 PE 間距離が短いほど通信遅延を抑えることができ，高スループットが期待できる．

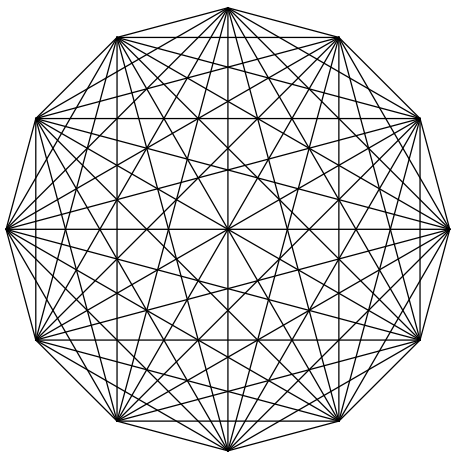


Fig. 2-4 Combinations of PE pair for communication.

12 台の PE がある場合の通信パターンを Fig. 2-4 に示す．Fig. 2-4 では 12 角形の頂点に PE をマップしている．頂点と頂点を結ぶ直線は通信を行うペアを示す．総通信距離を求める場合は，これらのペアに関して最短通信経路を計算し総和を取る必要がある．式(2-2)に PE 間の平均距離$\bar{d}$の定義を示す．総通信距離を PE の組み合わせ数の倍で除すことで求めることができる．

$$\bar{d} = \frac{1}{p(p-1)} \sum_{i=1}^{p} \sum_{j=1}^{p} Dist_{ij} = \frac{TotLen}{p(p-1)} \tag{2-2}$$

ここで，p は PE の台数，$Dist_{ij}$は i 番目の PE より，j 番目の PE までの距離を表し，$TotLen$ は PE 間通信距離の総和を表す．式(2-2)では，i および j を PE の id と考え用いたが，インターコネクトの形状が多次元になると，PE を特定するためには id ではなく，PE があるトポロジー内での場所によって識別する必要がある．すなわち，トポロジーが 2 次元であれば PE の位置を表すp_x，p_yを，3 次元であれば PE の位置を表すp_x，p_y，p_zを i の代わりに用いて式(2-2)を記述する．例えば Fig. 2-5 の 2 次元 Mesh および 2 次元 Torus 構造の場合，PE_{32}の位置は$p_x = 3$，$p_y = 2$と記述できる．

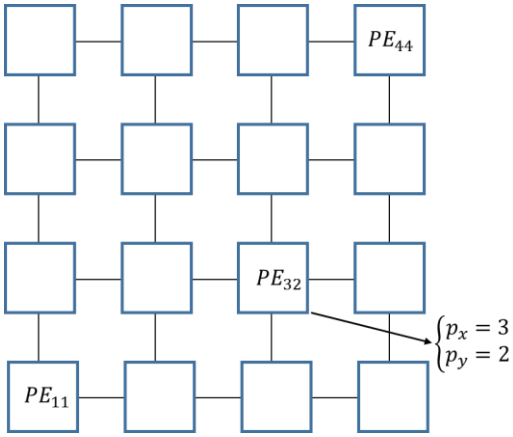


Fig. 2-5 Location expression of PE.

Ring，Torus，Hypercube 構造の場合は，基準とする PE からの通信距離の和に関して，基準とする PE のロケーションによる依存性がない．よって，式(2-2)の i を変化させて基準となる PE を変える必要がないため，式(2-2)は次式のように簡略化される．

$$\bar{d} = \frac{1}{p-1}\sum_{j=1}^{p} Dist_{1j} \tag{2-3}$$

2.2.1 Linear Array 構造

Linear Array の平均 PE 間距離を求めるために，初めに PE 間距離の総和を求める．Fig. 2-6 に PE 数p_xが 3 の場合の PE 間距離の例を示す．ここで，黒の矢印の長さがPE_1より他PEへの通信を行う場合の距離を示している．PE_1については合計 3，PE_2については合計 2，PE_3については合計 3,システム全体では，総計 3 の通信距離となる．これを PE 間の組み合わせ数の倍の数$p_x(p_x-1)$で除すと平均 PE 間距離$\bar{d}$を求めることができる．この場合は，$\bar{d}$は 4/3 となる．

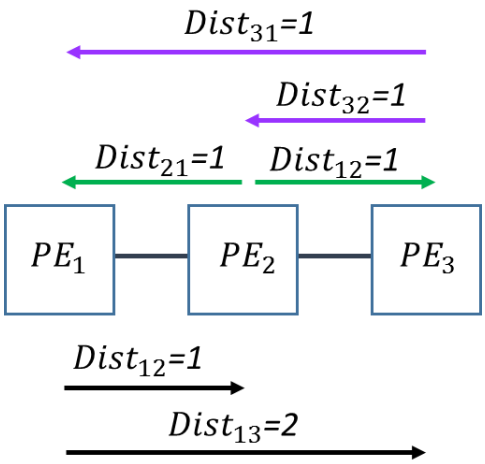


Fig. 2-6 Distance between PEs (Linear Array)

PE 間距離の総和 $TotLen$は式(2-4)であらわされる．

$$TotLen = \sum_{i=1}^{p_x}\left(\sum_{u=1}^{i-1}(i-u)+\sum_{u=i}^{p_x}(u-i)\right)=\frac{1}{3}p_x\left(p_x^2-1\right) \tag{2-4}$$

$$\bar{d}=\frac{1}{3}p_x\left(p_x^2-1\right)/\left(p_x\left(p_x-1\right)\right)=\frac{1}{3}\left(p_x+1\right) \tag{2-5}$$

よって，平均 PE 間距離 $\bar{d}$は，$\frac{1}{3}(p_x+1)$ となる．

2.2.2 2 次元 Mesh 構造

2 次元 Mesh 構造の平均 PE 間距離を求めるために，初めに PE 間距離の総和を求める．Fig. 2-7 に，x 方向に 3 台，y 方向に 2 台の PE より構成されるシステムの例を示す．Mesh 構造場合，距離はマンハッタン距離(市街区画距離)で定められる，各 PE に付けられた ID（PE 位置の座標成分）の差の絶対値の総和を 2 台の PE 間の距離と定義する，

Fig. 2-7 では，PE_{11}を基点とした場合のそれぞれの PE に対しての距離を示している 例えば，PE_{11}とPE_{32}の間の距離は 3 となる．

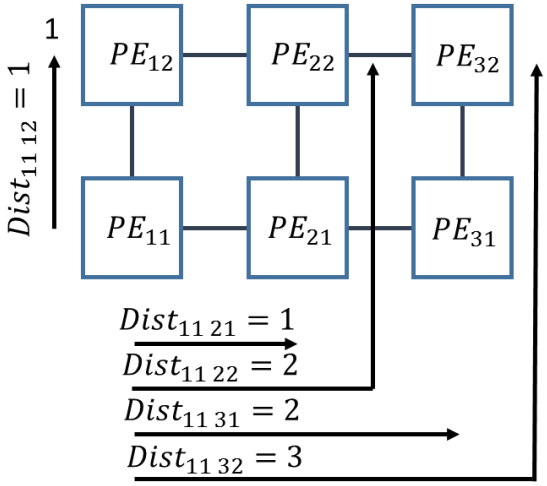


Fig. 2-7 Inter-PE-distance 2D Mesh example.

このような距離の集計を全ての PE について行えば，PE 間距離の総和が求められる．

PE_{ij}を基点としてPE_{uv}までの距離を考える．Fig. 2-8 にPE_{uv}の分割の方法を示す．領域を $\mathrm{A} = \{PE_{uv} | 1 \le \mathrm{u} < \mathrm{i}, 1 \le \mathrm{v} < \mathrm{j}\}$， $\mathrm{B} = \{PE_{uv} | \mathrm{i} \le \mathrm{u} \le \mathrm{p}_x, 1 \le \mathrm{v} < \mathrm{j}\}$， $\mathrm{C} = \{PE_{uv} | 1 \le \mathrm{u} < \mathrm{i}, \mathrm{j} \le \mathrm{v} \le \mathrm{p}_y\}$， $\mathrm{D} = \{PE_{uv} | \mathrm{i} \le \mathrm{u} \le \mathrm{p}_x, \mathrm{j} \le \mathrm{v} \le \mathrm{p}_y\}$のように 4 領域に分割しその合計を求める．

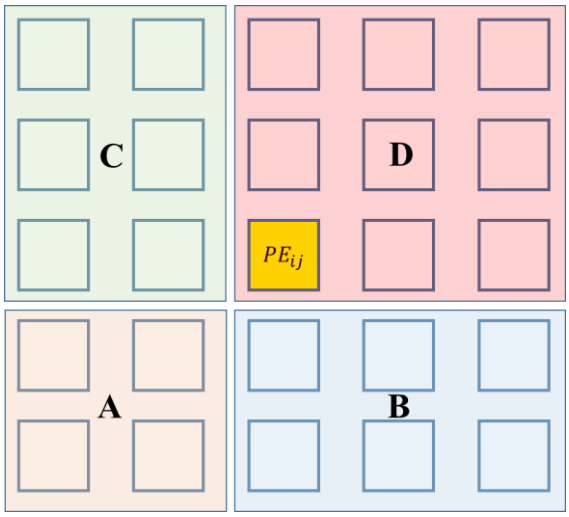


Fig. 2-8 Domain segmentation around PE_{ij}.

前記の内容を考慮すると，PE_{ij}より各 PE への距離の合計は式(2-6)のように表される．ここで，式(2-6)において，左の項より順に領域 A，領域 B，領域 C，領域 D に対応している．

$$\sum_{v=1}^{j-1}\left(\sum_{u=1}^{i-1}(i-u+j-v)+\sum_{u=i}^{p_x}(u-i+j-v)\right)+ \\ \sum_{v=j}^{p_y}\left(\sum_{u=1}^{i-1}(i-u+v-j)+\sum_{u=i}^{p_x}(u-i+v-j)\right) \tag{2-6}$$

式(2-6)より，i，jを全ての PE について合計すると全 PE についての合計距離 $TotLen$ が得られ，式(2-7)のように表される．

$$TotLen=\sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\left(\sum_{v=1}^{j-1}\left(\sum_{u=1}^{i-1}(i-u+j-v)+\sum_{u=i}^{p_x}(u-i+j-v)\right)\right)+ \\ \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\left(\sum_{v=j}^{p_y}\left(\sum_{u=1}^{i-1}(i-u+v-j)+\sum_{u=i}^{p_x}(u-i+v-j)\right)\right) \tag{2-7}$$

式(2-7)を変形すると式(2-8)となる．

$$\begin{aligned} TotLen &= \frac{1}{6}\left(p_xp_y-p_x^2p_y-p_x^3p_y-p_xp_y^2+p_x^3p_y^2+p_x^2p_y^3\right)+ \\ &\quad \frac{1}{6}\left(-p_xp_y-p_x^2p_y+p_x^3p_y-p_xp_y^2+p_x^3p_y^2+p_x^2p_y^3\right) \\ &= \frac{1}{3}p_xp_y\left(p_x+p_y\right)\left(p_xp_y-1\right) \end{aligned} \tag{2-8}$$

2 次元 Mesh 構造の場合，PE 数は$p=p_xp_y$となるため，式(2-2)より式(2-9)が求められ，平均 PE 間距離が得られる．

$$\bar{d}=\frac{TotLen}{p_xp_y\left(p_xp_y-1\right)}=\frac{1}{3}\left(p_x+p_y\right) \tag{2-9}$$

2.2.3 3 次元 Mesh 構造

3 次元 mesh 構造の PE 間距離の総和を求め，それを基に平均 PE 間距離を求めることを考える．PE_{ijk}を基点としてPE_{uvw}までの距離を考える．Fig. 2-8 を 3 次元に拡張し，PE_{uvw}の分割の方法を示す．領域は$\mathrm{AL} = \{PE_{uvw} | 1 \le \mathrm{u} < \mathrm{i}, 1 \le \mathrm{v} < \mathrm{j}, 1 \le \mathrm{w} < \mathrm{k}\}$，$\mathrm{BL} = \{PE_{uvw} | \mathrm{i} \le \mathrm{u} \le \mathrm{p}_x, 1 \le \mathrm{v} < \mathrm{j}, 1 \le \mathrm{w} < \mathrm{k}\}$，$\mathrm{CL} = \{PE_{uvw} | 1 \le \mathrm{u} < \mathrm{i}, \mathrm{j} \le \mathrm{v} \le \mathrm{p}_y, 1 \le \mathrm{w} < \mathrm{k}\}$，$\mathrm{DL} = \{PE_{uvw} | \mathrm{i} \le \mathrm{u} \le \mathrm{p}_x, \mathrm{j} \le \mathrm{v} \le \mathrm{p}_y, 1 \le \mathrm{w} < \mathrm{k}\}$，$\mathrm{AH} = \{PE_{uvw} | 1 \le \mathrm{u} < \mathrm{i}, 1 \le \mathrm{v} < \mathrm{j}, \mathrm{k} \le \mathrm{w} \le \mathrm{p}_z\}$，$\mathrm{BH} = \{PE_{uvw} | \mathrm{i} \le \mathrm{u} \le \mathrm{p}_x, 1 \le \mathrm{v} < \mathrm{j}, \mathrm{k} \le \mathrm{w} \le \mathrm{p}_z\}$，$\mathrm{CH} = \{PE_{uvw} | 1 \le \mathrm{u} < \mathrm{i}, \mathrm{j} \le \mathrm{v} \le \mathrm{p}_y, \mathrm{k} \le \mathrm{w} \le \mathrm{p}_z\}$，$\mathrm{DH} = \{PE_{uvw} | \mathrm{i} \le \mathrm{u} \le \mathrm{p}_x, \mathrm{j} \le \mathrm{v} \le \mathrm{p}_y, \mathrm{k} \le \mathrm{w} \le \mathrm{p}_z\}$のように$PE_{ijk} - PE_{uvw}$間のマンハッタン距離が正となるように 8 領域に分割しその合計を求める．

2 次元 Mesh 構造の場合に基づいて考えると，PE_{ijk}より各 PE への距離の合計は式(2-10)のように表される．ここで，式(2-10)において，左の項より順に領域 AL，領域 BL，領域 CL，領域 DL，領域 AH，領域 BH，領域 CH，領域 DH に対応している．

$$
\begin{aligned}
&\sum_{w=1}^{k-1}\left(\sum_{v=1}^{j-1}\left(\sum_{u=1}^{i-1}(i-u+j-v+k-w)+\sum_{u=i}^{p_x}(u-i+j-v+k-w)\right)\right)+\\
&\sum_{w=1}^{k-1}\left(\sum_{v=j}^{p_y}\left(\sum_{u=1}^{i-1}(i-u+v-j+k-w)+\sum_{u=i}^{p_x}(u-i+v-j+k-w)\right)\right)+\\
&\sum_{w=k}^{p_z}\left(\sum_{v=1}^{j-1}\left(\sum_{u=1}^{i-1}(i-u+j-v+w-k)+\sum_{u=i}^{p_x}(u-i+j-v+w-k)\right)\right)+\\
&\sum_{w=k}^{p_z}\left(\sum_{v=j}^{p_y}\left(\sum_{u=1}^{i-1}(i-u+v-j+w-k)+\sum_{u=i}^{p_x}(u-i+v-j+w-k)\right)\right)
\end{aligned}
\tag{2-10}
$$

式(2-11)を，全ての PE，すなわち i, j, k について総和を取ると全 PE についての合計距離が得られ，式のように表される．

$$
\begin{aligned}
TotLen = & \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=1}^{k-1}\left(\sum_{v=1}^{j-1}\left(\sum_{u=1}^{i-1}(i-u+j-v+k-w)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=1}^{k-1}\left(\sum_{v=1}^{j-1}\left(\sum_{u=i}^{p_x}(u-i+j-v+k-w)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=1}^{k-1}\left(\sum_{v=j}^{p_y}\left(\sum_{u=1}^{i-1}(i-u+v-j+k-w)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=1}^{k-1}\left(\sum_{v=j}^{p_y}\left(\sum_{u=i}^{p_x}(u-i+v-j+k-w)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=k}^{p_z}\left(\sum_{v=1}^{j-1}\left(\sum_{u=1}^{i-1}(i-u+j-v+w-k)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=k}^{p_z}\left(\sum_{v=1}^{j-1}\left(\sum_{u=i}^{p_x}(u-i+j-v+w-k)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=k}^{p_z}\left(\sum_{v=j}^{p_y}\left(\sum_{u=1}^{i-1}(i-u+v-j+w-k)\right)\right)\right)+ \\
& \sum_{i=1}^{p_x}\sum_{j=1}^{p_y}\sum_{k=1}^{p_z}\left(\sum_{w=k}^{p_z}\left(\sum_{v=j}^{p_y}\left(\sum_{u=i}^{p_x}(u-i+v-j+w-k)\right)\right)\right)
\end{aligned}
\tag{2-11}
$$

式(2-11)を変形すると式(2-12)となる.

$$
\begin{aligned}
TotLen = & \frac{1}{24}(-1+p_x)p_x(-1+p_y)p_y(-1+p_z)p_z(3+p_x+p_y+p_z)+ \\
& \frac{1}{24}p_x(1+p_x)(-1+p_y)p_y(-1+p_z)p_z(1+p_x+p_y+p_z)+ \\
& \frac{1}{24}(-1+p_x)p_xp_y(1+p_y)(-1+p_z)p_z(1+p_x+p_y+p_z)+ \\
& \frac{1}{24}p_x(1+p_x)p_y(1+p_y)(-1+p_z)p_z(-1+p_x+p_y+p_z)+ \\
& \frac{1}{24}(-1+p_x)p_x(-1+p_y)p_yp_z(1+p_z)(1+p_x+p_y+p_z)+ \\
& \frac{1}{24}p_x(1+p_x)(-1+p_y)p_yp_z(1+p_z)(-1+p_x+p_y+p_z)+ \\
& \frac{1}{24}(-1+p_x)p_xp_y(1+p_y)p_z(1+p_z)(-1+p_x+p_y+p_z)+ \\
& \frac{1}{24}p_x(1+p_x)p_y(1+p_y)p_z(1+p_z)(-3+p_x+p_y+p_z) \\
= & \frac{1}{3}p_x^2p_y^2p_z^2\left(p_x-\frac{1}{p_x}+p_y-\frac{1}{p_y}+p_z-\frac{1}{p_z}\right)
\end{aligned}
\tag{2-12}
$$

ここで，式(2-12)に$p_y = 1$，$p_z = 1$を代入すると，

$$
TotLen = \frac{1}{3}p_x\left(p_x^2-1\right) \tag{2-13}
$$

となり，式と一致する．また，式(2-12)に$p_z = 1$を代入すると，

$$
TotLen = \frac{1}{3}p_xp_y\left(p_x+p_y\right)\left(p_xp_y-1\right) \tag{2-14}
$$

となり，式(2-8)に一致する．これより式(2-12)の正当性を確認できる．

3 次元 Mesh 構造の場合，PE 数は$p = p_xp_yp_z$となるため，式(2-12)および式(2-2)より平均 PE 間距離が得られる．

$$
\begin{aligned}
\bar{d} &= \frac{1}{3}\frac{p_x^2p_y^2p_z^2}{p_xp_yp_z(p_xp_yp_z-1)}\left(p_x-\frac{1}{p_x}+p_y-\frac{1}{p_y}+p_z-\frac{1}{p_z}\right) \\
&= \frac{1}{3}\frac{p_xp_yp_z}{p_xp_yp_z-1}\left(p_x-\frac{1}{p_x}+p_y-\frac{1}{p_y}+p_z-\frac{1}{p_z}\right)
\end{aligned}
\tag{2-15}
$$

式(2-13)で，p_x，p_y，p_zが非常に大きな数の場合には，平均 PE 間距離は(2-16)のような単純な形に近似できる」．

$$\bar{d} \simeq \frac{p_x + p_y + p_z}{3} \tag{2-16}$$

2.2.4 Ring 構造

Ring 構造の平均 PE 間距離を求めるために， PE 間距離の総和を考える．Fig. 2-9 に，Ring 構造のシステムの例を示す．ここで（a）は PE 数 p が偶数の場合であり（b）は PE 数 p が奇数の場合である．

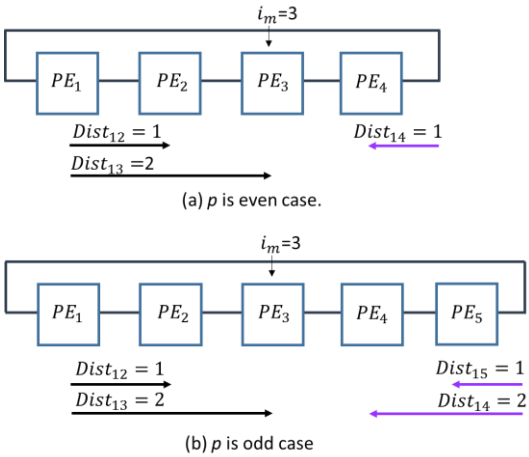


Fig. 2-9 Classification of Ring structure.

Fig. 2-9 では，PE_1より の距離を矢印の先に示した．ここで，PE_1より計測する場合は，反時計回り，時計回りで短い方を選択する．よって（a）(b）の両方の場合で，PE_1よりPE_3までは，PE_1より反時計回り方向の距離で計測し，PE_4より先の PE については，PE_1より時計回り方向の距離で計測する．

PE 間距離の総和は，(a)では 4，(b)では 6 となり，したがって，平均 PE 間距離は 4/3，3/2 となる．対称性よりどの PE を基準に選んでも PE 間距離の総和は同一な値となる．

PE_1より時計回り方向の距離で計測する一番離れた PE をPE_{i_m}と定義すると，PE 間距離の総和は式(2-17)で表される．

$$TotLen = \sum_{i=1}^{i_m}(i-1) + \sum_{i=i_m+1}^{p_x}(p_x - i + 1)$$
$$= \frac{1}{2}\left(2i_m^2 + p_x + p_x^2 - 2i_m(1+p_x)\right) \tag{2-17}$$

したがって，p_xが奇数の場合は，Fig. 2-9 より， $i_m = (p_x + 1)/2$となり，これを式(2-17)に代入すると，PE 間総通信距離が得られる．

$$TotLen = \frac{p_x^2 - 1}{4} \tag{2-18}$$

式(2-3)と式(2-17)より平均 PE 間距離が得られる．

$$\bar{d} = \frac{TotLen}{p_x - 1} = \frac{1}{4}(p_x + 1) \tag{2-19}$$

p_xが偶数の場合は，Fig. 2-9 より，$i_m = p_x/2 + 1$となり，これを式(2-17)に代入すると，PE 間総通信距離が得られる．

$$TotLen = \frac{p_x^2}{4} \tag{2-20}$$

式(2-3)と式(2-17)より平均 PE 間距離が得られる．

$$\bar{d} = \frac{p_x^2}{4(p_x - 1)} \tag{2-21}$$

2.2.5 2 次元 Torus 構造

2 次元 Torus 構造の平均 PE 間距離を求めるために，初めに PE 間距離の総和を求める．Fig. 2-10 に，PE_{11}を基点とした場合のそれぞれの PE への距離を PE id の下に示す．Fig.

2-10 では，PE 間の通信のためのリンクは割愛してある．このような距離を全ての PE について合計すると，PE_{11}を基点とした場合の PE 間距離の総和が求められる．

Ring 構造で用いたi_mを 2 次元に拡張し，j_mを導入する．Fig. 2-10 に示したように，PE_{11}を基点として y 軸方向に距離を計測する場合，PE_{21}，PE_{31}はPE_{11}より上方に向かって距離を計測するが，PE_{41}については，Torus 構造のためPE_{11}とPE_{41}を直接結合している Link を介してPE_{11}よりの距離を計測する必要がある．j_mはi_m同様，選択すべき計測アルゴリズムの境界点を示している．

PE_{11}を基点としてPE_{uv}までの距離の導出を考える．Fig. 2-10 にPE_{uv}の分割の方法を示す．領域は $\mathrm{A} = \{PE_{uv} | 1 \le \mathrm{u} \le i_m, 1 \le \mathrm{v} \le j_m\}$，$\mathrm{B} = \{PE_{uv} | i_{m+1} \le \mathrm{u} \le \mathrm{p}_x, 1 \le \mathrm{v} \le j_m\}$，$\mathrm{C} = \{PE_{uv} | 1 \le \mathrm{u} \le i_m, j_m \le \mathrm{v} \le \mathrm{p}_y\}$，$\mathrm{D} = \{PE_{uv} | i_m + 1 \le \mathrm{u} \le \mathrm{p}_x, j_m + 1 \le \mathrm{v} \le \mathrm{p}_y\}$のように 4 領域に分割しその合計を求める．

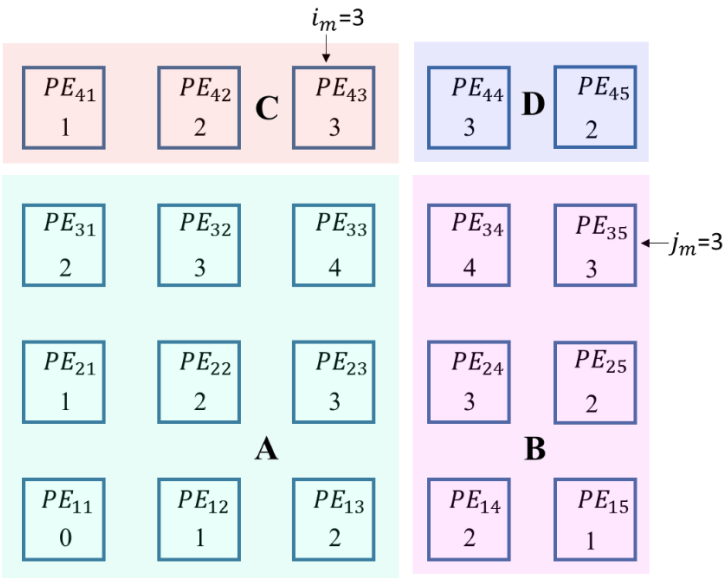


Fig. 2-10 Domain segmentation of 2D Torus.

距離計測アルゴリズムの切り換え点をi_mおよびj_mとすると，PE_{11}より各 PE への距離の合計は式(2-6)のように表される．ここで，式(2-6)において，左の項より順に領域 A，領域 B，領域 C，領域 D に対応している．

前記の領域分割より，PE 間距離の総和は，式(2-22)により求められる．

$$
\begin{aligned}
TotLen = & \sum_{j=1}^{j_m}\sum_{i=1}^{i_m}(i+j-2) \\
& + \sum_{j=1}^{j_m}\sum_{i=i_m+1}^{p_x}(p_x - i + 1 + j - 1) \\
& + \sum_{j=j_m+1}^{p_y}\sum_{i=1}^{i_m}(i - 1 + p_y - j + 1) \\
& + \sum_{j=j_m+1}^{p_y}\sum_{i=i_m+1}^{p_x}(p_x - i + 1 + p_y - j + 1)
\end{aligned} \tag{2-22}
$$

式(2-22)を具体的に計算し，整理すると式(2-23)になる．

$$
\begin{aligned}
TotLen = & \left(j_m^2 - j_m\right)p_x + \left(i_m^2 - i_m\right)p_y \\
& + \left(1 - i_m - j_m + \frac{1}{2}p_x + \frac{1}{2}p_y\right)p_x p_y
\end{aligned} \tag{2-23}
$$

PE 間距離の総和を求める式は，p_xおよびp_yが奇数および偶数であるかによって，下記に示す 4 種のパターンに分類される．

1) p_xが奇数，p_yが奇数の場合

この場合，$i_m =$ （$p_x + 1)/2$，$j_m = (p_y + 1)/2$となる．これらを式(2-23)に代入すると PE 間距離の総和として次式を得る．

$$
TotLen = \frac{1}{4}\left(p_x + p_y\right)\left(p_x p_y - 1\right) \tag{2-24}
$$

式(2-24)より，平均 PE 間距離は，式のようになる．

$$
\bar{d} = \frac{1}{4}\left(p_x + p_y\right) \tag{2-25}
$$

2) p_xが偶数，p_yが偶数の場合

この場合，$i_m = p_x/2 + 1$，$j_m = p_y/2 + 1$となる．これらを式(2-23)に代入すると PE 間距離の総和として次式を得る．

$$
TotLen = \frac{1}{4}p_x p_y\left(p_x + p_y\right) \tag{2-26}
$$

式(2-2)より，平均 PE 間距離は，式(2-27)のようになる．

$$\bar{d} = \frac{p_x + p_y}{4} \frac{p_x p_y}{p_x p_y - 1} \tag{2-27}$$

3) p_xが偶数，p_yが奇数の場合

この場合，$i_m = p_x/2 + 1$，$j_m = (p_y + 1)/2$となる．これらを式(2-23)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y \left(p_x + p_y\right) - p_x\right) \tag{2-28}$$

式(2-28)より，平均 PE 間距離は，式(2-29)のようになる．

$$\bar{d} = \frac{p_x p_y \left(p_x + p_y\right) - p_x}{4(p_x p_y - 1)} \tag{2-29}$$

4) p_xが奇数，p_yが偶数の場合

この場合，$i_m = (p_x + 1)/2$，$j_m = p_y/2 + 1$となる．これらを式(2-23)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y \left(p_x + p_y\right) - p_y\right) \tag{2-30}$$

式(2-62)より，平均 PE 間距離は，式(2-31)のようになる．

$$\bar{d} = \frac{p_x p_y \left(p_x + p_y\right) - p_y}{4(p_x p_y - 1)} \tag{2-31}$$

2.2.6 3 次元 Torus 構造

3 次元 Torus 構造の平均 PE 間距離を求めるために，初めに PE 間距離の総和を求める．Fig. 2-10 に示した関係を 3 次元に拡張することを考える．

PE_{111}を基点としてPE_{uvw}までの距離の導出を考える．距離計測アルゴリズムの z 軸方向の切り換え座標k_mを新たに追加して，Fig. 2-10 に示した 2 次元PE_{uv}の分割の方法を 3 次元に拡張すると，領域は，$\mathrm{AL} = \{PE_{uvw} | 1 \le \mathrm{u} \le i_m, 1 \le \mathrm{v} \le j_{m,}, 1 \le \mathrm{w} \le k_m\}$，$\mathrm{BL} = \{PE_{uv} | i_m + 1 \le \mathrm{u} \le \mathrm{p}_x, 1 \le \mathrm{v} \le j_m, 1 \le \mathrm{w} \le k_m\}$，$\mathrm{CL} = \{PE_{uv} | 1 \le \mathrm{u} \le i_m, j_m + 1 \le \mathrm{v} \le \mathrm{p}_y, 1 \le \mathrm{w} \le k_m\}$，$\mathrm{DL} = \{PE_{uv} | i_m + 1 \le \mathrm{u} \le \mathrm{p}_x, j_m + 1 \le \mathrm{v} \le \mathrm{p}_y, 1 \le \mathrm{w} \le k_m\}$，$\mathrm{AH} = \{PE_{uvw} | 1 \le \mathrm{u} \le i_m, 1 \le$

$\mathrm{v} \le j_{m,}, \mathrm{p}_z + 1 \le \mathrm{w} \le k_m\}$, $\mathrm{BH} = \{PE_{uv} | i_m + 1 \le \mathrm{u} \le \mathrm{p}_x, 1 \le \mathrm{v} \le j_m, \mathrm{p}_z + 1 \le \mathrm{w} \le k_m\}$, $\mathrm{CH} = \{PE_{uv} | 1 \le \mathrm{u} \le i_m, j_m + 1 \le \mathrm{v} \le \mathrm{p}_y \mathrm{p}_z + 1 \le \mathrm{w} \le k_m\}$, $\mathrm{DH} = \{PE_{uv} | i_m + 1 \le \mathrm{u} \le \mathrm{p}_x, j_m + 1 \le \mathrm{v} \le \mathrm{p}_y, \mathrm{p}_z + 1 \le \mathrm{w} \le k_m\}$のように 8 領域に分割しその合計を求める．

前記の領域分割より，点$(i_m,\ j_m,\ k_m)$を分割点とした場合の PE 間距離の総和は，式(2-22)により求められる．

$$
\begin{aligned}
TotLen = & \sum_{k=1}^{k_m}\sum_{j=1}^{j_m}\sum_{i=1}^{i_m}(i+j+k-3) \\
& + \sum_{k=k_m+1}^{p_z}\sum_{j=1}^{j_m}\sum_{i=1}^{i_m}(i-1+j-1+p_z-k+1) \\
& + \sum_{k=1}^{k_m}\sum_{j=j_m+1}^{p_y}\sum_{i=1}^{i_m}(i-1+p_y-j+1+k-1) \\
& + \sum_{k=k_m+1}^{p_z}\sum_{j=j_m+1}^{p_y}\sum_{i=1}^{i_m}(i-1+p_y-j+1+p_z-k+1) \\
& + \sum_{k=1}^{k_m}\sum_{j=1}^{j_m}\sum_{i=i_m+1}^{p_x}(p_x-i+j+k-1) \\
& + \sum_{k=k_m+1}^{p_z}\sum_{j=1}^{j_m}\sum_{i=i_m+1}^{p_x}(p_x-i+j+p_z-k+1) \\
& + \sum_{k=1}^{k_m}\sum_{j=j_m+1}^{p_y}\sum_{i=i_m+1}^{p_x}(p_x-i+1+p_y-j+1+k-1) \\
& + \sum_{k=k_m+1}^{p_z}\sum_{j=j_m+1}^{p_y}\sum_{i=i_m+1}^{p_x}(p_x-i+1+p_y-j+1+p_z-k+1)
\end{aligned}
\tag{2-32}
$$

式(2-32)を具体的に計算し，整理すると式(2-33)になる．

$$
\begin{aligned}
TotLen = & (k_m^2-k_m)p_xp_y + (i_m^2-i_m)p_yp_z + (j_m^2-j_m)p_zp_x \\
& + \frac{1}{2}(p_x+p_y+p_z-2i_m-2j_m-2k_m-3)p_xp_yp_z
\end{aligned}
\tag{2-33}
$$

PE 間距離の総和は，p_x，p_yおよびp_zの奇偶によって，下記に示す 8 種のパターンに分類される．

1) p_xが奇数，p_yが奇数，p_zが奇数の場合

この場合，$i_m =$ （$p_x + 1)/2$，$j_m = (p_y + 1)/2$，$k_m = (p_z + 1)/2$となる．これ

らを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y - p_y p_z - p_z p_y\right) \tag{2-34}$$

式(2-34)より，平均 PE 間距離は，式のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y - p_y p_z - p_z p_x}{4\left(p_x p_y p_z - 1\right)} \tag{2-35}$$

2) p_xが偶数，p_yが偶数，p_zが偶数の場合

この場合，$i_m = p_x/2 + 1$，$j_m = p_y/2 + 1$，$k_m = p_z/2 + 1$となる．これらを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4} p_x p_y p_z \left(p_x + p_y + p_z\right) \tag{2-36}$$

式(2-36)より，平均 PE 間距離は，式(2-37)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right)}{4\left(p_x p_y p_z - 1\right)} \tag{2-37}$$

3) p_xが偶数，p_yが偶数，p_zが奇数の場合

この場合，$i_m = p_x/2 + 1$，$j_m = p_y/2 + 1$，$k_m = (p_z + 1)/2$となる．これらを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y\right) \tag{2-38}$$

式(2-38)より，平均 PE 間距離は，式(2-39)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y}{4\left(p_x p_y p_z - 1\right)} \tag{2-39}$$

4) p_xが偶数，p_yが奇数，p_zが偶数の場合

この場合，$i_m = p_x/2 + 1$，$j_m = (p_y + 1)/2$，$k_m = p_z/2 + 1$となる．これらを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y\right) \tag{2-40}$$

式(2-40)より，平均 PE 間距離は，式(2-41)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y}{4\left(p_x p_y p_z - 1\right)} \tag{2-41}$$

5) p_xが偶数，p_yが奇数，p_zが奇数の場合

この場合，$i_m = p_x/2 + 1$，$j_m = (p_y + 1)/2$，$k_m = (p_z + 1)/2$となる．これらを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_z p_x\right) \tag{2-42}$$

式(2-42)より，平均 PE 間距離は，式(2-43)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_z p_x}{4\left(p_x p_y p_z - 1\right)} \tag{2-43}$$

6) p_xが奇数，p_yが偶数，p_zが偶数の場合

この場合，$i_m =$ （$p_x + 1)/2$，$j_m = p_y/2 + 1$，$k_m = p_z/2 + 1$となる．これらを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_y p_z\right) \tag{2-44}$$

式(2-44)より，平均 PE 間距離は，式(2-45)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_y p_z}{4\left(p_x p_y p_z - 1\right)} \tag{2-45}$$

7) p_xが奇数，p_yが偶数，p_zが奇数の場合

この場合，$i_m =$ （$p_x + 1)/2$，$j_m = p_y/2 + 1$，$k_m = (p_z + 1)/2$となる．これら

を式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y - p_y p_z\right) \tag{2-46}$$

式(2-46)より，平均 PE 間距離は，式(2-47)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_x p_y - p_y p_z}{4\left(p_x p_y p_z - 1\right)} \tag{2-47}$$

8) p_xが奇数，p_yが奇数，p_zが偶数の場合

この場合，$i_m = (p_x + 1)/2$，$j_m = (p_y + 1)/2$，$k_m = p_z/2 + 1$となる．これらを式(2-33)に代入すると PE 間距離の総和として次式を得る．

$$TotLen = \frac{1}{4}\left(p_x p_y p_z \left(p_x + p_y + p_z\right) - p_y p_z - p_z p_x\right) \tag{2-48}$$

式(2-48)(2-46)より，平均 PE 間距離は，式(2-61)のようになる．

$$\bar{d} = \frac{p_x p_y p_z \left(p_x + p_y + p_z\right) - p_y p_z - p_z p_x}{4\left(p_x p_y p_z - 1\right)} \tag{2-49}$$

2.2.7 Hypercube 構造

Hypercube 構造の平均 PE 間距離については，以前にも求められているが，一般的に知られている方法と異なる簡便な方法を導出できたため，ここで述べる．Fig. 2-11 に示すように，Hypercube 構造の各頂点に位置する PE は，id が 1bit だけ異なる PE と結合している．例えば，4 次元の Hypercube の場合，Fig. 2-11 より，PE id が 0000 は，{0001，0010，0100，1000} と結合している．これは，0000 より距離が 1 の PE を列挙したものと等価である．また，0000 より距離が 2 離れた PE は，2bit 異なる {0011，0101，1001，0110，1010，1100} の 6 台となる．Fig. 2-11 の場合，id の bit 数は 4，基準 PE からよりの距離，即ち，自らの id と異なる bit 数は 2 であるため，該当する PE 数は$\binom{4}{2}$=6 として求めることができる．したがって，一般に，d 次元の hypercube において，0000 より距離 k 離れた

PE の数は，$\binom{d}{k}$で求めることができる．したがって，距離 k にある PE 数を求め，式 (2-50) に示すように，k と PE 数の積の和を求めることによって PE 間総距離数が求められる．

$$TotLen = \sum_{k=1}^{d} \binom{d}{k} k \tag{2-50}$$

式(2-50)の右辺を求めるために，二項定理を考える．

$$(1+x)^d = \sum_{k=0}^{d} \binom{d}{k} x^k \tag{2-51}$$

式(2-51)の両辺を微分し，左辺と右辺を入れ替えると，式(2-52)を得る．

$$\sum_{k=0}^{d} \binom{d}{k} k\, x^{k-1} = d(1+x)^{d-1} \tag{2-52}$$

式(2-58)の両辺に$x = 1$を代入すると式(2-59)を得る．

$$\sum_{k=1}^{d} \binom{d}{k} k = d\, 2^{d-1} \tag{2-53}$$

よって，$p = 2^d$と式(2-56)より，PE 間総距離数が式(2-54)のように求められる．

$$TotLen = d\, 2^{d-1} = \frac{p}{2} \log_2(p) \tag{2-54}$$

式(2-3)より，平均 PE 間距離は，次式のように求められる．

$$\bar{d} = \frac{p \log_2(p)}{2(p-1)} \tag{2-55}$$

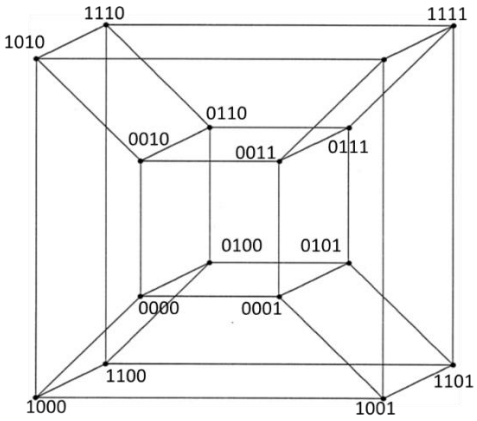


Fig. 2-11 4th Demensional Hypercube Structure.

2.2.8インターコネクトトポロジーの特性比較

前節で得られた結果を，Table 2-1，Table 2-2，Table 2-3 に示す．Table 2-1 に，1D の topology である Linear Array 構造と Ring 構造の結果を示す．Linear Array 構造と Ring 構造ともに PE 数p_xの偶奇による差はない．Ring 構造の場合，$p_x/2$より長い距離の PE との通信に関しては，反対側の Link を経由して通信を行う方が短距離であるため平均 PE 間距離が Linear Array 構造より小さく抑えられる結果となった．Ring 構造は実際に IBM/Sony Cell/B.E. [67]および Intel Xeon Phi の LSI Chip 内の PE(core)間インターコネクトに用いられている．

Table 2-1 Average distance of 1D topologies.

Topology	Average distance $\bar{d}$	$\bar{d}$ in large p area
Linear Array	$\frac{1}{3}(p_x+1)$	$\frac{p_x}{3}$

Ring	$\frac{p_x^2}{4(p_x-1)}$	$\frac{p_x}{4}$

Table 2-2 Average distance of 2D toplogies.

Topology	p_x	p_y	Average distance $\bar{d}$	$\bar{d}$ in large p area
2D Mesh			$\frac{1}{3}(p_x+p_y)$	$\frac{1}{3}(p_x+p_y)$
2D Torus	odd	odd	$\frac{1}{4}(p_x+p_y)$	$\frac{1}{4}(p_x+p_y)$
	even	even	$\frac{p_x+p_y}{4}\frac{p_xp_y}{p_xp_y-1}$	$\frac{1}{4}(p_x+p_y)$
	even	odd	$\frac{p_xp_y(p_x+p_y)-p_x}{4(p_xp_y-1)}$	$\frac{1}{4}(p_x+p_y)$
	odd	even	$\frac{p_xp_y(p_x+p_y)-p_y}{4(p_xp_y-1)}$	$\frac{1}{4}(p_x+p_y)$

Table 2-3 Average distance of 3D toplogies.

Topology	p_x	p_y	p_z	Average distance $\bar{d}$	$\bar{d}$ in large p area
3D Mesh				$\frac{1}{3}\frac{p_xp_yp_z}{p_xp_yp_z-1}\left(p_x-\frac{1}{p_x}+p_y-\frac{1}{p_y}+p_z-\frac{1}{p_z}\right)$	$\frac{1}{3}(p_x+p_y+p_z)$
3D Torus	odd	odd	odd	$\frac{p_xp_yp_z(p_x+p_y+p_z)-p_xp_y-p_yp_z-p_zp_x}{4(p_xp_yp_z-1)}$	$\frac{1}{4}(p_x+p_y+p_z)$
	even	even	even	$\frac{p_xp_yp_z(p_x+p_y+p_z)}{4(p_xp_yp_z-1)}$	$\frac{1}{4}(p_x+p_y+p_z)$

	even	even	odd	$\frac{p_x p_y p_z (p_x + p_y + p_z) - p_x p_y}{4(p_x p_y p_z - 1)}$	$\frac{1}{4}(p_x + p_y + p_z)$
	even	odd	even	$\frac{p_x p_y p_z (p_x + p_y + p_z) - p_x p_z}{4(p_x p_y p_z - 1)}$	$\frac{1}{4}(p_x + p_y + p_z)$
	odd	even	even	$\frac{p_x p_y p_z (p_x + p_y + p_z) - p_y p_z}{4(p_x p_y p_z - 1)}$	$\frac{1}{4}(p_x + p_y + p_z)$
	even	odd	odd	$\frac{p_x p_y p_z (p_x + p_y + p_z) - p_x p_y - p_z p_x}{4(p_x p_y p_z - 1)}$	$\frac{1}{4}(p_x + p_y + p_z)$
	odd	even	odd	$\frac{p_x p_y p_z (p_x + p_y + p_z) - p_x p_y - p_y p_z}{4(p_x p_y p_z - 1)}$	$\frac{1}{4}(p_x + p_y + p_z)$
	odd	odd	even	$\frac{p_x p_y p_z (p_x + p_y + p_z) - p_y p_z - p_z p_x}{4(p_x p_y p_z - 1)}$	$\frac{1}{4}(p_x + p_y + p_z)$

Table 2-2 に 2D Mesh 構造および 2D Torus 構造の平均 PE 間距離を示す．Table 2-2 では，横の PE 数をp_xとし，縦の PE 数をp_yとした場合（以降この構造を$p_x \times p_y$と表記する）の平均 PE 間距離を示している．Table 2-2 より， 2D Mesh で$p_x \times p_y$構造の場合は，p_xとp_yを交換しても常に同じ平均 PE 間距離となることがわかる．また，Table 2-2 より，2D Torus で$p_x \times p_y$構造の場合は，偶数×偶数構造もしくは奇数×奇数構造の場合は，p_xとp_yを交換した際平均 PE 間距離が等しくなるという性質を持つことが本研究によって新たに明らかになった．2D Torus 構造の平均 PE 間距離については，$p_x \times p_y$構造が奇数×奇数，偶数×偶数，偶数×奇数，奇数×偶数が考えられるが，Table 2-2 より平均 PE 間距離の値は，$p_x + p_y$の値によって支配的に決定されることがわかる．

Fig. 2-12 では，PE トポロジーが$p_x \times p_x$の正方形の 2D Mesh 構造に関して，p_xを連続する自然数として変化させた場合の平均 PE 間距離を紫色の線としてプロットした．$p_x \times p_x$構造の 2D Torus 構造に関しては，p_xを連続する奇数として変化させた結果（黄土色），および，p_xを連続する偶数として変化させた結果（青色）を示した．この場合，Table 2-2 よ

り，2D Torus 構造でp_xを連続する奇数として変化させた結果（黄土色）が 2D Mesh 構造に比べて平均 PE 間距離が一様に$3/4$となることがわかる．また，Fig. 2-12 中に示したように，$p_x \times p_x$構造であり且つp_xが偶数の 2D Torus 構造の結果（青色）については，PE 数（この場合，$p_x \times p_x$）が 100 を超えるあたりで，$p_x \times p_x$構造であり且つp_xが奇数の 2D Torus 構造の結果（黄土色）に急速に漸近することがわかる．Fig. 2-12 中に例示した，$p_x \times (p_x + 6)$および$p_x \times (p_x + 5)$で，p_xを連続する自然数として変化させた場合の 2D Torus 構造のケースでは，横と縦のサイズが異なるため奇数×奇数，もしくは，偶数×偶数の両方の場合が存在する．非対称性の崩れの大きさによって，絶えず$p_x \times (p_x + 5)$のケースが$p_x \times (p_x + 6)$のケースより平均 PE 間距離が小さく，また，両者とも PE 数 15 以下では，2D Mesh 構造より平均 PE 間距離が大きくなることがわかる．このように，本研究によって，縦と横のサイズが非対称な 2D Torus 構造の平均 PE 間距離を正確に求めることが可能となった．本結果は，ＬＳＩチップ内に面積の制約があるインターコネクトスイッチを構成する場合の事前評価に活用できる．2 次元のチップ内インターコネクトに関しては，Intel Knights Landing [8]および PEZY-SC2 [45]などの計画が知られている．

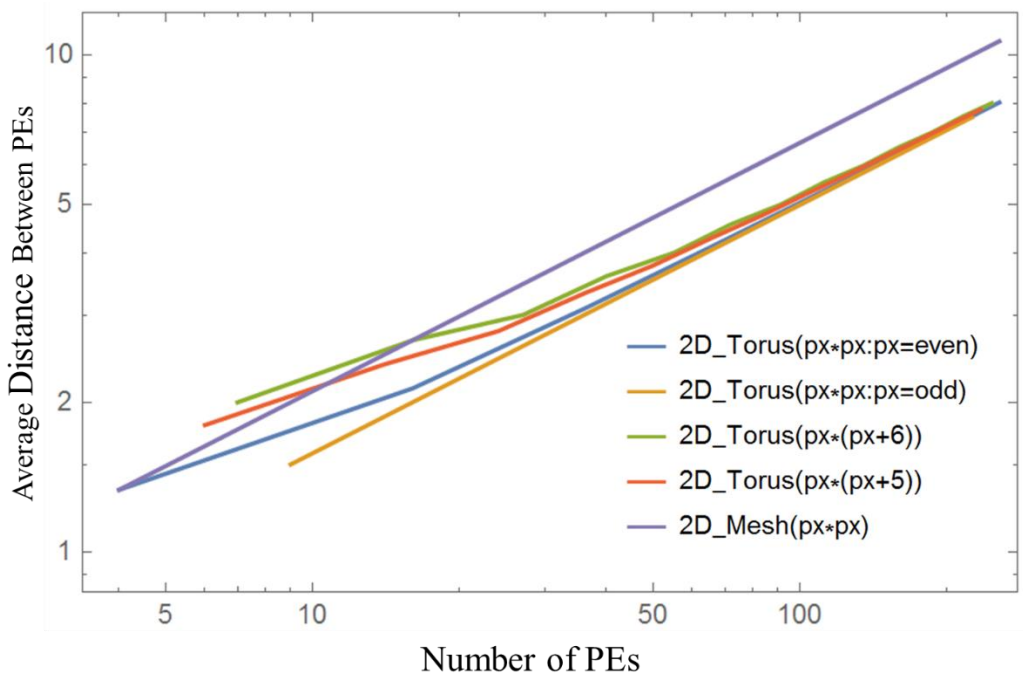


Fig. 2-12 Average Distance of 2D Topology cases

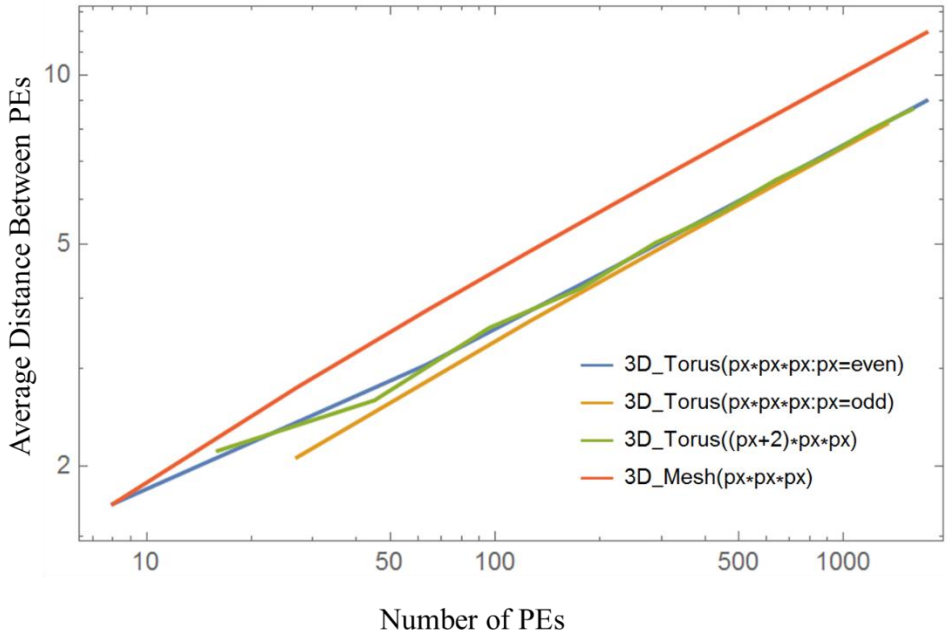


Fig. 2-13 Average Distance of 3D Topology cases

Fig. 2-13 に 3D Mesh 構造および 3D Torus 構造の平均 PE 間距離を示す．Fig. 2-13 は，横の PE 数をp_xとし，縦の PE 数をp_y，高さの PE 数をp_zの関数とした場合（以降これを$p_x \times p_y \times p_z$と表記する）の平均 PE 間距離を示している．また，Table 2-3 より，3D Torus で$p_x \times p_y \times p_z$構造の場合は，偶数×偶数×偶数構造の場合のみ，$p_x$，$p_y$，$p_z$を交換した際に平均 PE 間距離が等しくなるという性質を持ことが, 本研究によって新たに明らかになった．3D Torus 構造の平均 PE 間距離については，$p_x \times p_y \times p_z$構造がどのような遇奇の構造であっても，Table 2-3 より$p_x + p_y + p_z$の値によって支配的に決定されることがわかる．

Fig. 2-13 では，PE 構造が$p_x \times p_x \times p_x$構造の正方形の 3D Mesh 構造に関して，$p_x$を連続する自然数として変化させた場合の平均 PE 間距離をオレンジ色の線としてプロットした．$p_x \times p_x \times p_x$構造の 3D Torus 構造に関しては，$p_x$を連続する奇数として変化させた結果（黄土色），および，p_xを連続する偶数として変化させた結果（青色）を示した．この場合，Table 2-3 より，3D Torus 構造でp_xを連続する奇数として変化させた結果（黄土色）が 3D Mesh 構造（オレンジ色）に比べて平均 PE 間距離が一様に$3/4$となることがわかる．また，Fig. 2-13 中に示したように，$p_x \times p_x \times p_x$構造であり且つ$p_x$が偶数の 3D Torus 構造の結果

（青色）については，PE 数が小さな領域で，$p_x \times p_x \times p_x$構造であり且つ$p_x$が奇数の 3D Torus 構造の結果（黄土色）より差が広がることがわかる．Fig. 2-13 中に例示した$(p_x + 1) \times p_x \times p_x$構造は，$p_x$を連続する自然数として変化させた場合の 3D Torus 構造のケースでは，偶数×奇数×奇数構造と奇数×偶数×偶数構造のパターンが交互に現れるため，結果を直線で結ぶと滑らかではなくなった．現在，Samsung，Intel などにおいて 3 次元ＬＳＩの開発が進められている．当初は，メモリー分野への適用に限られるが，今後はＣＰＵ分野への拡張が考えられる．本結果は，ＬＳＩチップ内に 3 次元インターコネクトスイッチを構成する場合の事前評価に活用できる．

本章で既出の全てのインターコネクト構造について，2 次元の場合，縦と横，3 次元の場合，縦，横，高さが等しい系で評価を行う．Table 2-2 および Table 2-3 より，2D Torus 構造では，奇数×奇数構造が，3D Torus 構造では，奇数×奇数×奇数構造が平均 PE 間距離を最小とすることがわかり，これらに Linear Array, Ring, 2D Mesh, 3D Mesh, Hypercube 構造の結果を加えて，Fig. 2-14 に PE 数と平均 PE 間距離の関係を示す．Linear Array, Ring 構造は，他のインターコネクト構造に比べて，平均 PE 間距離が非常に大きいことがわかる．PE 数が大きな領域では，2D Mesh 構造，2D Torus 構造，3D Mesh 構造，3D Torus 構造，Hypercube 構造の順に平均 PE 間距離が小さくなり，PE 数が小さな領域ではこの関係が崩れる．PE 数が Fig. 2-14 の B 点を超えない領域では，3D Mesh 構造より 2D Torus 構造の方が平均 PE 間距離が小さいことがわかる．Hypercube 構造に関しては，PE 数が C 点より小さな領域では，3D Torus 構造より平均 PE 間距離が小さくなり，さらに A 点よりも小さな領域では，2D Torus 構造よりも平均 PE 間距離が小さくなる．このように，PE 数が C 点より小さな領域では，平均 PE 間距離の関係が複雑になることが，本研究によって明らかになり，CPU 内インターコネクトを決定する上で注意を払う必要があることが明らかになった．

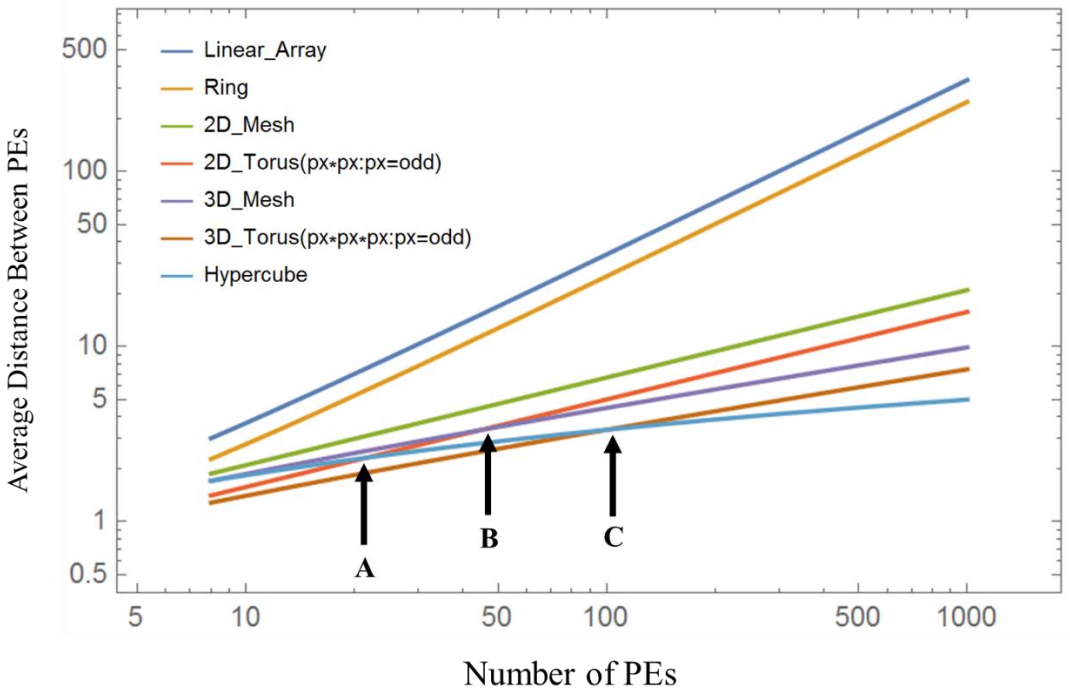


Fig. 2-14 Comparison of Average Distance

2.3 対象とする問題の定義

Poisson 方程式は，電磁気学，流体力学をはじめとする多くの分野で用いられている基本的な方程式で，実際に産業においても多くの分野で利用されている．したがって，本章では，下記の楕円型の 3 次元 Poisson 方程式で記述される場を解析の対象として選択した．

$$D(\boldsymbol{x})\nabla^2\phi(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in R^3 \tag{2-56}$$

ここで，$D(x), \phi(x)$，$f(x)$は実数であるとする．Fig. 2-15 に示すように，場の形状は直方体とし，第一種境界条件$(\partial\Omega_1)$，第二種境界条件$(\partial\Omega_2)$，第三種境界条件$(\partial\Omega_3)$を任意の形状で適応できるものとする．

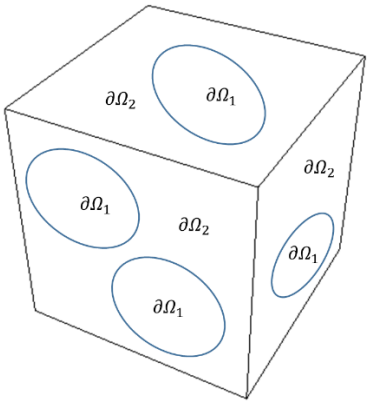


Fig. 2-15 Domain definition

定義された領域を計算機上で数値的に解く場合は，離散化を必要とする．本章では，生成される行列が規則的スパース行列となるため，アルゴリズムの評価がより容易となる FDM(Finite Difference Method）を離散化の方法として考える．

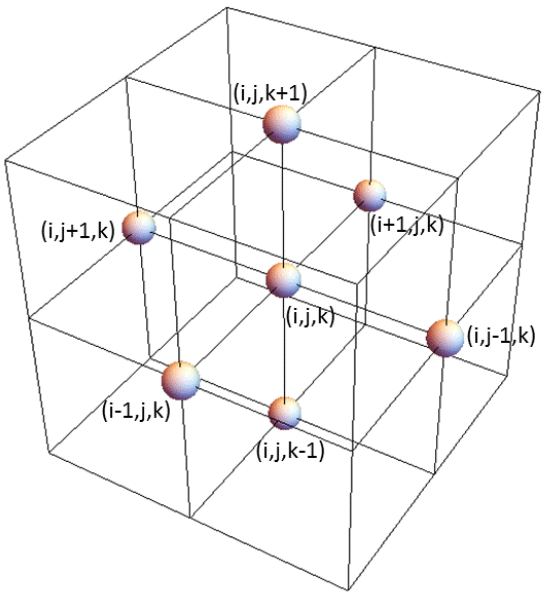


Fig. 2-16　Three dimensional stencil of FDM

3 次元空間を Fig. 2-16 のような格子に区切り，各格子点の座標を $(x_{i,}\, y_j,\, z_k)$ とする．格

子間隔は x 方向 がΔx, y 方向がΔy, z 方向がΔz とする．$x_{i\pm1} = x_i \pm \Delta x$, $y_{j\pm1} = y_j \pm \Delta y$, $z_{k\pm1} = z_k \pm \Delta z$である．以下，格子点番号 (i,j,k)を用いて，$u_{i,j,k} = u(x_i, y_j, z_k)$のように略記する． 着目している点$(x_{i,}\, y_j,\, z_k)$ のまわりでテイラー展開すると次式のようになる．

$$u_{i+1,j,k} = u(x_i + \Delta x, y_j, z_j) = u_{i,j,k} + \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2!}\left(\frac{\partial^2 u}{\partial^2 x}\right)_i + \frac{\Delta x^3}{3!}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i + \cdots \tag{2-57}$$

$$u_{i-1,j,k} = u(x_i - \Delta x, y_j, z_j) = u_{i,j,k} - \Delta x\left(\frac{\partial u}{\partial x}\right)_i + \frac{\Delta x^2}{2!}\left(\frac{\partial^2 u}{\partial^2 x}\right)_i + \frac{\Delta x^3}{3!}\left(\frac{\partial^3 u}{\partial^3 x}\right)_i + \cdots \tag{2-58}$$

式(2-57)と式(2-58)を辺々加えると，

$$u_{i-1,j,k} + u_{i+1,j,k} = 2u_{i,j,k} + \Delta x^2\left(\frac{\partial^2 u}{\partial^2 x}\right)_i + O(\Delta x^4) \tag{2-59}$$

となる．したがって，

$$\left(\frac{\partial^2 u}{\partial^2 x}\right)_i = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2} + O(\Delta x^2) \tag{2-60}$$

となる．これより，u の x に関する ニ階微分の係数$(\partial^2 u/\partial^2 x)_i$ を Δx について 2 次の精度で以下のように近似することができる．

$$\left(\frac{\partial^2 u}{\partial^2 x}\right)_i = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2} \tag{2-61}$$

y 方向，z 方向についても同様に次式のように表すことができる．

$$\left(\frac{\partial^2 u}{\partial^2 y}\right)_j = \frac{u_{u,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{\Delta y^2} \tag{2-62}$$

$$\left(\frac{\partial^2 u}{\partial^2 z}\right)_k = \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{\Delta z^2} \tag{2-63}$$

よって，式(2-56)を FDM により離散化した場合の表現式が，式(2-61)，式(2-62)，式(2-63)より次式として得られる．

$$
\begin{aligned}
D\nabla^2 u &= D\left(\frac{\partial^2 u}{\partial^2 x}\right)_i + D\left(\frac{\partial^2 u}{\partial^2 y}\right)_j + D\left(\frac{\partial^2 u}{\partial^2 z}\right)_k \\
&= D\frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2} + D\frac{u_{u,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{\Delta y^2} \\
&\quad + D\frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{\Delta z^2} \\
&= f_{i,j,k}
\end{aligned}
\tag{2-64}
$$

式(2-64)に現れる格子点(i,j,k)の周りの格子点は(i,j,k)を含めて 7 個である．これは 7 点差分であることを示している．この様子を Fig. 2-16 に示す．Fig. 2-16 中での球体が対応する格子点を示している．また，格子点(i,j,k)周りの関係を表す式(2-64)の演算をステンシル演算という．このステンシル演算は，Fig. 2-17 に示すように対称な疎行列と 7 個の要素を持つベクトル u の積とみなせる．

$$
\begin{bmatrix} \frac{D}{\Delta z^2} & \frac{D}{\Delta y^2} & \frac{D}{\Delta x^2} & -2\left(\frac{D}{\Delta x^2}+\frac{D}{\Delta y^2}+\frac{D}{\Delta z^2}\right) & \frac{D}{\Delta x^2} & \frac{D}{\Delta y^2} & \frac{D}{\Delta z^2} \end{bmatrix}
\begin{bmatrix} u_{i,j,k-1} \\ u_{i,j-1,k} \\ u_{i-1,j,k} \\ u_{i,j,k} \\ u_{i+1,j,k} \\ u_{u,j+1,k} \\ u_{i,j,k+1} \end{bmatrix} = f_{i,j,k}
$$

Fig. 2-17 **Stencil of 3D FDM.**

Fig. 2-18 に FDM で離散化した場合の格子点の結合状態を示す．これは，Fig. 2-16 に示した格子点(i,j,k)を中心とする部分的な格子を x 方向にn_x，y 方向にn_y，z 方向ににn_z，格子点数を拡張したたものと等価である．Fig. 2-18 を FDM で離散化した場合に生成される行列を Fig. 2-19 に示す．同一行で考えると，対角項を中心として，左右に 1，n_x，$n_x n_y$，離れた場所に 6 個の行列要素が存在する規則的対称スパース行列となっている．

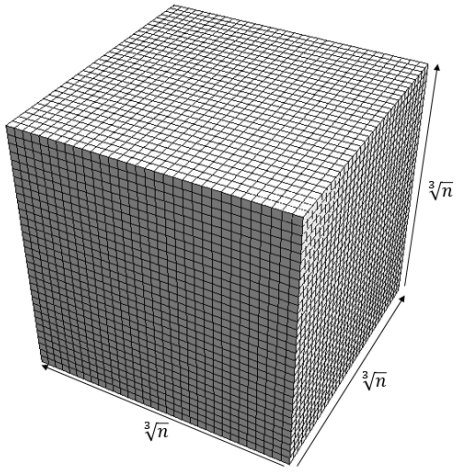


Fig. 2-18 Generated 3D Mesh structure by FDM.

Exascale 時代以降には, 百から数千 core が並列に動作する Intel Xeon Phi, nVIDIA GPU などのアクセラレータが現れる．これらで高速処理を行うためには，SIMD(Single Instruction Multiple Data）を適用する必要がある．SIMD 処理を行うためには，データ境界にアラインされた連続的なデータアクセスを行う必要がある．FDM は効率良く SIMD 処理を実現できる．

有限要素法の場合, 行列ベクトル積のデータアクセスが不規則となり, 反復法行列ソルバーの実行速度が低速する原因となる．今後は，3 次元メモリの普及などにより，高速で大容量のメモリの利用が容易となる．よって, 有限要素法の形状表現力を用いて節点数を減少させるアプローチより，多数の直方体メッシュ（VOXEL）を用いて形状を正確に記述し，かつ，高速に求解できる FDM のアプローチがより有効になるものと考えられる．

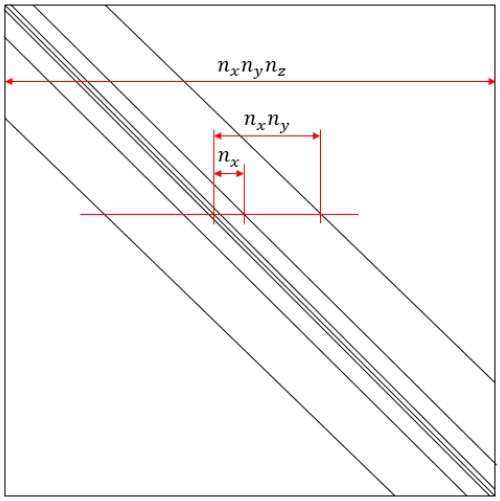


Fig. 2-19 Generated matrix from FDM.

2.4 並列環境で反復法を実行する場合の基本演算要素

本章では，式(2-56)により表される偏微分方程式を FDM で離散化した場合に生成される大規模スパース行列について，一般的に利用されている対称行列向け反復解法である CG 法を用いて並列計算機上で求解する場合を考える．一般的にアルゴリズムを並列計算機上で効率良く実行するためには，ロードバランスを考慮し負荷が均等になるように領域を分割する必要がある．具体的には，次の処理の均等な負荷分散を考慮する必要がある．

- 全領域における集団通信加算演算
- 分割領域内部の通信を伴わない演算
- 共有境界上データの隣接 PE との交換

並列計算機上で CG 法による行列解法を最適に行う場合には，

- 隣接 PE との通信を伴う行列ベクトル積
- システム内の全 PE とのグローバル通信・同期を伴う内積演算
- 分割領域内部でのベクトルの加算および減算

の全てについて，負荷を考慮し，実現する必要がある．

2.4.1 CG 法アルゴリズムと演算

Fig.2-20 に，CG 法のアルゴリズムを示す．Fig.2-20 より，CG 法の 1 反復は，1 回の行列ベクトル積（下線部分）と，2 回の内積（上波線部分），そして，2 回の Broadcast（上波線部分）より構成されることがわかる．

$$
\begin{aligned}
&\boldsymbol{r}_0 = \boldsymbol{b} - \underline{\boldsymbol{A}\boldsymbol{x}_0};\ \boldsymbol{p}_0 = \boldsymbol{r}_0 \\
&k = 0; \\
&while\left(\left\|\boldsymbol{r}_k\right\| > \varepsilon\right) \\
&\quad \boldsymbol{t}_k = \underline{\boldsymbol{A}\boldsymbol{p}_k} \\
&\quad \alpha_k = \frac{\left\langle \boldsymbol{r}_k , \boldsymbol{r}_k \right\rangle}{\left\langle \boldsymbol{p}_k , \boldsymbol{t}_k \right\rangle} \\
&\quad \boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{t}_k \\
&\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{p}_k \\
&\quad \beta_k = \frac{\left\langle \boldsymbol{r}_{k+1} , \boldsymbol{r}_{k+1} \right\rangle}{\left\langle \boldsymbol{r}_k , \boldsymbol{r}_k \right\rangle} \\
&\quad \boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \beta_k \boldsymbol{p}_k \\
&\quad k = k + 1 \\
&end\ while
\end{aligned}
$$

Fig.2-20　Algorithm of Conjugate Gradient method.

式(2-56)を FDM で離散化される場合の 1 差分格子点当たりの演算回数を考える． Fig. 2-17 の行ベクトル要素を$C_{ijk}^{0} \sim C_{ijk}^{3}$に置き換えると，規則的対称スパース行列とベクトルの積は次式のように表される．

$$
\begin{aligned}
&C_{ijk-1}^{3}\, u_{ijk-1} + C_{ij-1k}^{2}\, u_{ij-1k} + C_{i-1jk}^{1}\, u_{i-1jk} + C_{ijk}^{0}\, u_{ijk} + C_{ijk}^{1}\, u_{i+1jk} \\
&\qquad + C_{ijk}^{2}\, u_{ij+1k} + C_{ijk}^{3}\, u_{ijk+1}
\end{aligned}
\tag{2-65}
$$

式(2-65)より 1 度の行列ベクトル積において 1 格子点あたり 13 回の積和を含む浮動小数点演算を必要とすることがわかる．1 格子点あたりの内積演算は格子点上のベクトルをa, bとすると，$+a_{ijk}b_{ijk}$となり 2 回の浮動小数点演算を必要とする． Fig.2-20 より，r，x．p に関する通信を伴わないベクトル演算は, 1 格子点あたりそれぞれ 2 回の浮動小数点演算が必要となることがわかる．CG 法全体では，1 度の行列ベクトル積演算と 2 度の内積演算と 3 度の通信を伴わないベクトル加算演算の合計で 1 格子点あたり 23 回の浮動小数点演算を必要とする．

ここで，一般化のために，1 反復中 1 格子点当たりの各演算回数を，F_M（行列積演算由来），F_V（ベクトル演算由来）と定義し，1 反復中の格子点当たりのトータルの演算回数，$F_M + F_V$をF_Tと定義する．F_Tに全格子点数倍すれば全体の演算量が求められる．

2.4.2 行列ベクトル積に伴う通信

偏微分方程式の離散化から得られた疎行列とベクトルの積を並列計算機上で行う場合，①各 PE に存在しない境界データを隣接 PE より取得し，②自らの PE の境界データを隣接 PE に転送する必要がある．Fig. 2-21 にその様子を示す．Fig. 2-21 は 2 次元の領域を mesh 構造の PE に map した場合に起こる隣接 PE との間に発生する通信の x 方向のみを示している．実際には，図に示した x 方向の通信とは別に y 方向の通信も発生する．

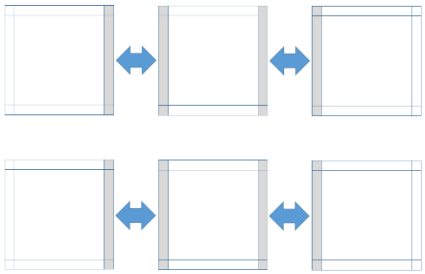


Fig. 2-21 Communication between neighbor PE

この場合の通信量は，各 PE が担う計算ボリュームの表面積に比例する．したがって，行

列ベクトル積の処理速度は，並列計算機のインターコネクトトポロジーには依存せず，LinkMode とデータのマッピング方法のみに依存することがわかる．

トータルn個の Grid を持つ立方体（一辺$\sqrt[3]{n}$ Grid）形状の領域を，トータル p 台の PE を有する Linear Array，2D Mesh(正方形，一辺$\sqrt{p}$台)，3D Mesh(立方体，一辺$\sqrt[3]{p}$台）形状の並列計算機上にマッピングする問題を考える．Fig.2-25 に，n個の Grid を持つ立方体の計算対象領域を，Linear Array，2D Mesh，3D Mesh で領域分割した際の PE 1 台分の領域形状を示す．

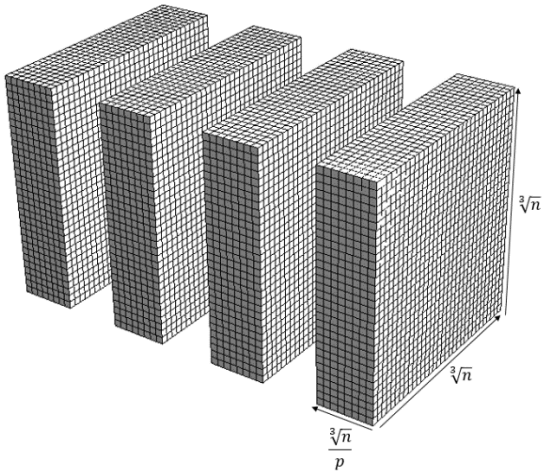


Fig. 2-22 Domain decomposition using Linear Array processor.

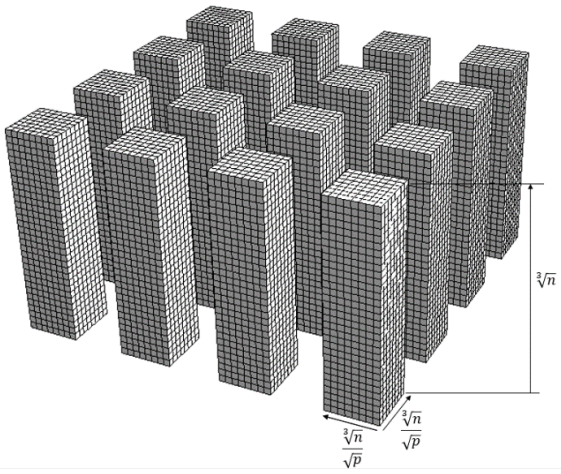


Fig. 2-23 Domain decomposition using 2D Mesh processor.

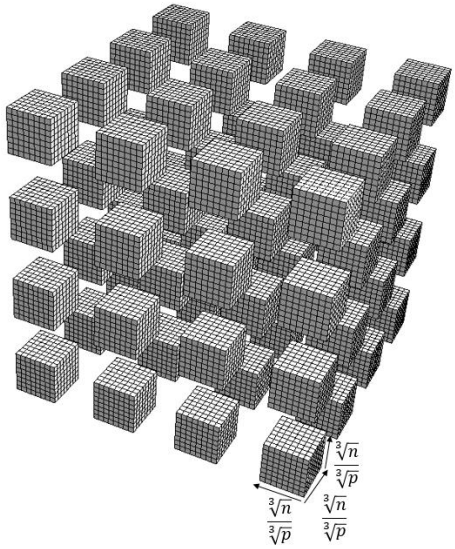


Fig. 2-24 Domain decomposition using 2D Mesh processor.

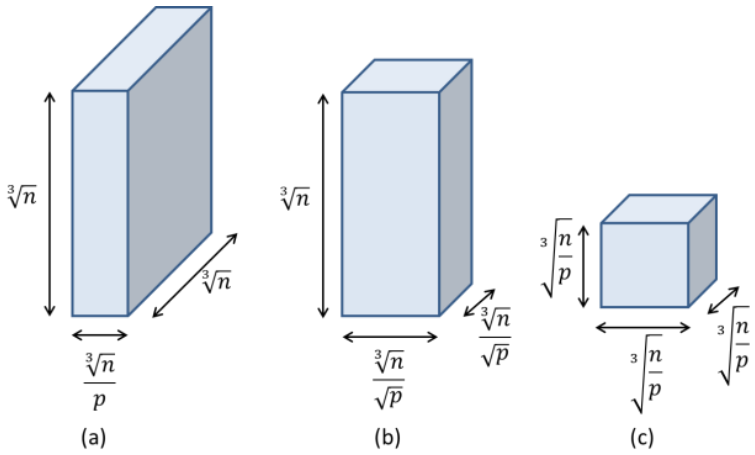


Fig.2-25　Mapping method of computational domain to single PE (a) Linear Array, Ring　(b) 2D Mesh, 2D Torus　(c)3D Mesh, 3D Torus, Hypercube.

Fig.2-25 において，行列ベクトル積の計算を行うためには，表面積に対応する Grid 分のデータを隣接 PE と交換する必要がある．例えば，Fig. 2-1 (a)の 1 次元 Linear Array で LinkMode1 を適用する場合であれば，Predecessor および Successor と相互に$n^{\frac{2}{3}}$個の Grid

データを交換する必要がある．「Predecessor よりの受信と，自らのデータの Successor への送信」とを同時に行うことができ，その逆である，「Successor よりの受信と，自らのデータの Predecessor への送信」も同時に行うことができるため，式(2-1)と Fig.2-25 より，隣接 PE との通信時間は$2T_c\left(n^{\frac{2}{3}}\right)$と見積もることができる．

Fig.2-26 より得られる 1 次元方向の交換データ量に伴う通信時間を次元数分加算すれば，2 次元，3 次元の場合にも容易に拡張することが可能である．求めた合計通信時間を Table 2-4 に行列ベクトル積を行う場合に発生する通信時間を，領域分割を行う際の次元と LinkMode の関係として示す．

.

Table 2-4　Communication cost of Matrix Vector Production for single PE. This table means the function　$LC(LinkMode, Topology, n, p)$.

	1D	2D	3D
LinkMode1	$2T_c\left(n^{\frac{2}{3}}\right)$	$4T_c\left(\frac{n^{\frac{2}{3}}}{\sqrt{p}}\right)$	$6T_c\left(\left(\frac{n}{p}\right)^{\frac{2}{3}}\right)$
LinkMode2	$T_c\left(n^{\frac{2}{3}}\right)$	$2T_c\left(\frac{n^{\frac{2}{3}}}{\sqrt{p}}\right)$	$3T_c\left(\left(\frac{n}{p}\right)^{\frac{2}{3}}\right)$

2.4.3 内積演算に伴う通信

並列計算機上で反復法による行列解法を実行する時，さらに必要な演算に内積演算がある．演算は，①各 PE 内部にあるデータを用いての内積演算の実行　②各 PE に存在する内積の部分和について集団通信加算演算を行い，root PE へ集積し内積値を求める．　③求めた内積の値を全 PE へ broadcast する，という 3 つのプロセスより成る．

Fig.2-26 に，Topology が Linear Array と Ring の場合の計算プロセスを示す．

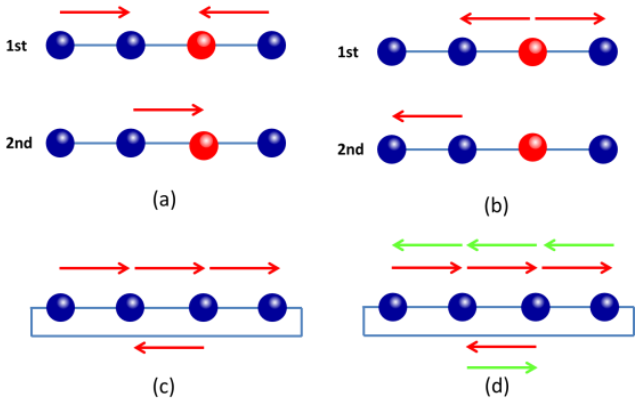


Fig.2-26 Inner product operation process on parallel computer (a) Reduction operation on Linear Array at LinkMode1 (b) Broadcast operation on Linear Array at LinkMode1 (c) Reduction and Broadcast simultatenous operation on Ring at LinkMode1 (d) Reduction and Broadcast simultatenous operation on Ring at LinkMode2.

Fig.2-26(a)は，LinkMode1 の Linear Array 上の集団通信加算演算を表す．ここで，矢印はメッセージの移動を表す．隣接 PE の計算結果を受け取った PE は自分の持つ内積部分和と加算し，次の PE へ送る．最終的に赤丸の root PE へ値が集積され，内積が求まる．したがって，PE 数を p とすると，Linear Array で内積を LinkMode1 で実行する場合は，$\left\lfloor \frac{p}{2} \right\rfloor$ 回の通信による $\left\lfloor \frac{p}{2} \right\rfloor T_c(1)$秒の時間的コストがかかることがわかる．

次に，root PE に集積された内積値を全 PE に Boradcast するプロセスを Fig.2-26(b)に示す．前述の集団通信加算演算の場合と逆プロセスの Broadcast となり，所要時間コストは$\left\lfloor \frac{p}{2} \right\rfloor T_c(1)$と同一になる．したがって，LinkMode1 の Linear Array で内積演算を実行する場合は，一回の内積演算部分和の Reduction 演算と結果の得られた結果の Broadcast に $2\left\lfloor \frac{p}{2} \right\rfloor T_c(1)$のコストがかかることがわかる．

Fig.2-26(c)は, LinkMode1 の Ring 上で集団通信加算演算を使った内積演算と Broadcast を同時に行った場合の 1 回分のプロセスを示す. 隣接 PE より他の PE が計算した内積の部

分和を受け取り，PE が p 台あれば，p-1 回同様のプロセスを繰り返せば，内積を求めると同時に結果の Broadcast も完了する．

LinkMode2 の Ring 上で集団通信加算演算を使った内積演算と broadcast を同時に行った場合のプロセスの 1 回分のプロセスを Fig.2-26(d)に示す．各 PE の内積の部分和を回す部分が LinkMode1 と同様の方向と逆の方向の 2 方向あるため，$\left\lfloor \frac{p}{2} \right\rfloor$回だけ Fig.2-26 (d)のプロセスを繰り返せば，内積演算を終了する．

2D Mesh は，ある次元軸に平行に複数の Linear Array を並べた構造と見ることができるため，異なる軸方向へ 2 回内積部分和の Reduction 演算を行えば，内積演算を行うことができる．同様に，3D Mesh 上の内積演算については，Linear Array 上の reduction および broadcast プロセス（Fig.2-26(a)，(b)）を次元数分だけ繰り返せば，実行することが可能となる．また，2D Torus，3D Torus に関しては，Ring のプロセス（Fig.2-26(c)，(d)）を次元分だけ繰り返せば，内積を計算できる．

d-dimensional Hypercube は，1 辺 2PE で構成される d 次元立方体と解釈できるので，$p = 2^{\mathrm{d}}$となる．したがって，LinkMode1 に関しては，Linear Array のプロセス（Fig.2-26(a)，(b)）を$\mathrm{Log}_2(\mathrm{p})$回繰り返せば，内積を計算することが可能となり，LinkMode2 に関しては，Ring のプロセス（Fig.2-26(c)，(d)）を$\mathrm{Log}_2(p)$回繰り返せば，内積を計算することが可能となる．上記と同様の方法で，他の高次元のトポロジーも内積演算が可能となる．

前記の内容を集約し， Table 2-5 に，1 回の内積演算に必要な集団通信加算演算と得られた結果の各 PE へのブロードキャストの双方を行う場合の通信コストを表す．

.

Table 2-5 Communication frequency with adjacent PEs in doing one innrer product operation and broadcasting result. This table means the function $GC(LinkMode, Topology, p)$

Topology	LinkMode1	LinkMode2

Linear Array	$2\left\lceil\frac{p}{2}\right\rceil$	$2\left\lfloor\frac{p}{2}\right\rfloor$
Ring	$p-1$	$\left\lfloor\frac{p}{2}\right\rfloor$
2D Mesh	$4\left\lceil\frac{\sqrt{p}}{2}\right\rceil$	$4\left\lfloor\frac{\sqrt{p}}{2}\right\rfloor$
2D Torus	$2(\sqrt{p}-1)$	$2\left\lfloor\frac{\sqrt{p}}{2}\right\rfloor$
3D Mesh	$6\left\lceil\frac{\sqrt[3]{p}}{2}\right\rceil$	$6\left\lfloor\frac{\sqrt[3]{p}}{2}\right\rfloor$
3D Torus	$3(\sqrt[3]{p}-1)$	$3\left\lfloor\frac{\sqrt[3]{p}}{2}\right\rfloor$
Hypercube	$2Log_2(p)$	$Log_2(p)$

2.5 並列計算機上での実行時間予測モデル

本節では，2.4 節で定義した行列ベクトル積演算と内積演算を用いて，3 次元空間を FDM で離散化した場合に生成される行列を，並列計算機上で CG 法によって解く際の計算時間推定の定式化を行う．その際，新たなパラメータ，通信遅延隠蔽閾値 PE 数を導入し定式化を進める．

2.5.1 処理時間の定式化

一般に，CG 法を mpi 等によって並列化する場合の擬似コードを Fig. 2-27 に示す．3 節で述べたように，外部ループ 1 反復は，通信を伴う行列ベクトル積を行うループ，通信を伴う内積を行うループ，通信を伴わないループの 3 種より構成される．

```
DO WHILE( not converged)

Start Communication
    DO inner_Loop1 k=1,n
            // Matrix Vector type   Loop
    END DO inner_Loop1
END   Communication

    DO inner_Loop2 k=1,n
        // Inner   Product type   Loop
    END DO inner_Loop2
Start Communication
END   Communication

   DO inner_Loop3 k=1,n
            //No communication type Loop
   END DO inner_Loop3
…..
END DO WHILE
```

Fig. 2-27 Pseudo code of parallelized CG method.

したがって，ループ長 n の複数のループ演算より構成される内積演算および行列ベクトル積演算を p 台の PE で並列実行する場合，各 PE での担当分のループ長が $\frac{n}{p}$ であることより，CG 法 1 反復当たりの処理時間は，次の式で見積もられる．

$$\sum_i \frac{n\,F_i}{p\,r_i} + ComCost_i \tag{2-66}$$

ここで，F_iは i 番目のループ演算での 1 格子点当たりの演算数とする．r_iは演算速度，$ComCost_i$は通信時間である．r_iは演算の種類によって異なる．

次に，CG 法の 1 反復あたりの計算時間について，インターコネクトトポロジーおよび通信リンクの特性を考慮したモデルを考える．行列ベクトル積演算にかかる通信時間は，Table 2-4 に示したように，$LinkMode$，$Topology$，問題サイズ n の関数として表せる．表の結果を関数$LC(LinkMode, Topology, n)$と定義する．同様に，内積演算の実行にかかる通信回数（通信時間）に関しては，Table 2-5 に示したように，$LinkMode$およびネットワーク$Topology$の関数として表せ，関数$GC(LinkMode, Topology)$と定義する．

次に，内積およびその他の通信が発生しない 1 反復当たりのベクトル演算について考える． 1 格子点あたりの 1 回の内積の演算量を F_I，その他の通信を伴わないベクトル演算の 1 格子点あたりの合算演算量をF_Lとする．1 反復中にα回の内積が実行されるとすると，1 反復中の 1 格子点当たりの演算量F_Vは，式(2-67)のように表される．

$$F_V = \alpha\, F_I + F_L \tag{2-67}$$

F_Vが処理速度r_1で実行されると仮定し，1 格子点当たりの行列ベクトル積の演算数をF_M，その処理速度をr_M，1 反復中にβ回の行列ベクトル積が実行されるとすると，一般的な反復法の 1 反復を p 台の PE で処理するために必要な時間T_eは，式(2-66)，式(2-67)，および Table 2-4，Table 2-5 より式(2-68)のように表される．

$$\begin{aligned} T_e = \beta \frac{n\, F_M}{p\, r_M} + \frac{n\, F_V}{p\, r_1} + \beta\ LC(LinkMode, Topology, n, p) \\ +\alpha\ GC(LinkMode, Topology, p)\, T_c(1) \end{aligned} \tag{2-68}$$

式(2-68)では，個別のループ演算ごとに，処理速度を設定できるが，後の議論のために，行列ベクトル積とそれ以外の演算の二種に分けて記述した．CG 法の場合は，$\alpha = 2$，$\beta = 1$ となるが，後述の近似式でローカル演算とグローバル演算の影響度を把握するため，ここで

は変数として残すこととした.

.

2.5.2 通信遅延隠蔽閾値 PE 数

隣接 PE との通信を伴う行列ベクトル積演算では, 演算開始前に自 PE が必要とされる境界データの取得および隣接 PE が必要とする自 PE 内の境界データの提供を行う. 非同期 recv および send を用いることで, この処理の通信による遅延を隠蔽できる場合がある. 大規模並列計算においては, 通信遅延はそのまま効率の低下に直結するため, その性質を知ることは重要である.

行列ベクトル積の通信に費やす時間をT_{cL}, 計算に費やす時間をT_{eL}とすると,

$$T_{eL} \geq T_{cL} \tag{2-69}$$

の時通信遅延隠蔽が成立する. PE 数を基準に考えると, ある PE 数P_{th}より小さな PE 数の時, 行列ベクトル演算の計算時間が, それに伴う通信時間より長くなることが, 通信遅延隠蔽が起こる原因となる. 今, このP_{th}を通信遅延隠蔽閾値 PE 数(Communication Latency Hiding Threshold PE Number) と定義する. P_{th}より大きな領域では, 必ず計算時間が通信時間を下回ることになる. これは, 式(2-68)と Table 2-4 より, 計算時間は, p^{-1}に比例し, それに比べ, 通信時間が, マッピングの次元数に依存して, p^0 (1D の時), $p^{-1/2}$ (2D の時), $p^{-2/3}$ (3D の時)に比例することより明らかである.

PE 数がP_{th}に等しい時, すなわち, $p = P_{th}$の時, 行列ベクトル積の計算時間と通信時間が等しくなるため, 式(2-68)より, 式(2-70)が成り立つ.

$$\frac{nF_M}{p\, r_M} = LC(LinkMode, Topology, n, p) \tag{2-70}$$

式(2-70)を p について解くことができれば, $LinkMode$, 空間分割$Topology$および問題サイズの関数としてP_{th}を求められる. P_{th}は近接 PE との通信により決定されるため, Fig.2-25

に示したマッピングの次元のみに依存し，インターコネクトトポロジーには依存しない．

例えば，空間分割に 3 次元分割を用いて，LinkMode1 の通信リンクを用いる場合，式(2-70)の具体的な形は，Table 2-4 より式(2-71)のようになる．

$$\frac{nF_M}{pr_M} - 6\left(\frac{1}{c}\left(\frac{n}{p}\right)^{2/3} + \tau_c\right) = 0 \tag{2-71}$$

式(2-71)をpについて解くことができれば，具体的にP_{th}を求めることができる．式(2-72)に 3D Mesh，3D Torus，Hypercube に関して 3D のマッピングと LinkMode1 の通信リンクを用いる場合のP_{th}の具体的な形を示す．

$$P_{th} = \frac{1}{6r_M^3}\left(\frac{nF_M r_M^2}{\tau_c} + \frac{26^{2/3} n^2 r_M^6}{\left(3c^6 n^3 F_M r_M^8 \tau_c^5 + \sqrt{3}\sqrt{c^9 n^6 r_M^{16} \tau_c^9 (16r_M^2 + 3c^3 F_M^2 \tau_c)}\right)^{1/3}} - \frac{(18c^6 n^3 F_M r_M^8 \tau_c^5 + 6\sqrt{3}\sqrt{c^9 n^6 r_M^{16} \tau_c^9 (16r_M^2 + 3c^3 F_M^2 \tau_c)})^{1/3}}{c^3 \tau_c^3}\right) \tag{2-72}$$

P_{th}は，Table 2-4 に示したように，マッピングの次元数と LinkMode の違いで，6 種類存在する．その 6 種類に関して，同様にP_{th}を決定することができる．

次に，本章で用いる並列計算機モデルを構成する要素の仕様を Table 2-6 に示す．Table 2-6 では，計算速度，Link の転送スピード，Link のセットアップ時間を定めた．

Table 2-6 Assumption of computer specification.

	Symbol	Value
Processing Speed of PE (Matrix Vector operation)	r_M	10 G Flops
Representative Processing Speed of PE	r_1	10 G Flops
Communication Latency	τ_c	1μs
Communication Speed	c	20 Gbps

同様に，他のケースについても，空間マッピング法，LinkMode，問題サイズの関数としてP_{th}を代数的に表すことができる．これらの結果に Table 2-6 を適用することで

Fig.2-28 を得る．マッピングの次元と LinkMode の組み合わせを変えることで，6 通りの状態を示す．ここで，LM は LinkMode を表す．

Fig.2-28 は，ある問題サイズ n で行列ベクトル積を行う場合に，PE が通信時間を隠蔽し滞りなく計算を進められる最大の PE 数を示している．よって，ある問題サイズ n でのP_{th}より小さな PE 数では通信時間が隠蔽されるため計算効率が高く，P_{th}より大きな PE 数では通信時間が隠蔽されないため計算効率が低くなる．

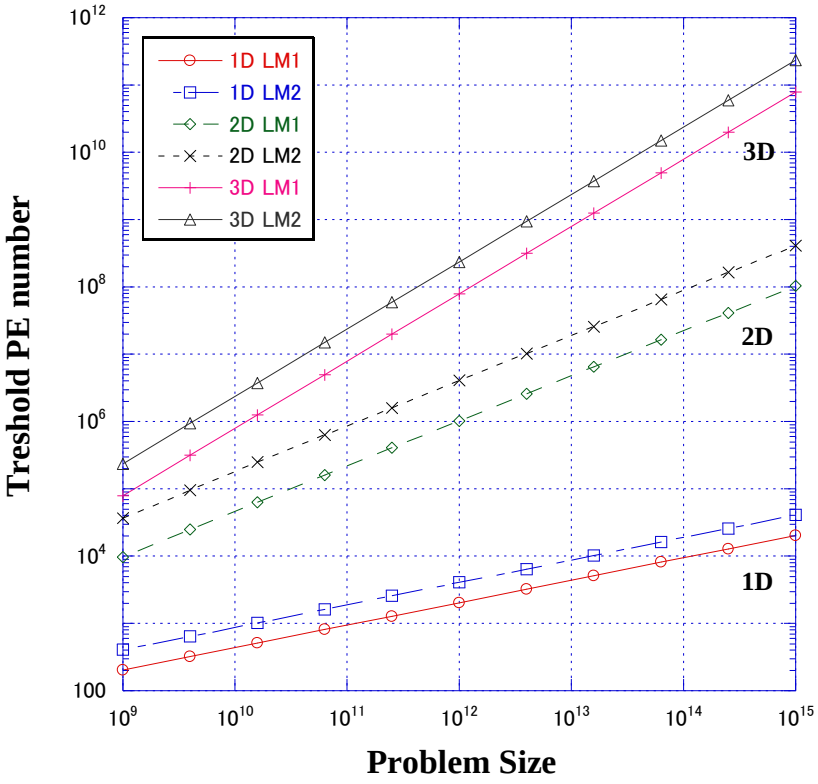


Fig.2-28　Latency Hiding Threshold PE Number.

Fig.2-28 より，LinkMode1 より LinkMode2 の方が PE 全体での通信速度が大きいため通信時間が短時間となり，P_{th} が大きくなっていることがわかる．また，グラフの傾きから，

P_{th}は，1D マッピング時に問題サイズ n の 1/3 乗，2D マッピング時に 2/3 乗，3D マッピング時に n の 1 乗に各々比例する性質を持つことがわかる．

Table 2-7 Approximated P_{th} (Latency Hiding Threshold PE Number).

	Link Mode	P_{th}
1D	LM1	$\frac{cF_M n^{1/3}}{2r_M}$
	LM2	$\frac{cF_M n^{1/3}}{r_M}$
2D	LM1	$\frac{c^2F_M^2 n^{2/3}}{16r_M^2}$
	LM2	$\frac{c^2F_M^2 n^{2/3}}{4r_M^2}$
3D	LM1	$\left(\left(\frac{2r_M}{9c^6F_M\tau_c^5}\right)^{1/3}+\frac{F_M}{6r_M\tau_c}-\frac{2}{9}\left(\frac{4r_M^5}{3c^{12}F_M^5\tau_c^7}\right)^{1/3}-\left(\frac{F_M}{6c^3r_M\tau_c^4}\right)^{1/3}\right)n$
	LM2	$\left(\frac{F_M}{3r_M\tau_c}-\left(\frac{F_M}{3r_Mc^3\tau_c^4}\right)^{1/3}+\left(\frac{r_M}{9c^6F_M\tau_c^5}\right)^{1/3}\right)n$

Fig.2-28 の結果より考えると，Table 2-6 の前提の範囲では，P_{th}は式(2-73)の形で近似できる．

$$\mathrm{C}_1 n^{\mathrm{C}_2} \tag{2-73}$$

ここで，C_1とC_2は定数である．式(2-70)に Table 2-4 を適用し代数的に解いた結果に関して，Table 2-6 の具体的な数値を基に，項の大小比較，級数展開等の代数的な方法で近似解を求めることができ，結果を Table 2-7 に示す．これは，Table 2-6 の仕様が前提の近似解であり，現状の計算機の仕様に近い．Table 2-6 より値が変化する場合は，新たな近似解を

求める必要がある．結果を Figure にまとめると解析解と Table 2-7 の近似解はほぼ一致する．また，

Fig.2-28 より求めた n の次数と Table 2-7 の近似解の n の次数もほぼ一致する．

2.5.3 PE 数と処理速度の関係

CG 法の実行性能を求めるために，1 反復当たりの実行時間について考える．システム構成上の可変な要素として想定するのは，インターコネクトトポロジー，LinkMode，問題サイズ n，PE 数 p である．これらの 4 種のパラメータが可変なモデルを考える．通信および演算に関するパラメータは Table 2-6 の定数を用いる．Fig. 2-27 で示した演算ループ毎に異なる演算処理性能を設定することは可能である．議論の本質には影響しないため，これらの定数の中の内部演算ループの処理速度を同一とし，代表処理速度r_1として進める．

今，PE 数が増加した場合の反復法の 1 反復当たりの処理時間について考える．前述の通り，$p > P_{th}$ において通信遅延隠蔽状態ではなくなり，その場合の 1 反復当たりの計算時間T_e^{NH}は，行列ベクトル積に伴う通信時間，内積に伴う通信時間，ベクトル演算に伴う時間，により構成される．式(2-68)より，

$$T_e^{NH} = \frac{n\,F_V}{p\,r_1} + \beta\ LC(LinkMode, Topology, n, p) + \alpha\ GC(LinkMode, Topology, p)\,T_c(1) \tag{2-74}$$

と表すことができる．また，$p \le P_{th}$ の場合の処理時間T_e^Hは，通信遅延隠蔽状態となり，行列ベクトル積に伴う通信時間が隠蔽されるため，式(2-68)より，

$$T_e^H = \frac{nF_T}{p\,r_1} + \alpha\,GC(LinkMode, Topology, p)\,T_c(1) \tag{2-75}$$

と表すことができる．ここで，

$$F_T = F_V + \beta\, F_M \tag{2-76}$$

とした．

2.5.4 Strong scaling

全体の問題サイズnを固定し，システムの並列度すなわち PE 数を増加させた場合の計算時間の特性を Strong scaling という．この場合，PE 数の増加に従って，各 PE へ割り当てられた計算領域は小さくなる．隣接 PE とのローカルな通信時間は，Table 2-4 より小さくなるが，内積等のグローバルな通信については，増加する．並列システムにおける Strong scaling は，式(2-74)および式(2-75)を用いて数値的に正確に表すことができる．

T_e を例示するために，Table 2-6 と，問題サイズ$n = 10^{10}$を仮定する．PE 数が$p > P_{th}$の場合，T_e は式(2-74)となり，$p \leq P_{th}$の場合は式(2-75)となる．T_e を求めた結果を Fig. 2-29 に示す．Fig. 2-29 より，CG 法の 1 反復の計算時間は，変化の大小はあるが，すべてのインターコネクトトポロジーで，PE 数が増加するに従い計算時間は減少し，ある PE 数を過ぎた後に増加することがわかる．これは，アーキテクチャに依存した極小値の存在を示している．1D の Linear Array，Ring 等の次元の低いインターコネクトトポロジー程，極小値が現れる PE 数が小さく，また，極小値を過ぎた後，計算時間が増加する率が速いことを示す．

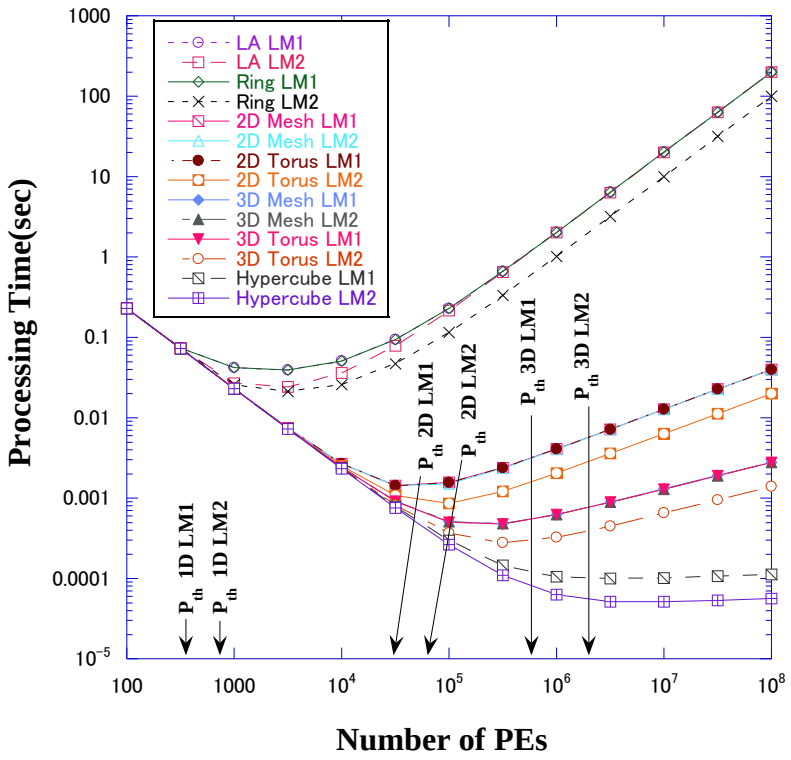


Fig. 2-29　Relation between PE number and operation time.

また，LinkMode1 の方が LinkMode2 より計算時間が，特に，極値付近で長いことがわかる．

Table 2-7 を用いて，$n = 10^{10}$の場合のP_{th}を Fig. 2-29 に矢印で示す．このP_{th}を境界として，PE 数が$p > P_{th}$の場合，T_e は式(2-74)となり，$p \leq P_{th}$の場合は式(2-75)となる． Fig. 2-29 より，全て極小値の手前にP_{th}が位置することがわかる．

式(2-74)，式(2-75)の 1 反復あたりの計算時間T_e と，1 反復あたり 1 格子点あたりの演算量を用いて，全体の処理速度を求めることが可能である．

$$\frac{nF_T}{T_e} \tag{2-77}$$

式(2-77)に Table 2-6 を適用することにより，$n = 10^{10}$の問題に関して，Table 2-6 の構成要素で並列計算機を組み上げた場合の PE 数と計算速度の関係を示すことができる．その結

果を Fig. 2-30 に示す.

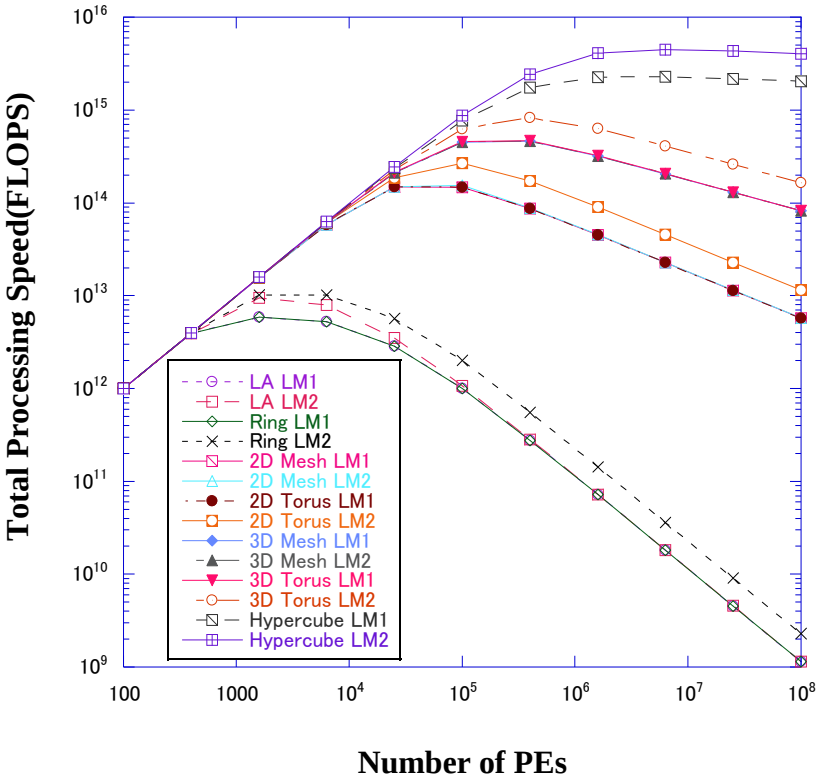


Fig. 2-30　Relation between PE number and operation speed.

1D のインタコネクトアーキテクチャが著しく速度が低いことがわかる． 3D Mesh，3D Torus と Hypercube を比較すると，インターコネクトが複雑な程，より高い性能が得られることがわかる．Hypercube の結果と Torus の結果と比べると，ピークで 7 倍程度高速な結果となることがわかる．これは，Hypercube が Link の数が多く，Log(p)オーダーで集団通信加算演算を行えることに起因する. Torus どうしの比較でも LinkMode の差によって，2 倍程度の速度差が生じていることがわかる.

2.6 並列処理システムの限界実行速度

2.4 節の Fig. 2-30 で示したように，並列計算機のインターコネクト毎に，固有の限界実行速度(Limit Execution Speed)があるということがわかった．この場合，限界実行速度を与える PE 数 $P_{opt}(Optimal\ PE\ Number)$において，式(2-78)が成り立つ．

$$\frac{\partial T_e}{\partial \mathrm{p}} = 0 \tag{2-78}$$

式(2-78)を求め，pについて解くことができれば，T_eに極小値を与えるP_{opt}を求められる．式(2-78)は，T_eに通信遅延隠蔽が起こるか否かにより，式(2-74)または式(2-75)のどちらかが選択され，どちらの状態になるかを考慮する必要がある．

通信遅延隠蔽が起こらない場合を考える．式(2-74)について，pに関する導関数を求め，式(2-78)が成り立つpを求める近似式P_{opt}^{NH}を導出し Table 2-8 に示す．同様に，通信遅延隠蔽が起こる場合について，式(2-75)に基づいてP_{opt}^{H}を求める近似式 Table 2-9 に示す． Table 2-8，Table 2-9 より，LinkMode に依存せず，与えられた条件の下では，

$$P_{opt}^{NH} < P_{opt}^{H} \tag{2-79}$$

となることがわかる．次に，P_{th}を基準に考え，下記の 3 種類の状態に分類して考える．

はじめに，$P_{opt}^{NH} > P_{th}$ の場合を考える．この場合は，通信遅延隠蔽状態ではないため，$P_{opt} = P_{opt}^{NH}$となる．次に，$P_{opt}^{H} < P_{th}$の場合である．このとき，通信遅延隠蔽状態であるため，$P_{opt} = P_{opt}^{H}$と考えられる．

最後に，$P_{opt}^{NH} < P_{th} < P_{opt}^{H}$ の状態について考える． 実際の 3D Mesh LinkMode1 のケースを Fig.2-31 に示す．式(2-74)，(2-75)を用いて処理時間をグラフ化し，右側に極小値があるグラフが，通信遅延隠蔽状態の処理時間を表し，左側に極小値があるグラフが，通信遅延非隠蔽状態の処理時間を表す．

Table 2-8 Optimal PE number for No Latency Hiding(P_{opt}^{NH})

	LM1	LM2
Linear Array	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$
Ring	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	$\left(\frac{2F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$
2D Mesh	$\left(\left(\frac{F_V}{\alpha r_1 \tau_c}\right)^{\frac{2}{3}} + \frac{4\beta}{3c\alpha\tau_c}\right) n^{\frac{2}{3}}$	$\left(\left(\frac{F_V}{\alpha r_1 \tau_c}\right)^{\frac{2}{3}} + \frac{2\beta}{3c\alpha\tau_c}\right) n^{\frac{2}{3}}$
2D Torus	$\left(\left(\frac{F_V}{\alpha r_1 \tau_c}\right)^{\frac{2}{3}} + \frac{4\beta}{3c\alpha\tau_c}\right) n^{\frac{2}{3}}$	$\left(\left(\frac{2F_V}{\alpha r_1 \tau_c}\right)^{\frac{2}{3}} + \frac{2\beta}{3c\alpha\tau_c}\right) n^{\frac{2}{3}}$
3D Mesh	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}} + \frac{3\beta}{c\alpha\tau_c} n^{\frac{2}{3}}$	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}} + \frac{3\beta}{2c\alpha\tau_c} n^{\frac{2}{3}}$
3D Torus	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}} + \frac{3\beta}{c\alpha\tau_c} n^{\frac{2}{3}}$	$\left(\frac{2F_V n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}} + \frac{3\beta}{2c\alpha\tau_c} n^{\frac{2}{3}}$
Hypercube	$\left(\frac{4\beta^2}{3c^2}\left(\frac{2\mathrm{Log}[2]^5 r_1}{F_V \alpha^5 \tau_c^5}\right)^{\frac{1}{3}} + \frac{F_V \mathrm{Log}[2]}{2\alpha r_1 \tau_c} + \frac{\beta}{c}\left(\frac{4F_V \mathrm{Log}[2]^4}{\alpha^4 r_1 \tau_c^4}\right)^{\frac{1}{3}}\right) n$	$\left(\frac{4\beta^2}{3c^2}\left(\frac{\mathrm{Log}[2]^5 r_1}{F_V \alpha^5 \tau_c^5}\right)^{\frac{1}{3}} + \frac{F_V \mathrm{Log}[2]}{\alpha r_1 \tau_c} + \frac{2\beta}{c}\left(\frac{F_V \mathrm{Log}[2]^4}{\alpha^4 r_1 \tau_c^4}\right)^{\frac{1}{3}}\right) n$

Table 2-9 Optimal PE number for Latency Hiding(P_{opt}^{H}).

	LM1	LM2
Linear Array	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$
Ring	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	$\left(\frac{2F_T n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$
2D Mesh	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{2}}$	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{2}}$
2D Torus	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{2}}$	$\left(\frac{2F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{2}}$

3D Mesh	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$
3D Torus	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$	$\left(\frac{2F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$
Hypercube	$\frac{F_T \mathrm{Log}[2]}{2\alpha r_1 \tau_c} n$	$\frac{F_T \mathrm{Log}[2]}{\alpha r_1 \tau_c} n$

この場合，矢印で示したように，P_{th}より小さい領域では，通信遅延隠蔽状態となり，大きな領域では，通信遅延非隠蔽状態となる．現実に選択可能な曲線は，$p = P_{th}$の時に最小値を取る V 字型の曲線となる．

次に， 3D Mesh LinkMode1 のケースに関して，問題サイズnとP_{opt}^{NH}，P_{opt}^{H}，P_{th}の関係を Figure にまとめた結果を Fig. 2-32 に示す．図中の問題サイズで，P_{opt}^{NH}およびP_{opt}^{H}について，式(2-79)の関係が成り立っている．また，P_{th}を表す曲線が，この二つの曲線を横断している．P_{th}を表す曲線より下の領域では通信遅延隠蔽状態となり，曲線より上の領域では通信遅延非隠蔽状態となることと，P_{opt}^{NH}およびP_{opt}^{H}を表す曲線の条件関係より，Fig. 2-32 に示した青色の曲線がP_{opt}となる．図に示した通り，問題サイズnが通信遅延の非隠蔽領域にある時は，P_{opt}=P_{opt}^{NH}となり，通信遅延の隠蔽領域では，$P_{opt} = P_{opt}^{H}$となることがわかる．図中のP_{th}領域では$P_{opt} = P_{th}$となる．

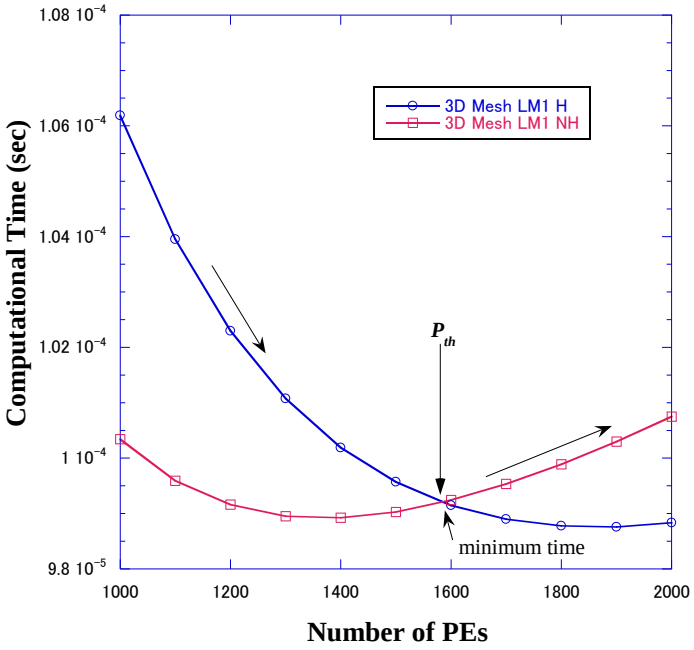


Fig.2-31　Relation between P_{th} and Communication Hiding mode(1).

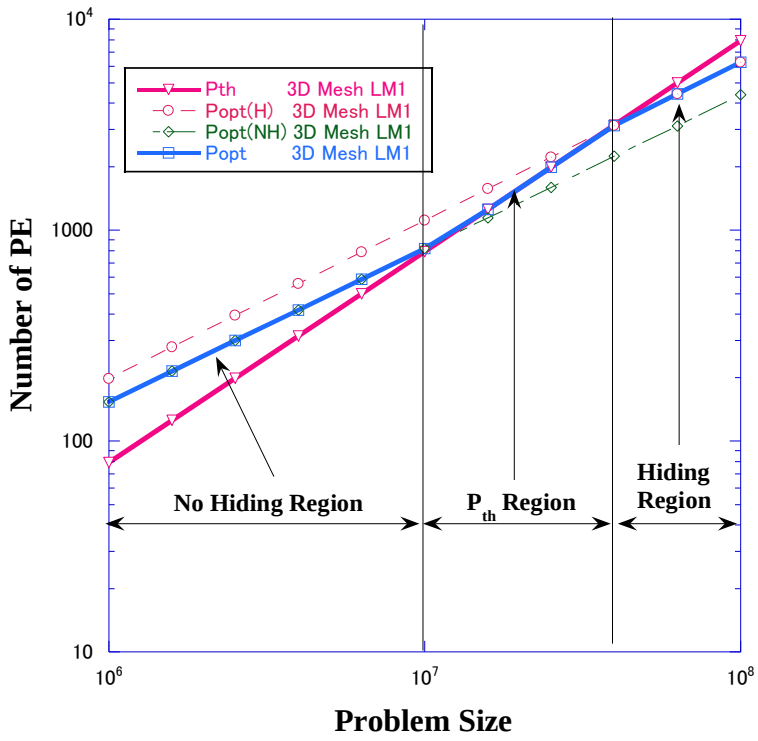


Fig. 2-32　Relation between P_{th} and Communication Hiding. mode(2).

前述のP_{th}を基準とする判別法によって，Table 2-8，　Table 2-9 を基にP_{opt}をグラフ化した結果を Fig.2-33 へ示す．1D，2D，3D，Hypercube と高次な構造になるにつれ，大きな PE 数で極値を持ち，またプロットの傾きが急になるのがわかる．Fig.2-34 に，問題サイズnが10^6以上10^8以下の領域について示す．前提としている並列計算機の仕様のもとで CG 法アルゴリズムを実行する場合，Fig.2-34 に示したようにこの問題サイズの領域ではP_{opt}^{NH}よりP_{opt}^{H}の乗り換えが起きる場合がある．しかし，これより問題サイズが大きな領域では，P_{th}，P_{opt}^{NH}，P_{opt}^{H}の位置関係は一定となり乗り換えはなくなる．

前述のP_{th}を基準とする判別法と，Table 2-8，Table 2-9 を基にして，$10^{10} \leq n$の場合のP_{opt}の近似式を Table 2-10 に示す．ここで，CLH は通信遅延隠蔽の状態を表し，H が通信遅延隠蔽，NH が通信遅延非隠蔽，TH がP_{th}領域を示す．

Table 2-10 より，P_{opt}のnの次数について，インターコネクト構造が 1D では$1/2$，2D では$2/3$，3D では$3/4$，Hypercube では 1 となることがわかる．

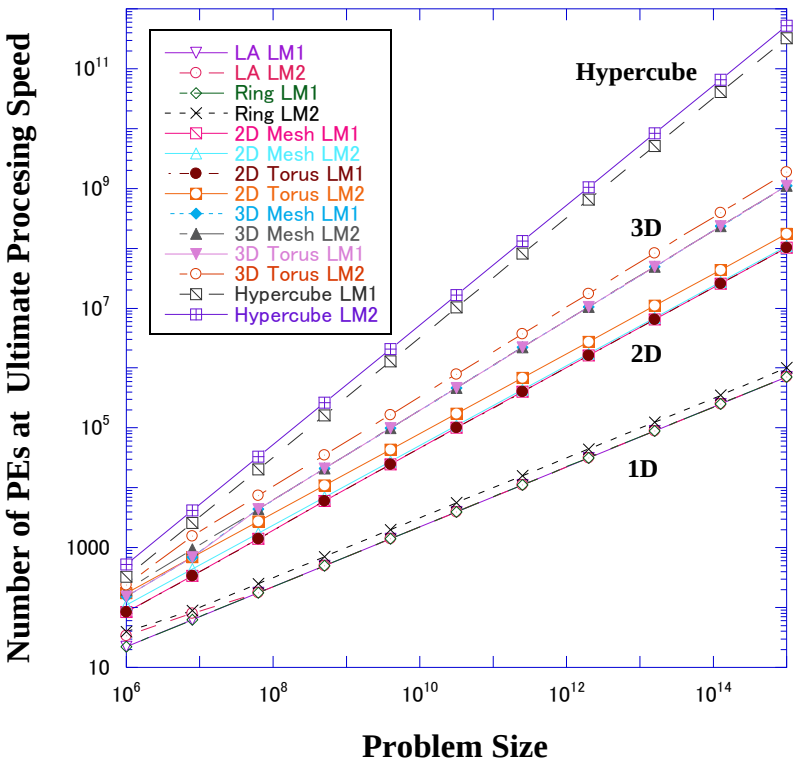


Fig.2-33 Number of PEs at Limit Execution Speed (P_{opt}).

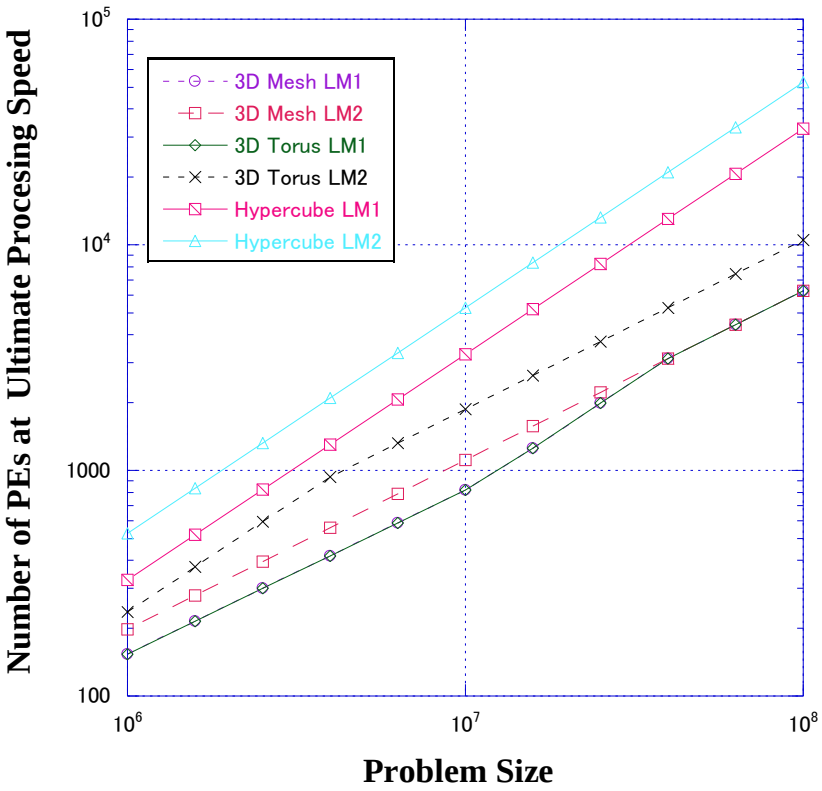


Fig.2-34　Number of PEs at Limit Execution Speed (P_{opt}) at $10^6 \leq n \leq 10^8$.

次に，Table 2-10 より得られたP_{opt}を通信遅延隠蔽の状態に従って，式(2-74)，式(2-75)に代入することで，計算時間を得る．計算時間と式(2-77)により，各インターコネクトの限界実行速度の Figure，Fig.2-35，Fig. 2-36 を得る．これらの Figure は，横軸に問題サイズn，縦軸にある問題サイズでの限界実行速度を，アーキテクチャおよび LinkMode をパラメータとして変化させプロットした．問題サイズの増加に伴って，限界実行速度も単調に増加している．各プロットが LogLog プロットで直線近似できることより，プロットは$C_1 n^{C_2}$の形で近似できることがわかる．

Table 2-10 Approximated analytical form of Optimal PE number at Limit Execution Speed.

	LM1		LM2	
	P_{opt}		P_{opt}	
Linear Array	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	NH	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	NH
Ring	$\left(\frac{F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	NH	$\left(\frac{2F_V n}{\alpha r_1 \tau_c}\right)^{\frac{1}{2}}$	NH
2D Mesh	$\frac{c^2 F_M^2 n^{2/3}}{16 r_1^2}$	TH	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{2}{3}}$	H
2D Torus	$\frac{c^2 F_M^2 n^{2/3}}{16 r_1^2}$	TH	$\left(\frac{2F_T n}{\alpha r_1 \tau_c}\right)^{\frac{2}{3}}$	H
3D Mesh	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$	H	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$	H
3D Torus	$\left(\frac{F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$	H	$\left(\frac{2F_T n}{\alpha r_1 \tau_c}\right)^{\frac{3}{4}}$	H
Hypercube	$\left(\frac{4\beta^2}{3c^2}\left(\frac{2\mathrm{Log}[2]^5 r_1}{F_V \alpha^5 \tau_c^5}\right)^{\frac{1}{3}} + \frac{F_V \mathrm{Log}[2]}{2\alpha r_1 \tau_c} + \frac{\beta}{c}\left(\frac{4F_V \mathrm{Log}[2]^4}{\alpha^4 r_1 \tau_c^4}\right)^{\frac{1}{3}}\right) n$	NH	$\left(\frac{4\beta^2}{3c^2}\left(\frac{\mathrm{Log}[2]^5 r_1}{F_V \alpha^5 \tau_c^5}\right)^{\frac{1}{3}} + \frac{F_V \mathrm{Log}[2]}{\alpha r_1 \tau_c} + \frac{2\beta}{c}\left(\frac{F_V \mathrm{Log}[2]^4}{\alpha^4 r_1 \tau_c^4}\right)^{\frac{1}{3}}\right) n$	NH

nが10^8以上の領域では，インターコネクト毎の性能が明確に分かれる．Fig. 2-36 に示すように，nが小さな領域では，傾きは同じであるが，y 切片の大きさが異なるために性能曲線が互いに交差したり，P_{th}の影響で性能曲線が大きく曲がったりしているため，複雑な状態となる．特筆すべきことは，nが10^8未満の領域では，Hypercube の LinkMode1 より 3D Torus の LinkMode2 の方が高速となることである．

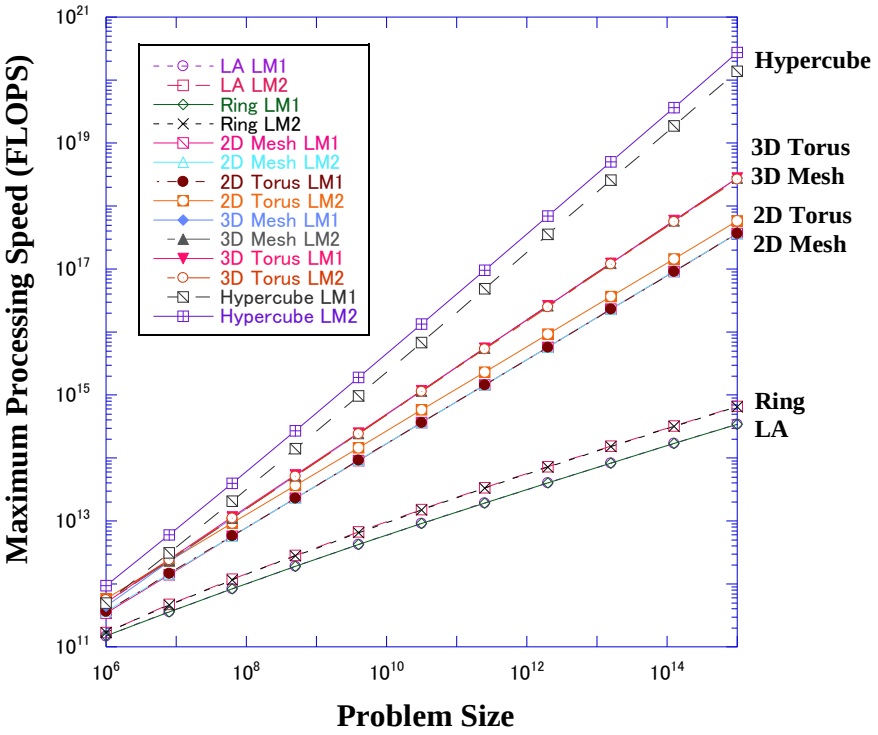


Fig.2-35　Maximum Processing Speed of CG method (1).

次に，Table 2-10 の限界実行速度をもたらすP_{opt}の解析式を，処理時間を表す式(2-74)，(2-75)に代入し限界実行速度を求め Table 2-11 に示す．Table 2-11 では，CG 法を各インターコネクトと各 LinkMode で構成した並列計算機上で実行した場合の最大性能を解析的に示している．

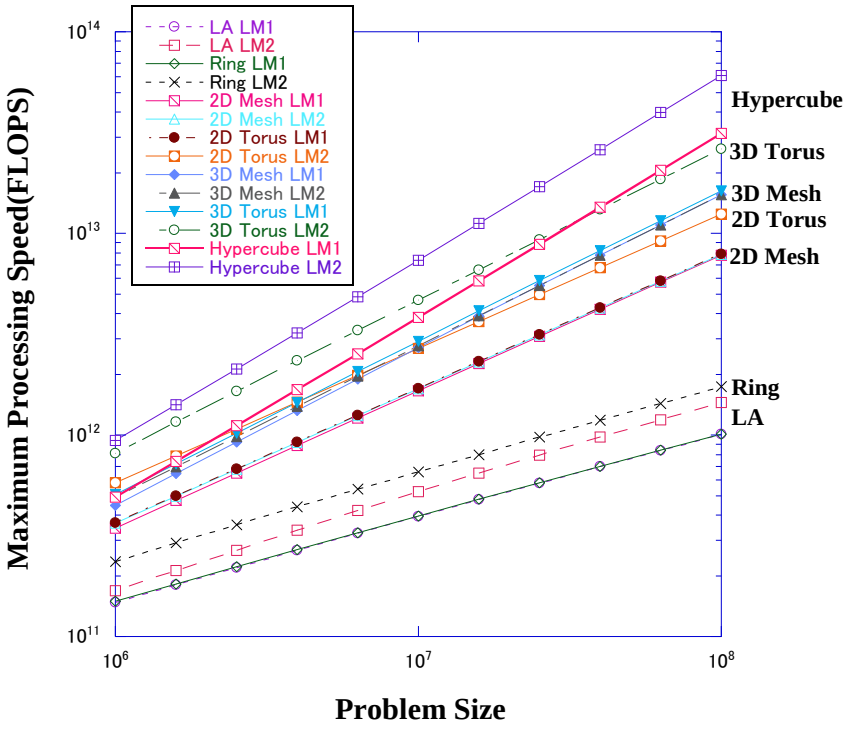


Fig. 2-36　Maximum Processing Speed of CG method (2).

Table 2-6 の前提で，問題サイズnが10^8以上の領域において誤差数%以内で求められた．Hypercube を除くインターコネクトは，限界実行速度が，1 次元で $n^{1/3}$，2 次元で$n^{2/3}$，3 次元で$n^{3/4}$に比例することが新たに判明し，これは，Fig.2-35 の結果とも一致する．

Table 2-11 Approximated analytical form of Limit Execution Speed of CG method execution.

	LM1		LM2	
	V_{max}		V_{max}	
Linear Array	$\frac{cF_T}{2\beta}n^{1/3}$	NH	$\frac{cF_T}{\beta}n^{1/3}$	NH
Ring	$\frac{cF_T}{2\beta}n^{1/3}$	NH	$\frac{cF_T}{\beta}n^{1/3}$	NH
2D Mesh	$\frac{2c^2F_M^2F_Tr_1}{32F_Tr_1^2+c^3\alpha F_M^3\tau_c}n^{2/3}$	TH	$\frac{1}{3}\left(\frac{F_T^2r_1}{\alpha^2\tau_c^2}\right)^{1/3}n^{2/3}$	H

2D Torus	$\frac{2c^2F_M^2F_Tr_1}{32F_Tr_1^2+c^3\alpha F_M^3\tau_c}n^{2/3}$	TH	$\frac{1}{3}\left(\frac{4F_T^2r_1}{\alpha^2\tau_c^2}\right)^{1/3}n^{2/3}$	H
3D Mesh	$\frac{1}{4}\left(\frac{F_T^3r_1}{\alpha^3\tau_c^3}\right)^{1/4}n^{3/4}$	H	$\frac{1}{4}\left(\frac{F_T^3r_1}{\alpha^3\tau_c^3}\right)^{1/4}n^{3/4}$	H
3D Torus	$\frac{1}{4}\left(\frac{F_T^3r_1}{\alpha^3\tau_c^3}\right)^{1/4}n^{3/4}$	H	$\frac{1}{2}\left(\frac{F_T^3r_1}{2\alpha^3\tau_c^3}\right)^{1/4}n^{3/4}$	H
Hypercube	$\frac{F_T}{2\alpha\mathrm{Log}_2[n\theta]\tau_c}n$ *,where* $\theta=\frac{4}{3c^2}2^{1/3}\beta^2\mathrm{Log}[2]^{5/3}\left(\frac{r_1}{\alpha^5F_V\tau_c^5}\right)^{1/3}$ $+2^{2/3}\beta\mathrm{Log}[2]^{4/3}\left(\frac{F_V}{c^3\alpha^4r_1\tau_c^4}\right)^{1/3}$ $+\frac{\mathrm{Log}[2]F_V}{2\alpha r_1\tau_c}$	NH	$\frac{F_T}{2\alpha\mathrm{Log}_2[n\theta]\tau_c}n$ *,where* $\theta=\frac{4}{3c^2}\beta^2\mathrm{Log}[2]^{5/3}\left(\frac{r_1}{\alpha^5F_V\tau_c^5}\right)^{1/3}$ $+2\beta\mathrm{Log}[2]^{4/3}\left(\frac{F_V}{c^3\alpha^4r_1\tau_c^4}\right)^{1/3}$ $+\frac{\mathrm{Log}[2]F_V}{\alpha r_1\tau_c}$	NH

Table 2-11 の結果と簡単な代数計算により，Hypercube の限界実行速度は，nの 1 乗に比例することが新たにわかった．

2.7 第２章のまとめ

本章では，ダイレクトネットワークを使用したローカルメモリー型の理想的な並列計算機上での実行性能を予測する方法を新たに提案した．提案法では，新たに通信遅延隠蔽閾値 PE 数を定義し，それを用いて実行時間を予測した．

さらに，提案法を用いて，FDM による離散化の際に生成される行列に，並列計算機上で CG 法を適用する場合の限界実行速度と限界実行速度をもたらす PE 数について解析的に示した．そして，与えられた問題のサイズと限界実行速度がどのような関係を持つのか新たに示すことができた．

第3章　逆行列ベースの要素棄却型の IC 分解前処理の性能評価

本章では，逆行列ベースの要素棄却型の IC 分解前処理の性能評価について述べる．3.1 節では本章の導入を企図して目的と意義について概観する．3.2 節では，IC 分解で生じる誤差に関する考察を行う．3.3 節では，前処理行列の逆行列のノルムの評価方法およびそれに基づきフィルインに対するドロッピング方法について述べる．3.4 節では IB ドロッピング手法に基づく IC 分解のアルゴリズムを記述する．3.5 節では，数値実験を通じて，閾値を変化させ，従来の IC 分解の収束不安定性と提案法の収束安定性の違いについて明らかにする．3.6 節では，本章の内容をまとめる．

3.1 は じ め に

連立一次方程式 $Ax = b$を解くことを考える．ここでAは対称かつ正則な行列 $A = (a_{ij}) \in R^{n\times m}$，解ベクトル$x \in R^n$，右辺ベクトル$b \in R^n$とする．このような連立一次方程式を反復法で解く場合，前処理として IC(Incomplete Cholesky，以下 IC と略す)分解が広く用いられている．IC 分解では，係数行列を近似した前処理行列 $M = U^T U(\approx A)$ を生成し，その逆行列を係数行列に作用させて変換し，変換された後の方程式に対して共役勾配(Conjugate Gradient，以下 CG と略す) 法などの反復法を適用する．さらに，ドロッピング付き IC 分解では，元の係数行列では値が零の要素が，前処理行列の分解過程で零でない値を持った要素(以下，フィルインと呼ぶ) に対して，予め定めた閾値より小さい場合はそれを棄却するという不完全分解を行う．その際，閾値の大きさによっては，反復法の収束性に大きなばらつきが生じることがある．この理由は，従来のドロッピングつき IC 分解では，「行列 U の誤差行列$\tilde{X}$が小さい場合には，その逆行列 U^{-1} の誤差行列列Xもおのずと小さいであろう」という前提で不完全分解がなされてきたことに起因する [15] [16] [46] [47]．しかし，その理論的な必然性はなく，逆行列 U^{-1}の誤差行列Xが偶然大きい場合に，その影響により前処理つき反復

法の収束性が悪化している危険性がある．しかし，逆行列 U^{-1} の誤差行列 X を計算する適当な計算手段があまりなかった．

そこで，非対称行列用の前処理として Crout 版 ILU 分解(以下，ILUC 分解と略す) が提案された [36]．原論文では，従来の ILUT(dual Threshold ILU) 分解 [49] [51]との収束性の違いが明らかにされ，さらに逆行列 U^{-1}の誤差行列 X の具体的な評価方法について Inverse-based ドロッピング手法と呼ばれる方法が提案された．元々逆行列 L^{-1}の行ノルムあるいは逆行列 U^{-1} の列ノルムを利用したドロッピング手法は文献 [15]で発表された．その後，ILUC 分解は，文献 [35] [42]などでピボッティング操作が組み入れられる等盛んに研究されている [75] [76]．正方行列の場合の逆行列を一般化した，例えば，Moore-Penrose 形の一般逆行列などを用いることも考えられる [74] [68]．しかし，Moore-Penrose 一般逆行列は特異値分解とも深く関係する．いま，行列の次元数を n とする時，特異値分解の計算量は $O(n^5)$で評価され非常に多いことが知られている．一方，反復法の計算量は $c \times O(n^2)$で評価される．ここで，係数 c は収束までの反復回数によって決まるため，そのため，反復法の研究では，少ない反復回数で近似解が得られることが重要とされる．また，係数 c の大きさも，次元数 n と比べてだいぶ少ないのが普通であり，したがってオーダーも$O(n^3)$を超えることはない．本研究の目的は，Inverse-based ドロッピング手法の考え方を対称行列に適用すること，さらに原論文では十分明らかにされなかった Inverse-based ドロッピング手法が持つ収束特性をより明確にすることである [70] [69] [54]．

本章の構成は以下の通りである．3.2 節では，従来の IC 分解とフィルインの棄却について述べる．3.3 節では，IC 分解で生じる誤差に関する考察を行う．3.4 節では，前処理行列 Uの逆行列 U^{-1}のノルムの評価方法およびそれに基づきフィルインに対するドロッピング方法について述べる．3.5 節では，Inverse-based ドロッピング手法に基づく IC 分解(以下，ib_IC 分解と略す) のアルゴリズムを記述する．3.6 節では，数値実験を通じて，閾値を変化させ，従来の IC 分解の収束不安定性と ib_IC 分解の収束安定性の違いについて明らかにする．3.7 節では，まとめと今後の課題を述べる．

3.2 従来の IC 分解とフィルインの棄却

連立一次方程式の係数行列 A を，上三角行列 $U = (u_{i,j})$ を使って $U^T U$ の形に完全(コレスキー) 分解する．このとき，元の行列 A では値が 0 であった要素が分解過程で非零要素になり，その結果行列 U は一般に密行列になる．また，行列 A を完全分解するのは，計算時間とメモリ量の増大を招くので，通常は行なわれない．そこで，行列 U も元の行列 A と同程度に疎性を保つように不完全に分解を行う．従来の IC 分解では，元の行列Aを次のように近似分解する．ここで，$R = (r_{i,j}) \in R^{n\times m}$ は狭義上三角行列，上付き添え字 T は行列の転置を各々表す．行列 R は分解の不完全さを表す形式的な行列で，IC 分解途中で棄却された行列 U の要素を表す．

$$A = U^T U + R + R^T \tag{3-1}$$

従来用いられてきたドロッピング手法では，IC 分解過程で現れたフィルインが，ある閾値 τ よりも小さい時そのフィルインを棄却するというものであった．すなわち，行列 R の要素の集合 P_{IC} とおくと，集合P_{IC}に含まれるフィルインは棄却していた．ここで，$a^*_{i,j}, \overline{a_{i,i}}$ は，最終的に $u_{i,j}, u_{i,i}$ となる要素を各々表す．

$$P_{IC} = \left\{ (i,j) \mid |a^*_{i,j}| / \sqrt{(\overline{a_{i,i}})(\overline{a_{j,i}})} \le \tau \right\} \tag{3-2}$$

3.3 IC 分解で生じる誤差に関する考察

前処理つき CG 法では，解くべき方程式$Ax = b$を次式のように変換して解くのが一般的である．

$$(U^{-T} A U^{-1})(Ux) = U^{-T} b \tag{3-3}$$

したがって，前処理後の係数行列は，$U^{-T} A U^{-1}$ になる．このとき，行列 A を完全分解して得られた上三角行列を$\overline{U}$とする時，従来の IC 分解により得られた行列Uは誤差行列$\tilde{X}$を用いて，次のように表せる．

$$U^T = \bar{U}^T + \tilde{X}^T,\ U = \bar{U} + \tilde{X} \tag{3-4}$$

同様に，別の誤差行列Xを用いて，行列Uの逆行列U^{-1}は次のように表せる．

$$U^{-T} = \bar{U}^{-T} + X^T,\ U^{-1} = \bar{U}^{-1} + X \tag{3-5}$$

これらの式(3-4), (3-5)を使うと，前処理後の係数行列$U^{-T}AU^{-1}$は次のように表せる．

$$U^{-T}AU^{-1} = I + \bar{U}X + X^T\bar{U}^T + X^TAX \tag{3-6}$$

ここで，式(3-6)中に式(3-4)で定義した誤差行列$\tilde{X}$が現れていないことに注意を要する．したがって，一般に行列 U の誤差行列$\tilde{X}$が小さい場合に逆行列U^{-1}の誤差行列 X も小さいという保証はまったくなく，式(3-6)から，もし逆行列 U^{-1}の誤差行列 X が大きい場合には前処理つき反復法の収束性に影響を及ぼす可能性があることがわかる．したがって，小さなフィルインを棄却するドロッピング操作が前処理行列の逆行列U^{-1} に与える影響の見積もり，すなわち，U^{-T}の行のノルム(U^{-1} の列のノルム)をできるだけ正確に見積もり，それに基づきドロッピングする ib_IC 分解の方が，より適切なドロッピングができると考えられる[8][18]．Table 3-1 に従来の IC 分解と ib_IC 分解における誤差行列およびドロッピングの時対象となる行列の違いを示す．

Table 3-1 Error matrix in conventional IC decomposition and ib_IC · target matrix in the case of dropping.

分解	Upper triangular matrix and the error matrix X of inverse matrix U^{-1}	Target matrix in the case of dropping
IC	$U \equiv \bar{U} + \tilde{X},\ U^{-1} \equiv \bar{U}^{-1} + X$	$U = \bar{U} + \widetilde{X^T}$
ib_IC	Same above	$U^{-1} = \bar{U}^{-1} + X$

3.4 Inverse-based ドロッピング手法に基づく IC 分解

ここでは，行列 $U^T = (u_{j,i})$ を下三角行列 $L = (l_{i,j})$を用いて表現する．同様に，行列$\bar{U}^T = (\bar{u}_{j,i})$も行列 $L = (l_{i,j})$ を用いて表現する．また，行列 L_kを，1 から k 列目まで前処理行列 L の 1 から k 列目までと同じ要素を持ち，k+1 から n 列目まで要素$\overline{a_{i,i}}$ $(1 \le i \le n)$ を対角要素に持つ対角行列と同じ要素を持つ下三角行列とする．ただし，要素$\overline{a_{i,i}}$は行列 A の第 i 行の対角要素に対応し，分解過程において次々と更新される．

$$L_k = \begin{pmatrix} l_{1,1} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{2,1} & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & l_{k,k} & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & \overline{a_{k+1,k+1}} & \ddots & & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \cdots & l_{n,k} & 0 & \cdots & 0 & \overline{a_{n,n}} \end{pmatrix}. \tag{3-7}$$

行列 L の k 列目を生成する時，初めてフィルイン$l_{j,k}(j > k)$ が棄却されたと仮定する．このとき，行列 L_k は以下のように表現できる．ただし，行列 L_k は，行列 L の k 列目を生成する時にフィルイン$l_{j,k}(j > k)$を棄却しなかった場合に相当する行列である．すなわち，行列 L_kは係数行列 A を k 列目まで完全分解した行列である．また，単位ベクトル e_i $(1 \le i \le n)$ はi行目が 1 で，i 以外の行は全て 0 の列ベクトルとする．

$$L_k = \bar{L}_k - l_{j,k} e_j e_k^T \tag{3-8}$$

$j > k$より，$L_k e_j = \overline{a_{j,j}} e_j$であることに注意すると，

$$L_k = \bar{L}_k - l_{j,k} e_j e_k^T = \bar{L}_k \left(I - l_{j,k} e_j e_k^T / \overline{a_{j,j}} \right) \tag{3-9}$$

である．式(3-9)より，行列L_k^{-1}を次のように表現できる．

$$\begin{aligned} L_k^{-1} &= \left(I - l_{j,k} e_j e_k^T / \overline{a_{j,j}} \right)^{-1} \bar{L}_k^{-1} \\ &= L_k^{-1} + l_{j,k} e_j e_k^T \bar{L}_k^{-1} / \overline{a_{j,j}} \end{aligned} \tag{3-10}$$

したがって，行列L_kの逆行列L_k^{-1}は$l_{j,k}/\overline{a_{j,j}}$と$\bar{L}_k^{-1}$の k 行目の積の値だけ影響を受けることがわかる．すなわち，$\left\| I - l_{j,k} e_j e_k^T / \overline{a_{j,j}} \right\|_\infty = \left| l_{j,k} / \overline{a_{j,j}} \right| \left\| e_j e_k^T \bar{L}_k^{-1} \right\|_\infty$でドロッピングを行うことにより，ドロッピングにより影響を受ける逆行列の行のノルムを限定することができる．ただし，$\|A\|_\infty$は，行列 A の最大ノルムで，

$$\|A\|_\infty = \max_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_i \sum_{j=1}^{n} |a_{i,j}| \tag{3-11}$$

のように定義される．また，$\left\| e_j e_k^T \bar{L}_k^{-1} \right\|_\infty$は全ての要素が 0 でないあるベクトル b を用いて，次のように近似できる．

$$\begin{aligned} \left\| e_j e_k^T \bar{L}_k^{-1} \right\|_\infty &= \left\| e_j e_k^T \bar{L}^{-1} b \right\|_\infty \\ &= \max_{b \neq 0} \frac{\left\| e_j e_k^T \bar{L}^{-1} b \right\|_\infty}{\|b\|_\infty} \end{aligned} \tag{3-12}$$

したがって，$\left\| e_j e_k^T \bar{L}^{-1} \right\|_\infty$を推定するためには，式(3-12)を満たすようなベクトル$b(\neq 0)$を決定すればよい．最大ノルムの定義式(3-11)において，$i = m$ の時，

$$\sum_{j=1}^{n} |a_{i,j}| \tag{3-13}$$

最大になったとすると，その最大値を与えるベクトル$\boldsymbol{x}$の第 j 成分は，

$$x_j = \mathrm{sign}(a_{m,j}), (j = 1,2,\cdots,n) \tag{3-14}$$

となることが知られている[13][14]．ただし，sign は,

$$\mathrm{sign}(t) = \begin{cases} +1 & (t \geq 0), \\ -1 & (t < 0) \end{cases}$$

で定義される符号関数である．しかし，式(3-12)では，行列$\overline{L}^{-1}$の計算をせず，ベクトル$\boldsymbol{b}$の要素の符号を決定できない．そこで,

$$b = (-b_1, -b_2, \cdots, -b_n,)^T, \tag{3-15}$$

$$(b_1, b_2, \cdots, b_n = \pm 1) \tag{3-16}$$

の中から，$\left\| e_j e_k^T \overline{L}^{-1} b \right\|_\infty$をできるだけ大きくするベクトル$\boldsymbol{b}$を探すことを考える．このようなベクトル$\boldsymbol{b}$を見つければ，次の近似式が成り立つ.

$$\left\| e_j e_k^T \overline{L}^{-1} \right\|_\infty \approx \frac{\left\| e_j e_k^T \overline{L}^{-1} b \right\|_\infty}{\| b \|_\infty} \tag{3-17}$$

式(3-17)の右辺項は，$\overline{L}\xi = b$ の解ξの第k成分で評価でき，ベクトル$\boldsymbol{\xi}$の第k 成分は，右辺ベクトル $\boldsymbol{b}$ の第 k 成分を用いて次の式で求められる．ただし，$\boldsymbol{\xi}_k = (\xi_1, \xi_2, \cdots, \xi_k, 0, \cdots, 0\ )^T$, $\boldsymbol{b}_k = (b_1, b_2, \cdots, b_k, 0, \cdots, 0\ )^T$とする.

$$\xi_k = (b_k - {e_k}^T \overline{L}_{k-1} \xi_{k-1}) / l_{k,k} \tag{3-18}$$

以上から，$\left\| e_j e_k^T \overline{L}^{-1} \right\|_\infty$を見積もる以下のアルゴリズムが得られる．ただし，アルゴリズム中のξ_kとv_kは，$\left\| e_j e_k^T \overline{L}^{-1} \right\|_\infty$，${e_k}^T \overline{L}_{k-1} \xi_{k-1}$に各々対応する.

アルゴリズム1　$\left\|e_j e_k^T \bar{L}^{-1}\right\|_\infty$を見積るアルゴリズム

set $\xi_1 = \frac{1}{l_{1,1}}$; $\nu_i = 0\ (i = 1, \cdots, n)$ (3-19)

for $k = 2, n$

$temp_+ = 1 - \nu_k$ (3-20)

$temp_- = -1 - \nu_k$ (3-21)

if $|temp_+| > |temp_-|$ then

$\xi_k = temp_+ / l_{k,k}$ (3-22)

else

$\xi_k = temp_- / l_{k,k}$ (3-23)

end if

for $j = k + 1, n\ \left(l_{j,k} \neq 0\right)$

$\nu_j = \nu_j + \xi_k l_{j,k}$ (3-24)

end for

end for

Inverse-based ドロッピング手法では，フィルイン$l_{j,k}$に対して，

$$\left|l_{j,k} e_j e_k^T / \overline{a_{j,j}}\right| \left\|e_j e_k^T \bar{L}^{-1}\right\|_\infty = \left|l_{j,k}\right| \left|\xi_k\right| / \left|\overline{a_{j,j}}\right| \tag{3-25}$$

の時棄却する．ただし，パラメータ τ は閾値とする．すなわち，棄却された行列 U の要素の集合P_{ib_IC}は次のように表せる．

$$P_{ib_IC} = \left\{(i,j) \middle| \left(\left|a_{i,j}^*\right| \left|\xi_i\right|\right) / \left(\left|\overline{a_{j,j}}\right| \sqrt{\overline{a_{i,i}}}\right) < \tau\right\} \tag{3-26}$$

3.5 Ib_IC 分解の算法

Ib_IC 分解の算法を以下に示す．Inverse-based ドロッピングに該当する部分に下線をつ

けた．算法の中の式(3-27)は非対角要素を生成する部分を表す．同様に，式(3-29)は非対角要素を棄却する部分，式(3-30)は対角要素を求める部分，式(3-31)は非対角要素を求める部分，式(3-32)は対角要素を生成する部分を各々表す

アルゴリズム 2 ib_IC 分解

for $i = 1, \cdots, n$

$\overline{a_{i,i}} = a_{i,i}$; $\underline{\nu_i = 0}$

end for

$\underline{\xi = 1/\sqrt{\overline{a_{1,1}}}}$

for $i = 1, \cdots, n$

if $i \geq 2$ then

$\underline{temp_+ = 1 - \nu_k}$

$\underline{temp_- = -1 - \nu_k}$

if $|temp_+| > |temp_-|$ then

$\underline{\xi = temp_+/\sqrt{\overline{a_{i,i}}}}$

$else$

$\underline{\xi = temp_+/\sqrt{\overline{a_{i,i}}}}$

end if

end if

for $j = i + 1, \cdots, n$

$a^*_{i,j} = a_{i,j}$

end for

for $k = 1, \cdots, i - 1$

for $j = i + 1, \cdots, n$

$$a^*_{i,j} = a^*_{i,j} - u_{k,i}u_{k,j} \tag{3-27}$$

$$\nu_j = \nu_j + \xi\left(a^*_{i,j}/\sqrt{\overline{a_{i,i}}}\right)$$

end for

end for

for $j = i+1, \cdots, n$

$$\underline{\psi = \left|a^*_{i,j}\right||\xi|/\left(\overline{a_{j,j}}\sqrt{\overline{a_{i,i}}}\right)} \tag{3-28}$$

$$\underline{\text{if } \psi \le \tau \ \ then}$$

$$a^*_{i,j} = 0 \tag{3-29}$$

end if

end for

$$u_{i,j} = \sqrt{\overline{a_{i,i}}} \tag{3-30}$$

for $j = i+1, \cdots, n$

$$u_{i,j} = a^*_{i,j}/u_{i,i} \tag{3-31}$$

$$\overline{a_{j,j}} = \overline{a_{j,j}} - {u_{i,j}}^2 \tag{3-32}$$

end for

end for

3.6 数値実験

3.6.1 計算機環境と計算条件

数値実験は，OS を Suse Linux ver.9.2，コンパイラーを Intel Fortran Compiler ver.8.1 を使用している PC で行った. コンパイラーの最適化オプションは“-O3” とした. Fortran90 でプログラムを実装し，計算は全て倍精度浮動小数点演算で行った. 反復法には CG 法を使

用し，前処理はシフト IC 分解とシフト ib_IC 分解の 2 種類を用いた．係数行列 A は予め対角項を 1 に揃える正規化処理を施した．CG 法の初期近似解 x_0 は全て 0 とした．Krylov 部分空間：$(r_0,\ Ar_0,\ A^2r_0, \cdots)$では，初期解 x_0 よりも，むしろ駆動源である右辺項に含まれる(残差を構成する）ベクトルの係数のうち，大きな値の係数がどれだけ含まれるかによって収束性が決まることが知られている．そのため，影響が相対的に少ないと思われる初期解x_0には通例に従って零ベクトルを与えた．反復繰り返し計算の収束判定は残差ベクトル l_2 ノルムの比が $\|r_k\|_2/\|r_0\|_2 \leq 10^{-8}$ を満たした時，最大反復回数は行列の次元数と同じ値にした．最大反復回数の設定値は，行列の次元数nなどを与えることが多いが，あくまで反復計算の暴走を阻止する意味しかないことが多い．また，Krylov 部分空間法の算法における残差ベクトルは，緩和法などの残差とは定義が異なる．そのため，残差の収束半径などの評価も直接的には難しい．

閾値 τ は，0.005 から 0.15 まで 0.005 刻みで変化させ合計 30 通りの場合の収束性を調べた．ただし，通常の IC 分解で分解の破綻が起こったので，解くべき方程式の係数行列 A を $A + \alpha I$ にシフトさせたものに対して分解を行った[10]．I は単位行列とする．シフト量 α は 0.1 に固定した．ただし，予め行列は 1 に正規化しているので，シフト量 α が 0.1 の場合は加速係数の値が 1.10 の場合に相当するとも考えることができる[10]．テスト行列の主な特徴を表 2 に示す. テスト行列は, MatrixMarket[11]から行列 BCSSTK10, BCSSTK11, BCSSTK13，BCSSTK15 の合計 4 種類，Kouhia の疎行列データベース[7]から行列 S3DKQ4M2，S3DKT3M2 の合計 2 種類を選出した．

Table 3-2 The main properties of the test matrix

Matrix	Rank	Number of non zero elements	Condition number	Area
BCSSTK10	1,086	11,578	1.30E+06	Structure Analysis
BCSSTK11	1,473	17,857	5.30E+08	↑
BCSSTK13	2,003	42,943	4.60E+10	↑
BCSSTK15	3,948	60,882	8.00E+09	↑

S3DKQ4M2	90,449	2,455,670	1.90E+11	FEM
S3DKT3M2	90,449	1,921,955	3.62E+11	↑

3.6.2 実験結果

まず，行列 BCSSTK10,同 11，同 13，そして同 15 についてシフト IC 分解つき CG 法とシフト ib_IC 分解つき CG 法の収束性について調べた．ここでは，それら二つの解法の反復回数の閾値依存性について調べるとともに，式(3-10)の行列 L_k^{-1} と行列 $\bar{L}_k^{-1}$ の誤差行列 $l_{j,k}e_je_k^T\bar{L}_k^{-1}/\overline{a_{j,j}}$ に着目し，その最大ノルムを定量的に示す．具体的には，フィルインが棄却された時のアルゴリズム **2** 中の式(3-28)の総和

$$\sum_{(j,k)\in P_{IC}} \psi = \sum_{(j,k)\in P_{IC}} \left|l_{j,k}/\overline{a_{j,j}}\right| \left\|e_je_k^T\bar{L}^{-1}\right\|_\infty \tag{3-33}$$

$$\sum_{(j,k)\in P_{ib_IC}} \psi = \sum_{(j,k)\in P_{ib_IC}} \left|l_{j,k}/\overline{a_{j,j}}\right| \left\|e_je_k^T\bar{L}^{-1}\right\|_\infty \tag{3-34}$$

(以下, “誤差総和” と呼ぶ) について，シフト IC 分解(式(3-33)), シフト ib_IC 分解(式(3-34)) ともに調べ，反復回数との相関について調べた．

Fig. 3-1 より Fig. 3-4 に，行列 BCSSTK10，同 11，同 13，そして同 15 に対するシフト IC 分解つき CG 法(図中で三角印付きの折れ線グラフ)とシフト ib_IC 分解つき CG 法（図中で丸印付きの折れ線グラフ）の実験結果を各々示す．ただし，三角印，丸印の内部が塗りつぶしてあるものは反復回数，塗りつぶしていないものは誤差総和である．また，図の横軸は閾値 τ の値，左側縦軸は反復回数，右側縦軸は誤差総和（常用対数表示)とし，途中で折れ線が途切れているところは，その閾値で不完全分解が破綻したことを表す．IC 分解では, 分解途中のある行の段階で, 丸め誤差などの影響で, 対角項の値が負の値になり, それ以降の行での分解計算が停止することがある．これは，経験的には，行列の要素の値のオーダが大きいときに現れることが多い．このような場合，閾値を替えるなどして，分

解を最初からやり直すしか現実的には手段がない.

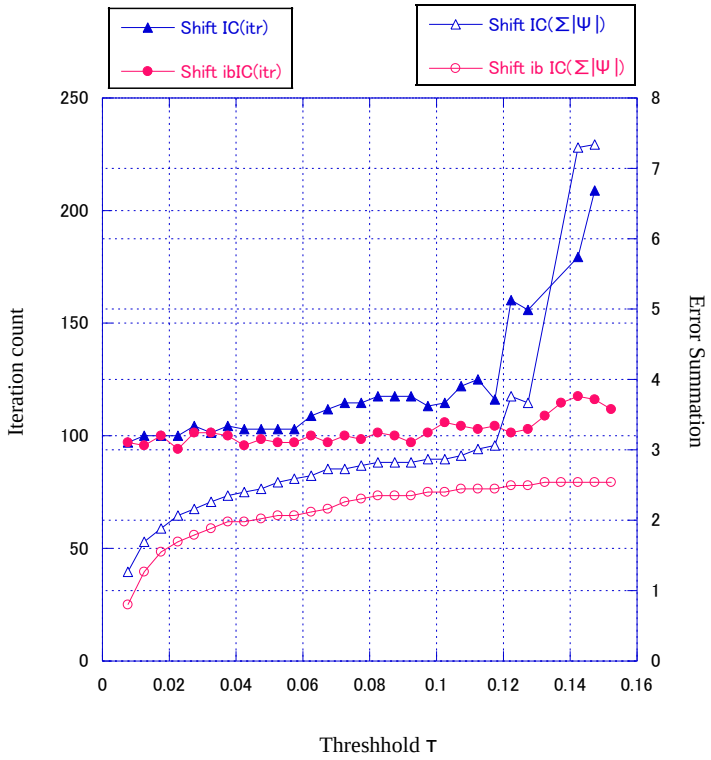


Fig. 3-1 Shifted IC of matrix BCSSTK10・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC

(▲・●：The left side vertical axis scale, △・○：The right side vertical axis scale).

Fig. 3-1 より Fig. 3-4 から以下のことが分かる.

- 行列 BCSSTK10 では，シフト IC 分解つき CG 法は，$\tau = 0.12$ の時反復回数が増え，$\tau = 0.12, 0.135, 0.15$ では分解が破綻した．また，$\tau = 0.14$, 0.145 では破綻は起きなかったものの，反復回数は極端に増えた．それらの異常が見られた閾値，

特にτ = 0.14, 0.145 では，誤差総和が極端に大きくなった．それに対して，シフト ib_IC 分解つき CG 法は，τの変化に対して反復回数に大きな変化がなかった．さらに，τの変化に対して誤差総和の変動は小さく穏やかに推移した．

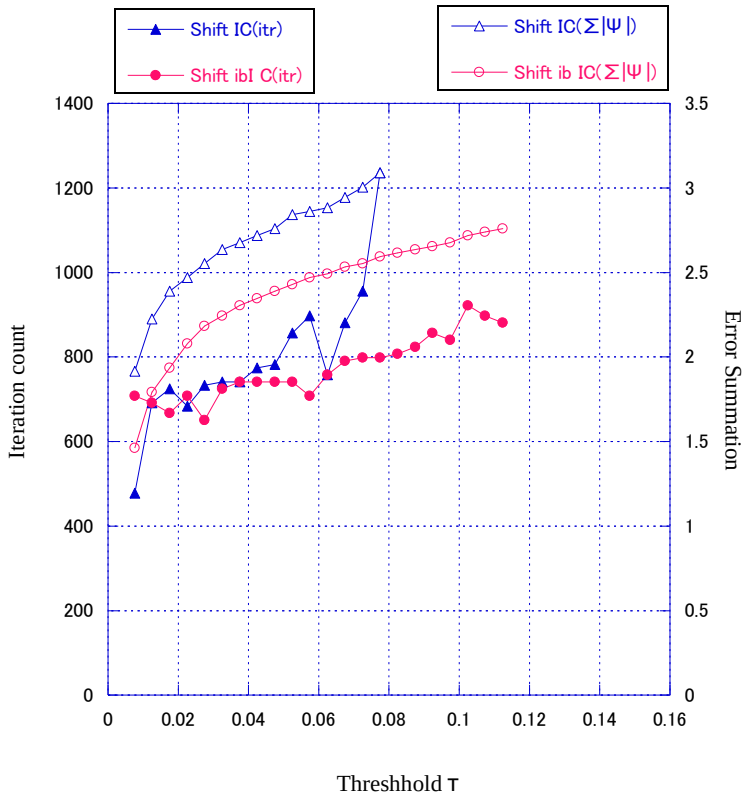


Fig. 3-2 Shifted IC of matrix BCSSTK11・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC.

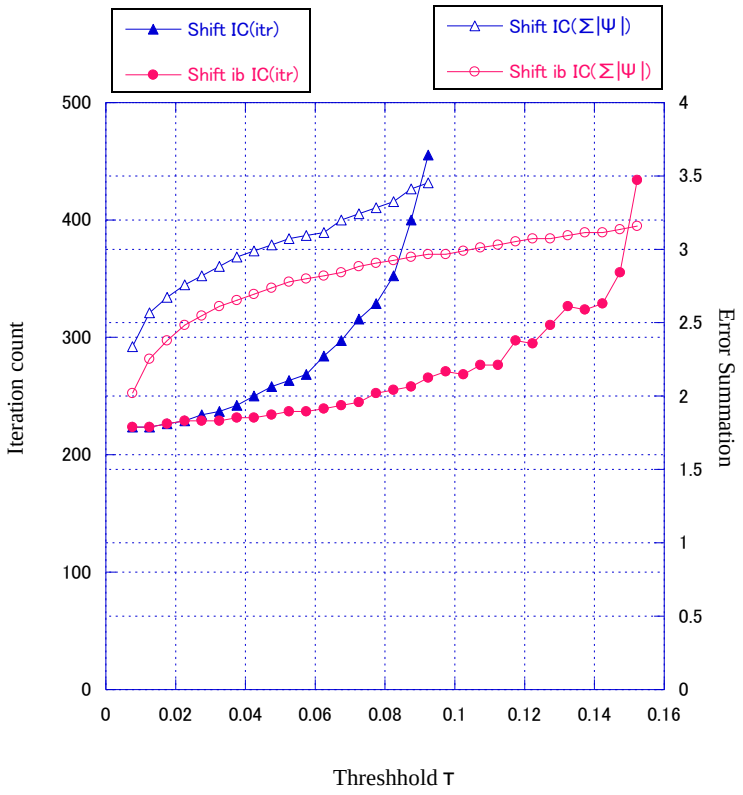


Fig. 3-3 Shifted IC of matrix BCSSTK13・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC.

- 行列 BCSSTK11 では，シフト IC 分解つき CG 法は，τ の変化に対して反復回数の変動は大きく，τ = 0.75 (分解が破綻した τ = 0.8 の前の閾値)では，極端に反復回数が増えた．誤差総和は単調増加ではあるが，シフト ib_IC 分解つき CG 法のそれと比べて，変化率は大きく，また全体的に$10^{0.4}$ (図では 0.4) 程上方にシフトしているように見られた．それに対して，シフト ib_IC 分解つき CG 法は，分解が破綻した τ = 0.115 の前の閾値（τ = 0.11）まで，反復回数および誤差総和の変動は小さく穏やかに推移した．

- 行列 BCSSTK13 では，シフト IC 分解つき CG 法は，τ の変化に対して反復回数は指数関数的に増えた．また，誤差総和の変動については，行列

BCSSTK11 の場合と同様の傾向を示した．それに対して，シフト ib_IC 分解つき CG 法は，τ = 0.145 の時までは反復回数の変動は小さく，緩やかな単調増加であったが，τ = 0.15 では，極端に反復回数が増えた．ただし，τ = 0.15 の誤差総和を見ると前の閾値(τ = 0.145)と比べ大きな変動もなかった．

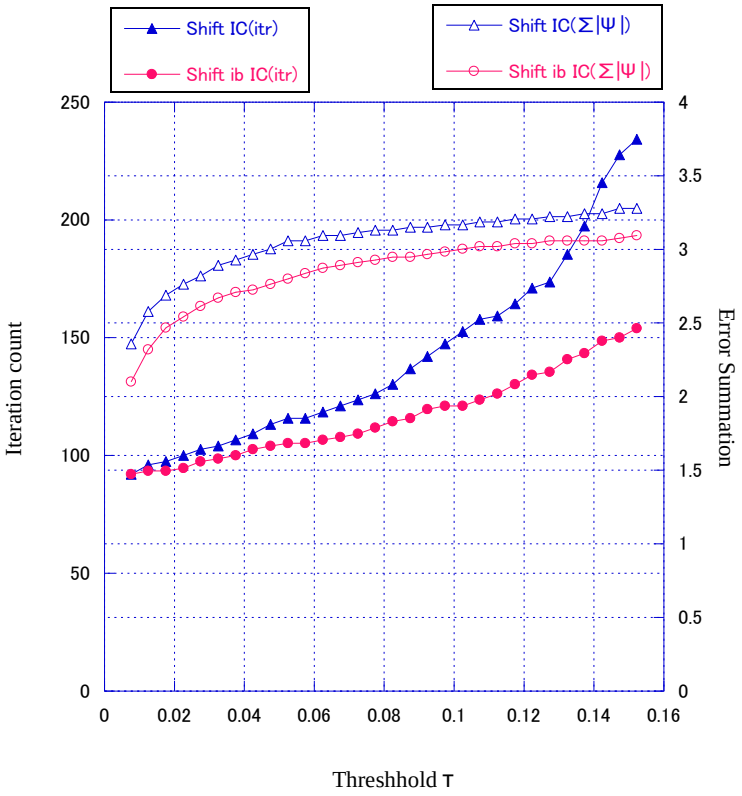


Fig. 3-4 Shifted IC of matrix BCSSTK15・The threshold dependency of iteration counts and error summation of CG method with

行列 BCSSTK15 では，シフト IC 分解つき CG 法は，τ の変化に対して反復回数は単調増加であったが，増加率の緩やかなところもあれば，急なところもあった．また，誤差総和は単調増加で，シフト ib_IC 分解つき CG 法のそれと比べて，変化率はほぼ同じで，全体的に$10^{0.2}$(図では 0:2) 程上方にシフトしているように見られた．それに対して，シフト ib_IC 分解つき CG 法は，τ の変化に対して反復回数は緩やかな単調増加であった．また，τ の変

化に対して誤差総和の変動は小さく穏やかに推移した.

- 行列 BCSSTK15 では，シフト IC 分解つき CG 法は，τ の変化に対して反復回数は単調増加であったが，増加率の緩やかなところもあれば，急なところもあった．また，誤差総和は単調増加で，シフト ib_IC 分解つき CG 法のそれと比べて，変化率はほぼ同じで，全体的に$10^{0.2}$(図では 0:2) 程上方にシフトしているように見られた．それに対して，シフト ib_IC 分解つき CG 法は，τ の変化に対して反復回数は緩やかな単調増加であった．また，τ の変化に対して誤差総和の変動は小さく穏やかに推移した.

次に，行列 S3DKQ4M2 と行列 S3DKT3M2 についてシフト IC 分解つき CG 法とシフト ib_IC 分解つき CG 法の収束性について調べた．ここでは，誤差総和について，シフト IC 分解・シフト ib_IC 分解ともに調べ，計算時間（前処理行列の作成に要した時間と CG 法の反復計算に要した時間の合計)[単位:秒]との相関について調べた. Fig. 3-5 と Fig. 3-6 に，行列 S3DKQ4M2，S3DKT3M2 に対するシフト IC 分解つき CG 法（図中で三角印付きの折れ線グラフ）とシフト ib_IC 分解つき CG 法（図中で丸印付きの折れ線グラフ）の実験結果を各々示す．ただし，三角印，丸印の内部が塗りつぶしてあるものは計算時間，塗りつぶしていないものは誤差総和である．また，図の横軸は閾値 τ の値，左側縦軸は計算時間，右側縦軸は誤差総和（常用対数表示）とし，途中で折れ線が途切れている所は，その閾値で不完全分解が破綻したことを表す．Fig. 3-5 と Fig. 3-6 から以下のことが分かる.

- 行列 S3DKQ4M2 では，シフト IC 分解つき CG 法は，$\tau = 0.05, 0.07, 0.09$ の時計算時間が多くかかった．さらに，$\tau = 0.1$ より大きい時は分解が破綻した．$\tau = 0.03, 0.04, 0.075$ では速く収束した．また，誤差総和を見ると，$\tau = 0.05$ の時その前後の閾値($\tau = 0.045, 0.055$) の時よりも大きくなった．それに対して，シフト ib_IC 分解つき CG 法は，$\tau = 0.115$ の時までは計算時間は穏やかに変化したが，$\tau = 0.12$ の時は再び少なくなり，以降大きな変動はなかった．また，誤差総和の変動は小さく穏やかに推移した.

- 行列 S3DKT3M2 では，シフト IC 分解つき CG 法は，$\tau = 0.09, 0.12, 0.14$で計算時間が多くかかった．さらに，$\tau = 0.125$ と$\tau = 0.130$の時分解が破綻し，$\tau = 0.135$ の時は収束し，収束が非常に不安定であった．誤差総和を見ると，$\tau = 0.09$の時その前後の閾値($\tau = 0.085, 0.095$) の時よりも大きくなった．それに対して，シフト ib_IC 分解つき CG 法は，全般にτの変化に対して計算時間は穏やかに変化し，特に$\tau = 0.04$から$\tau = 0.15$までは計算時間はほとんど変化がなかった．また，誤差総和の変動は小さく穏やかに推移した．

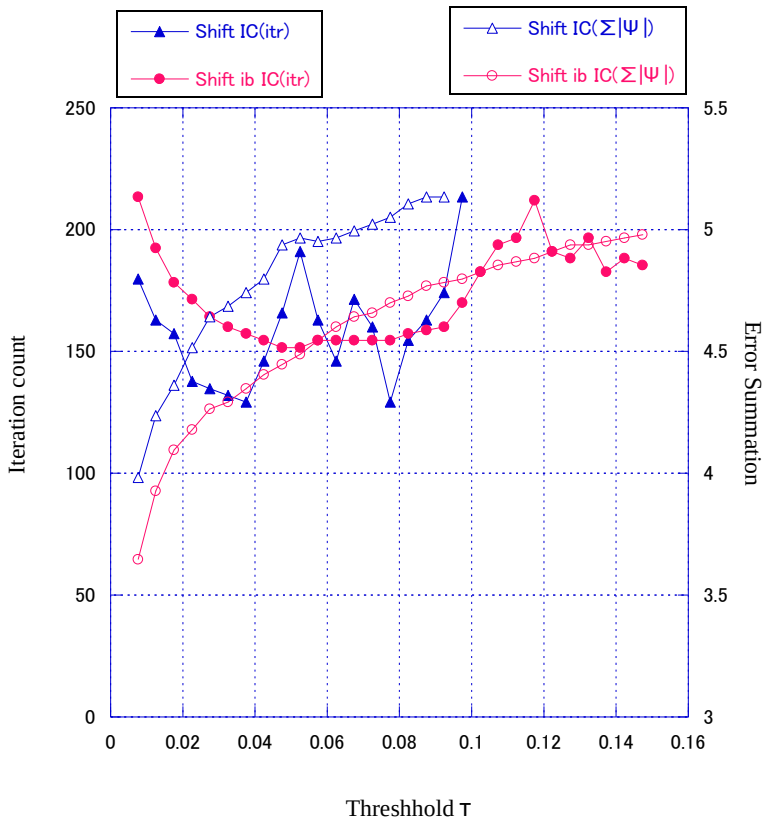


Fig. 3-5 Shifted IC of matrix S3DKQ4M2・The threshold dependency of computing time and error summation of CG method with shifted ib_IC.

(▲・● : The left side vertical axis scale, △・○ : The right side vertical axis scale).

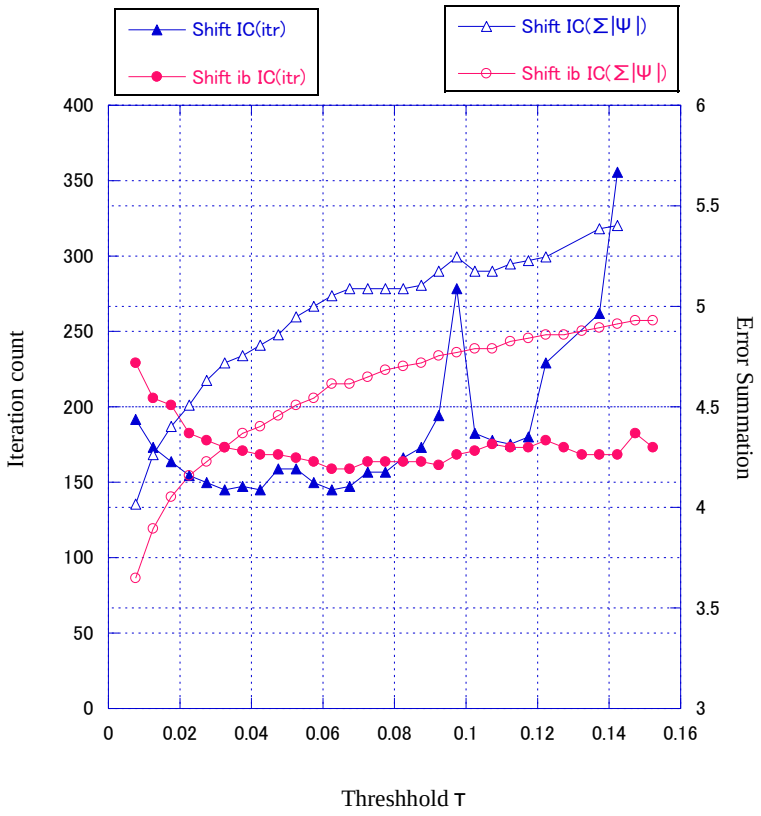


Fig. 3-6 Shifted IC of matrix S3DKT3M2・The threshold dependency of computing time and error summation of CG method with shifted ib_IC.

さらに，実験に用いた 6 種類の行列に対するシフト IC 分解およびシフト ib_IC 分解により生じたフィルイン数について Fig. 3-7，Fig. 3-8 に各々示す．ただし，図の横軸は閾値_の値, 左側縦軸は行列 BCSSTK10，同 11，同 13，そして同 15 に対するフィルイン数[単位: 百万個]，右側縦軸は行列 S3DKQ4M2，S3DKT3M2 に対するフィルイン数[単位：百万個]とし，途中で折れ線が途切れているところは，その閾値で不完全分解が破綻したことを表す.

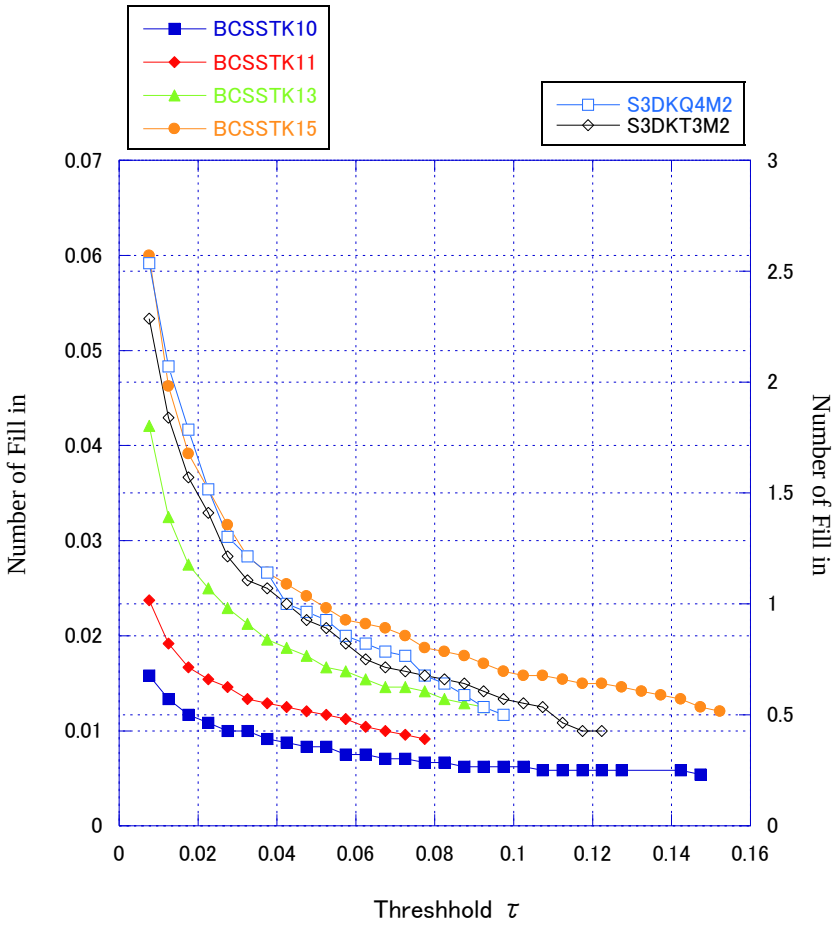


Fig. 3-7 Fill-in number generated from shifted IC of the 6 types of matrices.

(■・◆・▲・● : The left side vertical axis scale, □・◇ : The right side vertical

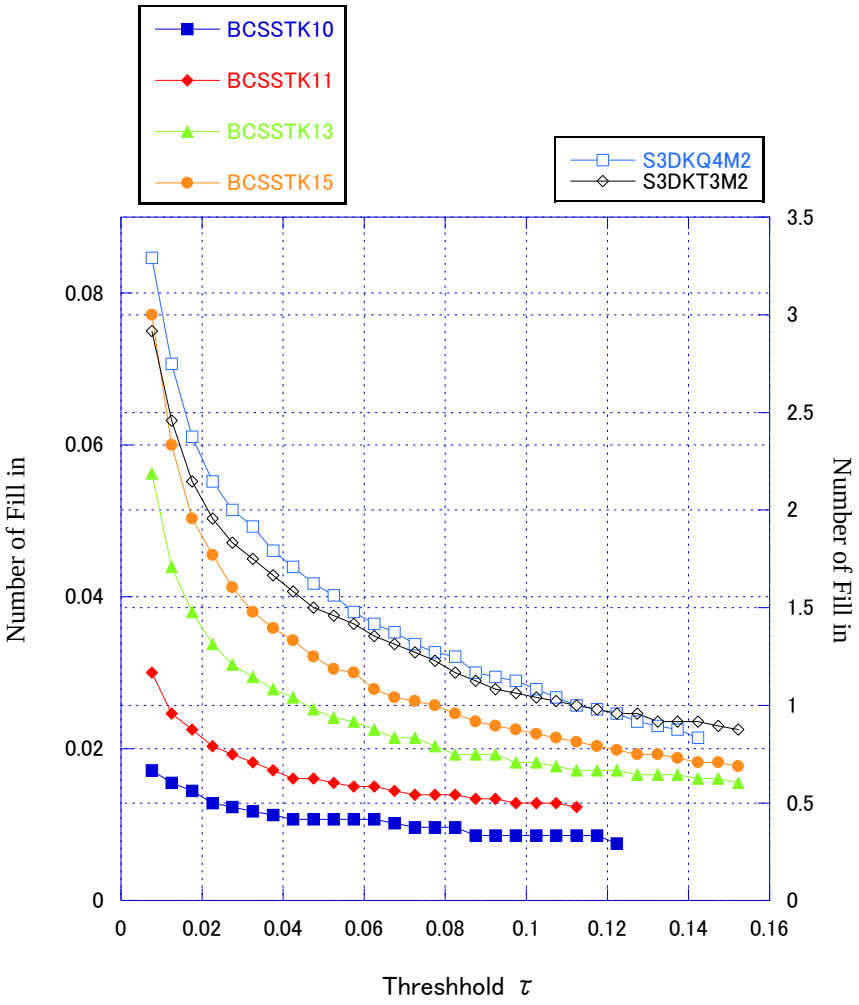


Fig. 3-8 Fill-in number generated from shifted IC of the 6 types of matrices.

これらの図から，シフト ib_IC 分解つき CG 法は，従来のドロッピング手法に基づくシフト IC 分解つき CG 法よりも同じ閾値で比較した時，多くのフィルインが生じ使用メモリ量が多くなることがわかる．また，閾値に対してフィルイン数は単調に減少することがわかる．フィルイン数が多ければ L_k と行列 $\overline{L}_k$ の誤差行列のノルムは小さく，逆にフィルイン数が少なければ L_k と行列 $\overline{L}_k$ の誤差行列のノルムは大きいので，閾値に対して L_k と行列 $\overline{L}_k$ の誤差行列のノルムの大きさは単調に増加すると考えられる．しかし，L_k と行列 $\overline{L}_k$ の誤差行列のノルムが単調に増加すると反復回数が単調に増加する(収束性が悪化する）という保障がないことは数値実験結果から明らかである．

3.6.3 考察

Ib_IC 分解で前処理行列 L の k 列目の生成の時に初めてフィルイン $l_{j,k}$ $(j > k)$が棄却された式(3-8)の場合を想定する．次に k +1 列目でフィルイン $l_{m,k+1}$ $(m > k+1)$ が棄却されたとすると，その時の行列 L_{k+1} は次のように表現できる．

$$
\begin{aligned}
L_{k+1} &= \overline{L}_{k+1} - l_{j,k} e_j e_k^T - l_{m,k+1} e_m e_{k+1}^T \\
&= \overline{L}_{k+1}\left(I - l_{m,k+1} e_m e_{k+1}^T / a_{m,m}\right) - l_{j,k} e_j e_k^T
\end{aligned}
\tag{3-35}
$$

式(3-35)の逆行列を求めるのは現実的ではないので，アルゴリズム 1 では，$\left\| e_m e_{k+1}^T \overline{L}^{-1} \right\|_\infty$ を見積もる場合，式(3-35)中の $l_{j,k} e_j e_k^T$を省略した

$$
L_{k+1} = \overline{L}_{k+1}\left(I - l_{m,k+1} e_m e_{k+1}^T / a_{m,m}\right) \tag{3-36}
$$

の逆行列

$$
\begin{aligned}
L_{k+1}^{-1} &= \left(I - l_{m,k+1} e_m e_{k+1}^T / a_{m,m}\right)^{-1} \overline{L}_{k+1}{}^{-1} \\
&= \overline{L}_{k+1}{}^{-1} + l_{m,k+1} e_m e_{k+1}^T / a_{m,m}
\end{aligned}
\tag{3-37}
$$

をノルム評価の対象とする．したがって，今まで述べてきた誤差行列，および誤差総和は，前処理行列の列生成毎の逆行列のノルムを評価する局所的なものであると考えられる．しかしながら，行列 BCSSTK10，同 11，同 13，そして同 15 の実験結果からは，誤差総和と反復回数に相関が見られ，同様に行列 S3DKQ4M2，S3DKT3M2 の実験結果からは，誤差総和と計算時間に相関が見られることから，局所的な評価ではあるが，不完全分解で生成された L の逆行列L^{-1}と完全分解で生成された$\overline{L}$の逆行列$\overline{L}^{-1}$の誤差行列の大きさを見積もることができる．しかし，シフト ib_IC 分解つき CG 法において Fig. 3-3 の行列 BCSSTK13 に対する実験結果では，$\tau = 0.15$の時極端に反復回数が増える現象が現れた．また Fig. 3-5 の行列 S3DKQ4M2 に対する実験結果では，$\tau = 0.115$の時までは計算時間は穏やかに変化したが，$\tau = 0.12$ の時は少なくなり，以降大きな変動がなくなるという現象が現れた．これらの場合は，前述したように逆行列のノルム評価が局所的であるため，真の逆行列のノルム値との誤

差が大きくなってしまったからと考えられる.

3.6.4 2 種類の IC 分解の収束速度と残差について

従来の IC 分解と逆行列ベースの IC 分解の収束速度を比較した場合，最短時間だけを見ると従来の IC 分解の方が僅かに勝っていることが多い．しかし，最適パラメータ以外の時では，従来の IC 分解の収束速度のバラツキや変動幅は非常に大きく，場合によっては収束しないことや収束するパラメータの区間もだいぶ狭いことなどがわかった.

3.7 第 3 章のまとめと今後の課題

分解過程で発生するフィルインを棄却するドロッピング操作が前処理行列の逆行列U^{-1}に与える影響を見積もり，すなわち，U^{-T} の行のノルム(U^{-1}の列のノルム) をできるだけ正確に見積もり，それに基づきドロッピングする対称行列用の ib_IC 分解を導出した．数値実験を通して，ib_IC 分解付き CG 法の収束は，従来のドロッピング手法に基づく IC 分解付き CG 法の収束と比較して，閾値の変化に対して安定であることがわかった．その収束の安定性は，ib_IC 分解が誤差総和の大きさを抑制するところに起因していると考えられる.
今後の課題は，ib_IC 分解で分解の破綻と収束性との関係，同じくシフト量 α と収束性との関係について調べることなどが上げられる.

第4章　行列ベクトル積の並列計算機用のブロック交換法

本章では, 行列ベクトル積の並列アーキテクチャ用のブロック交換法 BLEX を提案する. 4.1 節では本章の検討課題の導入を行う．4.2 節では，BLEX の詳細について述べる．提案法は, 並列計算機上の疎行列ベクトル積の高速化を行うために 2 種類の方法を用いている. 次いで 4.3 節では，数値実験を通して提案法の効果について評価する．4.4 節では本章の検討結果をまとめる.

4.1 はじめに

並列計算機上の効率的な行列ベクトル積の問題を考える. 周知のように, 行列ベクトル積は線形方程式を反復法で解く時によく用いられる．そしてその効率的な計算実行は重要な課題である. 特に, 行列ベクトル積は並列計算機上で線形方程式を解く時の計算において大きな部分を占める [50] [58]． 行列ベクトル積の高速計算については多くの研究が発表されている [14] [31] [64] [65]．計算の速度は，それぞれのプロセスにおける行列の均等に分割された非零行列成分に大きく依存する.

計算を速度は，行列の非零行列成分を並列計算機上の OpenMP スレッドにどのように割り当てるかに大きく依存する.

一般的に Block Cyclic (BC)分割 [37]は非零行列成分を均等に分割する点においては効果的な手段である．しかし，BC 分割における並列性能は，扱うブロックの数が変わることで大きく変化することは有名である．それゆえ，BC 分割においては，最適なブロックの数を決めることが難しい.

本章では, ブロック交換法によって, 行列の非零要素を均等に並列計算機上のスレッドに割り当てる方法を提案し, この方法を BLEX 法(BLocking Exchange for evenly distributed nonzero entries of matrix)と呼ぶことにする. 本章では, 通常の BC 分割に基づいて, BLEX

法を実現するにあたり，2 種の戦略を採用した．第一の戦略は，行列の非零要素をスレッド毎にほぼ均等に分割することである．第二の戦略は，ブロック交換法を用いることで，さらに均等な分割を行うことである．結果として，BLEX 法は BC 分割と伴に行列の非零要素を分散したスレッドに非常に均等に割り当てることを可能とした．

4.2 従来の分割方法

疎行列の非零要素は圧縮された記憶行列 Compressed Row Storage (CRS) [12] で記憶されているものとする．論文中の行列ベクトル積の計算プログラムは Fortran 90 [9]で表された擬似コードで記述した．ここで，行列 A は疎行列である．このケースでは，非零行列成分の計算方法を考えるにあたり，ブロックと BC 分割は単純な分割であるとする．下記に，ブロックと BC 分割についての概略を述べる．ncol は行列のサイズ，rowptr と colind，val はそれぞれ，各列の始点の配列，各要素の列インデックスと非零行列成分の値を示すものとする．

4.2.1 ブロック分割

ブロック分割においては，1 スレッドあたりの非零行列成分数が同数となるように処理ブロックを分割する．さらに，非零行列成分を分割する際，各ブロックの行列の行数もそれぞれ同じようにする．Fig. 4-1 では，非零要素数 8 の行列と 19 の行列をブロック分割する 2 つの例を示している．このケースでは，各ブロックに含まれる非零行列成分の数の違いは 3 個である．ここで，スレッド数を nth とし，ブロックの数を nblk とする．ブロック分割においては，nblk=nth とする．各ブロックの最初の行を記憶している配列 bst を生成する擬似プログラムを示す．

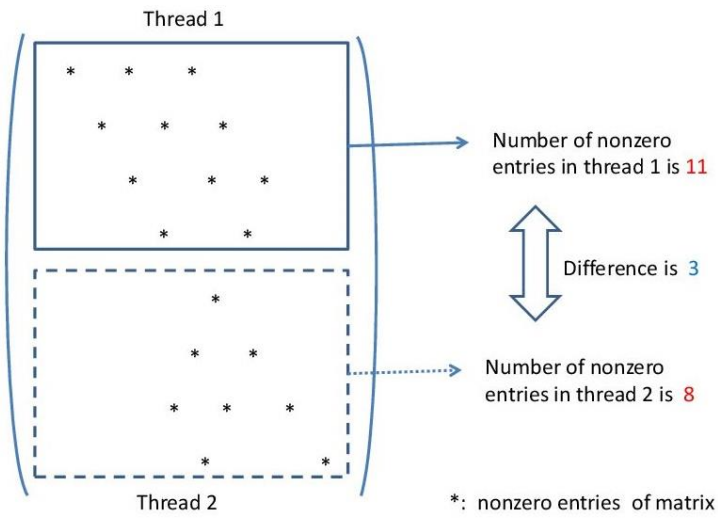


Fig. 4-1 An example of block distribution in case of two threads.

4.2.2 ブロック巡回分割

スレッドに割り当てられる block-id を asb(assigned block)に格納する．すなわち，i 番目のスレッドに割り当てられる j 番目のブロックの block-id を配列 asb (i,j)に格納する．以下のように BC 分割において asb 配列を作成する．以下に，BC 分割における OpenMP ライブラリ[4][9]による行列ベクトル積の並列化版を示す．bst 配列は各ブロックの開始列の行番号を格納する．ncol は行列のサイズを示し，rowptr，colind，val はそれぞれ各行の始点の配列，各要素の列インデックス，非零行列成分の値を示す．omp parallel do は，スレッド並列化を行う OpenMP ライブラリの指示行を意味している．Fig. 4-2 において，スレッドが 2 スレッドある場合の，BC 分割の例を示す．スレッド 1 の非零行列成分の数は 9，スレッド 2 の非零行列成分の数は 10 である．このケースにおいては，各ブロックに含まれる非零行列成分の数の違いは 1 だけである．

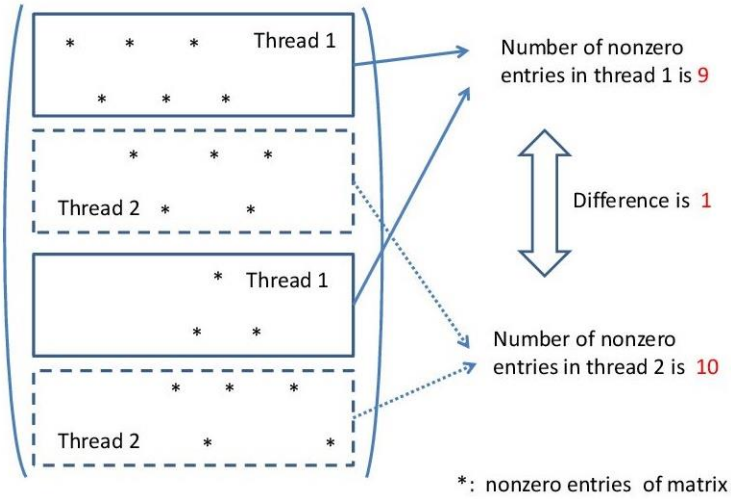


Fig. 4-2 An example of the BC distribution in case of two threads.

4.3 BLEX 法

本節では，BLEX 法を提案する．BLEX 法とは等しく割り当てられた行列の非零行列成分に関するブロック交換技術を意味する．また，BLEX 法は BC 分割に基づいて作成され，以下の二つの戦略を採用する．

1. 自動的にブロックの数を決定する．
2. ブロック交換技術を用いて行列の非零成分を均等に割り当てる．

4.3.1 ブロック数の自動決定

自動的にブロックの数を決定するために，指標 *Wnnt (Width of nnt)*を導入する．

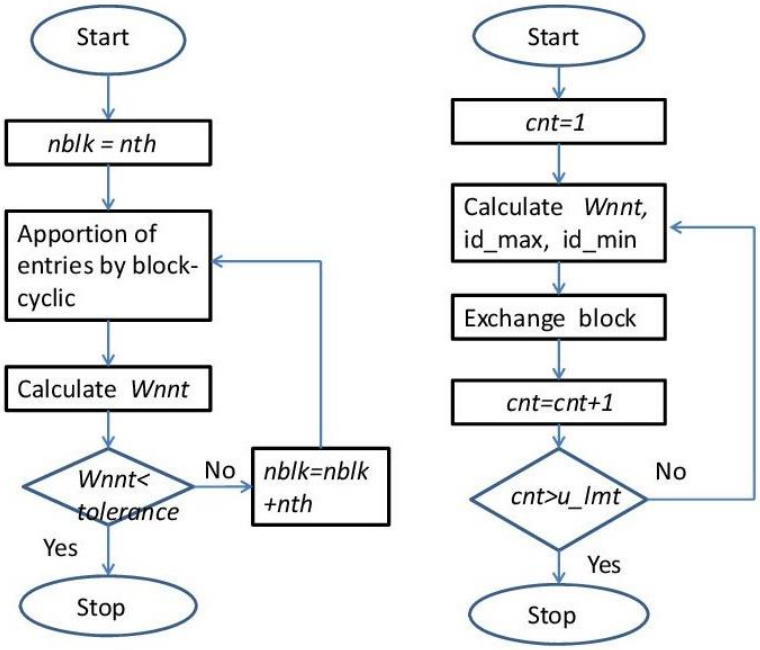
$$Wnnt := \max_{i\ in\ thread}\left(nnt(i)\right) - \min\left(nnt(i)\right) \qquad (1 \le i \le nth)$$

ここで，*nth* は，thread 数を示し，*nnt* は，thread あたりの非零行列成分数を示す．Fig.3(a)に自動的に thread あたりのブロック数を計算するアルゴリズムを示す．Fig.3 にお

いて，*nblk* は，ブロック数を示す．初めに，*nblk* は *nth* に初期化される．次に，指標 *Wnnt* を計算し，Wnnt < *tolerance* となるか否かを調べる．もし，Wnnt< *tolerance* の場合は，nblk は nth に決定とする．一方，*Wnnt*≧ *tolerance* の場合は，*nblk* を *nth* 増やす．*Wnnt*< *tolerance* となるまで，*nblk* は *nth* ずつ増加させる．

4.3.2 ブロック交換技術による均一な非零行列成分の割り当て

Fig. 4-3(b)に，ブロックを交換するアルゴリズムを示す．Fig. 4-3 において，*u_lmt* は，交換するブロックの数の上限を示している．id_max は，非零行列成分を最も割り当てられている thread ID を，id_min は非零行列成分が一番少ない処理の ID を示している．Fig. 4-3 (b)に，初めは *cnt* は 1 に初期化されている．次に *Wnnt* と id_max，id min を計算する．さらに，id max を割り当てられているブロック，id min を割り当てられているブロックを交換する．*cnt* を 1 ずつ増やしていき，*cnt* ＞ *u_lmt* となった時点で，交換は終了する．



(a) Determination of number of blocks

(b) Exchanging blocks

Fig. 4-3 Algorithm to automatically compute the number of block per thread.

4.4 数値実験

本節では，BC 分割と BLEX 法の数値実験について議論する．全ての計算は FUJITSU PRIMEQUEST 580(クロック周波数：1.6GHz)上で，倍精度の浮動小数点演算において行なわれた．FUJITSU の最適化 Fortran90 コンパイラーと，-Kfast, OMP のコンパイルオプションを使った．全てのプログラムを OpenMP ライブラリを用いて実装した．thread 数は 1，2，4，8，16，24，32，48，64 とした．BLEX 法の tolerance パラメータを 10000 とし実行し，u_lmt は thread 数と同じとする．4 つのテスト行列を Florida Sparse Matrix Collection [56]から選択した．テスト行列の性質を Table 4-1 に示す．この Table 4-1 において，nnz というのは非零行列成分の数，ave_nnz というのは 1 つの列における非零行列成分の数を示す．さらに，ave_nnz8 は 8 スレッド毎の全非零行列成分の平均数を表す．

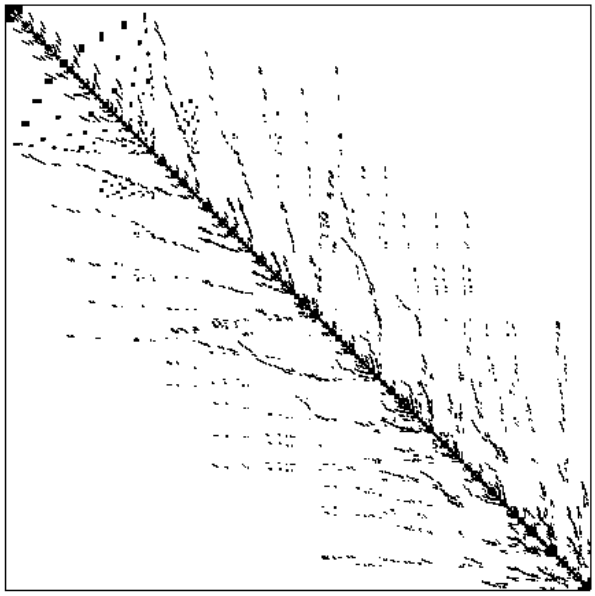
Table 4-1 The description of test matrices

matrix	dimension	*nnz*	*ave_nnz*	*ave_nnz8*
cage14	1,505,785	27,130,349	18.02	3,391,294
language	399,130	1,216,334	3.04	152,041
poisson3Db	85,623	2,374,949	27.74	296,869
sme3Dc	42,930	3,148,656	73.34	393,582

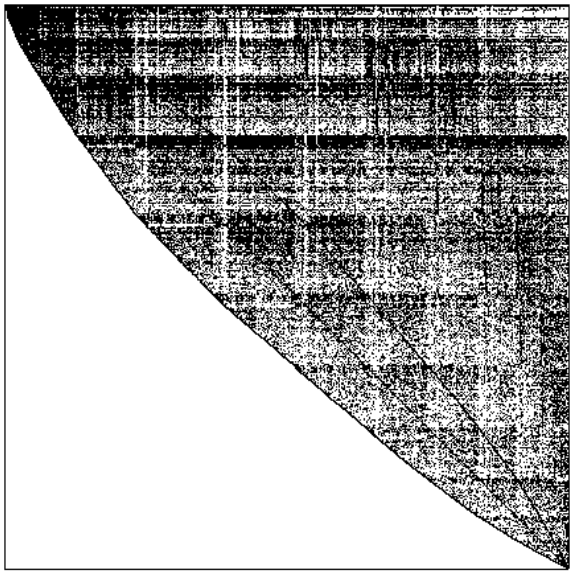
Fig. 4-4 は選択した 4 種の行列の構造を示す．つまり，Fig. 4-4 は行列における行列の非零要素を図示している．Fig. 4-4 より，下の行ほど，テスト行列 ”language” の非零行列成分の数が減っていることが分かる．それゆえ，BC 分割を採用した場合，テスト行列 ”language” に十分な非零行列成分を均一に割り当てることは難しくなる．また，行列 cage14 の非零行列成分の数は大変多いため，行列 cage14 の非零行列成分を割り当てることも難しくなる．

スレッド数が 8 と 64 の場合の非零行列成分の最小と最大の差異を Table 4-2 に示す．スレッド数が 8 と 64 の場合，BLEX 法における差は BC 分割における差よりもかなり小さい

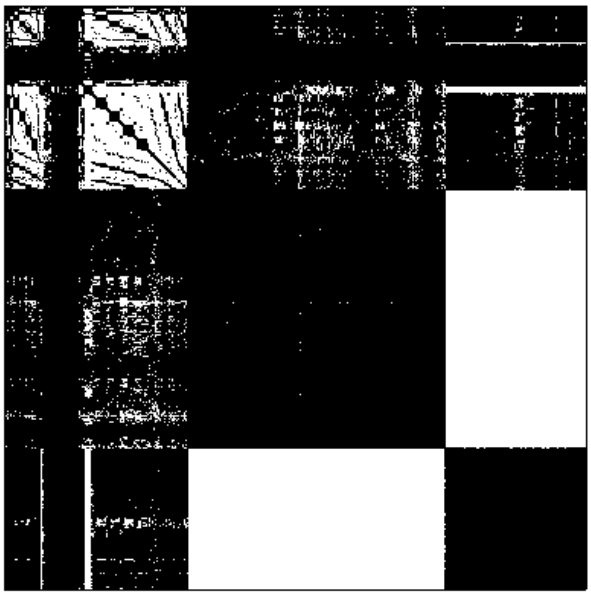
ことがわかる.



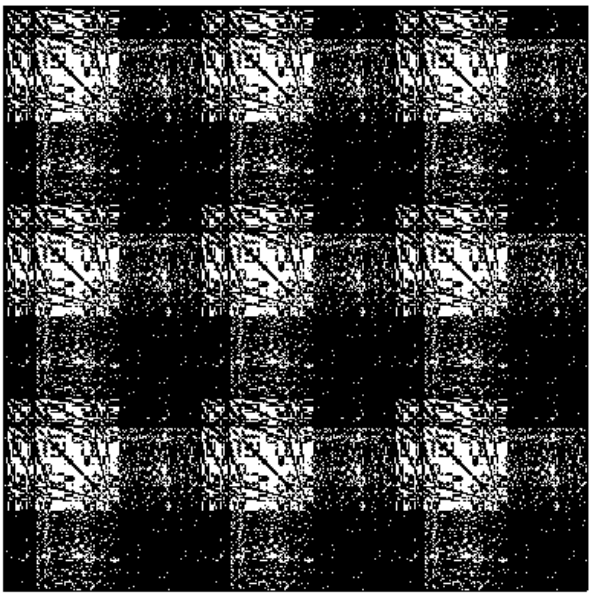
(a) cage14



(b) language



(c) poisson3Db



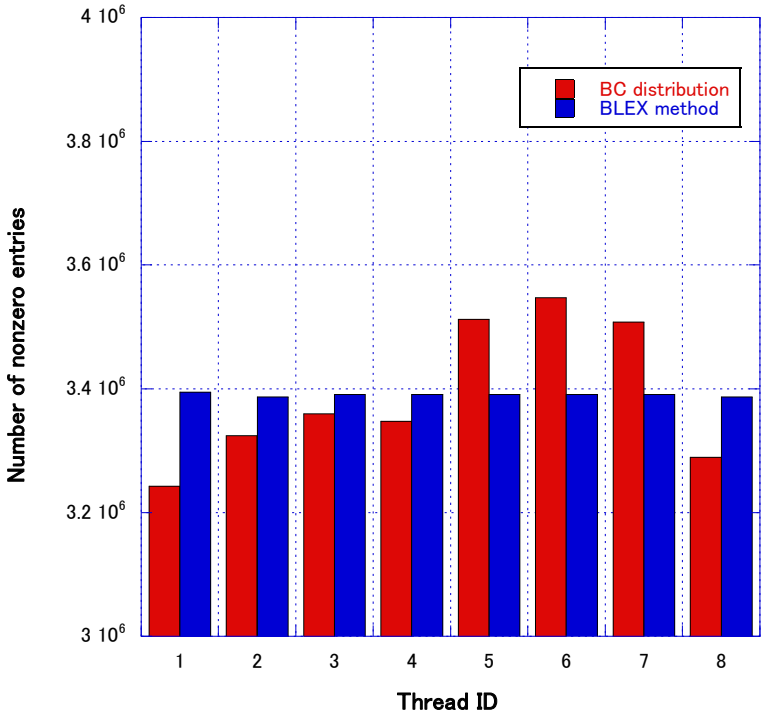
(d) sme3Dc

Fig. 4-4 Pattern of nonzero entries of four matrices.

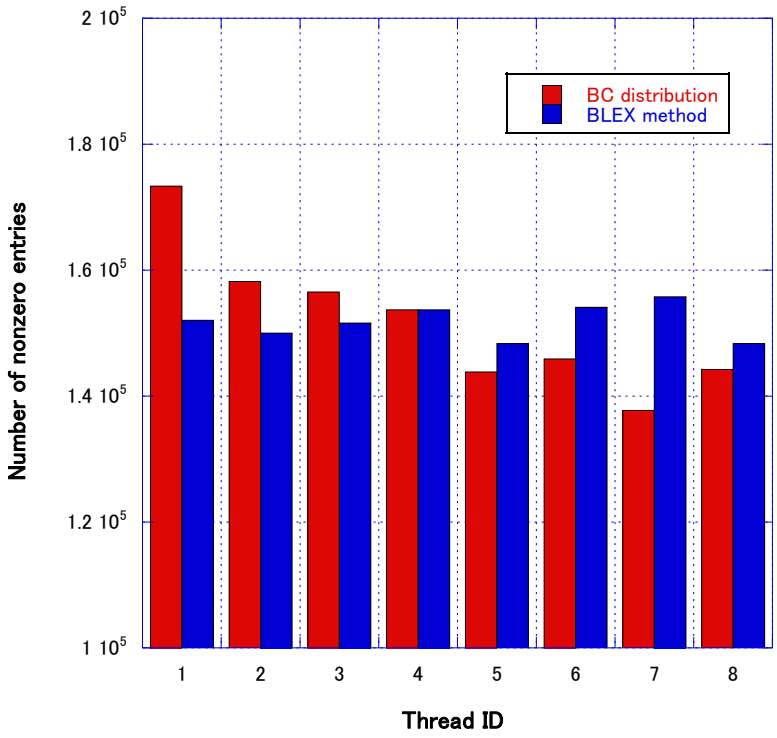
Table 4-2 Difference between minimum and maximum.

No.	matrix	ave. nnz	(a)diff. of BC	(b)diff. of BLEX	ratio(=(b)/(a))
8	cage14	3,391,294	303,593 (8.95%)	7,028 (0.21%)	1/42.6
	language	152,041	35,626 (23.4%)	8,121 (5.34%)	1/4.38
	poisson3Db	296,869	19,088 (6.43%)	1,511 (0.51%)	1/12.6
	sme3Dc	393,582	89,719 (22.8%)	7,923 (2.01%)	1/11.3
64	cage14	423,911	173,813 (41.0%)	8,942 (2.11%)	1/19.4
	language	19,005	13,099 (68.9%)	9,362 (49.3%)	1/1.40
	poisson3Db	37,109	13,636 (36.7%)	9,654 (26.0%)	1/1.41
	sme3Dc	49,198	61,864 (125.7%)	6,638 (13.5%)	1/9.32

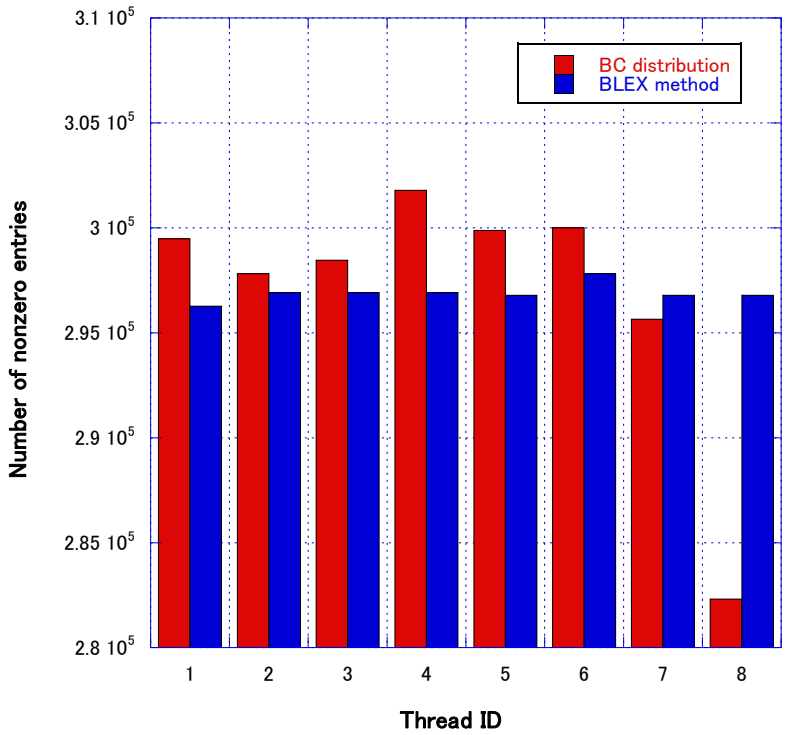
Fig. 4-5 はスレッド数が 8 の場合の 4 種の行列の非零行列成分の数を示している．黒いヒストグラムは BC 分割を採用した時の非零行列成分の数を，ヒストグラムは BLEX 法を採用した時の非零行列成分数を示している．BC 分割のブロック数は BLEX 法の時と同一にしてある．Fig. 4-5 は BLEX 法がいかにスレッドあたりの非零行列成分の割り当てを均等に行えるかということを示している．



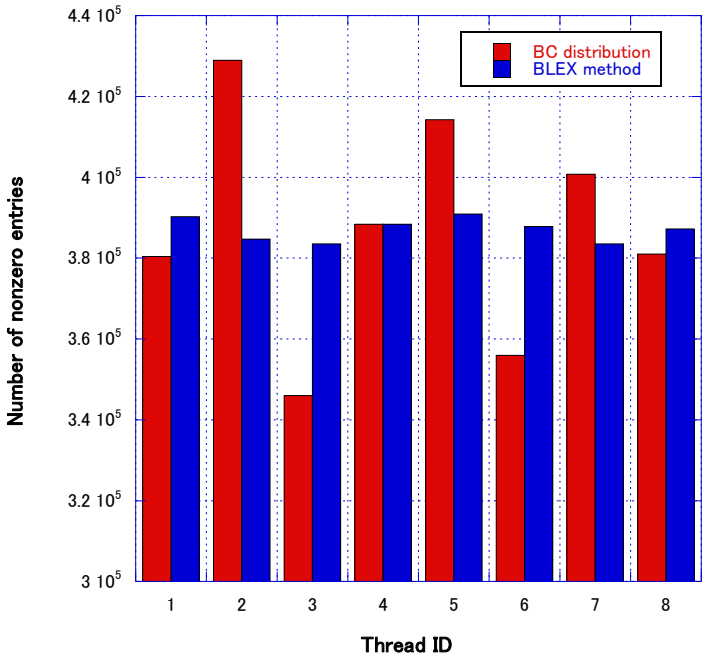
(a) matrix: cage14



(b) matrix: language



(c) poisson3Db



(d) Sme3Dc

Fig. 4-5 Number of nonzero entries of four matrices for each thread when number of thread is eight.

4.4.1 BLEX 法における計算コスト

D' をブロック交換した後の非零行列成分の最小と最大の差と定義する．m を最小非零行列成分数時の交換 block-id と定義し，M を最小非零行列成分数時の交換 block-id と定義する．さらに，bnz(m)を block-id が m の block の非零行列成分の数，bnz(M) を block-id が m の block の非零行列成分の数と，それぞれ定義する．したがって，D' は次のように記述することができる．

$$D' = \left| W_{nnt} - 2\left(bnz(M) - bnz(m) \right) \right|$$

つまり，初めにあらゆるブロック交換の組み合わせについて差 D' を計算し，次に差 D' に関して非ゼロ数が最小になるようにブロックを交換する．例えば，各スレッドに含まれるブ

ロックの数が 100 の場合，ブロック交換組み合わせ数は高々100×100 = 10, 000 通りと予測できる．次に，差 D' を 10000 回計算することとなる．それゆえ，これらの処理は高速に実行可能である．結果として，行列の非零要素を reordering する必要がないため，BLEX 法のコストは全く高くないと予想できる．

2 種類の実験を行った．最初の実験は BC 分割と BLEX 法に関する行列ベクトル積の並列計算の性能評価である．

4.4.2 ブロック巡回分割ならびに BLEX 法における行列ベクトル積の並列計算性能推定

本節では，ブロック巡回分割と BLEX 法に関して，行列ベクトル積の実行結果を示す．行列ベクトル積算については 1000 回の試行を行い結果を示した．Table 4-3 は BLEX 法の性能を示している．Table 4-3 において，nth は，スレッド数を，nblk はブロック数を示している．太字の数字は BC 分割と BLEX 法の最小合計計算時間を示している．

Table 4-3 Comparison of performance of paralleled Matrix-vector multiplication with the BC distributionand the BLEX

(a) matrix: cage14

method	*nth*	*nblk*	*Wnnt*		time[s]		speed-up
				ratio(%)	*Av-t*	total-t	
BC	1	1	0	-	184.598	184.598	1.00
BLEX							
BC	2	10	1,233,727	100	101.987	101.987	1.81
BLEX			6,633	0.54	101.423	**101.423**	1.82
BC	4	28	745,785	100	71.941	71.941	2.56
BLEX			7,435	1	71.606	**71.606**	2.57
BC	8	104	303,593	100	38.499	38.499	4.79
BLEX			7,028	2.31	37.522	**37.522**	4.92
BC	16	96	438,366	100	26.635	26.635	6.93

BLEX			9,191	2.1	25.127	**25.127**	7.34
BC	24	120	321,120	100	18.538	18.538	9.95
BLEX			9,748	3.04	17.974	**17.974**	10.27
BC	32	224	251,881	100	11.688	11.688	15.79
BLEX			8,272	3.28	10.993	**10.993**	16.79
BC	48	288	193,305	100	3.863	3.863	47.78
BLEX			6,300	3.26	3.212	**3.212**	57.47
BC	64	256	173,813	100	2.340	2.340	78.88
BLEX			8,942	5.14	2.138	**2.138**	86.34

(b) matrix: language

method	*nth*	*nblk*	*Wnnt*		time[s]		speed-up
				ratio(%)	*Av-t*	total-t	
BC	1	1	0	-	20.382	20.382	1.00
BLEX							
BC	2	10	######	100	8.964	8.964	2.27
BLEX			2,078	1.85	8.793	**8.793**	2.31
BC	4	28	54,286	100	4.034	4.034	5.05
BLEX			4,147	7.64	4.008	**4.008**	5.08
BC	8	280	35,626	100	2.007	2.007	10.15
BLEX			8,121	22.8	1.964	**1.965**	10.37
BC	16	368	28,770	100	1.046	1.046	19.48
BLEX			9,623	33.45	1.012	**1.013**	20.12
BC	24	600	27,922	100	0.744	0.744	27.39
BLEX			9,453	33.86	0.702	**0.703**	28.99
BC	32	800	21,982	100	0.602	0.602	33.85
BLEX			9,802	44.59	0.554	**0.556**	36.65
BC	48	1008	15,896	100	0.449	0.449	45.39
BLEX			9,916	62.38	0.421	**0.422**	48.29
BC	64	1152	13,099	100	0.391	0.391	52.12
BLEX			9,362	71.47	0.354	**0.355**	57.41

(c) matrix: poisson3Db

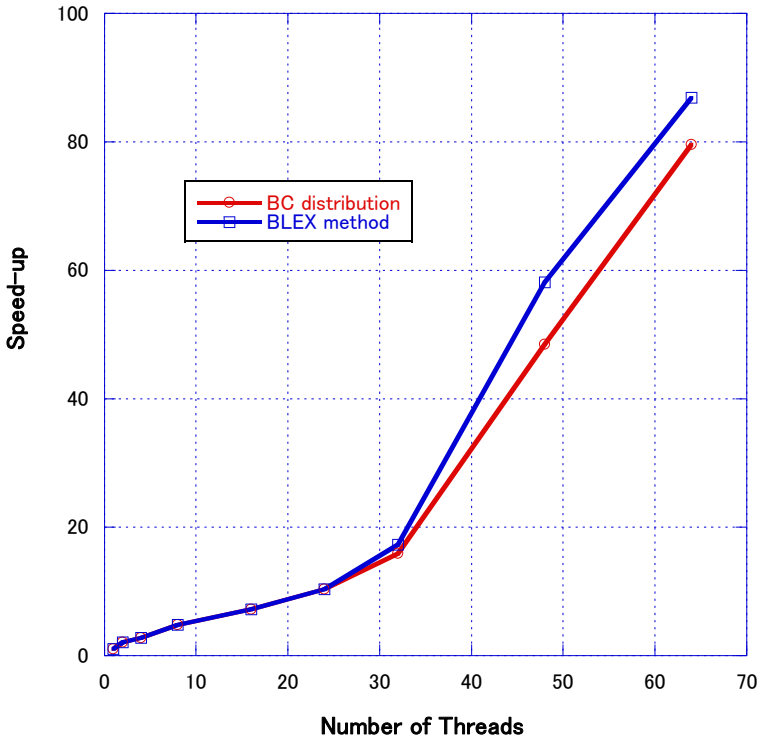
method	*nth*	*nblk*	*Wnnt*		time[s]		speed-up
				ratio(%)	*Av-t*	total-t	
BC	1	1	0	-	14.921	14.921	1.00
BLEX							
BC	2	14	10,581	100.0	6.853	6.853	2.17
BLEX			665	6.28	6.707	**6.707**	2.22
BC	4	64	21,988	100.0	3.036	3.036	4.91
BLEX			1,607	7.31	3.015	**3.015**	4.94
BC	8	128	19,088	100.0	1.527	1.527	9.77
BLEX			1,511	7.92	1.517	**1.517**	9.83
BC	16	368	33,619	100.0	0.833	0.833	17.91
BLEX			7,705	22.92	0.789	**0.787**	18.95
BC	24	216	21,758	100.0	0.573	0.573	26.04
BLEX			5,570	25.56	0.545	**0.542**	27.53
BC	32	256	11,863	100.0	0.434	0.435	34.3
BLEX			9,004	75.9	0.424	**0.424**	35.19
BC	48	384	8,781	100.0	0.318	0.318	46.92
BLEX			6,136	69.88	0.313	**0.313**	47.67
BC	64	384	13,636	100.0	0.296	0.296	50.4
BLEX			9,654	70.8	0.273	**0.273**	54.65

matrix: sme3Dc

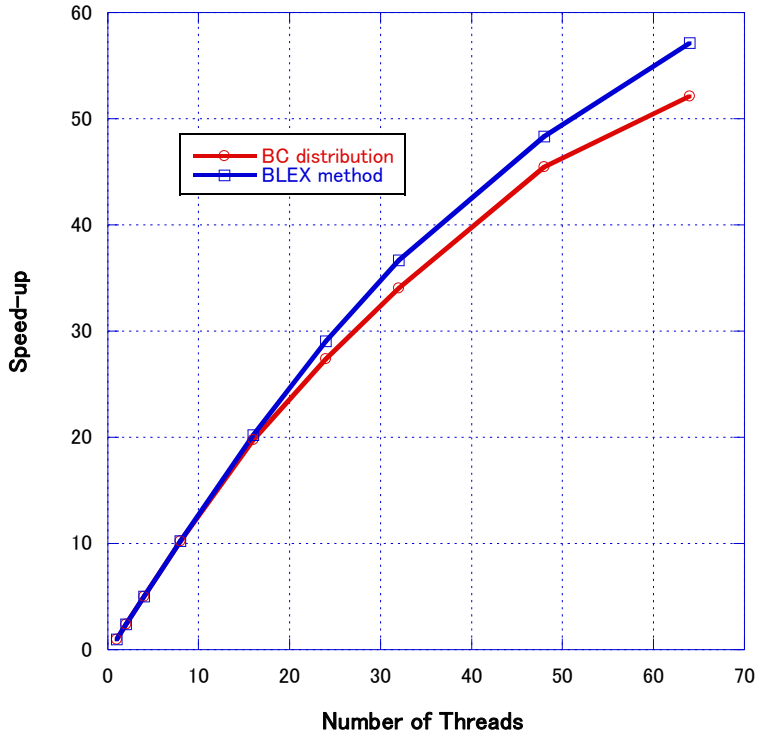
method	*nth*	*nblk*	*Wnnt*		time[s]		speed-up
				ratio(%)	*Av-t*	total-t	
BC	1	1	0	-	13.335	13.335	1.00
BLEX							
BC	2	8	76,358	100	6.719	6.719	1.98
BLEX			6,330	8.29	6.702	**6.702**	1.99
BC	4	20	80,777	100	2.680	2.680	4.97
BLEX			4,839	5.99	2.606	**2.606**	5.11
BC	8	40	89,719	100	1.344	1.344	9.92
BLEX			7,923	8.83	1.291	**1.291**	10.32

BC	16	112	41,814	100.0	0.720	0.720	18.52
BLEX			3,772	9.02	0.654	**0.654**	20.39
BC	24	168	28,798	100.0	0.496	0.496	26.88
BLEX			3,080	10.70	0.449	**0.449**	29.69
BC	32	224	24,769	100.0	0.386	0.386	34.54
BLEX			1,873	7.56	0.348	**0.348**	38.31
BC	48	336	16,990	100.0	0.292	0.292	45.66
BLEX			1,505	8.86	0.263	**0.263**	50.70
BC	64	384	61,864	100.0	0.334	0.334	39.92
BLEX			6,638	10.73	0.240	0.240	55.56

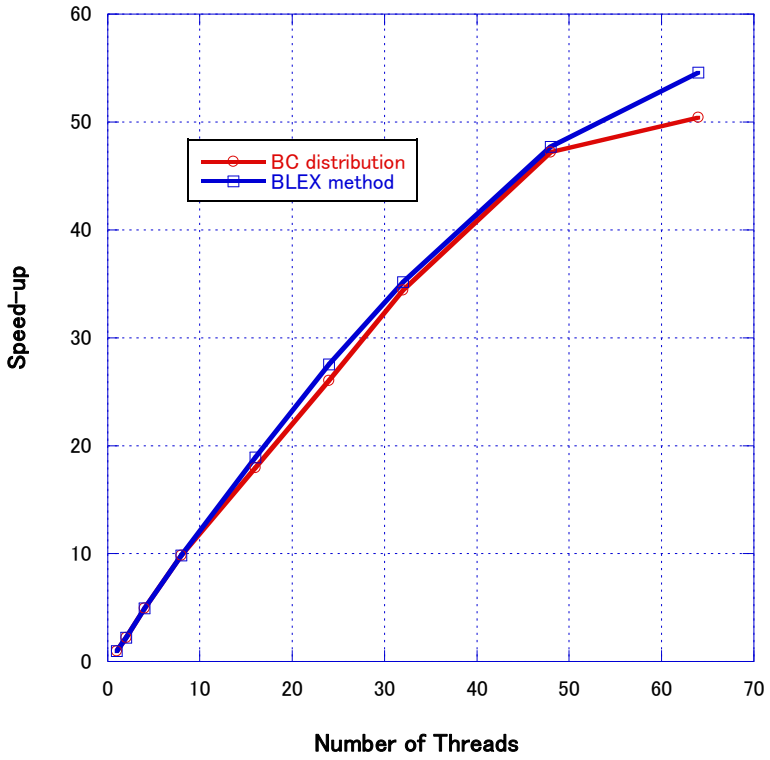
Table 4-3 から，BLEX 法の Wnnt は BC 分割のそれよりかなり小さく，全スレッド数において BLEX 法の計算時間は BC 分割よりも短いことがわかる．特に，BLEX 法はスレッド数が 32 より大きくなった場合，有効なことがわかる．さらに，BLEX 法は行列 sme3Dc においても大変効果的だと結論づけることができる．Fig. 4-6 (a)-(d) は，BC 分割と BLEX 法の行列ベクトル積の並列計算時の処理速度向上の度合いを示している. Fig. 4-6 (a)-(d) においては，破線が BC 分割を示し，実線が BLEX 法を示している．Fig. 4-6 (a)-(d) から，BLEX 法の並列処理による処理速度向上は BC 分割のそれよりも大きいことが分かる. Fig. 4-7 は，スレッド数が変化した時の *Wnnt* の割合の傾向を示している．4 つの行列の *Wnnt* の割合は BC 分割と比べると大変低いことが分かる.



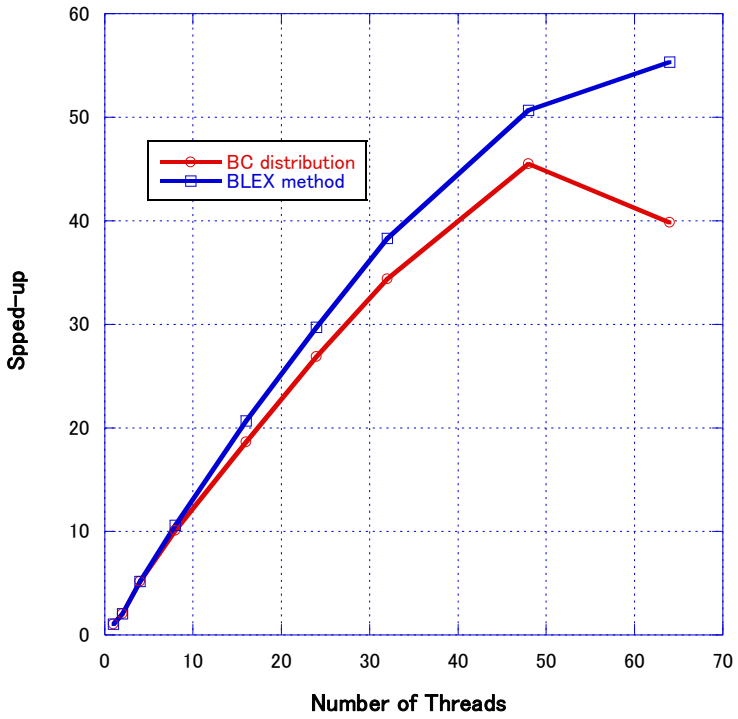
(a) matrix: cage14



(b) matrix: language



(c) matrix: poisson3Db



(d) matrix: sme3Dc

Fig. 4-6 Comparison of speed-up of parallel Matrix-vector multiplication with the BC distribution and the BLEX method.

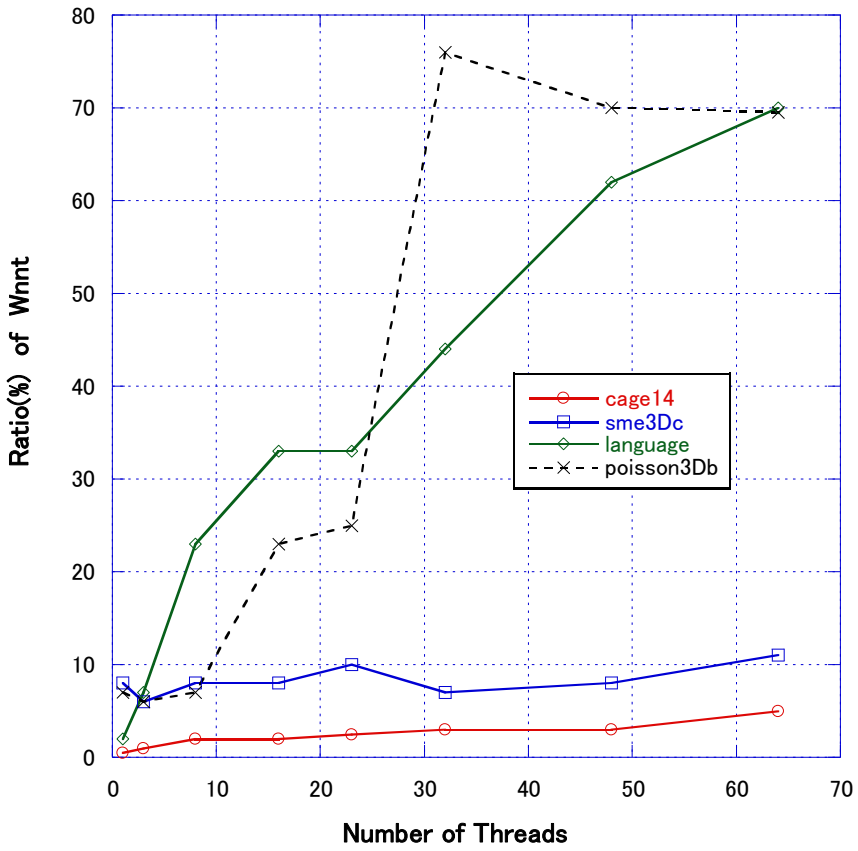


Fig. 4-7 Tendency of ratio(%) of Wnnt when the thread number changes.

4.5 第 4 章のまとめ

本章では，行列の非零要素を均等に分布するブロック交換技術(Blocking Exchange technique)を提案した．さらに，BLEX 法を用いた並列行列ベクトル積の性能を評価した．結果として，BLEX 法は従来の BC 分割法よりも均等に行列の非零成分を分布させることができることが判明した．さらに，BLEX 法を用いた並列行列ベクトル積演算は BC 分割法

を用いた場合よりも高速に計算できることが判明した.

第5章　結言

第五章では本論文の結言として本研究を総括する．5.1 節では本研究の成果を要約して結論とする． 5.2 節では本研究の成果に立脚して次のフェーズとして超並列アーキテクチャに関する学術上の課題と展望を与えて本論文の結語とする

5.1 本研究の結論

本研究では超並列アーキテクチャを志向した性能評価と行列演算のための高速算法について，3 つの課題を定めて検討した．

第 1 章では科学技術計算向けの CPU と並列計算機アーキテクチャの技術的な変遷について概説し，Exascale 以降の Capability Computing の方向性についても示した．またこれらに関連して，設計時に必要なアルゴリズム実行時の並列計算機における性能予測技術の不足と，需要が高い行列反復法において行列演算算法の高速化について指摘した．そこでそれらの解決あるいは緩和により当該分野の産業的発展に資すること，さらには同分野の学術的貢献を行うことを本研究の目的・意義とした．具体的に三項目の検討課題を定め，課題解決方法を概説した．

第 2 章では Chip 内インターコネクトの評価指標として平均 PE 間距離を採用した．新たに平均 PE 間距離導出法を提案することにより，正確な平均 PE 間距離を用いて Chip 内インターコネクトの評価を行った．次に，並列計算機上でのアルゴリズム実行性能を予測する方法を提案した．新たに通信遅延隠蔽閾値 PE 数を定義し，それを用いて実行時間予測法を提案した．さらに，提案法を用いて，FDM による離散化の際に生成される行列に，並列計算機上で CG 法を実行する場合の性能を予測する数値的な方法を提案した．限界実行速度と限界実行速度をもたらす PE 数を理想並列計算機を記述する際に用いたパラメータの関数として解析的に求める方法を新たに提案した．

第 3 章では逆行列ベースの要素棄却型の IC 分解前処理の性能評価を実施した．まず，分解過程で発生するフィルインを棄却するドロッピング操作が前処理行列の逆行列に与える

影響を見積もり，それに基づきドロッピングする対称行列用の ib_IC 分解を導出した．数値実験の結果，ib_IC 分解付き CG 法の収束は，従来のドロッピング手法に基づく IC 分解付き CG 法の収束と比較して，閾値の変化に対して安定であることがわかった．

第 4 章では，行列ベクトル積の超並列計算アーキテクチャを志向した行列の非零要素を均等に分布するブロック交換技術，BLEX 法を提案し，並列行列ベクトル積の性能を評価した．その結果，BLEX 法は従来の BC 分割法よりも均等に行列の非零行列成分を分布させることができ，提案法を用いた並列行列ベクトル積演算は BC 分割法を用いた場合よりも高速であることがわかった．

以上を総括して本研究で得られた知見は次の通りである．超並列アーキテクチャ技術の革新と世界各国での国家戦略とも相まって，Top500 [9]によると 2015 年 6 月現在の性能第 1 位は，中国の天河 2 号であり浮動小数点演算性能は 55PFLOPS に達し，300 万 core より構成されている．また，同サイトでの公式予測によると 2020 年には 1e18 FLOPS すなわち 1 Exa FLOPS に到達すると見込まれており，その能力を余すことなく活用することが求められる．そのためには，実行アルゴリズム×インターコネクトトポロジー×CPU（ベクトル処理能力）で構成される超並列計算機上の処理システムの性能予測技術が，並列計算機設計時のアーキテクチャ最適化およびアルゴリズム実行時の実行パラメータ最適化を行う際に最重要な技術となる．本研究ではそのような観点に立脚したアルゴリズム構成法と性能評価の一手法を与えた．この成果は今後の HPC の進展の一路標となりうると結論づける．

5.2 今後の課題と展望

2 章の今後の課題としては，実際の大規模並列計算機を用いての検証実験が考えられる．各演算の性能，インターコネクトの通信特性を実際に計測した値を，本研究によって理論的に導かれた理想並列計算機モデルに代入した各予測性能値と実測値の比較を行うことが考えられる．また，本研究では，FDM により離散化された問題に CG 法を用いた場合について論じたが，問題の対象を有限要素法に離散化された非対称行列の反復法に求解に広げることが考えられる．また，対象とする問題は反復法に限らず，大規模 FFT，分子動力学法など定式化が比較的単純な問題なら多くの分野に拡張可能となる．

3 章の今後の課題は，ib_IC 分解で分解の破綻と収束性との関係，同じくシフト量 α と収束性との関係について調べることなどが上げられる．

参考文献

[1] George Abe and Seiji Fujino, "Maximum Execution Performance of Iterative Methods on Parallel Computers," *Information*, vol. 18, no. 2, pp. 653-663, 2015.

[2] George Abe and Seiji Fujino, "Network Topology dependency of matrix solvers performance on future huge parallel machines," *Proc. of International Kyoto-Forum on Krylov Subspace method*, 2011.

[3] George Abe and Kunio Hane, "The preconditioned conjugate gradient method on the hypercube," *C3P Proceedings of the third conference on Hypercube concurrent computers and applications*, vol. 2, pp. 1676–1686, 1989.

[4] George Abe, Erika Misaki, Hane Kunio, and Mendez Raul, "Implimentation of the Concurrent Scaled Conjugate Gradient Method on the intel iPSC," in *Japanese Supercomputing: Architecture, Algorithms, and Applications*. New York: Springer-Verlag, 1988, pp. 73-79.

[5] George Abe, Yusuke Onoue, and Fujino Seiji, "Block Exchanging Method for Parallel Matrix-vector Multiplication," *Transaction of the Japan Society for Simulation Technology*, vol. 6, no. 4, pp. 53-59, 2014.

[6] George Abe, Makoto Shiode, and Seiji Fujino, "Performance Estimation of IC Factorization Preconditioning with Inverse-based Dropping," *Transaction of the Japan Society for Simulation Technology*, vol. 7, no. 1, pp. 27-35, 2015.

[7] Yuichiro Ajima, Shinj Sumimoto, and Toshiyuki Shimizu, "Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers," *Computer*, vol. 42, no. 11, pp. 36-40, 2009.

[8] Hisa Ando. (2015, Aug.) ISC 2015 - Intel が語った次世代 Xeon Phi「Knights Landing」. [Online]. http://news.mynavi.jp/series/isc2015_xeon_phi/004/

[9] C. Arai, *Introduction of Fortran90.*: Morikita Corp., 1998.

[10] A. H. Baker and et al. (2010, Dec.) Scaling Algebraic Matrix Solvers: On the Road to Exascale. [Online]. https://e-reports-ext.llnl.gov/pdf/460283.pdf

[11] A. H. Baker and et al. (2010, DEC) Scaling Algebraic Matrix Solvers: On the Road to Exascale. [Online]. https://e-reports-ext.llnl.gov/pdf/460283.pdf

[12] B. Barrett and et al, *Templates.*: SIAM, 2000.

[13] L. N. BHUYAN and et al., "Generalized Hypercube and Hyperbus Structures," *IEEE TRANSACTIONS ON COMPUTERS*, vol. C-33, no. 4, pp. 323-333, 1984.

[14] R. H. Bisseling and W. Meesen, "Communication balancing in parallel sparse matrix-vector multiplication," *Electronic Trans. on Numerical Analysis*, vol. 21, pp. 47-65, 2005.

[15] M. Bollhoefer, "A robust ILU with pivoting based on monitoring the growth of the inverse factors," *Linear Algebraand its Applications*, vol. 338, pp. 201-218, 2001.

[16] M. Bollhoefer and Y. Saad, "Multilevel Preconditioners Constructed From Inverse-Based ILUs," *SIAM J. Sci. Comput*, vol. 27, pp. 1627-1650, 2006.

[17] R. Chandra and et al., *Parallel programming in OpenMP*.: Morgan Kaufmann Publishers, 2001.

[18] Dong Chen and et al., "The IBM Blue Gene/Q Interconnection Network and Message Unit," *Supercomputing11*, 2011.

[19] M. J. Cordery and et al., "Analysis of Cray XC30 Performance Using Trinity-NERSC-8 Benchmarks and Comparison with Cray XE6 and IBM BG/Q," *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*, vol. 8551, pp. 52-72, 2014.

[20] P. W. Cotes and et al., "Technologies for Exascale systems," *IBM J. RES. & DEV.*, vol. 55, no. 5, pp. 1-12, 2011.

[21] David E. Culler and et al., "A new parallel machine model reflects the critical technology trends understanding parallel computers," *Communications of the ACM*, vol. 39, no. 11, 1996.

[22] William J. Dally, "Performance Analysis of k-ary n-cube Interconnection Network," *IEEE Trans. Comput.*, vol. 39, no. 6, 1990.

[23] W. J. Dally and et. al., "Technology-Driven, Highly-Scalable Dragonfly Topology," *International Symposium on Computer Architecture*, pp. 77-78, 2008.

[24] W. J. Dally and B. Towles, *Principles and Practice of Interconnect Networks.*, 2013.

[25] J. Dongarra. (2012, Nov.) Architecture-Aware Algorithm and Software for Peta and Exascale Computing. [Online]. http://charm.cs.illinois.edu/workshops/charmWorkshop2011/slides/CharmWorkshop2011_keynote_Dongarra.pdf

[26] J. Dongarra. (2012, Nov) Architecture-Aware Algorithm and Software for Peta and Exascale Computing. [Online].

http://charm.cs.illinois.edu/workshops/charmWorkshop2011/slides/CharmWorkshop2011_keynote_Dongarra.pdf

[27] Seiji Fujino, Shohei Kodama, and George Abe, "Estimation of Enhanced Performance of Variable ib_IC(tol） Factorization," *Transaction of the Japan Society for Simulation Technology*, vol. 7, no. 3, pp. 63-66, 2015.

[28] Roger W. Hockney, *The Science of Computer Bench Marking*. Philadelphia: SIAM.

[29] W. R. Hockney and R. C. Jesshope, *Parallel Computers 2: Architecture, Programming and Algorithms*.: Adam Hilger, 1988.

[30] Kai Hwang, *Advanced Computer Architecture for Parallel Processing*.: McGraw-Hill, 1993.

[31] E. J. Im, K. Yelick, and R. Vuduc, "Sparsity: Optimization framework for sparse matrix kernels," *Int. J. of High Performance Computing Applications*, vol. 18, no. 1, pp. 135-158, 2004.

[32] J. A. Kash and et al., "Communication Technologies for Exascale Systems," in *Photonics Packaging, Integration, and Interconnects IX*, vol. 7221, 2009.

[33] R. Kouhia. Sparse Matrices web. [Online]. http://www.hut.fi/kouhia/sparse.html

[34] S. Lee and R. Eigenmann, "Adaptive Runtime Tuning of Parallel Sparse Matrix-Vector Multiplication Distributed Memory Systems," in *The Proc. of the 22nd Annual Int. Conference on Supercomputing*, 2008, pp. 195–204.

[35] N. Li. and Y. Saad., "Crout versions of the ILU factorization with pivoting for sparse symmetric matrices," *ETNA*, vol. 20, pp. 75-85, 2005.

[36] N. Li., Y. Saad, and E. Chow, "Crout version of ILU for general sparse matrices," *SIAM J. Sci. Comput.*, vol. 25, pp. 716-728, 2003.

[37] J. J. Lo and et al., "Tuning Compiler Optimizations for Simultaneous Multithreading," *Int. J. of Parallel Programing*, vol. 27, no. 6, pp. 114-124, 1999.

[38] Junichiro Makino, Kei Hiraki, and Mary Inaba, "GRAPE-DR: 2-Pflops massively-parallel computer with," in *Supercomputing07*, 2007.

[39] T. A. Manteuffel, "An incomplete factorization technique for positive definite linear systems," *Math. of Computation*, vol. 34, pp. 473-497, 1980.

[40] Matrix Market. [Online]. http://math.nist.gov/MatrixMarket/

[41] T. G. Mattson, B. A. Sanders, and B. L. Massinggill, *Pattern for Parallel programing*.: Addison Wesley, 2005.

[42] J. Mayer, "ILUCP: a Crout ILU preconditioner with pivoting," *Numerical Linear*

Algebra with Applications, vol. 12, pp. 941-955, 2005.

[43] Timothy Prickett Morgan. Intel Stacks Knights Landing Chips Next To Xeons. [Online]. http://www.theplatform.net/2015/08/28/intel-stacks-knights-landing-chips-next-to-xeons

[44] Keigo Nitadori, Junichiro Makino, and George Abe, "High-Performance Small-Scale Simulation of Star Clusters Evolution on Cray XD1," *IPSJ*, vol. 48, no. ACS 18, 2007.

[45] (2015, Feb.) PEZY computing in PC クラスターワークショップ２０１５大阪. [Online]. http://www.pccluster.org/ja/event/%E9%BD%8A%E8%97%A4%E6%A7%98.pdf

[46] A. Rafiei and M. Bollhoefer. (2010, Sep.) Extension of inverse-based dropping techniques for ILU preconditioners. [Online]. http://www.digibib.tu-bs.de/?docid=00035624

[47] A Rafiei and F Toutounian, "Breakdown-free version of ILU factorization for nonsymmetric positive definite matrices," *Comput. Appl. Math.*, vol. 230, pp. 699-705, 2009.

[48] R. M. Russell, "The CRAY-1 Computer System," *Comm. ACM*, pp. 63-72, 1978.

[49] Y. Saad, "ILUT: a dual threshold incomplete LU factorization," *Numerical Linear Algebra with Applications*, vol. 1, pp. 387-402, 1994.

[50] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia: SIAM , 2003.

[51] Y. Saad, *Iterative methods for sparse linear systems 2nd edition*. Philadelphia: SIAM, 2003.

[52] D. SANCHEZ and et. al., "An Analysis of On-Chip Interconnection Networks for Large-Scale Chip Multiprocessors," *ACM Transactions on Architecture and Code Optimization*, vol. 7, no. 1, 2010.

[53] ZHIZHANG SHEN, "The Calculation of Average Distance," *Computers and Mathematics with Applications*, vol. 44, pp. 1379-1402, 2002.

[54] A. Shiode and S. Fujino, "Inverse-based dropping for IC decomposition of Conjugate Gradient method," *ICNAAM Extended Abstracts*, pp. 442-445, 2006.

[55] Edgar Solomonik, Abhinav Bhatele, and James Demmel, "Improving commutation performance in dense linear algebra via topology aware collectives," *Technical Report of UCB*, vol. UCB/EECS-2011-92, 2011.

[56] C T. Davis. Univ. of Florida sparse matrix collection. [Online]. http://www.cise.ufl.edu/research/sparse/matrices/index.html

[57] Top500. [Online]. http://www.top500.org/

[58] H. A. van der Vorst, *Iterative Krylov methods for large linear systems.*, 2003.

[59] Wikipedia. IBM Power4. [Online]. https://en.wikipedia.org/wiki/POWER4

[60] wikipedia. IBM system/370. [Online]. https://en.wikipedia.org/wiki/IBM_System/370

[61] Wikipedia. IBM system/370. [Online]. https://en.wikipedia.org/wiki/IBM_System/370

[62] wikipedia. intel 8087. [Online]. https://en.wikipedia.org/wiki/Intel_8087

[63] Wikipedia. intel 8087. [Online]. https://en.wikipedia.org/wiki/Intel_8087

[64] S. Williams and et al., "Optimization of sparse matrix-vector multiplication on emerging multicolor platforms," *Parallel Computing*, vol. 35, pp. 178-194, 2009.

[65] A. N. Yzelman and R. H. Bisseling, "Cacheoblivious sparse matrix-vector multiplication by using sparse matrix partitioning methods," *SIAM, J. on Scientific Computing*, vol. 31, no. 4, pp. 3128-3154, 2009.

[66] 阿部譲司, "ハイパーキューブ・マルチプロセッサ　～インテルiPSCを中心に～," *bit*, vol. 21, no. 5, pp. 22-33, 1989.

[67] 阿部譲司, 赤田圭史, and 小橋航, "Cell/B.E.およびRSXより構成されるクラスタ上でのリアルタイム並列可視化技術," *計算工学講演会論文集*, vol. 14, no. 2, pp. 595-598, 2009.

[68] 伊理正夫, *一般線形代数.*, 2003.

[69] 塩出亮, "対称行列に対するクリロフ部分空間法の新しい IC 分解型前処理の研究," *九州大学大学院システム情報科学府, 修士論文*, 2007.

[70] 塩出亮 and 藤野清次, "Inverse-based ドロッピング手法つき IC 分解について," *環瀬戸内海応用数理研究部会 10 回シンポジウム公園予稿集*, pp. 94-99, 2006.

[71] 奥川峻史, *並列計算機アーキテクチャ.*: コロナ社, 1991.

[72] 似鳥啓吾, 牧野淳一郎, and 阿部譲司, "Cray XD1 での星団進化の高性能「小規模」シミュレーション," *情報処理学会論文誌コンピューティングシステム*, vol. 48, no. 8, pp. 54-61, 2007.

[73] 森正武, *数値解析第 2 版.*, 2002.

[74] 青木繁 and 天谷賢治, "離散逆問題の数値計算," *応用数理*, vol. 10, no. 2, pp. 41-49, 2000.

[75] 藤原牧, "非対称行列に対する不完全分解前処理と反復法の評価," *九州大学大学院システム情報科学府, 修士論文*, 2006.

[76] 藤野清次，阿部邦美，杉原正顕，中嶋徳正, "線形方程式の反復解法," *丸善出版，東京*, 2013.

[77] 名取亮，*線形計算*., 1993.

図目次

Fig. 1-1 HPC Performance Development History 3
Fig. 2-1 Network Topologies：(a) Linear Array (b) Ring (c) 2D Mesh (d) 2D Torus (e) 3D Mesh (f) 3D Torus (g) Hypercube. 3
Fig. 2-2 Communication model of a PE. 4
Fig. 2-3 Definition of distance between PEi and PEj. 5
Fig. 2-4 Combinations of PE pair for communication. 6
Fig. 2-5 Location expression of PE. 7
Fig. 2-6 Distance between PEs (Linear Array) 8
Fig. 2-7 Inter-PE-distance 2D Mesh example. 9
Fig. 2-8 Domain segmentation around $PEij$. 9
Fig. 2-9 Classification of Ring structure. 14
Fig. 2-10 Domain segmentation of 2D Torus. 16
Fig. 2-11 4th Demensional Hypercube Structure. 24
Fig. 2-12 Average Distance of 2D Topology cases 27
Fig. 2-13 Average Distance of 3D Topology cases 28
Fig. 2-14 Comparison of Average Distance 30
Fig. 2-15 Domain definition 31
Fig. 2-16 Three dimensional stencil of FDM 31
Fig. 2-17 **Stencil of 3D FDM.** 33
Fig. 2-18 Generated 3D Mesh structure by FDM. 34
Fig. 2-19 Generated matrix from FDM. 35
Fig.2-20 Algorithm of Conjugate Gradient method. 36
Fig. 2-21 Communication between neighbor PE 37
Fig. 2-22 Domain decomposition using Linear Array processor. 38
Fig. 2-23 Domain decomposition using 2D Mesh processor. 39
Fig. 2-24 Domain decomposition using 2D Mesh processor. 39
Fig.2-25 Mapping method of computational domain to single PE (a) Linear Array, Ring (b) 2D Mesh, 2D Torus (c)3D Mesh, 3D Torus, Hypercube. 39
Fig.2-26 Inner product operation process on parallel computer (a) Reduction operation on Linear Array at LinkMode1 (b) Broadcast operation on Linear Array at LinkMode1 (c) Reduction and Broadcast simultatenous operation on Ring at LinkMode1 (d) Reduction and Broadcast simultatenous operation on Ring at LinkMode2. 41

Fig. 2-27 Pseudo code of parallelized CG method. 44
Fig.2-28 Latency Hiding Threshold PE Number. 48
Fig. 2-29 Relation between PE number and operation time..... 52
Fig. 2-30 Relation between PE number and operation speed..... 53
Fig.2-31 Relation between Pth and Communication Hiding mode(1). 57
Fig. 2-32 Relation between Pth and Communication Hiding. mode(2). 57
Fig.2-33 Number of PEs at Limit Execution Speed ($Popt$). 58
Fig.2-34 Number of PEs at Limit Execution Speed ($Popt$) at $106 \leq n \leq 108$. 59
Fig.2-35 Maximum Processing Speed of CG method (1). 60
Fig. 2-36 Maximum Processing Speed of CG method (2). 61
Fig. 3-1 Shifted IC of matrix BCSSTK10・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC 75
Fig. 3-2 Shifted IC of matrix BCSSTK11・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC. 76
Fig. 3-3 Shifted IC of matrix BCSSTK13・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC. 77
Fig. 4-4 Shifted IC of matrix BCSSTK15・The threshold dependency of iteration counts and error summation of CG method with shifted ib_IC. 78
Fig. 3-5 Shifted IC of matrix S3DKQ4M2・The threshold dependency of computing time and error summation of CG method with shifted ib_IC. 80
Fig. 3-6 Shifted IC of matrix S3DKT3M2・The threshold dependency of computing time and error summation of CG method with shifted ib_IC. 81
Fig. 3-7 Fill-in number generated from shifted IC of the 6 types of matrices. 82
Fig. 3-8 Fill-in number generated from shifted IC of the 6 types of matrices. 83
Fig. 4-1 An example of block distribution in case of two threads..... 88
Fig. 4-2 An example of the BC distribution in case of two threads..... 89
Fig. 4-3 Algorithm to automatically compute the number of block per thread..... 90
Fig. 4-4 Pattern of nonzero entries of four matrices. 93
Fig. 4-5 Number of nonzero entries of four matrices for each thread when number of thread is eight..... 96
Fig. 4-6 Comparison of speed-up of parallel Matrix-vector multiplication with the BC distribution and the BLEX method. 103
Fig. 4-7 Tendency of ratio(%) of Wnnt when the thread number changes. 103

表目次

Table 2-1 Average distance of 1D topologies. 24
Table 2-2 Average distance of 2D toplogies. 25
Table 2-3 Average distance of 3D toplogies. 25
Table 2-4 Communication cost of Matrix Vector Production for single PE. This table means the function $LCLinkMode, Topology, n, p$. 40
Table 2-5 Communication frequency with adjacent PEs in doing one innrer product operation and broadcasting result. This table means the function $GCLinkMode, Topology, p$ 42
Table 2-6 Assumption of computer specification. 47
Table 2-7 Approximated Pth (Latency Hiding Threshold PE Number). 49
Table 2-8 Optimal PE number for No Latency Hiding($PoptNH$) 55
Table 2-9 Optimal PE number for Latency Hiding($PoptH$). 55
Table 2-10 Approximated analytical form of Optimal PE number at Limit Execution Speed. 59
Table 2-11 Approximated analytical form of Limit Execution Speed of CG method execution. 61
Table 3-1 Error matrix in conventional IC decomposition and ib_IC ・target matrix in the case of dropping. 66
Table 3-2 The main properties of the test matrix 73
Table 4-1 The description of test matrices 91
Table 4-2 Difference between minimum and maximum. 93
Table 4-3 Comparison of performance of paralleled Matrix-vector multiplication with the BC distributionand the BLEX method 97

謝辞

本研究を進めるにあたりご指導頂いた，指導教員である九州大学情報基盤研究開発センターの藤野清次教授に心より感謝の意を表する．本論文の副査として，論文の適切な記述の仕方について，貴重なご助言を頂いた九州大学大学院システム情報科学研究院の櫻井幸一教授，九州大学情報基盤研究開発センターの天野浩文准教授に感謝の意を表する．論文執筆にあたり多大な貢献を頂いた藤野研究室 OB の尾上勇介氏，塩出亮氏に特別の感謝の意を表する．本論文作成に際し多忙な時期に時間を割いて頂いた藤野研究室の修士２年岩里洸介君、学部 4 年小玉捷平君に感謝の意を表する．

本研究の基盤である計算科学分野へ導いて頂いた慶應義塾大学大学院時代の指導教官である羽根邦夫元助教授に心よりの感謝の意を表する．

最後に，研究を進めるにあたり協力をしてくれた私の家族に感謝の意を表する．