JNC TN9400 2003-088

燃料集合体内詳細熱流動解析コード SPIRALの並列化

（研究報告）

２００３年３月

核燃料サイクル開発機構
大洗工学センター

〒319-1184　茨城県那珂郡東海村村松 4 番地49
核燃料サイクル開発機構
技術展開部　技術協力課
電話：029-282-1122（代表）
ファックス：029-282-7980
電子メール：jserv@jnc.go.jp

JNC TN9400 2003-088
２００３年３月

燃料集合体詳細熱流動解析コードSPIRALの並列化

(研究報告)

出口　明[1]　大島　宏之[2]

要　旨

実験では計測困難な変形した高燃焼度燃料内の熱流力挙動を数値実験で解明することを目的として、サブチャンネル解析コードと相互補完的に用いる局所詳細熱流動解析コードSPIRALの開発を進めている。本研究は、このSPIRALコードについて、実用炉等大規模解析への適用性を確保するため、コードの並列処理化を行なったものである。

有限要素法をベースとしたSPIRALコードを並列化するにあたり、通信が少なくて済むオーバーラップ要素を用いた領域分割を採用するとともに、並列化プログラミングツールとしてMPIを使用した。

検証計算には３次元のメッシュ数の規模が異なる２ケースのデータを用い、並列化前の解析結果と一致することを確認するとともに、最大16プロセスまで計算時間と経過時間について測定し、並列化効率を評価した。経過時間ベースでの比較において、小規模の体系では並列化処理のオーバーヘッドの影響が大きく、並列化効率向上はあまり得られなかったが、大規模体系においては16プロセスで約12倍の処理速度という効率が得られた。

１：有限会社イーエヌオー数理解析リサーチ
２：大洗工学センター　要素技術開発部　流体計算工学研究グループ

JNC TN9400 2003-088
March. 2 0 0 3

Parallelization of Detailed Thermal-Hydraulic Analysis Program SPIRAL

A. Deguchi[1] and H. Ohshima[2]

ABSTRACT

The detailed thermal-hydraulic analysis computer program SPIRAL is under development for the evaluation of local flow and temperature fields in wire-wrapped fuel pin bundles deformed by the influence of high burn-up, which are hard to reveal by experiment due to measurement difficulty. The coupling utilization of this program and a subchannel analysis program can offer a practical method to evaluate thermal-hydraulic behavior in a whole fuel assembly with high accuracy. This report describes the parallelization of SPIRAL for improving applicability to larger numerical simulations.

The domain decomposition method using overlapped elements was adopted to the parallelization because SPIRAL is based on finite element method and it can minimize the number of communications between processor elements. As a parallelization programming library, Massage Passing Interface (MPI) was applied.

Several numerical simulations were carried out to verify the parallelized version of SPIRAL and to evaluate parallelization efficiency. From these simulation results, the validity of this version was confirmed. Although no good parallelization efficiency was obtained in the case of small scale simulations due to overhead processes, approximately twelve times processing speed was acheieved by using 16 processor elements in larger scale simulations.

1: ENO Numerical Analysis Reseach Lab
2: Thermal-Hydraulic Research Group, Advanced Technology Division, O-arai Engineering Center, Japan Nuclear Cycle Development Institute

目　次

表リスト

図リスト

1. 概 要

（1）目的

流体計算工学研究グループでは、実験では計測困難な変形した高燃焼度燃料内の熱流力挙動を数値実験で解明することを目的として、サブチャンネル解析コードと相互補完的に用いる局所詳細熱流動解析コード SPIRAL【1】の開発を進めている。本件は、この SPIRAL コードについて、実用炉等大規模解析への適用性を確保するため、並列処理化するものである。

（2）並列計算の現状

科学技術計算プログラムにおける並列化は、ベクトルプロセッサーと並び、計算速度向上を図るための代表的な手段の一つである。また、並列化には、領域を分散することにより、限られた記憶装置や入出力装置などの計算機資源を有効に使用できるという利点もあり、現在、多くの並列研究グループが存在する。

現在、並列計算用にフリーで提供されているプラットフォームが幾つか有り、GEOFEM（RIST）【2】、Aztec（Sandia National Laboratory）【3】、PETSc（Argonne National Laboratory）【4】が代表的である。

並列化が有効なプログラムは、有限要素法や有限差分法など、計算対象を小さなメッシュで分割し各メッシュにおける編微分方程式を解くもののように、メッシュ数での DO ループが多く存在し、この部分の処理を分割並列処理することが可能なものであ。この点については DO ループをベクトル化して高速処理するベクトルプロセッサーと共通している。

並列処理をするためには、分割された領域間において境界データ等のやり取りが必要になるが、この通信を FORTRAN プログラムや C 言語プログラム上で可能とする並列化プログラミングツールとして MPI【5】が主流となっている。

並列処理が可能な計算機システムは、並列計算専用の共有メモリー型から、単にネットワークを介したワークステーション群など、いろんな形態が存在する。共有メモリー型は通信速度が速いため並列化効率上有利であるが一般に高価である。ネットワークを使用する場合は、極力高速なものが望まく、経験上、最低でも 100base より速いものが必要である。

並列化効率向上に最も重要な事項は以下の二点である。

・最少の通信回数

・均等な計算負荷分散

通信速度は並列化効率に影響を及ぼす極めて重要なファクターである。データ転送速度はハードで決まってしまうが、通信手続き等によるオーバーヘッドに対する対策はアプリケーションソフト側で行う必要が有る。同じタイミングで行える通信は一度に行う（通信データが多くなることよりも通信回数を減らすことが重要。）、不要な通信は極力避けることにより通信によるロスを最少にする。

計算負荷分散は、殆ど、領域分割で決まる。領域分割の理想は、分割後の各領域内において要素数・節点数が均一になること、及び、領域境界の要素数・節点数が最少になることである。領域境界の要素数・節点数を少なくすることは通信データを少なくすることに通じる。

（3）現行の SPIRAL の計算負荷状況

SPIRAL の計算内容は通常の有限要素法による流体解析コードと同様に、各保存式を弱形式し、要素マトリックスの組立・重ね合わせ・全体マトリックスを解くというステップに分けられる。係数行列は、過渡計算中においても不変であるため、過渡解析で負荷が大きくなるのは荷重ベクトルの計算と全体マトリックスを解く部分となる。特に全体マトリックスの負荷が過半数を占める。有限要素は、その解法自体の特質として、重ね合わせの前までは、ほぼ要素内で閉じた処理となる。このため、要素マトリックスの組立においては並列化処理は難なく可能である。したがって、重ね合わせから全体マトリックスを解く部分に対して特に注意して並列方法を選択した。

（4）SPIRAL の並列化手法の概要

様々な計算機環境に対応するべく、並列化プログラミングツールに MPI を用い分散メモリー型のプログラミングを行った。

有限要素法詳細熱流動解析コードである SPIRAL の並列処理は、形式上、SPMD（Single Program Multi Data）に分類される。（他の形式には MPMD（Multi Program Multi Data）がある。）SPMD 形式は、全ての並列プロセスが同一のプログラムであるものを指す。 従って、並列計算中に生

じる PE（Processor Element）間通信の送信側・受信側の手続きは一つのプログラム上で記述する。通信の送り手・受け手の認識は、各 PE に割り当てられる ID（MPI の場合は Rank）を用いる。

有限要素法の並列化でオーバーラップ要素を用いた領域分割は、少ない通信で並列化を実現できる優れた手法である。全節点への初期値設定後から有限要素法固有の手続きである重ね合わせまで通信無しで並列計算が可能となる。現在、幾つかの領域分割のフリーソフトが存在するが、中でも Metis【6】は複雑な計算メッシュ形状で優れたパフォーマンスを示す代表的なソフトである。SPIRAL の領域分割には、この Metis と合わせて単純メッシュ形状に適した RCB 法【7】を備えた GEOFEM の領域分割ツールを改良し利用した。

大規模体系計算時に計算負荷が集中する連立一次方程式は ICCG 法の並列処理で対応した。その他にも、非対称行列用ではあるが高速の手法である BiCGSTAB 法と GPBiCG 法のソルバーを用意し計算時間の比較を行った。

動作確認と並列化効率の検証をするために、三次元で規模のことなる体系を二ケース選び、PE 数に対する計算時間・経過時間について測定し評価した。

SPIRAL の並列化手法と検証計算の詳細については次章以降に、また、本報告書の最後に、並列版 SPIRAL の保守・運用の手助けになるように付録としてプログラムの仕様と実行方法について詳しく記述する。

2. SPIRALの並列化手法

2．1 領域分割型並列化

（1）分散メモリー型

有限要素法・有限差分法の解析コードの並列化では、領域分割による処理分散が一般に用いられる。ともに計算の対象となる物理空間を細かいメッシュでモデル化した手法にとっては、最も自然で且つ有効な処理分散手法と言える。有限要素法コードであるSPIARLの並列化も、この領域分割法により行う。

図2.1に領域分割法による並列処理の概念を示す。図では、元の全体領域（ドメイン）を４つのサブドメインに分割し、個々のサブドメインの処理をそれぞれ別のプロセッサーエレメント（PE）で行うことを表す。個々のPEはサブドメインの境界データ等を他のPEと通信により交換し合いながら処理を進めていく。

分割後の領域を処理する個々の並列プロセスは、分割後の領域のみをメモリーに格納して処理する分散メモリー型を採用する。これは、いろんな並列計算機環境に対応できるようにするためである。単にネットワークを介した複数のワークステーションを利用する場合などでは、個々に独立したメモリーを有効に使用できる。もちろん、共有メモリー型の計算機にも、プログラムの変更無しに対応できる。

分散メモリーを考慮したプログラミング法と共有メモリー型の計算機だけを対象とするプログラミング法には、次のような違いがある。分散メモリーを考慮する場合、他のサブドメインとの境界データとの受け渡しをプログラム上、通信ライブラリーを組み込む形式で行うのに対し、共有メモリーのみに対応するプログラムでは、共有メモリーに対して特殊な配列宣言方法を用いて領域全体を格納し、通信ライブラリーを使用することなく他のサブドメインが処理する領域にアクセスできる等の簡便さが利点であるが、汎用性に乏しい。

各要素・節点等で定義される主要変数は、要素数・節点数等（他に次元数など）をパラメータとして大きさが決まる配列形式を取っており、今回のメモリー分散型並列化では、分割領域内で最小の配列サイズを使用する。

オリジナルのSPIRALでは、FORTRANのparameter文で配列サイズが決定されるため、解析体系の規模に対し必要最小限の配列サイズにするには、その都度、プログラムレベルでの変更が必要であるが、FORTRAN90の動的割り当て機能を使用することにより（配列のallocatable宣言、及びallocate文）、体系の規模や分割後の体系のばらつきによらず、再コンパイルなしに、全てのPEで必要最小限の配列サイズが使用できるようにした。

（2）領域分割ツール（パーティショナー）

領域分割は、並列計算において最も重要な問題であり、これにより、並列化が成功するか失敗するか決まると言っても過言ではない。理想的な分割は、均等な負荷バランス（通信の待ち時間を最小にする。）と境界が最小（通信時間と通信のオーバーヘッドを最小にする。）となるものである。多くの研究グループによって開発されたが、現在では、以下の手法に集約される。

Metis はミネソタ大学でグラフ理論を応用して開発された分割用ライブラリーで、要素・節点の結合情報をグラフ（節点と辺の集合）として認識し、それに基づいて領域の分割を行う。特に、複雑形状の計算領域の分割に対して効率よい分割が行える。フリーウェアであり、多くの並列化で実績がある。

RCB (Recursive Coordinate Bisection) は、例えば、対称軸を持つような単純な形状の領域に対して Metis より有利な場合がある。

並列化プラットフォームである GEOFEM は、そのパーティショナーとして、この Metis と RCB 法を採用している。今回の並列化では、SPIRAL 本体の改造を軽減するため、SPIRAL 本体と入力データのインターフェースとしてのパーティショナーという形式をとった。GEOFEM パーティショナーをベースとして、SPIRAL のオリジナル入力形式のままで領域分割できるものを作成した。パーティショナーは、SPIRAL 入力ファイル内の要素・節点情報を元に領域分割を行い、各領域を担当する PE 用の入力データを新たに作成する。

（3）分割境界

図 2.2 に分割境界における要素・節点の関係を示す。

分割後の領域境界は要素表面となる。イメージ的には、要素の中心で分割し、分割後の両側の領域で、それぞれ分割要素を丸ごと有する手法を用いる。分割後に両側で持たれる要素はオーバーラップ要素として、分割後の各領域における通信のバッファー領域となる。

分割数を増やすと、全体系の要素数に対するオーバーラップ要素数の比率が高くなる。このため、PE 数に対して並列化効率が線形にならない原因の一つとなる。

（4）領域分割ツールによる要素形状の制約

使用する領域分割ツールがグラフ理論に基づくものであるという制限上、エッジ上に節点がある要素しか使用できない。言い換えると、二次元要素では面

内部、三次元要素では要素表面内部・体積内部に節点を持つ二次要素は対象外である。

このため、今回の並列化は一次要素のみを対象とした。ただし、SPIRAL 本体では、現行の SPIRAL が対象とする全ての要素形状に対応できるようになっている。

２．２ 計算負荷の分散方法

（１）領域分割による負荷分散

有限要素法プログラムの領域分割による計算負荷の分散は、プログラム上では、要素数・節点数で回すDO文内の処理の分散に他ならない。それ以外の処理については全ての並列プロセスが同じ処理を行う。DO文以外の部分を無理に並列化すれば、不必要な通信が発生し計算効率を損なう。

並列化がDO文内の処理を分散処理し計算速度を上げるのに対し、ベクトル化もDO文内の処理を高速化する手段である。方法は異なるが、高速化される箇所は共通である。並列化においては、できるだけ領域外の変数を参照しない（通信を避ける)、ベクトル化では回帰的参照（ベクトル化が不能）をしないという制約を考慮したプログラミングが必要となる。

今回の並列化では、DO文内の処理の分割を領域の分割に対応させることにより負荷分散を行った。

（２）要素番号・節点番号の分割方法

DO文の分割処理に対応して、分割後の要素番号・節点番号をどうするかという問題が発生する。

これに対しては二つの対応が存在する。一つは、分割後の領域毎に再番号付けを行う方法であり、他方は、要素番号・節点番号は分割後も不変とし、リストベクトル（DO変数を配列変数化）を使用する方法である。どちらも計算速度上で絶対優位になる理由はないものの、前者の方がプログラムの修正箇所が少なくて済むことがバグの発生確率を少なくさせると考え、前者の方法を採用した。

要素番号・節点番号の再番号付けのアルゴリズムは、パーティショナーで決められる。必ずしも、分割後の番号の昇順と分割前の昇順は一致しない。ただし、オーバーラップ要素上の節点の内、他の領域からデータを貰う外点と他の領域にデータを与える内点の関係は、内点から外点という順序で番号付けされる。

（3）オーバーラップ要素節点間通信

図 2.3 に領域境界上の通信の概念図を示す。

オーバーラップ要素上の任意の節点は、分割後の、何処か一つの領域の内点であり、また、隣接する他の全ての領域の外点である。言い換えれば、オーバーラップ上の内点の集合はオーバーラップ要素の全節点となる。

通信による領域間の境界データのやり取りは、全て、このオーバーラップ要素上の節点についてのデータのやり取りである。並列処理に必要な通信の大部分がこの処理である。その他、全体系内の最大値を求めるなど、通信が必要になる箇所もあるが、これについては後に詳しく説明する。

このオーバーラップ要素上の通信については、節点と通信相手の PE 番号を対応させる通信テーブルをパーティショナーが作成する。

２．３ 行列演算（反復法）の並列化手法

並列化・ベクトル化に関わらず、係数行列が疎である大規模な連立一次方程式を解く解法として、直接法よりもCG法系の反復法の方が使用するメモリーサイズが少なくて済むことや高速であるという理由で一般的に用いられている。SPIRALは連立方程式ソルバーとして、ICCG法と直接法を備えているが、並列処理が必要になるのは大規模大系の解析であるということを踏まえて、CG法系の反復法のみで対応した。そのかわり、ICCG法以外にCG法系で非対称用ではあるが高速と言われるILUBiCGSTAB（双対共役勾配安定化法）とILUGPBiCG（一般積型BiCG法）を並列化に対応して組み込んだ。なお、これらは、公開されているGEOFEMのプログラムをカスタマイズし、より通信回数を少なく高速化して使用した。

（１）局所前処理

前処理は不完全LDU分解した行列を用いる。L（下三角行列）とU（上三角行列）は元の係数行列のLの部分行列とUの部分行列をそのまま利用する。D（対角行列）は元の係数行列の各対角成分の逆数を取ったものとする。不完全LU分解では最も単純な手法であるが、それ故計算時間も少なく収束性も比較的良い。

前処理行列を使用すると、反復中に前処理行列を使った前進代入と後退代入の処理が発生する。全体系での前処理は並列化したとしても、逐次的な処理が必要になり並列化する意味がない。これに対処するため、GEOFEMが採用している局所的前処理を行う。

局所前処理は、分割領域内だけで前進代入と後退代入を行うというもので、前処理に関する通信が発生しないという利点がある。そのかわり、並列化以前より反復回数が増えることになるが、ここで通信が発生しない方が遙かに計算速度上有利である。

（２）各反復法の演算量と通信回数

表 2.1 に各反復法における演算の種類と並列計算時に必要となる通信種類を示す。表の中の数値は１反復計算あたりの回数

一般に、BiCGSTAB法は条件の良い時にBiCG法の半分の繰り返し回数で収束すると言われている。通常、対称な係数行列に使用されないため、CG法との比較がされたものは知られていない。BiCGSTAB法の演算量と通信回数はCG法のそれ

ぞれ約倍になるので、倍以上の収束性が得られないとCG法より有利にならない。GPBiCGは、BiCGSTABよりわずかに通信回数が少ないが演算量はさらに多い。

それぞれの収束性の実測データは次章の検証計算に記述するが、ICCG法が最も高速であった。ただ、収束性は解析条件により異なるので、最終的判断は今後の解析事例に委ねる。

（3）行列演算用配列インデックス

係数行列と計算に必要な係数行列のインデックス（対応する対角成分を示す）を格納する配列は一次元配列に詰めて格納する。三解法とも同一である。（現行のSPIRALでは、二次元配列を使用している。）

以下に形式を示す。

① 係数行列の格納順序

対角：1，2，3，・・・・・・・・・・・・・・・・・・・・・・・・・・，n

下三角：対角1に対する左側の係数，・・・，対角nに対する左側の係数

上三角：対角1に対する右側の係数，・・・，対角nに対する右側の係数

② 係数行列インデックスの格納順序

下三角：対角1に対する左側の係数，・・・，対角nに対する左側の係数

上三角：対角1に対する右側の係数，・・・，対角nに対する右側の係数

③ 係数の個数

下三角：第1行対角の左側の係数個数，・・・，第n行の左側の係数個数

上三角：第1行対角の右側の係数個数，・・・，第n行の右側の係数個数

２．４ 同期制御方法

並列処理を制御するために MPI 通信ライブラリーを使用した通信を必要な箇所で行う。通信によるオーバーヘッドによる並列化効率の悪化を防ぐため、極力、同時に通信できるものは一カ所でまとめて通信を行うようにし、通信回数は極限まで押さえた。

なお、通信に使用するコミュニケータには MPI のディフォルト・コミュニケータである MPI_COMM_WORLD のみを使用した。

以下に、通信箇所と内容を記す。

（１）通信種類（Point-to-Point、Global）

① Point-to-Point 通信

領域境界値の更新を行う時に用いる。各領域のオーバーラップ上の接点データのうち、内点として持っている接点情報を他の領域の対応する外点へ送信し、他の領域の内点から外点の接点情報を受信する。

参照）　使用 MPI ライブラリー

MPI_SEND：送信
MPI_RECV：受信
MPI_WAIT_ALL：送信・受信完了までの待機

② Global 通信

特定の値について、全領域内での最大値を求めたり加算を行う。さらに、全ての PE に加工後の値を送信する。

参照）　使用 MPI ライブラリー

MPI_ALLREDUCE：数値の加工（加算、最大値等）及び
ブロードキャスト

（2）通信箇所

以下に全ての通信箇所①～⑧と通信データの内容を示す。
なお、①は初期のみ、②～⑧は過渡計算ループ内である。（サブルーチンとの対応は付録A．1を参照）

① 乱流エネルギーの最大値
エネルギー散逸率の最大値

② 壁近傍の乱流エネルギー（領域境界の更新）
壁近傍のエネルギー散逸率（領域境界の更新）

③ セル間距離（上流化の場合のみ）（領域境界の更新）

④ 近似流速（次元分）（領域境界の更新）
温度（領域境界の更新）
乱流エネルギー（領域境界の更新）
エネルギー散逸率（領域境界の更新）

⑤ 圧力（領域境界の更新）

⑥ 流速（次元分）（領域境界の更新）

⑦ 圧力方程式連立ソルバー内での行列演算（領域境界の更新と内積）

⑧ 定常の判定（主要変数の最大値・最大変化量）

（3）通信テーブルの形式

領域境界の外点の数値の更新には、パーティショナーが作成する通信テーブルを使用する。

形式は一次元配列で以下の通り。

① 送信テーブル

オーバーラップ要素上の内点の節点番号 － 送信相手の PE 番号

② 受信テーブル

オーバーラップ要素上の外点の節点番号 － 受信相手の PE 番号

２．５ 入出力制御方法

本並列手法は汎用性という観点から分散メモリー型を採用した。分散メモリーにおいては、並列動作時の不要な通信を防ぐために、入出力ファイルとも PE 毎に完全独立させることが望ましい。ただし、出力ファイルについては、POST 処理用に個々の PE の結果ファイルを PE0 が全計算終了後に単独でマージ処理を行い、全体系のファイルを作成する。

共有ファイル方式の計算機環境においては、並列版 SPIRAL のロードモジュールを置いたディレクトリー上で全ての入出力ファイルがアクセスされる。また、ネットワークを介したワークステーションのように個々にディスク装置を有している場合は、全てのディスク装置において同一ディレクトリーパスを作成し、ロードモジュールと入力ファイルを置く。

（１）入力ファイル

実行に必要な入力データは以下の二つである。

① 領域分割された実行データ（プロセス数分）

② FILENAME（実行データのファイル名を指定するファイル）

（例）Solv.inp　というファイル名が指定されると、m+1 個のプロセスに対して Solv.inp.0、Solv.inp.1・・・・・Solv.inp.m という入力ファイルがあると認識される。

（参考）　①の実行データにおける要素・節点番号について

要素データにおける要素番号・節点番号は全体系・局所系、
節点データにおける節点番号は全体系、
境界データにおける要素番号・節点番号はともに局所系、
通信テーブルにおける節点番号は局所系

という形式で表されている。ただし、計算においては要素データの入力順、節点データの入力順（局所系）が実質の番号となる。

（2）出力ファイル

OUTPUT	実行結果ファイル
PLOTOUT	作図用ファイル
RESTART.0 ～ RESTART.m	リスタートファイル
RESTART	リスタートファイル（作図用）

（3）作業ファイル

これらのファイルは計算終了時に消去される。（ただし、1PE での計算ではマージしないので、OUTPUT・PLOTOUT・RESTART が作成されずに計算後も残る。）

OUTPUT.0～ OUTPUT..m	実行結果ファイル
PLOTOUT.0 ～PLOTOUT.m	作図用ファイル

２．６　運用上の制約事項

（１）要素形状

現行の SPIRAL の要素形状の内、エッジ上に節点があるものに限る。(パ ーティショナー上の制約。)

（２）分割数

ファイルの修飾子の桁数に制限がある。(10000 分割) ソフト上で変更可能。なお、隣接 PE 数は最大 26。

（３）メッシュ数

計算機のメモリーサイズのみ。ソフト上の制約は無い。

3. 検証計算

並列化したSPIRALコード並びに並列計算用領域分割ツール（パーティショナー）を流体計算工学研究グループの所有するクラスターマシンに移植し、使用環境を整備した。動作および解析精度の確認をするために、過去にサイクル機構で実施した検証計算のうち２ケースを使用した。

解析ケースには、三次元で要素形状および規模が異なる２ケースを選んだ。小規模／大規模体系において、計算に使用するPE数と圧力Poisson方程式の行列解法を変えて、反復回数と計算時間の測定を行った。計算PE数は、1, 2, 4, 8, 16 PEとして、また、行列解法には、従来のICCG法に加えて、GPBiCG法、BiCGSTAB法を選択した。

（1）計算条件

計算条件は、以下の通り。

・計算体系

２ケースのメッシュ図と領域分割を色分けして図3.1.1(Case1)および図3.1.2(Case2)に示す。

小規模体系であるCase1は、単ピンの燃料集合体試験体を模擬した要素数3200、節点数3587の体系である。6節点五面体要素と8節点六面体要素を混合して使用している。

また、大規模体系であるCase2は、プレナムを模擬した要素数198030、節点数217232の体系である。8節点六面体要素のみを使用している。

・並列計算用領域分割方法

並列計算用領域分割ツールはRCB法とMetis（RSB法）の二種類が利用できる。前者は単純な体系、後者は複雑な体系に対して有効である。本計算では、形状が規則的で単純なCase1ではRCB法、不規則的で複雑なCase2ではMetis（RSB法）を使用した。

・使用PE数と行列解法

計算に使用したPE数は1, 2, 4, 8, 16 PEという組み合わせで行った。解の精度の確認と並列化効率調べるケースでは、圧力方程式の行列解法にはすべてICCG法を選択した。GPBiCG法、BiCGSTAB法に関しては、Case1は1 PEでのみ、Case2は1, 16 PEで実施した。

・その他

測定を行った値は、以下の通り。

① 全体の計算時間（ 過渡計算の開始から終了までに要した計算時間 ）
② CG法の計算時間
③ CG法の反復計算回数

計算はCase1／Case2ともに100-stepまで実行して、それぞれ1-stepあたりの平均値に換算した。なお、プロットファイル ’ PLOTOUT ’ 、リスタートファイル ’ RESTART ’ への出力は、最終ステップでのみ行った。

（2）計算結果

・ 解の精度の確認

並列版SPIRALが正しく動作しているか確認するために現行のSPIRALの結果と比較した。その結果、1PEの計算では、バイナリーレベルで全ての計算値が一致した。また、複数PEを使用した並列計算でも、主要物理量の計算値に対して相対値で10^{-10}未満程度となり、計算順序等の違いによる丸め誤差程度しか違いが生じなかった。
これにより並列版SPIRALは正しく動作していると確認できた。また、高レイノルズ数の充分発達した乱流領域でも安定に現行のSPIRALの結果に追従できることが確認できた。

・Case1：小規模体系

表3.1.1、表3.1.2、表3.1.3、表3.1.4、および、図3.2.1、図3.2.2、

図 3.2.3 に、小規模体系における全体／CG 法／CG 法 1-iteration ／CG 法以外の計算時間測定結果を示す。小規模体系では、全体の計算速度は複数 PE を使用しても 1.1 ～ 1.2 倍とそれほど向上せず、特に 16 PE では 1 PE より計算時間は増大している。その原因は CG 法の並列化効率が極めて低いからであり、PE 数の増加に伴い、1-iteration の計算時間はわずかに減少するが、反復計算回数はそれ以上に増加するため、結果的に CG 法が収束するまでの計算時間は増大している。一方、CG 法以外の計算時間は PE 数に応じて減少しており、全体の計算速度がわずかながらも向上しているのに寄与している。また、各行列解法の比較では、反復計算回数が最小となるのは GPBiCG 法であるが、1-iteration の計算時間は ICCG 法が GPBiCG 法／BiCGSTAB 法の 1 / 2 程度であるため、結果的に ICCG 法が最速となっている。

・Case2：大規模体系

表 3.2.1、表 3.2.2、表 3.2.3、表 3.2.4、および、図 3.3.1、図 3.3.2、図 3.3.3、図 3.3.4 に、大規模体系における全体／CG 法／CG 法 1-iteration ／CG 法以外の計算時間測定結果を示す。大規模体系では、計算速度は PE 数に応じて効率的に向上している。小規模体系との最大の相違点は CG 法の並列化効率の向上であり、PE 数の増加に伴い、反復計算回数も増加するが、1-iteration の計算時間が劇的に減少するため、結果的に CG 法が収束するまでの計算時間も効率的に減少している。さらに、CG 法以外の計算速度は飛躍的に向上しており、特に 2 PE、8 PE では並列化効率が 1 を越える Super Linearity が測定されている。また、各行列解法の比較では、反復計算回数は ICCG 法が GPBiCG 法／BiCGSTAB 法の約 1.5 倍となるが、1-iteration の計算時間は小規模体系と同様に 1 / 2 程度であるため、結果的に ICCG 法が最速となっている。なお、1-iteration の計算時間から、計算量は GPBiCG 法、通信量は BiCGSTAB 法がそれぞれ最大となり、ICCG 法はともに最小となるのが分かる。

（3）まとめ

① 並列化 SPIRAL の 1PE の計算結果は、現行の SPIRAL のものとバイナリーレベルで完全に一致した。複数 PE を使用したケースでは、流速・圧力・乱流量の主要物理量に対し、１過度サイクルあたり 10^{-10} 未満の違いが生じた。ただし、これは、圧力方程式を解く際の演算内容の違い（領域分割数に応じて異なる前処理行列を使用する）や丸め誤差の影響の範囲であり、正しい計算を行っていると判断できる。

③ 計算要素数が 3,000 程度の小規模体系では、並列化の効率はほとんどあがらない。計算条件によっては、複数 PE の計算時間は 1 PE より増大する。（圧力方程式の並列化効率の低下が原因）

④ 計算要素数が 200,000 程度の大規模体系では、並列化の効率は飛躍的に向上する。計算条件によっては、圧力方程式の処理以外で並列化効率が 1 を越える Super Linearity も測定される。

⑤ 圧力方程式を解くための反復解法は、大規模体系においては経過時間レベルで PE 数を増やすほど高速であった。

4. 考　察

（1）並列化効率のスーパーリニア特性について

今回の計算時間の測定で、一部の演算箇所でスーパーリニア特性が観測された。特に、大規模体系での計算において、2PE と 8PE を使用したケースで目立つ。計算時間は圧力方程式とそれ以外について測定したが、どちらの場合も観測された。圧力方程式の計算以外の場所では、節点－要素接続テーブルやマトリックスソルバー用インデックス作成時等で領域（サブドメイン）内における多重ループでのサーチなど、領域の大きさに対して非線型な計算負荷が掛かるものが原因と推測される。また、圧力方程式の中では、並列処理により、丸め誤差の影響が少なくなり収束性が向上することも考えられる。また、特に８PE が目立った理由については、分割形状のバランスが最も効率的であったと推察される。

（2）マトリックスソルバーの計算効率について

圧力方程式の解法として用いるマトリックスソルバーとして、現行の SPIRAL が備えている ICCG 法以外に、高速反復解法と言われている BiCGSTAB 法と GPBiCG 法を並列化し組み込んで計算速度を比較した。SPIRAL の圧力方程式の係数行列は対称行列であり、ICCG 法は対称行列の高速解法として広く使用されている。一方、BiCGSTAB 法と GPBiCG 法は非対称行列用ソルバーとして、現在、最も高速と言われる解法である。

今回の二ケースの計算結果では、異なる収束性を示した。Case1 の集合体を模擬した小規模解析では、一回の演算量・通信回数とも最も少ない ICCG 法が最も少ない反復回数で収束し、従って、計算時間も最も高速であったのに対し、Case2 のプレナムを模擬した大規模解析では、GPBiCG 法が ICCG 法よりも少ない反復回数で収束した。GPBiCG 法は、ICCG 法に比べて一回の演算量・通信回数が多いため、計算時間では僅かに劣った。

一般に、CG 系の反復解法の収束性は係数行列の性質に大きくかかわる。解析条件によっては、これらの優劣が入れ替わる可能性がある。残念ながら、使い分けを明確に判断できる手法はない。しかしながら、係数行列が対称である以上、ICCG 法は演算量・通信回数・通信量とも他の解法の半分近くですみ、収束性も安定している。この理由で ICCG を推奨する。

（3）丸め誤差の影響について

並列化したSPIRALの1PE計算の結果は、現行のSPIRALのものとバイナリーレベルで一致した。(ただし、圧力方程式の係数行列における非ゼロ要素の配列への格納順序を同一にした場合。並列版では非ゼロ要素をソートして格納している。）複数のPEを用いた並列計算では、圧力方程式を解くマトリックスソルバー内の局所前処理による演算そのものの違いと内積計算での丸め誤差の影響によりバイナリーレベルでは一致しない。局所前処理による違いは反復回数に影響は出るものの解の精度に対しては問題ではない。しかしながら、局所前処理は局所的に不完全LU分解した行列を使った前進代入・後退代入操作をするのでPE数を増やす程、多くの反復回数を必要とする。

内積計算における桁落ち誤差は、自由度数が多くなるほど解の精度に影響を及ぼす。並列化した場合は、分割領域内の（部分的な）内積を求めてから全体領域分加算して計算するので、丸め誤差の影響を受けにくい。PE数を増やすほど、ここでの桁落ちを軽減できる。

（4）ベクトル化について

最近のMPUにはベクトル用パイプラインを備えたものもあり、FORTRANコンパイラーもベクトル化に対応しているものも出てきている。ベクトル化は、並列化とならび、計算の高速化に有用な手段である。

ベクトル化は、DO文内に回帰的参照が生じない場合に可能となり、特にベクトル長が大きな場合は、シングルプロセッサーながらも飛躍的な高速化が期待できる。ただし、DO文がネスト構造になっている場合、一番内側のDOしかベクトル化できない。このため、もともとベクトル化を意識して作成されていないプログラムは、そのままでは殆どベクトル化率は良くならず、オーバーヘッドによる計算効率の悪化を招くケースの方が多い。

通常、ベクトル化する場合、1）リストベクトルを使用し、多重DOループを一重にする。2）回帰演算が生じないようなアルゴリズムに変更する。などの方法で対応する。

有限要素法では、要素内節点でのDOループを多用するが、10位のオーダーのベクトル長ではオーバーヘッドのために効果が期待できない。また、リストベクトルを多用するにも一重ループ化のために膨大な配列が必要になる。また、非構造格子のように係数行列が規則的でないものは、CG法等のマトリックスソルバーの効率的なベクトル化は難しいなど困難な点が多い。

5. 結　論

サイクル機構流体計算工学研究グループでは、実験では計測困難な変形した高燃焼度燃料内の熱流力挙動を数値実験で解明することを目的として、サブチャンネル解析コードと相互補完的に用いる局所詳細熱流動解析コード SPIRAL の開発を進めている。この SPIRAL コードについて、実用炉等大規模解析への適用性を確保するため、並列処理化した。また、改良後のコードについて、並列計算機での動作および解析精度を確認するための例題解析を行った。

SPIRAL の並列化にあたり、並列計算の現状の動向を調査し、SPIRAL に最も適した手法により並列化を行った。

逐次処理を前提としてコーディングされている現行コードについて、各ルーチンの計算負荷状況およびデータフローを調査し、並列化処理化を行うルーチンを決定するとともに、以下に示す並列処理のための改良項目を実施した。

（1）各配列データの PE への分散
（2）DO ループ計算負荷の各 PE への分散
（3）局所数値計算アルゴリズムの改良
（4）反復計算における同期の確保
（5）入出力の制御

改良後の動作確認および精度確認解析を実施した。

改良した SPIRAL コードを流体計算工学研究グループが所有するクラスタマシンに移植し、解析のための環境を整備した。動作および解析精度確認のために、過去にサイクル機構にて実施した検証解析のうち体系規模の異なる２ケースを選択し解析を実施した。得られた計算結果を過去の結果と比較することにより、移植後の動作状況・解析精度・並列化効率を確認した。

並列処理化した SPIRAL は１PE の計算では、バイナリーレベルで現行の SPIRAL の解と一致した。また、複数の PE を使用した並列計算においても丸め誤差の範囲で現行の SPIRAL の解と一致した。並列化効率は、小規模体系のケースでは並列化による計算速度向上は得られなかったが、大規模体系のケースにおいては 16PE 使用時に対して 12 倍の計算速度向上という効果が得られた。

参考文献

【1】大島宏之,今井康友, 「高速炉燃料集合体内詳細熱流動解析手法の開発(1)」、JNC TN9400 2001－064 (2001).

【2】GEOFEM Web Site：http://geofem.tokyo.rist.or.jp/

【3】Aztec Web Site：http://www.cs.sandia.gov/CRF/aztec1.html

【4】PETSc Web Site：http://www-fp.mcs.anl.gov/petsc/

【5】MPI Web Site：http://www-unix.mcs.anl.gov/mpi/index.html

【6】Metis Web Site：http://www-users.cs.umn.edu/~karypis/metis/

【7】H.D.Simon,"Partitioning of unstructured problems for parallel processing",Comp.Sys,in Eng.,Vol.2,(1991).

表 2.1　各反復法における演算と通信の種類

	前処理行列の前進・後退代入	行列・ベクトル積	内積	加算通信	境界値の通信
ICCG	1	1	3	3	1
BiCGSTAB	2	2	4	4	4
GPBiCG	4	2	4	4	4

表 3.1.1　Case1：小規模体系における全体の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]	並列化効率 [-]
1 PE	2.641	2.641	0.000	518.9	1.000	1.000
2 PE	2.374	1.806	0.568	663.6	1.113	0.556
4 PE	2.338	1.110	1.229	726.4	1.130	0.282
8 PE	2.272	0.605	1.667	768.5	1.163	0.145
16 PE	3.212	0.531	2.681	897.2	0.822	0.051

行列解法との関係

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]
1 PE	ICCG	2.641	2.641	0.000	518.9	1.000
1 PE	GPBiCG	4.337	4.337	0.000	489.9	0.609
1 PE	BiCGSTAB	5.998	5.998	0.000	881.7	0.440

表 3.1.2　Case1：小規模体系における CG 法の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]	並列化効率 [-]
1 PE	1.489	1.489	0.000	518.9	1.000	1.000
2 PE	1.746	1.179	0.567	663.6	0.853	0.426
4 PE	1.933	0.818	1.115	726.4	0.770	0.193
8 PE	2.045	0.410	1.635	768.5	0.728	0.091
16 PE	3.030	0.398	2.632	897.2	0.491	0.031

行列解法との関係

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]
1 PE	ICCG	1.489	1.489	0.000	518.9	1.000
1 PE	GPBiCG	3.187	3.187	0.000	489.9	0.467
1 PE	BiCGSTAB	4.849	4.849	0.000	881.7	0.307

表 3.1.3　Case1：小規模体系における CG 法 1-iteration の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]	並列化効率 [-]
1 PE	0.002870	0.002870	0.000000	1.000	1.000
2 PE	0.002631	0.001777	0.000854	1.091	0.545
4 PE	0.002661	0.001126	0.001535	1.078	0.270
8 PE	0.002661	0.000533	0.002128	1.078	0.135
16 PE	0.003377	0.000444	0.002933	0.850	0.053

行列解法との関係

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]
1 PE	ICCG	0.002870	0.002870	0.000000	1.000
1 PE	GPBiCG	0.006506	0.006506	0.000000	0.441
1 PE	BiCGSTAB	0.005500	0.005500	0.000000	0.522

表 3.1.4　Case1：小規模体系における CG 法以外の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]	並列化効率 [-]
1 PE	1.152	1.152	0.000	1.000	1.000
2 PE	0.628	0.627	0.001	1.836	0.918
4 PE	0.405	0.291	0.114	2.844	0.711
8 PE	0.227	0.195	0.032	5.078	0.635
16 PE	0.182	0.132	0.049	6.348	0.397

表 3.2.1　Case2：大規模体系における全体の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]	並列化効率 [-]
1 PE	83.140	83.140	0.000	244.5	1.000	1.000
2 PE	46.795	46.111	0.684	298.0	1.777	0.888
4 PE	24.942	24.234	0.708	304.1	3.333	0.833
8 PE	11.514	10.065	1.449	312.2	7.221	0.903
16 PE	6.862	5.012	1.850	322.2	12.116	0.757

行列解法との関係（ 1 PE ）

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]
1 PE	ICCG	83.140	83.140	0.000	244.5	1.000
1 PE	GPBiCG	112.012	112.012	0.000	172.0	0.742
1 PE	BiCGSTAB	99.869	99.869	0.000	174.5	0.832

行列解法との関係（ 16 PE ）

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]
16 PE	ICCG	6.862	5.012	1.850	322.2	1.000
16 PE	GPBiCG	8.872	7.128	1.744	205.8	0.773
16 PE	BiCGSTAB	7.886	6.018	1.868	205.4	0.870

表 3.2.2　Case2：大規模体系における CG 法の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]	並列化効率 [-]
1 PE	48.596	48.596	0.000	244.5	1.000	1.000
2 PE	29.582	29.337	0.245	298.0	1.643	0.821
4 PE	16.083	15.323	0.760	304.1	3.022	0.755
8 PE	7.591	6.628	0.963	312.2	6.402	0.800
16 PE	4.623	3.503	1.120	322.2	10.512	0.657

行列解法との関係（ 1 PE ）

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]
1 PE	ICCG	48.596	48.596	0.000	244.5	1.000
1 PE	GPBiCG	77.269	77.269	0.000	172.0	0.629
1 PE	BiCGSTAB	65.369	65.369	0.000	174.5	0.743

行列解法との関係（ 16 PE ）

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	反復計算回数 [-]	計算速度 [-]
16 PE	ICCG	4.623	3.503	1.120	322.2	1.000
16 PE	GPBiCG	6.790	5.715	1.075	205.8	0.681
16 PE	BiCGSTAB	5.794	4.650	1.144	205.4	0.798

表 3.2.3　Case2：大規模体系における CG 法 1-iteration の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]	並列化効率 [-]
1 PE	0.198757	0.198757	0.000000	1.000	1.000
2 PE	0.099268	0.098446	0.000822	2.002	1.001
4 PE	0.052887	0.050388	0.002499	3.758	0.940
8 PE	0.024315	0.021230	0.003085	8.174	1.022
16 PE	0.014348	0.010872	0.003476	13.852	0.866

行列解法との関係（ 1 PE ）

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]
1 PE	ICCG	0.198757	0.198757	0.000000	1.000
1 PE	GPBiCG	0.449238	0.449238	0.000000	0.442
1 PE	BiCGSTAB	0.374607	0.374607	0.000000	0.531

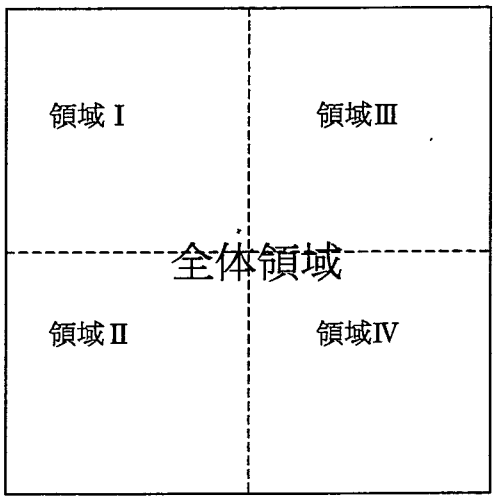
行列解法との関係（ 16 PE ）

PE 数	行列解法	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]
16 PE	ICCG	0.014348	0.010872	0.003476	1.000
16 PE	GPBiCG	0.032993	0.027770	0.005224	0.435
16 PE	BiCGSTAB	0.028208	0.022639	0.005570	0.509

表 3.2.4　Case2：大規模体系における CG 法以外の計算時間測定結果

PE 数との関係

PE 数	経過時間 [sec]	CPU 時間 [sec]	通信時間 [sec]	計算速度 [-]	並列化効率 [-]
1 PE	34.544	34.544	0.000	1.000	1.000
2 PE	17.213	16.773	0.440	2.007	1.003
4 PE	9.084	8.882	0.202	3.803	0.951
8 PE	3.923	3.437	0.486	8.806	1.101
16 PE	2.239	1.509	0.730	15.428	0.964



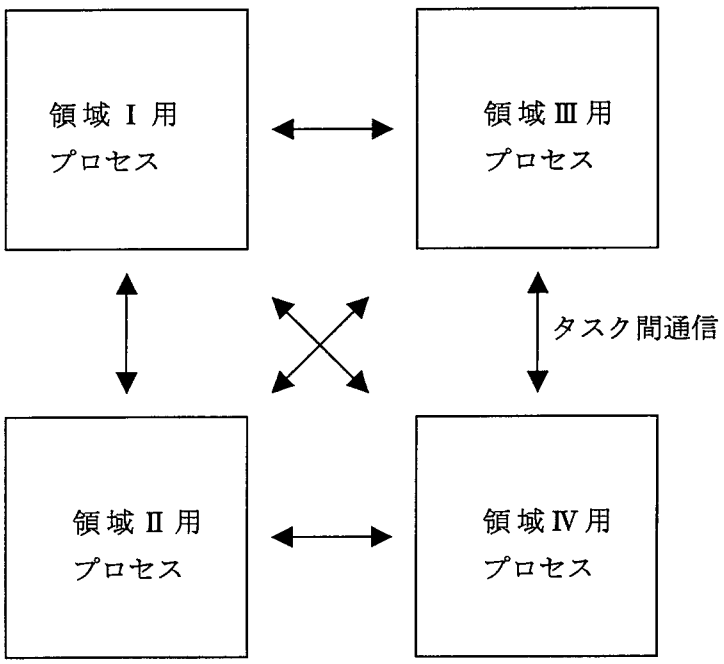


図 2.1 領域分割法による並列処理

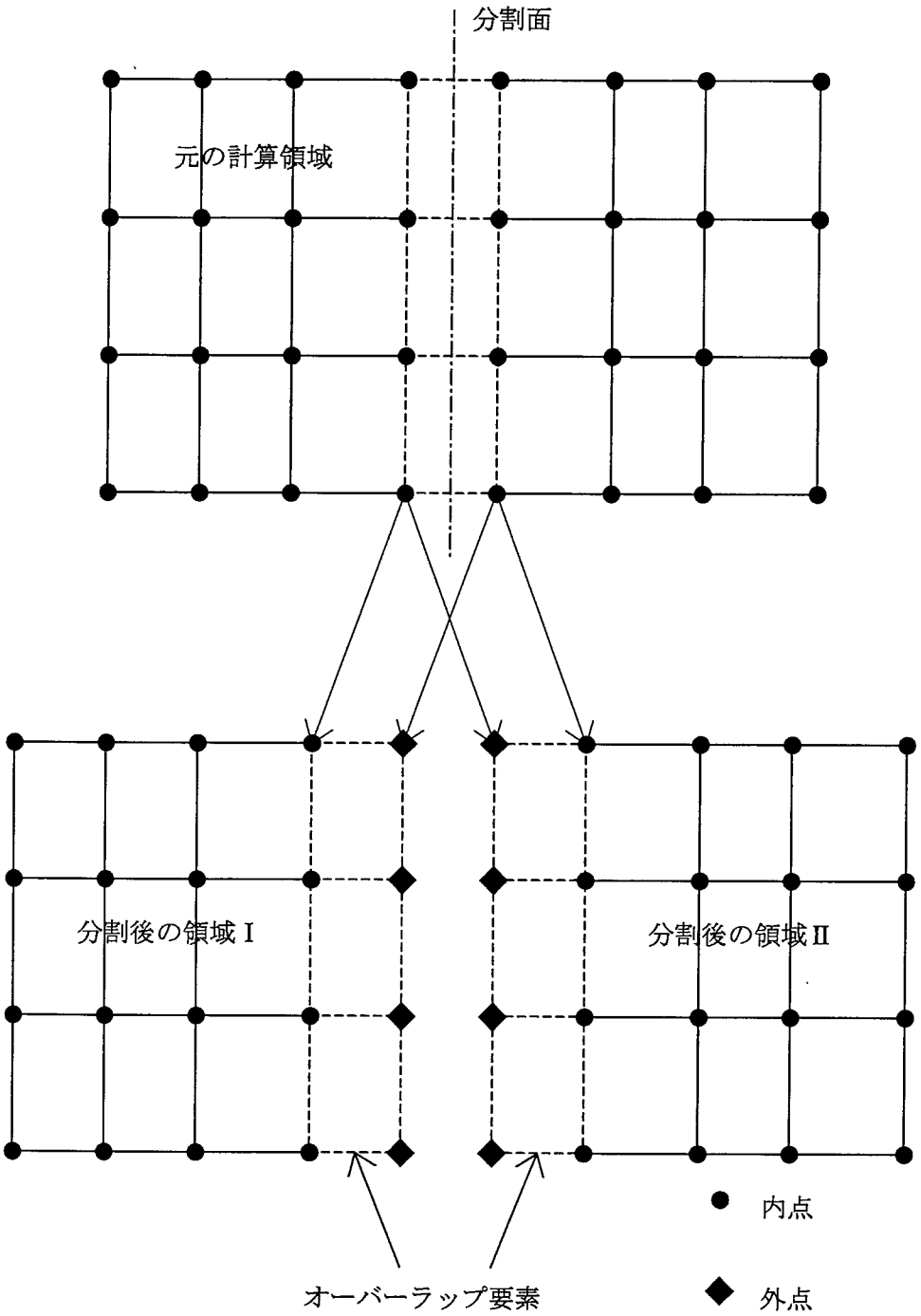


図 2.2 領域分割方法

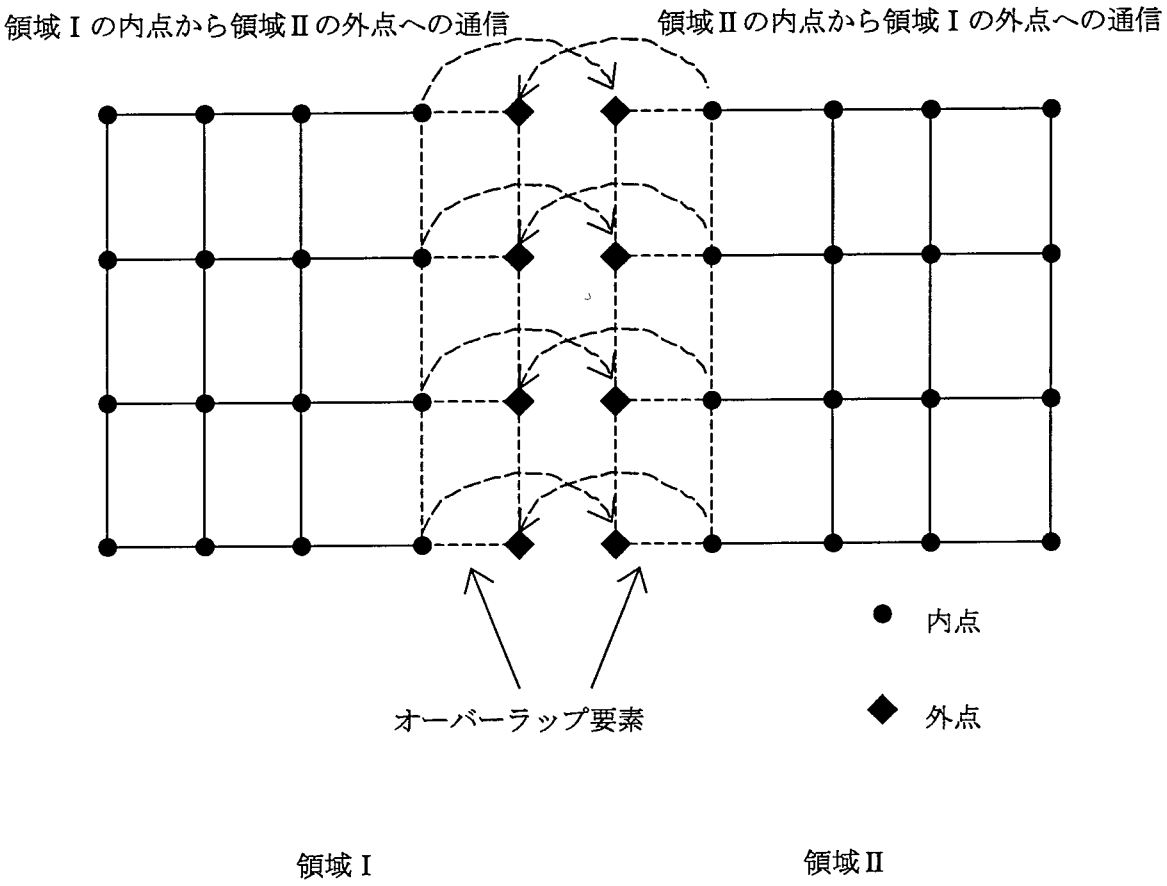
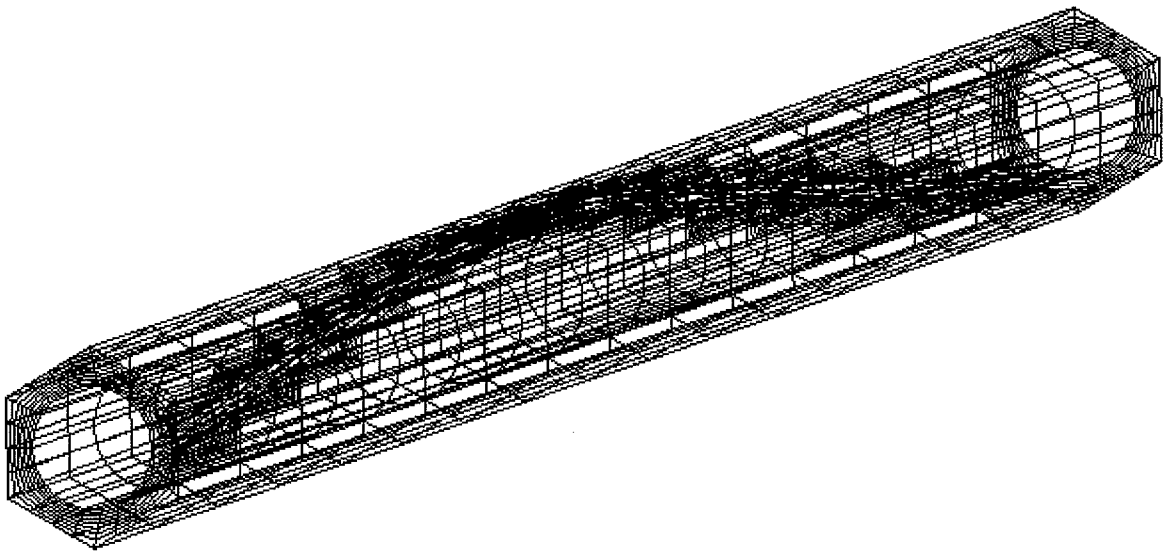
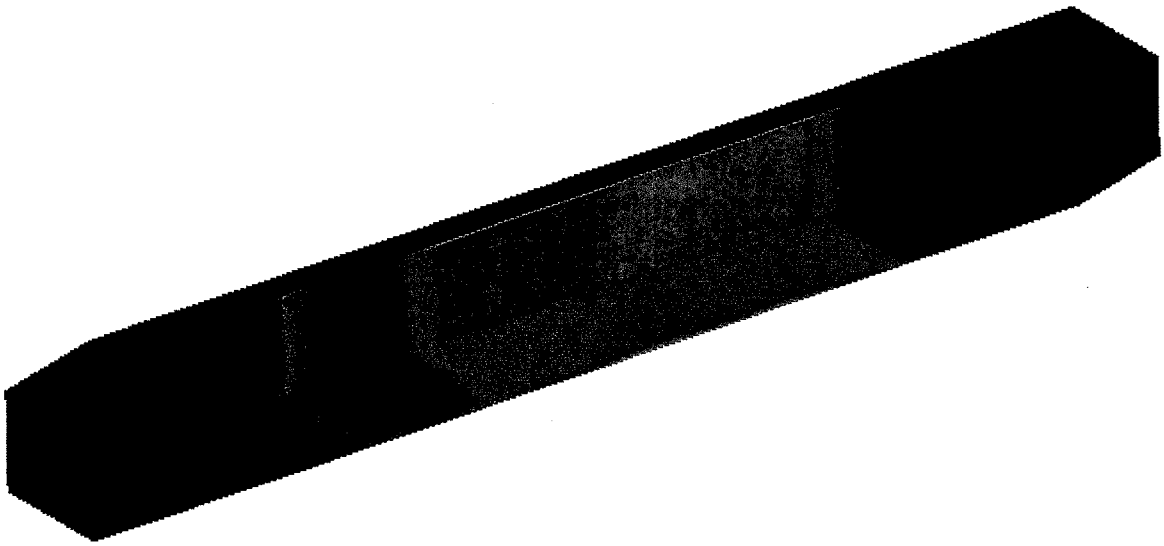


図 2.3 領域境界データの通信

計算体系



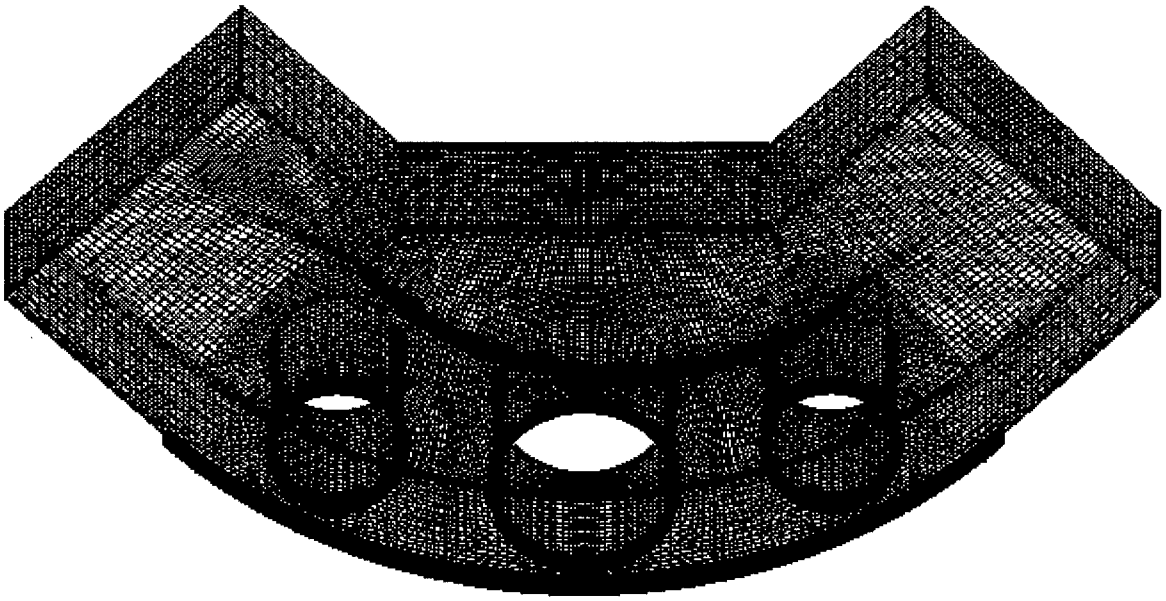
領域分割



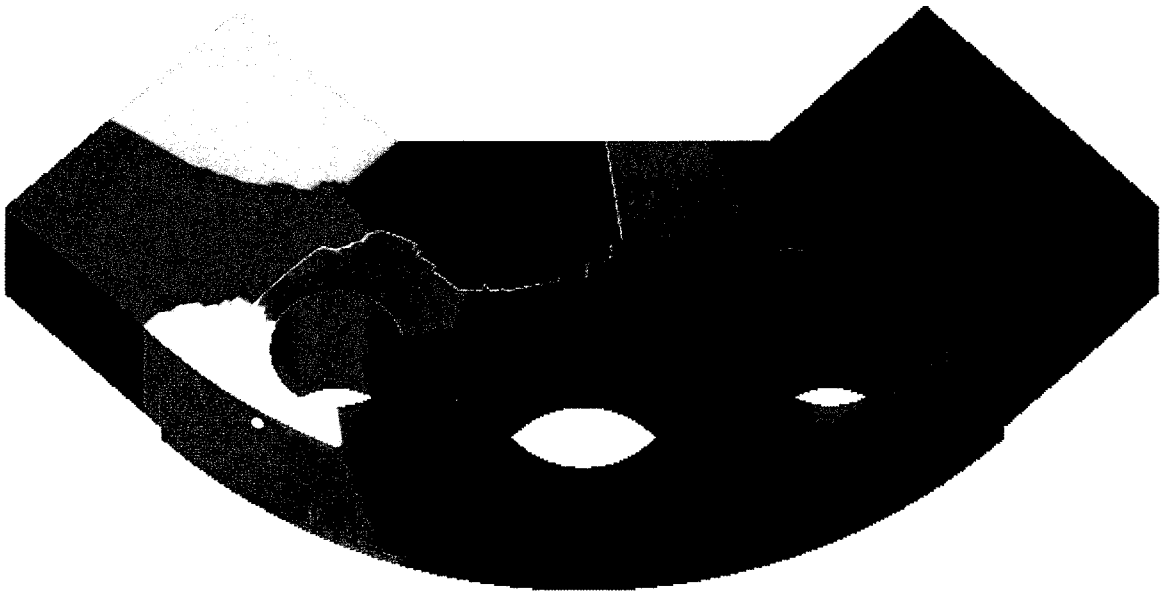
要素数　：　3200
節点数　：　3587
計算要素　：　6節点五面体要素と8節点六面体要素

図 3.1.1　計算体系と領域分割（ Case1：小規模体系 ）

計算体系



領域分割



要素数　　：　198030
節点数　　：　217232
計算要素　：　すべて 8 節点六面体要素

図 3.1.2　計算体系と領域分割（ Case2：大規模体系 ）

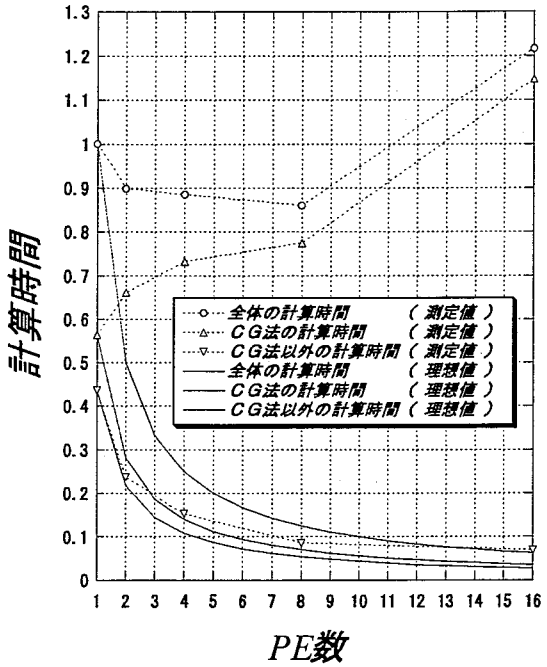


図 3.2.1 Case1：小規模体系における
全体の計算時間測定結果
(1PEの計算時間を 1.0とする)

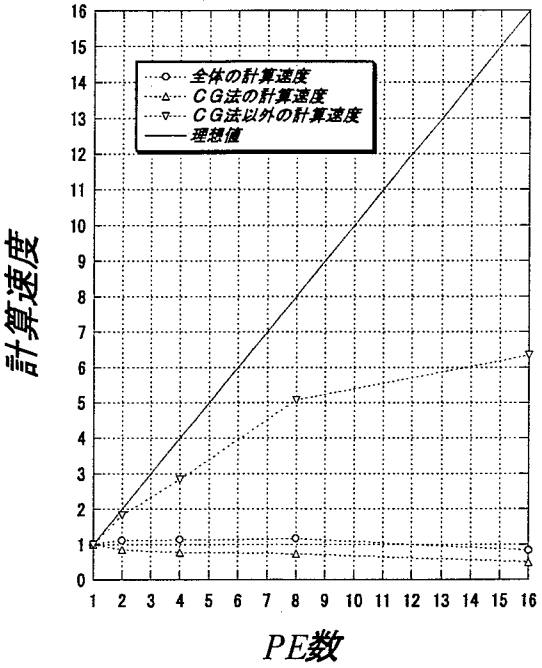


図 3.2.2 Case1：小規模体系における
全体の計算速度測定結果
(1PEの計算速度を 1.0とする)

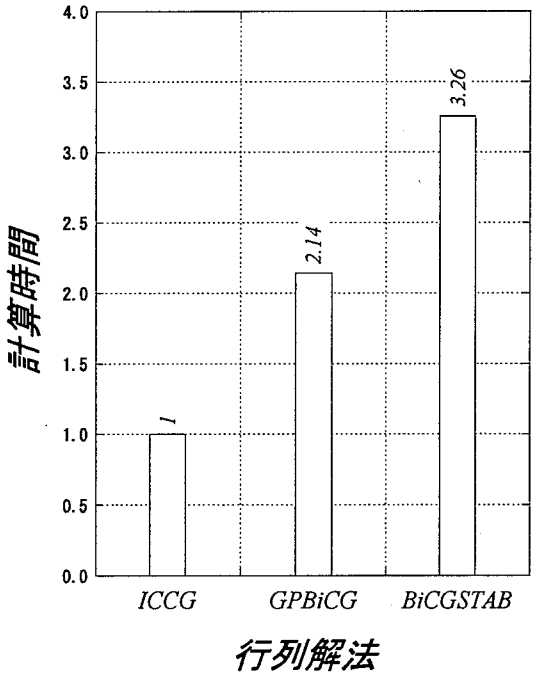


図 3.2.3 Case1：小規模体系における
CG 法の計算時間測定結果 (1PE)
(ICCG法の計算時間を 1.0とする)

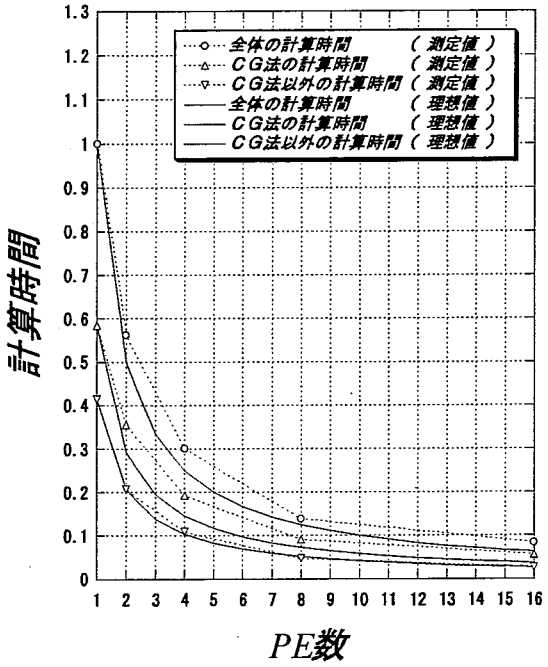


図 3.3.1 Case2：大規模体系における
全体の計算時間測定結果
(1PEの計算時間を 1.0とする)

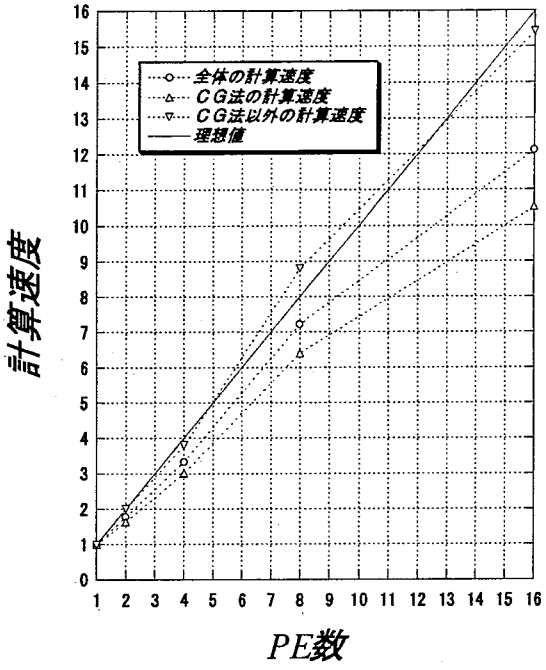


図 3.3.2 Case2：大規模体系における
全体の計算速度測定結果
(1PEの計算速度を 1.0とする)

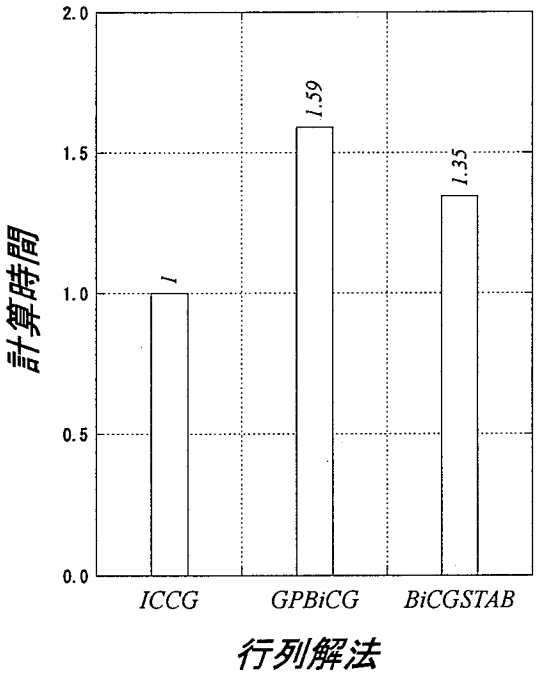


図 3.3.3 Case2：大規模体系における
CG 法の計算時間測定結果 (1 PE)
(ICCG法の計算時間を 1.0とする)

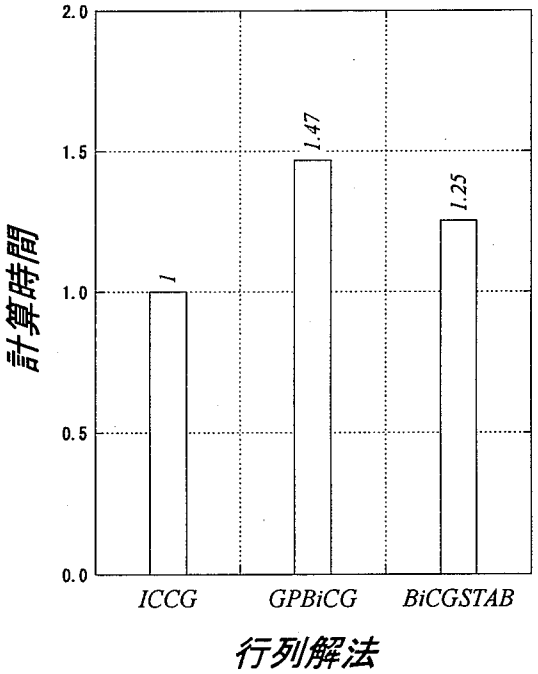


図 3.3.4 Case2：大規模体系における
CG 法の計算時間測定結果 (16 PE)
(ICCG法の計算時間を 1.0とする)

付録 A　並列化 SPIRAL の仕様説明

A．1　TREE 構造

並列化した SPIRAL の TREE 構造を以下に示す。大文字の名称は現行の SPIARL のサブルーチンを、小文字のものは並列化のため追加したものを示す。
なお、通信が行われる箇所に①～⑧を付す。

```
SPIRAL+---parallel_cal_int
       |---UOPEN1
       |---SETSIZE---SETCONST
       |---alloc_array
       |---UOPEN2
       |---INPUT
       |---VERIFY
       |---INITIAL+----RELOAD
       |           |---PLOTST
       |           |---PLOTTR
       |           +---SDWSTA+---CGAUSS
       |                      +---SFDF
       |---TRNIN
       |---input_para
       |---set_geometry_data
       |---dealloc_input
       |---set_aiccg_array
       |---iccg_parameter_init
       |---MTXDRV+---MTXEM+---MTXVI +---NGAUSS
       |           |           |          |---SDWSET
       |           |           |          +---JACOBV
       |           |           +---MTXSI +---NGAUSS
       |           |                      |---SDWSET
       |           |                      +---JACOBS
       |           |---MTXGL
       |           +---SOLVMX
       |---RENEW  ①
```

```
|---TFUNC
|---HCELL---HINVR
|---send_recv_hcell  ②
|---CTKEDRV+---CTKEEM+---COVEM
|              |             |---TEMEM
|              |             +---KEMEM
|              +---CTKEVI+---CENTER
|              |             |---JACOBV
|              |             |---RESTRV
|              |             |---RESTRW---WFUNC
|              |             |---RELOWV
|              |             |---RELOWW
|              |             |---REASMV+---LUDCMP
|              |             |             +---LUBKSB
|              |             |---REASMW +---LUDCMP
|              |             |              +---LUBKSB
|              |             |---NGAUSS
|              |             |---SDWSET
|              |             |---COVVI---HSUPG
|              |             |---TEMVI---HSUPG
|              |             +---KEMVI +---HSUPG
|              |                         |---CEF1F
|              |                         |---CEF2F
|              |                         +---CMUFF
|              |---CTKEGL
|              |---SOLVCV
|              |---SOLVTE
|              +---SOLVKE+---KEHIGH+---WFUNC
|                            |           +---send_recv_xke  ③
|                            +---KELOW
|---send_recv_ctk  ④
|---PREDRV +---PREEM
|             |---PREGL
|             +---SOLVPR +---CG  ⑤
|                         |---BiCGSTAB  ⑤
|                         +---GPBiCG  ⑤
```

```
|
|---send_recv_press  ⑥
|---VELDRV +---VELEM
|              |---VELGL
|              +---SOLVVE
|---send_recv_velo  ⑦
|---STEADY  ⑧
|---OUTDYN+---POTTR
|              +---RESAVE
+---RESAVE
```

以下に、各通信内容を示す。

① 乱流エネルギーの最大値
　エネルギー散逸率の最大値

② 壁近傍の乱流エネルギー（領域境界の更新）
　壁近傍のエネルギー散逸率（領域境界の更新）

③ セル間距離（上流化の場合のみ）（領域境界の更新）

④ 近似流速（次元分）（領域境界の更新）
　温度（領域境界の更新）
　乱流エネルギー（領域境界の更新）
　エネルギー散逸率（領域境界の更新）

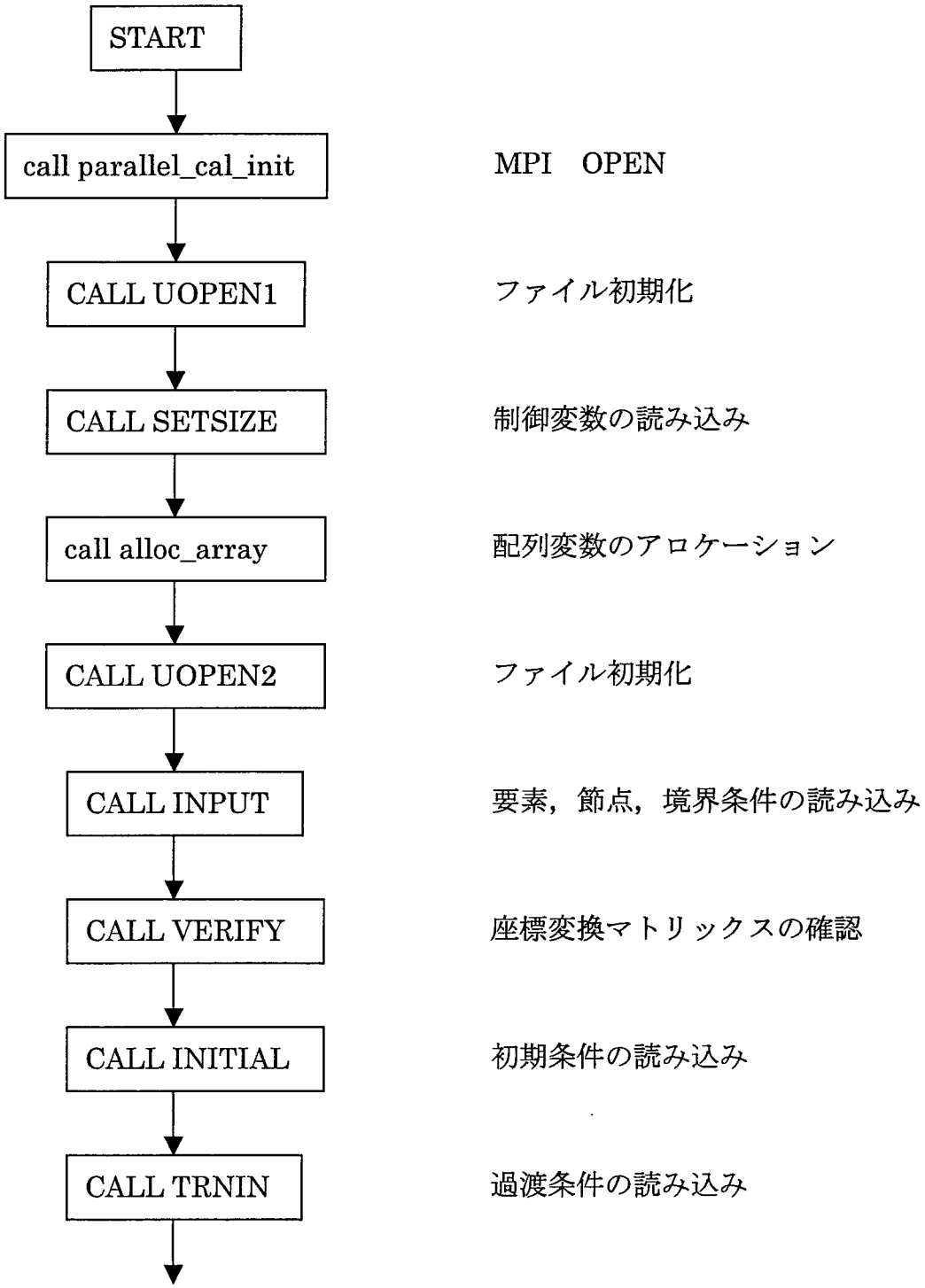
⑤ 圧力（領域境界の更新）

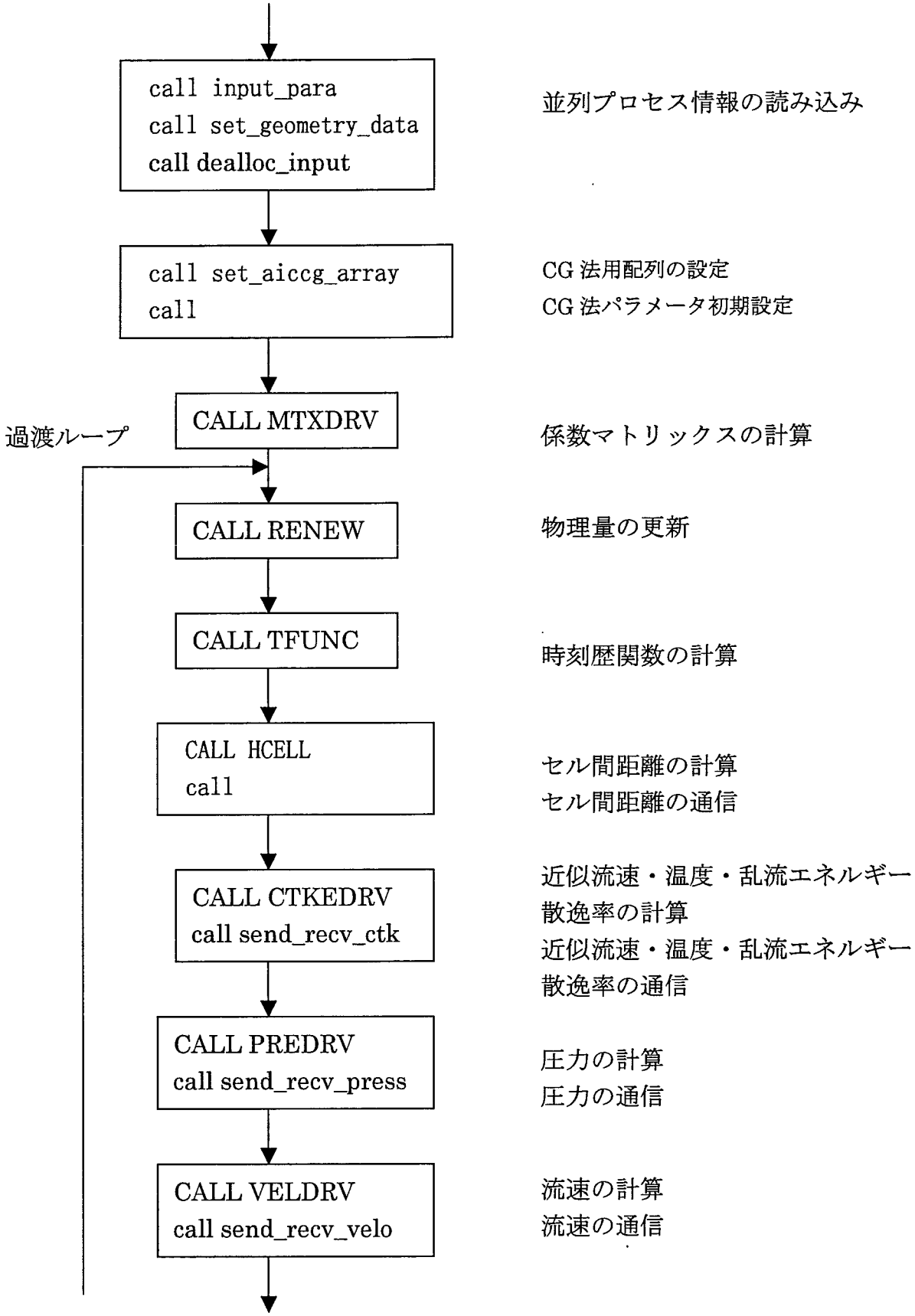
⑥ 流速（次元分）（領域境界の更新）

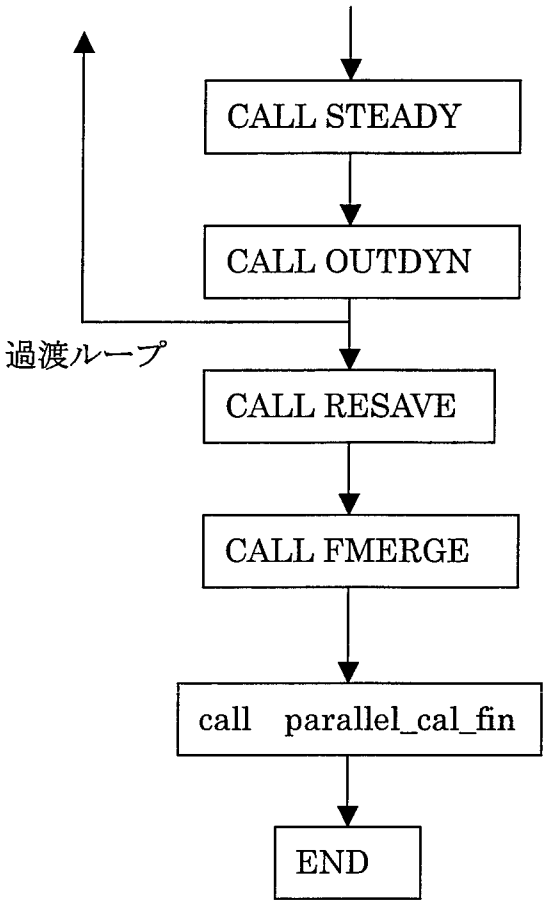
⑦ 圧力方程式連立ソルバー内での行列演算（領域境界の更新と内積）

⑧ 定常の判定（主要変数の最大値・最大変化量）

A．2　フローチャート


START
call parallel_cal_init
MPI　OPEN
CALL UOPEN1
ファイル初期化
CALL SETSIZE
制御変数の読み込み
call alloc_array
配列変数のアロケーション
CALL UOPEN2
ファイル初期化
CALL INPUT
要素，節点，境界条件の読み込み
CALL VERIFY
座標変換マトリックスの確認
CALL INITIAL
初期条件の読み込み
CALL TRNIN
過渡条件の読み込み


call input_para
call set_geometry_data
call dealloc_input
並列プロセス情報の読み込み
call set_aiccg_array
call
CG 法用配列の設定
CG 法パラメータ初期設定
過渡ループ
CALL MTXDRV
係数マトリックスの計算
CALL RENEW
物理量の更新
CALL TFUNC
時刻歴関数の計算
CALL HCELL
call
セル間距離の計算
セル間距離の通信
CALL CTKEDRV
call send_recv_ctk
近似流速・温度・乱流エネルギー
散逸率の計算
近似流速・温度・乱流エネルギー
散逸率の通信
CALL PREDRV
call send_recv_press
圧力の計算
圧力の通信
CALL VELDRV
call send_recv_velo
流速の計算
流速の通信



定常の判定

結果の出力

再スタートファイルの書き出し

出力ファイルのマージ

MPI CLOSE 処理

A．３　配列変数管理

（１）　COMMON・Local 配列からモジュールへ変更

従来の配列の固定サイズ方式から FORTRAN90 のダイナミックアロケーション機能を使った可変サイズ方式に変更した。これに伴い、従来の COMMON 変数を全てモジュール宣言に切り替えた。また、同様な理由で一部の Local 配列をモジュール化し可変サイズ方式に改めた。

m_header	旧 HEADER.inc
m_gauss	旧 GAUSS.inc
m_model	旧 MODEL.inc
m_pcg	旧 PCG.inc
m_COVVI	旧 sub.COVVI 内 Local 配列
m_HCELL	旧 sub. HCELL 内 Local 配列
m_KEHIGH	旧 sub. KEHIGH 内 Local 配列
m_KEMVI	旧 sub. KEMVI 内 Local 配列
m_RESTRW	旧 sub. RESTRW 内 Local 配列
m_STEADY	旧 sub. STEADY 内 Local 配列
m_TEMVI	旧 sub. TEMVI 内 Local 配列
m_merge	出力ファイルマージ作業用配列
m_geometry_data	並列処理用共通変数定義
m_geometry_parameter	並列処理用共通変数定義
m_iccg_matrix	並列処理用共通変数定義
m_iccg_parameter	並列処理用共通変数定義
m_parallel_iccg_data	並列処理用共通変数定義
m_parallel_var_dof	並列処理用共通変数定義
m_sorted_node	並列処理用共通変数定義
m_work_4_connect	並列処理用共通変数定義

（2） サイズパラメータ（入力データ）

配列サイズを決定するパラメータは、以前のparameter文（parameter.inc）ではなく、入力変数を使用してダイナミックアロケーションを行う。したがって、入力データの解析規模に応じてプログラムサイズを変更して再コンパイルする作業の必要が無くなる。

以下に、配列サイズの決定に使用するパラメータを示す。各モジュール内の配列のサイズ定義式で参照する。これ以外に、整数値で示すものはサイズが固定でありスタティックな割り当ての配列である。サイズ指定のない変数はスカラ変数である。

N1 = NPOINg	N11 = MANTSg	N21 = NVEBCg
N2 = NELEMg	N12 = NPROPg	N22 = NVNBCg
N3 = NDIMEg	N13 = NFUNCg	N23 = NPEBCg
N4 = No used	N14 = NDATAg	N24 = NPNBCg
N5 = No used	N15 = No used	N25 = NTEBCg
N6 = No used	N16 = No used	N26 = NTNBCg
N7 = NNODEg	N17 = MFRONg	N27 = NKEBCg
N8 = NSURFg	N18 = No used	N28 = NKNBCg
N9 = NEDGEg	N19 = No used	N29 = NEEBCg
N10 = LCONVg	N20 = NWLBCg	N30 = NENBCg

（3）各モジュールのコンテンツ

● モジュール m_header（旧 HEADER.inc）

・タイトル（旧 COMMON /CNTLT/）

TITLEg character(len=80) タイトル

・制御変数（旧 COMMON /CNTL1/）

NPOINg 節点総数

NELEMg	要素総数
NDIMEg	問題の次元数
NTYPEg	問題の座標系
NNODEg	節点数 / 要素
NSURFg	節点数 / 境界面
NEDGEg	境界面数 / 要素
NRESTg	計算開始時間ステップ数
NSTEPg	計算実行時間ステップ数
NINTVg	出力ステップ間隔
NMATSg	物性の数
NPROPg	物性の種類
NFUNCg	時刻歴関数の数
NDATAg	時刻歴関数に与えるデータの数
NGAUSg	ガウス数値積分点数（ 線要素 ）
NGAUTg	ガウス数値積分点数（ 三角形要素 ）
NITERg	最大反復計算回数
NLUMPg	流速方程式の計算オプション
NBANDg	圧力方程式の計算オプション
NTEMPg	温度方程式の計算オプション
NTURBg	乱流モデルの計算オプション
NPGVEg	流速方程式の風上化オプション
NPGTEg	温度方程式の風上化オプション
NPGKEg	乱流方程式の風上化オプション
NSUPGg	移流項の計算オプション

・境界条件，その他（旧　COMMON　/CNTL2/）

NVEBCg	流速の基本境界条件を与える節点の数
NVNBCg	流速の自然境界条件を与える境界面の数
NPEBCg	圧力の基本境界条件を与える節点の数
NPNBCg	圧力の自然境界条件を与える境界面の数
NTEBCg	温度の基本境界条件を与える節点の数
NTNBCg	温度の自然境界条件を与える境界面の数
NKEBCg	乱流エネルギーの基本境界条件を与える節点の数
NKNBCg	乱流エネルギーの自然境界条件を与える境界面の数
NEEBCg	散逸率の基本境界条件を与える節点の数

NENBCg	散逸率の自然境界条件を与える境界面の数
NWLBCg	壁関数を用いる境界面の数
MLUMPg	流速方程式のバンド幅
MBANDg	圧力方程式のバンド幅
MTEMPg	温度方程式のバンド幅
MFRONg	バンド幅
MNODEg	節点数 / 要素
MSURFg	節点数 / 境界面
MGASPg	ガウス数値積分点数

・計算時刻，重力定数（旧　COMMON　/CNTL3/）

TRESTg	開始時刻
TIMECg	計算時刻
DELTTg	計算時刻の刻み幅
EPSTRg	定常判定値
EPSTSg	定常判定値
GRAVCg	重力定数
CMASSg	（未使用）
CPOENg	（未使用）
CKIENg	（未使用）
CTOTLg	（未使用）

・物性値（旧　COMMON　/PROPS/）

DENSIg	密度
DVISCg	動粘性係数
ACONDg	温度伝導率
BEXPAg	体積膨張係数

・ 要素，節点（旧　COMMON　/IVARI/）

NNODSg(N2)	要素を構成する節点の数
LNODSg(N2,N7)	要素を構成する節点の番号
MATNOg(N2)	物性の番号

・ 境界条件（旧　COMMON　/IBOUN/）

LVEBCg(N21)　流速の基本境界条件を与える節点の番号
LVETPg(N3,N21)　= 0 : Free-Slip , = 1 : Non-Slip
LVEFNg(N3,N21)　流速の基本境界条件に用いる時刻歴関数の番号
LVNBCg(N22)　流速の自然境界条件を与える境界面が属する要素の番号
LVNNSg(N22)　流速の自然境界条件を与える境界面を構成する節点の数
LVNNDg(N8,N22)　流速の自然境界条件を与える境界面を構成する節点の番号
LVNTPg(N3,N22)　（未使用）
LVNFNg(N3,N22)　流速の自然境界条件に用いる時刻歴関数の番号
LPEBCg(N23)　圧力の基本境界条件を与える節点の番号
LPEFNg(N23)　圧力の基本境界条件に用いる時刻歴関数の番号
LPNBCg(N24)　圧力の自然境界条件を与える境界面が属する要素の番号
LPNNSg(N24)　圧力の自然境界条件を与える境界面を構成する節点の数
LPNNDg(N8,N24)　圧力の自然境界条件を与える境界面を構成する節点の
LPNFNg(N24)　圧力の自然境界条件に用いる時刻歴関数の番号
LTEBCg(N25)　温度の基本境界条件を与える節点の番号
LTEFNg(N25)　温度の基本境界条件に用いる時刻歴関数の番号
LTNBCg(N26)　温度の自然境界条件を与える境界面が属する要素の番号
LTNNSg(N26)　温度の自然境界条件を与える境界面を構成する節点の数
LTNNDg(N8,N26)　温度の自然境界条件を与える境界面を構成する節点の番号
LTNFNg(N26)　温度の自然境界条件に用いる時刻歴関数の番号
LKEBCg(N27)　乱流エネルギーの基本境界条件を与える節点の番号
LKEFNg(N27)　乱流エネルギーの基本境界条件に用いる時刻歴関数の番号
LKNBCg(N28)　乱流エネルギーの自然境界条件を与える境界面が属する要素の番号
LKNNSg(N28)　乱流エネルギーの自然境界条件を与える境界面を構成する節点の数
LKNNDg(N8,N28)　乱流エネルギーの自然境界条件を与える境界面を構成する節点の番号
LKNFNg(N28)　乱流エネルギーの自然境界条件に用いる時刻歴関数番号
LEEBCg(N29)　散逸率の基本境界条件を与える節点の番号
LEEFNg(N29)　散逸率の基本境界条件に用いる時刻歴関数の番号

LENBCg(N30)	散逸率の自然境界条件を与える境界面が属する要素の番号
LENNSg(N30)	散逸率の自然境界条件を与える境界面を構成する節点の数
LENNDg(N8,N30)	散逸率の自然境界条件を与える境界面を構成する節点の番号
LENFNg(N30)	散逸率の自然境界条件に用いる時刻歴関数の番号
LWLBCg(N20)	壁関数を用いる境界面が属する要素の番号
LWSNSg(N20)	壁関数を用いる固体側境界面を構成する節点の数
LWSNDg(N8,N20)	壁関数を用いる固体側境界面を構成する節点の番号
LWFNSg(N20)	壁関数を用いる流体側境界面を構成する節点の数
LWFNDg(N8,N20)	壁関数を用いる流体側境界面を構成する節点の番号
NWALLg(N2)	要素が属する境界の属性

・ 流速，圧力，温度，乱流（旧　COMMON　/RVARI/）

COORDg(N1,N3)	節点座標値
CORVEg(N1,N3)	近似流速
VELOCg(N1,N3)	現時刻の流速
VELOPg(N1,N3)	前時刻の流速
PRESSg(N1)	現時刻の圧力
PRESPg(N1)	前時刻の圧力
TEMPEg(N1)	現時刻の温度
TEMPPg(N1)	前時刻の温度
XKTBLg(N1)	現時刻の乱流エネルギー
XKTBPg(N1)	前時刻の乱流エネルギー
XETBLg(N1)	現時刻の散逸率
XETBPg(N1)	前時刻の散逸率

・その他（旧　COMMON　/RVARL/）

ACCEVg(N7,N3)	流速方程式における質量項
CONVVg(N7,N3)	流速方程式における対流項
DIFFVg(N7,N3)	流速方程式における粘性項
PRESVg(N7,N3)	流速方程式における圧力項
FORCVg(N7,N3)	流速方程式における外力項

ACCETg(N7)	温度方程式における質量項
CONVTg(N7)	温度方程式における移流項
DIFFTg(N7)	温度方程式における拡散項
ACCEKg(N7)	乱流エネルギー方程式における質量項
CONVKg(N7)	乱流エネルギー方程式における移流項
DIFFKg(N7)	乱流エネルギー方程式における拡散項
PRODKg(N7)	乱流エネルギー方程式における生成項
DISSKg(N7)	乱流エネルギー方程式における散逸項
ACCEEg(N7)	散逸率方程式における質量項
CONVEg(N7)	散逸率方程式における移流項
DIFFEg(N7)	散逸率方程式における拡散項
PRODEg(N7)	散逸率方程式における生成項
DISSEg(N7)	散逸率方程式における散逸項

・ 境界条件（旧 COMMON /RBOUN)

EBCOg(N3,N21)	流速の基本境界条件に用いる増倍係数
TNODEg(N3,N3,N21)	流速の基本境界条件に用いる座標変換マトリックス
VNBCOg(N3,N22)	流速の自然境界条件に用いる増倍係数
TRANSg(N3,N3,N22)	流速の自然境界条件に用いる座標変換マトリックス
PEBCOg(N23)	圧力の基本境界条件に用いる増倍係数
PNBCOg(N24)	圧力の自然境界条件に用いる増倍係数
TEBCOg(N25)	温度の基本境界条件に用いる増倍係数
TNBCOg(N26)	温度の自然境界条件に用いる増倍係数
XKEBCg(N27)	乱流エネルギーの基本境界条件に用いる増倍係数
XKNBCg(N28)	乱流エネルギーの自然境界条件に用いる増倍係数
XEEBCg(N29)	散逸率の基本境界条件に用いる増倍係数
XENBCg(N30)	散逸率の自然境界条件に用いる増倍係数
RNVVNg(N3,N22)	流速の自然境界条件を与える境界面に対する法線ベクトル
WSLENg(N20)	壁関数を用いる境界面が属する要素の番号
WSNORg(N3,N20)	壁関数を用いる境界面に対する法線ベクトル
WFLENg(N7,N20)	壁関数を用いる境界面が属する要素の番号
WFNORg(N3,N7,N20)	壁関数を用いる境界面に対する法線ベクトル

WSWGHg(N7,N7,N20)　壁関数を用いる境界面に対する法線ベクトル
HCELLg(N1)　　セル間距離

・ 係数マトリックス，荷重ベクトル（旧　COMMON　/RMTRX/）

ELOADg(N7*N3)	要素荷重ベクトル
GLOADg(N1*N3)	全体荷重ベクトル
GCOEFg(N1)	流速方程式の全体係数マトリックス
GCOEPg(N17,N1)	圧力方程式の全体係数マトリックス
GCMEBg(N17*2,N23)	圧力方程式の境界条件マトリックス
EMASSg(N7,N7,N2)	質量マトリックス
ECONVg(N7,N7,N7,N3,N10)	対流マトリックス
ESFORg(N7,N7,N2),	粘性マトリックス
EBFORg(N7,N2)	外力マトリックス
ECORRg(N7,N7,N3,N2)	流速修正マトリックス
EOMGAg(N8,N22)	流速の境界積分マトリックス
ESGMAg(N8,N24)	圧力の境界積分マトリックス
ETHTAg(N8,N26)	温度の境界積分マトリックス

・ 物性，重力，時刻歴関数（旧　COMMON　/RETAL/）

PROPSg(N11,N12)	物性値
GRAVYg(N3)	重力ベクトル
VALEXg(N13)	計算時刻における時刻歴関数値
TIMEXg(N14)	時刻歴関数に与える時間データ
DATAXg(N13,N14)	時刻歴関数に与える数値データ

・流速・温度・圧力・乱流（旧　COMMON　/RVAGP/）

VELOGg(N3)	流速
TEMPGg	温度
PRESGg	圧力
TKENGg	乱流エネルギー
TEPSGg	散逸率
TPKEGg	生成率
TVISCg	乱流動粘性係数

RESTRg(N3,N3)	レイノルズ応力
BESTRg(N3,N3)	レイノルズ応力
DVDXGg(N3,N3)	流速勾配
DPDXGg(N3)	圧力勾配
DTDXGg(N3)	温度勾配
DKDXGg(N3)	乱流エネルギー勾配
DEDXGg(N3)	散逸率勾配
WPGVEg(N7)	流速方程式における風上化重み関数
WPGTEg(N7)	温度方程式における風上化重み関数
WPGKEg(N7)	乱流エネルギー，散逸率方程式における風上化重み関数

新規追加

IGPOINg(N1)	全体系における節点NO
IGELEMg(N2)	全体系における要素NO
NGPOINg	全体系における節点総節点数
NGELEMg	全体系における節点総節点数

● モジュール m_pcg (旧PCG.inc)

BFORCE(N1)	荷重ベクトル
EPSCGg	CG法の収束条件
SIGMAg	CG法のスケーリングファクター

● モジュール m_COVVI (旧COVVIのLocal配列変数)

TERMV(N3)

● モジュール m_HCELL (旧HCELLのLocal配列変数)

XEAXA(N3)
XEAXB(N3)
CLINE(N3,N3)
CINVR(N3)

CNLNR(N3,N3)
CONST(N3)
CLOAD(N3)

● モジュール m_KEHIGH (旧 KEHIGH の Local 配列変数)

VEXI(N3)
VETA(N3)

● モジュール m_KEMVI (旧 KEMVI の Local 配列変数)

FLUXV(N3,N3)
FLUXT(N3)
FLUXK(N3)
FLUXE(N3)

● モジュール m_RESTRW (旧 RESTRW の Local 配列変数)

WSTR(N3,N3)
EMTX(N3,N3)
WMTX(N3,N3)
VEAV(N3)
VEXI(N3)
VETA(N3)

● モジュール m_STEADY (旧 STEADY の Local 配列変数)

AVMAX(N3)
DVMAX(N3)
RVMAX(N3)

● モジュール m_TEMVI (旧 TEMVI の Local 配列変数)

TERMV(N3)

● モジュール m_gauss

・アイソパラメトリック要素（旧　COMMON　/GAUSS/）

ELCODg(3,27)	要素を構成する節点の座標値
SRCODg(3,27)	境界を構成する節点の座標値
SHAPEg(27)	形状関数
DERIVg(3,27)	形状関数の局所座標系における微係数
DERCNg(3,27)	形状関数の全体座標系における微係数
XJACMg(3,3)	ヤコビアン行列
XJACIg(3,3)	ヤコビアン逆行列
DJACBg	ヤコビアン行列式
DVOLMg	体積要素
POSGPg(5)	ガウス積分点座標値
WEIGPg(5)	ガウス積分点重み係数
POSGTg(7,2)	ガウス積分点座標値
WEIGTg(7)	ガウス積分点重み係数
WEIGHg	重み係数
WEILNg(5)	線要素の重み係数
WEITRg(7)	三角形要素の重み係数
WEISQg(25)	四角形要素の重み係数
WEIWDg(35)	五面体要素の重み係数
WEICBg(125)	六面体要素の重み係数
SF02Lg(2,5)	２節点線要素の形状関数
DF02Lg(1,2,5)	２節点線要素の偏導関数
SF03Lg(3,5)	３節点線要素の形状関数
DF03Lg(1,3,5)	３節点線要素の偏導関数
SF03Tg(3,7)	３節点三角形要素の形状関数
DF03Tg(2,3,7)	３節点三角形要素の偏導関数
SF06Tg(6,7)	６節点三角形要素の形状関数
DF06Tg(2,6,7)	６節点三角形要素の偏導関数
SF04Sg(4,25)	４節点四角形要素の形状関数
DF04Sg(2,4,25)	４節点四角形要素の偏導関数
SF09Sg(9,25)	９節点四角形要素の形状関数
DF09Sg(2,9,25)	９節点四角形要素の偏導関数
SF06Wg(6,35)	６節点五面体要素の形状関数
DF06Wg(3,6,35)	６節点五面体要素の偏導関数

SF18Wg(18,35)	18節点五面体要素の形状関数
DF18Wg(3,18,35)	18節点五面体要素の偏導関数
SF08Cg(8,125)	８節点六面体要素の形状関数
DF08Cg(3,8,125)	８節点六面体要素の偏導関数
SF27Cg(27,125)	27節点六面体要素の形状関数
DF27Cg(3,27,125)	27節点六面体要素の偏導関数

● モジュール m_model (旧 MODEL.inc)

・ 乱流モデル定数 (旧 COMMON /KEM/)

CMU	C(μ)
FMU	F(μ)
ALK	α(k)
ALE	α(ε)
SGK	σ(k)
SGE	σ(ε)
CE1	C(e1)
FC1	F(e1)
CE2	C(e2)
FC2	F(e2)
RKA	κ

・ 乱流モデル定数 (旧 COMMON /RNG)

BETA	β
ETA0	η

・ 乱流モデル定数 (旧 COMMON /ASM/)

CSK	C(sk)
CSE	C(se)
RC1	C(R1)
RC2	C(R2)
RD1	D(R1)
RD2	D(R2)

RCL	C(R1)

● モジュール m_merge（新規追加）

・ 制御変数

NPOINLg	節点総数（全体系）
NELEMLg	要素総数（全体系）
NDIMELg	問題の次元数
TIMECGg	計算時刻

・ 要素，節点，座標

NNODSGg(N2)	要素を構成する節点の数
LNODSGg(N2,N7)	要素を構成する節点の番号
MATNOGg(N2)	物性の番号
GCOORDg(N1,N3)	節点座標値

・ 流速，圧力，温度，乱流

GVELOCg(N1,N3)	現時刻の流速
GPRESSg(N1)	現時刻の圧力
GTEMPEg(N1)	現時刻の温度
GXKTBLg(N1)	現時刻の乱流エネルギー
GXETBLg(N1)	現時刻の散逸率

● モジュール m_parallel_iccg_data（新規追加）

・ 行列インデックス

istack_l	圧力方程式係数行列下三角成分格納アドレス
istack_u	圧力方程式係数行列上三角成分格納アドレス
item_l	圧力方程式係数行列下三角成分インデックス
item_u	圧力方程式係数行列上三角成分インデックス
itotal_l	下三角成分の非ゼロ要素数
itotal_l	上三角成分の非ゼロ要素数

● モジュール m_iccg_matrix (新規追加)

・ 行列・ベクトル

x_vec(N1)	圧力方程式の右辺ベクトル
i_vect(1)	未使用
diag(N1)	圧力方程式係数行列の対角成分
aiccg_l(itotal_l)	圧力方程式係数行列の下三角成分
aiccg_u(itotal_u)	圧力方程式係数行列の上三角成分

● モジュール m_iccg_parameter (新規追加)

・ 圧力方程式制御

eps	圧力方程式の収束条件 (最小残差ノルム)
resid	圧力方程式の残差ノルム
nset	前処理スイッチ
sigma	スケーリングファクタ
sigma_diag	対角スケーリングファクタ
precond(character)	前処理種類 "SSOR" 固定

● モジュール m_parallel_var_dof (新規追加)

・ 並列制御

nprocs	並列プロセス数
my_rank	プロセスのランク
ierr, errno(整数)	通信エラーフラグ
neigh_pe_num	隣接 PE 数
num_import_data	受信外点数
num_export_data	送信内点数
neigh_pe_data	隣接 PE 番号
import_stack	受信外点格納アドレス
import_id_data	受信外点番号
import_ global _id_data	受信外点番号 (未使用)

export_stack	送信内点格納アドレス
export_id_data	送信内点番号
export_global_id_data	送信内点番号（未使用）
wsend	送信バッファー
wrecv	受信バファー
SOLVER_COMM(整数)	コミュニケータ(=MPI_COMM_WORLD)
status(整数)	MPI 用
max_neibpetot	最大隣接 PE 数(=26)

●その他のモジュール

並列処理用に幾つかの作業用配列・変数の定義用に使用した。内容は省略。

m_control_parameter
m_geometry_data
m_geometry_parameter
m_read_geometry_data_dummy
m_sorted_node
m_work_4_connect

A．4　並列化ライブラリー

以下に、SPIRALの並列化に関して作成したライブラリーの機能を示す。

parallel_cal_init	MPI initialize
parallel_cal_fin	MPI close
allocate_aiccg	配列動的アロケーション
allocate_iccg_matrix	配列動的アロケーション
allocate_istack	配列動的アロケーション
allocate_item	配列動的アロケーション
set_aiccg_array	係数行列インデックス作成
cal_inod_ele_max	節点－要素対応テーブル
cal_istack_l	下三角行列用インデックス
cal_istack_u	上三角行列用インデックス
cal_itotal	非ゼロ要素の数の計算
iccg_parameter_init	CG法パラメータ初期設定
quicksort	インデックスソーター
set_off_diag	係数行列インデックス用
sort_node_index	係数行列インデックス用
solver_CG	ICCG法連立方程式ソルバー
solver_BiCGSTAB	BiCGSTAB法連立方程式ソルバー
solver_GPBiCG	GPBiCG法連立方程式ソルバー
hcell_send_recv	境界節点情報通信(セル間距離)
ctk_send_recv	境界節点情報通信(近似流速・温度・乱流量)
xke_send_recv	境界節点情報通信(乱流量)
press_send_recv	境界節点情報通信(圧力)
velo_send_recv	境界節点情報通信(流速)
solver_SR	圧力方程式用境界データ通信ライブラリー（１節点－1自由度用)
solver_SR3	圧力方程式用境界データ通信ライブラリー（１節点－３自由度用)
time_prog_barrier	同期

A. 5 Makefile

(1) SPIRAL本体のMakefile

以下に並列版SPIRALのMakefileを示す。
ソースは、FORTRAN90 ベースでプログラミングしているので、FORTRAN90 に対応した MPI コンパイラーである mpif90 を使用する。 コンパイルオプション F90FLAG は使用するコンパイルによって異なるため、場合によっては変更が必要。下の例で使用しているオプション+r8 は、実数変数と実数定数を倍精度（８バイト）にするものである。
コンパイルの順序はモジュール、その他という形式にしなくてはならない。（コンパイル時に参照されるモジュルは既にコンパイル済みでなければならない。）

```
PROGRAM      =    ../run/mpi
F90           =   mpif90
FFLAGS       =
LFLAGS       =
F90FLAGS      =+r8
CC           =   cc -c
LD           =   mpif90

.SUFFIXES:
.SUFFIXES: .o .f90 .F .c

.f90.o:; $(F90) -c $(F90FLAGS)  $(F90OPTFLAG) $<
.F.o:; $(F90) -c $(F90FLAGS)  $(F90OPTFLAG) $<

OBJECTS = ¥
     m_HEADER.o m_GAUSS.o m_MODEL.o m_PCG.o ¥
     m_control_parameter.o m_geometry_data.o ¥
     m_geometry_parameter.o m_iccg_matrix.o ¥
     m_iccg_parameter.o m_parallel_iccg_data.o ¥
     m_parallel_var_dof.o m_read_geometry_data_dummy.o ¥
```

```
m_sorted_node.o m_work_4_connect.o ¥
m_COVVI.o m_HCELL.o m_KEHIGH.o m_KEMVI.o m_RESTRW.o ¥
m_SOLVPR.o m_STEADY.o m_TEMVI.o m_merge.o ¥
solver_SR.o solver_SR3.o ¥
solver_CG.o solver_BiCGSTAB.o solver_GPBiCG.o ¥
¥
CEF1F.o CEF2F.o CENTER.o CGAUSS.o CMUFF.o COVEM.o ¥
COVVI.o CTKEDRV.o CTKEEM.o CTKEGL.o CTKEVI.o ¥
HCELL.o HINVR.o HSUPG.o INITIAL.o INPUT.o ¥
JACOBS.o JACOBV.o KEHIGH.o KELOW.o KEMEM.o KEMVI.o ¥
LUBKSB.o LUDCMP.o MTXDRV.o MTXEM.o MTXGL.o MTXSI.o ¥
MTXVI.o NGAUSS.o OUTDYN.o PLOTST.o PLOTTR.o PREDRV.o ¥
PREEM.o PREGL.o REASMV.o REASMW.o RELOAD.o RELOWV.o ¥
RELOWW.o RENEW.o RESAVE.o RESTRV.o RESTRW.o ¥
SDWSET.o SDWSTA.o SETCONST.o SETSIZE.o SFDF.o ¥
SOLVCV.o SOLVKE.o SOLVMX.o SOLVPR.o SOLVTE.o SOLVVE.o ¥
SPIRAL.o STEADY.o TEMEM.o TEMVI.o ¥
TFUNC.o TRNIN.o UOPEN1.o UOPEN2.o VELDRV.o VELEM.o ¥
VELGL.o VERIFY.o WFUNC.o ¥
¥
alloc_array.o allocate_aiccg.o allocate_iccg_matrix.o ¥
allocate_istack.o allocate_item.o cal_inod_ele_max.o ¥
cal_istack_l.o cal_istack_u.o cal_itotal.o ¥
dealloc_input.o ¥
iccg_parameter_init.o input_para.o parallel_cal_fin.o ¥
parallel_cal_init.o send_recv_press.o quicksort.o ¥
set_aiccg_array.o set_geometry_data.o set_off_diag.o ¥
sort_node_index.o time_prog_barrier.o ¥
send_recv_press.o send_recv_velo.o send_recv_ctk.o ¥
send_recv_hcell.o send_recv_xke.o ¥
¥
FMERGE.o UOPENM.o alloc_buff.o dealloc_buff.o ¥
OMERGE.o UMERGE.o dealloc_array.o

.f.o:
	@echo "   Compiling $(PROGRAM) Fortran source files"
```

```
	$(FC) $(FFLAGS) $<

.c.o:
	@echo "   Compiling $(PROGRAM) C source files"
	$(CC) $(CFLAGS) $<

$(PROGRAM): $(OBJECTS)
	$(LD) $(LFLAGS) -o $(PROGRAM) $(OBJECTS)
```

（2）パーティショナーの Makefile

以下に、パーティショナーの Makefile を示す。
SPIRAL 本体と同様に FORTRAN90 でプログラミングしているため、FORTRAN90 に対応したコンパラーが必要である。

```
PROGRAM      =   part.run
F90           =   f90
FFLAGS       =
LFLAGS       =
CC            =   cc -c
LD            =   f90

.SUFFIXES:
.SUFFIXES: .o .f90 .F .c

.f90.o:; $(F90) -c $(F90FLAGS)  $(F90OPTFLAG) $<
.F.o:; $(F90) -c $(F90FLAGS)  $(F90OPTFLAG) $<

OBJECTS = partitioner.o analyzer.o ¥
  geofem_tiger.o calc_edgcut.o ¥
  cre_local_data.o create_metis_input.o interface_nodes.o ¥
 local_data.o neib_pe.o paraset.o proc_local.o rcm_gibbs.o util.o ¥
 spiral_geofem.o alloc_array.o

.f.o:
```

```
	@echo "   Compiling $(PROGRAM) Fortran source files"
	$(FC) $(FFLAGS) $<

.c.o:
	@echo "   Compiling $(PROGRAM) C source files"
	$(CC) $(CFLAGS) $<

$(PROGRAM): $(OBJECTS)
	$(LD) $(LFLAGS) -o $(PROGRAM) $(OBJECTS)
```

付録B　並列計算の運用方法

B．1　計算機環境

（1）計算機の種類

共有メモリー型並列計算機、ワークステーションクラスター、単にネットワークを介したワークステーション上で動作する。

使用可能なオペレーティングシステムはUNIX(LINUXを含む)を対象とする。コンパイラーとしてFORTRAN90、並列化プログラミングツールとしてMPIがインストールされていることを条件とする。

（2）ファイルシステム

共有メモリー型並列計算機やワークステーションクラスターでは、通常、全てのプロセッサーが同一ディスクを共有する共有ファイルシステムを使用するのに対し、単にネットワークを介したワークステーションでは、それぞれのワークステーションで独立したファイルシステムを持つ。

単にネットワークを介したワークステーションで並列版SPIRALを実行する場合は、使用するワークステーションの個々のディスク上に同一のディレクトリを構成してロードモジュールと入力ファイルを設定しなくてはならない。

B．2　実行手続きの概要

ここでは、並列版 SPIRAL のデータ作成から実行までの手続きについて説明する。

手続きの流れは以下のようになる。

① オリジナルの SPIRAL 実行データを作成する。(全体系でのデータの作成)

② パーティショナーにより各並列プロセス用データを作成する。

③ 並列計算を実行する。

ここから②以降の手続きについて実際のコマンド入力方法等について説明する。

(1) メッシュの領域分割と並列用入力データの生成

パーティショナーと Metis の二つのプログラムを使い分けて領域分割と並列用入力データを作成する。
領域分割法として RCB を選択するか Metis を選択するかで手続きが異なる。

RCB の場合は、一回のパーティショナーの実行のみであるのに対し、Metis を使用する場合は、二回のパーティショナーの起動と、一回の Metis の起動という三段階の手続きを踏む。(一回目のパーティショナーの起動は Metis 用入力データを作成するため。Metis の起動は Metis のアルゴリズムで領域分割を行うため、二回目のパーティショナーの起動は、Metis の出力情報を元に並列用入力データの作成を行うためである。)

この段階で使用するファイルは以下のものである。ただし、解析者が最初に用意しておく必要があるものは、全体系に対する SPIRAL 実行用データだけである。

SPIRAL 実行データ

全体領域に対する SPIRAL 実行データで、オリジナルと共通である。Partitioning 時のみに使用し、SPIRAL 実行時には使用しない。(1PE 計算時は使用可能である。ただし、SPIRAL が読み込まない不要なデータは削除しておく必要がある。)

分割後の小領域に対する SPIRAL データ

領域分割後の小領域に対する各 SPIRAL 並列プロセスに対する実行データ。様式はオリジナルに近く、小領域内の接点・要素・境界条件・初期条件及び共通制御データに加え、全体系での接点・要素 No と他のプロセスとの通信に必要なデータを含む。

Metis 用入力データ

Metis 領域分割法を選択してパーティショナーを実行した時にパーティショナーが SPIRAL 実行用データの要素・節点情報を元に作成するファイル。Metis 実行時の入力データとして使用する。

Metis 出力データ

Metis 領域分割法を選択してパーティショナーを実行した時の入力データとして Metis が作成するデータファイル。

B．3　領域分割ツール（パーティショナー）の使用方法

（1）RCB 法による領域分割で並列用入力ファイルを作成する場合

パーティショナーを起動する。（ここで影を付けた部分が解析者の入力項目である。）

実行コマンド

part.run

```
:   INPUT FILE NAME =
```

Solv.inp

```
:   SPIRAL DATA END
: *IEDGTOT=     3093    6360
: ********************
: ||                                ||
: ||   GeoFEM Partitioner Ver.5.00  ||
: ||                                ||
: ********************
:#   select PARTITIONING METHOD
:     RCB                                    (1)
:     未使用                                 (2)
:     MeTiS/RSB                              (3)
:     generate MeTiS/RSB INPUT               (4)
:  Please TYPE 1, 2, 3 or 4  !!
: >>>
```

1

```
: 1
: ***  RECURSIVE COORDINATE BISECTION   (RCB)
: How many partitions (2**n) ?
```

2

```
: 2
```

```
: ＊＊＊  4  REGIONS
:    0      Solv.inp.0
:    1      Solv.inp.1
:    2      Solv.inp.2
:    3      Solv.inp.3
: ##### 1-th BISECTION #####
: in which direction？ X：1, Y：2, Z：3
: >>>
        1
:  X-direction
: ##### 1-th BISECTION #####
: in which direction？ X：1, Y：2, Z：3
: >>>
        2
:  Y-direction
以下省略
```

（2）Metis による領域分割で並列用入力ファイルを作成する場合

① MeTiS 入力データを作成する

実行コマンド

part.run

```
:   INPUT FILE NAME =

      Solv.inp

:   SPIRAL DATA END
: *IEDGTOT=     3093    6360
: ********************
: ||                                ||
: ||   GeoFEM Partitioner Ver.5.00   ||
: ||                                ||
: ********************
:#   select PARTITIONING METHOD
:     RCB                                    (1)
:     未使用                                 (2)
:     MeTiS/RSB                              (3)
:     generate MeTiS/RSB INPUT               (4)
:  Please TYPE 1, 2, 3 or 4  !!
: >>>

      4
: generate MeTiS/RSB INPUT
: ***  GRID file
:#   MeTiS input file ?
: >>>
      Solv.inp.metis
***  MeTiS/RSB input   Solv.inp.metis
```

② MeTiS の実行

実行コマンド

kmetis　入力ファイル名　分割数

入力ファイルを　Solv.inp.metis　分割数を４とした場合、以下のファイルが生成
される。

Solv.inp.metis.part.4

このファイルを（３）MeTiS/RSB を選択した場合の partition-table file として
使用する

3
：3
：＊＊＊ MeTiS/RSB partition-table FILE？
>>>

Solv.inp.metispart.4

以下省略

③ MeTiS を選択する場合

実行コマンド

part.run

```
:   INPUT FILE NAME =
```

Solv.inp

```
:   SPIRAL DATA END
: *IEDGTOT=     3093    6360
: ********************
: ||                                ||
: ||   GeoFEM Partitioner Ver.5.00  ||
: ||                                ||
: ********************
:#   select PARTITIONING METHOD
:     RCB                                  (1)
:     未使用                               (2)
:     MeTiS/RSB                            (3)
:     generate MeTiS/RSB INPUT             (4)
:  Please TYPE 1, 2, 3 or 4  !!
: >>>
```

3

```
: 3
: ***  MeTiS/RSB partition-table FILE ?
>>>
```

Solv.inp.metispart.4

以下省略

B．4　並列計算実行コマンドの使用方法

実行コマンド

mpirun　-np プロセス数 -machinefile ホスト名リスト　SPIRAL ロードモジュール名

SPIRAL ロードモジュールとB．3で作成した全ての入力ファイルを同一ディレクトリに置き、上記コマンドを入力する。

入力ファイル

・実行データ（プロセス数分）

・FILENAME（実行データのファイル名を指定するファイル）

Solv.inp というファイル名が指定されると、m+1 個のプロセスに対して Solv.inp.0、Solv.inp.1・・・・・Solv.inp.m という入力ファイルがあると認識される。

出力ファイル

・ OUTPUT	実行結果ファイル
・ PLOTOUT	作図用ファイル
・ RESTART.0 ～ RESTART.m	リスタートファイル

（注意）１プロセスだけの実行の場合、OUTPUT、PLOTOUT の代わりに OUTPUT.0、PLOTOUT.0 というファイルが作成される。