文部科学省次世代ＩＴ基盤構築のための研究開発
「革新的シミュレーションソフトウエアの研究開発」

RSS21 フリーソフトウエア

HEC ミドルウェア（HEC-MW）

PC クラスタ用ライブラリ型 HEC-MW

（hecmw-PC-cluster） バージョン 2.01

線形ソルバライブラリ マニュアル

お問い合わせ先
（公開／契約窓口） （財）生産技術研究奨励会
〒153-8505 東京都目黒区駒場４－６－１
（ソフトウェア管理元） 東京大学生産技術研究所 計算科学技術連携研究センター
〒153-8505 東京都目黒区駒場４－６－１
Fax : 03-5452-6662
E-mail : software@rss21.iis.u-tokyo.ac.jp

目　　次

1. ソフトウェアの概要

1.1 概 要

「HEC ミドルウェア（HEC-MW)」プロジェクトにおいては，有限要素法（FEM)，有限体積法（FVM）など，非構造格子を使用した科学技術シミュレーションにおける計算手法の計算処理パターン，それらの連成に必要な共通インタフェースを抽出し，ハードウェアに依存しない開発基盤，すなわち HEC ミドルウェア（HEC-MW）の機能設計，開発，実装を実施する。PC で開発されたソースプログラム（FORTRAN90 または C 言語で記述）をネットワーク上の各ハイエンド計算機にインストールされた HEC-MW にプラグインすることにより，PC クラスタからベクトル並列計算機まで，それぞれのハードウェアに対して最適化されたコードが自動的に生成される。

有限要素法によるシミュレーションコードは：

- データ入出力
- マトリクス生成
- 線形ソルバー
- 領域間通信

などのいくつかの典型的なプロセスから構成されている。これらのプロセスを各ハードウェアに対して最適化するとともに，FEM 本体の開発者から見ると，同じインタフェースで使用可能である必要がある。特に領域間通信部分に関しては，開発者から隠蔽することが重要である。

本マニュアルは PC クラスタ向け HEC-MW ライブラリの一部である反復法による線形ソルバーに関する説明書である。

HEC-MW では，反復法としては：

- Conjugate Gradient （CG)
- Bi-Conjugate Gradient Stabilized （BiCGSTAB)
- Generalized Minimal Residual （GMRES)
- Generalized Product-type Bi-Conjugate Gradient (GPBiCG)

を利用できる。1 節点に 1 自由度のスカラーソルバーのほか，1 節点に多自由度が存在する場合に関するブロックタイプのソルバーも用意する。ブロック内の自由度が変化する場合，一つのプログラムの中でブロックサイズが変化する場合に関しても対応する。

前処理手法としては，以下の手法に対応する。

- 局所 ILU(k)／IC(k)法
- Block Scaling, Point Jacobi
- SAI (Sparse Approximate Inverse)
- Algebraic Multigrid (Algebraic Multigrid)
- Selective Blocking

1.2 反復法を使用した有限要素法の並列化

有限要素法（Finite Element Method, FEM）における典型的な処理プロセスは以下の通りである（Fig.1）：

- プリ･プロセッシング（例：メッシュ生成）
- シミュレーション本体（例：構造解析，熱流体解析）
- ポスト・プロセッシング（例：可視化，データマイニング）

並列計算で扱うデータのサイズ（メッシュ数）は非常に大きいため，全体領域を一括して取り扱うことは困難で，全体データを部分領域（局所データ）に分割する必要がある。それぞれの領域（domain，partition）は並列計算機の各プロセッサ（Processing Element，PE）に割り当てられる（Fig.2）。有限要素法は差分法などと比較して並列化が困難であると考えられてきた。間接データ参照があるため，1プロセッサ（Processing Element，PE）あたりの計算効率は差分法と比較して低いが，有限要素法の処理は基本的に要素単位の局所的な処理が中心となるため並列化には適している。

FEMの処理は，線形，非線形いずれの場合も，支配方程式を線形化し，要素単位で重みつき残差法を適用して得られる要素剛性マトリクスを足し合わせて得られる，疎な全体剛性マトリクスを係数とする大規模連立一次方程式を解くことに帰着される。FEMの計算のほとんどの部分は：

- 係数マトリクス生成
- 線形ソルバーによる大規模連立一次方程式求解

に費やされる。係数マトリクス生成に関しては，要素単位に実行が可能なため，並列化は容易であり，領域間の通信無しに実行することが可能である。並列FEMの計算において，通信が発生する可能性があるのは線形ソルバーの部分のみである。したがって，線形ソルバーの部分を除くと1CPUのPC向けに開発されたFEMコードは並列計算機上で容易に動かすことが可能である。

領域間の通信は Fig.2 に示すように線形ソルバーの部分だけで生じる。この特性を最大限利用し，適切なデータ構造を設定，並列計算に適した反復法を採用することによって後述するように 95%を越えるような並列化効率を達成することも可能である。

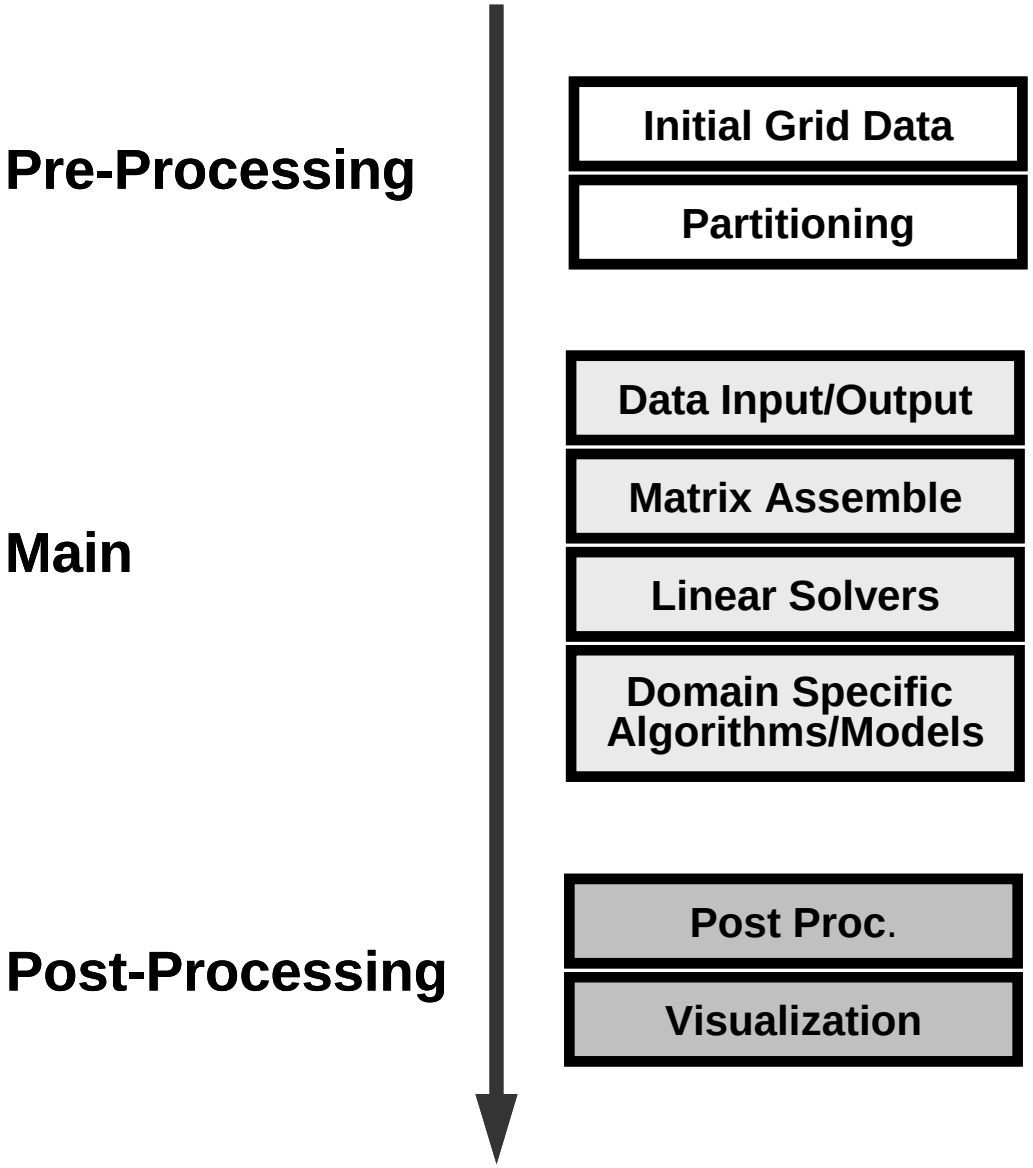


Fig. 1 Parallel FEM Procedure

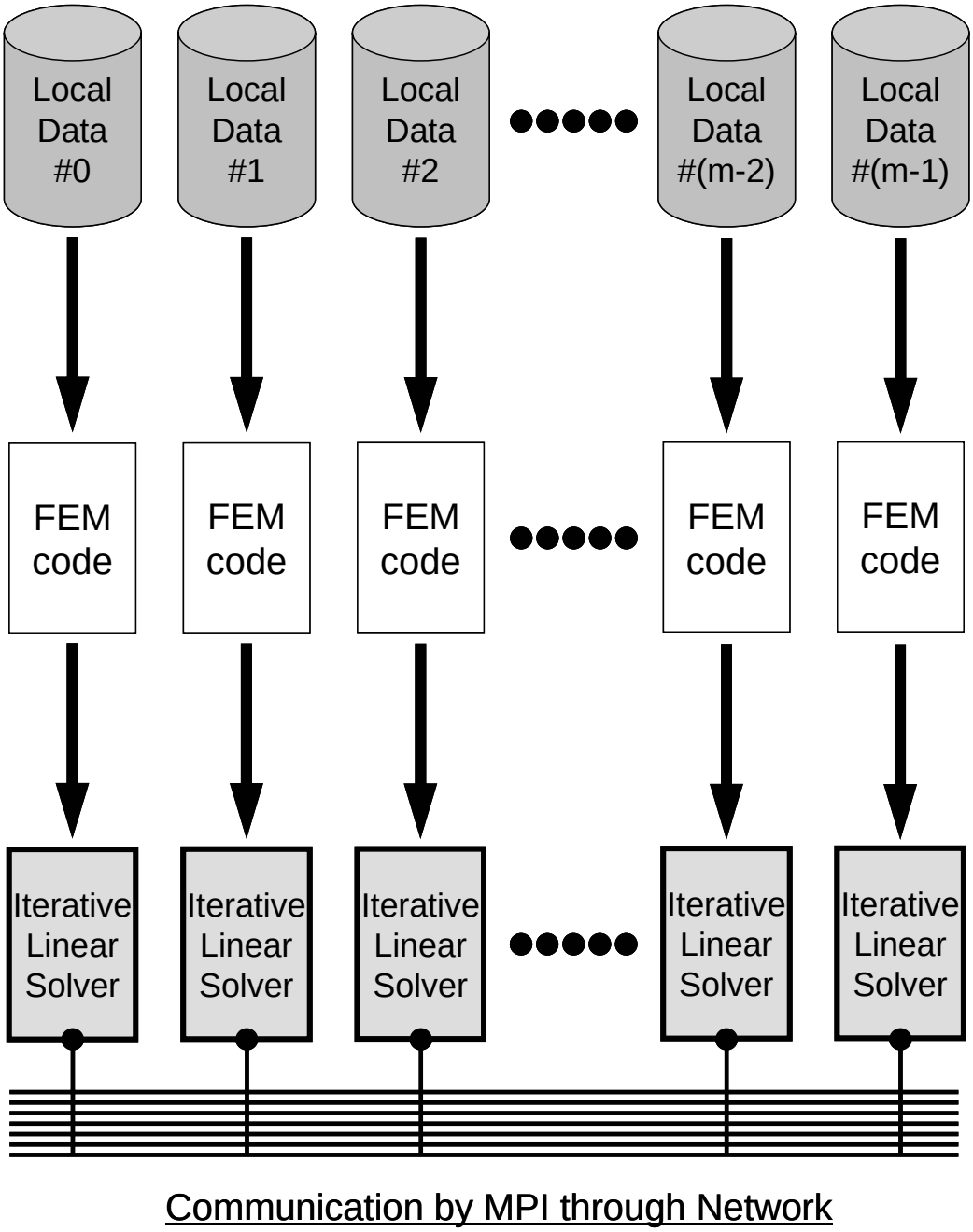


Fig. 2 Parallel FEM procedure and distributed local data sets in HEC-MW 〔5,6,7〕

1.3 分散データ構造

FEMに代表される非構造格子（Unstructured Mesh）を使用したアプリケーションにおいて，適切な分散データ構造を決定することは，並列計算を効率的に実施する上で重要である。HEC-MWの局所メッシュデータは節点ベース（node-based）および要素ベース（element-based）の領域分割の両者をサポートしているが，ここでは，領域間オーバーラップ要素を含む節点ベース領域分割を前提とする。

FEMにおいて，速度，温度，変位など，線形方程式の解となるような変数は節点において定義される。従って，並列計算における効率という観点からは領域間の節点数が均等であることが望ましい。節点ベースの領域分割を使用した場合，剛性マトリクス生成に代表されるような要素単位の処理を各領域において局所的に実施するためには，領域間のオーバーラップ要素が必要である。Fig.3はこのようなオーバーラップ要素の例を示したものである。ここで，灰色に塗られた要素は複数の領域によって共有されており，各領域における各節点に対する剛性マトリクスの足し込みなどの処理を，並列に実施するためにはこれらオーバーラップ要素の情報が必要である。

HEC-MWでは領域間の通信の記述にはMPI〔15〕を使用している。差分法などに使用されている構造格子（Structured Grids）に関してはMPI固有の領域間通信用のサブルーチンが準備されているが，有限要素法に代表される非構造格子（Unstructured Grids）では，プログラム開発者が独自にデータ構造と領域間通信を設計しなくてはならない。

HEC-MWにおいて，各領域は以下の情報を含んでいる：

- 各領域に割り当てられた節点
- 各領域に割り当てられた節点を含む要素
- 他の領域に割り当てられているが上記の要素に含まれている節点
- 領域間の通信テーブル（送信，受信用）
- 節点グループ，要素グループ，面グループ
- 材料物性

節点は，通信という観点から以下の3種類に分類される：

- 内点（Internal Nodes）：各領域に割り当てられた節点
- 外点（External Nodes）：他領域に属しているが，各領域の要素に含まれている節点
- 境界点（Boundary Nodes）：他領域の外点となっている節点

Fig.4は，局所データの実例である。Fig.4とFig.5におけるPE#2において，節点は以下のように分類される：

- 内点　{1,2,3,4,5,6}
- 外点　{7,8,9,10,11,12}
- 境界点 {1,2,5,6}

局所データには領域間の通信テーブルの情報も含まれる。境界点における値は隣接領域へ「送信（send)」され，送信先では外点として「受信（receive)」される。Fig.4に示す局所データ構造とFig.5に示す領域間通信によって非常に高い並列性能が達成されている〔3,5,6,7,8,9〕。

HEC-MWでは全体データから局所データを自動的に生成するためのツールとして領域分割用ユーティリティが用意されている。利用者は実際には上記の通信テーブルについては陽に意識することなく並列有限要素法コードの開発，利用が可能である。領域分割にあたっては：

- 各領域の負荷が均等となっていること
- 領域間の通信が少ないこと

が重要である。特に前処理つき反復法を使用する場合には収束を速めるために（2）が重要なポイントである〔7〕。この両条件を満たす手法としてはMETIS〔14〕が良く知られている。HEC-MW の領域分割ツールでは文献〔12〕で紹介されている RCB 法（Recursive Coordinate Bisection）等のほか，METIS による領域分割も利用できる。

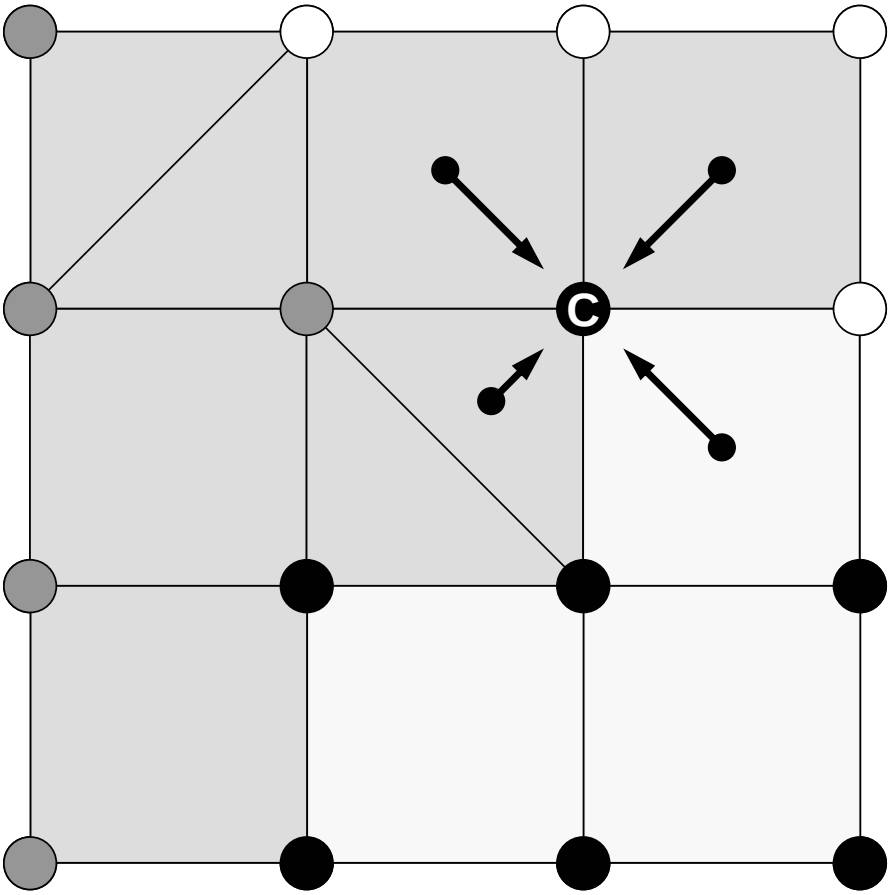


Fig. 3 *Element-by-element* operations around node *C*.
Gray meshes are *overlapped* among domains.

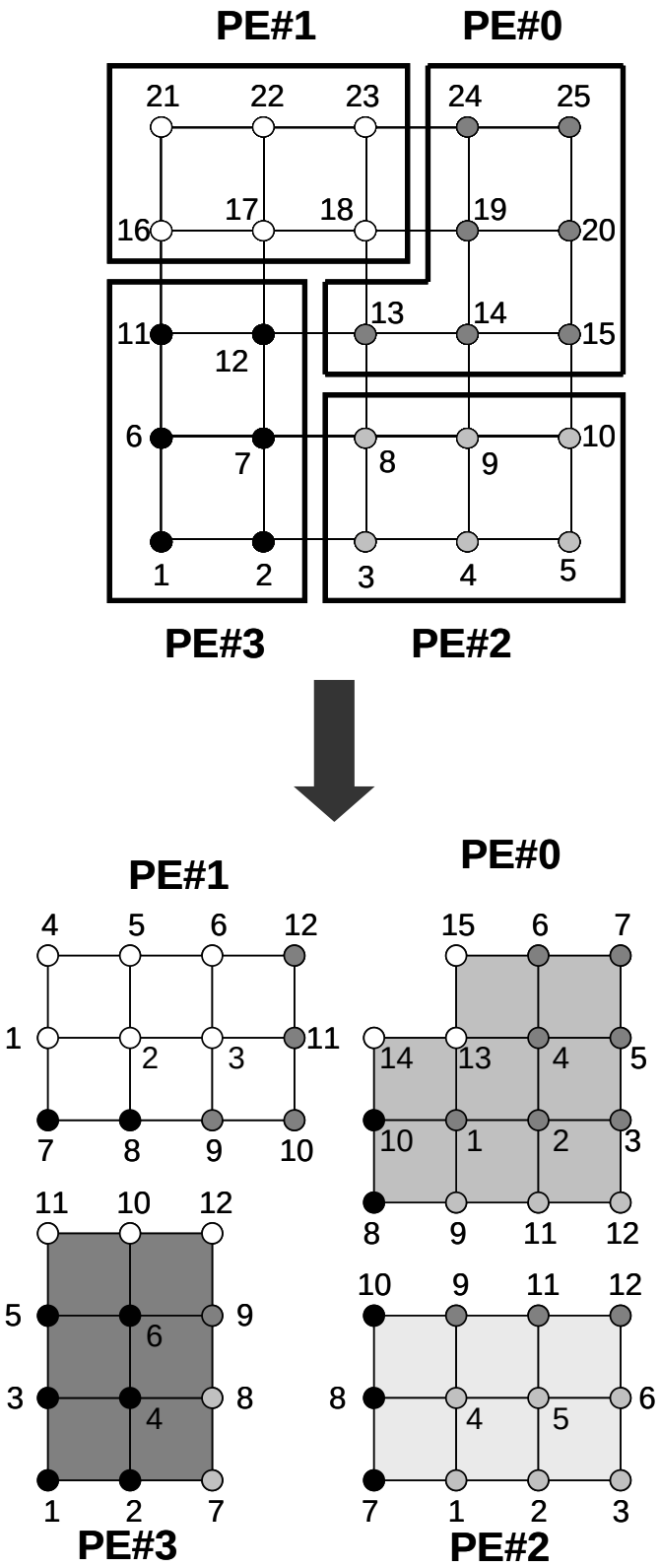


Fig. 4 Node-based partitioning into 4 PEs 〔5,6,7〕

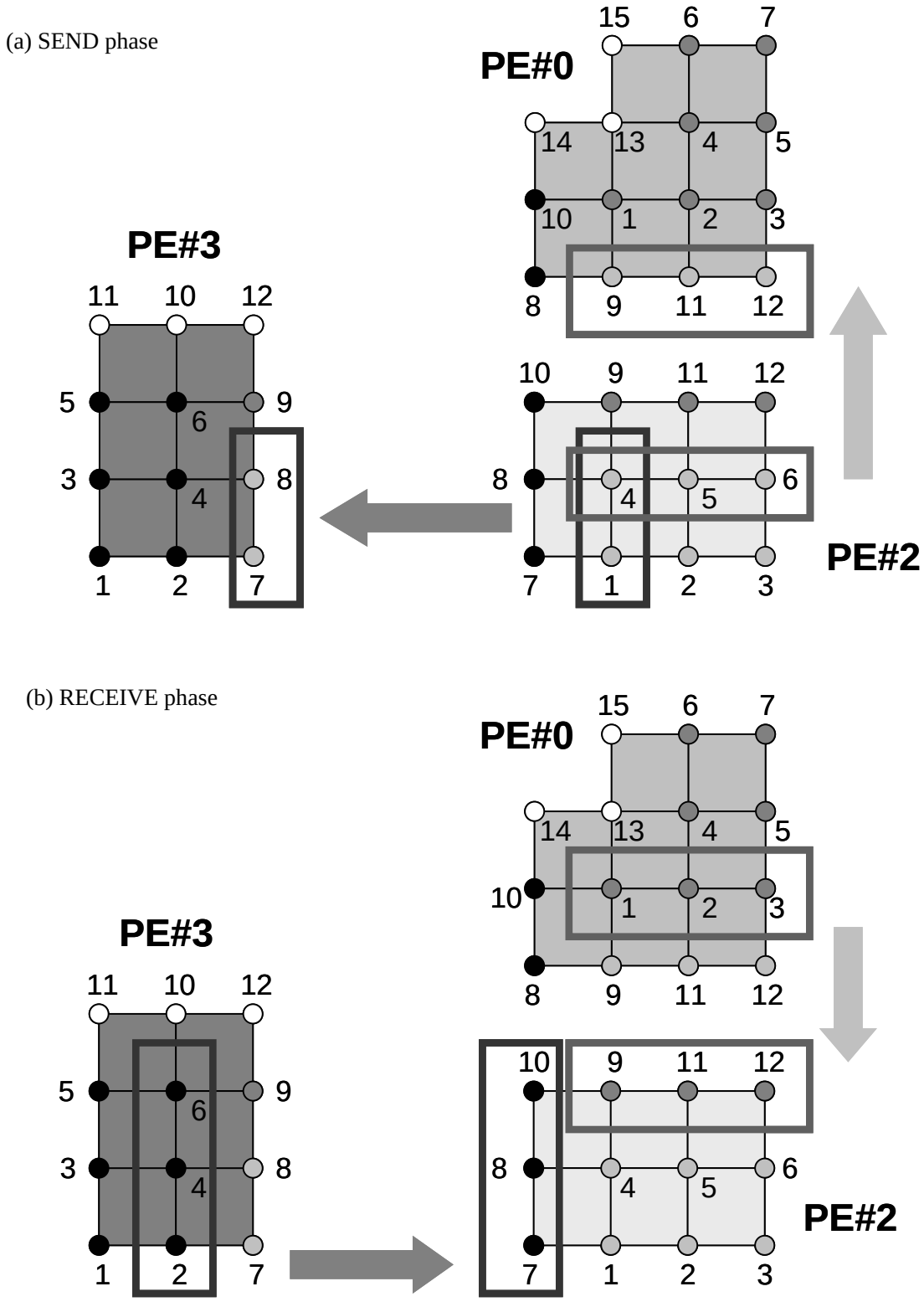


Fig. 5 Communication among processors 〔5,6,7〕

1.4 前処理付反復法

連立一次方程式の解法としてはガウスの消去法などの直接法（Direct Method）〔2〕が広く使用されてきたが，大域的（Global）処理が生ずるため，並列計算には適していない。並列計算に適した手法として共役勾配法（Conjugate Gradient Method，CG）〔2〕などのKrylov型反復法（Krylov Iterative Method）が注目されている。

反復法の収束特性は係数行列の固有値分布に依存するため，実用的な問題に適用するためには前処理（Preconditioning）を施し，固有値分布を変えた行列を解く手法が一般的である。反復法の前処理手法としては不完全LU分解（Incomplete LU Factorization，ILU）あるいは不完全コレスキー分解（Incomplete Cholesky Factorization, IC）などがよく使用される〔2〕。

前処理つき反復法における計算プロセスは以下の4種類に分類される：

（1） 行列ベクトル積
（2） ベクトル～ベクトル内積
（3） ベクトル（およびその実数倍）の加減（DAXPY）
（4） 前処理

Fig.7は前処理付CG法 〔1,2〕の処理手順を示したものである。これによると，一回の反復で以下に示す回数だけ各プロセスが発生する：

（1）	行列ベクトル積	1
（2）	ベクトル～ベクトル内積	2
（3）	ベクトル（およびその実数倍）の加減（DAXPY）	3
（4）	前処理	1

このうち（3）を除く各プロセスでは領域間の通信が発生する。（1）は計算前に2. で述べた通信を実施すれば局所的な処理が可能である。（2）は「MPI_ALLREDUCE」などのサブルーチンを使用して容易に達成可能である。Fig.8は（1）～（3）の各プロセスについてMPIを使用してFORTRANで記述した場合のプログラム例である。

（4）については前処理手法によって異なる。例えばILUなどの手法は前進／後退代入（Foward Backward Substitution，FBS）により大域的な変数の依存性が生じるため，並列化が困難である。単独プロセッサを使用した計算の場合，Fill-inのないILU（0）法を前処理として使用すると，前処理計算部分が全体の50%程度を占めるため〔7〕，前進／後退代入部分の並列化は計算効率の向上のために不可欠である。前進／後退代入は以下のような処理である。

- 前進代入 $$y_k = b_k - \sum_{j=1}^{k-1} L_{kj} y_j \quad (k=2,\cdots N)$$

- 後退代入 $$x_k = \tilde{D}_k \left(b_k - \sum_{j=k+1}^{N} U_{kj} y_j \right) \quad (k=N,N-1,\cdots 1)$$

HEC-MWではブロックヤコビ法に基づく，局所前処理法（局所ILU（0）法，Localized ILU（0））〔7〕を使用している。局所ILU（0）法は一種の「擬似」ILU（0）法である。局所ILU（0）法では前進／後退代入計算時に領域外からの影響（すなわち外点の影響）を0とすることによって，前処理の局所化を行い，並列性の高いアルゴリズムを実現している（Fig.9参照）。

この処理は，各プロセッサにおいて，境界＝0のディリクレ型境界条件のもとで前処理を実施することと同等であり，完全に並列な前処理を実施することが可能となる。この考え方は不完全ヤコビ前処理〔2,12〕に基づくものである。Fig.10は局所ILU（0）法を使用した場合のCG法の計算アルゴリズムである。一回の反復計算あたりで領域間通信が生じるのは3回であり，そのうち2つは内積計算のためにスカラーをBROADCASTするだけであり，残りの1回が通信テーブルを使用したオーバーラップ領域情報の通信である。

局所ILU（0）法は並列性能には優れていると考えられるが，グローバルな処理によっている本来のILU（0）法と比較すると前処理の機能は強力ではない。一般的に局所ILU（0）型の前処理では，領域数が増加するほど収束が悪くなる〔3,5,6,7〕。領域数と自由度数が同じになると，対角スケーリングと同じになってしまう。Table 1は3×44^3 自由度の一様な立方体領域における三次元線形弾性解析を局所IC(0)前処理つきのCG法で計算した例である。計算には東京大学情報基盤センターのHitachi SR2201を使用した。領域数の増加とともに収束までの反復回数は増加しているが，1PEから32PEまで増えた場合でも30%程度の増加である。

同様にHitachi SR2201を使用して，並列性能の評価を実施した。Fig.11は一様な立方体領域における三次元線形弾性解析を様々な問題規模において解いた場合の並列性能（work ratio＝計算時間／合計実行時間）である〔8,9〕。これらのケースでは1PEあたりの問題規模は固定されている。最大規模の問題では1024 PEを使用して196,608,000自由度の問題を解いている。Fig.11によると1PEあたりの問題規模が充分大きければ，並列性能は95％以上である。

文献〔10〕などに見られるように，適切なオーダリング，領域分割を実施することによって，並列計算の場合のILU／IC処理においても完全にグローバルな処理を達成することは可能である。しかしながらこれが可能となるのはあらかじめ全体マトリクスが得られているときのみである。並列有限要素法においては，局所領域ごとに係数マトリクスが生成されるため，このような手法は本研究においては採用していない。文献〔4〕においては深さ

kのFill-inレベルを持つILU（k）法のグローバル並列化に関して検討されている。非常に安定な収束を示しているが，並列性能は低い。

(a) Calling interface for communication among domains (1x1 scalar and 3x3 block)

1x1 Scalar

```
      allocate (WS(NP), WR(NP), X(NP)
      call SOLVER_SEND_RECV                                          &
     &   ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_NODE,         &
     &     EXPORT_INDEX, EXPORT_NODE, WS, WR, X , SOLVER_COMM,       &
     &     my_rank)
```

3x3 Block

```
      allocate (WS(3*NP), WR(3*NP), X(3*NP)
      call SOLVER_SEND_RECV_3                                        &
     &   ( NP, NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_NODE,         &
     &     EXPORT_INDEX, EXPORT_NODE, WS, WR, X , SOLVER_COMM,       &
     &     my_rank)
```

(b) Subroutines for communication among domains

- SEND phase

```
do neib= 1, NEIBPETOT
  istart= EXPORT_INDEX(neib-1)
  inum  = EXPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    WS(k)= X(EXPORT_NODE(k))
  enddo
  call MPI_ISEND
       (WS(istart+1), inum, MPI_DOUBLE_PRECISION, &
        NEIBPE(neib), 0, SOLVER_COMM,             &
        req1(neib), ierr)
enddo
```

- RECEIVE phase

```
do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  call MPI_IRECV
       (WR(istart+1), inum, MPI_DOUBLE_PRECISION, &
        NEIBPE(neib), 0, SOLVER_COMM,             &
        req2(neib), ierr)
enddo

call MPI_WAITALL (NEIBPETOT, req2, sta2, ierr)

do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    X(IMPORT_NODE(k))= WR(k)
  enddo
enddo

call MPI_WAITALL (NEIBPETOT, req1, sta1, ierr)
```

Fig. 6 Communication procedures among domains in HEC-MW 〔5,6,7,8,9〕

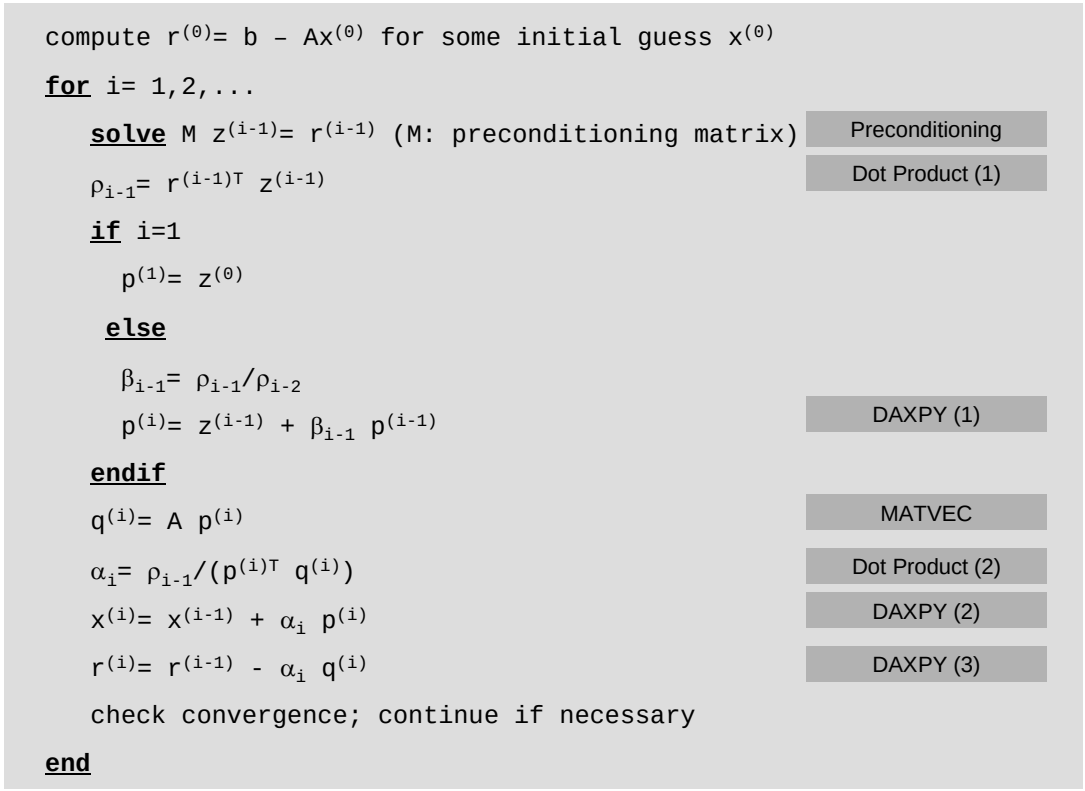


Fig. 7 Procedures in CG iterative method 〔1,2〕

(a) Matrix-vector products

```
 do i= 1, N
   isL= INL(i-1) + 1
   ieL= INL(i  )
   WVAL= WW(i,R)
   do j= isL, ieL
     inod = IAL(j)
     WVAL=  WVAL -  AL(j) * WW(inod,Z)
   enddo
   WW(i,Z)= WVAL * DD(i)
 enddo

 do i= N, 1, -1
   SW = 0.0d0
   isU= INU(i-1) + 1
   ieU= INU(i  )
   do j= isU, ieU
     inod = IAU(j)
     SW= SW + AU(j) * WW(inod,Z)
   enddo
   WW(i,Z)= WW(i,Z) - DD(i) * SW
 enddo
```

(b) Inner dot products

```
 RHO0= 0.0
 do i= 1, N
   RHO0= RHO0 + WW(i,R)*WW(i,Z)
 enddo

 call MPI_allREDUCE (RHO0, RHO, 1, MPI_DOUBLE_PRECISION, &
&                    MPI_SUM, SOLVER_COMM, ierr)
```

(c) DAXPY

```
 do i= 1, N
   X (i)  = X (i)   + ALPHA * WW(i,P)
   WW(i,R)= WW(i,R) - ALPHA * WW(i,Q)
 enddo
```

Fig. 8 Parallelization of typical processes in iterative solvers in FORTRAN with MPI 〔5,6,7,8,9〕

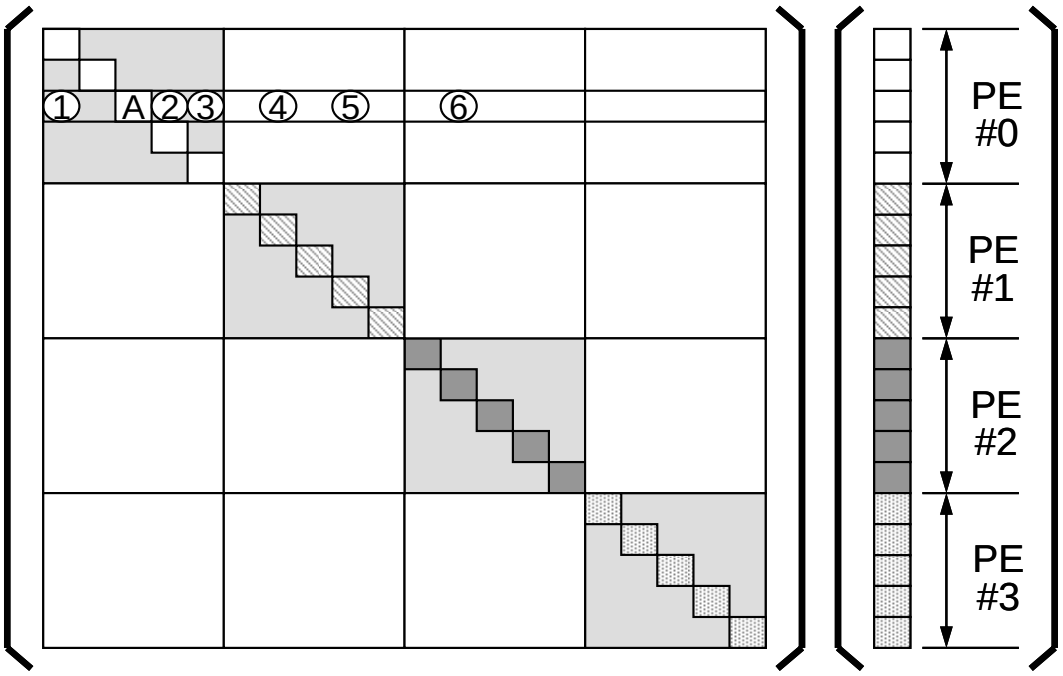


Fig. 9 *Localized* ILU(0) Operation: Matrix components whose column numbers are outside the processor are ignored (set equal to 0) at *localized* ILU(0) factorization. For example the element *A* on PE#0 has 6 non-zero components but only 1,2,3 are considered and 4,5,6 are ignored and set to 0 〔5,6,7,8,9〕

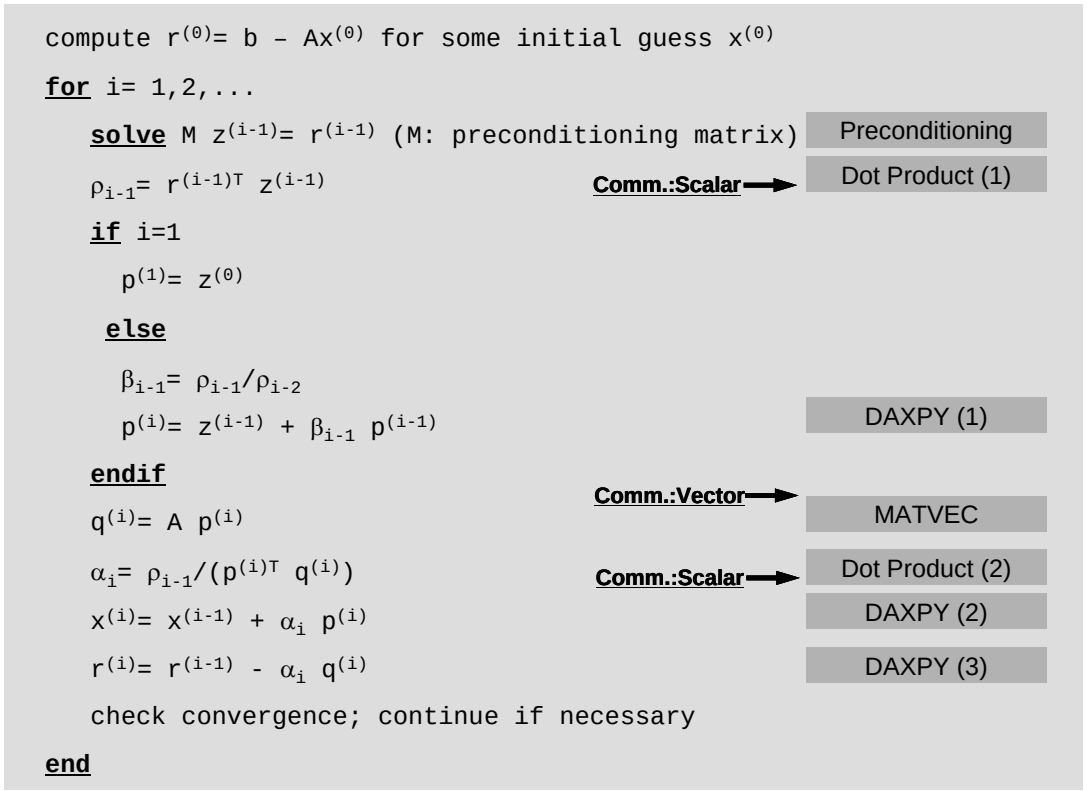


Fig. 10 Parallel CG iterative method by localized preconditioning in HEC-MW

Table 1 Homogeneous solid mechanics example with 3×44^3 DOF on Hitachi SR2201 solved by CG method with localized IC(0) preconditioning (Convergence Criteria $\varepsilon=10^{-8}$).

PE #	Iter. #	sec.	Speed Up
1	204	233.7	-
2	253	143.6	1.63
4	259	74.3	3.15
8	264	36.8	6.36
16	262	17.4	13.52
32	268	9.6	24.24
64	274	6.6	35.68

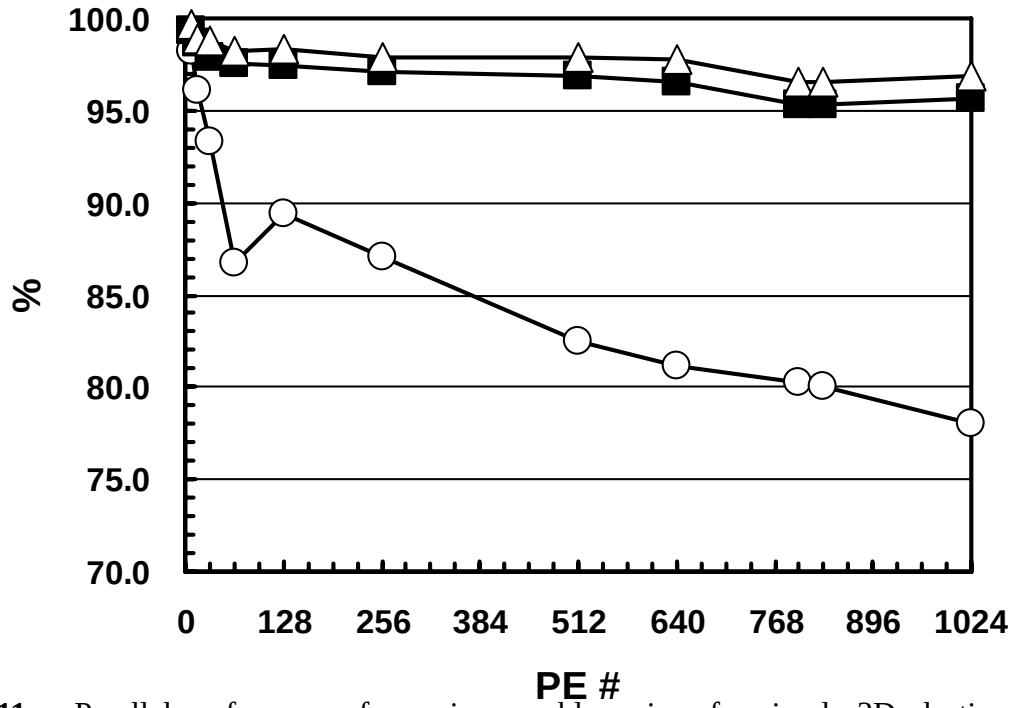
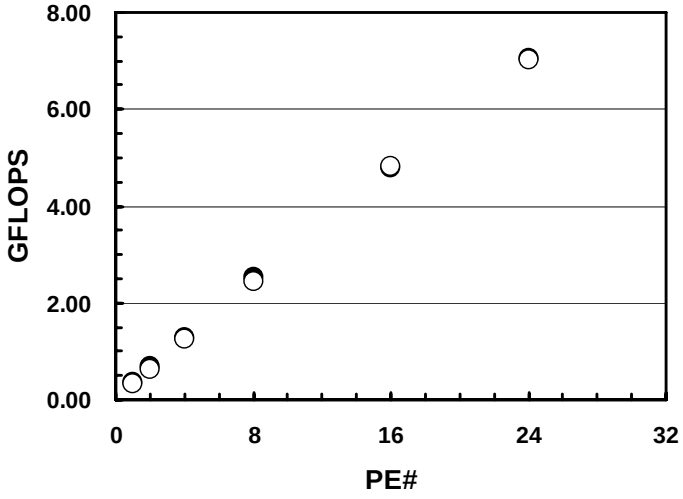
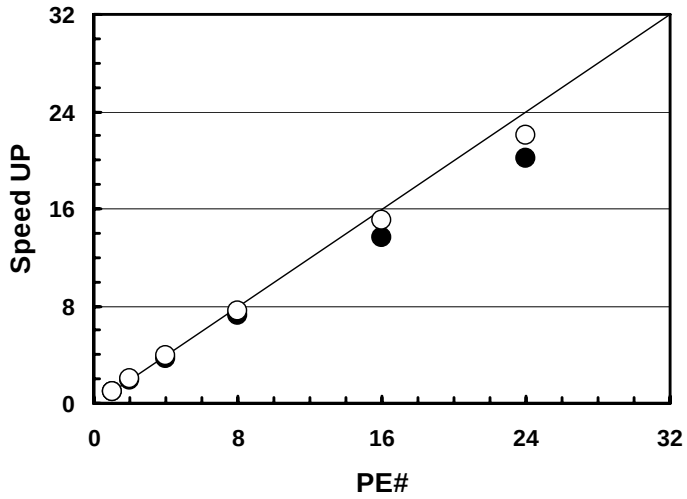


Fig. 11 Parallel performance for various problem sizes for simple 3D elastic solid mechanics on Hitachi SR2201. Problem size/PE is fixed. Largest case is 196,608,000 DOF on 1024 PEs. (Circles: 3×16^3 (= 12,288) DOF/PE, Squares: 3×32^3 (= 98,304), Triangles: 3×40^3 (= 192,000)).

(a) GFLOPS



(b) Parallel Speed-UP



(c) Parallel Work Ratio

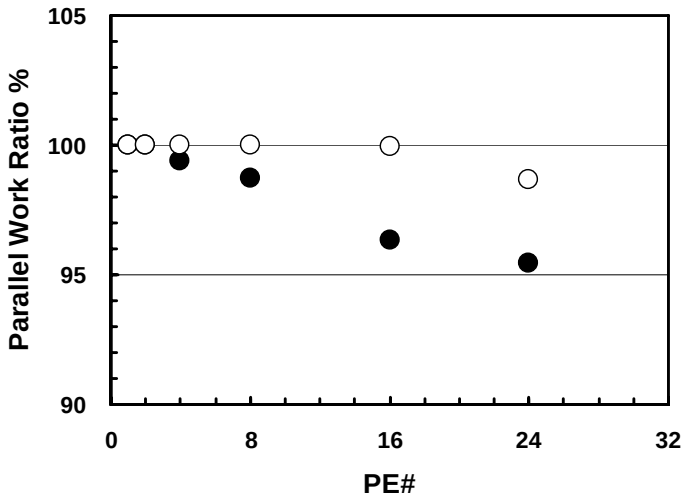
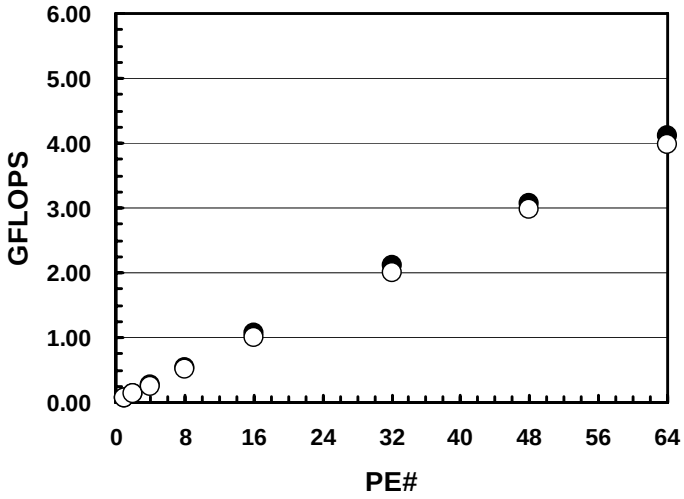
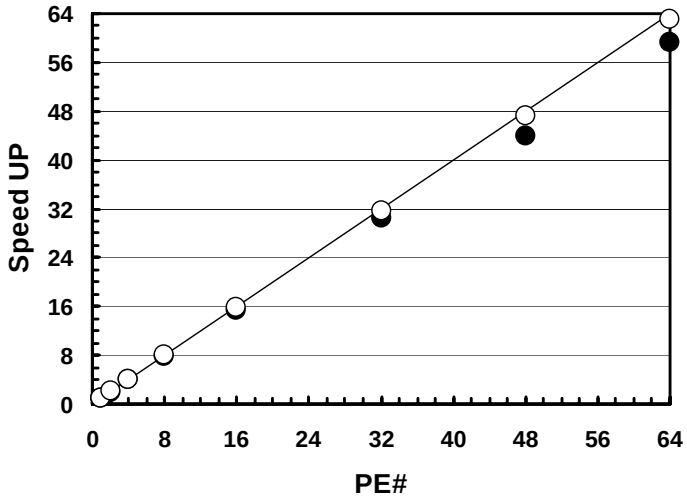


Fig. 12 Parallel performance for various problem sizes for simple 3D elastic solid mechanics on Xeon 2.8GHz Cluster. Problem size/PE is fixed. Largest case is 2,359,296 DOF on 24 PEs. (Black Circles: 3×16^3 (= 12,288) DOF/PE, White Circles: 3×32^3 (= 98,304)).

(a) GFLOPS



(b) Parallel Speed-UP



(c) Parallel Work Ratio

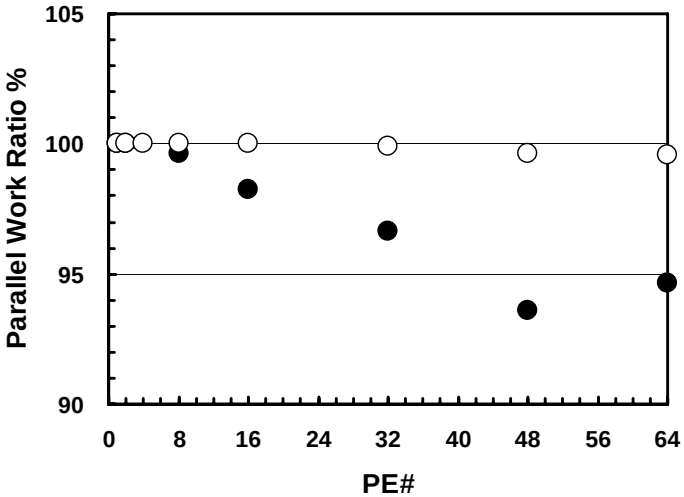


Fig. 13 Parallel performance for various problem sizes for simple 3D elastic solid mechanics on Hitachi SR2201. Problem size/PE is fixed. Largest case is 6,291,456 DOF on 64 PEs. (Black Circles: 3×16^3 (= 12,288) DOF/PE, White Circles: 3×32^3 (= 98,304)).

1.5 Additive Schwartz Domain Decomposition

局所ILU（0）前処理法を安定化させる手法として，領域間オーバーラップ領域に加法型Schwartz領域分割法（Additive Schwartz Domain Decomposition，ASDD）〔12〕を適用した。ASDDの手順は以下のとおりである：

(1) 以下の前処理を実施する $Mz = r$ ここで M：前処理行列， r,z：ベクトル，である。

(2) 全体領域がFig.14 (a) に示すように Ω_1 および Ω_2 の2領域に分かれているものと仮定すると前処理は各領域において，以下のように局所的に実施される。

$$z_{\Omega_1} = M_{\Omega_1}^{-1} r_{\Omega_1}, \quad z_{\Omega_2} = M_{\Omega_2}^{-1} r_{\Omega_2}$$

(3) 局所的な前処理を実行したのちに，領域間オーバーラップ領域 Γ_1 および Γ_2 において以下の計算を実施する（Fig.14（b））：

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1}(r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

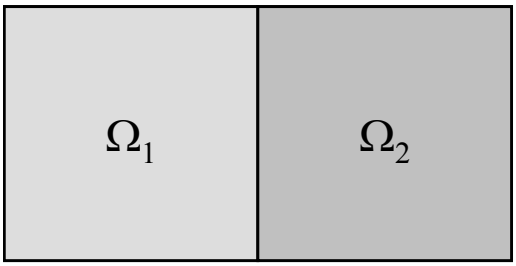
$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1}(r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$

ここでnはADSSの繰り返し数である。

(4) （2）および（3）のプロセスを収束まで繰り返す。

Table 2 は ASDD の効果を 1.4 で紹介した 3×44^3 自由度の一様な立方体領域における三次元線形弾性解析に適用した例である。計算には東京大学情報基盤センターの Hitachi SR2201 を使用した。すでに示したように，ASDD 無しの場合は領域数とともに反復回数は増加するが，ASDD の導入により，一回の反復計算あたりの計算量は増加するものの，領域数増加にともなう反復回数の増加はわずかに抑制される。

(a) Local operation



(b) Global nesting correction

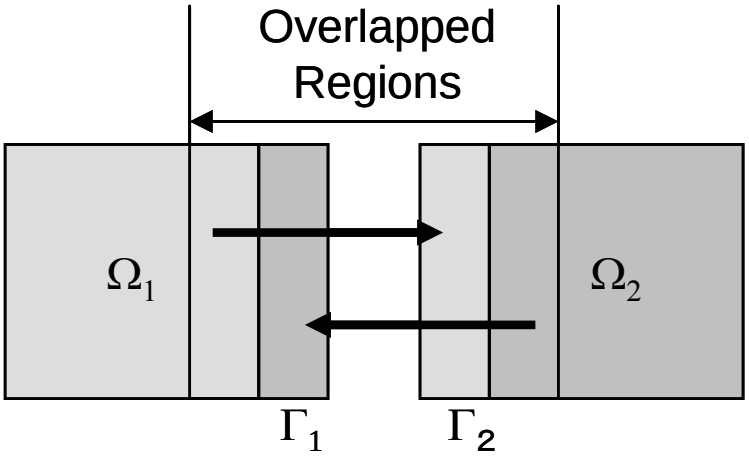


Fig. 14 Operations in ASDD for 2 domains 〔12〕

Table 2 Effect of ASDD for solid mechanics with 3×44^3 DOF on a Hitachi SR2201.

	NO Additive Schwarz			WITH Additive Schwarz		
PE #	Iter. #	Sec.	Speedup	Iter.#	Sec.	Speedup
1	204	233.	-	144	325.	-
2	253	143.	1.63	144	163.	1.99
4	259	74.3	3.15	145	82.4	3.95
8	264	36.8	6.36	146	39.7	8.21
16	262	17.4	13.52	144	18.7	17.33
32	268	9.6	24.24	147	10.2	31.80
64	274	6.6	35.68	150	6.5	50.07

Number of ASDD cycle/iteration = 1, Convergence Criteria $\varepsilon=10^{-8}$

1.6 悪条件問題に関する前処理手法

1.6.1 各前処理手法について

（1）概 要

接触条件に基づく境界非線形性を有するアプリケーションは，工学，自然科学の広い分野にわたって非常に重要である。有限要素法で接触問題を解く場合には，拡大ラグランジェ法（Augmented Lagrange Method，ALM）とペナルティ法が使用されることが多く，接触による拘束条件は大きなペナルティ数λを導入することによって表される〔9〕。非線形過程はNewton-Raphson法（NR法）によって線形化され，反復的に解かれている。

一般に，λの値が大きいほど，精度よく接触条件を模擬することができ，Newton-Raphson法の非線形計算に要する反復回数は減少するが，係数行列の条件数が大きくなる。したがって，反復法ソルバーの収束のためには多くの反復を必要とする。条件数の大きい悪条件問題を解く場合には安定した信頼性の高い前処理手法が不可欠である。

ILU/IC系の前処理手法を使用する場合, このような悪条件問題に関する対処法としては以下のようなものがあげられる：

- ブロッキング
- 深い Fill-in レベル
- オーダリング

（2）ブロッキング

三次元固体力学の問題においては，各節点あたり，三方向の変位成分を自由度として持っている。そこで，Fig.15に示すように3×3ブロックに関する完全LU分解を，対角ブロックに導入する。これが，ILU/IC系前処理のブロック化バージョンである，ブロックILU/IC（BILU/BIC）前処理である。このような処理を導入することによって，各節点上の三自由度は同時に計算されるため，各自由度を独立に計算する場合と比較してより効果的な前処理が可能となる。

（3）深い Fill-in レベル

続いて，BILU/BIC前処理において深いレベルのFill-inを考慮するBILU（n）/BIC（n）前処理について検討する（「n」はFill-inレベル）。直接法で使用される完全LU分解（あるいはガウスの消去法）のアルゴリズムは以下の通りである：

Gaussian Elimination

```
do i= 2, n
  do k= 1,  i-1
    a_jk := a_jk/a_kk
    do j= k+1, n
      a_ij := a_ij - a_ik*a_kj
    enddo
  enddo
enddo
```

完全LU分解では, 分解のたびに多量のFill-inが発生するため, もとの行列が疎でも完全LU分解によって生成される逆行列は密行列となる可能性がある。ILU（n）またはIC（n）は「n」レベル分のFill-inを許容する手法である。「n」の値が大きいほど，分解の精度は高くなり，より安定した前処理行列が得られるが，計算量，メモリのコストは高くなる。一般工学分野においては，Fill-inを許さず，オリジナル行列と同じ非ゼロ成分パターンを保持する以下のに示すILU（0）/IC（0）前処理が，効率の観点から広く使用されている。

ILU(0)

```
do i= 2, n
  do k= 1,  i-1
    if ((i,k) ∈ NonZero(A)) then
      a_jk := a_jk/a_kk
    endif
    do j= k+1, n
      if ((i,j) ∈ NonZero(A)) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

一方「n」レベルのFill-inを考慮するILU（n）/IC（n）法のアルゴリズムは以下の通りである：

ILU(n)

```
LEV_ij=0 if ((i,j) ∈ NonZero(A)) otherwise LEV_ij= p+1
do i= 2, n
  do k= 1,  i-1
    if (LEV_ik ≤ p) then
      a_jk := a_jk/a_kk
    endif
    do j= k+1, n
      if (LEV_ij = min(LEV_ij,1+LEV_ik+ LEV_kj) ≤ p) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

（4）Selective Blocking

Fill-inレベルを深くとる手法に加えて，接触問題向けに「Selective Blocking」法を開発した〔9〕。「Selective Blocking」法は，接触要素によって決定され，ペナルティ数により強くカップルした「接触節点グループ」〔9〕に属する節点が，連続する節点番号を持って同じグループ（「Selective Block」または「Super Node」）になるようにオーダリングし，この「Selective Block」内に完全LU分解を適用する手法である。三次元問題の場合，ある「Selective Block」に含まれる節点数をNBとすると，（3×NB）×（3×NB）ブロックに関する完全LU分解が必要となる（Fig.16およびFig.17参照）。つまり，前処理計算のプロセスにおいて，接触節点グループに属する節点群に関しては直接法的な処理が適用される。

この「Selective Block」の考え方は，「Clustered Element-by-Element（CEBE）」法あるいはブロックICT法と類似したものである。これらの手法は，全体領域を複数の節点のクラスター（Cluster）に分割し，各クラスター内に直接法的な処理を導入するものである。クラスターの大きさは任意に選ぶことが可能であるが，最適な計算効率を得るためのパラメータとして考えることができる。CEBE法はクラスターの大きさが節点数と同じになった場合に直接法と同等と見なすことができる。一般的にクラスターの大きさが大きいほど安定した収束が得られるが，Fig.18に示すように各反復における計算コストは増大する。収束と計算コストのトレードオフは必ずしも明確ではないが，クラスターの大きさが大きいほど概して効率的である。

「Selective Blocking」では，CEBE法などにおけるクラスターは接触節点グループの情報によって決定される。個々のクラスターの大きさはCEBE法と比較すると一般的に小さい。例えば，接触節点グループに属さない節点の場合，その節点単独で「大きさ＝1」のクラスターを形成することになる。「Selective Blocking」法におけるサイクルあたりの計算コスト

は，クラスター間のFill-inを考慮しない場合は，BILU（0）/BIC（0）とほぼ同等である。

（5）各手法の評価

Table 3は，Fig.20に示す形状の問題について，ブロック間の接触面の各節点についてMPC（Multi-Point Constraint）条件を課した，弾性静解析についての計算結果である。3×3ブロック処理を導入することにより，Fill-inを考慮しないIC法前処理（BIC（0））においてもλ=10^6の場合に計算を実施することが可能となった。Fill-inを深くすることにより，更に効率的に解を得ることが可能であるが，SB-BIC（0）前処理（BIC（0）前処理と「Selective Blocking」オーダリングを組み合わせた手法）によるものが最も効率が良いことがわかった（Table 3）。

SB-BIC（0）は収束までの反復回数はBIC（1）やBIC（2）（Fill-inレベル＝1または2のブロックIC前処理）と比較して多いが，一回の反復あたりの計算コストが低いため，計算時間が少なくなる。SB-BIC（0）では，「Selective Block」間のFill-inは考慮されておらず，「Selective Block」内のFill-inのみ考慮されているため，計算コスト，必要メモリ量は既に述べたようにBILU（0）/BIC（0）とほぼ同等である。

「Selective Blocking」を考慮した前処理によるKrylov反復解法は反復法と直接法のハイブリッド解法と見なすことができる。すなわち，接触節点グループに属する節点に関しては，前処理においては直接法的な処理が適用されている。この手法は反復法の効率，スケーラビリティと直接法の安定性の両方を備えた解法と考えることができる。

1.6.2 並列計算手法

局所ILU/IC前処理は効率的な並列前処理手法であるが, 悪条件問題に関しては安定な手法ではない。Table 4はTable 3で示した問題によるマトリクスに関して，8 PEのPCクラスタを使用して，局所ICCG法によって計算を実施した例である。なお，領域分割はMETISの中のk-way METISを使用している。計算結果によるとペナルティ数の増加とともに収束は悪化し，$\lambda=10^6$の場合は収束までの反復回数が10倍のオーダーで増加している。これは同じ接触節点グループに属する節点から構成される要素（すなわち「接触要素」）が，領域間境界に存在しているため，これらの辺で「edge-cut」が生じているためであると考えられる〔9〕。

このような「edge-cut」を解消するため，同じ接触節点グループに属する節点が必ず同じ領域に属するような特殊な領域分割手法を開発した。さらに，領域間の負荷が均等となるような，負荷再配分手法も併せて開発した（Fig.19）。

Table 4 は，この新しい領域分割法を適用して得られる計算結果である。反復回数は，いずれの前処理手法においても従来と比較して著しく減少していることがわかる。

1.6.3 ベンチマーク

（1）概 要

接触問題用に開発された前処理手法および領域分割手法の効率と安定性について，二種類の三次元計算例を使用して検証を実施した。

Fig.20は，第一の計算対象である，3つの単純形状ブロックから構成される対象（簡易ブロックモデル）と，三次元弾性静解析のための境界条件について説明したものである。この計算例では，以下に示すように，線形多点拘束（Multiple Point Constraint，MPC）の条件が接触節点グループ内の各節点に適用されている。

- 接触節点グループを構成する各節点の座標は同一である。
- 接触節点グループを構成する各節点の変位は三方向において拘束されている。
- 固体力学における微小弾性変形理論に基づき，節点の座標は不変であり，したがって接触に関する関係も不変である。
- ペナルティ数による拘束が接触節点グループを構成する各節点に適用される。「111」型トラス要素が Fig.21 に示すように各接触節点グループの節点によって形成される。このトラス要素の剛性はペナルティ数に対応する。Fig.22 は接触節点グループの行列処理を示す。前述のように（x,y,z）三方向の変位が拘束されている。

ペナルティ数が大きくなるほど，強い拘束条件となるが係数行列の条件数は大きくなる。したがって，ペナルティ数が大きくなると，反復法の収束は悪化する。このベンチマークで扱う問題自体は線形の弾性解析であるが，得られる連立一次方程式は非線形接触問題を解く場合と同様に，反復法で解くことは困難である。解析モデルと境界条件は以下の通りである：

- ヤング率＝1.00, ポアソン比＝0.30 の一様物性からなる 3 つのブロックを含んでいる。一次線形アイソパラメトリック六面体要素を使用している。
- 各ブロックの境界に沿って一様な MPC 条件が適用されている。したがって, Fig.22 に示すように各接触節点グループに属する節点数は異なる（2 または 3）。
- x=0 および y=0 の面においては対称境界条件が適用されている。
- x=Xmax および y=Ymax の面においては拘束条件は適用されていない。
- z=0 の面においては固定境界条件が適用されている。
- z=Zmax の面においては z 方向に一様分布荷重がかけられている。
- 接触面における摩擦を考慮しない場合には，係数行列は対称正定となる。したがって共益勾配法（CG 法）の適用が可能である。

この計算例における要素の形状は全て立法体である。

Fig.23に示す二番目の解析モデルは，西南日本域における地震シミュレーション〔9〕に使用される複雑な形状を持ったメッシュである。このモデルは大陸側プレート地殻（濃い灰色）と沈み込みプレート（淡い灰色）の2つの部分から構成されている。節点数は27,195であり，一次線形アイソパラメトリック六面体要素 23,831個から構成されている。簡易ブロックモデルと同様の境界条件を適用しているが，この西南日本モデルでは，z=Zmaxの面におけるz方向への表面分布荷重ではなく，各要素にz方向に大きさ-1.0の体積分布荷重が適用されている。また，xおよびy方向への対称境界条件も適用されていない。西南日本モデルでは，各要素の形状は不規則で，場所によっては非常に大きく歪んでいるものもある。物性は簡易ブロックモデルと同様一様で，ヤング率＝1.00，ポアソン比＝0.30である。

以下，これら2種類の計算モデルについて，1 PE（Compaq Alpha 21164/600MHz）によって小規模問題を解いた場合，日立SR2201においてFlat-MPI並列プログラミングモデルを使用して並列化した場合の大規模計算例を示す。

（2）ベンチマークI（簡易ブロックモデル）

まず最初にFig.20に示す簡易ブロックモデルについて，ペナルティ数をパラメータとして，様々な前処理手法を適用した。計算は1領域について実施し，Compaq Alpha 21164/600MHzを使用した。このベンチマークでは，以下のようなモデルを使用している：

- NX1＝20，NX2=20，NY=15，NZ1=20，NZ2=20（Fig.20（a））
- 要素数＝24,000，節点数＝27,888，自由度数＝83,664

Table 5（a）は反復法の収束状況を示したものである。ペナルティ数が10^4より大きくなるとBIC（0）は収束しなくなる。BIC（1），BIC（2）とSB-BIC（0）は広範囲のペナルティ数に対して安定である。SB-BIC（0）はBIC（1），BIC（2）と比較して収束までの反復回数は多いが計算時間は短く，最も効率的である。

続いて，前処理手法の安定性を$[M]^{-1}[A]$の固有値分布に基づき文献〔1, 3〕に示された手法によって推測した。ここで[A]はもとの係数行列であり，$[M]^{-1}$は前処理行列の逆行列である。

対称正定行列において，条件数κは

$$\kappa = E_{max} / E_{min}$$

によって得られる。ここでE_{max}，E_{min}は$[M]^{-1}[A]$の最大および最小固有値である。

Table 5（b）は各前処理手法において，様々なペナルティ数に関して得られるE_{max}，E_{min}および κ の値である。Table 5によると，BIC（0）以外の前処理手法では，広範囲のペナル

ティ数において全ての固有値はほぼ同じ値であり，1.00に近い。BIC（1）とBIC（2）からはSB-BIC（0）と比較して若干小さい κ が得られる。

（3）ベンチマークII（西南日本モデル）

Fig.23に示すように複雑な形状をもった西南日本モデルについて，ペナルティ数をパラメータとして，様々な前処理手法を適用した。計算は1領域について実施し，Compaq Alpha 21164/600MHzを使用した。このベンチマークでは，以下のようなモデルを使用している：

Table 6は反復法の収束状況を示したものである。ペナルティ数が10^4より大きくなるとBIC（0）は収束しなくなる。BIC（1），BIC（2）とSB-BIC（0）は広範囲のペナルティ数に対して安定であるが，BIC（1）とBIC（2）はペナルティ数が10^2から10^4に増加すると反復回数が増加する（BIC（1）は201から259に，BIC（2）は176から232）。SB-BIC（0）はBIC（1），BIC（2）と比較して収束までの反復回数は多いが計算時間は短く，最も効率的である。

Table 7 は各前処理手法において，様々なペナルティ数に関して，$[M]^{-1}[A]$の固有値分布から得られるE_{max}，E_{min}および κ の値である。SB-BIC（0）についてはペナルティ数が変化しても，E_{max}，E_{min}および κ の値は変化しないが，BIC（1）とBIC（2）については，ペナルティ数が10^2から10^4に増加すると κ の値が増加する。これはTable 6に示したBIC（1）とBIC（2）の反復回数が増加したのに対応している。この計算例では，形状が簡易ブロックモデルと比較すると形状は複雑であり，メッシュ形状も不規則で歪んでいるものもある。要素の歪みは，係数行列[A]，$[M]^{-1}[A]$の固有値分布に直接に影響する。SB-BIC（0）はこのような条件下においても安定している。

簡易ブロックおよび西南日本のいずれのモデルにおいても，$[M]^{-1}[A]$の条件数は前処理手法の収束性の評価に役立つパラメータである。簡易ブロックモデルにおいては，BIC（1）およびBIC（2）に関する条件数はSB-BIC（0）のそれと比較して小さく，収束までの反復回数は少ない（Table 4およびTable 5）。一方，西南日本モデルにおいては，ペナルティ数が10^4よりも大きくなると，BIC（1）およびBIC（2）に関する条件数はSB-BIC（0）のそれよりも大きくなるが，収束までの反復回数は少ない（Table 6およびTable 7）。

1.6.4 大規模並列計算

（1）簡易ブロックモデル

Fig.20に示す簡易ブロックモデルについて，大規模計算を実施した。以下のようなモデルを使用している：

- NX1＝70，NX2=70，NY=40，NZ1=70，NZ2=70（Fig.8（a））
- 要素数＝784,000，節点数＝823,813，自由度数＝2,471,439

各種前処理手法について，様々なペナルティ数に関して並列計算を実施した。並列計算にあたっては，1.6.3で述べた接触問題用領域分割手法を使用した。計算にあたっては日立SR2201を16～128 PE 使用した。並列プログラミングモデルとしてはFlat-MPIを使用した。

Table 8は128 PEを使用した場合の各種前処理における計算結果である。ペナルティ数が10^4より大きくなるとBIC（0）は収束しなくなる。BIC（1），BIC（2）とSB-BIC（0）は広範囲のペナルティ数に対して安定である。SB-BIC（0）はBIC（1），BIC（2）と比較して収束までの反復回数は多いが計算時間は短く，最も効率的である。Table 9と Fig.12は同じ問題をPE数を16から128まで変化させて解いた場合の結果である。BIC（1）はPE数が64より小さいとメモリ不足で動かず，BIC（2）に至っては128 PEの場合のみ計算が可能であった。Table 9と Fig.24からわかるように，BIC（0）とSB-BIC（0）において，PE数を16から128に変化させると，局所前処理の影響で反復回数は増大するが，16 PEから128 PEでの増加は11%程度である。16 PEの場合を基準とすると，128PEにおける加速率は120以上である。

Fig.25は各前処理手法における必要メモリサイズに関して比較したものである。日立SR2201の各プロセッサのメモリは256MBであるが，このうちアプリケーションに利用可能なのは224MBである。例えば，BIC（2）は14.4 GBのメモリを必要とするが，64 PEでは利用可能メモリの合計は，224MB×64／1000＝14.34GBであるため，64 PEでは動かないということになる。SB-BIC（0）の必要メモリ容量はBIC（0）とほぼ同じであり，BIC（1）の50%以下，BIC（2）の25%程度である。SB-BIC（0）の必要メモリ容量は接触要素の数や，各「Selective Block」のサイズによって変化しうるが，いずれにせよ，BIC（1）やBIC（2）と比較すると少ない。

（2）西南日本モデル

西南日本モデルについて，SB-BIC（0）前処理を使用して，大規模並列計算を実施した。計算にあたっては日立SR2201を16～128 PE 使用した。Fig.23に示すメッシュをグローバルに2回細分化し得られた997,422節点，960,509要素の解析モデルを使用した。総自由度数は2,992,264である。

Table 10と Fig.26は同じ問題をPE数を16から128まで変化させて解いた場合の結果である。

BIC（0）とSB-BIC（0）において，PE数を16から128に変化させると，局所前処理の影響で反復回数は増大するが，16 PEから128 PEでの増加は15%程度である。16 PEの場合を基準とすると，128PEにおける加速率は107である。

Table 3 Iterations/computation time for convergence ($\varepsilon=10^{-8}$) on a single PE of Intel Xeon 2.8 GHz by preconditioned CG for the 3D elastic fault-zone contact problem in Fig.20 (83,664 DOF).: **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering.

Preconditioning	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up+Solve (sec.)	Single Iteration (sec.)	Memory Size (MB)
Diagonal	10^2	1531	<0.01	75.1	75.1	0.049	119
Scaling	10^6	No Conv.	-	-	-	-	
IC(0)	10^2	401	0.02	39.2	39.2	0.098	119
(Scalar Type)	10^6	No Conv.	-	-	-	-	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

$Mp=q$ where $M=(L+D)D^{-1}(D+U)$

Forward Substitution

$(L+D)p= q : p= D^{-1}(q-Lp)$

Backward Substitution

$(I+ D^{-1} U)p_{new}= p_{old} : p= p - D^{-1} Up$

```
do i= 1, N
  SW1= Z(3*i-2)
  SW2= Z(3*i-1)
  SW3= Z(3*i  )
  isL= INL(i-1)+1
  ieL= INL(i)
  do j= isL, ieL
      k= IAL(j)
     X1= Z(3*k-2,Z)
     X2= Z3*k-1,Z)
     X3= WW(3*k  ,Z)
    SW1= SW1 - AL(1,1,j)*X1 - AL(1,2,j)*X2 - AL(1,3,j)*X3
    SW2= SW2 - AL(2,1,j)*X1 - AL(2,2,j)*X2 - AL(2,3,j)*X3
    SW3= SW3 - AL(3,1,j)*X1 - AL(3,2,j)*X2 - AL(3,3,j)*X3
  enddo
```

Full LU factorization for 3x3 block

```
  X1= SW1
  X2= SW2
  X3= SW3
  X2= X2 - ALU(2,1,i)*X1
  X3= X3 - ALU(3,1,i)*X1 - ALU(3,2,i)*X2
  X3= ALU(3,3,i)*  X3
  X2= ALU(2,2,i)*( X2 - ALU(2,3,i)*X3 )
  X1= ALU(1,1,i)*( X1 - ALU(1,3,i)*X3 - ALU(1,2,i)*X2)
  WW(3*i-2,Z)= X1
  WW(3*i-1,Z)= X2
  WW(3*i  ,Z)= X3
enddo
```

```
WW (:,Z) : work vector
INL(:)   : coefficient index of the lower triangular matrix (LTM)
IAL(:)   : connected nodes of LTM

AL (3,3,:): 3x3 coefficient matrix component of the LTM
ALU(3,3,:): 3x3 LU factorization for each 'node'
```

Fig. 15 Procedure of the 3×3 block ILU(0) preconditioning: forward substitution [9]

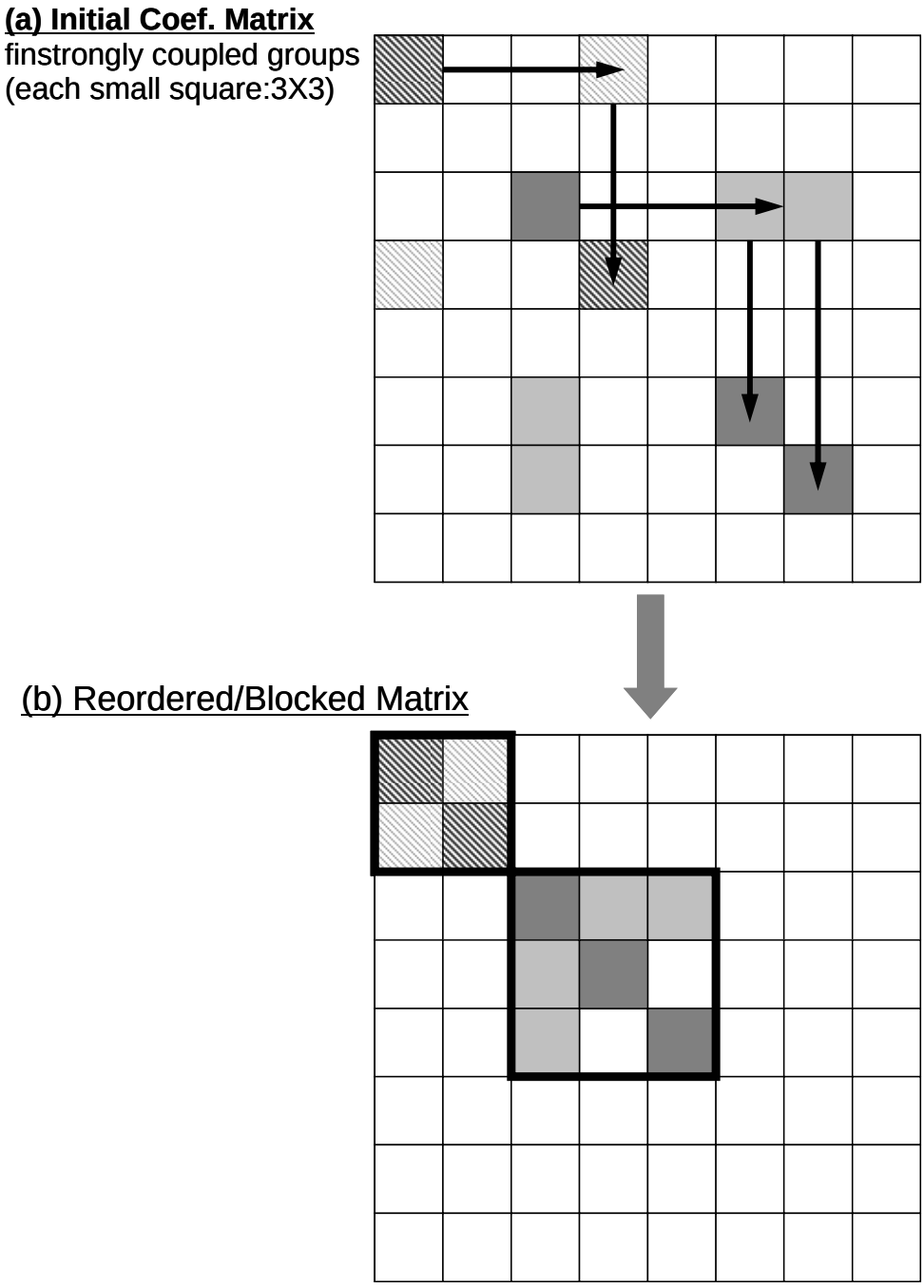


Fig. 16 Procedure of the *selective blocking* : Strongly coupled elements are put into the same *selective block*. (a) searching for strongly coupled components and (b) reordering and selective blocking. Full LU factorization procedure is applied to each *selective block*. Coupled finite-element nodes in contact groups can be solved in direct method during preconditioning procedure. In SB-BIC(0) (BIC(0) preconditioning combined with the *selective blocking* reordering), no *inter-block* fill-in is considered. Only *inter-node* fill-in in each *selective block* is considered in SB-BIC(0) [9].

```
      do ib= 1, NBLOCKtot
        NB0size= BLOCKstack(ib) - BLOCKstack(ib-1)
        (FORWARD SUBSTITUTIONS)
        do i= 1, NB0size
          ii= i + iBS
          WVAL1= 0.d0
          WVAL2= 0.d0
          WVAL3= 0.d0
          do j= 1, NB0size
            WR1= WKB(3*j-2)
            WR2= WKB(3*j-1)
            WR3= WKB(3*j  )
            WVAL1= WVAL1 + ALU(3*i-2,3*j-2,ib) * WR1                    &
     &                   + ALU(3*i-2,3*j-1,ib) * WR2                    &
     &                   + ALU(3*i-2,3*j  ,ib) * WR3
            WVAL2= WVAL2 + ALU(3*i-1,3*j-2,ib) * WR1                    &
     &                   + ALU(3*i-1,3*j-1,ib) * WR2                    &
     &                   + ALU(3*i-1,3*j  ,ib) * WR3
            WVAL3= WVAL3 + ALU(3*i  ,3*j-2,ib) * WR1                    &
     &                   + ALU(3*i  ,3*j-1,ib) * WR2                    &
     &                   + ALU(3*i  ,3*j  ,ib) * WR3
          enddo
          WW(3*ii-2,Z)= WVAL1
          WW(3*ii-1,Z)= WVAL2
          WW(3*ii  ,Z)= WVAL3
        enddo
      enddo
```

Fig. 17 Procedure of the *selective blocking*: Full LU factorization procedure for a (3×NB) × (3×NB) size *selective block.* [9]

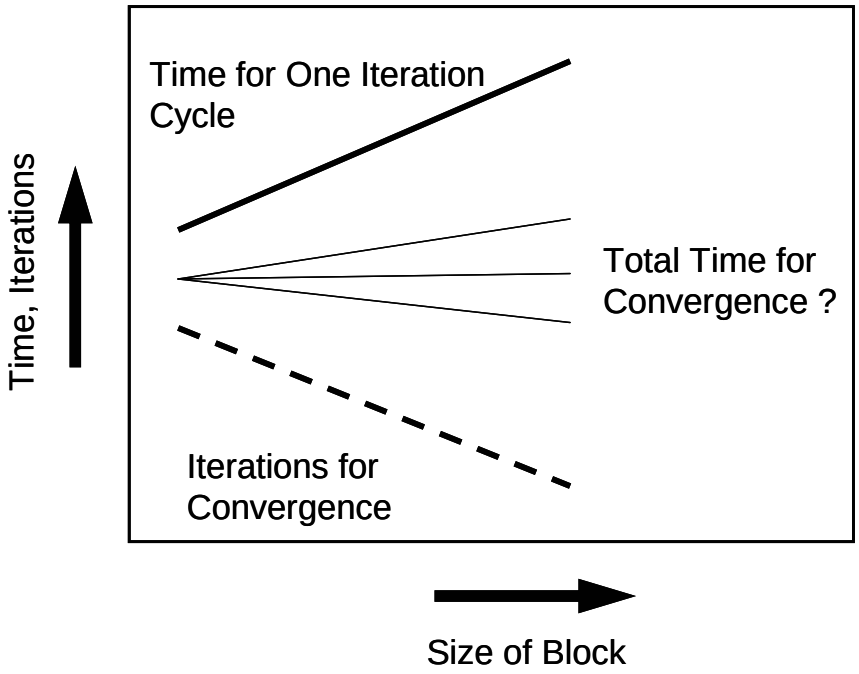


Fig. 18 Trade-off between convergence and computational cost per one iteration cycle according to block size in CEBE type method.

Table 4 Iterations/computation time for convergence ($\varepsilon=10^{-8}$) on 8 PEs of Intel Xeon 2.8 GHz cluster by preconditioned CG for the 3D elastic fault-zone contact problem (83,664 DOF).: **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering. Effect of repartitioning method in Fig.20 is evaluated.

		ORIGINAL Partitioning		**IMPROVED** Partitioning	
Preconditioning	λ	Iterations	Set-up+Solve (sec.)	Iterations	Set-up+Solve (sec.)
BIC(0)	10^2	703	7.5	489	5.3
	10^6	4825	50.6	3477	37.5
BIC(1)	10^2	613	11.3	123	2.7
	10^6	2701	47.7	123	2.7
BIC(2)	10^2	610	19.5	112	4.7
	10^6	2448	73.9	112	4.7
SB-BIC(0)	10^0	655	10.9	165	2.9
	10^6	3498	58.2	166	2.9

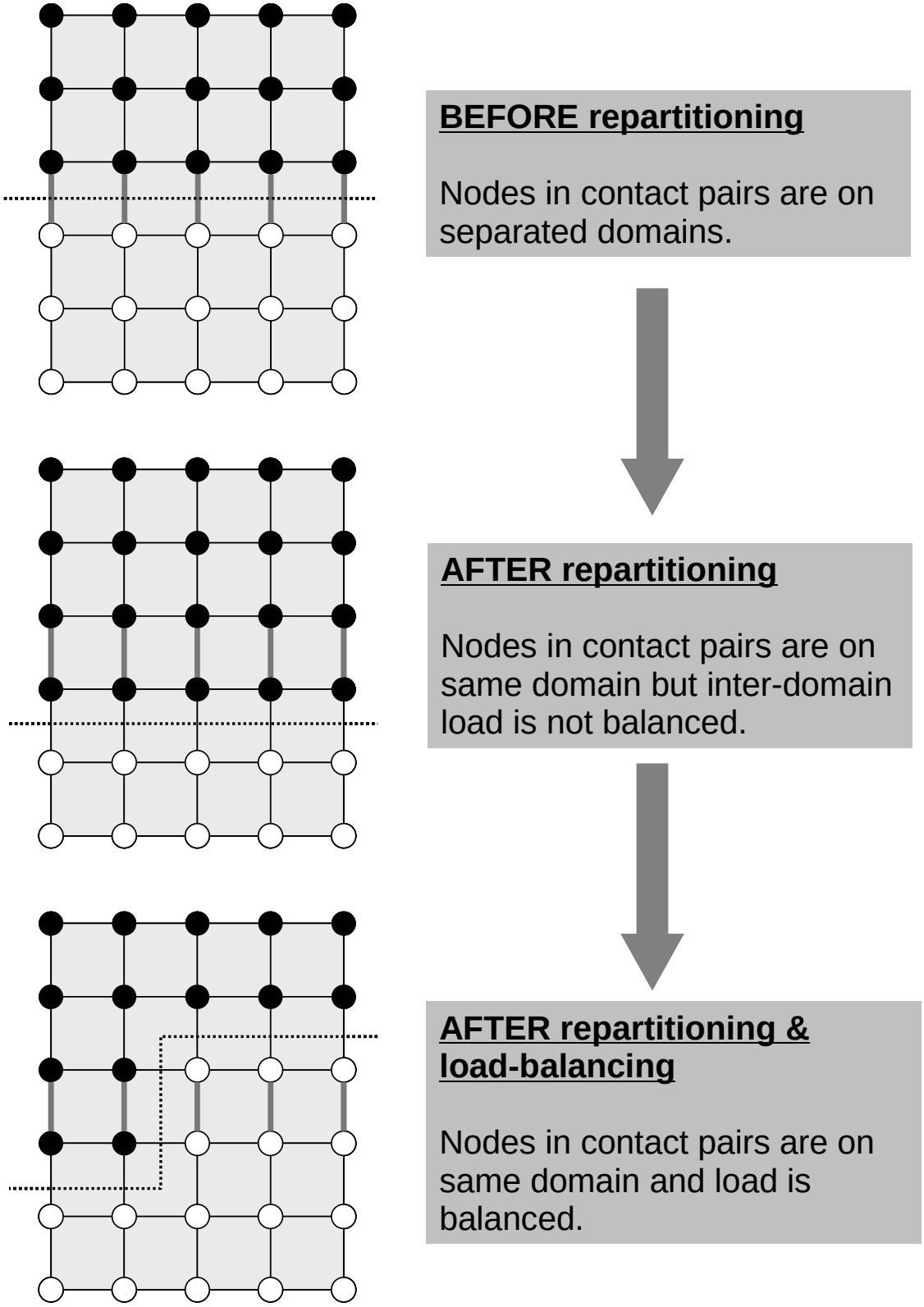
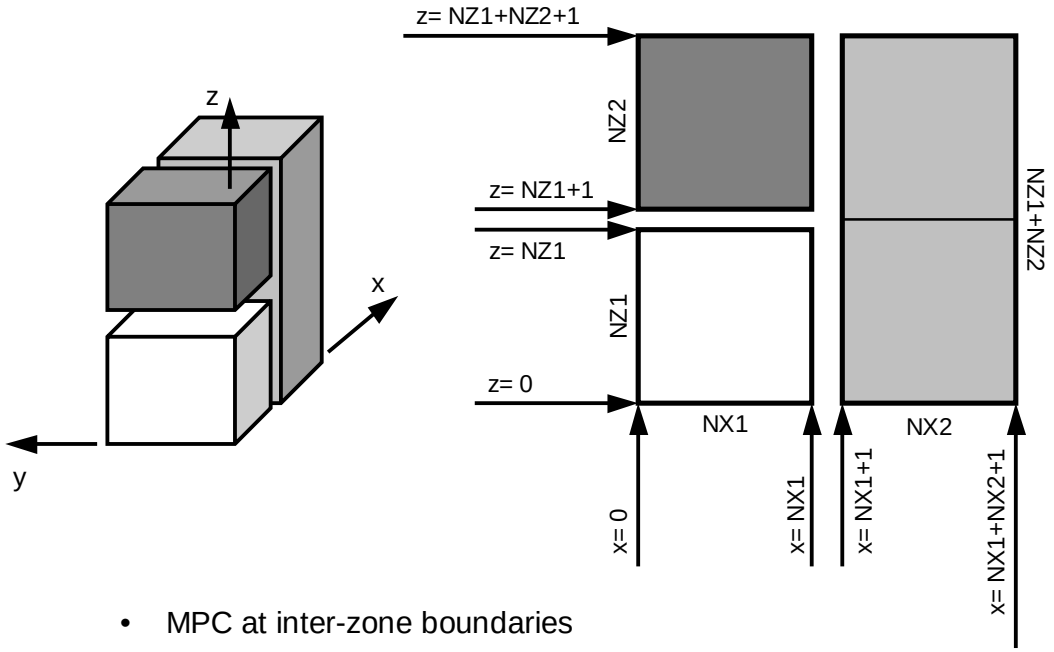
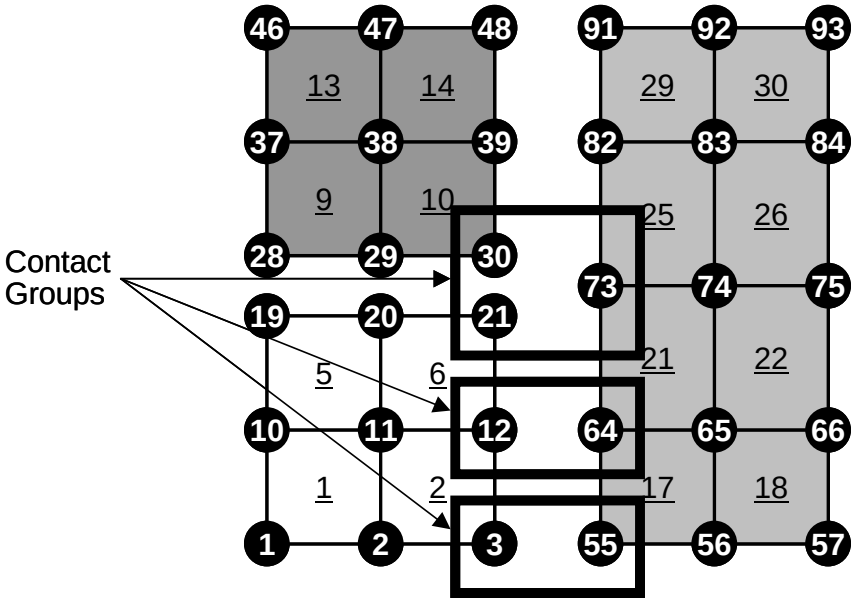


Fig. 19 Partitioning strategy for the nodes in contact groups [9]



- MPC at inter-zone boundaries
- Symmetric condition at the x=0 and y=0 surfaces
- Dirichlet fixed condition at the z=0 surface
- Uniform distributed load at the z= Zmax surface

(a) Model and boundary conditions



(b) Node, elements and contact groups

Fig. 20 Description of the simple block model [9]

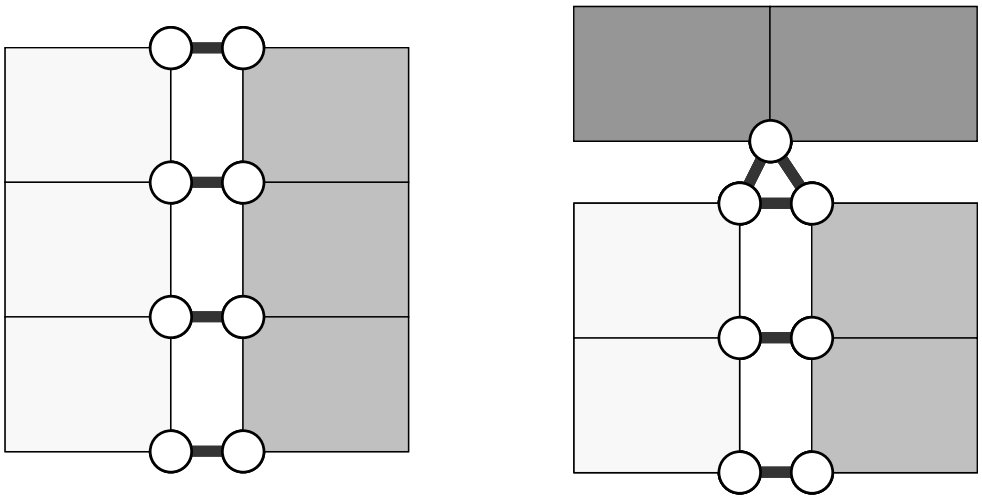


Fig. 21 111-type element (Rod/Beam) is put in each contact group and very large stiffness corresponding to penalty is applied [9]

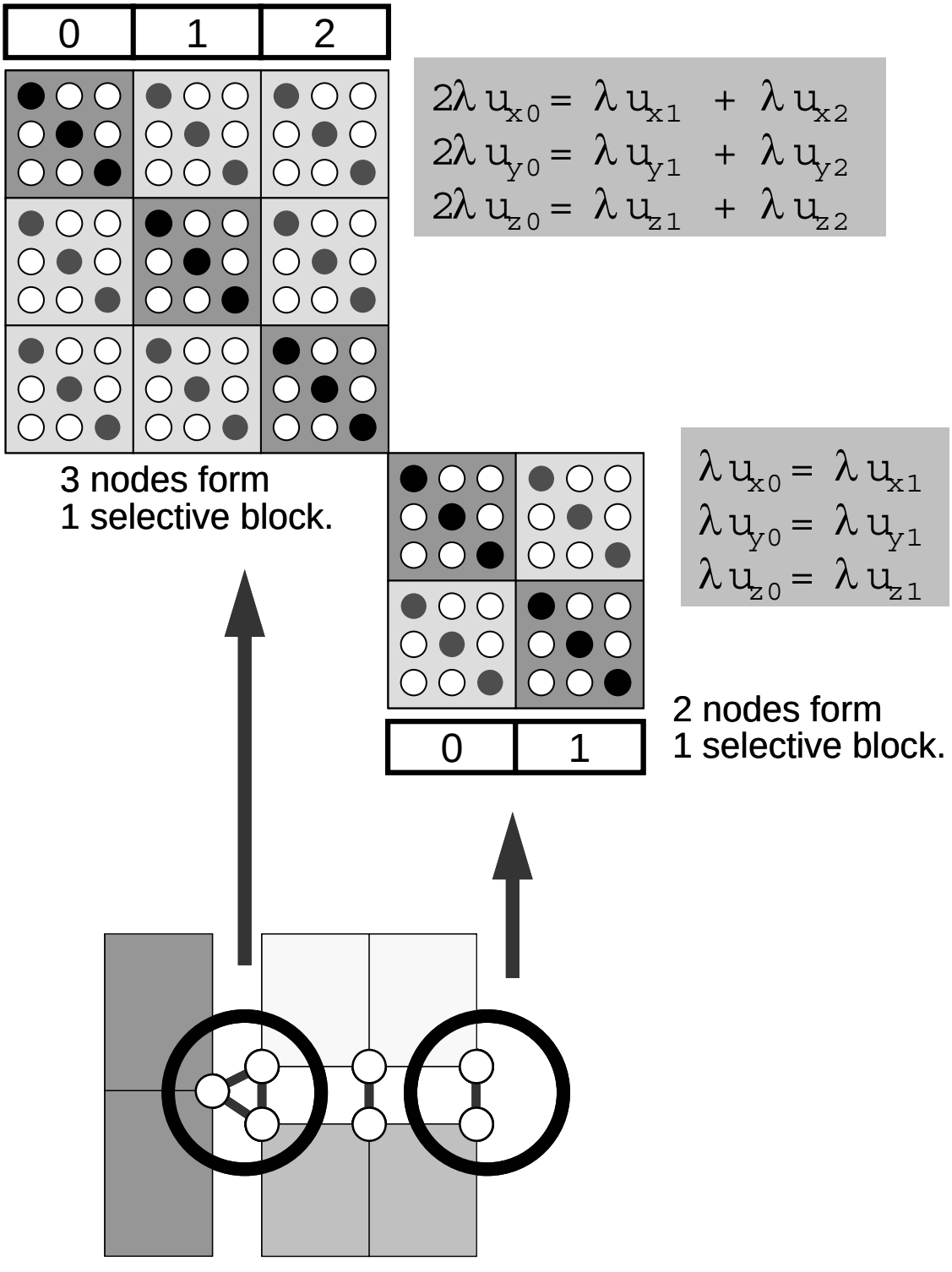


Fig. 22 Matrix operation of nodes in a contact group [9]

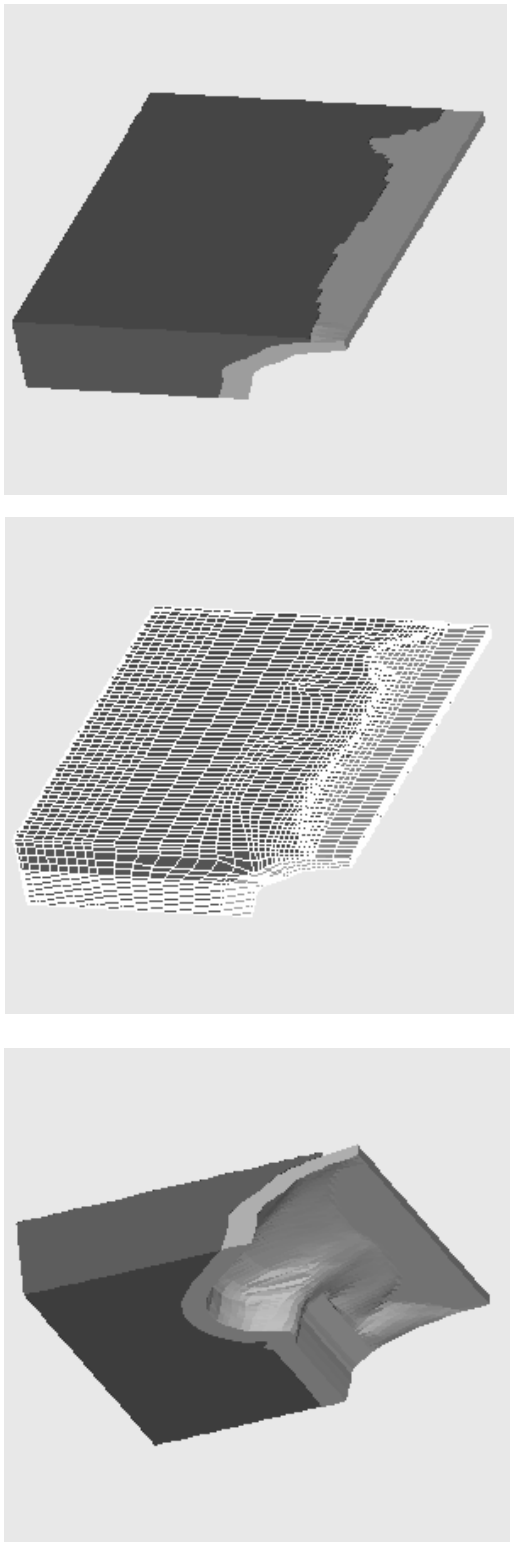


Fig. 23 Description of the Southwest Japan model This model consists crust (dark gray) and subduction plate (light gray). 27,195 nodes and 23,831 tri-linear (1st order) hexahedral elements are included [9].

Table 5 (a) Iterations/CPU time (includes factorization) for convergence ($\varepsilon=10^{-8}$) on a single PE of COMPAQ Alpha 21164/600MHz by preconditioned CG for the 3D elastic contact problem for simple block model with MPC condition in Fig.8 (83,664DOF).: **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering.

Preconditioning	λ	Iter #	sec.
BIC(0)	10^2	388	202.
	10^4	No Conv.	N/A
BIC(1)	10^2	77	89.
	10^6	77	89.
	10^{10}	78	90.
BIC(2)	10^2	59	135.
	10^6	59	135.
	10^{10}	60	137.
SB-BIC(0)	10^2	114	61.
	10^6	114	61.
	10^{10}	114	61.

Table 5 (b) Largest and smallest eigenvalues (E_{min}, E_{max}) and $\kappa= E_{max}/E_{min}$ of $[M]^{-1}[A]$ for a wide range of penalty parameter values: 3D elastic contact problem for simple block model with MPC condition in Fig.20 (83,664DOF).

Preconditioning		$\lambda=10^2$	$\lambda=10^6$	$\lambda=10^{10}$
BIC(0)	E_{min}	4.845568E-03	4.865363E-07	4.865374E-11
	E_{max}	1.975620E+00	1.999998E+00	2.000000E+00
	κ	4.077170E+02	4.110686E+06	4.110681E+10
BIC(1)	E_{min}	8.901426E-01	8.890643E-01	8.890641E-01
	E_{max}	1.013930E+00	1.013863E+00	1.013863E+00
	κ	1.139065E+00	1.140371E+00	1.140371E+00
BIC(2)	E_{min}	9.003662E-01	8.992896E-01	8.992895E-01
	E_{max}	1.020256E+00	1.020144E+00	1.020144E+00
	κ	1.133157E+00	1.134388E+00	1.134389E+00
SB-BIC(0)	E_{min}	6.814392E-01	6.816873E-01	6.816873E-01
	E_{max}	1.005071E+00	1.005071E+00	1.005071E+00
	κ	1.474924E+00	1.474387E+00	1.474387E+00

Table 6 Iterations/CPU time (includes factorization) for convergence ($\varepsilon=10^{-8}$) on a single PE of COMPAQ Alpha 21164/600MHz by preconditioned CG for the 3D elastic contact problem for Southwestern Japan model with MPC condition in Fig.23 (81,585DOF).: **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering.

Preconditioning	λ	Iter #	sec.
BIC(0)	10^2	344	172.
	10^4	No Conv.	N/A
BIC(1)	10^2	201	192.
	10^4	256	237.
	10^6	256	237.
	10^8	258	240.
	10^{10}	259	241.
BIC(2)	10^2	176	288.
	10^4	229	360.
	10^6	230	361.
	10^8	230	361.
	10^{10}	232	364.
SB-BIC(0)	10^2	297	149.
	10^4	295	148.
	10^6	295	148.
	10^8	295	148.
	10^{10}	295	148.

Table 7 Largest and smallest eigenvalues (E_{min}, E_{max}) and $\kappa = E_{max}/E_{min}$ of $[M]^{-1}[A]$ for a wide range of penalty parameter values: 3D elastic contact problem for Southwestern Japan model with MPC condition in Fig.23 (81,585DOF).

Preconditioning		$\lambda=10^2$	$\lambda=10^4$	$\lambda=10^6$	$\lambda=10^{10}$
BIC(0)	E_{min}	1.970395E-02	1.999700E-04	1.999997E-06	2.000000E-10
	E_{max}	1.005194E+00	1.005194E+00	1.005194E+00	1.005194E+00
	κ	5.101486E+01	5.026725E+03	5.025979E+05	5.025971E+09
BIC(1)	E_{min}	3.351178E-01	2.294832E-01	2.286390E-01	2.286306E-01
	E_{max}	1.142246E+00	1.142041E+00	1.142039E+00	1.142039E+00
	κ	3.408491E+00	4.976580E+00	4.994944E+00	4.995128E+00
BIC(2)	E_{min}	3.558432E-01	2.364909E-01	2.346180E-01	2.345990E-01
	E_{max}	1.058883E+00	1.088397E+00	1.089189E+00	1.089196E+00
	κ	2.975702E+00	4.602277E+00	4.642391E+00	4.642800E+00
SB-BIC(0)	E_{min}	2.380572E-01	2.506369E-01	2.507947E-01	2.507963E-01
	E_{max}	1.005194E+00	1.005455E+00	1.005465E+00	1.005466E+00
	κ	4.222491E+00	4.011600E+00	4.009117E+00	4.009092E+00

Table 8 Iterations/elapsed execution time (includes factorization, communication overhead) for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 128 PEs using preconditioned CG for the 3D elastic contact problem for simple block model with MPC condition in Fig.20 (2,471,439 DOF). Domains are partitioned according to the contact group information.: **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering.

Preconditioning	λ	Iter #	sec.
BIC(0)	10^2	998	118.
	10^4	No Conv.	N/A
BIC(1)	10^2	419	98.
	10^6	419	98.
	10^{10}	421	99.
BIC(2)	10^2	394	171.
	10^6	394	171.
	10^{10}	396	172.
SB-BIC(0)	10^2	565	71.
	10^6	566	71.
	10^{10}	567	72.

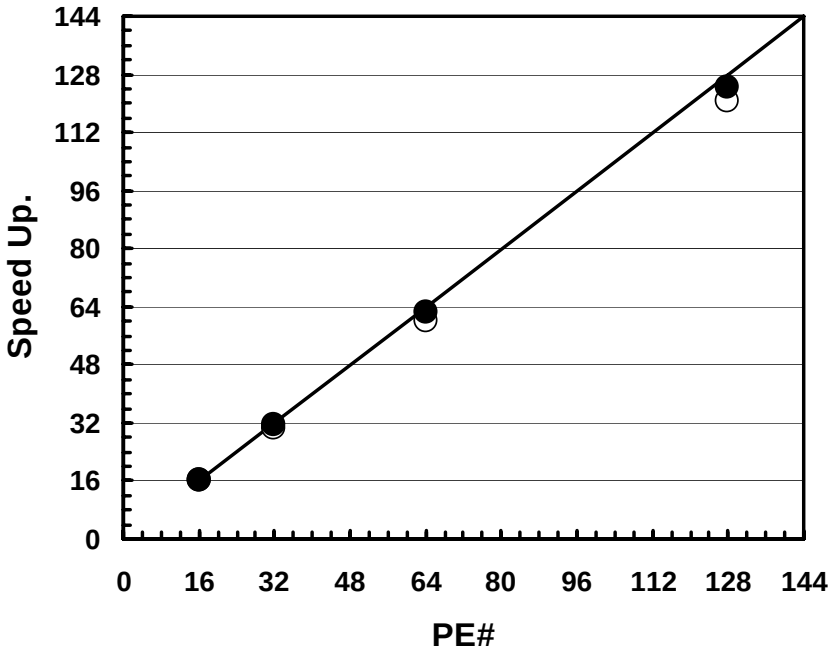
Table 9 Iterations/elapsed execution time (includes factorization, communication overhead) for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 16 to 128 PEs using preconditioned CG for the 3D elastic contact problem for simple block model with MPC condition in Fig.20 (2,471,439 DOF). Domains are partitioned according to the contact group information.: **BIC(n)**: Block IC with n-level fill-in, **SB-BIC(0)**: BIC(0) with the selective blocking reordering.

$\lambda=10^2$

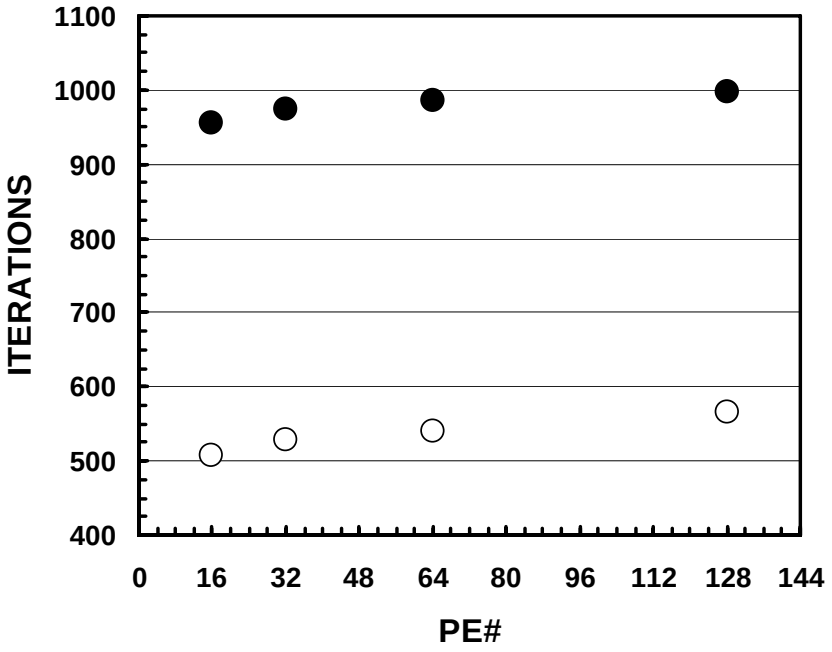
Preconditioning		16 PEs	32 PEs	64 PEs	128 PEs
BIC(0)	iters	956	975	986	998
	sec.	919	469	236	118
	ratio	16.0	31.4	62.5	124.4
BIC(1)	iters			396	419
	sec.	N/A	N/A	190	98
	ratio			64.0	124.3
BIC(2)	iters				394
	sec.	N/A	N/A	N/A	171
	ratio				-
SB-BIC(0)	iters	508	529	541	565
	sec.	540	282	144	71
	ratio.	16.0	30.7	60.2	120.7

$\lambda=10^6$

Preconditioning		16 PEs	32 PEs	64 PEs	128 PEs
BIC(1)	iters			395	419
	sec.	N/A	N/A	190	98
	ratio			64.0	124.2
BIC(2)	iters				394
	sec.	N/A	N/A	N/A	171
	ratio				-
SB-BIC(0)	iters	510	532	543	566
	sec.	542	283	144	71
	ratio	16.0	30.6	60.5	121.9



(a) Speed-up ratio



(b) Iteration number for convergence

Fig. 24 Parallel performance based on elapsed execution time including communication and iterations for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 16 to 128 PEs using preconditioned CG for the 3D elastic contact problem with MPC condition ($\lambda=10^{2}$) in Fig.20 (2,471,439 DOF). Domains are partitioned according to the contact group information. (White Circles: SB-BIC(0), Black-Circles: BIC(0)).

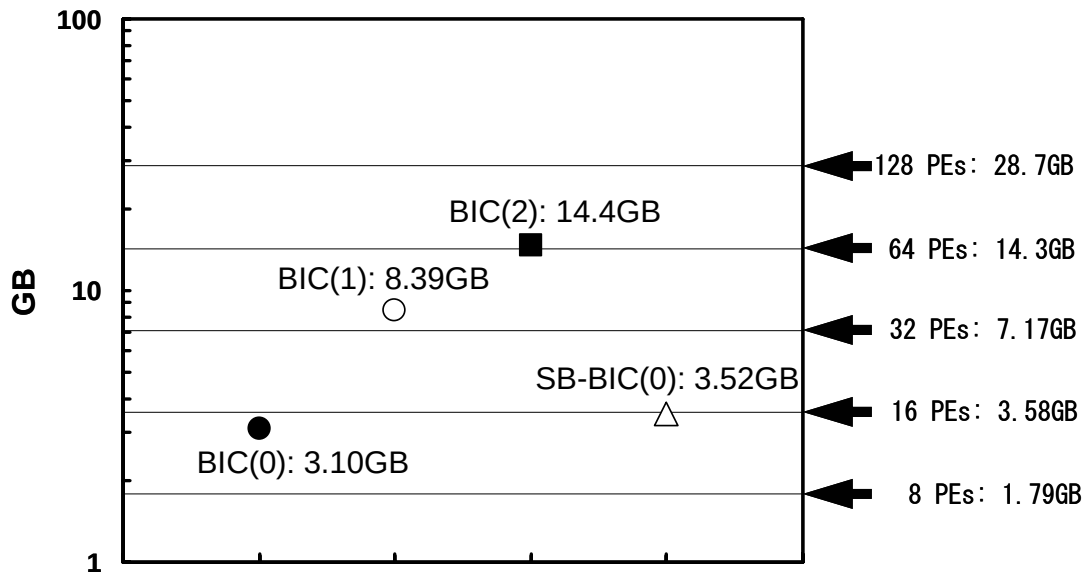
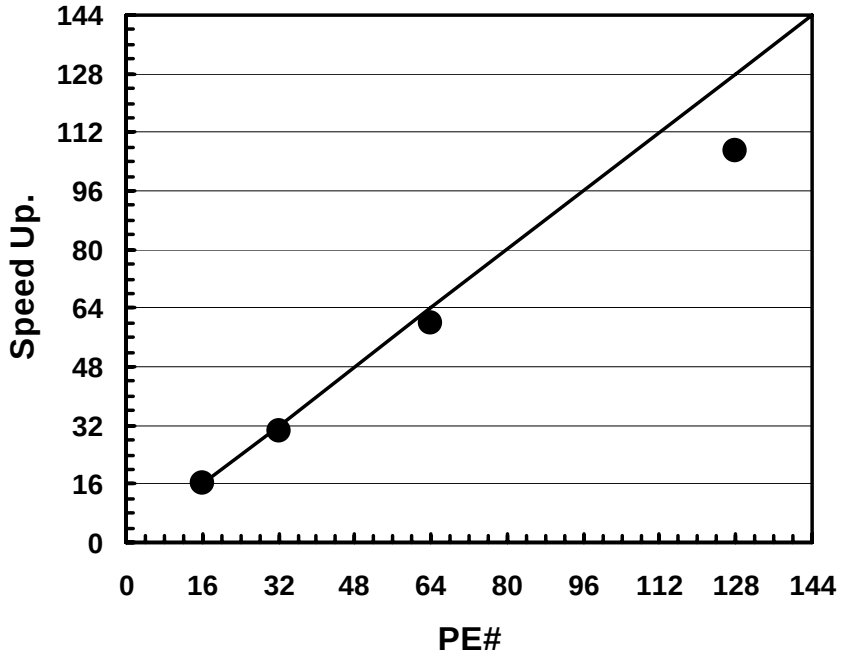


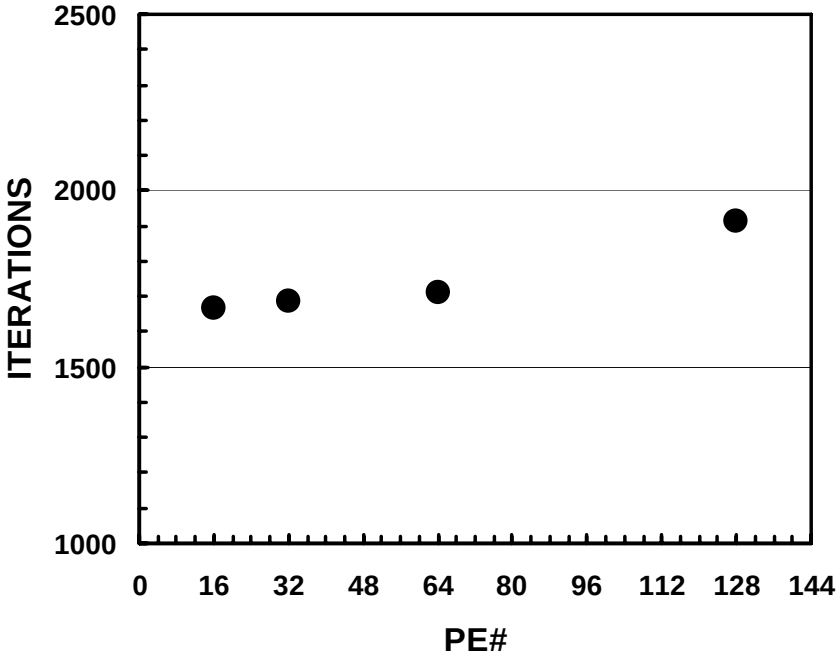
Fig. 25 Required memory size of CG solvers with various types preconditioners for the 3D elastic contact problem with MPC condition ($\lambda=10^2$) in Fig.20 (2,471,439 DOF) and available memory size on Hitachi SR2201 (Black-Circles: BIC(0), White-Circles: BIC(1), Black-Squares: BIC(2), White Triangles: SB-BIC(0)).

Table 10 Iterations/elapsed execution time (including factorization, communication overhead) for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 16 to 128 PEs using SB-BIC(0) CG for the 3D elastic contact problem for Southwest Japan model with MPC condition ($\lambda=10^6$) in Fig.23 (2,992,264 DOF). Domains are partitioned according to the contact group information.

Preconditioning		16 PEs	32 PEs	64 PEs	128 PEs
SB-BIC(0)	iters	1665	1686	1710	1912
	sec.	1901.	993.	506.	284.
	ratio	16.0	30.6	60.1	107.2



(a) Speed-up ratio



(b) Iteration number for convergence

Fig. 26 Parallel performance based on elapsed execution time including communication and iterations for convergence ($\varepsilon=10^{-8}$) on a Hitachi SR2201 with 16 to 128 PEs using SB-BIC(0) CG for the 3D elastic contact problem with MPC condition ($\lambda=10^{6}$) in Fig.23 (2,992,266 DOF). Domains are partitioned according to the contact group information.

1.6.5 Sparse Approximate Inverse (SAI)

SAI は「近似逆行列」であり，以下に示す式（1.7-1）の最小二乗問題によって係数行列 A の逆行列を近似するような前処理行列 M^{-1} を求めるものである：

$$\min \left\| AM^{-1} - E \right\|_F^2 = \sum_{k=1}^{n} \min_{m_k \in R^n} \left\| Am_k - e_k \right\|_2^2 \qquad (1.7\text{-}1)$$

ただし，ベクトル $\mathbf{m}_k$ は行列 M^{-1} の k 列目の列成分，ベクトル $\mathbf{e}_k$ は単位行列 E の k 列目の列成分を表している。ベクトル $\mathbf{m}_k$ は式(1.7-1)の右辺を最小化するように求めればよいが，計算量を減少させるために，あらかじめ前処理行列 M^{-1} の非ゼロ要素の場所を決定しておき，その非ゼロ要素のみを取り出すことを考える。最も簡単な方法としては，前処理行列 M^{-1} の非ゼロ要素の位置を係数行列 A と同じにすることである。ILU 系前処理と同様，Fill-in を許せば，前処理行列 M^{-1} の性能は向上するが，計算量，記憶容量ともに増加する。前処理行列 M^{-1} の非ゼロ要素の行インデックス集合を J，J の各々の行インデックスに対応する非ゼロ要素の列インデックス集合を I とすると，式(1.7-1)は：

$$\sum_{k=1}^{n} \min_{m_k(J) \in R^J} \left\| A(I, J)\mathbf{m}_k(J) - \mathbf{e}_k(I) \right\|_2^2 \qquad (1.7\text{-}2)$$

を解くことに帰着できる。ここで式（1.6-2）は，n 本の I×J 次の最小二乗問題：

$$\min_{m_k(J) \in R^J} \left\| A(I, J)\mathbf{m}_k(J) - \mathbf{e}_k(I) \right\|_2^2 \qquad (1 \le k \le n) \qquad (1.7\text{-}3)$$

の計算を独立に行なうことができるので，式（1.7-3）の最小二乗問題を各 CPU に均等に割り当てて，並列に計算することが可能となる。式（1.7-3）の最小二乗問題は Givens 回転行列を使用して QR 分解によって解くことが可能である。

HEC-MW では LAPACK の DGELS〔13〕を使用している。

Fig.15 に示すブロック間の接触問題〔9〕について，SAI を適用して計算し，他の前所為手法と比較した例を示す。Fig.20 に示す接触グループ（Contact Groups）を構成する節点にペナルティ条件を課する場合，並列計算においてはこれらの節点が同じ領域上にある場合（coupled partition）に収束の速いことが知られている〔9〕。逆に，同じ領域上にない場合（decoupled）は収束が非常に遅い。Table 11 に示すように，SAI は「decoupled partition」の場合も良好な収束を示すことがわかる。Table 12 に示すように，他の手法と比較して，

「set-up」に時間を要しているものの，概ね良好な収束を示していることがわかる。打ち切りパラメータを小さくすることによって，反復回数は減るが，計算時間は増加する。

Table 11 Comparison of preconditioners for simple block prblem with 27,888 nodes (83,664 DOF), SAI works well for 8PE case in "decoupled" partitioning, Xeon 2.8GHz Cluster.

1 PE

	SAI/GPBiCG	SAI/BiCGSTAB	SB-BILU(0)CG	BILU(1)-CG
ITERs	190	187	114	78
set-up	10.7	10.7	<0.01	8.2
solver	23.9	17.3	18.6	11.7

8 PEs (decoupled)

	SAI/GPBiCG	SAI/BiCGSTAB	SB-BILU(0)CG	BILU(1)-CG
ITERs	191	193	3498	2701
set-up	1.0	1.0	<0.01	0.5
solver	4.1	3.0	56.9	46.9

8 PEs (coupled)

	SAI/GPBiCG	SAI/BiCGSTAB	SB-BILU(0)CG	BILU(1)-CG
ITERs			166	123
set-up			<0.01	0.5
solver			2.8	2.2

Table 12 Comparison of preconditioners for simple block prblem with 2.4M DOF, coupled partitttioing, Xeon 2.8GHz Cluster with 32 PEs.

Time	BIC(1)-CG	BIC(2)-CG	SB-BIC(0)CG	SAI/BiCGSTAB		
				0.20-0.20	0.10-0.10	0.05-0.05
ITERs	382	333	535	843	671	545
Solver	87.4	117.6	124.0	136.0	121.7	119.1
Set-up+ Solver	108.2	149.6	124.0	155.2	142.7	200.2

2. ソフトウェアの使用法

2.1 概 要

本年度開発した「線形ソルバー」機能では，Table 13 に示すような解法，前処理手法を使用することが可能である。

Table 13 Available solvers and preconditioning methods in ***hecmw-PC-cluster*** (red circles: recommended methods)

		CG	BiCGSTAB	GPBiCG	GMRES
1D	ILU(0)	●			
	P-Jacobi	○			
	SAI				
2D	BILU(0)	●			
	BILU(1)	●			
	BILU(2)	○			
	B-Jacobi	○			
	SAI				
3D	BILU(0)	●	●	●	○
	BILU(1)	●	●	●	
	BILU(2)	○	○	○	
	B-Jacobi	○	○	○	○
	SAI		○	○	○
	S-B	○			

大きくわけて，1D（1節点1自由度），2D（1節点2自由度），3D（3節点3自由度）の3種類のソルバーを扱うことができ，各節点上の自由度はブロック化されて扱われている。

各ソルバーの呼び出しは，Fig. 16に示すように，以下の手順で実施される：

- モジュール「hecmw」のuse
- 構造体「hecmwST_local_mesh」，「hecmwST_matrix」の定義
- サブルーチン「hecmw_solve_init」による初期化
- サブルーチン「hecmw_solve_NN（1D:NN=11，2D:NN=22，3D:NN=33)」による計算

このうち，「hecmw_solve_NN」については，次期バージョンでは統一されたインタフェース

で使用できるように変更する予定である。

```
 use hecmw
 type (hecmwST_matrix)     :: hecMAT
 type (hecmwST_local_mesh) :: hecMESH
...
 call hecmw_solve_init (hecMAT)
...
 call hecmw_solve_33 (hecMAT, hecMESH)
```

Fig.28 How to use linear solvers in HEC-MW

これ以外に，並列計算補助サブルーチンとして：

- 領域間通信：
  - ✧ hecmw_update_1_R
  - ✧ hecmw_update_2_R
  - ✧ hecmw_update_3_R
  - ✧ hecmw_update_m_R
- MPI_ALLREDUCE 相当：
  - ✧ hecmw_allreduce_R
  - ✧ hecmw_allreduce_I
- MPI_BCAST 相当
  - ✧ hecmw_bcast_R
  - ✧ hecmw_bcast_I
  - ✧ hecmw_bcast_C
- MPI_BARRIER 相当
  - ✧ hecmw_barrier

などのサブルーチンを利用することができる。これらは，MPI のサブルーチンを内部から呼び出しているが，MPI を直接呼ぶよりもはるかに少ない引数で容易に使用することができる。利用法の詳細は，「2.3 API」に示す。

2.2 使用法

Fig. 17 は HEC-MW を使用した FEM プログラムのメイン処理部である。太字で書かれているサブルーチンが HEC-MW でサポートしているサブルーチンである。ソルバーを使用するために，利用者がやらなければならないことは：

- 係数マトリクスの記憶領域を確保し，内容を定義する。
- 解法，前処理，反復回数等に関わるパラメータを定義する。

ことである。以下にそれぞれについて示す。

```
program psan_main
use m_psan

implicit REAL*8 (A-H,O-Z)
type (hecmwST_local_mesh) :: hecMESH
type (hecmwST_matrix    ) :: hecMAT
type (psan_solid        ) :: psanSOLID
type (hecmwST_result_data):: psanRESULT
character(len=HECMW_FILENAME_LEN) :: name_ID

call hecmw_init()

name_ID='psanMSH'
call hecmw_get_mesh(name_ID,hecMESH)

name_ID='psanCNT'
call hecmw_ctrl_get_control_file(name_ID,cntfilNAME)
call psan_input_cntl(hecMESH,hecMAT)

call psan_init_prop (hecMESH, psanSOLID)
call psan_mat_con   (hecMESH, hecMAT)

call psan_mat_ass(hecMESH, hecMAT, psanSOLID)

call hecmw_solve_33 (hecMESH, hecMAT)
call hecmw_update_3_R (hecMESH, hecMAT%X, hecMAT%NP)
call psan_output3d(hecMESH,hecMAT,psanSOLID)

call psan_post(hecMESH,psanSOLID)
call psan_make_result(hecMESH,psanSOLID,psanRESULT)

call hecmw_init_for_visualize
call hecmw_visualize( psanRESULT, 0, 0, 0)
call hecmw_finalize_for_visualize

call hecmw_finalize()

end program psan_main
```

Fig.17 FEM procedure developed on HEC-MW
(Subroutines in bold letters are provided by HEC-MW)

(1) 係数マトリクスの定義

HEC-MW における線形ソルバの係数マトリクスの格納法は，ハードウェアによって異なるが，「hecmw-PC-cluster ver2.1」では，一次元圧縮係数行列（Compressed Row Storage，CRS，http://www.netlib.org/templates/index.html，〔2〕）の形式によっている。Fig.18 にその概要を示す。上下三角成分，対角成分を別々の配列に格納しているのが特徴である。このような手法によって，非対角成分の数が，要素によって異なるような不規則な行列においても，最小限の記憶容量で係数マトリクスを記憶させることができる。Fig.19 に例（1D の場合）を示す。

Fig.19 は１D の場合であるが，2D，3D の場合は各節点に 2 または 3 の自由度が定義される。したがってマトリクスの成分（hecMAT%D，hecMAT%AL，hecMAT%AU）においては，4 または 9 の要素が定義される。HEC-MW ではメモリアクセスの負担を減らすために，これらの要素をブロック化してまとめて取り扱い，一次元配列に格納している。したがって，例えば，2D の場合に下三角成分の k 番目の要素では AL(4k-3)，AL(4k-2)，AL(4k-1)，AL(4k) の 4 成分が定義され,3D の場合に上三角成分の m 番目の要素では AU(9m-8)，AU(9m-7)，AU(9m-6)，AU(9m-5)，AU(9m-4)，AU(9m-3)，AU(9m-2)，AU(9m-1)，AU(9m) の 9 成分が定義される（Fig.20)。

```
use hecmw
type hecmwST_matrix
          integer  NDOF                                  !節点自由度数
          integer  N                                     !内点数
          integer  NP                                    !総節点数
          integer  NPL                                   !下三角成分総数
          integer  NPU                                   !上三角成分総数
          real(kind=kreal), pointer :: D(NP)             !対角成分
          real(kind=kreal), pointer :: B(NP)             !右辺ベクトル
          real(kind=kreal), pointer :: X(NP)             !解ベクトル
          real(kind=kreal), pointer :: AL(NPL)           !下三角成分
          real(kind=kreal), pointer :: AU(NPU)           !上三角成分
          integer(kind=kint), pointer :: indexL(0:NPL)   !下三角成分インデクス
          integer(kind=kint), pointer :: indexU(0:NPU)   !上三角成分インデクス
          integer(kind=kint), pointer :: itemL(0:NP)     !下三角インデクス
          integer(kind=kint), pointer :: itemU(0:NP)     !上三角インデクス
```

Fig.18　係数行列の格納（CRS 形式）

	1	2	3	4	5	6	7	8
1	1		1		3	4		
2		2		2	3	5		
3	1		4		4		5	
4		9		3		10		
5			32		4		2	
6		1		5		6		7
7	1		1				1	8
8		3		5	6			2

```
NP, NPL, NPU   8, 10, 12
D              {1, 2, 4, 3, 4, 6, 1, 2}
AL             {1, 9, 32, 1, 5, 1, 1, 3, 5, 6}
indexL         {0: 0, 0, 1, 2, 3, 5, 7, 10}
itemL          {1, 2, 3, 2, 4, 1, 3, 2, 4, 5}
AU             {1, 3, 4, 2, 3, 5, 4, 5, 10, 2, 7, 8}
indexU         {0: 3, 6, 8, 9, 10, 11, 12, 12}
itemU          {3, 5, 6, 4, 5, 6, 5, 7, 6, 7, 8, 8}
```

Fig.19 Example of Coefficient Matrix in CRS Format (1D, 8x8)

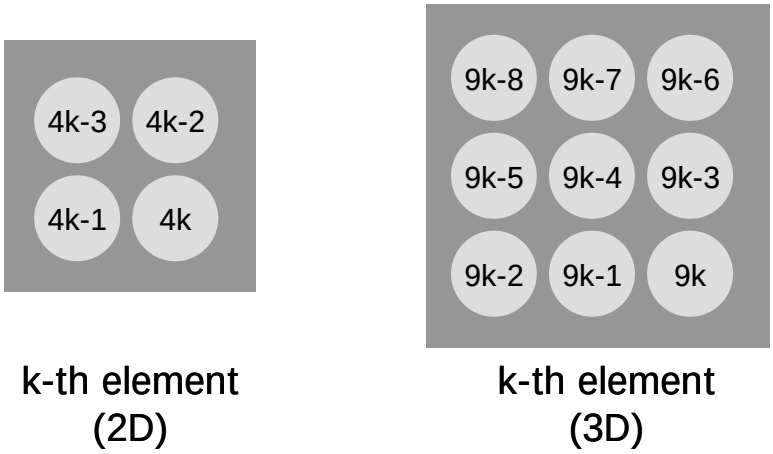


Fig.20 Block Storage in CRS Format

（2）パラメータの定義

HEC-MW における線形ソルバーでは，hecMAT%Iarray，hecMAT%Rarray という整数型，実数型のパラメータ用の配列が用意されており，それらの値を定義することによって，解法の選択，収束判定を実施する。以下に各パラメータの内容を示す。

hecMAT%Iarray (1)

反復回数（必須），デフォルト値＝100

hecMAT%Iarray (2)

反復解法（必須）： 1:CG, 2:BiCGSTAB, 3:GMRES, 4:GPBiCG

hecMAT%Iarray (3)

前処理手法（必須）

- 1:(B)IC(0), 2:(B)SSOR(0), 3：(B)DIAG,
- 10:(B)IC(0), 11：(B)IC(1), 12：(B)IC(2),
- 21:SAI （Sparse Approximate Inverse）

注 1） 前処理手法と反復解法の可能な組み合わせは「Table 5」に示した通りである。これ以外の場合はエラーとなる。

注 2） 前処理として SAI を使用する場合は，領域分割におけるオーバーラップ深さを「2」にしなくてはならない。

hecMAT%Iarray (4)

NSET（必須）

- 0：初めてソルバーを呼ぶ場合（default）
- -1：行列の内容が変わった場合
- +1：上記のいずれでもない場合

hecMAT%Iarray (5)

Additive Schwartz の繰り返し数（必須）

=2 がおすすめ（デフォルト値＝0）

hecMAT%Iarray (6)

クリロフ部分空間数（解法として GMRES を選択した場合のみ）（デフォルト値＝10）

hecMAT%Iarray (21)

ソルバー収束履歴出力の有無（1:YES, 0：NO（デフォルト値））

hecMAT%Iarray (22)

ソルバー計算時間出力の有無（1:YES, 0：NO（デフォルト値））

hecMAT%Rarray (1)

残差，打ち切り誤差（必須），（デフォルト値=1.d-08）

hecMAT%Rarray (2)

=1.d0 （固定）

hecMAT%Rarray (3)

=0.d0 （固定）

hecMAT%Rarray (4)

SAI パラーメータ①（THRESH），前処理として SAI を使用した場合（デフォルト値＝0.10）

hecMAT%Rarray (5)

SAI パラーメータ②（FILTER），前処理として SAI を使用した場合（デフォルト値＝0.10）

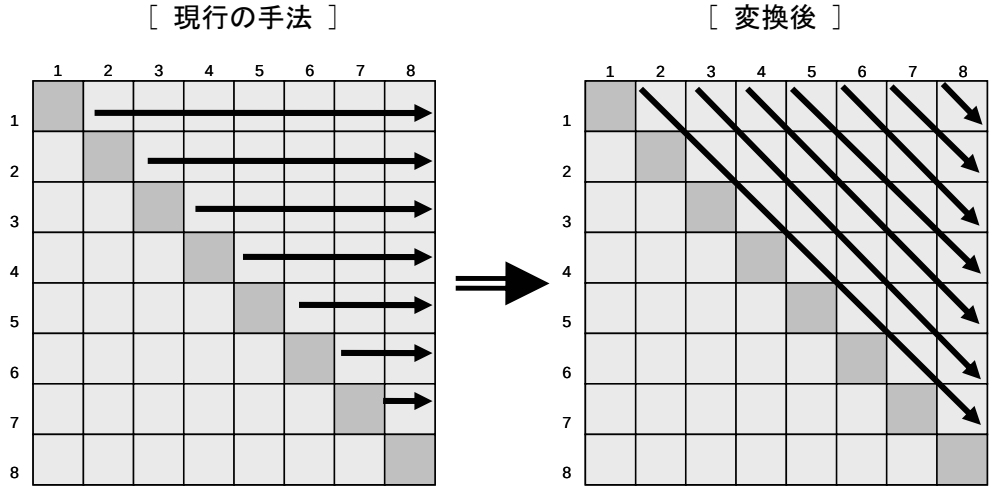
hecMAT%Rarray (11)

多点拘束の組込みに用いるペナルティ値（デフォルト値＝10^4）

HEC-MW を使用して開発された並列有限要素法による三次元弾性解析コード「FrontSTR」では，これらのパラメータを制御ファイルから定義できるようになっている。詳細は「FrontSTR」利用マニュアルを参照されたい。

(3) ベクトル型スーパーコンピュータ向けルーチンについて

ベクトル型スーパーコンピュータを効率良く利用する為には、[行列]x[ベクトル]の計算式実行部分の最内ループにおいて、そのループ長が長くなるような配列参照である必要がある。以下の図は現行ルーチンと今回リリースのベクトル型スーパーコンピュータ向けルーチンそれぞれの解法の構造である。



矢印方向が、それぞれの手法のデータの格納順序を表している。

上の例のように上三角成分のみに着目した場合、左図の現行の方法では配列要素の行方向に向かって順次積算するが、この方法ではループ長の最大値は「要素成分の数」となってしまう。これを右図のように斜め方向に要素を並べるよう、行列の変換プログラムを用意した。これが以下の変換サブルーチンである：

hecmw_mat_conv

なお、ディレクティブについてはユーザが適切な記述に書き換える必要がある。これは、ディレクティブはそのコンパイラの種類・バージョン、あるいはまたそのマシンのメーカーにより異なることが普通であるからである。詳しくはその環境に応じたマニュアルを参照されたい。

ディレクティブ例（NEC FORTRAN90/SX の場合）:

参照に重複や矛盾は生じないのでベクトル化せよ→	!CDIR NODEP
長いループ長が取れないのでベクトル化を禁止せよ→	!CDIR NOVECTOR

2.3 API

各サブルーチンの呼び出し API を以下に示す。

hecmw_solve_11

線形ソルバー（1 節点あたり 1 自由度）を呼び出します。

```
subroutine hecmw_solve_11(mesh, matrix)

type(hecmwST_local_mesh) :: mesh
type(hecmwST_matrix    ) :: matrix
```

引数

mesh

メッシュデータの格納先

matrix

マトリクスデータおよびソルバー制御情報の格納先

説明

- 線形ソルバー（1 節点あたり 1 自由度）を呼び出します。
- 係数マトリクスに関する情報，右辺ベクトル，ソルバー制御情報は全て「matrix」に格納されています。
- 「matrix」の内容については，「2.2」をご覧ください。

hecmw_solve_22

線形ソルバー（1 節点あたり 2 自由度）を呼び出します。

```
subroutine hecmw_solve_22(mesh, matrix)

type(hecmwST_local_mesh) :: mesh
type(hecmwST_matrix    ) :: matrix
```

引数

mesh
: メッシュデータの格納先

matrix
: マトリクスデータおよびソルバー制御情報の格納先

説明

- 線形ソルバー（1 節点あたり 2 自由度）を呼び出します。
- 係数マトリクスに関する情報，右辺ベクトル，ソルバー制御情報は全て「matrix」に格納されています。
- 「matrix」の内容については，「2.2」をご覧ください。

hecmw_solve_33

線形ソルバー（1 節点あたり 3 自由度）を呼び出します。

```
subroutine hecmw_solve_33(mesh, matrix)

type(hecmwST_local_mesh) :: mesh
type(hecmwST_matrix    ) :: matrix
```

引数

mesh
: メッシュデータの格納先

matrix
: マトリクスデータおよびソルバー制御情報の格納先

説明

- 線形ソルバー（1 節点あたり 3 自由度）を呼び出します。
- 係数マトリクスに関する情報，右辺ベクトル，ソルバー制御情報は全て「matrix」に格納されています。
- 「matrix」の内容については，「2.2」をご覧ください。

hecmw_barrier

MPI プロセスの同期のための Barrier を設定します。

```
subroutine hecmw_barrier(mesh)

type(hecmwST_local_mesh) :: mesh
```

引数

mesh

対象とするコミュニケーショングループのメッシュデータの格納先

説明

- 内部からは，MPI_BARRIER を呼び出しています。

hecmw_allREDUCE_R

実数値または実数ベクトルの各成分に関してグローバルな演算を実施します。

```
subroutine hecmw_allREDUCE_R(mesh, VAL, m, FLAG)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: m, FLAG

real(kind=kreal) :: VAL
   or
real(kind=kreal), dimension(m) :: VAL
```

引数

mesh

対象とするコミュニケーショングループのメッシュデータの格納先

VAL

対象とする実数ベクトル（またはスカラー）

m

ベクトルのサイズ（スカラーの場合は 1）

FLAG

グローバル演算の識別子

hecmw_sum	総　和
hecmw_min	最小値
hecmw_max	最大値

説明

- 内部からは，MPI_ALLREDUCE を呼び出しています。

hecmw_allREDUCE_I

整数値または整数ベクトルの各成分に関してグローバルな演算を実施します。

```
subroutine hecmw_allREDUCE_I(mesh, VAL, m, FLAG)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: m, FLAG

integer(kind=kint) :: VAL
   or
integer(kind=kint), dimension(m) :: VAL
```

引数

mesh

対象とするコミュニケーショングループのメッシュデータの格納先

VAL

対象とする整数ベクトル（またはスカラー）

m

ベクトルのサイズ（スカラーの場合は 1）

FLAG

グローバル演算の識別子

hecmw_sum	総　和
hecmw_min	最小値
hecmw_max	最大値

説明

- 内部からは，MPI_ALLREDUCE を呼び出しています。

hecmw_bcast_R

実数値または実数ベクトル値を全プロセッサに送信します。

```
subroutine hecmw_bcast_R(mesh, VAL, m, nbase)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: m, nbase

real(kind=kreal) :: VAL
   or
real(kind=kreal), dimension(m) :: VAL
```

引数

mesh
: 対象とするコミュニケーショングループのメッシュデータの格納先

VAL
: 対象とする実数ベクトル（またはスカラー）

m
: ベクトルのサイズ（スカラーの場合は 1）

nbase
: 発信元の PE のプロセス ID（0 から開始）

説明

- 内部からは，MPI_BCAST を呼び出しています。

hecmw_bcast_I

整数値または整数ベクトル値を全プロセッサに送信します。

```
subroutine hecmw_bcast_I(mesh, VAL, m, nbase)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: m, nbase

integer(kind=kint) :: VAL
   or
integer(kind=kint) , dimension(m) :: VAL
```

引数

mesh
: 対象とするコミュニケーショングループのメッシュデータの格納先

VAL
: 対象とする整数ベクトル（またはスカラー）

m
: ベクトルのサイズ（スカラーの場合は 1）

nbase
: 発信元の PE のプロセス ID（0 から開始）

説明

- 内部からは，MPI_BCAST を呼び出しています。

hecmw_bcast_C

Character 変数値または Character 変数ベクトル値を全プロセッサに送信します。

```
subroutine hecmw_bcast_I(mesh, VAL, m, LEN, nbase)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: m, length, nbase

character (length=LEN) :: VAL
   or
character (length=LEN), dimension(m) :: VAL
```

引数

mesh
: 対象とするコミュニケーショングループのメッシュデータの格納先

VAL
: 対象とする Character 変数ベクトル（またはスカラー）

m
: ベクトルのサイズ（スカラーの場合は 1）

LEN
: Character 変数の文字長

nbase
: 発信元の PE のプロセス ID（0 から開始）

説明

- 内部からは，MPI_BCAST を呼び出しています。

hecmw_update_1_R

オーバーラップ領域における実数ベクトル（1 節点あたり 1 自由度）の値を，領域間通信によって更新します。

```
subroutine hecmw_update_1_R (mesh, VAL, N)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: N
real(kind=kreal), dimension(N) :: VAL
```

引数

mesh
: メッシュデータの格納先

VAL
: 実数ベクトル値

N
: ベクトルのサイズ

説明

hecmw_update_2_R

オーバーラップ領域における実数ベクトル（1 節点あたり 2 自由度）の値を，領域間通信によって更新します。

```
subroutine hecmw_update_2_R(mesh, VAL, N)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: N
real(kind=kreal), dimension(2*N) :: VAL
```

引数

mesh

メッシュデータの格納先

VAL

実数ベクトル値

N

ベクトルのサイズ

説明

hecmw_update_3_R

オーバーラップ領域における実数ベクトル（1 節点あたり 3 自由度）の値を，領域間通信によって更新します。

```
subroutine hecmw_update_3_R(mesh, VAL, N)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: N
real(kind=kreal), dimension(3*N) :: VAL
```

引数

mesh

メッシュデータの格納先

VAL

実数ベクトル値

N

ベクトルのサイズ

説明

hecmw_update_m_R

オーバーラップ領域における実数ベクトル（1 節点あたり m 自由度）の値を，領域間通信によって更新します。

```
subroutine hecmw_update_2_R(mesh, VAL, N, m)

type(hecmwST_local_mesh) :: mesh
integer(kind=kint) :: N, m
real(kind=kreal), dimension(2*N) :: VAL
```

引数

mesh
: メッシュデータの格納先

VAL
: 実数ベクトル値

N
: ベクトルのサイズ

m
: 節点あたりの自由度数

説明

hecmw_solve_direct

直接法線形ソルバーを呼び出します。

```
subroutine hecmw_solve_direct(mesh,mat,msg)

type(hecmwST_local_mesh) :: mesh
type(hecmwST_matrix) ::mat
```

引数

mesh

メッシュデータの格納先(解析対象の自由度数を得る)

mat

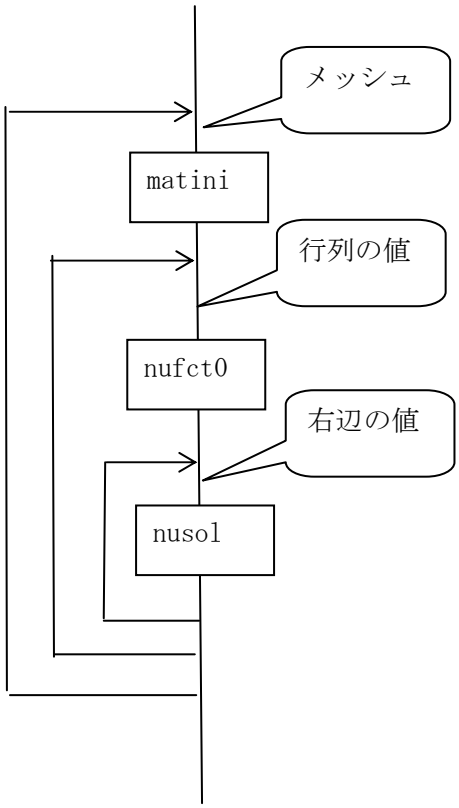
行列データ

説明

有限要素法のプログラムにおいて次ページ図のようなループが存在する。直接法の行列ソルバーを利用するに際してループに合わせて適切にルーチンを呼び出すことにより効率よい計算を行える。例えば行列の値は変更なく右辺だけが変更になるような時は、分解ルーチン(nufct)を呼び出すことはせず、求解ルーチンを繰り返す呼び出すようにする。
このように直接法ソルバーでは３つのルーチンの API が存在する。
この制御のために mat%Iarray(97)と mat%Iarray(98)が使われている。

mat%Iarray(98)=0: matini は呼び出さない。
mat%Iarray(98)=1: matini を呼び出す

mat%Iarray(97)=0:nufct0 は呼び出さない。
mat%Iarray(97)=1:nufct0 を呼び出す



図：FEM プログラムの制御

hecmw_solve_direct_parallel

並列直接法線形ソルバーを呼び出します。。

```
subroutine hecmw_solve_direct_parallel(mesh,mat,msg)

type(hecmwST_local_mesh) :: mesh
type(hecmwST_matrix) ::mat
integer(kind=kint) :: msg
```

引数

mesh

メッシュデータの格納先(解析対象の自由度数を得る)

mat

行列データ

msg

メッセージ出力ファイル装置番号

説明

mpirun 等で指定されたプロセス数で並列計算を行います。領域分割はソルバー内部で行うため、メッシュデータは全体領域メッシュを用います。領域分割は二分割を繰り返して行なわれ、計算全体をコントロールするプロセスが一つ存在するため、全体で 2**n+1 個(3,5,9,17…)プロセスが必要となります。分割数が増えるに従って子プロセスが使用するメモリ量は減少しますが、コントロールプロセスが使用するメモリ量が増えるため、問題の大きさに対して適切なプロセス数を指定して下さい。行列が小さすぎて分割に失敗した場合、標準出力にエラーを表示して終了します。

求解後、全プロセスが同じ解ベクトルを返します。

ソルバを呼び出す前に次の二つのフラグを設定しておいて下さい。

mat%Iarray(98)=1: matini を呼び出す(行列の設定を行う)

mat%Iarray(97)=1: nufct0 を呼び出す(行列の LU 分解を行う)

一度ソルバを呼び出した後、異なる右辺で再度求解を行う場合は、上記の二つのフラグを 0 に設定した上で再度ソルバを呼び出して下さい。メッシュ及び行列要素の値が変更される場合の再求解には対応していません。

2.4 エラーメッセージ

HEC-MW-SOLVER-E-1001

inconsistent solver/preconditioning

反復解法，前処理手法の組み合わせが正しくない。

HEC-MW-SOLVER-E-2001

ZERO component in diagonal block

対角成分に 0 が含まれている。

HEC-MW-SOLVER-W-2002

ZERO RHS norm

右辺ベクトルのノルムが 0 である（警告のみで計算継続)。

HEC-MW-SOLVER-W-3001

not converged within ceratin iterations

指定回数以内に収束しない（警告のみで計算継続)。

参考文献

（1） R. Barrett, M. Berry, T.F. Chan, J.W. Demmel, J. Donato, J.J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Horst: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994.

（2） J.J. Dongarra, I. Duff, D.C. Sorensen and H.A. van der Vorst: *Numerical Linear Algebra for High-Performance Computers*, SIAM, 1998.

（3） K. Garatani, H. Nakamura, H. Okuda and G. Yagawa: *GeoFEM: High Performance Parallel FEM for Solid Earth*, Lecture Notes in Computer Science No.1593, pp.132-140, 1999.

（4） D. Hysom and A. Pothen: *Efficient Parallel Computaion of ILU(k) Preconditioners*, NASA/CR-2000-210210, ICASE Report No.2000-23, 2000.

（5） K. Nakajima, H. Nakamura and T. Tanahashi: *Parallel Iterative Solvers with Localized ILU Preconditioning*, Lecture Notes in Computer Science 1225, pp.342-350, 1997.

（6） K. Nakajima and H. Okuda: *Parallel Iterative Solvers with Localized ILU Preconditioning for Unstructured Grids*, IMACS Series in Computational and Applied Mathematics Volume 5: Iterative Methods in Scientific Computation IV, p.85-98, 1999.

（7） K. Nakajima and H. Okuda: *Parallel Iterative Solvers with Localized ILU Preconditioning for Unstructured Grids on Workstation Cluster*, International Journal for Computational Fluid Dynamics, Vol.12, pp.315-322, 1999.

（8） Nakajima, K.: "OpenMP/MPI Hybrid vs. Flat MPI on the Earth Simulator: Parallel Iterative Solvers for Finite Element Method", *International Workshop on OpenMP: Experiences and Implementations (WOMPEI 2003)*, Tokyo, Japan, *Lecture Notes in Computer Science 2858*, pp.486-499, Springer, 2003.

（9） K.Nakajima, " Parallel Iterative Solvers of GeoFEM with Selective Blocking Preconditioning for Nonlinear Contact Problems on the Earth Simulator", *SC2003 Technical Session*, Phoenix, Arizona, 2003.

（10） Y. Saad: *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1996.

（11） H.D. Simon: *Partitioning of unstructured problems for parallel processing*, *Computing Systems in Engineering*, Vol.2, pp.135-148, 1991.

（12） B. Smith, P. Bjϕrstad and W. Gropp: *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.

（13）*LAPACK User's Guide*
（14）*METIS* Web Site: http://www-users.cs.umn.edu/~karypis/metis/
（15）*MPI (Message Passing Interface) Forum* Web Site: http://www.mpi-forum.org/