東京大学大学院新領域創成科学研究科
人間環境学専攻

平成 26 年度
修士論文

並列有限要素解析のための A-直交過程に基づく RIF 前処理

2015 年 3 月提出
指導教員 奥田 洋司 教授

47-136725 森田 直樹

目次

図目次

表目次

第 1 章　序論

1.1　研究背景

有限要素法 (Finite Element Method, FEM) を用いた構造解析は，工学分野で広く利用されている．計算機の発展と高精度な解析の必要性に伴って，構造解析の計算規模も増大している．工学分野の中で板状構造は，航空機，自動車，建屋など広範囲に使用され工業的に重要である．有限要素法を用いた構造解析では，板状構造を有する場合，多くの解析メッシュにシェル要素が用いられる．シェル要素は板状構造の挙動を良く再現できる．鉄筋コンクリートや積層材料など，板厚方向に物性が違う層が重なっているものには，積層シェル要素が使用されることが多い [1]．要素をより精細にする場合，ソリッド要素に比べ節点数を大きく削減できる利点がある．

有限要素法は最終的に，正定値対称な疎行列を係数行列としてもつ連立一次方程式をこと解くことに帰着する．連立一次方程式を解く線形ソルバは，直接法に基づくものと反復法に基づくものの二つに大きく分けられる．

直接法は，Gauss の消去法 (LU 分解と前進後退代入) に基づき，有限回の演算で解を得る堅固な手法である．このとき，LU 分解の過程で元々の行列の値が零である部分に零以外の値が出現する，フィルインが発生することに注意しなければならない．シェル要素を用いた 2 次元問題や薄い形状の 3 次元問題を扱う場合，適切なオーダリングを施すことにより，LU 分解に必要な演算量とメモリ量を抑制することが可能となっている [2]．

反復法は，係数行列に関するベクトル演算を繰り返すことによって，反復的に解を収束させる手法である．求解時のメモリ使用量が少なく，並列計算に適したアルゴリズムであり，適した問題条件の場合，非常に少ない反復回数で収束することができる．しかし，シェル要素から得られる剛性マトリクスは大きな条件数を持つため，求解時に共役勾配法 (Conjugate Gradient Method, CG 法) のような反復法ソルバの収束性が著しく低下する [3]．これは悪条件な問題として位置づけられている (ill-condition)．そのため現状では，シェル要素を用いた解析の多くは直接法ソルバが用いられている．

シェル要素から得られるような悪条件の問題を解くには，直接法ソルバを用いるのが現在の主流であると述べた．しかし，大規模な構造解析では演算時間やメモリ使用量の観点

から，線形ソルバの並列化を考慮しなければならない．

大規模問題を並列直接法ソルバを用いて解く場合，京コンピュータ等の計算資源の増大があっても，計算時間やメモリ使用量等の面から求解が困難である問題に直面している．並列直接法は並列プロセス数が増えると，オーダリングによるフィルインの抑制が難しくなり，演算量とメモリ使用量が増大する．この理由から，計算時間が膨大になり計算機のメモリ環境の限界に達する場合がある．また，LU 分解のアルゴリズムが逐次演算に基づくため，並列計算性能が伸びにくい．

並列反復法ソルバを用いて解く場合，高い並列計算性能を得ることやメモリ使用量を抑えることはできるものの，悪条件な問題では収束までの反復回数が増大するため，現状ではこのような問題を解くことに適しているとは言い難い．反復法ソルバが求解時に必要とするメモリ量は直接法ソルバに比べ少ないため，限られた計算資源でできる限り大きな問題を扱うことを考えると，シェル要素から得られるような悪条件問題を並列反復法ソルバによって高速に解けるようになることは非常に重要である．

反復法では，前処理という，悪条件を改善し収束までの反復回数を削減する手法が用いられる [4]．前処理には様々な手法があるが [5]，それぞれの前処理で前処理行列の生成手法や適用方法が異なっていることや，前処理がいかなる問題にも有効に作用するものではないことに十分留意しなければならない．

悪条件問題に有用な場合があるという点で，前処理行列として係数行列の逆行列を直接近似する，近似逆行列分解系前処理が注目されてきた [6][7][8]．Benzi らは，大きな条件数を持つ連立一次方程式の前処理として，A-直交過程 [9] に基づく近似逆行列分解 (Approximate Inverse，AINV)[6] を，係数行列の正定値性を利用し安定化した，安定化近似逆行列分解 (Stabilized Approximate Inverse，SAINV)[7] を提案した．この前処理手法は，生成する前処理行列のスパース性を保つために，あらかじめ指定した値によって処理中に得られる値の棄却を行う必要がある．さらに Benzi らは，SAINV から得られる分解因子を用いて新たな不完全分解因子を生成する，ロバスト不完全分解 (Robust Incomplete Factorization，RIF)[10] を提案した．池田らは，新たな閾値の導入により前処理性能を改善させた改良型安定化近似逆行列 (Improved SAINV, ISAINV)[11]，改良型ロバスト不完全分解 (Improved RIF, IRIF)[8] を提案した．

しかしこれらの前処理は，大規模問題に適用すると前処理生成に膨大な演算回数が必要

となり，さらに並列計算に対応していない．このため，大きな条件数をもつ大規模問題を並列反復法で解くための解決手法のひとつとして，悪条件問題に有用で高い並列性能が見込める堅固な前処理の開発が急がれている．

高精度な解析の必要性に伴う影響は計算規模の増大の他に，従来に比べより高い計算精度を要求する，高精度数値計算が必要となることが考えられる．高精度数値計算は，高い精度の浮動小数点数演算を用いて桁落ちや情報落ちを抑えることで，従来の低い精度では表現できずに解けなかった問題を解くことができる．しかし，高い精度の浮動小数点数を表現するためには，計算機上で浮動小数点数を表現するビット数を増やす必要がある．ビット数の増加は，演算量とメモリ使用量の増加に直接影響する．例えば，単精度浮動小数点数 (以下単精度) は 32bit，倍精度浮動小数点数 (以下倍精度) は 64bit で表現されるが，CPU によっては 2 倍の演算性能差が発生する．さらに，bit 数が小さい低い精度の演算は，高い精度の演算に比べキャッシュメモリに格納される可能性が高くなるため，キャッシュ効率が向上する．これらを踏まえ，高精度な浮動小数点数を扱う方法として，疑似精度演算と混合精度演算がよく知られている．

疑似精度演算は，二つの低い精度の浮動小数点数で，ひとつの高い精度の浮動小数点数を表し演算する．具体例として，二つの倍精度でひとつの四倍精度を表現する，疑似四倍精度浮動小数点数 (double-double precision, quasi-quadruple precision) が挙げられる．現状では，四倍精度浮動小数点数はハードウェアで広くサポートされておらず，四倍精度演算が可能であっても演算性能は高くない．このため，プログラムの可搬性や演算性能の観点から，ほとんどの四倍精度演算はこの疑似四倍精度演算が利用されている [12][13].

混合精度演算は，解の精度を保ちながら，解に影響しない部分を低い精度の浮動小数点数で計算する方法である．具体例として，低い精度の解法によって得られた解を補正し高い精度の解を保証する，mixed precision iterative refinement が挙げられる [14][15].

さらに近年，省エネルギ性から CPU と GPU といった異なるアーキテクチャのプロセッサを統合して利用する，ヘテロジニアスな計算機環境（以下ヘテロ環境）における並列計算が注目されている [16]．現在広く流通している GPU などのメニーコアアクセラレータのほとんどは，倍精度演算性能に比べて単精度演算性能が 2 倍以上高いため，高精度数値計算への適用を視野に，単精度と倍精度の混合精度演算をヘテロ環境に適用させる意義は大きい．効率のよい演算を実現するために，数値計算のアルゴリズムは，このよう

な高精度数値計算の適用可能性を考慮する必要がある．

1.2 本研究の目的

本研究では，堅固な反復法前処理として注目される RIF 前処理を，領域分割に基づく並列有限要素法に適用する．RIF 前処理は，前処理行列の非零要素プロファイルを前処理生成中に更新して定めるため，大きな計算コストがかかる．そこで，従来の RIF 前処理に対して，係数行列の非零要素プロファイルを元にした，フィルインを考慮せず局所並列化した IRIF(0) 前処理と，フィルインを考慮し局所並列化した IRIF(1) 前処理を提案する．提案手法は，前処理生成より前に前処理行列の非零要素プロファイルを定めることができる点で優れており，これら前処理を用いた並列共役勾配法の有効性について検討する．

また，単精度と倍精度を用いた混合精度演算に着目し，提案手法を単精度を用いて計算する，混合精度演算を用いた前処理つき共役勾配法の有効性について検討する．本研究で提案する前処理生成手法は，係数行列の直交過程を用いて前処理行列の対角項が常に正の値となるように分解を進めるため，低い精度の浮動小数点数を用いた計算でも，安定した前処理を生成できる可能性がある．

さらに，これら提案手法を実問題として設定したエネルギ関連重要施設の構造解析に適用し，実問題における提案手法の有効性について検討する．

1.3 本論文の構成

本論文は，8 章で構成されている．第 1 章では，序論として本研究の背景と目的を述べた．第 2 章では，有限要素法の理論と，数値計算における行列の性質について述べる．第 3 章では，疎行列連立一次方程式を解くための，有限要素法ソルバについて述べる．第 4 章では，本研究で開発基盤として用いた，大規模並列有限要素解析ソフトウェア FrontISTR について述べる．第 5 章では，本復法ソルバの前処理について述べる．第 6 章では，提案手法であるロバスト不完全分解の高速化と並列化について述べる．第 7 章では，提案手法を用いた数値例について述べる．特に，実問題の適用事例として，エネルギ関連重要施設の構造解析についても検証した．第 8 章では，結論として本研究を総括する．

第 2 章　有限要素法

2.1　諸言

この章では，本論文で扱う有限要素法の線形弾性体の境界値問題の理論について述べる．次に，8 節点六面体ソリッド要素と 4 節点平面シェル要素の定式化について述べる．

2.2　線形弾性体の境界値問題の有限要素法

2.2.1　支配方程式

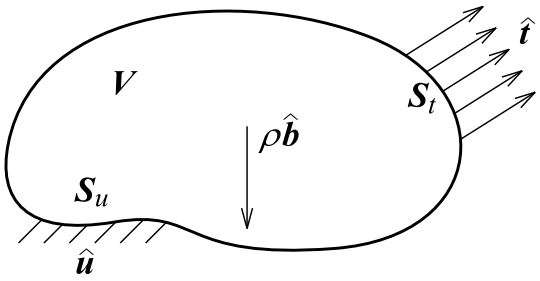


Fig. 1　Boundary value problem of linear elastic material

線形弾性体の境界値問題は，平衡方程式，応力ひずみ関係式，ひずみ変位関係式を支配方程式として，境界条件式のもとに変位 $\boldsymbol{u}$ を求める問題である [17][18]．図 1 に，線形弾性体の境界値問題の概略図を示した．ここで，$\boldsymbol{V}$ は物体の領域，ρ は密度，$\widehat{\boldsymbol{b}}$ は単位質量あたりの体積力ベクトル，$\boldsymbol{S}_u$ は変位境界条件 $\widehat{\boldsymbol{u}}$ が与えられる境界，$\boldsymbol{S}_t$ は応力境界条件 $\widehat{\boldsymbol{t}}$ が与えられる境界，$\widehat{\ }$は既知の量である．

このとき，平衡方程式は式 (1)，変位境界条件式は式 (2)，応力境界条件式は式 (3) で表される．

$$\nabla \cdot \boldsymbol{T} + \rho\widehat{\boldsymbol{b}} = \boldsymbol{0} \tag{1}$$

$$\boldsymbol{u} = \widehat{\boldsymbol{u}} \text{ on } \boldsymbol{S}_u \tag{2}$$

$$\boldsymbol{n} \cdot \boldsymbol{T} = \widehat{\boldsymbol{t}} \text{ on } \boldsymbol{S}_t \tag{3}$$

ここで，$\boldsymbol{T}$ は応力テンソル，$\boldsymbol{n}$ は物体表面における外向き単位法線ベクトルである．

微小変形を考えると応力ひずみ関係式は式 (4)，ひずみ変位関係式は式 (5) で表される．

$$\begin{aligned}\boldsymbol{T} &= \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{\varepsilon}} \\ &= \boldsymbol{C} : \boldsymbol{\varepsilon} \\ &= \lambda \,\mathrm{tr}(\boldsymbol{\varepsilon})\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}\end{aligned} \tag{4}$$

$$\boldsymbol{\varepsilon} = \frac{1}{2}\{\nabla \otimes \boldsymbol{u} + (\nabla \otimes \boldsymbol{u})^t\} \tag{5}$$

ここで，$\boldsymbol{W}$ は弾性ポテンシャル関数，$\boldsymbol{C}$ は 4 階の弾性テンソル，$\boldsymbol{\varepsilon}$ は微小ひずみテンソル，$\boldsymbol{I}$ は恒等テンソルである．上付き添え字 t はテンソルの転置を示す．Lame 定数 λ，μ は，Young 率 E と Poisson 比 ν を用いて，式 (6)，式 (7) で表される．

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{6}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{7}$$

式 (1) と式 (2)，式 (3) からなる線形弾性体の境界値問題は，変位 $\boldsymbol{u}$ が C^2 級であることを必要としており，解に強い制約が施されているため強形式と呼ばれる．

2.2.2 仮想仕事の原理

強形式の境界値問題に対して，弱形式は解の制約を緩めた拡張にあたる．ここではエネルギ最小化問題を経て，境界値問題の弱形式である仮想仕事の原理を求める．

エネルギ最小化問題は，式 (8) と式 (9) を満たす変位 $\boldsymbol{u} \in \boldsymbol{\Phi}$ を求めるものである．ここで，関数空間 $\boldsymbol{\Phi} = \{\boldsymbol{v} | \boldsymbol{v} \in H^1(\boldsymbol{V})^N, \boldsymbol{v} = \widehat{\boldsymbol{u}} \text{ on } \boldsymbol{S}_u\}$，$N$ は次元数である．

$$\Pi(\boldsymbol{u}) \leq \Pi(\boldsymbol{v}) \ \ \forall \boldsymbol{v} \in \boldsymbol{\Phi} \tag{8}$$

$$\Pi(\boldsymbol{v}) = \int_{\boldsymbol{V}} \boldsymbol{W}(\boldsymbol{v}) d\boldsymbol{V} - \int_{\boldsymbol{S}_t} \widehat{\boldsymbol{t}} \cdot \boldsymbol{v} d\boldsymbol{S} - \int_{\boldsymbol{V}} \rho \boldsymbol{b} \cdot \boldsymbol{v} d\boldsymbol{V} \tag{9}$$

このとき任意の $\boldsymbol{v}$ に対して $\boldsymbol{v} = \boldsymbol{u} + \delta\boldsymbol{u}$ となる $\delta\boldsymbol{u} \in \boldsymbol{\Phi}$ が存在するので，Π の変分量，式 (10) を考える．

$$\delta\Pi = \Pi(\boldsymbol{u} + \delta\boldsymbol{u}) - \Pi(\boldsymbol{u}) \tag{10}$$

停留条件 $\delta\Pi = 0$ のもとに, 式 (9) と式 (10) から，式 (11) が得られる．

$$\begin{aligned}\delta\Pi &= \int_{\boldsymbol{V}} \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{\varepsilon}} : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} - \int_{\boldsymbol{S}_t} \widehat{\boldsymbol{t}} \cdot \delta\boldsymbol{u} d\boldsymbol{S} - \int_{\boldsymbol{V}} \rho\widehat{\boldsymbol{b}} \cdot \delta\boldsymbol{u} d\boldsymbol{V} \\ &= \int_{\boldsymbol{V}} \boldsymbol{T} : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} - \int_{\boldsymbol{S}_t} \widehat{\boldsymbol{t}} \cdot \delta\boldsymbol{u} d\boldsymbol{S} - \int_{\boldsymbol{V}} \rho\widehat{\boldsymbol{b}} \cdot \delta\boldsymbol{u} d\boldsymbol{V} \\ &= 0\end{aligned} \tag{11}$$

よって，エネルギ最小化問題と等価な仮想仕事の原理は式 (12) で表される．仮想仕事の原理は仮想変位 $\delta\boldsymbol{u}$ のもとで，仮想内力仕事と仮想外力仕事が等しいことを示している．

$$\int_{\boldsymbol{V}} \boldsymbol{T} : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} = \int_{\boldsymbol{S}_t} \widehat{\boldsymbol{t}} \cdot \delta\boldsymbol{u} d\boldsymbol{S} + \int_{\boldsymbol{V}} \rho\widehat{\boldsymbol{b}} \cdot \delta\boldsymbol{u} d\boldsymbol{V} \tag{12}$$

式 (4) の関係から式 (13) が得られる．

$$\int_{\boldsymbol{V}} (\boldsymbol{C} : \boldsymbol{\varepsilon}) : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} = \int_{\boldsymbol{S}_t} \widehat{\boldsymbol{t}} \cdot \delta\boldsymbol{u} d\boldsymbol{S} + \int_{\boldsymbol{V}} \rho\widehat{\boldsymbol{b}} \cdot \delta\boldsymbol{u} d\boldsymbol{V} \tag{13}$$

ここで，$\delta\boldsymbol{\varepsilon}$ は式 (14) で与えられる．

$$\delta\boldsymbol{\varepsilon} = \frac{1}{2}\{\nabla \otimes \delta\boldsymbol{u} + (\nabla \otimes \delta\boldsymbol{u})^t\} \tag{14}$$

式 (14) を式 (12) の左辺に代入すると，式 (15) が得られる．

$$\begin{aligned}\int_{\boldsymbol{V}} \boldsymbol{T} : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} &= \int_{\boldsymbol{V}} \boldsymbol{T} : \frac{1}{2}\{\nabla \otimes \delta\boldsymbol{u} + (\nabla \otimes \delta\boldsymbol{u})^t\} d\boldsymbol{V} \\ &= \int_{\boldsymbol{V}} \boldsymbol{T} : (\nabla \otimes \delta\boldsymbol{u}) d\boldsymbol{V} \\ &= \int_{\boldsymbol{V}} \nabla \cdot (\boldsymbol{T} \cdot \delta\boldsymbol{u}) d\boldsymbol{V} - \int_{\boldsymbol{V}} (\nabla \cdot \boldsymbol{T}) \cdot \delta\boldsymbol{u} d\boldsymbol{V} \\ &= \oint_{\partial\boldsymbol{V}} (\boldsymbol{n} \cdot \boldsymbol{T}) \cdot \delta\boldsymbol{u} d\boldsymbol{S} - \int_{\boldsymbol{V}} (\nabla \cdot \boldsymbol{T}) \cdot \delta\boldsymbol{u} d\boldsymbol{V} \\ &= \int_{\boldsymbol{S}_t} (\boldsymbol{n} \cdot \boldsymbol{T}) \cdot \delta\boldsymbol{u} d\boldsymbol{S} - \int_{\boldsymbol{V}} (\nabla \cdot \boldsymbol{T}) \cdot \delta\boldsymbol{u} d\boldsymbol{V}\end{aligned} \tag{15}$$

式 (12) の右辺と式 (15) を比べれば，式 (1) と式 (3) に等しくなる．よって，境界値問題とエネルギ最小化問題，仮想仕事の原理が等価であることが示された．

2.2.3 有限要素による離散化と全体剛性マトリクス

弱形式の仮想仕事の原理が得られたことで，解析対象の領域 $\boldsymbol{V}$ を式 (16) のような有限要素 $\boldsymbol{V}^e$ で離散化することができる．

$$\int_{\boldsymbol{V}} d\boldsymbol{V} = \sum_e \int_{\boldsymbol{V}^e} d\boldsymbol{V} \tag{16}$$

式 (16) を用いて式 (13) を書き換えると，式 (17) が得られる．

$$\sum_e \int_{\boldsymbol{V}^e} (\boldsymbol{C} : \boldsymbol{\varepsilon}) : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} = \sum_e \int_{\boldsymbol{S}_t^e} \widehat{\boldsymbol{t}} \cdot \delta\boldsymbol{u} d\boldsymbol{S} + \sum_e \int_{\boldsymbol{V}^e} \rho\widehat{\boldsymbol{b}} \cdot \delta\boldsymbol{u} d\boldsymbol{V} \tag{17}$$

式 (17) を Voigt 表記に書き換えて式 (18) を得る．

$$\sum_e \int_{\boldsymbol{V}^e} (\boldsymbol{B}\delta\boldsymbol{u}^e)^t \boldsymbol{D}(\boldsymbol{B}\boldsymbol{u}^e) d\boldsymbol{V} = \sum_e \int_{\boldsymbol{S}_t^e} (\boldsymbol{N}\delta\boldsymbol{u}^e)^t \widehat{\boldsymbol{t}} d\boldsymbol{S} + \sum_e \int_{\boldsymbol{V}^e} (\boldsymbol{N}\delta\boldsymbol{u}^e)^t \rho\widehat{\boldsymbol{b}} d\boldsymbol{V} \tag{18}$$

ここで，$\boldsymbol{B}$ はひずみ変位マトリクス，$\boldsymbol{D}$ は応力ひずみマトリクス，$\boldsymbol{N}$ は形状関数マトリクス，$\boldsymbol{u}^e$ は要素節点変位ベクトルであり，$\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{u}^e$ の関係をもつ．

式 (18) をまとめると式 (19) を得る．

$$\sum_e (\delta\boldsymbol{u}^e)^t \int_{\boldsymbol{V}^e} \boldsymbol{B}^t \boldsymbol{D}\boldsymbol{B} d\boldsymbol{V} \boldsymbol{u}^e = \sum_e (\delta\boldsymbol{u}^e)^t \left(\int_{\boldsymbol{S}_t^e} \boldsymbol{N}^t \widehat{\boldsymbol{t}} d\boldsymbol{S} + \int_{\boldsymbol{V}^e} \boldsymbol{N}^t \rho\widehat{\boldsymbol{b}} d\boldsymbol{V} \right) \tag{19}$$

要素剛性マトリクス $\boldsymbol{K}^e$ を式 (20) とおく．

$$\boldsymbol{K}^e = \int_{\boldsymbol{V}^e} \boldsymbol{B}^t \boldsymbol{D}\boldsymbol{B} d\boldsymbol{V} \tag{20}$$

要素外力ベクトル $\boldsymbol{f}^e$ を式 (21) とおく．

$$\boldsymbol{f}^e = \int_{\boldsymbol{S}_t^e} \boldsymbol{N}^t \widehat{\boldsymbol{t}} d\boldsymbol{S} + \int_{\boldsymbol{V}^e} \boldsymbol{N}^t \rho\widehat{\boldsymbol{b}} d\boldsymbol{V} \tag{21}$$

式 (19) を，式 (20) と式 (21) で書き換えて式 (22) を得る．

$$\sum_e (\delta\boldsymbol{u}^e)^t \boldsymbol{K}^e \boldsymbol{u}^e = \sum_e (\delta\boldsymbol{u}^e)^t \boldsymbol{f}^e \tag{22}$$

全体剛性マトリクスを $\boldsymbol{K}$，全体節点変位ベクトルを $\boldsymbol{u}$，全体外力ベクトルを $\boldsymbol{f}$ とすると式 (23) を得る．

$$(\delta\boldsymbol{u})^t \boldsymbol{K}\boldsymbol{u} = (\delta\boldsymbol{u})^t \boldsymbol{f} \tag{23}$$

式 (23) より，有限要素法による離散化式 (24) を得る．

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \tag{24}$$

2.3 要素

2.3.1 ソリッド要素

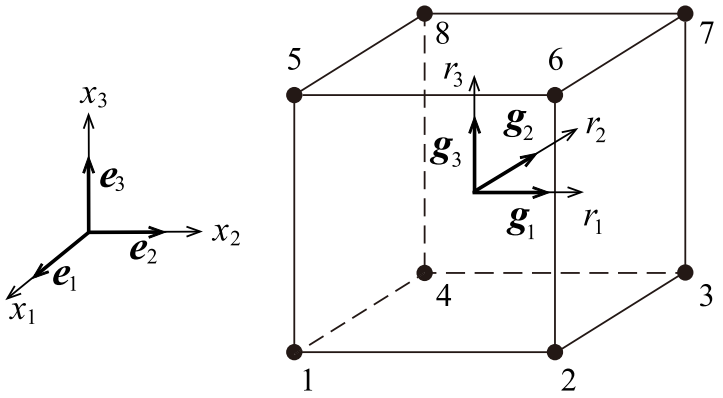


Fig. 2 8 node hexahedral solid element

ソリッド要素は，離散化を行う場合に用いる要素の種類のひとつで，立体形状のものをいう．ソリッド要素には様々あるが，ここでは図 2 のような 3 次元 8 節点六面体要素を示す [18]．ここで $\boldsymbol{e}_1$，$\boldsymbol{e}_2$，$\boldsymbol{e}_3$ は全体座標系 x_1，x_2，x_3 の直交基底ベクトル，$\boldsymbol{g}_1$，$\boldsymbol{g}_2$，$\boldsymbol{g}_3$ は局所座標系 r_1，r_2，r_3 の共変基底ベクトルである．8 節点六面体ソリッド要素では，全体座標系 $\boldsymbol{x}$ 内の六面体から局所座標系 $\boldsymbol{r}$ 内への立方体へ写像し，形状関数 $\boldsymbol{N}$ によって各々の要素内を補完する．形状関数 $\boldsymbol{N}$ は式 (25) で表される．

$$
\begin{aligned}
N^{(1)} &= \frac{1}{8}(1-r_1)(1-r_2)(1-r_3) \\
N^{(2)} &= \frac{1}{8}(1+r_1)(1-r_2)(1-r_3) \\
N^{(3)} &= \frac{1}{8}(1+r_1)(1+r_2)(1-r_3) \\
N^{(4)} &= \frac{1}{8}(1-r_1)(1+r_2)(1-r_3) \\
N^{(5)} &= \frac{1}{8}(1-r_1)(1-r_2)(1+r_3) \\
N^{(6)} &= \frac{1}{8}(1+r_1)(1-r_2)(1+r_3) \\
N^{(7)} &= \frac{1}{8}(1+r_1)(1+r_2)(1+r_3) \\
N^{(8)} &= \frac{1}{8}(1-r_1)(1+r_2)(1+r_3)
\end{aligned}
\tag{25}
$$

共変基底ベクトル $\boldsymbol{g}$ は式 (26) で表される．

$$\begin{aligned} \boldsymbol{g}_1 &= \frac{\partial \boldsymbol{x}}{\partial r_1} \\ \boldsymbol{g}_2 &= \frac{\partial \boldsymbol{x}}{\partial r_2} \\ \boldsymbol{g}_3 &= \frac{\partial \boldsymbol{x}}{\partial r_3} \end{aligned} \tag{26}$$

ひずみ変位マトリクス $\boldsymbol{B}$ は，$\boldsymbol{B}^{(\alpha)}$ を横に並べたもので，式 (27) で表される．

$$\boldsymbol{B} = (\boldsymbol{B}^{(1)}\ \boldsymbol{B}^{(2)}\ \cdots\ \boldsymbol{B}^{(\alpha)}\ \cdots\ \boldsymbol{B}^{(8)}) \tag{27}$$

ここで $\boldsymbol{B}^{(\alpha)}$ は式 (28) である．

$$\boldsymbol{B}^{(\alpha)} = \begin{pmatrix} \dfrac{\partial N^{(\alpha)}}{\partial x} & 0 & 0 \\ 0 & \dfrac{\partial N^{(\alpha)}}{\partial y} & 0 \\ 0 & 0 & \dfrac{\partial N^{(\alpha)}}{\partial z} \\ \dfrac{\partial N^{(\alpha)}}{\partial y} & \dfrac{\partial N^{(\alpha)}}{\partial x} & 0 \\ 0 & \dfrac{\partial N^{(\alpha)}}{\partial z} & \dfrac{\partial N^{(\alpha)}}{\partial y} \\ \dfrac{\partial N^{(\alpha)}}{\partial z} & 0 & \dfrac{\partial N^{(\alpha)}}{\partial x} \end{pmatrix} \tag{28}$$

応力ひずみマトリクス $\boldsymbol{D}$ は式 (29) で表される．

$$\boldsymbol{D} = \begin{pmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix} \tag{29}$$

式 (27) と式 (29) より，8 節点六面体ソリッド要素の要素剛性マトリクスが生成できる．

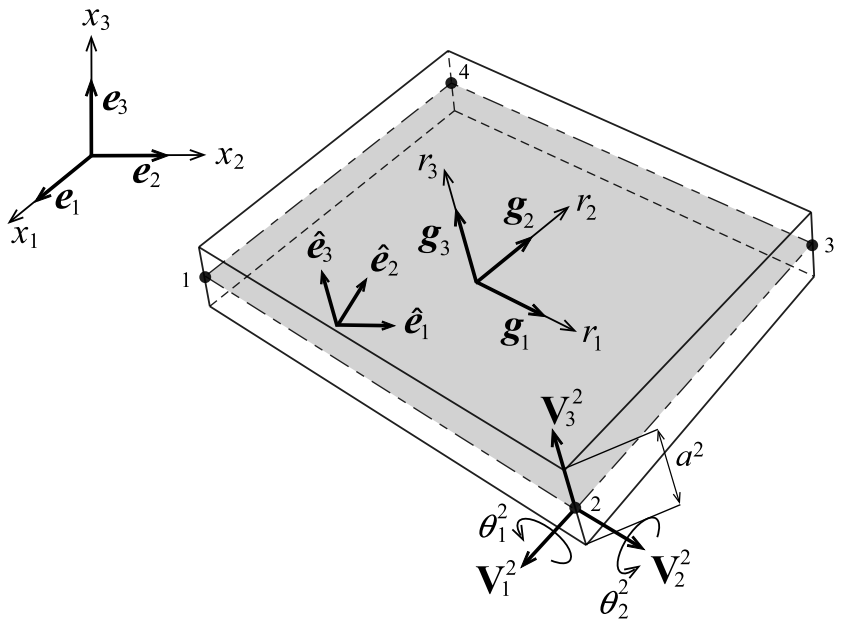


Fig. 3　4 node degenerated shell element

2.3.2　シェル要素

シェル要素は，ソリッド要素同様に要素の種類のひとつで，板形状のものをいう．ここではシェル要素で代表的な要素のひとつである，3 次元 4 節点四角形 MITC シェル要素について示す [19]．MITC（Mixed Interpolation of Tensorial Components）シェル要素は，degenerate シェル要素 [20] のシェアロッキングを防ぐために Assumed Strain を考え再補完を行った要素である．

図 3 に degenerate シェル要素を示す．ここで $\boldsymbol{e}$ は直交基底ベクトル，$\hat{e}$ は単位直交ベクトル，$\boldsymbol{V}$ は任意の節点での単位直交ベクトル，$\boldsymbol{\theta}$ は任意の節点での軸性ベクトル，α^k は板厚，上付き添字 k は任意の節点を示す．特に $\boldsymbol{V}_3$ は director ベクトルと呼ばれる．

ここで共変基底ベクトル $\boldsymbol{g}_1$，$\boldsymbol{g}_2$，$\boldsymbol{g}_3$ は式 (30) のように定義する．

$$
\begin{aligned}
\boldsymbol{g}_1 &= \frac{\partial \boldsymbol{x}}{\partial r_1} \\
\boldsymbol{g}_2 &= \frac{\partial \boldsymbol{x}}{\partial r_2} \\
\boldsymbol{g}_3 &= \frac{\partial \boldsymbol{x}}{\partial r_3} \times \boldsymbol{g}_2
\end{aligned}
\tag{30}
$$

式 (30) から，単位直行ベクトル $\hat{\boldsymbol{e}}_1$，$\hat{\boldsymbol{e}}_2$，$\hat{\boldsymbol{e}}_3$ を式 (31) のように定義する．

$$\begin{aligned}\hat{\boldsymbol{e}}_3 &= \frac{\boldsymbol{g}_3}{|\boldsymbol{g}_3|} \\ \hat{\boldsymbol{e}}_1 &= \frac{\boldsymbol{g}_2 \times \hat{\boldsymbol{e}}_3}{|\boldsymbol{g}_2 \times \hat{\boldsymbol{e}}_3|} \\ \hat{\boldsymbol{e}}_2 &= \hat{\boldsymbol{e}}_3 \times \hat{\boldsymbol{e}}_1\end{aligned} \tag{31}$$

さらに式 (31) から，単位直行ベクトル $\boldsymbol{V}_1^k$，$\boldsymbol{V}_2^k$，$\boldsymbol{V}_3^k$ を式 (32) のように定義する．

$$\begin{aligned}\boldsymbol{V}_3^k &= \hat{\boldsymbol{e}_3} \\ \boldsymbol{V}_2^k &= \frac{\hat{\boldsymbol{e}}_3 \times \boldsymbol{e}_1}{|\hat{\boldsymbol{e}}_3 \times \boldsymbol{e}_1|} \\ \boldsymbol{V}_1^k &= \boldsymbol{V}_2^k \times \boldsymbol{V}_3^k\end{aligned} \tag{32}$$

要素内の任意の点における位置ベクトル $\boldsymbol{x}$ は，式 (33) で表される．

$$\boldsymbol{x} = \sum_{k=1}^{4} N^{(k)} \boldsymbol{x}^k + \frac{r_3}{2} \sum_{k=1}^{4} N^{(k)} \alpha^k \boldsymbol{V}_3^k \tag{33}$$

ここで形状関数 $\boldsymbol{N}$ は，式 (34) で表される．

$$\begin{aligned}N^{(1)} &= \frac{1}{4}(1-r_1)(1-r_2) \\ N^{(2)} &= \frac{1}{4}(1+r_1)(1-r_2) \\ N^{(3)} &= \frac{1}{4}(1-r_1)(1+r_2) \\ N^{(4)} &= \frac{1}{4}(1+r_1)(1+r_2)\end{aligned} \tag{34}$$

式 (33) を用いて，基準時刻 t_1 から微小時間後の t_2 における変位ベクトル $\boldsymbol{u}$ を式 (35) に示す．

$$\boldsymbol{u} = {}^{t_2}\boldsymbol{x} - {}^{t_1}\boldsymbol{x} = \sum_{k=1}^{4} N^{(k)} \left({}^{t_2}\boldsymbol{x}^k - {}^{t_1}\boldsymbol{x}^k\right) + \frac{r_3}{2} \sum_{k=1}^{4} N^{(k)} \alpha^k \left({}^{t_2}\boldsymbol{V}_3^k - {}^{t_1}\boldsymbol{V}_3^k\right) \tag{35}$$

${}^{t_2}\boldsymbol{x} - {}^{t_1}\boldsymbol{x}$ は節点変位，${}^{t_2}\boldsymbol{V}_3^k - {}^{t_1}\boldsymbol{V}_3^k$ は director ベクトル $\boldsymbol{V}_3^k$ の回転を表している．

このとき微小回転を仮定すると，図 4 に示すように，director ベクトル $\boldsymbol{V}_3^k$ の回転は軸性ベクトル $\boldsymbol{\theta}$ によって，式 (36) で近似される．

$${}^{t_2}\boldsymbol{V}_3^k \cong {}^{t_1}\boldsymbol{V}_3^k + \boldsymbol{\theta} \times {}^{t_1}\boldsymbol{V}_3^k \tag{36}$$

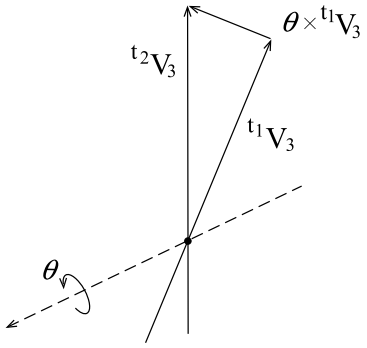


Fig. 4　Rotation of director vector within infinitesimal time

式 (**??**) を用いて，軸性ベクトル $\boldsymbol{\theta}$ を式 (37) のように成分分解する．

$$\boldsymbol{\theta}^k = \theta_1^k \boldsymbol{V}_1^k + \theta_2^k \boldsymbol{V}_2^k \tag{37}$$

式 (37) を式 (36) に代入して式 (38) を得る．

$${}^{t_2}\boldsymbol{V}_3^k \cong {}^{t_1}\boldsymbol{V}_3^k - \theta_1^k \boldsymbol{V}_2^k + \theta_2^k \boldsymbol{V}_1^k \tag{38}$$

式 (38) を式 (35) に代入して式 (39) を得る．

$$\boldsymbol{u} = {}^{t_2}\boldsymbol{x} - {}^{t_1}\boldsymbol{x} = \sum_{k=1}^{4} N^{(k)}({}^{t_2}\boldsymbol{x}^k - {}^{t_1}\boldsymbol{x}^k) + \frac{r_3}{2}\sum_{k=1}^{4} N^{(k)}\alpha^k(-\theta_1^k \boldsymbol{V}_2^k + \theta_2^k \boldsymbol{V}_1^k) \tag{39}$$

式 (39) をまとめて，式 (40) を得る．

$$\boldsymbol{u} = \sum_{k=1}^{4} N^{(k)}\{\boldsymbol{u}^k + \frac{r_3}{2}\alpha^k(\boldsymbol{\theta}^k \times \boldsymbol{V}_3^k)\} \tag{40}$$

弾性テンソル $\boldsymbol{C}^{ijkl}$ は，単位直交ベクトル $\hat{\boldsymbol{e}}$ と反変基底ベクトル $\boldsymbol{g}^i$ から，式 (41) のように得られる．

$$\boldsymbol{C}^{ijkl} = \hat{\boldsymbol{C}}^{mnop}(\hat{\boldsymbol{e}}_m \cdot \boldsymbol{g}^i)(\hat{\boldsymbol{e}}_n \cdot \boldsymbol{g}^j)(\hat{\boldsymbol{e}}_o \cdot \boldsymbol{g}^k)(\hat{\boldsymbol{e}}_p \cdot \boldsymbol{g}^l) \tag{41}$$

degenerated シェル要素は，面外せん断ひずみによってシェアロッキングを発生する場合がある．MITC シェル要素では，この現象を回避するために，図 5 に示すのように，A 点 B 点 C 点 D 点から補完される Assumed Strain による，面外せん断ひずみ成分についての内挿式 (42) を得る．図 5 では，ε_{13}^A について示した．

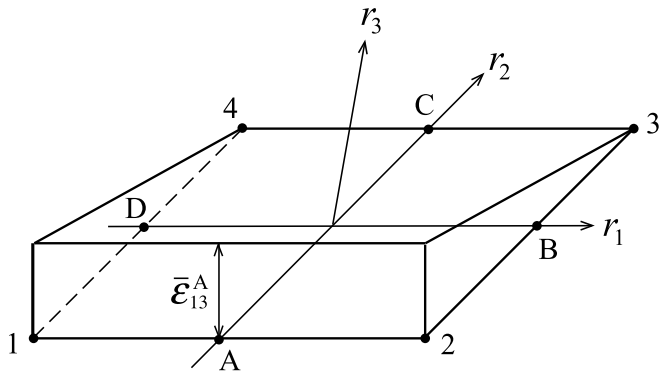


Fig. 5 Interpolation functions for the transverse shear strains

$$\varepsilon_{13} = \varepsilon_{31} = \frac{1}{2}(1 - r_2)\varepsilon^A_{13} + \frac{1}{2}(1 + r_2)\varepsilon^C_{13}$$
$$\varepsilon_{23} = \varepsilon_{32} = \frac{1}{2}(1 + r_1)\varepsilon^B_{13} + \frac{1}{2}(1 - r_2)\varepsilon^D_{13} \tag{42}$$

MITC シェル要素による定式化は，1 節点あたり 5 自由度（並進 3 自由度，回転 2 自由度）で定義するものである．5 自由度の要素は，他の要素を混在させて解析する場合など，プログラミングによる実装が複雑になる．この解決のために，式 (43) のように板厚方向を示す director ベクトルに対し drilling 自由度を考慮する [21][22][23].

$$\boldsymbol{u}^k = u^k_1\boldsymbol{e}_1 + u^k_2\boldsymbol{e}_2 + u^k_3\boldsymbol{e}_3$$
$$\boldsymbol{\theta}^k = \theta^k_1\boldsymbol{e}_1 + \theta^k_2\boldsymbol{e}_2 + \theta^k_3\boldsymbol{e}_3 \tag{43}$$

drilling 自由度を考慮した場合，解くべき仮想仕事の原理は式 (44) となる．

$$\int_{\boldsymbol{V}} \boldsymbol{T} : \delta\boldsymbol{\varepsilon} d\boldsymbol{V} + \int_{\boldsymbol{V}} \alpha\{\boldsymbol{V}_3 \cdot (\boldsymbol{\theta} - \frac{1}{2}\boldsymbol{e} : \boldsymbol{\Theta})\}\{\boldsymbol{V}_3 \cdot (\delta\boldsymbol{\theta} - \frac{1}{2}\boldsymbol{e} : \delta\boldsymbol{\Theta})\} d\boldsymbol{V}$$
$$= \int_{\boldsymbol{S}_t} \widehat{\boldsymbol{t}} \cdot \delta\boldsymbol{u} d\boldsymbol{S} + \int_{\boldsymbol{V}} \rho\widehat{\boldsymbol{b}} \cdot \delta\boldsymbol{u} d\boldsymbol{V} \tag{44}$$

ここで，α はペナルティ係数，$\boldsymbol{\Theta}$ は反対称テンソルである．

これら関係から MITC シェル要素の要素剛性マトリクスを生成する．

2.4 結言

この章では，本論文で扱う有限要素法の線形弾性体の境界値問題の理論について述べた．さらに，8 節点六面体ソリッド要素と 4 節点平面シェル要素の定式化について述べた．

第 3 章　線形ソルバ

3.1　諸言

この章では，有限要素法の離散化から得られる連立方程式を解くソルバについて述べる．前章までに，有限要素法を用いた離散化から，仮想仕事の原理の離散化式を示した．有限要素法は最終的に，大規模な疎行列を係数行列としてもつ連立一次方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ を解くことに帰着する．連立一次方程式を解く線形ソルバは，直接法に基づくものと反復法に基づくものの二つに大きく分けられる．まず，直接法ソルバと反復法ソルバについて述べる．次に，線形ソルバで解く全体剛性マトリクスの性質について述べる．最後に，条件数が共役勾配法の反復回数に与える影響について述べる．

3.2　直接法

3.2.1　直接法の概要

直接法は，Gauss の消去法 (LU 分解と前進後退代入) に基づいており，有限回の演算で解を得る堅固な手法である．直接法の特徴は，LU 分解の過程において元々の行列で要素の値が零である部分に零以外の値が出現する，フィルインが発生することである．大規模疎行列を求解するときは，フィルインの数が元の行列の非零要素数に比べて非常に大きくなり，計算量とメモリ使用量が増大する．フィルインの増大を抑制させるため，適当な置換行列を用いて行と列を置換するオーダリングを行う必要がある [24].

よく用いられるオーダリングの代表例としては，Minimum Degree 法と Nested Dissection 法が挙げられる [3]．Minimum Degree 法は，LU 分解の各ステップで生じるフィルインを評価値として，フィルインが少なくなるように置換行列を決定する greedy algorithm の一種である [25]．Nested Dissection 法は，ひとつの領域を二つの互いに独立した領域とその間を結ぶセパレータ領域に分割する手法である [26]．二つの互いに独立した領域は，各々再帰的に分割できる．セパレータ領域の大きさは通信量に強く関係するため，独立した領域に含まれる要素数が等しくなり，かつ，セパレータ領域をできるだけ小さくするように分割を行うことで，並列計算に適したオーダリングが可能な手法として注目されている．

2 次元問題や薄い構造物のような 3 次元問題を扱う場合，適切なオーダリングを施すことによって，LU 分解に必要な演算量とメモリ量を抑制することが可能となっている [2]．このため現状では，シェル要素を用いた解析の多くは直接法ソルバが用いられている．

並列計算を行う場合，LU 分解は逐次分解を行う性質上，アルゴリズムの並列化が難しいため並列計算性能が得にくい．また，領域分割に基づいた並列有限要素法のソルバとして用いる場合，領域分割の影響でフィルインを抑制することが難しくなり，この影響からも通信量とメモリ使用量が増大する．

直接法は，計算時に多くのメモリ使用量を必要とすることや並列化性能が得にくいことを述べたが，有限回の演算で解を得ることのできる堅固な手法であり，直接法を用いて解ける問題ならば，一般的に反復法に比べて計算時間は短くなる．

現在広く知られている疎行列直接法ソルバのライブラリとして，MUMPS[27]，PARADISO[28]，WSMP[29] などが挙げられる．

3.3 反復法と反復法前処理

3.3.1 反復法の概要

反復法は，係数行列に関するベクトル演算を繰り返すことによって，反復的に解を収束させる手法である．反復法の特徴は，解く問題の条件が適していると非常に少ない反復回数で収束することである．さらに，大規模疎行列を求解するときにメモリ使用量が少ない．並列計算を行う場合，ベクトル演算に関して並列処理を適用できるため，大規模問題に関しては反復法が用いられる．

3.3.2 共役勾配法

本研究では，Krylov 部分空間による反復法として広く知られている共役勾配法を用いる [3]．共役勾配法は，正定値性対称行列を係数行列にもつ連立一次方程式 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ を反復的に解くことの他に，式 (45) のような関数の最小化問題を解くと考えることもできる．

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} \tag{45}$$

図 6 に，共役勾配法を示した．ここで，$\boldsymbol{A}$ は $n \times n$ 次の係数行列，$\boldsymbol{b}$ は n 次の右辺ベクトル，$\boldsymbol{r}$ は残差ベクトル，$\boldsymbol{p}$ は探索方向ベクトル，ε は収束判定値である．

1: $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$
2: **for** $m = 1, 2, 3, \ldots,$ **do**
3: $\quad \alpha_m = \dfrac{(\boldsymbol{r}_{m-1}, \boldsymbol{r}_{m-1})}{(\boldsymbol{p}_{m-1}, \boldsymbol{A}\boldsymbol{p}_{m-1})}$
4: $\quad \boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \alpha_m \boldsymbol{p}_{m-1}$
5: $\quad \boldsymbol{r}_m = \boldsymbol{r}_{m-1} - \alpha_m \boldsymbol{A}\boldsymbol{p}_{m-1}$
6: $\quad$ **if** $(\|\boldsymbol{r}_m\|_2 / \|\boldsymbol{r}_0\|_2 \leqq \varepsilon)$ **then stop**
7: $\quad \beta_m = \dfrac{(\boldsymbol{r}_m, \boldsymbol{r}_m)}{(\boldsymbol{r}_{m-1}, \boldsymbol{r}_{m-1})}$
8: $\quad \boldsymbol{p}_m = \boldsymbol{r}_m + \beta_m \boldsymbol{p}_{m-1}$
9: **end for**

Fig. 6 Conjugate gradient method

共役勾配法の特徴は，1 反復あたりの演算量とメモリ使用量が少ないこと，解くべき行列の次元数 N に対して理論上 N 回反復すれば収束することである．共役勾配法の収束のしやすさは，最大固有値と最小固有値の比で表される条件数 (Comdition number) を指標にすることができる．条件数の推定については 3.4 章で詳しく述べる．

3.3.3 前処理つき共役勾配法

共役勾配法は，解くべき行列の条件数が増大すると収束に必要な反復回数も増大することが知られている．そのため，係数行列の条件数を低減させることを目的として，元の方程式 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ を解くかわりに，式 (46) のような方程式を解くことを考える [3]．

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b} \tag{46}$$

ここで，$\boldsymbol{M}$ は前処理行列である．この方程式 (46) に共役勾配法を適用したものが，前処理つき共役勾配法である．図 7 に，前処理つき共役勾配法を示した．

前処理行列 $\boldsymbol{M}$ は，$\boldsymbol{M} \simeq \boldsymbol{A}$ であるほど前処理の効果が大きくなるが，$\boldsymbol{M}^{-1}$ を計算する必要があることから計算量は増大する．そのため，前処理生成時間と，前処理行列によって短縮された反復回数分の計算時間のトレードオフとなる．例えば，前処理行列 $\boldsymbol{M} = \boldsymbol{A}$ として $\boldsymbol{M}^{-1}$ を完全に解けば 1 回の反復で解を得られるが，直接法を用いて解を得ることと同等の演算量とメモリ使用量が必要になる．

1: $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0$
2: **for** $m = 1, 2, 3, \ldots,$ **do**
3: $\quad \alpha_m = \dfrac{(\boldsymbol{r}_{m-1}, \boldsymbol{M}^{-1}\boldsymbol{r}_{m-1})}{(\boldsymbol{p}_{m-1}, \boldsymbol{A}\boldsymbol{p}_{m-1})}$
4: $\quad \boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \alpha_m \boldsymbol{p}_{m-1}$
5: $\quad \boldsymbol{r}_m = \boldsymbol{r}_{m-1} - \alpha_m \boldsymbol{A}\boldsymbol{p}_{m-1}$
6: $\quad$ **if** $(\|\boldsymbol{r}_m\|_2 / \|\boldsymbol{r}_0\|_2 \leqq \varepsilon)$ **then stop**
7: $\quad \beta_m = \dfrac{(\boldsymbol{r}_m, \boldsymbol{M}^{-1}\boldsymbol{r}_m)}{(\boldsymbol{r}_{m-1}, \boldsymbol{M}^{-1}r_{m-1})}$
8: $\quad \boldsymbol{p}_m = \boldsymbol{M}^{-1}\boldsymbol{r}_m + \beta_m \boldsymbol{p}_{m-1}$
9: **end for**

Fig. 7 Preconditioned conjugate gradient method

3.4 全体剛性マトリクスの条件数推定

全体剛性マトリクスの性質を評価する指標のひとつとして，共役勾配法の収束のしやすさに影響する条件数の推定は重要である．条件数は，式 (47) で定義される．

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\| \tag{47}$$

このときノルムの定義は任意である．特に，ノルムの定義が 2 ノルムで，行列 $\boldsymbol{A}$ が正定値対象であれば，条件数は式 (48) のように，最大固有値と最小固有値の比となる．

$$\kappa(\boldsymbol{A}) = \frac{\lambda_{max}(\boldsymbol{A})}{\lambda_{min}(\boldsymbol{A})} \tag{48}$$

これまで，条件数の推定方法はいくつか提案されているが，本研究では大規模問題でも推定が可能である共役勾配法を利用した方法を用いる [3]．共役勾配法は，式 (49) に示すように反復中から得られる係数 α，β を用いて，Lanczos 法の第 m 回の反復までに得られる三重対角行列に対応する行列 $\boldsymbol{T}_m$ を生成することができる．この三重対角行列 $\boldsymbol{T}_m$ の最小固有値と最大固有値を求めることで，条件数の推定を行う．

三重対角行列の固有値計算は，QR 法などを用いて解かれる．本研究では，数値計算ライブラリ LAPACK[30] のルーチン DSTEQR を用いて QR 法による固有値計算を行った．

$$\boldsymbol{T}_m = \begin{pmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{\alpha_0} & & & & & \mathbf{0} \\ \frac{\sqrt{\beta_0}}{\alpha_0} & \frac{\beta_0}{\alpha_0}+\frac{1}{\alpha_1} & \frac{\sqrt{\beta_1}}{\alpha_1} & & & & \\ & \frac{\sqrt{\beta_1}}{\alpha_1} & \frac{\beta_1}{\alpha_1}+\frac{1}{\alpha_2} & \frac{\sqrt{\beta_2}}{\alpha_2} & & & \\ & & & \ddots & & & \\ & & & \frac{\sqrt{\beta_{m-3}}}{\alpha_{m-3}} & \frac{\beta_{m-3}}{\alpha_{m-3}}+\frac{1}{\alpha_{m-2}} & \frac{\sqrt{\beta_{m-2}}}{\alpha_{m-2}} & \\ \mathbf{0} & & & & \frac{\sqrt{\beta_{m-2}}}{\alpha_{m-2}} & \frac{\beta_{m-2}}{\alpha_{m-2}}+\frac{1}{\alpha_{m-1}} & \end{pmatrix} \tag{49}$$

この方法は共役勾配法の係数を用いるため，収束自体が不十分な場合には，最小固有値を大きく推定することになる．最小固有値を大きく推定することは，条件数を小さく推定することになるため，推定には十分な反復を行うことに注意する．

前処理付共役勾配法に対して，式 (49) の行列 $\boldsymbol{T}_m$ から条件数推定を行うと，前処理行列を適用した後の行列 $\boldsymbol{M}^{-1}\boldsymbol{A}$ の条件数を推定することができる．

表 1 に，数値例として，条件数の推定値と収束に必要な共役勾配法の反復回数を示す．図 8 に，解析モデル ESTCOND-SOLID と ESTCOND-SHELL を示す．ESTCOND-SOLID と ESTCOND-SHELL のモデルは，縦 100mm × 横 100mm × 厚さ t mm であり，厚さ t を 10mm，1mm，0.1mm と変化させた．要素数は縦 10 要素 × 横 10 要素 × 厚さ方向 1 要素，ヤング率 $E = 2.0 \times 10^5$，ポアソン比 $\nu = 0.3$，収束判定値 $\varepsilon = 1.0 \times 10^{-8}$ であり，シェル要素は MITC4[19] を使用した．厚さ t を薄くしていくと，物理的には面内方向の振動数と面外方向の振動数の比が大きくなり，条件数が増大する．つまり，薄いソリッド要素やシェル要素を用いて，薄い板形状を有する構造解析を反復法で解く場合，条件数が増大の影響によって，収束に必要な反復回数は増加する．

3.5 結言

この章では，有限要素法の離散化から得られる連立方程式を解くソルバに関し，直接法ソルバと反復法ソルバについて述べた．直接法は有限回の演算で解を得ることのできる堅固な手法であり，反復法はメモリ使用量が少なく，並列計算が比較的容易な手法であることが特徴である．次に，線形ソルバで解く全体剛性マトリクスの性質について述べた．最後に，条件数が共役勾配法の反復回数に与える影響について述べた．

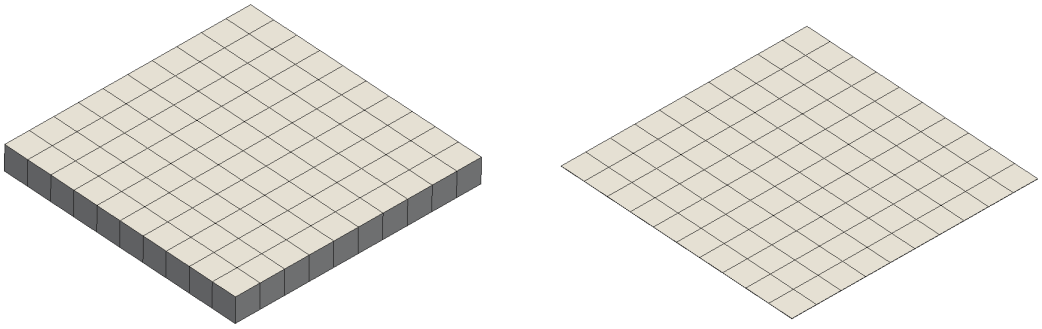


Fig. 8　Analysis mesh of ESTCOND-SOLID and ESTCOND-SHELL

Table 1　Estimated condition number and iteration of CG method

Matrix	Thickness t	Itetation	Condition Number
ESTCOND-SOLID	10	140	6.46E+04
	1	5691	6.20E+08
	0.1	173564	6.11E+12
ESTCOND-SHELL	10	281	4.41E+05
	1	1381	3.71E+07
	0.1	11533	3.71E+09

第 4 章　大規模並列有限要素解析ソフトウェア FrontISTR

4.1　諸言

この章では，本研究で開発基盤として用いた，大規模並列有限要素解析ソフトウェア FrontISTR とそのミドルウェア HEC-MW について述べる．次に，並列計算に必要な行列ベクトル積とベクトル内積の並列化に関して述べる．前処理の並列計算については，次章以降で述べる．

4.2　HEC-MW と FrontISTR

4.2.1　HEC-MW と FrontISTR の概要

HEC-MW(High End Computing Middleware) は，要素メッシュ情報やファイル入出力，線形ソルバ，可視化などの大規模並列有限要素解析のための基盤となる機能を提供するミドルウェアである．FrontISTR は HEC-MW 上で構築された大規模並列有限要素解析ソフトウェアであり [31][32]，HEC-MW の機能によって，並列計算を行う場合に必要な計算領域の分割やその領域間の通信を意識しないプログラム構造になっている．

本研究では，並列計算を簡便に実現できることや提案手法の実装が比較的容易なことから，FrontISTR を提案手法の開発基盤として利用した．

HEC-MW と FrontISTR は，東京大学生産技術研究所革新的シミュレーション研究センターにおける文部科学省次世代 IT 基盤構築のための研究開発「イノベーション基盤シミュレーションソフトウェアの研究開発」プロジェクト等において開発された．

4.2.2　領域分割とパーティショナ

有限要素法の並列化を行うために，FrontISTR では要素情報に基づく領域分割に基づいた有限要素法を採用し，並列計算を実現している．このとき並列計算領域は，あらかじめ元の一領域メッシュから，HEC-MW で提供されるパーティショナを用いて分割される．パーティショナは，一領域メッシュの節点番号と要素情報から得られる情報を元にしたグラフ構造を用いて並列計算領域を決定する．パーティショナのグラフ分割は，外部のグラフライブラリを参照し，利用できるようになっており，本研究においては Metis[33]

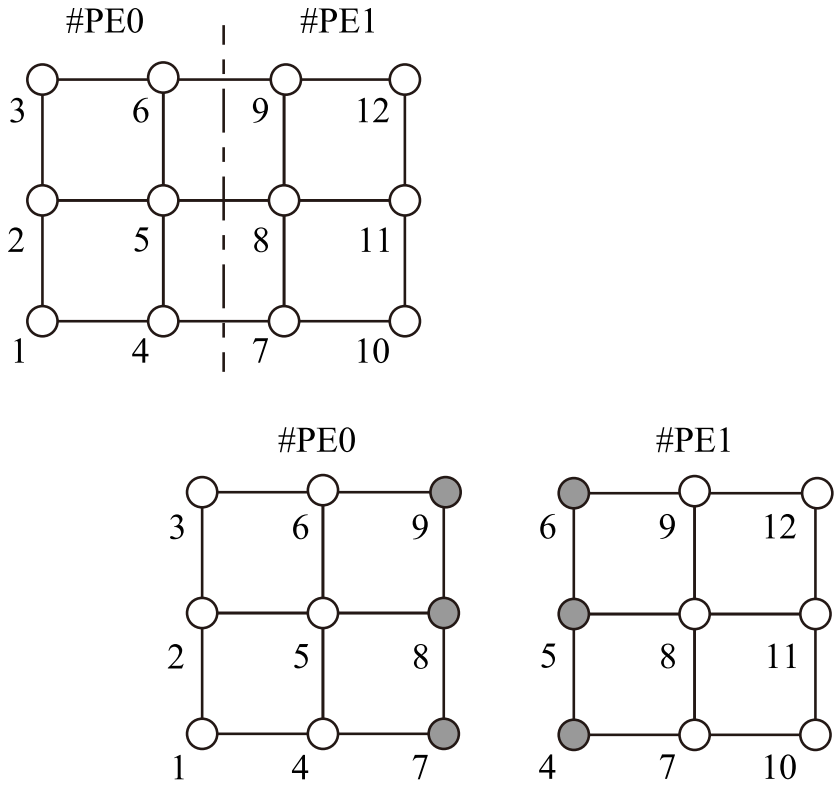


Fig. 9 Schematic representation of Node-based domain decomposition into 2PEs

を利用した．このとき，各領域の節点数がほぼ均等となるように領域を分割することで，並列計算時に計算負荷が均等になり，高い並列計算性能を得ることができる．

図 9 に，要素情報に基づく領域分割の概要を示す．ここでは上段に示した非構造格子を，下段のように 2 つの領域の非構造格子に分割している．濃く示した節点は外点と呼ばれ，並列計算において，各領域は外点を通じて他領域とデータ通信を行う．外点の数はデータ通信が必要な節点数と対応しているため，通信が必要な節点数をできるだけ少なくすることは，並列計算効率を高める上で重要である．

4.2.3 有限要素のリファイン

高精度な解析のために，精細な解析メッシュを作成する必要がある．並列計算が必要な大規模解析メッシュをより精細なものにすると，データ容量が莫大になりひとつの計算機上ではデータを保持できない状況が見受けられるようになった．この解決のため，並列計算のために領域分割されたメッシュを分散メモリ環境に分配した後，それぞれの環境で個

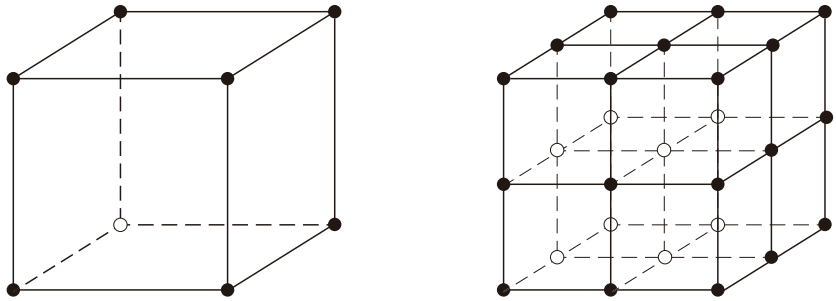


Fig. 10　Schematic representation of refinement of solid element

Fig. 11　Schematic representation of refinement of shell element

別にメッシュを微細化するリファインという手法がとられる．並列計算でなくとも，精細な解析メッシュの生成が容易になるため，リファインの利用価値は高い．

HEC-MW では，既存のメッシュを要素情報に基づいてリファインするライブラリである，リファイナが提供されている．図 10 と図 11 に，HEC-MW のリファイナの概要について，8 節点六面体ソリッド要素と 4 節点四角形シェル要素を例に挙げて示す．

図 10 左のようなソリッド要素を 1 回リファインすると，図 10 右のように，リファインされたソリッド要素が得られる．要素数は，1 要素から 8 要素に増加する．図 11 左のようなシェル要素を 1 回リファインすると，図 11 右のように，リファインされたシェル要素が得られる．要素数は，1 要素から 4 要素に増加する．このような要素の精密化を再帰的に繰り返すことで，ひとつの計算機上では保持できないような巨大な要素情報も，分散メモリ環境の並列計算機上で保持することで解析が可能になる．

HEC-MW のリファイン機能では，リファインによって新しく生成された節点情報が，既存の節点情報に続いて定義される．節点情報が計算結果に影響する前処理アルゴリズムを用いる際には，オーダリングの適用を考慮するなど注意が必要になる．

4.2.4 反復法ソルバにおける並列計算

並列反復法の計算において重要なのは，ベクトル内積と行列ベクトル積，前処理の並列計算を実現することである．HEC-MW の並列反復法ソルバは，MPI(Message Passing Interface) を用いた分散メモリ環境での並列計算を可能としている．前処理の並列計算については，次章以降で述べる．

ベクトル内積の並列計算は，各領域でのベクトル内積の結果を，通信時にデータの総和を取る集団通信 (MPI_ALLREDUCE) を行うことで計算できる．

行列ベクトル積の並列計算は，各領域での行列ベクトル積の結果が全て得られた後，隣接する領域間でデータ通信 (MPI_SEND_RECIEVE) を行うことで計算できる．各領域で行われる行列ベクトル積は，それぞれ完全に並列計算が可能であるため，HEC-MW ではこの部分を OpenMP[34] を用いたスレッド並列によって並列計算を行っている．

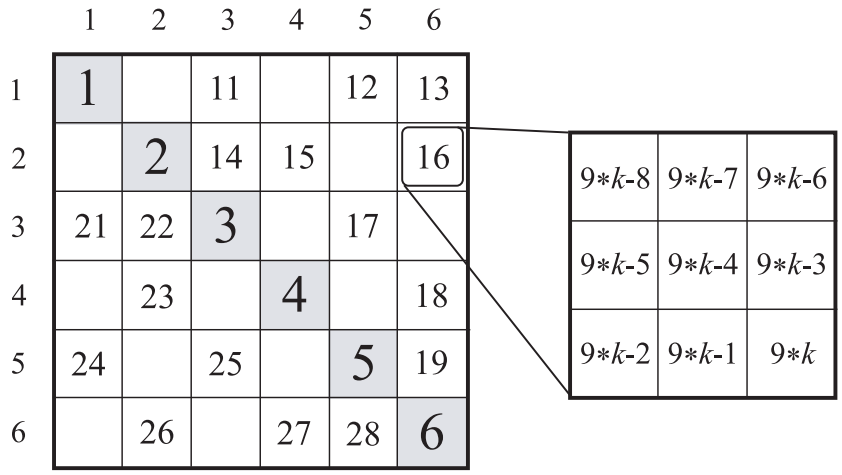
これら行列ベクトル演算が適用される係数行列は，HEC-MW 上に BCSR (Block Compressed Sparse Row, または Block Compressed Row Storage) 方式で格納されている．図 12 に，BCSR 方式の概要を示す．

BCSR 方式は，CSR 方式による格納を拡張したものであり，一行にいくつの要素が存在しているかを格納した Index 配列と，各々の要素が何列目に存在しているかを格納した Item 配列によって，疎行列の要素情報を効率良く表現できる．

3×3 BCSR 方式の場合，図 12 右のように，CSR 方式のひとつの要素に対応するブロックに 9 つの情報が格納されていることを表している．k は，Index 配列から得られる，要素配列を参照するインデックスである．有限要素法において，ひとつの節点あたり 3 つの自由度を持つ要素から得られる全体剛性マトリクスは，3 行 3 列の小行列を用いて係数行列の非零プロファイルを簡潔に表現できる．この 3 行 3 列の小行列の要素を連続的にメモリ配列へ格納することで，計算効率を向上させることができる．

4.3 結言

この章では，本研究で開発基盤として用いた，大規模並列有限要素解析ソフトウェア FrontISTR とそのミドルウェア HECMW について述べた．並列計算に必要な行列ベクトル積とベクトル内積の並列化に関して述べた．



D = [1,2,3,4,5,6]
AU = [11,12,13,14,15,16,17,18,19]
IndexU = [0,3,6,7,8,9,9]
ItemU = [3,5,6,3,4,6,5,6,6]
AL = [21,22,23,24,25,26,27,28]
IndexL = [0,0,0,2,3,5,8]
ItemL = [1,2,2,1,3,2,4,5]

Fig. 12 Schematic representation of Block Compressed Sparse Row

第 5 章　反復法前処理

5.1　諸言

この章では，反復法として本研究で扱う共役勾配法の前処理について述べる．まず，最も簡易的な前処理として知られる対角スケーリング前処理について述べる．次に，対称逐次過緩和前処理について述べる．次に，不完全 LU 分解前処理について，フィルインの有無に合わせて述べる．最後に，本研究で注目する，A-直交過程に基づく近似逆行列系前処理について述べる．

5.2　対角スケーリング前処理

対角スケーリング前処理は，前処理行列 $\boldsymbol{M}$ の対角成分を係数行列 $\boldsymbol{A}$ の対角成分として用いるものである．図 13 に，対角スケーリング前処理を示す．このとき，$\boldsymbol{m}_{ii}$ は前処理行列 $\boldsymbol{M}$ の i 番目の対角要素，$\boldsymbol{a}_{ii}$ は係数行列 $\boldsymbol{A}$ の i 番目の対角要素，n は次元数である．計算負荷が小さく，完全に並列計算が可能であるのが特徴である．係数行列 $\boldsymbol{A}$ の強い近似ではないが，対角優位な行列に対して有効な前処理能力をもつ場合がある．

ブロック対角スケーリング前処理では，対角ブロックの逆行列を計算して前処理行列を生成する．図 14 に，ブロック対角スケーリング前処理を示す．このとき，$\boldsymbol{M}_{ii}$ は前処理

1: $\boldsymbol{M} = \boldsymbol{0}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad m_{ii} = a_{ii}$
4: **end for**

Fig. 13　Preconditioner of diagonal scaling

1: $\boldsymbol{M} = \boldsymbol{0}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{M}_{ii} = \boldsymbol{A}_{ii}$
4: **end for**

Fig. 14　Preconditioner of block diagonal scaling

行列 $\boldsymbol{M}$ の i 番目の対角ブロック，$\boldsymbol{a}_{ii}$ は係数行列 $\boldsymbol{A}$ の i 番目の対角ブロック，n は節点数，対角ブロックは $ndof \times ndof$ の大きさで，$ndof$ は係数行列を BCSR 形式で格納した場合，1 節点あたりの自由度である．FrontISTR では対角成分の逆行列は陽に保持せず，LU 分解後の成分を保持することで，計算効率を高める方法を採用している．

5.3 対称逐次過緩和前処理

対象逐次過緩和前処理 (Symmetry Successive Over-Relaxation preconditioning, SSOR) は，式 (50) に示すように係数行列 $\boldsymbol{A}$ を行列の和で表すことで得られる分離型前処理である [3]．このとき，前処理行列 $\boldsymbol{M}$ を式 (51) のように生成する．

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U} \tag{50}$$

$$\boldsymbol{M} = (\frac{\boldsymbol{D}}{\omega} + \boldsymbol{L})\frac{\omega}{2-\omega}\boldsymbol{D}^{-1}(\frac{\boldsymbol{D}}{\omega} + \boldsymbol{U}) \tag{51}$$

ここで，$\boldsymbol{L}$ は対角成分が 0 の下三角行列，$\boldsymbol{D}$ は対角行列，$\boldsymbol{U}$ は対角成分が 0 の上三角行列，ω は緩和パラメータであり $0 < \omega < 2$ をとる．

5.4 不完全 LU 分解前処理

5.4.1 フィルインを考慮しない不完全 LU 分解前処理

不完全 LU 分解前処理 (Incomplete LU factorization preconditioning, ILU) は，Gauss の消去法に基づいて近似的な LU 分解を施すことで前処理行列 $\boldsymbol{M} = \boldsymbol{LU}$ を生成する [3]．特に，分解時の前処理行列の非零プロファイルを係数行列と同じものに固定して分解を行うフィルインを考慮しない不完全 LU 分解前処理は，ILU(0) と呼ばれる．図 15 に，フィルインを考慮しない不完全 LU 分解前処理 (ILU(0)) を示す．

このアルゴリズムでは，下三角行列 $\boldsymbol{L}$ の対角成分が 1 であることを自明として，対角を含まない前処理行列 $\boldsymbol{M}$ の下三角部分に，対角を含まない下三角行列 $\boldsymbol{L}$ を格納する．その前処理行列 $\boldsymbol{M}$ の対角を含む上三角行列部分に，上三角行列 $\boldsymbol{U}$ を格納する．

ILU 前処理は，解く問題によって分解が不安定になり，前処理行列の対角項に負の値が発生する場合がある．そのような問題には，ILU 分解前の前処理行列 $\boldsymbol{M}$ の対角項を，あらかじめ修正係数を用いて定数倍して分解を行うなど対策が必要である．

```
1: M = A
2: for i = 1, 2, 3, ..., n do
3:   for k = 1, 2, 3, ..., n do
4:     if m_ik ≠ 0 then
5:       m_ik = m_ik / m_kk
6:     end if
7:     for j = 1, 2, 3, ..., n do
8:       if m_ij ≠ 0 then
9:         m_ij = m_ij − m_ik m_kj
10:      end if
11:    end for
12:  end for
13: end for
```

Fig. 15 Preconditioner of incomplete LU(0)

5.4.2 p 段のフィルインを考慮した不完全 LU 分解前処理

前述の不完全 LU 分解前処理は，前処理行列の非零プロファイルを係数行列と同じものに固定して分解を行った．一般に数段のフィルインを考慮して非零プロファイルの数を増やせば，フィルインを考慮しない場合に比べて，前処理行列が係数行列の良い近似に近づくため，強い前処理能力をもつ．このような，p 段のフィルインを考慮した不完全 LU 分解前処理は，ILU(p) と呼ばれる [3]．図 16 に，p 段のフィルインを考慮した不完全 LU 分解前処理 (ILU(p)) を示す．ここで，p はフィルインの段数，$\boldsymbol{F}$ はフィルインを決定するために用いる行列である．

```
1: fill-in level = p, F = 0, M = A
2: for all i, j do
3:   if a_ij = 0 then
4:     f_ij = p + 1
5:   end if
6: end for
7: for i = 1, 2, 3, ..., n do
8:   for k = 1, 2, 3, ..., n do
9:     if f_ik ≤ p then
10:      m_ik = m_ik / m_kk
11:    end if
12:    for j = 1, 2, 3, ..., n do
13:      if min(f_ij, f_ik + f_kj + 1) ≤ p then
14:        m_ij = m_ij − m_ik m_kj
15:      end if
16:    end for
17:  end for
18: end for
```

Fig. 16 Preconditioner of incomplete LU(p)

5.5 A-直交過程に基づく近似逆行列系前処理

5.5.1 近似逆行列分解

近似逆行列分解 (Approximate inverse, AINV) は，A-直交過程に基づき前処理行列の逆行列 $\boldsymbol{M}^{-1} = \boldsymbol{Z}\boldsymbol{D}^{-1}\boldsymbol{Z}^t$ として生成するものである．

係数行列 $\boldsymbol{A}$ に対して共役なベクトル $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n$ を並べたものを，式 (52) に示す．

$$\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n] \tag{52}$$

行列 $\boldsymbol{Z}$ と係数行列 $\boldsymbol{A}$ の正定値性を利用すれば，式 (53) のように分解できる．

$$\boldsymbol{Z}^T\boldsymbol{A}\boldsymbol{Z} = \boldsymbol{D}, \quad \begin{cases} d_i = \boldsymbol{z}_i^T\boldsymbol{A}\boldsymbol{z}_i \\ 0 = \boldsymbol{z}_i^T\boldsymbol{A}\boldsymbol{z}_j \quad (i \neq j) \end{cases} \tag{53}$$

ここで，d_i は対角行列 $\boldsymbol{D}$ の第 i 番目の対角要素である．

式 (53) から，係数行列 $\boldsymbol{A}$ の逆行列は式 (54) で得られる．

$$\boldsymbol{A}^{-1} = \boldsymbol{Z}\boldsymbol{D}^{-1}\boldsymbol{Z}^t \tag{54}$$

このような共役なベクトル $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n$ を生成するための方法として，図 17 に示す A-直交過程（共役グラムシュミット法）が知られている [6][35]．ここで，$\boldsymbol{e}_j$ は単位行列 $\boldsymbol{E}$ の第 j 番目の列ベクトルである．A-直交過程は，行列の完全分解を行う方法であり，一般に，逆行列を生成する場合元の行列が疎であっても逆行列は疎にならない．そこで，前処理行列としてスパース性を保つように，閾値 tol より小さい分解因子の値は零に棄却する dropping 処理を行うことで，近似逆行列分解前処理として利用することが提案された．

図 18 に，近似逆行列分解前処理を示す．ここで，$\boldsymbol{z}_j$ は対角成分が 1 の上三角行列 $\boldsymbol{Z}$ の第 j 番目の列ベクトル，$\boldsymbol{a}_j$ は係数行列 $\boldsymbol{A}$ の第 j 番目の列ベクトル，d_j は対角行列 $\boldsymbol{D}$

1: $\boldsymbol{Z} = \boldsymbol{E}$
2: **for** $i = 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{z}_i = \boldsymbol{e}_i + \sum_{j=1}^{i-1} -\frac{{}^t\boldsymbol{e}_i\boldsymbol{A}\boldsymbol{z}_j}{{}^t\boldsymbol{z}_j\boldsymbol{A}\boldsymbol{z}_j}\boldsymbol{z}_j$
4: **end for**

Fig. 17 A-orthogonalization process

1: $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: **for** $j = i, i+1, i+2, \ldots, n$ **do**
4: $d_j = {}^t\boldsymbol{a}_i \boldsymbol{z}_j^{(i-1)}$
5: **end for**
6: **for** $j = i+1, i+2, \ldots, n$ **do**
7: **for** $k = 1, 2, 3, \ldots, n$ **do**
8: $z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
9: **end for**
10: **end for**
11: **end for**

Fig. 18　Preconditioner of approximate inverse

の第 j 番目の対角要素，$\boldsymbol{E}$ は単位行列，上付き添字 (i) は i 回目の分解を示す．閾値 tol より小さい分解因子の値は零に棄却する dropping 処理は，$\mathrm{drop}(*, tol)$ で示した．

不完全 LU 分解前処理は，前処理行列 $\boldsymbol{M} = \boldsymbol{LU}$ を生成するものであった．前処理行列 $\boldsymbol{M} = \boldsymbol{LU}$ として生成すると，共役勾配法の反復の中では，$\boldsymbol{LU}$ を用いた前進消去と後退代入によって，前処理行列の逆行列 $\boldsymbol{M}^{-1}$ を計算しなければならない．前進消去と後退代入は計算の前後に依存関係があるため，この部分の並列計算は難しい．近似逆行列系前処理では，前処理行列の逆行列 $\boldsymbol{M}^{-1} = \boldsymbol{Z}\boldsymbol{D}^{-1}\boldsymbol{Z}^t$ とするため，共役勾配法における前処理行列の適用を完全に並列計算することができる．

1: $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{s} = \boldsymbol{A}\boldsymbol{z}_i^{(i-1)}$
4: $\quad$ **for** $j = i, i+1, i+2, \ldots, n$ **do**
5: $\quad\quad d_j = {}^t\boldsymbol{s}\boldsymbol{z}_j^{(i-1)}$
6: $\quad$ **end for**
7: $\quad$ **for** $j = i+1, i+2, \ldots, n$ **do**
8: $\quad\quad$ **for** $k = 1, 2, 3, \ldots, n$ **do**
9: $\quad\quad\quad z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
10: $\quad\quad$ **end for**
11: $\quad$ **end for**
12: **end for**

Fig. 19 Preconditioner of stabilized approximate inverse

5.5.2 安定化近似逆行列分解

安定化近似逆行列分解 (Stabilized approximate inverse, SAINV) は，逆行列分解中に正定値性を保ちながら計算をすすめる手法である [7]．近似逆行列分解では，対角成分 $d_j = {}^t\boldsymbol{a}_i\boldsymbol{z}_j^{(i-1)}$ として計算した．しかし，dropping 処理の影響で分解後の前処理行列の正定値性が崩れることがある．そこで安定化近似逆行列分解では，対角成分 $d_j = {}^t(\boldsymbol{z}_i^{(i-1)})\boldsymbol{A}\boldsymbol{z}_j^{(i-1)}$ として計算する．係数行列 $\boldsymbol{A}$ が正定値行列のため，任意の零でないベクトルに対して対角成分 d_j は正の値になる．dropping 処理による不安定な分解を抑制でき，より堅固な前処理行列を生成することができる．

図 19 に，安定化近似逆行列分解前処理を示す．ここで，$\boldsymbol{s}$ は中間ベクトルである．

5.5.3 ロバスト不完全分解

ロバスト不完全分解 (Robust Incomplete Factorization, RIF) は，安定化近似逆行列分解の分解過程の中で，新たに定めた不完全分解因子 $\boldsymbol{L}$ から前処理行列 $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^t$ を得る手法である [10].

ここではまず，係数行列 $\boldsymbol{A}$ と係数行列の逆行列 $\boldsymbol{A}^{-1}$ を式 (55)，式 (56) のように完全分解することを考える．

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}_a\boldsymbol{L}^t \tag{55}$$

$$\boldsymbol{A}^{-1} = \boldsymbol{Z}\boldsymbol{D}_b^{-1}\boldsymbol{Z}^t \tag{56}$$

式 (55) は完全コレスキー分解であり，この逆行列は式 (57) となる．

$$\boldsymbol{A}^{-1} = \boldsymbol{L}^{-t}\boldsymbol{D}_a^{-1}\boldsymbol{L}^{-1} \tag{57}$$

式 (56) と式 (57) は等しいことから，式 (58) と式 (59) が成り立つ．

$$\boldsymbol{Z}^t = \boldsymbol{L}^{-1} \tag{58}$$

$$\boldsymbol{D}_a = \boldsymbol{D}_b \tag{59}$$

これらの関係から式 (60) が導かれる．

$$\boldsymbol{L} = \boldsymbol{A}\boldsymbol{Z}\boldsymbol{D}_b^{-1} \tag{60}$$

式 (60) は，係数行列の逆行列 $\boldsymbol{A}^{-1}$ の分解過程から，係数行列 $\boldsymbol{A}$ の分解因子 $\boldsymbol{L}$ を得ることが可能であると示している．式 (61) に示すように，安定化近似逆行列分解の分解過程の中で計算される d_j/d_i は，式 (60) の要素と等しい．

$$\frac{d_j}{d_i} = \frac{{}^t\boldsymbol{a}_i\boldsymbol{z}_j^{(i-1)}}{d_i} \tag{61}$$

このように，ロバスト不完全分解は，安定化近似逆行列分解の分解過程の中で計算される d_j/d_i を用いて，不完全分解因子 $\boldsymbol{L}$ を $l_{ji} = d_j/d_i$ として，前処理行列 $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^t$ を構成する．不完全 LU 分解とは異なった考えに基づく不完全分解であり，A-直交過程に基づく不完全分解と呼ばれる．基本となる安定化近似逆行列分解が行列の正定値性を保ちながら計算するため，不完全 LU 分解前処理では不安定な前処理行列が生成される問題に対しても，強い前処理能力が得られる場合がある．

1: $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$, $\boldsymbol{L}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{s} = \boldsymbol{A}\boldsymbol{z}_i^{(i-1)}$
4: $\quad$ **for** $j = i, i+1, i+2, \ldots, n$ **do**
5: $\quad\quad d_j = {}^t\boldsymbol{s}\boldsymbol{z}_j^{(i-1)}$
6: $\quad$ **end for**
7: $\quad$ **for** $j = i+1, i+2, \ldots, n$ **do**
8: $\quad\quad$ **if** $(|\ \frac{d_j}{d_i}\ | > tol)$ **then**
9: $\quad\quad\quad l_{ji} = \dfrac{d_j}{d_i}$
10: $\quad\quad$ **end if**
11: $\quad\quad$ **for** $k = 1, 2, 3, \ldots, n$ **do**
12: $\quad\quad\quad z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
13: $\quad\quad$ **end for**
14: $\quad$ **end for**
15: **end for**

Fig. 20 Preconditioner of robust incomplete factorization

図 20 に，ロバスト不完全分解前処理を示す．ここで，l_{ji} は対角成分が 1 の下三角行列 $\boldsymbol{L}$ の第 (j,i) 番目の行列要素である．近似逆行列分解前処理と同様にスパース性を保つため，下三角行列 $\boldsymbol{L}$ にも dropping 処理を行う．

5.5.4 改良型安定化近似逆行列分解

改良型安定化近似逆行列分解 (Improved SAINV, ISAINV) は，安定化近似逆行列の分解中に新たな閾値を用いて近似分解の精度を向上させたものである [11][36].

安定化近似逆行列分解に基づく完全分解は，上三角行列 $\boldsymbol{Z}$ の列ベクトル $\boldsymbol{z}_i$ を，A-直交過程に基づいての式 (62) となるように生成する．

$$\boldsymbol{z}_j \boldsymbol{A} \boldsymbol{z}_i = 0 \quad (i \neq j) \tag{62}$$

対角項にのみ非零要素が生じるので，式 (63) と表すことができる．

$$\boldsymbol{Z}^t \boldsymbol{A} \boldsymbol{Z} = \boldsymbol{D} \tag{63}$$

1: $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{s} = \boldsymbol{A}\boldsymbol{z}_i^{(i-1)}$
4: $\quad$ **for** $j = i, i+1, i+2, \ldots, n$ **do**
5: $\quad\quad d_j = {}^t\boldsymbol{s}\boldsymbol{z}_j^{(i-1)}$
6: $\quad$ **end for**
7: $\quad$ **for** $j = i+1, i+2, \ldots, n$ **do**
8: $\quad\quad$ **if** $(|\ \frac{d_j}{d_i}\ | > toldd)$ **then**
9: $\quad\quad\quad$ **for** $k = 1, 2, 3, \ldots, n$ **do**
10: $\quad\quad\quad\quad z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
11: $\quad\quad\quad$ **end for**
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**

Fig. 21 Preconditioner of improved SAINV

dropping 処理の値の棄却によって不完全に分解が行われた場合は，誤差行列 $\boldsymbol{E}$ を用いて，式 (64) と表すことができる．

$$\boldsymbol{Z}^t\boldsymbol{A}\boldsymbol{Z} = \boldsymbol{D} + \boldsymbol{E} \tag{64}$$

式 (64) から，安定化近似逆行列の分解中に生じる $d_i = \boldsymbol{z}_i^t\boldsymbol{A}\boldsymbol{z}_i$，$d_j = \boldsymbol{z}_j^t\boldsymbol{A}\boldsymbol{z}_i$ は，それぞれ対角行列 $\boldsymbol{D}$ と誤差行列 $\boldsymbol{E}$ の要素に等しい．このとき，計算中に現れる d_j/d_i は，更新に用いる因子 d_j を対角因子 d_i で正規化した値になっている．これに注目し，新たな閾値 $toldd$ によって，対角因子に対してある閾値よりも大きな割合をもつ因子による値の更新のみを行うことで，近似逆行列分解の精度を向上させる．この新たな dropping 処理は double dropping と呼ばれる．

図 21 に，改良型安定化近似逆行列分解前処理を示す．ここで，閾値 $toldd$ はベクトル $\boldsymbol{z}_j^{(i-1)}$ の更新判定として使用される．

ISAINV については同様の手法が，Lee and Zhang により Factored Sparse Approximate Inverse (FAPINV) として独立に提案されている [37][38].

1: $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$, $\boldsymbol{L}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{s} = \boldsymbol{A}\boldsymbol{z}_i^{(i-1)}$
4: $\quad$ **for** $j = i, i+1, i+2, \ldots, n$ **do**
5: $\quad\quad d_j = {}^t\boldsymbol{s}\boldsymbol{z}_j^{(i-1)}$
6: $\quad$ **end for**
7: $\quad$ **for** $j = i+1, i+2, \ldots, n$ **do**
8: $\quad\quad$ **if** $(|\ \frac{d_j}{d_i}\ | > tol)$ **then**
9: $\quad\quad\quad l_{ji} = \dfrac{d_j}{d_i}$
10: $\quad\quad$ **end if**
11: $\quad\quad$ **if** $(|\ \frac{d_j}{d_i}\ | > toldd)$ **then**
12: $\quad\quad\quad$ **for** $k = 1, 2, 3, \ldots, n$ **do**
13: $\quad\quad\quad\quad z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
14: $\quad\quad\quad$ **end for**
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end for**

Fig. 22 Preconditioner of improved RIF

5.5.5 改良型ロバスト不完全分解

改良型ロバスト不完全分解 (Improved RIF, IRIF) は，改良型安定化近似逆行列分解に基づいて，不完全分解因子 $\boldsymbol{L}$ から前処理行列 $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^t$ を得る手法である．[8] 図 22 に，改良型ロバスト不完全分解を示す．

5.6 結言

この章では，反復法として本研究で扱う共役勾配法の前処理について述べた．まず，対角スケーリング前処理について述べた．次に，対称逐次過緩和前処理について述べた．次に，不完全 LU 分解前処理について，フィルインの有無に合わせて述べた．最後に，本研究で注目する，A-直交過程に基づく近似逆行列系前処理について述べた．

第 6 章　並列有限要素法解析のためのロバスト不完全分解の高速化と並列化

6.1　諸言

この章では，本研究が提案する並列有限要素法解析のためのロバスト不完全分解の高速化と並列化について述べる．まず，計算高速化のための係数行列の非零プロファイルの利用について述べる．次に，並列化のための Localized 処理について述べる．最後に，計算高速化と将来的な高精度演算に追従するための混合精度演算について述べる．

6.2　係数行列の非零プロファイルの利用

従来の RIF 前処理は，前処理行列 $\boldsymbol{M}$ のスパース性を保つために，前処理行列生成中に閾値 tol を用いた値の棄却を行う必要がある．この方法では，様々な問題の係数行列に対して，それぞれ閾値 tol を経験的に定めなければならず，スパース性も閾値 tol に依存する．並列計算の場合，各計算領域に属する非零プロファイルの数が値の棄却により偏ることで，計算負荷が均一にならずに計算性能が低下することが考えられる．このように，最適な閾値 tol を定めるのは容易ではない．さらに，各計算領域間での通信に用いる情報を更新しなければならず，実装時のプログラムの複雑化や計算負荷の増大に結びつく．

また，前処理行列生成中に得られる全ての要素に対して閾値 tol による棄却判定を行わなければならず，前処理行列における非零要素の位置を動的に更新しながら定めることは，前処理行列生成時に大きな計算負荷になり得るため，従来の処理では前処理行列生成の高速化には向いておらず，手法の改良が必要である．

この問題を解決するために，係数行列 $\boldsymbol{A}$ の非零要素プロファイルを前処理生成に利用した IRIF(0) と，IRIF(0) から一段のフィルインを考慮した IRIF(1) を提案する [39]．このとき IRIF(1) の非零プロファイルは，フィルインを考慮した不完全 LU 分解 (ILU(1)) の非零プロファイルと同じものを用いる．

図 23 に，係数行列の非零プロファイルを利用した IRIF の概要を示す．図 24 に IRIF(0)，図 25 に IRIF(1) を示す．提案手法では，係数行列 $\boldsymbol{A}$ の非零要素プロファイルを，前処理行列 $\boldsymbol{M}$ を生成する上三角行列 $\boldsymbol{Z}$ と下三角行列 $\boldsymbol{L}$ に適用し，非零要素以外の

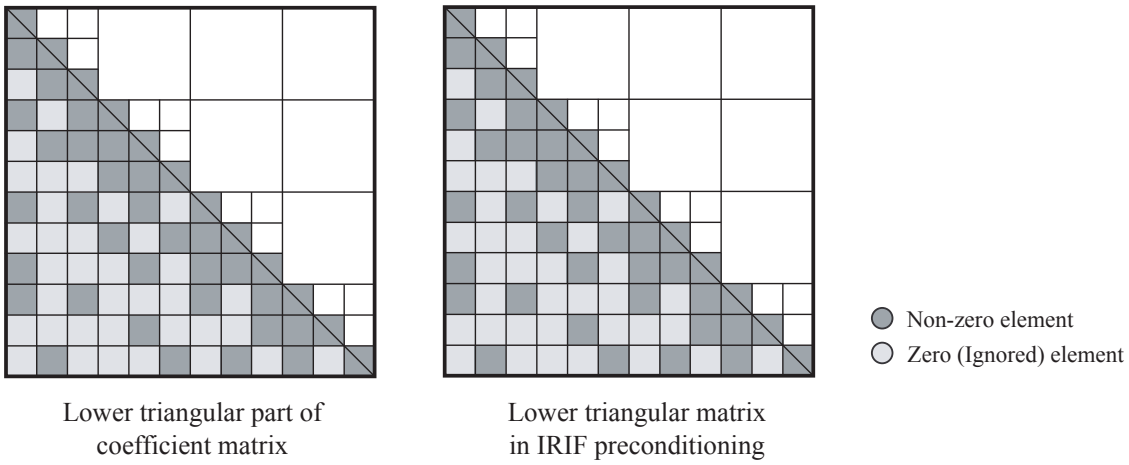


Fig. 23 Schematic representation of using lower triangular part of non-zero element profile of coefficient matrix in IRIF preconditioning

要素をあらかじめ零と固定することで，前処理生成の高速化を図る．そのため提案手法では，従来必要だった閾値 tol を用いた値の棄却処理 (drop($*$, tol)) の必要はない．一方，double dropping 処理で導入された閾値 $toldd$ によるベクトル $\boldsymbol{z}_j^{i-1}$ の更新判定は前処理行列生成時の分解精度を高めるため，閾値 $toldd$ による値の更新判定を追加する．

ここで，係数行列 $\boldsymbol{A}$ が自由度 n の十分大きな疎行列で，バンド幅 K の行列と見なすことを仮定する．このとき適切に係数行列の非零要素を保持すれば，行列ベクトル積は乗算 Kn 回，加算 $(K-1)n$ 回の演算が必要となる．係数行列 $\boldsymbol{A}$ のあるベクトル $\boldsymbol{a}_i$ と $\boldsymbol{a}_j$ のベクトル内積は乗算 K 回，加算 $K-1$ 回が必要となる．すなわち，行列ベクトル積は演算量 $O(n)$，ベクトル内積は演算量 $O(n^0)$ である．

以上を踏まえ，図 24 の IRIF(0) では，前処理生成時の外側ループ i 毎に行列ベクトル積を 1 回 (3 行目) と，ベクトル内積を $(n-i+1)$ 回 (4〜6 行目) 行う必要がある．この外側ループ i は 1〜n まで演算するため，外側ループ i あたり 1 回の行列ベクトル積，$(n-i+1)$ 回のベクトル内積により，全体として演算量 $O(n^2)$ となる．

1: $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$, $\boldsymbol{L}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{s} = \boldsymbol{A}\boldsymbol{z}_i^{(i-1)}$
4: $\quad$ **for** $j = i, i+1, i+2, \ldots, n$ **do**
5: $\quad\quad d_j = {}^t\boldsymbol{s}\boldsymbol{z}_j^{(i-1)}$
6: $\quad$ **end for**
7: $\quad$ **for** $j = i+1, i+2, \ldots, n$ **do**
8: $\quad\quad$ **if** $(|\ \frac{d_j}{d_i}\ | > tol$ **and** $\boldsymbol{A}_{ij} \neq 0)$ **then**
9: $\quad\quad\quad l_{ji} = \dfrac{d_j}{d_i}$
10: $\quad\quad$ **end if**
11: $\quad\quad$ **if** $(|\ \frac{d_j}{d_i}\ | > toldd)$ **then**
12: $\quad\quad\quad$ **for** $k = 1, 2, 3, \ldots, n$ **do**
13: $\quad\quad\quad\quad$ **if** $\boldsymbol{A}_{kj} \neq 0$ **then**
14: $\quad\quad\quad\quad\quad z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
15: $\quad\quad\quad\quad$ **end if**
16: $\quad\quad\quad$ **end for**
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: **end for**

Fig. 24 Preconditioner of IRIF(0)

1: fill-in level = 1, $\boldsymbol{F} = \boldsymbol{0}$, $\boldsymbol{Z}^{(0)} = \boldsymbol{E}$, $\boldsymbol{L}^{(0)} = \boldsymbol{E}$
2: **for all** i, j **do**
3: **if** $a_{ij} = 0$ **then**
4: $f_{ij} = 2$
5: **end if**
6: **end for**
7: **for** $i = 1, 2, 3, \ldots, n$ **do**
8: $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{z}_i^{(i-1)}$
9: **for** $j = i, i+1, i+2, \ldots, n$ **do**
10: $d_j = {}^t\boldsymbol{s}\boldsymbol{z}_j^{(i-1)}$
11: **end for**
12: **for** $j = i+1, i+2, \ldots, n$ **do**
13: **if** ($|\ \frac{d_j}{d_i}\ | > tol$ **and** $\min(f_{ik}, f_{ij} + f_{jk} + 1) \leq 1$) **then**
14: $l_{ji} = \dfrac{d_j}{d_i}$
15: **end if**
16: **if** ($|\ \frac{d_j}{d_i}\ | > toldd$) **then**
17: **for** $k = 1, 2, 3, \ldots, n$ **do**
18: **if** $\min(f_{ik}, f_{ij} + f_{jk} + 1) \leq 1$ **then**
19: $z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
20: **end if**
21: **end for**
22: **end if**
23: **end for**
24: **end for**

Fig. 25 Preconditioner of IRIF(1)

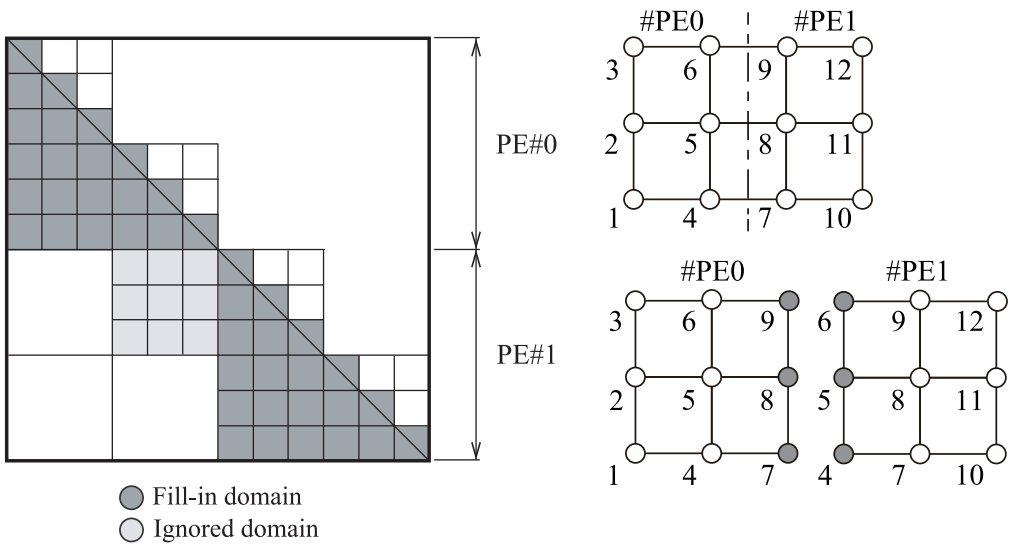


Fig. 26 Schematic representation of fill-in and ignored domains of lower triangular matrix in localized IRIF preconditioning

6.3 Localized IRIF

大規模計算を考える場合，前処理の並列化は欠かすことができない．そのため，いかに堅固な前処理であっても高並列化を考慮しなければならない．一口に並列化といってもその手法は様々あり，フィルイン無視，並列計算領域間の通信無視，局所ブロック化等が挙げられる．その中で本研究では，Localized IRIF を提案する [39].

Localized IRIF の概要を図 26 に示す．ここで，分割された前処理用下三角行列を 2 つのプロセスで並列計算する場合を考える．Localized IRIF では，図 26 左で薄く示すような，複数のプロセス間に跨る領域にフィルインを認めないことで高い並列化と高速化を見込んでいる．非対角部分にフィルインを認めないことは，他領域と外点を介した通信を無視することにあたる．各プロセスで独立して前処理を実施できるため並列性能には優れるが，前処理の効果は並列数に依存する．並列数を増やした場合無視されるフィルインの量も増加するが，正定値性を維持するように分解を行う RIF は，フィルインの減少による不安定な分解を抑制する働きがある．IRIF(0)，IRIF(1) は，演算量 $O(n^2)$ のアルゴリズムであるため，局所並列化により，並列プロセス数の 2 乗に沿って演算量は減少する．

6.4 前処理行列の混合精度演算

将来の高精度な解析の必要性に伴って，従来よりも高い計算精度が要求される数値計算が必要となることが予想される．高い精度の浮動小数点数演算を実用化するためには，計算時間の増大やメモリ使用量を改善しなければならない．そこで，解の精度に影響しない部分を低い精度の浮動小数点数で計算することで，計算時間とメモリ使用量を削減できる混合精度演算が提案された．

本研究では，単精度浮動小数点（以下単精度）と倍精度浮動小数点（以下倍精度）を用いた混合精度演算に着目し，単精度を用いて前処理生成を行うことで，前処理行列に関わる計算時間の削減することを提案する．

図 27 に，混同精度演算を用いた IRIF(0) を示す．図 28 に，混同精度演算を用いた前処理付共役勾配法を示す．ここで，下付き添え字 low は単精度浮動小数点で演算することを示す．混同精度演算を用いた RIF では，倍精度の値を単精度に変換し，前処理行列生成に関わる演算を全て単精度で行い，前処理行列を生成する．

提案手法は，前述した係数行列の直交過程を用いて，前処理行列の対角項が常に正の値となるように分解を進めるため，低い精度の浮動小数点数を用いた計算でも，安定した前処理を生成できる可能性がある．

ここで，共役勾配法の反復中において前処理の逆行列 $\boldsymbol{M}^{-1} = (\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^t)^{-1}$ を適用する方法は，以下の二つが考えられる．

1. 反復中で前処理の逆行列 $\boldsymbol{M}^{-1} = \boldsymbol{L}^{-t}\boldsymbol{D}^{-1}\boldsymbol{L}^{-1}$ を計算する．
2. 新たな下三角行列 $\boldsymbol{W} = \boldsymbol{D}\boldsymbol{L}^t$ を予め計算しておいて，反復中では前処理の逆行列 $\boldsymbol{M}^{-1} = \boldsymbol{W}^{-1}\boldsymbol{L}^{-1}$ を計算する．

混合精度演算の場合，単精度で保持された前処理行列の因子 $\boldsymbol{L}$ と $\boldsymbol{D}$ から新たな下三角行列 $\boldsymbol{W}$ を予め計算する際は，単精度同士の演算による誤差の影響で計算結果が変化すると考えられる．数値例題では，この観点からも検証を行う．

1: $\boldsymbol{Z}_{low}^{(0)} = \boldsymbol{E}$, $\boldsymbol{L}_{low}^{(0)} = \boldsymbol{E}$
2: **for** $i = 1, 2, 3, \ldots, n$ **do**
3: $\quad \boldsymbol{s}_{low} = \boldsymbol{A}\boldsymbol{z}_{low,i}^{(i-1)}$
4: $\quad$ **for** $j = i, i+1, i+2, \ldots, n$ **do**
5: $\quad\quad d_{low,j} = {}^t\boldsymbol{s}_{low}\boldsymbol{z}_{low,j}^{(i-1)}$
6: $\quad$ **end for**
7: $\quad$ **for** $j = i+1, i+2, \ldots, n$ **do**
8: $\quad\quad$ **if** $(|\ \frac{d_{low,j}}{d_{low,i}}\ | > tol$ **and** $\boldsymbol{A}_{ij} \neq 0)$ **then**
9: $\quad\quad\quad l_{low,ji} = \dfrac{d_{low,j}}{d_{low,i}}$
10: $\quad\quad$ **end if**
11: $\quad\quad$ **if** $(|\ \frac{d_{low,j}}{d_{low,i}}\ | > toldd)$ **then**
12: $\quad\quad\quad$ **for** $k = 1, 2, 3, \ldots, n$ **do**
13: $\quad\quad\quad\quad$ **if** $\boldsymbol{A}_{kj} \neq 0$ **then**
14: $\quad\quad\quad\quad\quad z_{low,kj}^{(i)} = \mathrm{drop}\ (\ z_{low,kj}^{(i-1)} - \dfrac{d_{low,j}}{d_{low,i}} z_{low,ki}^{(i-1)},\ tol\ )$
15: $\quad\quad\quad\quad$ **end if**
16: $\quad\quad\quad$ **end for**
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: **end for**

Fig. 27 Preconditioner of mixed precision IRIF(0)

1: $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{M}_{low}^{-1}\boldsymbol{r}_0$
2: **for** $m = 1, 2, 3, \ldots,$ **do**
3: $\quad \alpha_m = \dfrac{(\boldsymbol{r}_{m-1}, \boldsymbol{M}_{low}^{-1}\boldsymbol{r}_{m-1})}{(\boldsymbol{p}_{m-1}, \boldsymbol{A}\boldsymbol{p}_{m-1})}$
4: $\quad \boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \alpha_m \boldsymbol{p}_{m-1}$
5: $\quad \boldsymbol{r}_m = \boldsymbol{r}_{m-1} - \alpha_m \boldsymbol{A}\boldsymbol{p}_{m-1}$
6: $\quad$ **if** $(\|\boldsymbol{r}_m\|_2 / \|\boldsymbol{r}_0\|_2 \leqq \varepsilon)$ **then stop**
7: $\quad \beta_m = \dfrac{(\boldsymbol{r}_m, \boldsymbol{M}_{low}^{-1}\boldsymbol{r}_m)}{(\boldsymbol{r}_{m-1}, \boldsymbol{M}_{low}^{-1} r_{m-1})}$
8: $\quad \boldsymbol{p}_m = \boldsymbol{M}_{low}^{-1}\boldsymbol{r}_m + \beta_m \boldsymbol{p}_{m-1}$
9: **end for**

Fig. 28 Mixed precision preconditioned conjugate gradient method

6.5 結言

この章では，本研究が提案する並列有限要素法解析のためのロバスト不完全分解の高速化と並列化について述べた．まず，計算高速化のための係数行列の非零プロファイルの利用について述べた．次に，並列化のための Localized 処理について述べた．最後に，計算高速化のための混合精度演算について述べた．

第 7 章　数値例

7.1　諸言

この章では，提案手法と既存の手法を数値例によって比較する．まず，計算条件と計算機環境について述べる．数値例題 1 では，提案手法の逐次計算性能を評価する．数値例題 2 では，前処理性能の並列分割数による依存性を評価する．数値例題 3 では，単精度浮動小数点数で計算された前処理行列の性能を評価する．数値例題 4 では，実問題への適用事例として，エネルギ関連重要施設の数値解析を評価する．

7.2　計算条件と計算機環境

前処理を用いない共役勾配法，フィルインを考慮しない ILU(0) と IRIF(0)，1 段のフィルインを考慮した ILU(1) と IRIF(1) の前処理を有する共役勾配法，単精度で計算し 1 段のフィルインを考慮した ILU(1) と IRIF(1) の前処理を有する混合精度演算共役勾配法に対して，反復収束性と計算時間を比較する．並列計算を行う場合，前処理は Localized ILU，Localized IRIF として並列共役勾配法に適用される．ILU(0) と ILU(1) が分解中に破綻した場合は，対角項に修正係数 α を乗じて，対角項が全て正の値で分解されるよう修正した．IRIF(0) と IRIF(1) の閾値 *toldd* は，各行列ごとに並列数が最小の問題で，0.01 刻みで反復回数が最小となる値を定めた．収束条件は，相対残差に対し収束判定値 $\varepsilon = 1.0 \times 10^{-8}$ とした．

本提案手法は，領域分割に基づく並列有限要素法ソルバである FrontISTR[32] に実装した．全体剛性マトリクスは 3×3BCSR 形式で保持している．

数値例題 1 では，東京大学奥田研究室内のコンピュータを利用した．表 2 に，計算機環境を示す．数値例題 2 と数値例題 3，数値例題 4 では，東京大学情報基盤センター内の Fujitsu HPC Prime FX10 を利用した．表 3 に，計算機環境を示す．

Table 2 Specification of PC

Component	Configuration of a node
CPU	Xeon E5-1620 v2
Memory	32 GB
OS	CesntOS 6.6
Compiler	PGI Compiler 14.3-0 64bit
FEM	FrontISTR 4.4

Table 3 Specification of FX-10

Component	Configuration of a node
CPU	SPARC64 IXfx (16cores/node, 1.85GHz)
Memory	32 GB
OS	XTCOS
Network	Tofu Interconnect
Compiler	Fujitsu Compiler ver 1.2.1
FEM	FrontISTR 4.4

7.3 数値例題 1

数値例題 1 は，提案手法の逐次計算性能を評価するものである．流体の一般化固有値問題 BCSSTK13，ドームから得られた剛性行列 BCSSTK14，海上プラットフォームから得られた剛性行列 BCSSTK15，ダムから得られた剛性行列 BCSSTK16，圧力容器から得られた剛性行列 BCSSTK17，プラントから得られた剛性行列 BCSSTK18，板形状から得られた剛性行列 BCSSTK21，ドームから得られた剛性行列 BCSSTK24，高層ビルから得られた剛性行列 BCSSTK25，筒状構造から得られた剛性行列 S1RMQ4M1，筒状構造から得られた剛性行列 S2RMQ4M1，エンジンブロックから得られた剛性行列 CT20STIF,，図 29 に示す板状構造から得た剛性行列 SHELL-A の 4 つに対して，計算を行った．適用させた前処理は，フィルインを考慮しない ILU(0) と IRIF(0)，1 段のフィルインを考慮した ILU(1) と IRIF(1) である．計算には，東京大学奥田研究室内のコンピュータを用いた．

SHELL-A のモデルは，幅 200mm × 高さ 10mm × 奥行 1,000mm，シェル厚さ 0.1mm であり，要素は MITC4[19] を使用した．一要素の最大長さは 20mm で，このときシェル厚さとの比は 200 となる．

行列 BCSSTK13，BCSSTK14，BCSSTK15，BCSSTK16，BCSSTK17，BCSSTK-18，BCSSTK21，BCSSTK24，BCSSTK25，S1RMQ4M1，S2RMQ4M1 は Matrix Market[40] から，行列 CT20STIF は The University of Florida Sparse Matrix Collection[41] から取得した．Matrix Market と The University of Florida Sparse Matrix Collection から得た行列は，3×3 CBSR 方式に格納し直しており，この状態で行列の性質を評価した．

表 4 に，これらテスト行列の性質を示す．図 42 に，これらテスト行列の非零プロファイルを示す．ここで，N は行列の次元数，Non-zero Element は非零要素数，Sparsity は非零要素比，Condition number は共役勾配法を用いて推定した条件数である．非零要素比は，全体に占める全ての非零要素数の割合を計算した．

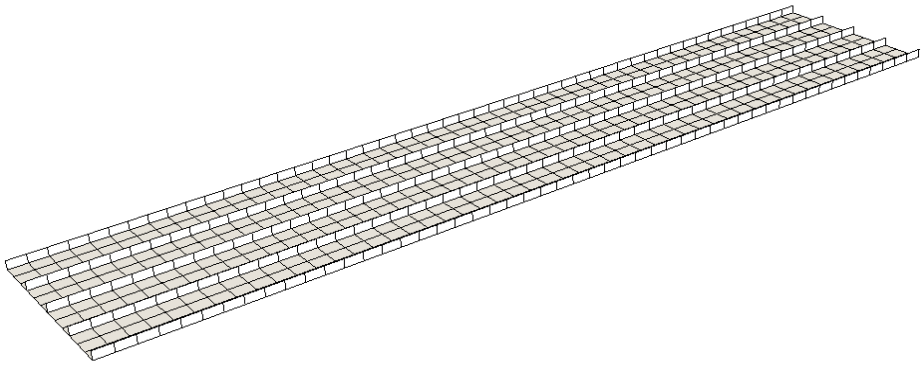
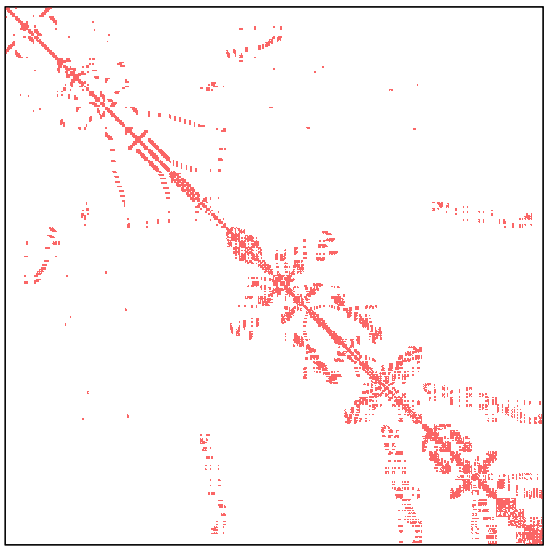


Fig. 29 Analysis mesh of SHELL-A in Problem 1

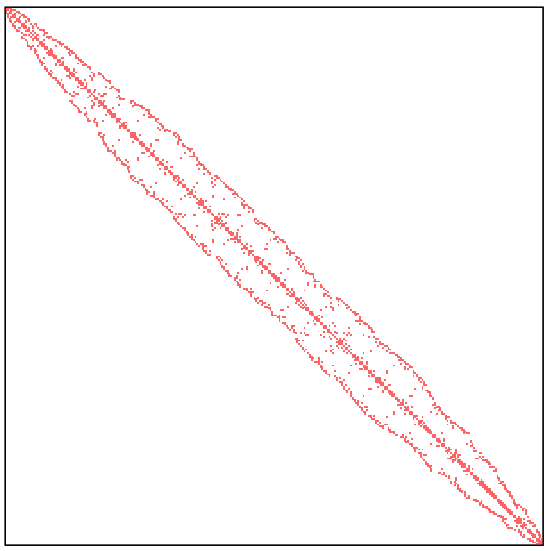
Table 4 Matrix information in Problem 1

Name	N	Non-zero element	Sparsity [%]	Condition number
BCSSTK13	2004	170604	4.25	3.08E+12
BCSSTK14	1806	74196	2.27	1.19E+10
BCSSTK15	3948	206424	1.32	6.54E+09
BCSSTK16	4884	296010	1.24	4.94E+09
BCSSTK17	10974	549918	0.457	1.30E+10
BCSSTK18	11949	557667	0.391	1.62E+11
BCSSTK21	3600	93600	0.722	1.76E+07
BCSSTK24	3564	232038	1.83	2.24E+11
BCSSTK25	15441	712125	0.299	4.12E+12
S1RMQ4M1	5490	282672	0.938	1.81E+06
S2RMQ4M1	5490	282672	0.938	1.69E+08
CT20STIF	52329	4187691	0.153	2.41E+13
SHELL-A	5508	282672	0.932	3.58E+09



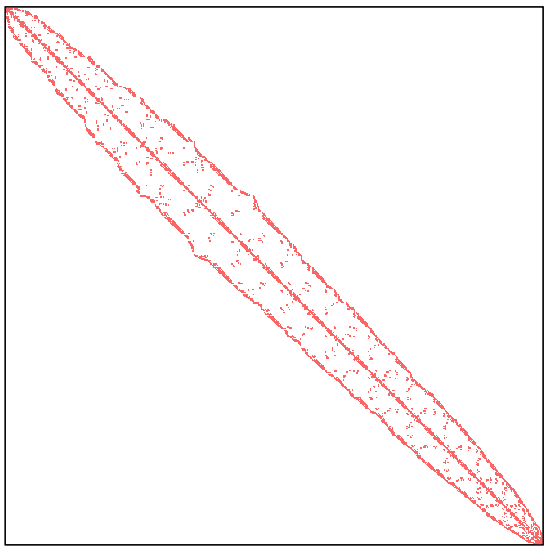
Name : BCSSTK13
N : 2004
Non-Zero Elem. : 170604
Density [%] : 4.25E+00
Condition Num. : 3.08E+12

Fig. 30 Non-zero profile of BCSSTK13



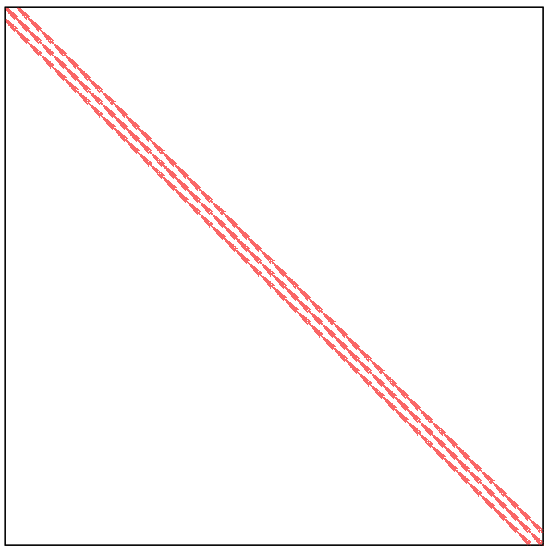
Name : BCSSTK14
N : 1806
Non-Zero Elem. : 74196
Density [%] : 2.27E+00
Condition Num. : 1.19E+10

Fig. 31 Non-zero profile of BCSSTK14



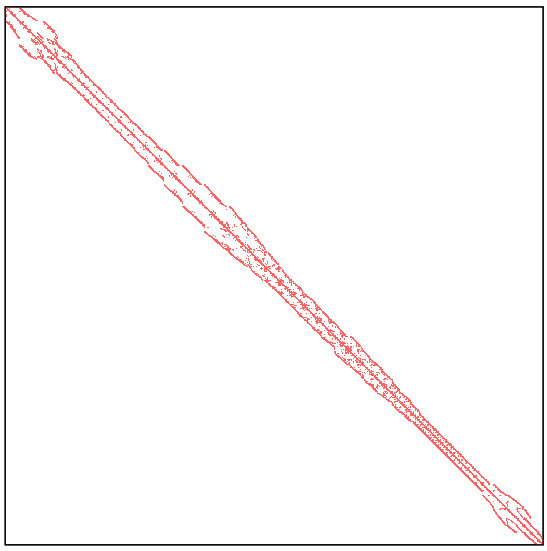
Name : BCSSTK15
N : 3948
Non-Zero Elem. : 206424
Density [%] : 1.32E+00
Condition Num. : 6.54E+09

Fig. 32 Non-zero profile of BCSSTK15



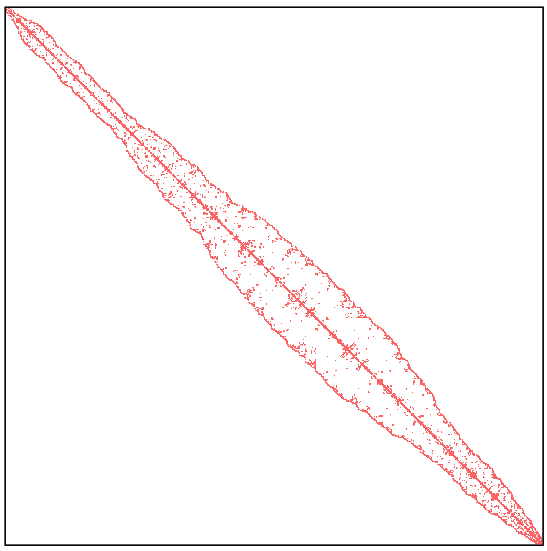
Name : BCSSTK16
N : 4884
Non-Zero Elem. : 296010
Density [%] : 1.24E+00
Condition Num. : 4.94E+09

Fig. 33 Non-zero profile of BCSSTK16



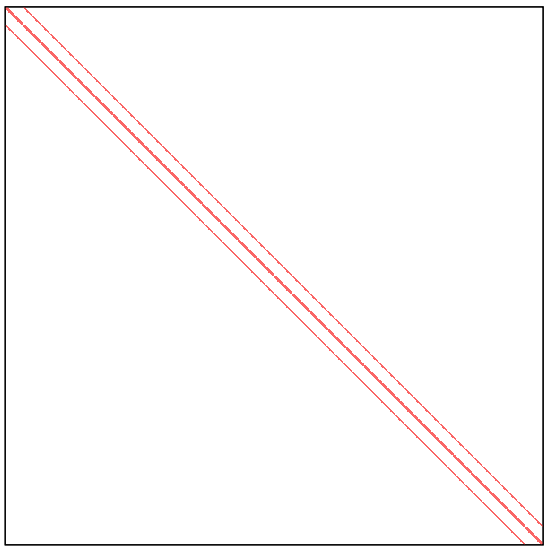
Name : BCSSTK17
N : 10974
Non-Zero Elem. : 549918
Density [%] : 4.57E-01
Condition Num. : 1.30E+10

Fig. 34 Non-zero profile of BCSSTK17



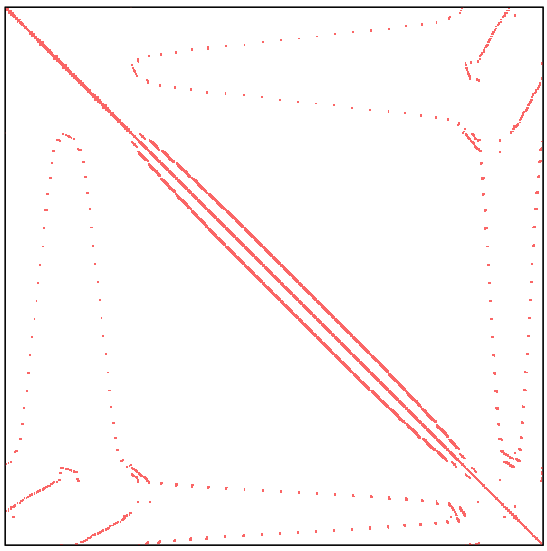
Name : BCSSTK18
N : 11949
Non-Zero Elem. : 557667
Density [%] : 3.91E-01
Condition Num. : 1.62E+11

Fig. 35 Non-zero profile of BCSSTK18



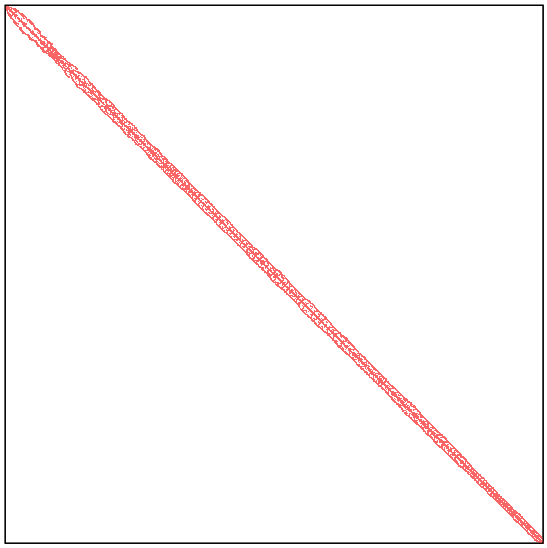
Name : BCSSTK21
N : 3600
Non-Zero Elem. : 93600
Density [%] : 7.22E-01
Condition Num. : 1.76E+07

Fig. 36 Non-zero profile of BCSSTK21



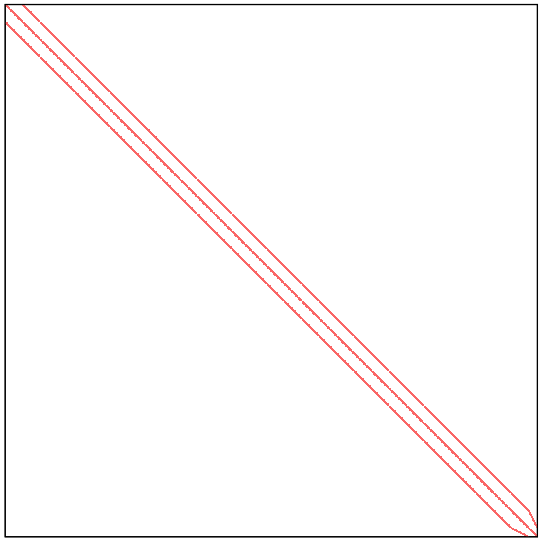
Name : BCSSTK24
N : 3564
Non-Zero Elem. : 232038
Density [%] : 1.83E+00
Condition Num. : 2.24E+11

Fig. 37 Non-zero profile of BCSSTK24



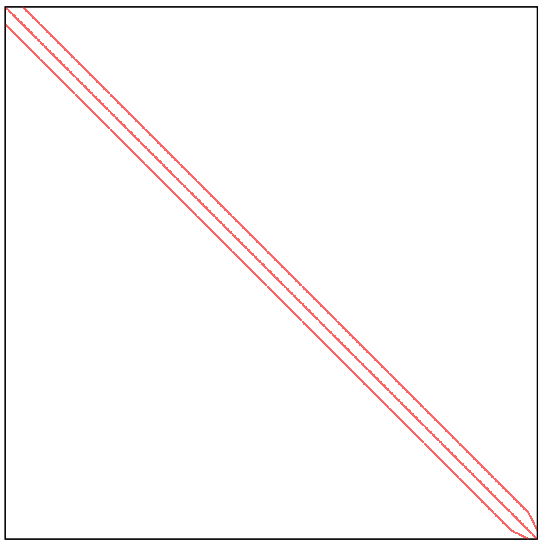
Name : BCSSTK25
N : 15441
Non-Zero Elem. : 712125
Density [%] : 2.99E-01
Condition Num. : 4.12E+12

Fig. 38 Non-zero profile of BCSSTK25



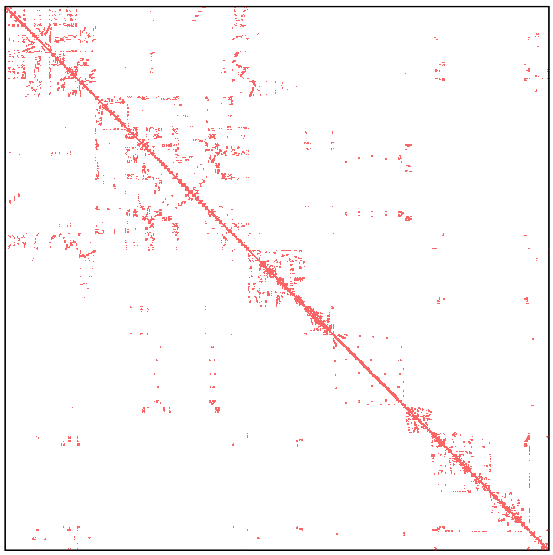
Name : S1RMQ4M1
N : 5490
Non-Zero Elem. : 282672
Density [%] : 9.38E-01
Condition Num. : 1.81E+06

Fig. 39 Non-zero profile of S1RMQ4M1



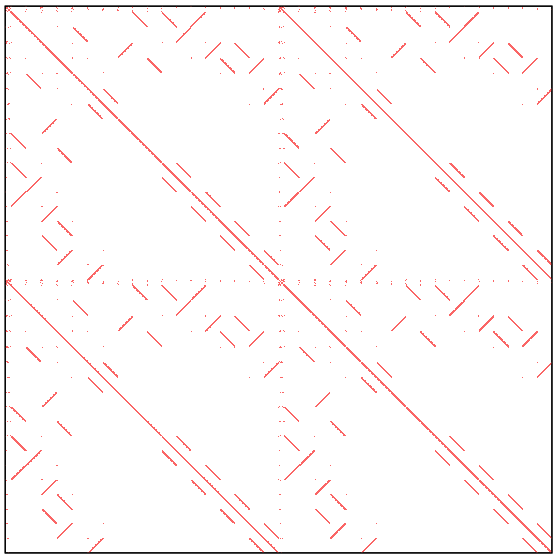
Name : S2RMQ4M1
N : 5490
Non-Zero Elem. : 282672
Density [%] : 9.38E-01
Condition Num. : 1.69E+08

Fig. 40 Non-zero profile of S2RMQ4M1



Name : CT20STIF
N : 52329
Non-Zero Elem. : 4187691
Density [%] : 1.53E-01
Condition Num. : 2.41E+13

Fig. 41 Non-zero profile of CT20STIF



Name : SHELL-A
N : 5508
Non-Zero Elem. : 282672
Density [%] : 9.32E-01
Condition Num. : 3.58E+09

Fig. 42 Non-zero profile of SHELL-A

数値例題 1 では，前処理性能の並列分割数による依存性を排除するため，全て 1 プロセスで計算を行った．表 5 に，剛性行列 BCSSTK13，BCSSTK14，BCSSTK15，BCSSTK16，BCSSTK17，BCSSTK18，BCSSTK21，BCSSTK24，BCSSTK25，S1RMQ4M1，S2RMQ4M1，CT20STIF，SHELL-A の解析結果を示す．図 43 に，剛性行列 BCSSTK24 の相対残差履歴を示す．図 44 に，剛性行列 CT20STIF の相対残差履歴を示す．図 45 に，剛性行列 SHELL-A の相対残差履歴を示す．ここで，Precond. は計算行列に適用した前処理行列，Option の係数 α は ILU(0) と ILU(1) の対角要素の修正係数，閾値 *toldd* は IRIF(0) と IRIF(1) のベクトル $\boldsymbol{z}_j^{(i-1)}$ の更新判定閾値，Condition number は推定した前処理適用後の条件数，Iteration は反復回数，Precond. time は前処理生成に要した時間 [sec]，CG time は CG 法の反復に要した時間 [sec]，Total time はこれらの合計計算時間 [sec] である．Iteration に記載のないものは未収束を意味する．それぞれの計算で最も短かった計算時間を記載した．

表 5 より，剛性行列 BCSSTK24，CT20STIF など多くの例題では，IRIF(0)，IRIF(1) とも，それぞれ ILU(0)，ILU(1) に対して反復回数の改善が見られた．推定した前処理適用後の条件数は，多くの例題で収束に必要な反復回数と正の相関があり，共役勾配法の収束のしやすさの指標となりえる．

剛性行列 BCSSTK24 の計算では，図 43 のように，IRIF(1) の相対残差履歴に RIF 系

前処理の特徴である急激な相対残差の減少がみられた．

剛性行列 CT20STIF の計算では，図 44 のように，1 段のフィルインを考慮した ILU(1) の前処理能力が ILU(0) より弱くなる一方，IRIF(0) と IRIF(1) は，ILU(0) と ILU(1) に比べ強い前処理能力を有することがわかった．

剛性行列 SHELL-A の計算では，図 45 のように，IRIF(0) を ILU(0) と比較すると前処理能力の性能が劣るものの，IRIF(1) は ILU(1) よりも高い前処理性能を有することがわかった．特に，IRIF(0) と IRIF(1) を比較すると反復回数の大きな改善が見られた．

IRIF(0) と IRIF(1) は，ILU(0) と ILU(1) に比べ強い前処理能力を有し，共役勾配法の反復に必要な計算時間を削減できることが明らかになったものの，前処理生成に演算量 $O(n^2)$ が必要であるため，前処理生成時間は ILU(0) と ILU(1) に比べ大きく増大した．

図 46 に，剛性行列 BCSSTK18 の計算における RIF(1) 前処理生成時の閾値 $toldd$ が，反復回数に与える影響を示す．閾値を設定しない場合 (閾値 $toldd = 0.0$) のとき反復回数は 364 回であり，閾値 $toldd = 0.1$ のとき反復回数は 310 回で，最小となった．閾値 $toldd$ を適当に変化させることで，収束回数が減少する場合があるが，閾値 $toldd$ を大きく設定しすぎると，本来の前処理性能が失われ，反復回数は大きく増加する．

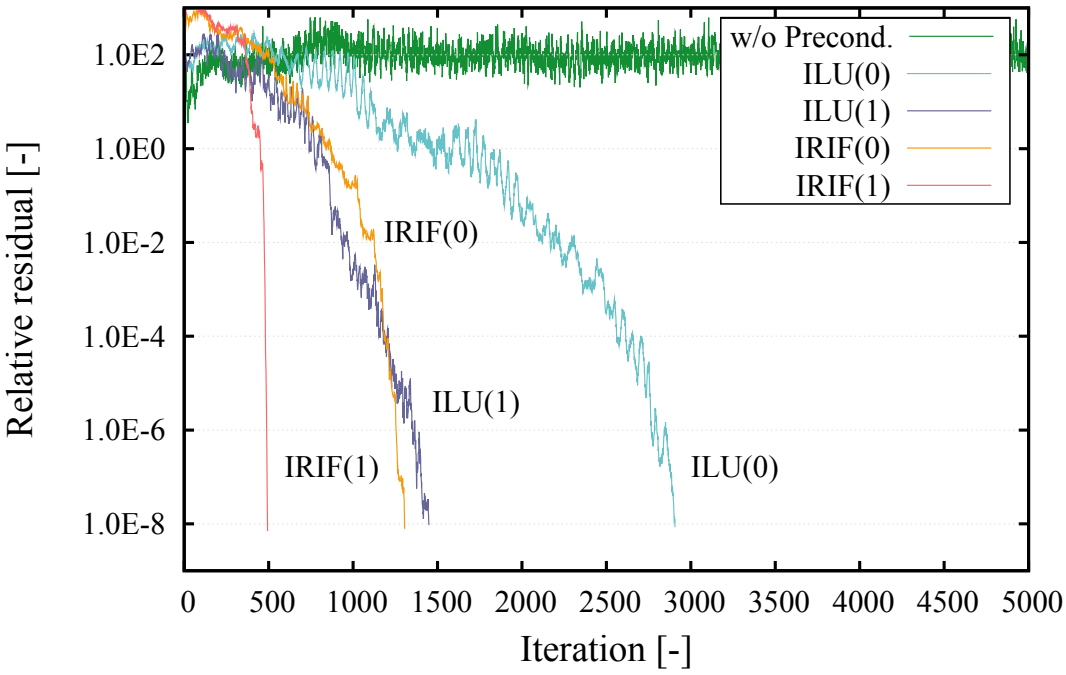


Fig. 43 Convergence histories in Problem 1 (BCSSTK24)

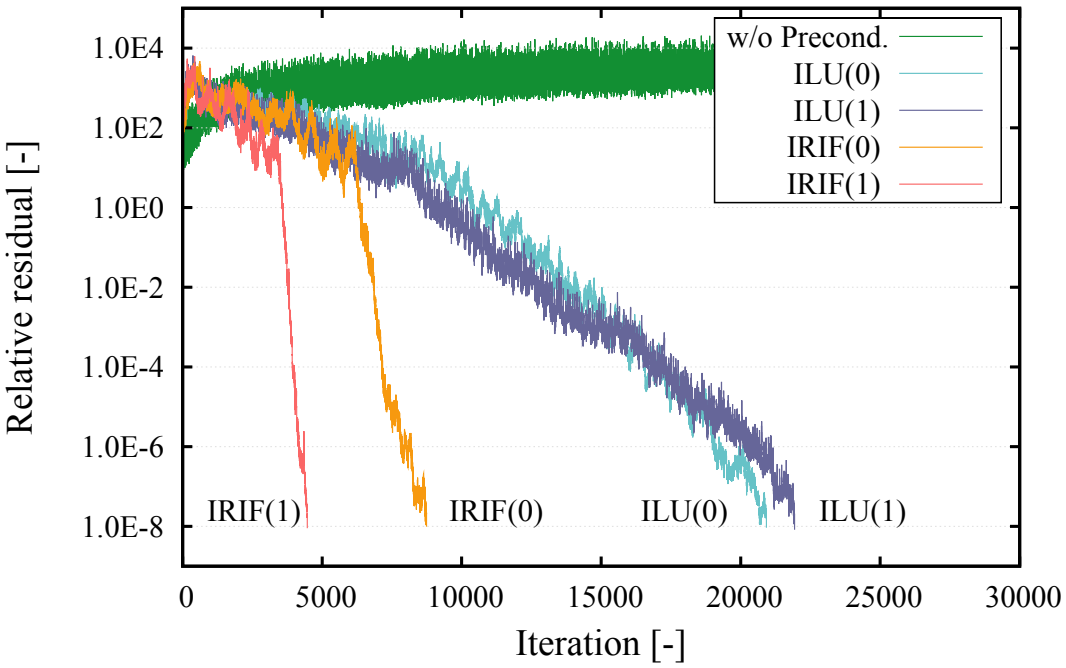


Fig. 44 Convergence histories in Problem 1 (CT20STIF)

Table 5 Numerical results of preconditioned CG methods in Problem 1

Matrix	Precond.	Option (α, $toldd$)	Iteration	Condition number	Precond. time [sec]	CG time [sec]	Total time [sec]
BCSSTK13	None	-	-	3.08E+12	-	-	-
	ILU(0)	-	485	5.48E+04	0.067	0.99	1.06
	ILU(1)	1.1	357	3.48E+04	0.138	1.06	1.20
	IRIF(0)	0.01	385	4.53E+04	1.82	0.776	2.60
	IRIF(1)	0	300	2.39E+04	2.87	0.893	2.76
BCSSTK14	None	-	15775	1.19E+10	0	12.9	12.9
	ILU(0)	-	85	2.06E+02	0.0159	0.0884	0.204
	ILU(1)	-	41	8.58E+01	0.0288	0.0495	0.0783
	IRIF(0)	0	65	1.23E+02	0.8	0.0654	0.865
	IRIF(1)	0.01	52	6.86E+01	0.888	0.0611	0.949
BCSSTK15	None	-	24877	6.54E+09	0	44.1	44.1
	ILU(0)	-	207	6.30E+03	0.0577	0.558	0.616
	ILU(1)	-	98	1.81E+03	0.108	0.316	0.424
	IRIF(0)	0	150	4.15E+03	4.69	0.396	5.09
	IRIF(1)	0	123	2.67E+03	5.43	0.393	5.82
BCSSTK16	None	-	502	4.94E+09	0	0.981	0.981
	ILU(0)	-	62	5.84E+01	0.0904	0.232	0.322
	ILU(1)	-	31	1.68E+01	0.182	0.158	0.34
	IRIF(0)	0	48	3.92E+01	8.11	0.179	8.29
	IRIF(1)	0	36	2.08E+01	10.6	0.186	10.8
BCSSTK17	None	-	25793	1.30E+10	0	107	107
	ILU(0)	-	790	9.45E+04	0.214	5.74	5.95
	ILU(1)	1.2	582	6.02E+04	0.394	4.94	5.33
	IRIF(0)	0	581	4.93E+04	34.2	4.15	38.4
	IRIF(1)	0	531	2.81E+04	39.2	4.5	43.7
BCSSTK18	None	-	-	1.62E+11	-	-	-
	ILU(0)	-	670	7.01E+03	0.225	5.01	5.24
	ILU(1)	1.1	400	2.32E+03	0.491	4.28	4.77
	IRIF(0)	0	445	4.67E+03	37.8	3.23	41.0
	IRIF(1)	0.01	310	2.56E+03	52.7	3.36	56.1

Matrix	Precond.	Option (α, $toldd$)	Iteration	Condition number	Precond. Time [sec]	CG time [sec]	Total time [sec]
BCSSTK21	None	-	11933	1.76E+07	0	12.8	12.8
	ILU(0)	-	301	6.75E+03	0.0136	0.456	0.470
	ILU(1)	1.1	250	4.47E+03	0.0398	0.443	0.483
	IRIF(0)	0	181	2.09E+03	2.14	0.264	2.40
	IRIF(1)	0	133	1.10E+03	2.49	0.229	2.72
BCSSTK24	None	-	-	2.24E+11	-	-	-
	ILU(0)	-	2906	1.32E+06	0.078	8.61	8.69
	ILU(1)	1.1	1449	2.02E+05	0.151	5.79	5.94
	IRIF(0)	0.01	1305	4.56E+05	4.67	3.78	8.45
	IRIF(1)	0	494	1.25E+05	7.81	1.96	9.77
BCSSTK25	None	-	-	4.12E+12	-	-	-
	ILU(0)	-	4319	6.77E+06	0.3	41.6	41.9
	ILU(1)	1.1	2654	2.63E+06	0.689	27.9	28.6
	IRIF(0)	0	4171	6.34E+07	62.9	39.9	103
	IRIF(1)	0.01	2282	3.82E+06	87.6	33.1	121
S1RMQ4M1	None	-	7557	1.81E+06	0	15.9	15.9
	ILU(0)	-	172	1.96E+04	0.0865	0.639	0.726
	ILU(1)	-	85	6.01E+03	0.153	0.367	0.520
	IRIF(0)	0	145	1.44E+04	8.82	0.532	9.35
	IRIF(1)	0	133	1.15E+04	10.1	0.575	10.7
S2RMQ4M1	None	-	24316	1.69E+08	0	58.2	58.2
	ILU(0)	-	248	6.78E+05	0.0865	0.921	1.01
	ILU(1)	-	104	4.01E+05	0.153	0.45	0.603
	IRIF(0)	0	187	4.75E+05	8.83	0.686	9.52
	IRIF(1)	0	173	3.24E+05	10.1	0.739	10.8
CT20STIF	None	-	-	2.41E+13	-	-	-
	ILU(0)	-	20924	3.25E+07	3.62	1079	1082
	ILU(1)	1.1	21931	3.82E+07	7.88	1854	1861
	IRIF(0)	0.01	8743	6.83E+07	1210	443	1653
	IRIF(1)	0.01	4468	9.38E+06	1821	379	2200
SHELL-A	None	-	-	3.58E+09	-	-	-
	ILU(0)	-	8409	2.79E+07	0.088	32.5	32.6
	ILU(1)	1.3	7228	1.71E+07	0.212	42.5	42.7
	IRIF(0)	0.01	9538	3.06E+07	9.58	36.6	46.2
	IRIF(1)	0.1	5351	6.98E+06	15.3	31.1	46.4

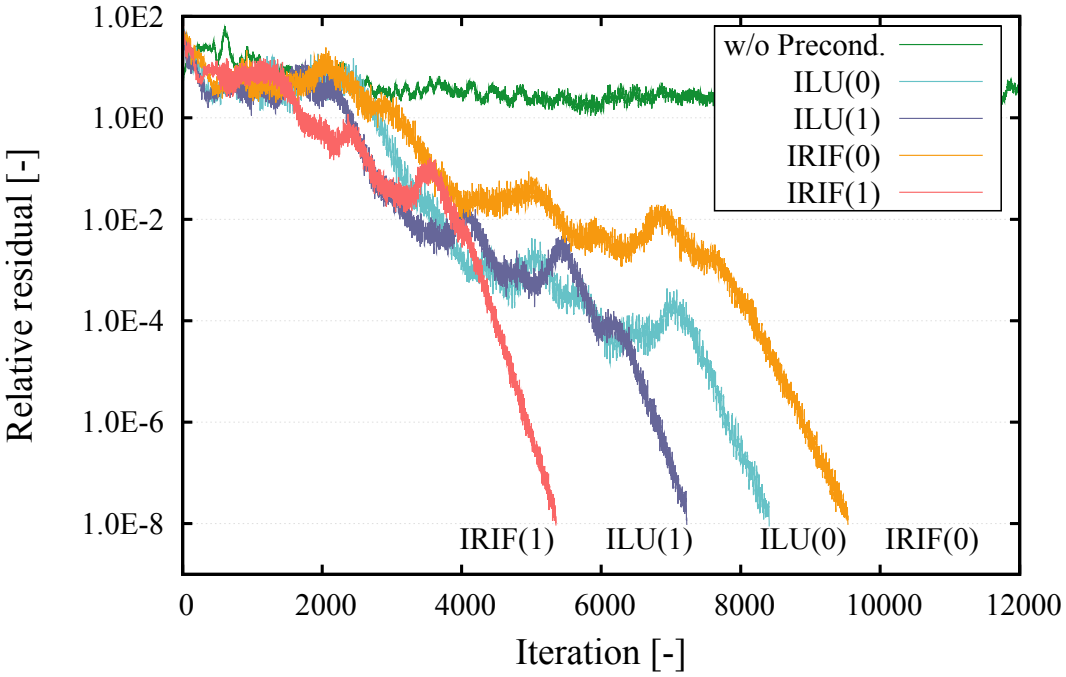


Fig. 45 Convergence histories in Problem 1 (SHELL-A)

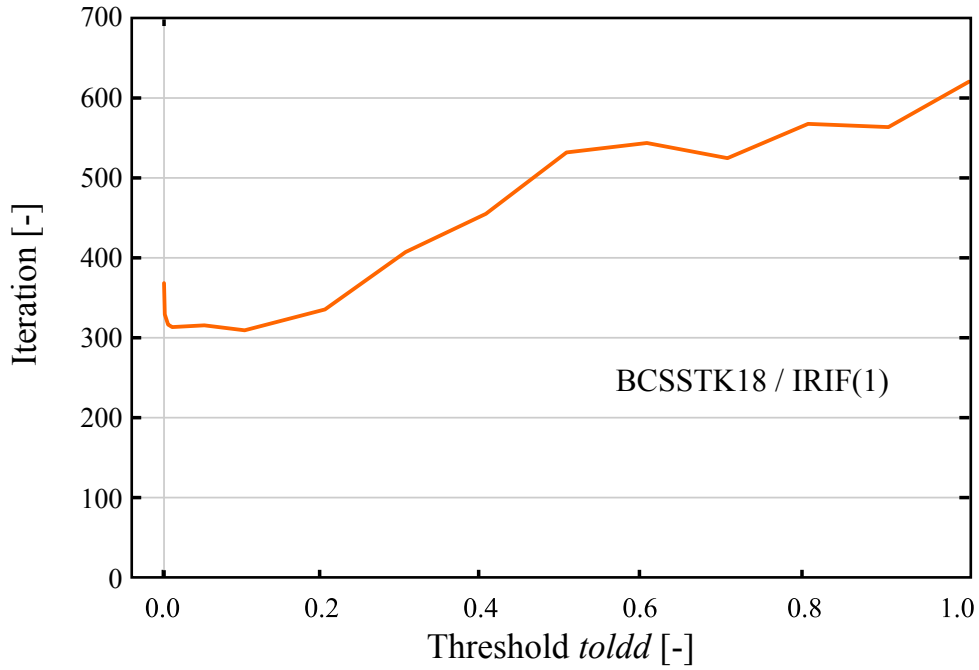


Fig. 46 Relationship between Threshold *toldd* and CG iteration in Problem 1 (BCSSTK18)

7.4 数値例題 2

数値例題 2 は，前処理性能の並列分割数による依存性を評価するものである．計算には，東京大学情報基盤センター内の FX10 を用いた．図 29 に示したモデルのメッシュをより細かく生成して得た剛性行列 SHELL-B に対して，領域分割数を 1，2，4，8，16 と変化させて計算を行った．使用ノード数は 1 で，MPI は OpenMPI[42] を用いた．SHELL-B は，節点数 10,848，要素数 10,000，シェル厚さ 0.01mm であり要素は MITC4[19] を使用した．一要素の最大長さは 5mm で，このときシェル厚さとの比は 500 となる．表 6 に SHELL-B の性質を示す．図 47 に SHELL-B の非零プロファイルを示す．図 48 に並列数 16 の場合の領域分割を示す．

表 7 に，SHELL-B の前処理として Localized ILU(1) と Localized IRIF(1) を適用させ並列計算を行った解析結果を示す．図 49 に，Localized ILU(1) と Localized IRIF(1) の前処理生成時間と CG 反復時間の合計時間の増速率を示す．ここで，PE は並列プロセス数，Accel. は 1 プロセスでの計算時間を基準とした加速率である．

数値例題 2 では，表 7 のように，解の収束に必要な反復回数は，並列プロセス数の増加に従って増加した．特に，Localized IRIF(1) に比べ，Localized ILU(1) は反復回数増加が顕著である．局所化した前処理は，並列プロセス数の増加に伴って分割領域間に跨るフィルインも増加するため，これらフィルインが無視されることで，前処理行列生成時の不安定化につながる．Localized IRIF(1) は前処理行列の対角項が負になることを避けることができるため，Localized ILU(1) に見られる分解の不安定性に比べ，無視されるフィルイン増加の影響を少なくできると考えられる．

Localized IRIF(1) の前処理生成時間の増速率はおよそ並列プロセス数の 2 乗に沿うため，並列計算によって前処理生成の大きな計算高速化が実現された．そのため，並列プロセス数が増加するに従って，並列効果によって計算時間の多くを占めていた前処理生成時間の影響が緩和され，合計計算時間とその増速率で Localized IRIF(1) が Localized ILU(1) に比べ，優位となる結果が得られた．計算例のすべての場合，Localized IRIF(1) の合計計算時間が Localized ILU(1) の合計計算時間より早くなった．

Table 6　Matrix information in Problem 2

Name	N	Non-zero element	Sparsity [%]	Condition number
SHELL-B	65088	3331728	0.0786	9.49E+09

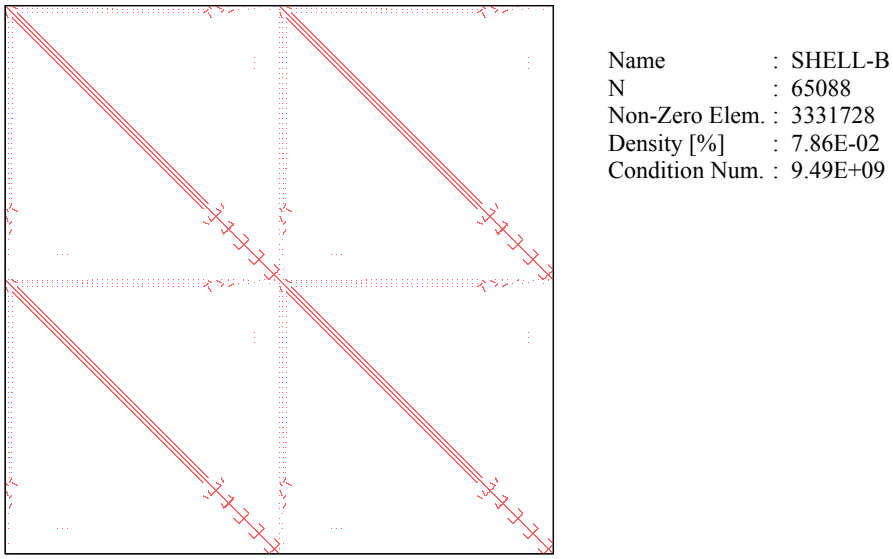


Fig. 47　Non-zero profile of SHELL-B

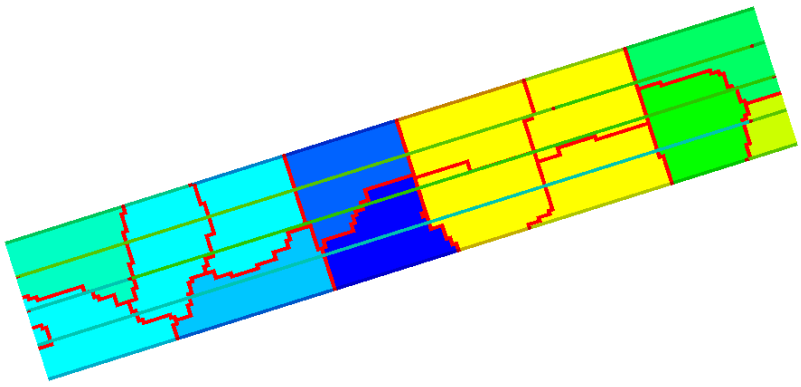


Fig. 48　Analysis mesh of SHELL-B in Problem 2 (domain decomposition into 16PE)

Table 7 Numerical results of parallel preconditioned CG methods in Problem 2

Matrix	Precond.	Option (α, $toldd$)	PE	Iteration	Condition number	Precond. time [sec]	CG time [sec]	Total time [sec]	Accel.
SHELL-B	Localized	1.5	1	144343	3.52E+10	2.72	2740	2743	1
	ILU(1)	1.5	2	229277	6.53E+10	0.853	2294	2295	1.2
		1.5	4	226431	6.43E+10	0.315	1229	1229	2.23
		1.5	8	229507	6.40E+10	0.136	679	679	4.04
		1.5	16	245880	6.82E+10	0.0708	459	459	5.98
	Localized	0.08	1	114751	1.40E+10	554	2136	2690	1
	IRIF(1)	0.08	2	140205	1.40E+10	141	1364	1505	1.79
		0.08	4	142978	1.40E+10	37.3	754	791	3.40
		0.08	8	145151	1.39E+10	9.45	419	428	6.29
		0.08	16	152768	1.41E+10	2.5	284	289	9.31

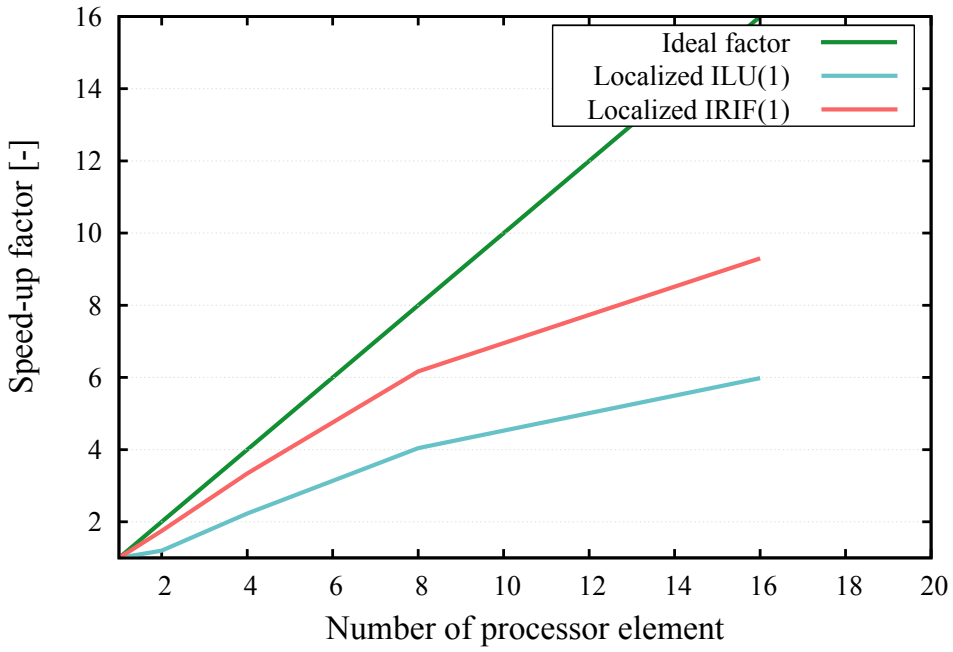


Fig. 49 Speed-up factors obtained by executing the PCG algorithms in Problem 2 (IC(1) vs IRIF(1))

7.5 数値例題 3

数値例題 3 は，単精度浮動小数点数で計算された前処理行列の性能を評価する．

混合精度演算では，単精度演算浮動小数点数による誤差の影響で，共役勾配法反復中の前処理の逆行列 $\boldsymbol{M}^{-1} = (\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^t)^{-1}$ を適用する方法によって演算結果が変化すると考えられる．そこで，反復中で前処理の逆行列 $\boldsymbol{M}^{-1} = \boldsymbol{L}^{-t}\boldsymbol{D}^{-1}\boldsymbol{L}^{-1}$ を直接計算する方法をプログラム 1，新たな下三角行列 $\boldsymbol{W} = \boldsymbol{D}\boldsymbol{L}^t$ をあらかじめ計算しておき，反復中では前処理の逆行列 $\boldsymbol{M}^{-1} = \boldsymbol{W}^{-1}\boldsymbol{L}^{-1}$ を計算する方法をプログラム 2 として，計算方法の異なるふたつのプログラムを検証した（これまでに述べた数値例題 1 と数値例題 2 では，プログラム 1 の方法を採用している）．

計算には，東京大学情報基盤センター内の FX10 を用いた．数値例題 2 で用いた剛性行列 SHELL-B に対して，単精度によって計算された Localized ILU(1) と Localized IRIF(1) 前処理を適用し，領域分割数を 1，2，4，8，16 と変化させて計算した．

表 8 に，単精度によって計算された Localized ILU(1) と Localized IRIF(1) を適用させ，並列計算を行った解析結果を示す．比較のために，倍精度によって前処理を計算した場合（数値例題 2 の表 7）の解析結果を合わせて記載した．ここで特に，単精度で計算した前処理には Single precision，プログラム 2 の方法を用いて計算した Localized IRIF(1) には Program 2 の記載をしている．

数値例題 3 の結果，単精度と倍精度の前処理を比較した場合，ほぼ全ての計算例で，単精度の前処理を用いると収束に必要な反復回数が増加した．一方，計算時間を比較した場合，単精度の前処理を用いた方が，合計計算時間が短くなるという結果が得られた．

図 50 に，単精度の前処理を用いることで増加した反復回数を示す．図 51 に，単精度と倍精度によって前処理行列が計算された，Localized ILU(1) と Localized IRIF(1)(プログラム 1) の相対残差履歴を示す．Localized ILU(1) とプログラム 2 の方法を用いて計算した Localized IRIF(1) は，混合精度演算の影響で反復回数が大きく増加している．一方，プログラム 1 の方法を用いて計算した Localized IRIF(1) は，混合精度演算の影響を大きく抑えることができ，並列数が増えても反復回数の変化は非常に少ない．ここで，単精度 16 並列の計算では反復回数が減少しているが，特異な計算事例であると考えられる．

今回用いた計算機環境では，CPU の特徴上，倍精度と単精度に演算性能に大きな差は

ないにも関わらず，単精度を用いた場合の方が，倍精度を用いた場合に比べ，収束までに必要な計算時間は短くなった．これはメモリ利用の観点から，単精度の方がキャッシュメモリに多くのデータ配列を格納することができるために，計算時間短縮に優位な影響を与えたものだと考えられる．

単精度で計算した前処理は，倍精度に比べ有効桁数が小さいことから，浮動小数点数演算による誤差が前処理行列生成時の不安定化につながる．提案手法は，前処理生成中のドロッピングによる影響がある場合にも安定に分解できたように，浮動小数点数の情報落ちや桁落ちによる値の欠落のような，浮動小数点数演算が引き起こす誤差による影響がある場合でも，安定に分解できると考えられる．

Table 8 Numerical results of parallel preconditioned CG methods in Problem 3

Matrix	Precond.	Option (α, *toldd*)	PE	Iteration	Condition number	Precond. time [sec]	CG time [sec]	Total time [sec]
SHELL-B	Localized	1.5	1	145265	3.52E+10	2.69	2703	2706
	ILU(1)	1.5	2	236442	6.54E+10	0.84	2317	2318
	Single	1.5	4	232386	6.43E+10	0.306	1231	1231
	precision	1.5	8	241825	6.55E+10	0.125	693	693
		1.5	16	239214	6.72E+10	0.0668	399	400
	Localized	1.5	1	144343	3.52E+10	2.72	2740	2743
	ILU(1)	1.5	2	229277	6.53E+10	0.853	2294	2295
		1.5	4	226431	6.43E+10	0.315	1229	1229
		1.5	8	229507	6.40E+10	0.136	679	679
		1.5	16	245880	6.82E+10	0.0708	459	459
SHELL-B	Localized	0.08	1	115353	1.40E+10	586	2088	2674
	IRIF(1)	0.08	2	140451	1.40E+10	149	1330	1479
	Single	0.08	4	143220	1.40E+10	38.8	730	769
	precision	0.08	8	145154	1.39E+10	9.88	298	308
		0.08	16	152784	1.41E+10	2.44	244	246
	Localized	0.08	1	114751	1.40E+10	554	2136	2690
	IRIF(1)	0.08	2	140205	1.40E+10	141	1364	1505
		0.08	4	142978	1.40E+10	37.3	754	791
		0.08	8	145151	1.39E+10	9.45	419	428
		0.08	16	152768	1.41E+10	2.5	284	289
SHELL-B	Localized	0.08	1	116531	1.40E+10	570	2110	2680
	IRIF(1)	0.08	2	145754	1.40E+10	145	1389	1534
	Single	0.08	4	147956	1.40E+10	37.8	763	800
	precision	0.08	8	154742	1.40E+10	9.62	433	443
	by Program 2	0.08	16	156077	1.41E+10	2.39	257	260
	Localized	0.08	1	114394	1.40E+10	542	2117	2659
	IRIF(1)	0.08	2	140182	1.40E+10	139	1377	1516
	by Program 2	0.08	4	143277	1.40E+10	36.5	759	796
		0.08	8	145515	1.39E+10	9.24	422	431
		0.08	16	152506	1.41E+10	2.46	283	286

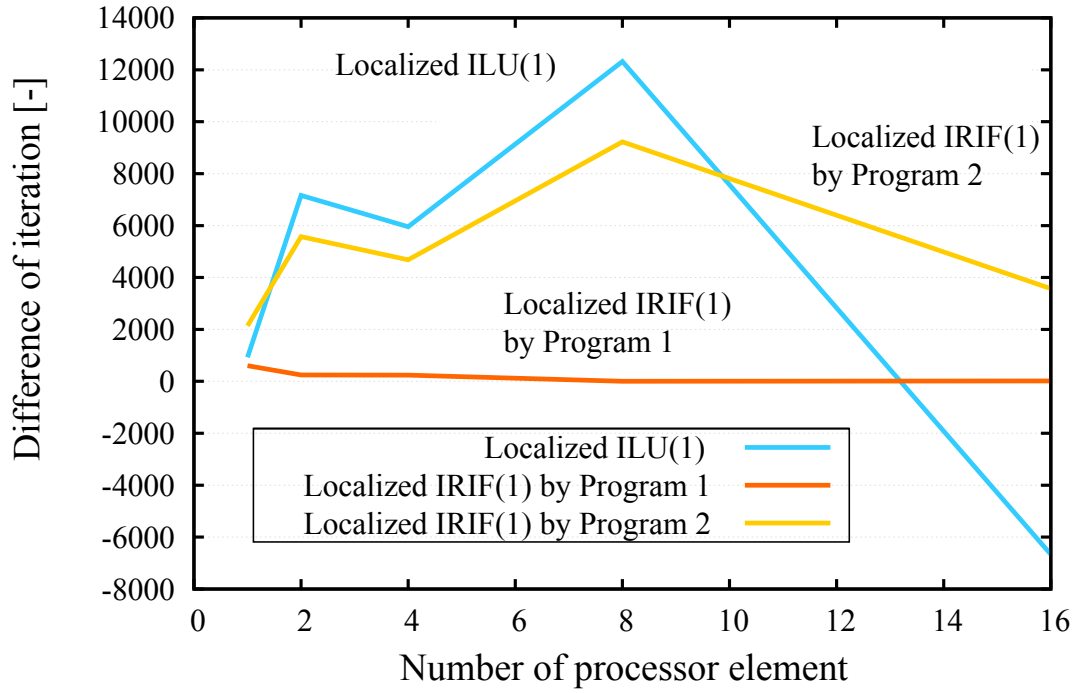


Fig. 50 Relationship between number of processor element and difference of iteration in Problem 3

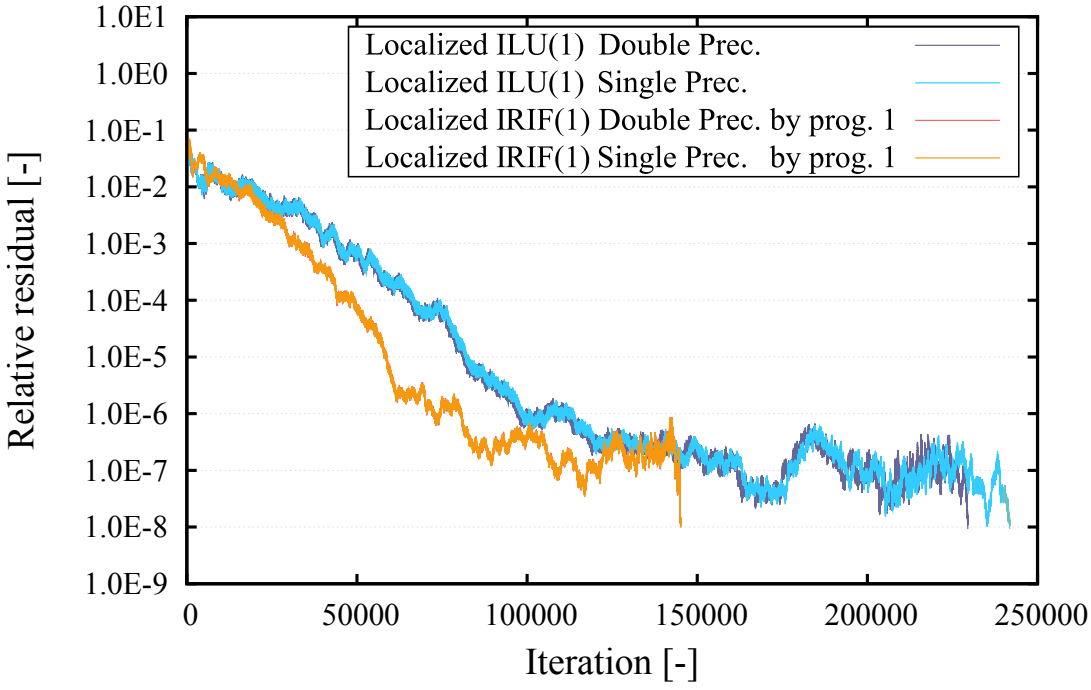


Fig. 51 Convergence histories in Problem 3 (SHELL-B, PE8)

7.6 数値例題 4

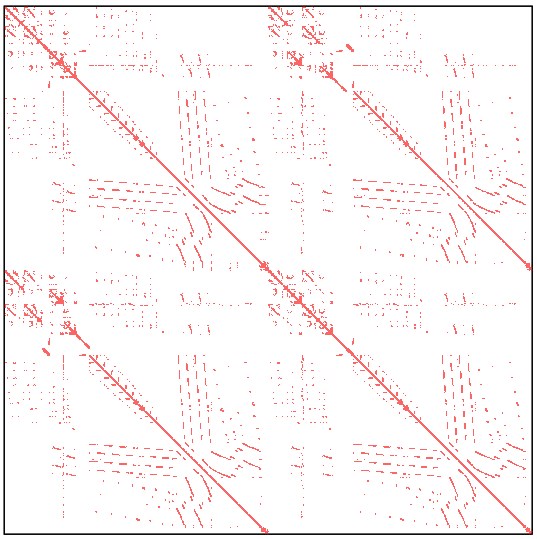
数値例 4 は，実問題への適用事例として，エネルギ関連重要施設の数値解析を評価するものである．実問題への適用は，提案手法の前処理性能を評価する上で非常に重要である．図 53 と図 54 に，エネルギ関連重要施設の解析モデルを示す．このモデルは，節点数 25,668，要素数 15,270，次元数 77,004，非零要素数 4,188,672，非零要素比 0.0706%，条件数 1.82×10^9 である．ソリッド要素，シェル要素，梁要素，トラス要素が混在したモデルである．シェル要素は直交異方性を考慮した積層シェルを FrontISTR に実装し使用した（詳細は付録に記載した）[43]．施設最下部基礎部分のシェル要素と施設壁部分は多点拘束 (Multiple Point Constraint，MPC) を用いて拘束されている．同様に，施設屋根部分の梁トラス構造と施設壁部分も MPC を用いて拘束されている．図 52 に，この問題の非零プロファイルを示す．境界条件として，施設の基礎部分を完全固定し，施設全体に重力を想定した体積力を横方向に負荷した．

さらに本解析では，上記のモデルに 1 回のリファインを行ったものを考慮する．図 55 と図 56 に，1 回のリファインを行ったエネルギ関連重要施設の解析モデルを示す．リファイン後のモデルは，節点数 103,490，要素数 56,026，次元数 310,470，非零要素数 16,740,324，非零要素比 0.0174%，条件数 3.56×10^9 である．

このふたつの問題に対して，Localized ILU(1) と Localized IRIF(1)，混合精度 Localized IRIF(1) の前処理を適用し，共役勾配法の反復収束性と計算時間を比較する．収束条件は，相対残差に対して収束判定値 $\varepsilon = 1.0 \times 10^{-8}$ とした．計算機は，東京大学情報基盤センター内の FX10 を利用した．リファインしないモデルは領域分割数を 1，2，4，8，16 と変化させ，1 回のリファインを行ったモデルは領域分割数を 4，8，16，32，64 と変化させて計算を行った．

表 9 に，リファインをしない場合の数値解析の結果を示す．表 10 に，1 回のリファインを行った場合の数値解析の結果を示す．

この問題では，全ての計算例において，前処理に混合精度 Localized IRIF(1) を用いた場合，Localized IRIF(1) を用いた場合，Localized ILU(1) を用いた場合の順に，合計計算時間が短くなった．リファインをしない場合，混合精度 Localized IRIF(1) は Localized ILU(1) に対し，8 並列のとき合計計算時間を約 60% に短縮できた．1 回のリファインを



Name : Energy related installation
N : 77004
Non-Zero Elem. : 4188672
Density [%] : 7.06E-02
Condition Num. : 1.82E+09

Fig. 52 Non-zero profile of energy related installation

を行った場合，混合精度 Localized IRIF(1) は Localized ILU(1) に対し，8 並列のとき合計計算時間を約 70% に短縮できた．リファインによって新しく生成された節点は既存の節点に続いて定義されるため，非零プロファイルが元のものに比べ複雑になる．複雑な非零プロファイルは，フィルインを考慮した前処理に少なからぬ影響を与えると考えられる．この影響を低減するには節点情報のオーダリングが挙げられるが，オーダリングが収束に与える影響について検証することは，今後の課題としたい．

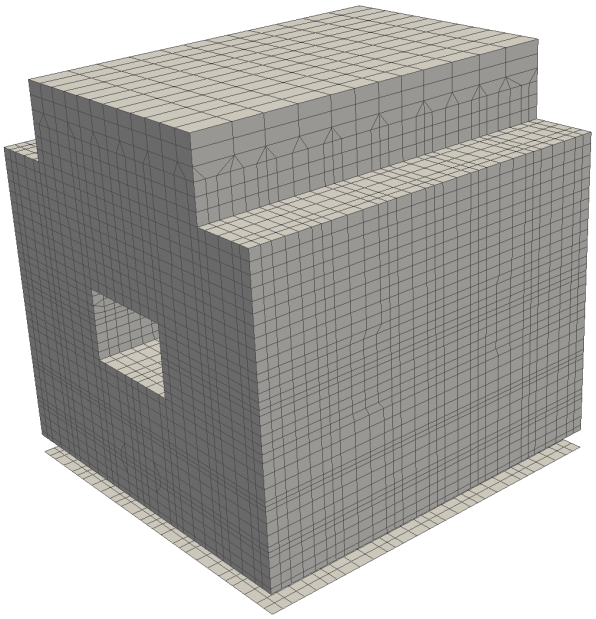


Fig. 53　Analysis mesh of energy related installation

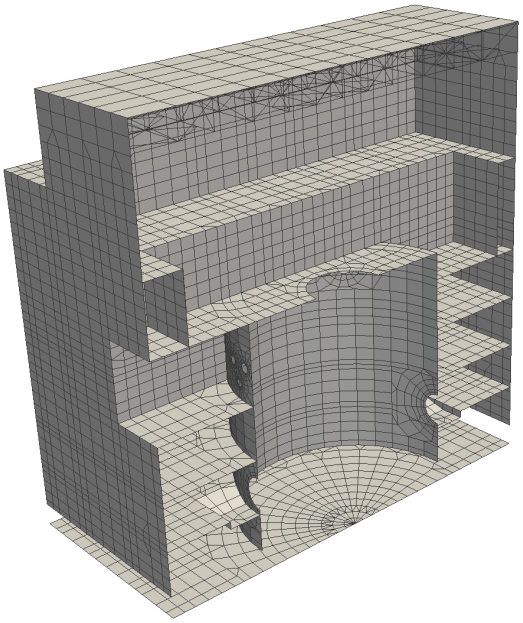


Fig. 54　Analysis mesh of energy related installation (cross section)

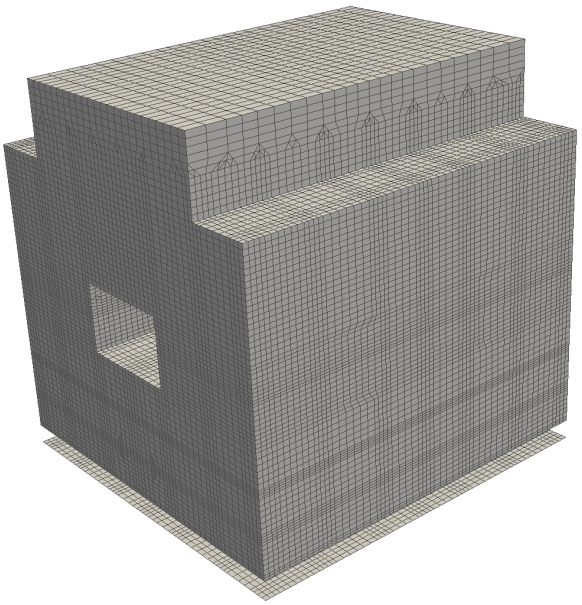


Fig. 55　Analysis mesh of energy related installation (Refine=1)

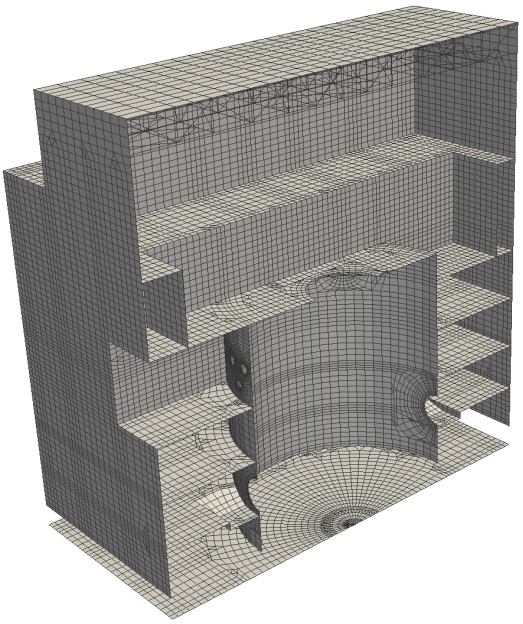


Fig. 56　Analysis mesh of energy related installation (cross section, Refine=1)

Table 9 Numerical results of preconditioned CG methods in Problem of energy related installation (Refine = 0)

Matrix	Precond.	Option (α, *toldd*)	PE	Iteration	Precond. time [sec]	CG time [sec]	Total time [sec]
Energy	Localized	1.5	1	166621	3.84	4259	4263
related	ILU(1)	1.5	2	168009	1.27	2278	2279
installation		1.5	4	173883	0.478	1236	1236
Refine=0		1.5	8	176477	0.208	695	695
		1.5	16	175602	0.107	433	433
	Localized	0.01	1	94727	940	2372	3312
	IRIF(1)	0.01	2	101239	237	1350	1587
		0.01	4	107128	60	723	783
		0.01	8	112900	15.1	437	452
		0.01	16	119512	3.92	299	303
	Localized	0.01	1	94788	957	2269	3226
	IRIF(1)	0.01	2	101354	242	1259	1501
	Single prec.	0.01	4	106863	61.2	688	749
		0.01	8	112578	15.4	392	407
		0.01	16	119892	3.98	258	262

Table 10 Numerical results of preconditioned CG methods in Problem of energy related installation (Refine = 1)

Matrix	Precond.	Option (α, *toldd*)	PE	Iteration	Precond. time [sec]	CG time [sec]	Total time [sec]
Energy	Localized	1.5	4	323456	4.33	8342	8346
related	ILU(1)	1.5	8	324913	1.48	4528	4529
installation		1.5	16	319730	0.599	2875	2876
Refine=1		1.5	32	322207	0.265	1548	1548
		1.5	64	321639	0.123	929	930
	Localized	0.01	4	223235	906	5645	6551
	IRIF(1)	0.01	8	230670	248	3264	3512
		0.01	16	270675	69.8	2468	2538
		0.01	32	276776	18.1	1352	1370
		0.01	64	297093	5.11	859	864
	Localized	0.01	4	224764	941	5246	6187
	IRIF(1)	0.01	8	231847	257	3010	3267
	Single prec.	0.01	16	272494	69.4	2133	2202
		0.01	32	276666	18.3	1148	1166
		0.01	64	298450	5.21	766	771

7.7 結言

この章では，提案手法と既存の手法を数値例によって比較した．まず，計算条件と計算機環境について述べる．

数値例題 1 では，提案手法の逐次計算性能を評価した．提案手法の IRIF(0) と IRIF(1) は，ILU(0) と ILU(1) に比べ強い前処理能力を有し，共役勾配法の反復に必要な計算時間を削減できるという結果が得られた．

数値例題 2 では，前処理性能の並列分割数による依存性を評価した．提案手法の Localized IRIF(1) は，並列プロセス数を多くするに従って，並列効果によって計算時間の多くを占めていた前処理生成時間の影響が緩和され，合計計算時間とその増速率で Localized ILU(1) に比べ優位な結果が得られた．

数値例題 3 では，単精度浮動小数点数で計算された前処理行列の性能を評価した．提案手法の単精度で計算された Localized IRIF(1) は，混合精度演算の低い演算精度による誤差影響を大きく抑えることができ，メモリ利用の観点から収束までに必要な計算時間を短縮できるという結果が得られた．

数値例題 4 では，実問題への適用事例として，エネルギ関連重要施設の数値解析について述べた．提案手法は，様々な要素が混在し MPC で拘束されたエネルギ関連重要施設の数値解析において，Localized ILU(1) に比べ合計計算時間を最大約 60% に短縮できるという結果が得られた．

第 8 章　結論

本研究では，大規模並列有限要素法を用いた構造解析における，シェル要素や薄い板状構造から得られるような大きな条件数をもつ問題に対して，並列反復法を適用することを目的とした．大きな条件数をもつ大規模問題を並列直接法ソルバを用いて解く場合，計算時間やメモリ使用量などの面から求解が困難である問題に直面しており，並列反復法ソルバを用いて解けることは非常に重要である．以下に，各章の内容を簡潔にまとめる．

第 2 章では，有限要素法の線形弾性体の境界値問題について述べた．次に，一般に広く用いられる 8 節点六面体ソリッド要素と，4 節点平面シェル要素の定式化について述べた．

第 3 章では，有限要素法で解くべき連立一次方程式の線形ソルバである，直接法ソルバと反復法ソルバについて述べた．さらに，解くべき問題の性質を表す条件数について述べ，反復法ソルバを用いた条件数の推定方法について述べた．大規模な構造解析を並列有限要素法を用いて解く場合，並列反復法の適用が必要であることを述べた．

第 4 章では，本研究の開発基盤として利用した，大規模並列有限要素解析ソフトウェア FrontISTR と HEC-MW について述べた．さらに，大規模並列計算に必要なパーティショナ，リファイナ，ソルバの並列計算と行列の格納方法について述べた．

第 5 章では，共役勾配法の前処理手法として，対角スケーリング前処理，対称逐次過緩和前処理，不完全 LU 分解前処理について述べた．続いて，堅固な反復法前処理として注目される A-直交過程に基づく近似逆行列分解系前処理として，近似逆行列分解前処理，安定化近似逆行列分解前処理，ロバスト不完全分解 (RIF) 前処理，改良型安定化近似逆行列分解前処理，改良型ロバスト不完全分解前処理について述べた．

第 6 章では，堅固な反復法前処理として注目される RIF 前処理を，領域分割に基づく並列有限要素法に適用することについて述べた．RIF 前処理は，前処理行列の非零要素プロファイルを前処理生成中に更新して定めるため，大きな計算コストがかかる．そこで，従来の RIF 前処理に対して，係数行列の非零要素プロファイルを元にした，フィルインを考慮せず局所並列化した IRIF(0) 前処理と，フィルインを考慮し局所並列化した IRIF(1) 前処理を提案した．提案手法は，前処理生成より前に前処理行列の非零要素プロファイルを定めることができる点で優れている．また，単精度と倍精度を用いた混合精度演算に着

目し，提案手法を単精度を用いて計算する，混合精度演算を用いた前処理つき共役勾配法を提案した．提案した前処理生成手法は，係数行列の直交過程を用いて前処理行列の対角項が常に正の値となるように分解を進めるため，局所並列化による並列領域間のフィルイン無視や，低い精度の浮動小数点数を用いた計算でも，安定した前処理を生成できる．

第 7 章では，数値例を示した．数値例題 1 では逐次計算を行った場合，数値例題 2 では並列計算を行った場合，数値例題 3 では混合精度演算を用いて並列計算を行った場合について解析結果を示した．数値例題 4 では実問題としてエネルギ関連重要施設の構造解析に提案手法を適用し，その有効性について検討した．解析結果により，以下の結果を得た．

- IRIF(0) では ILU(0) の前処理より性能が劣る場合もあるものの，フィルインを考慮した IRIF(1) は IRIF(0) に比べ強い前処理能力を有し，ILU(1) と IRIF(1) を比較しても共役勾配法の収束向上が確認された．
- IRIF として提案されていた閾値 *toldd* によるベクトル更新判定が，IRIF(0) や IRIF(1) にも有効であることが確認された．
- 複数のプロセス間に跨る要素にフィルインを認めないことで並列化した Localized IRIF(1) は，計算時間の多くを占めていた前処理生成時間を並列効果により緩和できた．数値例により，Localized IRIF(1) が Localized ILU(1) に比べて，並列共役勾配法の収束までに必要な反復回数と計算時間で優位である結果が得られた．
- 混合精度演算を用いて計算した Localized IRIF(1) は，浮動小数点数演算による誤差の影響がある場合でも，安定した前処理能力をもつことが確認された．
- 実問題として設定したエネルギ関連重要施設の構造解析では，リファインをしない場合，混合精度 Localized IRIF(1) は Localized ILU(1) に対し，8 並列のとき合計計算時間を約 60% に短縮できた．

今後の展望として，CPU と GPU といった異なるアーキテクチャのプロセッサを統合して利用する，ヘテロジニアスな計算機環境において，高い並列計算性能を実現することが挙げられる．本研究での計算機では考慮しなかったが，GPU などのメニーコアアクセラレータのほとんどは，倍精度演算性能に比べて単精度演算性能が 2 倍以上高いため，提案手法のような混合精度演算でも安定して計算できる手法を，ヘテロ環境に適用させる意義は大きいと考える．

謝辞

本論文は，東京大学大学院新領域創成科学研究科人間環境学専攻修士課程における成果をまとめたものです．研究の遂行にあたり，多くの方々に多大なご指導，ご協力いただきましたことを，ここに厚く御礼申し上げます．

はじめに，指導教官であります新領域創成科学研究科人間環境学専攻 奥田洋司教授に深く感謝の意を表します．奥田教授には，研究生活における様々な面でご指導頂きました．

本論文に関しまして副査を務めて頂き，貴重なご教示を多数いただきました新領域創成科学研究科人間環境学専攻 小竹元基准教授に，ここに謹んで感謝の意を表します．

奥田教授とともに，研究生活，大学生活における様々な面でご指導，ご協力頂きました橋本学講師，北山健特任研究員，渡辺夏実秘書に感謝の意を表します．

本研究で用いた数値例題およびモデルの一部は，鹿島建設株式会社様にご提供いただいたものです．ここに厚く御礼申し上げます．

これまで研究生活をともにしてきました，奥田研究室の皆様に心から感謝いたします．

研究室の先輩として，親身に相談に乗ってくださり，有益な意見を多数いただきました後藤和哉氏と稲垣和久氏に心から感謝いたします．

研究室の同期として，これまでたくさんお世話になりました井原遊氏に心から感謝いたします．今後ともどうかよろしくお願いいたします．

最後に，これまで一貫して暖かく応援くださった両親と家族に心から感謝いたします．

本学博士課程へにおいては，研究生活，私生活とも今以上に力強く邁進する所存です．最後にこれまでご協力くださった皆様に感謝の気持ちを申し上げ，謝辞にかえさせていただきます．

参考文献

[1] Henry TY Yang, S Saigal, A Masud, and RK Kapania. A survey of recent shell finite elements. *International Journal for Numerical Methods in Engineering*, Vol. 47, No. 1-3, pp. 101–127, 2000.

[2] Iain S Duff, Albert Maurice Erisman, and John Ker Reid. *Direct methods for sparse matrices*. Clarendon Press Oxford, 1986.

[3] Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[4] Nocedal Jorge and J Wright Stephen. Numerical optimization. *Springerverlang, USA*, 1999.

[5] Michele Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, Vol. 182, No. 2, pp. 418–477, 2002.

[6] Michele Benzi, Carl D Meyer, and Miroslav Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, Vol. 17, No. 5, pp. 1135–1149, 1996.

[7] Michele Benzi, Jane K Cullum, and Miroslav Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, Vol. 22, No. 4, pp. 1318–1332, 2000.

[8] 池田優介, 藤野清次, 柿原正伸, 井上明彦. A-直交過程に基づく RIF 前処理の効率化について. 情報処理学会論文誌. コンピューティングシステム, Vol. 45, No. 6, pp. 95–104, 2004.

[9] Michele Benzi and Carl D Meyer. A direct projection method for sparse linear systems. *SIAM Journal on Scientific Computing*, Vol. 16, No. 5, pp. 1159–1176, 1995.

[10] Michele Benzi and Miroslav Tůma. A robust incomplete factorization preconditioner for positive definite matrices. *Numerical Linear Algebra with Applications*, Vol. 10, No. 5-6, pp. 385–400, 2003.

[11] 池田優介, 藤野清次. 二重ドロッピングによる安定化近似逆行列前処理の改良. 情報

処理学会論文誌. コンピューティングシステム, Vol. 45, No. 1, pp. 10–17, 2004.

[12] Harold Bailey David. High-precision software directory. `http://crd-legacy.lbl.gov/~dhbailey/mpdist/`. 2015 年 1 月 1 日閲覧.

[13] 小武守恒, 藤井昭宏, 長谷川秀彦, 西田晃. 反復法ライブラリ向け 4 倍精度演算の実装と SSE2 を用いた高速化. 情報処理学会論文誌. コンピューティングシステム, Vol. 1, No. 1, pp. 73–84, 2008.

[14] Julie Langou, Piotr Luszczek, Jakub Kurzak, Alfredo Buttari, and Jack Dongarra. Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pp. 50–50. IEEE, 2006.

[15] Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Piotr Luszczek, and Stanimir Tomov. Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy. *ACM Transactions on Mathematical Software (TOMS)*, Vol. 34, No. 4, p. 17, 2008.

[16] Serban Georgescu, Peter Chow, and Hiroshi Okuda. GPU acceleration for FEM-based structural analysis. *Archives of Computational Methods in Engineering*, Vol. 20, No. 2, pp. 111–121, 2013.

[17] Olgierd Cecil Zienkiewicz and Kenneth Morgan. 有限要素と近似. ワイリージャパン, 1984.

[18] 久田俊明, 野口裕久. 非線形有限要素法の基礎と応用. 丸善, 1995.

[19] Eduardo N Dvorkin and Klaus-Jürgen Bathe. A continuum mechanics based four-node shell element for general non-linear analysis. *Engineering computations*, Vol. 1, No. 1, pp. 77–88, 1984.

[20] Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[21] Thomas JR Hughes and F Brezzi. On drilling degrees of freedom. *Computer Methods in Applied Mechanics and Engineering*, Vol. 72, No. 1, pp. 105–121, 1989.

[22] G Pimpinelli. An assumed strain quadrilateral element with drilling degrees of freedom. *Finite elements in analysis and design*, Vol. 41, No. 3, pp. 267–283,

2004.

[23] Craig S Long, Sannelie Geyer, and Albert A Groenwold. A numerical study of the effect of penalty parameters for membrane elements with independent rotation fields and penalized equilibrium. *Finite elements in analysis and design*, Vol. 42, No. 8, pp. 757–765, 2006.

[24] Timothy A Davis. *Direct methods for sparse linear systems*, Vol. 2. Siam, 2006.

[25] Alan George and Joseph WH Liu. The evolution of the minimum degree ordering algorithm. *Siam review*, Vol. 31, No. 1, pp. 1–19, 1989.

[26] Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, Vol. 10, No. 2, pp. 345–363, 1973.

[27] MUMPS: a multifrontal massively parallel sparse direct solver. `http://mumps.enseeiht.fr/`. 2015 年 1 月 1 日閲覧.

[28] PARADISO 5.0.0 solver project. `http://www.pardiso-project.org/`. 2015 年 1 月 1 日閲覧.

[29] Watson sparse matrix package (WSMP). `http://researcher.watson.ibm.com/researcher/view_group.php?id=1426`. 2015 年 1 月 1 日閲覧.

[30] LAPACK - linear algebra package. `http://www.netlib.org/lapack/`. 2015 年 1 月 1 日閲覧.

[31] FrontISTR 研究会. `http://www.multi.k.u-tokyo.ac.jp/FrontISTR/`. 2015 年 1 月 1 日閲覧.

[32] Hiroshi Okuda. Nonlinear structual analysis open software FrontISTR. `http://www.ciss.iss.u-tokyo.ac.jp/riss/english/project/structure/FISTR_JE_1303.pdf`. 2015 年 1 月 1 日閲覧.

[33] METIS - serial graph partitioning and fill-reducing matrix ordering. `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`. 2015 年 1 月 1 日閲覧.

[34] OpenMP. `http://openmp.org/wp/`. 2015 年 1 月 1 日閲覧.

[35] Leslie Fox, Harry D Huskey, and James Hardy Wilkinson. Notes on the solution of algebraic linear simultaneous equations. *The Quarterly Journal of Mechanics and Applied Mathematics*, Vol. 1, No. 1, pp. 149–173, 1948.

[36] 藤野清次. 近似逆行列による前処理の特性について. 数理解析研究所講究録, Vol. 1320, pp. 179–189, 2003.

[37] Eun-Joo Lee and Jun Zhang. A two-phase preconditioning strategy of sparse approximate inverse for indefinite matrices. *Computers & Mathematics with Applications*, Vol. 58, No. 6, pp. 1152–1159, 2009.

[38] Eun-Joo Lee and Jun Zhang. Factored approximate inverse preconditioners with dynamic sparsity patterns. *Computers & Mathematics with Applications*, Vol. 62, No. 1, pp. 235–242, 2011.

[39] 森田直樹, 橋本学, 奥田洋司. 並列有限要素法のための A-直交過程に基づく RIF 前処理. 日本計算工学会論文集, 第 2014 巻, p. 20140015, 2014.

[40] Matrix market. `http://math.nist.gov/MatrixMarket/`. 2015 年 1 月 1 日閲覧.

[41] The university of florida sparse matrix collection. `http://www.cise.ufl.edu/research/sparse/matrices/`. 2015 年 1 月 1 日閲覧.

[42] Open MPI: Open source high performance computing. `http://www.open-mpi.org/`. 2015 年 1 月 1 日閲覧.

[43] Junuthula Narasimha Reddy. *Mechanics of laminated composite plates and shells: theory and analysis*. CRC press, 2004.

[44] Suresh Panda and R Natarajan. Analysis of laminated composite shell structures by finite element method. *Computers & Structures*, Vol. 14, No. 3, pp. 225–230, 1981.

[45] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. *Introduction to algorithms*, Vol. 2. MIT press Cambridge, 2001.

付録

A. 直交異方性材料を考慮した積層シェル要素の FrontISTR への実装

積層シェルへ拡張させる際に最も一般的な手法は，Equivalent Single Layer Theories（等価単層理論）を用いることである [43][44]．この方法は，積層方向に要素剛性マトリクスを積分することと解釈でき，解析対象全体の挙動を十分精度よく捉えることができることが知られている．単層シェル要素の要素剛性マトリクスは，式 (65) から求められる．

$$\begin{aligned} \boldsymbol{K}^e &= \int_{\boldsymbol{V}^e} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} d\boldsymbol{V} \\ &= \iiint_{-1}^{1} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} |\boldsymbol{J}| dr_1 dr_2 dr_3 \end{aligned} \tag{65}$$

この式 (65) を，積層要素の要素剛性マトリクスの算出のために拡張する．図 57 に，積層シェル要素の断面を示した．ここでは，等方性材料を用いると仮定した．

ここで，t_i は各層の厚さ，a は板厚，Z_i は各層の境界，i は層の番号，$\boldsymbol{j}$ は全体座標系から局所座標系へのヤコビヤン，r_3 は等価単層としたときの厚さ方向の座標軸，r_3^k は各層ごとの厚さ方向の座標軸，$-1 \leq (r_3, r_3^k) \leq 1$ をとり，$\times$ 印はガウスの数値積分点である．

積層シェルの場合，r_3 は，r_3^k を用いて式 (66) のように表される．

$$r_3 = -1 + \sum_{i=1}^{k} 2\frac{t_i}{a} - \frac{t_k}{a}(1 - r_3^k) \tag{66}$$

$r_3^k = \pm 0.5773$ ととれば，図 57 に示す積分点と一致する．

また，式 (66) を微分して式 (67) を得る．

$$dr_3 = d\frac{t_k}{a} r_3^k \tag{67}$$

式 (66) と式 (67) を式 (65) に適用し，式 (68) のように要素剛性マトリクスを拡張する．

$$\boldsymbol{K}^e = \sum_{k=1}^{n} \iiint_{-1}^{1} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} |\boldsymbol{J}| (\frac{t_k}{a}) dr_1 dr_2 dr_3^k \tag{68}$$

ここで n は積層数である．よって式 (68) に，積層シェル要素における要素剛性マトリクスが示された．

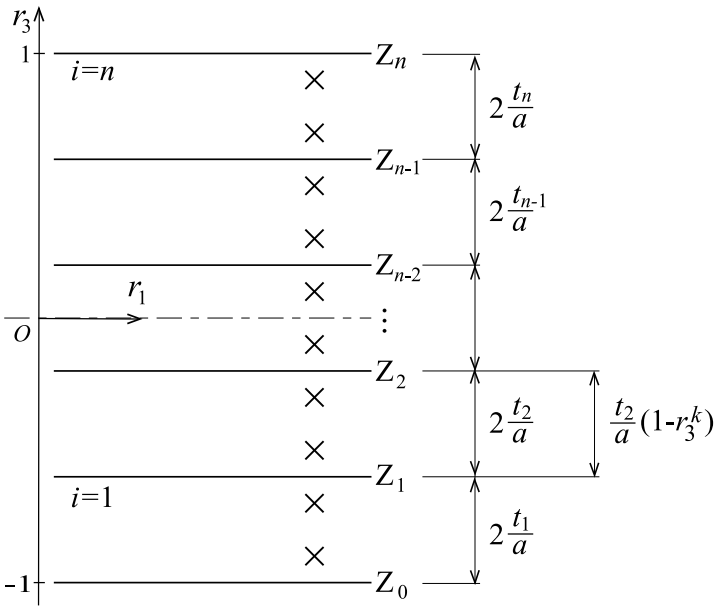


Fig. 57 Schematic representation of integration point of laminated shell element

直交異方性材料を考慮する場合，面内材料方向がある基準軸から角度 θ だけ回転することを考える．Shell 要素における応力の座標変換は式 (69) で表される．

$$\boldsymbol{\sigma}' = \boldsymbol{T}_1\boldsymbol{\sigma}$$

$$\begin{pmatrix} \sigma'_x \\ \sigma'_y \\ \sigma'_z \\ \tau'_{xy} \\ \tau'_{yz} \\ \tau'_{zx} \end{pmatrix} = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 0 & 2\sin\cos\theta & 0 & 0 \\ \sin^2\theta & \cos^2\theta & 0 & -2\sin\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\sin\cos\theta & \sin\cos\theta & 0 & \cos^2\theta - \sin^2\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{pmatrix} \tag{69}$$

Shell 要素におけるひずみの座標変換は式 (70) で表される．

$$\boldsymbol{\varepsilon}' = \boldsymbol{T}_2\boldsymbol{\varepsilon}$$

$$\begin{pmatrix} \varepsilon'_x \\ \varepsilon'_y \\ \varepsilon'_z \\ 2\gamma'_{xy} \\ 2\gamma'_{yz} \\ 2\gamma'_{zx} \end{pmatrix} = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 0 & \sin\cos\theta & 0 & 0 \\ \sin^2\theta & \cos^2\theta & 0 & -\sin\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -2\sin\cos\theta & 2\sin\cos\theta & 0 & \cos^2\theta - \sin^2\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ 2\gamma_{xy} \\ 2\gamma_{yz} \\ 2\gamma_{zx} \end{pmatrix} \tag{70}$$

Shell 要素における直交異方性は式 (71) によって表される．

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon}$$

$$\begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{pmatrix} = \begin{bmatrix} \dfrac{E_1}{1-\nu_{12}\nu_{21}} & \dfrac{\nu_{21}E_1}{1-\nu_{12}\nu_{21}} & 0 & 0 & 0 & 0 \\ \dfrac{\nu_{21}E_1}{1-\nu_{12}\nu_{21}} & \dfrac{E_2}{1-\nu_{12}\nu_{21}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & G_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma G_{23} & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma G_{31} \end{bmatrix} \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{pmatrix} \tag{71}$$

ここで，γ はせん断補正係数で，$\gamma = 5/6$ である．式 (69) を $\boldsymbol{\sigma} = \boldsymbol{T}_1^{-1}\boldsymbol{\sigma}'$，式 (70) を $\boldsymbol{\varepsilon} = \boldsymbol{T}_2^{-1}\boldsymbol{\varepsilon}'$，式 (71) を $\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon}$ と表すことができるので，これらの関係から，式 (72) に，θ による座標変換後の応力 $\boldsymbol{\sigma}'$ とひずみ $\boldsymbol{\varepsilon}'$ の関係を示す．

$$\begin{aligned} \boldsymbol{\sigma} &= \boldsymbol{D}\boldsymbol{\varepsilon} \\ \boldsymbol{T}_1^{-1}\boldsymbol{\sigma}' &= \boldsymbol{D}\boldsymbol{T}_2^{-1}\boldsymbol{\varepsilon}' \\ \boldsymbol{\sigma}' &= \boldsymbol{T}_1\boldsymbol{D}\boldsymbol{T}_2^{-1}\boldsymbol{\varepsilon}' \end{aligned} \tag{72}$$

ここで $\boldsymbol{T}_2^{-1}$ は回転行列であるから，$\boldsymbol{T}_2^{-1}(\theta) = \boldsymbol{T}_2(-\theta)$ として，式 (73) のようになる．

$$\boldsymbol{T}_2^{-1}(\theta) = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 0 & -2\sin\cos\theta & 0 & 0 \\ \sin^2\theta & \cos^2\theta & 0 & 2\sin\cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \sin\cos\theta & -\sin\cos\theta & 0 & \cos^2\theta - \sin^2\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & 0 & 0 & \sin\theta & \cos\theta \end{bmatrix} = \boldsymbol{T}_1^t \tag{73}$$

よって式 (72) は，式 (74) で表され，直交異方性の応力変換式を得る．

$$\boldsymbol{\sigma}' = \boldsymbol{T}_1\boldsymbol{D}\boldsymbol{T}_1^t\boldsymbol{\varepsilon}' \tag{74}$$

$\boldsymbol{T}_1 = \boldsymbol{T}$ と置きなおし式 (68) で得られた要素剛性マトリクスを式 (74) で拡張して，式 (75) を得る．

$$\boldsymbol{K}^e = \sum_{k=1}^{n} \iiint_{-1}^{1} \boldsymbol{B}^T\boldsymbol{T}\boldsymbol{D}\boldsymbol{T}^t\boldsymbol{B}|\boldsymbol{J}|(\frac{t_k}{a})dr_1dr_2dr_3^k \tag{75}$$

式 (75) に基づいて，直交異方性を考慮した積層シェル要素を FrontISTR に実装した．

B. 3×3 BCSR 形式で格納されたシェル要素の FrontISTR への実装

従来，FrontISTR で実装されているシェル要素は，1 節点が並進 3 自由度と回転 3 自由度の合計 6 自由度に基づく定式化により，6×6 BCSR 形式で格納されている．3×3 BCSR 形式で格納されたソリッド要素，梁要素，トラス要素など，シェル要素以外の要素が混在した解析モデルを解く場合，BCSR 形式の違いから全体剛性行列を生成することができない．この解決のため，3×3 BCSR 形式で格納されたシェル要素（以下 3×3 シェル要素）を FrontISTR へ実装した．

3×3 シェル要素の概要を図 58 に示す．FrontISTR のように体系化された有限要素法プログラムでは，新しい要素の定義とその要素に関する関数の実装が容易ではないことから，3×3 BCSR 形式で格納された場合の既存の関数を用いて計算を実現する手法を選ぶ．そこで，シェル要素（4 節点 ×6 自由度=1 要素あたり 24 自由度）と 1 要素あたりの自由度が同じである，ソリッド要素（8 節点 ×3 自由度=1 要素あたり 24 自由度）に注目し，プログラム内の構造をソリッド要素と同じように保持する．具体的には，シェル要素の節点番号 1〜4 の並進自由度がそれぞれソリッド要素の節点番号 1〜4 に対応し，シェル要素の節点番号 1〜4 の回転自由度がそれぞれソリッド要素の節点番号 5〜8 に対応する．このように定義された 3×3 シェル要素は，3×3 BCSR 形式で格納された場合の既存の関数に対して図 59 のような変換テーブルをもつ．

以上のように，図 58 と図 59 に基づいて，3×3 BCSR 形式で格納されたシェル要素を FrontISTR へ実装した．本研究ではこの実装を経て，シェル要素とその他の要素が混在した問題を取り扱った．

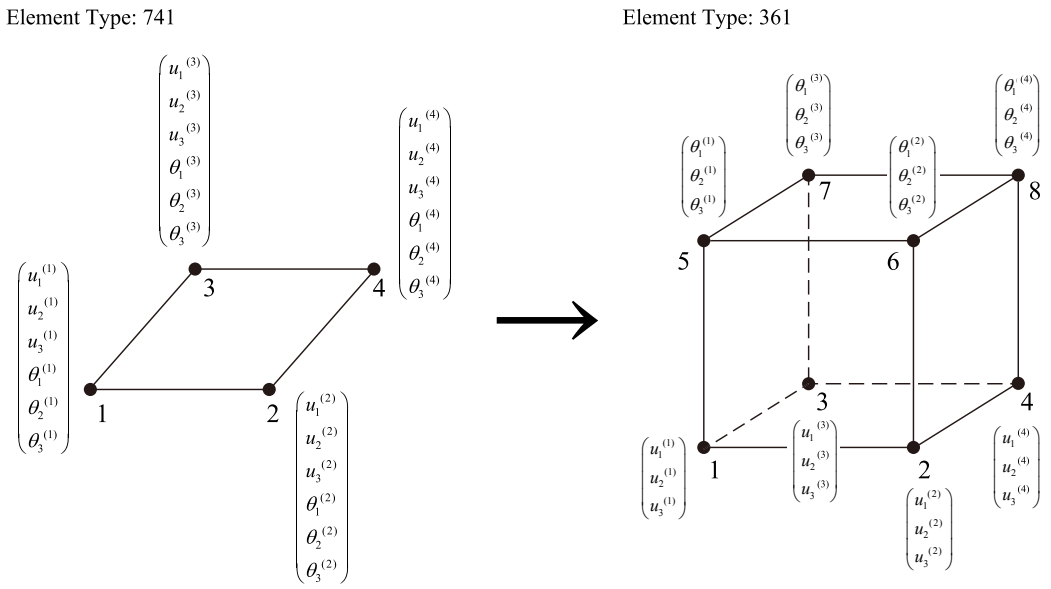


Fig. 58 Schematic representation of shell element storaged 3×3 BCSR using topology features of solid element

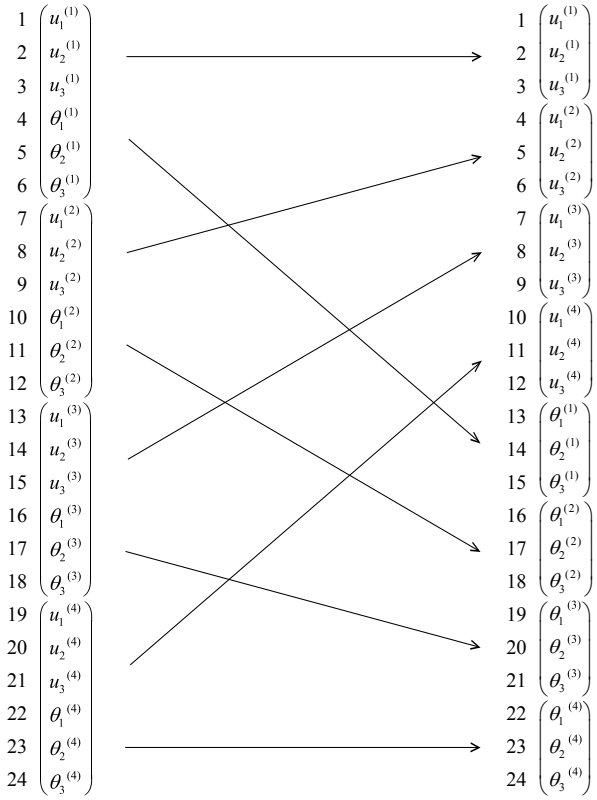


Fig. 59 Schematic representation of conversion table of shell element storaged 3×3 BCSR

C. 局所ベクトルの検索方法変更による FrontISTR の高速化

直交異方性の材料をもつ要素を用いる場合，各々の要素に対して材料方向を定める局所ベクトルを定義する必要がある．FrontISTR では，定義された局所ベクトルと要素を結びつける際に，局所ベクトルにつけられた名前ラベルを検索する．従来の FrontISTR では，検索に線形探索を用いるため，要素数 N として N 個の局所ベクトルが一対一で対応している場合，探索に必要な演算量は $O(N^2)$ である．

例えば，球状モデルに局所ベクトルを定義する場合には，要素数と同じ数だけの局所ベクトルを定義しなければならないように，曲面を有する大規模解析モデルの場合，定義される局所ベクトルの数が膨大になるため探索に必要な計算時間が問題となっていた．

この解決のために，汎用のハッシュテーブル [45] を FrontISTR に実装し，局所ベクトルの検索時間を削減した．