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

2014 年度修了人間環境学専攻 47136725 森田 直樹
指導教員 奥田 洋司 教授

Robust incomplete factorization (RIF) based on the A-orthogonalization process is an effective preconditioning technique for the conjugate gradient (CG) method to solve highly ill-conditioned linear systems. This research aims to accelerate the convergence of the finite element analysis of shell structures by holding the sparsity in the preconditioner and by parallelizing the localized process of RIF preconditioning and by using mixed precision arithmetic. In the numerical results, the proposed RIF preconditioner has shown better convergence and faster computing time than the conventional preconditioner.

Key words: Preconditioned CG method, Sparse Approximate Inverse, Parallel Finite Element Method

1 緒言

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ており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使用される．シェル要素は板状構造の挙動を良く再現でき，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フィルインが発生する．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で，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．

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．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．

大規模問題を並列直接法ソルバで解く場合，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．

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．しかし，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，悪条件を改善し収束までの反復回数を削減する，前処理が用いられる．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．

Benzi らは，悪条件をもつ連立一次方程式に有用な前処理として，安定化近似逆行列分解(Stabilized Approximate Inverse，SAINV)[1] から得られる分解因子を用い新たな不完全分解因子を構成する，ロバスト不完全分解(Robust Incomplete Factorization，RIF)[2] を提案した．

池田らは新たな閾値により前処理性能を改善した，改良型安定化近似逆行列(Improved SAINV，ISAINV)[3]，改良型ロバスト不完全分解(Improved RIF, IRIF)[3]を提案した．

しかしこれら前処理を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，従来より高い計算精度を要求する，高精度数値計算の必要が挙げられる．しかし，高い精度の浮動小数点数は，演算量とメモリ使用量の増加に直接影響する．この解決のために，解に影響しない部分を低い精度の浮動小数点数で計算し，得られた解を補正し高い精度の解を保証する，混合精度 iterative refinement が注目されている．

さらに近年，省エネルギー性から 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における並列計算が注目されている．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．

本研究では，RIF 前処理を領域分割に基づく有限要素法に適用する．そこで，係数行列 A の非零プロファイルを元にした，フィルインを考慮しない局所並列化した RIF(0) と，フィルインを考慮し局所並列化した RIF(1) を提案する．提案手法は，前処理生成より前に前処理行列 M の非零要素プロファイルを定める点で優れており，これら前処理を用いた並列共役勾配法の有用性について検討する．

さらに，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．

$r_0 = b - Ax_0$, $p_0 = M^{-1}r_0$
for $m = 1, 2, 3, \ldots,$ **do**
  $\alpha_m = \dfrac{(r_{m-1}, M^{-1}r_{m-1})}{(p_{m-1}, Ap_{m-1})}$
  $x_m = x_{m-1} + \alpha_m p_{m-1}$
  $r_m = r_{m-1} - \alpha_m A p_{m-1}$
  if $(\|r_m\|_2 / \|r_0\|_2 \leqq \varepsilon)$ **then stop**
  $\beta_m = \dfrac{(r_m, M^{-1}r_m)}{(r_{m-1}, M^{-1}r_{m-1})}$
  $p_m = M^{-1}r_m + \beta_m p_{m-1}$
end for

Fig.1 Preconditioned conjugate gradient method

$Z^{(0)} = E$
for $i = 1, 2, 3, \ldots, n$ **do**
  $s = Az_i^{(i-1)}$
  for $j = i, i+1, i+2, \ldots, n$ **do**
    $d_j = {}^t s z_j^{(i-1)}$
  end for
  for $j = i+1, i+2, \ldots, n$ **do**
    if $(|\frac{d_j}{d_i}| > toldd)$ **then**
      for $k = 1, 2, 3, \ldots, n$ **do**
        $z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
      end for
    end if
  end for
end for

Fig. 2 Preconditioner of ISAINV

2 安定化近似逆行列分解とロバスト不完全分解

2.1 前処理つき共役勾配法

前処理つき共役勾配法は，式(1)のような正定値対称の係数行列 A を持つ連立一次方程式の反復解法として非常によく知られている．ここで，A は $n \times n$ 次正方行列，$\boldsymbol{x}$ は n 次の解ベクトル，$\boldsymbol{b}$ は n 次の右辺ベクトルである．

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

図 1 に，前処理つき共役勾配法を示す．ここで，$\boldsymbol{r}$ は残差ベクトル，$\boldsymbol{p}$ は探索方向ベクトル，M は前処理行列，ε は収束判定値である．前処理行列 M は，M≅A であるほど前処理の効果が大きくなるが，その反面計算量は増大する．

2.2 改良型安定化近似逆行列

改良型 SAINV は，図 2 に示すように前処理行列 M の逆行列 $M^{-1}=ZD^{-1}Z^t$ を構成する．ここで，z_j は上三角行列 Z の第 j 番目の列ベクトル，a_j は係数行列 A の第 j 番目の列ベクトル，d_j は対角行列 D の第 j 番目の対角要素，$\boldsymbol{s}$ は中間ベクトル，E は単位行列，上付き添え字(i)は i 回目の分解を示す．SAINV は中間ベクトル $\boldsymbol{s}$ を用いて正定値性を維持し安定して分解を行う．分解中は，閾値 *tol* より小さな分解因子の値は零に棄却され，前処理行列 M のスパース性が保たれる．本研究ではこれを，drop(*,*tol*)で示した．閾値 *toldd* はベクトル $z_j^{(i+1)}$の更新判定として使用される．

$Z^{(0)} = E$, $L^{(0)} = E$
for $i = 1, 2, 3, \ldots, n$ **do**
  $s = Az_i^{(i-1)}$
  for $j = i, i+1, i+2, \ldots, n$ **do**
    $d_j = {}^t s z_j^{(i-1)}$
  end for
  for $j = i+1, i+2, \ldots, n$ **do**
    if $(|\frac{d_j}{d_i}| > tol)$ **then**
      $l_{ji} = \dfrac{d_j}{d_i}$
    end if
    if $(|\frac{d_j}{d_i}| > toldd)$ **then**
      for $k = 1, 2, 3, \ldots, n$ **do**
        $z_{kj}^{(i)} = \mathrm{drop}\ (\ z_{kj}^{(i-1)} - \dfrac{d_j}{d_i} z_{ki}^{(i-1)},\ tol\ )$
      end for
    end if
  end for
end for

Fig. 3 Preconditioner of IRIF

2.3 改良型ロバスト不完全分解

改良型 RIF は，図 2 に示した改良型 SAINV 処理中に得られるd_j/d_iを用いて，図 3 に示すように不完全分解因子 L を構成し，前処理行列 $M=LDL^t$ を構成する．ここで，l_{ji}は下三角行列 L の第(j,i)番目の行列要素である．RIF は，不完全コレスキー分解とは異なる手法によって不完全分解を行うことから，これまで安定して計算できなかった問題に対しても，有用な前処理結果を得る場合がある．

3 並列有限要素法解析のための近似逆行列分解の高速化と並列化

3.1 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 M のスパース性を保つため，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，さらに前処理行列生成中に得られる全ての要素に対して棄却判定を行う必要があるため，前処理行列生成の高速化には適していない．そのため，係数行列 A の非零要素プロファイルを前処理生成にも用いた 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 [4]．図 4 に IRIF(1) 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．

3.2 Localized IRIF

大規模計算を考慮すると，いかに堅固な前処理であっても，前処理並列化は欠かすことができない．並列化の手法は様々あるが，その中で本研究では，Localized IRIF を提案する [4]．Localized IRIF の概要を図 5 に示す．Localized IRIF では，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高い並列化と高速化を見込んでいる．この処理では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，本来の RIF 前処理と比較すると前処理性能は分割数の依存性が懸念される．

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. 4 Preconditioner of IRIF(1)

3.3 前処理行列の混合精度演算

単精度浮動小数点数と倍精度浮動小数点数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演算時間を削減する方法を提案する．図 6 に，混合精度演算を用いた前処理つき共役勾配法を示す．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．提案手法は，係数行列 **A** 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．

4 数値例

フィルインを考慮した Localized ILU(1)と Localized IRIF(1), 単精度で計算しフィルインを考慮した Localized ILU(1)と Localized IRIF(1)の前処理を有する共役勾配法に対して, 並列前処理性能の並列分割数による反復収束性への影響と, 混合精度演算による反復収束性への影響を比較した．並列分割数は，1，2，4，8，16 と変化させ計算を行った．計算機は東京大学情報基盤センター内の FX10 を利用した. 収束判定値 ε =1.0×10^{-8} とした．数値例には，板状構造から得た剛性行列 SHELL を用いる．このモデルは，幅 200mm×高さ 10mm×奥行 1,000mm，シェル厚さ 0.01mm であり，要素は MITC4 を使用した．節点 10,848，要素数 10,000，自由度 65,088，非零要素数 3,331,728，非零要素比 0.0786%，条件数の推定値は 9.49 ×10^{9} である．

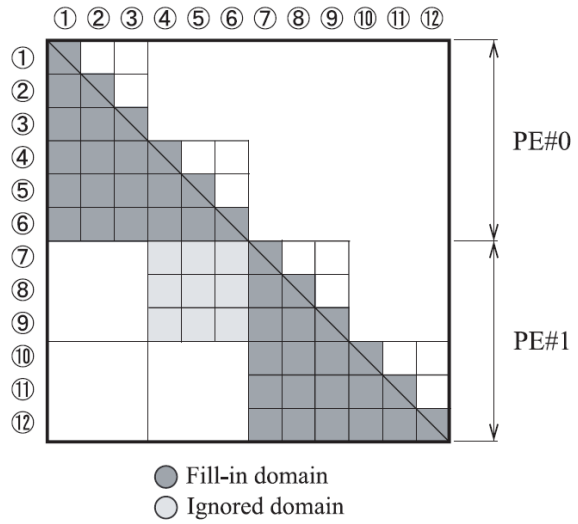

Fig. 5 Schematic representation of sill-in and ignored domains of lower triangular matrix in localized IRIF

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

Fig.6 Mixed precision preconditioned conjugate gradient method

表 1 に，フィルインを考慮した Localized ILU(1)と Localized IRIF(1)，単精度で計算しフィルインを考慮した Localized ILU(1)と Localized IRIF(1)を適用させ，並列計算を行った結果を示す．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 $\mathbf{z}^{(i-1)}{}_j$の更新判定閾値，PE は並列プロセス数，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．単精度で計算した前処理を特に Single Precision と示した．それぞれの計算で最も短かった計算時間を記載した．

数値例の結果，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)

Table 1 Numerical results of mixed precision parallel preconditioned CG method

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
	RIF(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
	RIF(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

で生じ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. 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.

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. 特に, 単精度 Localized ILU(1)を適用した場合, 反復回数が大きく増加している. 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. 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.

5 結言

本研究では, 並列有限要素法の共役勾配法前処理として, 係数行列 A の非零要素プロファイルを元にしたフィルインを考慮しない局所並列化した 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.

さらに, 提案手法を単精度で演算し, 混合精度演算を用いた共役勾配法に適用させた.

例題解析の結果, 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. 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.

参考文献

1) Michele Benzi, Cullum, J.K. and Tuma M., "Robust approximate inverse preconditioning for the conjugate gradient method", *SIAM J. on Scientific Computing*, Vol.22, pp.1318-1332 (2000).
2) Michele Benzi and Miroslav Tuma, "A robust incomplete factorization preconditioner for positive define matrices", *Numerical Linear Algebra with Applications*, Vol.10, pp.385-400 (2003).
3) 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, Vol.45, No. SIG6(ACS6), pp.95-104 (2004).
4) 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, Vol. 2014, p. 20140015 (2014).