



地盤有限要素解析における求解時収束性の検討

河野, 博人
竹山, 智英
橘, 伸也
高山, 裕介

(Citation)

神戸大学都市安全研究センター研究報告, 29:34-43

(Issue Date)

2025-03

(Resource Type)

departmental bulletin paper

(Version)

Version of Record

(JaLCD0I)

<https://doi.org/10.24546/0100496648>

(URL)

<https://hdl.handle.net/20.500.14094/0100496648>



地盤有限要素解析における求解時収束性の検討

Study on convergence of solutions in geotechnical finite element analysis

河野 博人¹⁾
Hiroto Kono
竹山 智英²⁾
Tomohide Takeyama
橘 伸也³⁾
Shinya Tachibana
高山 裕介⁴⁾
Yusuke Takayama

概要：大規模な有限要素解析を行う際に解決すべき問題として計算時間の短縮と使用メモリ量の削減が挙げられ，計算時間短縮のための手法として代表的なものに並列化がある．また，反復法による求解も計算時間短縮に寄与すると考えられる．数値計算上の誤差によって収束性が悪化する恐れがあるという反復法の性質に対しては前処理を導入することで対処する．また，前処理の効果をより高めるために AMD ordering を導入する．本研究では DACSAR-I4 のソルバーとして AMD ordering を導入した前処理付き共役勾配法を用いた際の計算時間短縮効果を測定する．

キーワード：共役勾配法，並列化，前処理，AMD ordering

1. 研究背景及び目的

DACSAR-I (Deformation Analysis Considering Stress Anisotropy and Reorientation – Integration：動的土/水/空気連成有限要素解析コード)¹⁾によって大規模な解析を行う際に，解析時間が長いことと使用メモリ量が多いことが課題である．計算速度を向上させる手法として有力なものがプログラムの並列化である．並列化手法はメモリとプロセッサの関係からプロセス並列とスレッド並列の2種類に分けられ，本研究では MPI(Message Passing Interface)²⁾を用いたプロセス並列を採用した．プロセス並列はスレッド並列よりも実装難易度が高いがスケーラビリティが高いという特徴がある．図1に各並列の模式図を示す．

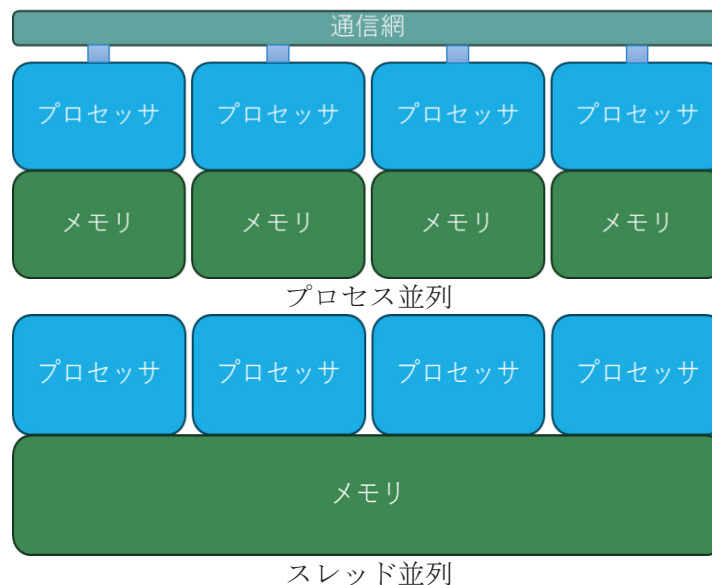


図1 各並列形式の模式図

また、並列化の時短効果には方程式解法による側面も大きい。直接法と反復法を比較した場合、計算手法の特性上並列化効果を得やすいのは反復法である。しかし、反復法は数値計算上の誤差によって解を求められない可能性があり、特に DACSAR-I で扱う連成問題の剛性マトリクスのようなオーダーの差が大きく性質の悪いマトリクスにおいて顕著である。収束性の改善には前処理が用いられ、反復操作開始前に性質の悪い方程式を収束性がよい形に変換する。本研究では不完全 LU 分解前処理を用いた。これは LU 分解の過程で発生するフィルインを無視して行列の分解を行うというものである。さらに、補助的な操作として AMD ordering³⁾がある。これは LU 分解前にフィルイン発生数が最小となるように行と列を並び替えるもので、不完全 LU 分解に対しては無視されるフィルイン数が減り前処理効果を高められると考えられる。

使用メモリ量の観点では、係数行列については直接法の場合は求解のための行列分解によって発生するフィルインを考慮してメモリを確保しなければならないが、反復法の場合はその必要はないため使用メモリ量が削減できると考えられる。

以上より本研究では DACSAR-I4 のソルバーに不完全 LU 分解前処理付き共役勾配法を採用する。また、計算実行時の反復回数削減を目的として AMD ordering をソルバーに導入する。使用言語は C, C++ である。

2. 手法

(1) 前処理

前処理とは扱う方程式を条件の良い問題に変形することであり、元の方程式の両辺に変換行列 P^{-1} を左側からかける操作を行う。変換行列は元の係数行列 A の逆行列に近い性質をもつものが好ましい。

$$P^{-1}Ax = P^{-1}b \quad (1)$$

(2) 不完全 LU 分解

不完全 LU 分解は前処理行列を求める手法の一つである。フィルインがすべて無視されるため通常の LU 分解と異なり行列分解の前後で総非ゼロ数は変化しない。

$$LU = M \approx A \quad (2)$$

(3) 不完全 LU 分解前処理を導入した共役勾配法

まず、不完全 LU 分解によって下三角行列 L と上三角行列 U を求める。

$$LU = M \approx A \quad (3)$$

前処理方程式を解き初期残差ベクトル r_0 を求める。($x_0 = 0$)

$$r_0 = b - Ax_0 \quad (4)$$

r_0 を右辺に置き、代入操作により前処理済みの残差ベクトル z_0 を求める。

$$Mz_0 = LUz_0 = r_0 \quad (5)$$

$$y = Uz_0 \quad (6)$$

前進代入を行う。

$$Ly = r_0 \quad (7)$$

$$y = L^{-1}r_0 \quad (8)$$

後退代入を行い、初期探索方向ベクトル p_0 を求める。

$$z_0 = U^{-1}y \quad (9)$$

$$p_0 = z_0 \quad (10)$$

修正量 α_k を計算する. ($k = 0,1,2,3 \dots$)

$$\alpha_k = \frac{r_k^T z_k}{(Ap_k)^T p_k} \quad (11)$$

解と残差を更新する. ($k = 0,1,2,3 \dots$)

$$x_{k+1} = x_k + \alpha_k p_k \quad (12)$$

$$r_{k+1} = r_k - \alpha_k A p_k \quad (13)$$

次ステップの残差を計算し修正量 β_k を計算する. ($k = 0,1,2,3 \dots$)

$$Mz_{k+1} = LUz_{k+1} = r_{k+1} \quad (14)$$

$$\beta_k = \frac{r_{k+1}^T z_{k+1}}{r_k^T z_k} \quad (15)$$

探索方向ベクトルを更新する.

$$p_{k+1} = z_{k+1} + \beta_k p_k \quad (16)$$

残差ベクトルのノルム $\|r_{k+1}\|$ を求め収束判定を実施する.

$$\frac{\|r_{k+1}\|}{\|b\|} (< or >) \epsilon \quad (17)$$

前処理は初回の操作時のみ行われる. 各種ベクトルの更新操作は定めた閾値よりも残差ベクトルのノルムが小さくなるまで繰り返される. 以上が不完全 LU 分解前処理を導入した共役勾配法のプロセスである.

3. 疎行列ソルバーの並列化

(1) MPI(Message Passing Interface)

疎行列ソルバーの並列化には MPI(Message Passing Interface)を使用する. MPI のライブラリ内には様々な通信関数が用意されており必要に応じてこれらの関数を使用する. 用意されている通信関数の種類は大きく分けると以下の通りである. ⁵⁾

- 1 対 1 通信
- 1 対全通信
- 全対 1 通信
- 集団通信

(2) 領域分割

プロセス並列では計算に必要な情報を分割して各プロセッサの固有メモリ領域に格納することが必要である. DACSAR-I4 でメッシャーとして用いている Gmsh⁶⁾には Metis という領域分割機能があり, これを使用する.

(3) 領域分割された領域の全体剛性マトリクス生成

領域分割を行った場合は, 全体剛性マトリクスは分割された領域ごとに生成される. これが可能になるのは共役勾配法の操作が方程式の各式に対して独立的に行われるためである. 前進代入

と後退代入のような前ステップの結果が次ステップに影響するような操作がないため並列的な操作が可能になる。領域が分割されると、本来は 1 つのメモリ領域にあった情報がバラバラに独立したメモリ領域に格納されるため、反復操作を行う上で通信をして各プロセッサが必要な情報を他のプロセッサから手に入れる必要がある。図 2 に節点が 20 個含まれる 2 次元領域を 3 つに分けたものを示す。この図では領域 0 の節点 8 の反復操作に必要な通信を表現している。節点 8(領域 0)は周囲の 8 個の点と関係を持っており、節点 8 に働く外力に関するつり合い式を立てるためには周囲 8 個の節点に関する情報が必要である。このうち節点 14 と 15 は領域 0 にとっては外点と呼ばれる点でありこれらの情報は外部から通信して取得する。節点 14 は節点 4(領域 1)に、節点 15 は節点 4(領域 2)に相当するので、それぞれの領域を担当するプロセッサ同士が通信を行う。通信は通信テーブル(送信元と受信先を格納したデータ)に基づき、各反復操作で実行される。

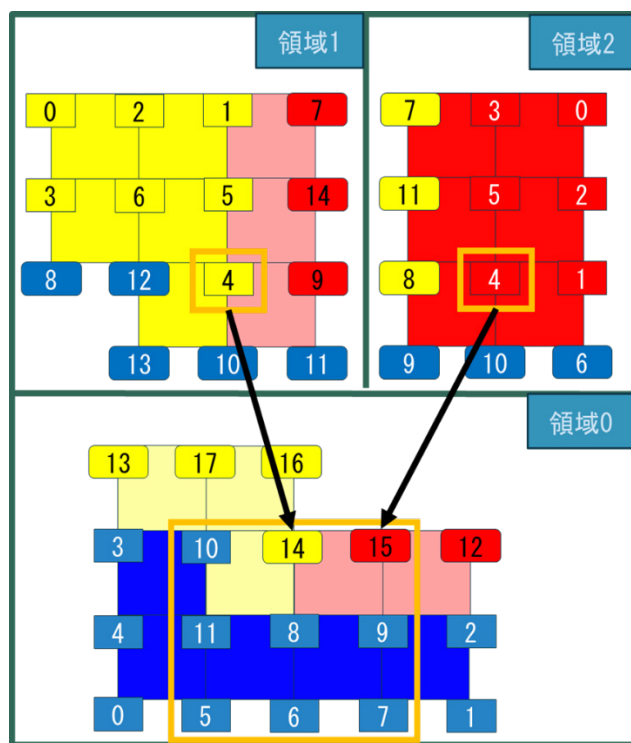


図 2 領域分割の様子

(4) 不完全 LU 分解前処理の並列化

不完全 LU 分解前処理を並列化するには領域分割された全体剛性マトリクスを「領域分割されていない全体剛性マトリクスが列方向ブロック分割された形」へ変形する必要がある。図 3 に変形の様子を示す。この操作は局所的な全体剛性マトリクスを領域分割されていない形に変形することと列方向に等しい大きさのブロックに分割することを同時に行っている。

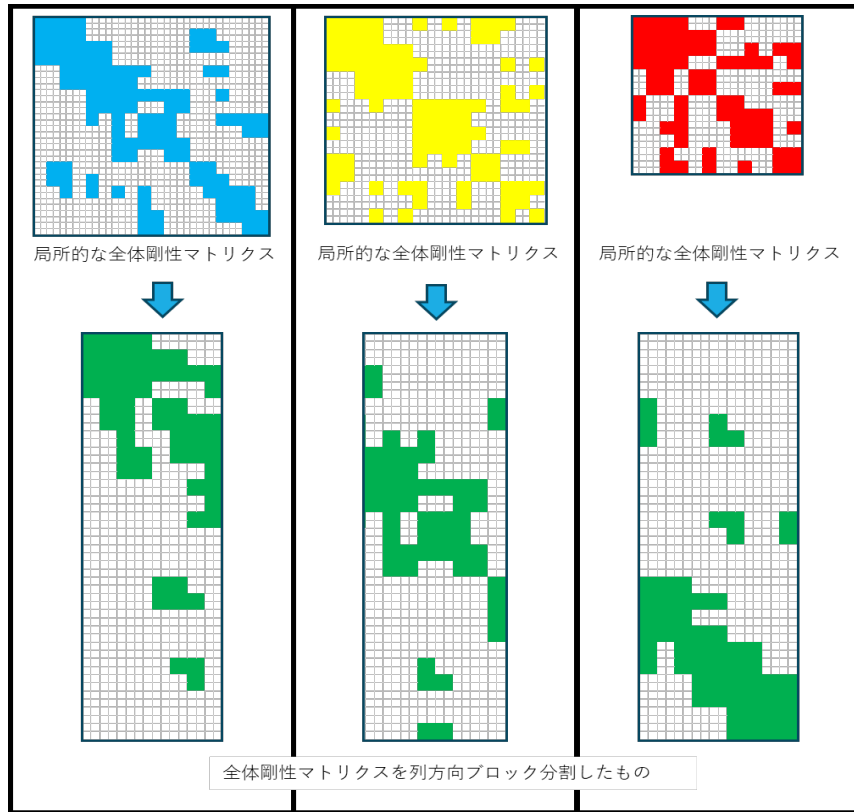


図3 局所の全体剛性マトリクスを不完全LU分解用マトリクスへ変換

行列の変換が終了後、これらのブロックに対して外積形式ガウス法を用いて並列的に不完全LU分解を行う。図4に外積形式ガウス法⁴⁾の操作を示す。外積形式ガウス法は次のような手順でLU分解を行う。

①k行目の対角成分をピボットとしてk+1行目からn行目までの直下の成分をピボットで割る。

$$a'_{k+l,k} = a_{k+l,k} / a_{kk} (l = 1, 2, \dots, n - k) \quad (18)$$

②k列目に対して対称な成分 $a_{k,k+l}$ に $a'_{k+l,k}$ をかけて、一つ下の行の成分 $a_{k+l+1,k}$ から引く

$$a'_{k+l+1,k} = a_{k+l+1,k} - a_{k,k+l} \times a'_{k+l,k} (l = 1, 2, \dots, n - k - 1) \quad (19)$$

③手順1と2を行列の次数回繰り返して行列の上三角部分と下三角部分を更新する。

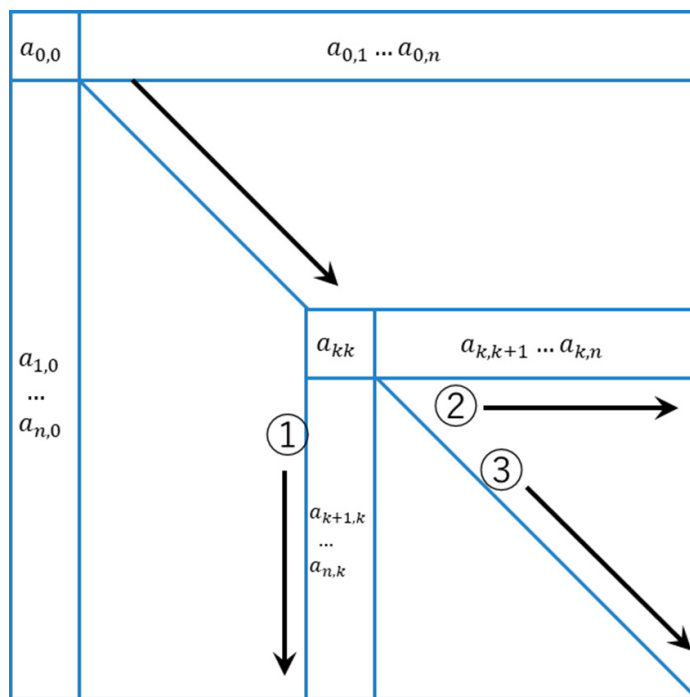


図 4 外積形式ガウス法

ピボット成分があるブロックを持つプロセッサよりも左側のプロセッサへ操作 1 が行われた列を送信し、各プロセッサ内で並行的に操作 2 が行われることで不完全 LU 分解の実行時間が短縮される。代入操作もこれらのブロックに対して行われる。図 5 に示す。

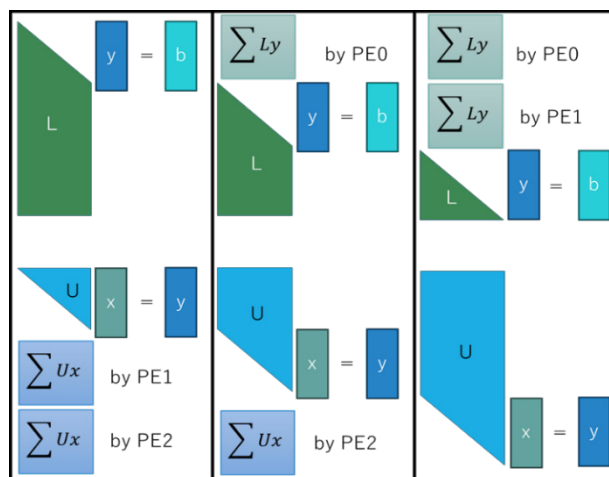


図 5 通信を伴う代入操作

代入操作の前には右辺ベクトルを列ブロック分割に対応した形で分解しておく。まず、最も左のブロックで列の幅分の前進代入が完了しベクトル y が求められる。その後右のブロックへ代入操作が移行していく。後退代入は最も右のブロックで列の幅分の操作が終わり、その後左のブロックへ代入操作外交していく。行列がブロック分割されて格納されているので左辺の総和計算を行う際には通信が必要となり、前進代入では左から右へ、後退代入では右から左へ局所的な左辺の総和が代入操作を行っているプロセスへ送信される。

4. 不完全 LU 分解前処理に対する AMD ordering 導入

共役勾配法の反復回数を減少させるための前処理に用いる行列は元の係数行列に近い方が好ましいということは先ほど述べた。不完全 LU 分解はそのような行列を求めるために有用である。AMD ordering は LU 分解実行時のフィルイン発生数を抑制する操作である。不完全 LU 分解の場合は、AMD ordering を行うとオーダリングを行わない場合よりも本来の LU 分解に近い結果が得られる。これにより元の係数行列により近い性質の前処理行列を得ることができ、不完全 LU 分解前処理効果を高めることができると考えられる。共役勾配法では数値計算上の誤差が収束性に悪影響を及ぼすため、連成計算のような条件の悪い問題や大規模な問題を扱う場合には前処理がほぼ必須であると考えられる。図 6 に行列の自由度数と LU 分解によるフィルイン発生数をオーダリングの有無で比較したものを示す。自由度数が多いほどオーダリングによる抑制効果が大きいことが分かる。

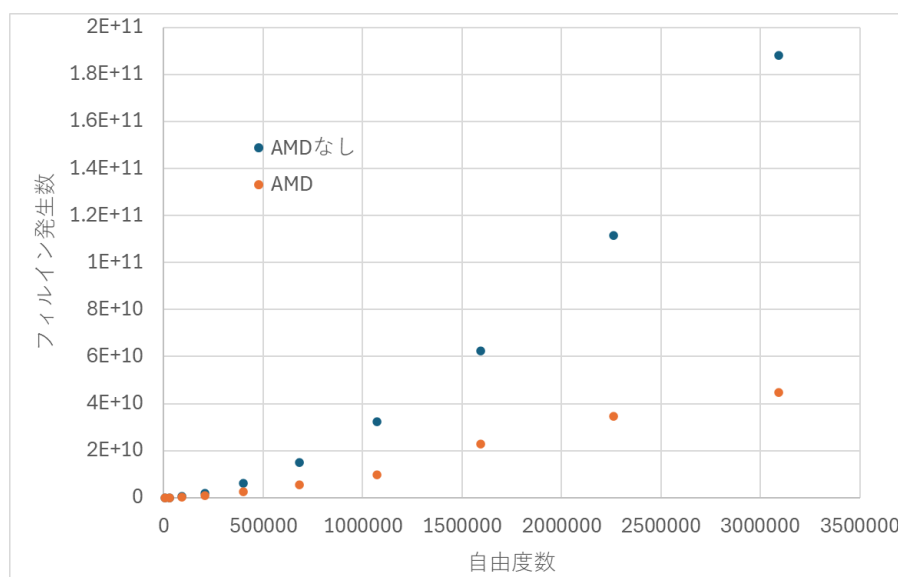


図 6 全体剛性マトリクスの自由度数とフィルイン発生数の関係

以上を踏まえて前処理や AMD ordering の有無で共役勾配法の反復回数を比較した。解析対象領域は 8 節点六面体要素で要素分割され、要素数は 125,000、全体剛性方程式の自由度数は 397,953 とした。表 1 には各条件の反復回数を示す。図 8 には収束解と反復ごとの解の差のノルムと反復回数との関係のグラフを示す。AMD ordering による効果を検証する際には扱う関数が並列化に未対応であるため並列化は行わず、1 プロセッサでの計算を行った。

表 1 各条件の反復回数

条件	前処理無/AMD 無	前処理有/AMD 無	前処理有/AMD 有
反復回数	338	179	307

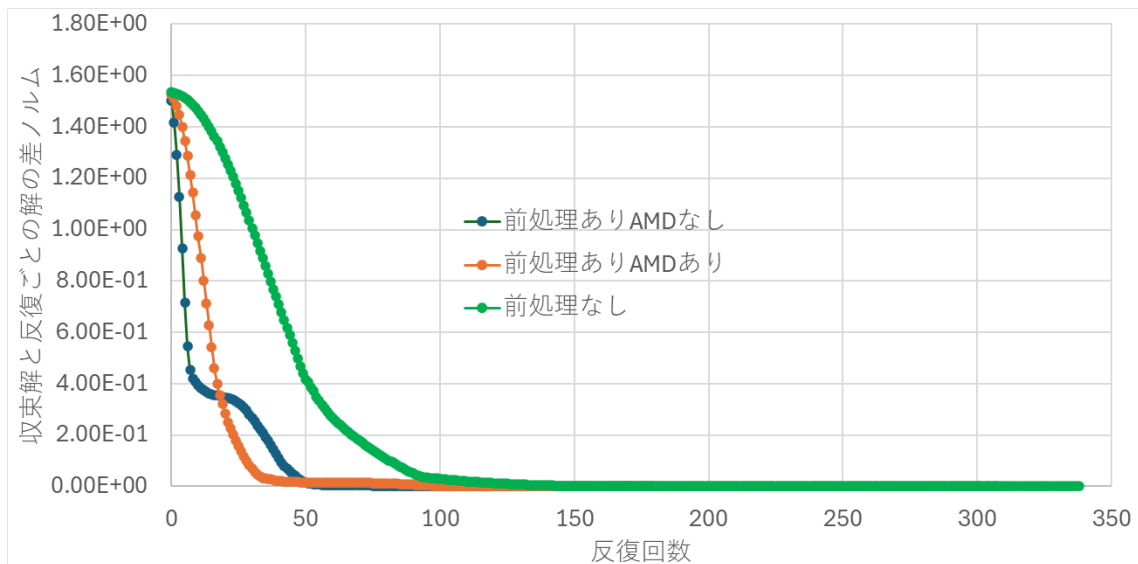


図 8 収束解と反復ごとの解の差のノルムと反復回数との関係

表 1 の結果では前処理有/AMD ordering 無の場合の反復回数が最も少ないという結果になった．図 8 では初期解から収束解へ解が近づいていく様子を示しているが，前処理有/AMD ordering 無の場合が最も早く収束解に近づいていくことが分かる．この結果から AMD ordering は不完全 LU 分解前処理の効果を打ち消していると考えられる．

5. 結論

本研究では不完全 LU 分解前処理付き共役勾配法の反復回数の減少を目的として AMD ordering を導入したが，得られた結果は予想と異なり反復回数が増加するというものだった．非連成弾性問題では前処理を行わなくても比較的少ない数で収束するが，連成問題では行列の条件が悪いため前処理なしの共役勾配法では計算が発散する恐れがあり，前処理はほぼ必須であると考えられる．また，前処理を行った場合でも解の収束には非連成の弾性計算よりかなり多い反復を必要とすることが考えられる．

参考文献

- 1) Tomohide Takeyama, Shinya Tachibana, Aiko Furukawa : A finite element method to describe the cyclic behavior of saturated soil, International Journal of Material Science & Engineering, Vol.2 Issue.1, April, 2015
- 2) 安藤嘉倫, 連載「分子動力学計算の大規模並列化のための技術 2」, 分子シミュレーション学会誌”アンサンブル”Vol. 25, No. 2, April 2023(通巻 102 号)
- 3) Patric R. Amestoy, Timothy A. Davis, Iain S. Duff, SIAM J. Matrix Analysis & Applic, Vol17, no 4, pp. 806-905, Dec. 1996 An Approximate Minimum Degree Ordering Algorithm
- 4) 堂ヶ原建, 竹山智英, 飯塚敦, 銭谷誠司, 地盤応答解析コードの GPU 並列化と液状化解析 pp.64, 2022
- 5) 片桐孝洋 スパコンプログラミング入門~並列処理と MPI の学習~ 東京大学出版会 pp.17, 2013
- 6) Gmsh4.13.1
<https://gmsh.info/doc/texinfo/gmsh.html>(最終閲覧日:2025/03/13)

筆者:1)河野 博人, 神戸大学大学院工学研究科市民工学専攻, 学生;2) 竹山 智英, 神戸大学大学院工学研究科, 教授; 3) 橘 伸也, 神戸大学都市安全研究センター, 教授;4) 高山 裕介, 神戸大学都市安全研究センター, 准教授

Study on convergence of solutions in geotechnical finite element analysis

Hiroto Kono
Tomohide Takeyama
Shinya Tachibana
Yusuke Takayama

Abstract

The problems to be solved in large-scale finite element analysis are the reduction of computation time and the amount of memory used. Parallelization is a typical method for reducing computation time. In addition, iterative solution methods are also considered to contribute to the reduction of computation time. The convergence of the iterative method may deteriorate due to errors in numerical calculations, a property of the iterative method that is addressed by introducing preprocessing. AMD ordering is also introduced to enhance the effect of preprocessing. In this study, we measure the effect of using the conjugate gradient method with AMD ordering as a solver for DACSAR-I4 to reduce the computation time.

©2025 Research Center for Urban Safety and Security, Kobe University, All rights reserved.