# 大規模動的有限要素解析へ向けた効率化とその性能

九州大学 学生会員 ○野中 翔
九州大学大学院 正会員 浅井 光輝

## １．緒言

64 ビット機の導入とほぼ同時期に CPU のマルチコア化が加速し，大容量メモリを搭載したメモリ共有型並列計算機が比較的安価に購入できるようになった．このため，環境にあわせたプログラミングをすれば，以前はスーパーコンピュータなどの大型計算機に頼らざるを得なかった規模の大規模計算がパソコンあるいはワークステーションレベルの環境で実施できるようになりつつある．

本研究では，地盤 - 構造物の複合領域を同時に有限要素メッシュで離散化する地震時応答解析を行うことを目標した動的有限要素解析プログラムを独自開発している．特に，陰解法において計算時間および消費メモリともにボトルネックとなる連立一次方程式の解法について改良を施した．本稿では，それらの方法の概要と，その効率について報告する．

## ２．陰的な動的有限要素法の並列化

### 2.1 前処理付き共役勾配法

離散化解析における連立一次方程式の解法は，直接法と反復法の 2 種類に大別できる．直接法は，有限回の式変形により連立一次方程式の解を求める方法であり，安定して求解できることが特徴である．しかしながら，使用メモリ量が多く，また並列化にはあまり適していない．一方，反復法は繰り返し計算により反復的に解を更新することで近似解を求める手法である．問題によっては解の収束が悪くなる場合もあるものの，直接法に比べると使用メモリ量は少なく，並列化も容易に行えることから，大規模計算にしばしば採用される．なお，収束の悪さについては，前処理によりある程度緩和することができる．

上記の理由から，反復法の一種である前処理付き共役勾配法を選択し，共有メモリ型並列計算機の環境にあわせて改良をおこなった．その計算フローを**図-1** に示す．ここで，**A** は係数行列，$\mathbf{x}^{(i)}$は解ベクトル，**b** は既知ベクトル，**M** は前処理行列，$\mathbf{r}^{(i)}$は解の収束判定に使用する残差，$\alpha^{(i)}$，$\beta^{(i)}$，$\rho^{(i)}$，$\mathbf{z}^{(i)}$，$\mathbf{p}^{(i)}$，$\mathbf{q}^{(i)}$はそれぞれ解の更新に必要なスカラー，ベクトルである．上付きの($i$)は $i$ 回目の反復における値であることを示す．

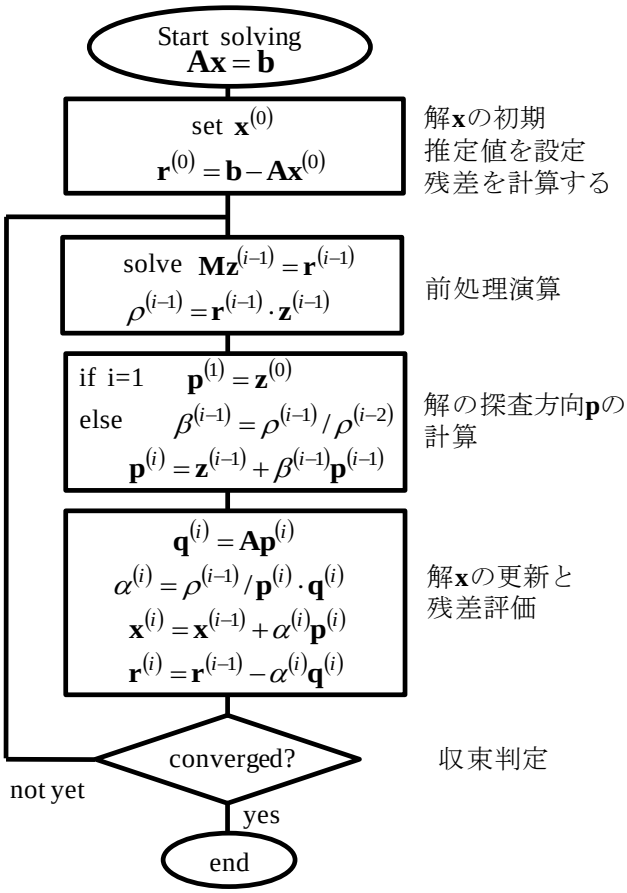


**図-1 共役勾配法の計算フロー**

### 2.2 node by node 法

**図-1** に示すように，共役勾配法では行列・ベクトル積などの代数演算の繰り返しにより解の更新が行われる．このため，いかにこの演算の効率を向上させるかが，重要なポイントである．これまでには，特にメモリ効率の観点から element by element 法が用いられた例が多く報告されている．ただし，共有メモリ型並列計算においては，メモリへの同時アクセスによるデータレースの恐れがあり，行列の保存方法，行列・ベクトル積の演算順序を工夫しなくてはならない．そこで本研究では，node by node 法により行列を記憶することにし，上記の問題を回避した．node by node 法とは，行列・ベクトル積で必要になる行列，ベクトルの非ゼロ成分とその位置関係を記憶する方法であり，適度に消費メモリ，演算量を節約することができる．

図-2 に，node by node 法による行列・ベクトル積の演算過程を示す．黒丸や白抜きの丸はそこが非ゼロの成分であることを示す．図-2(a)より，行列のある行(白抜きの丸で示した部分)について，非ゼロの成分とそこに対応するベクトルの成分との積のみが値をもち，その行の積の結果に影響することがわかる．この対応関係を記憶すれば図-2(b)のように，行列の非ゼロ成分のみを左に寄せるような状態で行列を記憶しても，行列・ベクトル積を求めることが可能である．有限要素法では通常，剛性行列や質量行列が連立一次方程式の係数行列となり，その形は正方行列である．node by node 法により係数行列を記憶すれば，それらの行列が正方行列であった場合との差の分(図-2(b)において点線で示した部分)だけメモリ使用量と演算量を節約することができる．

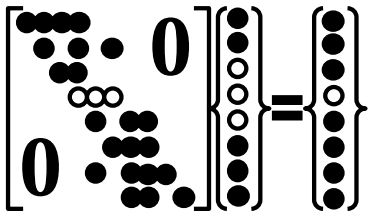

図-2(a)　行列・ベクトル積

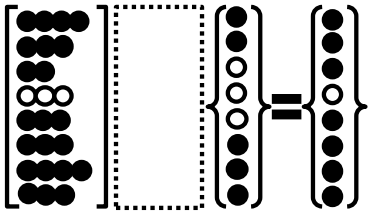

図-2(b)　node by node 法における行列・ベクトル積

### 2.3 並列化

本研究では，OpenMP を用いた共有メモリ型の並列化を試みた．使用方法は，並列化したいアルゴリズムの部分に特定のソースコードを加えるだけである．本研究で使用したプログラムでは，反復法における行列・ベクトル積の演算の部分についてのみ並列処理することにした．その並列化効率については，次項に具体的な解析例とともに示す．

## ３．解析例

並列化効率を調べるため，簡単な動的解析を行った．図-3 にはそのモデルが示されている．モデルは要素数 6000，総節点数 7381，総自由度数 22143 とし，6 面体要素を使用した．底面($z$=0)の変位は $x$，$y$，$z$ 方向について完全に拘束し，上面の $x$ 軸に垂直な辺に $x$ 軸正の方向の分布荷重 $q$ を与え，1.0 秒を 1000 ステップに分割した動的解析を行った．時間積分は Newmark-$\beta$ 法による．計算には，ネイティブ 4 コアの core i7 搭載した PC を使用した．(注．ハイパースレッド技術により仮想的に 8 コアまで使用可能)また共役勾配法における解の収束条件は残差が $1\times10^{-5}$ 以下と設定した．

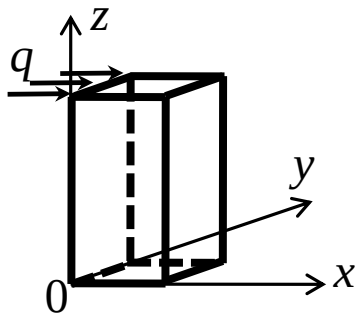


図-3　解析モデル

図-4 にはスレッド数を変えながら上述の計算を行い，それぞれ 1000 ステップ解くのに要した時間とスレッド数の関係を示している．また参考に，solver に直接法(skyline 法)を使用した場合の結果をプロットしている(並列化なし)．スレッド数を増やすほどに解析に要する時間が小さくなり，並列化効率が出ていることが確認できる．直接法の結果と比較してみると，使用スレッド数にかかわらず，反復法の方が短い計算時間で解を求めていることがわかる．

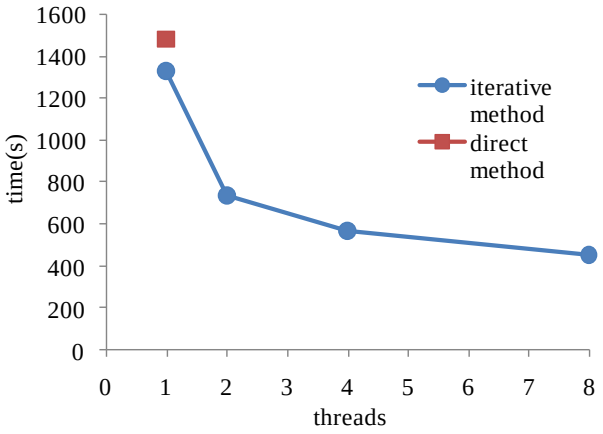


図-4　スレッド数と解析時間の関係

## ４．結論

本報告では，大規模な有限要素解析を行うためのいくつかの手法について述べた．反復法，node by node 法による使用メモリ量の低減や，並列化による計算時間の短縮を試みた．並列化については，具体的な解析例と共にその効率を示し，並列化が計算時間の短縮に有効であることを確認した．