論文の内容の要旨

論文題目　分散メモリ型並列計算を用いた 3 次元非線形有限要素法による動的応答解析手法に関する研究

氏　　名　　大西　直毅

本論文は 3 次元地盤を含む構造物の動的応答解析における計算量の大規模化に対応する一つの方法として並列計算を取り上げ，動的応答解析を行うために必要な手法についてまとめるものである。

第 1 章　序論

構造物と地盤の相互作用には，入力の相互作用と慣性の相互作用があり，さらに地盤の非線形性や地盤と構造物の境界での滑動，浮上りなどの非線形性による，応答への影響として非線形相互作用がある。弾性地盤では理論解が導かれており，成層地盤の場合には薄層要素法が広く使われているが，非線形地盤や非成層地盤では 3 次元有限要素法を用いた解析を行うことが多い。しかし 3 次元有限要素法による解析は計算する規模が地盤の自由度数の増加とともに増大し，ときに 1 台の計算機では賄いきれない。並列計算は複数の計算機で一つの問題を切り分けて計算し，結果を集約することにより，全体で 1 つの計算機として機能するようにする手法であり，より大規模な計算が行えるほか，計算機の台数を増やすことで計算速度の向上も期待できる。しかしながら既存のプログラムを並列化するには大変な労力がかかり，並列化の知識も必要となることから多くの場合スーパーコンピュータを用いるような大規模計算の場合くらいにしか取り扱われない。以上の観点から，3 次元有限要素法による動的応答解析に限り，並列計算に必要な方法と全体の解析の流れについてまとめることを本研究の目的とする。

第 2 章　並列計算

並列計算には計算機のメモリの持ち方によって大きく分けて 2 つの方法がある。一つは共有メモリ型並列計算であり，この方法では 1 つの巨大メモリを複数の CPU で共有する計算機を用いる。プログラミングでは命令を分散することだけを考慮すればよいので比較的簡単に並列化ができるが，CPU の数が増えると共有メモリへのアクセスが競合してしまう。もう一つは分散メモリ型並列計算であり，この方法は完全に独立した計算機をネットワークによって繋ぎ，通信によって各計算機の結果を共有することで 1 つの問題を解くものである。この方法では共有メモリ型のデメリットであるメモリの競合は生じないが，ネットワークが高速でないとそこがボトルネックになることや，プログラマが効率よくメモリにアクセスできるようなプログラミングを意識する必要がある。

分散メモリ型並列計算での計算機間の通信の方法で最も一般的なのがメッセージパッシングインターフェイス（MPI）である。本研究で用いたのはそのうち MPICH1 と呼ばれるものである。MPICH1 の基本は複数の CPU がそれぞれ他の CPU に対してデータをメッセージとして送受信することである。各 CPU が自分の認識 ID を持っており，データの通信では自分と送り先の認識 ID，データの格納先のアドレス，データ型，データ量，タグ番号などをしていすることで CPU 間の通信を行っている。MPI による並列化はこうした通信に必要なデータのやり取りを全てプログラマが把握している必要があり，高度なプログラミング技術を要する。

それに対して自動的に Fortran プログラムから MPI を用いたソースコードへと変換するためのツールが考案された。ハイパフォーマンスフォートラン（HPF）と呼ばれるそのツールは Fortran90 で書かれたプログラムの並列化したい配列を各計算機に分散確保することができ，その配列に関する DO ループの計算を並列化することができる。

有限要素法の計算では領域分割法を用いることがある。領域分割法は有限要素法における解析対象領域をいくつかの部分領域に分割し，各部分領域毎の計算を行い，解析対象全体の解析を行うための数値解析手法である。並列計算を効率的に行うには各計算機の負荷を均等にし，計算機間の通信をなるべく減らすことが重要であるが，この手法は部分領域毎に係数行列が作成でき，領域内部節点自由度に関しては完全に通信なしで計算が独立に行えるため，並列化に向いている。この章では領域分割法の基本的な原理についての説明を行った。また並列化とは関係ないが，有限要素法は係数行列が疎行列となるため，圧縮行格納方式（CRS）を用いて計算量を削減した。CRS は行列同士の掛け算が難しくなるが，連立方程式のソルバーを共役勾配法にすることで行列ベクトル積だけで計算できるようにしている。共役勾配法は前処理に単純な対角スケーリング行列を用いている。

アムダールの法則により並列化可能な比率が低ければいくら並列台数を増やしても並列化効率が落ち，スピードアップには上限があることがわかる。効率を考えるのであれば計算は規模に応じて各計算機のパフォーマンスを最大に利用するのがよい。また，大規模計算になればおのずと並列化可能な比率も上がるのでスケーラブルな並列計算が可能となる。

第 3 章　3 次元地盤モデル

この章では 3 次元非線形地盤の動的応答解析に必要な手法として，1) 有限要素法に基づく地盤の解析に必要な運動方程式の立式，2) 3 次元粘性境界，3) 地盤の弾完全塑性モデルについてまとめ，それらを用いて領域分割法で解くための流れについて示した。

次に作成した 3 次元地盤プログラムについて，逐次プログラムと並列プログラムの計算時間について考え，領域分割と並列プロセス数が計算時間に与える影響について調べた。まず並列プロセス数を増加させると規模が大きいほど並列化によるスピードアップが見られた。しかしながらその値は 8PE で 2〜3 程度の小さいものであることがわかった。そもそも並列化効率は 1 を上回らない。したがってパラメトリック解析を行う場合を考えれば，N 通りの計算を N 台で行えるときに，N 台で並列化して N 通りの計算を順々に行うのは無駄である。

領域分割法の分割数を増やすことでも計算の高速化が可能であり，その最適な台数についての検討を行った。1PE では通信が発生しないため分割数が多いほど計算時間は短くなったが，並列計算にするとある分割数以上で計算時間が長くなる傾向にあり，通信のオーバーヘッドとのバランスが必要であることが確認できた。また分割の方向により各プロセスが実行する計算に偏りが生じると分割数が増えても計算時間が減らない場合が生じることがあり，これを避けるには HPF では CYCLIC 分散が効果的であるが，CYCLIC 分散は DO ループ変数のアドレス解決のためのオーバーヘッドにより，他の部分で計算時間が増えることも考慮しておかなければならない。

大規模計算を解くときには 1 台で規模 $N / p$ の問題を $T$ 秒で解けたときに，規模 N の問題が $p$ 台で $T$ 秒に近い速さで解けることが理想的である。しかし領域分割の粗さがほぼ同じ場合で計算時間を比較すると，要素数が増えて通信のオーバーヘッドと収束回数の増加などにより計算時間が大幅に伸びることがわかった。

最後に 3 次元弾塑性地盤に正弦波加速度を入力した解析を行ってみた。自由度数 153015，増分ステップ数

1000, 領域分割法の不釣合力の絶対誤差, 相対誤差を 10 の-8 乗としたところ, 計算には約 12 時間を要した。

第 4 章　構造物と地盤の動的相互作用解析

第 3 章で作成した 3 次元地盤モデルと構造物との動的相互作用解析モデルの作成について説明した。ただし地盤の一部を剛体とし基礎と見立て，その剛体基礎を有する一質点せん断ばねモデルが構造物である。地盤の一部を剛体とする処理を行うには領域分割法の処理の一部に剛体導入のための行列 $R$ を乗じる操作を入れる必要がある。また前処理の対角スケーリング行列も $R$ と $R$ の転置を係数行列の対角項の左右に乗じて新たな対角スケーリング行列を作成する必要がある。

次に全体を構造物と剛体を含む地盤とに分けて計算するサブストラクチャ法について述べた。手順はまず外力と慣性力などから地盤についての運動方程式を解き，その結果得られる基礎の加速度を入力加速度として上部構造物の応答を計算，さらにそこで得られた基礎の反力を次ステップで地盤に与える外力として計算する，というものである。非線形では次ステップへの反力の持ち越しを行うと不釣合力が溜まって発散する可能性があるため注意を要するが，弾性の場合にはほとんど影響はない。

最後に，解析例として建物と地盤を弾性とした自由振動解析を行った。解析は大きさの違う地盤を 3 種類行った。地盤の領域分割は分割の粗さを揃え，計算機の PE 数は 8 とした。解析時間は地盤の自由度数に比例するかたちで増加することが分かり，大規模地盤モデルの場合でも計算時間の凡その予測が出来ることが確認できた。また自由振動解析の変位応答波形から対数減衰率による減衰定数を求め，建物の弾性一次固有周期を 0.4 秒，地盤の弾性一次固有周期を 0.53 秒としたとき，減衰定数は約 2.2%となった。ただし粘性境界は完全に反射しないわけではないので，地盤の規模が小さいときには境界からの反射波によって建物の減衰にも影響が現れることがわかった。

第 5 章　結論

3 次元有限要素法でモデル化した大規模な地盤の解析では, 領域分割法による解析領域の分割と MPI（HPF）による並列化が有効な方法であり，さらに係数行列に CRS を用いて必要メモリと計算量の削減を行うことも効果的である。本研究で作成した並列プログラムは領域分割法における領域境界上節点での適合条件を満足するように収束する部分があるために，領域分割法を用いないプログラムに比べて余計な計算が必要であるものの，計算規模が大きくなったときに領域分割による高速化と並列計算による高速化により領域分割法を用いないプログラムよりも高速化することが可能である。