音響解析ソフトウェアAdvance/FrontNoiseによる大規模解析

松原　聖*　　桑原　匠史**

Large-Scale Acoustic Simulation of Advacne/FrontNoise

Kiyoshi Matsubara* and Takuhito Kuwabara**

従来の音響解析は、あらかじめ設計上でクリティカルになる部位を設計者が想定し、その部位に対する解析を行うことが一般的であった。近年の計算機能力の飛躍的向上を背景として、産業界での製品に対する品質向上への要求から製品全体に対する大規模シミュレーションのニーズが高まっている。

このような背景から、音響解析ソフトウェア Advance/FrontNoise を、一千万要素を超える大規模な計算に適用することを目的として Advance/FrontNoise を改良した。ここでは、主として並列機能を改良し、その性能について計測した。ここで報告するモデルの最大規模は、8300 万要素、1400 万節点、2800 万自由度である。このモデルにつき、16 並列の計算機で、1 周波数当たり処理時間 2 時間で結果を得た。本稿では、大規模解析とその並列性能について報告する。

また、Advance/FrontNoise には従来からの並列機能もあったが、それは大規模計算のためというより高速な処理を行うためであった。ここでは、従来の機能と改良した機能との比較も行う。

Key word: 音響解析、並列計算機、並列化、大規模計算

1. はじめに

近年の製造業における品質保証への要求に応えるため、より精度の高い振動解析等のニーズが高まっている。また、計算機の性能向上および価格の低下により、大規模な計算により精度の高い解析が可能となってきている。その 1 つの手段として、当社では、並列化を中心としたシミュレーションの大規模化および高速化を実現するためのソフトウェアの開発に取り組んできた。

この方針のもと、当社では、音響解析ソフトウェア Advance/FrontNoise を独自に開発し、これまで、機械による環境騒音低減[1][2]や騒音低下のための機器設計[3]の一環として、音響シミュレーションに関するサービスを提供してきた。また、音響シミュレーションについては、構造解析における固有値の観点から別のアプローチも行ってきた[4]。

*アドバンスソフト株式会社　技術第 5 部
5th Technical Division, AdvanceSoft Corporation
**アドバンスソフト株式会社　技術第 3 部
3rd Technical Division, AdvanceSoft Corporation

当社における大規模シミュレーションの中心となる技術は、ソフトウェアの並列化の技術である。音響解析の周波数空間での基礎方程式を利用した音響シミュレーションを並列化する方法には、「周波数分割による並列化」と「領域分割による並列化」の 2 つの方法が考えられる。

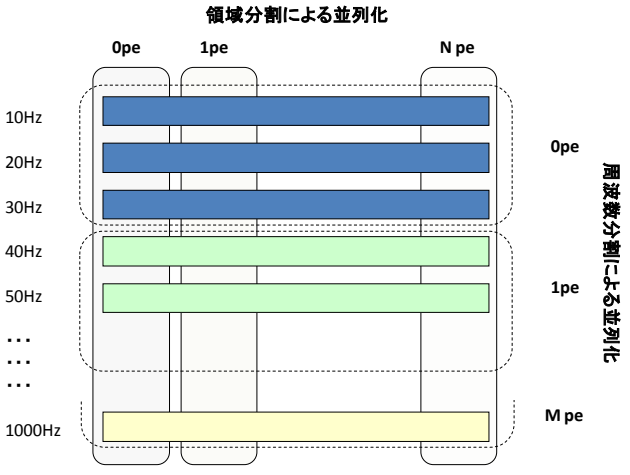

図 1.1　2 つの並列化手法の概要

このうち、「周波数分割による並列化」は、異な

る周波数を異なるプロセッサで処理する方法である。周波数分割による並列化のメリットは、並列化効率が 100%に近いこと（例えば、nCPU で処理速度が n 倍になること）、および、プログラムの実装が容易なことである。一方、並列化による 1 プロセス当たりの使用記憶容量が一定である（例えば、nCPU で使用記憶容量も n 倍必要である）ことがデメリットである。

また、「領域分割による並列化」は、解析領域を分割して異なるプロセッサに処理を担当させる方法である。領域分割による並列化のメリットとデメリットは、周波数分割による並列化と表裏一体である。つまり、そのメリットは使用記憶容量が節約できること、デメリットは並列化効率が落ちることである。

従来の Advance/FrontNoise は周波数分割の並列のみであったが、今回、領域分割による並列化の手法を実装した。本稿では、両者の比較を行い、新たに実装した領域分割による並列化で可能となった大規模音響シミュレーションの結果を紹介する。

2. Advance/FrontNoise の機能

2.1. 音響解析の基礎方程式

Advance/FrontNoise では、音響の基礎方程式を周波数空間に変換し、その周波数空間における 3 次元空間に有限要素法を適用して解く。本ソフトウェアでは、複素速度ポテンシャル $\Psi(x,t)$ を解く。

$x \in R^3$ は位置を表し、$t \in R^1$ は時間である。速度ポテンシャルに対して、粒子速度 $V(x,t)$ と音圧 $P(x,t)$ は、$V(x,t) = grad(\Psi(x,t))$、

$P(x,t) = -\rho \dfrac{\partial \Psi(x,t)}{\partial t}$ の関係式がある。

この速度ポテンシャル $\Psi(x,t)$ に対しては、波動方程式

$$\nabla^2 \Psi(x,t) = \frac{1}{c^2} \frac{\partial^2 \Psi(x,t)}{\partial t^2} \qquad (1)$$

が成り立つ。また、同様にして、

$$\nabla^2 V(x,t) = \frac{1}{c^2} \frac{\partial^2 V(x,t)}{\partial t^2} \qquad (2)$$

$$\nabla^2 P(x,t) = \frac{1}{c^2} \frac{\partial^2 P(x,t)}{\partial t^2} \qquad (3)$$

が成り立つ。ここで、速度ポテンシャル $\Psi(x,t)$ を Fourier 展開

$$\Psi(x,t) = \sum_{\nu=1,\infty} \varphi(x) e^{-i\omega t} \ 、\ \omega = 2\pi\nu \qquad (4)$$

をすると、各周波数成分 $\varphi(x)$ に対して、

$$\nabla^2 \varphi(x) + k^2 \varphi(x) = 0 \quad in \quad \Omega \qquad (5)$$

$$k = \frac{2\pi\nu}{c} \qquad (6)$$

となり、Helmholz 方程式に変換される。境界要素法では、この Helmholz 方程式を境界上の積分方程式に変換した後に離散化して解を求める。境界条件は、各周波数成分 $\varphi(x)$ に対して、

$$\alpha(x)\varphi(x) + \beta(x) \frac{\partial \varphi(x)}{\partial n} = f(x) \quad on \quad \partial\Omega \qquad (7)$$

の混合境界条件を課す。第 1 項が圧力、第 2 項が粒子速度に関する境界条件である。

2.2. 音響解析のアルゴリズム

Advance/FrontNoise では、音響の基礎方程式を周波数空間に変換し、その周波数空間において有限要素法を適用して解く。本ソフトウェアでは、どんな複雑形状に対しても容易にメッシュを作成できるように、4 面体 1 次要素を採用している。本問題はポテンシャル問題であるため、4 面体 1 次要素で精度面でも問題なく解析を実行することが可能である。4 面体要素を採用していることにより、ユーザーによるメッシュ作成の自由度が大幅に向上している。

処理時間のほとんどは、複素数の行列要素から構成される連立 1 次方程式の解を求める過程に費やされる。Advance/FrontNoise では、1 つの周波数につき、1 回の連立方程式を解いている。領域分割による並列化においては、この連立方程式を解く部分を並列化する。このため、非常に大きなサイズとなる複素数から成る行列要素を各プロセッサに分散させて使用記憶容量上に持ち、高速に並列処理することが要求される。一方で周波数分割する並列方法においては、この大きなサイズの行列を各プロセ

ッサですべて持っておく必要がある。従って周波数分割による大規模解析の並列化では、非常に大きな使用記憶容量を要することになる。

3. 計算機環境

ベンチマークを実施した計算機の環境は表 3.1 の通りである。この計算機環境は、特殊な計算機環境ではなく、比較的安価に企業で導入できる程度の計算機環境である。

表 3.1　解析に利用した計算機環境

項目	内容
OS	CentOS release 4.4 (Final)
CPU	Dual Core Opteron280(2.4GHz)
構成	16nodes(4Core/node) 2nodes(interactive node) Total 72 Cores
使用記憶容量	16GB/node(4GB/Core)

4. 周波数分割による並列性能

4.1. 目的

本節では、Advance/FrontNoise で従来から利用してきた周波数分割による並列化の実行性能について述べる。

4.2. 処理速度および使用記憶容量

次のようなサイズの問題に対して、処理時間および使用記憶容量を測定した。

表 4.1　周波数分割による 1 周波数の処理時間

No.	要素数	節点数	使用記憶容量	処理時間
①	630, 761	115, 292	150MB	2 分 02 秒
②	2, 176, 068	390, 341	490MB	16 分 30 秒
③	3, 403, 839	606, 721	780MB	32 分 07 秒

まず、使用記憶容量については、要素数に比例して大きくなる。現在では、1CPU に 2GB 程度を搭載している計算機が多いが、要素数が 300 万程度であれば、十分に 1CPU で計算することができる。使用記憶容量については、利用する計算機環境で実行可能であるかどうかの予測が可能である。また、処理時間については、ほぼ比例して増加するが、大規模になると増加の割合が大きくなる。このデータについては、領域分割の並列化手法で大規模計算を実施した結果に関する記述をした次の節で詳細に述べる。並列性能については、8CPU でほぼ 8 倍の処理速度となり、並列性能は 100%に近い。

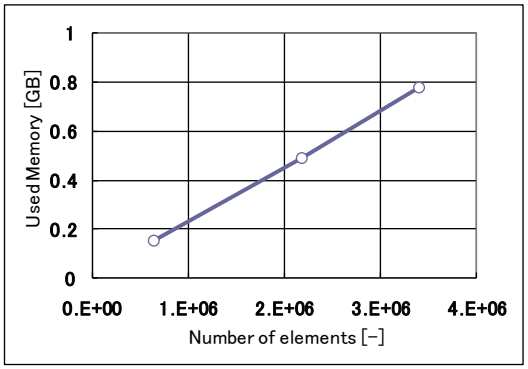

図 4.1　周波数分割による使用記憶容量（1 プロセス当たり）

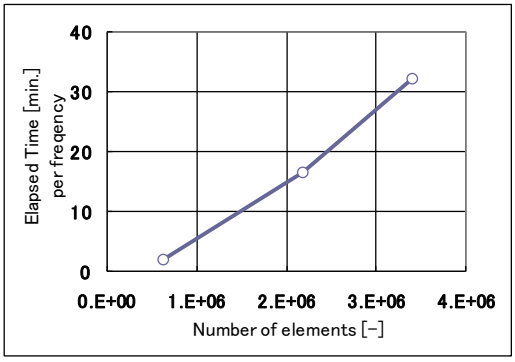

図 4.2　周波数分割による処理時間（1 周波数）

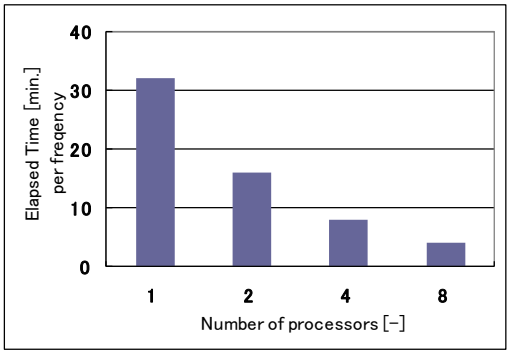

図 4.3　周波数分割による並列性能

5. 領域分割による並列化

5.1. 目的

従来の並列化手法は前節で述べた通り周波数分割であり、並列化効率については、n プロセッサでほぼ n 倍の処理速度を得ることができた。しかし、それぞれのプロセッサで 1 つの周波数を実行することが理由で、この方法は大規模化に対応できないことが課題である。

ここでは、新たに実装した領域分割による並列化機能について述べる。この並列化の目的は、周波数分割並列機能で不得意としていた部分を解決し、より大規模な計算を可能にすることである。

領域分割並列は、周波数分割の並列化手法ほどのスケーラビリティを得ることは期待できないが、どの程度まで並列性能を上げることが課題である。また、この手法で十分に効果を得ることができれば、将来的に周波数並列手法、領域分割並列手法を組み合わせて、計算機環境に最適な手法の選択をすることで、計算機資源を有効に活用することが可能となる。

5.2. 実装方法の検討

ここでは、従来の並列手法における課題を解決しつつ、従来の並列化手法のようなスケーラビリティを得ることができるか、すなわち、反復法を利用した線型ソルバの部分のスケーラビリティを得ることが課題となる。すなわち、具体的には、

① 並列化により線型ソルバの反復回数が増加しないようにすること
② 並列化により反復解法の中で 1 回の反復計算が高速化できること
③ ソルバの並列化のために、前処理および後処理に負荷がかからないこと

の 3 つの課題を解決することである。

5.3. 解決の方針

音響解析における複素行列に対して、Advance/FrontNoise で実績があり、頑強であった BiCG 系反復法に SOR の前処理を施した手法を中心にいくつかの反復法の手法を試み、領域分割による並列計算に最適な手法の組み合わせを提供する。例えば、反復法では、BiCG 系、GMRES 系、QMR 系、マルチグリッド系を試し、前処理では、Jacobi 系、ILU 系、SOR 系、加法シュワルツ系、マルチグリッド系を試した。これらの手法はいずれも、現時点で実用化され、十分に汎用的になっている反復法のアルゴリズムの中から選択した。この多くの組み合わせの中から、いくつかのテストにより、次の 4 つの手法に絞り込んだ。

表 5.1　試行した方法

方法	前処理法	反復法
方法 A	SOR 系	BiCG 系
方法 B	ILU 系	GMRES 系
方法 C	シュワルツ系	GMRES 系
方法 D	Jacobi 系	BiCG 系

5.4. 手法の選択

試作において手法の検討を行い、処理時間について、次のような結果を得た。処理時間については ILU の前処理を行い GMRES 法で処理した方法 B の結果が最も高速であった。ただし、従来から利用している SOR 系の前処理を行った BiCG 法を用いた方法 A も並列で安定に解くことができている。また、方法 C については、並列数が増えても反復回数の増加が少ないという利点があった。また、方法 D については、その他の手法と比較して特段の優位性を認めることができる点はなかった。

表 5.2　試行した方法と評価結果

方法	周波数並列	領域分割並列
方法 A	◎	○
方法 B	○	◎
方法 C	－	バックアップ
方法 D	－	バックアップ

方法 A は、従来から周波数方向の並列処理に向いており、これまでに実績がある方法である。方法 B は、今回のベンチマークにおいて、領域分割並列手

法に最も効果があることが分かった。方法 C は、並列数が増加しても反復回数が増加しない傾向にあるため、収束性は一番良好である。ただし、並列化の効率が低い。並列で収束しない場合のバックアップに利用できる。方法 D は、一般的に実績のある手法であり、この手法もバックアップとして用意しておく。

表 5.3　試行した 4 つの方法の処理時間

単位:秒

	方法 A	方法 B	方法 C	方法 D
1PE	848.665	196.312	372.95	756.79
2PE	601.238	113.751	226.232	355.086
4PE	548.944	102.679	108.727	426.55
8PE	245.635	102.5	100.454	220.273
16PE	190.523	86.447	89.561	174.965

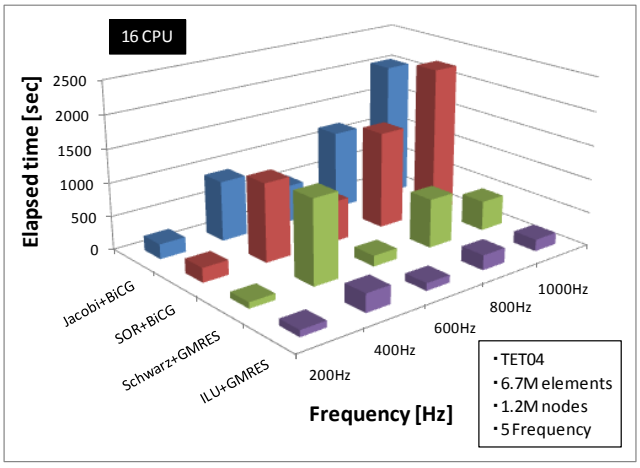

図 5.1　手法による処理時間の比較

6. 並列化性能のための検証

6.1. 目的

本検証例題では、数百万要素のデータに対して、並列性能を計測することを目的とする。具体的には、並列処理に対するデータを比較検討し、問題なく大規模計算を実施することができることを確認する。

6.2. 解析モデル

テストデータとして、マフラーとその出口の開空間を対象としたデータを利用した。ここでは、マフラーの入り口に圧力の境界条件を与え、開空間には開放の境界条件を与えている。

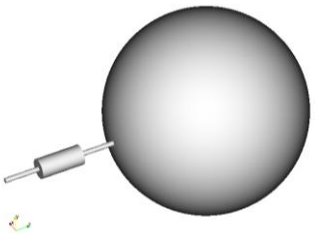

図 6.1　検証 A に利用したモデル

表 6.1　検証に利用したデータ

項目	内容
要素数	6,708,664
節点数	1,142,034
自由度数	2,284,068
周波数	100～500Hz の 5 周波数

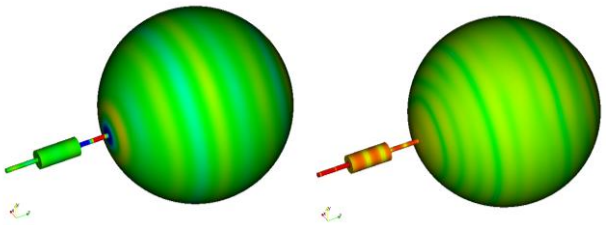

図 6.2　解析結果

(左：速度ポテンシャル、右：音圧レベル)

6.3. アルゴリズムの違いによる処理時間の比較

16CPU を利用することでプログラム全体の処理速度をほぼ 8 倍にできた。これまで実績のある方法 A よりも方法 B が高速であり、並列時の性能も低下しない。ここでは、方法 A および方法 B に関する各処理に必要とされる処理時間およびその並列化に関する処理時間について示す。

方法 A および方法 B ともに、処理時間のほとんどを、反復法の処理に費やしている。また、反復に要する時間は、収束するまでの反復数に比例するため、並列数にのみ依存しない部分もある。

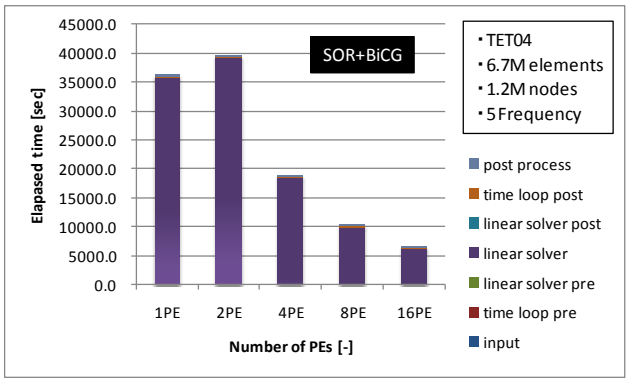

図 6.3　手法 A の並列化性能

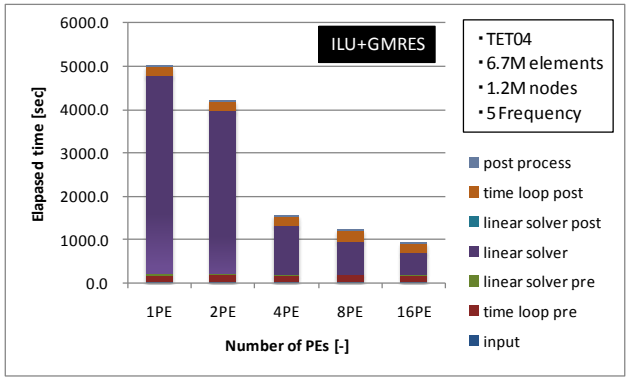

図 6.4　手法 B の並列化性能

方法 B は方法 A よりも数倍の処理速度となっている。また、方法 B において、「time loop post」という部分に時間を要し、かつ、並列数が上がっても処理時間が短縮されていない。これは、本計算ではすべての周波数で全点の結果を出力していることに起因しており、使う用途によっては削減可能な部分である。方法 A および方法 B ともに、並列数の増加により、同程度の反復回数の増加が見られる。また、方法 A および方法 B ともに反復 1 回あたりの並列性能については、十分に得られていると考えられる。

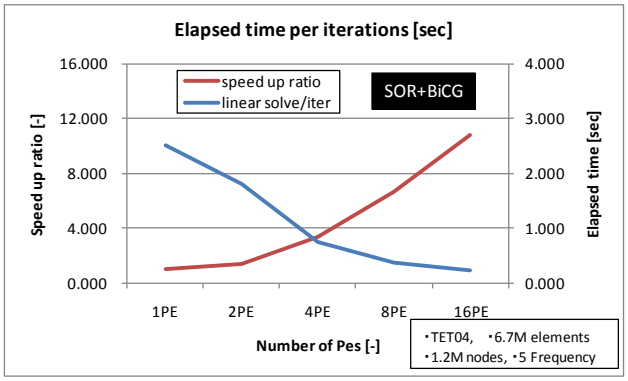

図 6.5　反復 1 回の処理の並列性能

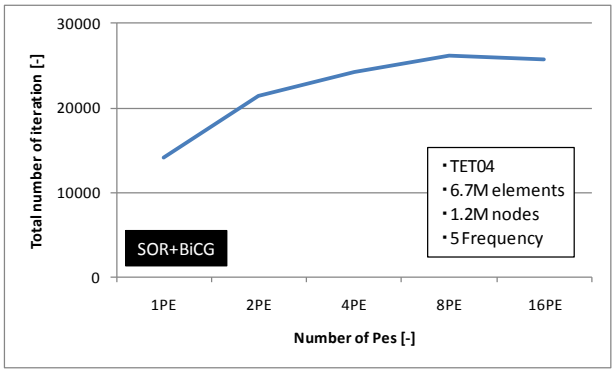

図 6.6　並列処理に伴う反復回数の変化

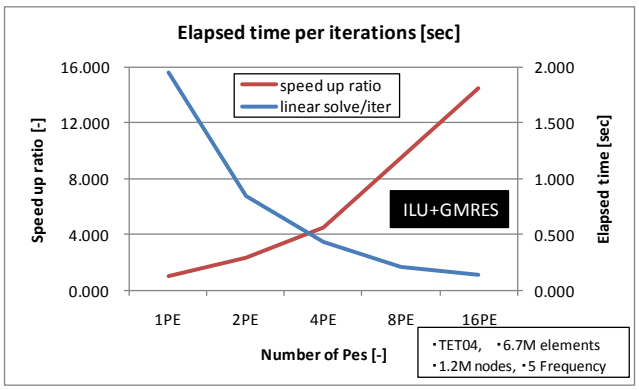

図 6.7　反復 1 回の処理の並列性能

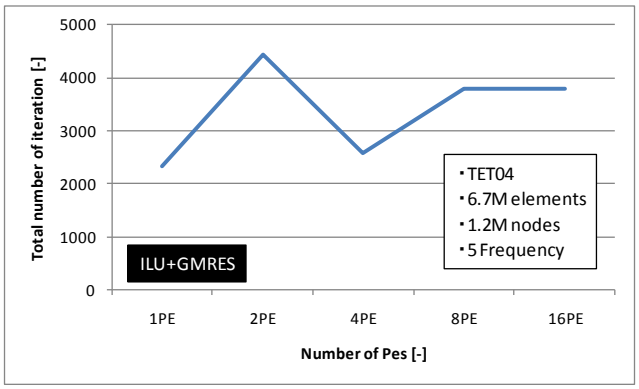

図 6.8　並列処理に伴う反復回数の変化

6.4. 使用記憶容量

音響シミュレーションの大規模計算においては、処理速度のみならず、ユーザーの利用する環境の使用記憶容量の制限がボトルネックとなる。ここでは、並列計算を実施する場合の各プロセスに必要な使用記憶容量および全体で必要な使用記憶容量について示す。

並列化した場合には、もとのデータがそのまま等分に分割されるのではなく、各プロセスで重複して持たざるを得ないデータがある。従って、並列化を

することで、解析全体で利用する使用記憶容量は増加する。今回計測したデータでは、16 プロセスを利用することにより 1 プロセスの場合の約 2 倍の使用記憶容量が必要である。しかし、重要なことは並列数を上げることで 1 つのプロセスの使用記憶容量が減少していくことであり、これにより大規模計算が可能となる。

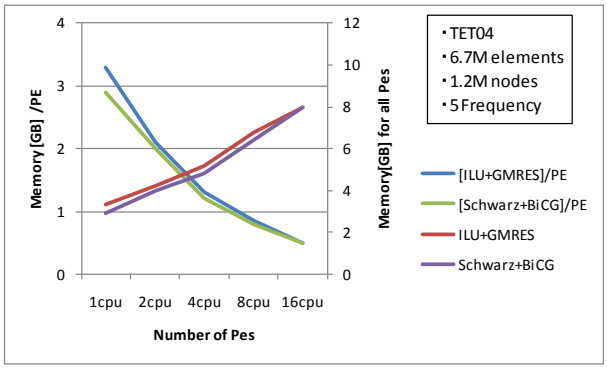

図 6.9　全プロセスと各プロセスの使用記憶容量

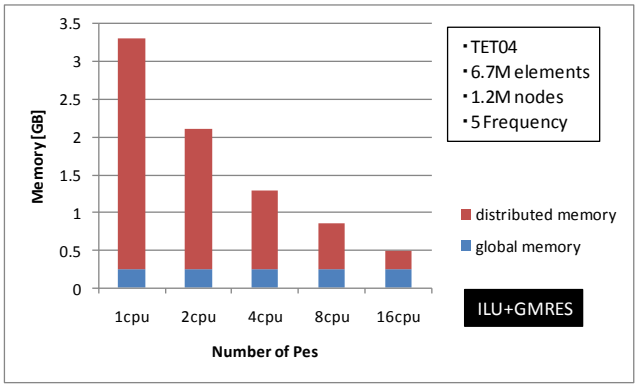

図 6.10　各プロセスの使用記憶容量

7. 大規模解析のための検証

7.1. 目的

本検証では、ここで利用している計算機環境において、どの程度の規模までの解析が実行でき、また、さらなる大規模化に対する課題等について検討するための材料を示すことを目的とした。

7.2. 解析モデル

1 辺 1m の立方体の内部に音源を設定するモデルで、メッシュサイズを自由に設定できるモデルとした。音源には、立方体の中央位置に単極子音源、双極子音源、四重極音源を設定した。また、立方体の周囲は完全反射の壁とした。

解析モデルの規模としては、次のような大きさの規模のデータを用いた。

表 7.1　検証例題の規模

No.	要素数	節点数	自由度
①	6,000,000	1,030,301	2,060,602
②	12,002,256	2,048,383	4,096,766
③	23,665,872	4,019,679	8,039,358
④	48,000,000	8,120,601	16,241,202
⑤	58,802,064	9,938,375	19,876,750
⑥	71,114,112	12,008,989	24,017,978
⑦	82,944,000	13,997,521	27,995,042

ここで示す処理時間等の測定結果は、双極子の問題で計測した数字である。単極子音源、双極子音源、四重極音源のいずれの場合にも下記のような妥当な結果を得ており、データをとるためには妥当な問題であることが分かる。

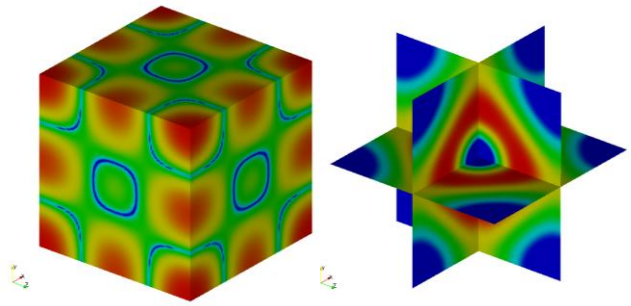

図 7.1　単極子の結果（左；立方体表面の速度ポテンシャル分布、右；中央断面の音圧レベル）

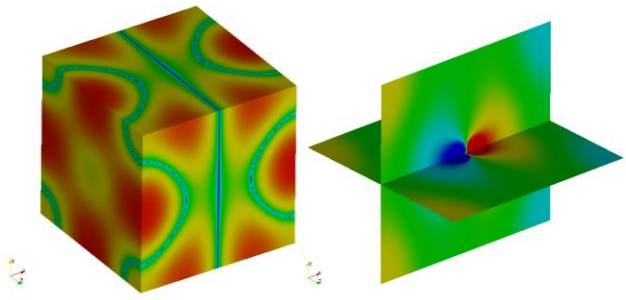

図 7.2　双極子の結果（左；立方体表面の速度ポテンシャル分布、右；中央断面の音圧レベル）

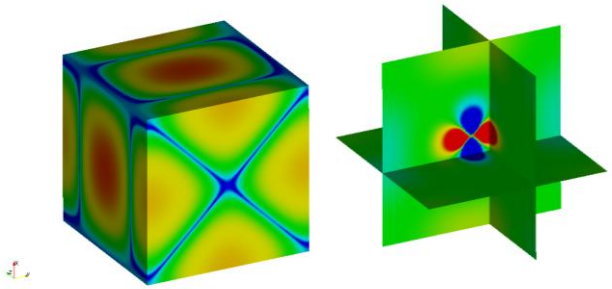

図 7.3　四重極の結果（左；立方体表面の速度ポテンシャル分布、右；中央断面の音圧レベル）

7.3. 処理時間

本検証では、3 節に示した計算機の 16CPU を利用して実施した。その経過時間および収束までに要する反復回数を示す。経過時間は、処理が開始されてから終了するまでの実時間を計測している。

表 7.2　16CPU における処理時間

単位：秒

自由度	経過時間 [sec]	ソルバ [sec]	入出力 [sec]	反復数 [回]
1,000,000	118.00	43.76	74.25	551
2,000,000	225.00	83.86	141.14	547
4,000,000	608.00	334.29	273.71	959
8,000,000	1,830.00	1,276.31	553.69	1,759
10,000,000	3,150.00	2,114.36	1,035.64	1,204
12,000,000	5,030.00	3,794.80	1,235.20	2,488
14,000,000	6,570.00	5,138.70	1,431.30	1,908

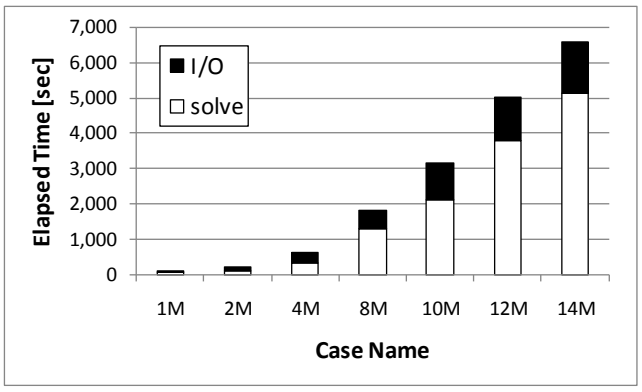

図 7.4　解析規模と経過時間（16CPU）

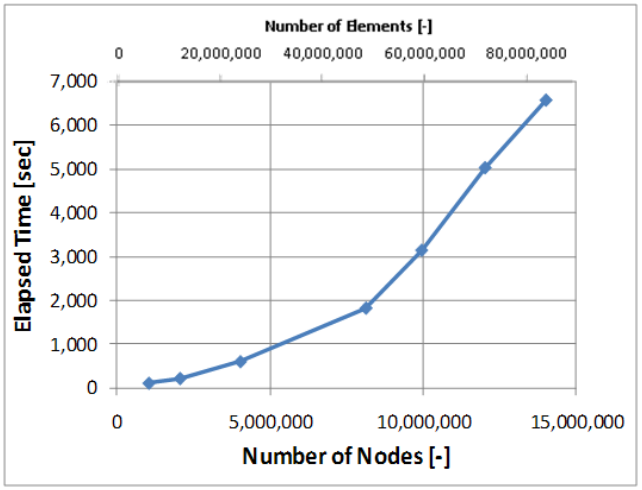

図 7.5　節点数と処理時間

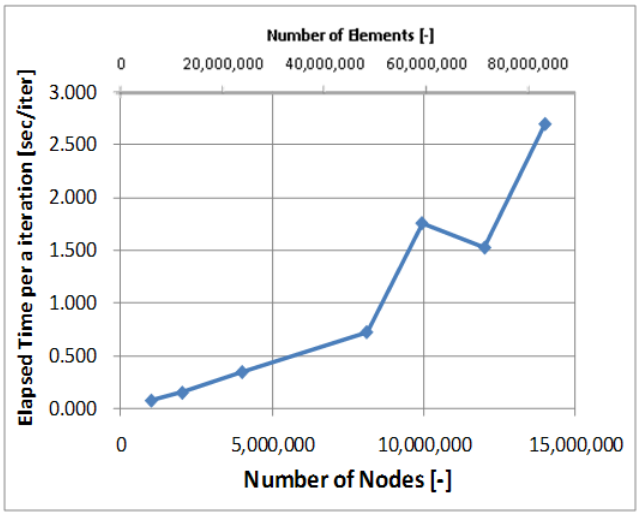

図 7.6　節点数と 1 反復当たりの処理時間

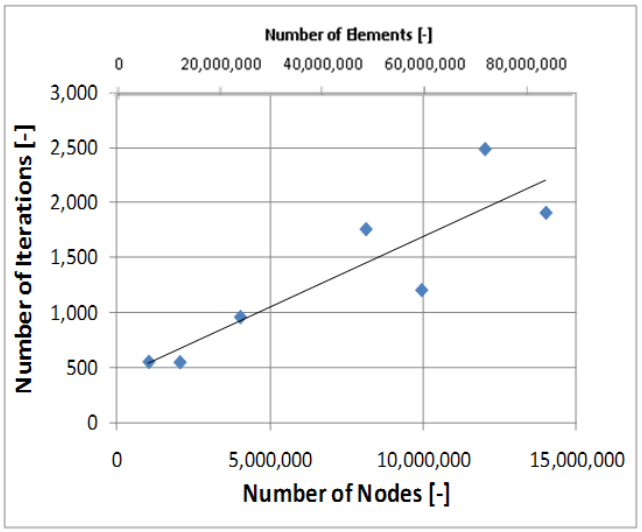

図 7.7　節点数と反復回数

7.4. 使用記憶容量

16CPU を利用した場合の使用記憶容量を下記に示す。ここで利用した 16CPU で 64GB の使用記憶容量の計算機では、要素数 82,944,000、節点数 13,997,521、自由度数 27,995,042 のモデルが実行できた。使用記憶容量については、下記のグラフにより容易に見積もることができる。また、1CPU における使用記憶容量については、6 節の内容から見積もることができる。

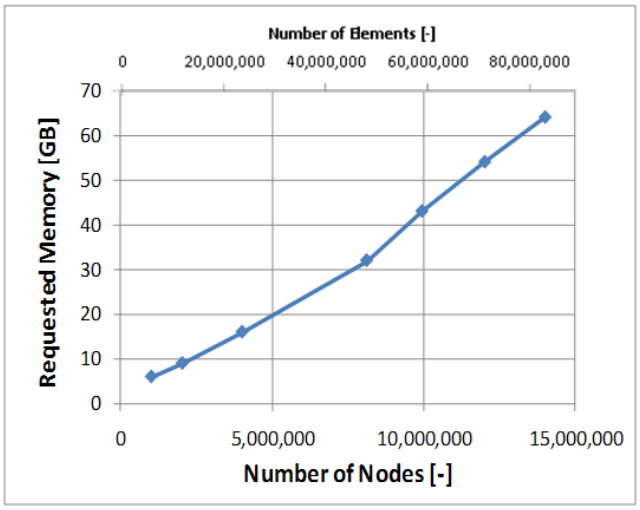

図 7.8　使用記憶容量（16CPU 利用時）

8. まとめ

当社のソフトウェアでは、流体解析、構造解析、音響解析ともに、まるごと解析が実施できることを 1 つの目標としている。音響解析ソフトウェア Advance/FrontNoise においても、大規模解析を行うために、頑強で並列効率を得ることができる実用的な並列アルゴリズムを実装した。

本稿の実施内容により、16CPU で 8 倍程度の高速化が可能となった。4 面体 1 次要素 2000 万要素のデータでは、並列化効率は、16PEs で 8 倍程度、使用記憶容量は 1PE あたり 500MB 程度であった。また、1PE あたり 4GB の使用記憶容量で 16CPU を利用すると、4 面体 1 次要素 8300 万要素程度（1400 万節点）が 2 時間程度で可能となった。

また、現状の Advance/FrontNoise では、周波数分割による並列化と領域分割による並列化は別々の機能となっている。周波数分割による並列化、領域分割による並列化の双方の利点を利用すべく、そのハイブリッドな並列化方法も取り入れる価値があると考えられ、今後の課題として検討する。

Advance/FrontNoise に関しても、今後、まるごと解析のために必要とされるさらなる大規模解析の検証を行う。

参考文献

[1] 赤間誠他, “鉄道用低応力・低騒音軽量車輪の開発,” 日本機会学会論文誌 A, Vol.73, No.730, (2007.07)

[2] M.Akama, et al., "Design and analysis of low-stress and low-noise lightweight railway wheel," Proc. IMechE Vol. 223 Part F: J. Rail and Rapid Transit, (June, 2008)

[3] 桑原匠史, “Advance/FrontNoise を用いた音響解析,” アドバンスシミュレーション, Vol. 2, (2010.9)

[4] 松原聖, 桑原匠人ら, “数千万自由度を対象とした大規模並列固有値ソルバー,” 日本機械学会, 第 19 回計算力学講演会, (2006.11.05)