

大規模固有値計算ソフトウェア Advance/NextNVH

松原 聖* 桑原 匠史**

The solver for large scale eigenvalue problems, Advance/NextNVH

Kiyoshi Matsubara* and Takuhito Kuwabara**

振動設計における「構造物の軽量化」、「環境適合性や快適性」、「嗜好性を対象とした音質」などの問題を解決するための大規模固有値計算ソフトウェアは、品質・環境対策・コスト削減などの開発・設計技術を大幅に向上させる可能性がある。本稿では、大規模固有値計算ソフトウェア Advance/NextNVH で採用した新規手法である多階層モード座標結合モード合成法による大規模固有値計算の手法と性能について述べる。

Key words: eigenvalue problems, large-scale, parallel processing, mode synthesis

1. はじめに

構造設計において、製品性能と環境対策やユーザー満足度の要求を満たすための設計技術として、最適な音響/構造連成を求めるシミュレーション技術が要求されている。現在の音響構造連成システムでは、動的応答（変位、応力、最大音圧、音圧積分、モード共振ピークの分散など）を最適化する構造を求めることが困難な機能上の問題と、使い勝手の点で利用上の制約がある。

その解決案のひとつは、系全体を最適化変更の対象として、固有値問題を解くことである[1]。具体的には、自動車等の設計における振動・音響問題で必要となる数千万自由度に対する数千個の固有値を計算機で求める問題を解くことを可能にすることである。ここでは、Householder法やLanczos法のように直接固有値を求める方法ではなく、計算機の処理速度や記憶容量で有利なモード結合による多階層のモード合成法等の近似解法が必須である。しかし、これまでに知られていた実空間とモード空間を交互に解く方式の多階層モード合成法[1]では、大規模になればそれらの行列を関係付ける処理に非

現実的な処理時間がかかり、大規模な問題への適用が困難であった。

一方では、近年の計算機（ハードウェア）の性能は飛躍的に向上している。また、並列計算機を利用したシミュレーション技術の進展にも目を見張るものがある。また、現在の構造解析ソフトウェアでは、大規模固有値解析向けのLanczos法等でかなり大規模の問題も解けるようになってきた。しかし、サブスペース法やLanczos法のように系全体の剛性行列や質量行列を持ち、それら全体を同時に処理する必要のあるアルゴリズムは、数千自由度ともなるとなかなか適用が難しくなっている。

このような背景のもと、大規模構造モデルの音響構造連成応答を最適化解析するシミュレーションのための固有値問題を解く計算機プログラムが必要となってきた。

本提案によるモード結合による多階層のモード合成法では、解析領域全体を木の形式で多階層に領域分割し、その領域において実空間とモード空間を完全に分離する[4]。木に領域分割した場合の根元を上階層、枝の方を下階層と呼ぶ。まず、実空間を下階層から上階層に順番に固有値問題を解く。ここでは、その階層ごとの実空間の固有ベクトルを基底として、上階層から下階層にモード空間での固有値問題を解き、最終的に全体領域の固有値・固有ベクトルを求めるアルゴリズムを構成した。

*アドバンスソフト株式会社 代表取締役社長

President, AdvanceSoft Corporation

**アドバンスソフト株式会社 第1事業部

1st Computational Science and Engineering Group,

AdvanceSoft Corporation

本稿では、そのアルゴリズムの詳細を述べるとともに、そのアルゴリズムをソフトウェアとして実現して検証した結果について述べる。

2. 機能

本ソフトウェアでは、実 n 次元空間における一般化固有値問題 $Kx = \lambda Mx$ を対象とする。すなわち、 $n \times n$ 対称行列 K と M が与えられたときに、実固有値 λ および n 次元固有ベクトル x を求める。特に、行列 K および M がスパース行列（行列の多くの要素が0である行列）である場合に有効なアルゴリズムについて述べる。一般的に、大規模な工学的な振動問題では、スパース率は99.9%以上である。ここでは、 n が数千万に対して、数千個の固有値を求める問題を対象とするものである。

3. 理論概要

3.1. アルゴリズム

本稿で提案するアルゴリズムは、すべての一般化固有値問題に適用可能な方法であり、また、先に述べた規模の大規模な問題に対しては、特に有効な高速演算処理方法に関する手法である。

前節で述べた方法（既存の多階層のモード座標結合モード合成法のアルゴリズム）では、2.3節の(3)(4)の処理において、上位領域と下位領域の関係を表す行列（[1]におけるP行列）の計算に多大な処理時間がかかることが短所であった。本稿で提案する新しい方法のひとつの特長は、処理時間を要するP行列を計算する必要のない、従来の方法に替わる高速演算処理手法である。そのため、ここでは次のようなアルゴリズムを利用する。

ここまでの道具立てを利用した高速な多階層モード座標結合モード合成法のアルゴリズムは次の通りである。

(1)内部領域の固有値問題；まず、最下位の階層の領域（内部領域） $k=m$ における内部領域群の固有値問題 $Kx = \lambda Mx$; D^m_i で固有値・固有ベクトルを求める。
(2)境界領域の固有値問題；その親($k-1$ 階層)の静縮合された固有値問題 $Kx = \lambda Mx$; $D^k_{B,j}$ で固有値・固有ベクトルを求める。さらに、 k をひとつずつ小さく

くして、すべての境界領域 $D^l_{B,j}$ ($l=1, m$) の静縮合された固有値・固有ベクトルを求める。次に、 $k=1$ として、

(3)モード空間での固有値問題；この2つの固有値問題の固有ベクトルを基底として、

$Kx = \lambda Mx$; $\left(\bigcup_{i=1, P_i^{(l)}} D^k_i\right) \cup D^k_{B,j}$ をモード空間で、固有値・固有ベクトルを求める。

(4)固有ベクトルの計算；モード空間で求めた固有ベクトルを $\left(\bigcup_{i=1, P_i^{(l)}} D^k_i\right) \cup D^k_{B,j}$ （実空間とも呼ぶ）における固有ベクトルに変換する。

(5)1階層下の階層へ；この固有値問題の解を改めて、境界領域の解と考え、さらに $k+1$ 階層の固有値問題を解く。すなわち、その子を含めたモード空間での固有値問題を解くステップ(3)に戻る。最終的に、 $k=m$ となるまで同じ処理を行い、最終的な固有値・固有ベクトルが、一般化固有値問題 $Kx = \lambda Mx$ の解である。

ここに示した新しいアルゴリズムを図2に示す。

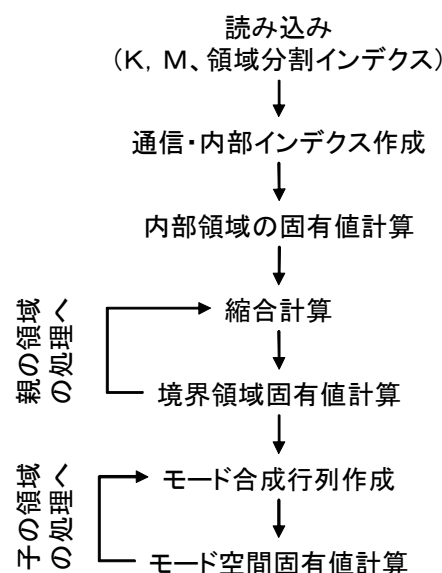


図 1 新規開発した多階層モード座標結合モード合成法アルゴリズム

3.2. 特長

ここで開発したアルゴリズムは次のような3つの特長がある。

- ・ 新規アルゴリズムにより、高速なモード空間で

の固有値計算が可能である。

- ・ モード空間では、安定な固有値計算が可能な定式化を利用している。
- ・ 物理空間およびモード空間での固有値計算には最適な固有値解法を採用し、頑強なアルゴリズムとなっている。

3.3. 記号の定義

上記の課題を解決するために、領域分割を次のように定義する。ベクトル x の n 成分をノードとし、行列 K および M の非零成分をエッジとするグラフ

D^1 を考える。このグラフを p 個に分離されたグラフ $D^1_i (i=1, p_1)$ とその境界のノードで構成されるグラフ $D^1_{B,j} (j=1)$ を作る。すなわち、 D^1_i と $D^1_{B,j}$ のノード間にはエッジ $D^1_{i,j}$ は存在するが、 D^1_i と $D^1_j (i \neq j)$ のノード間にエッジは存在しない。また、 $D^1_{B,j} (j=1)$ と $D^1_i (i=1, p_1)$ および $D^1_{B,i}$ で、もとのグラフ D^1 全体を構成する。分割された $D^1_i (i=1, p_1)$ を子、その子とエッジを共有する $D^1_{B,j} (j=1)$ を親と呼ぶ。この操作を、 $D^1_i (i=1, p_1)$ に対して再帰的に行うことにより、 $D^2_i (i=1, p_2)$ と $D^2_{B,j} (j=1, p_1)$ を構成する。注意すべきことは、 $D^2_{B,j}$ の個数が D^1_i の個数となることである。さらに、 $D^m_i (i=1, p_m)$ に対して、同様の操作を行えば、 $D^{m+1}_i (i=1, p_{m+1})$ と $D^{m+1}_{B,j} (j=1, p_m)$ を構成できる。子と親の関係も同様に定義できる。また、ひとつの親 $D^m_{B,j}$ の子たちを $D^m_i (i=1, p_m^{(j)})$ と書くことにする。

階層を次のように定義する。領域分割の手順により、次の階層のグラフ（以下、領域とも呼ぶ）が定義できる。

$$\begin{aligned} D^1 &\rightarrow D^1_{B,j} (j=1) \rightarrow D^2_{B,j} (j=1, p_1) \rightarrow \cdots \\ &\rightarrow D^m_{B,j} (j=1, p_{m-1}) \rightarrow D^m_j (j=1, p_m) \end{aligned} \quad (1)$$

ここで、 $D^1_{B,j} (j=1)$ を階層 1、 $D^2_{B,j} (j=1, p_1)$ を階層 2、 $D^m_{B,j} (j=1, p_{m-1})$ を階層 m の（境界）領域と

呼び、最後の $D^m_j (j=1, p_m)$ を $m+1$ 階層の領域または内部領域と呼ぶ。

縮合を次のように定義する。ある内部領域または境界領域およびその子すべてを含む領域 D において、 $Kx = \lambda Mx$ を考える。ここでは、他の領域の成分は、 0 と考える。このような問題を、 $Kx = \lambda Mx; D$ と書くことにする。この問題は、 D のノードの個数の次元 $\dim(D)$ を持つ。ここで、

$$Kx = \lambda Mx; \left(\bigcup_{i=1, p_m^{(j)}} D^m_i \right) \cup D^m_{B,j} \quad (2)$$

を考える。 $\left(\bigcup_{i=1, p_m^{(j)}} D^m_i \right) \cup D^m_{B,j}$ は、境界領域およびその子すべてを含む領域である。

これは、 $\dim(D^m_{B,j}) + \sum_{i=1, p_m^{(j)}} \dim(D^m_i)$ の次元を持つ。この問題に対して、

$$Kx = 0; D^m_i \cup D^m_{B,j} \quad (i=1, p_m^{(j)}) \quad (3)$$

の制約条件をつける。これは、 $\sum_{i=1, p_m^{(j)}} \dim(D^m_i)$ 個の式であり、(1)と(2)を連立させて解けば $\dim(D^m_{B,j})$ の次元の固有値問題となる。これを、 $\left(\bigcup_{i=1, p_m^{(j)}} D^m_i \right) \cup D^m_{B,j}$ から $D^m_{B,j}$ に静縮合した固有値問題と呼ぶ。これも内部領域と同様に、

$Kx = \lambda Mx; D^m_{B,j}$ と書くものとする。これは、一般に知られている Guyan の静的縮約を含む概念である。以下、親の領域を上位の領域と呼び、子の領域を下位の領域と呼ぶこともあるが、本稿では、それらは同値な用語として利用している。また、最下位の階層に所属する領域を、その形状から内部領域と呼ぶこともある。

3.4. 縮合の処理

縮合の演算を次のように定義する。本アルゴリズムでは、縮合は K と M に関する縮合からなる。両者は基本的に同じ手順であるが、詳細な式が若干異なることに注意する。まず、1 階層を対象とした K の縮合は、

$$\begin{bmatrix} K_{1,j}^m & 0 & 0 & 0 & 0 & K_{1,j}^m \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & K_{i,j}^m & 0 & 0 & K_{i,j}^m \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & K_{p_m^{(i)},j}^m & K_{p_m^{(i)},j}^m \\ K_{1,j}^{m,T} & \dots & K_{i,j}^{m,T} & \dots & K_{p_m^{(i)},j}^{m,T} & K_{B,j}^{m,T} \end{bmatrix} \quad (4)$$

から

$$K_{B,j}^{m'} = K_{B,j}^m - \sum_{i=1, p_m^{(i)}} K_{i,j}^{m,T} K_{i,j}^{m-1} K_{i,j}^m \quad (5)$$

を構成する手続きである。全体の K の縮合は、この手続きを再帰的に実行する手順である。次に、1 階層を対象とした M の縮合は、

$$\begin{bmatrix} M_{1,j}^m & 0 & 0 & 0 & 0 & M_{1,j}^m \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & M_{i,j}^m & 0 & 0 & M_{i,j}^m \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & M_{p_m^{(i)},j}^m & M_{p_m^{(i)},j}^m \\ M_{1,j}^{m,T} & \dots & M_{i,j}^{m,T} & \dots & M_{p_m^{(i)},j}^{m,T} & M_{B,j}^{m,T} \end{bmatrix} \quad (6)$$

から

$$M_{B,j}^{m'} = M_{B,j}^m - \sum_{i=1, p_m^{(i)}} M_{i,j}^{m,T} K_{i,j}^m M_{i,j}^m K_{i,j}^{m-1} M_{i,j}^m \quad (7)$$

を構成する手続きである。全体の M の縮合は、この手続きを再帰的に実行する手順である。すなわち、静縮合とは、

$$\begin{bmatrix} K_{1,j}^m & 0 & 0 & 0 & 0 & K_{1,j}^m \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & K_{i,j}^m & 0 & 0 & K_{i,j}^m \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & K_{p_m^{(i)},j}^m & K_{p_m^{(i)},j}^m \\ K_{1,j}^{m,T} & \dots & K_{i,j}^{m,T} & \dots & K_{p_m^{(i)},j}^{m,T} & K_{B,j}^{m,T} \end{bmatrix} x = \lambda \begin{bmatrix} M_{1,j}^m & 0 & 0 & 0 & 0 & M_{1,j}^m \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & M_{i,j}^m & 0 & 0 & M_{i,j}^m \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & M_{p_m^{(i)},j}^m & M_{p_m^{(i)},j}^m \\ M_{1,j}^{m,T} & \dots & M_{i,j}^{m,T} & \dots & M_{p_m^{(i)},j}^{m,T} & M_{B,j}^{m,T} \end{bmatrix} x \quad (8)$$

を

$$K_{B,j}^{m'} x = \lambda M_{B,j}^{m'} x \quad (9)$$

$$K_{B,j}^{m'} = K_{B,j}^m - \sum_{i=1, p_m^{(i)}} K_{i,j}^{m,T} K_{i,j}^{m-1} K_{i,j}^m \quad (10)$$

$$M_{B,j}^{m'} = M_{B,j}^m - \sum_{i=1, p_m^{(i)}} M_{i,j}^{m,T} K_{i,j}^m M_{i,j}^m K_{i,j}^{m-1} M_{i,j}^m \quad (11)$$

の固有値問題に順次変換する再帰的な手順である。

次に、モード空間を次のように定義する。まず、領域 $(\bigcup_{i=1, p_m^{(i)}} D_{B,j}^m) \cup D_{B,j}^m$ における固有値問題を

考える。(1)境界領域の静縮合された固有値問題

$Kx = \lambda Mx; D_{B,j}^m$ の固有ベクトル、および、(2)内部領域 $Kx = \lambda Mx; D_{B,j}^m$ の固有ベクトルを基底とする関数空間(これらの固有ベクトルの線形結合で作られる空間)を考え、その空間の範囲で、

$$Kx = \lambda Mx; (\bigcup_{i=1, p_m^{(i)}} D_{B,j}^m) \cup D_{B,j}^m \quad (12)$$

を解く。これをモード空間の固有値問題と呼ぶ。すなわち基底空間の係数 η をモード空間の座標として利用し、任意の x は $x = T\eta$ と表現する。この表現を利用して、 $Kx = \lambda Mx$ を $(T^T K T)\eta = \lambda (T^T M T)\eta$ と変換し、モード空間での固有値問題に帰着する。 $Kx = \lambda Mx; D_{B,j}^m$ の固有ベクトルを、 $\phi_i^m (i=1, p_m^{(i)})$ とおき、 $Kx = \lambda Mx; D_{B,j}^m$ の固有ベクトルを、 $\phi_{B,j}^m$ とする。

3.5. モード空間での処理

モード空間の演算を次のように定義する。モード空間の基底については、次のような方法で求める。まず、 $Kx = \lambda Mx; D_{B,j}^m$ の固有ベクトルを $\phi_i^m (i=1, p_m^{(i)})$ とおき、 $Kx = \lambda Mx; D_{B,j}^m$ の固有ベクトルを、 $\phi_{B,j}^m$ とおく。モード空間の基底は、

$$\Phi_{1,j}^{m,T} = (\phi_{1,j}^m \ 0 \ 0 \ 0 \ 0 \ 0)^T \quad (13)$$

...

$$\Phi_{i,j}^{m,T} = (0 \ 0 \ \phi_{i,j}^m \ 0 \ 0 \ 0)^T \quad (14)$$

...

$$\Phi_{p_m^{(i)},j}^{m,T} = (0 \ 0 \ 0 \ 0 \ \phi_{p_m^{(i)},j}^m \ 0)^T \quad (15)$$

$$\Phi_{B,j}^{m,T} = (\phi_{B,j}^{m(1)} \ \dots \ \phi_{B,j}^{m(i)} \ \dots \ \phi_{B,j}^{m(p_m^{(i)})} \ \phi_{B,j}^m)^T$$

である。ここで、最後の基底は、Guyan の静的縮約条件

$$\begin{bmatrix} K_{1,j}^m & 0 & 0 & 0 & 0 & K_{1,j}^m \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & K_{i,j}^m & 0 & 0 & K_{i,j}^m \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & K_{p_m^{(i)},j}^m & K_{p_m^{(i)},j}^m \\ K_{1,j}^{m,T} & \dots & K_{i,j}^{m,T} & \dots & K_{p_m^{(i)},j}^{m,T} & K_{B,j}^{m,T} \end{bmatrix} \begin{bmatrix} \phi_{B,j}^{m(1)} \\ \vdots \\ \phi_{B,j}^{m(i)} \\ \vdots \\ \phi_{B,j}^{m(p_m^{(i)})} \\ \phi_{B,j}^m \end{bmatrix} = 0 \quad (16)$$

を満たす。すなわち、

$$\phi_{B,j}^{m(i)} = -K_i^{m-1} K_{i,j}^m \phi_{B,j}^m \quad (17)$$

として決定できる。これらの基底を利用して

$$T = (\Phi_{m_1} \cdots \Phi_{m_i} \cdots \Phi_{m_{p_m}^{(i)}} \Phi_{B,j}^m)$$

とする。 $Kx = \lambda Mx$ は、 $x = T\eta$ を利用すると、

$$(T^T K T) \eta = \lambda (T^T M T) \eta \text{ となって、モード空間での}$$

固有値問題に帰着できる。また、 $x = T\eta$ により、

モード空間座標を物理空間座標に変換することができる。ただし、ここで、注意すべきことは、

$\phi_{B,j}^{m(i)} = -K_i^{m-1} K_{i,j}^m \phi_{B,j}^m$ のうち K_i^{m-1} は少ない演算回数で計算できることである。

ここで、具体的な処理においても

$\phi_{B,j}^{m(i)} = -K_i^{m-1} K_{i,j}^m \phi_{B,j}^m$ を求める必要がある。 K_i^m は

$\phi_{B,j}^m$ 以下の領域に関する K 行列全体であり、

K_i^{m-1} 逆行列がすでに求められているため、高速に処理することができる。具体的には、モード空間では、

$$\begin{bmatrix} \lambda_i I & 0 & 0 & 0 & 0 \\ 0 & \lambda_i I & 0 & 0 & 0 \\ 0 & 0 & \lambda_i I & 0 & 0 \\ 0 & 0 & 0 & \lambda_i I & 0 \\ 0 & 0 & 0 & 0 & \lambda_i I \end{bmatrix} \begin{bmatrix} I & 0 & 0 & 0 & \phi_1^T (M_{11} G_{1B} + M_{1B}) \phi_B \\ & I & 0 & 0 & \phi_2^T (M_{22} G_{2B} + M_{2B}) \phi_B \\ & & I & 0 & \phi_3^T (M_{33} G_{3B} + M_{3B}) \phi_B \\ & & & I & \phi_4^T (M_{44} G_{4B} + M_{4B}) \phi_B \\ -\lambda & & & & I \end{bmatrix} \begin{bmatrix} \eta \\ \eta \\ \eta \\ \eta \\ \eta \end{bmatrix} = 0 \quad (18)$$

を解くことになり、この演算のみを行うことで十分である。

4. 解析事例

4.1. 解析条件

検証例題として 160 万自由度の問題を解いた。これは実用的な例題であり、シェルを中心とした有限要素法でモデル化を行ったモデルに、マスやバネを追加したモデルである。また、質量行列と剛性行列が異なる非零パターンを持つようなケースとなっている。

本検証例題において、精度比較には Strum 列によるチェックを利用した。また、Strum 列で正しいことが確認されたデータ（具体的には、計算パラメータをかなり安全側にとった条件で得られた結果）を

正しいとして、それ以外のケースを比較した計算も行った。

本検証例題で比較した項目は、下記の項目の通りである。

- Strum 列の解析結果から得られた固有値の数と本固有値ソルバーで得られた固有値数の比較による解の妥当性チェック
- 上限周波数および s パラメータに対する処理時間および精度の検証
- 領域分割数に対する処理時間と精度の検証
- 最小固有値パラメータに対する処理時間と精度の検証
- 並列計算時の処理時間と速度向上比の検証

これらの項目に関し、実施した検証結果を以下の節で述べる。

4.2. Strum 列による結果との比較

本固有値ソルバーに付属する Strum 列算出機能を利用して、Strum 列による結果と本固有値ソルバーで得られた結果の比較を行った。まず、対象とした問題に対し、50Hz、100Hz、120Hz、200Hz について、Strum 列を求め、その周波数以下の固有値の数を求めた。また、本固有値ソルバーでは、周波数の小さい方から順番に解が得られる。従って、Strum 列の結果と比較することにより、本固有値ソルバーの妥当性がチェックできる。下図は、Strum 列から得られた結果と本固有値ソルバーで得られた固有値を比較したものである。横軸が固有値数であり、縦軸が対応する周波数である。また、その次の図は、周波数 100Hz 付近を拡大して示した図である。本固有値ソルバーで得られた結果上に、Strum 列で得られた結果が乗っていることから、本固有値ソルバーで得られた結果と Strum 列で得られた結果は完全に一致していることが分かる。

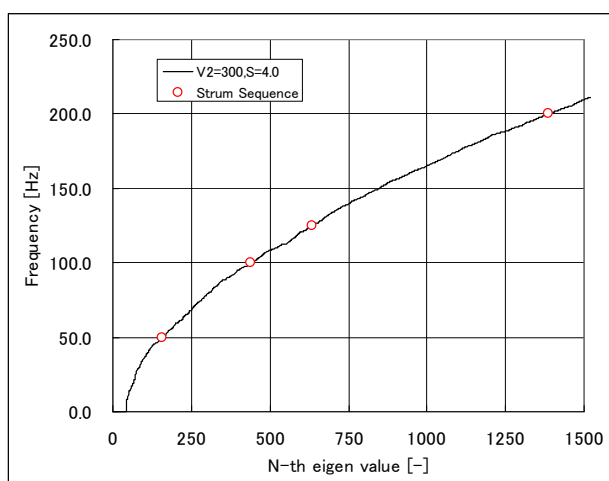


図 2 Strum 列結果との精度比較

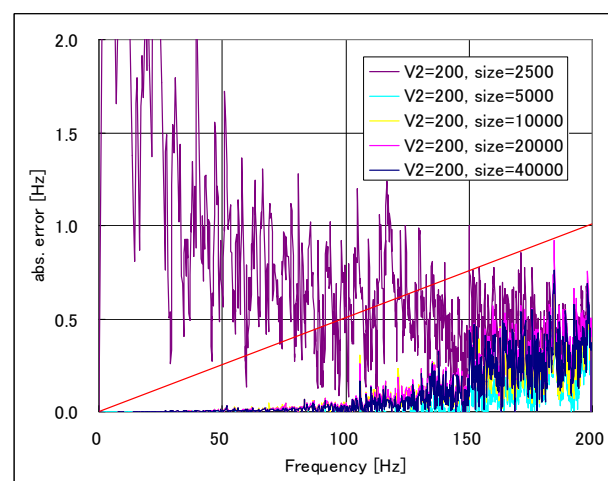


図 4 精度の領域分割サイズ依存性

4.3. 領域分割サイズ依存性

本検証例題は、160 万自由度であり、領域分割数による処理速度と精度を検討する必要がある。本節では、その結果を示す。まず、前節までで妥当な解が得られている領域分割数 512 を基準とし、64 領域～1024 領域まで、2 の累乗数の領域で領域を分割とし、その他のパラメータを同じにして処理時間を確認した。ここでその他のパラメータについては、他の検証ケースで得られた推奨値を利用した。処理時間については、細分化しすぎると処理時間が遅くなるが、512 分割以下、または 1 領域の自由度数が 5000 以上であれば、処理時間は変化ない。また、図 10 から領域数が小さいと精度も得られにくいことが分かる。

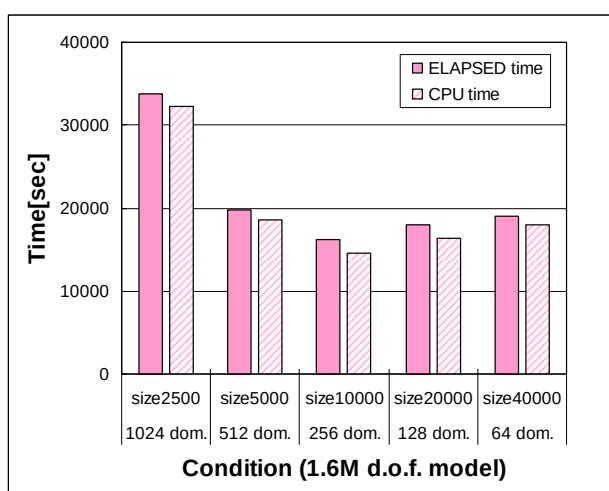


図 3 処理時間の領域分割サイズ依存性

4.4. 上限周波数依存性

ここでのパラメータは、ユーザーの与える上限周波数： f [Hz]、および、計算内部で設定する上限の周波数と f との比： s [-] の 2 つのパラメータが重要である。例えば、200Hz までの周波数を求める場合には、 $f=200\text{Hz}$ と指定し、このケースで $s=2.0$ と指定すれば、本固有値ソルバー内部では、 $f \cdot s=200\text{Hz} \times 2.0=400\text{Hz}$ までの固有値が計算され、自動的に 200Hz までの固有値が出力されることになる。

まず、 $s=4.0$ に固定した場合の上限周波数の処理時間への依存性を確認した。このテスト項目については、当然上限周波数が多いほど処理時間は長くなる。ほぼ周波数に比例する程度に処理時間は増大する。

次に、 $f=200$ に固定し、 $s=1.5, 2.0, 2.5, 3.0, 4.0$ とした場合に、その処理時間と精度について検証した。処理時間については、当然 $f \cdot s$ に比例する結果となったが、ここで確認したいこと、および、実用上重要なのは精度である。精度については、 $s=2.0$ 以上とすると、ほぼ 1.0% 以下の精度が保証されていることが分かる。

従って、以上の検証結果から、 $s=2.0$ を推奨値とした。また、当然ながら、上限周波数 f はユーザーの都合で与えるパラメータである。 f はユーザーが利用目的により意識して決めなくてはならないが、 s は数値計算のパラメータでありユーザーには簡単には決められないという事情がある。

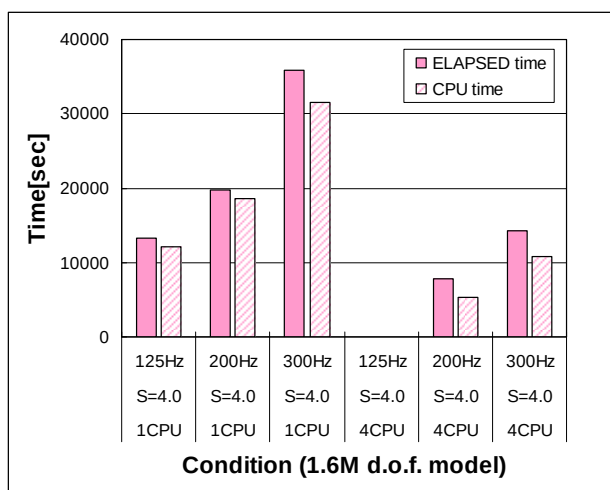


図 5 処理時間の上限周波数依存性 (その 1)

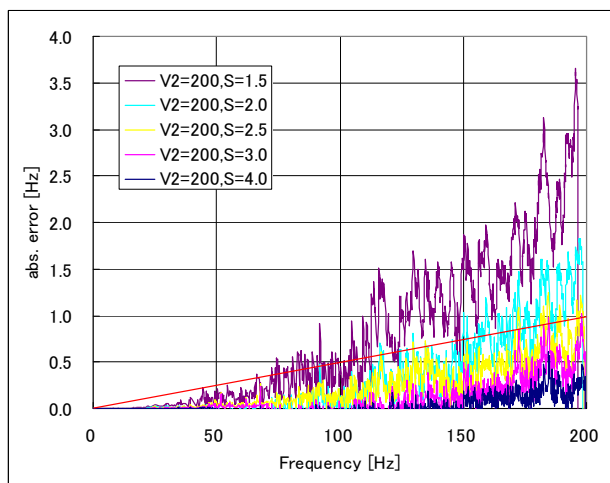


図 6 精度の上限周波数依存性 (その 1)

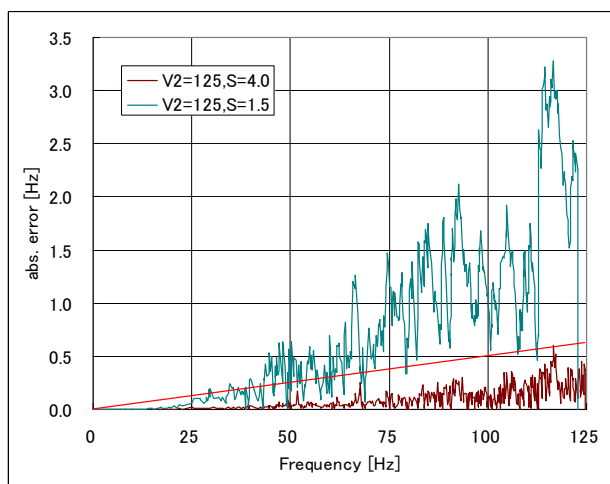


図 7 精度上限周波数依存性 (その 2)

4.5. 部分領域最小固有値数依存性

内部領域での最小固有値数および境界領域での最

小固有値数については、ひとつの節点の自由度以上が必要である。例えば、シェルを利用したデータの場合には、それぞれ最低 6 個が妥当であると考えられる。また、本検証問題においては、最初に設定した最小固有値数は 10 である。従って、検証問題では、念のため 6 と 10 の場合を確認した。

いずれも、処理時間および精度とも大きな変化はない (若干、双方を 6 に設定した場合に精度が悪くなっている)。従って、内部領域での最小固有値数および境界領域での最小固有値数については、安全側の 10 に設定することが妥当と考えられる。

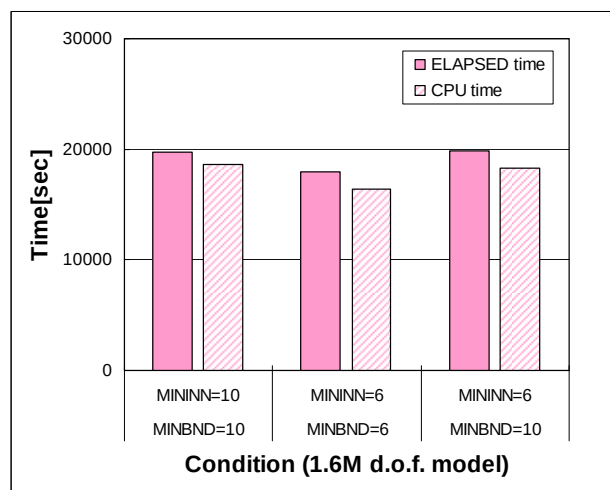


図 8 処理時間の部分領域固有値数への依存性

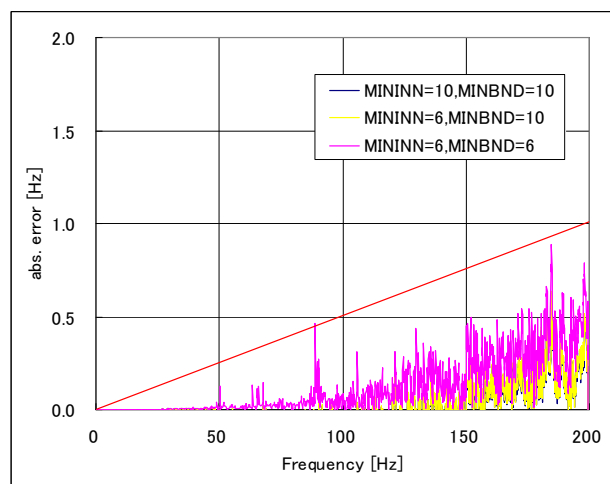


図 9 精度の部分領域固有値数への依存性

5. まとめ

アドバンスソフト株式会社では、新しいアルゴリズムに基づく開発およびその検証を実施してきた。現

時点では、課題がふたつ残されている。

そのうちひとつは、詳細な並列化のチューニングによる並列処理における処理速度の改善である。または必要に応じて別の並列化手法も導入する必要があると考えている。階層型の数値計算アルゴリズムについては、いくつもの並列化手法が提案されており、それらのいくつかが、多階層モード座標結合モード合成法に適用できると考えている。

ふたつめの課題は、解析対象のさらなる大規模化である。アドバンスソフト株式会社では、引き続きテスト計算として、数千万規模の大規模な固有値問題に着手したところである。

これらの2点を中心として、広い範囲のユーザーのニーズに応えるとともに、今後とも飛躍的に増大する計算機資源およびその性能に合わせて、さらに処理効率の高い計算アルゴリズムを開発していく予定である。本ソフトウェアは、アドバンスソフト株式会社が独自に改良し販売する構造解析有限要素法プログラム Advance/FrontSTR の大規模固有値解析のオプションとして販売および保守を行っている。利用者からの声を反映させながら、より頑強でより高速な固有値ソルバーとしていくことを計画している。

参考文献

- [1] 長松 昭男, 大熊 政明, “部分構造合成法”, 培風館 (1991)
- [2] 松原聖, 中村寿, 月森和之, 矢川元基; “並列処理による構造解析コードの試作研究 (その 3: 固有値ソルバーの並列化手法とその評価)”, 日本機械学会第 74 期通常総会講演会(1997), pp.73-75.
- [3] K.Garatani, K.Kitagawa, K.Matsubara, H.Nakamura, K.Tsukimori, G.Yagawa;"Study on Parallelization Method of Structural-Analysis Code,"High Performance Computing and Networking '97 in Europe.(1997), pp.1044-1046
- [4] 松原聖, 桑原匠史 他; “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)

※技術情報誌アドバンスシミュレーションは、アドバンスソフト株式会社 ホームページのシミュレーション図書館から、PDF ファイルがダウンロードできます。(ダウンロードしていただくには、アドバンス/シミュレーションフォーラム会員登録が必要です。)