大規模固有値計算プログラムAdvance/NextNVH

松原　聖*　　桑原　匠史**

The solver for large scale eigenvalue problems: Advance/NextNVH

Kiyoshi MATSUBARA* and Takuhito KUWABARA**

構造設計に対する振動騒音への要求は、「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、一層重要な課題になってきており、それらの系全体の固有値計算を可能とするソフトウェアの出現が期待されている。そのようなソフトウェアは、例えば「自動車全体および機械装置全体を丸ごと」等の振動騒音設計における処理スピード、および、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を持っている。本稿では、アドバンスソフトで開発した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に対する新しい提案を述べ、その性能について言及する。これをソフトウェアとして実現した Advance/NextNVH は、大規模固有値計算ソルバーであり、汎用的な計算機環境において高速な固有値解析を可能にするソフトウェアである。

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。筆者らは、並列計算機の市販が開始され、並列計算におけるメッセージパッシングライブラリが普及してきた1990年代前半から、並列計算を利用したシミュレーションの高速化および大規模化に着目して、ソフトウェアの開発および事業を実施してきた[2][3]。これら文献は、サブスペース法およびLanczos法による固有値計算の並列化に関する文献である。ここでは、当時の並列計算機で十分なスケーラビリティを持った並列化性能を得ることができたということを報告した。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

*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

ってきている。

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ており、また、それを実現できる計算機環境が整いつつある。

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の階層、枝の方を下の階層と呼ぶ。まず、実空間を下の階層から上の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、上の階層から下の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。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. 開発した手法の概要

2.1. 目的

ここでは、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を求める問題を対象とする。また、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。

本稿で提案するアルゴリズムは、このような条件を満たす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。

2.2. 既存の固有値解法

まず、数十万次元（有限要素法での数十万自由度）程度の大きさのスパースな行列に関して、部分的な固有値を求める（全固有値を求める問題に対して、部分的に求めるという言葉を使った）ための既存の手法としては、Block Lanczos法やサブスペース法等が利用されている。歴史的には、これ以外の多くの手法が利用されてきたが、上記の2つ以外の手法の多くは大規模向けではない。大規模向けではない手法や既存の手法で大規模向けの手法に関する手法の詳細については、例えば[1], [5], [6], [7], [8], [9]等の参考文献を参照していただきたい。

次に、全体構造を分割することで、より大規模な固有値問題を解くための近似解法として、いろいろな手法が開発されてきた。初期の段階では、領域分割を行い、その内部領域ごとに一般に知られているGuyanの縮約法または静的縮約(Guyan's reductionまたはstatic condensation、以下Guyanの静的縮約の用語で統一する)を適用した部分構造モード合成法が利用されてきた。そこでは、静縮小した行列の固有値をGuyanの静的縮約により内部領域の自由度を消去した境界領域の固有値問題に変換して、規模が小さくなった問題を解く。その規模が小さくなった問題の解を最終的な解として利用することは、実用的に現実的な解法であった。

また、モード合成法、そのうちで、物理座標結合モード合成法は、内部領域のモードと静縮小した境界領域の自由度を結合させた固有値問題として解く方法である。モード合成を利用したその他の手法として、モード座標結合モード合成法では、内部領域モードと静縮小した境界領域モードを結合させた固有値問題である。モード合成を利用した手法では、基本的に関数空間の基底の考えから構成されるアルゴリズムである。これらはすべて近似手法であること、および、基底とする関数のとり方にはいろいろな手法が考えられることから、ソフトウェアとしては多様な方法が実装され、試されてきた。その中で、多階層モード座標結合モード合成法では、先に述べたモード合成法等の近似手法に対して、その領域に階層を持たせた領域分割に適用する方法である。

さらに、以上で述べた方法を複数取り入れた手法

を考えることもできる。そのうちのひとつであるハイブリッド型の多階層モード座標結合モード合成法では、モード座標結合モード合成法と物理座標結合モード合成法の混合手法である。原理的には、モード合成の範囲内であるが、このようなアルゴリズムをソフトウェアとして実現する場合には、かなり複雑な制御の必要なソフトウェアとなることを覚悟する必要がある。また、階層型の領域を制御するために、最も単純な二分木を利用する方法、四分木を利用する方法および任意の木を利用できるようにする方法が考えられている。これらは、ソフトウェアの複雑さよりも、処理効率に関係する。

本固有値ソルバーでは、大規模な問題に適合可能なように、多階層のモード座標結合モード合成法を利用する。また、本固有値ソルバーで取り扱う領域としては、多階層型の二分木を利用するアルゴリズムを利用する。これは、多階層を利用することで、大規模問題にも対応できるようにすることと、物理的な意味合いの明確なモード座標結合モード合成法を利用するものとする。ただし、ソフトウェアとしての実装には、文献には公開されていない各種の工夫が必要である。

表 1　大規模固有値問題向け数値解法の比較(1)

解法	精度	処理時間		使用メモリ	
		小規模	大規模	小規模	大規模
①	◎	大	×	大	×
②	△	中	△	中	×
③	○	中	△	中	×
④	○	中	△	中	×
⑤	○	中	○	中	○
⑥	○	中	○	中	○

① 従来の方法（Lanczos 法）
② 静縮小した行列の固有値
③ 物理座標結合モード合成法
④ モード座標結合モード合成法
⑤ 多階層モード座標結合モード合成法
⑥ ハイブリッド型多階層モード座標結合モード合成法

表 2　大規模固有値問題向け数値解法の比較(2)

解法	プログラム開発	並列プログラム
①	ライブラリあり	直接法が並列に不向き
②	従来から利用	直接法が並列に不向き
③	複雑	共有メモリで実現された
④	複雑	共有メモリで実現された
⑤	かなり複雑	分散メモリ用は複雑
⑥	非常に複雑	さらに複雑

① ⑥については、表 1 と同じ。

表 3　階層構造処理方法の比較(1)

手法	精度	処理時間		使用メモリ	
		小規模	大規模	小規模	大規模
二分木	○	中	中	中	中
四分木	○	大	大	大	大
任意木	○	個別に方法を決める必要あり			

表 4　階層構造処理方法の比較(1)

手法	プログラム開発	並列プログラム
二分木	全体の制御が比較的簡単である	比較的簡単、MPMD でも制御可能
四分木	同上	同上
任意木	表 3 と同じく個別の手法に依存	master-slave 方式が妥当

2.3. 既存の多階層モード座標結合モード合成法

まず、既存のアルゴリズムについて述べる。これまで一般的に利用されてきた文献[1]等で公開されている既存の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；多階層に領域分割された最下位の階層（内部領域）における内部領域群の固有値問題の固有値・固有ベクトルを求める。

(2)境界領域の固有値問題；その親（ひとつ上の階層の領域）に静縮合された固有値問題の固有値・固有

ベクトルを求める。

(3)モード空間での固有値問題；上記(1)と(2)の固有値問題の固有ベクトルを基底とした空間（モード空間）での、固有値・固有ベクトルを求める。

(4)固有ベクトルの計算;モード空間で求まった固有ベクトルを実空間における固有ベクトルに変換する。

(5)上位の階層へ;以上のステップで解いた固有値問題の解を改めて、下位の領域の解と考え、さらにひとつ上位の階層について固有値問題を解く。すなわち、その上位の領域について静的縮約された固有値問題を解くステップ(2)に戻る。

図 1 にそのアルゴリズムを示す。

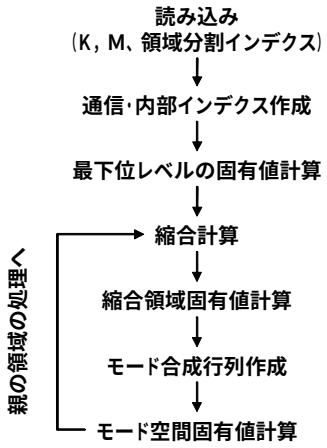

図 1　既存の多階層モード座標結合モード合成法アルゴリズム[1]

2.4. 開発した多階層モード座標結合モード合成法

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。

(1) 内部領域の固有値問題；多階層に領域分割された最下位の階層（内部領域）における領域群の固有値問題の固有値・固有ベクトルを求める。

(2)境界領域の固有値問題；その親（ひとつ上の階層の領域）に静縮合された固有値問題の固有値・固有ベクトルを求める。さらにひとつ上の階層の固有値問題を解く。すなわち、その上位の領域に静的縮約された固有値問題を最上位に到達するまで繰り返して解く。

(3)モード空間の固有値問題；上記(1)とすべての(2)における固有値問題で求めた固有ベクトルを基底とた空間（モード空間）で、固有値・固有ベクトルを求める。まず最上位のモード空間で固有値を求め、そこで固有ベクトルを求める。その次に、ひとつ下位の階層の処理を行う。この処理を、最下位の領域に到達するまで繰り返す。

(4)固有ベクトルの計算;最下位の領域の処理で求まった固有値が系全体の固有値である。最後に、最下位の領域の処理におけるモード空間で求まった固有ベクトルを実空間における固有ベクトルに変換する。

ここに示した新しいアルゴリズムを図2に示す。

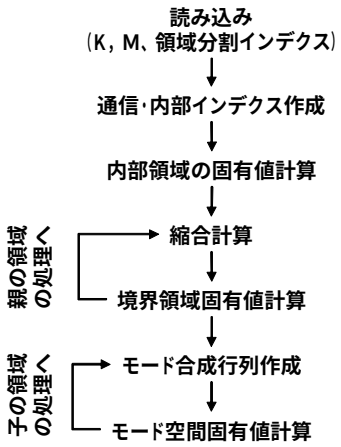

図 2　新規開発した多階層モード座標結合モード合成法アルゴリズム

3. 新規開発したアルゴリズムの特長と例示

3.1. 既存のアルゴリズムの例示

3.1.1. 二領域二階層の定式化

(1) 概要

領域全体を分割し、その領域ごとに固有値を求め、それらの結果をモード合成して全体の固有値にすることが領域分割の考え方である。モード合成法は、

各節点の変位を境界節点を拘束したときの動的変位（振動モード）と境界節点の変位に従属して決まる静的変位とに分離することができ、それらを独立に取り扱うことができるということを基本としている。従って、各レベルの変位は、それより上位の変位を利用して、表現することができる。

本節で議論する問題は、図3で示すような二領域二階層を対象としている。

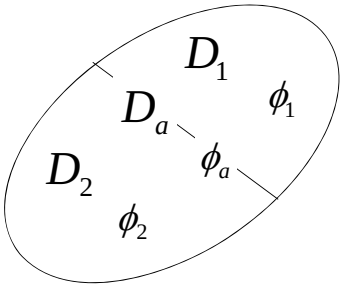

図 3　二領域二階層の場合のモデル

ここでは、まず、

$$\begin{pmatrix} M_{11} & 0 & M_{1a} \\ 0 & M_{22} & M_{2a} \\ M_{a1} & M_{a2} & M_{aa} \end{pmatrix}\begin{pmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_a \end{pmatrix}+\begin{pmatrix} K_{11} & 0 & K_{1a} \\ 0 & K_{22} & K_{2a} \\ K_{a1} & K_{a2} & K_{aa} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_a \end{pmatrix} =\begin{pmatrix} f_1 \\ f_2 \\ f_a \end{pmatrix} \tag{1}$$

を解くことを考える。対応する固有値問題は、

$$\begin{pmatrix} K_{11} & 0 & K_{1a} \\ 0 & K_{22} & K_{2a} \\ K_{a1} & K_{a2} & K_{aa} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_a \end{pmatrix} =\lambda\begin{pmatrix} M_{11} & 0 & M_{1a} \\ 0 & M_{22} & M_{2a} \\ M_{a1} & M_{a2} & M_{aa} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_a \end{pmatrix} \tag{2}$$

である。ここで下付きの添え字は、それぞれの領域を表す。本問題に対して、Guyanの静的縮約を具体的に書き下すと、

$$\begin{aligned} K_{11}u_1 + K_{1a}u_a &= 0 \\ K_{22}u_2 + K_{2a}u_a &= 0 \end{aligned} \tag{3}$$

の通りである。

これらの記号を使って、全体の固有値を求める手順は次の通りである。まず、内部領域D_1とD_2について独立に固有値問題を解き、ϕ_1とϕ_2を求める。次に境界領域については、Guyan の静的縮約条件を利用した固有値問題をD_aについて解き、ϕ_aを求める。ここまでは物理空間での固有値問題である。処理の後半では、モード空間の固有値問題を解く。モード空間の基底には、ϕ_1とϕ_2とϕ_aを利用する。ここでは、実空間とモード空間の間の変換の行列Tを定義し、その固有値問題を解く。以下では、その処理の流れと定式化について述べる。

(2) 内部領域の固有値問題

まず、内部領域の固有値問題として、

$$\begin{aligned} K_{11}u_1 &= \lambda M_{11}u_1 \\ K_{22}u_2 &= \lambda M_{22}u_2 \end{aligned} \tag{4}$$

を解く。ここで解くことのできた問題の固有ベクトル群を、ϕ_1、ϕ_2とする。ϕ_1は、行数はu_1と同じで、列数は求めた固有値の数と等しい。従って、ϕ_1、ϕ_2は、行数も列数もサイズは異なる行列である。また、それぞれの問題で計算された固有ベクトル群は、すべてM 直交となるように正規化する。すなわち、

$$\varphi^t{}_i M \varphi_j = \delta_{ij} \tag{5}$$

であり、行列の形式であれば、

$$\phi^t{}_i M \phi^t{}_j = I \tag{6}$$

となるようにする。ここで、φ_iは、ϕ_1またはϕ_2中のひとつの固有ベクトルである。また、δ_{ij}はクロネッカのデルタであり、I は単位行列である。本稿中で、以下、すべての固有ベクトルは M 直交を仮定するものとする。

(3) 境界領域の固有値問題

次に、Guyan の静的縮約条件を

$$\begin{aligned} u_1 &= -K_{11}{}^{-1}K_{1a}u_a \\ u_2 &= -K_{22}{}^{-1}K_{2a}u_a \end{aligned} \tag{7}$$

の形で利用して、運動方程式に代入し、u_1とu_2を消去して、u_aの方程式を求める。すなわち、

$$\begin{aligned} &\left(M_{aa} - M_{a1}K_{11}{}^{-1}K_{1a} - M_{a2}K_{22}{}^{-1}K_{2a}\right)\ddot{u}_a \\ &+\left(K_{aa} - K_{a1}K_{11}{}^{-1}K_{1a} - K_{a2}K_{22}{}^{-1}K_{2a}\right)u_a = f_a \end{aligned} \tag{8}$$

となる。この固有値問題は、

$$\begin{aligned} &M'_{aa}\ddot{u}_a = \lambda K'_{aa}u_a \\ &K'_{aa} = K_{aa} - K_{a1}K_{11}{}^{-1}K_{1a} - K_{a2}K_{22}{}^{-1}K_{2a} \\ &M'_{aa} = M_{aa} - M_{a1}K_{11}{}^{-1}K_{1a} - M_{a2}K_{22}{}^{-1}K_{2a} \end{aligned} \tag{9}$$

として解くことができる。ここで求めた固有ベクトル群を、ϕ_aとする。

(4) モード空間の固有値問題

ここで解けた固有値問題は、Guyan の静的縮約条件のもとで正しいが、求める問題の解ではない。そこで、得られた固有ベクトルϕ_1、ϕ_2、ϕ_aを基底とした空間（モード空間）を考え、その空間での固有値問題を再度解くことにする。$u_1 = -K_{11}^{-1}K_{1a}u_a$であることを考えると、$u_1$は$\phi_1$と$-K_{11}^{-1}K_{1a}\varphi_a$（以下、$G_{1a}\phi_a$と書く）で張られる空間に存在すると仮定しても大きな間違いではない。また、u_2はϕ_2と$-K_{22}^{-1}K_{2a}\varphi_a$（以下、$G_{2a}\phi_a$と書く）で張られる空間に存在すると仮定する。この仮定は近似ではあるが、実用上は問題ないレベルである。また、これは、Guyan の静的縮約条件のもとで得られた解を補正する役割を果たしている。

これらの記号を利用して、モード空間での固有値を求める。まず、モード座標系をξ_Aと書くと、もともとの運動方程式の解は、

$$\begin{pmatrix} u_1 \\ u_2 \\ u_a \end{pmatrix} = \begin{pmatrix} \phi_1 & 0 & G_{1a}\phi_a \\ 0 & \phi_2 & G_{2a}\phi_a \\ 0 & 0 & \phi_a \end{pmatrix} \xi_A \tag{10}$$

となる。これは、モード座標系を物理座標系に変換する式である。この行列をT_Aと書く。すなわち、

$$T_A = \begin{pmatrix} \phi_1 & 0 & G_{1a}\phi_a \\ 0 & \phi_2 & G_{2a}\phi_a \\ 0 & 0 & \phi_a \end{pmatrix} \tag{11}$$

である。モード座標系を物理座標系に変換する式をもとの固有値問題に代入し、そのあとで、左からT_A^tを乗ずると

$$\begin{aligned} & T_A^t \begin{pmatrix} K_{11} & 0 & K_{1a} \\ 0 & K_{22} & K_{2a} \\ K_{a1} & K_{a2} & K_{aa} \end{pmatrix} T_A \xi_A \\ & = \lambda T_A^t \begin{pmatrix} M_{11} & 0 & M_{1a} \\ 0 & M_{22} & M_{2a} \\ M_{a1} & M_{a2} & M_{aa} \end{pmatrix} T_A \xi_A \end{aligned} \tag{12}$$

となる。この式を利用して、求めるべき問題の運動方程式は

$$\begin{aligned} & \begin{pmatrix} \varphi_1 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_a \end{pmatrix}^T \begin{pmatrix} K_{11} & 0 & K_{1a} \\ 0 & K_{22} & K_{2a} \\ K_{1a}^T & K_{2a}^T & K_{aa} \end{pmatrix} \begin{pmatrix} \varphi_1 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_a \end{pmatrix} \ddot{\xi}_A \\ & + \begin{pmatrix} \varphi_1 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_a \end{pmatrix}^T \begin{pmatrix} K_{11} & 0 & K_{1a} \\ 0 & K_{22} & K_{2a} \\ K_{1a}^T & K_{2a}^T & K_{aa} \end{pmatrix} \begin{pmatrix} \varphi_1 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_a \end{pmatrix} \xi_A \\ & = f' \end{aligned} \tag{13}$$

となる。ここで、次の課題は、このモード空間での方程式を解くことである。

そのために、この固有値問題の左辺と右辺を展開してみる。まず、左辺は、

$$\begin{pmatrix} \varphi_1^t & 0 & 0 \\ 0 & \varphi_2^t & 0 \\ (G_{1a}\varphi_a)^t & (G_{2a}\varphi_a)^t & \varphi_a^t \end{pmatrix} \begin{pmatrix} K_{11} & 0 & K_{1a} \\ 0 & K_{22} & K_{2a} \\ K_{a1} & K_{a2} & K_{aa} \end{pmatrix} \begin{pmatrix} \varphi_1 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_a \end{pmatrix}$$

$$= \begin{pmatrix} \varphi_1^t & 0 & 0 \\ 0 & \varphi_2^t & 0 \\ (G_{1a}\varphi_a)^t & (G_{2a}\varphi_a)^t & \varphi_a^t \end{pmatrix} \begin{pmatrix} K_{11}\varphi_1 & 0 & (K_{11}G_{1a} + K_{1a})\varphi_a \\ 0 & K_{22}\varphi_2 & (K_{22}G_{2a} + K_{2a})\varphi_a \\ K_{a1}\varphi_1 & K_{a2}\varphi_2 & (K_{aa} + K_{11}G_{1a} + K_{22}G_{2a})\varphi_a \end{pmatrix}$$

$$= \begin{pmatrix} \varphi_1^t & 0 & 0 \\ 0 & \varphi_2^t & 0 \\ (G_{1a}\varphi_a)^t & (G_{2a}\varphi_a)^t & \varphi_a^t \end{pmatrix} \begin{pmatrix} K_{11}\varphi_1 & 0 & 0 \\ 0 & K_{22}\varphi_2 & 0 \\ K_{a1}\varphi_1 & K_{a2}\varphi_2 & (K_{aa} + K_{11}G_{1a} + K_{22}G_{2a})\varphi_a \end{pmatrix}$$

$$= \begin{pmatrix} \phi_1^t K_{11}\phi_1 & 0 & 0 \\ 0 & \phi_2^t K_{22}\phi_2 & 0 \\ (G_{1a}\phi_a)^t K_{11}\phi_1 + \phi_a^t K_{a1}\phi_1 & (G_{2a}\phi_a)^t K_{22}\phi_2 + \phi_a^t K_{a2}\phi_2 & \phi_a^t K'_{aa}\phi_a^t \end{pmatrix}$$

$$= \begin{pmatrix} \phi_1^t K_{11}\phi_1 & 0 & 0 \\ 0 & \phi_2^t K_{22}\phi_2 & 0 \\ 0 & 0 & \phi_a^t K'_{aa}\phi_a^t \end{pmatrix}$$

$$= \begin{pmatrix} \lambda_1\phi_1^t M_{11}\phi_1 & 0 & 0 \\ 0 & \lambda_2\phi_2^t M_{22}\phi_2 & 0 \\ 0 & 0 & \lambda_a\phi_a^t M'_{aa}\phi_a^t \end{pmatrix}$$

$$= \begin{pmatrix} \lambda_1 I & 0 & 0 \\ 0 & \lambda_2 I & 0 \\ 0 & 0 & \lambda_a I \end{pmatrix} \tag{14}$$

となる。ここで、

$$\begin{aligned} & (G_{1a}\phi_a)^t K_{11}\phi_1 = -\left(K_{11}^{-1}K_{1a}\phi_a\right)^t K_{11}\phi_1 \\ & = -\phi_a^t K_{1a}^t K_{11}^{-1} K_{11}\phi_1 = -\phi_a^t K_{1a}^t \phi_1 \end{aligned} \tag{15}$$

および

$$\begin{aligned} & (G_{2a}\phi_a)^t K_{22}\phi_2 = -\left(K_{22}^{-1}K_{2a}\phi_a\right)^t K_{22}\phi_2 \\ & = -\phi_a^t K_{2a}^t K_{22}^{-1} K_{22}\phi_2 = -\phi_a^t K_{2a}^t \phi_2 \end{aligned} \tag{16}$$

を利用した。この式変形では、固有ベクトルはM直交であること等を十分に利用している。この式変形が可能であることから、モード座標結合モード座標系の定式化全体の式変形の見通しがかなり良くなっている。次に右辺を変形する。

$$
\begin{aligned}
&\begin{pmatrix} \varphi_1^t & 0 & 0 \\ 0 & \varphi_2^t & 0 \\ \left(G_{1a}\varphi_a\right)^t & \left(G_{2a}\varphi_a\right)^t & \varphi_a^t \end{pmatrix}
\begin{pmatrix} M_{11} & 0 & M_{1a} \\ 0 & M_{22} & M_{2a} \\ M_{a1} & M_{a2} & M_{aa} \end{pmatrix}
\begin{pmatrix} \varphi_1 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_a \end{pmatrix} \\
&= \begin{pmatrix} \varphi_1^t & 0 & 0 \\ 0 & \varphi_2^t & 0 \\ \left(G_{1a}\varphi_a\right)^t & \left(G_{2a}\varphi_a\right)^t & \varphi_a^t \end{pmatrix}
\begin{pmatrix} M_{11}\varphi_1 & 0 & \left(M_{11}G_{1a}+M_{1a}\right)\varphi_a \\ 0 & M_{22}\varphi_2 & \left(M_{22}G_{2a}+M_{2a}\right)\varphi_a \\ M_{a1}\varphi_1 & M_{a2}\varphi_2 & \left(M_{aa}+M_{11}G_{1a}+M_{22}G_{2a}\right)\varphi_a \end{pmatrix} \\
&= \begin{pmatrix} \varphi_1^t M_{11}\varphi_1 & 0 & \varphi_1^t\left(M_{11}G_{1a}+M_{1a}\right)\varphi_a \\ 0 & \varphi_2^t M_{22}\varphi_2 & \varphi_1^t\left(M_{11}G_{1a}+M_{1a}\right)\varphi_a \\ \varphi_a^t\left(M_{11}G_{1a}+M_{1a}\right)^t\varphi_a & \varphi_a^t\left(M_{11}G_{1a}+M_{1a}\right)^t\varphi_a & \varphi_a^t M_{aa}'\varphi_a^t \end{pmatrix} \\
&= \begin{pmatrix} I & 0 & \varphi_1^t\left(M_{11}G_{1a}+M_{1a}\right)\varphi_a \\ & I & \varphi_1^t\left(M_{11}G_{1a}+M_{1a}\right)\varphi_a \\ sym. & & I \end{pmatrix}
\end{aligned}
\tag{17}
$$

となる。ここで、sym は対称行列であることを表している（本稿中、以下、同様である）。ここでも、固有ベクトルはM 直交としたことで式変形の見通しがよくなっている。モード空間の固有値問題は、

$$
\begin{pmatrix} \lambda_1 I & 0 & 0 \\ 0 & \lambda_2 I & 0 \\ 0 & 0 & \lambda_a I \end{pmatrix}\xi_A = \lambda \begin{pmatrix} I & 0 & \varphi_1^t\left(M_{11}G_{1a}+M_{1a}\right)\varphi_a \\ & I & \varphi_2^t\left(M_{22}G_{2a}+M_{2a}\right)\varphi_a \\ sym. & & I \end{pmatrix}\xi_A \tag{18}
$$

の形の固有値問題に帰着できることがわかる。ここで求めることができる固有値をλ_A および固有ベクトルをθ_A とする。ここで計算できるモード空間の固有値問題の固有値は、もとの問題の固有値と一致する。もちろん、固有ベクトルはもとの問題の固有ベクトルとは一致しない。固有ベクトルについては、次の節で求め方を示す。

(5) 固有ベクトルの計算

最後に、これまでに得られた固有値と固有ベクトルから、もとの問題の固有ベクトルを再構成する方法について述べる。モード空間で得られた固有ベクトルをθ_A とすると、このθ_A からもとの固有値問題の固有ベクトルを算出する手順を確認する。モード空間から物理空間への変換はT_A を乗ずればいいので、

$$\phi_A = T_A \xi_A \tag{19}$$

で、もとの物理空間の固有ベクトルを求めることができる。

(6) アルゴリズムの注意点

これらの定式化で注意すべき点は、物理空間では非常に多い次元（有限要素法での自由度数）であるが、モード空間に変換されるとその次元は求めるべき固有値の数程度となる。従って、その数が激減することに注意を払って定式化する必要がある。

3.1.2. 多領域二階層の定式化

(1) 概要

多領域二階層の定式化においては、本節では、下図のような状況を考える。ここでは、境界領域（図中では線分の集合）のすべてをひとつの領域とする。

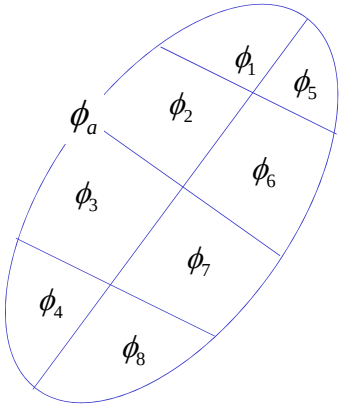

図 4　多領域二階層の場合のモデル

多領域二階層のモデルにおいて、全体の固有値を求める手順は次の通りである。まず、内部領域D_1 からD_8 について独立に固有値問題を解き、ϕ_1 からϕ_8 を求める。次に境界領域については、D_1 からD_8 までの Guyan の静的縮約条件を利用した固有値問題をD_a について解き、ϕ_a を求める。ここまでは物理空間での固有値問題である。処理の後半では、モード空間の固有値問題を解く。モード空間の基底には、ϕ_1 からϕ_8 とϕ_a を利用する。ここでは、実空間とモード空間の間の変換の行列T を定義し、その固有値問題を解く。以下では、二領域二階層と異なる部分を中心に、多領域二階層での処理の流れについて、述べる。

(2) 内部領域の固有値問題

まず、内部領域の固有値問題として、

$$K_{ii}u_i = \lambda M_{ii}u_i \tag{20}$$

を解く。ここで解くことのできた問題の固有ベクトル群をϕ_i とする。本節で取り扱っている問題では、$1 \le i \le 8$ である。

(3) 境界領域の固有値問題

次に、Guyan の静的縮約条件を利用して、縮合された行列に対する固有値問題

$$M'_{aa}\ddot{u}_a = \lambda K'_{aa}u_a \qquad (21)$$

を解く。ここで、M'_{aa} と K'_{aa} は縮合された合成行列と質量行列である。また、固有ベクトル群を ϕ_a とする。

(4) モード空間の固有値問題

二領域二階層と同じ考え方で定式化すると、次のような実空間とモード空間の変換行列が得られることがわかる。

$$\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_a \end{pmatrix} = \begin{pmatrix} \varphi_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{1a}\varphi_a \\ 0 & \varphi_2 & 0 & 0 & 0 & 0 & 0 & 0 & G_{2a}\varphi_a \\ 0 & 0 & \varphi_3 & 0 & 0 & 0 & 0 & 0 & G_{3a}\varphi_a \\ 0 & 0 & 0 & \varphi_4 & 0 & 0 & 0 & 0 & G_{4a}\varphi_a \\ 0 & 0 & 0 & 0 & \varphi_5 & 0 & 0 & 0 & G_{5a}\varphi_a \\ 0 & 0 & 0 & 0 & 0 & \varphi_6 & 0 & 0 & G_{6a}\varphi_a \\ 0 & 0 & 0 & 0 & 0 & 0 & \varphi_7 & 0 & G_{7a}\varphi_a \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_8 & G_{8a}\varphi_a \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_a \end{pmatrix} \xi_A \qquad (22)$$

ここで、前節と同じ記法ではあるが、

$$G_{ia} = -K_{ii}^{-1}K_{ia} \qquad (23)$$

である。ここで求めることができる固有値を λ_A および固有ベクトルを θ_A とする。

(5) 固有ベクトルの計算

次に、固有ベクトルを θ_A とする。ここでも、モード座標系から物理座標系への変換は T_A を乗ずればいいので、

$$\phi_A = T_A\xi_A \qquad (24)$$

で物理空間の固有ベクトルを求めることができる。これは、二領域二階層の場合とまったく同様である。

3.1.3. 多領域三階層の定式化

(1) 概要

多領域三階層における定式化では、ここまでに説明した二階層の定式化より、少し複雑になる。その理由は、最下位と最上位の領域が中間の領域を通して関連しているためである。本節では、図のような状況を考える。ここでは、境界領域（図中では線分の集合）のすべてをひとつの領域とする。最終的に目指す定式化は、次の図のような多領域多階層の定式化である。

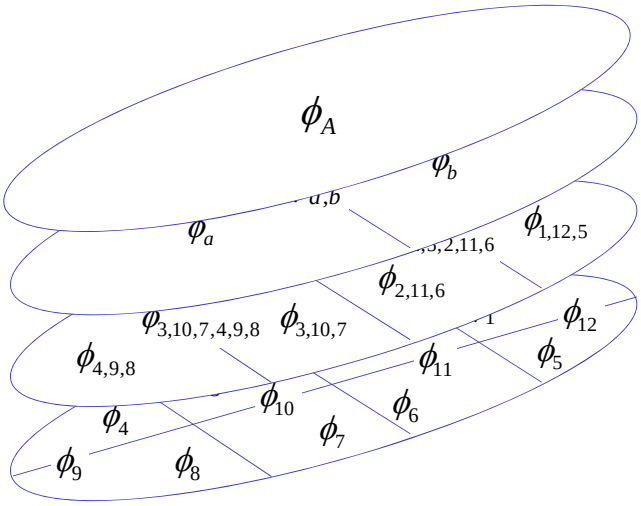

図 5　多領域多階層の場合のモデル

多領域三階層において、全体の固有値を求める手順は次の通りである。領域は、図 5 のように階層化され、それらの領域は二分木で構成されているものとする。まず、内部領域 D_1 から D_8 について独立に固有値問題を解き、ϕ_1 から ϕ_8 を求める。次に境界領域については、二分木の上位に向かって処理する。この場合には、まず 4 つの境界領域について処理する。最初は D_1 と D_4 の Guyan の静的縮約条件を利用した固有値問題を D_9 について解き、ϕ_9 を求める。同様にして、ϕ_{10} から ϕ_{12} までを求める。次の上位のレベルについて、次々に縮合された固有値問題を解いていく。最後に、最上位の領域に縮合された固有値問題を解いて、ϕ_A を求める。ここまでは物理空間での固有値問題である。

処理の後半では、下位のレベルからモード空間での固有値問題を解く。最初は D_1 と D_4 について、モード空間の基底には ϕ_1 と ϕ_4 と ϕ_9 を利用してその固有値問題を解く。同様にして、ϕ_{10} から ϕ_{12} までのモード空間の処理を行う。次の上位のレベルについて、次々にモード空間でも固有値問題を解いていく。最後に、最上位の領域についてモード空間での固有値問題を解く。モード空間の基底には、ϕ_a と ϕ_b と ϕ_A を利用する。ここでは、実空間とモード空間の間の変換の行列 T を定義し、その固有値問題を解く。以下では、その処理の流れを述べる。

(2) 内部領域の固有値問題

まず、内部領域の固有値問題として、

$$K_{ii}u_i = \lambda M_{ii}u_i \tag{25}$$

を解く。ここで解くことのできた問題の固有ベクトル群をϕ_iとする。

(3) 境界領域の固有値問題

次に、Guyan の静的縮約条件を利用して、縮合された行列に対する固有値問題

$$M'_{aa}\ddot{u}_a = \lambda K'_{aa}u_a \tag{26}$$

を解く。ここで、M'_{aa}とK'_{aa}は縮合された合成行列と質量行列である。また、固有ベクトル群をϕ_aとする。

(4) モード空間の固有値問題

ここでは、3 階層のデータにおけるT行列およびモード空間での方程式

$$\overline{K}x = \lambda\overline{M}x \tag{27}$$

$$\overline{K} = T^t K T \tag{28}$$

$$\overline{M} = T^t M T \tag{29}$$

における$\overline{K}$と$\overline{M}$とを示す。まず、多階層の場合には、Tは次のような形となる。この式の導き方は、前節までと同様に、Guyan の静的縮約条件を利用して定式化する。

$$T = \begin{bmatrix} \varphi_1 & 0 & 0 & 0 & G_{15}\varphi_5 & 0 & \left(G_{17}+G_{15}G'_{57}\right)\varphi_7 \\ 0 & \varphi_2 & 0 & 0 & G_{25}\varphi_5 & 0 & \left(G_{27}+G_{25}G'_{57}\right)\varphi_7 \\ 0 & 0 & \varphi_3 & 0 & 0 & G_{36}\varphi_6 & \left(G_{37}+G_{36}G'_{67}\right)\varphi_7 \\ 0 & 0 & 0 & \varphi_4 & 0 & G_{36}\varphi_6 & \left(G_{47}+G_{46}G'_{67}\right)\varphi_7 \\ 0 & 0 & 0 & 0 & \varphi_5 & 0 & G_{57}\varphi_7 \\ 0 & 0 & 0 & 0 & 0 & \varphi_6 & G_{67}\varphi_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & \varphi_7 \end{bmatrix} \tag{30}$$

ここで、

$$\begin{aligned} G'_{57} &= -K'^{-1}_{55}K'_{57} \\ K'_{55} &= K_{55} - K^T_{15}K^{-1}_{11}K_{15} - K^T_{25}K^{-1}_{22}K_{25} \\ K'_{57} &= K_{57} - K^T_{15}K^{-1}_{11}K_{17} - K^T_{25}K^{-1}_{22}K_{27} \\ G'_{67} &= -K'^{-1}_{66}K'_{67} \\ K'_{66} &= K_{66} - K^T_{36}K^{-1}_{33}K_{36} - K^T_{46}K^{-1}_{44}K_{46} \\ K'_{67} &= K_{67} - K^T_{36}K^{-1}_{33}K_{37} - K^T_{46}K^{-1}_{44}K_{47} \end{aligned} \tag{31}$$

である。また、解くべき固有値問題の$\overline{K}x = \lambda\overline{M}x$における$\overline{K}$と$\overline{M}$は次の通りである。

$$\overline{K} = \begin{bmatrix} \lambda_1 I & 0 & 0 & 0 & 0 & 0 & 0 \\ & \lambda_2 I & 0 & 0 & 0 & 0 & 0 \\ & & \lambda_3 I & 0 & 0 & 0 & 0 \\ & & & \lambda_4 I & 0 & 0 & 0 \\ & & & & \lambda_5 I & 0 & 0 \\ & & & & & \lambda_6 I & 0 \\ sym & & & & & & \lambda_7 I \end{bmatrix} \tag{32}$$

$$\overline{M} = \begin{bmatrix} I & 0 & 0 & 0 & \varphi_1^t\overline{M}_{15}\varphi_5 & 0 & \varphi_1^t\overline{M}_{17}\varphi_7 \\ & I & 0 & 0 & \varphi_2^t\overline{M}_{25}\varphi_5 & 0 & \varphi_2^t\overline{M}_{27}\varphi_7 \\ & & I & 0 & 0 & \varphi_3^t\overline{M}_{36}\varphi_6 & \varphi_3^t\overline{M}_{37}\varphi_7 \\ & & & I & 0 & \varphi_4^t\overline{M}_{46}\varphi_6 & \varphi_4^t\overline{M}_{47}\varphi_7 \\ & & & & I & 0 & \varphi_5^t\overline{M}_{57}\varphi_7 \\ & & & & & I & \varphi_6^t\overline{M}_{67}\varphi_7 \\ sym & & & & & & I \end{bmatrix}$$

ただし、ここで、次のようにおいた。

$$\begin{aligned} \overline{M}_{15} &= M_{11}G_{15} + M_{15} \\ \overline{M}_{25} &= M_{22}G_{25} + M_{25} \\ \overline{M}_{36} &= M_{33}G_{36} + M_{36} \\ \overline{M}_{46} &= M_{44}G_{46} + M_{46} \end{aligned} \tag{33}$$

$$\begin{aligned} \overline{M}_{17} &= M_{11}\left(G_{17}+G_{15}G'_{57}\right) + M_{15}G'_{57} + M_{17} \\ \overline{M}_{27} &= M_{22}\left(G_{27}+G_{25}G'_{57}\right) + M_{25}G'_{57} + M_{27} \\ \overline{M}_{37} &= M_{33}\left(G_{37}+G_{36}G'_{67}\right) + M_{36}G'_{67} + M_{37} \\ \overline{M}_{47} &= M_{44}\left(G_{47}+G_{46}G'_{67}\right) + M_{46}G'_{67} + M_{47} \\ \overline{M}_{57} &= M'_{55}G'_{57} + M'_{57} \\ \overline{M}_{67} &= M'_{66}G'_{67} + M'_{67} \end{aligned} \tag{34}$$

ここで、これらの M や G 同士の演算には、多大なコストがかかることを認識しておく必要がある。従って、大規模計算において、この課題を解決しなくてはならない。

(5) 固有ベクトルの計算

最終的に求めることができたモード空間の固有ベクトルについては、前節と同じようにTでモード空間から実空間に変換することができる。次に、固有ベクトルをθ_Aとする。モード空間から物理空間への変換はT_Aを乗ずればいいので、

$$\phi_A = T_A\xi_A \tag{35}$$

で物理空間の固有ベクトルを求めることができる。この式自体は二階層の場合とまったく同様であるが、ソフトウェアでの実装では大きく異なる。

3.2. 開発したアルゴリズムの例示

既存の手法では、さらに階層が増えた場合に、Tの定式化が複雑になり、また、演算量も増加する。従って、本ソルバーの開発では、この部分のアルゴリズムおよび全体の制御方法を新規に開発した。

3.2.1. 既存のアルゴリズムの欠点

前節で(30)式に示した通り、従来のアルゴリズムでは、

$$T=\begin{bmatrix} \varphi_1 & 0 & 0 & 0 & G_{15}\varphi_5 & 0 & \left(G_{17}+G_{15}G'_{57}\right)\varphi_7 \\ 0 & \varphi_2 & 0 & 0 & G_{25}\varphi_5 & 0 & \left(G_{27}+G_{25}G'_{57}\right)\varphi_7 \\ 0 & 0 & \varphi_3 & 0 & 0 & G_{36}\varphi_6 & \left(G_{37}+G_{36}G'_{67}\right)\varphi_7 \\ 0 & 0 & 0 & \varphi_4 & 0 & G_{36}\varphi_6 & \left(G_{47}+G_{46}G'_{67}\right)\varphi_7 \\ 0 & 0 & 0 & 0 & \varphi_5 & 0 & G_{57}\varphi_7 \\ 0 & 0 & 0 & 0 & 0 & \varphi_6 & G_{67}\varphi_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & \varphi_7 \end{bmatrix} \quad (36)$$

における(1,7)成分等のように、T 行列を構成する一部の G について、G 同士を演算して、修正する必要がある。また、対角ブロックおよび再右列以外にも(1,5)成分等の位置の成分が出現する。これらの状況により、処理方式が複雑となるとともに、演算量が爆発的に増大する。

3.2.2. 開発したアルゴリズム

(1) 開発したアルゴリズムの特長

前節で述べた欠点を克服するために、新たなアルゴリズムを開発した。ここで開発したアルゴリズムでは、モード空間における固有値計算については、G を修正する必要がなく、また、対角ブロックおよび再右列以外には、非零のブロックはあらわれない。

ここでは、次の手順で処理することを考える。まず、最下位の領域の固有値問題を解く。その後、最下位のレベルから最上位のレベルまで順番に縮合計算と縮合領域の固有値計算を実施する。ここまでは、従来の方法と同じである。次に、モード空間においては、上のレベルから 1 階層ずつ下位に降りながら処理する。すなわち、モード空間における処理を既存の方法とは逆の順序で行う。

具体的には、まず、モード空間では、レベル 1（最上位）とレベル 2（上位から 2 番目）の 2 つの階層で、レベル 1+2 のモード合成処理（固有値を計算）を行う。次に、レベル 1+2 とレベル 3 の 2 つの階層で、レベル 1+2+3 のモード合成処理を行う。これを繰り返す。当然ながら、その処理量は増大していく。最後に最上位から最下位の直前までのレベルにおけるモード空間で処理した結果と最下位レベルの 2 つの階層で、全体のモード剛性処理を行う。ここで、求まった解が最終的な全体の系の固有値となる。

ここで述べた処理を三階層モデルでの定式化について述べる。まず、最下位のレベルで固有値計算を行う。次に、最下位のレベルから、順次、上のレベルに上がりながらレベル 1（最上位のレベル）まで縮合計算と縮合領域の固有値計算を繰り返し実施する。ここまでは、既存のアルゴリズムと同等の物理座標系での処理を行う。ただし、ここまでては、モード空間での処理は一切行わないことに注意する。

そののちに実施すべきモード空間での処理手順は次の通りである。まず、モード空間における最初の処理は、レベル 1 とレベル 2 を対象とする。この処理では、

$$[T]=\begin{bmatrix} \phi_{2,1} & 0 & G_{1B,1}\phi_{B,1} \\ 0 & \phi_{2,2} & G_{2B,1}\phi_{B,1} \\ 0 & 0 & \phi_{B,1} \end{bmatrix} \quad (37)$$

のT行列を利用する。ここで、$\varphi_{B,1}$は最上位で求めた縮合領域の固有ベクトルである。これを用いてモード空間の固有値を計算する。ここで求められた固有ベクトルを$\varphi_{B,1+2}$とすると、次のモード空間での手順では、

$$[T]=\begin{bmatrix} \phi_{3,1} & 0 & 0 & 0 & G_{1B,1+2}\phi_{B,1+2} \\ 0 & \phi_{3,2} & 0 & 0 & G_{2B,1+2}\phi_{B,1+2} \\ 0 & 0 & \phi_{3,3} & 0 & G_{3B,1+2}\phi_{B,1+2} \\ 0 & 0 & 0 & \phi_{3,4} & G_{4B,1+2}\phi_{B,1+2} \\ 0 & 0 & 0 & 0 & \phi_{B,1+2} \end{bmatrix} \quad (38)$$

を利用する。ここで注意すべきことは、対角ブロックおよび再右列以外には、非零のブロックはあらわれないことである。これを用いてモード空間の固有値を計算する。ここで求められた固有ベクトルを$\varphi_{B,1+2+3+4}$とすると、最終的なモード空間での手順では、

$$[T]=\begin{bmatrix} \varphi_{4,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{1B,1+2+3}\varphi_{B,1+2+3} \\ 0 & \varphi_{4,2} & 0 & 0 & 0 & 0 & 0 & 0 & G_{2B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & \varphi_{4,3} & 0 & 0 & 0 & 0 & 0 & G_{3B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & 0 & \varphi_{4,4} & 0 & 0 & 0 & 0 & G_{4B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & 0 & 0 & \varphi_{4,5} & 0 & 0 & 0 & G_{5B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & 0 & 0 & 0 & \varphi_{4,6} & 0 & 0 & G_{6B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{4,7} & 0 & G_{7B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{4,8} & G_{8B,1+2+3}\varphi_{B,1+2+3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_{B,1+2+3} \end{bmatrix} \tag{39}$$

を利用する。ここで求められた固有値および固有ベクトルが最終的な全体系の固有値および固有ベクトルとなる。

ここでの多領域三階層での定式化で確認したように、新規に開発したアルゴリズムでは、G を修正する必要がない。従来のアルゴリズムではモード空間での処理に処理時間がかかっており、その原因は G の修正(P 行列)により、T 行列を構成する必要があったことである。この新規アルゴリズムはモード空間の処理では、その必要がなく、既存のアルゴリズムよりも高速に処理可能である。確認しておくと、処理速度の違いは、T 行列（モード空間から実空間へ変換する行列）の違いのみである。

(2) 処理速度の見込み

新規に開発したアルゴリズムでは、この T 行列を利用した演算について、その演算量から処理速度を見積もるとつぎの通りとなる。処理時間の支配的な部分は、

$$[T]^T[M][T]$$

$$=\begin{bmatrix} I & 0 & 0 & 0 & \phi_1^T(M_{11}G_{1B}+M_{1B})\phi_B \\ & I & 0 & 0 & \phi_2^T(M_{22}G_{2B}+M_{2B})\phi_B \\ & & I & 0 & \phi_3^T(M_{33}G_{3B}+M_{3B})\phi_B \\ & & & I & \phi_4^T(M_{44}G_{4B}+M_{4B})\phi_B \\ sym. & & & & I \end{bmatrix} \tag{40}$$

における $\phi_1^T(M_{11}G_{1B}+M_{1B})\phi_B$ の演算である。実際の演算では、G_{1B}, M_{1B} のスパース性を利用して、$\phi_1^T\left(M_{11}G_{1B}\phi_B+M_{1B}\phi_B\right)$ を演算する。その演算量は、例えば 160 万自由度のシェルを中心とした実用的な検証例題 B では、

- $G_{1B}\phi_B, M_{1B}\phi_B$ ；演算回数、約 30 億回
- $M_{11}G_{1B}\phi_B$ ；演算回数、約 9 億回
- ϕ_1^T ×上記の量 ；演算回数、約 2 億回

となる（和 1 回と積 1 回で演算 1 回とカウントした）。検証計算の条件設定では、これを 512 領域で繰り返すため、1sec に 1G 回の演算が可能として、この演算のみで 2,000sec、その他の処理をあわせてモード空間での処理を 3000sec 程度で演算できることになる。一方、従来のアルゴリズムでは、モード計算 1 回に対して、この数倍の演算量が必要であったことがわかる。階層が多くなれば、さらにこの処理は増大する。このことから本アルゴリズムの優位性は明らかである。

(3) 計算精度の見込み

既存のアルゴリズムは、境界領域を多階層のモード合成法で解き、境界領域をひとつの階層として全体を 1 階層のモード合成法として解く方法である。理論上は、既存のアルゴリズムと新しく開発したアルゴリズムとでは、モード空間において、同等のレベルの基底関数を利用しているため、同等の精度が出ると考えられる。しかし、これについては、実際にソフトウェアを作成したのちに、精度を確認する必要がある。

我々は、改良したアルゴリズムに関して、該当する部分の精度について、ソフトウェアの開発の段階で、既存のアルゴリズムと同等の精度が出ることを確認した。この結果については、5 章に示す検証問題 A で述べることとする。ここで具体的に実施したことは、

$$\left(\begin{bmatrix} \lambda_1 I & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 I & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 I & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 I & 0 \\ 0 & 0 & 0 & 0 & \lambda_B I \end{bmatrix} - \lambda \begin{bmatrix} I & 0 & 0 & 0 & \phi_1^T(M_{11}G_{1B}+M_{1B})\phi_B \\ & I & 0 & 0 & \phi_2^T(M_{22}G_{2B}+M_{2B})\phi_B \\ & & I & 0 & \phi_3^T(M_{33}G_{3B}+M_{3B})\phi_B \\ & & & I & \phi_4^T(M_{44}G_{4B}+M_{4B})\phi_B \\ sym. & & & & I \end{bmatrix}\right)[\eta]=0 \tag{41}$$

のタイプの固有値問題に関する精度の確認したことである。ここで、λ_1 から λ_4 の個数は任意であり、内部領域の個数となる。これに対し、これまでの多

階層モード合成法で解いていたタイプは、内部領域が 2 つに限定されたモデル

$$\left(\begin{bmatrix} \lambda_a I & 0 & 0 \\ 0 & \lambda_b I & 0 \\ 0 & 0 & \lambda_B I \end{bmatrix} - \lambda \begin{bmatrix} I & 0 & \overline{\phi}_a^T (\overline{M}_{aa}\overline{G}_{aB} + \overline{M}_{aB})\tilde{\phi}_B \\ & I & \overline{\phi}_b^T (\overline{M}_{bb}\overline{G}_{bB} + \overline{M}_{bB})\tilde{\phi}_B \\ sym. & & I \end{bmatrix} \right) [\eta] = 0 \quad (42)$$

を繰り返し解くことに相当している。

3.2.3. まとめ

本稿のここまでに示した通り、新規に開発したモード空間での固有値問題解法における処理は、精度では既存のアルゴリズムと同程度であること、および、演算量の面で既存のアルゴリズムと比較して非常に有利なことがわかった。このような理由により、新規に開発した本アルゴリズムを採用とすることを決めた。

3.3. 定式化における注意

3.3.1. 頻繁にあらわれる行列の演算について

既存のアルゴリズムと同様に新しく開発したアルゴリズムにおいても、階層化された行列の演算を繰り返す場合に、本定式化では、

$$\begin{aligned} M'_{aa} u_a &= \lambda K'_{aa} u_a \\ K'_{aa} &= K_{aa} - K_{a1} K_{11}^{-1} K_{1a} - K_{a2} K_{22}^{-1} K_{2a} \\ M'_{aa} &= M_{aa} - M_{a1} K_{11}^{-1} K_{1a} - M_{a2} K_{22}^{-1} K_{2a} \end{aligned} \quad (43)$$

で示す特徴的な演算が随所にあらわれる。これは、一般によく知られている LDU 分解においてあらわれる手順と全く同じものである。従って、ソフトウェアを実装する際の該当箇所におけるプログラミングにおいては、従来の LDU 分解をブロック型の行列に対して行うプログラムを作成することと同値である。従って、これらのプログラム開発においては、従来型の技術が応用できる。ここでは、ブロックごとの fill-in の処理も必要になってくることには、当然、注意しなくてはならない。すなわち、ブロック行列内のみならず、ブロックごとの接続の情報も管理しておく必要がある。

3.3.2. Strum 列の計算について

前節で述べたように本アルゴリズムはLU 分解と同様の手順を踏んで行われる。この手順において、Strum 列も同時に計算することができる。確認のため定義を確認しておくと、Strum 列とは下記のような定義である。

ある与えられた実数 a に対して、$Kx = \lambda Mx$ の実数 a 以下の大きさの固有値の数を数えることは、固有値の計算を必要としなく、それよりもかなり少ない演算数でこの数を求めることが可能となる。具体的には、$K - aM$ を LDU のように LDU 分解する。ここで、D の対角項のうち、非負の対角項の数を数えることで、これが $Kx = \lambda Mx$ に対する a 以下の固有値数となる。これは、線型代数の入門編であらわれるシルベスターの慣性則を用いるれば、明らかな事実である。

実用的には、固有値計算の精度を確認するために、しばしば Strum 列の計算は必要である。本固有値ソルバーにおいて、シフトした系に対する縮合演算をすることで、本固有値ソルバーを Strum 列の演算に利用できる。本演算が、同じ使用方法で利用可能なソフトウェアで実施できることは、実用的に非常に有効であることが多い。

3.3.3. 領域分割について

本アルゴリズムにおける領域分割では、2 つの要請がある。そのひとつは、プロセス間の通信量を減少させるために領域間の境界をできるだけ少なくすることである。ふたつめは、各プロセスでの処理量を均一化するために、領域ごとの節点を均一にすることが必要である。領域分割におけるこれらの判断基準は、並列計算において、プロセス間の通信をための領域分割と同等のものである。本プログラムでの領域分割は、MeTiS 等の汎用のライブラリを利用することを前提としている。従って、領域分割については、本固有値ソルバーにおける開発要素には含めていない。本開発では、MeTiS をひとつのツールとして利用している。

4. 開発したアルゴリズムの定式化

3 章では、サンプル例題によるアルゴリズムの例示を行った。それらのアルゴリズムを踏まえ、本章では、開発したアルゴリズムの正確な定式化について述べる。

4.1. 開発したモード座標結合モード合成法

4.1.1. 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 $D^1{}_i\left(i=1, p_1\right)$その境界のノードで構成されるグラフ $D^1{}_{B,j}\left(j=1\right)$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 $D^1{}_j\left(i \neq j\right)$のノード間にエッジは存在しない。また、$D^1{}_{B,j}\left(j=1\right)$ と $D^1{}_i\left(i=1, p_1\right)$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 $D^1{}_i\left(i=1, p_1\right)$を子、その子とエッジを共有する $D^1{}_{B,j}\left(j=1\right)$を親と呼ぶ。この操作を、$D^1{}_i\left(i=1, p_1\right)$ に対して再帰的に行うことにより、 $D^2{}_i\left(i=1, p_2\right)$ と $D^2{}_{B,j}\left(j=1, p_1\right)$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、 $D^m{}_i\left(i=1, p_m\right)$ に対して、同様の操作を行えば、$D^{m+1}{}_i\left(i=1, p_{m+1}\right)$ と $D^{m+1}{}_{B,j}\left(j=1, p_m\right)$ 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 $D^m{}_i\left(i=1, p_m{}^{(j)}\right)$ 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。

$$D^1 \rightarrow D^1{}_{B,j}\left(j=1\right) \rightarrow D^2{}_{B,j}\left(j=1, p_1\right) \rightarrow \cdots \rightarrow D^m{}_{B,j}\left(j=1, p_{m-1}\right) \rightarrow D^m{}_j\left(j=1, p_m\right) \tag{44}$$

ここで、 $D^1{}_{B,j}\left(j=1\right)$を階層 1、 $D^2{}_{B,j}\left(j=1, p_1\right)$を階層 2、 $D^m{}_{B,j}\left(j=1, p_{m-1}\right)$ を階層 m の（境界）領域と呼び、最後の $D^m{}_j\left(j=1, p_m\right)$ 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 $\dim\left(D\right)$を持つ。ここで、

$$Kx = \lambda Mx; \left(\cup_{i=1, p_m{}^{(j)}} D^m{}_i\right) \cup D^m{}_{B,j} \tag{45}$$

を考える。$\left(\cup_{i=1, p_m{}^{(j)}} D^m{}_i\right) \cup D^m{}_{B,j}$ 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、 $\dim\left(D^m{}_{B,j}\right) + \sum_{i=1, p_m{}^{(j)}} \dim\left(D^m{}_i\right)$ の次元を持つ。この問題に対して、

$$Kx = 0; D^m{}_i \cup D^m{}_{B,j} \quad \left(i=1, p_m{}^{(j)}\right) \tag{46}$$

の制約条件をつける。これは、 $\sum_{i=1, p_m{}^{(j)}} \dim\left(D^m{}_i\right)$ 個の式であり、(1)と(2)を連立させて解けば $\dim\left(D^m{}_{B,j}\right)$ の次元の固有値問題となる。これを、$\left(\cup_{i=1, p_m{}^{(j)}} D^m{}_i\right) \cup D^m{}_{B,j}$ 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。

4.1.2. 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、

$$\begin{bmatrix} K^m{}_1 & 0 & 0 & 0 & 0 & K^m{}_{1,j} \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & K^m{}_i & 0 & 0 & K^m{}_{i,j} \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & K^m{}_{p^{(i)}{}_m} & K^m{}_{p^{(i)}{}_m,j} \\ K^m{}_{1,j}{}^T & \cdots & K^m{}_{i,j}{}^T & \cdots & K^m{}_{p^{(i)}{}_m,j}{}^T & K^m{}_{B,j} \end{bmatrix} \tag{47}$$

から

$$K^m{}_{B,j}{}' = K^m{}_{B,j} - \sum_{i=1,p^{(j)}{}_m} K^m{}_{i,j}{}^T K^m{}_i{}^{-1} K^m{}_{i,j} \tag{48}$$

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、

$$\begin{bmatrix} M^m{}_1 & 0 & 0 & 0 & 0 & M^m{}_{1,j} \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & M^m{}_i & 0 & 0 & M^m{}_{i,j} \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & M^m{}_{p^{(i)}{}_m} & M^m{}_{p^{(i)}{}_m,j} \\ M^m{}_{1,j}{}^T & \cdots & M^m{}_{i,j}{}^T & \cdots & M^m{}_{p^{(i)}{}_m,j}{}^T & M^m{}_{B,j} \end{bmatrix} \tag{49}$$

から

$$M^m{}_{B,j}{}' = M^m{}_{B,j} - \sum_{i=1,p^{(i)}{}_m} M^m{}_{i,j}{}^T K^m{}_i M^m{}_i K^m{}_i{}^{-1} M^m{}_{i,j} \tag{50}$$

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、

$$\begin{bmatrix} K^m{}_1 & 0 & 0 & 0 & 0 & K^m{}_{1,j} \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & K^m{}_i & 0 & 0 & K^m{}_{i,j} \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & K^m{}_{p^{(i)}{}_m} & K^m{}_{p^{(i)}{}_m,j} \\ K^m{}_{1,j}{}^T & \cdots & K^m{}_{i,j}{}^T & \cdots & K^m{}_{p^{(i)}{}_m,j}{}^T & K^m{}_{B,j} \end{bmatrix} x = \lambda \begin{bmatrix} M^m{}_1 & 0 & 0 & 0 & 0 & M^m{}_{1,j} \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & M^m{}_i & 0 & 0 & M^m{}_{i,j} \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & M^m{}_{p^{(i)}{}_m} & M^m{}_{p^{(i)}{}_m,j} \\ M^m{}_{1,j}{}^T & \cdots & M^m{}_{i,j}{}^T & \cdots & M^m{}_{p^{(i)}{}_m,j}{}^T & M^m{}_{B,j} \end{bmatrix} x \tag{51}$$

を

$$K^m{}_{B,j}{}' x = \lambda M^m{}_{B,j}{}' x \tag{52}$$

$$K^m{}_{B,j}{}' = K^m{}_{B,j} - \sum_{i=1,p^{(j)}{}_m} K^m{}_{i,j}{}^T K^m{}_i{}^{-1} K^m{}_{i,j} \tag{53}$$

$$M^m{}_{B,j}{}' = M^m{}_{B,j} - \sum_{i=1,p^{(i)}{}_m} M^m{}_{i,j}{}^T K^m{}_i M^m{}_i K^m{}_i{}^{-1} M^m{}_{i,j} \tag{54}$$

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 $\left(\cup_{i=1,p_m{}^{(j)}} D^m{}_i\right) \cup D^m{}_{B,j}$ 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 $Kx = \lambda Mx; D^m{}_{B,j}$ の固有ベクトル、および、(2)内部領域 $Kx = \lambda Mx; D^m{}_i$ 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、

$$Kx = \lambda Mx; \left(\cup_{i=1,p_m{}^{(j)}} D^m{}_i\right) \cup D^m{}_{B,j} \tag{55}$$

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 $\left(T^T KT\right)\eta = \lambda\left(T^T MT\right)\eta$ と変換し、モード空間での固有値問題に帰着する。$Kx = \lambda Mx; D^m{}_i$ の固有ベクトルを、$\varphi^m{}_i\left(i = 1, p_m{}^{(i)}\right)$ とおき、$Kx = \lambda Mx; D^m{}_{B,j}$ の固有ベクトルを、$\phi^m{}_{B,j}$ とする。

4.1.3. 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、$Kx = \lambda Mx; D^m{}_i$ の固有ベクトルを $\phi^m{}_i\left(i = 1, p_m{}^{(i)}\right)$ とおき、$Kx = \lambda Mx; D^m{}_{B,j}$ の固有ベクトルを、$\phi^m{}_{B,j}$ とおく。モード空間の基底は、

$$\Phi^m{}_1{}^T = \left(\phi^m{}_1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\right)^T \tag{56}$$

…

$$\Phi^m{}_i{}^T = \left(0 \quad 0 \quad \phi^m{}_i \quad 0 \quad 0 \quad 0\right)^T \tag{57}$$

…

$$\Phi^m{}_{p_m{}^{(i)}}{}^T = \left(0 \quad 0 \quad 0 \quad 0 \quad \phi^m{}_{p_m{}^{(i)}} \quad 0\right)^T$$

$$\Phi^m{}_{B,j}{}^T = \left(\phi^m{}_{B,j}{}^{(1)} \quad \cdots \quad \phi^m{}_{B,j}{}^{(i)} \quad \cdots \quad \phi^m{}_{B,j}{}^{\left(p_m{}^{(i)}\right)} \quad \phi^m{}_{B,j}\right)^T \tag{58}$$

である。ここで、最後の基底は、Guyan の静的縮

約条件

$$\begin{bmatrix} K^m{}_1 & 0 & 0 & 0 & 0 & K^m{}_{1,j} \\ 0 & \ddots & 0 & 0 & 0 & \vdots \\ 0 & 0 & K^m{}_i & 0 & 0 & K^m{}_{i,j} \\ 0 & 0 & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & K^m{}_{p^{(i)}_m,j} & K^m{}_{p^{(i)}_m,j} \\ K^m{}^T_{1,j} & \cdots & K^m{}^T_{i,j} & \cdots & K^m{}^T_{p^{(i)}_m,j} & K^m{}_{B,j} \end{bmatrix} \begin{bmatrix} \varphi^m{}^{(1)}_{B,j} \\ \vdots \\ \varphi^m{}^{(i)}_{B,j} \\ \vdots \\ \varphi^m{}^{(p_m^{(i)})}_{B,j} \\ \varphi^m{}_{B,j} \end{bmatrix} = 0 \quad (59)$$

を満たす。すなわち、

$$\phi^m{}^{(i)}_{B,j} = -K^m{}^{-1}_i K^m{}_{i,j} \phi^m{}_{B,j} \quad (60)$$

として決定できる。これらの基底を利用して

$T = \left(\Phi^m{}_1 \quad \cdots \quad \Phi^m{}_i \quad \cdots \quad \Phi^m{}_{p_m^{(i)}} \quad \Phi^m{}_{B,j}\right)$

とする。$Kx = \lambda Mx$ は、$x = T\eta$ を利用すると、$\left(T^T KT\right)\eta = \lambda\left(T^T MT\right)\eta$ 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、$\varphi^m{}^{(i)}_{B,j} = -K^m{}^{-1}_i K^m{}_{i,j} \varphi^m{}_{B,j}$ のうち $K^m{}^{-1}_i$ 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

$\varphi^m{}^{(i)}_{B,j} = -K^m{}^{-1}_i K^m{}_{i,j} \varphi^m{}_{B,j}$ を求める必要がある。$K^m{}_i$ は $\phi^m{}_{B,j}$ 以下の領域に関する K 行列全体であり、$K^m{}^{-1}_i$ 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、

$$\left(\begin{bmatrix} \lambda_1 I & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 I & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 I & 0 & 0 \\ 0 & 0 & 0 & \lambda_4 I & 0 \\ 0 & 0 & 0 & 0 & \lambda_B I \end{bmatrix} - \lambda \begin{bmatrix} I & 0 & 0 & 0 & \varphi_1^T (M_{11} G_{1B} + M_{1B}) \varphi_B \\ & I & 0 & 0 & \varphi_2^T (M_{22} G_{2B} + M_{2B}) \varphi_B \\ & & I & 0 & \varphi_3^T (M_{33} G_{3B} + M_{3B}) \varphi_B \\ & & & I & \varphi_4^T (M_{44} G_{4B} + M_{4B}) \varphi_B \\ sym. & & & & I \end{bmatrix} \right) [\eta] = 0 \quad (61)$$

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。

4.1.4. 全体の制御手順

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して、すべての境界領域 $D^l{}_{B,j}\left(l = 1, m\right)$ 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、

$Kx = \lambda Mx; \left(\cup_{i=1, p_k^{(j)}} D^k{}_i\right) \cup D^k{}_{B,j}$ 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；モード空間で求まった固有ベクトルを $\left(\cup_{i=1, p_k^{(j)}} D^k{}_i\right) \cup D^k{}_{B,j}$（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。

4.2. 本手法の特長

4.2.1. 特長とする項目

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。

(1) 高速な固有値計算

新規に開発したアルゴリズムにより、モード空間での高速な固有値計算を実現している。式

$\phi^{m}{}_{B,j}{}^{(i)} = -K^{m}{}_{i}{}^{-1}K^{m}{}_{i,j}\phi^{m}{}_{B,j}$ において、$K^{m}{}_{i}{}^{-1}$ は、対角ブロック行列であり、縮合過程ですでに計算されているため、本アルゴリズムでは、高速に処理できる。

一般化固有値問題を標準固有値問題に変換したことで短時間に固有値を求めることができ、さらに、その標準固有値問題を一般化固有値問題に変換することで、その固有値を高速に、かつ、安定に求めることができる。

まず、一般化固有値問題 $Kx = \lambda Mx$ を解くための、$n \times n$ 行列 K と M のデータを読み込み、領域分割を実行し、$D^{1} \to D^{1}{}_{B,j}(j=1) \to D^{2}{}_{B,j}(j=1, p_1)$ $\to \cdot\cdot\cdot \to D^{m}{}_{B,j}(j=1, p_{m-1}) \to D^{m}{}_{j}(j=1, p_m)$ のデータを作成する。

内部領域 $D^{m}{}_{j}(j=1, p_m)$ の固有値を計算する。

まず、k=m として縮合計算を行い、

$D^{k}{}_{B,j}(j=1, p_{k-1})$ の固有値を求める。すなわち、

$\left(\cup_{i=1,p_k{}^{(j)}} D^{k}{}_{i}\right) \cup D^{k}{}_{B,j}$ から $D^{m}{}_{B,j}$ に静縮合した固有値問題を解く。k を 1 だけ減じながら、k が 1 になるまで処理を繰り返す。

$D^{1}{}_{B,j}(j=1)$ におけるモード空間での固有値を求める。次に $D^{2}{}_{B,j}(j=1, p_1)$ を求める。以下、k を 1 だけ増加させながら、$D^{k}{}_{j}(j=1, p_k)$ のモード空間での固有値を求める。k=m となった時点の解が、一般化固有値問題 $Kx = \lambda Mx$ の解である。

4.2.2. 安定な固有値計算

本ソフトウェアでは、モード空間の固有値問題の等価な変換を行い、安定な固有値計算を実現している。一般化固有値問題から標準固有値問題への変換を行い、モード空間の固有値を求める。モード空間では、一般化固有値問題 $Kx = \lambda Mx$ を解く必要があるが、

$$K = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} \tag{62}$$

$$M = I + \begin{pmatrix} 0 & 0 & * \\ 0 & 0 & * \\ 0 & 0 & 0 \end{pmatrix} \tag{63}$$

の行列になっている。従って、上記の一般化固有値問題を等価な標準固有値問題へ変換すると、

$$\frac{1}{\lambda}\left(K^{1/2}x\right) = \left(K^{-1/2}MK^{-1/2}\right)\left(K^{1/2}x\right) \tag{64}$$

となる。行列 $\left(K^{-1/2}MK^{-1/2}\right)$ の固有値と固有ベクトルを求めると、$\frac{1}{\lambda}$ と $\left(K^{1/2}x\right)$ を求めることができる。

ここで、$K^{-1/2}MK^{-1/2}$ は、$M = \left(m_{ij}\right)$ とすると、

$$K^{-1/2}MK^{-1/2} = \left(\frac{m_{ij}}{\sqrt{\lambda_i}\sqrt{\lambda_j}}\right) \tag{65}$$

となる。また、M の対角項は 1 であったことから、$K^{-1/2}MK^{-1/2}$ の対角項は、$1/\lambda_i$ となる。従って、これを変換して、もとの一般化固有値問題の固有値 λ と固有ベクトル $K^{-1/2}\left(K^{1/2}x\right)$ を求める。ただし、ここで、

$$K^{1/2} = \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \end{pmatrix} \tag{66}$$

$$K^{-1/2} = \begin{pmatrix} 1/\sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & 1/\sqrt{\lambda_n} \end{pmatrix} \tag{67}$$

とおいた。

モード空間における固有ベクトルの変換は次のように行う。$\frac{1}{\lambda_i}\tilde{x}_i = \left(K^{-1/2}MK^{-1/2}\right)\tilde{x}_i$ を求めることができれば、$\tilde{x}_j{}^{T}\tilde{x}_i = \delta_{ij}$ を満たす固有ベクトルを求めることができる。この直交化の式から、

$$\delta_{ij} = \tilde{x}_j^{\,T}\tilde{x}_i = \lambda\tilde{x}_j^{\,T}\left(K^{-1/2}MK^{-1/2}\right)\tilde{x}_i = \lambda\left(K^{-1/2}\tilde{x}_j\right)^T M\left(K^{-1/2}\tilde{x}_j\right) \tag{68}$$

および、

$$\frac{1}{\lambda_i}\tilde{x}_i = \left(K^{-1/2}MK^{-1/2}\right)\tilde{x}_i \tag{69}$$

から、

$$K\left(K^{-1/2}\tilde{x}_i\right) = \lambda_i M\left(K^{-1/2}\tilde{x}_i\right) \tag{70}$$

なので、この 2 つの式から、$\left(K^{-1/2}\tilde{x}_i\right)$ は、固有値問題 $Kx = \lambda_i Mx$ の M で規格化された解となっている。

モード空間では、シフト可能な一般化固有値問題への再変換を行う。モード空間での一般化固有値問題 $Kx = \lambda Mx$ を

$$\frac{1}{\lambda}\left(K^{1/2}x\right) = \left(K^{-1/2}MK^{-1/2}\right)\left(K^{1/2}x\right) \tag{71}$$

と変換して解いた通常、標準固有値問題 $Ax = \lambda x$ に対しては、大きい固有値から計算できるため、この方法は都合がよかった。ただし、小さい固有値から計算するためには、$A^{-1}x = \frac{1}{\lambda}x$ と変換して解く必要がある。また、求める固有値の数が多い場合にはシフトして計算が必要であるが、(1)通常の標準固有値問題では、固有値範囲を指定できない。(2) $Ax = \lambda x$ のままシフトして解くとマイナスの固有値も計算される。(3)上記の理由から slice 処理が有効でない。の 3 つの理由から、多くの固有値を求めると、固有値が求まらないか、または、多くの処理時間を要することになる。従って、標準固有値問題の解きやすいという性質を保存しつつ、次の一般化固有値として、slice 処理を利用できる形式で利用する。

以下では、$\frac{1}{\lambda}\left(K^{1/2}x\right) = \left(K^{-1/2}MK^{-1/2}\right)\left(K^{1/2}x\right)$ を

$$A = K^{-1/2}MK^{-1/2} \tag{72}$$

$$x' = K^{1/2}x \tag{73}$$

として、$A^{-1}x' = \lambda x'$ として、シフトした系

$$\left(A^{-1} - \lambda_0 I\right)x' = \left(\lambda - \lambda_0\right)x' \tag{74}$$

を計算する。Lanczos 法の過程では、上式の左辺の行列の逆を、与えられた U に対して

$$\left(A^{-1} - \lambda_0 I\right)V = U \tag{75}$$

を満たす V を計算する必要がある。A のスパース性を利用するため、

$$V = \left(I - \lambda_0 A\right)^{-1}AU \tag{76}$$

で計算する。ここで、$\left(I - \lambda_0 A\right)^{-1}$ は A の M 行列と同じスパース性を保ったまま、疎行列の演算を利用して計算できる。

4.2.3. 個別領域の固有値解法

(1) 検証計算からのフィードバック

検証計算において、Block Lanczos 法が安定して解けないケースが発生したため、subspace 法および Householder 法が、スパース行列、密行列、モード空間用行列に利用できるように、本固有値ソルバーでは整備されている。ただし、Householder のスパース行列用は意味がないため整備していない。

また、いくつかの検証計算において、内部領域および境界領域については、Block Lanczos 法で問題ない場合がほとんどであった。また、固有値の数が多い場合には、slice の手法を利用して、安定に解を求めることができる。ただし、内部領域については全体の 0.5%以下のケース(200 回に 1 回という意味)であるが Block Lanczos では解を求めることができない場合もあり、その場合には、subspace 法をバックアップとして利用する。ここで利用した subspace 法の部分空間で密行列となる小さなサイズの固有値解法には Jacobi 法を利用している。さらに、境界領域については、Block Lanczos 法をメインのソルバーとするが、密行列であるため Housholder 法によるバックアップも可能である。本システムで解く 3 つの固有値問題では、モード空間の固有値を求めることが最も不安定であり、いくつかの手法を検討し、標準固有値問題に変換して Block Lanczos 法で解く手法が最も安定であることがわかり、その手法を採用した。ただし、モード空

間専用の標準固有値問題向けの Householder 法のバックアップとして利用可能である。

(2) 個別領域での固有値解法のまとめ

内部領域、境界領域、および、モード空間での固有値問題を解く必要があるが、これらの解法に関して、検証計算から次のことがわかった。

- モード空間であらわれる特殊な行列の形式を保ったまま、Householder 変換を行うことは困難である。
- 通常の Householder による三重対角化の方法では、1 段の処理を終了した時点で、小行列が密行列となる。
- Householder 変換におけるベクトル $\mathbf{w}_k$ の選択には自由度が多いため、モード空間の行列の形式を保ったまま Householder 変換ができる可能性があるように見えるが、次の理由で不可能である。1 回の Householder 変換は、鏡像変換の意味を持つため、行列の形を変えないで（モード空間での行列の形式である、左下にある空間以外は不変に保ったまま）Householder 変換を行うことは不可能であるため。

従って、下記のような手法を整備するとともに、本固有値ソルバーを頑強な固有値ソルバーとするために、本稿に示す対策を行った。

これらの手法を手軽に利用するには、ARPACK (Arnoldi Package)、SLEPc (Scalable Library for Eigenvalue Problem Computations)、MUMPS (MUltifrontal Massively Parallel sparse direct Solver)、PETSc (Portable, Extensible Toolkit for Scientific Computation)、LAPACK (Linear Algebra PACKage)、ScaLAPACK (Scalable LAPACK)等を利用できる[10] [11] [12] [13] [14] [15]。ただし、これらのライブラリを使いこなすことは、かなりの経験とノウハウが必要であることも付け加えておく。また、どのような解法をどのような基準で選択するかについては、かなりの経験を必要とすることである。これらのライブラリについては別の機会に本雑誌において概説したい。

表 5 利用した個別領域での固有値解析手法

手法	固有値問題	内部領域用	境界領域用	モード空間用
BlockLanczos	一般化	○	○	×
BlockLanczos	一般化	○	○	－
BlockLanczos	標準	－	－	○
subspace	一般化	△	△	×
Householder	一般化	－	△	×
Householder	標準	－	－	△

（○を利用、△がバックアップ、×は利用不可）

5. 検証

5.1. 検証問題A

5.1.1. 解析条件

検証問題 A として 1 万自由度（1 節点 2 自由度の 1 次四角形要素約 5000 要素）の問題を解いた。本問題は、質量行列と剛性行列が同じ非零パターンを持つようなケースを設定した。本問題は小規模であるため、従来型の疎行列用でも密行列用の固有値ソルバーでも簡単に解くことができる。従って、精度を検証するための検証例題として利用した。また、1 万自由度であるため、領域分割をしてもひとつの領域が小さすぎることがない。従って、例えば、5 階層、すなわち、32 領域に分割することも可能である。従って、本ソフトウェアをいくつかの面から検証するためには、妥当なサイズの検証問題となった。

ここでは、解析解と本固有値ソルバーで得られた解を比較した。ここで解析解とは、正確には解析解ではないが、既存の小規模固有値ソルバーで解いて、その精度を十分に検証できている値であるという意味で用いた。

用いた問題のモード図は次の通りである。検証計算では、20 次までの固有値を求めて、その精度を検証した。ただし、この解析対象においては、1,2,3 次の固有値は剛体モードとなる。すなわち、ここでは、1,2,3 次の固有値が 0 となるような問題を設定している。

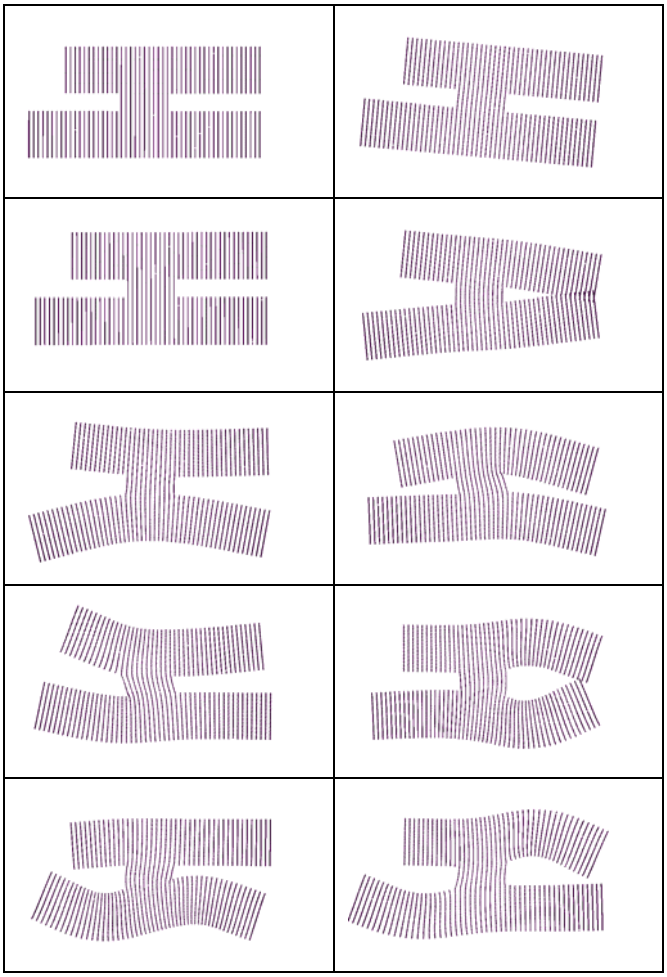

図 6　検証問題Aの振動モード

※ 左上が 1 次、右上が 2 次、順次 3 次から 10 次までの振動モードを示す。

5.1.2. モード空間の固有値解法の比較

本固有値ソルバーは、小領域の固有値を求めることとモード空間の固有値を求めること、および、全体を統合して固有値を求める部分から構成されている。まず、ここでは、モード空間の固有値を求める部分に着目した。

検証では、まず、モード空間の固有値を求める場合に、一般化固有値問題に対する Block Lanczos 法では、反復法を利用しているため、その結果に誤差を含んでいる場合があることがわかった。一方、モード空間以外の一般化固有値問題では、このような問題は発生しなかった。

この問題は、もちろん、（反復法に対する）直接法である Householder 法で解くことにより、精度が改善させる。しかし、前章に述べたように、Householder 法では、モード空間固有の行列の形を利用して演算量を節約することができないため、高速な処理をする場合に、Block Lanczos 法を利用することが必須である。従って、Blocked Lanczos 法および Householder 法に対して、一般化固有値問題、および、標準固有値問題を解いて、全体の固有値に対する精度を確認した。

表 6　各種手法による固有値の比較

解法		①		②		③		④	
	解析解	固有値	誤差	固有値	誤差	固有値	誤差	固有値	誤差
0次	0.0	0.0		0.0		0.0		0.0	
1次	0.0	0.0		0.0		0.0		0.0	
2次	0.0	0.0		0.0		0.0		0.0	
3次	28.0	28.2	0.44%	28.2	0.44%	28.2	0.44%	28.2	0.44%
4次	48.2	48.6	0.70%	48.6	0.70%	48.6	0.70%	48.6	0.71%
5次	69.7	69.9	0.30%	69.9	0.30%	69.9	0.30%	69.9	0.30%
6次	95.0	94.7	0.30%	95.2	0.13%	95.2	0.13%	95.2	0.14%
7次	155.4	95.2	38.75%	156.3	0.64%	156.3	0.64%	156.4	0.65%
8次	159.4	156.4	1.91%	160.7	0.83%	160.7	0.83%	160.7	0.84%
9次	167.7	160.7	4.17%	168.0	0.13%	168.0	0.13%	168.0	0.15%
10次	195.1	168.0	13.91%	192.8	1.21%	192.8	1.21%	192.8	1.20%
11次	207.7	192.8	7.18%	206.8	0.44%	206.8	0.44%	206.9	0.40%
12次	274.8	206.9	24.72%	273.8	0.37%	273.8	0.37%	274.0	0.31%
13次	307.3	281.9	8.26%	308.7	0.45%	308.7	0.45%	309.0	0.53%
14次	329.9	311.1	5.68%	328.7	0.35%	328.7	0.35%	329.0	0.28%
15次	333.7	338.5	1.46%	338.9	1.56%	338.9	1.56%	339.0	1.60%
16次	382.2	339.5	11.17%	380.9	0.35%	380.9	0.35%	381.1	0.30%
17次	413.6	419.5	1.41%	413.3	0.09%	413.3	0.09%	413.5	0.02%
18次	432.0	432.7	0.16%	441.7	2.24%	441.8	2.25%	442.2	2.34%
19次	461.8	486.1	5.27%	453.3	1.82%	453.3	1.83%	453.9	1.69%

本表は、以下の方法をモード空間の固有値手法として利用した結果である。

① 一般化固有値問題に対する Block Lanczos
② 標準固有値問題に対する Block Lanczos
③ 一般化固有値問題に対する Householder
④ 標準固有値問題に対する Householder

この結果、一般化固有値問題の Block Lanczos 法以外については、解析解と同等レベルまで精度が出ていることがわかる。

従って、一般化固有値問題を標準固有値問題に変換して解くことは妥当と考えられ、かつ、標準固有値問題に変換した Block Lanczos 法を最終的には採用することにした。ただし、反復法であるため収束しない場合もあり、その場合には、バックアップとして、標準固有値問題に対する Householder 法を用いるものとする。

ここで、全体の固有値を求める場合の制御は、内部領域では 15 個の固有値を求め、境界領域では 15 個の固有値を求め、モード空間では 30 個の固有値を求める制御方法を利用した。また、Block Lanczos 法には、ブロック数というパラメータがある。この

詳細については、ここでは述べないが[10]等を参照のこと。ここでは、ブロック数は 6 を利用した。以下特に断らない限り、ブロック数は 6 を利用している。

5.1.3. 精度のパラメータの依存性

本固有値ソルバーには、いくつかのパラメータがある。ここでは、そのパラメータのうち、最小固有値数と最大周波数に関する感度解析を行い、精度を比較した。

本固有値ソルバーでは、内部領域と境界領域およびモード空間において部分的な固有値を求めながら、全体の固有値を求めるという手法である。その際には、全固有値を求めるわけではなく、部分的に固有値を求め、演算を継続するという手法を利用している。従って、部分的な固有値を求める場合にいくつの固有値を求めていくかという戦略は全体の精度を保証する上で重要である。本固有値ソルバーでは、指定された上限周波数以下の固有値またはここで指定する最小固有値数のうち、数の多い方の固有値を求めるという制御をしている。

また、20 次の固有値までを求めるため、上限の周波数は 470Hz 程度となる。上限の周波数の指定については、反復法系列の固有値ソルバーを利用する場合には必ずパラメータとして指定することが必要である。上限周波数については、次のような注意が必要である。例えば、470Hz 以下の固有値を求める場合には、上限周波数の何倍かまでを上限として固有値を求めることで 470Hz 以下の固有値の精度を上げることができる。また、この上限固有値をどの程度にするかという基準は最終的な処理時間に大きく依存するため、非常に重要なパラメータとなっている。

ここで、精度を比較する対象は前節で述べた解析解（この解の妥当性は前節で述べた）である。また、すべての計算において、ここでは、次の固有値解法を利用した。内部領域では一般化固有値問題に対する Block Lanczos 法、 境界領域では一般化固有値問題に対する Block Lanczos 法、モード空間においては、標準固有値問題に対する Block Lanczos 法を利用した。

さらに、最小固有値を 10 と固定して、上限周波数を求めるべき値の 470Hz に対して約 20 倍の 10000Hz までとしてみた。また、ここでのテストでは、多階層のモード合成法を利用しているが、その階層数は 5 としている。本固有値ソルバーでは、階層数を 1 と指定した場合には、一般化固有値問題に対する Block Lanczos 法が適用される。この場合には、モード空間の固有値ソルバーは起動されない。さらに以下のテストでは、最小固有値数を、10 から、15, 20, 30 と変更してその精度を確認した。

テストで変更したパラメータのうち、上限周波数については、その区間に存在する固有値の数に比例して処理量が増大する。また、最小固有値数については、その数に比例して処理時間は増大する。

表 7　精度の周波数依存性；最小固有値数 10

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	10	10	10	10	10	10	10
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4次	28.0	28.2	28.2	28.2	28.2	28.2	28.2	28.2
5次	48.2	48.6	48.6	48.6	48.6	48.6	48.6	48.6
6次	69.7	69.9	69.9	69.9	69.9	69.9	69.9	69.9
7次	95.0	95.3	95.3	95.3	95.3	95.3	95.3	95.3
8次	155.4	156.3	156.3	156.3	156.3	156.3	156.3	156.2
9次	159.4	161.0	161.0	161.0	161.0	161.0	160.9	160.9
10次	167.7	169.2	169.2	169.2	169.2	169.2	169.1	169.1
11次	195.1	195.3	195.3	195.3	195.3	195.3	195.3	195.2
12次	207.7	209.0	209.0	208.9	208.9	208.9	208.8	208.8
13次	274.8	277.1	277.5	276.9	276.8	276.6	276.5	276.5
14次	307.3	309.8	309.8	309.6	309.5	309.3	309.2	309.1
15次	329.9	331.8	331.8	331.6	331.5	331.3	331.2	331.1
16次	333.7	338.1	337.6	337.3	337.2	337.0	336.8	336.8
17次	382.2		384.1	383.9	383.8	383.6	383.4	383.3
18次	413.6		418.9	416.0	415.7	415.3	415.1	415.0
19次	432.0		469.5	438.3	437.5	437.0	436.6	436.6
20次	461.8		562.6	466.6	466.1	465.5	465.2	465.2

表 8　誤差の周波数依存性；最小固有値数 10

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	10	10	10	10	10	10	10
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	-	-	-	-	-	-	-
2次	0.0	-	-	-	-	-	-	-
3次	0.0	-	-	-	-	-	-	-
4次	28.0	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%
5次	48.2	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%
6次	69.7	0.35%	0.35%	0.34%	0.34%	0.34%	0.34%	0.34%
7次	95.0	0.26%	0.26%	0.25%	0.25%	0.25%	0.25%	0.25%
8次	155.4	0.61%	0.61%	0.61%	0.61%	0.59%	0.58%	0.58%
9次	159.4	1.00%	1.00%	0.99%	0.99%	0.98%	0.97%	0.97%
10次	167.7	0.87%	0.87%	0.86%	0.86%	0.85%	0.83%	0.83%
11次	195.1	0.10%	0.10%	0.09%	0.09%	0.08%	0.07%	0.07%
12次	207.7	0.61%	0.63%	0.58%	0.58%	0.55%	0.53%	0.53%
13次	274.8	0.84%	0.97%	0.75%	0.71%	0.65%	0.61%	0.60%
14次	307.3	0.79%	0.80%	0.73%	0.70%	0.63%	0.60%	0.59%
15次	329.9	0.58%	0.58%	0.53%	0.50%	0.43%	0.39%	0.39%
16次	333.7	1.31%	1.17%	1.10%	1.06%	1.01%	0.93%	0.92%
17次	382.2	-	0.50%	0.44%	0.42%	0.35%	0.31%	0.29%
18次	413.6	-	1.27%	0.56%	0.49%	0.40%	0.34%	0.34%
19次	432.0	-	8.66%	1.45%	1.26%	1.16%	1.06%	1.05%
20次	461.8	-	21.83%	1.04%	0.95%	0.81%	0.75%	0.74%

表 9　精度の周波数依存性；最小固有値数 15

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	15	15	15	15	15	15	15
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4次	28.0	28.2	28.2	28.2	28.2	28.2	28.2	28.2
5次	48.2	48.6	48.6	48.6	48.6	48.6	48.6	48.6
6次	69.7	69.9	69.9	69.9	69.9	69.9	69.9	69.9
7次	95.0	95.3	95.3	95.3	95.3	95.3	95.3	95.3
8次	155.4	156.3	156.3	156.3	156.3	156.3	156.3	156.2
9次	159.4	161.0	161.0	161.0	161.0	161.0	160.9	160.9
10次	167.7	169.2	169.2	169.2	169.2	169.1	169.1	169.1
11次	195.1	195.3	195.3	195.3	195.3	195.3	195.2	195.2
12次	207.7	208.9	208.9	208.9	208.9	208.8	208.8	208.8
13次	274.8	276.7	276.6	276.6	276.6	276.6	276.5	276.5
14次	307.3	309.4	309.4	309.3	309.3	309.2	309.2	309.1
15次	329.9	331.5	331.4	331.3	331.3	331.2	331.2	331.1
16次	333.7	337.7	337.3	337.0	337.0	336.9	336.8	336.8
17次	382.2		383.7	383.6	383.6	383.5	383.4	383.3
18次	413.6		416.5	415.6	415.3	415.2	415.1	415.0
19次	432.0		438.6	437.5	437.0	436.9	436.6	436.6
20次	461.8		469.8	465.9	465.7	465.4	465.2	465.2

表 10　誤差の周波数依存性；最小固有値数 15

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	15	15	15	15	15	15	15
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	-	-	-	-	-	-	-
2次	0.0	-	-	-	-	-	-	-
3次	0.0	-	-	-	-	-	-	-
4次	28.0	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%
5次	48.2	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%
6次	69.7	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%
7次	95.0	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%
8次	155.4	0.59%	0.59%	0.59%	0.59%	0.58%	0.58%	0.58%
9次	159.4	0.98%	0.98%	0.98%	0.98%	0.97%	0.97%	0.97%
10次	167.7	0.84%	0.84%	0.84%	0.84%	0.84%	0.83%	0.83%
11次	195.1	0.09%	0.09%	0.08%	0.08%	0.08%	0.07%	0.07%
12次	207.7	0.56%	0.55%	0.55%	0.55%	0.54%	0.53%	0.53%
13次	274.8	0.68%	0.66%	0.65%	0.65%	0.63%	0.61%	0.60%
14次	307.3	0.68%	0.66%	0.64%	0.63%	0.62%	0.59%	0.59%
15次	329.9	0.49%	0.47%	0.45%	0.44%	0.42%	0.39%	0.38%
16次	333.7	1.21%	1.08%	1.00%	0.99%	0.96%	0.93%	0.92%
17次	382.2	-	0.40%	0.37%	0.36%	0.33%	0.30%	0.29%
18次	413.6	-	0.70%	0.48%	0.41%	0.39%	0.34%	0.33%
19次	432.0	-	1.53%	1.27%	1.16%	1.13%	1.06%	1.05%
20次	461.8	-	1.74%	0.91%	0.85%	0.79%	0.75%	0.74%

表 11　精度の周波数依存性；最小固有値数 20

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	20	20	20	20	20	20	20
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4次	28.0	28.2	28.2	28.2	28.2	28.2	28.2	28.2
5次	48.2	48.6	48.6	48.6	48.6	48.6	48.6	48.6
6次	69.7	69.9	69.9	69.9	69.9	69.9	69.9	69.9
7次	95.0	95.3	95.3	95.3	95.3	95.3	95.3	95.3
8次	155.4	156.3	156.3	156.3	156.3	156.3	156.3	156.2
9次	159.4	161.0	160.9	160.9	160.9	160.9	160.9	160.9
10次	167.7	169.1	169.1	169.1	169.1	169.1	169.1	169.1
11次	195.1	195.3	195.3	195.3	195.3	195.3	195.2	195.2
12次	207.7	208.8	208.8	208.8	208.8	208.8	208.8	208.8
13次	274.8	276.6	276.6	276.6	276.5	276.5	276.5	276.5
14次	307.3	309.3	309.3	309.2	309.2	309.2	309.2	309.1
15次	329.9	331.3	331.3	331.2	331.2	331.2	331.1	331.1
16次	333.7	337.2	337.2	336.9	336.9	336.8	336.8	336.8
17次	382.2	383.7	383.7	383.5	383.5	383.5	383.4	383.3
18次	413.6	416.9	416.4	415.5	415.2	415.2	415.1	415.0
19次	432.0	439.0	438.3	437.2	436.9	436.8	436.6	436.6
20次	461.8	492.9	469.7	465.7	465.5	465.4	465.2	465.2

表 12　誤差の周波数依存性；最小固有値数 20

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	20	20	20	20	20	20	20
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	-	-	-	-	-	-	-
2次	0.0	-	-	-	-	-	-	-
3次	0.0	-	-	-	-	-	-	-
4次	28.0	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%
5次	48.2	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%
6次	69.7	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%
7次	95.0	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%
8次	155.4	0.58%	0.58%	0.58%	0.58%	0.58%	0.58%	0.58%
9次	159.4	0.97%	0.97%	0.97%	0.97%	0.97%	0.97%	0.97%
10次	167.7	0.84%	0.84%	0.84%	0.84%	0.84%	0.83%	0.83%
11次	195.1	0.08%	0.08%	0.08%	0.07%	0.07%	0.07%	0.07%
12次	207.7	0.54%	0.54%	0.54%	0.54%	0.54%	0.53%	0.53%
13次	274.8	0.65%	0.64%	0.63%	0.63%	0.63%	0.61%	0.60%
14次	307.3	0.63%	0.64%	0.62%	0.61%	0.61%	0.59%	0.59%
15次	329.9	0.45%	0.45%	0.42%	0.41%	0.41%	0.39%	0.39%
16次	333.7	1.06%	1.05%	0.96%	0.95%	0.95%	0.93%	0.92%
17次	382.2	-	0.38%	0.33%	0.33%	0.32%	0.30%	0.29%
18次	413.6	-	0.67%	0.44%	0.38%	0.37%	0.34%	0.33%
19次	432.0	-	1.45%	1.20%	1.13%	1.11%	1.06%	1.05%
20次	461.8	-	1.71%	0.85%	0.81%	0.78%	0.75%	0.74%

表 13　精度の周波数依存性；最小固有値数 30

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	30	30	30	30	30	30	30
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3次	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4次	28.0	28.2	28.2	28.2	28.2	28.2	28.2	28.2
5次	48.2	48.6	48.6	48.6	48.6	48.6	48.6	48.6
6次	69.7	69.9	69.9	69.9	69.9	69.9	69.9	69.9
7次	95.0	95.3	95.3	95.3	95.3	95.3	95.3	95.3
8次	155.4	156.3	156.3	156.3	156.3	156.3	156.3	156.2
9次	159.4	160.9	160.9	160.9	160.9	160.9	160.9	160.9
10次	167.7	169.1	169.1	169.1	169.1	169.1	169.1	169.1
11次	195.1	195.3	195.3	195.3	195.3	195.3	195.2	195.2
12次	207.7	208.8	208.8	208.8	208.8	208.8	208.8	208.8
13次	274.8	276.5	276.5	276.5	276.5	276.5	276.5	276.5
14次	307.3	309.2	309.2	309.2	309.2	309.2	309.2	309.1
15次	329.9	331.2	331.2	331.2	331.2	331.2	331.1	331.1
16次	333.7	336.9	336.9	336.8	336.8	336.8	336.8	336.8
17次	382.2	383.5	383.5	383.4	383.4	383.4	383.4	383.3
18次	413.6	416.0	416.0	415.4	415.1	415.1	415.0	415.0
19次	432.0	437.4	437.4	437.1	436.7	436.7	436.6	436.6
20次	461.8	466.3	466.3	465.7	465.3	465.3	465.2	465.2

表 14　誤差の周波数依存性；最小固有値数 30

方法	解析解	多階層	多階層	多階層	多階層	多階層	多階層	多階層
階層	1	5	5	5	5	5	5	5
最小固有値数	ー	30	30	30	30	30	30	30
上限周波数	1000	400	600	1000	2000	3000	5000	10000
1次	0.0	-	-	-	-	-	-	-
2次	0.0	-	-	-	-	-	-	-
3次	0.0	-	-	-	-	-	-	-
4次	28.0	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%	0.45%
5次	48.2	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%	0.72%
6次	69.7	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%	0.34%
7次	95.0	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%	0.25%
8次	155.4	0.58%	0.58%	0.58%	0.58%	0.58%	0.58%	0.58%
9次	159.4	0.97%	0.97%	0.97%	0.97%	0.97%	0.97%	0.97%
10次	167.7	0.83%	0.83%	0.83%	0.83%	0.83%	0.83%	0.83%
11次	195.1	0.07%	0.07%	0.07%	0.07%	0.07%	0.07%	0.07%
12次	207.7	0.54%	0.54%	0.54%	0.54%	0.54%	0.53%	0.53%
13次	274.8	0.62%	0.62%	0.62%	0.61%	0.61%	0.61%	0.60%
14次	307.3	0.61%	0.61%	0.61%	0.60%	0.60%	0.59%	0.59%
15次	329.9	0.42%	0.42%	0.41%	0.40%	0.40%	0.39%	0.39%
16次	333.7	0.95%	0.95%	0.95%	0.93%	0.93%	0.93%	0.92%
17次	382.2	-	0.34%	0.32%	0.31%	0.31%	0.30%	0.29%
18次	413.6	-	0.57%	0.42%	0.36%	0.35%	0.34%	0.33%
19次	432.0	-	1.24%	1.18%	1.08%	1.07%	1.06%	1.05%
20次	461.8	-	0.98%	0.86%	0.78%	0.77%	0.75%	0.74%

5.1.4. 検証問題Aのまとめ

これらの結果から、まず、モード空間における固有値ソルバーには、標準固有値問題に変換して Block Lanczos 法を適用することが最も妥当であることがわかった。また、計算パラメータの最小固有値および上限周波数については、最小固有値は 10 程度および上限周波数は求めたい周波数の 2 倍か 3 倍程度をとればいいことがわかった。

5.2. 検証問題 B

5.2.1. 解析条件

検証例題 B 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。

5.2.2. 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わかる。

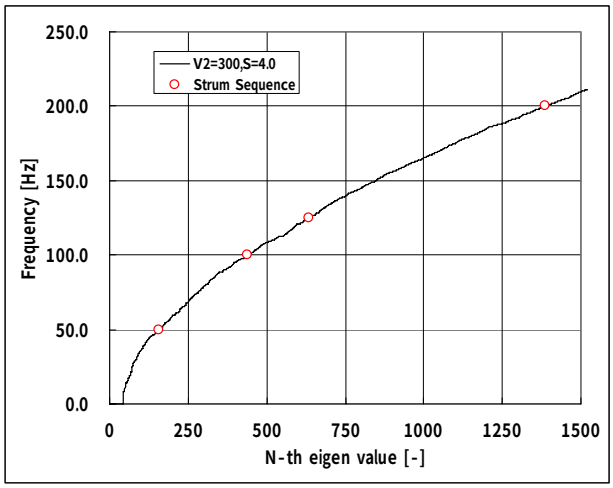

図 7 Strum 列結果との精度比較

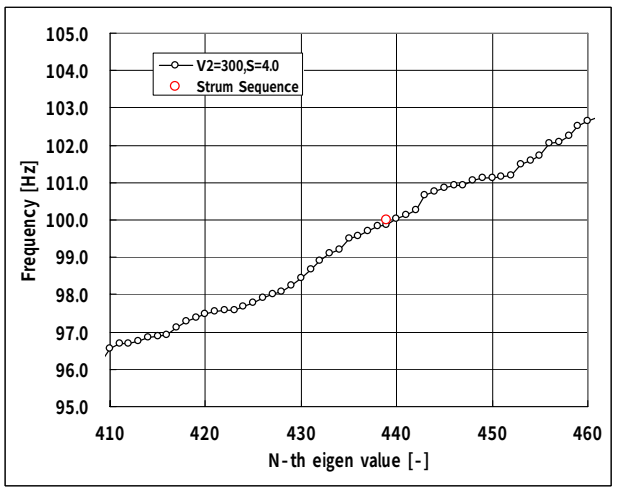

図 8 Strum 列結果との精度比較（拡大図）

5.2.3. 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わかる。

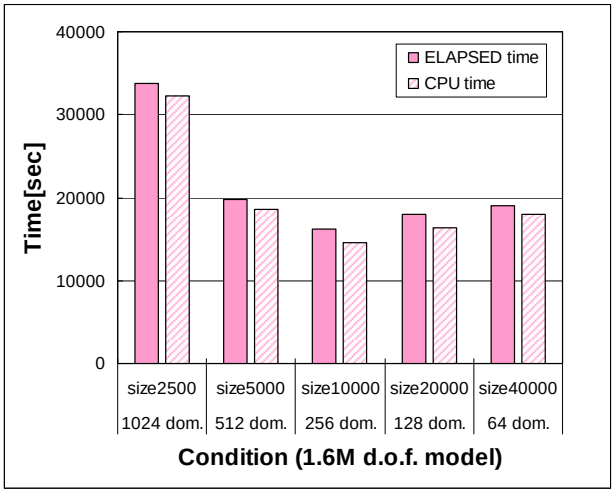

図 9　処理時間の領域分割サイズ依存性

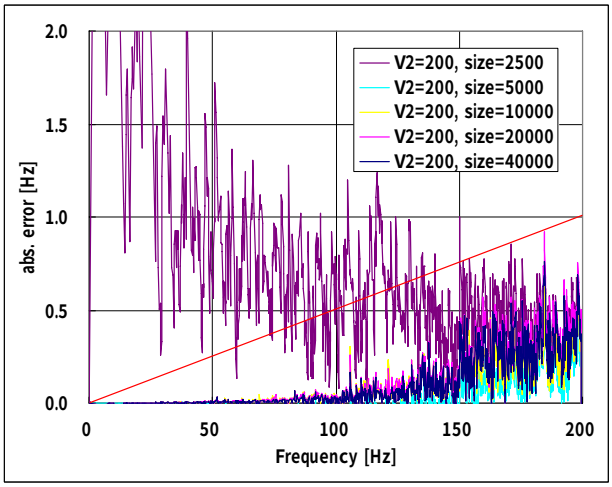

図 10　精度の領域分割サイズ依存性

5.2.4. 上限周波数依存性

ここでのパラメータは、ユーザ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=200Hz 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・s=200Hz ×2.0=400Hz 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、当然上限周波数が大き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・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わかる。

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 はユーザの都合で与えるパラメータである。f はユーザ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には簡単には決められないという事情がある。

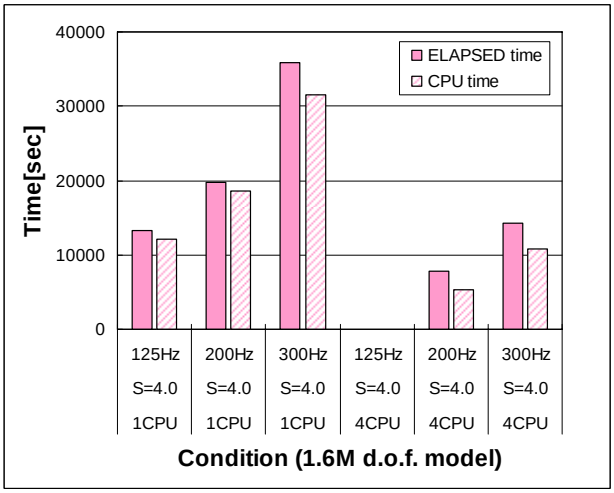

図 11 処理時間の上限周波数依存性（その 1）

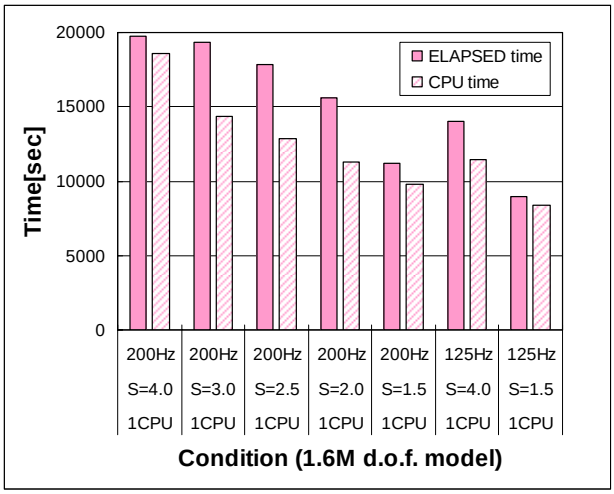

図 12 処理時間の上限周波数依存性（その 2）

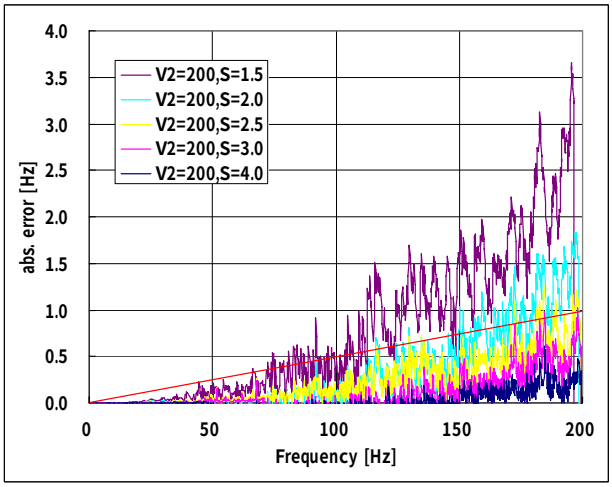

図 13 精度の上限周波数依存性（その 1）

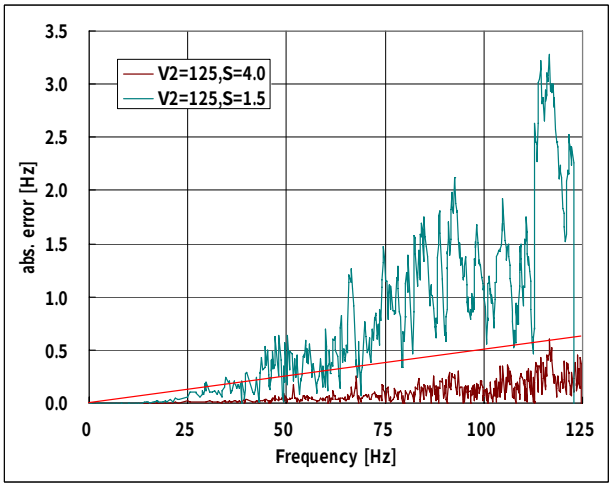

図 14 精度上限周波数依存性（その 2）

5.2.5. 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 B 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 B 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。

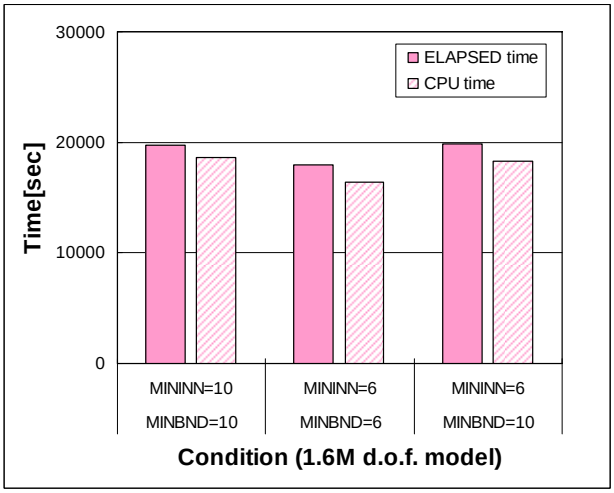

図 15 処理時間の部分領域固有値数への依存性

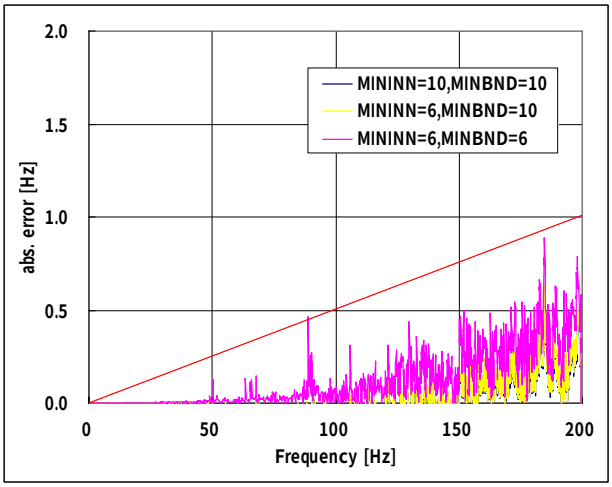

図 16 精度の部分領域固有値数への依存性

5.2.6. 並列計算と処理速度

検証の最後として、並列処理と処理速度の関係につき検証した。本固有値ソルバーでは、内部領域および境界領域における固有値を求める場合には領域ごとの並列化を、モード空間の固有値を求める場合には周波数に関する並列化および領域ごとの並列化を行っている。また、縮合計算においても、領域ごとの並列化を行っている。

本固有値ソルバーは、本開発ではアルゴリズムの開発に主眼をおいたため、並列化については、代表的な手法を取り入れたのみであり、詳細な並列のチューニングは行っていない。その前提の上で、本ソフトウェアで処理時間は、1CPU と 4CPU の処理時間の比較で、ほぼ 2.5 倍から 3.0 倍の処理速度が出ていることがわかる。もちろん、処理結果については、それぞれのケースで完全に一致している。

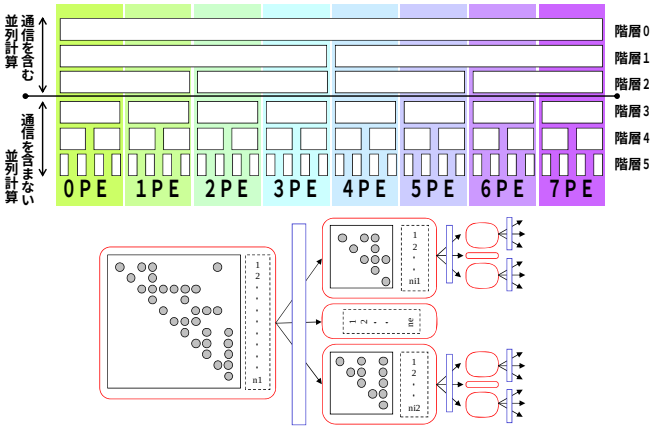

図 17　並列計算の方法

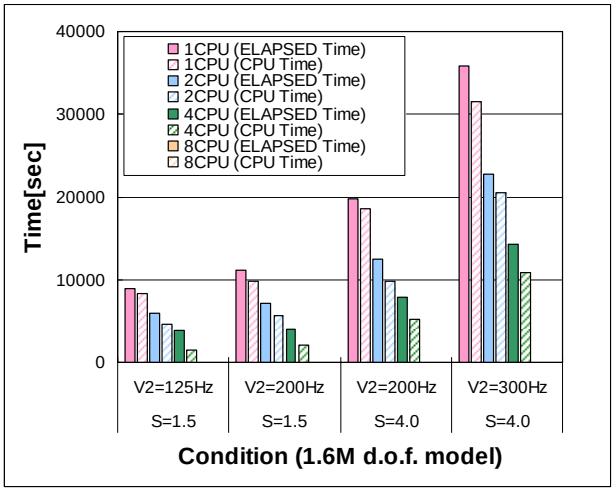

図 18 並列計算と処理速度（その 1）

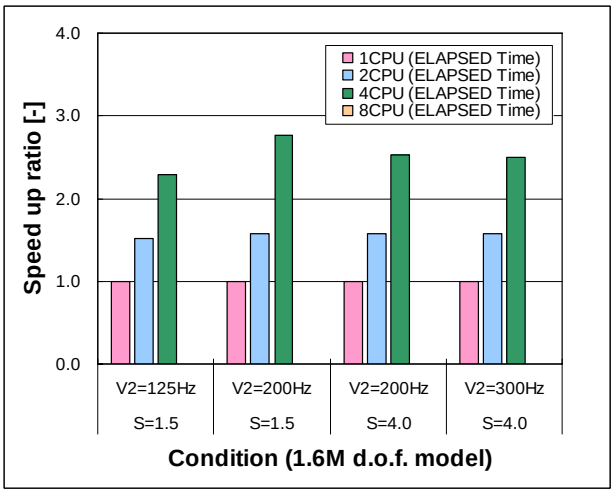

図 19 並列計算と処理速度（その 1）

5.2.7. 検証問題 B のまとめ

ここで実施した計算から、各種のパラメータの推奨値を得ることができた。ここで得られたのは、内部領域に対する計算する固有値の最小個数と最大個数、境界領域に対する計算する固有値の最小個数と最大個数、モード空間に対する計算する固有値の最小個数と最大個数の推奨値である。

また、本問題における特殊なケースとして、自由度が小さすぎた場合に適用できない手法も存在することもわかった。従って、境界領域の自由度数またはモード空間の自由度数が、指定した値よりも小さい場合には、Householder 法で実施するというソフトウェアとして対策も行った。この方法は、処理時間と精度にはまったく影響がない。

表 15 計算パラメータの推奨値

内容	推奨値
内部領域固有値の最小個数	10
内部領域固有値の最大個数	500
境界領域固有値の最小個数	10
境界領域固有値の最大個数	500
モード空間固有値の最小個数	10
モード空間固有値の最大個数	1,000
境界領域の自由度数の最小	10
モード空間の自由度数の最小	10

6. 残された課題と将来計画

本稿では、既存のアルゴリズムでは大規模問題に対して適用が困難であったモード合成法に対して、それを改良し大規模固有値計算に適合したアルゴリズムを提案した。そのソフトウェアを開発し、その検証を実施した。ここでは、見込み通りの性能を得られたことについて報告した。従って、ここで提案した多階層モード座標結合モード合成法のアルゴリズムは、既存の方法を演算量の面で改良した方法として極めて有用であり、大規模問題が実用的な演算時間および使用記憶容量での処理が可能であることがわかった。また、ここで提案した多階層モード座標結合モード合成法は、精度の面でも十分に妥当な計算結果を得ることができた。最後に、検証計算では、実用的な計算で必要となる計算パラメータの決定を行い、今後の本ソフトウェアの普及を進めるための情報を得た。

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が、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、本文でも述べたように、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の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),"日本機会学会第 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, 桑原匠人 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)

[5] 戸川隼人;”マトリクスの数値計算,” オーム社(1971)

[6] 一松信;”数値解析,”朝倉書店(1982)

[7] 村田健郎;”線形代数と線形計算法序説,“サイエンス社(1986).

[8] F.シャトラン著, 伊理 正夫, 伊理由美訳;”行列の固有値問題,”シュプリンガー・フェアラーク東京 (1993)

[9] 矢川元基,青山裕司;”有限要素固有値解析,“森北出版(2001)

[10] ARPACK;http://www.caam.rice.edu/software/ARPACK/

[11] SLEPc;http://www.grycap.upv.es/slepc/

[12] MUMPS;http://www.enseeiht.fr/lima/apo/MUMPS/

[13] PETSc;http://www.mcs.anl.gov/petsc/

[14] LAPACK;http://www.netlib.org/lapack/

[15] ScaLAPACK;http://www.netlib.org/scalapack/scalapack_home.html