Fri. Jun 2, 2023

Room C

OS27 先進並列シミュレーション

[C-10] OS27 先進並列シミュレーション (1)

座長:奥田 洋司(東京大学)
9:00 AM - 10:15 AM Room C (2F Conference Room 201A)

[C-10-01] AIを用いたピクトグラムのモーション化システムの開発

*岡谷 夏実[1]、塩谷 隆二[1]、中林 靖[1]、多田 光利[1] （1. 東洋大学）

9:00 AM - 9:15 AM

[C-10-02] AI技術を用いた油中ガス分析用アプリケーションの開発

*鄭 宏杰[1]、塩谷 隆二[1]、久冨 友里奈[1]、的場 大[2]、中嶋 恵一[2]、岡倉 秀行[2]、中村 広樹[2] （1. 東洋大学、2. 富士電機株式会社）

9:15 AM - 9:30 AM

[C-10-03] 実モデルを用いたヴァイオリンの大規模振動試解析

小嶋 美宇[1]、*塩谷 隆二[1]、横山 真男[2]、武居 周[3]、矢川 元基[1] （1. 東洋大学、2. 明星大学、3. 宮崎大学）

9:30 AM - 9:45 AM

[C-10-04] 行列固有値問題のフィルタを用いた近似固有対の精度改善法

*村上 弘[1] （1. 東京都立大学）

9:45 AM - 10:00 AM

[C-10-05] oneAPIを用いた様々なデバイス上でのステンシル計算の実装

*佐久間 大我[1]、下川辺 隆史[1]、大森 拓郎[1] （1. 東京大学）

10:00 AM - 10:15 AM

OS27 先進並列シミュレーション

[C-11] OS27 先進並列シミュレーション (2)

座長:塩谷 隆二(東洋大学)
10:30 AM - 11:15 AM Room C (2F Conference Room 201A)

[C-11-01] 並列有限要素構造解析コード ADVENTURE_Solidの富岳向けチューニング

*宮村 倫司[1]、小池 邦昭[2]、吉村 忍[3] （1. 日本大学、2. 株式会社先端力学シミュレーション研究所、3. 東京大学）

10:30 AM - 10:45 AM

[C-11-02] Accelerating lattice Boltzmann method simulation with GPU computation using C++ standard language parallelism

*袁 子恒[1]、下川辺 隆史[1] （1. 東京大学）

10:45 AM - 11:00 AM

[C-11-03] 階層型領域分割法による電磁場解析での四倍精度演算 CG法， COCG法

*杉本 振一郎[1] （1. 八戸工業大学）

11:00 AM - 11:15 AM

OS27 先進並列シミュレーション

[C-10] OS27 先進並列シミュレーション (1)

座長:奥田 洋司(東京大学)

Fri. Jun 2, 2023 9:00 AM - 10:15 AM Room C (2F Conference Room 201A)

[C-10-01] AIを用いたピクトグラムのモーション化システムの開発

*岡谷 夏実[1]、塩谷 隆二[1]、中林 靖[1]、多田 光利[1] （1. 東洋大学）

9:00 AM - 9:15 AM

[C-10-02] AI技術を用いた油中ガス分析用アプリケーションの開発

*鄭 宏杰[1]、塩谷 隆二[1]、久冨 友里奈[1]、的場 大[2]、中嶋 恵一[2]、岡倉 秀行[2]、中村 広樹[2] （1. 東洋大学、2. 富士電機株式会社）

9:15 AM - 9:30 AM

[C-10-03] 実モデルを用いたヴァイオリンの大規模振動試解析

小嶋 美宇[1]、*塩谷 隆二[1]、横山 真男[2]、武居 周[3]、矢川 元基[1] （1. 東洋大学、2. 明星大学、3. 宮崎大学）

9:30 AM - 9:45 AM

[C-10-04] 行列固有値問題のフィルタを用いた近似固有対の精度改善法

*村上 弘[1] （1. 東京都立大学）

9:45 AM - 10:00 AM

[C-10-05] oneAPIを用いた様々なデバイス上でのステンシル計算の実装

*佐久間 大我[1]、下川辺 隆史[1]、大森 拓郎[1] （1. 東京大学）

10:00 AM - 10:15 AM

計算工学講演会論文集　Vol.28 (2023年5月)　　計算工学会

AIを用いたピクトグラムのモーション化システムの開発

Development of a system to convert pictograms into motion-pictograms using AI

岡谷 夏実[1)]，塩谷 隆二[2)]，中林 靖[3)]，多田 光利[4)]
Natsumi Okatani, Ryuji Shioya, Yasushi Nakabayashi and Terutoshi Tada

1) 東洋大学大学院（〒350-0815 埼玉県川越市鯨井2100, E-mail: s4b102210019@toyo.jp)
2) 工博 東洋大学 総合情報学研究科 教授（〒350-0815 埼玉県川越市鯨井2100, E-mail: shioya@toyo.jp）
3) 工博 東洋大学 総合情報学研究科 教授（〒350-0815 埼玉県川越市鯨井2100, E-mail: nakabayashi@toyo.jp）
4) 東洋大学 総合情報学研究科 教授（〒350-0815 埼玉県川越市鯨井2100, E-mail: t_tada@toyo.jp）

Pictograms are graphic symbols that expresses the meaning or concept of an idea by using the form of the target item. They are used for emergency exit signs, public facilities, etc., in order to inform the information in a non-verbal way. However, due to the limitations of expression through still images and differences in expression by nationality and culture, not all pictograms can be understood clearly by everyone. Therefore, a system is developed that recognizes still pictograms using AI and converts them into motion graphics.

Key Words : *AI, Image Recognition, Pictograms, Motion Graphics*

1.　はじめに

ピクトグラムとは意味するものの形を使ってその意味概念を表すグラフィックシンボル（図記号）であり，言語に頼ることなく，特定の意味を視覚的に認知させる [1]．従来は静止画により，非常口サインや公共施設の案内表示などに使用されてきたが，近年では，東京五輪において「動くピクトグラム」[2]が登場したように，アニメーションにより情報を拡張，モーショングラフィックス化されたピクトグラムが用いられ始めている．本研究では，これを「モーションピクトグラム」と称する．

ピクトグラムは，意味伝達の目的と，設置場所の景観を損なわない程度の存在感が求められることから，シンプルかつ安易な表現で，また，利用者が素早く理解できるデザインが求められる．しかしながら，表そうとする事象によっては伝達内容の複雑さや国籍・文化による表現の違いを乗り越えなければならないといった課題もあり，静止画による表現では限界がある．誰もが正確な意味を一見して理解できるものばかりではないのが現状である．

そこで，アニメーションによって情報補完することで，複雑な事象や静止画では伝わりづらい動きも表現することができ，より直感的で正確な理解が可能となる．「言語を必要とせず意味内容を伝える」というピクトグラムの意義を強化することとなり，例えば，表示が母国語でない場合や，利用者が子どもである場合のように，文字での説明が理解できなくても情報を伝えられることで，危険の回避や注意に役立つ．国際化，多様化の進む現代社会において，誰もが使いやすい街づくりに貢献することができる．

2.　目的

ピクトグラムはJIS規格ピクトグラム（案内用図記号 JIS Z8210）[3]として策定されている104項目をはじめ，公用，民間合わせ数多くの種類が存在する．国内外を合わせればさらに多種多様なピクトグラムがみられる．また，新型コロナウイルスによる社会的な影響を受けて，ソーシャルディスタンスやマスク着用，黙食といった数多くのピクトグラムが考案されたように，社会の状況や必要に応じて日々新たなピクトグラムが作成，更新されている．

これに対し映像作成の手間は静止画の作成に比べ多大かつ専門性も求められ，時間や人的資源も要する．こうしたアニメーション作成のコストは，モーションピクトグラムが浸透しづらい理由のひとつでもある．

そこで，静止画のピクトグラムからモーションピクトグラムを自動的に生成することができれば，モーションピクトグラムによる情報伝達を，実現可能かつ一般に取り入れやすい手段として浸透させることができる．言語を要しないコミュニケーション手段として，現状の静止画ピクトグラムの抱える課題を克服し，安定かつ広範な情報供給が可能になると考えた．本研究では，AIを用いて静止画のピクトグラムを認識することで，モーショングラフィックス化を行うシステムを開発する．

3.　AIと画像認識を用いたコンテンツ生成

AIと画像認識を用いて認識結果からコンテンツを生成することは，これまで，教育分野に適用し，紙面に描かれた空間図形を認識し，結果をもとにした3Dモデルを生成することにより，空間図形の学習支援に適用してきた．[4]

本研究では，これを公共空間デザインの分野に適用する．なかでも，静止画ピクトグラムの判定，画像認識や特徴量抽出といった部分が活用できる．認識内容をもとにしたコンテンツ生成という部分では共通するが，3Dモデルからアニメーションへと生成内容が変化する点で，これに応じたシステムの構築が必要となる．

4. モーションピクトグラム

最終的な成果のイメージとして，静止画ピクトグラムをもとにしたモーションピクトグラムを手動で作成した．例として，図1に静止画のピクトグラムを示す．

図1は単体かつ静止画の状態では何を示したものか分かりにくいが，JIS Z8210「転落注意（Caution, drop）」を示したピクトグラムである．そこで，モーションピクトグラム化を行えば，落下の動作はアニメーションによって明確に示され，転落に対する注意喚起の意義を強めることになる．作成したモーションピクトグラムについて，フレーム分割した様子を図2に示す．

5. システム構成

対象のピクトグラムが表現しようとしているモチーフ，意味するものの形を，AIおよび画像認識を用いて判定し，判定結果から対象の動きを推定する．例えば，対象が人型ピクトグラムであれば「歩く」「走る」といった動作，対象が水や煙であれば物理法則に基づいた「流れる」「漂う」「落下する」といった動作，矢印による図示があればその方向に対象が移動，といったように対象の動作を推測し決定する．

図 1 静止画ピクトグラム「転落注意」

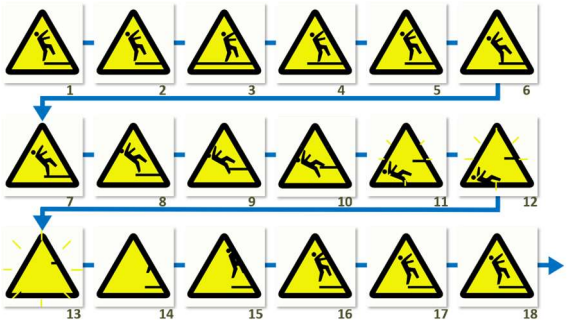


図 2 モーションピクトグラム「転落注意」
（フレーム分割）

こうした推定結果をもとにモーショングラフィックスを生成し，モーションピクトグラム化を行う．特に，人型や動物型のピクトグラムでは，判定した対象に応じた実写映像や既存のアニメーションの動きを対象の静止画ピクトグラムへと転移させることで，新たなもの，未知のものであってもモーショングラフィックス化へ対応することを考える．

6. 人型ピクトグラムのモーション化

図-3は「非常口」を示す静止画ピクトグラムである．この人型ピクトグラムに対し，実写映像から「走る」動作の転移を行い，新たなモーションピクトグラム生成を行った．

(1) Thin-Plate Spline Motion Modelの使用

モーションピクトグラム生成のため，Thin-Plate Spline Motion Model[5]を用いた動作転移を行った．動作転移には事前学習済みモデル「TaiChiHD」を使用した．生成したモーションピクトグラムについて，動作転移の様子を図-4に示す．各フレーム左下部分に動作転移に使用した実写映像，右上に対象とする静止画ピクトグラムを配置している．

図-4に示した生成結果では，人型ピクトグラムの周囲にグレーのノイズが生じた．また，右足先端部分の動作が転移せずに一点に固定され，右膝のみが動作している状態となった．これは，人型ピクトグラムと非常口の周囲の壁面を示す枠が重なっており，それらが同色で表現されていることが原因と考えられる．

図 3 静止画ピクトグラム「非常口」

図 4 モーションピクトグラム「非常口」
（フレーム分割）

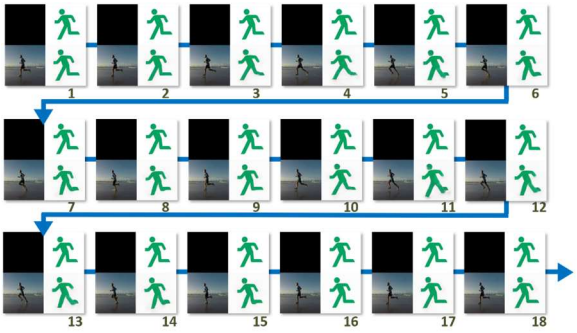


図 5 モーションピクトグラム
「非常口（背景なし）」（フレーム分割）

そこで，非常口周囲の壁面部分を除去し，人型ピクトグラムにフォーカスした状態での動作転移を試行した．動作転移の様子を図-5に示す．

図-5に示した生成結果では，図-4の生成結果で見られたようなノイズは低減することができた．しかし，右足先端部分の動作は転移できず，固定されたままであった．これは，使用した実写映像と対象の人型ピクトグラムの初期姿勢の違いと，使用した学習モデルが原因と考えられる．人型ピクトグラムでは右足が地面に平行な状態まで曲げられているが，実写映像ではそこまで高く曲げられておらず，右足先端部分がやや下の位置にある．また，動作転移において，人体検出に使用した学習済みモデルが，実写映像に対応したものであり，ピクトグラムのようなデフォルメされた表現には適していなかったものと考えられる．これらの問題を解決するためには，初期姿勢の差分を修正する処理を加え，ピクトグラムの表現に対応した学習モデルを作成，対応させることが必要である．

（2）Liquid Warping GAN with Attentionの使用

Liquid Warping GAN with Attention[6]では，3次元人体メッシュ復元モジュールによりモデルの体型を再現し，回転や関節位置に対応可能であることが示されている．Liquid Warping GAN with Attention の再現手法であるiPERCoreを用いた動作転移を試行した．しかしながら，図-3「非常口」の静止画ピクトグラムでは，画像内に人体が認識されず，動作転移を行うことができなかった．また，背景部分を手動で削除した画像を用いての実行でも，同様に画像内に人体が認識されなかったため，使用した人体検出モデルが，デフォルメされた人型ピクトグラムに対しては適応されないことが示唆された．更に，静止画ピクトグラムの種類を変え，図-6に示される「転落注意」「更衣室」といった姿勢や色の異なる対象でも試行したものの，人体は認識されない結果となった．一方，「陸上競技場」のピクトグラムでは人体の認識に成功した．しかしながら，動画との対応クロッピングおよび正面画像の生成部分が動作せず，モーションピクトグラムの生成はできなかった．現状のシステムでは，実写画像の特徴に基づいた検出や生成を行っているため，ピクトグラムの表現に適した改善が必要である．対応する認識モデルを作成，適用させることで，人型ピクトグラム専用の3次元人体メッシュ復元の方法を検討，動作転移の実行を目指す．

図 6 ピクトグラム
「転落注意」「更衣室マーク」「陸上競技場」

7. まとめ

本研究では，静止画ピクトグラムからモーションピクトグラムの自動生成を行うことにより，モーションピクトグラムの利活用をより一般的で現実的な手段とすることを目指す．静止画ピクトグラムの課題である表現の限界や複雑な事象の伝わりにくさをモーションピクトグラムの活用によって解決することで，言語によらない情報伝達をより安定的で効果的なものとし，公共の場での注意喚起や情報伝達といった役割を高める．静止画のピクトグラムを入力として用い，ピクトグラムが表現する意味概念を考慮した適切な動作を自動的に生成システム開発に取り組んだ．

将来的な展望としては，駅や博物館などの公共施設に設置されている電子ディスプレイでの利用および，認識したピクトグラムをマーカ，生成結果を表示コンテンツとしたAR化への活用も期待できる．ARによるモーションピクトグラム化が実現できれば，現実に設置されている静止画のピクトグラムをそのままに，拡張的にモーションピクトグラムを取り入れることが可能となる．新たなピクトグラムに対してもリアルタイムで情報の拡張，更新を行うことができる．ARグラスをはじめとした関連デバイスの発展および一般への普及とともに，本研究における認識や自動生成といった利点の活用にも期待したい．

参考文献

[1] 太田幸夫: 国際安全標識のピクトグラムデザインの研究, 平成15年度 共同研究費研究成果報告書, 2004.

[2] 野波健祐: 入場行進にドラクエ 世界に魅力伝わった？五輪開会式, 朝日新聞デジタル, 2021-7-27.

[3] 国土交通省: 案内用図記号（JIS Z8210）, 案内用図記号（JIS Z8210）（令和元年7月20日）, 2019.

[4] 岡谷夏実, 塩谷隆二, 中林靖, 多田光利: AI と AR を用いた平面図の 3 次元モデル化システムの開発, 日本計算工学会論文集, 2022 巻, p.20220009, 2022.

[5] Jian Zhao Hui Zhang: Thin-Plate Spline Motion Model for Image Animation, School of Software, BNRist, Tsinghua University, Beijing, China, 2022.

[6] Wen Liu, Zhixin Piao, Zhi Tu, Wenhan Luo, Lin Ma, Shenghua Gao: Liquid Warping GAN with Attention: A Unified Framework for Human Image Synthesis, School of Software, BNRist, Tsinghua University, Beijing, China, 2022.

9:15 AM - 9:30 AM (Fri. Jun 2, 2023 9:00 AM - 10:15 AM Room C)

[C-10-02] AI技術を用いた油中ガス分析用アプリケーションの開発

*鄭 宏杰[1]、塩谷 隆二[1]、久冨 友里奈[1]、的場 大[2]、中嶋 恵一[2]、岡倉 秀行[2]、中村 広樹[2] （1. 東洋大学、2. 富士電機株式会社）

計算工学講演会論文集　Vol.28 (2023年5月)　　計算工学会

実モデルを用いたヴァイオリンの大規模振動試解析

Large scale vibration analysis of a violin using a real model

小嶋 美宇[1)], 塩谷 隆二[1)], 横山 真男[2)], 武居 周[3)], 矢川 元基[1)]
Misora Kojima, Ryuji Shioya, Masao Yokoyama, Amane Takei and Genki Yagawa

1) 東洋大学（〒350-8585 埼玉県川越市鯨井2100, E-mail: shioya@toyo.jp）
2) 明星大学（〒191-8506 東京都日野市程久保2-1-1）
3) 宮崎大学（〒889-2155 宮崎県宮崎市学園木花台西1-1）

The numerical simulations of the mode vibrations of violin are performed by the finite element method using super computer and the software. The geometry of violin is scanned by micro CT scanner and the orthotropic properties of spruce and maple, such as the Young's modulus, the rigidity modulus and the Poisson's ratio, are set in the parameters of numerical simulation. The shapes of the main mode vibration are shown in this paper.

Key Words : *Vibrations, Violin, Numerical Simulation, Eigenvalue analysis*

1.　はじめに

ヴァイオリンの振動解析は，近年の数値シミュレーションによって解析が進んでいる[1-3]．しかし，楽器の音色（Timbre）の良し悪しの議論はいまだに経験的であり主観的である．これまでにも多くの科学者や製作者がヴァイオリンの名器として知られるStradivariusの作品を様々な角度から分析し，音色の美とは何かを説明しようと試みてきた．本研究では，実際のStradivariusによるヴァイオリンをスキャンしたモデルを用いた大規模振動解析の試みについて紹介する．また，三次元解析が可能なソフトウェアを用いて木材の性質によってヴァイオリンの振動数がどのように変化していくのかを調査した．

2.　数値シミュレーションについて

(1) マイクロ CT スキャンとメッシュ生成

数値シミュレーションに用いる 3D のジオメトリデータの取得について概要を示す．マイクロ CT スキャナ(精度，0.1mm)を用いて，17 世紀から 18 世紀のイタリアのヴァイオリン製作者の一人であるAntonio Stradivari(1719年)の楽器をスキャンした．スキャン時のエラーやノイズである断片や小孔を除去したのちSTLファイルとして保存した．そのポリゴンデータをADVENTURECluster のBuilderにインポートした．メッシュの要素種別は四面体二次要素に設定し，基本節点間隔は30mmで定めた．

(2) 木材の直行異方性の考慮

ヴァイオリンに用いられるような木材を計算対象とする場合では，木材の直行異方性を数値シミュレーションに設定する必要がある．ヴァイオリンの材料の物理的特性を，木目方向（Longitudinal）とそれに直交する年輪方向（Radial），そして年輪の周方向 （Tangential）の3方向の直交異方性でモデル化した．直交異方性の設定値については，Table 1 に示すように，ヤング率（縦弾性係数，Young's modulus），剛性率（Rigidity modulus），ポアソン比（Poisson's ratio）であり，それぞれの値は文献[4]で計測された代表値を用いた．ここで，Table 1のE_T, E_R と剛性率の各値は長さ方向のヤング率 E_L に対する比率である．また，Table 2はポアソン比に対する比率である.ヤング率 E_L については，文献[4]のメープルの代表値である12.6GPa を，また，スプルースは9.9Gpaを用いた．ヴァイオリンの材料には主にスプルースとメープル，黒檀などが用いられる．密度は，スプルースが0.36×10^3 kg/m^3，メープルが0.63×10^3kg/m^{3}である．黒檀は等方性材料とみなし，比重1.2×10^3kg/m^3，ヤング率1,200kg/m^3，ポアソン比0.2とした．しかし，これらの値はスキャンしたヴァイオリンの実際の物性値とは異なるため，今後のオールド楽器の物性値をどう計測するかを検討する必要がある．

Table 1 Values for Orthotropic properties for setting in numerical simulation

Property	Maple	Spruce
Young's module E_R / E_L	0.132	0.078
E_T / E_L	0.065	0.043
Rigidity modulus G_{LR} / E_L	0.111	0.064
G_{RT} / E_L	0.021	0.003
G_{LT} / E_L	0.063	0.061

Table 2 Values for Orthotropic properties for setting in numerical simulation

Property	Maple	Spruce
Poisson's ratio μLR	0.424	0.372
μRT	0.774	0.435
μLT	0.476	0.467
Density(kg/m^3)	0.63×10^3	0.36×10^3

(3) 有限要素法解析

商用ソフトウェアであるADVENTUREClusterを用いて，固有値解析，モーダル周波数応答解析を行った．求める周波数は，文献[3]より200Hzから500Hzの範囲とした．Intel Xeon W-2223，256GByteのメモリを搭載したWindowsサーバを用いて2Coreでの計算を行った．

3. シミュレーション結果

(1) ヴァイオリンの振動モード

実際のヴァイオリンの板の厚みは均一ではなく，中央部が厚く周辺部へかけて緩やかに薄くなっている．数値シミュレーションにおいても，この厚みを考慮しないと正しい計算結果が得られなく，表面のアーチの形状のみをスキャンするだけでは不十分である[3]．よって，本研究のように3Dモデリングなどにより楽器の内側の座標の取得による正しい厚みの情報が重要である．また，スーパーコンピューターを用いてさらなる大規模振動解析を行った．今回は，モード振動の条件下でメープルとスプルースの2種類の材料について調査した．ここで，Table 3 にヴァイオリン本体の振動モードを示す[1,2]．A0 は breath mode とも呼び表板全体が上下に振動する吸込み湧き出しの単極子のような振動モードで，音量に関連すると言われる．また，CBR (Center Bout Rotation)モードというねじれモードや，表板の振動に関連する B1-と裏板の振動に関連する B1+(Corpus mode)がある．B1-は音のダイナミクスや音色(明るい，暗い)に，そして B1+はパワーに関係すると言われている．今回の研究では計算コストなどの制約もあり，楽器の表板のみの振動と周辺音場に絞って計算を行った．

Table 3 Frequencies of A0, A1, B1-, B1+ and CBR[1,2]

Mode	Frequency
A0 (first cavity mode)	270 - 290
CBR (Center bout rotation)	380 - 420
B1- (corpus mode, body mode)	445 – 465
B1 + (corpus mode, body mode)	520 - 540
A1 (second cavity mode)	460

(2) 楽器の周波数

固有値解析の結果では，節点数430,340，要素数247,183，計算時間は96,754.2秒という結果が得られた．周波数350Hzのときに A0モードの振動に近い解析結果が出力された．Fig.1は350Hzにおける，ヴァイオリンの真上(表板側)から見た振動の様子を示している．Fig.2は350Hzにおける，ヴァイオリンの真下(裏板側)から見た振動の様子を示している．

モーダル周波数応答解析については，現在解析中であり，解析終了後に振動モードを分析する予定である．

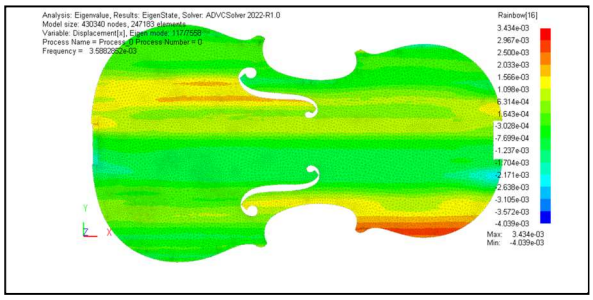


Fig.1 Displacement of violin (Front Side)

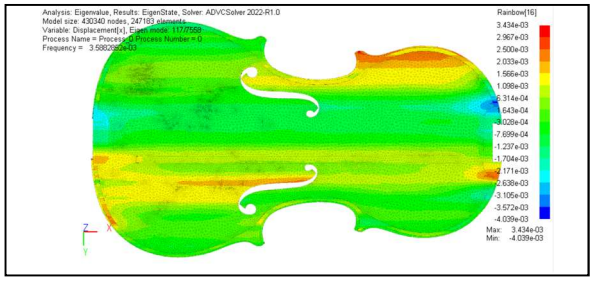


Fig.2 Displacement of violin (Back Side)

4. おわりに

本研究ではヴァイオリンの実物モデルを用いた大規模振動解析を行った．今回は表板のみの解析を実現したが，今後，すでにモデル化を終えている部材を追加し，ヴァイオリン全体解析を行う予定である．さらに，弦の加振による楽器の振動と音場の解析に拡張する予定である．

参考文献

[1] Woodhouse J., "The acoustics of the violin: a review." Reports on Progress in Physics 77.11 (2014) 115901.
[2] Gough, C. E. "A violin shell model: vibrational modes and acoustics" The Journal of the Acoustical Society of America 137.3 (2015): 1210-1225.
[3] Yokoyama, M., DE Lucia, R. R., Antonacci, F., & Sarti, A. Influence of orthotropic properties on vibration of violin top plates, Proceedings of International Conference on Acoustics 2019 (2019).
[4] Green, D. W., Winandy, J. E., and Kretschmann, D. E. “Mechanical properties of wood. Wood handbook: wood as an engineering material” Madison, WI: USDA Forest Service, Forest Products Laboratory, 1999. General technical report FPL; GTR-113: (1999): 4.1-4.45, 113.

計算工学講演会論文集 Vol. 28 (2023 年 5 月)
計算工学会

行列固有値問題のフィルタを用いた近似固有対の精度改善法

Accuracy Improvement of Approximate Eigenpairs of Matrix Eigenproblems by Using Filters

村上弘[1)]
Hiroshi Murakami

[1)]東京都立大学（〒 192-0397 東京都八王子市南大沢 1-1 理学部 8 号館 6 階数理科学事務室気付，E-mail: mrkmhrsh@tmu.ac.jp)

By using a filter, we approximate a part of eigenpairs of a matrix eigenproblem. The filter is applied to a set of randomly generated vectors to make another set of vectors which contains required eigenvectors rich, and from the span of the set of vectors the required approximate eigenpairs are extracted well. However, if the characteristics of the filter are not good, or if the amount of the required eigenvectors contained in the set of initial random vectors is very little, then the accuracy of the approximate eigenpairs obtained will be poor. Therefore, we study some method to improve the accuracy of the approximate eigenpairs to be solved.

Key Words : *Eigenproblem, Eigenpairs, Filter, Improvement, Preconditioning*

1. はじめに

実対称定値の一般固有値問題に対して，固有値が指定された狭い区間に入る固有対の近似をフィルタを用いて求める．ランダムに生成されたベクトルの組に対してフィルタを適用して，必要な固有ベクトルを豊富に含むベクトルの組を作り，そのベクトルの組の張る空間から必要な固有対の近似をうまく取り出す．フィルタの特性が良くなかったり，あるいは最初のランダムなベクトルの組に含まれている必要な固有ベクトルの量が少ないとそれだけ近似固有対の精度は悪くなる．そこで本論文ではフィルタを利用した前処理により必要な固有対の近似精度を改善する方法について検討する．

2. 準備

実対称定値（行列 A と B が実対称で B が正定値）の一般固有値問題 $A\mathbf{v} = \lambda B\mathbf{v}$ の固有対でその固有値が指定された区間 $[a, b]$ にあるものの数が数百程度以下の場合に，フィルタを利用してそれらの近似を一斉に求める．そのために通常は以下のような方法で計算を行う．

1. 固有値が $[a, b]$ にある必要な固有ベクトルは良く伝達するが不要なものは強く減衰させる性質を持つように構成された線形作用素をフィルタ $\mathcal{F}$ とする．
2. 十分多くのランダムなベクトルを生成してベクトルの組 Y とする．
3. ベクトルの組 Y に B-正規直交化を施してベクトルの組 X を作る（$X^T BX = I$ となる）．
4. ベクトルの組 X にフィルタを適用して $Y \equiv \mathcal{F}X$ を作る．フィルタの持つ性質から，Y は不要な固有ベクトルをほとんど含まないベクトルの組になる．
5. Y の線形結合をうまく構成して，固有値が区間 $[a, b]$ にある固有ベクトル全体で張られた不変部分空間を良く近似する空間の基底 $\mathcal{V}$ を作る．
6. 基底 $\mathcal{V}$ に Rayleigh-Ritz 法を適用することにより必要な固有対の近似を一斉に得る．

必要な固有ベクトルに対するフィルタの伝達率の大きさが不均一であるほど，必要な固有対の近似の精度が不均一となる傾向が生じるが，ベクトルの組 Y に対して「B-正規直交化に続いてフィルタを適用する」という上記のステップ 3 と 4 の操作を繰り返すことで必要な固有対全体の近似の精度や近似の不均一さも改善できる（このことは既知である）．たとえば B-正規直交化を間にはさんで同じフィルタを 2 回適用する場合には，ランダムに生成したベクトルの組を B-正規直交化したベクトルの組と比べて 1 回目のフィルタを適用して得られたベクトルの組の含む不要な固有ベクトルの量は減少するが，それを再び B-正規直交化して 2 回目のフィルタを適用すると得られたベクトルの組の含む不要な固有ベクトルの量はさらに減少するので，それが張る部分空間は不変部分空間のより良い近似となる．

フィルタを計 2 回適用する場合に，最初に適用するものを前処理用のフィルタとして，2 回目に適用するものを本来のフィルタとすると，前処理を行わない場合と行う場合とでは操作はそれぞれ以下のようになる．

前処理を行わない場合： 乱数ベクトルの組 Y から B-正規直交化により m 個のベクトルの組 X を作る．その X に（本来の）フィルタを適用して Y を作る．

前処理を行う場合： 乱数ベクトルの組 Y から B-正規直交化により m 個のベクトルの組 X を作る．その X に前処理用のフィルタを適用して Y を作る．さらにそのベクトルの組 Y から（閾値切断付き）B-正規直交化により m' 個のベクトルの組 X を作る．その X に本来のフィルタを適用して Y を作る．しかも前処理用と本来の 2 つのフィルタは違うものでも構わない．

今回用いるフィルタの構成はレゾルベントの線形結合 $\mathcal{L}$ に Chebyshev 多項式を合成したものである．Chebyshev の 3 項漸化式に基づいて多項式の次数が n 次であるフィルタを適用する計算は，ベクトルの組に対する $\mathcal{L}$ の作用を n 回順次に繰り返して適用することになる．

今回のフィルタの構成方式自体は前処理用も本来のものも同じである．レゾルベントの線形結合 $\mathcal{L}$ は共通であるが，Chebyshev 多項式の次数を本来のフィルタでは n とするが前処理用のものでは ν とする．そうして

通常は $\nu < n$ とする．すると近似固有対に対して要求される精度を達成できる次数 ν をうまく選べるのであれば，次数 n のフィルタを単純に 2 回用いるよりは必要な計算の手間を減らせるという利点が生じる．

3. フィルタの構成

シフトが ρ のレゾルベントの定義を $\mathcal{R}(\rho) \equiv (A-\rho B)^{-1}B$ とする．シフトが相異なる K 個のレゾルベント $\mathcal{R}(\rho_j)$ の線形結合を $\mathcal{L}$ として，フィルタ $\mathcal{F}$ は $\mathcal{L}$ の n 次 Chebyshev 多項式の定数 g_s 倍であるとする（式 (1)）．

$$\left\{ \begin{array}{lcl} \mathcal{F} & = & g_s\, T_n(\mathcal{L}), \\ \mathcal{L} & = & c_\infty\, \mathcal{I} + \sum_{j=1}^{K} \gamma_j\, \mathcal{R}(\rho_j). \end{array} \right. \tag{1}$$

レゾルベントの線形結合 $\mathcal{L}$ が実作用素となるように，定数 c_∞ は実数とし，虚数のシフトは複素共役対を為して現れ，シフトが複素共役であるレゾルベントについての結合係数は複素共役であり，シフトが実数のレゾルベントの結合係数は実数とする．

いま扱っている一般固有値問題の任意の固有対 $(\lambda, \mathbf{v})$ に対しては $\mathcal{F}\mathbf{v} = f(\lambda)\mathbf{v}$ が成立する．ここで $f(\lambda)$ はフィルタ $\mathcal{F}$ の伝達関数と呼ばれる実関数で，フィルタ $\mathcal{F}$ の式 (1) に対応して実有理関数 $y(\lambda)$ の n 次 Chebyshev 多項式の定数 g_s 倍である（式 (2)）．逆に式 (2) の形の伝達関数に対応して式 (1) の構成を持つフィルタが決まる．

$$\left\{ \begin{array}{lcl} f(\lambda) & \equiv & g_s\, T_n(y(\lambda)), \\ y(\lambda) & \equiv & c_\infty + \sum_{j=1}^{K} \gamma_j \,/\, (\lambda - \rho_j). \end{array} \right. \tag{2}$$

下端の固有対を求める場合には，区間 $[a, b]$ の幅を $\mu\,(>1)$ 倍に拡げた区間を $[a, b']$ として，伝達関数 $f(\lambda)$ が以下の条件 (3) を満たすように結合係数 γ_j とシフト ρ_j，j=1, 2, …, K，実数 c_∞，および Chebyshev 多項式の次数 n をうまく選ぶ．ここで μ，g_s，g_p は，伝達関数 $f(\lambda)$ のグラフの形状を代表する 3 つの形状パラメタで，$0 < g_s \ll g_p < 1$ を満たす．

$$\left\{ \begin{array}{ll} \text{通過域 } \lambda \in [a, b] & \text{で } g_p \le f(\lambda) \le 1, \\ \text{遷移域 } \lambda \in (b, b') & \text{で } g_s < f(\lambda) < g_p, \\ \text{阻止域 } \lambda \in [b', \infty) & \text{で } |f(\lambda)| \le g_s. \end{array} \right. \tag{3}$$

求めたい固有値の区間 $\lambda \in [a, b]$ と単位区間 $t \in [0, 1]$ を同じ向きに対応させる λ から t への線形変換を用いて固有値の座標 λ に対応する正規化座標 t を定義する．そうして正規化座標 t を引数とする伝達関数を $g(t) \equiv f(\lambda)$ により定義する（そのグラフの例を図 1 に示す）．

(1) フィルタのベクトルの組への作用

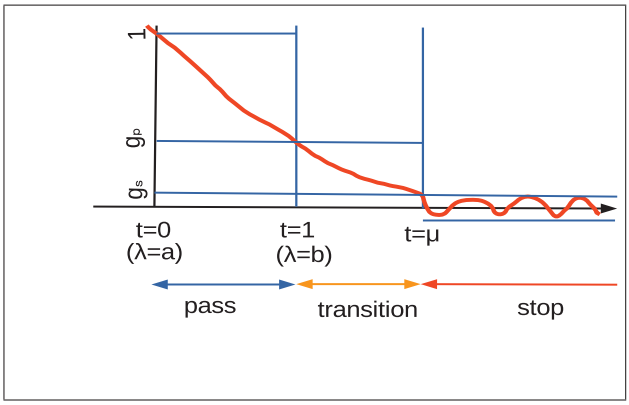
フィルタ $\mathcal{F}$ をベクトルの組 X に適用する計算 $\mathcal{F}X = g_s\, T_n(\mathcal{L})\, X$ では，まず $X^{(j)} \equiv T_j(\mathcal{L})\, X$ とおいて 3 項漸化式 (4) を用いて順番に $X^{(j)}$，j=1, …, n を計算すれば，$\mathcal{F}X$ は $g_s\, X^{(n)}$ に等しい．

$$\left\{ \begin{array}{l} X^{(0)} = X, \quad X^{(1)} = \mathcal{L}X, \\ X^{(\ell)} = 2\mathcal{L}X^{(\ell-1)} - X^{(\ell-2)}, \ (\ell \ge 2). \end{array} \right. \tag{4}$$

4. 多項式の次数だけを変えたフィルタによる前処理

(1) 同じフィルタを 2 回繰り返す計算の場合

レゾルベントの線形結合 $\mathcal{L}$ の n 次 Chebyshev 多項式のフィルタは式 (1) で与えられる．B-正規直交化を間に



図–1 伝達関数 $g(t) \equiv f(\lambda)$ の概形

はさんで同じフィルタ $\mathcal{F}$ を 2 回適用する計算は式 (5) になる．

$$\left\{ \begin{array}{lcl} X & \leftarrow & \text{ランダムな } B\text{–正規直交ベクトル } m \text{ 個の組}; \\ Y & \leftarrow & g_s\, T_n(\mathcal{L})\, X; \quad \text{※ 1 回目の } n \text{ 次のフィルタ} \\ X & \leftarrow & Y \text{ の（閾値切断付きの）} B\text{–正規直交化}; \\ Y & \leftarrow & g_s\, T_n(\mathcal{L})\, X. \quad \text{※ 2 回目の } n \text{ 次のフィルタ} \end{array} \right. \tag{5}$$

すると作用素 $\mathcal{L}$ の適用回数が 2 倍に増えて $2n$ になる．しかし我々は作用素 $\mathcal{L}$ の適用回数を 2 倍にまで増やさずにある程度の近似の向上を得たいとする．

(2) 次数を変えたフィルタによる前処理

両方のフィルタでレゾルベントの組 $\mathcal{L}$ は共通にするが，Chebyshev 多項式の次数を ν（$\nu < n$）に変えた前処理用のフィルタをランダムな B-正規直交ベクトルの組に先に適用して，その結果に B-正規直交化を施してから多項式の次数が n である本来のフィルタを適用すると，レゾルベントの線形結合 $\mathcal{L}$ の適用回数は合計で $\nu + n$ になる．

$$\left\{ \begin{array}{lcl} X & \leftarrow & \text{ランダムな } B\text{–正規直交ベクトル } m \text{ 個の組}; \\ Y & \leftarrow & \widetilde{g}_s\, T_\nu(\mathcal{L})\, X; \quad \text{※ 前処理の } \nu \text{ 次のフィルタ} \\ X & \leftarrow & Y \text{ の（閾値切断付きの）} B\text{–正規直交化}; \\ Y & \leftarrow & g_s\, T_n(\mathcal{L})\, X. \quad \text{※ 本来の } n \text{ 次のフィルタ} \end{array} \right. \tag{6}$$

ここで $\widetilde{g}_s$ は ν 次のフィルタの最大伝達率を 1 に規格化するための定数で，式 (7) で計算できる．

$$\widetilde{g}_s \equiv 1/\cosh\left(\frac{\nu}{n}\, \cosh^{-1}\frac{1}{g_s}\right). \tag{7}$$

5. 数値実験

(1) 例題の設定と固有対の近似精度の評価法

例題にする行列の対称定値一般固有値問題 $A\mathbf{v} = \lambda B\mathbf{v}$ は，1 辺の長さ π の立方体に零境界条件を課した 3 次元ラプラス作用素 $-\Delta$ の固有値問題を有限要素法で離散化して導かれるものである．有限要素法の要素は立方体の各辺方向を N_1+1，N_2+1，N_3+1 に等分した辺を持つ直方体とし，要素内の基底関数には 3 重線形関数を用いた．以下の例題では $(N_1, N_2, N_3) = (40, 50, 60)$ としたので，係数の行列 A と B は共に次数 N は 120,000 で半帯幅 w_h は 2,041 である．計算で使用したすべての

数値と演算は倍精度（IEEE 754，binary64）である．求めた近似固有対 $(\lambda, \mathbf{v})$ の精度の評価には相対残差の大きさである $\Theta \equiv \|A\mathbf{v} - \lambda B\mathbf{v}\|_2 \,/\, \|\lambda B\mathbf{v}\|_2$ の値を用いた．

(2) 実験に用いた計算機環境

使用した計算機は東大情報基盤センター Oakbridge-CX の 1 ノードである．ノードは Dual CPU で CPU は Intel Xeon Platinum 8280(CascadeLake)(2.7GHz)．CPU のコア数は 28（ノードあたり 56）である．CPU の L3 キャッシュのサイズは 38.5MiB であり，拡張命令セットは AVX-512 である．ノードあたりの理論ピーク性能は 4.84TFLOPS(倍精度) で，主記憶 DDR4 メモリの容量は 192GiB，メモリバンド幅は 281.6GB/s である．プログラムは Fortran90 で記述して，コンパイラは `intelifort`，バージョン`"19.1.3.304 20200925"`を使用した．計算は 1 ノード上で，OpenMP 並列化でノード内のコア数に等しい 56 スレッドで実行した．正定値実対称な帯行列の分解と複数右辺の前進後退代入計算には intel MKL ライブラリに含まれる Lapack のルーチンを用いた．

(3) 固有値が分布の下端にある固有対を求めた例

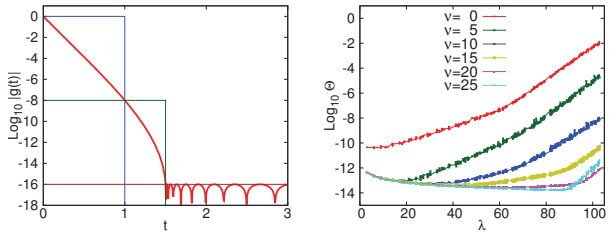
ここでは例題の一般固有値問題の固有対で固有値が固有値分布の下端の区間 $\lambda \in [3, 103]$ に含まれるものを求めた．そのような固有対の正しい数は 422 である．なお最初に生成する乱数ベクトルの数 m は 850 とした．

実験に用いた 5 通りのフィルタそれぞれについて，フィルタ名，使用した実シフトのレゾルベントの数 K，Chebyshev 多項式の次数 n，フィルタの伝達特性を与える 3 つの形状パラメタ μ，g_p，g_s を表 1 に示す．これらのフィルタの使用するレゾルベントはすべてシフトが実数である（これらのフィルタ F-1R，F-2R，F-3R-a，F-3R-b，F-4R は文献 [1] の中での対応する名称はそれぞれ Ex1-4，Ex2-1，Ex3-I-1，Ex3-I-5，Ex4-I-1 であり，具体的に構成するための数値も詳しく示してある）．

表–1 固有値が下端の固有対を求めるために用いたフィルタ

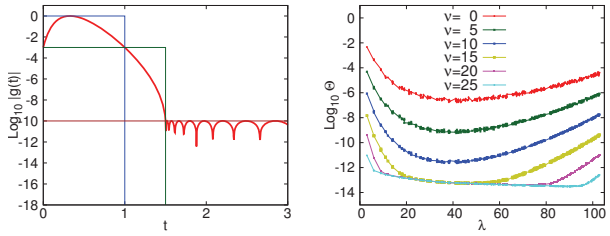
フィルタ名	K	n	μ	g_p	g_s
F-1R	1	25	1.5	1E-8	1E-16
F-2R	2	23	1.5	1E-3	1E-10
F-3R-a	3	20	1.5	1E-3	1.1E-10
F-3R-b	3	20	1.5	1E-3	1.2E-12
F-4R	4	23	1.5	1E-2	1E-12

5 通りのフィルタを前処理付きで適用した実験結果をそれぞれ，図 2 から図 6 までの 5 枚の図に示す．各左側の図は正規化座標 t を引数として Chebyshev 多項式の次数が n である本来のフィルタの伝達関数の大きさ $|g(t)|$ を対数でプロットしたものである．そうして各右側の図は，得られた各近似固有対について横軸に固有値をとり，縦軸には相対残差の大きさ Θ の対数値をとってプロットしたグラフを前処理用フィルタの次数 ν のそれぞれについて描いたものである．前処理用のフィルタの Chebyshev 多項式の次数 ν を 5 刻みで 25 まで変えて実験をした．ただし $\nu = 0$ は前処理をしない場合である．前処理用の次数 ν を増やすとそれに対応して各近似固有対の相対残差が減少する様子が見てとれる．



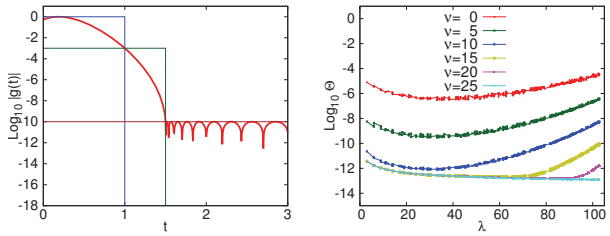
伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–2 フィルタ F-1R を前処理付きで適用した例



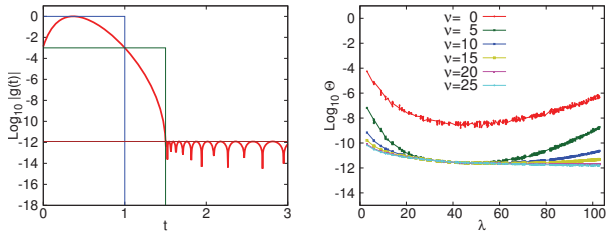
伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–3 フィルタ F-2R を前処理付きで適用した例



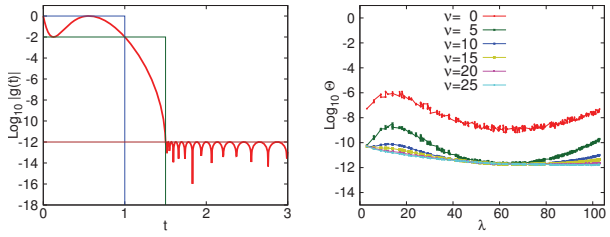
伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–4 フィルタ F-3R-a を前処理付きで適用した例



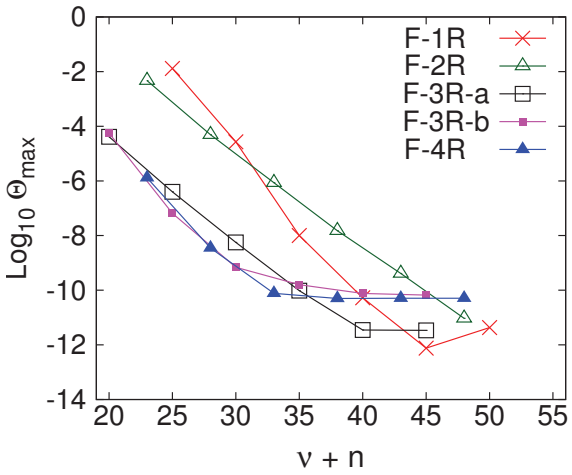
伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–5 フィルタ F-3R-b を前処理付きで適用した例

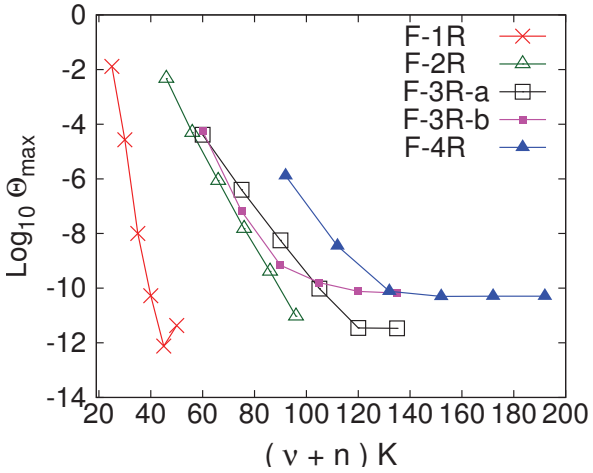


伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

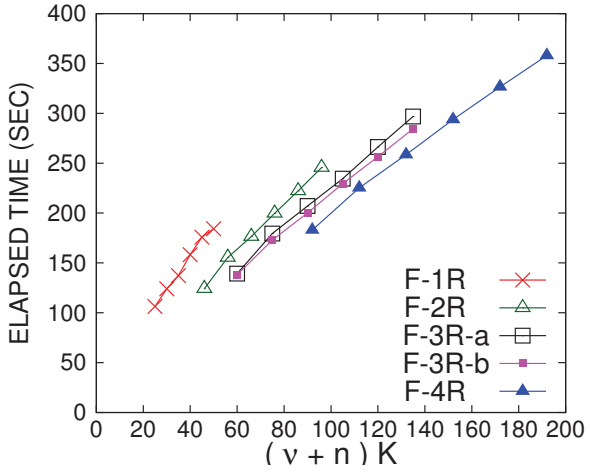
図–6 フィルタ F-4R を前処理付きで適用した例



図–7 次数の和に対する相対残差の大きさの最大値の対数



図–8 次数の和とレゾルベントの数の積に対する相対残差の大きさの最大値の対数



図–9 次数の和とレゾルベントの数の積に対する対角化完了までの経過時間（秒）

図 7 は，5 通りの各フィルタを用いた結果をそれぞれ，横軸には前処理と本来のフィルタの Chebyshev 多項式の次数の和 $\nu+n$ をとり，縦軸には求めた近似固有対すべてについての相対残差の大きさの最大値を対数でプロットしたものである．また図 8 は，5 通りの各フィルタを用いた結果をそれぞれ，横軸には多項式の次数の和にレゾルベントの数 K を乗じた値 $(\nu+n)K$ をとり，縦軸には求めた近似固有対全体について相対残差の大きさの最大値を対数でプロットしたものである．もしも K 個のレゾルベントを作用させる計算を完全に並列に処理できるのであれば，計算の主要部の経過時間は $\nu+n$ に比例するが，そうではなくてレゾルベントの作用を与えるための K 個の行列分解やレゾルベントの適用で複数右辺に対する行列分解の因子を用いた前進後退代入の計算を K 個のレゾルベントについて順次に行うのであれば，経過時間あるいは全体の計算の手間はほぼ $(\nu+n)K$ に比例することになる．どちらの場合も，フィルタの計算効率はグラフが下側にあるものほど良いことになる．そのため，K 個のレゾルベントについての処理を順次に行う場合は，図 8 にある 5 通りのフィルタのうちで最も計算効率が良いのはレゾルベント 1 つで構成された F-1R ということになる．

図 9 は 5 通りのフィルタのそれぞれについて，前処理の次数 ν を 0 から 25 まで 5 刻みで選んで，横軸には $(\nu+n)K$ の値を，縦軸には対角化が終了するまでの経過時間をとってプロットしたグラフを描いたものである．対角化が完了するまでの全体の経過時間には，たとえばランダムな m 個のベクトルの生成，m 個のベクトルの B-正規直交化，フィルタ処理を終えてから近似固有対を抽出するまでの処理などのレゾルベントの個数 K には依らないものが含まれるので，これらのグラフは図中で 1 つの直線には乗らずに，レゾルベントの数 K が多いフィルタほど対応するグラフが下側にきている．ベクトル m 個の組にフィルタを適用する計算の手間は m に比例するが，ベクトルの組の B-正規直交化の手間は m の 2 乗に比例するので，同じレゾルベントの組を使う場合，m が大きくなるほど計算中で B-正規直交化の占める割合が増える．Rayleigh-Ritz 法もその中で次数 m の行列を作る手間が m の 2 乗に比例し，また小さい m 次行列の完全対角化には m の 3 乗の定数倍の手間がかかるなど，レゾルベントを逐次に処理する場合には行列次数 N と半帯幅 w_h やベクトルの数 m が一定ならば経過時間は $(\nu+n)K$ に比例する，という単純な見積りは m が大きいければそれだけずれることになる．

(4) 固有値が分布の内側にある固有対を求めた例

こんどは例題の一般固有値問題の固有対で固有値が固有値分布の内側の区間 $\lambda \in [2000, 2020]$ にあるものを求める．そのような固有対の正しい数は 429 である．

フィルタで使用する K 個のレゾルベントはすべてシフトが虚数であり，複素対称性を利用することでシフトの虚部が正であるもの $K' = K/2$ 個だけを用いてフィルタの作用を実現する．実験に用いた 6 通りのフィルタそれぞれについて，本論文でのフィルタ名，使用するレゾルベントの数 $K' = K/2$，Chebyshev 多項式の次数 n，伝達関数の 3 つの形状パラメタ ξ，g_p，g_s を表 2 に掲げる（ξ の定義は，通過域 $\lambda \in [a, b]$ を標準区間 $t \in [-1, 1]$ に対応させる線形変換により λ に対応する正規化座標 t を定義するとき，阻止域の端の位置が $t = \pm\xi$ である）．

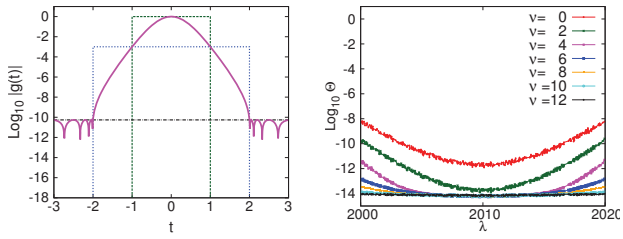
フィルタの特性が $\xi = 2.0$ の場合には，阻止域は $\lambda \in (-\infty, 1990] \cup [2030, +\infty)$ であり，最初に生成する乱数ベクトルの数 m を 900 とした．またフィルタの遮断

表–2 固有値が内側の固有対を求めるために用いたフィルタ

フィルタ名	K'	n	ξ	g_p	g_s
F-1C	1	12	2.0	1E-3	5.49E-11
F-2C	2	4	2.0	1E-3	1.92E-13
F-3C	3	3	2.0	1E-1	1.30E-13
F-1C-s	1	14	1.5	1E-4	1.72E-10
F-2C-s	2	6	1.5	1E-2	1.11E-12
F-3C-s	3	3	1.5	1E-1	7.96E-11

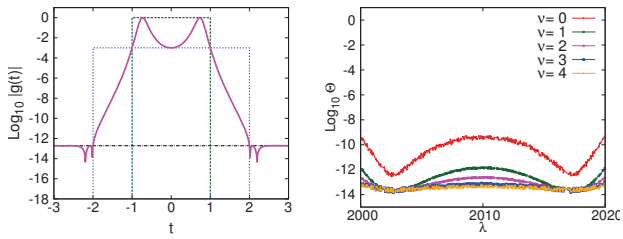
特性がより先鋭な $\xi = 1.5$ の場合には，阻止域は $\lambda \in (-\infty, 1990] \cup [2025, +\infty)$ であり，m は 700 とした．

フィルタを前処理付きで適用した 6 通りの実験の結果を $\xi = 2.0$ の場合は図 10 から図 12 の 3 枚の図に，$\xi = 1.5$ の場合は図 13 から図 15 の 3 枚の図にそれぞれ示す．それら 6 枚それぞれの左側は Chebyshev 多項式の次数が n である本来のフィルタの伝達関数の大きさを正規化座標 t を引数として表した $|g(t)|$ の対数プロットであり，右側は横軸には得られた各近似固有対の固有値をとり，縦軸には相対残差の大きさ Θ の対数値をとってプロットしたグラフを前処理用フィルタの次数 ν を変えた各場合について描いたものである．前処理用のフィルタに用いた Chebyshev 多項式の次数 ν は F-1C と F-1C-s の場合には（煩雑になるのを避けて）2 刻みでプロットしている．ただし $\nu = 0$ は前処理を行わない場合である．前処理用のフィルタの ν を増やすとそれにともなって各近似固有対の相対残差が減少していく様子が見てとれる．図 13 の右側で $\nu = 0$ の場合の赤い折れ線グラフに鋭い縦線が 4 箇所あるのは，相対残差の大きい偽の近似固有対が 4 つ混入したためである．



伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

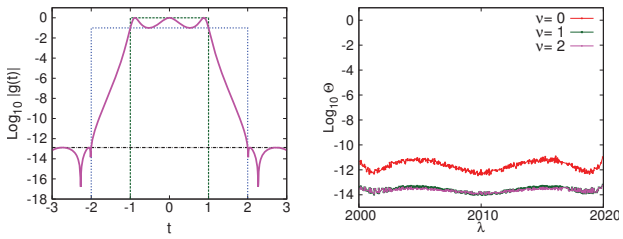
図–10 フィルタ F-1C を前処理付きで適用した例



伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

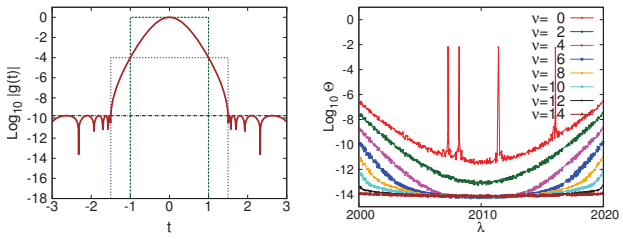
図–11 フィルタ F-2C を前処理付きで適用した例

図 16 は $\xi = 2.0$ の場合について，図 17 は $\xi = 1.5$ の場合について，それぞれ 3 通りのフィルタを用いた結果を，横軸には前処理用と本来のフィルタの Chebyshev 多項式の次数の和である $\nu + n$ をとり，縦軸には近似固



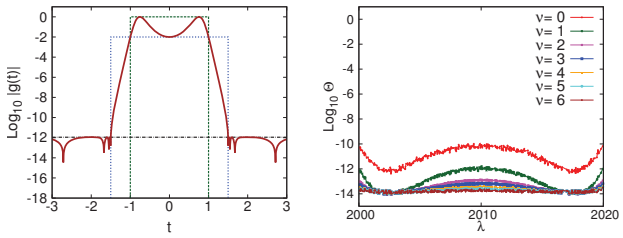
伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–12 フィルタ F-3C を前処理付きで適用した例



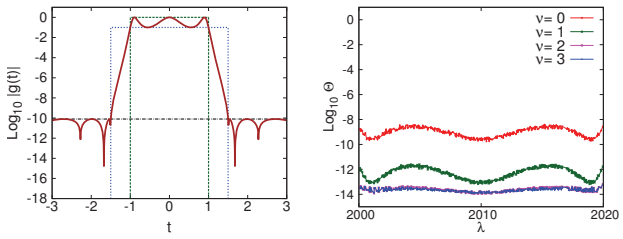
伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–13 フィルタ F-1C-s を前処理付きで適用した例



伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

図–14 フィルタ F-2C-s を前処理付きで適用した例

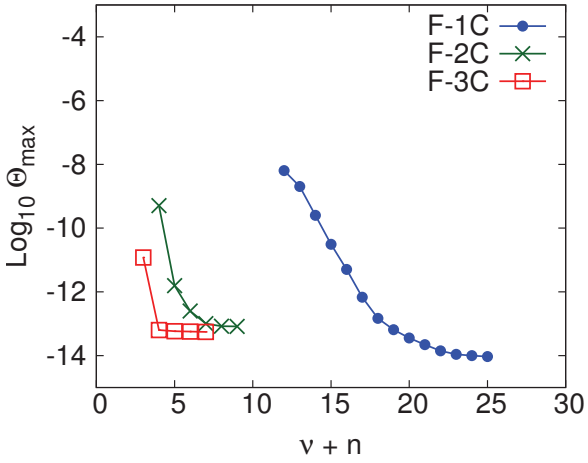


伝達関数の大きさ $|g(t)|$ の対数　相対残差の大きさ Θ の対数

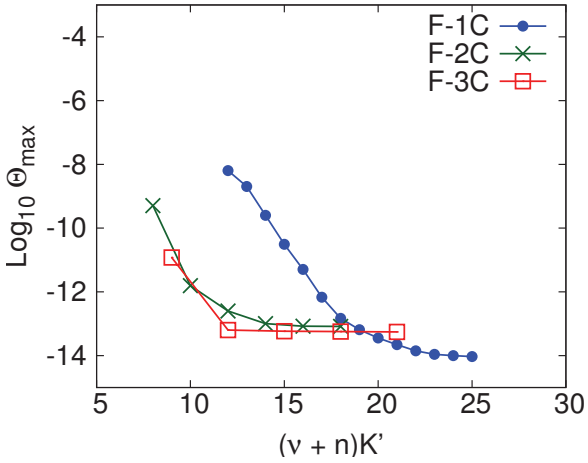
図–15 フィルタ F-3C-s を前処理付きで適用した例

有対全体についての相対残差の大きさの最大値 $\Theta_{\max}$ を対数でプロットして折れ線で描いたものである．同様に図 18 は $\xi = 2.0$ の場合について，図 19 は $\xi = 1.5$ の場合について，それぞれ 3 通りのフィルタを用いた結果を，横軸には多項式の次数の和と使用したレゾルベントの数の積である $(\nu + n)K'$ をとり，縦軸には求めた近似固有対全体についての相対残差の大きさの最大値 $\Theta_{\max}$ を対数でプロットして折れ線で描いたものである．

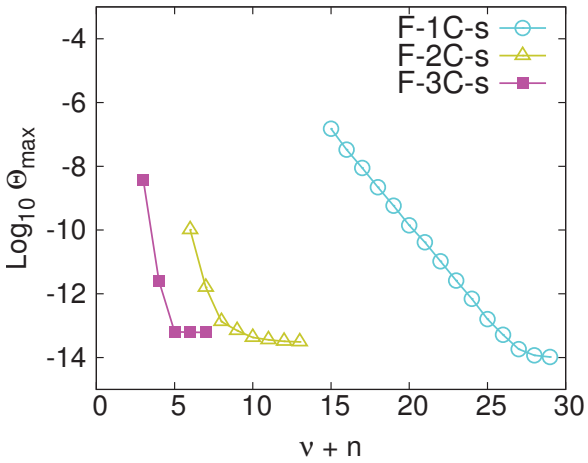
もしも使用するレゾルベント K' 個についてレゾルベントを準備するための行列分解やレゾルベントの作用を実現する行列分解因子を用いた複数右辺の前進後退代入の計算を完全に並行して行えるのならば，主要部であるフィルタ処理の経過時間は $\nu + n$ に比例することになる．あるいは K' 個のレゾルベントについて順次



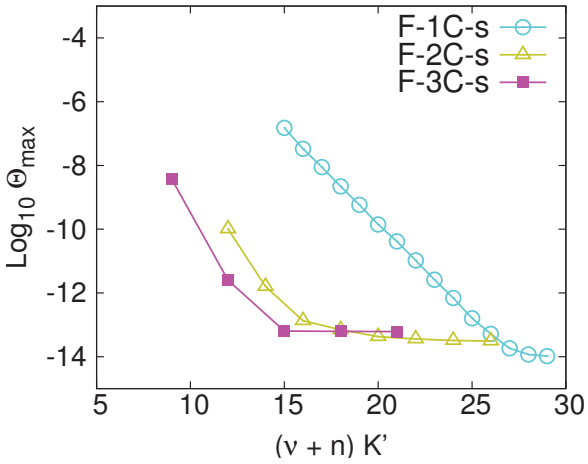
図–16 次数の和に対する相対残差の大きさの最大値の対数（$\xi = 2.0$ のフィルタ 3 通り）



図–18 次数の和とレゾルベントの数の積に対する相対残差の大きさの最大値の対数（$\xi = 2.0$ のフィルタ 3 通り）



図–17 次数の和に対する相対残差の大きさの最大値の対数（$\xi = 1.5$ のフィルタ 3 通り）



図–19 次数の和とレゾルベントの数の積に対する相対残差の大きさの最大値の対数（$\xi = 1.5$ のフィルタ 3 通り）

に計算を行うのであれば，主要部の経過時間あるいは計算の手間は $(\nu + n)K'$ に比例する．いずれの場合も，計算効率の良いフィルタほどグラフが下側にくるので，いまの場合には K' 個のレゾルベントのための処理を完全並行に行える場合と順次に行う場合のどちらも，計算効率が最も良いフィルタは $\xi = 2.0$ では F-3C であり，$\xi = 1.5$ では F-3C-s であり，レゾルベントを 3 つ用いたものであるという結果になった．固有値分布の一般的な位置に固有値がある固有対を求めるためにシフトが虚数のレゾルベントだけで構成された今回のフィルタ 3 通りによる比較では，レゾルベントの数 K' が 1，2，3 と増すほど計算効率が良いことがわかる．ただし行列分解因子を格納するための記憶量はレゾルベントの数 K' に比例するので，計算の制約が記憶量である場合には使用するレゾルベントの数を抑えなければならない．

6. まとめ

フィルタをレゾルベントの線形結合の Chebyshev 多項式とする場合には，その多項式の次数を下げたものを前処理用のフィルタとして利用できる．

乱数で生成したベクトルの組を B-正規直交化したものに対して本来のフィルタを直接適用せずに，まず先に前処理用のフィルタを適用してその結果を B-正規直交化したものを作り，それに対して本来のフィルタを適用すれば得られる近似固有対の精度を向上できる．近似固有対の精度は前処理用のフィルタの多項式の次数を増すほど良くなる．

前処理用と本来のフィルタの多項式の次数の和を一定とする場合には，本来のフィルタだけで行うよりも，前処理に手間の一部をかける方が結果の精度が向上して効率的であろう．また，近似固有対への精度要求を前処理を採用することにより満たせるのならば，性能の悪い単一のレゾルベントで構成されたフィルタでも，記憶容量が制約である場合には有用な計算手段になる．

参考文献

[1] 村上 弘：チェビシェフ多項式と極がすべて実数である低次有理関数の合成により実対称定値一般固有値問題の少数の下側固有対を解くためのフィルタの伝達関数を構成する方法，情報処理学会論文誌コンピューティングシステム *(ACS)*，Vol.15, No.3 (ACS78), pp.1–28, 2022.

計算工学講演会論文集 Vol. 28 (2023 年 5 月)
計算工学会

oneAPIを用いた様々なデバイス上でのステンシル計算の実装

Implementation of Stencil Calculation on Various Devices using oneAPI

佐久間大我 [1] 下川辺 隆史 [2] 大森拓郎 [3]
Taiga Sakuma and Takashi shimokawabe, Takuro Omori

[1]東京大学大学院 工学系研究科 電気系工学専攻 (〒 113-8658 東京都文京区弥生 2-11-16, E-mail: sakuma-taiga616@cspp.cc.u-tokyo.ac.jp)
[2]東京大学情報基盤センター (〒 277-0882 千葉県柏市柏の葉 6-2-3, E-mail: shimokawabe@cc.u-tokyo.ac.jp)
[3]東京大学大学院 工学系研究科 電気系工学専攻 (〒 113-8658 東京都文京区弥生 2-11-16)

In recent years, accelerators such as GPUs and FPGAs have attracted attention in the HPC field. Therefore, Intel oneAPI was introduced as a programming model for heterogeneous environments. Intel oneAPI has a SYCL-compliant compiler, DPC++, which is used to implement stencil applications that run on both CPUs and GPUs. We confirmed that the same code runs on both CPU and GPUd and that the performance is basically comparable to the legacy code.

Key Words : *oneAPI, SYCL, DPC++*

1. はじめに

近年、HPC 分野ではアクセラレータに注目が集まり、多くのスーパ-コンピュータが GPU 等のアクセラレータを搭載するようになっている。従来アクセラレータを利用するのであれば、そのデバイスに対して専用のプログラミングモデルを用いて専用のコードを書く必要があった。そこで、全てのデバイスに統一的なプログラミングモデルを提供することを目的として、Intel 社が発表したのが Intel oneAPI である。

Intel oneAPI は、複数アーキテクチャ向けのプログラミングモデルであり、対応するデバイスは、Intel 製の CPU や GPU は勿論のこと、NVIDIA 社や AMD 社などの他社製品にまで及んでいる。また DPC++(Data Parallel C++) という、既存の Intel C/C++ Complier をベースとして、SYCL が利用可能になった C++コンパイラが付属している [1]。

GPU 向けに NVIDIA 社が提供する CUDA や、Khronos Group が提供する、SYCL のベースとなったクロスプラットフォーム向けの API である OpenCL で書かれたコードの利用もサポートしており、先行研究では GPU-FPGA 混合環境を、oneAPI 上で既存のコードを統合し一つのアプリケーションとして実装している [2]。加えて、CUDA で書かれたコードを SYCL のコードに変換することも可能である [3]。

そこで Intel oneAPI を利用してステンシル計算と呼ばれる反復的な処理を要するアプリケーションを CPU、GPU 両方で動作させる SYCL コードを実装すると、実際に同じコードで動作する。すると、実際に同じコードで動作し、実行時に渡すパラメーターを変えるだけでも既存のコードに匹敵する性能を出せる程度の最適化が可能である。

本稿の構成としては、2 章でまず SYCL によるデバイスへのオフロード方法について紹介したのち、3 章で実際に SYCL を用いて開発した Intel oneAPI 環境において CPU と GPU 両方で動作する拡散方程式の計算のアプリケーションについて検討し、４章でまとめに入る。

2. SYCL

SYCL は、OpenCL を開発している Khronos Group により提供されている、ロイヤリティフリーのクロスアーキテクチャ向け抽象化レイヤーである [4]。異種デバイスを一つのアプリケーションで、最新の ISO C++を用いて記述される。本節では、SYCL コードの概要と、各デバイスへのオフロード方法、並列処理の方法などを紹介する。

(1) SYCL コード概要

まず SYCL のコードは、ホストデバイスで動作するホストコードとオフロード先のデバイスで動作するデバイスコードに分けられる。ホストコードで、オフロード先のデバイスを選択する Queue クラスを作成、Queue を submit することで、デバイスにオフロードする。

デバイスコードは、Listing 1 における、”h.parallel_for” のラムダ式で表される部分がデバイスコード SYCL で利用できる代表的なカーネルは、SIMD 的な並列処理をサポートする *parallel for kernel* (コード内 : parallel_for) と、パイプラインによる並列処理サポートする *single task* (コード内 : single_task) の二つである。

本稿の実装では GPU にオフロードすることを考え、parallel_for を採用した。

Listing 1 SYCL コード例

```
// キューの作成
queue q(d_selector, exception_handler);

buffer<int> a_buf(a.data());
buffer<int> b_buf(b.data());
buffer<int> sum_buf(sum.data());

// キューをデバイスに送り、データの移動
    やデバイスでの処理を書く。
q.submit([&](handler &h){
    accessor a_vec(a_buf, h);
    accessor b_vec(b_buf, h);
    accessor sum_vec(sum_buf, h);

    h.parallel_for(n,[=](nd_item<1> i){
      sum[i] = a[i] + b[i];
    });
});
```

(2) parallel_for

parallel_for では、ループ範囲の指定に 2 次元、3 次元の配列を指定できる。また、ループ範囲の指定には、nd_range クラスを使用することができる。nd_range クラスは、(global_range, local_range) で構成され、globa_range はループ範囲を表し、local_range は各スレッドに割り当てられる work-group(CUDA の thread-block に対応) の大きさを表す [5]。

local_range の全体や各次元のサイズは、デバイス毎に指定できる大きさに制限があり、当然最適な local_range は異なる。ただ、コンパイル時に決定する必要はなく、実行時に値を渡せばいいため、CPU と GPU で共有のコードでも対応可能である。

3. 拡散方程式シミュレーション

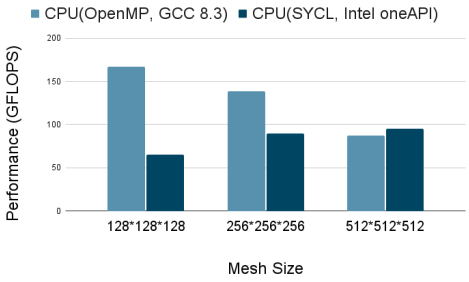
3 次元に離散化された拡散方程式は各点 (x_i,y_i,z_i) の濃度 $f_{i,j,k}$ として

$$f^{n+1}_{i,j,k} = (f^n_{i+1,j,k} + f^n_{i-1,j,k} + f^n_{i,j+1,k} + f^n_{i,j-1,k} + f^n_{i,j,k+1} + f^n_{i,j,k-1} + 4f^n_{i,j,k})/10 \quad (1)$$

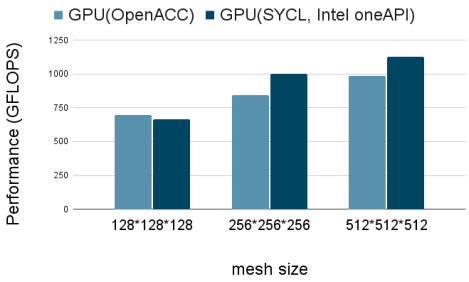
と表せる。ある時間ステップにおける、各点の f の更新は互いに独立であり、同時に処理が可能である。

性能評価

実行環境については、Wisteria-Aquarius 上で行った。CPU は Intel Xeon Platinum 8360Y、GPU は NVIDIA A100 が搭載されており、SYCL コードのコンパイラーは Intel® oneAPI DPC++/C++ Compiler 2023.0.0 である。各測定はグリッドサイズ (global_range) は (nx,ny,nz) に (128,128,128)、(256,256,256)、(512,512,512) の 3 つに対して行われる。先述した nd_range を利用しており、デバイス毎の制約は今回の環境の場合、CPU は local_range のサイズが 8192 までで、GPU は local_range の各次元のサイズが 64 までとなっている。



図–1 CPU 上での拡散方程式の実行性能比較



図–2 GPU 上での拡散方程式の実行性能比較

a) CPU の場合

local_range に関しては、global_range が (128,128,128) の場合は (128,64,1)、(256,256,256) の場合は (256,32,1)、(512,512,512) の場合は (512,8,1) とした。CPU の性能は、OpenMP のコードと比較した。比較に用いた OpenMP のコードは、GCC8.3 でコンパイルしている。Intel Compiler も OpenMP には対応しているが、動作を試したところ GCC の場合よりも性能が低下したため、今回は GCC を選択した。

図-1 が CPU の場合の性能測定結果である。メッシュサイズが小さい場合、つまり global_range が (128,128,128) のときには、SYCL では OpenMP の三分の一程度の性能にとどまった。ただ、メッシュサイズが大きくなるとその差は縮まり、global_range が (512,512,512) の場合には、SYCL が OpenMP を追い越している。

b) GPU の場合

local_range は、全てのメッシュサイズに対して (64,4,1) としている。GPU の性能は、OpenACC のコードと比較した。コンパイル環境は NVIDIA HPC SDK Version 22.7 である。

図-2 の GPU の場合の性能測定結果を見ると、(128,128,128) の場合には、SYCL の性能が OpenACC の性能にほぼ匹敵しており、それよりもメッシュサイズが大きくなると SYCL の性能が OpenACC の性能を追い越している。

4. まとめ

oneAPI 環境において SYCL で書かれたコードが CPU、GPU で動作し、また実行時に local_range を与えることで、基本的には既存のコードと遜色のない性

能を出せることが確認できた。ただ、CPU において、メッシュサイズが小さい場合に非常に性能が下がってしまう点、また CPU において実行するたびに性能が数十 GFLOPS の単位で変動する点が気になった。まだ、SYCL のコードについて最適化の余地があるためそれを探ることと、より複雑なステンシルアプリケーションを SYCL で開発することが今後の課題である。

謝辞: 本研究の一部は科学研究費補助金 課題番号 20K21787,20H00580, および, 学際大規模情報基盤共同利用・共同研究拠点, および, 革新的ハイパフォーマンス・コンピューティング・インフラ課題番号 jh220036,jh230037 から支援を頂いた. 記して謝意を表す.

参考文献

[1] Intel. Data parallel c++ language. https://www.intel.com/content/www/us/en/developer/tools/oneapi/data-parallel-c-plus-plus.html#gs.ugtz08. 2023.

[2] 柏野隆太, 小林諒平, 藤田典久, 朴泰祐ほか. oneapi を用いた gpu・fpga 混載ノードにおける宇宙物理シミュレーションコード argot の実装. 研究報告ハイパフォーマンスコンピューティング (HPC), Vol. 2022, No. 12, pp. 1–9, 2022.

[3] Steffen Christgau and Thomas Steinke. Porting a legacy cuda stencil code to oneapi. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 359–367, 2020.

[4] Khronos Group. Sycl website. https://www.khronos.org/api/index_2017/sycl, 2023.

[5] Kamalavasan Kamalakkannan, Gihan R Mudalige, Istvan Z Reguly, and Suhaib A Fahmy. Fpga acceleration of structured-mesh-based explicit and implicit numerical solvers using sycl. In *International Workshop on OpenCL*, pp. 1–11, 2022.

OS27 先進並列シミュレーション

[C-11] OS27 先進並列シミュレーション (2)

座長:塩谷 隆二(東洋大学)

Fri. Jun 2, 2023 10:30 AM - 11:15 AM Room C (2F Conference Room 201A)

[C-11-01] 並列有限要素構造解析コード ADVENTURE_Solidの富岳向けチューニング

*宮村 倫司[1]、小池 邦昭[2]、吉村 忍[3] （1. 日本大学、2. 株式会社先端力学シミュレーション研究所、3. 東京大学）

10:30 AM - 10:45 AM

[C-11-02] Accelerating lattice Boltzmann method simulation with GPU computation using C++ standard language parallelism

*袁 子恒[1]、下川辺 隆史[1] （1. 東京大学）

10:45 AM - 11:00 AM

[C-11-03] 階層型領域分割法による電磁場解析での四倍精度演算 CG法， COCG法

*杉本 振一郎[1] （1. 八戸工業大学）

11:00 AM - 11:15 AM

計算工学講演会論文集　Vol.28 (2023年5月)　　計算工学会

並列有限要素構造解析コードADVENTURE_Solidの富岳向けチューニング

Tuning Up Parallel Finite Element Structural Analysis Code ADVENTURE_Solid for Supercomputer Fugaku

宮村倫司[1)]，小池邦昭[2)]，吉村忍[3)]
Tomoshi Miyamura, Kuniaki Koike, and Shinobu Yoshimura

1) 日本大学工学部情報工学科（〒963-8642 郡山市田村町徳定字中河原１番地)
2)（株）先端力学シミュレーション研究所（〒112-0002文京区小石川5-5-5 プライム茗荷谷ビル5F）
3) 東京大学大学院工学系研究科システム創成学専攻（〒113-8656 文京区本郷7-3-1）

ADVENTURE_Solid is a parallel finite element structural analysis code that utilizes the hierarchical domain decomposition method. The code incorporates the balancing domain decomposition (BDD) method as a parallel linear solver. Supercomputer Fugaku is a massively parallel supercomputer consisting of approximately one hundred and sixty thousand computation nodes. In this study, an enhanced parallel implementation of the BDD method in ADVENTURE_Solid is presented for optimal performance on Supercomputer Fugaku.

Key Words : *Parallel finite element analysis, ADVENTURE_Solid, Domain decomposition, Supercomputer Fugaku, Sparse direct solver*

1.　はじめに

ADVENTURE_Solidは階層型領域分割法をベースとした陰的な有限要素法に基づく並列構造解析コードであり，オープンソースのCAEシステムであるADVENTUREシステムに含まれる解析モジュールのひとつである[1][2]．一方，スーパーコンピュータ「富岳」[3]は，48コアのCPUを持つ約16万個の計算ノードを6次元メッシュ／トーラスネット―ワークであるTofuD インターコネクトで結合した構成のスーパーコンピュータ（スパコン）である．多数の計算ノードを持つ富岳では，数千から数万個の計算ノードを用いた計算ジョブを日常的に実行することが可能であるため，その計算資源を活用できるようなソフトウェアの開発が求められる．

本研究では，最初に従来のADVENTURE_Solid（ver. 2）を富岳で実行した場合の問題点について分析する．次に，数万個の計算ノードを用いても十分な並列化効率が得られるような実装方法を提案する．それに基づいて実装したADVENTURE_Solid（ver. 3）の計算性能を「富岳」上で測定し，その結果を報告する．

2.　BDD法

Mandel[4]により提案されたバランシング領域分割（Balancing Domain Decomposition; BDD）法は，部分構造型領域分割法に基づく，線形問題に対する前処理付きの反復解法である．ここではその定式化の概要を示す．BDD法の計算過程では元の線形問題よりも小規模ないくつかの線形問題を解く必要がある．実装方法を検討するためにこれらを整理する．

(1) BDD法の定式化

有限要素法に基づき離散化された静的線形構造解析における釣り合い式は，次のような連立一次方程式となる．

$$\mathbf{K}\mathbf{u} = \mathbf{p} \tag{1}$$

ここに，$\mathbf{K}$ は剛性行列，$\mathbf{u}$ は節点変位ベクトル，$\mathbf{p}$ は節点荷重ベクトルである．

ここでは，領域分割法の中でも部分領域内部の自由度を消去する部分構造型領域分割法を用いる．まず，解析領域をN個のオーバーラップのない部分領域に分割する．部分領域 k に対する剛性行列を $\mathbf{K}^{(k)}$，節点変位を $\mathbf{u}^{(k)}$，節点荷重を $\mathbf{p}^{(k)}$ としたとき，これらの行列，ベクトルを次式のように部分領域内部の自由度と部分領域間境界の自由度に分割する．

$$\mathbf{K}^{(k)} = \begin{bmatrix} \mathbf{K}_{\mathrm{II}}^{(k)} & \mathbf{K}_{\mathrm{IB}}^{(k)} \\ \mathbf{K}_{\mathrm{BI}}^{(k)} & \mathbf{K}_{\mathrm{BB}}^{(k)} \end{bmatrix}, \quad \mathbf{u}^{(k)} = \begin{bmatrix} \mathbf{u}_{\mathrm{I}}^{(k)} \\ \mathbf{u}_{\mathrm{B}}^{(k)} \end{bmatrix}, \quad \mathbf{p}^{(k)} = \begin{bmatrix} \mathbf{p}_{\mathrm{I}}^{(k)} \\ \mathbf{p}_{\mathrm{B}}^{(k)} \end{bmatrix} \tag{2}$$

ここに，添字Iは部分領域内部の自由度を，添字Bは部分領域間境界上の自由度を表す．領域間境界全体の自由度数を n_{B} とする．また，部分領域 k に関係する領域間境界自由度の自由度数を $n_{\mathrm{B}}^{(k)}$ とする．

部分領域の内部の自由度に関する釣り合いは次式のように各部分領域において独立に成立する．

$$\mathbf{K}_{\mathrm{II}}^{(k)}\mathbf{u}_{\mathrm{I}}^{(k)}+\mathbf{K}_{\mathrm{IB}}^{(k)}\mathbf{u}_{\mathrm{B}}^{(k)}=\mathbf{p}_{\mathrm{I}}^{(k)} \tag{3}$$

一方，部分領域間境界自由度の釣り合いは次式で表される．

$$\sum_{k=1}^{N}\mathbf{N}_{\mathrm{B}}^{(k)}\mathbf{K}_{\mathrm{BI}}^{(k)}\mathbf{u}_{\mathrm{I}}^{(k)}+\sum_{k=1}^{N}\mathbf{N}_{\mathrm{B}}^{(k)}\mathbf{K}_{\mathrm{BB}}^{(k)}\mathbf{u}_{\mathrm{B}}^{(k)}=\sum_{k=1}^{N}\mathbf{N}_{\mathrm{B}}^{(k)}\mathbf{p}_{\mathrm{B}}^{(k)} \tag{4}$$

ここに，$\mathbf{N}_{\mathrm{B}}^{(k)}$ は部分領域 k に関する領域間境界自由度を全ての部分領域に対する領域間境界自由度に変換するブーリアン行列である．式(3)は次のように内部節点変位について解くことができる．

$$\mathbf{u}_{\mathrm{I}}^{(k)}=\left(\mathbf{K}_{\mathrm{II}}^{(k)}\right)^{-1}\left(\mathbf{p}_{\mathrm{I}}^{(k)}-\mathbf{K}_{\mathrm{IB}}^{(k)}\mathbf{u}_{\mathrm{B}}^{(k)}\right) \tag{5}$$

式(5)を式(4)に代入すると次式のようにまとめられる．

$$\mathbf{S}\bar{\mathbf{u}}_{\mathrm{B}}=\bar{\mathbf{p}}_{\mathrm{B}} \tag{6}$$

ここに，$\mathbf{S}$ はSchur complement，$\bar{\mathbf{u}}_{\mathrm{B}}$ は部分領域間境界上の全ての節点に対する節点変位ベクトル，$\bar{\mathbf{p}}_{\mathrm{B}}$ は静的縮約された部分領域間境界上の節点荷重ベクトルである．行列 $\mathbf{S}$ は $n_{\mathrm{B}}\times n_{\mathrm{B}}$ 型行列となる．部分領域 k に対するローカルSchur complementを $\mathbf{S}^{(k)}$ とすると，全体のSchur complementである $\mathbf{S}$ は次式となる．

$$\mathbf{S}=\sum_{k=1}^{N}\mathbf{N}_{\mathrm{B}}^{(k)}\mathbf{S}^{(k)}\mathbf{N}_{\mathrm{B}}^{(k)\mathrm{T}} \tag{7}$$

ここに，

$$\mathbf{S}^{(k)}=\mathbf{K}_{\mathrm{BB}}^{(k)}-\mathbf{K}_{\mathrm{BI}}^{(k)}\mathbf{K}_{\mathrm{II}}^{(k)-1}\mathbf{K}_{\mathrm{IB}}^{(k)} \tag{8}$$

である．

式(6)は前処理付き共役勾配法で解くものとする．Mandel[4]が提案したバランシング領域分割法（Balancing Domain Decomposition Method, BDD法）はこの問題に対する効果的な前処理手法であり，ADVENTURE_Solidにも実装されている．その前処理行列は次式となる．

$$\begin{aligned}\mathbf{M}_{\mathrm{BDD}}^{-1}=&\left(\mathbf{I}-\mathbf{R}\mathbf{K}_{\mathrm{C}}^{-1}\mathbf{R}^{\mathrm{T}}\mathbf{S}\right)\mathbf{M}_{\mathrm{NN}}^{-1}\left(\mathbf{I}-\mathbf{S}\mathbf{R}\mathbf{K}_{\mathrm{C}}^{-1}\mathbf{R}^{\mathrm{T}}\right)\\&+\mathbf{R}\mathbf{K}_{\mathrm{C}}^{-1}\mathbf{R}^{\mathrm{T}}\end{aligned} \tag{9}$$

ここに，$\mathbf{M}_{\mathrm{NN}}^{-1}$ はNeumann-Neumann前処理のための前処理行列である．$\mathbf{R}$ と $\mathbf{R}^{\mathrm{T}}$ はそれぞれマルチグリッド法prolongation行列とrestriction行列に相当する．固体，構造解析では行列 $\mathbf{R}$ は $n_{\mathrm{B}}\times 6N$ 型行列となる．$\mathbf{K}_{\mathrm{C}}$ は次式によって計算される．

$$\mathbf{K}_{\mathrm{C}}=\mathbf{R}^{\mathrm{T}}\mathbf{S}\mathbf{R} \tag{10}$$

行列 $\mathbf{K}_{\mathrm{C}}$ はCoarse grid（コース）行列と呼ばれ，剛性行列 $\mathbf{K}$ を部分領域の剛体運動を利用して低次元で近似した行列となる．$\mathbf{K}_{\mathrm{C}}$ を係数行列とした線形問題をコース（グリッド）問題と呼ぶ．コース問題を解いた結果を用いてCG法の収束性を高める処理は，前処理の一種でありコースグリッド修正と呼ばれる．初期残差に対してコースグリッド修正をするとそれ以降はコース空間の成分が取り除かれるため，式(9)の第1項の最初のコースグリッド修正と最後の項を省略できる．このとき，前処理行列は，

$$\mathbf{M}_{\mathrm{BDD}}^{-1}=\left(\mathbf{I}-\mathbf{R}\mathbf{K}_{\mathrm{C}}^{-1}\mathbf{R}^{\mathrm{T}}\mathbf{S}\right)\mathbf{M}_{\mathrm{NN}}^{-1} \tag{11}$$

となる．

Neumann-Neumann前処理を表す $\mathbf{M}_{\mathrm{NN}}^{-1}$ は，

$$\mathbf{M}_{\mathrm{NN}}^{-1}=\sum_{k=1}^{N}\mathbf{N}_{\mathrm{B}}^{(k)}\mathbf{D}^{(k)}\left(\mathbf{S}^{(k)}\right)^{+}\mathbf{D}^{(k)\mathrm{T}}\mathbf{N}_{\mathrm{B}}^{(k)\mathrm{T}} \tag{12}$$

と表される．ここに，$\mathbf{D}^{(k)}$ は領域間境界自由度を共有する部分領域の数の逆数を対角成分とした対角行列であり，重み行列と呼ばれる．$\left(\mathbf{S}^{(k)}\right)^{+}$ はローカルSchur complementの一般逆行列を表す．荻野[2]は一般逆行列を使う代わりに次式のようにペナルティ項を加えて正則化し，逆行列を使う方法を提案した．

$$\left(\mathbf{S}^{(k)}\right)^{+}\approx\left(\mathbf{S}^{(k)}+\max\left(diag\left(\mathbf{K}^{(k)}\right)\right)\mathbf{I}\right)^{-1} \tag{13}$$

(2) 解くべき線形問題の整理

BDD法の計算過程で解く必要がある線形問題を整理する．以下では，一般的なベクトルを $\mathbf{a}$，$\mathbf{b}$ あるいはそれらに添字を加えた記号で表す．

CG法の計算過程における残差ベクトルの計算，式(10)のコース行列の作成，式(11)のNeumann-Neumann前処理の後に $\mathbf{S}$ との積の計算がある．そのために，次式のようなローカルSchur complement $\mathbf{S}^{(k)}$ とベクトルの積の計算が必要である．

$$\mathbf{b}_{\mathrm{B}}^{(k)}=\mathbf{S}^{(k)}\mathbf{a}_{\mathrm{B}}^{(k)} \tag{14}$$

$\mathbf{S}^{(k)}$ を陽に作る代わりに次式のように本来のディリクレ境界に加えて部分領域境界自由度もディリクレ境界として与えた問題（ディリクレ問題）を解く．ディリクレ境界の処理は，ディリクレ境界自由度を除く方法ではなく，次式のようにその自由度に関する係数行列の対角項に1を入れる方法で処理する．

$$\begin{bmatrix}\mathbf{K}_{\mathrm{II}}^{(k)} & \mathbf{O}\\ \mathbf{O} & \mathbf{I}_{\mathrm{BB}}\end{bmatrix}\begin{bmatrix}\mathbf{u}_{\mathrm{I}}^{(k)}\\ \mathbf{u}_{\mathrm{B}}^{(k)}\end{bmatrix}=\begin{bmatrix}-\mathbf{K}_{\mathrm{IB}}^{(k)}\mathbf{b}_{\mathrm{B}}^{(k)}\\ \mathbf{a}_{\mathrm{B}}^{(k)}\end{bmatrix} \tag{15}$$

ここに，$\mathbf{I}_{\mathrm{BB}}$ は単位行列を表す．左辺の係数行列の逆行列を陽に求めるのではなく，線形ソルバーを使う．次に，$\mathbf{b}_{\mathrm{B}}^{(k)}$ を次のように求める．

$$\mathbf{b}_{\mathrm{B}}^{(k)}=\begin{bmatrix}\mathbf{K}_{\mathrm{BI}}^{(k)} & \mathbf{K}_{\mathrm{BB}}^{(k)}\end{bmatrix}\begin{bmatrix}\mathbf{u}_{\mathrm{I}}^{(k)}\\ \mathbf{u}_{\mathrm{B}}^{(k)}\end{bmatrix} \tag{16}$$

一方，Neumann-Neumann前処理では，次式のように式

(13)の一般逆行列を近似した行列とベクトルの積の計算が必要である．

$$\mathbf{b}_{\mathrm{B}}^{(k)} = \left(\mathbf{S}^{(k)} + \max\left(diag\left(\mathbf{K}^{(k)}\right)\right)\mathbf{I}\right)^{-1}\mathbf{a}_{\mathrm{B}}^{(k)} \qquad (17)$$

ここでも $\mathbf{S}^{(k)}$ を陽に作る代わりに次の方程式（ノイマン問題）を線形ソルバーにより解く．

$$\begin{bmatrix} \mathbf{K}_{\mathrm{II}}^{(k)} & \mathbf{K}_{\mathrm{IB}}^{(k)} \\ \mathbf{K}_{\mathrm{BI}}^{(k)} & \mathbf{K}_{\mathrm{BB}}^{(k)} + \max\left(\mathrm{diag}\left(\mathbf{K}^{(k)}\right)\right)\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{a}_{\mathrm{I}}^{(k)} \\ \mathbf{a}_{\mathrm{B}}^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_{\mathrm{B}}^{(k)} \end{bmatrix} \qquad (18)$$

最後に，コースグリッド修正に含まれる $\mathbf{K}_{\mathrm{C}}^{-1}$ とベクトルの積についても，逆行列を陽に作る代わりに線形ソルバーを用いて計算する．すなわち，

$$\mathbf{b} = \mathbf{K}_{\mathrm{C}}^{-1}\mathbf{a} \qquad (19)$$

を求めるために，

$$\mathbf{K}_{\mathrm{C}}\mathbf{b} = \mathbf{a} \qquad (20)$$

を線形ソルバーにより解く．

以上のように，BDD法の実装では式(15)，(18)，(20)の求解の実装方法がポイントとなる．式(15)，(18)は部分領域単位の局所的な計算であるため，ローカル問題と呼ぶ．式(20)の計算はコース問題と呼ぶ．

3. ADVENTURE_Solid におけるバランシング領域分割法の従来の実装方法

(1) 階層型領域分割法

ADVENTURE_Solid は Yagawa and Shioya[5]が提案した階層型領域分割法をベースとして並列実装をしている．図-1 のようにメッシュはまず Part と呼ばれる領域に分割され，各 Part がさらに部分領域に分割される．階層型領域分割はドメインデコンポーザーADVENTURE_Metis で行う．その中で，Part への分割には ParMetis[6]が，Part の部分領域への分割には Metis[7]が使われている．

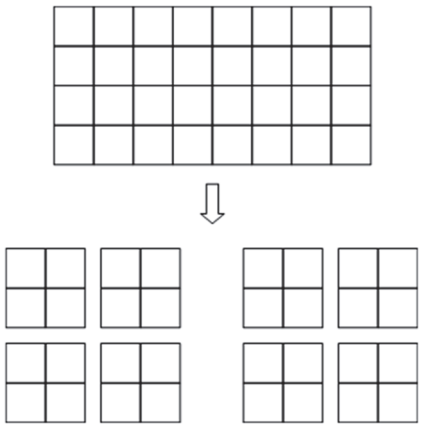


図-1 階層型領域分割

(2) 従来の実装方法

非公開のADVENTURE_Solid FS版では式(15)，(18)のローカル問題は直接法の一種であるスカイライン法のシリアル処理版で解いている．一方，式(20)のコース問題はスパース直接法ソルバーMUMPS[8]で解いている．ADVENTURE_SolidはMPIにより並列化されており，1個のPartを1個のMPIプロセスに割り当てている．マルチコアCPUを使う場合には，1個のMPIプロセスの中でOpenMPによるスレッド並列化を行っており，1個のPartの中にある複数の部分領域に対する式(15)，(18)のローカル問題をコア数と同数のスレッドに割り当てることで並列に計算する．この並列化では理想的なスケーラビリティが得られている．一方，コース問題に使うMUMPSもMPI並列化されているが，超並列計算機ではPart数と同数のMPIプロセス数は非常に大きいため，Partの情報をMUMPSに適した数のMPIプロセスに集約し，少ないMPIプロセス数で並列計算を行っている．また，MUMPSで使われるBLASをスレッド並列版とすることで複数のコアを使ったマルチスレッド並列化ができる．

(3) 従来版による原子力発電所モデルの解析

富岳に実装したADVENTURE_Solid FS版の従来版により，四面体二次要素による福島第1原子力発電所1号機モデルの静的解析を富岳により行う．このモデルの自由度数は1,497,322,665（約15億）である．なお，富岳の1個の計算ノードは4個のCMGと呼ばれる単位により構成され，48個のコアとメインメモリの関係は均質ではない．ここでは1個のCMGに1個のMPIプロセスを割り当てて計算する．したがって，1個のMPIプロセスには12個のコアが割り当てられる．

表-1に5種類の領域分割のケースを示す．ここではCase AとBに対して従来版により解析を行った結果を示す．表-2にそれぞれのケースにおけるコース問題の規模とローカル問題の規模を示す．コース問題の規模は全部分領域数×6（剛体モードの数）となる．ローカル問題の規模は，全自由度数/全部分領域数で概算したものである．実際には部分領域間境界上の自由度が重複することや，式(15)のディリクレ問題よりも式(18)のノイマン問題の方が係数行列の非零成分少ないという差異はあるものの，計算のオーダーの見積りにはなっている．富岳上の従来版により計算性能に対するパラメータスタディをした結果，従来版ではCase Aの領域分割によって最も速い計算速度が得られている．

図-2にCG法の1反復ステップ（1CG）の計算におけるコース問題の計算時間とその他の計算時間の割合を示す．Case Aのローカル問題の数はCase Bの10倍以上で，ローカル問題の規模は1/10以下となっている．ローカル問題の求解に用いているスカイライン法は自由度に対してスケーラブルな解法ではないため，全てのローカル問題に対する計算時間はCase Aの方が圧倒的に短い．しかし，コース問題の規模が大きいため，MUMPSによるコース問題の求

解の時間が全体の半分近くを占めている．MUMPSに割り当てるMPIプロセス数は96～192程度であり，全体の6144個のMPIプロセスの中のほんの一部に過ぎないため，MUMPSの計算の間はほとんどのMPIプロセスは何もしていないことになる．一方，Case Bでは1個のPartの部分領域数を1個に減らすことでコース問題の規模を小さくしている．コース問題とローカル問題の規模は同程度である．ただし，コース問題はMUMPSで，ローカル問題はスカイラインソルバーで解いているため，計算時間の大半はローカル問題の求解時間になっている．全体の計算速度はCase Aよりも大幅に遅く，収束まで計算していない．

プロセスとスレッドの割り当て，および，1CGおよび全体の計算時間とCG法の反復回数は後の表-3, 4にそれぞれ改良版の情報と共に示す．CG法の収束判定では，後の改良版による解析も含めて相対残差の閾値を1.0×10^{-3}としている．

4. ローカル問題にもMUMPSを使った実装

(1) 実装

3(3)節の数値実験の結果から，部分領域数を調整することで，コース問題とローカル問題の規模を同程度にすることができ，多くの計算ノードを有効活用できる可能があることがわかった．しかし，その結果，ローカル問題の規模が大きくなり，スカイライン法を使うと計算時間が長くなる．そこで，ローカル問題にもMUMPSを使えるように解析コードを改良する．

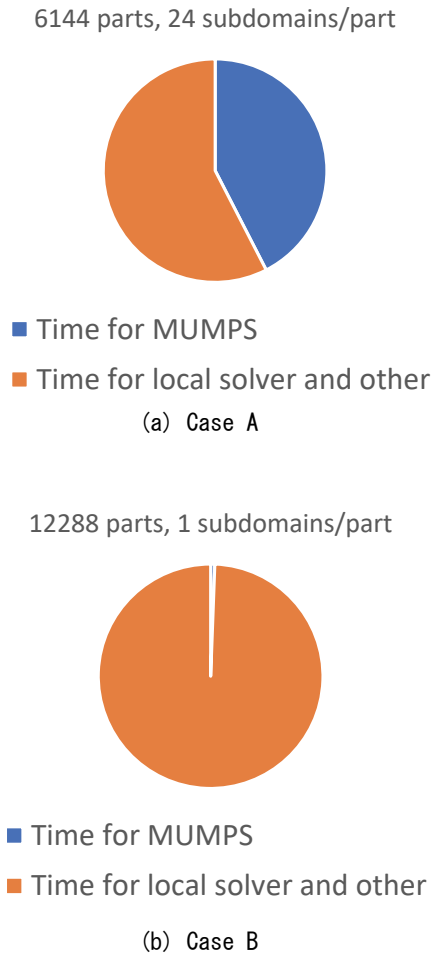


図-2 1CG の計算におけるコース問題の計算時間とその他の計算時間の割合（従来版による）

表-1 領域分割

ケース	Part数	1Partあたりの部分領域数	全部分領域数
Case A	6,144	24	147,456
Case B	12,288	1	12,288
Case C	6144	4	24576
Case D	12,288	2	24576
Case E	24000	1	24000

表-2 コース問題とローカル問題の規模

ケース	コース問題の規模	ローカル問題の規模
Case A	約88万自由度	約1万自由度
Case B	約7.3万自由度	約12.2万自由度
Case C	約15万自由度	約6.1万自由度
Case D	約15万自由度	約6.1万自由度
Case E	約14万自由度	約6.3万自由度

1個の部分領域に関するローカル問題をMPIにより分散並列化されているMUMPSで解く場合，1個の部分領域に複数のMPIプロセス（以下，プロセス）を割り当てる必要がある．従来のADVENTURE_Solidでは，図-3のように部分領域を管理するPartに1個のプロセスを割り当てていたため，プロセスの割り当て方法を見直す必要がある．

MPIでは集団的操作の対象となるプロセスの集まりをコミュニケータとして定義する．図-4に改良したADVENTURE_Solid（改良版）におけるMPIのコミュニケータの配置を示す．従来の実装ではPart数と同数のプロセスで構成されるMaster Groupのみであったのに対し，今回の実装ではWorker Groupを加える．MPI_COMM_WORLDは全プロセスを表すコミュニケータである．PLUコースグリッドMUMPSコミュニケータは，コースグリッド問題の求解に使うMUMPS用である．MPC MUMPSコミュニケータは，MPCを含む問題において拘束条件に関する射影を行うMUMPS用のコミュニケータである．このコミュニケータは複数個配置することもある．以上のコミュニケータはMaster Groupの中に置く．MUMPSローカルコミュニケータは，ローカル問題の求解に使うMUMPS用であり，Partの数と同じ数になるように生成する．コミュニケータ

に2個以上のプロセスが含まれる場合には，Master Groupのプロセスだけでは足らないので, Worker Groupのプロセスを使う．1個のPartに複数の部分領域がある場合には，MUMPSのオブジェクトは部分領域毎に生成するが，計算は順番に行うものとしてコミュニケータは同じもの（各Partに一つ配置）を使う．

(2) 計算性能評価

表-1のCase C, D, Eについて，改良版で解析を行う．表-3に各ケースのプロセスとスレッドの割り当てを，表-4に計算時間とCG法の反復回数を示す．改良版ではローカル問題に対して多くのMPIプロセスが必要なこと，また，スレッド並列が部分領域毎の処理に対するものではなく，スレッド並列版のBLASに対するものであるため，並列性能が低いことから, 1個の計算ノードに8個のMPIプロセスを割り当てている．Case C, D, Eの全部分領域数はほぼ同じであり，違いは1個のPartあたりの部分領域数である．

Case CとDについて，Case C のCG法の反復数がCase Dと同じであると仮定した上で並列化効率を求めると，66.4%となり良好な結果となっている．1CGループの計算時間で比較すると，Case Eではさらに高速になっている．しかし，Case Eでは収束までのCG法の反復回数が多く，全計算時間は従来版の最速値であるCase Aの計算時間よりも遅くなっている．BDD法の性質により，部分領域数が少なくなるとコースグリッド修正の性能が低下して反復回数が増大する．それだけでなく，1個のPartあたりの部分領域数が1個の場合，収束性がより低下する現象がみられる．部分領域数を少し変えたケースでも同様の傾向となった．Partあたりの部分領域数が1の場合，領域分割はParMetisのみで行うことになるので，ParMetisによる領域分割の品質が，Metisによるものと比べてやや劣っている可能性が考えられる．

5. おわりに

本研究では，階層型領域分割法に基づく並列有限要素構造解析コードADVENTURE_Solid (ver. 2) を富岳で実行した場合の問題点について分析した．次に，数万個の計算ノードを用いても十分な並列化効率が得られるように，ローカル問題をスパース直接法ソルバーMUMPSで解くような実装を行い（ADVENTURE_Solid（ver. 3））, その計算性能を評価した．

謝辞：本研究は，文部科学省「富岳」成果創出加速プログラム「スーパーシミュレーションとAIを連携活用した実機クリーンエネルギーシステムのデジタルツインの構築と活用」（JPMXP1020200303）の一環として実施されたものです．また, 本研究の一部は, スーパーコンピュータ「富岳」の計算資源の提供を受け，実施しました（課題番号：hp210175, hp220169）．

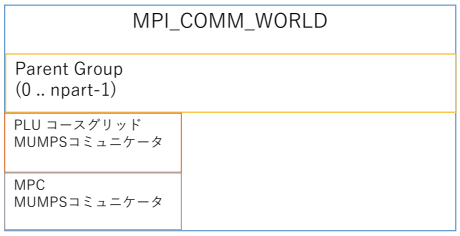


図-3 従来版におけるコミュニケータの配置

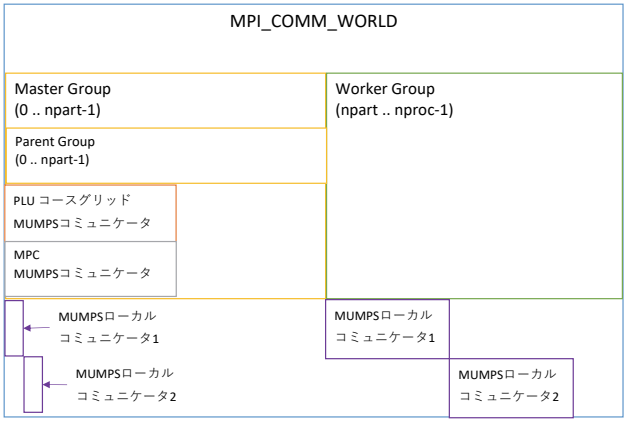


図-4 改良版におけるコミュニケータの配置

表-3 MPIプロセスとスレッドの割り当て（Case A, Bは従来版，Case C, D, Eは改良版）

ケース	ノード数	MPIプロセス数	部分領域あたりのプロセス数	コース問題用プロセス数	プロセスあたりのスレッド数
Case A	1,536	6,144	1	192	12
Case B	3,072	12,288	1	16	12
Case C	3,072	24,576	4	24	6
Case D	6,144	49,152	4	24	6
Case E	12,000	96,000	4	24	6

表-4 計算時間とCG法の反復数

ケース	1CGの計算時間（s）	CG法の反復回数	全計算時間（s）
Case A	0.55	678	461
Case B	3.72	—	—
Case C	0.86	1,260	1,193
Case D	0.53	1,024	594
Case E	0.38	1,321	524

参考文献

[1] Home page of ADVENTURE Project, https://adventure.sys.t.u-tokyo.ac.jp
[2] 荻野正雄，塩谷隆二，金山寛，田上大助，吉村忍：バランシング領域分割法による並列弾性有限要素解析,

日本機械学会論文集 A 編, Vol.69, No.685, pp.1360-1367, 2003.

[3] Web site of Fugaku, https://www.r-ccs.riken.jp/en/fugaku/

[4] Mandel, J: Balancing domain decomposition, *Communications on Num. Methods in Eng*. Vol. 9, pp. 233-241, 1993.

[5] Yagawa, G., Shioya, R.: Parallel finite elements on a massively parallel computer with domain decomposition, *Comput. Syst. Engng*., Vol. 4, pp. 495–503, 1993.

[6] Karypis, G, Kumar, V: Parallel Multilevel K-Way Partitioning Scheme for Irregular Graphs, *Tech. Rep., Dept. of Comp. Sci. Univ. of Minnesota*, TR 96-036, 1996.

[7] Karypis. G, Kumar, V: Multilevel K-Way Partitioning Scheme for Irregular Graphs, *Tech. Rep., Dept. of Comp. Sci. Univ. of Minnesota*, TR 95-064, 1995.

[8] Home page of MUMPS: a parallel sparse direct solver, https://mumps-solver.org/

10:45 AM - 11:00 AM (Fri. Jun 2, 2023 10:30 AM - 11:15 AM Room C)

[C-11-02] Accelerating lattice Boltzmann method simulation with GPU computation using C++ standard language parallelism

*袁 子恒[1]、下川辺 隆史[1] （1. 東京大学）

計算工学講演会論文集　Vol.28 (2023年5月)　　計算工学会

階層型領域分割法による電磁場解析での四倍精度演算CG法，COCG法

CG and COCG methods with Quadruple-Precision Arithmetic in Electromagnetic Field Analysis by HDDM

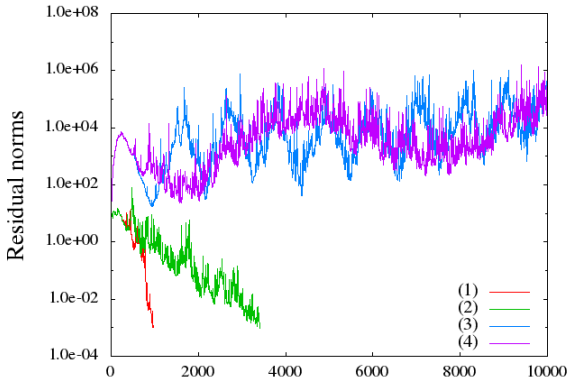
杉本振一郎[1)]
Shin-ichiro Sugimoto

1) 博(工) 八戸工業大学 工学部 准教授(〒031-8501 青森県八戸市妙字大開88-1, E-mail: sugimoto@hi-tech.ac.jp)

This paper deals with a parallel finite element analysis by Hierarchical Domain Decomposition Method (HDDM) of an electromagnetic field problem with a quadruple precision floating point number. A Conjugate Gradient (CG) method for real symmetric matrices and a Conjugate Orthogonal Conjugate Gradient (COCG) method for complex symmetric matrices are implemented in the quadruple precision for the parallel computing. Then, the HDDM with the quadruple precision arithmetic for the electromagnetic field problem is implemented by the C language. As a result, the convergence of the iterative method is improved, and the minimum residual norm is reduced by approximately four orders of magnitude in a high-frequency electromagnetic field analysis.

Key Words : *Parallel Computing, Finite Element Method, Hierarchical Domain Decomposition Method, Electromagnetic Field Analysis, Quadruple-Precision Arithmetic*

1. はじめに



Iteration counts on the interface
(1) 44M DOF (*A*-ϕ), (2) 44M DOF (*A*)
(3) 55M DOF (*A*-ϕ), (4) 55M DOF (*A*)
図 1. 時間調和渦電流問題の収束履歴[1].

電磁界解析は，温熱療法など電磁波を用いる医療行為の効果を明らかにするための数値人体モデルの解析や，変圧器などの静止器，モータや発電機などの回転機といった電気機器の設計などに広く活用されている．しかし人体や機器だけでなくそのまわりの空間も解析対象としなければならない電磁界解析のメッシュは自由度が大きくなりやすく，他の物理現象の解析に比べて演算量がかなり多い傾向にある．また電磁界解析で解くべき方程式には不定性があるため求解には反復法を用いざるを得ないが，条件数も悪いためその反復回数は非常に多く，演算量をさらに増加させている．そのうえ，方程式を構成する各項の係数となる物理量は材質によって桁が大きく異なる．演算量の増加は丸め誤差の蓄積を促進し，物理量の桁の違いは情報落ちを招く．

解析ソフトウェアのプログラミングでは実数の取り扱いに倍精度の浮動小数点数を用いるのが一般的であるが，電磁界解析においては丸め誤差の蓄積と情報落ちが倍精度演算では解析の破綻(=解が得られない)につながる恐れがあるほどに深刻である．例えばより高い精度の解を得るために同じモデルで自由度を増やすだけで解が得られなくなることがある．**図 1**は無限長ソレノイドコイルモデル[2]の時間調和渦電流問題を階層型領域分割法Hierarchical Domain Decomposition Method: HDDM)で解いた際の収束履歴である．4,400万自由度のメッシュでは*A*-ϕ法，*A*法とも収束しているが，5,500万自由度では残差が減ることなく発散してしまっている．

この問題を解決するには，実数の表現としてより精度の高い四倍精度や八倍精度などの多倍長精度浮動小数点数を用いることが考えられる．しかし多倍長精度演算には倍精度演算に比べてより長い時間がかかったり，メモリ使用量が増えたりといった問題がある．例えば四倍精度演算は倍精度演算に比べて演算時間が20～30倍，メモリ使用量は2倍となる．かつては十分な精度を得られるだけの自由度で実用的な時間内に解析を行うにはコンピュータの性能をギリギリまで使わなければならなかった．そのため計算時間が大幅に増え，メモリ使用量も数倍になる多倍長精度演算は，それを利用しなければ解を得られないようなごく一部の解析での利用に限られていた．しかしコンピュータ技術の進展により，演算性能が高く100 GB以上のメモリを搭載したパソコンが数十万円程度で手に入るようになった．また京と富岳を頂点とする

HPCI環境の整備により，年数万円程度で並列数数万，搭載メモリ100 TB以上のスーパーコンピュータを利用できるようになっている．さらにGNU Compiler Collection (GCC)やIntel C++ Compiler (ICC)などのC言語コンパイラでも四倍精度浮動小数点型__float128をサポートするようにもなり，多倍長精度演算を活用するための環境が整ってきたと考えられる．

本稿では四倍精度演算を用いた共役勾配法(Conjugate Gradient method: CG法) [3]，複素対称行列向けに共役直交共役勾配法(Conjugate Orthogonal Conjugate Gradient method: COCG法) [4]に基づく階層型領域分割法(Quadruple)を実装し，倍精度演算の階層型領域分割法(Double)と収束履歴，残差の最小値，計算時間を比較する．

2. 四倍精度演算での実装

本稿ではC言語で四倍精度演算を用いた有限要素解析の実装を行う．パソコンからスーパーコンピュータまで様々な環境に対応できるよう，コンパイラにはそれらで幅広く用いられているGCCとICCを用いることとし，これらでコンパイルできるようにプログラムを記述する．前述のとおり，この2つのコンパイラでは四倍精度浮動小数点型として__float128型(先頭の「_」(アンダーバー)は2つ)をサポートしているので，__float128型を用いて実装する．数学関数ライブラリは標準のlibquadmath.aを用いる．

また四倍精度の複素数型としては__complex128型(先頭の「_」(アンダーバー)は2つ)がサポートされている．ただし__float128型と__complex128型は相互に代入することが標準で有効なためバグに気づきにくくなること，環境によっては構造体の複素数の方が計算時間面で有利になることがあることから，**図 2**のように四倍精度の複素数型は__complex128型と構造体を切り替えて利用できるようにする．

```
typedef __complex128 Complex128 ;
```

(a) __complex128型の別名を定義

```
typedef struct {
  __float128 re ; __float128 im ;
} Complex128 ;
```

(b) 構造体による四倍精度複素数型

図 2. 四倍精度複素数型.

次に，CG法，COCG法に基づく階層型領域分割法を実装した．解くべき行列と既知ベクトルは四倍精度で作成する．ただし，実験で四倍精度に相当する精度で物性値を決定することが困難であること，四倍精度に相当する精度の座標値を持つメッシュを生成できるメッシャがないことから，入力データは倍精度であり，それらを四倍精度に型キャストしたうえで用いる．また，同様に四倍精度に相当する精度のデータを扱う可視化を行えるソフトウェアもないことから，出力データも倍精度とする．

3. 階層型領域分割法

階層型領域分割法(は領域分割法[5]-[7]を並列計算機環境に効率よく実装するための1手法である．大規模問題を効率よく数値計算することのできる手法としてよく知られており，分散メモリ環境で良好な並列効率を得られることが期待できる[8]．階層型領域分割法は大規模な構造解析[9]や熱伝導解析[10]に適用され，また電磁界解析でも数値人体モデルの高周波電磁界解析について2016年に300億自由度[11]，2019年に1,300億自由度[12]の解析に成功している．

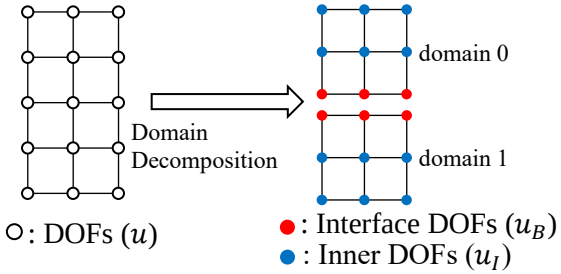


図 3. 領域分割.

元の問題：$Ku = f$

⇩ 領域分割

$$\begin{bmatrix} K_{II} & K_{IB} \\ K_{IB}^T & K_{BB} \end{bmatrix} \begin{bmatrix} u_I \\ u_B \end{bmatrix} = \begin{bmatrix} f_I \\ f_B \end{bmatrix}$$

⇩ 静的縮約

インターフェース問題：$Su_B = g$

図 4. 自由度の静的縮約.

$$y = Sx$$
$$= \sum_{i=0}^{1} R_B^{iT} S^i R_B^i \cdot x = \sum_{i=0}^{1} R_B^{iT} S^i x^i = \sum_{i=0}^{1} R_B^{iT} y^i,$$
$$y^i = S^i x^i = \left(K_{BB}^i - K_{IB}^{iT} \left(K_{II}^i\right)^{-1} K_{IB}^i\right) x^i$$
$$= K_{BB}^i x^i + K_{IB}^{iT} z^i,$$
$$K_{II}^i z^i = -K_{IB}^i x^i. \quad : \text{小領域での有限要素計算}$$

図 5. Sの行列ベクトル積.

階層型領域分割法では解析領域を並列数よりもはるかに多くの小領域(1小領域あたり数百自由度程度)に分割し，自由度を領域分割によって生じる領域間のインターフェース自由度(**図 3**)に静的縮約したインターフェース問題(**図 4**)を並列反復法で解く．解くべき問題そのものが変わるため並列性能を阻害する不完全Cholesky分解(IC)前処理を使わずとも収束解を得られる．また分割数を変えずに並列数を変えられるため並列反復法の収束性に並列数は影響しない．そのため，多倍長精度演算を階層型領域分割法に適用すれば並列数を増やすことにより計算時間とメモリ使用量の問題をともに解決することが期待できる．一方で解くべき行列Sが密であり，作成してしまうと効率的な解析ができなくなるので，並列反復法で必要なSの行

列ベクトル積は小領域での有限要素計算の結果の重ね合わせ(**図 5**)で行っている．

並列処理についてはMPIとOpenMPを用いるハイブリッド並列とする．階層型領域分割法ではまずメッシュをMPIのプロセス数と同じpart数に分割する．次にpart内で**図 3**のdomainに相当するsubdomainに分割する．ノード内のスレッド並列ではsubdomain単位で演算をスレッドに割り当てる．

4. 数値計算例

数値計算例としてTEAM20モデル[13]の静磁場解析(Magnetostatic), 無限長ソレノイドコイルのA-ϕ法での時間調和渦電流解析(TH_Eddy)，TEAM29モデル[14]の高周波電磁波解析(HF_EM)をそれぞれ倍精度演算，四倍精度演算で行う．それぞれの有限要素方程式の特徴を**表 1**に示す. モデルはいずれも四面体に分割され，それぞれの要素数は857,468，843,594，838,803，自由度はいずれも約100万自由度である．解析には東京大学情報基盤センターOakbridge-CXスーパーコンピュータ(OBCX) [15]を用いる. OBCXは1,240ノードのFujitsu PRIMERGY CX2550 M5 (Intel Xeon Platinum 8280 (2.7GHz，28+28コア)，192 GB memory搭載)をIntel Omni-Pathで接続したスーパーコンピュータである. 本稿では32ノード用いることとし，それぞれのメッシュは32 parts，各part内は224 subdomainsに分割する．静磁場解析ではCG法，時間調和渦電流解析，高周波電磁波解析ではCOCG法に基づく階層型領域分割法を簡易対角スケーリング前処理[16]とともに用いる. 収束判定値は設けず10,000回反復させて収束履歴, 残差の最小値, 計算時間を比較する．

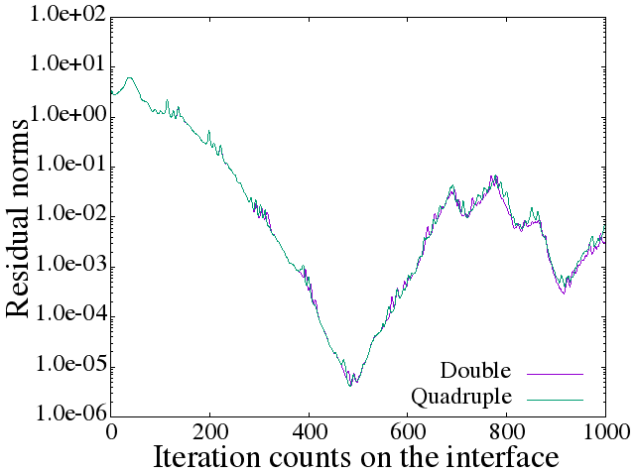
表 1. 有限要素方程式の特徴.

	Matrix	Singular	DOFs on
Magnetostatic	Symmetric Real	Yes	Edges
TH_Eddy	Symmetric Complex	Yes	Edges Apexes
HF_EM	Symmetric Complex	No	Edges

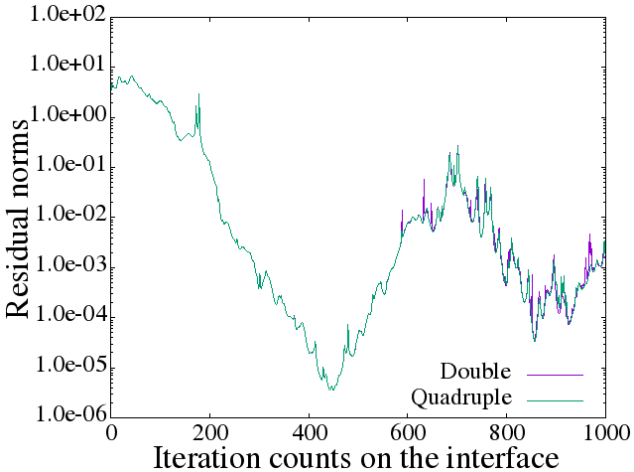
DOFs: Degrees of freedom

複素行列である時間調和渦電流解析，高周波電磁波解析で__complex128型と構造体の実装を計算時間で比較した結果，違いは最大でも1割にも満たず，どちらかが必ず短いということもなかった．そのためバグの発見のしやすさから，以降は構造体により複素数を表現する実装を用いることとする．

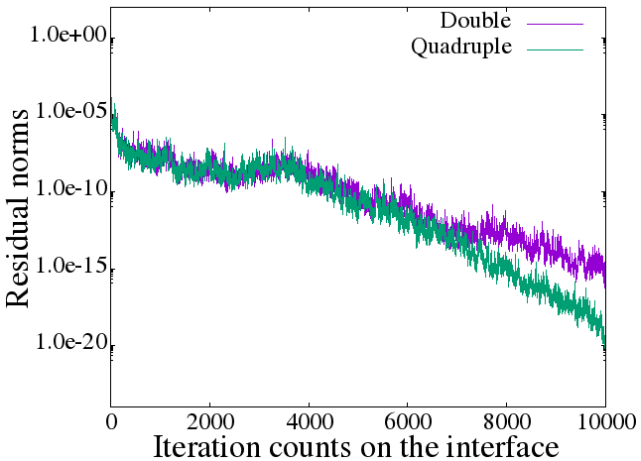
図 6にそれぞれの収束履歴を示す．解くべき方程式に不定性がある静磁場解析，時間調和渦電流解析は反復500回あたりで最小値をとって以降は発散するので，1,000回までの履歴である．**表 2**に残差の最小値と倍精度演算に対する四倍精度演算での減少率を示す．いずれも比較的解きやすいモデルであるためか静磁場解析，時間調和渦電流解析では最小値に大きな差はなかった．一方，不定性はないが静磁場解析，時間調和渦電流解析よりも条件数の悪い高周波電磁波解析では10,000反復までで4桁近く残差が小さくなった．倍精度演算に対する四倍精度演算の計算時間は静磁場解析で15.6倍，時間調和渦電流解析で23.5倍，高周波電磁波解析で14.4倍であった．



(a) Magnetostatic analysis.



(b) Time-Harmonic Eddy Current analysis.



(c) High-Frequency Electromagnetic Field analysis.

図 6. 収束履歴の比較.

表 2. 残差最小値の比較.

	Double	Quadruple	Q / D
Magnetostatic	4.278591e-06	4.108636e-06	9.60278e-01
TH_Eddy	3.564778e-06	3.564556e-06	9.99938e-01
HF_EM	5.549787e-17	6.805313e-21	1.22623e-04

5. おわりに

四倍精度演算を用い，ハイブリッド並列でCG法，およびCOCG法に基づく階層型領域分割法を実装した．約100万自由度のメッシュで倍精度演算に比べて四倍精度演算では反復法の残差の最小値が高周波電磁波解析で4桁減少し，四倍精度演算の利用が収束性の改善に効果があることを確認できた．今後はCG法，COCG法以外の反復法に基づく階層型領域分割法の実装を行うとともに，より大規模な問題やより複雑な問題で四倍精度演算導入の効果を検証していく．

謝辞

本研究の一部はJSPS科研費22H03605，22K19779の助成を受けて実施された．ここに記し，感謝の意を示す.

参考文献

[1] 杉本振一郎，田上大助，荻野正雄，武居周，金山寛: 階層型領域分割法による時間調和渦電流解析の収束性改善, 日本シミュレーション学会論文誌, Vol.7, No.1, pp.11-17, 2015.

[2] 金山寛: 計算電磁気学　岩波講座 現代工学の基礎　＜空間系 IV＞, 岩波書店, 2000.

[3] M. R. Hestenes and E. Stiefel: Methods of conjugate gradients for solving linear systems, *Journal of Research of the National Bureau of Standards*, Vol.49, No.6, pp.409-436, 1952.

[4] H. A. Vorst and J. B. M. Melissen: A Petrov-Galerkin type method for solving *Ax=b* and where *A* is symmetric complex, *IEEE Transactions on Magnetics*, Vol.26, Issue 2, pp.706-708, 1990.

[5] R. Glowinski, Q.V. Dinh and J. Periaux: Domain decomposition methods for nonlinear problems in fluid dynamics, *Computer Methods in Applied Mechanics and Engineering*, Vol.40, Issue 1, pp.27-109, 1983.

[6] A. Quarteroni and A. Valli: Domain Decomposition Methods for Partial Differential Equations, *Clarendon Press, Oxford*, 1999.

[7] A. Toselli and O. Widlund: Domain Decomposition Methods: Algorithms and Theory (Springer Series in Computational Mechanics), *Springer*, 2004.

[8] R. Shioya and G. Yagawa: Iterative domain decomposition FEM with preconditioning technique for large scale, problem, *ECM'99 Progress in Experimental and Computational Mechanics in Engineering and Material Behaviour*, pp.255-260, 1999.

[9] S. Yoshimura, R. Shioya, H. Noguchi and T. Miyamura: Advanced general-purpose computational mechanics system for large-scale analysis and design, *Journal of Computational and Applied Mathematics*, Vol.149, Issue 1, pp.279-296, 2002.

[10] A.M.M. Mukaddes, M. Ogino, M, H. Kanayama and R. Shioya: A scalable balancing domain decomposition based preconditioner for large scale heat transfer problems, *JSME International Journal Series B Fluids and Thermal Engineering*, Vol.49, No.2, pp.533-540, 2006.

[11] S. Sugimoto, A. Takei and M. Ogino: Finite element analysis with tens of billions of degrees of freedom in a high-frequency electromagnetic field, *Mechanical Engineering Letters*, Vol.3, 2017.

[12] S. Sugimoto, A. Takei and M. Ogino: High-Frequency Electromagnetic Field Analysis with 130 Billion of Degrees of Freedom, *The 38th JSST Annual Conference, International Conference on Simulation Technology*, pp.290-295, 2019.

[13] 回転機電磁界解析ソフトウェアの適用技術調査専門委員会: 回転機電磁界解析ソフトウェアの適用技術，電気学会技術報告，No.486，1994.

[14] Y. Kanai: Description of TEAM Workshop Problem 29: Whole body cavity resonator, TEAM Workshop in Tucson, 1998.

[15] 東京大学情報基盤センター スーパーコンピューティング部門HP：https://www.cc.u-tokyo.ac.jp/

[16] H. Kanayama and S. Sugimoto: Effectiveness of A-phi Method in a parallel computing with an iterative domain decomposition method, IEEE Transactions on Magnetics, Vol.42, No.4, pp.539-542, 2006.