計算数理工学論文集 Vol. 13 (2013 年 11 月), 論文 No. 26-131129 JASCOME

ACA を用いた演算子積分時間領域境界要素法の効率化

Efficient calculation of the convolution quadrature boundary element method using ACA

伊海田 明宏 [1)], 斎藤 隆泰 [2)], 廣瀬 壮一 [3)]

Akihiro IKAIDA, Takahiro SAITOH and Sohichi HIROSE

1) 東京工業大学大学院情報理工学研究科 (〒 152-8552 東京都目黒区大岡山 2-12-1, E-mail: ikaida.a.aa@m.titech.ac.jp)
2) 群馬大学理工学研究院環境創生部門 (〒 376-8515 群馬県桐生市天神町 1-5-1, E-mail: t-saitoh@gunma-u.ac.jp)
3) 東京工業大学大学院情報理工学研究科 (〒 152-8552 東京都目黒区大岡山 2-12-1, E-mail: shirose@cv.titech.ac.jp)

This paper presents a fast and efficient convolution quadrature boundary element method for SH wave propagation. Adaptive cross approximation (ACA) is applied to efficiently calculate matrix-vector product of the discretized time-domain boundary integral equation. Two types of ACAs are considered. The one is "ACAstore" which stores both full and low rank blocks, and the other is "ACArelease" which releases allocated memory for low rank blocks in each time step. As numerical examples, the both proposed methods are applied to a multiple scattering problem. "ACAstore" and "ACArelease" are effective for reduction of computational time and memory, respectively.

Key Words: Time-domain BEM, Convolution quadrature method (CQM), Adaptive cross approximation (ACA), $\mathcal{H}$-matrices

1. はじめに

本論文では, Adaptive Cross Approximation (ACA) による演算子積分時間領域境界要素法の効率化について述べる.

演算子積分法 (CQM: Convolution Quadrature Method)[(1)] は, Lubich が開発した畳み込み積分の離散化手法である. この CQM を時間領域境界要素法の畳み込み積分に用いた方法が演算子積分時間領域境界要素法 (CQ-BEM: Convolution Quadrature Boundary Element Method) であり, 波動伝搬問題において時間増分が小さい場合でも安定的に解を求める事が可能である. また CQ-BEM は Laplace 変換域の基本解を用いるため, 時間領域の基本解が閉じた形で求まらない粘弾性波動問題等にも用いることができる. 著者らのグループでは, これまでにスカラー波動 [(2)] や弾性波動 [(3)], 粘弾性波動 [(4)], 異方性弾性波動問題 [(5)] といった数多くの問題に CQ-BEM を適用してきた.

演算子積分法を用いた時間領域境界要素法により大規模問題を解く場合, 多大な計算時間と記憶容量が必要となる. そのため, これらの計算コストを削減するために, 高速多重極法 (FMM: Fast Multipole Method) [(6)] が主に用いられてきた [(2)(4)(7)(8)]. 高速多重極法は非常に有用な計算効率化手法であるが, 実装には基本解の多重極展開を導く必要があるため, 異方性弾性波動問題など, 基本解の多重極展開を導くことが困難な問題に対しては適用が難しいのが現状である.

2013 年 9 月 30 日受付, 2013 年 10 月 18 日受理

したがって, 本研究では ACA を CQ-BEM に適用し, 計算の効率化を図ることを検討する. ACA[(9)(10)(11)] は, 行列を近似圧縮した形で求めることで行列の生成や行列ベクトル積の計算を効率化する手法である. ACA は基本解の形に依存せず実行できるため, その境界要素法への適用は比較的容易である. しかし, CQ-BEM に ACA を適用した研究例は数少ない. 例えば, 吉川ら [(12)] は ACA の CQ-BEM への適用を検討しているが, 影響関数の計算に高速 Fourier 変換 (FFT) を援用していない.

そこで本論文では,CQ-BEM における行列の成分計算および行列ベクトル積の計算を ACA により効率化する方法について検討する. ただし, 著者らのグループの過去の適用例 [(8)] と同様に, CQ-BEM の影響関数の計算に FFT を用いる. 以下では, 簡単のため二次元面外波動問題を対象に CQ-BEM の定式化について簡単に説明した後, ACA のアルゴリズム, 適用法について述べる. 数値解析例として, 開発した手法により多重散乱問題を解き, 従来法と結果を比較することで, 計算効率や精度を確認するとともに, 本手法の有効性を検討する. 最後に結論および今後の課題について述べる.

2. 二次元面外波動の散乱問題

ここでは, Fig.1 に示すような等方均質な二次元無限領域 D 中における, 散乱体 D_c による面外波動の散乱問題について考える. ただし, 入射波 $u^{\mathrm{in}}(\mathbf{x},t)$ が D_c の境界 ∂D に到達した時間を時刻 $t=0$ とし, $t<0$ では面外変位 $u(\mathbf{x},t)$ および

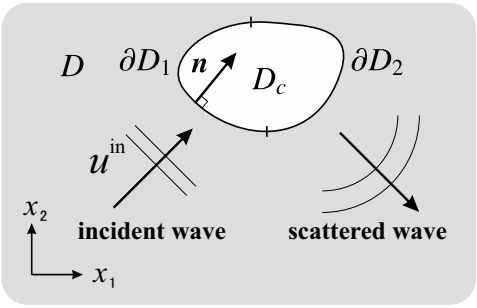

Fig. 1　SH wave scattering by a scatterer.

表面力 $t(\mathbf{x},t)$ は静止過去の条件を満足すると仮定する. このとき初期条件は

$$u(\mathbf{x},0)=0,\quad t(\mathbf{x},0)=\mu\frac{\partial u(\mathbf{x},0)}{\partial n}=0\ \text{ in } D \tag{1}$$

で与えられるとする. ここで, μ はせん断弾性定数, $\partial/\partial n$ は法線方向微分を表す. 今, 物体力を無視すれば, 支配方程式および境界条件はそれぞれ次のように表される.

$$\mu\nabla^2 u(\mathbf{x},t)=\rho\frac{\partial^2 u(\mathbf{x},t)}{\partial t^2}\quad \text{in } D,\ t\geq 0 \tag{2}$$

$$u=\tilde{u}\ \text{ on } \partial D_1,\quad t=\tilde{t}\ \text{ on } \partial D_2,\quad \partial D_2=\partial D\backslash\partial D_1 \tag{3}$$

ここで, ρ は密度, $\tilde{(\ )}$ は与えられた境界条件を表す.

この問題の解は, 次の時間領域境界積分方程式を解くことで求まる.

$$C(\mathbf{x})u(\mathbf{x},t)=u^{\text{in}}(\mathbf{x},t)+\int_{\partial D}U(\mathbf{x},\mathbf{y},t)*t(\mathbf{y},t)ds_{\mathbf{y}} -\int_{\partial D}T(\mathbf{x},\mathbf{y},t)*u(\mathbf{y},t)ds_{\mathbf{y}} \tag{4}$$

ここで, $C(\mathbf{x})$ は自由項であり, 境界 ∂D を一定要素で離散化する場合は $C=1/2$ となる. $U(\mathbf{x},\mathbf{y},t)$ および $T(\mathbf{x},\mathbf{y},t)$ は, それぞれ二次元面外波動問題に対する時間領域での基本解および二重層核である. また $*$ は時間に関する畳み込み積分を表す. 式 (4) の畳み込み積分を, Lubich の演算子積分法を用いて次節で述べるようにして離散化する.

3. 演算子積分時間領域境界要素法 (CQ-BEM)

3.1. 演算子積分法 (CQM)

CQ-BEM の定式化の前に, Lubich の演算子積分法 (CQM)[1] について簡単に述べる. Lubich は, 畳み込み積分 $f(t)*g(t)$ を, 関数 $f(t)$ の Laplace 変換を用いて離散近似する手法を提案した. 畳み込み積分は一般に次のように表される.

$$f(t)*g(t)=\int_0^t f(t-\tau)g(\tau)d\tau \tag{5}$$

Lubich の方法を用いることで, 式 (5) の畳み込み積分は, 時間増分 Δt を用いて時間 t を N_t ステップに分割することで, 次のように近似することができる.

$$f(n\Delta t)*g(n\Delta t)\approx\sum_{j=0}^{n}\omega_{n-j}(\Delta t)g(j\Delta t),\quad n=0,1,\ldots,N_t \tag{6}$$

ここで, $\omega_j(\Delta t)$ は重み関数であり, 関数 $f(t)$ の Laplace 変換

$$\hat{f}(s)=\int_0^\infty f(t)e^{-st}dt \tag{7}$$

を用いて次のように表される.

$$\omega_n(\Delta t)\simeq\frac{\mathcal{R}^{-n}}{L}\sum_{l=0}^{L-1}\hat{f}\left(\frac{\delta(z_l)}{\Delta t}\right)e^{-\frac{2\pi\mathrm{i}nl}{L}} \tag{8}$$

ただし, $\mathrm{i}=\sqrt{-1}$ である. また, $\hat{f}$ が存在するためには, 引数の実部が正の時に正則であることが必要である. 式 (8) の $\delta(z_l)$ は線形多段法における生成多項式の商であり, 引数 z_l は半径 $\mathcal{R}<1$ の円周上の L 等分点を考えて $z_l=\mathcal{R}e^{2\pi\mathrm{i}l/L}$ と表される. $\mathcal{R}$ は目標とする精度 ε によって決定されるパラメータであり, $\mathcal{R}=\varepsilon^{1/(2L)}$ で与える.

3.2. CQ-BEM の定式化

式 (4) の畳み込み積分を式 (6),(8) を用いて離散近似し, 境界 ∂D を N_e 個の一定要素 ∂D_i $(i=1,\ldots,N_e)$ を用いて選点法により離散化することで, 次のような第 n ステップ $(n=1,\ldots,N_t)$ における離散化された境界積分方程式を得ることができる.

$$\frac{1}{2}u_i(n\Delta t)=u_i^{\text{in}}(n\Delta t) +\sum_{j=1}^{N_e}\sum_{k=1}^{n}\left[A_j^{n-k}(\mathbf{x}_i)t_j(k\Delta t)-B_j^{n-k}(\mathbf{x}_i)u_j(k\Delta t)\right] \tag{9}$$

ここで, Δt は時間増分である. また, $u_i(n\Delta t)$ および $t_i(n\Delta t)$ はそれぞれ要素 ∂D_i の選点 $\mathbf{x}_i$ における第 n ステップの面外変位および表面力である. 一方, A_j^m および B_j^m は基本解および二重層核に関する影響関数であり,

$$A_j^m(\mathbf{x}_i):=\frac{\mathcal{R}^{-m}}{L}\sum_{l=0}^{L-1}\left[\int_{\partial D_j}\hat{U}(\mathbf{x}_i,\mathbf{y},s_l)ds_{\mathbf{y}}\right]e^{-\frac{2\pi\mathrm{i}ml}{L}}$$
$$B_j^m(\mathbf{x}_i):=\frac{\mathcal{R}^{-m}}{L}\sum_{l=0}^{L-1}\left[\int_{\partial D_j}\hat{T}(\mathbf{x}_i,\mathbf{y},s_l)ds_{\mathbf{y}}\right]e^{-\frac{2\pi\mathrm{i}ml}{L}} \tag{10}$$

と表される. ここで, $s_l=\delta(z_l)/(\Delta t)$ である. また, $\hat{U}(\mathbf{x},\mathbf{y},s)$ および $\hat{T}(\mathbf{x},\mathbf{y},s)$ は, それぞれ二次元面外波動問題に対する Laplace 変換域の基本解および二重層核であり,

$$\hat{U}(\mathbf{x},\mathbf{y},s)=\frac{1}{2\pi\mu}K_0(sr)$$
$$\hat{T}(\mathbf{x},\mathbf{y},s)=\mu\frac{\partial}{\partial n_{\mathbf{y}}}\hat{U}(\mathbf{x},\mathbf{y},s)=-\frac{s}{2\pi}K_1(sr)\frac{\partial r}{\partial n_{\mathbf{y}}} \tag{11}$$

と表される. ただし, $r:=|\mathbf{x}-\mathbf{y}|$ であり, $\partial/\partial n_{\mathbf{y}}$ は点 $\mathbf{y}$ における法線方向微分, K_h は h 次の第二種変形 Bessel 関数である. 式 (9) 右辺の第 n ステップにおける境界値 $u_j(n\Delta t)$ および $t_j(n\Delta t)$ を含む項を左辺に移項すると次のようになる.

$$\frac{1}{2}u_i(n\Delta t)-\sum_{j=1}^{N_e}\left[A_j^0(\mathbf{x}_i)t_j(n\Delta t)-B_j^0(\mathbf{x}_i)u_j(n\Delta t)\right]$$
$$=u_i^{\text{in}}(n\Delta t)$$
$$+\sum_{j=1}^{N_e}\sum_{k=1}^{n-1}\left[A_j^{n-k}(\mathbf{x}_i)t_j(k\Delta t)-B_j^{n-k}(\mathbf{x}_i)u_j(k\Delta t)\right] \tag{12}$$

実際に第 n ステップにおける境界値を計算する際には, 式 (12) 右辺の第 n ステップ以前の境界値を全て求めておいた上で右辺を計算し, 左辺の連立方程式を解く, といった手順をとる. 以下では, 式 (12) 右辺の計算例として, 影響関数 B_j^{n-k} を含む項 $\mathbf{b}^n := \sum_{k=1}^{n-1} \mathbf{B}^{n-k}\mathbf{u}^k$, $\{\mathbf{B}^{n-k}\mathbf{u}^k\}_i = \sum_{j=1}^{N_e} B_j^{n-k}(\mathbf{x}_i)u_j(k\Delta t)$ の計算について述べるが, この計算は影響関数 A_j^{n-k} を含む項に関しても同様に実行される. 式 (10) より, $\mathbf{B}^m\mathbf{u}^k$ は次のように表せる.

$$\mathbf{B}^m\mathbf{u}^k = \frac{\mathcal{R}^{-m}}{L}\sum_{l=0}^{L-1}\hat{\mathbf{T}}^l\mathbf{u}^k e^{-\frac{2\pi iml}{L}} \tag{13}$$

ここで, $\{\hat{\mathbf{T}}^l\}_{ij} = \int_{\partial D_j}\hat{T}(\mathbf{x}_i, \mathbf{y}, s_l)ds_{\mathbf{y}}$ である. ただし, $\mathbf{B}^m\mathbf{u}^k$ は式 (13) のようにして $\hat{\mathbf{T}}^l$ と $\mathbf{u}^k$ の積から求めるため, $\mathbf{B}^m$ そのものを直接計算することはしない. 式 (13) 右辺は離散 Fourier 変換の形をしているため, $L = N_t$ とすれば, FFT を適用することで, $\mathbf{B}^m\mathbf{u}^k$ を $m = 1, \ldots, N_t$ について高速に計算することができる. 計算した $\mathbf{B}^m\mathbf{u}^k$ の総和を次のように計算することで $\mathbf{b}^n$ を求める [8].

$$\begin{Bmatrix}\mathbf{b}^1\\ \mathbf{b}^2\\ \mathbf{b}^3\\ \vdots\\ \mathbf{b}^{N_t}\end{Bmatrix} = \begin{Bmatrix}0\\ \mathbf{B}^1\mathbf{u}^1\\ \mathbf{B}^2\mathbf{u}^1 \quad +\mathbf{B}^1\mathbf{u}^2\\ \vdots\\ \mathbf{B}^{N_t-1}\mathbf{u}^1+\mathbf{B}^{N_t-2}\mathbf{u}^2+\cdots+\mathbf{B}^1\mathbf{u}^{N_t-1}\end{Bmatrix} \tag{14}$$

すなわち, 式 (14) において, 第 k ステップにおける変位 $\mathbf{u}^k$ が得られた時点で, 式 (14) 右辺の $\mathbf{u}^k$ に関する項を全て計算し $\mathbf{b}^i$ $(i = 1, \ldots, N_t)$ に加算する.

式 (13) 右辺の二重層核についての係数行列 $\hat{\mathbf{T}}^l$ は一般に密行列となり, l に関して $L = N_t$ 個存在するため (実際の計算では, 複素引数 s_l についての Laplace 変換域基本解が複素平面上の実軸に関して対称となる事を利用するため, $L/2 + 1$ 個だけ計算すれば良い [8]), N_e が大きい問題を解くには計算効率の向上が不可欠となる. ACA や高速多重極法等の計算効率化手法を用いないで計算した場合, 全時間ステップの計算時間は $O(N_e^2N_t^2)$, 行列の格納に必要となる記憶容量は $O(N_e^2N_t)$ となる [7]. そこで次節では, 式 (12) 右辺の係数行列を ACA を用いて圧縮した形で求めることで計算時間や記憶容量といった計算コストを削減する手法について述べる.

4. ACA を用いた CQ-BEM の計算効率化

4.1. 低ランク近似

ACA を用いた低ランク近似により境界要素法の係数行列を圧縮し, 効率的に計算を行う事が出来る. 低ランク近似 [9][11] とは, 行列 $\mathbf{M} \in \mathbb{C}^{t\times s}$ を

$$\mathbf{M} \approx \mathbf{M}_k = \mathbf{U}\mathbf{V}^{\mathrm{T}} = \sum_{r=1}^{k}\mathbf{u}_r\mathbf{v}_r^{\mathrm{T}}$$
$$\mathbf{U} \in \mathbb{C}^{t\times k},\ \mathbf{V} \in \mathbb{C}^{s\times k} \tag{15}$$

と表されるような行列積 $\mathbf{U}\mathbf{V}^{\mathrm{T}}$(低ランク行列) で近似することで圧縮する方法である. ただし, $\mathbf{V}^{\mathrm{T}}$ は $\mathbf{V}$ の転置行列を表す. また, k は近似のランクであり, $k(t+s) < ts$ を満たすものとする. 低ランク近似により, 行列の大きさおよび行列の保存に必要となる記憶容量は $O(k(t+s))$ となる. また, 低ランク行列に対する行列ベクトル積は次式のように表される.

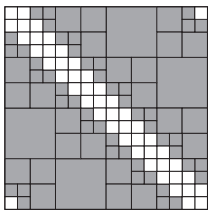

Fig. 2 Example of $\mathcal{H}$-matrices with low rank blocks (gray) and full rank blocks (white).

$$\mathbf{M}_k\mathbf{a} = \mathbf{U}\mathbf{V}^{\mathrm{T}}\mathbf{a} = \mathbf{U}\mathbf{b} = \mathbf{c} \tag{16}$$

したがって, 行列ベクトル積の計算量は行列の大きさと同様に $O(k(t+s))$ となる.

4.2. 階層行列 ($\mathcal{H}$-matrices)

低ランク近似は係数行列全体に適用することは出来ないため, 係数行列を境界要素代表点の配置に応じて分割し, 階層行列 ($\mathcal{H}$-matrices) 化する必要がある. 階層行列 [11] とは, 行列とその部分行列を再帰的に分割した行列の集合であり, 例えば Fig.2 のように表される. 階層行列では部分行列が階層状に配置されており, その各部分行列について後述の近似許容条件を考えて, 低ランク近似を行うか否かを判断する. すなわち, Fig.2 の low rank block は近似を行う部分行列, full rank block は近似を行わない部分行列を表す. 階層行列は以下のような手順で作成される.

まず, 係数行列の行および列番号の集合を再帰的に二分割し, 分割後の集合をまとめてクラスタ木とする. クラスタ木とは行および列番号の集合 (クラスタ) からなる木構造である. Fig.3 (a) はクラスタ分割の例であり, 対応する境界要素代表点を用いてクラスタを可視化している. また, 分割後のクラスタを分割前から見て「子」と呼び, 各々の子クラスタに対しても, クラスタに含まれる要素数があらかじめ与えた $n_{\min}$ 以下になるまで分割を繰り返す. なお, 分割には bounding box を用いる手法 [11] を使用した. Bounding box とは, Fig.3 (a) に表されているように, クラスタの成分が対応する境界要素代表点が全て含まれるような最小の長方形 (三次元では直方体) を, 各辺が座標軸に平行になるように定義したものである. この bounding box の長辺の中点を全て含むような直線 (三次元では平面) で境界要素代表点の集まりを二分割し, 子クラスタの成分とする.

次に, クラスタ木の組み合わせからなるブロック木により階層行列を定義する. ブロック木は, 分割回数が同じクラスタ同士の組み合わせ (ブロック) からなる木構造であり, ブロックには, 各クラスタに含まれる行・列番号に対応した部分行列が割り当てられる. ブロックの分割は各クラスタの子を組み合わせることで定義されるが, いずれかのクラスタの子が存在しない場合や, 近似の許容条件を満たす場合はそのブロッ

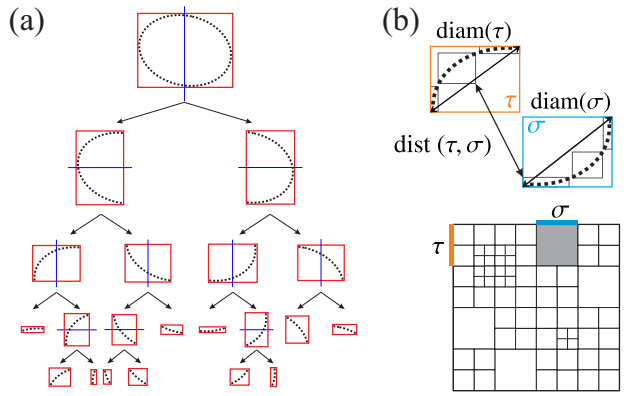

Fig. 3 Subdivision of (a) clusters by bounding box and (b) matrices and admissibility condition of a subblock which corresponds to a cluster set τ and σ.

クの分割は行わない. 近似の許容条件は次式で与えられる.

$$\min\{\mathrm{diam}(\tau), \mathrm{diam}(\sigma)\} \leq \eta \ \mathrm{dist}(\tau, \sigma) \tag{17}$$

ここで, τ, σ はそれぞれ行および列番号に対応したクラスタであり, $0 < \eta < 1$ は許容パラメータである. η が小さいほど許容条件は厳しくなるため低ランク近似の精度は向上するが, 圧縮効率は低下する. また, 式 (17) の $\mathrm{diam}(\tau)$, $\mathrm{dist}(\tau, \sigma)$ はそれぞれクラスタ τ の径, クラスタ τ, σ 間の距離を表し, bounding box を用いる場合は, Fig.3(b) のように bounding box の対角長, bounding box 間の最短距離と定義できる. ただし, τ, σ に子クラスタ $\tau_s \in \tau$, $\sigma_s \in \sigma$ が存在する場合, $\mathrm{dist}(\tau, \sigma)$ は分割不可能な子クラスタ同士の全ての組み合わせについての距離の最小値とする. このようにクラスタ間の距離を決めることで, 境界の形状により近い距離を少ない時間で計算できる.

あるブロックが式 (17) を満たす場合はクラスタ間の距離が遠いと判断し, 対応する部分行列を式 (15) の低ランク近似した形で保存する (low rank block). 式 (17) を満たさない場合は分割を行い, 子ブロックに対して式 (17) を考える. もし分割をそれ以上行えなくなった場合は, 対応する部分行列を通常の行列の形で保存する (full rank block).

4.3. Adaptive Cross Approximation (ACA)

階層行列内の部分行列のうち, 式 (17) の許容条件を満たしたものを Adaptive Cross Approximation (ACA) により低ランク行列に圧縮する. ACA[9][10][11] とは低ランク行列を高速, かつ純粋な代数計算により求めるアルゴリズムである. 一般に用いられる ACA では, 式 (15) における圧縮前の行列 $\mathbf{M} \in \mathbb{C}^{t \times s}$ を $\mathbf{M} = \mathbf{M}_k + \mathbf{R}_k$ として, ランク k の低ランク行列 $\mathbf{M}_k$ と残差 $\mathbf{R}_k$ に分けて考える. これらの初期値 ($r = 0$) を

$$\mathbf{M}_0 := \mathbf{0}, \quad \mathbf{R}_0 := \mathbf{M} \tag{18}$$

とし, 残差から低ランク行列へ成分を抜き出す操作を, 残差が微小となるまで繰り返すことで低ランク行列を作成する. 以下では, この繰り返し計算ステップ $r = 1, 2, \ldots$ で行われる操作について述べる.

まず, 残差 $\mathbf{R}_{r-1}$ から $r > 0$ ステップ目のピボット $\gamma_r = (\mathbf{R}_{r-1})^{-1}_{i^* j^*}$ を選択する. ここで, i^*, j^* は選択されたピボット行および列番号であり, $(\cdot)_{ij}$ は i 行 j 列成分を表すとする. ピボット行および列はピボット γ_r の絶対値が大きいものを選択するが, 選択方法[11] には幾つか種類があり, そのうちの一つを後述する.

次に, 以下のように $\mathbf{R}_{r-1}$ から $\mathbf{M}_r$ へ成分を抜き出す.

$$(\mathbf{u}_r)_i := \gamma_r (\mathbf{R}_{r-1})_{ij^*} \tag{19}$$

$$(\mathbf{v}_r^{\mathrm{T}})_j := (\mathbf{R}_{r-1})_{i^* j} \tag{20}$$

$$\mathbf{M}_r := \mathbf{M}_{r-1} + \mathbf{u}_r \mathbf{v}_r^{\mathrm{T}} \tag{21}$$

$$\mathbf{R}_r := \mathbf{R}_{r-1} - \mathbf{u}_r \mathbf{v}_r^{\mathrm{T}} \tag{22}$$

ここで, 式 (19), (20) において必要となる $(\mathbf{R}_{r-1})_{ij}$ は, 式 (21), (22) から次のように書ける.

$$(\mathbf{R}_{r-1})_{ij} = (\mathbf{M})_{ij} - (\mathbf{M}_{r-1})_{ij}, \quad r > 0 \tag{23}$$

したがって, 行列 $\mathbf{M}$ の成分全てを計算する必要は無く, ベクトル $\mathbf{u}_r, \mathbf{v}_r$ の計算に必要となる行, 列のみを計算すれば良い. なお, ここで求めた $\mathbf{u}_r, \mathbf{v}_r$ を利用し, ピボット行 i^*, 列 j^* を

$$i^* = \begin{cases} \text{arbitrary} & \text{for} \quad r = 1 \\ \underset{i}{\mathrm{argmax}} \, |(\mathbf{u}_{r-1})_i| & \text{for} \quad r > 1 \end{cases} \tag{24}$$

$$j^* = \underset{j}{\mathrm{argmax}} \, |(\mathbf{v}_r)_j| \tag{25}$$

と決める事ができる. すなわち, この方法では計算を式 (24), (20), (25), (19), (21) といった順序で実行することとなる. なお, この方法では境界に角があるような場合に二重層核の係数行列の圧縮効率が低下することがあり, その場合は改善されたピボット選択法[11][13] を用いるのが望ましい.

最後に, 次のような収束条件式を考える.

$$\|\mathbf{u}_r\|_{\mathrm{F}} \|\mathbf{v}_r\|_{\mathrm{F}} < \varepsilon_{\mathrm{ACA}} \|\mathbf{M}_r\|_{\mathrm{F}} \tag{26}$$

これを満たす場合は近似のランクを $k = r$ と決定しループを脱出, 満たさない場合は r を $r+1$ としてピボットの選択から同様に繰り返す. ここで, $\varepsilon_{\mathrm{ACA}}$ は ACA の精度を決めるパラメータである. また, 近似行列の Frobenius ノルム $\|\mathbf{M}_r\|_{\mathrm{F}}$ は,

$$\begin{aligned} \|\mathbf{M}_r\|_{\mathrm{F}}^2 &= \sum_{i=1}^{t} \sum_{j=1}^{s} |(\mathbf{M}_r)_{ij}|^2 \\ &= \sum_{i=1}^{t} \sum_{j=1}^{s} |(\mathbf{M}_{r-1})_{ij} + (\mathbf{u}_r)_i (\mathbf{v}_r)_j|^2 \\ &= \|\mathbf{M}_{r-1}\|_{\mathrm{F}}^2 + \sum_{\nu=1}^{r-1} (\mathbf{u}_\nu^{\mathrm{T}} \bar{\mathbf{u}}_r \mathbf{v}_\nu^{\mathrm{T}} \bar{\mathbf{v}}_r + \bar{\mathbf{u}}_\nu^{\mathrm{T}} \mathbf{u}_r \bar{\mathbf{v}}_\nu^{\mathrm{T}} \mathbf{v}_r) + \|\mathbf{u}_r\|_{\mathrm{F}}^2 \|\mathbf{v}_r\|_{\mathrm{F}}^2 \end{aligned} \tag{27}$$

のように漸化式に変形することで効率的に計算出来る. ただし, $|z| = \sqrt{z\bar{z}}$, $z \in \mathbb{C}$ であり, $\bar{z}$ は z の共役複素数を表す.

ACA による低ランク行列の計算量は $O(k(t+s))$ となる. 厳密には, ACA には式 (23), (26) といった $O(k^2(t+s))$ となる計算が存在するが, 成分計算の計算量が支配的であるため,

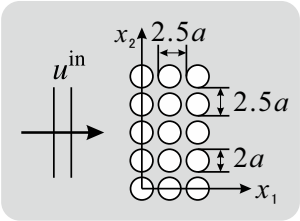

Fig. 4　Multiple scattering model.

$O(k(t+s))$ と見なすことが出来る. 成分の計算は, 前述の通り近似後の行列に必要となる分のみで済ませる事が出来る.

4.4. CQ-BEM に対する ACA 適用上の注意点

本論文では, 式 (13) の $\hat{\mathbf{T}}^l$ 等と表せるような CQ-BEM の係数行列を, ACA を用いた階層行列の形で生成することにより計算コストを削減する. 式 (13) から明らかなように, 係数行列が各 l について存在するため, ACA を適用する部分行列も l ごとに作成する. ただし, 部分行列の中身は異なるものの, 階層行列の構造自体は境界要素代表点により決まるため, l に関わらず一定となる.

また, l の変化に応じて部分行列の成分の性質が異なるため, ACA の精度 ε_{ACA} を変化させて対応する. l は, Laplace 変換域における波数に相当するパラメータである s_l の値を変化させるが, 周波数域の波動問題に対する境界要素法に ACA を適用する場合, 波数が増加すると低ランク近似のランクが増大することが知られている [11]. そのため, 今回のような Laplace 変換域における問題でも同様の現象が発生し, l の増加に伴い圧縮効率が悪化する. ところで, 式 (10) の基本解および二重層核に含まれる h 次の第二種変形 Bessel 関数 $K_h(z)$ は, $|\text{Re}[z]|$ が大きい場合に $K_h(z) \simeq e^{-z}$ と表現する事ができる. したがって, $|\text{Re}[z]|$ が大きい値を取る場合の基本解や二重層核の値は十分小さくなり, 式 (13) 等の計算精度にさほど影響しないと考えられる. この事を利用し, $|\text{Re}[z]|$ が十分大きい値となる場合に, ACA による近似の精度を次のように l の値に応じて緩和する事が出来る.

$$\varepsilon'_{\text{ACA}} := \varepsilon_{\text{ACA}} / e^{-\text{Re}[s_l] r_d} \tag{28}$$

ただし, r_d は部分行列に対応するクラスタ τ, σ 同士の最短距離とした. $\varepsilon'_{\text{ACA}}$ を用いることで, r_d が大きい場合には l が増加したときの圧縮効率の悪化を抑制し, それでいて階層行列全体で見た場合の計算精度への影響を抑える事が出来る.

5. 数値解析例

数値解析例として, Fig.4 のような 15 個の空洞の近傍での入射波 u^{in} により生じる散乱波動場を解析した結果を示す. Fig.4 において $a = 1.0$ とし, 各散乱体を 64〜4,096 個の境界要素に分割した. また, 総時間ステップ数は, $N_t = L = 256$ とした. 入射波および境界条件は次のように与える.

$$u^{\text{in}}(\mathbf{x}, t) = u_0(1 - \cos 2\pi\alpha),$$
$$\alpha = \begin{cases} \dfrac{c_T}{\lambda}\left(t - \dfrac{x_1 + a}{c_T}\right) & \text{for} \quad 0 \le \alpha \le 1 \\ 0 & \text{for} \quad \text{otherwise} \end{cases} \tag{29}$$

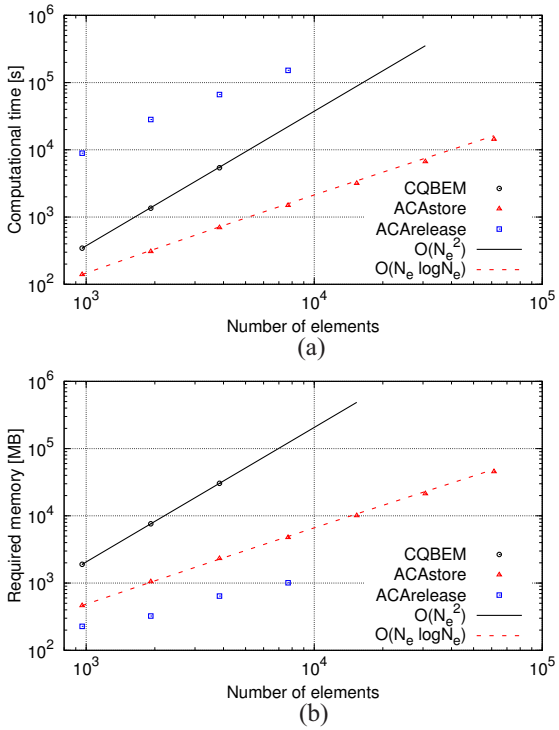

Fig. 5　Analysis results with $\varepsilon_{\text{ACA}} = 10^{-6}$: (a) Computational time. (b) Total sizes of coefficient matrices on memory.

$$\frac{\partial u(\mathbf{x}, t)}{\partial n} = 0 \quad \text{on } \partial D, \ t \ge 0, \tag{30}$$

ただし, c_T はせん断波速度であり, $u_0 = 1.0$, $\lambda = 2.0$, $c_T = 1.0$, $\Delta t = 0.03$ とした. 演算子積分法の精度は $\varepsilon = 10^{-10}$ とし, 階層行列に関するパラメータは $\varepsilon_{\text{ACA}} = 10^{-3}$〜$10^{-6}$, $\eta = 0.9$, $n_{\min} = 20$ とした. また, 計算は東京工業大学の TSUBAME2.0 上で行い, CPU (Intel Xeon 2.93 GHz) の 1 コアを使用した. OpenMP 等の並列化は行っていない. ここでは, 計算時間の上限を 24 時間, 記憶容量の上限を 50GB とした.

$\varepsilon_{\text{ACA}} = 10^{-6}$ とした場合の, 要素数 N_e ごとの計算時間および係数行列の保存に必要な記憶容量を, それぞれ Fig.5(a), (b) にまとめた. ここで言う係数行列とは, 式 (10) 右辺の積分項を成分とした行列であり, そのうち, 式 (12) 右辺の境界未知量に対応する影響関数の計算に必要となる部分のみが保存される. Fig.5(a), (b) における “CQ-BEM” は係数行列を全て保存する ACA を適用しない通常の CQ-BEM, “ACAstore” は階層行列に係数行列を圧縮した形で全て保存した計算を表す. また, “$O(N_e^2)$” および “$O(N_e \log N_e)$” は各計算オーダーの傾きを表しており, それぞれ CQ-BEM および ACAstore の要素数が最も少ない場合の点を通るように描かれている. “ACArelease” は, full rank block については ACAstore と同じように成分を保存するが, low rank block の成分は保存せずに, 時間ステップ毎に式 (12) 右辺の計算の直前に成分を計算し直し, 行列ベクトル積の計算後に low rank block が保存されているメモリを解放する, という処理を行っている. この手法を用意した背景には, ACAstore は計算時間を低く抑えられるものの, その分かなりの記憶容量を要する点がある.

Table 1　ACAstore with $N_e = 7,680$.

$\varepsilon_{\mathrm{ACA}}$	Time(s)	MB	Rel. Err.
10^{-2}	933	2,634	8.1×10^{-4}
10^{-3}	1,027	3,267	1.4×10^{-4}
10^{-4}	1,116	3,779	7.4×10^{-6}
10^{-5}	1,203	4,312	1.5×10^{-6}
10^{-6}	1,493	4,858	1.4×10^{-6}

そこで, 計算時間の増加を多少許容する代わりに記憶容量を抑える事で, より大規模な問題を解くことを可能とするための検討を行う.

Fig.5(a), (b) から, CQ-BEM では計算時間, 記憶容量は $O(N_e^2)$ で増加しているのに対し, ACAstore では $O(N_e \log N_e)$ で増加している事がわかる. また, ACAstore は要素数が最も少ない $N_e = 960$ の場合でも, CQ-BEM より短い計算時間, 少ない記憶容量で計算が行われている. 一方, ACArelease は記憶容量に関しては ACAstore から更に削減されているが, その分計算時間は CQ-BEM と比較しても大幅に長くなってしまっている. これは, ACA による圧縮を行ったとしても, low rank block の再計算を N_t 回行う計算に時間が掛かるため, 少ない要素数の場合では CQ-BEM に勝ることが出来なかったと思われる. しかしながら, Fig.5(a) より ACArelease が CQ-BEM に比べて速くなる計算時間を見積もると, 概ね要素数 $N_e = 10^5$ 以上で, ACArelease の方が速くなると予想される. また, Fig.5(a), (b) では CQ-BEM および ACArelease は要素数 N_e がそれぞれ 7,680 および 15,360 以上で計算されていないが, これは CQ-BEM が記憶容量, ACArelease が計算時間の上限を超えてしまったためである. ACAstore についても, 記憶容量が上限に近いことから, これ以上 N_e を大きくすることは困難である.

Table 1 に, $N_e = 7,680$ とした場合の ACAstore による計算時間, 記憶容量, 圧縮後の係数行列 $\tilde{G}$ の相対誤差を, ACA の精度 $\varepsilon_{\mathrm{ACA}}$ 毎にまとめた. ただし, 圧縮前の係数行列 G に対する $\tilde{G}$ の相対誤差は $\|G - \tilde{G}\|_{\mathrm{F}} / \|\tilde{G}\|_{\mathrm{F}}$ と求めた. Table 1 から, $\varepsilon_{\mathrm{ACA}}$ を小さくすると圧縮率と記憶容量はほぼ $O(\log(\varepsilon_{\mathrm{ACA}}))$ で向上していることがわかる. 一方, $\varepsilon_{\mathrm{ACA}} = 10^{-3}$ 以上では相対誤差が悪化している. これは $\varepsilon'_{\mathrm{ACA}}$ を用いることで, 相対誤差の変化の度合いが増加したためと思われる.

今回用いた ACAstore, ACArelease はそれぞれ計算時間, 記憶容量の削減に関しては有用だが, 性能が極端であり, 計算資源を十分に活用しているとは言い難い. そのため, 両者のバランスを改善するために, 階層行列の再圧縮法の検討, 近似圧縮手法の改良, 保存する, しない部分行列の配分の見直し等を行う必要がある. 特に, 階層行列 ($\mathcal{H}$-matrices) の代わりに改善型の $\mathcal{H}^2$-matrices[11] を用いることで, 要素数 N_e に関しての計算コストを $O(N_e)$ に押し下げることができると思われる.

6. おわりに

本論文では, ACA を用いた CQ-BEM の計算効率化手法について述べた. ACA を用いた手法の二次元面外波動の数値計算結果では, 要素数が N_e の場合の計算時間と記憶容量を $O(N_e \log N_e)$ とすることができた. 今後は計算時間と記憶容量のバランスを改善した上で, より大規模の問題や異方性弾性波動問題へと適用していく予定である.

参考文献

(1) Lubich, C.: Convolution quadrature and discretized operational calculus I, *Numerische Mathematik*, **52**(2)(1988), pp. 129–145.

(2) 福井卓雄，岡山美央，石田貴之 : 2 次元波動伝播問題における演算子積分時間領域境界要素法および高速多重極法の適用，計算数理工学論文集，**6**(2)(2006)，pp. 153–158.

(3) 斎藤隆泰，廣瀬壮一，福井卓雄，石田貴之: 三次元スカラー波動および弾性波動問題における演算子積分時間領域境界要素法，応用力学論文集，土木学会，**10**(2007)，pp. 217–224.

(4) 斎藤隆泰，石田貴之，福井卓雄，廣瀬壮一 : 粘弾性面外波動問題における演算子積分時間領域境界要素法および高速多重極法の適用，計算工学論文集，(2008)，原稿番号 200803031.

(5) 斎藤隆泰，田中遊雲，廣瀬壮一 : 3 次元異方性弾性波動問題における Lubich の方法を用いた時間領域境界要素法，計算数理工学論文集，**10**(2010)，pp. 111–116.

(6) Rokhlin, V.: Rapid solution of integral equations of classical potential theory, *Journal of Computational Physics*, **60**(2)(1985), pp. 187–207.

(7) 福井卓雄，斎藤隆泰 : Lubich の演算子積分法における高速多重極法，日本シミュレーション学会論文誌，小特集 : 境界要素法の新展開，**28**(3)(2009)，pp. 17–22.

(8) 斎藤隆泰，瀬川尚揮，石田貴之，廣瀬壮一 : 並列化された演算子積分時間領域高速多重極境界要素法による大規模多重散乱解析，計算数理工学論文集，**11**(2011)，pp. 95–100.

(9) Bebendorf, M.: Approximation of boundary element matrices, *Numerische Mathematik*, **86**(2000), pp. 565–589.

(10) Kurz, S., Rain, O. and Rjasanow, S.: The adaptive cross approximation technique for the 3-D boundary element method, *IEEE Transactions on Magnetics*, **38**(2)(2002), pp. 421–424.

(11) Bebendorf, M.: Hierarchical matrices, *Springer*, (2008).

(12) 吉川仁，松浦亮介，西村直志 : Lubich の CQM を用いた時間域境界積分方程式法の ACA による高速化について，土木学会論文集 A2(応用力学)，**69**(2)(2013)，pp. I_187–I_193.

(13) Grasedyck, L.: Adaptive recompression of $\mathcal{H}$–matrices for BEM, *Computing*, **74**(3)(2005), pp. 205–223.