確率的固有値分布推定法におけるフィルタ特性および並列実装に関する研究

2016年3月

前田　恭行

確率的固有値分布推定法におけるフィルタ特性および並列実装に関する研究

前田　恭行

システム情報工学研究科
筑波大学

2016年3月

論文要旨

本論文では，固有値問題における固有値の分布を推定する方法に対して，フィルタ特性の解析による解法の性能比較および，非線形固有値問題への応用，特殊な領域に対する固有値数の推定，さらに並列実装による高速化を目標としている．本論文で扱う確率的固有値分布推定法は，固有値の分布を推定する方法の一つであり，有理式型推定法と多項式型推定法の 2 つが存在する．本論文では，2 つの推定法のフィルタ特性を解析することによって，これまで行われていなかった推定法の性能比較を行う．また，非線形固有値問題に対する確率的固有値分布推定法は提案されていないため，拡張された Smith の標準形を用いて，非線形固有値問題への拡張を行う．そして，有理式型推定法のフィルタ特性を変更することで，特殊な領域に対する固有値数が推定可能であることを示す．さらに，マスター・ワーカ方式並列処理を用いて，並列計算機での有理式型推定法の実装手法を示し，高速化を図る．

目 次

図 目 次

表 目 次

第1章　序論

現在，航空機や自動車等の構造解析による製品の設計・開発や，タンパク質やDNA等の機能解析による創薬，気象現象のシミュレーションによる，精密な天気予測・解明など，様々な分野において科学技術計算が必要とされている．そして科学技術計算では，コンピュータを用いた数値シミュレーションが盛んに行われている．この数値シミュレーションでは固有値問題という問題が現れることがあり，計算の規模が大規模になるにつれて，その求解には膨大な時間を要する．そのため，スーパーコンピュータなどの大規模並列環境での並列計算によって固有値問題の求解に要する時間を軽減させる必要がある．

固有値問題とは行列値関数 $F(\lambda)$ について，

$$F(\lambda)\boldsymbol{x} = \mathbf{0}, \quad (\lambda \in \mathbb{C},\ \boldsymbol{x} \in \mathbb{C}^n \neq \{\mathbf{0}\}),$$

を満たす固有値 λ と固有ベクトル $\boldsymbol{x}$（固有対）を求める問題である．固有値問題には標準固有値問題，一般化固有値問題，非線形固有値問題がある．標準固有値問題，一般化固有値問題はそれぞれ，

$$F(\lambda) = \lambda I - A, \quad F(\lambda) = \lambda B - A,$$

となる問題である．ここで，行列 $A, B \in \mathbb{C}^{n\times n}$ であり，I は n 次の単位行列である．また，非線形固有値問題は多項式固有値問題，指数型固有値問題，平方根型固有値問題，など様々なタイプが存在し，それらはそれぞれ，

$$F(\lambda) = A_0 + \lambda A_1 + \ldots + \lambda^p A_p, \quad F(\lambda) = A_0 + \lambda A_1 + \mathrm{e}^{\lambda} A_2, \quad F(\lambda) = A_0 + \lambda A_1 + \sqrt{\lambda} A_2,$$

で表される．ここで，行列 $A_0, A_1, \ldots, A_p$ は $n \times n$ の行列である．

様々な科学や工学の分野において現れる固有値問題には，一部の固有対のみが必要な問題がある．例として，量子ドットの電子状態計算で現れる固有値問題 [1] では，最小固有値から数個の固有対が必要となる．また，遅延微分方程式から現れる固有値問題 [2] では，実数部が正の値を持つ固有値が重要となる．また，ナノエレクトロニクスデバイスの構造解析から現

れる固有値問題 [3] では，複素平面上の特定の円の円周近傍の固有値およびそれに対応する固有ベクトルが必要となる．そのような固有値問題の解法としては，全固有値を求める方法と，一部の固有対を求める方法が存在する．全固有値を求める方法には，行列の変形を伴う方法を用いた QZ 法 [4, 5] が代表される，一部の固有対を求める方法には，shift-invert-Arnoldi 法 [6, 7] や Sakurai-Sugiura 法 [8, 9, 10]，FEAST 法 [11, 12, 13]，Spectrum Slicing 法 [14] などが代表される．特に，一部の固有対を求める方法は，不必要な固有対を求める必要がなくなるため，大規模固有値問題に対して有効な解法である．しかし，一部の固有対を求める方法は，求めたい固有値付近にどの程度固有値が存在するのかという，固有値の分布情報に基づいたパラメータ設定が必要であり，適切なパラメータを設定をしない場合，解の精度が低くなることがある．以上により，固有値問題において，固有値の分布を推定することは，固有値解法の適切なパラメータを設定などに利用することができるため重要である．また，遅延微分方程式から現れる固有値問題や，3 次元フォトニック結晶構造解析から現れる固有値問題 [15, 16] では，固有値が存在しない区間幅を求める必要となる．そのため，固有値の分布を推定する方法はそのような科学技術計算への応用が期待できる．

固有値分布を推定する方法には，Gershgorin 定理 [17, 18] や Sylvester 慣性則 [19]，Algebraic Sub-structuring method[20]，確率的固有値分布推定法 [21, 22] などがある．各手法について，Gershgorin 定理は，標準・一般化固有値問題に対する手法であり，全固有値が存在する範囲を推定する．しかし，この方法は固有値の密集情報を推定することができないため，固有値の粗密を求められないという問題がある．Sylvester 慣性則は，標準・一般化エルミート固有値問題に対する手法であり，指定した値よりも大きい，もしくは小さい固有値の数を求める．この方法は固有値数を厳密に求めることができるが，計算の過程で修正コレスキー分解 [19] という行列分解が必要となり，行列が大規模な疎行列である場合，適用が難しいという問題がある．AS 法を用いた方法は，標準・一般化エルミート固有値問題に対する手法であり，近似固有値・固有ベクトルを求めて固有値分布を推定する．この方法は，固有値の粗密を推定することができるが，Sylvester 慣性則と同様に計算の過程で修正コレスキー分解が必要となり，こちらも大規模な疎行列に対して適用が難しい．確率的固有値分布推定法は，指定した領域内部の固有値数を推定する手法である．この方法は行列分解を行わずに固有値の粗密を推定することができるため，大規模な疎行列に対して，適用が可能となる．本論文では，確率的固有値分布推定法について考える．

確率的固有値分布推定法には多項式型推定法 [22]，有理式型推定法 [21] の 2 つが存在する．多項式型推定法は標準・一般化エルミート固有値問題に対する手法であり，指定した区間内部の固有値数を推定する．多項式型推定法の計算過程では複数の行列・ベクトル積の計算を

行う．有理式型推定法は標準・一般化固有値問題に対する手法であり，指定した複素平面上の円領域内部の固有値数を推定する．有理式型推定法の計算過程では複数の線形方程式の求解を行う．この線形方程式の求解に反復法の一つである Krylov 部分空間反復法を用いることで，行列分解を行わずに固有値数を推定することが可能となる．また複数の線形方程式の計算は独立した計算が可能となるため，並列計算機によって線形方程式を並列に計算することで高速な計算が可能となる．

確率的固有値分布推定法の問題点として，多項式型推定法および有理式型推定法の計算過程が異なることから，計算コストによる比較が困難となっていることがあげられる．また，確率的固有値分布推定法を含め，前述した固有値の分布を推定する方法は，標準・一般化固有値問題を対象としており，非線形固有値問題に対する固有値分布を推定する方法が提案されていない．有理式型推定法の問題点として，前述したように，ナノエレクトロニクスデバイスの構造解析から現れる固有値問題では，特定の円の円周近傍の固有対が必要となる．そのような問題に対しては円周上の固有値の分布情報が必要となるため，有理式型推定法の適用が困難となっている．また有理式型推定法で現れる線形方程式に Krylov 部分空間反復法を用いて，並列環境で計算する場合，各計算機での計算時間ののばらつきが発生し，それによって並列効率の悪化が発生する．

以上のことから本論文では，2 つの推定法のフィルタ特性を解析することによって，これまで行われていなかった推定法の性能比較を行う．また，有理式型推定法に対して，拡張された Smith の標準形を用いて，非線形固有値問題への拡張を行う．そして，有理式型推定法のフィルタ特性を変更することで，特殊な領域に対する固有値数が推定可能であることを示す．さらに，マスター・ワーカ方式並列処理を用いて，並列計算機での有理式型推定法の効率的な実装手法を示し，高速化を図る．

1.1 本論文の構成

本論文の構成を以下に述べる．第 2 章では，先行研究である確率的固有値分布推定法およびその推定法によって作成されるフィルタ関数について述べる．第 3 章では，有理式型推定法および多項式型推定法のフィルタ特性を解析し，各推定法の性能比較を行う．第 4 章では，拡張された Smith の標準形を用いて，非線形固有値問題に対する有理式型推定法を提案する．第 5 章では，有理式型推定法のフィルタ特性を変更し，特殊な領域に対する有理式型推定法の拡張手法を提案する．第 6 章では，有理式型推定法を並列計算機で実装した場合の問題を示し，その問題を軽減させるためにマスター・ワーカ方式並列処理を用いた実装手法を示す．

最後に第 7 章で本論文で得られた結論をまとめる.

第2章　確率的固有値分布推定法

本章では先行研究である確率的固有値分布推定法について述べる．確率的固有値分布推定法には，有理式型推定法 [21] と多項式型推定法 [22] が存在する．本章では各手法の概要および各手法で作成されるフィルタ関数について述べる．

2.1　有理式型推定法

有理式型推定法は，複素平面上に任意に設定された閉曲線内部の固有値の数を数値積分によって計算する方法である．同手法は一般化固有値問題

$$F(\lambda)\boldsymbol{x} = (\lambda B - A)\boldsymbol{x} = \boldsymbol{0}, \quad (A, B \in \mathbb{C}^{n\times n}, \boldsymbol{x} \in \mathbb{C}^n \backslash \{\boldsymbol{0}\}, z \in \mathbb{C}),$$

を対象としている．

複素平面上に閉曲線 Γ を任意に置き，Γ 上の点を z とし，Γ 内の固有値の数を m_{R} とする．このとき，m_{R} は以下の周回積分により求めることができる．

$$m_{\mathrm{R}} = \frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} \mathrm{tr}\left((zB - A)^{-1} B \right) \mathrm{d}z. \tag{2.1}$$

ここで，tr は行列の対角要素の総和 (トレース) を表す．また，$(zB - A)$ は正則な行列束であるとする．

図 2.1 に有理式型推定法の概略図を示す．横軸，縦軸はそれぞれ実軸，虚軸を表している．緑色の点は固有値を表しており，橙色の領域は閉曲線 Γ とその内部の領域を表している．概略図では $m_{\mathrm{R}} = 4$ となる．

以下に，式 (2.1) により固有値数を求めることができることを示す．

行列 A および B に対し QZ 分解を行うことにより，以下のように表すことができる．

$$A = URW^{\mathrm{H}}, \quad B = UTW^{\mathrm{H}}.$$

ここで，U および W はユニタリ行列であり，T および R は上三角行列となる．このとき一般

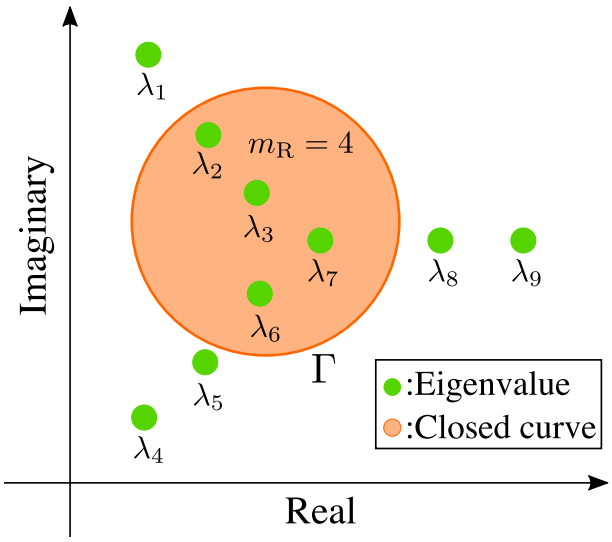

図 2.1: 有理式型推定法の概略図

化固有値問題の固有値 λ_i は，行列 T の対角要素 t_{ii} および行列 R の対角要素 r_{ii} を用いて，

$$\lambda_i = \begin{cases} \dfrac{r_{ii}}{t_{ii}}, & (t_{ii} \neq 0) \\ \infty, & (t_{ii} = 0) \end{cases},$$

と表すことができる．

式 (2.1) で現れる $\mathrm{tr}((zB - A)^{-1}B)$ に対して，以上の行列分解を用いることにより，

$$\mathrm{tr}\left((zB - A)^{-1} B\right) = \mathrm{tr}\left(\left(zUTW^{\mathrm{H}} - URW^{\mathrm{H}}\right)^{-1} UTW^{\mathrm{H}}\right),$$

と表せる．行列のトレースの性質により，

$$\mathrm{tr}\left(\left(zUTW^{\mathrm{H}} - URW^{\mathrm{H}}\right)^{-1} UTW^{\mathrm{H}}\right) = \mathrm{tr}\left((zT - R)^{-1} T\right) = \sum_{i=1}^{n} \frac{t_{ii}}{zt_{ii} - r_{ii}} = \sum_{i=1}^{n'} \frac{1}{z - \lambda_i},$$

と表すことができる．ここで n' は，$\lambda_i \neq \infty$ である固有値の数である．

複素平面上での周回積分について，定理 2.1 が成り立つ．

定理 2.1 (留数定理)**.** 複素平面内のある開領域 D で連続な複素関数を $f(z)$ とする．開領域 D 内に閉曲線 Γ を置き，その閉曲線内部で $f(z)$ が特異点 $\lambda_1, \lambda_2, \ldots, \lambda_m$ を持つとき，以下の式

が成り立つ．

$$\frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} f(z)\,\mathrm{d}z = \sum_{j=1}^{m} \left(\lim_{z \to \lambda_j} f(z)\,(z-\lambda_j) \right).$$

定理 2.1 を用いることで，以下の式が導出される．

$$\frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} \mathrm{tr}\left((zB-A)^{-1} B \right) \mathrm{d}z = \sum_{j=1}^{m_{\mathrm{R}}} \left(\lim_{z \to \lambda_j} \sum_{i=1}^{n'} \frac{1}{z-\lambda_i} (z-\lambda_j) \right) = m_{\mathrm{R}}.$$

以上により式 (2.1) は閉曲線 Γ 内部の固有値数を求めることができる．

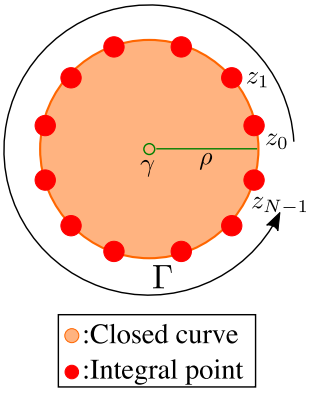

図 2.2: 積分点の配置

式 (2.1) の周回積分を計算機上で計算させるために，N 点台形則による近似計算を行う．これにより，式 (2.1) 式の m_{R} は $\hat{m}_{\mathrm{R}}$ に近似される．

$$m_{\mathrm{R}} \approx \hat{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\mathrm{tr}\left((z_j B - A)^{-1} B \right) \right). \tag{2.2}$$

ここで，z_j は中心 γ，半径 ρ の円 Γ 上に等間隔に N 個配置させた点であり，w_j は各積分点に対応した重みである．それらは以下の式で表される．

$$w_j = \frac{\rho}{N} \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}, \qquad z_j = \gamma + \rho \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}.$$

図 2.2 に，有理式型推定法における積分点 z_j の配置の概略図を示す．赤点が積分点を表している．

式 (2.2) について，$\mathrm{tr}((z_jB - A)^{-1}B)$ の計算は行列の逆行列の計算があるため計算コストが非常に高い．そこで，その計算を高速化するために行列のトレースの確率的推定 [23, 24] を用いる．

定理 2.2 (行列のトレースの確率的推定 [23, 24])**.** 行列 A を n 次対称行列とし，$\boldsymbol{v}$ を要素が等確率で 1 か -1 である n 次の確率変数ベクトルとする．このとき以下の式が成り立つ．

$$\mathrm{E}\left(\boldsymbol{v}^{\mathrm{T}}A\boldsymbol{v}\right) = \mathrm{tr}\left(A\right),$$
$$\mathrm{Var}\left(\boldsymbol{v}^{\mathrm{T}}A\boldsymbol{v}\right) = 2\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} a_{ij}.$$

ここで，$\mathrm{E}(\cdot)$ および $\mathrm{Var}(\cdot)$ はそれぞれ平均値および分散を表し，a_{ij} は行列 A の i 行 j 列の要素を表している．

定理 2.2 を用いることで，式 (2.2) は式 (2.3) で近似される．

$$\hat{m}_{\mathrm{R}} \approx \tilde{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\frac{1}{L}\sum_{\ell=1}^{L}\left(\boldsymbol{v}_\ell^{\mathrm{T}}\left(z_jB - A\right)^{-1}B\boldsymbol{v}_\ell\right)\right). \tag{2.3}$$

ここで，$\boldsymbol{v}_\ell$, $(\ell = 1, 2, \ldots, L)$ は要素がランダムで 1 か -1 となる L 本のベクトルである．式 (2.3) は $B\boldsymbol{v}_\ell$ を右辺ベクトルとした連立一次方程式

$$\left(z_jB - A\right)\boldsymbol{x}_{j,\ell} = B\boldsymbol{v}_\ell, \quad (j = 0, 1, \ldots, N-1, \quad \ell = 1, 2, \ldots, L),$$

を解き，その解ベクトルを用いて以下のように計算することができる．

$$\tilde{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\frac{1}{L}\sum_{\ell=1}^{L}\left(\boldsymbol{v}_\ell^{\mathrm{T}}\boldsymbol{x}_{j,\ell}\right)\right).$$

そのため，逆行列の計算をする必要が無く，L の数が $z_jB - A$ の次数よりも小さい場合，式 (2.2) の計算を行うよりも高速な計算が可能となる．

Algorithm 1 に有理式型推定法のアルゴリズムを示す．

有理式型推定法では，複数の閉曲線 $\Gamma_1, \Gamma_2, \ldots$ を配置し，各閉曲線に対して式 (2.3) の計算を行い固有値の数を推定することで，固有値分布を推定する．このとき閉曲線を多くとることでより詳細な固有値の分布を求めることができる．

図 2.3 および図 2.4 に閉曲線の配置例を示す．図 2.3 および図 2.4 では区間 $([a_0, b_0], [a_1, b_1])$,

Algorithm 1 有理式型推定法のアルゴリズム

Input: A, B, N, L, γ, ρ

Output: $\tilde{m}_{\mathrm{R}}$

set $\boldsymbol{v}_\ell$ of which the elements take 1 or -1 with equal probability for $\ell = 1, 2, \ldots, L$

$\tilde{m}_{\mathrm{R}} \leftarrow 0$

for $j = 0$ to $N - 1$ **do**

$z_j \leftarrow \gamma + \rho \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}$

$w_j \leftarrow \frac{\rho}{N} \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}$

for $\ell = 1$ to L **do**

solve $(z_j B - A)\boldsymbol{x}_{j,\ell} = B\boldsymbol{v}_\ell$ for $\boldsymbol{x}_{j,\ell}$

$\tilde{m}_{\mathrm{R}} \leftarrow \tilde{m}_{\mathrm{R}} + w_j(\boldsymbol{v}_\ell^{\mathrm{T}} \boldsymbol{x}_{j,\ell})/L$

end for

end for

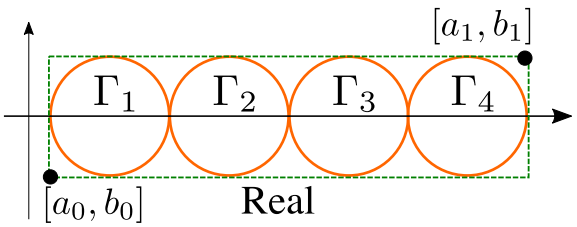

図 2.3: 閉曲線の配置例 ($K = 4$)

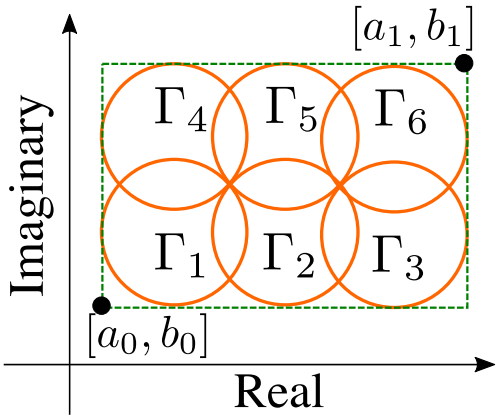

図 2.4: 閉曲線の配置例 ($K = 6$)

$(a_0, a_1, b_0, b_1 \in \mathbb{R})$ 内部に同じ大きさの閉曲線 $\Gamma_1, \Gamma_2, \ldots \Gamma_K$ を等間隔に配置している．全固有値が全て実数となる場合は図 2.3 のような配置が考えられ，固有値が複素平面上に存在する場合は図 2.4 のような配置が考えられる．

2.2 多項式型推定法

多項式型推定法は，任意に設定された区間内部の固有値の数を Chebyshev 多項式を用いて計算する方法である．同手法ではエルミート標準固有値問題および対称一般化固有値問題を対象としている．このとき，各固有値問題の全固有値は実数となる．

2.2.1 エルミート標準固有値問題における多項式型推定法

エルミート標準固有値問題

$$F(\lambda)\boldsymbol{x} = (\lambda I - A)\boldsymbol{x} = \mathbf{0}, \quad (A \in \mathbb{C}^{n\times n}, \boldsymbol{x} \in \mathbb{C}^n, z \in \mathbb{C}), \quad A : \text{エルミート行列}, \ I : \text{単位行列}$$

において，任意の実数区間 $[a, b]$, $(a, b \in \mathbb{R})$ を置き，その区間内部の固有値数を m_{P} とする．このとき，m_{P} は以下の式で求めることができる．

$$m_{\mathrm{P}} = \sum_{i=1}^{n} h_{\mathrm{P}}(\lambda_i) = \mathrm{tr}(h_{\mathrm{P}}(A)). \tag{2.4}$$

ここで，$h_{\mathrm{P}}(\cdot)$ は区間 $[a, b]$ 内部では 1 となり外部では 0 となるフィルタ関数である．

$$h_{\mathrm{P}}(\lambda) = \begin{cases} 1, & (a \le \lambda \le b) \\ 0, & (\text{otherwise}) \end{cases}. \tag{2.5}$$

式 (2.4) について，固有値の計算は計算コストが高い．そのためフィルタ関数を Chebyshev 多項式を用いて近似する．これにより，フィルタ関数は以下の式に近似される．

$$h_{\mathrm{P}}(A) \approx \sum_{j=0}^{p} \zeta_j(a, b) T_j(A). \tag{2.6}$$

ここで，$T_j(A)$ は j 次の第 1 種 Chebyshev 多項式

$$T_j(A) = \begin{cases} I, & (j=0) \\ A, & (j=1) \\ 2AT_{j-1}(A) - T_{j-2}(A), & (j>1) \end{cases},$$

であり，$\zeta_j(\cdot)$ は Chebyshev 多項式に対する重みである．

$$\zeta_j(a,b) = \begin{cases} \dfrac{\arccos(a) - \arccos(b)}{\pi}, & (j=0) \\ \dfrac{2(\sin(j\arccos(a)) - \sin(j\arccos(b)))}{j\pi}, & (j>0) \end{cases}. \tag{2.7}$$

ここで，A の全固有値は $[-1,1]$ に存在している場合を想定している．そのため，行列 A の全固有値が $[-1,1]$ に存在するように，A, a, b に対してマッピングを行う．

$$M(A) = \frac{2A - (\lambda_n + \lambda_1)I}{\lambda_n - \lambda_1}, \quad M(a) = \frac{2a - \lambda_n + \lambda_1}{\lambda_n - \lambda_1}, \quad M(b) = \frac{2b - \lambda_n + \lambda_1}{\lambda_n - \lambda_1}.$$

ここで，λ_n は最大固有値，λ_1 は最小固有値である．

以上により式 (2.4) を以下の式で近似することで，固有値数を近似計算することができる．

$$m_{\mathrm{P}} \approx \hat{m}_{\mathrm{P}} = \mathrm{tr}\left(\sum_{j=0}^{p} \zeta_j(M(a), M(b)) T_j(M(A))\right). \tag{2.8}$$

式 (2.8) では，全固有値の計算が必要ないため計算コストを抑えることができる．

式 (2.8) について，計算コストが高い部分は第 1 種 Chebyshev 多項式 $T_j(A)$ の計算で現れる行列・行列積の計算部分である．そこで，その計算を抑えるために前節と同様に行列のトレースの確率的推定 [23, 24] を用いる．それにより，$\hat{m}_{\mathrm{P}}$ は式 (2.9) で近似できる．

$$\hat{m}_{\mathrm{P}} \approx \tilde{m}_{\mathrm{P}} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{p} \zeta_j(M(a), M(b)) \left(\boldsymbol{v}_\ell^{\mathrm{T}} T_j(M(A)) \boldsymbol{v}_\ell\right) = \frac{1}{L}\sum_{\ell=1}^{L} \boldsymbol{v}_\ell^{\mathrm{T}} P_p(A) \boldsymbol{v}_\ell. \tag{2.9}$$

ここで，$\boldsymbol{v}_\ell, (\ell = 1, 2, \ldots, L)$ は要素がランダムで 1 か -1 となる L 本のベクトルであり，$P_p(A)$

は p 次の多項式である．式 (2.9) では，$T_j(M(A))$ の計算の代わりに以下の行列・ベクトル積

$$\boldsymbol{w}_{j,\ell} = T_j(M(A))\boldsymbol{v}_\ell = \begin{cases} \boldsymbol{v}_\ell, & (j=0) \\ M(A)\boldsymbol{v}_\ell, & (j=1) \\ 2M(A)\boldsymbol{w}_{j-1,\ell} - \boldsymbol{w}_{j-2,\ell}, & (j>1) \end{cases}, \tag{2.10}$$

を計算し，得られたベクトル $\boldsymbol{w}_j$ を用いて以下のように計算することができる．

$$\tilde{m}_{\mathrm{P}} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{p}\zeta_j(M(a),M(b))\left(\boldsymbol{v}_\ell^{\mathrm{T}}\boldsymbol{w}_{j,\ell}\right).$$

そのため，行列・行列積の計算をする必要が無く，L の数が行列の次数よりも小さい場合，式 (2.8) の計算を行うよりも高速な計算が可能となる．

Algorithm 2 に標準固有値問題における多項式型推定法のアルゴリズムを示す．

Algorithm 2 エルミート標準固有値問題における多項式型推定法のアルゴリズム

Input: A, a, b, P, L
Output: $\tilde{m}_{\mathrm{P}}$
  set $\boldsymbol{v}_\ell$ of which the elements take 1 or -1 with equal probability for $\ell = 1, 2, \ldots, L$
  compute λ_1, λ_n
  $\tilde{m}_{\mathrm{P}} \leftarrow 0$
  for $j = 0$ to p **do**
    if $j = 0$ **then**
      $\zeta_j(M(a),M(b)) \leftarrow \frac{\arccos(M(a))-\arccos(M(b))}{\pi}$
    else
      $\zeta_j(M(a),M(b)) \leftarrow \frac{2(\sin(j\arccos(M(a)))-\sin(j\arccos(M(b))))}{j\pi}$
    end if
    for $\ell = 1$ to L **do**
      if $j = 0$ **then**
        $\boldsymbol{w}_{j,\ell} \leftarrow \boldsymbol{v}_\ell$
      else if $j = 1$ **then**
        $\boldsymbol{w}_{j,\ell} \leftarrow M(A)\boldsymbol{v}_\ell$
      else
        $\boldsymbol{w}_{j,\ell} \leftarrow 2M(A)\boldsymbol{w}_{j-1,\ell} - \boldsymbol{w}_{j-2,\ell}$
      end if
      $\tilde{m}_{\mathrm{P}} \leftarrow \tilde{m}_{\mathrm{P}} + \zeta_j(M(a),M(b))(\boldsymbol{v}_\ell^{\mathrm{T}}\boldsymbol{w}_{j,\ell})/L$
    end for
  end for

2.2.2 対称一般化固有値問題における多項式型推定法

対称一般化固有値問題

$$F(\lambda)\boldsymbol{x} = (\lambda B - A)\boldsymbol{x} = \mathbf{0}, \quad (A, B \in \mathbb{C}^{n\times n}, \boldsymbol{x} \in \mathbb{C}^n, z \in \mathbb{C}), \quad A : \text{対称行列}, \ B : \text{正定値対称行列}$$

において，多項式型推定法行う場合，2 つの行列 A, B が現れるため，式 (2.4) で表すことができない．

このような問題に対して，行列 B が正定値対称行列であることからコレスキー分解を用いて，

$$B = LL^{\mathrm{T}}$$

とする．ここで，L は下三角行列である．このとき，対称一般化固有値問題を以下の標準固有値問題

$$(\lambda LL^{\mathrm{T}} - A)\boldsymbol{x} = \mathbf{0} \Rightarrow (\lambda I - L^{-1}AL^{-\mathrm{T}})(L^{\mathrm{T}}\boldsymbol{x}) = \mathbf{0}$$

に帰着させることで，対称標準固有値問題における多項式型推定法を適用することが可能となる．しかしこの場合，行列 B に対する行列分解を行う必要がある．そのため，行列が大規模疎行列の場合，行列分解の計算コストが高い．

以下では対称一般化固有値問題に対して，行列分解を行わずに固有値分布を推定する方法について述べる．

対称一般化固有値問題において，定理 2.3 が成り立つ．

定理 2.3 (Sylvester 慣性則 [19])**.** 任意の行列 A および X がそれぞれ対称行列，非特異行列とする．この時，行列 A の正および負の固有値の数は，行列 $X^{\mathrm{T}}AX$ のそれらと一致する．

補題 2.3 より，任意の値 $\sigma \in \mathbb{R}$ を用いた行列 $A - \sigma B$ の正および負の固有値の数は，行列 $L^{-1}(A - \sigma B)L^{-\mathrm{T}}$ のそれらと一致することがわかる．$B = LL^{\mathrm{T}}$ であることから，

$$L^{-1}(A - \sigma B)L^{-\mathrm{T}} = L^{-1}AL^{-\mathrm{T}} - \sigma I$$

となる．これにより $A - \sigma B$ の正および負の固有値数を推定することで，$L^{-1}AL^{-\mathrm{T}} - \sigma I$ のそれらを推定することができる．そのため，対称一般化固有値問題における区間 $[a, b]$ 内部の固有値は行列 $A - aB$ の正の固有値数および $A - bB$ の正の固有値数を推定し，その差分を取

ることで求めることができる．

行列 $\sigma B - A$ に対して正の固有値数を $m_{\mathrm{P}}^{(\sigma)}$ とする．$m_{\mathrm{P}}^{(\sigma)}$ は以下の式により近似することができる．

$$m_{\mathrm{P}}^{(\sigma)} \approx \hat{m}_{\mathrm{P}}^{(\sigma)} = \mathrm{tr}\left(\sum_{j=0}^{p} \zeta_j(0,1) T_j(M(A-\sigma B))\right).$$

行列のトレースの確率的推定を用いることで $\hat{m}_{\mathrm{P}}^{(\sigma)}$ は $\tilde{m}_{\mathrm{P}}^{(\sigma)}$ に近似される．

$$\hat{m}_{\mathrm{P}}^{(\sigma)} \approx \tilde{m}_{\mathrm{P}}^{(\sigma)} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{p} \zeta_j(0,1)\left(\boldsymbol{v}_\ell^{\mathrm{T}} T_j(M(A-\sigma B))\boldsymbol{v}_\ell\right).$$

対称一般化固有値問題において，区間 $[a,b]$ 内部の固有値数を m_{P} とする．このとき，m_{P} は以下の式で近似計算できる．

$$m_{\mathrm{P}} \approx \tilde{m}_{\mathrm{P}} = \tilde{m}_{\mathrm{P}}^{(b)} - \tilde{m}_{\mathrm{P}}^{(a)}. \tag{2.11}$$

多項式型推定法では，複数の区間 $[a_1,b_1],[a_2,b_2]\ldots$ を配置し，各区間に対して式 (2.9) もしくは式 (2.11) の計算を行い固有値の数を推定することで，固有値分布を推定する．このとき区間を多くとることでより詳細な固有値の分布を求めることができる．Algorithm 3 に一般化固有値問題における多項式型推定法のアルゴリズムを示す.

2.3 固有値分布推定法におけるフィルタ関数

多項式型推定法では，フィルタ関数を多項式近似して固有値数を推定している．有理式型推定法では，周回積分を積分近似することで固有値数を推定しており，その計算ではフィルタ関数を有理式近似している．本節では有理式型および多項式型推定法で用いられているフィルタ関数について述べる．

Algorithm 3 対称一般化固有値問題における多項式型推定法のアルゴリズム

Input: A, a, b, P, L

Output: $\tilde{m}_{\mathrm{P}}$

set $\boldsymbol{v}_\ell$ of which the elements take 1 or -1 with equal probability for $\ell = 1, 2, \ldots, L$
compute λ_1, λ_n
$\tilde{m}_{\mathrm{P}}^{(a)} \leftarrow 0$
$\tilde{m}_{\mathrm{P}}^{(b)} \leftarrow 0$
for $j = 0$ to p **do**
  if $j = 0$ **then**
    $\zeta_j(0,1) \leftarrow \frac{\arccos(0) - \arccos(1)}{\pi}$
  else
    $\zeta_j(0,1) \leftarrow \frac{2(\sin(j \arccos(0)) - \sin(j \arccos(1)))}{j\pi}$
  end if
  for $\ell = 1$ to L **do**
    if $j = 0$ **then**
      $\boldsymbol{w}_{j,\ell}^{(a)} \leftarrow \boldsymbol{v}_\ell$
      $\boldsymbol{w}_{j,\ell}^{(b)} \leftarrow \boldsymbol{v}_\ell$
    else if $j = 1$ **then**
      $\boldsymbol{w}_{j,\ell}^{(a)} \leftarrow M(A - aB)\boldsymbol{v}_\ell$
      $\boldsymbol{w}_{j,\ell}^{(b)} \leftarrow M(A - bB)\boldsymbol{v}_\ell$
    else
      $\boldsymbol{w}_{j,\ell}^{(a)} \leftarrow 2M(A - aB)\boldsymbol{w}_{j-1,\ell}^{(a)} - \boldsymbol{w}_{j-2,\ell}^{(a)}$
      $\boldsymbol{w}_{j,\ell}^{(b)} \leftarrow 2M(A - bB)\boldsymbol{w}_{j-1,\ell}^{(b)} - \boldsymbol{w}_{j-2,\ell}^{(b)}$
    end if
    $\tilde{m}_{\mathrm{P}}^{(a)} \leftarrow \tilde{m}_{\mathrm{P}}^{(a)} + \zeta_j(0,1)(\boldsymbol{v}_\ell^{\mathrm{T}} \boldsymbol{w}_{j,\ell}^{(a)})/L$
    $\tilde{m}_{\mathrm{P}}^{(b)} \leftarrow \tilde{m}_{\mathrm{P}}^{(b)} + \zeta_j(0,1)(\boldsymbol{v}_\ell^{\mathrm{T}} \boldsymbol{w}_{j,\ell}^{(b)})/L$
  end for
end for
$\tilde{m}_{\mathrm{P}} \leftarrow \tilde{m}_{\mathrm{P}}^{(b)} - \tilde{m}_{\mathrm{P}}^{(a)}$

2.3.1 有理式型推定法におけるフィルタ関数

有理式型推定法のフィルタ関数について述べる．2.1 章と同様に，有理式型推定法において，式 (2.2) に対して QZ 分解を行う．これにより，

$$\hat{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\mathrm{tr} \left((z_j B - A)^{-1} B \right) \right) = \sum_{j=0}^{N-1} w_j \sum_{i=1}^{n'} \frac{1}{z_j - \lambda_i} = \sum_{i=1}^{n'} f(\lambda_i). \tag{2.12}$$

と表せる．ここで，

$$f(\lambda_i) = \sum_{j=0}^{N-1} \frac{w_j}{z_j - \lambda_i}$$

である．式 (2.12) により，有理式型推定法は，各固有値 λ_i に対して，フィルタ関数 $f(\lambda_i)$ を行っているとみなせる．

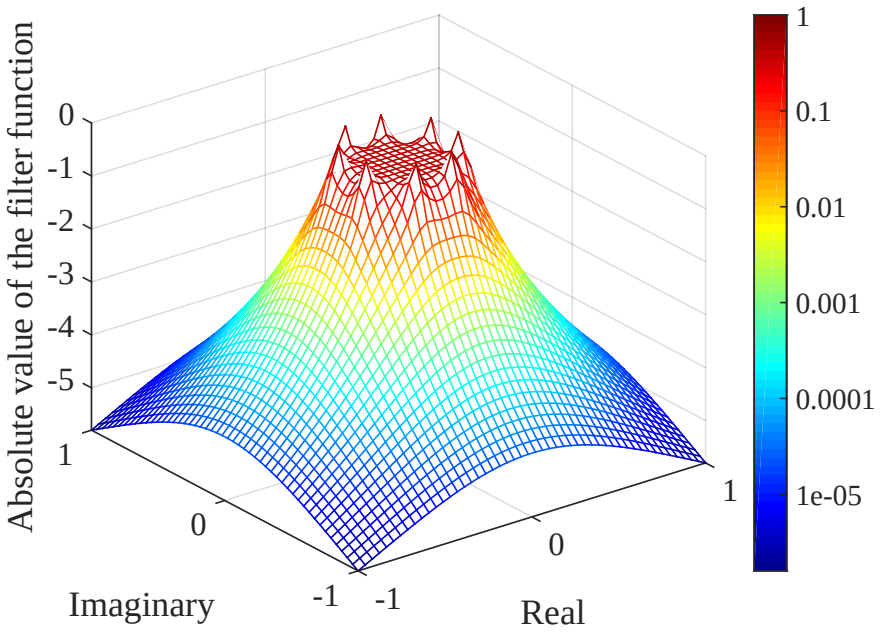

図 2.5: 複素平面上での有理式型推定法のフィルタ関数 ($N = 8$)

図 2.5 および図 2.6 に有理式型推定法におけるフィルタ関数 $f(\lambda_i)$ を示す．閉曲線 Γ を $\gamma = 0,\ \rho = 0.3$ としてフィルタ関数 $f(\lambda_i)$ を計算した．$N = 8$ とする．図 2.5 について，右下の軸は実数軸，左下の軸は虚数軸，縦軸はフィルタ関数値を表しており，フィルタ関数 $f(\lambda_i)$ の

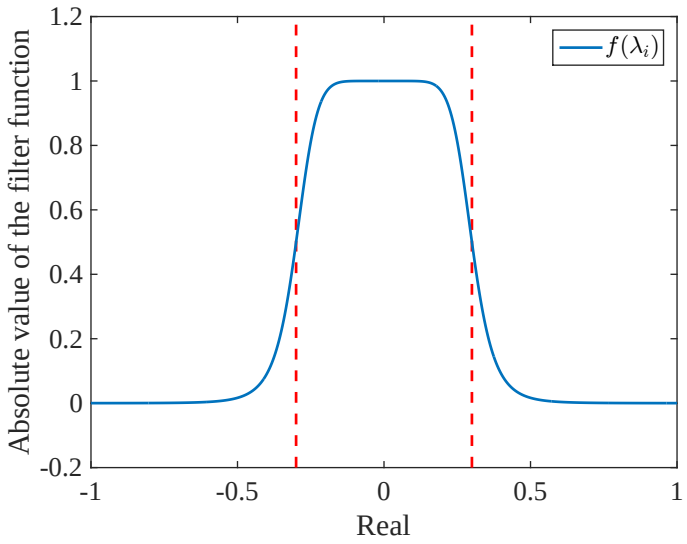

図 2.6: 実軸上での有理式型推定法のフィルタ関数 ($N = 8$)

値をメッシュで示す．図 2.6 について，横軸は実数軸，縦軸はフィルタ関数値を表している．閉曲線の両端 $\gamma + \rho$ および $\gamma - \rho$ を赤点線で示し，フィルタ関数 $f(\lambda_i)$ の値を青線で示す．図 2.5 および図 2.6 から，フィルタ関数 $f(\lambda_i)$ が閉曲線 Γ 内部では 1 となり，外部では 0 となることがわかる．

2.3.2 多項式型推定法におけるフィルタ関数

多項式型推定法のフィルタ関数について述べる．エルミート標準固有値問題における多項式推定法に対して，固有値分解を行う．これにより，

$$A = U \Lambda U^{\mathrm{H}}$$

となる．ここで，U はユニタリ行列であり，Λ は対角要素が固有値となる対角行列である．式 (2.8) に対して固有値分解を行うと，

$$\hat{m}_{\mathrm{P}} = \mathrm{tr}\left(\sum_{j=0}^{p} \zeta_j(a,b) T_j(M(A)) \right) = \mathrm{tr}\left(U \sum_{j=0}^{p} \zeta_j(a,b) T_j(M(\Lambda)) U^{\mathrm{H}} \right) = \sum_{i=1}^{n} P_p(\lambda_i) \quad (2.13)$$

となる．ここで，

$$P_p(\lambda_i) = \sum_{j=0}^{p} \zeta_j(a,b) T_j(M(\lambda_i))$$

である．式 (2.13) により，多項式型推定法は，各固有値 λ_i に対して，フィルタ関数 $P_p(\lambda_i)$ を行っているとみなせる．

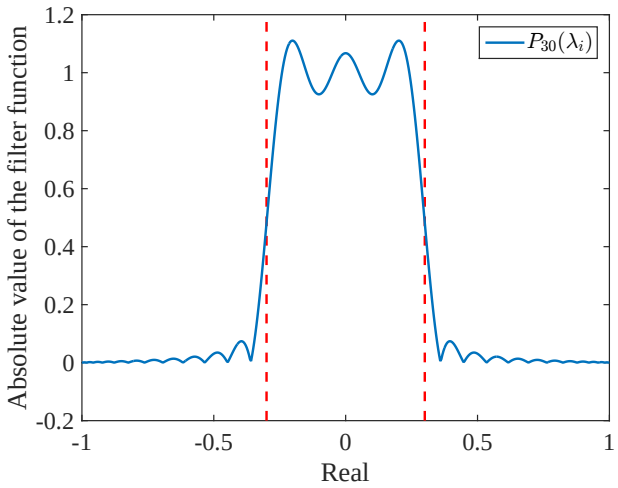

図 2.7: 多項式型推定法のフィルタ関数 ($p = 30$)

図 2.7 に多項式型推定法におけるフィルタ関数 $P_p(\lambda_i)$ を示す．実数区間 $[a,b] = [-0.3, 0.3]$ としてフィルタ関数 $P_p(\lambda_i)$ を作成した．$p = 30$ とする．図 2.7 について．横軸は実数軸，縦軸はフィルタ関数値を表している．区間の端点である a および b を赤点線で示し，フィルタ関数 $P_p(\lambda_i)$ の値を青線で示す．図 2.7 から，フィルタ関数 $P_p(\lambda_i)$ が区間 $[a,b]$ 内部では 1 となり，外部では 0 となることがわかる．

図 2.7 から，フィルタ関数 $P_p(\lambda_i)$ に振動が発生していることがわかる．これは，$P_p(\lambda_i)$ の計算に第 1 種 Chebyshev 多項式を用いており，第 1 種 Chebyshev 多項式から発生する Gibbs 振動が影響していると考えられる．$P_p(\lambda_i)$ の振動によって，指定区間外部の固有値が 0 に減衰しなくなるため，振動は少ないほうが望ましい．

Gibbs 振動に対して，以下の Jackson 係数を加えることによって振動を抑えることができる

[25].

$$J_{j,p} = \frac{\left(1 - \frac{j}{p+2}\right)\sin(\alpha_p)\cos(j\alpha_p) + \frac{1}{p+2}\cos(\alpha_p)\sin(j\alpha_p)}{\sin(\alpha_p)}, \quad \alpha_p = \frac{\pi}{p+2}$$

これにより，以下に示す Jackson 係数を加えたフィルタ関数 $P_p^{(\mathrm{J})}(\lambda_i)$ が得られる．

$$P_p^{(\mathrm{J})}(\lambda_i) = \sum_{j=0}^{p} J_{j,p}\zeta_j(a,b)T_j(M(\lambda_i))$$

また，以下に示す Jackson 係数を加えた多項式型推定法による推定値 $\hat{m}_{\mathrm{P}}^{(\mathrm{J})}$ が得られる．

$$\hat{m}_{\mathrm{P}}^{(\mathrm{J})} = \sum_{i=1}^{n} P_p^{(\mathrm{J})}(\lambda_i) \tag{2.14}$$

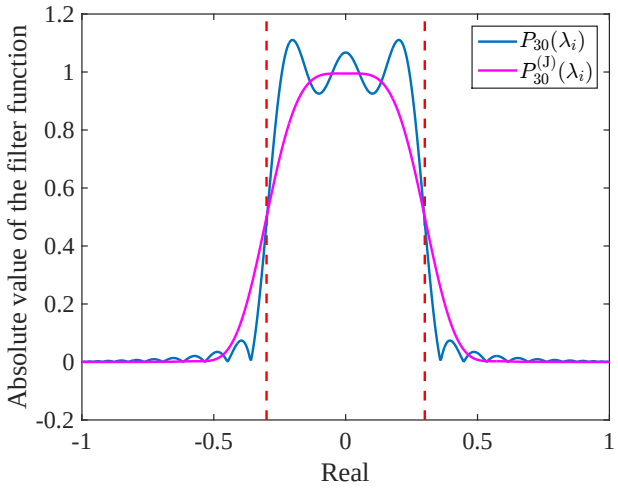

図 2.8: 多項式型推定法+Jackson 係数のフィルタ関数 ($p = 30$)

図 2.8 に多項式型推定法における Jackson 係数を加えたフィルタ関数 $P_p^{(\mathrm{J})}(\lambda_i)$ を示す．実数区間 $[a,b] = [-0.3, 0.3]$，$p = 30$ としてフィルタ関数 $P_p^{(\mathrm{J})}(\lambda_i)$ を作成した．図 2.8 について．横軸は実数軸，縦軸はフィルタ関数値を表している．区間の端点である a および b を赤点線で示し，フィルタ関数 $P_p(\lambda_i)$ の値を青線で示し，フィルタ関数 $P_p^{(\mathrm{J})}(\lambda_i)$ の値を紫線で示す．図 2.8 から，フィルタ関数 $P_p^{(\mathrm{J})}(\lambda_i)$ では $P_p(\lambda_i)$ で発生した振動が取り除かれていることがわかる．しかし，$P_p^{(\mathrm{J})}(\lambda_i)$ は指定した区間内部 $[-0.3, 0.3]$ でも減衰している．そのため，Jackson 係数を加えた多項式型推定法による推定値は，実際の固有値数より小さくなる可能性がある．

第3章　有理式型推定法におけるKrylov部分空間反復法を利用した多項式表現

2章では固有値の分布を求める方法の一つである確率的固有値分布推定法について述べた．本章では確率的固有値分布推定法の一つである有理式型推定法に対するKrylov部分空間反復法を利用した多項式表現について述べる．さらに，多項式型推定法と有理式型推定法を比較することで各手法の有効性について検証する．

3.1　序説

本章では標準エルミート固有値問題

$$F(\lambda)\boldsymbol{x} = (\lambda I - A)\boldsymbol{x} = \boldsymbol{0}, \quad (A \in \mathbb{C}^{n\times n}, \boldsymbol{x} \in \mathbb{C}^n, z \in \mathbb{C}), \quad A : \text{エルミート行列}, \ I : \text{単位行列}$$

における固有値の分布計算について考える．

2章で述べたように，固有値の分布を求める方法の一つに確率的固有値分布推定法がある．確率的固有値分布推定法には有理式型推定法と多項式型推定法の2つがあり，どちらも求めたい固有値成分のみを透過するフィルタ関数を作成し，固有値数を推定している．有理式型推定法および多項式型推定法のフィルタ関数はそれぞれ，有理関数近似および多項式近似によって作成しており，関数の近似にかかる計算コストがそれぞれ異なる．そのため，有理式型推定法と多項式型推定法のフィルタ関数に対する計算コストの比較を行うことが困難となっている．

2.1節で述べたように，有理式型推定法では計算過程で複数の線形方程式の求解が必要となる．線形方程式に対する解法として，Gaussの消去法やLU分解法[19]に代表される直接解法と，Krylov部分空間反復法[26]に代表される反復解法がある．特にKrylov部分空間反復法は反復計算によって線形方程式の近似解列を生成し，その近似解は多項式で表現できる．そのため，有理式型推定法で現れる複数の線形方程式に対してKrylov部分空間反復法を利用することで，有理式型推定法を多項式で表現することが可能となる．これにより，有理式型推

定法におけるフィルタ関数は多項式近似で表すことが可能となるため，多項式型推定法とのフィルタ関数に対する計算コストの比較を行うことが容易となる．本章では，有理式型推定法に対して，Krylov 部分空間反復法を利用した多項式表現を行い，多項式型推定法との比較を行う．

本章ではまず，Krylov 部分空間反復法および Krylov 部分空間反復法によって得られる近似解の多項式表現について述べる．次に，有理式型推定法に対して，Krylov 部分空間反復法を利用した多項式表現について述べる．さらに，数値実験によって多項式型推定法とのフィルタ関数の比較を行い，各手法の有効性を検証する．

3.2 Krylov 部分空間反復法

本節では線形方程式

$$A\boldsymbol{x} = \boldsymbol{b}, \quad (A \in \mathbb{C}^{n\times n},\ \boldsymbol{b}, \boldsymbol{x} \in \mathbb{C}^n),$$

の近似解を求める反復解法の一つである Krylov 部分空間反復法について述べる．ここで，A は正則な行列であり，$\boldsymbol{b}$ は既知のベクトルである．

反復解法とは，任意の初期解 $\boldsymbol{x}_0$ を基に，反復計算を行い，それによって近似解 $\boldsymbol{x}_i,\ i = 1, 2, \ldots$ を更新し，近似解を真の解 $\boldsymbol{x}$ に近づける解法である．反復解法の一つである Krylov 部分空間反復法では，k 反復目の近似解 $\boldsymbol{x}_k$ に対して，以下の条件を満たすように求める．

$$\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k(A; \boldsymbol{r}_0). \tag{3.1}$$

ここで，$\boldsymbol{x}_0$ は初期解，$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ は初期残差ベクトルを表し，$\mathcal{K}_k(A; \boldsymbol{r}_0)$ は行列 A および初期残差ベクトル $\boldsymbol{r}_0$ によって生成される以下の Krylov 部分空間を表す．

$$\mathcal{K}_k(A; \boldsymbol{r}_0) \equiv \mathrm{span}(\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0).$$

以上の条件によって $\boldsymbol{x}_k$ および $\boldsymbol{r}_k$ は式 (3.2) および式 (3.3) で表される．

$$\boldsymbol{x}_k = \boldsymbol{x}_0 + \sum_{i=0}^{k-1} h_{ik} A^i \boldsymbol{r}_0 = \boldsymbol{x}_0 + S_{k-1}(A)\boldsymbol{r}_0, \tag{3.2}$$

$$\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{b} - A\left(\boldsymbol{x}_0 + \sum_{i=0}^{k-1} h_{ik} A^i \boldsymbol{r}_0\right) = \boldsymbol{r}_0 - \sum_{i=0}^{k-1} h_{ik} A^{i+1} \boldsymbol{r}_0 = C_k(A)\boldsymbol{r}_0. \tag{3.3}$$

ここで，$h_{ik} \in \mathbb{C}$ は定数であり，$S_{k-1}(A)$ および $C_k(A)$ はそれぞれ $k-1$ 次の多項式および k 次の多項式である．特に，$C_k(A)$ は残差多項式と呼ばれ，式 (3.4) で表される．

$$I - C_k(A) = AS_{k-1}(A). \tag{3.4}$$

以上によって近似解 $\boldsymbol{x}_k$ および残差ベクトル $\boldsymbol{r}_k$ を更新する．しかし，以上の式では近似解 $\boldsymbol{x}_k$ および残差ベクトル $\boldsymbol{r}_k$ が一意に定まらないため，条件を付加する必要がある．このとき，近似解 $\boldsymbol{x}_k$ もしくは残差ベクトル $\boldsymbol{r}_k$ に対する条件のかけ方に応じて，様々な解法に変化する．本論文では Krylov 部分空間反復法の一つである BiCG(Biconjugate Gradient) 法 [27] および COCG(Conjugate Orthogonal Conjugate Gradient) 法 [28] について述べる．

3.2.1 BiCG 法

BiCG 法は，非エルミート行列 $A \neq A^{\mathrm{H}}$ に対する線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解く数値解法である．BiCG 法では線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の他にもう一つ別の線形方程式

$$A^{\mathrm{H}}\boldsymbol{x}^* = \boldsymbol{b}^*,$$

に対する Krylov 部分空間を考える．ここで，$\boldsymbol{b}^* \in \mathbb{C}^n$ は既知のベクトルである．

BiCG 法では線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の残差ベクトル $\boldsymbol{r}_k$ に対して，以下の双直交条件 (Petrov-Galerkin 条件) を課すことで近似解 $\boldsymbol{x}_k$ を一意に定めている．

$$\boldsymbol{r}_k \perp \mathcal{K}_k(A^{\mathrm{H}}; \boldsymbol{r}_0^*). \tag{3.5}$$

ここで，$\mathcal{K}_k(A^{\mathrm{H}}; \boldsymbol{r}_0^*)$ は，線形方程式 $A^{\mathrm{H}}\boldsymbol{x}^* = \boldsymbol{b}^*$ の行列 A^{H} および初期残差ベクトル $\boldsymbol{r}_0^*$ によって生成される Krylov 部分空間である．

以下に，BiCG 法による k 反復目の近似解 $\boldsymbol{x}_k$ の計算について述べる．Krylov 部分空間 $\mathcal{K}_k(A; \boldsymbol{r}_0)$ を張る正規直交ベクトル列を $\{\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_{k-1}\}$ とする．また，Krylov 部分空間 $\mathcal{K}_k(A^{\mathrm{H}}; \boldsymbol{r}_0^*)$ を張る正規直交ベクトル列を $\{\boldsymbol{r}_0^*, \boldsymbol{r}_1^*, \ldots, \boldsymbol{r}_{k-1}^*\}$ とする．このとき，式 (3.5) から，それぞれの直交ベクトル列は Gram-Schmidt の直交化法を用いて式 (3.6) と表すことができる．

$$\boldsymbol{r}_{k+1} = \alpha_k \left(\sum_{i=0}^{k} \frac{(\boldsymbol{r}_i^*, A\boldsymbol{r}_k)}{(\boldsymbol{r}_i^*, \boldsymbol{r}_i)} \boldsymbol{r}_i - A\boldsymbol{r}_k \right), \quad \boldsymbol{r}_{k+1}^* = \bar{\alpha}_k \left(\sum_{i=0}^{k} \frac{(\boldsymbol{r}_i, A^{\mathrm{H}}\boldsymbol{r}_k^*)}{(\boldsymbol{r}_i, \boldsymbol{r}_i^*)} \boldsymbol{r}_i^* - A^{\mathrm{H}}\boldsymbol{r}_k^* \right). \tag{3.6}$$

ここで，$(\boldsymbol{x}, \boldsymbol{y})$ はベクトル $\boldsymbol{x}$ と $\boldsymbol{y}$ の内積を表し，$\bar{\cdot}$ は複素共役を表す．また，$\alpha_k, \bar{\alpha}_k \in \mathbb{C}$ はそ

れぞれ $\boldsymbol{r}_{k+1}, \boldsymbol{r}^*_{k+1}$ を正規化するためのパラメータである．

次に式 (3.6) が 3 項間漸化式で表せることについて示す．式 (3.6) の内積計算について以下の式が成り立つ．

$$(\boldsymbol{r}^*_i, A\boldsymbol{r}_k) = (A^{\mathrm{H}}\boldsymbol{r}^*_i, \boldsymbol{r}_k), \quad (\boldsymbol{r}_i, A^{\mathrm{H}}\boldsymbol{r}^*_k) = (A\boldsymbol{r}_i, \boldsymbol{r}^*_k).$$

また，式 (3.6) から $A^{\mathrm{H}}\boldsymbol{r}^*_i$ は $\{\boldsymbol{r}^*_0, \boldsymbol{r}^*_1, \ldots, \boldsymbol{r}^*_{i+1}\}$ の線形結合で表すことができる．さらに，$\boldsymbol{r}_k$ は式 (3.5) の双直交条件を満たすことから，以下の式が成り立つ．

$$(A^{\mathrm{H}}\boldsymbol{r}^*_i, \boldsymbol{r}_k) = \begin{cases} 0, & (i = 1, 2, \ldots, k-2) \\ -\frac{1}{\alpha_{k-1}}(\boldsymbol{r}^*_k, \boldsymbol{r}_k), & (i = k-1) \end{cases}, \quad (A\boldsymbol{r}_i, \boldsymbol{r}^*_k) = \begin{cases} 0, & (i = 1, 2, \ldots, k-2) \\ -\frac{1}{\bar{\alpha}_{k-1}}(\boldsymbol{r}_k, \boldsymbol{r}^*_k), & (i = k-1) \end{cases}.$$

そのため，式 (3.6) は以下の 3 項間漸化式で表すことができる．

$$\boldsymbol{r}_{k+1} = \alpha_k \frac{(\boldsymbol{r}^*_k, A\boldsymbol{r}_k)}{(\boldsymbol{r}^*_k, \boldsymbol{r}_k)}\boldsymbol{r}_k - \frac{\alpha_k}{\alpha_{k-1}}\frac{(\boldsymbol{r}^*_k, \boldsymbol{r}_k)}{(\boldsymbol{r}^*_{k-1}, \boldsymbol{r}_{k-1})}\boldsymbol{r}_{k-1} - \alpha_k A\boldsymbol{r}_k, \tag{3.7}$$

$$\boldsymbol{r}^*_{k+1} = \bar{\alpha}_k \frac{(\boldsymbol{r}_k, A^{\mathrm{H}}\boldsymbol{r}^*_k)}{(\boldsymbol{r}_k, \boldsymbol{r}^*_k)}\boldsymbol{r}^*_k - \frac{\bar{\alpha}_k}{\bar{\alpha}_{k-1}}\frac{(\boldsymbol{r}_k, \boldsymbol{r}^*_k)}{(\boldsymbol{r}_{k-1}, \boldsymbol{r}^*_{k-1})}\boldsymbol{r}^*_{k-1} - \bar{\alpha}_k A^{\mathrm{H}}\boldsymbol{r}^*_k. \tag{3.8}$$

ここで，式 (3.3) から $\boldsymbol{r}_k$ は $\boldsymbol{r}_0$ と $C_k(A)$ の積で表されることから，式 (3.7) により，以下の式が成り立つ．

$$C_{k+1}(A) = \alpha_k \frac{(\boldsymbol{r}^*_k, A\boldsymbol{r}_k)}{(\boldsymbol{r}^*_k, \boldsymbol{r}_k)}C_k(A) - \frac{\alpha_k}{\alpha_{k-1}}\frac{(\boldsymbol{r}^*_k, \boldsymbol{r}_k)}{(\boldsymbol{r}^*_{k-1}, \boldsymbol{r}_{k-1})}C_{k-1}(A) - \alpha_k A C_k(A).$$

式 (3.4) より，残差多項式 $C_{k+1}(A)$ の定数項は I なので，式 (3.9) が成り立つ．

$$\alpha_k \frac{(\boldsymbol{r}^*_k, A\boldsymbol{r}_k)}{(\boldsymbol{r}^*_k, \boldsymbol{r}_k)} - \frac{\alpha_k}{\alpha_{k-1}}\frac{(\boldsymbol{r}^*_k, \boldsymbol{r}_k)}{(\boldsymbol{r}^*_{k-1}, \boldsymbol{r}_{k-1})} = 1. \tag{3.9}$$

次に，式 (3.7) および式 (3.8) が交代漸化式で表せることについて示す．補助ベクトルを $\boldsymbol{p}_k = \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{\alpha_k}$, $\boldsymbol{p}^*_k = \frac{\boldsymbol{x}^*_{k+1} - \boldsymbol{x}^*_k}{\bar{\alpha}_k}$ とする．ここで，$\boldsymbol{x}^*_k$ は線形方程式 $A^{\mathrm{H}}\boldsymbol{x}^* = \boldsymbol{b}^*$ の k 反復目の近似解である．このとき，$\boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{r}_k$, $\boldsymbol{b}^* - A\boldsymbol{x}^*_k = \boldsymbol{r}^*_k$ であることから，式 (3.10) が得られる．

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k, \quad \boldsymbol{r}^*_{k+1} = \boldsymbol{r}^*_k - \bar{\alpha}_k A^{\mathrm{H}}\boldsymbol{p}^*_k. \tag{3.10}$$

式 (3.10) および式 (3.9) を式 (3.7) に代入することで，以下の交代漸化式が得られる．

$$\begin{aligned}
\boldsymbol{r}_{k+1} &= \left(1 + \frac{\alpha_k}{\alpha_{k-1}} \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_{k-1}^*, \boldsymbol{r}_{k-1})}\right) \boldsymbol{r}_k - \frac{\alpha_k}{\alpha_{k-1}} \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_{k-1}^*, \boldsymbol{r}_{k-1})} \boldsymbol{r}_{k-1} - \alpha_k A \boldsymbol{r}_k, \\
\boldsymbol{r}_{k+1} - \boldsymbol{r}_k &= \frac{\alpha_k}{\alpha_{k-1}} \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_{k-1}^*, \boldsymbol{r}_{k-1})} (\boldsymbol{r}_k - \boldsymbol{r}_{k-1}) - \alpha_k A \boldsymbol{r}_k, \\
\boldsymbol{p}_k &= \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_{k-1}^*, \boldsymbol{r}_{k-1})} \boldsymbol{p}_{k-1} + \boldsymbol{r}_k = \beta_{k-1} \boldsymbol{p}_{k-1} + \boldsymbol{r}_k. \qquad (3.11)
\end{aligned}$$

同様に，式 (3.10) および式 (3.9) を式 (3.8) に代入することで以下の交代漸化式が得られる．

$$\boldsymbol{p}_k^* = \frac{(\boldsymbol{r}_k, \boldsymbol{r}_k^*)}{(\boldsymbol{r}_{k-1}, \boldsymbol{r}_{k-1}^*)} \boldsymbol{p}_{k-1}^* + \boldsymbol{r}_k^* = \bar{\beta}_{k-1} \boldsymbol{p}_{k-1}^* + \boldsymbol{r}_k^*. \qquad (3.12)$$

以上により $k+1$ 反復目の近似解 $\boldsymbol{x}_{k+1}$ は $\boldsymbol{x}_{k+1} = \alpha_k \boldsymbol{p}_k + \boldsymbol{x}_k$ で求めることができる．

最後に α_k の値を示す．式 (3.1) より，$\boldsymbol{p}_k \in \mathcal{K}_{k+1}(A; \boldsymbol{r}_0)$，$\boldsymbol{p}_k^* \in \mathcal{K}_{k+1}(A^{\mathrm{H}}; \boldsymbol{r}_0^*)$ となるため，式 (3.10) に対して，$\boldsymbol{p}_k^*$，$\boldsymbol{p}_k$ の内積をとると，以下の式が得られる．

$$(\boldsymbol{p}_k^*, \boldsymbol{r}_k) = \alpha_k (\boldsymbol{p}_k^*, A\boldsymbol{p}_k), \quad (\boldsymbol{p}_k, \boldsymbol{r}_k^*) = \bar{\alpha}_k (\boldsymbol{p}_k, A^{\mathrm{H}} \boldsymbol{p}_k^*).$$

また，式 (3.11)，式 (3.12) に対してそれぞれ，$\boldsymbol{r}_{k+1}^*$，$\boldsymbol{r}_{k+1}$ の内積をとると，以下の式が成り立つ．

$$(\boldsymbol{p}_{k+1}, \boldsymbol{r}_{k+1}^*) = (\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1}^*), \quad (\boldsymbol{p}_{k+1}^*, \boldsymbol{r}_{k+1}) = (\boldsymbol{r}_{k+1}^*, \boldsymbol{r}_{k+1}).$$

以上により α_k，$\bar{\alpha}_k$ は以下の式となる．

$$\alpha_k = \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{p}_k^*, A\boldsymbol{p}_k)}, \quad \bar{\alpha}_k = \frac{(\boldsymbol{r}_k, \boldsymbol{r}_k^*)}{(\boldsymbol{p}_k, A^{\mathrm{H}} \boldsymbol{p}_k^*)}. \qquad (3.13)$$

以上をまとめることによって，Algorithm 4 に示す BiCG 法のアルゴリズムとなる．Algorithm 4 について，$\epsilon \in \mathbb{R}$ は収束判定定数を表している．

3.2.2 COCG 法

COCG 法は，複素対称行列 $A = A^{\mathrm{T}} \neq A^{\mathrm{H}}$ に対する線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解く数値解法である．COCG 法も BiCG 法と同様に式 (3.5) に示す双直交条件を用いて，近似解 $\boldsymbol{x}_k$ および残差ベクトル $\boldsymbol{r}_k$ を一意に定める．そのため，$\boldsymbol{r}_{k+1}$，$\boldsymbol{r}_{k+1}^*$，$\boldsymbol{p}_{k+1}$，$\boldsymbol{p}_{k+1}^*$ は BiCG 法と同様に，以

Algorithm 4 BiCG 法のアルゴリズム

Input: $A, \boldsymbol{b}, \boldsymbol{x}_0$

Output: $\boldsymbol{x}_{k+1}$

$\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - A\boldsymbol{x}_0$

set $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq \boldsymbol{0}$

$\boldsymbol{p}_0 \leftarrow \boldsymbol{r}_0,\ \boldsymbol{p}_0^* \leftarrow \boldsymbol{r}_0^*,\ k \leftarrow 0$

while $\|\boldsymbol{r}_k\|_2 \leq \epsilon \|\boldsymbol{b}\|_2$ **do**

$$\alpha_k \leftarrow \frac{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}{(\boldsymbol{p}_k^*, A\boldsymbol{p}_k)}$$

$\boldsymbol{x}_{k+1} \leftarrow \alpha_k \boldsymbol{p}_k + \boldsymbol{x}_k$

$\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k, \quad \boldsymbol{r}_{k+1}^* \leftarrow \boldsymbol{r}_k^* - \bar{\alpha}_k A^{\mathrm{H}} \boldsymbol{p}_k^*$

$$\beta_k \leftarrow \frac{(\boldsymbol{r}_{k+1}^*, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k^*, \boldsymbol{r}_k)}$$

$\boldsymbol{p}_{k+1} \leftarrow \beta_k \boldsymbol{p}_k + \boldsymbol{r}_{k+1}, \quad \boldsymbol{p}_{k+1}^* \leftarrow \bar{\beta}_k \boldsymbol{p}_k^* + \boldsymbol{r}_{k+1}^*$

$k \leftarrow k + 1$

end while

下の式で表せる.

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k, \quad \boldsymbol{r}_{k+1}^* = \boldsymbol{r}_k^* - \bar{\alpha}_k A^{\mathrm{H}} \boldsymbol{p}_k^*,$$
$$\boldsymbol{p}_{k+1} = \beta_k \boldsymbol{p}_k + \boldsymbol{r}_{k+1}, \quad \boldsymbol{p}_{k+1}^* = \bar{\beta}_k \boldsymbol{p}_k^* + \boldsymbol{r}_{k+1}^*.$$

ここで，$\boldsymbol{r}_0^* = \bar{\boldsymbol{r}}_0$ としたとき，$\boldsymbol{p}_0^* = \bar{\boldsymbol{r}}_0$ となることから，以下の式が成り立つ.

$$\boldsymbol{p}_{k+1}^* = \bar{\boldsymbol{p}}_{k+1} = \bar{\beta}_k \bar{\boldsymbol{p}}_k + \bar{\boldsymbol{r}}_{k+1}.$$

また，$A = A^{\mathrm{T}} \neq A^{\mathrm{H}}$ であることから，$A^{\mathrm{H}} = \bar{A}$ となるため，以下の式が成り立つ.

$$\boldsymbol{r}_{k+1}^* = \bar{\boldsymbol{r}}_{k+1} = \bar{\boldsymbol{r}}_k - \bar{\alpha}_k \bar{A} \bar{\boldsymbol{p}}_k.$$

また，式 (3.7) に示す 3 項間漸化式は，式 (3.9) を用いることで，式 (3.14) で表すことができる.

$$\boldsymbol{r}_{k+1} = \left(1 + \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}}\right) \boldsymbol{r}_k - \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}} \boldsymbol{r}_{k-1} - \alpha_k A\boldsymbol{r}_k. \tag{3.14}$$

以上により，$\boldsymbol{r}_{k+1}^*,\ \boldsymbol{p}_{k+1}^*$ の計算を行わずに近似解を求めることができるため，計算効率が向上する.

Algorithm 5 に COCG 法のアルゴリズムを示す.

Algorithm 5 COCG 法のアルゴリズム

Input: $A, \boldsymbol{b}, \boldsymbol{x}_0$

Output: $\boldsymbol{x}_{k+1}$

$\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - A\boldsymbol{x}_0$

$\boldsymbol{p}_0 \leftarrow \boldsymbol{r}_0,\ k \leftarrow 0$

while $\|\boldsymbol{r}_k\|_2 \leq \epsilon \|\boldsymbol{b}\|_2$ **do**

$\alpha_k \leftarrow \dfrac{(\bar{\boldsymbol{r}}_k, \boldsymbol{r}_k)}{(\bar{\boldsymbol{p}}_k, A\boldsymbol{p}_k)}$

$\boldsymbol{x}_{k+1} \leftarrow \alpha_k \boldsymbol{p}_k + \boldsymbol{x}_k$

$\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$

$\beta_k \leftarrow \dfrac{(\bar{\boldsymbol{r}}_{k+1}, \boldsymbol{r}_{k+1})}{(\bar{\boldsymbol{r}}_k, \boldsymbol{r}_k)}$

$\boldsymbol{p}_{k+1} \leftarrow \beta_k \boldsymbol{p}_k + \boldsymbol{r}_{k+1}$

$k \leftarrow k + 1$

end while

3.3 Shifted Krylov 部分空間反復法

前節では線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の近似解を求める Krylov 部分空間反復法について述べた．本節では任意のシフト値 $\sigma_j \in \mathbb{C},\ (j = 1, 2, \ldots)$ を含むシフト行列 $A + \sigma_j I$ によるシフト線形方程式

$$(A + \sigma_j I)\boldsymbol{x}_j = \boldsymbol{b}, \quad (A \in \mathbb{C}^{n\times n},\ I : \text{単位行列},\ \boldsymbol{b}, \boldsymbol{x}_j \in \mathbb{C}^n,\ j = 1, 2, \ldots, N)$$

に対する Krylov 部分空間が満たす性質について述べる．また Krylov 部分空間が満たす性質を利用して，シフト線形方程式を効率的に求める Krylov 部分空間反復法の一例である Shifted COCG 法 [29] について述べる．

シフト線形方程式に対して Krylov 部分空間反復法を用いた場合，式 (3.1) に示すように，得られる k 反復目の近似解 $\boldsymbol{x}_k^{(j)}$ は以下の式を満たすように決定する．

$$\boldsymbol{x}_k^{(j)} \in \boldsymbol{x}_0^{(j)} + \mathcal{K}_k(A + \sigma_j I; \boldsymbol{r}_0^{(j)}).$$

ここで，$\boldsymbol{r}_0^{(j)} = \boldsymbol{b} - (A + \sigma_j I)\boldsymbol{x}_0^{(j)}$ は σ_j によるシフト線形方程式の初期残差を表し，$\mathcal{K}_k(A + \sigma_j I; \boldsymbol{r}_0^{(j)})$ は以下で定義される Krylov 部分空間である．

$$\mathcal{K}_k(A + \sigma_j I; \boldsymbol{r}_0^{(j)}) \equiv \mathrm{span}(\boldsymbol{r}_0^{(j)}, (A + \sigma_j I)\boldsymbol{r}_0^{(j)}, (A + \sigma_j I)^2\boldsymbol{r}_0^{(j)}, \ldots, (A + \sigma_j I)^{k-1}\boldsymbol{r}_0^{(j)}).$$

以上の条件により，σ_j によるシフト線形方程式の k 反復目の近似解 $\boldsymbol{x}_k^{(j)}$ および k 反復目の残差 $\boldsymbol{r}_k^{(j)}$ はそれぞれ式 (3.2)，式 (3.3) によって以下で表現できる．

$$\begin{aligned}\boldsymbol{x}_k^{(j)} &= \boldsymbol{x}_0^{(j)} + S_{k-1}^{(j)}(A+\sigma_j I)\boldsymbol{r}_0^{(j)},\\ \boldsymbol{r}_k^{(j)} &= C_k^{(j)}(A+\sigma_j I)\boldsymbol{r}_0^{(j)} \in \mathcal{K}_{k+1}(A+\sigma_j I;\boldsymbol{r}_0^{(j)}).\end{aligned}$$

ここで，$S_{k-1}^{(j)}(A+\sigma_j I)$ は $k-1$ 次の多項式であり，$C_k^{(j)}(A+\sigma_j I)$ は式 (3.15) で表される k 次の残差多項式である．

$$I - C_k^{(j)}(A+\sigma_j I) = (A+\sigma_j I)S_{k-1}^{(j)}(A+\sigma_j I). \tag{3.15}$$

このとき，Krylov 部分空間および残差多項式について定理 3.1 が成り立つ．

定理 3.1 (Krylov 部分空間におけるシフト不変性 [30])**.** 任意の行列 A およびベクトル $\boldsymbol{b}$ によって生成される Krylov 部分空間を $\mathcal{K}_k(A;\boldsymbol{r}_0)$ とし，シフト行列 $A+\sigma I$ によって生成される Krylov 部分空間を $\mathcal{K}_k(A+\sigma_j I;\boldsymbol{r}_0^{(j)})$ とする．このとき，$\boldsymbol{r}_0 = \boldsymbol{r}_0^{(j)}$ であれば，2 つの Krylov 部分空間は一致する．

$$\mathcal{K}_k(A;\boldsymbol{r}_0) = \mathcal{K}_k(A+\sigma_j I;\boldsymbol{r}_0^{(j)}).$$

定理 3.1 により，残差ベクトル $\boldsymbol{r}_k^{(j)}$ は $\boldsymbol{r}_k^{(j)} \in \mathcal{K}_{k+1}(A;\boldsymbol{r}_0)$ となる．

3.3.1 Shifted COCG 法

Shifted Krylov 部分空間反復法は，シフト不変性を効率的に利用して，複数のシフト線形方程式を同時に解く解法である．本節ではその Shifted Krylov 部分空間反復法の一例である Shifted COCG 法について述べる．

COCG 法では線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ に対して，以下の直交条件を課すことで近似解 $\boldsymbol{x}_k$ を一意に定めている．

$$\boldsymbol{r}_k \perp \mathcal{K}_k(\bar{A};\bar{\boldsymbol{r}}_0). \tag{3.16}$$

一方，シフト線形方程式 $(A+\sigma_j I)\boldsymbol{x}^{(j)} = \boldsymbol{b}$ に対して　COCG 法を用いた場合は，以下の直

交条件が課される．

$$\boldsymbol{r}_k^{(j)} \perp \mathcal{K}_k(\bar{A} + \bar{\sigma}_j I; \bar{\boldsymbol{r}}_0^{(j)}).$$

このとき，定理 3.1 により，残差ベクトル $\boldsymbol{r}_k^{(j)}$ は，$\boldsymbol{r}_k^{(j)} \in \mathcal{K}_{k+1}(A; \boldsymbol{r}_0)$ となり，Krylov 部分空間は，$\mathcal{K}_k(\bar{A}; \bar{\boldsymbol{r}}_0) = \mathcal{K}_k(\bar{A} + \bar{\sigma}_j I; \bar{\boldsymbol{r}}_0^{(j)})$ となる．以上により，シフト線形方程式の残差ベクトル $\boldsymbol{r}_k^{(j)}$ は，$\boldsymbol{r}_k$ と同じ $k+1$ 次の空間に属し，同じ k 次の空間に直交するため，$\boldsymbol{r}_k$ と同じ方向のベクトルとなる．つまり $\boldsymbol{r}_k^{(j)}$ はスカラー値 $\xi_k^{(j)} \in \mathbb{C}$ によって式 (3.17) で表される．

$$\boldsymbol{r}_k^{(j)} = \xi_k^{(j)} \boldsymbol{r}_k. \tag{3.17}$$

また，$\boldsymbol{r}_k^{(j)}$ は，$\boldsymbol{r}_0^{(j)}$ と残差多項式 $C_k^{(j)}(A + \sigma_j I)$ との積で表わされることから，式 (3.18) が成り立つ．

$$C_k^{(j)}(A + \sigma_j I) = \xi_k^{(j)} C_k(A). \tag{3.18}$$

ここで $C_k(A)$ は式 (3.3) に示す残差多項式である．

次に Shifted COCG 法による補助ベクトル，近似解の計算について述べる．残差ベクトル $\boldsymbol{r}_{k+1}^{(j)}$ は式 (3.14) から，以下の 3 項間漸化式で表すことができる．

$$\boldsymbol{r}_{k+1}^{(j)} = \left(1 + \frac{\alpha_k^{(j)} \beta_{k-1}^{(j)}}{\alpha_{k-1}^{(j)}}\right) \boldsymbol{r}_k^{(j)} - \frac{\alpha_k^{(j)} \beta_{k-1}^{(j)}}{\alpha_{k-1}^{(j)}} \boldsymbol{r}_{k-1}^{(j)} - \alpha_k^{(j)} (A + \sigma_j I) \boldsymbol{r}_k^{(j)}. \tag{3.19}$$

式 (3.19) の右辺に対して，式 (3.17) を用いることで，式 (3.20) が得られる．

$$\boldsymbol{r}_{k+1}^{(j)} = \left(1 + \frac{\alpha_k^{(j)} \beta_{k-1}^{(j)}}{\alpha_{k-1}^{(j)}}\right) \xi_k^{(j)} \boldsymbol{r}_k - \frac{\alpha_k^{(j)} \beta_{k-1}^{(j)}}{\alpha_{k-1}^{(j)}} \xi_{k-1}^{(j)} \boldsymbol{r}_{k-1} - \alpha_k^{(j)} \xi_k^{(j)} (A + \sigma_j I) \boldsymbol{r}_k. \tag{3.20}$$

また，式 (3.17) の右辺に対して，式 (3.14) を用いることで，式 (3.21) が得られる．

$$\boldsymbol{r}_{k+1}^{(j)} = \left(1 + \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}}\right) \xi_{k+1}^{(j)} \boldsymbol{r}_k - \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}} \xi_{k+1}^{(j)} \boldsymbol{r}_{k-1} - \alpha_k \xi_{k+1}^{(j)} A \boldsymbol{r}_k. \tag{3.21}$$

このとき，式 (3.20) および式 (3.21) の $\boldsymbol{r}_k, \boldsymbol{r}_{k-1}, A\boldsymbol{r}_k$ それぞれの係数を比較することで，式

(3.22) が得られる．

$$\alpha_{k-1}^{(j)} = \alpha_k \frac{\xi_{k+1}^{(j)}}{\xi_k^{(j)}}, \quad \beta_{k-1}^{(j)} = \beta_{k-1} \frac{(\xi_k^{(j)})^2}{(\xi_{k-1}^{(j)})^2}, \quad \xi_{k+1}^{(j)} = \frac{\alpha_{k-1}\xi_k^{(j)}\xi_{k-1}^{(j)}}{\alpha_{k-1}(1+\alpha_k\sigma_j) + \alpha_k\beta_{k-1}(\xi_{k-1}^{(j)} - \xi_k^{(j)})}. \tag{3.22}$$

ただし，$\xi_{-1}^{(j)} = \xi_0^{(j)} = \alpha_{-1} = 1$, $\beta_{-1} = 0$ とする．シフト線形方程式に対する，近似解 $\boldsymbol{x}_{k+1}^{(j)}$ および補助ベクトル $\boldsymbol{p}_{k+1}^{(j)}$ は，3.2.2 項により，式 (3.23) で表される．

$$\boldsymbol{x}_{k+1}^{(j)} = \alpha_k^{(j)}\boldsymbol{p}_k^{(j)} + \boldsymbol{x}_k^{(j)}, \quad \boldsymbol{p}_{k+1}^{(j)} = \beta_k^{(j)}\boldsymbol{p}_k^{(j)} + \boldsymbol{r}_{k+1}^{(j)} = \beta_k^{(j)}\boldsymbol{p}_k^{(j)} + \xi_{k+1}^{(j)}\boldsymbol{r}_{k+1}. \tag{3.23}$$

以上により，$\alpha_k^{(j)}$, $\beta_k^{(j)}$ は，式 (3.22) により，スカラー値の計算で求めることができる．

COCG 法の計算過程で一番計算コストの高い部分は，α_k の計算および $\boldsymbol{r}_{k+1}$ の計算で現れる行列・ベクトル積の計算である．そのため，複数のシフト線形方程式を COCG 法によって計算する場合，行列・ベクトル積の計算は反復回数とシフト点数の積 ($k \times N$ 回) となる．しかし，Shifted COCG 法では，式 (3.22) および式 (3.23) に示すように，α_k および $\boldsymbol{r}_{k+1}$ の計算結果を再利用している．そのため，Shifted COCG 法の行列・ベクトル積の計算は反復回数のみ (k 回) となり，計算効率を向上させることができる．

Algorithm 6 に Shifted COCG 法のアルゴリズムを示す．Algorithm 6 において，$\sigma_{\mathrm{s}} \in \mathbb{C}$ は任意のシフト値であり，シードシフト値と呼ばれる．

3.4 有理式型推定法における Krylov 部分空間反復法を用いた多項式表現

3.2 節および 3.3 節では，線形方程式に対する解法の一つである Krylov 部分空間反復法について述べた．2 章で述べた有理式型推定法では線形方程式の求解が必要となるため，それに対して Krylov 部分空間反復法を用いることで，固有値数を推定することができる．特に，Krylov 部分空間反復法を利用した場合，その反復過程で推定値の計算が可能となる．さらに，Krylov 部分空間反復法で得られる反復ごとの近似解は，多項式で表現できることから，有理式型推定法も多項式で表現できると考えられる．

本節では，有理式型推定法に対して Krylov 部分空間反復法を用いた場合，有理式型推定法が多項式で表現可能であることを明らかにし，Chebyshev 多項式を用いた方法との多項式次数による比較を行う．本節ではエルミート標準固有値問題 $F(\lambda)\boldsymbol{x} = (\lambda I - A)\boldsymbol{x} = \boldsymbol{0}$ を対象と

Algorithm 6 Shifted COCG 法のアルゴリズム

Input: $A, \boldsymbol{b}, \sigma_{\mathrm{s}}, \sigma_j,$ for $j = 1, 2, \ldots, N$

Output: $\boldsymbol{x}_{k+1}^{(j)},$ for $j = 1, 2, \ldots, N$

$\boldsymbol{x}_0 \leftarrow \boldsymbol{0},\ \boldsymbol{r}_0 \leftarrow \boldsymbol{b},\ \boldsymbol{p}_0 \leftarrow \boldsymbol{b},\ \alpha_{-1} \leftarrow 1,\ \beta_{-1} \leftarrow 0$

$\boldsymbol{x}_0^{(j)} \leftarrow \boldsymbol{0},\ \xi_{-1}^{(j)} \leftarrow 1,\ \xi_0^{(j)} \leftarrow 1$ for $j = 1, 2, \ldots, N$

$k \leftarrow 0$

while $\max(|\xi_k^{(j)}|\|\boldsymbol{r}_k\|_2, \|\boldsymbol{r}_k\|_2) \leq \epsilon\|\boldsymbol{b}\|_2$ **do**

$$\alpha_k \leftarrow \frac{(\bar{\boldsymbol{r}}_k, \boldsymbol{r}_k)}{(\bar{\boldsymbol{p}}_k, (A + \sigma_{\mathrm{s}} I)\boldsymbol{p}_k)}$$

$\boldsymbol{x}_{k+1} \leftarrow \alpha_k \boldsymbol{p}_k + \boldsymbol{x}_k$

for $j = 1$ to N **do**

$$\xi_{k+1}^{(j)} \leftarrow \frac{\alpha_{k-1}\xi_k^{(j)}\xi_{k-1}^{(j)}}{\alpha_{k-1}(1 + \alpha_k\sigma_j) + \alpha_k\beta_{k-1}(\xi_{k-1}^{(j)} - \xi_k^{(j)})}$$

$$\beta_{k-1}^{(j)} \leftarrow \beta_{k-1}\frac{(\xi_k^{(j)})^2}{(\xi_{k-1}^{(j)})^2}$$

$$\alpha_{k-1}^{(j)} \leftarrow \alpha_k\frac{\xi_{k+1}^{(j)}}{\xi_k^{(j)}}$$

$\boldsymbol{p}_{k+1}^{(j)} \leftarrow \beta_k^{(j)}\boldsymbol{p}_k^{(j)} + \xi_{k+1}^{(j)}\boldsymbol{r}_{k+1}$

$\boldsymbol{x}_{k+1}^{(j)} \leftarrow \alpha_k^{(j)}\boldsymbol{p}_k^{(j)} + \boldsymbol{x}_k^{(j)}$

end for

$\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k - \alpha_k(A + \sigma_s)\boldsymbol{p}_k$

$$\beta_k \leftarrow \frac{(\bar{\boldsymbol{r}}_{k+1}, \boldsymbol{r}_{k+1})}{(\bar{\boldsymbol{r}}_k, \boldsymbol{r}_k)}$$

$\boldsymbol{p}_{k+1} \leftarrow \beta_k\boldsymbol{p}_k + \boldsymbol{r}_{k+1}$

$k \leftarrow k + 1$

end while

した固有値分布推定計算を行う.

3.4.1 有理式型推定法の多項式表現

標準固有値問題に対する有理式型推定法は，式 (2.3) より，以下の式を計算する.

$$\tilde{m}_{\mathrm{R}} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{N-1} w_j \boldsymbol{v}_\ell^{\mathrm{T}} \boldsymbol{x}_{j,\ell}, \quad (z_j I - A)\boldsymbol{x}_{j,\ell} = \boldsymbol{v}_\ell. \tag{3.24}$$

ここで，線形方程式 $(z_j I - A)\boldsymbol{x}_{j,\ell} = \boldsymbol{v}_\ell$ の計算に Krylov 部分空間反復法を用いる．このとき，$(k+1)$ 反復目の近似解を $\boldsymbol{x}_{j,\ell}^{(k+1)}$ としたとき，式 (3.2) から，以下の多項式で表現できる.

$$\boldsymbol{x}_{j,\ell}^{(k+1)} = S_k^{(j,\ell)}(z_j I - A)\boldsymbol{v}_\ell.$$

ここで $S_k^{(j,\ell)}(z_j I - A)$ は k 次の多項式であり，初期解 $\boldsymbol{x}_{j,\ell}^{(0)} = \boldsymbol{0}$ とする．これにより，$\tilde{m}_{\mathrm{R}}$ は以下の式で表すことができる.

$$\tilde{m}_{\mathrm{R}} \approx \tilde{m}_{\mathrm{R}}^{(k)} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{N-1} w_j \boldsymbol{v}_\ell^{\mathrm{T}} \boldsymbol{x}_{j,\ell}^{(k+1)} = \frac{1}{L}\sum_{j=1}^{N}\sum_{\ell=1}^{L} w_j \boldsymbol{v}_\ell^{\mathrm{T}} S_k^{(j,\ell)}\left(z_j I - A\right)\boldsymbol{v}_\ell. \tag{3.25}$$

式 (3.25) に対して，3.3 節で述べた Shifted Krylov 部分空間反復法を用いる．Shifted Krylov 部分空間反復法のシフト不変性を利用することにより，$S_k^{(j,\ell)}(z_j I - A)$ は以下の k 次の多項式 $Q_k^{(N,\ell)}(A)$ で表現することができる.

$$Q_k^{(N,\ell)}\left(A\right) = \sum_{j=1}^{N} w_j S_k^{(j,\ell)}\left(z_j I - A\right).$$

以上により，有理式型推定法による推定値 $\tilde{m}_{\mathrm{R}}^{(k)}$ は多項式型推定法と同様に k 次の多項式によって表現できる.

$$\tilde{m}_{\mathrm{R}}^{(k)} = \frac{1}{L}\sum_{\ell=1}^{L} \boldsymbol{v}_\ell^{\mathrm{T}} Q_k^{(N,\ell)}\left(A\right)\boldsymbol{v}_\ell. \tag{3.26}$$

3.4.2 有理式型推定法と多項式型推定法の比較

本項では，有理式型推定法と多項式型推定法の計算コストおよび，フィルタ関数の比較を行う.

多項式型推定法によって計算した k 反復目の推定値を $\tilde{m}_{\mathrm{P}}^{(k)}$ としたとき，式 (2.9) に示すように，$\tilde{m}_{\mathrm{P}}^{(k)}$ は k 次の多項式 $P_k(A)$ で求められる．

$$\tilde{m}_{\mathrm{P}}^{(k)} = \frac{1}{L}\sum_{\ell=1}^{L} \boldsymbol{v}_\ell^{\mathrm{T}} P_k(A)\boldsymbol{v}_\ell.$$

$\tilde{m}_{\mathrm{P}}^{(k)}$ の計算で最も計算コストの高い部分は行列・ベクトル積となり，計算回数は反復回数と $\boldsymbol{v}_\ell$ の本数の積 ($k \times L$ 回) となる．

有理式型推定法は Shifted Krylov 部分空間反復法を用いることで，式 (3.26) に示すように k 次の多項式 $Q_k^{(N,\ell)}(A)$ で表される．多項式 $Q_k^{(N,\ell)}(A)$ は，Shifted Krylov 部分空間反復法による近似解の多項式から得られる．そのため，$\tilde{m}_{\mathrm{R}}^{(k)}$ の計算で最も計算コストの高い部分は，Shifted Krylov 部分空間反復法の計算過程で現れる行列・ベクトル積の計算となる．3.3.1 項で述べたように，Shifted Krylov 部分空間反復法を用いた際の行列・ベクトル積の計算回数は反復回数のみ (k 回) となる．そのため，$\tilde{m}_{\mathrm{R}}^{(k)}$ の行列・ベクトル積の計算回数は，多項式型推定法と同様に，反復回数と $\boldsymbol{v}_\ell$ の本数の積 ($k \times L$ 回) となる．

以上により，多項式型推定法および有理式型推定法の行列・ベクトル積の回数は，反復回数と $\boldsymbol{v}_\ell$ の本数が同じ場合，同数となる．

次に各推定法のフィルタ関数の比較を行う．

多項式型推定法のフィルタ関数は式 (2.13) で述べたように，k 次の多項式 $P_k(\cdot)$ で表される．このフィルタ関数は，第 1 種 Chebyshev 多項式によって求めているため，行列の性質に関わらず，多項式の次数 k のみに依存した関数となる．

有理式型推定法のフィルタ関数について述べる．行列 A は固有値分解により $A = U\Lambda U^{\mathrm{H}}$ と表すことができる．ここで，Λ は固有値 $\lambda_1, \lambda_2, \ldots, \lambda_n$ を対角要素とする対角行列であり，U は i 列目のベクトル $\boldsymbol{u}_i$ が固有ベクトルとなるユニタリ行列である．式 (3.26) に対して固有値分解を用いることで，以下のように展開できる．

$$\tilde{m}_{\mathrm{R}}^{(k)} = \frac{1}{L}\sum_{\ell=1}^{L} \boldsymbol{v}_\ell^{\mathrm{T}} U Q_k^{(N,\ell)}(\Lambda)\, U^{\mathrm{H}}\boldsymbol{v}_\ell = \sum_{i=1}^{n}\frac{1}{L}\sum_{\ell=1}^{L} Q_k^{(N,\ell)}(\lambda_i)\left|\boldsymbol{v}_\ell^{\mathrm{T}}\boldsymbol{u}_i\right|^2.$$

これにより有理式型推定法では多項式の次数ごとに以下のフィルタ関数 $R_k^{(N,L)}$ を作成する．

$$R_k^{(N,L)}(\lambda_i) = \frac{1}{L}\sum_{\ell=1}^{L} Q_k^{(N,\ell)}(\lambda_i).$$

フィルタ関数 $R_k^{(N,L)}(\cdot)$ は多項式の次数 k に依存した関数となる．

次に有理式型推定法の誤差について調べる．有理式型推定法による推定値と，Shifted Krylov 部分空間反復法を用いた有理式型推定法による推定値との誤差を $e_{\mathrm{R}} = \tilde{m}_{\mathrm{R}} - \tilde{m}_{\mathrm{R}}^{(k)}$ とする．式 (3.24) および式 (3.25) から以下の式が得られる．

$$e_{\mathrm{R}} = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{N-1} w_j \boldsymbol{v}_\ell^{\mathrm{T}}((z_j I - A)^{-1}\boldsymbol{v}_\ell - \boldsymbol{x}_{j,\ell}^{(k+1)}) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{N-1} w_j \boldsymbol{v}_\ell^{\mathrm{T}}(z_j I - A)^{-1}\boldsymbol{r}_{j,\ell}^{(k+1)}.$$

ここで，$\boldsymbol{r}_{j,\ell}^{(k+1)}$ は Krylov 部分空間反復法によって得られる $(k+1)$ 反復目の残差ベクトルである．このとき，初期解 $\boldsymbol{x}_{j,\ell}^{(0)} = \boldsymbol{0}$ として，Shifted COCG 法を用いた場合，シフト不変性によって以下の式が得られる．

$$\boldsymbol{r}_{j,\ell}^{(k+1)} = \xi_{j,\ell}^{(k+1)}\boldsymbol{r}_\ell^{(k+1)} = \xi_{j,\ell}^{(k+1)} C_{k+1}^{(\ell)}(\sigma I - A)\boldsymbol{v}_\ell.$$

ここで，$\boldsymbol{r}_\ell^{(k+1)}$ は任意の $\sigma \in \mathbb{C}$ による線形方程式 $(\sigma I - A)\boldsymbol{x} = \boldsymbol{v}_\ell$ の $k+1$ 反復目の残差ベクトルであり，$C_{k+1}^{(\ell)}(\sigma I - A)$ は $\boldsymbol{r}_\ell^{(k+1)}$ から得られる $k+1$ 次の残差多項式である．上記の式および固有値分解を用いることで，

$$\begin{aligned} e_{\mathrm{R}} &= \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=0}^{N-1} w_j \boldsymbol{v}_\ell^{\mathrm{T}}(z_j I - A)^{-1}\xi_{j,\ell}^{(k+1)} C_{k+1}^{(\ell)}(\sigma I - A)\boldsymbol{v}_\ell \\ &= \frac{1}{L}\sum_{\ell=1}^{L}\sum_{i=1}^{n} g_{k+1}^{(\ell)}(\lambda_i)\|\boldsymbol{r}_\ell^{(k+1)}\|_2^{-1} C_{k+1}^{(\ell)}(\sigma - \lambda_i)\left|\boldsymbol{v}_\ell^{\mathrm{T}}\boldsymbol{u}_i\right|^2, \end{aligned}$$

となる．ここで，$g_{k+1}^{(\ell)}(\lambda_i) = \|\boldsymbol{r}_\ell^{(k+1)}\|_2 \sum_{j=0}^{N-1} w_j \xi_{j,\ell}^{(k+1)}/(z_j - \lambda_i)$ である．このとき，行列 $\sigma I - A$ は正定値エルミート行列である場合，多項式 $C_{k+1}^{(\ell)}(\sigma - \lambda_i)$ は，各固有値 λ_i, $i = 1, 2, \ldots, n$ に対して，以下の式を最小化するように構成される．

$$\min_{\lambda_i}\left\{\sum_{i=1}^{n}(\sigma - \lambda_i)^{-1}(C_{k+1}^{(\ell)}(\sigma - \lambda_i)|\boldsymbol{v}_\ell^{\mathrm{T}}\boldsymbol{u}_i|)^2\right\}$$

以上により，$g_{k+1}^{(\ell)}(\lambda_i)$ は閉曲線 Γ から離れるにつれて減少していく関数となり，フィルタ関数 $R_k^{(N,L)}(\cdot)$ は，多項式の次数 k の他に，行列の性質に依存した関数となる．

3.5 数値実験

本節では，エルミート標準固有値問題 $F(\lambda)\boldsymbol{x} = (\lambda I - A)\boldsymbol{x} = \boldsymbol{0}$ に対して，多項式型推定法および有理式型推定法をもちいて固有値数を推定し，その性能を比較する．有理式型推定法では線形方程式の求解に Shifted COCG 法を用いた．数値実験は，
CPU: MacBook Air 1.3GHz Intel Core i5,
Memory: 8GB,
OS: Mac OS ver.10.9.5,
上で行い，MATLAB8.5.0 を用いた．本実験において $\boldsymbol{v}_\ell$ は MATLAB の関数である `rand` を用いており，そのシード値は ‘twister’ の 0 シードを用いた．

3.5.1 数値実験 3-1

本項では，多項式型推定法と有理式型推定法のフィルタ関数が，行列によってどのように変化するのかを調べる．対象とする行列 A は，3 つの 1000 次元の対角行列を用いた．各対角行列の対角要素 $A = \{a_{ii}\}_{i=1,2,\ldots,1000}$ はそれぞれ，
パターン 1: $a_{ii} = c_i$，パターン 2: $a_{ii} = \mathrm{sign}(c_i) * c_i^2$，パターン 3: $a_{ii} = \mathrm{sign}(c_i) * \sqrt{|c_i|}$
とした．ここで，c_i は区間 $[-1, 1]$ 内で線形に等間隔な値 $c_i = -1 + 2 * (i-1)/999$ であり，$\mathrm{sign}(\cdot)$ 符号関数である．

固有値数を推定する区間 $[a, b]$ は $[-0.035, 0.035]$ とした．各行列における，区間内部の実際の固有値数はそれぞれ，
パターン 1: 34，パターン 2: 186，パターン 3: 2
である．確率的固有値分布推定法のパラメータについて，$\boldsymbol{v}_\ell$ の本数は $L = 30$ とし，多項式の次数は $k = 200$ とした．有理式型推定法では，閉曲線の中心 γ を $(a+b)/2$，半径 ρ を $(a-b)/2$ とした．これにより多項式型推定法と有理式型推定法は同じ区間の固有値数を推定する．

有理式型推定法について，積分点数は $N = 16$ とし，Shifted COCG 法のシードシフト値は $\sigma = 2 > \max(\lambda_i)$ とした．

各行列における実験結果をそれぞれ図 3.1，図 3.2，図 3.3 に示す．横軸はフィルタ関数の引数部分の値を示しており，縦軸はフィルタ関数の絶対値を表している．黒点は固有値の分布を示しており，青点，緑点，赤点はそれぞれ，多項式型推定法によって得られるフィルタ関数 $P_k(\lambda)$，多項式型推定法+Jackson 係数によって得られるフィルタ関数 $P_k(\lambda)$(Jackson)，有理式型推定法によって得られるフィルタ関数 $R_k^{(N,L)}(\lambda)$ を示している．また $h(\lambda)$ は，理想のフィルタ関数を示している．図 3.1，図 3.2，図 3.3 から，フィルタ関数 $P_k(\lambda)$ および $P_k(\lambda)$(Jackson)

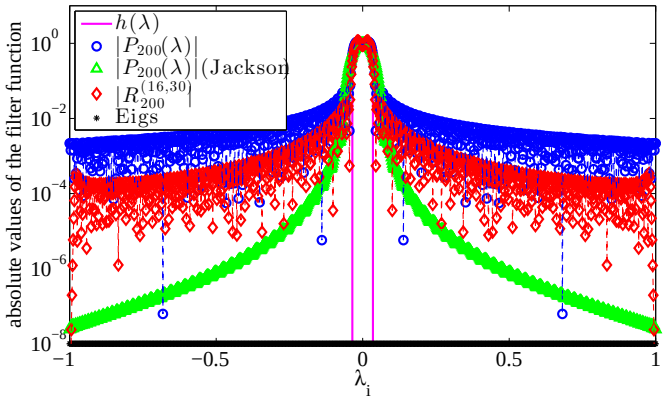

図 3.1: パターン 1 行列における 200 次のフィルタ関数

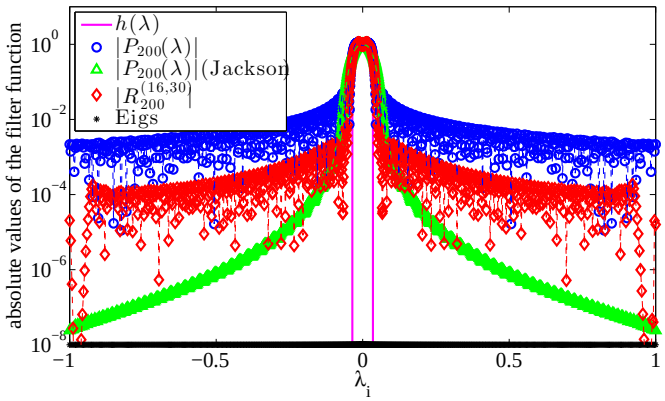

図 3.2: パターン 2 行列における 200 次のフィルタ関数

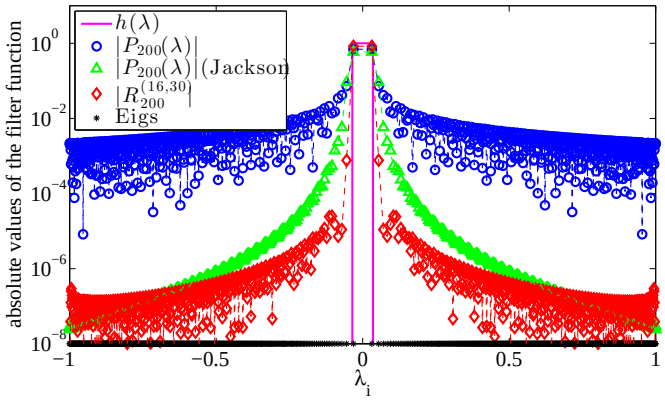

図 3.3: パターン 3 行列における 200 次のフィルタ関数

は，どのような行列でもフィルタ関数が変化していないことがわかる．これにより，多項式型推定法は行列の性質に依存しないフィルタ関数を作成する．一方，フィルタ関数 $R_k^{(N,L)}(\lambda)$ は，行列によって形が変化していることがわかる．また，$g_{k+1}^{(\ell)}(\lambda)$ は固有値数を求める区間から離れるにつれて，0 に減衰していることがわかる．これにより，有理式型推定法は行列に性質に依存したフィルタ関数を作成する．特に図 3.3 から，$R_k^{(N,L)}(\lambda)$ および $g_{k+1}^{(\ell)}(\lambda)$ は $P_k(\lambda)$ および $P_k(\lambda)$(Jackson) と比べて，固有値数を求める区間外部の減衰が早いことがわかる．以上によって，有理式型推定法は，指定した区間内部の固有値数が少ない場合，多項式型推定法よりも効果的なフィルタ関数を作成することがわかる．

3.5.2 数値実験 3-2

本項では，多項式型推定法と有理式型推定法のフィルタ関数および推定値が，多項式の次数によってどのように変化するのかを調べる．対象とする行列は ELSES matrix library[31, 32] の VCNT4000 を用いた．VCNT4000 はガーボンナノチューブの振動計算から現れる，実対称 4000 次元の行列である．本実験では MATLAB の関数である `eig` で求めた固有値を実際の固有値とした．固有値数を推定する区間 $[a, b]$ は 3 つの区間を用いた．各区間はそれぞれ，
パターン 1: $[0.1, 0.15]$，パターン 2: $[-0.125, -0.075]$，パターン 3: $[0.33, 0.38]$
とし，全ての区間の区間幅は同じものとした．各区間での実際の固有値数は，
パターン 1: 32，パターン 2: 4，パターン 3: 0
である．確率的固有値分布推定法のパラメータについて，$\boldsymbol{v}_\ell$ の本数は $L = 30$ とし，多項式の次数は $k = 10, 20, 50, 100, 150, 200$ と変化させた．有理式型推定法では，閉曲線の中心 γ を $(a+b)/2$，半径 ρ を $(a-b)/2$ とした．これにより多項式型推定法と有理式型推定法は同じ区間の固有値数を推定する．有理式型推定法について，積分点数は $N = 16$ とし，Shifted COCG 法のシードシフト値は $\sigma = 2 > \max(\lambda_i)$ とした．

表 3.1，表 3.2，表 3.3 に各推定法の推定値を示す．全ての表から，どの解法においても，多項式の次数 k を増やすことで，推定値が実際の値に近づいていくことがわかる．特に，表 3.3 から，固有値が存在しない区間 $[0.33, 0.38]$ の場合は，有理式型推定法が多項式型推定法よりも少ない多項式の次数で実際の値に近づくのがわかる．これは，前項の実験で示したように，推定する区間内部の固有値数が少ない場合，有理式型推定法が多項式型推定法よりも効果的なフィルタ関数を作成することが理由と考えられる．図 3.4，図 3.5，図 3.6 に各推定法の区間ごとのフィルタ関数を示す．図の横軸，縦軸，各マーカの意味は，前項の実験結果と同じである．全ての図から，どの解法においても，推定する区間外部では減衰するフィルタ関数が作成されていることがわかる．特に，図 3.6 では，有理式型推定法のフィルタ関数 $R_k^{(N,L)}(\lambda)$

表 3.1: 区間 $[0.1, 0.15]$ における推定値の変化

k	多項式型推定法	多項式型推定法+Jackson 係数	有理式型推定法
10	134.83	110.55	279.15
20	72.75	118.51	159.12
50	24.30	88.22	33.80
100	34.18	52.55	36.75
150	32.45	42.54	31.13
200	31.52	38.62	31.19
実際の値	32	32	32

表 3.2: 区間 $[-0.125, -0.075]$ における推定値の変化

k	多項式型推定法	多項式型推定法+Jackson 係数	有理式型推定法
10	86.22	115.63	223.48
20	22.80	101.49	89.00
50	2.52	49.83	7.07
100	1.39	18.71	4.24
150	3.22	9.57	3.98
200	4.49	6.38	3.96
実際の値	4	4	4

表 3.3: 区間 $[0.33, 0.38]$ における推定値の変化

k	多項式型推定法	多項式型推定法+Jackson 係数	有理式型推定法
10	96.86	154.60	0.00
20	8.50	122.17	0.00
50	6.01	41.10	0.01
100	3.93	11.05	0.02
150	0.86	3.86	0.02
200	0.33	1.43	0.02
実際の値	0	0	0

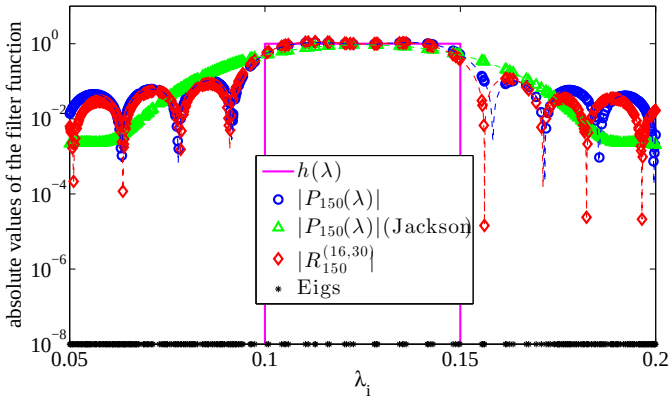

図 3.4: 区間 $[0.1, 0.15]$ における 150 次のフィルタ関数

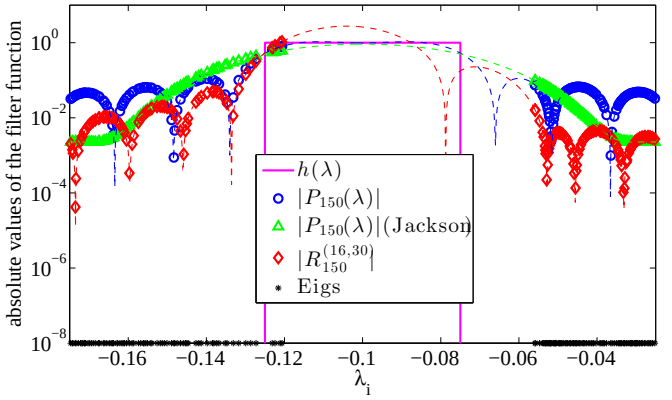

図 3.5: 区間 $[-0.125, -0.075]$ における 150 次のフィルタ関数

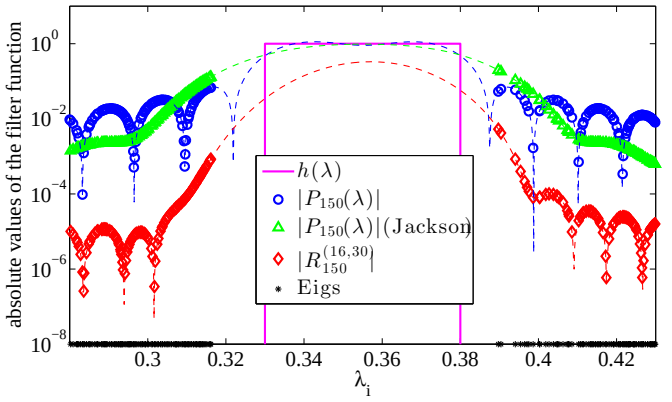

図 3.6: 区間 $[0.33, 0.38]$ における 150 次のフィルタ関数

が，多項式型推定法よりも，区間外部で減衰していることがわかる．以上により，固有値が存在しない区間を推定する際には，有理式型推定法の方が多項式型推定法よりも有効である．

固有値が存在しない区間を求めることは，遅延微分方程式から現れる固有値問題 [2] や，3 次元フォトニック結晶構造解析から現れる固有値問題 [15, 16] などにおいて必要とされる．そのため，有理式型推定法はそのような問題への応用が期待できる．

3.6 小括

本章では線形方程式を求解する反復解法の一つである Krylov 部分空間反復法および Shifted Krylov 部分空間反復法について述べ，それらによって得られる近似解が多項式で表現可能であることを示した．また，Krylov 部分空間反復法の多項式表現を利用して，有理式型推定法が多項式で表現できることを示した．これにより，多項式型推定法との計算コストによる比較およびフィルタ関数による比較が可能となることを示した．特に Krylov 部分空間反復法を利用した有理式型推定法によって得られるフィルタ関数は，多項式の次数および行列の性質によって変化することを示した．数値実験から，固有値数を推定する領域内部の固有値数が少ない場合，多項式型推定法よりも，領域外部の減衰が強いことがわかった．また，固有値が存在しない領域に対して，有理式型推定法は，多項式有理式型推定法よりも少ない多項式の次数によって固有値数を推定できることがわかった．この結果より，有理式型推定法は，多項式型推定法よりも固有値が存在しない領域の固有値数を効果的に求めることができる．

第4章　非線形固有値問題における有理式型固有値分布推定法

2 章および 3 章では，標準・一般化固有値問題に対する固有値の分布を求める方法の一つである，確率的固有値分布推定法について述べた．本章では確率的固有値分布推定法の一つである有理式型推定法をを拡張された Smith の標準形を用いて拡張し，非線形固有値問題に適用する方法について述べる．

4.1　序説

本章では非線形固有値問題

$$F(\lambda)\boldsymbol{x} = \mathbf{0}, \quad (\boldsymbol{x} \in \mathbb{C}^n \backslash \{\mathbf{0}\}, z \in \mathbb{C}, F(\cdot) : \mathbb{C} \to \mathbb{C}^{n \times n})$$

における固有値の分布推定について考える．

非線形固有値問題は様々な科学技術計算において現れる．例として，遅延微分方程式から指数関数を含む非線形固有値問題 [2] が現れ，量子ドットの電子状態計算から 5 次多項式固有値問題 [1] が現れ，高エネルギー加速器の設計から平方根を含む非線形固有値問題 [33] が現れる．このような問題に対して固有値の分布を推定することは，固有値解法のパラメータなどに利用することができるため重要となる．

1 章で述べたように，固有値の分布を求める方法には様々な方法が存在する．しかし，それらの解法は全て標準・一般化固有値問題を対象としており，非線形固有値問題に対する分布推定法は提案されていない．

そのため，本章では，非線形固有値問題に対する固有値の分布を推定する方法を提案する．本章で提案する方法は，固有値分布推定法の一つである有理式型推定法を非線形固有値問題に拡張した手法である．

本章では，まず提案法である非線形固有値問題における有理式型推定法について述べ，提案法が，一般化固有値問題における有理式型推定法から拡張が可能となることを理論的に説

明する．次に，数値実験によって提案方法の有効性を検証する．

4.2 有理式型推定法の非線形固有値問題に対する拡張

非線形固有値問題

$$F(\lambda)\boldsymbol{x} = \mathbf{0}, \quad (\boldsymbol{x} \in \mathbb{C}^n \backslash \{\mathbf{0}\}, z \in \mathbb{C}, F(\cdot) : \mathbb{C} \to \mathbb{C}^{n \times n})$$

について，$F(z)$ を解析的行列値関数とする．ここで解析的行列値関数とは複素平面上の領域 Ω 内で解析的な関数 $f_{ij}(z) : \mathbb{C} \to \mathbb{C},\ (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n)$ を要素として持つ n 次行列とする．つまり，領域 Ω 内であれば $F(z)$ は正則であり，z に対する微分が可能である．

複素平面上の任意の閉曲線 Γ 内部の固有値の数を m_{R} とする．拡張法では，式 (4.1) を計算することにより，m_{R} を求める．

$$m_{\mathrm{R}} = \frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} \mathrm{tr}\left(F(z)^{-1} \frac{\mathrm{d}F(z)}{\mathrm{d}z}\right) \mathrm{d}z, \quad (z \in \Omega). \tag{4.1}$$

以下に，式 (4.1) から固有値数を求めることが可能であることを証明する．

解析的行列関数について定理 4.1 が成り立つ．

定理 4.1 (拡張された Smith の標準形 [34, 35])**.** 解析的行列関数 $F(z) : \mathbb{C} \to \mathbb{C}^{n \times n}$ は，次のように表される．

$$F(z) = U(z)\, D(z)\, V(z).$$

ここで行列 $U(z) : \mathbb{C} \to \mathbb{C}^{n \times n}$ および $V(z) : \mathbb{C} \to \mathbb{C}^{n \times n}$ は正則であり，行列式が非零の定数となる．また $D(z)$ は $d_j(z) : \mathbb{C} \to \mathbb{C},\ (j = 1, 2, \ldots, n-1)$ を対角要素とする対角行列で表される．d_j は領域 Ω 内で解析的な関数であり，$j = 2, 3, \ldots, n-1$ のとき $d_j(z)$ は $d_{j-1}(z)$ で割り切れ，$d_1(z) \neq 0$ である．

式 (4.1) のトレース以下の式に対して，拡張された Smith の標準形を用いることにより，式 (4.2) が成り立つ．

$$\mathrm{tr}\left(F(z)^{-1} \frac{\mathrm{d}F(z)}{\mathrm{d}z}\right) = \mathrm{tr}\left(U(z)^{-1} \frac{\mathrm{d}U(z)}{\mathrm{d}z}\right) + \mathrm{tr}\left(D(z)^{-1} \frac{\mathrm{d}D(z)}{\mathrm{d}z}\right) + \mathrm{tr}\left(V(z)^{-1} \frac{\mathrm{d}V(z)}{\mathrm{d}z}\right). \tag{4.2}$$

ここで，補題 4.2 および補題 4.3 から定理 4.4 が成り立つ．

定理 4.2 (Jaccobi's formula[36])**.** 行列 A が正則の時，$\tilde{A}$ を行列 A の余因子行列とすると，以下の式が成り立つ.

$$\frac{\mathrm{d}\left(\det A\right)}{\mathrm{d}z} = \mathrm{tr}\left(\tilde{A}\frac{\mathrm{d}A}{\mathrm{d}z}\right). \tag{4.3}$$

ここで $\det A$ は行列 A の行列式を表す.

定理 4.3 (クラメルの公式)**.** 行列 A が正則の時，$\tilde{A}$ を行列 A の余因子行列とすると，以下の式が成り立つ.

$$A^{-1} = \frac{\tilde{A}}{\det A}. \tag{4.4}$$

定理 4.4. 行列 A が正則のとき，以下の式が成り立つ.

$$\mathrm{tr}\left(A^{-1}\frac{\mathrm{d}A}{\mathrm{d}z}\right) = \frac{1}{\det A}\frac{\mathrm{d}\left(\det A\right)}{\mathrm{d}z}. \tag{4.5}$$

式 (4.2) に対して定理 4.4 適用すると，行列式が非零の定数となることから，以下の式が成り立つ.

$$\mathrm{tr}\left(U\left(z\right)^{-1}\frac{\mathrm{d}U\left(z\right)}{\mathrm{d}z}\right) = \frac{1}{\det\left(U\left(z\right)\right)}\frac{\mathrm{d}\left(\det\left(U\left(z\right)\right)\right)}{\mathrm{d}z} = 0,$$
$$\mathrm{tr}\left(V\left(z\right)^{-1}\frac{\mathrm{d}U\left(z\right)}{\mathrm{d}z}\right) = \frac{1}{\det\left(V\left(z\right)\right)}\frac{\mathrm{d}\left(\det\left(V\left(z\right)\right)\right)}{\mathrm{d}z} = 0.$$

これにより，式 (4.2) は式 (4.6) となる.

$$\mathrm{tr}\left(F\left(z\right)^{-1}\frac{\mathrm{d}F\left(z\right)}{\mathrm{d}z}\right) = \mathrm{tr}\left(D\left(z\right)^{-1}\frac{\mathrm{d}D\left(z\right)}{\mathrm{d}z}\right) = \sum_{j=1}^{n}\frac{\mathrm{d}d_j\left(z\right)}{\mathrm{d}z}\frac{1}{d_j\left(z\right)}. \tag{4.6}$$

ここで，$\lambda_i\ (i = 1, 2, \ldots, s)$ を多項式 $d_j(z)$ の相異なる零点とする．$d_j(z)$ は $d_{j-1}(z)$ で割り切れることから，

$$d_j(z) = h_j(z)\prod_{i=1}^{s}(z-\lambda_i)^{\alpha_{ji}},$$

と表すことができる．このとき，$\alpha_{ji} \in \mathbb{Z}^+$ で，$h_j(z)$ は $z \in \Omega$ で非零となる解析的な関数で

ある．このことから，式 (4.6) は以下の式で表せる．

$$\sum_{j=1}^{n} \frac{\mathrm{d}d_j(z)}{\mathrm{d}z} \frac{1}{d_j(z)} = \sum_{i=1}^{s} \frac{\sum_{j=1}^{n} \alpha_{ji}}{z-\lambda_i} + \sum_{j=1}^{n} \frac{\mathrm{d}h_j(z)}{\mathrm{d}z} \frac{1}{h_j(z)} . \tag{4.7}$$

式 (4.7) に対して周回積分を行うと，留数定理より以下の式が導き出せる．

$$\begin{aligned}
&\frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} \left(\sum_{i=1}^{s} \frac{\sum_{j=1}^{n} \alpha_{ji}}{z-\lambda_i} + \sum_{j=1}^{n} \frac{\mathrm{d}h_j(z)}{\mathrm{d}z} \frac{1}{h_j(z)} \right) \mathrm{d}z \\
&\quad = \sum_{k=1}^{t} \left(\lim_{z \to \lambda_k} \sum_{i=1}^{s} \frac{\sum_{j=1}^{n} \alpha_{ji}}{z-\lambda_i} (z-\lambda_k) \right) + \quad \sum_{k=1}^{t} \left(\lim_{z \to \lambda_k} \sum_{j=1}^{n} \frac{\mathrm{d}h_j(z)}{\mathrm{d}z} \frac{1}{h_j(z)} (z-\lambda_k) \right) \\
&\quad = \sum_{k=1}^{t} \sum_{j=1}^{n} \alpha_{ji} = m_{\mathrm{R}},
\end{aligned}$$

となる．ここで，t は閉曲線 Γ 内部の相異なる固有値数を表しており，$\sum_{j=1}^{n} \alpha_{ji}$ は固有値 λ_i の重複度を表している．以上により，式 (4.1) は閉曲線 Γ 内部の固有値数を求める．

拡張法の計算機上での計算には，2.1 章で述べた N 点台形則を用いる．それにより式 (4.1) は式 (4.8) に近似される．

$$m_{\mathrm{R}} \approx \hat{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\mathrm{tr} \left(F(z_j)^{-1} F'(z_j) \right) \right), \quad F'(z_j) = \left. \frac{\mathrm{d}F(z)}{\mathrm{d}z} \right|_{z=z_j} . \tag{4.8}$$

ここで，z_j は中心 γ，半径 ρ の円 Γ 上に等間隔に N 個配置させた点であり，w_j は各積分点に対応した重みである．それらは以下の式で表される．

$$w_j = \frac{\rho}{N} \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}, \qquad z_j = \gamma + \rho \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}} .$$

式 (4.8) について，2.1 章と同様に行列のトレースの確率的推定を行う．定理 2.2 で対象とする行列は対称行列である．しかし $F(z)$ は非対象となる場合が存在する．そのため，対象とする行列が非対称行列の場合でも定理 2.2 が成り立つことを示す．

行列 A が非対称行列の場合，

$$\mathrm{tr}A = \frac{1}{2} \mathrm{tr} \left(A + A^{\mathrm{T}} \right)$$

となる．ここで $\left(A + A^{\mathrm{T}}\right)$ は対称行列なので，定理 2.2 から

$$\begin{aligned}\frac{1}{2}\mathrm{tr}\left(A + A^{\mathrm{T}}\right) &= \frac{1}{2}\mathrm{E}\left(\boldsymbol{v}^{\mathrm{T}}\left(A + A^{\mathrm{T}}\right)\boldsymbol{v}\right)\\ &= \frac{1}{2}\mathrm{E}\left(2\left(\boldsymbol{v}^{\mathrm{T}}A\boldsymbol{v}\right)\right)\\ &= \mathrm{E}\left(\boldsymbol{v}^{\mathrm{T}}A\boldsymbol{v}\right)\end{aligned}$$

が成り立つ．これにより，行列 A が非対称行列の場合でも定理 2.2 が成り立つ．

以上により式 (4.8) は以下の式で近似できる．

$$\hat{m}_{\mathrm{R}} \approx \tilde{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\frac{1}{L}\sum_{\ell=1}^{L}\left(\boldsymbol{v}_\ell^{\mathrm{T}} F\left(z_j\right)^{-1} F'\left(z_j\right)\boldsymbol{v}_\ell\right)\right). \tag{4.9}$$

式 (4.9) は $F'\left(z_j\right)\boldsymbol{v}_\ell$ を右辺ベクトルとした線形方程式

$$F\left(z_j\right)\boldsymbol{x}_{j,\ell} = F'\left(z_j\right)\boldsymbol{v}_\ell \quad (j = 0, 1, \ldots, N-1, \quad \ell = 1, 2, \ldots, L)$$

を解き，その解ベクトルを用いて以下のように計算することができる．

$$\tilde{m}_{\mathrm{R}} = \sum_{j=0}^{N-1} w_j \left(\frac{1}{L}\sum_{\ell=1}^{L}\left(\boldsymbol{v}_\ell{}^{\mathrm{T}}\boldsymbol{x}_{j,\ell}\right)\right).$$

そのため，逆行列の計算をする必要が無く，L の数が $F(z_j)$ の次数よりも小さい場合，式 (4.8) の計算を行うよりも高速な計算が可能となる．Algorithm 7 に拡張法のアルゴリズムを示す．

拡張法は，2.1 節で述べた一般化固有値問題に対する有理式型推定法と同様に，複数の閉曲線 $\Gamma_1, \Gamma_2, \ldots$ を配置し，各閉曲線に対して式 (4.9) の計算を行い，固有値の数を推定することで，固有値分布を推定する．このとき閉曲線を多くとることでより詳細な固有値の分布を求めることができる．

4.3 数値実験

本節では非線形固有値問題に対する有理式型推定法の有効性を検証する．対象とする行列は表 4.1 に示す．Butterfly は NLEVP[37] の行列”Butterfly”である．Quantum Dot（以下 QD）は量子ドットの電子状態計算から現れる 5 次多項式固有値問題 [1] である．Delay-Differential Equation (以下 DDE) は遅延微分方程式の数値計算から現れる要素に指数関数を含む固有値問

Algorithm 7 非線形固有値問題における有理式型推定法のアルゴリズム

Input: $F(z), N, L, \gamma, \rho$

Output: $\tilde{m}_{\mathrm{R}}$

set $\boldsymbol{v}_\ell$ of which the elements take 1 or -1 with equal probability for $\ell = 1, 2, \ldots, L$
$\tilde{m}_{\mathrm{R}} \leftarrow 0$
for $j = 0$ to $N - 1$ **do**
  $z_j \leftarrow \gamma + \rho \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}$
  $w_j \leftarrow \frac{\rho}{N} \mathrm{e}^{\frac{2\pi \mathrm{i}(j+1/2)}{N}}$
  for $\ell = 1$ to L **do**
    solve $F(z_j)\boldsymbol{x}_{j,\ell} = F'(z_j)\boldsymbol{v}_\ell$ for $\boldsymbol{x}_{j,\ell}$
    $\tilde{m}_{\mathrm{R}} \leftarrow \tilde{m}_{\mathrm{R}} + w_j(\boldsymbol{v}_\ell^{\mathrm{T}} \boldsymbol{x}_{j,\ell})/L$
  end for
end for

題 [2] である．Accelerator Designs（以下 SLAC）は高エネルギー加速器の設計から発生する，要素に平方根を含む固有値問題 [33] である．数値実験は，

CPU: MacBook Air 1.3GHz Intel Core i5,

Memory: 8GB,

OS: Mac OS ver.10.9.5,

上で行い，MATLAB8.5.0 を用いた．線形方程式の数値計算は MATLAB の関数である `mldivide` を用いた．サンプルベクトル $\boldsymbol{v}_\ell$ は MATLAB のコマンドである `rand` を用いており，そのシード値は ‘twister’ の 0 シードを用いた．本実験では非線形固有値問題に対応した Sakurai-Sugiura 法 [38, 39] によって求めた固有値を実際の固有値とした．

表 4.1: 数値実験で対象とする行列

	$F(\lambda)$	Size
Butterfly	$\sum_{i=0}^{4} \lambda^i A_i$	64
QuantumDot (QD)	$\sum_{i=0}^{5} \lambda^i A_i$	2475
Delay-Differential Equation (DDE)	$\lambda I - A_0 - A_1 e^{-\tau\lambda}$	3600
Accelerator Designs (SLAC)	$A_0 - \lambda A_1 + \mathrm{i}\sqrt{\lambda - \sigma^2}\mathrm{A}_2$	5384

各問題における行列 $A_0, A_1, A_2, A_3, A_4, A_5$ はそれぞれ異なる

4.3.1 数値実験 4-1

本項では非線形固有値問題に対する有理式型推定法による推定値が，積分点数によってどのように変化するのかを調べる．対象とする行列の積分路 Γ の中心 γ，半径 ρ を表 4.2 に示す．積分点数 $N = 4, 6, 8, 16, 32, 64$ と変化させ，式 (4.8) の $\hat{m}_{\mathrm{R}}$ がどのように変化するかを調べる．

実験結果を表 4.3 に示す．実験結果から，全ての $\hat{m}_{\mathrm{R}}$ について，一桁目は実際の固有値の数と一致している．このことから，式 (4.8) によって閉曲線内部の固有値の数が近似できることがわかる．そして，積分点数を増やしても推定値 $\hat{m}$ の値にはあまり変化が見られない．提案手法である固有値数推定計算は，$N \times L$ の線形方程式を求解する必要があり，計算コストを抑えるためには，N は小さい方が望ましい．以上により，固有値分布推定に用いる N の値は 4 から 8 程度であればよいと考えられる．

表 4.2: 数値実験 4-1 で用いる閉曲線 Γ のパラメータ

	γ	ρ
Butterfly	$1 + 0.7\mathrm{i}$	0.5
QD	1	0.06
DDE	$-4.3 + 6.3\mathrm{i}$	0.2
SLAC	360000	25000

表 4.3: 数値実験 4-1 の実験結果 $\hat{m}_{\mathrm{R}}$

N	Butterfly	QD	DDE	SLAC
4	26.65	35.54	22.77	10.48
6	27.27	33.10	23.30	10.35
8	27.10	32.23	23.55	10.97
16	28.50	31.12	23.69	9.89
32	27.30	30.81	24.97	9.98
64	28.19	31.08	23.75	10.00
実際の値	28	31	24	10

4.3.2 数値実験 4-2

本項では行列のトレースの確率的推定による推定値が，ベクトル $\boldsymbol{v}_\ell$ の数 L によってどのように変化するのかを調べる．本項では，異なる 30 通りのランダムシーケンスを用いて，式

(4.9) の

$$\tilde{t}_j = \frac{1}{L}\sum_{\ell=1}^{L}\left(\boldsymbol{v}_\ell^{\mathrm{T}} F\left(z_j\right)^{-1} F'\left(z_j\right)\boldsymbol{v}_\ell\right).$$

を L を 1 から 100 まで計算し，L の値ごとにランダムシーケンス同士の平均と標準偏差を計算し，その値の変化を調べる．各行列値関数 $F(z_j)$ に対する z_j の値は表 4.4 に示す．

実験結果を図 4.1，図 4.2，図 4.3，図 4.4 に示す．横軸はベクトル $\boldsymbol{v}_\ell$ の数 L の値を示し，縦軸は平均値と標準偏差を示している．破線は $\tilde{t}_j$ の平均値を表しており，点線は $\tilde{t}_j$ の標準偏差を表しており，実線は実際の行列のトレースの計算結果を示している．実験結果から，$\tilde{t}_j$ の標準偏差は L が大きくなるにつれて 0 に近くなっている．また，全ての計算結果について，L が 0 から 30 の値の間で $\tilde{t}_j$ 標準偏差が急激に減少し，その後は減少幅が小さくなっている．このことから，$L = 30$ 以上であれば異なるランダムシーケンスを用いた場合でも，式 (4.9) の値はお互いに近い値となることがわかる．行列のトレースの確率的推定の近似精度は，$F(z)^{-1}F'(z)$ の対角優位性に依存する．そのため，$F(z)^{-1}F'(z)$ の対角要素の値が非対角要素の値よりも十分に大きい行列（SLAC）は L が小さい値でもトレースの値に近づき，対角要素の値があまり大きくない行列（Butterfly，QD，DDE）はトレースの値に近づくのが遅い．

表 4.4: 数値実験 4-2 の z_j の値

	Butterfly	QD	DDE	SLAC
z_j	$1 + 0.7i$	0.1	$-3 + 6i$	10

4.3.3 数値実験 4-3

本項では非線形固有値問題に対する有理式型推定法による推定値が，ベクトル $\boldsymbol{v}_\ell$ の数 L によってどのように変化するのかを調べる．対象とする行列の積分路 Γ は数値実験 1 と同様の値を用いる．積分点数 $N = 8$ とし，ベクトル $\boldsymbol{v}_\ell$ の数 L を 10 から 1000 まで変化させて，式 (4.9) の $\tilde{m}_{\mathrm{R}}$ を計算し，その値がどのように変化するかを調べる．

実験結果を表 4.5 に示す．実験結果から，式 (4.9) の $\tilde{m}_{\mathrm{R}}$ の値は，式 (4.8) の $\hat{m}_{\mathrm{R}}$ に近い値を示している．このことから，非線形固有値問題に対する有理式型推定法は，閉曲線内部の固有値の数を近似できていることがわかる．また，本実験で示した $\tilde{m}$ と $\hat{m}$ との差と，数値実験 1 で示した $\hat{m}$ と実際の値との差を比べると，$\tilde{m}$ と $\hat{m}$ との差の方が大きい．このことから，周回積分の近似による誤差の値より，行列のトレースの確率的推定による誤差の方が大きいこ

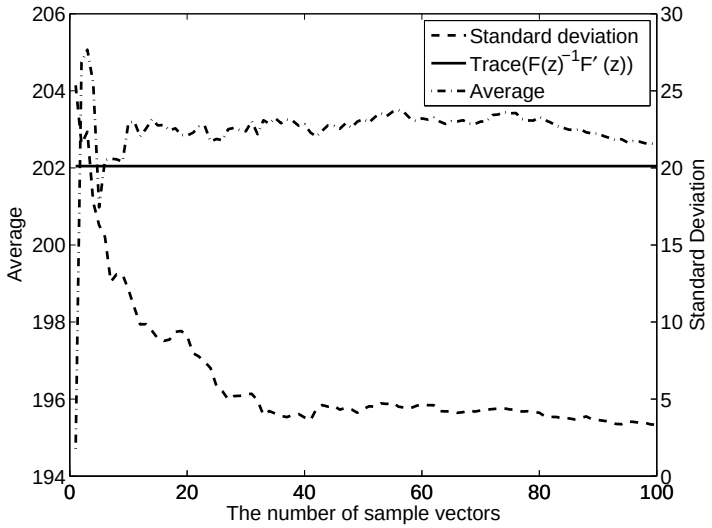

図 4.1: 数値実験 4-2 の実験結果 (Butterfly)

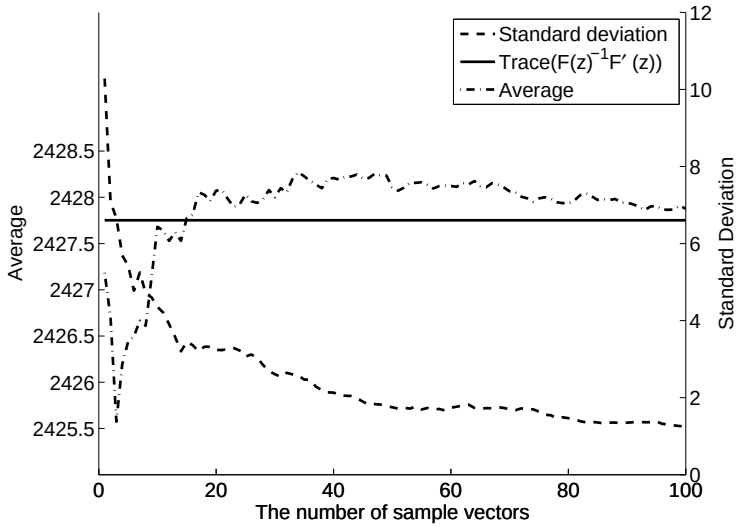

図 4.2: 数値実験 4-2 の実験結果 (QD)

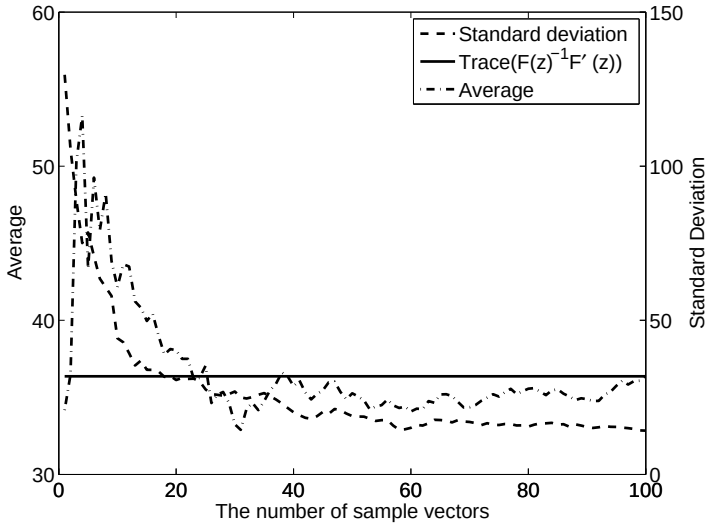

図 4.3: 数値実験 4-2 の実験結果 (DDE)

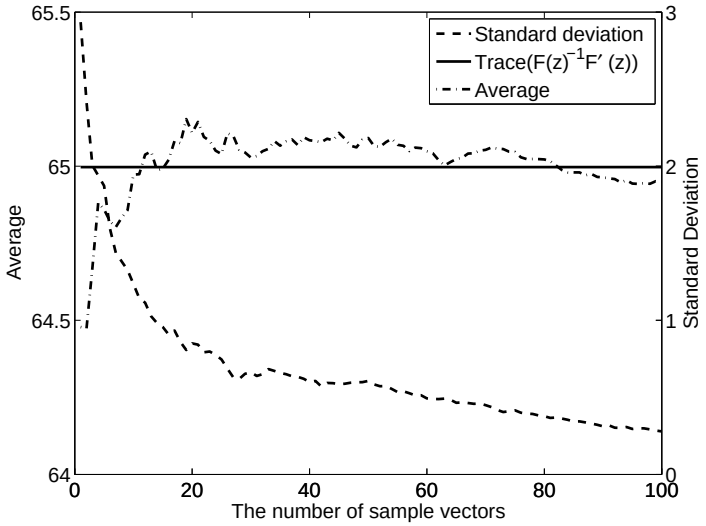

図 4.4: 数値実験 4-2 の実験結果 (SLAC)

とがわかる．そして，L の値を大きくしても，$\tilde{m}$ の値にあまり変化が見られない．4.3.1 項で述べたように，L の値は計算する線形方程式の数に影響するため，L は小さい方が望ましい．また 4.3.2 項で述べたように，$L = 30$ 以上であれば，行列のトレースの推定値はどのランダムシーケンスでも近い値を取る．そのため，L の値は 30 程度であればよいと考えられる．さらに，実験結果の値は実際の値と小数桁まで一致していないことから，線形方程式の求解に高精度な解は必ずしも必要でない．

表 4.5: 数値実験 4-3 の実験結果 $\tilde{m}_{\mathrm{R}}$

L	Butterfly	QD	DDE	SLAC
10	27.48	38.89	36.64	11.63
20	23.49	35.86	26.94	10.18
30	24.06	35.44	19.08	10.38
40	25.70	34.92	17.90	9.41
50	27.44	33.63	19.10	8.89
100	28.19	33.14	22.06	9.64
500	28.42	32.92	22.76	10.71
1000	27.46	33.15	23.65	10.50
$\hat{m}_{\mathrm{R}}$	27.10	32.23	23.55	10.97

4.3.4 数値実験 4-4

本項では，非線形固有値問題における有理式型推定法を用いて固有値の分布を推定し，その有効性を検証する．対象とする行列のパラメータを表 4.6 に示す．非線形固有値問題における有理式型推定法を用いて，複数の閉曲線に対する固有値数の推定を行う．閉曲線の配置は図 4.5 に示す．N, L の値は数値実験 4-1,4-2,4-3 から考察した結果を基に設定した．

実験結果を図 4.6，図 4.7，図 4.8，図 4.9 に示す．横軸は γ_k の実数の値を，縦軸は固有値の数を示しており，Exact は実際の固有値の数，Estimate は $\tilde{m}_{\mathrm{R}}$ の値を示している．実験結果から有理式型推定法による計算結果と実際の固有値数が近い値となっており，固有値の多い箇所，少ない箇所が推定されている．特に，固有値が存在しない箇所に対して, 推定値は存在していないことを示している．また，固有値が存在している箇所の固有値数に関しては，正確な固有値の数を推定していない箇所が存在するが，固有値の粗密情報は推定されている．このことから，固有値分布推定法は固有値が存在している箇所，存在していない箇所の判別に利用することができ，また，固有値の粗密を推定できることがわかる．

表 4.6: 非線形固有値問題に対する有理式型推定法のパラメータ

	N	L	K	(a_0, b_0)	(a_1, b_1)
Butterfly	8	30	30	$(-3.2, 0.5)$	$(3.2, 0.5)$
QD	8	30	30	$(0, 0)$	$(2, 0)$
DDE	8	30	30	$(-10, 0)$	$(5, 0)$
SLAC	8	30	30	$(0.02 \times 10^6, 0)$	$(1.02 \times 10^6, 0)$

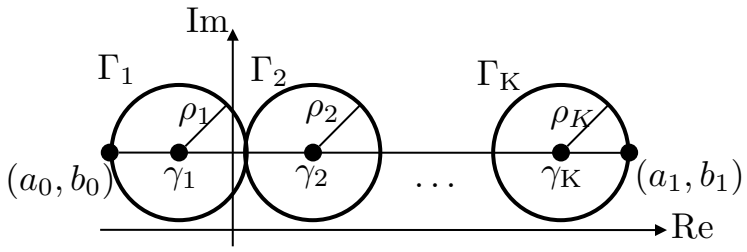

図 4.5: 数値実験 4-4 の閉曲線の配置

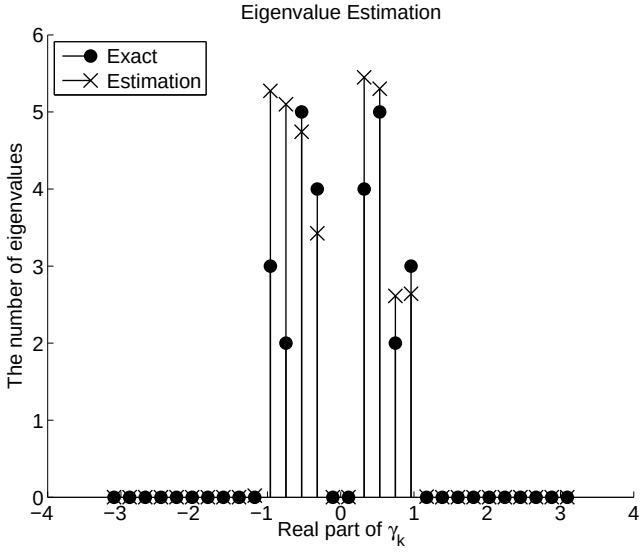

図 4.6: 数値実験 4-4 の実験結果（Butterfly）

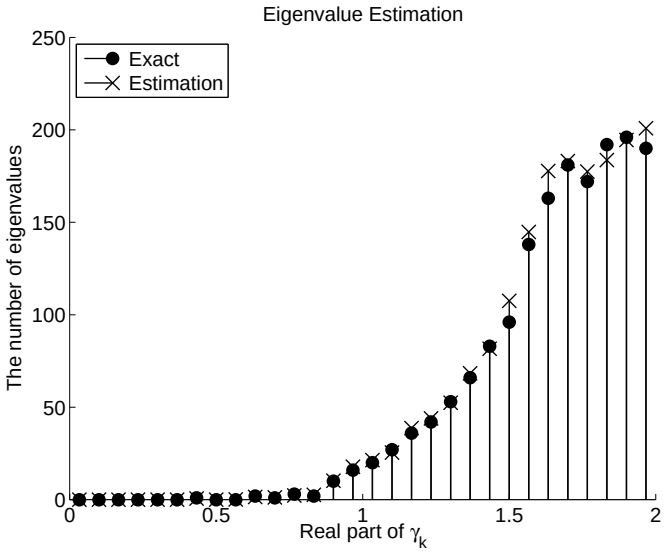

図 4.7: 数値実験 4-4 の実験結果（QD）

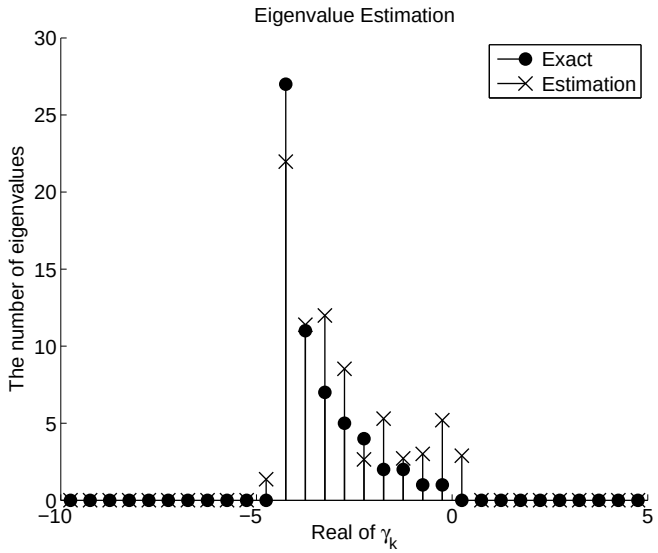

図 4.8: 数値実験 4-4 の実験結果（DDE）

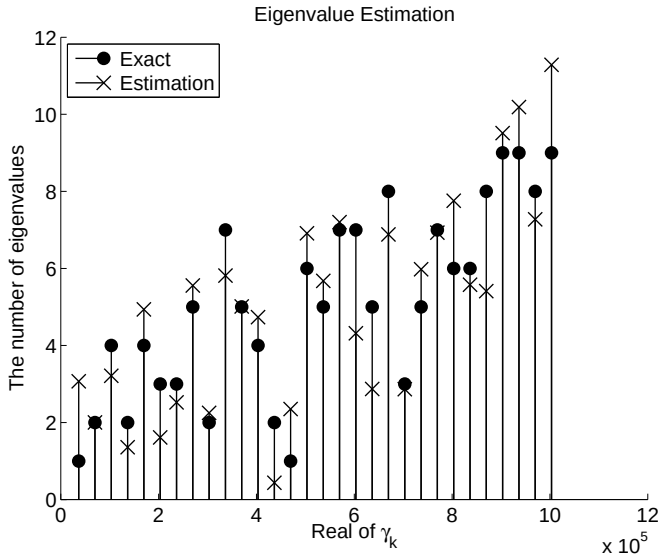

図 4.9: 数値実験 4-4 の実験結果（SLAC）

4.4 小括

本章では非線形固有値問題に対する，固有値の分布を推定する方法として，有理式型推定法を拡張した方法を提案した．これにより，有理式型推定法が標準・一般化固有値問題以外に，非線形固有値問題に適用可能であることを示した．数値実験から，有理式型推定法において，積分点数は 4 点もしくは 8 点程度で固有値数を推定することができることを示した．また，行列のトレースの確率的推定によって近似される値は，ランダムベクトル $\boldsymbol{v}_\ell$ の本数が $L = 30$ 以上であれば，どのようなランダム値であっても，互いに近い値となることを示した．さらに，有理式型推定法において，ランダムベクトル $\boldsymbol{v}_\ell$ の本数が $L = 30$ 程度であれば固有値数が推定できることを示した．この結果より，非線形固有値問題における有理式型推定法は，非線形固有値問題の固有値の分布情報を効果的に求めることができる．

第5章　円弧近傍に対する確率的固有値分布推定法

2 章，3 章，4 章では確率的固有値分布推定法の一つである，有理式型推定法について述べた．有理式型推定法では任意の円領域内部の固有値の数を推定した．本章では有理式型推定法を拡張し，任意の円弧近傍の固有値数を求める方法を提案する．

5.1　諸言

本章では一般化固有値問題

$$F(\lambda)\boldsymbol{x} = (\lambda B - A)\boldsymbol{x} = \boldsymbol{0}, \quad (A, B \in \mathbb{C}^{n\times n}, \boldsymbol{x} \in \mathbb{C}^n, z \in \mathbb{C})$$

および非線形固有値問題

$$F\left(\lambda\right)\boldsymbol{x} = \boldsymbol{0}, \quad (\boldsymbol{x} \in \mathbb{C}^n \backslash \{\boldsymbol{0}\}, z \in \mathbb{C}, F(\cdot) : \mathbb{C} \to \mathbb{C}^{n\times n})$$

における固有値の分布計算について考える．

このような固有値問題は，様々な科学技術計算において現れ，それらの問題には特定の領域の固有値の分布情報が必要となる問題がある．特に，ナノエレクトロニクスデバイスの構造解析から現れる固有値問題では，複素平面上の特定の円の円弧近傍の固有値分布が必要となる [3]．本章では，そのような円弧近傍の固有値分布を求める問題について考える．

2 章で述べたように，確率的固有値分布推定法の一つに有理式型推定法がある．有理式型推定法は，指定した閉曲線内部の固有値数を推定する方法であり，特定の領域の固有値数を推定することが可能となる．しかし，有理式型推定法では閉曲線に円を用いているため，円弧近傍の固有値数を求める場合には，複数の閉曲線を配置した計算などが必要となり，計算コストが増加する問題が発生する．

本論文ではそのような問題に対して，実数区間に対するフィルタ関数を応用し，円弧近傍の固有値数を推定する拡張方法を提案する．提案法により，円弧近傍の固有値数を効率的に

求めることができる．

本章ではまず，実数区間に対するフィルタ関数について述べる．次に，そのフィルタ関数を応用し，円弧近傍の固有値数を推定する有理式型推定法の拡張ついて述べる．さらに，数値実験によって，拡張手法の有効性を検証する．

5.2 実数区間に対するフィルタ関数

有理式型推定法では，式 (2.12) の $f(\lambda_i)$ によってフィルタ関数を作成している．このフィルタ関数 $f(\lambda_i)$ は，指定した区間外部の固有値を減衰する関数となっており，これによって固有値数を推定している．フィルタ関数は積分点 $z_j,\ j = 0, 1, \ldots, N-1$ および積分点に対応する重み $w_j\ j = 0, 1, \ldots, N-1$ によって作成されており，これらの値を変化することで減衰する固有値が変わる．文献 [40, 41, 42] では，実軸上に配置した Chebyshev 点によってある実数区間近傍の固有値を透過させるフィルタ関数を作成している．

線分 $[-1, 1]$ に対して，その線分近傍の固有値を透過するフィルタ関数を考える．このとき z_j, w_j を以下のように設定することで，線分 $[-1, 1]$ 外部では 0 に減衰するフィルタ関数が作成される [40, 41, 42].

$$z_j = \cos\left(\frac{2j-1}{2N}\pi\right), \quad w_j = \frac{T_{N-1}(z_j)}{N}. \tag{5.1}$$

ここで，$T_j(x)$ は以下に定義される第 1 種 Chebyshev 多項式であり，z_j は $T_N(x)$ の零点である．

$$T_j(x) = \begin{cases} 1, & (j = 0) \\ x, & (j = 1) \\ 2xT_{j-1}(x) - T_{j-2}(x), & (j \geq 2) \end{cases}.$$

以上の z_j, w_j を用いて線分近傍に対するフィルタ関数 $f^{(\mathrm{S})}(\lambda_i)$ を計算することにより，フィルタ関数は $f^{(\mathrm{S})}(\lambda_i) = 1/T_N(\lambda_i)$ で表される．多項式型推定法によるフィルタ関数 $P_k(\lambda_i)$ においても，線分近傍に対するフィルタ関数 $f^{(\mathrm{S})}(\lambda_i)$ と同様に，第 1 種 Chebyshev 多項式を用いているが，$f^{(\mathrm{S})}(\lambda_i)$ は，第 1 種 Chebyshev 多項式の逆数を用いているため，$P_k(\lambda)$ とは異なるフィルタ関数となる．

図 5.1，図 5.2 に線分近傍に対するフィルタ関数 $f^{(\mathrm{S})}(\lambda_i)$ を示す．式 (5.1) を用いて，$f^{(\mathrm{S})}(\lambda_i)$ を計算した．積分点数 $N = 8$ とした．図 5.1 について，横軸は実数軸，縦軸はフィルタ関数値を表している．実数値 -1 および 1 を赤点線で示し，フィルタ関数を $f^{(\mathrm{S})}(\lambda_i)$ の値を青線で示す．図 5.2 について，右下の軸は実数軸，左下の軸は虚数軸を表しており，フィルタ関

数 $f^{(\mathrm{S})}(\lambda_i)$ の値をメッシュで示す．図 5.1，図 5.2 から線分近傍に対するフィルタ関数 $f^{(\mathrm{S})}(\lambda_i)$ が，線分 $[-1,1]$ 外部では 0 に減衰することがわかる．また，図 2.7 に示す，多項式型推定法によるフィルタ関数 $P_k(\lambda_i)$ とは異なるフィルタ関数となることがわかる．

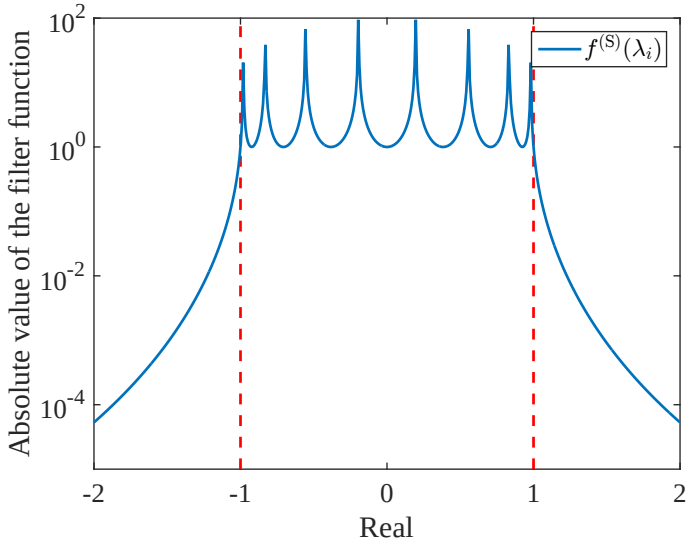

図 5.1: 線分 $[-1,1]$ に対するフィルタ関数 ($N = 8$)

以上の積分点 z_j および積分点に対応する重み w_j を用いて，式 (2.3) を計算することにより，線分 $[-1,1]$ 近傍近傍の固有値数を推定することができる．

5.3 円弧近傍に対する有理式型推定法の拡張

前節では線分近傍に対するフィルタ関数について述べた．本節では線分近傍に対するフィルタ関数を応用し，円弧近傍に対するフィルタ関数を作成し，それを用いて有理式型推定法を拡張する．

図 5.3 に示すように，複素平面上の中心 γ，半径 ρ によって作成される円に対して，開始角度 θ_{a} および終了角度 θ_{b} によって作成される円弧線を $\mathbb{L}$ とする．

$$\mathbb{L} : z = \gamma + \rho \mathrm{e}^{\mathrm{i}\theta}, \quad \theta_{\mathrm{a}} \leq \theta \leq \theta_{\mathrm{b}}. \tag{5.2}$$

ここで，$0 \leq \theta_{\mathrm{a}} < \theta_{\mathrm{b}} \leq 2\pi$ である．このとき積分点 z_j および積分点に対する重み w_j を以下

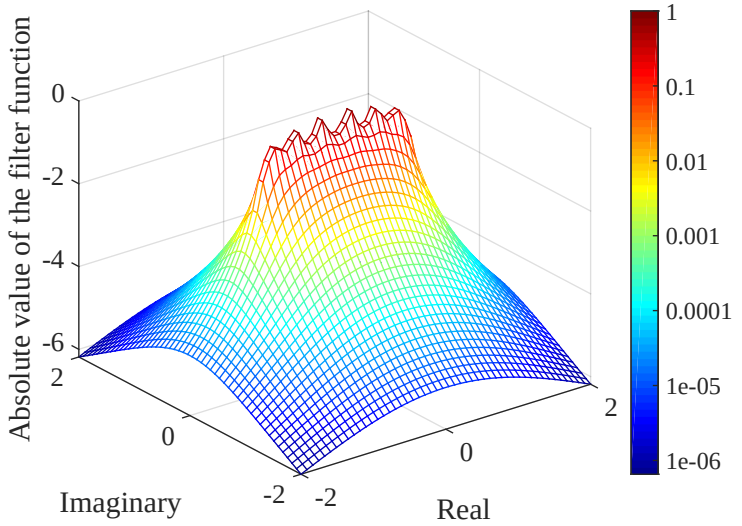

図 5.2: 複素平面上での線分 $[-1, 1]$ に対するフィルタ関数 ($N = 8$)

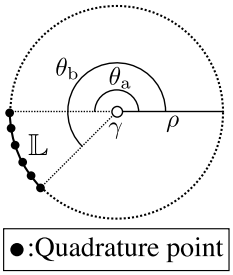

図 5.3: 複素平面上の円弧線 $\mathbb{L}$ および積分点 z_j の概略図

のように設定する．

$$z_j = \gamma + \rho \mathrm{e}^{\mathrm{i}\theta_j}, \quad w_j = \frac{T_{N-1}(z_j)}{N}, \tag{5.3}$$
$$\theta_j = \theta_{\mathrm{a}} + (\theta_{\mathrm{b}} - \theta_{\mathrm{a}})\frac{\zeta_j + 1}{2}, \quad \zeta_j = \cos\left(\frac{2j-1}{2N}\pi\right).$$

ζ_j および w_j は式 (5.1) の z_j および w_j と同じものを用いる．式 (5.3) により，実数区間 $[-1, 1]$ 上の積分点が円弧線 $\mathbb{L}$ 上にマッピングされる．図 5.3 に積分点の配置を示す．5.2 節で示したように，実数区間に対するフィルタ関数は線分 $[-1, 1]$ 外部では 0 に減衰するフィルタ関数が作成される．式 (5.3) では，実数区間 $[-1, 1]$ 上の積分点を円弧線 $\mathbb{L}$ 上にマッピングしているため，円弧線のフィルタ関数 $f^{\mathbb{L}}(\lambda_i)$ も円弧線から離れるにつれて 0 に減衰するフィルタ関数が作成されると考えられる．以上の積分点 z_j および積分点に対応する重み w_j を用いて，式 (2.3) を計算することにより，円弧線近傍の固有値数を推定することができる．また，4 章で示したように，非線形固有値問題に対する，有理式型推定法においても，積分点 z_j および積分点に対応する重み w_j を用いて円領域内部の固有値数を推定している．そのため，式 (5.3) の z_j, w_j を用いて式 (4.9) を計算することで，非線形固有値問題の円弧線上の固有値数を推定することができる．

拡張法の問題点として，積分点 z_j と固有値が完全に一致した場合は数値破綻が起こることが考えられる．実数区間に対するフィルタ関数は，積分点近傍で 1 よりも大きい値を示している．そのため，拡張法のフィルタ関数においても，積分点近傍では 1 よりも大きい値となると考えられる．それにより，z_j が固有値と限りなく近い場合，推定値に影響を及ぼすことが考えられる．

5.4 数値実験

本章ではいくつかの数値実験によって，提案法の有効性を検証する．数値実験は，MATLAB 8.3 を用いた．

本節では，円弧近傍に対する有理式型推定法の拡張法によって円弧近傍の固有値数を推定し，その性能を比較する．数値実験は，

CPU: MacBook Air 1.3GHz Intel Core i5,

Memory: 8GB,

OS: Mac OS ver.10.9.5,

上で行い，MATLAB8.5.0 を用いた．本実験において $\boldsymbol{v}_\ell$ は MATLAB の関数である `rand` を

用いており，そのシード値は ‘twister’ の 0 シードを用いた．また，線形方程式の数値計算は MATLAB の関数である `mldivide` を用いた．本実験では MATLAB の関数である `eig` によって求めた固有値を実際の固有値とした．

5.4.1 数値実験 5-1

本項では，円弧近傍に対する有理式型推定法によって作成されるフィルタ関数が積分点数によってどのように変化するのかを調べる．円弧線 $\mathbb{L}$ のパラメータについて $\gamma = 0,\ \rho = 1,\ \theta_{\mathrm{a}} = 0,\ \theta_{\mathrm{b}} = \pi$ とする．積分点数 N を $N = 4, 8, 16$ と変化させ拡張法のフィルタ関数 $f^{\mathbb{L}}(\lambda_i)$ の変化を調べた．

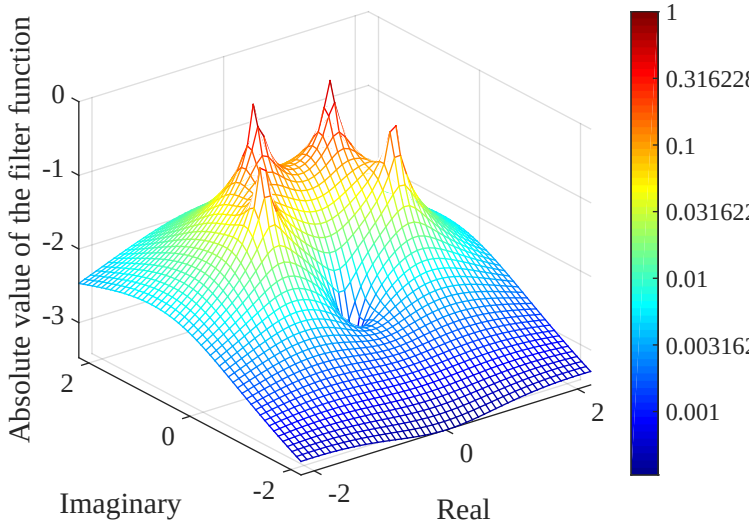

図 5.4: 複素平面上での円弧近傍に対する有理式型推定法のフィルタ関数 $(N = 4)$

図 5.4，図 5.5，図 5.6 に複素平面上での拡張手法によって得られる複素平面上でのフィルタ関数を示す．右下は実数軸，左下は虚数軸を表しており，拡張手法によるフィルタ関数の絶対値 $|f^{\mathbb{L}}(\lambda_i)|$ を色で表している．拡張手法によるフィルタ関数は，指定した円弧から外部に行くにつれて減衰していることがわかる．また，積分点数を増やすことで，減衰が強くなることがわかる．図 5.7，に実数軸上での拡張手法によって得られる複素平面上でのフィルタ関数を示す．横軸は実数軸，縦軸はフィルタ関数の絶対値 $|f^{\mathbb{L}}(\lambda_i)|$ を表している．図 5.7 からも，実数部が $-1,\ 1$ 付近においてフィルタ関数 $|f^{\mathbb{L}}(\lambda_i)|$ が 1 に近く，外部では 0 に減衰していることがわかる．また，積分点数を増やすことで，減衰が強くなっていることがわかる．以上

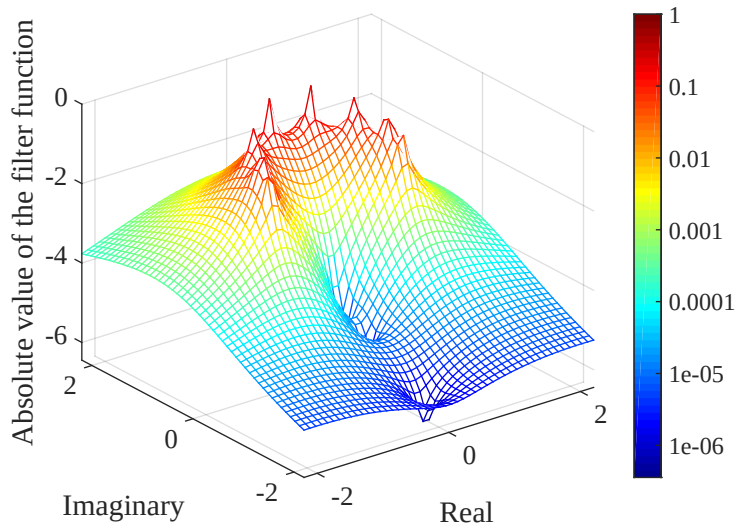

図 5.5: 複素平面上での円弧近傍に対する有理式型推定法のフィルタ関数 ($N = 8$)

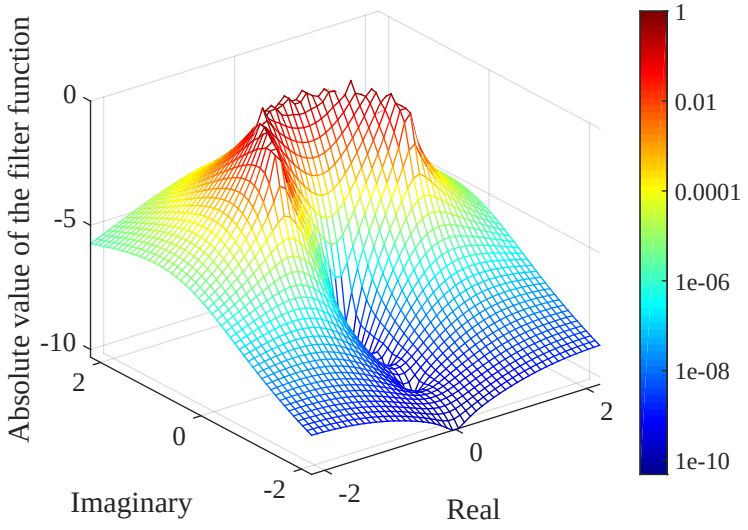

図 5.6: 複素平面上での円弧近傍に対する有理式型推定法のフィルタ関数 ($N = 16$)

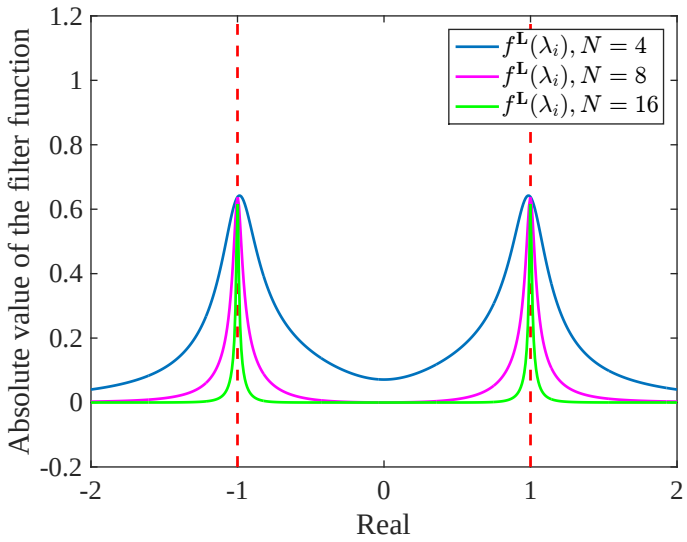

図 5.7: 実数軸上での円弧近傍に対する有理式型推定法のフィルタ関数

のことから，拡張手法が円弧近傍の固有値のみを透過するフィルタ関数を作成している．

5.4.2 数値実験 5-2

本項では円弧近傍に対する有理式型推定法による推定値が，積分点数によってどのように変化するのかを調べる．表 5.1 に対象とする固有値問題を示す．各行列について，Sample は複素平面上の中心 0，半径 1 の円周上に等間隔に置いた 64 点と，複素平面上の中心 0，半径 0.8 の円内の乱数 936 点を要素に持つ対角行列である．OLM5000 は Matrix market[43] の行列 “OLM5000” である．図 5.8，図 5.9 に各行列の固有値分布を示す．

表 5.1: 数値実験 5-2 で対象とする固有値問題

	行列タイプ	固有値問題	n
Sample	複素対称	標準固有値問題	1000
OLM5000	実非対称	標準固有値問題	5000

本項では，積分点数 $N = 4, 8, 16, 32, 64$ と変化させて，式 (5.3) の z_j, w_j を用いて式 (2.3) を計算して，その推定値の変化を調べる．表 5.1 に実験で用いた領域のパラメータを示す．その他のパラメータとして $L = 32$ とする．

実験結果を表 5.3 に示す．実験結果から，積分点数 N を増やすことで推定値が実際の値に

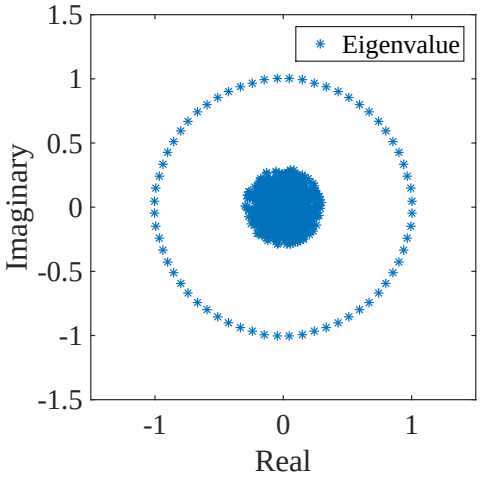

図 5.8: 行列 Sample の固有値分布

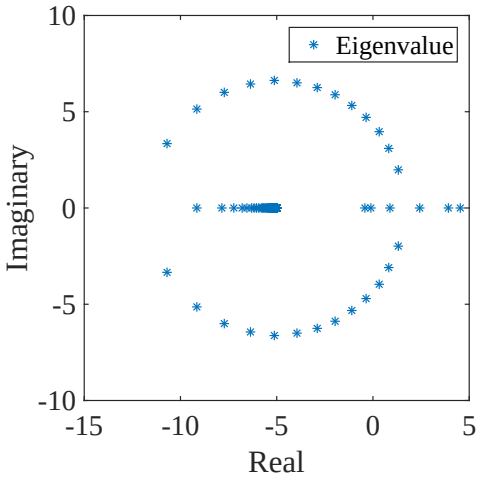

図 5.9: 行列 OLM5000 の固有値分布

表 5.2: 数値実験 5-2 のパラメータ

	次元数	$(\gamma, \rho, [\theta_{\mathrm{a}}, \theta_{\mathrm{b}}])$
Sample	1000	$(0, 1, [0, \pi])$
OLM5000	5000	$(-5, 6.6, [0, \pi])$

表 5.3: 積分点数 N の変化による有理式型推定法の推定値 $\tilde{m}_{\mathrm{R}}$ の変化

N	Sample	OLM5000
4	61.92	170.93
8	6.27	5.47
16	29.33	10.93
32	22.33	13.50
64	32.91	9.95
実際の数	32	13

近づいていくことがわかる．特に行列 Sample では $N = 16$ 以上で推定値に近い値が得られており，行列 OLM5000 では $N = 8$ 以上で推定値に近い値が得られていることがわかる．以上により，拡張手法によって円弧近傍の固有値の数が推定できていることがわかる．

5.5 小括

本章では実数区間に対するフィルタ関数について述べ，フィルタ関数を有理式型推定法に応用できることを示した．また，実数区間に対するフィルタ関数の積分点を円弧上にマッピングすることにより，円弧近傍の固有値が透過されるフィルタ関数が作成できることを示した．そして，円弧近傍に対するフィルタ関数を有理式型推定法に応用した拡張手法を示した．数値実験から，円弧近傍に対するフィルタ関数は円弧外部で 0 に減衰することがわかった．特に，積分点数を増やすことによって，円弧外部ではより 0 に減衰するフィルタ関数が作成されることがわかった．また，拡張法によって円弧近傍の固有値数が推定できることがわかった．この結果より，円弧近傍に対する有理式型推定法は，円弧近傍の固有値数を効果的に求めることができる．

第6章　確率的固有値分布推定法の並列実装

これまでの章では，確率的固有値分布推定法の一つである有理式型推定法および，その拡張法について述べてきた．本章では，マスター・ワーカ方式並列処理を用いて，並列計算機での有理式型推定法の効率的な実装手法を示し，高速化を図る．

6.1　序説

本章では一般化固有値問題

$$F(\lambda)\boldsymbol{x} = (\lambda B - A)\boldsymbol{x} = \boldsymbol{0}, \quad (A, B \in \mathbb{C}^{n\times n}, \boldsymbol{x} \in \mathbb{C}^n, z \in \mathbb{C})$$

および非線形固有値問題

$$F\left(\lambda\right)\boldsymbol{x} = \boldsymbol{0}, \quad (\boldsymbol{x} \in \mathbb{C}^n \backslash \{\boldsymbol{0}\}, z \in \mathbb{C}, F(\cdot) : \mathbb{C} \to \mathbb{C}^{n\times n})$$

における有理式型推定法の並列計算ついて考える．

2 章で述べたように，有理式型推定法では複数の閉曲線を設定し，各閉曲線で固有値数を推定することで，固有値の分布を推定している．このとき各閉曲線での固有値数推定法は独立に計算できるため閉曲線数分の並列計算が可能である．また，有理式型推定法は，計算過程で積分点 N ごとにベクトル $\boldsymbol{v}_\ell$ の本数 L 本の線形方程式の計算が発生する．このとき各積分点での線形方程式の計算は独立に計算できるため，$N \times L$ 個分の並列計算が可能である．これにより，十分な計算ノードが存在する場合，(閉曲線数) × (積分点数) × ($\boldsymbol{v}_\ell$の本数) 個の並列計算が可能となるため，非常に並列性が高い．

また，3 章で述べたように，線形方程式の求解に反復法の一つである Krylov 部分空間反復法を用いることで，行列分解を行わずに固有値数を推定することが可能となる．さらに，反復過程で固有値数の推定値を得ることにより，高速に推定値を求めることができる．そのとき，線形方程式の解の精度は低くなるが，有理式型推定法による推定値も近似値であることから，解の精度が低くても問題はない．

しかしながら，線形方程式の求解に Krylov 部分空間反復法を用いた場合，閉曲線ごとに反復回数が異なるため，図 6.1 に示すように，待機する計算資源が生じ，ロードバランスが悪くなる問題が発生する．

本章ではまず，ロードバランスを軽減させるためにマスター・ワーカ方式を用いた有理式型推定法の並列実装について述べる．次に，複数の閉曲線によって作成される，積分点の総数を減らす実装アルゴリズムの提案を行う．さらに，数値実験によって実装アルゴリズムの有効性を検証する．本章では，有理式型推定法の閉曲線は円を用いる．

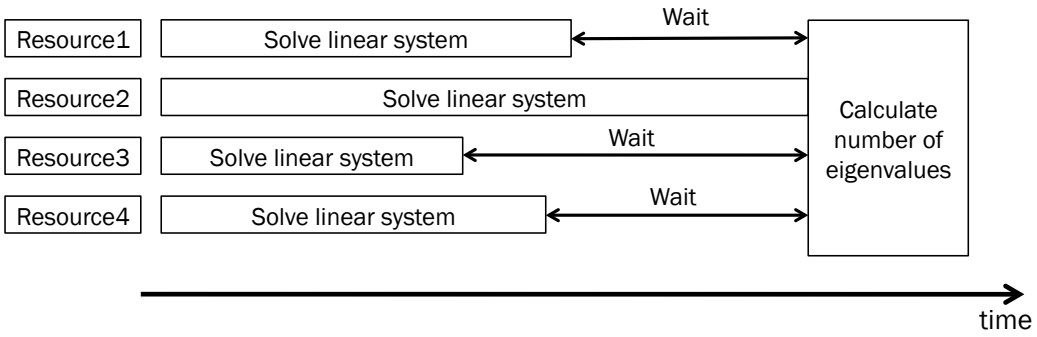

図 6.1: 有理式型推定法の並列計算モデル例：Wait が待機時間を示す

6.2 マスター・ワーカ方式並列計算を用いた有理式型推定法の並列実装

前節で示したように，有理式型推定法から現れる線形方程式に対して並列実装を行う場合，ロードバランスが悪くなる問題が発生する．本節では，この問題を解決するために，マスター・ワーカ方式を用いた実装アルゴリズムについて述べる．本論文では，有理式型推定法におけるマスター・ワーカ方式を用いた実装アルゴリズムを完全形メッシュアルゴリズムと呼ぶ．

6.2.1 マスター・ワーカ方式並列計算

マスター・ワーカ方式とは，並列実行時の複数の計算プロセスをマスターとワーカに分ける並列実装方式である．マスターとワーカーに割り当てられたプロセスはそれぞれ処理が異なる．マスターはワーカへのタスク割り当てと結果の管理を行い，ワーカは割り当てられたタスクの処理を行う．図 6.2 にマスター・ワーカ方式並列処理の概略図を示す．本論文では，マスターのプロセス数を 1 とし，ワーカのプロセス数を残りのプロセス数 N_{w} とする．

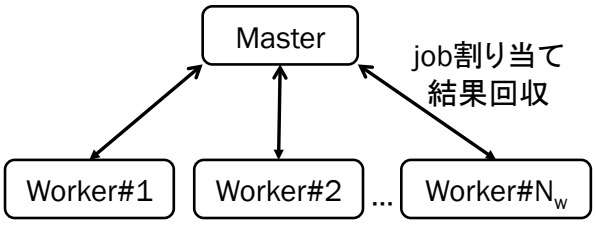

図 6.2: マスター・ワーカ方式並列処理の概略図

6.2.2 完全型メッシュアルゴリズム

完全型メッシュアルゴリズムとはマスター・ワーカ方式並列処理を用いて有理式型推定法を計算する並列計算手法である．完全型メッシュアルゴリズムでは，図 6.3 のように一様に閉曲線を配置し，積分点数を $N = 4$ とする．これにより，積分点を正方格子状になり，積分点ごとの線形方程式の計算結果を共有することができる．このとき生成される格子を完全型メッシュと呼び，完全型メッシュによって生成される総積分点数を N_{all} とする．

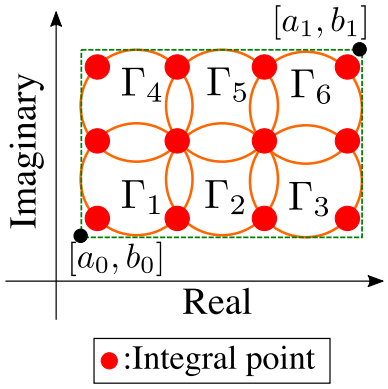

図 6.3: 有理式型推定法における閉曲線およびその積分点の配置

初めにマスターは N_{all} 個のタスクをタスクセットに追加する．ここで，タスクはある 1 つの積分点における L 本分の線形方程式の求解とし，タスクセットは未処理のタスクの集合とする．マスターはタスクセットの中からワーカ数に応じて，タスクの処理を各ワーカに命じる．各ワーカは受け取ったタスクの情報に基づいて線形方程式を反復解法で解き，収束判定を満たしたとき，以下の式の $\tilde{t}_{jk}$ をマスターに送る．

$$\mathrm{tr}\left(F\left(z_{jk}\right)^{-1} F'\left(z_{jk}\right)\right) \approx \tilde{t}_{jk} = \frac{1}{L}\sum_{\ell=1}^{L}\left(\boldsymbol{v}_\ell^{\mathrm{T}} F(z_{jk})^{-1} F'(z_{jk}) \boldsymbol{v}_\ell\right) \tag{6.1}$$

ここで，$z_{jk},\ j = 0, 1, 2, 3,\ k = 1, 2, \ldots$ は k 番目の閉曲線の j 番目の積分点を表す．ワーカから計算結果を受け取ったマスターは 4 点分の計算が終わった領域内部の固有値数を推定する．

$$\tilde{m}_{\mathrm{R}}^{(k)} = \frac{1}{N} \sum_{j=0}^{3} w_j \tilde{t}_{jk} \tag{6.2}$$

この後，マスターはタスクセットが空でないならばタスクの処理をワーカに命じる．このタスクはランダムに選択する．マスターが全ての領域について推定固有値数を求めたならば，最後に全ワーカに終了命令を出して終了する．

Algorithm 8 および Algorithm 9 に完全型メッシュアルゴリズムのマスターとワーカのアルゴリズムを示す．Algorithm 8 について，$\tilde{m}_{\mathrm{R}}^{(k)}$ は，k 番目の閉曲線における推定値である．Algorithm 9 について，$\boldsymbol{x}_{jk}^{(\ell)}$ は，線形方程式 $F(z_{jk})\boldsymbol{x}_{jk}^{(\ell)} = F'(z_{jk})\boldsymbol{v}_\ell$ の近似解である．

Algorithm 8 完全型メッシュアルゴリズム（マスター）

Input: $F(z), (a_0, b_0), (a_1, b_1), N_{\mathrm{all}}$
Output: $\tilde{m}_{\mathrm{R}}^{(k)}, \quad k = 1, 2, \ldots$

Compute integral points $z_p,\ p = 1, 2, \ldots, N_{\mathrm{all}}$
Add N_{all} tasks to TaskSet
Send tasks to all workers
while there exist $\tilde{m}_{\mathrm{R}}^{(k)}$ which have not been computed **do**
    Receive $\tilde{t}_{jk}$ from a worker
    if Converge all integral points in Γ_k **then**
        Compute $\tilde{m}_{\mathrm{R}}^{(k)}$ by Eq.(6.2)
    end if
    if TaskSet is not empty **then**
        Send a task to free worker
    end if
end while
Send END to all workers

図 6.4 に完全型メッシュの例を示す．格子点が積分点となる．

6.3 適応型メッシュアルゴリズム

前節で述べた完全型メッシュアルゴリズムは，固有値の分布に関わらず一様に格子を配置し固有値数推定を行う手法である．しかし，有理式型推定法を固有値解法のパラメータ設定に用いる場合，固有値分布推定法はできるだけ早く計算結果を出すのが望ましい．そのため

Algorithm 9 完全型メッシュアルゴリズム（ワーカ）

1: **Receive** task from master
2: **while** have not received END **do**
3: **Solve** $F(z_{jk})\boldsymbol{x}_{jk}^{(\ell)} = F'(z_{jk})\boldsymbol{v}_\ell$ for $\boldsymbol{x}_{jk}^{(\ell)}, \quad \ell = 1, 2, \ldots, L$
4: **Compute** $\tilde{t}_{jk}$ by Eq.(6.1)
5: **Send** $\tilde{t}_{jk}$ to master
6: **Receive** task or END from master
7: **end while**

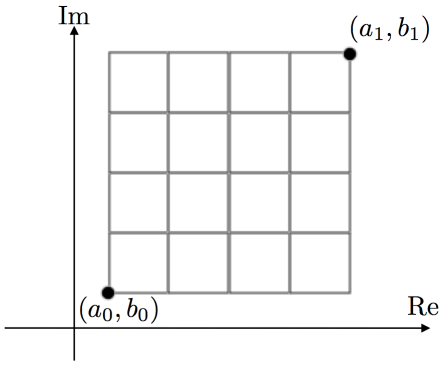

図 6.4: 完全型メッシュ例（N_{all}=25）

固有値が存在しない領域を粗く推定し，固有値が存在している領域を細かく推定することで計算量を削減することができる．適応型メッシュアルゴリズムは階層的に有理式型推定法を計算し，固有値の粗密に応じた格子での固有値分布推定を行うアルゴリズムである．

適応型メッシュアルゴリズムのワーカのアルゴリズムは完全型メッシュアルゴリズムと変わらないが，マスターのアルゴリズムは異なる．マスターは，はじめに完全型メッシュの積分点のうち N_{part} 個の初期格子点を選び，粗い格子による初期領域を決定し，タスクセットにタスクを追加する．次にマスターはタスクセットの中からワーカ数 N_{w} に応じて，タスクの処理を各ワーカに命じる．各ワーカは受け取ったタスクの情報に基づいて線形方程式を反復解法で解き，収束判定を満たしたとき，$\tilde{t}_{jk}$ をマスターに送る．ワーカから計算結果を受け取ったマスターは 4 点分の計算が終わった領域内部の固有値数を推定する．そして推定値がユーザが与えた閾値 m' よりも大きい場合，その格子を分割し新たに計算を必要とする積分点をタスクセットに追加する．この後，マスターはタスクセットが空でないならばタスクの処理をワーカに命じる．このアルゴリズムによって生成される格子を適応型メッシュとよぶ．適応型メッシュアルゴリズムにより，総積分点数の削減が可能となる．

Algorithm 10 および Algorithm 11 に適応型メッシュアルゴリズムを示す．

Algorithm 10 適応型メッシュアルゴリズム（マスター）

Input: $F(z), (a_0, b_0), (a_1, b_1), N_{\mathrm{all}}, N_{\mathrm{part}}, m'$
Output: $\tilde{m}_{\mathrm{R}}^{(k)}, \quad k = 1, 2, \ldots$

1: **Compute** integral points $z_p,\ p = 1, \ldots, N_{\mathrm{all}}$
2: **Add** N_{part} initial tasks to TaskSet
3: **Send** tasks for all workers
4: **while** there exist $\tilde{m}_{\mathrm{R}}^{(k)}$ which have not been computed **do**
5: **Receive** $\tilde{t}_{jk}$ from a worker
6: **if** Converge all integral points in Γ_k **then**
7: **Compute** $\tilde{m}_{\mathrm{R}}^{(k)}$ by Eq.(6.2)
8: **end if**
9: **if** $\tilde{m}_{\mathrm{R}}^{(k)} > m'$ **then**
10: **Add** new tasks to TaskSet
11: **end if**
12: **if** TaskSet is not empty **then**
13: **Send** tasks to free workers
14: **end if**
15: **end while**
16: **Send** END

Algorithm 11 適応型メッシュアルゴリズム（ワーカ）

1: **Receive** task from master
2: **while** have not received END **do**
3: **Solve** $F(z_{jk})\boldsymbol{x}_{jk}^{(\ell)} = F'(z_{jk})\boldsymbol{v}_\ell$ for $\boldsymbol{x}_{jk}^{(\ell)}, \quad \ell = 1, 2, \ldots, L$
4: **Compute** $\tilde{t}_{jk}$ by Eq.(6.1)
5: **Send** $\tilde{t}_{jk}$ to master
6: **Receive** task or END from master
7: **end while**

適応型メッシュの例を図 6.5 に示す．完全型メッシュアルゴリズムとの違いは以下の 2 点である．まず，アルゴリズムの 1 行目で初期タスクの決定を行っている点である．適応型メッシュの初期点は指定領域全体をカバーすることが望ましいため，ワーカプロセス数に応じた初期格子点を求める必要がある．次に，領域分割とタスクの追加である．適応型メッシュでは 4 点が収束した領域から有理式型推定法を行い，m' と比較して分割の必要性を判断する．分割が必要な場合は，新しく計算が必要となる積分点のうち，タスクセットに入っていないものを追加する．このとき，分割された領域の最小サイズを完全型メッシュの領域サイズと同じにすることで，適応型アルゴリズムの最大積分点数は完全型メッシュアルゴリズムの積分点数と等しくなる．そのため，適応型メッシュは完全型メッシュと比べて計算する積分点数を減らすことができ，完全型メッシュアルゴリズムよりも計算量を削減する事ができる．しかし，適応型メッシュアルゴリズムは最初粗い格子で有理式型推定法を行い，固有値数が推定された後でタスクを追加していくため，負荷分散がとりづらくなる．また，分割の判断は格子の 4 点目の計算終了後に判断するため，4 点目の計算が遅いと分割の判断ができない．これにより，タスクセットが空になりやすく，計算待ち状態のワーカが発生しやすいという問題がある．

6.4 先読み付き適応型メッシュアルゴリズム

前節で述べたように，適応型メッシュアルゴリズムは，タスクセットが空になりやすく，計算待ち状態のワーカが発生しやすいという問題がある．その問題を軽減させる手法として先読み付き適応型メッシュアルゴリズムを提案する．

先読み付き適応型メッシュアルゴリズムは，有理式型推定法に必要な 4 点目の計算が終わるよりも前に分割の判断をし，分割の必要性があるとわかれば，新しく計算が必要な積分点をより早くタスクセットに追加する方法である．これにより待ち状態のワーカにより早くタスクを与えることができ，負荷バランスが向上する可能性がある．

4 点目の計算が終わるよりも前に分割の判断をするために，積分点 3 点が収束し収束していない点の線形方程式解法の反復回数が上限に達したならば，数値積分に必要な重み w_j を収束した 3 点で計算するよう再定義して，有理式型推定法を行う．このとき，積分点が円周上に等間隔に並んでいない場合の w_j の計算は N 次線形方程式

$$\sum_{j=0}^{N-1} w_j \zeta_j^{k-1} = \begin{cases} 1, & k = 0 \\ 0, & k = 1, \ldots, N-1 \end{cases}, \tag{6.3}$$

を解くことで求められる [44, 45, 46].

式 (6.3) で得られた w_j を用いて式 (6.2) の計算を行うことで，先読みした推定固有値数が求められる．この先読みを行うために，各ワーカにおける線形方程式解法の反復回数上限の設定で反復を途中で停止させ，マスターから命令が来たときに計算を再開するようにする．

Algorithm 12，Algorithm 13 に先読み付き適応型メッシュアルゴリズムを示す．適応型メッシュアルゴリズムとの変更，追加は以下の通りである．マスターに関しては，6-9 および 14 行目に先読みに関する追加を行った．また，15-19 行目ではワーカの反復の停止と再開を実現するためにマスターからの命令，ワーカからの結果の受け取りに変更を行った．ワーカのアルゴリズムは 5-12 行目で反復の停止と再開を実現するために，マスターとのやり取りに関して変更を行った．

3 点積分による先読みで得られる推定固有値数は，一般に 4 点積分の場合と異なる．そのため，先読みの判断の誤りには 2 種類のタイプがある．1 つ目は 3 点では分割の必要がないと判断したが，4 点で求めた場合では分割が必要と判断した場合である．これはタスクセットへの追加のタイミングが先読みなしと同じになるため，先読みのメリットは得られないが，デメリットもない．2 つ目は 3 点では分割が必要と判断したが，4 点で求めた場合では分割が不必要だった場合である．この場合は，先読み無しの場合には計算の必要がなかった積分点をタスクセットに追加してしまうため，計算量を増やしてしまうというデメリットがある．

先読みの目的は，タスクセットの中身が 0 のために次のタスクが割り振られず，待ち状態となったワーカを出さないことにある．そのため，ワーカに割り振るタスクがあるときはリスクを冒してまで先読みをする必要はない．この先読みのリスクを軽減するために，先読みはタスクセットが 0 のときのみ行うようにした．

Algorithm 12 先読み付き適応型メッシュアルゴリズム（マスター）

Input: $F(z), (a_0, b_0), (a_1, b_1), N_{\mathrm{all}}, N_{\mathrm{part}}, m'$
Output: $\tilde{m}_{\mathrm{R}}^{(k)}, \quad k = 1, 2, \ldots$

Compute integral points $z_p,\ p = 1, \ldots, N_{\mathrm{all}}$
Add N_{part} initial tasks to TaskSet
Send tasks to all workers
while there exist $\tilde{m}_{\mathrm{R}}^{(k)}$ which have not been computed **do**
  Receive $result$
  if Converge integral points in $\Gamma_k \geq 3$ **then**
    if 1 point NotConverge and TaskSet is empty **then**
      Compute w_j by Eq.(6.3)
    end if
    Compute $\tilde{m}_{\mathrm{R}}^{(k)}$ by Eq.(6.2)
    if $\tilde{m}_{\mathrm{R}}^{(k)} > m'$ **then**
      Add new tasks to TaskSet
    end if
  end if
  if $result$ is NotConverge **then**
    Send CONTINUE
  else if TaskSet is not empty **then**
    Send tasks to free workers
  end if
end while
Send END

Algorithm 13 先読み付き適応型メッシュアルゴリズム（ワーカ）

Receive task from master
while have not received END **do**
  Solve $F(z_{jk})\boldsymbol{x}_{jk}^{(\ell)} = F'(z_{jk})\boldsymbol{v}_\ell$ for $\boldsymbol{x}_{jk}^{(\ell)}, \quad \ell = 1, 2, \ldots, L$
  if equation solver converge **then**
    Compute $\tilde{t}_{jk}$ by Eq.(6.1)
    $result \leftarrow \tilde{t}_{jk}$
  else
    $result \leftarrow$ NotConverge
  end if
  Send $result$ to master
  Receive task, CONTINUE or END
end while

6.5 数値実験

本章では，有理式型推定法の並列化実装アルゴリズムの有効性を検証する．数値実験は，CPU: AMD Opteron(tm) Processor 6180 SE(2.5GHz) 12-Core × 4，Memory: DDR3 SDRAM 8GB × 32 (Total 256 GB)，OS: Cent OS 5.4 上で行い，C 言語と MPI，ソフトウェア・ライブラリ PETSc[47, 48] を用いて実装した．対象とする行列は表 6.1 に示す．QDSub2 は QD の 2 次の項までを用いた問題であり，QDSub4 は 4 次の項までを用いた問題である．ベクトル $\boldsymbol{v}_\ell$ の本数は $L = 32$ とする．線形方程式の反復解法にはリスタート付き GMRES 法 [49] を用い，収束条件は相対残差が 10^{-3} を下回ったときとし，リスタート数を 30 とした．線形方程式の前処理には ILU(0) 前処理 [50] を用いる．

表 6.1: 数値実験で対象とする行列

	$F(\lambda)$	Size	(a_0, b_0)	(a_1, b_1)
QDSub2	$\sum_{i=0}^{2} \lambda^i A_i$	245	$(-0.56, -0.19)$	$(0.04, 0.21)$
QDSub4	$\sum_{i=0}^{4} \lambda^i A_i$	2475	$(-1.1, -0.7)$	$(0.9, 1.3)$
Butterfly	$\sum_{i=0}^{4} \lambda^i A_i$	64	$(-2, -2)$	$(2, 2)$

6.5.1 数値実験 6-1

本項では適応型メッシュアルゴリズムの有効性を調べる．対象とする行列は，QDSub2 と Butterfly を用いる．本項では，タスク数 N_{all} を変化させて，完全型メッシュアルゴリズム，適応型メッシュアルゴリズムを行い，それぞれで生成された，完全型メッシュ，適応型メッシュの総積分点数を調べた．今回の実験では，固有値が存在しないと思われる領域では分割を行わないようにするために，閾値 $m' = 0.5$ とした．適応型メッシュアルゴリズムの初期タスク N_{part} は，QDSub2 では指定領域を実軸 3 分割，虚軸 2 分割したときの積分点 12 点を与え，Butterfly では指定領域の端点 4 つを与える．

実験結果を図 6.6，図 6.7，表 6.3 に示す．図 6.6，図 6.7 は，QDSub2，Butterfly をそれぞれ適応型メッシュアルゴリズムで解いたときのメッシュ構造と固有値の分布である．横軸は実軸，縦軸は虚軸を示しており，黒い点が固有値，四角の枠が適応型メッシュアルゴリズムで生成される格子を示している．固有値は MATLAB の関数 `polyeig` で求めた値である．

図 6.6，図 6.7 から，固有値の存在する周辺の領域ほど分割が行われており，固有値密度に応じた格子を生成していることがわかる．表 6.3 は完全型メッシュと適応型メッシュの総積分点数である．表 6.3 から，タスク数 N_{all} を変化させた場合，完全型メッシュアルゴリズムで

は，N_{all} 個の積分点での計算を行っているが，適応型メッシュアルゴリズムは，N_{all} 個の積分点での計算を行なっていないことがわかる．これによりタスク数 N_{all} を広く変化させても，適応型メッシュでは完全型メッシュより，積分点数を増やさずに固有値が密に集まっている部分での詳細な分布情報を得ることができていることがわかる．

有理式型推定法を固有値解法のパラメータ設定に用いる際，固有値が少ない領域での詳細な分布情報の必要性は少ないため，適応型メッシュアルゴリズムでも十分な情報が得られている．また，有理式型推定法の計算はなるべく早く計算させるのが好ましく，少ない計算量で固有値分布が得られる適応型メッシュアルゴリズムが有効であることがわかる．

表 6.2: 数値実験 6-1 の各メッシュの積分点数（QDSub2）

N_{all}	完全型メッシュ	適応型メッシュ	適応型/完全型
35	35	35	1.00
117	117	79	0.67
425	425	167	0.39
1617	1617	315	0.19
6305	6305	560	0.09
24897	24897	859	0.03

表 6.3: 数値実験 6-1 の各メッシュの積分点数（Butterfly）

N_{all}	完全型メッシュ	適応型メッシュ	適応型/完全型
25	25	25	1.00
81	81	81	1.00
289	289	217	0.75
1089	1089	505	0.46
4225	4225	1081	0.26
16641	16641	2443	0.17

6.5.2 数値実験 6-2

本項では，先読み付き適応型メッシュアルゴリズムの有効性を調べる．対象とする行列は，QDSub4 を用いる．本項では，適応型メッシュアルゴリズムと先読み付き適応型メッシュアルゴリズムを行い，先読みの正誤と各ワーカが仕事をしている時間を調べる．

本実験では，閾値 $m' = 20$ をとし，適応型メッシュアルゴリズムの初期タスク N_{part} は，指定領域を実軸，虚軸ともに 4 分割し，25 点の積分点を与える．ワーカプロセス数 N_{w} は 48，

マスタープロセス数は 1 とする．

実験結果は図 6.8，図 6.9，表 6.4 に示す．図 6.8 は，適応型メッシュアルゴリズムの各ワーカプロセスの実行状況である．図 6.9 は，先読み付き適応型メッシュアルゴリズムの各ワーカプロセスの実行状況である．横軸はマスタープロセスの経過時間，縦軸が各ワーカプロセスの番号を示している．ワーカが線形方程式を解いている時間を実行時間（busy time）と呼び，各図で赤い線で示した．また，線形方程式を解いていない時間を待ち時間（wait time）と呼び，各図の線のない部分にあたる．グラフの結果から，先読み付き適応型メッシュアルゴリズムは，初期タスクが割り当てられなかったワーカ（26 番から 48 番）に対して適応型メッシュアルゴリズムよりも早くタスクを渡し計算の始まりを早くすることができている．これは，先読みによってタスクセットにタスクが早く追加されたことによるものである．また，適応型メッシュアルゴリズムでは 3 秒前後においてタスクリストが空になり，待ち状態のワーカが多数現れたが，先読み付き適応型メッシュアルゴリズムでは同じ時間において十分にワーカが働いている．このことから，先読みが約 4 秒までの結果において有効であったことがわかる．本実験では，最後にいくつかの線形方程式の反復数が非常に多くなり，実行時間が長くなった．これは領域を細かくした際に積分点が固有値に近い値をとってしまったため，反復数が非常に増えたと考えられる．このような状況に対する改善案として，線形方程式の最大反復回数を分割前の 1 つ粗い領域における各積分点の反復回数から定めるといった方法が考えられる．

表 6.4 は先読みの正誤表である．先読みは全部で 106 回行われた.「先読みで分割し実際は分割する」と「先読みで分割せず実際は分割しない」場合は先読みが正しく行われており，合計で 90 回あった．そのため，正答率は約 84.9%となった．先読みが失敗した内訳は「先読みで分割せず実際は分割する」という場合は 2 回発生した．これは先読み無しと同じであるため，問題はない．しかし,「先読みで分割し実際は分割しない」という判断が 14 回行われてしまった．この場合は本来必要のない計算が発生するため，計算量が増加してしまう．失敗の原因としては 3 点積分による先読みにおける推定値の精度悪化が考えられる．この改善案として，対象とする領域の分割前の一階層粗い領域の推定数と，そこから分割され計算の終わった別の領域における推定数を用いて対象とする領域の推定数の精度を改善していくことが挙げられる．

表 6.4: 先読みの正誤

	先読みで分割する	先読みは分割しない
実際に分割する	42	2
実際に分割しない	14	48

6.6 小括

本章ではマスター・ワーカ方式並列計算について述べ，有理式型推定法の計算過程で発生する複数の線形方程式に対して，マスター・ワーカ方式を用いた並列実装アルゴリズム（完全型メッシュアルゴリズム）を提案した．提案アルゴリズムによって，ロードバランスの悪化の軽減が可能であることを示した．また，固有値の粗密状況によって閉曲線の大きさを変更する並列実装アルゴリズム（適応型メッシュアルゴリズム）を提案した．これにより，総積分点数が減少し，計算する線形方程式の数を減らせることを示した．さらに，以上のアルゴリズムに固有値数の推定値の先読み計算を付加した並列実装アルゴリズム（先読み付き適応型メッシュアルゴリズム）を提案した．これにより，タスクを早く追加することができ，タスクが計算途中で空になることを防げることを示した．数値実験から，適応型メッシュアルゴリズムによって作成された総積分点数は完全型メッシュアルゴリズムの総積分点数よりも少ない数になることがわかった．これにより，線形方程式の求解回数を減らすことが可能となった．また，適応型メッシュアルゴリズムの並列実装では計算途中でタスクが空になり，タスク待ち状態となるワーカが発生していたが，先読み付き適応型メッシュアルゴリズムによって，タスク待ち状態を軽減できることがわかった．以上の結果により，提案した 3 つの並列実装アルゴリズムの有効性を示した．

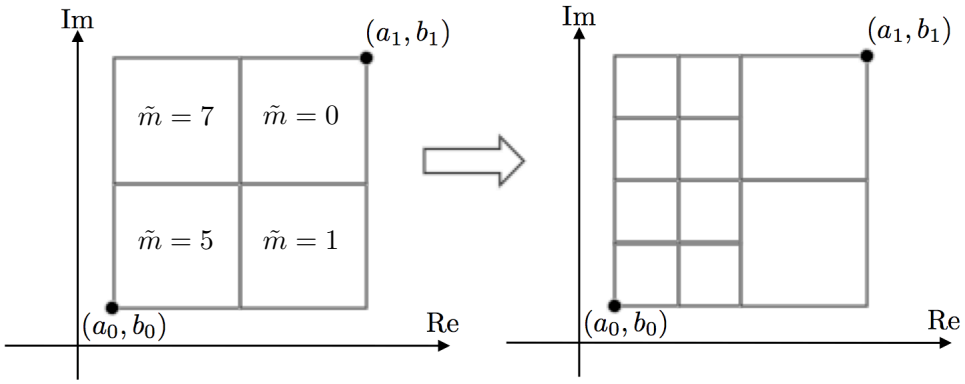

図 6.5: 適応型メッシュ例（N_{part}=9，m'=2）

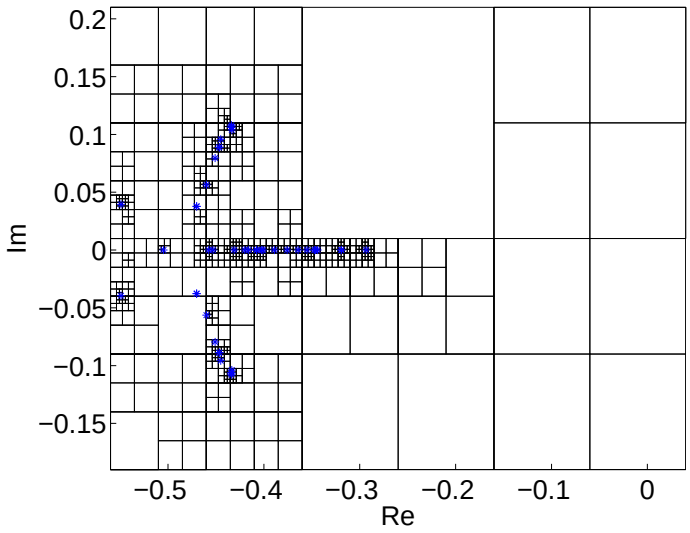

図 6.6: 数値実験 6-1 の実験結果（QDSub2）

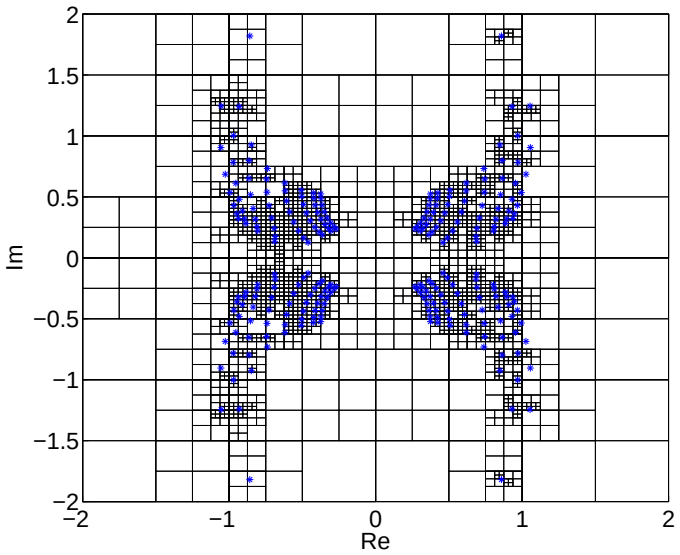

図 6.7: 数値実験 6-1 の実験結果（Butterfly）

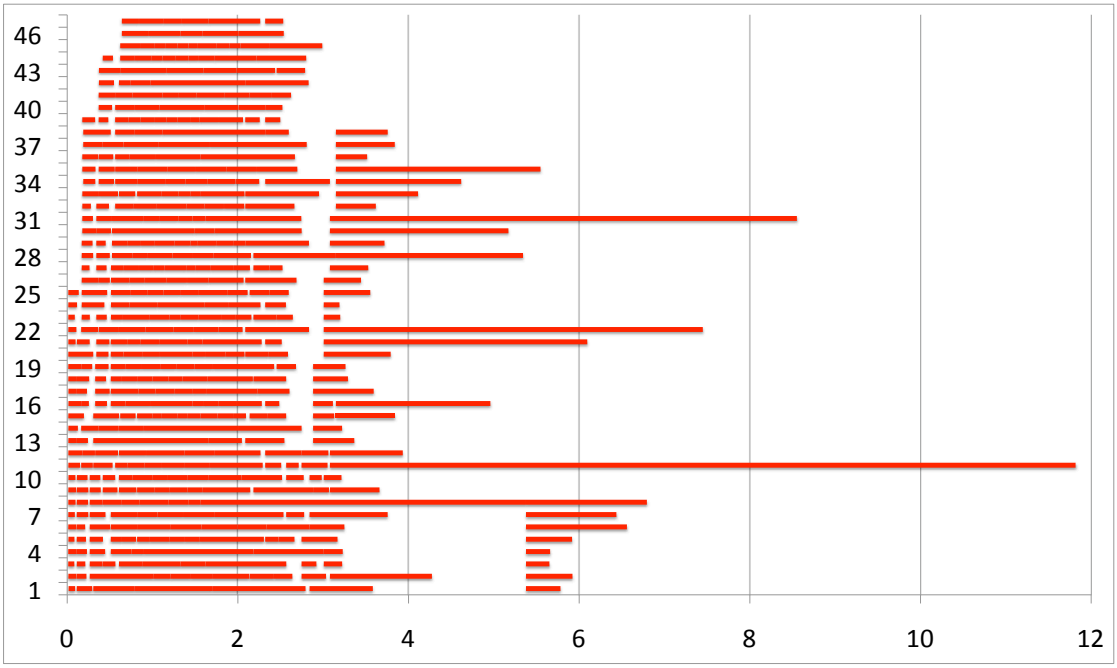

図 6.8: 適応型メッシュアルゴリズムの各ワーカプロセスの実行状況

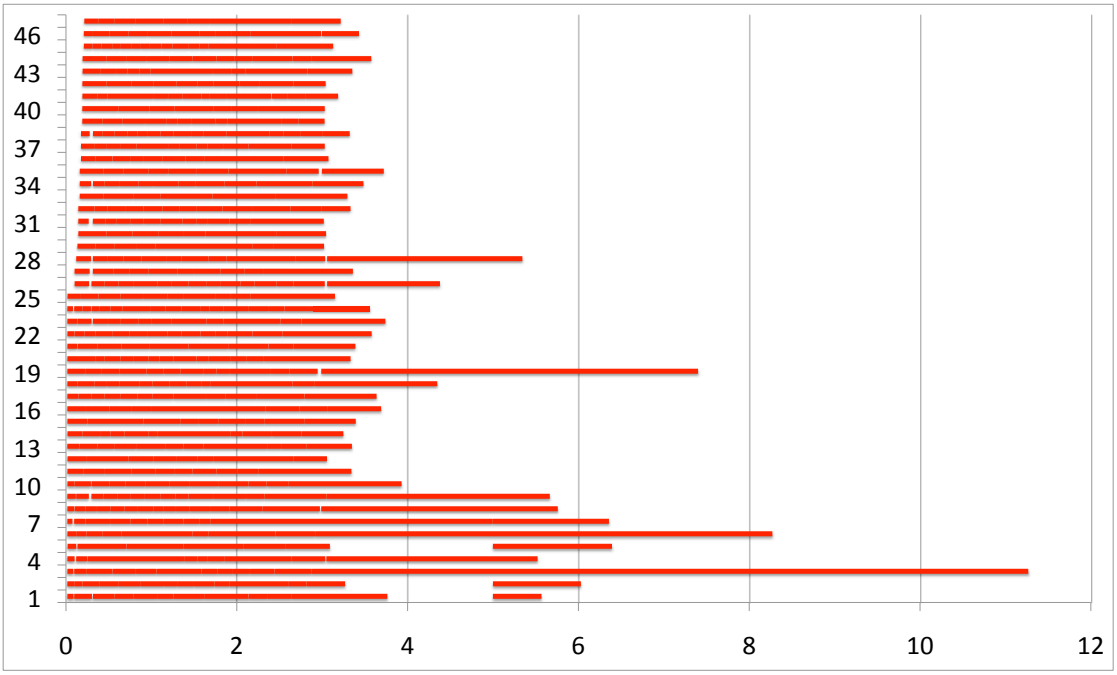

図 6.9: 先読み付き適応型メッシュアルゴリズムの各ワーカプロセスの実行状況

第7章　結論

本論文では，確率的固有値分布推定法の一つである有理式型推定法に対して，Krylov 部分空間法を用いた際のフィルタ関数の解析および，確率的固有値分布推定法の一つである多項式型推定法との比較を行った．また有理式型推定法に対して，非線形固有値問題への拡張手法の提案，円弧領域における拡張手法の提案，マスター・ワーカ方式による並列実装アルゴリズムの提案を行った．

第 2 章では，確率的固有値分布推定法である有理式型推定法および多項式型推定法について説明した．さらに，有理式型推定法および多項式型推定法によって作成されるフィルタ関数について説明した．

第 3 章では，線形方程式を求解する反復解法の一つである Krylov 部分空間反復法について説明し，Krylov 部分空間反復法の一つである BiCG 法および COCG 法について述べた．また，複数のシフト線形方程式を同時に解く Shifted Krylov 部分空間反復法について説明し，Shifted Krylov 部分空間反復法の一つである Shifted COCG 法およびその多項式表現について述べた．さらに，有理式型推定法の計算過程で発生する線形方程式に対して，Shifted Krylov 部分空間反復法を用いることで，有理式型推定法が多項式で表現可能であることを示し，有理式型推定法の行列・ベクトル積の計算回数が，多項式型推定法と同様に多項式の次数によって決まることを示した．そして，数値実験によって，有理式型推定法のフィルタ関数は，多項式の次数と行列に依存することを示した．また，固有値数を推定する区間内部の固有値数が少ない場合，有理式型推定法は，多項式型推定法よりも，区間外部のフィルタ関数の減衰が強く，少ない多項式の次数で推定値を求めることができることを示した．以上により，有理式型推定法は固有値が存在しない区間を求めるのに有効であることがわかる．

第 4 章では，有理式型推定法を非線形固有値問題に拡張した方法を提案し，拡張した Smith の標準形を用いて，拡張法によって固有値数を求めることができることを数式によって示した．数値実験から，有理式型推定法において，積分点数は 4，点もしくは 8 点程度で固有値数を推定することができることを示した．また，行列のトレースの確率的推定によって近似される値は，ランダムベクトル $\boldsymbol{v}_\ell$ の本数が $L = 30$ 以上であれば，どのようなランダム値であっても，互いに近い値となることを示した．さらに，有理式型推定法において，ランダム

ベクトル $\boldsymbol{v}_\ell$ の本数が $L = 30$ 程度であれば固有値数が推定できることを示した．以上により，非線形固有値問題に対する有理式型推定法の拡張法を用いることで，これまで不可能であった非線形固有値問題に対する固有値数の推定が可能となる．

第 5 章では．実数区間に対するフィルタ関数について述べ，実数区間に対するフィルタ関数を有理式型推定法に応用できることを示した．また，実数区間に対するフィルタ関数を応用し，円弧近傍に対する有理式型推定法の拡張手法を示した．数値実験から，拡張手法のフィルタ関数は円弧外部で 0 に減衰することを示し，積分点数を増やすことによって，円弧外部ではより 0 に減衰するフィルタ関数が作成されることを示した．また，拡張法によって円弧近傍の固有値数が推定できることを示した．以上により，円弧近傍に対する有理式型推定法の拡張手法は，円弧近傍の固有値数を効果的に求めることができる．

第 6 章では，並列実装方式の一つであるマスター・ワーカ方式並列計算について述べた．また，有理式型推定法の計算過程で発生する複数の線形方程式に対して，マスター・ワーカ方式を用いた並列実装アルゴリズム（完全型メッシュアルゴリズム）を提案し，ロードバランスの悪化の軽減が可能であることを示した．また，固有値の粗密状況によって閉曲線の大きさを変更する並列実装アルゴリズム（適応型メッシュアルゴリズム）を提案した．これにより，総積分点数が減少し，計算する線形方程式の数を減らせることを示した．適応型メッシュアルゴリズムに固有値数の推定値の先読み計算を付加した並列実装アルゴリズム（先読み付き適応型メッシュアルゴリズム）を提案した．適応型メッシュアルゴリズムの並列実装では計算途中でタスクが空になり，タスク待ち状態となるワーカが発生していたが，先読み付き適応型メッシュアルゴリズムによってタスクを早く追加することができ，タスクが計算途中で空になることを防げることを示した．以上により，有理式型推定法に対する 3 つの並列実装アルゴリズムの有効性を示した．

謝辞

本論文を作成するにあたり，常日頃から終始丁寧にご指導いただきました筑波大学 システム情報系 櫻井 鉄也 教授に心より感謝いたします．筆者が情報数理研究所に配属してからの 6 年間，数多くのご助言を頂きました．筆者が学部 4 年生の頃に情報学群長賞をいただき，修士 1 年生の頃にコンピュータサイエンス専攻長賞をいただくことができたのも櫻井 鉄也 教授が親身に相談にのって下さったおかげです．重ねてお礼申し上げます．

大変お忙しい所，本論文の審査をお引き受けいただいた筑波大学 システム情報系 高橋 大介 教授，河辺 徹 教授，佐野 良夫 准教授，筑波大学 数理物質系 重田 育照 教授に心より感謝いたしますとともに御礼申し上げます．

筑波大学 システム情報系 北川 高嗣 教授から，研究を行う上で，快適な研究環境を提供して頂きました．ここに感謝の意を表します．D. Sistemes Informàtics i Computació, Universitat Politècnica de València, Jose E. Roman 准教授には本研究を進めるにあたり，多くのご助言を賜りました．また，筆者が実装したプログラムがソフトウェア・ライブラリ SLEPc に組み込まれましたのも，Jose E. Roman 准教授のご協力によるものです．心から感謝いたします．Network for Computational Nanotechnology Purdue University, Gerhard Klimeck 教授，Michael Povolotskyi 特任助教には本研究に必要な行列の提供およびご助言を賜りまして心より感謝いたします．

常日頃から研究に関して熱心にご指導をいただいた筑波大学 システム情報系 多田野 寛人 助教，今倉 暁 助教，Claus Aranha 助教，二村 保徳 助教，保國 惠一 助教に心より感謝申しあげます．同じ研究室にこれだけ多くの先生方がいてくださったおかげで非常に心強く感じました．ここに御礼申し上げます．特に二村 保徳 助教は筆者が研究室に配属してから 6 年間，数値解析の基礎から応用まで様々なことを教えていただき，非常に多くのことを支えていただきました．心より感謝いたします．

6 年間とても快適で楽しい研究生活を送らせていただきましたことを情報数理研究室の皆様に深く感謝いたします．

最後に筆者の 9 年間に渡る大学生活を生活面で支えてくださった両親に心から感謝いたします．

関連図書

[1] Feng-Nan Hwang, Zih-Hao Wei, Tsung-Ming Huang, and Weichung Wang. A parallel additive schwarz preconditioned jacobi-davidson algorithm for polynomial eigenvalue problems in quantum dot simulation. *J. Comput. Phy.*, 229(8):2932–2947, 2010.

[2] Elias Jarlebring, Karl Meerbergen, and Wim Michiels. An arnoldi method with structured starting vectors for the delay eigenvalue problem. In *9th IFAC Workshop on Time Delay Systems*, pages 57–62, Czech Republic, 2010. International Federation of Automatic Control.

[3] Mathieu Luisier, Andreas Schenk, Wolfgang Fichtner, and Gerhard Klimeck. Atomistic simulation of nanowires in the sp3d5s* tight-binding formalism: From boundary conditions to strain calculations. *Phys. Rev. B*, 74, 2006.

[4] Cleve B. Moler and Gilbert W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973.

[5] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

[6] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. SIAM, 2nd edition, 2011.

[7] Danny C. Sorensen. Implicit application of polynomial filters in a k-step arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[8] Tetsuya Sakurai and Hiroshi Sugiura. A projection method for generalized eigenvalue problems using numerical integration. *J. Comput. Appl. Math.*, 159:119–128, 2003.

[9] Tsutomu Ikegami, Tetsuya Sakurai, and Umpei Nagashima. A filter diagonalization for generalized eigenvalue problems based on the sakurai-sugiura projection method. *J. Comput, Appl. Math.*, 233:1927–1936, 2010.

[10] Tsutomu Ikegami and Tetsuya Sakurai. Contour integral eigensolver for non-hermitian systems: A rayleigh-ritz-type approach. *Taiwanese J. Math.*, 14:825–836, 2010.

[11] Eric Polizzi. A density matrix-based algorithm for solving eigenvalue problems. *Phys. Rev. B*, 79, 2009.

[12] Feast eigenvalue solver. `http://www.ecs.umass.edu/~polizzi/feast/`.

[13] Eric Polizzi. High-performance numerical library for solving eigenvalue problems. *arXiv:1203.4031v3*, 2015.

[14] Grady Schofield, James R. Chelikowsky, and Yousef Saad. A spectrum slicing method for the kohnsham problem. *Comput. Phys. Comm.*, 183:497–505, 2012.

[15] Ruey-Lin Chern, C. Chung Chang, Chien C. Chang, and Robert R. Hwang. Numerical study of three-dimensional photonic crystals with large band gaps. *J. Phys. Soc. Jpn.*, 73:727–737, 2004.

[16] Tsung-Ming Huang, Han-En Hsieh, Wen-Wei Lin, and Weichung Wang. Eigenvalue solvers for three dimensional photonic crystals with face-centered cubic lattice. *J. Comput. Appl. Math.*, 272:350–361, 2014.

[17] Semyon Aronovich Geršhgorin. Über die abgrenzung der eigenwerte einer matrix. *Izv. Akad. Nauk SSSR Ser. Mat. 1*, 7:749–755, 1931.

[18] Yuji Nakatsukasa. Gerschgorin ’s theorem for generalized eigenvalue problems in the euclidean metric. *Math. Comput.*, 80(276):2127–2142, 2011.

[19] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[20] Kenta Senzaki, Hiroto Tadano, Tetsuya Sakurai, and Zhaojun Bai. A method for profiling the distribution of eigenvalues using the as method. *Taiwanese Journal of Mathematics*, 14(3A):839–853, 2010.

[21] Yasunori Futamura, Hiroto Tadano, and Tetsuya Sakurai. Parallel stochastic estimation method for eigenvalue distribution. *JSIAM Letters*, 2:127–130, 2010.

[22] Eric Polizzi Edoardo Di Napoli and Yousef Saad. Effcient estimation of eigenvalue counts in an interval. *arXiv:cs/1308.4275v2*, 2014.

[23] Zhaojun Bai, Mark Fahey, and Gene H. Golub. Some large scale matrix computation problems. *J. Comput. Appl. Math.*, 74:71–89, 1996.

[24] Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Commun. Stat., Simulation Comput.*, 19:433–450, 1990.

[25] Laurent O. Jay, Hanchul Kim, Yousef Saad, and James R. Chelikowsky. Electronic structure calculations using plane wave codes without diagonlization. *Comput. Phys. Comm.*, 118:21–30, 1999.

[26] 藤野 清次 and 張 紹良. **反復法の数理**. 朝倉書店, 1996.

[27] Roger Fletcher. Conjugate gradient methods for indefinite systems. *Lecture Notes in Math.*, 506:73–89, 1976.

[28] Henk Albertus van der Vorst. A petrov-galerkin type method for solving ax=b, where a is symmetric complex. *IEEE Trans. on Magn.*, 26:706–708, 1990.

[29] Ryu Takayama, Takeo Hoshi, Tomohiro Sogabe, Shao-Liang Zhang, and Takeo Fujiwara. Linear algebraic calculation of the green 's function for large-scale electronic structure theory. *Phys. Rev. B*, 73, 2006.

[30] Beat Jegerlehner. Krylov space solvers for shifted linear systems. *arXiv:hep-lat/9612014*, 1996.

[31] Takeo Fujiwara Ryu Takayama, Takeo Hoshi. Krylov subspace method for molecular dynamics simulation based on large-scale electronic structure theory. *J. Phys. Soc. Jpn.*, 73(6):1519–1524, 2004.

[32] Takeo Hoshi. Elses matrix library. `http://www.elses.jp/matrix/`.

[33] Ben-Shan Liao. *Subspace projection methods for model order reduction and nonlinear eigenvalue computation*. PhD thesis, University of California, Davis, 2007.

[34] Israel Gohberg, Peter Lancaster, and Leiba Rodman. *Matrix Polynomials*. SIAM, 1982.

[35] Israel Gohberg and Leiba Rodman. Analytic matrix functions with prescribed local data. *Journal d'Analyse Mathématique*, 40:90–128, 1981.

[36] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley, 1999.

[37] Timo Betcke, Nicholas J. Higham, Volker Mehrmann, Christian Schröder, and Françoise Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. `http://www.mims.manchester.ac.uk/research/numerical-analysis/nlevp.html`.

[38] Shinnosuke Yokota and Tetsuya Sakurai. A projection method for nonlinear eigenvalue problems using contour integrals. *JSIAM Letters*, 5:41–44, 2013.

[39] Junko Asakura, Tetsuya Sakurai, Hiroto Tadano, Tsutomu Ikegami, and Kinji Kimura. A numerical method for polynomial eigenvalue problems using contour integral. *Japan J. Indust. Appl. Math.*, 27:73–90, 2010.

[40] Anthony P. Austin and Lloyd N. Trefethen. Computing eigenvalues of real symmetric matrices with rational filters in real arithmetic. *Preprint,* `http://people.maths.ox.ac.uk/austin/`, 2014.

[41] 村上 弘. レゾルベントの線形結合によるフィルタ対角化法. **情報処理学会論文誌　コンピューティングシステム**, 49(SIG 2(ACS 21)):66–87, 2008.

[42] 村上 弘. 帯対称定値一般固有値問題のフィルタ対角化法の実験. **情報処理学会研究報告ハイパフォーマンスコンピューティング（*HPC*）**, 59((2007-HPC-110)):31–36, 2007.

[43] NIST. Matrix market. `http://math.nist.gov/MatrixMarket/`.

[44] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric lagrange interpolation. *SIAM Rev.*, 46:501–517, 2004.

[45] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, Philadelphia, 2003.

[46] Tetsuya Sakurai, Yasunori Futamura, and Hiroshi Sugiura. Efficient parameter estimation and implementation of a contour integral-based eigensolver. *J. Algo. Comput. Tech.*, 7:249–269, 2013.

[47] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley,

Lois Curfman McInnes, Karl Rupp, Barry F. Smith, Stefano Zampini, and Hong Zhang. PETSc Web page, 2015. http://www.mcs.anl.gov/petsc.

[48] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[49] Yousef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithms for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[50] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2nd edition, 2003.

研究業績一覧

査読付き論文

[1] 前田 恭行, 櫻井 鉄也, 周回積分を用いた固有値解法の円弧領域に対する拡張, 情報処理学会論文誌 コンピューティングシステム, 8(4):88–97, 2015.

[2] Yasuyuki Maeda, Yasunori Futamura, Akira Imakura and Tetsuya Sakurai, Filter analysis for the stochastic estimation of eigenvalue counts, JSIAM Letters, 7:53–56, 2015.

[3] 山本和磨, 前田恭行, 二村保徳, 櫻井鉄也, 非線形固有値問題の固有値密度推定法における適応的並列アルゴリズム, 情報処理学会論文誌 コンピューティングシステム, 5,(3):22–29, 2012.

[4] Yasuyuki Maeda, Yasunori Futamura and Tetsuya Sakurai, Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane, JSIAM Letters, 3:61–64, 2011.

国際会議発表

[1] Yasuyuki Maeda, Tetsuya Sakurai, James Charles, Michael Povolotskyi, Gerhard Klimeck and Jose E. Roman, A parallel eigensolver using numerical quadrature for annular regions, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing 2015, Tsukuba, Japan, September 14-16, 2015.

[2] Yasuyuki Maeda, Tetsuya Sakurai, Contour Integral Spectral Slicing Solver CISS in SLEPc, 8th International Workshop on Parallel Matrix Algorithms and Applications, Lugano, Switzerland, July 2-4, 2014.

[3] James Charles, Michael Povolotskyi, Yu He, Yasuyuki Maeda, Daniel Lemus, Tillmann Kubis, Gerhard Klimeck, Tetsuya Sakurai, Applications of Eigenvalue Solvers in Nanoelectronic

Device Modeling, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing 2014, Tsukuba, Japan, March 7-9, 2014.

[4] Yasuyuki Maeda, Tetsuya Sakurai, Contour Integral Spectral Slicing Solver CISS in SLEPc, International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing 2014, Tsukuba, Japan, March 7-9, 2014.

[5] Yasuyuki Maeda, Parallel stochastic estimation method of eigenvalue density for nonlinear eigenvalue problems, International Workshop on Computer Science and Technology 2013, China, October 26-27, 2013.

[6] Yasuyuki Maeda, Yasunori Futamura and Tetsuya Sakurai, Parameter Auto-tuning for a Contour Integral based Eigensolver using Stochastic Estimation of Eigenvalue Count, SIAM conference on Computational Science and Engineering, Boston, USA, February 25 - March 1, 2013.

[7] Yasuyuki Maeda, Yasunori Futamura and Tetsuya Sakurai, Effective resource utilization for a contour integral based parallel eigensolver, 7th International Workshop on Parallel Matrix Algorithms and Applications, London, UK, June 28-30, 2012.

[8] Kazuma Yamamoto, Yasuyuki Maeda, Yasunori Futamura and Tetsuya Sakurai, A scalable parallel method for large-scale nonlinear eigenvalue problems, 2012 SIAM Conference on Applied Linear Algebra, Valencia, Spain, June 17-22, 2012.

[9] Yasuyuki Maeda, Yasunori Futamura and Tetsuya Sakurai, Stochastic Estimation Method of Eigenvalue Density for Nonlinear Eigenvalue Problems on the Complex Plane, SIAM Conference on Parallel Processing for Scientific Computing, Georgia, USA, February 14-19, 2012.

国内会議発表

[1] 前田恭行，櫻井鉄也，周回積分型固有値解法における円弧領域への拡張，2015 年並列/分散/協調処理に関する『別府』サマー・ワークショップ, ビーコンプラザ別府国際コンベンションセンター，8/4-8/6，2015.

[2] 前田恭行，櫻井鉄也，確率的固有値分布推定法を用いたバンドギャップの高速計算，日本応用数理学会 2014 年度年会，政策大学院大学，9/3-9/5，2014.

[3] 前田恭行，Claus Aranha，櫻井鉄也， 周回積分を用いた並列固有値解法に対するパラメータ設定について，日本応用数理学会 2013 年度年会，アクロス福岡，9/9-9/11，2013.

[4] 前田恭行，Claus Aranha，櫻井鉄也，固有値分布推定による周回積分を用いた並列固有値解法パラメータ自動設定，並列/分散/協調処理に関するサマー・ワークショップ, 北九州国際会議場, 7/31-8/2，2013.

[5] 前田恭行，櫻井鉄也， ゲルシュゴリン集合の非線形固有値問題への拡張， 日本応用数理学会 2013 年度連合発表会，東洋大学 白山キャンパス，3/14-3/15，2013.

[6] 前田恭行，櫻井鉄也，大規模固有値問題に対する固有値分布推定について，行列・固有値研究部会 第 14 回研究会，筑波大学計算科学研究センター，11/20，2012.

[7] 前田恭行， 周回積分を用いた固有値解法における計算資源の効率的利用， 2012 年度数値解析研究集会，少年自然の家 八ヶ岳荘，9/6-9/8，2012.

[8] 前田恭行，櫻井鉄也，周回積分を用いた固有値解法における計算資源の効率的利用，日本応用数理学会 2012 年度年会，稚内全日空ホテル，8/28-9/2，2012.

[9] 前田恭行，山本和磨，二村保徳，櫻井鉄也，固有値密度推定法における適応型並列アルゴリズムの適用，第 41 回数値解析シンポジウム，伊香保温泉旅館よろこびの宿しん喜，6/6-6/8，2012.

[10] 山本和磨，前田恭行，二村保徳，櫻井鉄也， 非線形固有値問題の固有値密度推定法における適応的並列アルゴリズム，第 12 回先進スーパーコンピューティング環境研究会，東京大学情報基盤センター，4/25，2012.

[11] 前田恭行，二村保徳，櫻井鉄也，非線形固有値問題における複素平面上での大域的固有値分布推定，日本応用数理学会 2011 年度年会，同志社大学今出川キャンパス，9/14-16，2011.

[12] 前田恭行，二村保徳，櫻井鉄也，非線形固有値問題における複素平面上での大域的固有値分布推定，第 40 回数値解析シンポジウム，鳥羽シーサイドホテル，6/20-6/22，2011.

[13] 前田恭行，二村保徳，櫻井鉄也，非線形固有値問題における複素平面上での大域的固有値分布推定，行列・固有値研究部会 第 10 回，国立情報学研究所，11/24，2010.

[14] 前田恭行，二村保徳，櫻井鉄也，非対称行列における複素平面上での大域的固有値分布推定， 日本応用数理学会 2010 年度年会，明治大学，9/7，2010.

[15] 前田恭行，二村保徳，櫻井鉄也，非対称行列における複素平面上での大域的固有値分布推定，2010 年度数値解析研究集会，国立信州高遠青少年自然の家，8/29，2010.