Several Algorithms for the Computation of Matrix Functions

（行列関数計算のためのアルゴリズムの研究）

Fuminori TATSUOKA

Doctoral Thesis

Submitted to
Department of Applied Physics,
Graduate School of Engineering,
Nagoya University

January 2020

Abstract

Matrix functions have been a fundamental topic in matrix analysis. The term "matrix function" means a kind of matrix-valued functions obtained by generalizing scalar functions. Matrix functions are important in the numerical analysis as well as matrix analysis because they arise in many applications. For example, the matrix logarithm arises in quantum information theory, and the matrix fractional power arises in the computational methods for the space-fractional partial differential equation.

Although matrix functions are the generalization of scalar functions, computational methods obtained by substituting matrices into computational methods of scalar functions may have room for improvement. For example, we may improve algorithms in terms of computational cost, accuracy, robustness, and efficiency for high-performance computing. Thus, many computational frameworks and algorithms have been proposed for matrix functions and have been studied still now. The aim of this thesis is to contribute to the computation of matrix functions by proposing several algorithms. In this thesis, we focus on two frameworks of computational methods: numerical quadrature (for the matrix logarithm and the matrix fractional power) and Newton's method (for matrix pth root).

For numerical quadrature, we tried reducing the number of abscissas because it is a crucial factor in the computational cost. In previous studies, the Gaussian quadratures are considered; however, their convergence becomes slow for some cases, and the number of abscissas to obtain an accurate result becomes large. To avoid slow convergence, we consider the double exponential (DE) formula as a new choice of quadrature formulas. However, to utilize the DE formula, we have to choose a finite interval in the computation, which affects the accuracy. Thus, we propose a method of selecting a finite interval on the basis of error analysis, and propose an algorithm for practical use. In addition, we analyze the convergence of the DE formula for SPD matrices to compare with that of the Gaussian quadratures. Besides, we propose preconditioning for quadrature-based algorithms to reduce the number of abscissas with estimates of the effect of the preconditioning. Numerical results showed that the DE formula worked well as intended, the DE formula is effective when the convergence of the Gaussian quadratures are slow, and the preconditioning reduced the number of abscissas as the estimates.

For Newton's method for the matrix pth root, we focus on the incremental Newton (IN) iteration proposed by Iannazzo. The IN iteration is numerically stable, and its computational cost is $\mathcal{O}(n^3 p)$ flops per iteration. We present a cost-efficient variant of the IN iteration, whose computational cost well agrees with $\mathcal{O}(n^3 \log p)$ flops per iteration when $p \leq 100$. Stability analysis shows the variant is also stable. Numerical results support that the variant is stable, and its computational time is smaller than the original one.

Acknowledgments

I would like to express the deepest appreciation to my supervisor Prof. Shao-Liang Zhang for his support during these five years. His advice has expanded my viewpoint on my studies. His support always helped me not only to advance the research but also to mature as a researcher.

I would like to express my gratitude to Prof. Tomohiro Sogabe for his professional advice. His advice has helped improve my skills to be a researcher, including skills in discovering and solving research problems, technical writing, and presentation. My Ph.D. studies would not be possible without their constant support.

I would like to thank Prof. Takahiro Katagiri, Prof. Ken'ichiro Tanaka, Prof. Yukio Tanaka for being a member of the thesis committee out of a busy schedule. Their useful feedback improved the quality of this thesis. Especially, I am grateful for Prof. Ken'ichiro Tanaka for his careful reading and theoretical suggestions.

I am very grateful to Prof. Yuto Miyatake for insightful comments and suggestions on my studies, and his encouragement. I am also very grateful to Prof. Tomoya Kemmochi for his advice on mathematical problems and help with the problems I faced.

I would like to thank all members of Prof. Zhang's research group for their helpful supports during the period of my studies. Especially, I should thank Ms. Yumi Kobayashi and Ms. Kaori Iwata for their supports.

I was supported by JSPS Research Fellowship for Young Scientists.

Finally, I would like to thank my parents and sister for their kind support.

Contents

List of Tables

List of Figures

List of Algorithms

Notations

Symbols

$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$	set of natural/integer/real/complex numbers
$a, b, c, \ldots$ and $\alpha, \beta, \gamma, \ldots$	scalars
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$	vectors
$A, B, C, \ldots$	matrices (except for L)
O	zero matrix
I	identity matrix
L	the Fréchet derivative of matrix functions
A^{T}	transpose of A
A^{H}	conjugate transpose of A
$\|\cdot\|$	vector norms or consistent matrix norms
$\|\cdot\|_p$	p-norms
$\|\cdot\|_F$	the Frobenius norm
$\kappa(A)$	the condition number of A, i.e., $\kappa(A) = \|A\|_2\|A^{-1}\|_2$
$\Lambda(A)$	the set of all eigenvalues of A
$\rho(A)$	the spectral radius of A, i.e., $\rho(A) = \max_{\lambda\in\Lambda(A)} \lvert\lambda\rvert$
δ_{ij}	the Kronecker delta, i.e., $\delta_{ij} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$

Other notations

- The notation $X = \mathcal{O}(\|E\|)$ denotes that $\|X\| \leq c\|E\|$ for some constant c for all sufficiently small $\|E\|$, while $X = o(\|E\|)$ means that $\|X\|/\|E\| \longrightarrow 0$ as $E \longrightarrow O$ [39, p.321].

Chapter 1

Introduction

1.1 Background

Matrix functions have been a fundamental topic in matrix analysis, one of study in linear algebra. In this thesis, the term "matrix function" indicates a matrix-valued function $F : \Psi \longrightarrow \mathbb{C}^{n\times n}$ ($\Psi \subset \mathbb{C}^{n\times n}$) generated from a scalar function $f : \Omega \longrightarrow \mathbb{C}$ ($\Omega \subset \mathbb{C}$) by substituting a square matrix for the variable of f. For example, for a square matrix A, $F(A) = (I - 2A)(3I + 4A + 5A^2)^{-1}$ is a matrix function generated from $f(\lambda) = (1 - 2\lambda)/(3 + 4\lambda + 5\lambda^2)$. A more exact definition of matrix functions is shown in Chapter 2.

In recent years, the computation of matrix functions (and its multiplication by a vector) has attracted attention because matrix functions (and functions of linear operators) appear in many fields of science. For example, the matrix exponential $\exp(A)$ and the matrix trigonometric functions, $\sin(A)$ and $\cos(A)$, appear in exact solutions of linear ordinary differential equations (e.g., [43]). The matrix logarithm $\log(A)$ appears in von Neumann entropy as a quantity in quantum information theory [67]. The matrix (fractional) power A^α appears in the computation of fractional partial differential equations (e.g., [9, 48]).

Studies on matrix functions began in the middle of the 19th century. For example, Cayley referred to the square root of a matrix in [13]. Until the first half of the 20th century, fundamental theories of matrix functions had been developed. In the 20th century, computers and fundamental algorithms for numerical linear algebra have been developed. Accordingly, the computation of matrix functions became one of the hot topics in numerical linear algebra. For more details of the historical background of matrix functions, see, e.g., [39].

Although matrix functions are the generalization of scalar functions, computational methods obtained by substituting matrices into computational methods of scalar functions may have room for improvement. For example, we may improve algorithms in terms of computational cost, accuracy, robustness, and efficiency for high-performance computing. More specifically, the computational cost of such algorithms can be large. The computational cost of matrix-matrix multiplication and matrix inversion are much larger than that of matrix summation and scalar-matrix multiplication. Thus, the computational cost of algorithms depends on the number of multiplication and inversion of matrices. Thinking of accuracy and robustness of algorithms, a single algorithm may not deal with all input matrices because the performance of the algorithm can be largely affected by properties of input matrices such as symmetric structure and condition number.

As a result of great efforts by contributors, many computational methods for matrix functions have been proposed, and some of them have been implemented and provided

as libraries. For example, 14 functions for matrix functions are implemented in Scipy (a Python library for scientific computing) [66]. The status of the implementation of functions concerned with matrix functions in other programming languages as of 2016 is reported in [40]. Though the study on the computation of matrix functions (and its multiplication by a vector) seems not to be completed. In order to make anyone compute matrix functions without any difficulty as if we compute basic scalar functions with built-in functions, we try enhancing the computation of matrix functions.

1.2 Contribution of this thesis

The aim of this thesis is to contribute to the computation of the matrix functions by proposing several algorithms. In this thesis, we focus on the two computational frameworks: numerical quadrature and Newton's method. The details of each framework and our contributions are shown in the following:

Numerical quadrature

The first framework is numerical quadrature. When a matrix function has an integral representation, we can compute the matrix function via numerical quadrature. In this thesis, we focus on the following two integral representation

$$\log(A) = (A - I)\int_0^1 [t(A - I) + I]^{-1}\,\mathrm{d}t\,, \tag{1.1}$$

$$A^\alpha = \frac{\sin(\alpha\pi)}{\alpha\pi}\int_0^\infty (t^{1/\alpha}I + A)^{-1}\,\mathrm{d}t \quad \big(\alpha \in (0,1)\big). \tag{1.2}$$

We refer to two advantages of quadrature-based algorithms. The first one is that numerical quadrature is suitable for parallel computing because the computation of the integrand on each abscissa (integration node) is independent. The second one is that numerical quadrature can compute $\log(A)\boldsymbol{b}$ and $A^\alpha\boldsymbol{b}$ without direct computation of $\log(A)$ and A^α.

Although numerical quadrature is suitable for parallel computing, it is important to reduce the computational cost and save computational resources for practical use. The computational cost of numerical quadrature for (1.1) and (1.2) largely depends on the number of abscissas because there are matrix inversions in both of the integral representations. Therefore reducing the number of abscissas is effective for reducing the computational cost.

In previous studies, the Gaussian quadratures with some variable transformations is considered to compute (1.1) and (1.2). When $A = I$, the transformed integrands become constant function, and the Gaussian quadratures converge within one abscissa. On the other hand, the convergences of the Gaussian quadratures will be slow when A is far away from I (and α is not reciprocal of natural numbers for A^α) because it will be difficult to approximate the integrands with low-degree polynomials.

For example, suppose that A is a symmetric positive definite (SPD) matrix, and we compute (1.2) with Gauss–Jacobi (GJ) quadrature after substituting $t(v) = (1 - v)^\alpha/(1 + v)^\alpha$. In this case, the analysis in [28] shows that the convergence of the GJ quadrature is slow when the condition number of A, $\kappa(A)$, is large. It is because the integrand has singularities at $(1 + \sqrt{\lambda})/|1 - \sqrt{\lambda}|$ for λ being an eigenvalue of A, and the analytic region of the integrand is narrow when $\kappa(A)$ is large.

In this study, we first consider the double exponential (DE) formula as a new choice of quadrature formula for (1.1) and (1.2) to reduce the number of abscissas. Since the DE formula deals with integrals even if the integrand has end point singularities, the DE formula may be useful when the Gaussian quadratures do not perform well. However, the utilization of the DE formula is not straightforward because we need to choose some parameters that affect accuracy. Thus, we propose a method of choosing appropriate parameters on the basis of error analysis, and propose algorithms for practical use. In addition, we analyze the convergence rate of the DE formula for SPD matrices because it allows us to choose a fast quadrature formula.

Besides, focusing on the case where A is an SPD matrix, we discuss another approach for reducing the number of abscissas based on the relations

$$\log(A) = \log(AP) - \log(P), \quad A^\alpha = (AP)^\alpha P^{-\alpha}. \tag{1.3}$$

for a suitable matrix P[1]. Since we can compute $\log(A)$ and A^α via the right-hand-side (RHS) of the relations, these procedures can be regarded as preconditioning, and it may reduce the number of abscissas. We show a choice of P and estimate of the effect of preconditioning.

Newton's method

The second framework is Newton's method. When a matrix function is a solution of a matrix equation, we can compute the matrix function by solving the equation. In this study, we consider the computation of matrix pth root $A^{1/p}$ ($p \in \mathbb{N}$) by solving the equation

$$X^p = A. \tag{1.4}$$

When applying Newton's method for the equation (1.4) directory, the computational cost of the Newton correction is more than $\mathcal{O}(n^3)$ flops, and therefore it is impractical. If we assume that the initial guess X_0 is commute with A, the computation of the Newton correction can be simplified and its cost is $\mathcal{O}(n^3 \log(p))$ flops; however, the simplification cause numerical instability [58]. In [45], a variant of the simplified Newton's iteration called the incremental Newton (IN) iteration which is numerically stable and has computational cost $\mathcal{O}(n^3 p)$ is proposed.

In this study, by rewriting the variant iteration, we obtain another variant whose computational cost is $\mathcal{O}(n^3 \log(p))$ without loss of numerical stability.

1.3 Outline

This thesis consists of two introductory chapters and three chapters referring to our algorithms based on [61, 62, 63, 64] ([64] is a review article for the computational methods of A^α especially quadrature-based algorithms). The connection between the papers [61, 62, 63] is illustrated in Figure 1.1.

In Chapter 2, we first show a brief overview of the theory of matrix functions based on [39] and computational methods of matrix functions. After that, we introduce some fundamental concepts on numerical quadrature and Newton's method.

In Chapter 3 we propose several quadrature-based algorithms for $\log(A)$. We first consider applying the DE formula for the integral representation of $\log(A)$. We show an algorithm for utilizing the DE formula based on error analysis. After that, focusing on the SPD

[1] Some conditions for the relation (1.3) to hold are known [39, pp. 270–271]. Though the commutativity of A and P are required in these conditions, it could not be a necessary condition for the relation (1.3).

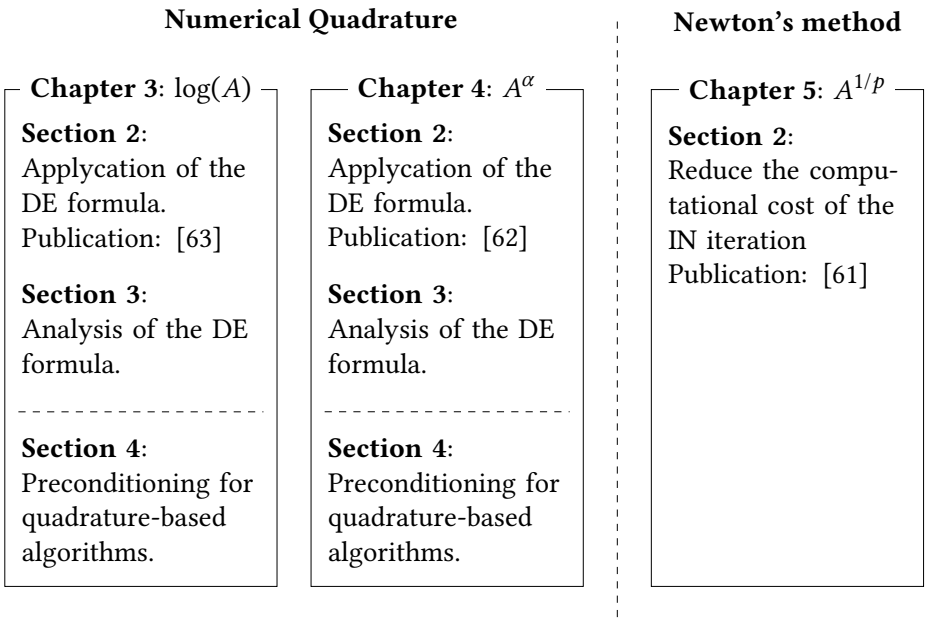


Figure 1.1: Outline of this thesis.

matrices, we analyze the convergence rate of the DE formula and compare it with that of the Gaussian quadrature. After that, we propose preconditioning for reducing the number of abscissas. Lastly, we show numerical results for checking the performance of our algorithms.

In Chapter 4, we propose several quadrature-based algorithms for A^α. The structure of Chapter 4 is similar to that of Chapter 3. We first apply the DE formula for A^α and propose an algorithm for practical use. After that, focusing on the case where A is an SPD matrix, we analyze the convergence rate of DE formula and discuss the preconditioning. Then, we show numerical results for checking the performance of our algorithms.

In Chapter 5, we propose a variant of the IN iteration for the matrix pth root $A^{1/p}$. After recalling the results of the previous studies, we rewrite the IN iteration to the special form whose computational cost is smaller than that of the IN iteration. After that, we check the numerical stability of the proposed iteration. Lastly, we show numerical results for checking computational time and stability.

In Chapter 6, we conclude the study and show some future work.

Chapter 2

Preliminaries

In this chapter, we first recall the theoretical backgrounds of matrix functions based on [39], and we give a brief overview of computational methods for matrix functions. After that, we introduce some fundamental concepts in the numerical analysis that will be used in the following chapters.

2.1 Theory and computational methods of matrix functions

2.1.1 Definitions and properties of a matrix function

Historically, several definitions of matrix function had been proposed. For example, in [55], eight definitions of matrix functions were reviewed, and equivalences of some definitions were shown. In this subsection, we recall two definitions of matrix functions.

We first refer to a definition based on the Cauchy integral formula:

Definition 2.1 ([39, Def. 1.11, p. 8]). *For $A \in \mathbb{C}^{n\times n}$,*

$$F(A) := \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - A)^{-1}\,\mathrm{d}z \tag{2.1}$$

where f is analytic on and inside a closed contour Γ that encloses $\Lambda(A)$.

When the size of A is 1, Definition 2.1 reduces to the Cauchy integral formula. Definition 2.1 is useful because we can define functions of linear operators through Definition 2.1 and we can compute any matrix function by numerical quadrature if we know the eigenvalue distribution of A.

The second definition is based on the Jordan canonical form. The Jordan canonical form is one of the generalizations of matrix diagonalization. If A is diagonalizable, a matrix function $F(A)$ will be simply defined as $F(A) = Z\mathrm{diag}(f(\lambda_1), \dots, f(\lambda_n))Z^{-1}$ where λ_i ($i = 1, \dots, n$) are the eigenvalues of A and Z is the eigenvectors of A. Thus, it is natural to use the Jordan canonical form for extending the definition of functions of diagonalizable matrices to general matrices.

We recall the Jordan canonical form and its existence based on [39, p. 322]. Any matrix

$A \in \mathbb{C}^{n\times n}$ can be expressed in the Jordan canonical form

$$Z^{-1}AZ = J = \operatorname{diag}(J_1, J_2, \dots, J_p), \tag{2.2}$$
$$J_k = J_k(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k\times m_k},$$

where Z is non-singular and $m_1 + m_2 + \cdots + m_p = n$, and λ_k are the eigenvalues of A. The matrices J_k are called Jordan blocks. The Jordan matrix J is unique up to the ordering of the blocks J_k but the transforming matrix Z is not unique.

Before showing the definition, we recall the following terminology which will be used in the definition of a matrix function:

Definition 2.2 ([39, Def. 1.1, p. 3])**.** *Let $\lambda_1, \dots, \lambda_s$ be the distinct eigenvalues of $A \in \mathbb{C}^{n\times n}$, and let n_i be the order of the largest Jordan block in which λ_i appears. Then, the function f is said to be defined on the spectrum of A if the values*

$$f^{(j)}(\lambda_i), \quad (j = 0, \dots, n_i - 1, \quad i = 1, \dots, s)$$

exist.

The definition of a matrix function based on the Jordan canonical form as follows:

Definition 2.3 ([39, Def. 1.2,])**.** *Let f be defined on the spectrum of $A \in \mathbb{C}^{n\times n}$ and let A have the Jordan canonical form* (2.2)*. Then*

$$F(A) := ZF(J)Z^{-1} = Z\operatorname{diag}(F(J_1), F(J_2), \dots, F(J_p))Z^{-1},$$

where

$$F(J_k) := \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \cdots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix}.$$

When the size of A is 1, Definition 2.1 reduce to the scalar function f itself. Definition 2.3 is useful because some matrix functions can be regarded as a solution of the matrix equation. For example, the square root of A can be regard as a solution of the equation $X^2 = A$ of X, the logarithm $\log(A)$ is a solution of $\exp(X) = A$, and the Lambert W function is a solution of $X\exp(X) = A$. It is known that Definition 2.1 and Definition 2.3 are equivalent when f is analytic [39, Thm. 1.12].

In the final part of this subsection we recall several properties of matrix functions in the following theorem:

Theorem 2.4 ([39, Thm. 1.13])**.** *Let $A \in \mathbb{C}^{n\times n}$ and let f be defined on the spectrum of A. Then*

1. *$F(A)$ commutes with A;*
2. *$F(A^{\mathsf{T}}) = (F(A))^{\mathsf{T}}$;*
3. *$F(XAX^{-1}) = XF(A)X^{-1}$;*

4. *The eigenvalues of $F(A)$ are $f(\lambda_i)$, where λ_i are the eigenvalues of A;*

5. *If X computes with A then X commutes with $F(A)$;*

6. *If $A = (A_{ij})$ is block triangular then $F = F(A)$ is block triangular with the same block structure as A, and $F_{ii} = f(A_{ii})$;*

7. *If $A = \mathrm{diag}(A_{11}, A_{22}, \dots, A_{mm})$ is block diagonal then*

$$F(A) = \mathrm{diag}(F(A_{11}), F(A_{22}), \dots, F(A_{mm}));$$

8. *$F(I \otimes A) = I \otimes F(A)$, where $\otimes$ is the Kronecker product;*

9. *$F(A \otimes I) = F(A) \otimes I$.*

Some of the statements in Theorem 2.4 help us compute matrix functions and analyze algorithms. For example, the third statement allows us to compute matrix functions after Schur factorization, and it may reduce the computational cost. Another example is that the fourth statement allows us to reduce the analysis of algorithms for $F(A)$ to that of $f(\lambda_i)$.

2.1.2 Computational methods of matrix functions

In this subsection, we give a brief overview of methods of computing matrix functions. The methods can be classified roughly into two groups depending on the purpose. The first group is algorithms for computing $F(A)$ itself. The other group is algorithms specialized to compute $F(A)\boldsymbol{b}$ for a vector $\boldsymbol{b} \in \mathbb{C}^n$.

Computational methods for $F(A)$ itself

The Schur–Parlett algorithm [16] is a method that can compute arbitrary matrix functions if we have a scalar function f (and its derivative). The key idea of the Schur–Parlett algorithm is the fact that we can compute functions of triangular matrices explicitly. To the best of our knowledge, there are no computational methods other than the Schur–Parlett algorithm which can compute matrix functions for arbitrary f [1].

One of the computational methods for matrix exponential $\exp(A)$ is the scaling and squaring algorithm, e.g., [38], which is based on the relation $\exp(A) = [\exp(2^{-s}A)]^{2^s}$ and the Padé approximation of $\exp(A)$. In recent years, the scaling and squaring algorithm has been improved in order to reduce the computational cost, obtain an accurate result, or work with multiprecision arithmetic [2, 7, 26, 33, 54]. See, e.g., [39, Chap. 10], for further computational methods for $\exp(A)$.

For the matrix logarithm $\log(A)$, the inverse scaling and squaring algorithm [49] is one of the standard algorithms. Like the scaling and squaring algorithm for $\exp(A)$, the inverse scaling and squaring algorithm has been improved in terms of the computational cost, accuracy, and support for multiprecision environments [3, 25]. See, e.g., [39, Chap. 11], for further computational methods for $\log(A)$.

Computational methods for the matrix fractional power $A^\alpha = \exp(\alpha \log(A))$ include Padé approximation [41, 42], Schur logarithmic algorithm [47], and Numerical quadrature

[1] Since matrix functions is defined based on the Cauchy integral formula, one may compute matrix functions by numerical quadrature. However, in order to compute, we have to know the region which f is analytic and the eigenvalue distribution of A. To the best of our knowledge, no algorithm can deal with all function f and a matrix A.

[10, 28, 62]. When α is reciprocal of a natural number, i.e., $\alpha = 1/p$ ($p \in \mathbb{N}$), we can use a more variety of methods. For example, direct methods called the Schur algorithm is proposed in [8, 58], and an efficient implementation for $\alpha = 1/2$ is studied in [20]. Since $A^{1/p}$ is the solution of the equation $X^p = A$, we can compute $A^{1/p}$ by solving the equation by iterative methods [21, 31, 37, 45, 46]

We only briefly refer to computational methods for other matrix functions. For the computation of matrix sign function, see, e.g., [39, Chap. 5] and [14, 15, 21]. For the computation of matrix trigonometric functions, see, e.g., [39, Chap. 12] and [4, 56]. In recent years, computational methods have been developed for special functions such as the Gamma function [12] and the Lambert W function [27]. The verified computation of matrix functions has also attracted attention, e.g., [30, 50, 51, 52, 53].

Computational methods for $F(A)\boldsymbol{b}$

Methods for computing the multiplication of a matrix function and a vector can be classified into two groups: quadrature-based algorithms and algorithms based on the projection to a suitable subspace.

Since $F(A)$ has the integral representation (2.1), we can compute $F(A)\boldsymbol{b}$ via

$$F(A)\boldsymbol{b} = \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - A)^{-1}\boldsymbol{b}\, \mathrm{d}z$$

if we know the eigenvalue distribution of A. By solving linear systems $(zI - A)\boldsymbol{x} = f(z)\boldsymbol{b}$ for $\boldsymbol{x}$, we can compute $F(A)\boldsymbol{b}$ without direct computation of $F(A)$. In fact, some matrix functions such as $\log(A)$, A^α, and $\mathrm{sign}(A)$ have integral representations whose integral interval is on the real axis, and numerical quadrature is suitable for such functions.

The projection method is as follows. Let $V_m = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_m] \in \mathbb{C}^{n\times m}$ ($m \ll n$) be an orthonormal basis of a subspace in $\mathbb{C}^n$ and let $H_m = V_m^{\mathsf{H}} A V_m \in \mathbb{C}^{m\times m}$. Then, we approximate $F(A)\boldsymbol{b}$ as

$$F(A)\boldsymbol{b} = \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - A)^{-1}\boldsymbol{b}\, \mathrm{d}z \approx \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z) V_m (zI - H_m)^{-1} V_m^{\mathsf{H}}\boldsymbol{b}\, \mathrm{d}z = V_m F(H_m) V_m^{\mathsf{H}}\boldsymbol{b}.$$

For the projection method, we generally consider the Krylov subspace $\mathcal{K}_m(A, \boldsymbol{b}) = \mathrm{span}\{\boldsymbol{b}, A\boldsymbol{b}, \ldots, A^{m-1}\boldsymbol{b}\}$ or the extended Krylov subspace $\mathcal{K}_{q+1,m}(A, \boldsymbol{b}) = \mathcal{K}_{q+1}(A, \boldsymbol{b}) \cap \mathcal{K}_m(A^{-1}, \boldsymbol{b})$, which includes the rational Krylov subspace $\mathcal{Q}_m(A, \boldsymbol{b}) = \mathcal{K}_{m,m}(A, \boldsymbol{b})$. One of the disadvantages of the projection method is to compute and store the dense matrix V_m. In order to overcome this disadvantage, a technique called restart, which is originally developed for solving linear systems, has been studied in recent years. See e.g., [1, 29] for more details.

It is not easy to compare quadrature-based algorithms with projection methods for $F(A)\boldsymbol{b}$ because the computation time will be largely affected by the properties of A, computational environments, and implementations of algorithms. For example, quadrature-based algorithms are suitable for parallel computing while it requires solving linear systems. On the other hand, the projection methods with the Krylov subspace are free from solving linear systems while these algorithms are sequential.

2.2 Basics of numerical analysis

2.2.1 Numerical quadrature

Generally, we approximate a given integral by a linear combination of the integrand, i.e., $\int_a^b f(x)\,\mathrm{d}x \approx \sum_{k=1}^n w_k f(x_k)$ with suitable abscissas x_k and weights w_k. More precisely, we approximate the given integral f as a function whose integral can be computed exactly, and we perform integration of the approximated function. In this subsection, we recall some results on the Gaussian quadrature and the double exponential formula. For more details of general topics of numerical quadrature, see e.g. [18].

The Gaussian quadrature

First, we recall results on the Gaussian quadrature formula. Let $[a, b]$ be an interval and $w(x)$ be a function such that there exists $\int_a^b w(x)\,\mathrm{d}x$. Suppose that $p_m(x)$ $(m = 0, 1, 2, \dots)$ be polynomials of degree m where they are orthonormal polynomials with respect to $w(x)$, i.e., $\int_a^b w(x)p_i(x)p_j(x)\,\mathrm{d}x = \delta_{ij}$. Let x_k $(k = 1, \dots, m)$ be the roots of $p_m(x)$ and

$$w_k = -\frac{c_{m+1}}{c_m}\frac{1}{p_{m+1}(x_k)p'_m(x_k)} \quad (k = 1, \dots, m)$$

where $c_{\tilde{m}}$ is the coefficient of $x^{\tilde{m}}$ in $p_{\tilde{m}}(x)$. Then, for any polynomial $\tilde{p}(x)$ whose degree is less than or equal to $2m - 1$, it holds that

$$\int_a^b w(x)\tilde{p}(x)\,\mathrm{d}x = \sum_{k=1}^m w_k\tilde{p}(x_k).$$

See, e.g., [18] for more details. When abscissas and weights of quadrature formula are chosen as above, we call the quadrature formula the Gaussian quadrature. Especially, we call the formula the Gauss–Legendre quadrature when $[a, b] = [-1, 1]$ and $w(x) = 1$, we call the formula the Gauss–Jacobi quadrature when $[a, b] = [-1, 1]$ and $w(x) = (1 - x)^\alpha(1 + x)^\beta$ for $\alpha, \beta > -1$.

When the integrand in a given integral is analytic on the integral intervals, the Gaussian quadrature converges exponentially. More precisely, the error of the m-point Gaussian quadrature where $[a, b] = [-1, 1]$ is known as follows [28]. Suppose that Γ is ellipse in the complex plane such that its foci are ± 1 and f is analytic in its inside. Let τ be the sum of semi-axes of Γ. Then, the error of m-point Gaussian quadrature is

$$\left|\int_{-1}^1 w(x)f(x)\,\mathrm{d}x - \sum_{k=1}^m w_k f(x_k)\right| \le 4\mu_0\frac{1}{\tau^{2m}}\frac{\tau^2}{\tau^2 - 1}\max_{z\in\Gamma}|f(z)|,$$

where $\mu_0 = \int_{-1}^1 w(x)\,\mathrm{d}x$.

Double exponential formula

The double exponential (DE) formula is proposed in [60] based on the fact that trapezoidal rule for the integral of an analytic function on the real axis converges exponentially:

Theorem 2.5 ([65, Thm. 5.1]). *Suppose g is analytic in the strip $|\mathrm{Im}(z)| < d$ for some $d > 0$. Suppose further that $g(z) \longrightarrow 0$ uniformly as $|z| \longrightarrow \infty$ in the strip, and for some c, it satisfies*

$$\int_{-\infty}^{\infty} |g(x + \mathrm{i}y)| \, \mathrm{d}x \leq c$$

for all $y \in (-d, d)$. Then, for any $h > 0$, the sum $h \sum_{k=-\infty}^{\infty} g(kh)$ exists and satisfies

$$\left| \sum_{k=-\infty}^{\infty} g(kh) - \int_{-\infty}^{\infty} g(x) \, \mathrm{d}x \right| \leq \frac{2c}{\exp(2\pi d/h) - 1}$$

and the quantity $2c$ in the numerator is as small as possible.

The fundamental idea of the DE formula is to transform a given integral into an integral on the real axis by appropriate transformations.

The followings show procedures of the DE formula. Consider the computation of the integral in the form of

$$\int_a^b f(x) \, \mathrm{d}x$$

where $f(x)$ is analytic on (a, b). First, we apply a variable transformation $x = x(t)$ such that the transformed interval is $(-\infty, \infty)$ and the transformed integrand decays double exponentially at $t \longrightarrow \pm\infty$. For example, we use $x(t) = \tanh(\frac{\pi}{2} \sinh(t))$ when $(a, b) = (-1, 1)$, and we use $x(t) = \exp(\frac{\pi}{2} \sinh(t))$ when $(a, b) = (0, \infty)$ and $f(x)$ decays polynomially as $x \longrightarrow \infty$. Then, we truncate the infinite integral interval into a finite interval:

$$\int_{-\infty}^{\infty} x'(t) f(x(t)) \, \mathrm{d}t \approx \int_l^r x'(t) f(x(t)) \, \mathrm{d}t \, .$$

Since the transformed integrand decays very rapidly, we obtain accurate results if the finite interval $[l, r]$ is sufficiently wide. Generally, we truncate the infinite interval so that the magnitude of the trapezoidal error is equal to that of the interval truncation error. When the number of abscissas is m ($m \gg 1$), we select the finite interval $[-\log(4\tau m)/2, \log(4\tau m)/2]$ where τ is a constant depending on $x'(t)f(x(t))$. See, e.g., [59, 60] for more details. At last, we compute the truncated integral by using the trapezoidal formula.

2.2.2 Newton's method

Newton's method is one of the computational methods of solving equations. Consider the equation $f(x) = 0$ where $f : \mathbb{C} \longrightarrow \mathbb{C}$ is a differential function. Then, Newton's iteration for the equation is

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

If x_0 is sufficiently close to the solution of the equation, then the sequence generated by Newton's iteration converges to the solution quadratically.

Newton's method can be extended for solving the equation $f(x) = 0$ where f is a function between Banach spaces. In this subsection, we focus on Newton's method for an equation $F(X) = O$ where F is a matrix function. For more general discussions on Newton's method, see, e.g., [68]. For solving $F(X) = O$, we usually consider the Fréchet derivative. The Fréchet derivative for matrix functions is defined as follows:

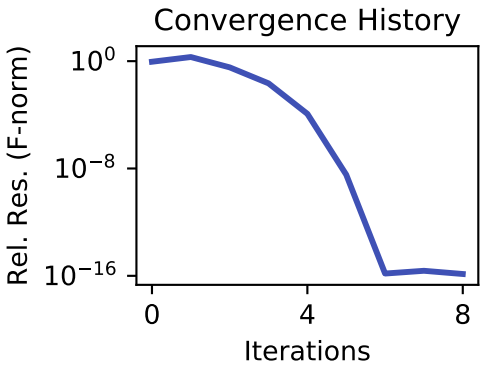


Figure 2.1: Convergence history of Newton's method for the equation $X^2 - A = O$ where A is the Frank matrix of size 5.

Definition 2.6 ([39, p. 56]). *The Fréchet derivative of a matrix function $F : \mathbb{C}^{n\times n} \longrightarrow \mathbb{C}^{n\times n}$ at a point $X \in \mathbb{C}^{n\times n}$ is a linear mapping*

$$\begin{aligned} \mathbb{C}^{n\times n} &\xrightarrow{L} \mathbb{C}^{n\times n} \\ E &\mapsto L(X, E) \end{aligned}$$

such that for all $E \in \mathbb{C}^{n\times n}$

$$F(X + E) - F(X) - L(X, E) = o(\|E\|).$$

Our notations regarding the Fréchet derivative follow the notations in [39, p. 56]. If we need to show the dependence on F we will write $L_F(X, E)$. When we want to refer to the mapping at X and not its value in a particular direction we will write $L(X)$. For a more general definition of the Fréchet derivative, see, e.g., [5]. Then, Newton's iteration for $F(X) = O$ is as follows:

$$\begin{cases} \text{solve } L(X_k, E_k) = F(X_k) \text{ for } E_k, \\ X_{k+1} = X_k - E_k. \end{cases}$$

For example, consider the case where $F(X) = X^2 - A$. Then, the Fréchet derivative of F at X is $L(X, E) = XE + EX$, and Newton's iteration can be written as

$$\begin{cases} \text{solve } X_kE_k + E_kX_k = X_k^2 - A \text{ for } E_k, \\ X_{k+1} = X_k - E_k. \end{cases}$$

We show a numerical example of Newton's method for $X^2 - A = O$. The convergence history of Newton's method where A is the Frank matrix of size 5 generated by using `MatrixDepot.jl` [69] and $X_0 = I$ is illustrated in Figure 2.1. Figure 2.1 shows the quadratically converges of Newton's method.

Chapter 3

Quadrature-based algorithms for $\log(A)$

3.1 Introduction

In this chapter, we consider quadrature-based algorithms for the matrix logarithm. Before discussing quadrature algorithms, we give a brief overview of applications and computational methods of the matrix logarithm.

Matrix logarithms arise in many fields of science. For example, von Neumann entropy $s = -\operatorname{tr}(A\log(A))$ appears as a physic physical quantities in Quantum information theory [67]. In the context of machine learning, log-determinant $\log(\det(A)) = \operatorname{tr}(\log(A))$ is computed through Hutchinson's trace estimator for the matrix logarithm [23, 35, 44]. Other examples include buckling simulation [57] and quantum chemistry [36].

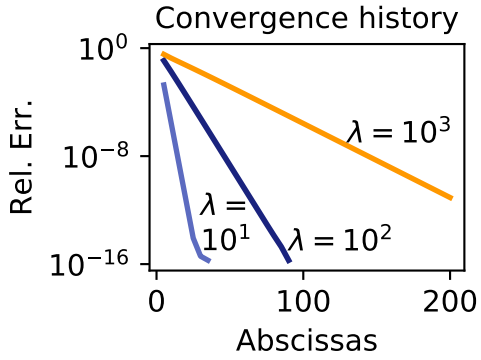
Computational methods of the matrix logarithm include the inverse scaling and squaring algorithms [3, 25], an iterative method based on the arithmetic–geometric mean iteration [11], and numerical quadrature [17]. In this chapter, we consider numerical quadrature, which employs the following integral representation:

$$\log(A) = \int_0^1 (A - I)\left[t(A - I) + I\right]^{-1} \mathrm{d}t\,. \tag{3.1}$$

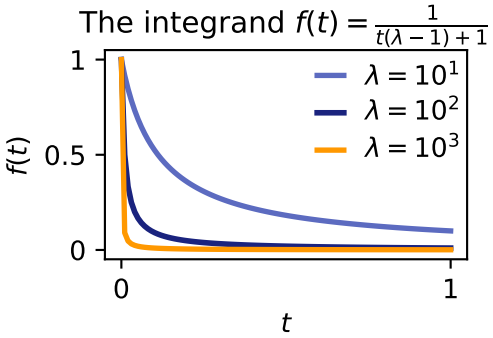
As stated in Section 1.2, the computational cost of numerical quadrature for (3.1) largely depends on the number of abscissas because the integrand in (3.1) includes a matrix inversion. For this reason, it is important to obtain accurate results with a small number of abscissas. Thus, our aim in this chapter is reducing the number of abscissas.

In previous studies, e.g. [17], the Gauss–Legendre quadrature is considered for computing (3.1). In [22], it is shown that the approximate of the m-point GL quadrature for (3.1) is equivalent to the Padé approximate of order $[m/m]$ for the $\log(A)$ at I when $\rho(A - I) < 1$. From the fact that the GL quadrature is equivalent to the rational approximate at I, the GL quadrature converges fast when A is close to I. Conversely, the convergence of the GL quadrature will be slow when A is far away from I. Here, as a simple example, we show the convergence of the GL quadrature of the scalar logarithm in Figure 3.1. Figure 3.1a shows that the convergence of the GL quadrature becomes slow when λ is far away from 1, and Fig. 3.1b suggest that the reason of the slow convergence of the GL quadrature is the rapid changes of values of the integrand.

In this study, to avoid slow convergence, we first consider the double exponential (DE) formula [60] as another quadrature formula for computing $\log(A)$. As mentioned in Chapter 2, the DE formula can compute integrals efficiently, even if the integrands have endpoint singularities. For this reason, the DE formula may be useful in scenarios in which the GL



(a) Convergence histories of the GL quadrature



(b) The integrand of the integral representation of $\log(\lambda)$

Figure 3.1: The integrand for of a integral representation of the scalar logarithm $\log(\lambda) = (\lambda - 1)\int_0^1 1/(t(\lambda-1)+1)\,\mathrm{d}t$ and convergence histories of the GL quadrature for the integral representation.

quadrature does not perform well. However, utilization of the DE formula is not straightforward because the integrand in (3.1) is a matrix-valued function. When we use the DE formula, we truncate the infinite interval after the DE transformation. We generally truncate it so that the magnitude of the interval truncation error and the trapezoidal error are both equal to each other by using the decay rate of the integrand. However, it is complicated to calculate the decay rate of each element of the integrand in (3.1) and to determine the appropriate finite interval. Alternatively, we estimate the upper bound on the truncation error directory instead of considering the decay rate of the integrand. Besides, we propose an algorithm which determine a finite interval according to a given tolerance. In addition, focusing on the case where A is an SPD matrix, we analyze the convergence rate of the DE formula, and compare it to that of the GL quadrature.

Besides, we consider the preconditioning for an SPD matrix A based on the relation

$$\log(A) = \log(AP) - \log(P) \tag{3.2}$$

for a suitable matrix $P \in \mathbb{C}^{n\times n}$. After selecting P, we can compute $\log(A)$ via the RHS of (3.2). Thus, this procedure can be regarded as preconditioning, and it may reduce the number of abscissas. We show a choice of P and estimate the effect of preconditioning.

The remainder of this chapter is as follows. In Section 3.2, we propose a method to choose an appropriate interval for the DE formula based on truncation error analysis. We also show an algorithm for practical use. In Section 3.3, we analyze the convergence rate of the DE formula focusing on SPD matrices, and we compare it with that of the GL quadrature. In Section 3.4, we show a choice of P for the latter approach. In Section 3.5, we show numerical results. In Section 3.6, we conclude this chapter.

3.2 An algorithm for $\log(A)$ based on the double exponential formula

In this section we propose a method of selecting a finite interval for the DE formula by estimating the interval truncation error and present two algorithms. Before considering the truncation error, let us apply the following transformations to (3.1). By substituting

$t(u) = (u+1)/2$ in (3.1), we obtain

$$\log(A) = \int_{-1}^{1} F(u)\,\mathrm{d}u \quad \big(F(u) = (A-I)[(1+u)A + (1-u)I]^{-1}\big). \tag{3.3}$$

Then, by applying the DE transformation $u(x) = \tanh(\sinh(x))$, it follows that

$$\log(A) = \int_{-\infty}^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x, \tag{3.4}$$

where

$$F_{\mathrm{DE}}(x) = u'(x)F(u(x)) = \frac{\cosh(x)}{\cosh^2(\sinh(x))}(A-I)\Big[\big(1+\tanh(\sinh(x))\big)A + \big(1-\tanh(\sinh(x))\big)I\Big]^{-1}.$$

The matrix $[(1+\tanh(\sinh(x)))A + (1-\tanh(\sinh(x))I)]$ in the integrand F_{DE} is non-singular for any $x \in \mathbb{R}$. In Subsection 2.1, we derive an upper bound on the error between the integral in (3.4) and the same integral defined in the finite interval $[l, r]$,

$$\left\| \log(A) - \int_{l}^{r} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\|. \tag{3.5}$$

In Subsection 2.2, we propose a method of selecting the interval $[l, r]$ so that the relative truncation error is small than or equal to a given tolerance ε. Our algorithm is described in Subsection 2.3[1].

3.2.1 Estimation of the error from the interval truncation

The error stems from the interval truncation (3.5) can be rewritten as

$$\left\| \log(A) - \int_{l}^{r} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| = \left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x + \int_{r}^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\|.$$

Using the triangle inequality in the right-hand side (RHS) of (3.5), it holds that

$$\left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x + \int_{r}^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \le \left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| + \left\| \int_{r}^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \tag{3.6}$$

By estimating the RHS of (3.6), we obtain an upper bound on (3.5).

Initially, we focus on the first term of the RHS in (3.6). To avoid cumbersome notation, instead of $F_{\mathrm{DE}}(x)$, which includes hyperbolic functions, we consider the integrand of (3.3). By considering the transformation $u(x) = \tanh(\sinh(x))$, we have

$$\left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| = \left\| \int_{-1}^{a} F(u)\,\mathrm{d}u \right\|, \tag{3.7}$$

where $a = \tanh(\sinh(l))$. The following lemma shows an upper bound on the RHS of (3.7) when $a \approx -1$ enough to warrant the use of the Neumann series expansion of $F(u)$.

[1] In [63], the algorithm for selecting $[l, r]$ suppose that ε is sufficiently small because it uses a first-order approximation of upper bound on the truncation error. In this thesis, by slightly modifying the derivation of upper bounds, we develop an algorithm without first-order approximations.

Lemma 3.1. *Suppose that $A \neq I$ and $-1 < a \leq -1 + 1/\|A - I\|$. Then, it holds that*

$$\left\| \int_{-1}^{a} F(u)\,\mathrm{d}u \right\| \leq \|A - I\|(1 + a), \tag{3.8}$$

where $F(u)$ is defined in (3.3)*. Therefore,*

$$\left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \leq \|A - I\|(1 + a) \tag{3.9}$$

where $l = \operatorname{asinh}(\operatorname{atanh}(a))$.

Proof. The integrand $F(u)$ can be rewritten as

$$F(u) = [(1 + u)A + (1 - u)I]^{-1} = [(1 + u)(A - I) + 2I]^{-1} = \frac{1}{2}\left[I - \left(-\frac{1 + u}{2}(A - I)\right)\right]^{-1}, \tag{3.10}$$

and the Neumann series expansion is applicable to (3.10) because for all $u \in [-1, a]$ where $a \leq 1/\|A - I\| - 1$, it holds that

$$\left\| \frac{1 + u}{2}(A - I) \right\| = \frac{1 + u}{2}\|A - I\| \leq \frac{1 + a}{2}\|A - I\| \leq \frac{1}{2}\ (< 1).$$

After applying the Neumann expansion to (3.10), we have

$$F(u) = \frac{1}{2}\sum_{k=0}^{\infty}\left(-\frac{1 + u}{2}(A - I)\right)^{k}. \tag{3.11}$$

By substituting (3.11) to the integral in the LHS of (3.8), we have

$$\begin{aligned}
\int_{-1}^{a} F(u)\,\mathrm{d}u &= (A - I)\int_{-1}^{a}\frac{1}{2}\sum_{k=0}^{\infty}\left(-\frac{1 + u}{2}(A - I)\right)^{k}\mathrm{d}u \\
&= \sum_{k=0}^{\infty}\frac{(-1)^k}{2^{k+1}}(A - I)^{k+1}\int_{-1}^{a}(1 + u)^k\,\mathrm{d}u \\
&= \sum_{k=0}^{\infty}\frac{(-1)^k}{2^{k+1}}(A - I)^{k+1}\frac{(1 + a)^{k+1}}{k + 1} \\
&= \sum_{k=0}^{\infty}(-1)^k\frac{1}{k + 1}\left[\frac{1 + a}{2}(A - I)\right]^{k+1}.
\end{aligned}$$

By considering triangle inequality and consistency of the norm we get the following:

$$\begin{aligned}
\left\| \int_{-1}^{a} F(u)\,\mathrm{d}u \right\| &\leq \sum_{k=0}^{\infty}\frac{1}{k + 1}\left\| \frac{1 + a}{2}(A - I) \right\|^{k+1} \\
&= \left\| \frac{1 + a}{2}(A - I) \right\| \sum_{k=0}^{\infty}\frac{1}{k + 1}\left\| \frac{1 + a}{2}(A - I) \right\|^{k} \\
&\leq \frac{1 + a}{2}\|A - I\| \sum_{k=0}^{\infty}\frac{1}{0 + 1}\left(\frac{1}{2}\right)^{k} \\
&= (1 + a)\|A - I\|.
\end{aligned}$$

We obtain (3.8) substituting $u(x) = \tanh(\sinh(x))$ to □

The calculation to estimate the second term on the RHS of (3.6) is similar to that of the first term. By applying the transformation $u(x) = \tanh(\sinh(x))$, we get the following:

$$\left\| \int_r^\infty F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| = \left\| \int_b^1 F(u)\,\mathrm{d}u \right\|, \tag{3.12}$$

where $b = \tanh(\sinh(r))$.

The following lemma shows an upper bound on (3.12), if b is close enough to 1 to warrant the use of the Neumann series expansion of $F(t)$.

Lemma 3.2. *Suppose that $A \neq I$ and $1 - 1/\|A^{-1}\|\|A - I\| \leq b < 1$. Then, it holds that*

$$\left\| \int_b^1 F(u)\,\mathrm{d}u \right\| \leq \|A^{-1}\|\|A - I\|(1 - b), \tag{3.13}$$

and therefore

$$\left\| \int_r^\infty F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \leq \|A^{-1}\|\|A - I\|(1 - b) \tag{3.14}$$

where $b = \tanh(\sinh(r))$.

Proof. The outline of this proof is similar to that of Lemma 3.1. The integrand $F(u)$ can be rewritten as

$$\begin{aligned} F(u) &= [(1+u)A + (1-u)I]^{-1} \\ &= [u(A-I) + (A+I)]^{-1} \\ &= [-(1-u)(A-I) + 2A]^{-1} \\ &= \frac{1}{2}A^{-1}\left[I - \frac{1-u}{2}A^{-1}(A-I)\right]^{-1} \end{aligned}, \tag{3.15}$$

and the Neumann series expansion is applicable to (3.15) because for all $u \in [b, 1]$ where $b \geq 1 - 1/\|A^{-1}\|\|A - I\|$, it holds that

$$\left\| \frac{1-u}{2}A^{-1}(A-I) \right\| \leq \frac{1-u}{2}\|A^{-1}\|\|A-I\| = \frac{1-b}{2}\|A^{-1}\|\|A-I\| \leq \frac{1}{2}\ (< 1).$$

After applying the Neumann expansion to (3.15), we have

$$F(u) = \frac{1}{2}A^{-1}\sum_{k=0}^{\infty}\left(\frac{1-u}{2}A^{-1}(A-I)\right)^k. \tag{3.16}$$

By substituting (3.16) to the integral in the LHS of (3.13), we have

$$\begin{aligned} \int_b^1 F(u)\,\mathrm{d}u &= (A-I)\int_b^1 \frac{1}{2}A^{-1}\sum_{k=0}^{\infty}\left(\frac{1-u}{2}A^{-1}(A-I)\right)^k \mathrm{d}u \\ &= \sum_{k=0}^{\infty}\frac{1}{2^{k+1}}A^{-(k+1)}(A-I)^{k+1}\int_b^1 (1-u)^k\,\mathrm{d}u \\ &= \sum_{k=0}^{\infty}\frac{1}{2^{k+1}}A^{-(k+1)}(A-I)^{k+1}\frac{(1-b)^{k+1}}{k+1} \\ &= \sum_{k=0}^{\infty}\frac{1}{k+1}\left[\frac{1-b}{2}A^{-1}(A-I)\right]^{k+1}. \end{aligned}$$

By considering triangle inequality and consistency of the norm we get the following:

$$\begin{aligned}\left\|\int_b^1 F(u)\,\mathrm{d}u\right\| &\le \sum_{k=0}^{\infty}\frac{1}{k+1}\left\|\frac{1-b}{2}A^{-1}(A-I)\right\|^{k+1}\\ &= \left\|\frac{1-b}{2}A^{-1}(A-I)\right\|\sum_{k=0}^{\infty}\frac{1}{k+1}\left\|\frac{1-b}{2}A^{-1}(A-I)\right\|^{k}\\ &\le \frac{1-b}{2}\|A^{-1}\|\|A-I\|\sum_{k=0}^{\infty}\frac{1}{0+1}\left(\frac{1}{2}\right)^{k}\\ &= (1-b)\|A^{-1}\|\|A-I\|\end{aligned}$$

□

In the final part of this subsection, we have the following upper bound by combining (3.9) and (3.14):

Proposition 3.3. *Suppose that $A \neq I$. For given $[l, r]$, let $a = \tanh(\sinh(l))$ and $b = \tanh(\sinh(r))$. Then, if $a \le -1 + 1/\|A-I\|$ and $b \ge 1 - 1/\|A-I\|\|A^{-1}\|$, it holds that*

$$\left\|\log(A) - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\| \le \|A-I\|(1+a) + \|A-I\|\|A^{-1}\|(1-b). \tag{3.17}$$

3.2.2 Setting the integration interval

To develop algorithms for computing $\log(A)$ based on the DE formula, we need to determine the appropriate finite integration interval, $[l, r]$ in advance. The finite interval should be ideally set so that the relative error is guaranteed to be smaller than or equal to a given tolerance, $\epsilon > 0$, i.e.,

$$\frac{\left\|\log(A) - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\|}{\|\log(A)\|} \le \epsilon.$$

To accomplish this, a lower bound on $\|\log(A)\|$ must be estimated. The following lemma shows a lower bound in terms of the spectral radius of A.

Lemma 3.4 ([63, Lem. 4])**.** *Let $\rho(\cdot)$ be the spectral radius. Then, the following two inequalities hold:*

$$\|\log(A)\| \ge |\log(\rho(A))|, \tag{3.18}$$
$$\|\log(A)\| \ge |\log(\rho(A^{-1}))|. \tag{3.19}$$

Proof. By using the consistency of the norm and the fact that any eigenvalue of $\log(A)$ is equal to $\log(\lambda)$, for some λ being an eigenvalue of A, we have the lower bound in (3.18) as $\|\log(A)\| \ge \rho(\log(A)) \ge |\log(\rho(A))|$. The lower bound in (3.19) can be obtained in a similar way. □

In the following proposition, we show how to set a finite interval such that the relative truncation error in 2-norm is smaller than or equal to a given tolerance, $\epsilon > 0$.

Proposition 3.5. *Suppose that $A \neq I$. Let θ be a lower bound on $\|\log(A)\|_2$, and the tolerance $\epsilon > 0$ satisfy*

$$\epsilon < \min\left\{\frac{4\|A-I\|_2\|A^{-1}\|_2}{\theta(1+\|A^{-1}\|_2)}, \frac{2}{\theta}\right\}.$$

Define

$$l = \operatorname{asinh}(\operatorname{atanh}(a)), \quad r = \operatorname{asinh}(\operatorname{atanh}(b))$$

where

$$a = -1 + \frac{\varepsilon\theta}{2\|A-I\|_2}, \quad b = 1 - \frac{\varepsilon\theta}{2\|A-I\|_2\|A^{-1}\|_2},$$

then, it holds that

$$\frac{\left\|\log(A) - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\|_2}{\|\log(A)\|_2} \leq \epsilon$$

Proof. It is true that $l < r$ because $l < r$ is equivalent to $a < b$ and

$$\begin{aligned} b - a &= 1 - \frac{\varepsilon\theta}{2\|A-I\|_2\|A^{-1}\|_2} - \left(-1 + \frac{\varepsilon\theta}{2\|A-I\|_2}\right) \\ &= 2 - \frac{\varepsilon\theta(1+\|A^{-1}\|_2)}{2\|A-I\|_2\|A^{-1}\|_2} \\ &> 2 - \frac{\theta(1+\|A^{-1}\|_2)}{2\|A-I\|_2\|A^{-1}\|_2}\frac{4\|A-I\|_2\|A^{-1}\|_2}{\theta(1+\|A^{-1}\|_2)} = 0. \end{aligned}$$

In addition it follows that

$$\begin{aligned} a &= -1 + \frac{\varepsilon\theta}{2\|A-I\|_2} < -1 + \frac{\theta}{2\|A-I\|_2}\frac{2}{\theta} = -1 + \frac{1}{\|A-I\|_2}, \\ b &= 1 - \frac{\varepsilon\theta}{2\|A-I\|_2\|A^{-1}\|_2} > 1 - \frac{\theta}{2\|A-I\|_2\|A^{-1}\|_2}\frac{2}{\theta} = 1 - \frac{1}{\|A-I\|_2\|A^{-1}\|_2}. \end{aligned}$$

Therefore, we can choose a finite interval $[l, r]$ that satisfies the assumptions of Proposition 3.3. By dividing the inequality (3.17) by $\|\log(A)\|_2 \geq \theta$, we have the following:

$$\begin{aligned} \frac{\left\|\log(A) - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\|}{\|\log(A)\|_2} &\leq \frac{\|A-I\|_2}{\theta}(1+a) + \frac{\|A-I\|_2\|A^{-1}\|_2}{\theta}(1-b) \\ &= \frac{\|A-I\|_2}{\theta}\left(1 - 1 + \frac{\varepsilon\theta}{2\|A-I\|_2}\right) + \frac{\|A-I\|_2\|A^{-1}\|_2}{\theta}\left(1 - 1 + \frac{\varepsilon\theta}{2\|A-I\|_2\|A^{-1}\|_2}\right) \\ &= \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \end{aligned}$$

□

Algorithm 1 Computation of $\log(A)$ based on the m-point DE formula.

Input: $A \in \mathbb{R}^{n\times n}$, $m \in \mathbb{N}$, $\epsilon > 0$ a tolerance for the interval truncation error

1: Set $t(x) = \tanh(\sinh(x))$

2: Set $\tilde{F}_{\mathrm{DE}}(x) = \dfrac{\cosh(x)}{\cosh^2(\sinh(x))}\left[(1+t(x))A + (1-t(x))I\right]^{-1}$

3: Compute $\|A-I\|_2$, $\|A^{-1}\|_2$, $\rho(A)$

4: $\theta = |\log(\rho(A))|$

5: $\alpha = \dfrac{\varepsilon\theta}{2\|A-I\|_2}, \quad \beta = \dfrac{\varepsilon\theta}{2\|A-I\|_2\|A^{-1}\|_2}$

6: $\texttt{atanh\_a} = (\log(\alpha) - \log(2) - \log(1-\alpha/2))/2$

7: $\texttt{atanh\_b} = (-\log(\beta) + \log(2) + \log(1-\beta/2))/2$

8: $l = \operatorname{asinh}(\texttt{atanh\_a}), \quad r = \operatorname{asinh}(\texttt{atanh\_b})$

9: $h = (r-l)/(m-1)$

10: $T = \dfrac{h}{2}(F_{\mathrm{DE}}(l) + F_{\mathrm{DE}}(r)) + h\displaystyle\sum_{i=1}^{m-2} F_{\mathrm{DE}}(l+ih)$

Output: $(A-I)T \approx \log(A)$

3.2.3 Algorithm

In this subsection we show an algorithms based on the results in subsections 2.1 and 2.2. The following Algorithm 1 summarize the computation of $\log(A)$ using the m-point DE formula whose truncation error is smaller than or equal to ϵ.

When ϵ is sufficiently small, accurate computation of $\|I-A\|_2$, $\|A^{-1}\|_2$ and $\rho(A)$ at Step 3 of Algorithm 1 may not be required because the errors that stem from these values have little effect on the accuracy of $\log(A)$. We give more detail in the following paragraph.

Assume that ϵ in Algorithm 1 is sufficiently small and a, b in Step 9 are chosen as

$$a = \frac{\theta\epsilon}{3\|A-I\|_2}, \quad b = 1 - \frac{\theta\epsilon}{3\|A-I\|_2\|A^{-1}\|_2},$$

where θ is a lower bound on $\|\log(A)\|_2$. Let Δ_1 and Δ_2 be the relative errors of $\theta/\|A-I\|_2$ and $1/\|A^{-1}\|_2$ respectively. Then, the computational result of a is equal to

$$\frac{\epsilon}{3}\frac{\theta}{\|A-I\|_2}(1+\Delta_1) = \frac{\theta}{3\|A-I\|_2}\epsilon(1+\Delta_1),$$

and that of b is equal to

$$1 - \frac{\epsilon}{3}\frac{\theta}{\|A-I\|_2}(1+\Delta_1)\frac{1}{\|A^{-1}\|_2}(1+\Delta_2) = 1 - \frac{\theta}{3\|A-I\|_2\|A^{-1}\|_2}\epsilon(1+\Delta_1+\Delta_2+\Delta_1\Delta_2).$$

Therefore, the upper bound on the truncation error ϵ computed by considering the relative errors Δ_1, Δ_2 is almost equal to the upper bound on the truncation error when the tolerance is set as $\epsilon(1+\Delta_1+\Delta_2+\Delta_1\Delta_2)$. For example, when $\Delta_1, \Delta_2 = 10^{-2}$, $\epsilon(1+\Delta_1+\Delta_2+\Delta_1\Delta_2) \approx 1.02\epsilon$, which means that the upper bound on the truncation error changes by approximately 2%. If ϵ is sufficiently small, the effect of these errors will be negligible.

At Step 4, a lower bound on $\|\log(A)\|_2$ is computed based on (3.18). By setting $\theta = \max\{|\log(\rho(A))|, |\log(\rho(A^{-1}))|\}$, a tighter lower bound can be obtained. In particular, when A is a SPD matrix, $|\log(\rho(A^{-1}))|$ can be obtained without additional computation because $\rho(A^{-1}) = \|A^{-1}\|_2$ is already computed in Step 3.

The computational cost of Algorithm 1 for dense A is $(2m+2)n^3 + \mathcal{O}(n^2)$. When A is sparse, evaluating $\log(A)\boldsymbol{b}$ using Algorithm 1 has computational cost $mc_{\text{abscissa}} + c_{\text{mul}} + c_{\text{param}}$, where c_{abscissa} is the computational cost of computing $F_{\text{DE}}(x)\boldsymbol{b}$, c_{mul} is the computational cost of a matrix-vector multiplication, and c_{param} is the computational cost of computing parameters $\rho(A)$, $\|A-I\|_2$, and $\|A^{-1}\|_2$. If the parameters are computed approximately and $F_{\text{DE}}(x)\boldsymbol{b}$ is computed accurately, then c_{param} will be smaller than c_{abscissa}. Therefore, the computational cost will largely depend on m.

In practical situations, one may improve the approximation of the m-point DE formula by adding $m-1$ abscissas to the centers of subintervals $l+ih/2 (i = 1, \dots, m)$. We can estimate the trapezoidal error from the difference between the m-point approximate and $2m-1$-point approximate, i.e., we can use the DE formula as adaptive quadrature. See Appendix A for more details.

3.3 Convergence of the double exponential formula for the logarithms of SPD matrices

It is important to know the convergence speed of quadrature formulas because we can choose the fastest convergence quadrature formula and reduce the computational cost. In [24], the convergence of the Gauss–Jacobi (GJ) quadrature for the integral

$$\int_{-1}^{1} (1-s)^{-\alpha}(1+s)^{\alpha-1}\left[(1+s)A + (1-s)I\right]^{-1} \mathrm{d}s$$

is analyzed. Under the assumption that $\lambda_{\max}\lambda_{\min} = 1$ where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of A respectively, the upper bound on the m-point GJ quadrature is estimated as:

$$\left\| \int_{-1}^{1} (1-s)^{-\alpha}(1+s)^{\alpha-1}\left[(1+s)A + (1-s)I\right]^{-1} \mathrm{d}s - \sum_{i=1}^{m} w_i \left[(1+s_i)A + (1-s_i)I\right]^{-1} \right\| \leq \mathcal{O}(\tau^{-2m}),$$

where w_i and s_i are weights and nodes of the GJ quadrature, and $\tau = \{[\kappa(A)]^{1/4}+1\}/\{[\kappa(A)]^{1/4}-1\}$. In this section, following [24], we estimate the convergence rate of the DE formula (3.4) for SPD matrices. After that, we compare it to that of the Gaussian quadratures.

When A is an SPD matrix, we can reduce the convergence analysis of the DE formula for $\log(A)$ to that of the DE formula for the scalar logarithm because it holds that

$$\left\| \int_{-\infty}^{\infty} F_{\text{DE}}(x)\,\mathrm{d}x - \sum_{i=-\infty}^{\infty} hF_{\text{DE}}(ih) \right\|_2 = \max_{\lambda\in\Lambda(A)} \left| \int_{-\infty}^{\infty} f_{\text{DE}}(x,\lambda)\,\mathrm{d}x - \sum_{i=-\infty}^{\infty} hf_{\text{DE}}(ih,\lambda) \right|,$$

where h is mesh size, $\Lambda(A)$ is the spectrum of A, and

$$f_{\text{DE}}(x,\lambda) = (\lambda-1)\int_{-1}^{1} \frac{\cosh(x)}{\cosh^2(\sinh(x))} \frac{1}{(1+t(x))\lambda + 1 - t(x)}\,\mathrm{d}x\,.$$

In Subsection 3.1, we first estimate the convergence rate of the trapezoidal rule for $f_{\text{DE}}(x,\lambda)$, i.e., the convergence rate of the DE formula for λ^α. In Subsection 3.2, we discuss the convergence rate of the DE formula for A^α. In Subsection 3.3, we compare the convergence speed of the DE formula and that of the Gaussian quadratures.

3.3.1 Convergence of the DE formula for $\log(\lambda)$

In this subsection, we estimate the convergence rate of the trapezoidal rule for

$$f_{\mathrm{DE}}(x,\lambda) = \frac{\cosh(x)}{\cosh^2(\sinh(x))} \frac{\lambda - 1}{(1+\tanh(\sinh(x)))\lambda + (1-\tanh(\sinh(x)))} \tag{3.20}$$

by using Theorem 2.5.

In order to apply Theorem to f_{DE}, we need a strip region in which f_{DE} is analytic. We show such a strip region in the following proposition:

Proposition 3.6. *For $\lambda > 0$, $f_{\mathrm{DE}}(z,\lambda)$ defined in (3.20) is analytic in the strip $\{z' \in \mathbb{C} : |\mathrm{Im}(z')| < d_0(\lambda)\}$ where*

$$d_0(\lambda) = \arcsin\left(\sqrt{\frac{(\log(\lambda))^2 + \pi^2 + 4 - \sqrt{[(\log(\lambda))^2 + \pi^2 + 4]^2 - 16\pi^2}}{8}}\right). \tag{3.21}$$

Proof. In this proof, we show Proposition 3.6 by calculating

$$d_0(\lambda) = \min_{\omega \in \Omega(\lambda)} |\mathrm{Im}(\omega)|, \tag{3.22}$$

where $\Omega(\lambda)$ is the set of the singular points of $f_{\mathrm{DE}}(z,\lambda)$.

First, the singular points of $1/\cosh(\sinh(z))$ nearest to the real axis are $\pm\mathrm{acosh}(k\pi/2) \pm \mathrm{i}\pi/2$ where k is an odd number [2].

Next, we consider the singular points of

$$\frac{1}{(1+\tanh(\sinh(z)))\lambda + (1-\tanh(\sinh(z)))}. \tag{3.23}$$

The singular points of the denominator are the solutions of the equation

$$(1+\tanh(\sinh(z)))\lambda + (1-\tanh(\sinh(z))) = 0. \tag{3.24}$$

From now on, we calculate (3.22) by considering the imaginary parts of the solutions of (3.24). Let $w = \sinh(z)$. Then, we have

$$\begin{aligned}(1+\tanh w)\lambda + (1-\tanh w) = 0 &\iff \tanh w \left(= \frac{\mathrm{e}^w - \mathrm{e}^{-w}}{\mathrm{e}^w + \mathrm{e}^{-w}}\right) = \frac{1+\lambda}{1-\lambda} \\ &\iff 2\lambda \mathrm{e}^{2w} + 2 = 0 \\ &\iff \exp(2\sinh(z)) = -\frac{1}{\lambda}. \end{aligned} \tag{3.25}$$

To consider the imaginary parts of solution of (3.25), let x and y be the real part and the imaginary part of z in (3.25) respectively. Then, the LHS and the RHS of (3.25) can be rewritten as

$$\exp(2\sinh(z)) = \exp(2\sinh(x)\cos(y) + 2\mathrm{i}\cosh(x)\sin(y)), \tag{3.26}$$

$$-\frac{1}{\lambda} = \exp(-\log(\lambda) + \mathrm{i}k\pi) \quad (k = \pm 1, \pm 3, \pm 5, \dots). \tag{3.27}$$

[2] The calculation of singular points of $1/\cosh(\sinh(z))$ is equivalent to solving the equation $\cosh(\sinh(z)) = 0$. By separating real and imaginary parts of $\cosh(\sinh(z))$, we have $\sinh x \cos y = 0$ and $\cosh x \sin y = k\pi/2$ where $x, y \in \mathbb{R}$ and k is any odd number. When $\cos y = 0$, $x = \pm\mathrm{acosh}(k\pi/2)$ is the solution of $\cosh x \sin y = k\pi/2$. When $\sinh x = 0$, there is no solutions because $\sin y \le 1$. Therefore, the singular points nearest to the real axis are $\pm\mathrm{acosh}(k\pi/2) \pm \mathrm{i}\pi/2$.

By comparing the RHS of (3.26) with that of (3.27), we have

$$\begin{cases} 2\sinh(x)\cos(y) = -\log\lambda, \\ 2\cosh(x)\sin(y) = k\pi. \end{cases} \tag{3.28}$$

By squaring the left and right of both equation of (3.28), we obtain

$$\begin{cases} 4\sinh^2(x)\cos^2(y) = (\log\lambda)^2, \\ 4\cosh^2(x)\sin^2(y) = k^2\pi^2. \end{cases} \tag{3.29}$$

By substituting $\cosh^2(x) = 1 + \sinh^2(x)$ and $\cos^2(y) = 1 - \sin^2(x)$ to (3.29), we have

$$\left(4 + \frac{(\log(\lambda))^2}{1 - \sin^2(y)}\right)\sin^2(y) = k^2\pi^2. \tag{3.30}$$

Let us consider the minimum absolute solution of (3.30). The equation (3.30) has just one solution in $(0, \pi/2)$ for each k because $s(y) := [1 + (\log\lambda)^2/(1 - \sin^2 y)]\sin^2 y$ monotonically increases in $(0, \pi/2)$, and $s(y)$ satisfies $s(0) = 0$ and $\lim_{y\to\pi/2} s(y) = \infty$. In addition, the absolute minimum solution of (3.30) lies in $(0, \pi/2)$ because $s(y)$ is an even function. Furthermore, when $k^2 = 1$ we have the minimum solution among all odd number k because $s(y)$ monotonically increases. Therefore, the minimum absolute solution of (3.30) is obtained by solving (3.30) with $k = 1$, and the solution is the form of the RHS of (3.21).

In conclusion, by comparing the singular points of $1/\cosh(\sinh(z))$ with that of (3.23), we have (3.21). □

Next, we make sure that Theorem 2.5 is applicable to f_{DE}. Let $\tilde{D}$ be the strip $\{z' \in \mathbb{C} : |\mathrm{Im}(z')| < d_0(\lambda) - \delta\}$ where δ is a small positive number. Assume that $z = x + \mathrm{i}y \in \tilde{D}$ for avoiding effects of singular points. Then, we have $|f_{\mathrm{DE}}(x + \mathrm{i}y, \lambda)| = \mathcal{O}(\exp(-\mathrm{e}^{|x|}\cos y))$ as $x \longrightarrow \pm\infty$ from some calculations, and therefore $f_{\mathrm{DE}}(z, \lambda) \longrightarrow 0$ as $|z| \longrightarrow \infty$. For more details of the calculations, see Appendix 3.B. In addition, there exists a constant c such that

$$\int_{-\infty}^{\infty} |f_{\mathrm{DE}}(x + \mathrm{i}y)|\,\mathrm{d}x \le c$$

because $|f_{\mathrm{DE}}(z, \lambda)|$ is bounded in $\tilde{D}$ and $f_{\mathrm{DE}}(x + \mathrm{i}y, \lambda)$ decays faster than $\exp(-|x|)$ as $x \longrightarrow \pm\infty$. Therefore, Theorem 2.5 is applicable to f_{DE}, and for sufficiently small h, the convergence of the DE formula for $\log(\lambda)$ is approximately exponential with rate $\exp(-2\pi d_0(\lambda))$.

We have two observations for the convergence. The first one is that the convergence rate of the DE formula for $\log(\lambda)$ is approximately equal to that of the DE formula for $\log(1/\lambda)$ because $d_0(\lambda) = d_0(1/\lambda)$. The second one is that for two positive scalar λ and $\tilde{\lambda}$ with $|\log(\lambda)| < |\log(\tilde{\lambda})|$, the convergence of the DE formula for $\log(\lambda)$ is faster than that of the DE formula for $\log(\tilde{\lambda})$. This is because $\mathrm{d}d_0(\lambda)/\mathrm{d}(\log(\lambda))^2 > 0$ for $\lambda < 1$ and $\mathrm{d}d_0(\lambda)/\mathrm{d}(\log(\lambda))^2 < 0$ for $\lambda > 1$. We plot $d_0(\lambda)$ in Figure 3.2 for reference, and the graph in Fig 3.2 supports our observations.

3.3.2 Convergence of the DE formula for $\log(A)$

In Subsection 3.1, we found that the convergence speed of the DE formula for $\log(\lambda)$ is slow when $|\log(\lambda)|$ is large, i.e., λ is far away from 1. Therefore, for sufficiently small h, it holds that

$$\left\|\log(A) - \sum_{i=-\infty}^{\infty} hF_{\mathrm{DE}}(ih)\right\|_2 \le \max_{\lambda\in\{\lambda_{\max},\lambda_{\min}\}} \left|\log(\lambda) - \sum_{i=-\infty}^{\infty} hf_{\mathrm{DE}}(ih, \lambda)\right|.$$

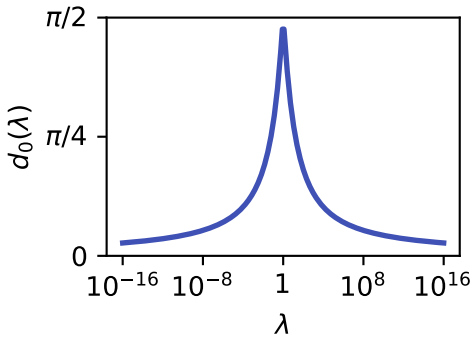


Figure 3.2: The minimum distance between the singular values of $f_{\mathrm{DE}}(z, \lambda)$ and the real axis.

If we assume that $\lambda_{\max}\lambda_{\min} = 1$, i.e., $|\log(\lambda_{\max})| = |\log(\lambda_{\min})|$, we have

$$\left\|\log(A) - \sum_{i=-\infty}^{\infty} hF_{\mathrm{DE}}(ih)\right\|_2 \le \mathcal{O}\left(\exp\left(-\frac{2\pi d_0(\lambda_{\max})}{h}\right)\right) = \mathcal{O}\left(\exp\left(-\frac{2\pi d_0(\sqrt{\kappa(A)})}{h}\right)\right)$$

because $d_0(\lambda_{\max}) = d_0(1/\lambda_{\max}) = d_0(\lambda_{\min})$. Note that the condition $\lambda_{\max}\lambda_{\min} = 1$ can be satisfied by computing $\log(pA)(= \log(A) + \log(p)I)$ where $p = 1/\sqrt{\lambda_{\max}/\lambda_{\min}}$, and can accelerate the convergence of the DE formula for matrices such that $\lambda_{\max}\lambda_{\min}$ is far away from 1.

If we further assume that we have a finite interval $[l, r]$ whose truncation error is smaller than the trapezoidal error, then the trapezoidal error can be represented using the number of abscissas $m = (r - l)/h + 1$, i.e.,

$$\begin{aligned}&\left\|\log(A) - \sum_{i=1}^{m-2} hF_{\mathrm{DE}}(l + ih) - \frac{h}{2}(F_{\mathrm{DE}}(l) + F_{\mathrm{DE}}(r))\right\|_2 \\ &\le \mathcal{O}\left(\exp\left(-\frac{2\pi d_0\left(\sqrt{\kappa(A)}\right)}{r - l}m\right)\right).\end{aligned} \tag{3.31}$$

3.3.3 Comparison between the DE formula and the GL quadrature

In this subsection, we compare convergence speed of the DE formula (DE) with that of the GL quadrature for (3.3) (GL) under the assumption that $\lambda_{\max}\lambda_{\min} = 1$. Since both of quadrature formulas converges exponentially, we can write the upper bounds on these quadrature formulas in the form of $\mathcal{O}(\exp(-\phi m))$, where $\phi = \phi(\kappa(A))$ and m is the number of abscissas. The convergence speed of DE is $\phi_{\mathrm{DE}}(\kappa) = 2\pi d_0(\kappa^{1/2})/(r - l)$, and the convergence speed of GL is $\phi_{\mathrm{GL}}(\kappa) = 2\log((\kappa^{1/4} + 1)/(\kappa^{1/4} - 1))$. We numerically calculated the coefficient ϕ_{DE} and ϕ_{GL} for various κ. We set $\kappa_2(A) \in [10^{-16}, 10^{16}]$, and plotted results in Figure 3.3. In this calculation, we obtain a finite interval for DE by using Algorithm 1 with $\varepsilon = 2^{-53} \approx 1.1 \times 10^{-16}$.

Figure 3.3 shows that the convergence speed of DE is higher than that of GL when $\kappa(A)$ is larger than about 3.7×10^3. Conversely, the convergence speed of DE is slower than that of GL for small $\kappa(A)$. By using Fig. 3.3, we can select the fastest convergence quadrature formula. For example, when we compute $\log(A)$ for $\kappa(A) = 10^9$, DE converges about ten times faster than GL.

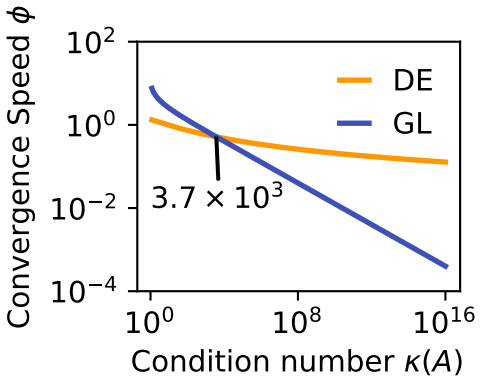


Figure 3.3: Comparison of the convergence speed of the GL quadrature and the DE formula.

3.4 Preconditioning of quadrature formulas based on the relation $\log(A) = \log(AP) - \log(P)$

In the previous subsection, we analyzed the convergence speed of the GL quadrature and the DE formula for the logarithm of an SPD matrix A. In addition, we found that both of quadrature formulas converge slowly when $\kappa(A)$ is large. In this study, we consider another approach to reducing the number of abscissas. For some matrix P, the following relation holds

$$\log(A) = \log(AP) - \log(P), \tag{3.32}$$

and a condition for the relation (3.32) to hold is known as follows:

Theorem 3.7 ([39, Thm. 11.3]). *Suppose $A, P \in \mathbb{C}^{n\times n}$ both have no eigenvalues on $\mathbb{R}^-$ and that $AP = PA$. If for every eigenvalue λ_j of A and the corresponding eigenvalue μ_j of P,*

$$|\arg \lambda_j + \arg \mu_j| < \pi,$$

then $\log(AP) = \log(A) + \log(P)$.

Thus, we can compute $\log(A)$ via the RHS of (3.32) after choosing P. This procedure can be regarded as preconditioning, and it may reduce the number of abscissas. In this section, for an SPD matrix A, we propose a choice of P and discuss the effect of the preconditioning.

In order to reduce the number of abscissas for computing $\log(A)$, P must be selected so that $\kappa(P) < \kappa(A)$ and $\kappa(AP) < \kappa(A)$. Here, we observe that when P is a $[0, 1]$ type rational function of A whose eigenvalue is positive, i.e., $P \in \Phi$, $\Phi := \{(A+pI)^{-1} : p > 0\}$, then $\kappa(AP)$ and $\kappa(P)$ are smaller than $\kappa(A)$. We can check the observation as follows. Let $P = (A+pI)^{-1}$, and $\lambda_{\max}$ and $\lambda_{\min}$ be the maximum and minimum eigenvalues of A respectively. Then, the maximum and minimum eigenvalues of P are $1/(\lambda_{\min} + p)$ and $1/(\lambda_{\max} + p)$ respectively. Hence, it follows that

$$\kappa(P) - \kappa(A) = \frac{\lambda_{\max} + p}{\lambda_{\min} + p} - \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{(\lambda_{\min} - \lambda_{\max})p}{\lambda_{\min}(\lambda_{\min} + p)} < 0.$$

Similarly, the maximum and minimum eigenvalues of AP are $\lambda_{\max}/(\lambda_{\max} + p)$ and $\lambda_{\min}/(\lambda_{\min} + p)$ respectively, and it follows that

$$\kappa(AP) - \kappa(A) = \frac{\lambda_{\max}}{\lambda_{\max} + p}\frac{\lambda_{\min} + p}{\lambda_{\min}} - \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{\lambda_{\max}(\lambda_{\min} - \lambda_{\max})}{\lambda_{\min}(\lambda_{\max} + p)} < 0.$$

Algorithm 2 Splitting Precondition for the logarithm of SPD matrices.

Input: $A \in \mathbb{C}^{n\times n}$: a SPD matrix.

1: Compute the maximum eigenvalue of A, $\lambda_{\max}$, and the minimum eigenvalue of A, $\lambda_{\min}$.

2: Set $\tilde{F}_{\mathrm{DE}}(x) = \dfrac{\cosh(x)}{\cosh^2(\sinh(x))}\left[(1+t(x))A + (1-t(x))I\right]^{-1}$

3: $\tilde{A} = A/\sqrt{\lambda_{\max}\lambda_{\min}}, \quad \kappa = \lambda_{\max}/\lambda_{\min}$

4: $p_{\tilde{A}+I} = 1/\sqrt{2+\sqrt{\kappa}+1/\sqrt{\kappa}}, \quad p_{\tilde{A}(\tilde{A}+I)^{-1}} = (1+\sqrt{\kappa})\kappa^{-1/4}$

5: Compute $\log(\tilde{A}+I) = \log\left(p_{\tilde{A}+I}(\tilde{A}+I)\right) - \log(p_{\tilde{A}+I})I$

6: Compute $\log\left((\tilde{A}+I)^{-1}\tilde{A}\right) = \log\left(p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A}\right) - \log(p_{\tilde{A}(\tilde{A}+I)^{-1}})I$

7: $X = \log\left((\tilde{A}+I)^{-1}\tilde{A}\right) + \log(\tilde{A}+I) + \log(\lambda_{\max}\lambda_{\min})I/2$

Output $X \approx \log(A)$

In addition, $P \in \Phi$ holds the condition of Theorem 3.7. Thus, we can use $P \in \Phi$ for the preconditioning.

In order to accelerate the convergence of a quadrature formula, it is natural to choose P so that both of the condition numbers $\kappa(P)$ and $\kappa(AP)$ are small, i.e., $P = \arg\min_{P'\in\Phi}\{\kappa(P'), \kappa(AP')\}$. In fact, we can calculate the minimizer explicitly. We show the minimizer in the following proposition.

Proposition 3.8. *Let $A \in \mathbb{C}^{n\times n}$ be an SPD matrix and $\Phi = \{(A+pI)^{-1} : p > 0\}$. Then, it holds that*

$$\left(A + \sqrt{\lambda_{\max}\lambda_{\min}}I\right)^{-1} = \arg\min_{P\in\Phi} \max\{\kappa(P), \kappa(AP)\}. \tag{3.33}$$

Proof. Let $P(p) = (A+pI)^{-1}$. Then, $\kappa(P(p))$ monotonically decreases while $\kappa(AP(p))$ monotonically increases. Therefore, we obtain the minimizer $p = \arg\min_{p>0} \max\{\kappa(P(p)), \kappa(AP(p))\}$ by solving the equation $\kappa(P(p)) = \kappa(AP(p))$, and we have (3.33). □

As a summary of the above discussions, we show the preconditioning for numerical quadrature for $\log(A)$ in Algorithm 2. At Step 6 of Algorithm 2, we compute $\log(p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A})$. By rewriting the integral representation, we can avoid computing $(\tilde{A}+I)^{-1}$ directory:

$$\log(p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A}) = (p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A} - I)\int_{-1}^{1}\left[(1+t)p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A} + (1-t)I\right]^{-1}\mathrm{d}t$$

$$= ((p_{\tilde{A}(\tilde{A}+I)^{-1}} - 1)\tilde{A} - I)\int_{-1}^{1}\left[(1+t)p_{\tilde{A}(\tilde{A}+I)^{-1}}\tilde{A} + (1-t)(\tilde{A}+I)\right]^{-1}\mathrm{d}t\,.$$

Since we have the convergence rate of the GL quadrature and the DE formula in the previous section, we can estimate the effect of the preconditioning. Let $\phi(\kappa)$ be the convergence speed of a quadrature formula for computing $\log(A)$ where $\kappa = \kappa(A)$, i.e., the error of a m-point quadrature formula is $\mathcal{O}(-\exp(\phi(\kappa)m))$. Then, the convergence speed of the quadrature formula for $\log(AP)$ and $\log(P)$ is $\phi(\kappa^{1/2})$. Since we compute $\log(AP)+\log(P)(=\log(A))$, the error of m-point quadrature formula after preconditioning is $\mathcal{O}(-\exp(\phi(\kappa^{1/2})m/2))$. Now we compare the following four convergence speeds: the GL quadrature $\phi_{\mathrm{GL}}(\kappa)$, the GL quadrature with preconditioning $\phi_{\mathrm{GL}}(\kappa^{1/2})/2$, the DE formula $\phi_{\mathrm{DE}}(\kappa)$, and the DE formula with preconditioning $\phi_{\mathrm{DE}}(\kappa^{1/2})/2$. The comparison of the convergence speed for various κ is shown in Figure 3.4.

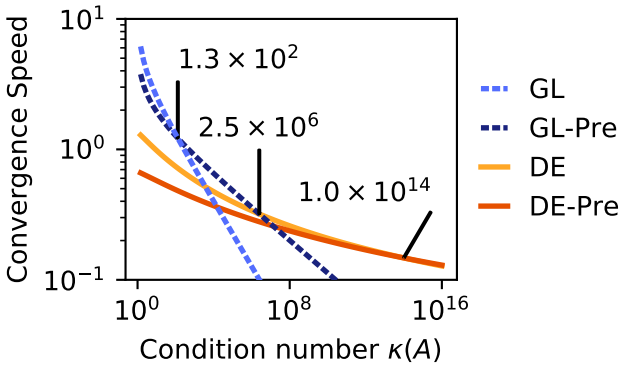


Figure 3.4: Comparison of the convergence speed of the GL quadrature and the DE formula when considering the preconditioning $\log(A) = \log(AP) - \log(P)$ where P is defined in (3.33). GL and DE denote the convergence speed $\phi_{\mathrm{GL}}(\kappa)$ and $\phi_{\mathrm{DE}}(\kappa)$ respectively as in Fig. 3.3. GL-Pre and DE-Pre denote that the convergence speeds of the quadrature formulas with preconditioning respectively, i.e., $\phi_{\mathrm{GL}}(\kappa^{1/2})/2$ and $\phi_{\mathrm{DE}}(\kappa^{1/2})/2$.

Table 3.1: Test matrices used for numerical experiments of Chapter 3

Test	Matrix	n	$\kappa(A)$	Properties
1	`ex5` [19]	27	6.6×10^7	SPD
	`pores_1` [19]	30	1.8×10^6	Nonsymmetric
2, 4	`nos4` [19]	100	1.6×10^3	SPD
	`bcsstk04` [19]	132	2.3×10^6	SPD
	`lund_b` [19]	147	3.0×10^4	SPD
3	`SPD_well`	100	1.0×10^2	SPD
	`SPD_ill`	100	1.0×10^7	SPD
	`NS_well`	100	1.0×10^2	Nonsymmetric
	`NS_ill`	100	1.0×10^7	Nonsymmetric

From the results in Fig. 3.4, we can choose the fastest quadrature formula when computing $\log(A)$. When $\kappa(A) \lesssim 1.3 \times 10^2$, the GL quadrature without preconditioning is the fastest among the four quadrature formulas. Similarly, the GL quadrature with preconditioning is the fastest when $1.3 \times 10^2 \lesssim \kappa(A) \lesssim 2.5 \times 10^6$, the DE formula without preconditioning is the fastest when $2.5 \times 10^6 \lesssim \kappa(A) \lesssim 1.0 \times 10^{14}$, and the DE formula with preconditioning is the fastest when $\kappa(A) \gtrsim 1.0 \times 10^{14}$

3.5 Numerical experiments

The numerical experiments were carried out using Julia 1.2 on an AMD EPYC 7501 CPU (2 GHz, 32 cores) with 64 GB RAM. We used the IEEE double precision arithmetic unless otherwise stated. When we consider arbitrary precision arithmetic, we implement the code by using the `BigFloat` type of Julia whose machine epsilon is about 1.7×10^{-77}. We computed abscissas and weights in the GJ quadrature with `QuadGK.jl` [3]. Our test matrices are shown in 3.1.

[3] `https://github.com/JuliaMath/QuadGK.jl`

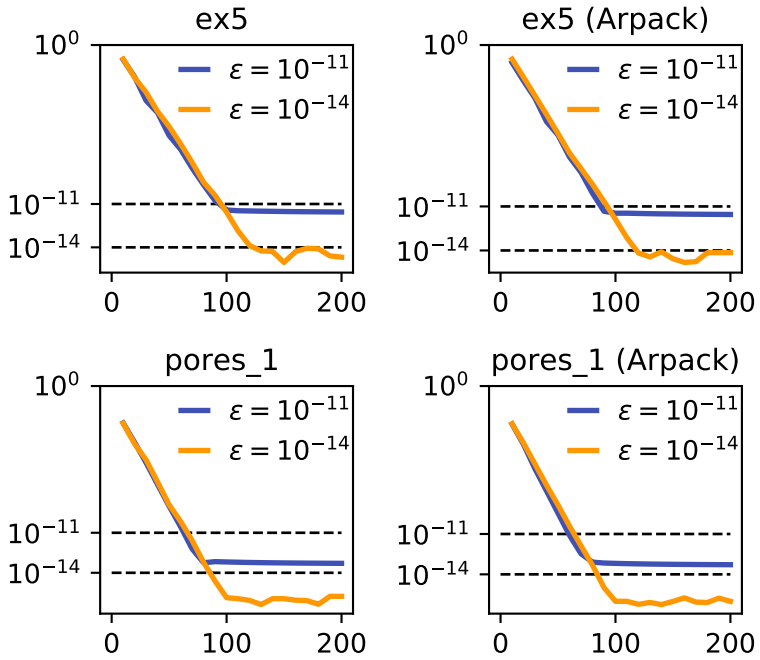


Figure 3.5: Convergence histories of the DE formula for Test 1. We consider on the x-axis the number of abscissas and on the y-axis the relative error in 2-norm $\|\log(\tilde{A}) - X\|_2 / \|\log(\tilde{A})\|_2$. Each title represents the test matrix. If we write (Arpack) in the title, it means that the parameters in Algorithm 1 are computed with `Arpack.jl`. Otherwise, the parameters are computed with `eigvals` and `svdvals`.

3.5.1 Test 1: checking the appropriateness of the determined finite interval

In this test, we check that Algorithm 1 selects a finite interval appropriately. First, we scaled the matrix as $\tilde{A} = A/\sqrt{\sigma_{\max}\sigma_{\min}}$ where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and the minimum singular value of A respectively. Then, we computed the reference solution $\log(\tilde{A})$ by using the inverse scaling and squaring algorithm [25, Algorithm 5.1] with arbitrary precision [4]. After that, with the tolerance $\varepsilon = 10^{-11}, 10^{-14}$, we computed $\log(\tilde{A})$ based on Algorithm 1. At Step 3 of Algorithm 1, we compute the parameters $\|A - I\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$ in two ways. The one are computed by using `eigvals` and `svdvals` in Julia which are function computing all the eigenvalues and the singular values respectively. The others are computed by using `Arpack.jl` [5], which compute eigenvalues and singular values with restarted Lanczos or Arnoldi iterations. We computed parameters $\|A - I\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$ with three-digit accuracy. Furthermore, at the steps 9 and 10 of Algorithm 1, we used arbitrary precision arithmetic in order to avoid the error stems from matrix inversion.

The convergence histories of the DE formula is illustrated in Figure 3.5. Figure 3.5 shows that we obtain approximations with the desired accuracy in all cases. Therefore, Algorithm 1 gives us appropriate finite intervals. In addition, the approximates are accurate regardless of the accuracy of the parameters $\|A\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$. This results support the discussions in Subsection 3.2.3, and we can compute these parameters approximately.

For `pores_1`, Algorithm 1 seemed to give a wide finite interval. It may because the lower bound on $\|\log(\tilde{A})\|$ was not sharp. Indeed, $|\log(\rho(\tilde{A}))| \approx 7.0 \times 10^0$ while $\|\log(\tilde{A})\| \approx 3.8 \times 10^2$.

[4] We computed $\log(-\tilde{A})$ for `pores_1` alternatively because the eigenvalues of `pores_1` lie in the set $\mathbb{C}\backslash[0, \infty)$.
[5] `https://github.com/JuliaLinearAlgebra/Arpack.jl`

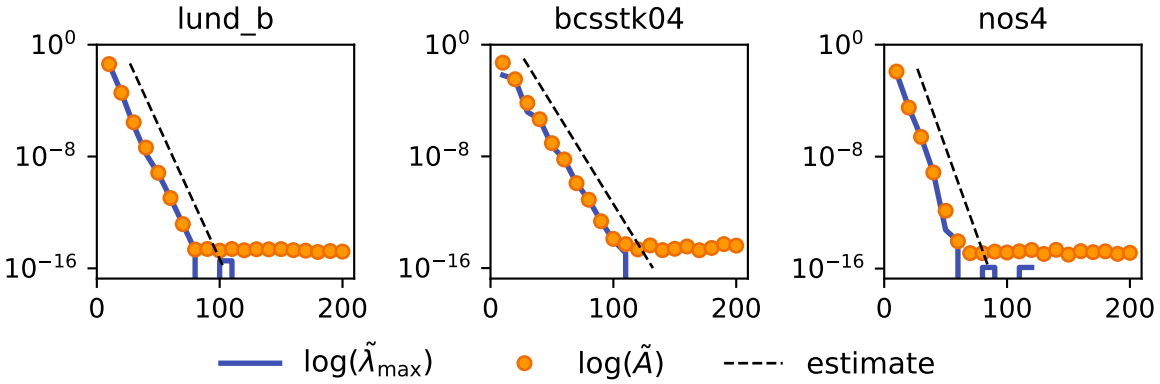


Figure 3.6: Convergence histories of the DE formula for Test 2. We consider on the x-axis the number of abscissas and on the y-axis the relative error in 2-norm $\|\log(\tilde{A})-X\|_2/\|\log(\tilde{A})\|_2$ (or $|\log(\tilde{\lambda})|/|\log(\tilde{\lambda})|$). Each title represents the test matrix.

3.5.2 Test 2: checking the convergence

In this test, we check the convergence speed of the DE formula for the fractional power of SPD matrices. First, we scaled the test matrices $\tilde{A} = A/\sqrt{\lambda_{\max}\lambda_{\min}}$ and compute the reference solution $\log(\tilde{A})$ as done in Test 1. Then, we compute $\log(\tilde{A})$ by using the DE formula (Algorithm 1) with $\varepsilon = 2^{-53} \approx 1.1 \times 10^{-16}$ in the double precision environment. After that, we compute the square root of the maximum eigenvalue of $\tilde{A}$, $\tilde{\lambda}_{\max}$, by using the DE formula whose integral interval is the same as that of the DE formula for $\log(\tilde{A})$. We plotted the convergence histories of the DE formula for $\log(\tilde{A})$ and $\tilde{\lambda}_{\max}$ in Figure 3.6. In addition, we plotted the estimates of the convergence rate (3.31) in Fig. 3.6.

Figure 3.6 shows the convergence of the DE formula for $\log(\tilde{A})$ is almost the same as that of the DE formula for $\log(\tilde{\lambda}_{\max})$. Furthermore, the convergence rates of the quadrature formulas are approximately equal to the estimates. These results support our convergence analysis in Section 3.3.

3.5.3 Test 3: comparison between the DE formula and the GL quadrature

In this test, we compare the convergence of the DE formula with that of the GJ quadrature. For this test, we generate four test matrices as follows:

1. We generated two SPD matrices (`SPD_well` and `SPD_ill`).
   (a) We generated an orthogonal matrix Q by QR decomposition of a random 100×100 matrix.
   (b) We generated a diagonal matrix whose diagonal elements were from the geometric sequence: $\{d_i\}_{i=1,\ldots,100}$ where $d_1 = \kappa^{-1/2}$ and $d_{100} = \kappa^{1/2}$ for $\kappa = 10^2$.
   (c) $A_{\text{SPD\_well}} = QDQ^{\mathsf{T}}$
   (d) We repeated Step (b) and (c) with the setting $\kappa = 10^7$, and generate $A_{\text{SPD\_ill}}$.
2. We generated two nonsymmetric matrices (`NS_well` and `NS_ill`).
   (a) We generated a random 100×100 matrix R.

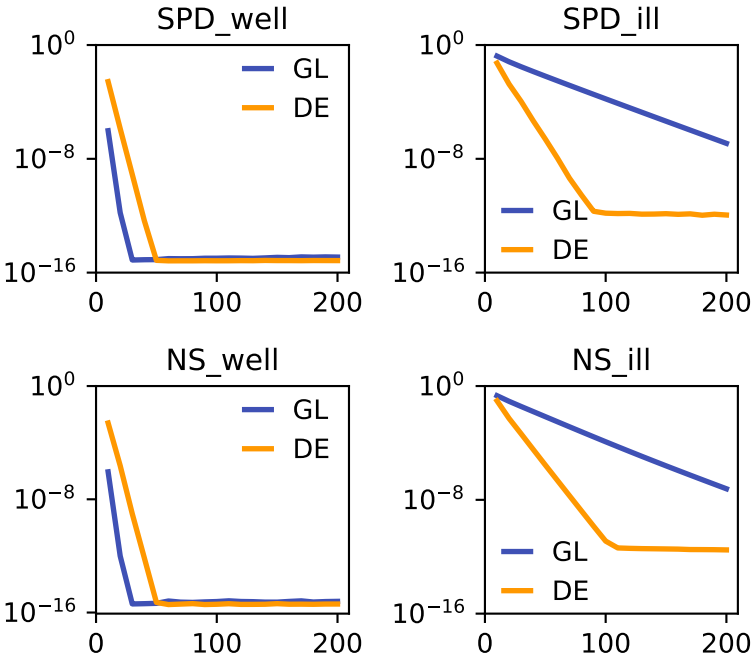


Figure 3.7: Convergence histories of the GL quadrature and the DE formula for Test 3. We consider on the x-axis the number of abscissas, and on the y-axis the relative error in F-norm $\|\log(\tilde{A}) - X\|_F / \|\log(\tilde{A})\|_F$. Each title represents the test matrix.

(b) We generated $A_{\mathrm{NS\_well}} = \exp(cR)$ with $c = c_{\mathrm{well}} \approx 8.535 \times 10^{-2}$ so that $\kappa(A_{\mathrm{NS\_well}}) \approx 10^2$. The matrix exponential is computed by using exp function is Julia[6].

(c) We scaled $A_{\mathrm{NS\_well}}$ so that the multiplication of the maximum singular value and the minimum singular value becomes 1.

(d) We repeated Step (b) and (c) with the setting $c = c_{\mathrm{ill}} \approx 2.998 \times 10^{-1}$, and generated $A_{\mathrm{NS\_ill}}$.

Then, we computed $\log(A)$ and plotted convergence histories in Figure 3.7. Figure 3.7 shows that when $\kappa(A)$ is large (`SPD_ill` and `NS_ill`), DE converged faster than GL. These results support that the DE formula is effective when the convergence of GJ quadrature is slow.

3.5.4 Test 4: checking the effect of the preconditioning

In this test, we check the effect of the preconditioning with test matrices used in Test 2. We compute $\log(\tilde{A})$ by using the GL quadrature and the DE formula with and without preconditioning (Algorithm 9). The convergence histories are illustrated in Figure 3.8. In Fig. 3.8, we consider the x-axis the total number of abscissas to compute $\log(\tilde{A})$ in order to compare results with and without preconditioning.

Since the condition numbers of test matrices are larger then 1.3×10^2 and smaller than 2.5×10^6, the GL quadrature with preconditioning converges the fastest among the four quadrature formulas. These results support the conclusion of Subsection 3.4. On the other hand, the preconditioned quadrature formulas are slightly more inaccurate than quadrature formulas without preconditioning. We leave the discussions on the accuracy for future work.

[6] In Julia, `exp(A)` computes the matrix exponential while `exp.(A)` performs element-wise computation.

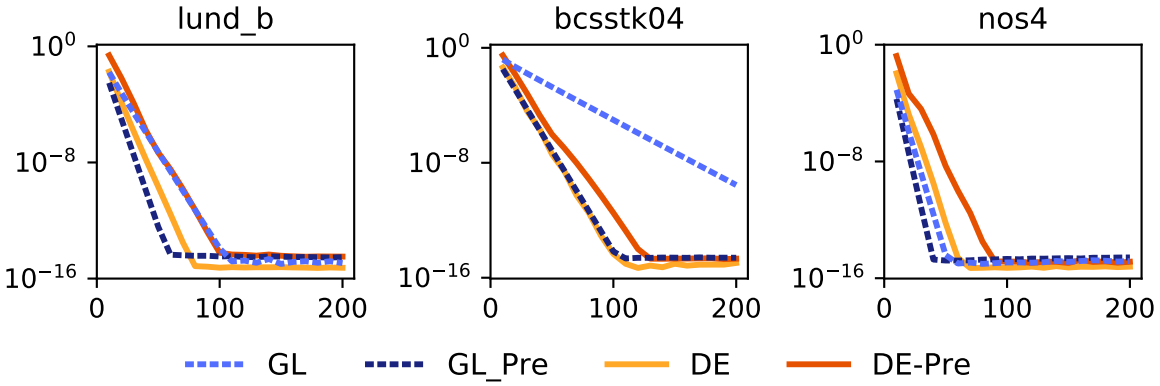


Figure 3.8: Convergence histories of the GL quadrature and the DE formula for Test 4. GL-Pre and DE-Pre mean the convergence histories of the GL quadrature and the DE formula with the preconditioning respectively. We consider on the x-axis the total number of abscissas.In other words, when considering the preconditioning, we used $m/2$-point quadrature formulas for computing $\log(p_{\tilde{A}+I}(\tilde{A}+I))$ and $\log(p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A})$. We consider on the y-axis the relative error in F-norm $\|\log(\tilde{A}) - X\|_F/\|\log(\tilde{A})\|_F$. Each title represents the test matrix.

3.6 Conclusion

In this chapter, we focused on the DE formula as a new choice for the numerical quadrature formula of $\log(A)$. In order to utilize the DE formula, we proposed a method for selecting an appropriate finite interval based on error analysis, and we proposed an algorithm for practical computation. By focusing on SPD matrices, we analyzed the convergence rate of the DE formula. Then, we found that the DE formula converges faster than the GL quadrature formula when $\kappa(A) \gtrsim 3.7 \times 10^3$ while the GL quadrature converges faster when $\kappa(A) \lesssim 3.7 \times 10^3$. In addition, we considered preconditioning based on the relation $\log(A) = \log(AP) - \log(P)$, and show a choice of P when A is an SPD matrix. Estimates of convergence speed showed that the preconditioning accelerate the GL quadrature when $1.3{\times}10^2 \lesssim \kappa(A) \lesssim 2.5 \times 10^6$ and accelerate the DE formula when $\kappa(A) \gtrsim 1.0 \times 10^{14}$.

We carried out four numerical experiments, and all of the results supported our discussions in previous sections. The first one showed that our interval selection method gave us a finite interval such that the truncation error is smaller than a given tolerance, and the second one showed that our estimation on convergence rate is appropriate. The third one showed that the DE formula is effective when the GL quadratures converge slow, and the fourth one showed the preconditioning accelerated the GL quadrature for the test matrices.

Our future work will focus on four problems. The first one is to obtain a sharper lower bound on $\log(A)$ than $|\log(\rho(A))|$ to avoid the overestimation of the truncation error. The second one is a further error analysis of quadrature formulas. For example, one may consider the convergence rate for nonsymmetric matrices and the coefficient of the error which affects the convergence when the number of abscissas is small. The third one is developing more efficient quadrature formulas. For example, one may modify the variable transformation for the DE formula. The fourth one is the verification of the practical performance of the presented algorithms when applied to large (sparse) matrices from current research problems.

Appendix 3.A Two algorithms for $\log(A)$ based on the DE formula

In this section, we show two algorithms which compute $\log(A)$ adding abscissas automatically. The first algorithm can be used for general matrices while the other algorithms is specialized to SPD matrices. After proposing algorithms, we show some numerical results.

Once an appropriate finite interval is obtained, the accuracy of the DE formula can be improved with the following procedure. Let m be the number of abscissas, $h = (r-l)/(m+1)$ be the mesh size, and $T(h)$ be the trapezoidal rule for the mesh size h:

$$T(h) := \frac{h}{2}\left(F_{\mathrm{DE}}(l) + F_{\mathrm{DE}}(r)\right) + h\sum_{i=1}^{m-2} F_{\mathrm{DE}}(l+ih).$$

Then, $T(h/2)$ can be computed using $T(h)$:

$$T\left(\frac{h}{2}\right) = \frac{1}{2}T(h) + \frac{h}{2}\sum_{i=1}^{m-1} F_{\mathrm{DE}}\left(l + (2i-1)\frac{h}{2}\right).$$

In addition, we can apply the following estimation of the trapezoidal error for a sufficiently small value of h using [10, Eq. (4.3)]:

$$\left\| \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x - T\left(\frac{h}{2}\right)\right\| \approx \frac{1}{3}\left\| T(h) - T\left(\frac{h}{2}\right)\right\|. \tag{3.34}$$

We show an adaptive quadrature algorithm based on (3.34) in Algorithm 3. The computational cost of Algorithm 3 for dense A is $(2m_{k+1}+2)n^3 + \mathcal{O}(n^2) = [2^{k+1}(m_0-1)+4]n^3 + \mathcal{O}(n^2)$, and the computational cost of $\log(A)\boldsymbol{b}$ with sparse A is $[2^k(m_0-1)+1]c_{\mathrm{abscissa}} + c_{\mathrm{mul}} + c_{\mathrm{param}}$.

Algorithm 3 Adaptive quadrature algorithm for $\log(A)$ based on the DE formula.

Input A, $m_0 \le 2$, $\varepsilon > 0$. ▷ For example, one can set $m_0 = 16$.

Set $t(x) = \tanh(\sinh(x))$

Set $\tilde{F}_{\mathrm{DE}}(x) = \dfrac{\cosh(x)}{\cosh^2(\sinh(x))}\left[(1+t(x))A + (1-t(x))I\right]^{-1}$

Compute $\|A-I\|_2$, $\|A^{-1}\|_2$, $\rho(A)$

Compute l, r, θ using Algorithm 1 from step 4 to step 8

$h_0 = (r-l)/(m_0-1)$

$T_0 = \frac{h_0}{2}\tilde{F}_{\mathrm{DE}}(l) + \frac{h_0}{2}\tilde{F}_{\mathrm{DE}}(r) + h_0\sum_{i=1}^{m-2}\tilde{F}_{\mathrm{DE}}(l+ih)$

for $k = 0, 1, 2, \ldots$ until convergence **do**

  $h_{k+1} = h_k/2$

  $T_{k+1} = \frac{1}{2}T_k + h_{k+1}\sum_{i=1}^{m_k-1}\tilde{F}_{\mathrm{DE}}(l+(2i-1)h_{k+1})$

  $m_{k+1} = 2m_k - 1$

  if $\frac{1}{3}\|T_{k+1} - T_k\|/\theta \le \varepsilon$ **then**

    $T = T_{k+1}$

    break

  end if

end for

Output $(A-I)T \approx \log(A)$

Algorithm 4 Computing the logarithm of an SPD matrix with the appropriate number of abscissas.

Input: an SPD matrix A, $\varepsilon > 0$

1: Compute $\lambda_{\max}, \lambda_{\min}$, the maximum and the minimum eigenvalues of A respectively.

2: $\tilde{A} = (\lambda_{\max}\lambda_{\min})^{-1/2}A$, and $\tilde{\lambda}_{\max} = (\lambda_{\max}\lambda_{\min})^{-1/2}\lambda_{\max}$.

3: Compute l, r by using from Step 4 to Step 8 of Algorithm 1 for $\tilde{A}$.

4: Find the minimum positive integer m satisfying

$$\frac{\left|\log(\tilde{\lambda}_{\max}) - h(m)\left[\frac{1}{2}f_{\mathrm{DE}}(l, \tilde{\lambda}_{\max}) + \sum_{j=2}^{m-1} f_{\mathrm{DE}}(l + jh(m), \tilde{\lambda}_{\max}) + \frac{1}{2}f_{\mathrm{DE}}(r, \tilde{\lambda}_{\max})\right]\right|}{\log(\tilde{\lambda}_{\max})} \leq \varepsilon$$

where $h(m) = (r - l)/(m - 1)$.

5: Compute $\log(\tilde{A})$ by using the m-point DE formula (Step 10 of Algorithm 1)

Output $\log(\tilde{A}) + \log(\lambda_{\max}\lambda_{\min})/2 \approx \log(A)$

Table 3.2: Test matrices for checking Algorithm 3.

Matrix	size	$\kappa_2(A)$	Structure
`parter` [69]	10	2.4×10^{0}	Nonsymmetric
`frank` [69]	10	2.9×10^{7}	Nonsymmetric
`vand` [69]	10	3.1×10^{12}	Nonsymmetric
`bcsstk02` [19]	66	4.3×10^{3}	SPD
`bcsstk03` [19]	112	6.8×10^{6}	SPD
`ck104` [19]	104	5.5×10^{3}	Nonsymmetric

When A is an SPD matrix, we no longer need to apply the adaptive quadrature algorithm because we can predict the number of abscissas to get an approximation with the required accuracy. Indeed, since the convergence of the DE formula for $\log(A)$ is the same as that of the DE formula for $\log(\lambda_{\max})$ with $\lambda_{\max}\lambda_{\min} = 1$, we can obtain the appropriate number of abscissa after computing $\log(\lambda_{\max})$ [7]. We show the procedures in Algorithm 4. The computational cost of Algorithm 4 is the same as that of Algorithm 1 because the computational cost of Step 4 is negligible.

At the end of this section, we show numerical results of which is used for checking Algorithm 3. We skip checking Algorithm 4 because we already check the convergence of the DE formula for $\log(A)$ and $\log(\lambda_{\max})$ at the second experiment in Subsection 3.5.2. Our test matrices are presented in Table 3.2, and the procedures for this test is as follows:

1. We scaled test matrices: $\tilde{A} = (10/\rho(A))A$ because some matrices have too large value to deal with.

2. We computed $\log(\tilde{A})$ by Algorithm 3. We set $m_0 = 16$ and $\varepsilon \in \{10^{-8}, 10^{-11}\}$. At Step 2 of Algorithm 3, where Algorithm 1 was used, we computed spectrum radius and 2-norm of matrices using `eigvals` and `svdvals`. We stopped the computation once the number of integrand evaluations reached 1921.

We show the results in Table 3.3. Table 3.3 shows that Algorithm 3 successfully computed the logarithm with the desired accuracy within 2000 integrand evaluations for all test

[7] The idea predicting the number of abscissas using the maximum eigenvalue is pointed out in [28] for computing the matrix geometric mean based on the Gauss–Jacobi quadrature.

Table 3.3: Checking Algorithm 3, in terms of the number of integrand evaluations (in bold) and the relative error when the algorithm stopped (in parentheses). The notation "– (–)" means that the algorithms did not stop before the number of integrand evaluations reached the limit.

ε	10^{-8}	10^{-11}
`parter`	**61** (2.6×10^{-9})	**121** (2.3×10^{-12})
`frank`	**481** (1.0×10^{-12})	**1921** (2.1×10^{-13})
`vand`	– (–)	– (–)
`bcsstk02`	**121** (2.8×10^{-9})	**121** (3.1×10^{-12})
`bcsstk03`	**241** (1.4×10^{-9})	**241** (1.5×10^{-12})
`ck104`	**121** (6.7×10^{-10})	**121** (7.4×10^{-13})

matrices, except `vand`. For `vand`, the stopping criterion may be too strict.

Appendix 3.B On the behavior of $|f_{\mathrm{DE}}(x+\mathrm{i}y, \lambda)|$ as $x \longrightarrow \pm\infty$

In this section, under the assumption that $|y| < d_0(\lambda) - \delta$ for sufficiently small positive δ, we check that $|f_{\mathrm{DE}}(x + \mathrm{i}y, \lambda)| = \mathcal{O}(\exp(-\mathrm{e}^{|x|} \cos y))$ as $x \longrightarrow \pm\infty$. The definitions of f_{DE} and d_0 are in (3.20) and (3.21) respectively. As a preparation of discussions, we recall the real part and the imaginary part values of hyperbolic functions:

$$\begin{aligned}
\sinh(x + \mathrm{i}y) &= \sinh x \cos y + \mathrm{i} \cosh x \sin y, \\
\cosh(x + \mathrm{i}y) &= \cosh x \cos y + \mathrm{i} \sinh x \sin y, \\
\tanh(x + \mathrm{i}y) &= \frac{\sinh(2x) + \mathrm{i} \sin(2y)}{\cosh(2x) + \cos(2y)}.
\end{aligned}$$

First we consider $|\cosh(x + \mathrm{i}y)|$ and $|\cosh^2(\sinh(x + \mathrm{i}y))|$. It holds that

$$\begin{aligned}
|\cosh(x + \mathrm{i}y)|^2 &= \cos^2 y \cosh^2 x + \sin^2 y \sinh^2 x \\
&= \cos^2 y \cosh^2 x + (1 - \cos^2 y)(\cosh^2 x - 1) = \cosh^2 x + \cos^2 y - 1, \\
|\cosh(x + \mathrm{i}y)| &= \sqrt{\cosh^2 x + \cos^2 y - 1},
\end{aligned}$$

and we have $|\cosh(x+\mathrm{i}y)| = \mathcal{O}(\exp(|x|))$ as $x \longrightarrow \pm\infty$. In addition, by defining $a = \sinh x \cos y$ and $b = \cosh x \sin y$, it follows that

$$|\cosh^2(\sinh(x + \mathrm{i}y)| = |\cosh(a + \mathrm{i}b)|^2 = \cosh^2 a + \cos^2 b - 1,$$

and we obtain $|\cosh^2(\sinh(x + \mathrm{i}y))| = \mathcal{O}(\exp(\mathrm{e}^{|x|} \cos y))$ as $x \longrightarrow \pm\infty$.

Next, we consider the absolute value of

$$\frac{\lambda - 1}{(1 + \tanh(\sinh(x + \mathrm{i}y)))\lambda + (1 - \tanh(\sinh(x + \mathrm{i}y)))} = \frac{1}{\tanh(\sinh(x + \mathrm{i}y)) + \frac{\lambda+1}{\lambda-1}}.$$

Here, it holds that

$$\tanh(\sinh(x + \mathrm{i}y)) = \tanh(a + \mathrm{i}b) = \frac{\sinh(2a) + \mathrm{i} \sin(2b)}{\cosh(2a) + \cos(2b)},$$

and we have

$$\left|\tanh(\sinh(x+\mathrm{i}y))+\frac{\lambda+1}{\lambda-1}\right| = \sqrt{\left(\frac{\sinh(2a)}{\cosh(2a)+\cos(2b)}+\frac{\lambda+1}{\lambda-1}\right)^2+\left(\frac{\sin(2b)}{\cosh(2a)+\cos(2b)}\right)^2}.$$

Therefore, $|\tanh(\sinh(x+\mathrm{i}y))|+(\lambda+1)/(\lambda-1)\longrightarrow 2\lambda/(\lambda-1)$ as $x\longrightarrow\infty$, and $|\tanh(\sinh(x+\mathrm{i}y))|+(\lambda+1)/(\lambda-1)\longrightarrow 2/(\lambda-1)$ as $x\longrightarrow-\infty$. Finally, we have

$$\begin{aligned}|f_{\mathrm{DE}}(x+\mathrm{i}y,\lambda)| &= \frac{|\cosh(x+\mathrm{i}y)|}{|\cosh^2(\sinh(x+\mathrm{i}y))|}\frac{1}{\left|\tanh(\sinh(x+\mathrm{i}y))+\frac{\lambda+1}{\lambda-1}\right|}\\ &= \mathcal{O}(\exp(-\mathrm{e}^{|x|}\cos y))\end{aligned}$$

as $x\longrightarrow\pm\infty$.

Chapter 4

Quadrature-based algorithms for A^α

4.1 Introduction

In this chapter, we consider quadrature-based algorithms for the matrix fractional power. The matrix fractional power is defined as $A^\alpha := \exp(\alpha \log(A))$ for $\alpha \in \mathbb{R}$, where exp is the matrix exponential and log is the principal matrix logarithm. Therefore, A^α exists uniquely when $\Lambda(A) \subset \mathbb{C} \setminus (-\infty, 0]$.

We first give a brief overview of applications and computational methods of the matrix fractional power. The matrix fractional power arises in many situations of scientific computing such as lattice QCD calculation [6], fractional reaction-diffusion equations [9], and optimization in machine learning [32]. Computational methods for matrix fractional power can be classified roughly into two groups depending on the purpose. The first group is algorithms for computing A^α itself. This group includes the Schur–Padé algorithm [42], the Schur recurrence algorithm [47], and quadrature-based algorithms [10, 24, 34]. In addition, when α is reciprocal of a natural number, i.e., $\alpha = 1/p$ for $p \in \mathbb{N}$, we can use algorithms specialized to the matrix pth root such as Newton's method [45]. The other group is algorithms specialized to compute $A^\alpha \boldsymbol{b}$ for a vector $\boldsymbol{b}$ ($\boldsymbol{b} \in \mathbb{R}^n$). This group includes Krylov subspace methods [1, 29] and quadrature-based algorithms.

Quadrature-based algorithms have been developed on the basis of some integral representations of the matrix fractional power. A well-known integral representation is

$$A^\alpha = \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^\alpha (zI - A)^{-1} \, \mathrm{d}z \tag{4.1}$$

where Γ is a closed contour in $\mathbb{C} \setminus (-\infty, 0]$ that encloses the spectrum of A [41]. Based on (4.1), a quadrature formula specialized to SPD matrices was proposed in [34]. When $\alpha \in (0, 1)$, A^α can also be represented as the integral on the real axis [41]

$$A^\alpha = \int_0^\infty F(t) \, \mathrm{d}t \quad \left(F(t) := \frac{\sin(\alpha\pi)}{\alpha\pi} A (t^{1/\alpha} I + A)^{-1} \right). \tag{4.2}$$

In terms of the computation of A^α, the integral representation (4.2) is free from selecting an integral path, and therefore we can deal with nonsymmetric matrices as well as SPD matrices. On the other hand, we have to compute the half infinite integral. In this chapter, we consider methods of computing (4.2) efficiently.

In previous studies, the Gaussian quadratures were applied to (4.2) after some variable transformations. For example, the transformation $t(u) = (1 + u)/(1 - u)$, which is considered

for computing the matrix pth root in [10], gives the following integral representation:

$$A^{\alpha} = \frac{2\sin(\alpha\pi)}{\pi\alpha} A \int_{-1}^{1} (1-u)^{1/\alpha-2} \left[(1+u)^{1/\alpha} I + (1-u)^{1/\alpha} A\right]^{-1} u. \tag{4.3}$$

For computing (4.3), the Gauss–Legendre (GL) quadrature or the Gauss–Jacobi (GJ) quadrature is suitable. As another example, the transformation $t(v) = (1-v)^{\alpha}/(1+v)^{\alpha}$ gives the following integral representation:

$$A^{\alpha} = \frac{2\sin(\alpha\pi)}{\pi} A \int_{-1}^{1} (1-v)^{\alpha-1}(1+v)^{-\alpha} \left[(1-v)I + (1+v)A\right]^{-1} v. \tag{4.4}$$

A similar integral representation to (4.4) is considered in [24] for computing matrix weighted geometric mean. For computing (4.4), the GJ quadrature is suitable.

Since these methods are based on the Gaussian quadrature, the convergence will be fast when the integrand is close to low degree polynomials. Conversely, when the integrand is far away from polynomials, the convergence may be slow. For example, when α is not reciprocal of natural numbers, the convergence of the Gaussian quadrature for (4.3) will be slow because $(1+u)^{1/\alpha}$ and $(1-u)^{1/\alpha}$ are not analytic on $u = -1$ and $u = 1$ respectively. As another example, when the condition number of A is large, the convergence of the GJ quadrature for (4.4) will be slow because the integrand value in (4.4) changes rapidly at the endpoint of the interval[1]. Therefore, when α is not reciprocal of natural numbers and $\kappa(A)$ is large, the convergence of the Gaussian quadrature for both of (4.3) and (4.4) will be slow.

To avoid the slow convergence we first consider the double exponential (DE) formula [60] as a new approach to computing (4.2). In order to utilize the DE formula, we propose a method to chose a finite interval based on truncation error estimation, as done in Chapter 3. Then, We summarize the interval selection method as an algorithm for practical situations. After that, focusing on SPD matrices, we estimate the convergence rate of the DE formula, and we compare it to that of the Gaussian quadratures. Besides, focusing on an SPD matrix A, we consider the preconditioning for reducing the number of abscissas based on the relation

$$A^{\alpha} = (AP)^{\alpha} P^{-\alpha}. \tag{4.5}$$

The relation (4.5) holds when $\log(A) = \log(AP) - \log(P)$ because $A^{\alpha} = \exp(\alpha \log(A))$. Thus, as done in Section 3.4, we choose P from a rational function of A to accelerate the convergence of numerical quadrature.

The remainder of this chapter as follows. In Section 4.2, we propose a method of selecting an appropriate interval for the DE formula based on a truncation error analysis. We also show an algorithm for practical use. In Section 4.3, we analyze the convergence rate of the DE formula focusing on SPD matrices, and we compare it with that of the GL quadrature. In Section 4.4, we show a choice of P for the preconditioning. In Section 4.5, we show numerical results. In Section 4.6, we conclude this chapter.

4.2 Algorithms for A^{α} based on the DE formula

In this section, we propose a method of selecting a finite interval for the DE formula by estimating the interval truncation, and we show an algorithm for practical computation.

[1]Convergence analysis of the GJ quadrature for (4.4) and (4.3) where A is an SPD matrix is performed in [24]. When the condition number of A is large and $\alpha = 1/2$, the GJ quadrature for (4.3) converges faster than the GJ quadrature for (4.4) under the assumption that the product of the maximum eigenvalue and the minimum eigenvalue is 1.

Before considering the truncation error, let us apply the DE transformations to (4.2). By substituting $t(x) = \exp(\alpha \sinh(x))$ to (4.2), we have

$$A^\alpha = \int_{-\infty}^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x\,, \tag{4.6}$$

where

$$F_{\mathrm{DE}}(x) := \frac{\sin(\alpha\pi)}{\pi} \exp(\alpha \sinh(x)) \cosh(x) A \left[\exp(\sinh(x))I + A\right]^{-1}.$$

In Subsection 4.2.1, we estimate an upper bound on the interval truncation error for a given finite interval $[l, r]$:

$$\left\| A^\alpha - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \tag{4.7}$$

as done in Chapter 3. In Subsection 4.2.2, we propose a method of setting the interval $[l, r]$ so that

$$\frac{\left\| A^\alpha - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\|_2}{\|A^\alpha\|_2} \le \varepsilon$$

for a given tolerance ε. In Subsection 4.2.3, we summarize discussions in subsections 2.1 and 2.2 as an algorithm.

4.2.1 Estimation of the interval truncation error

In this subsection, we obtain an upper bound on the interval truncation error (4.7). The error (4.7) can be rewritten as

$$\left\| A^\alpha - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| = \left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x + \int_r^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\|. \tag{4.8}$$

Then, by considering the triangle inequality of the right-hand side (RHS) of (4.8), we have

$$\left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x + \int_r^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \le \left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| + \left\| \int_r^{\infty} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\|. \tag{4.9}$$

Therefore, we can obtain an upper bound on (4.7) by estimating the RHS of (4.9).

In [10], an upper bound on

$$\left\| \int_b^{\infty} (t^{1/\alpha} I + A)^{-1}\,\mathrm{d}t \right\|$$

is derived as the following lemma:

Lemma 4.1 ([10]). *Let A have no eigenvalues on the closed negative real axis. If $b \ge (2\|A\|)^\alpha$, then*

$$A^\alpha = \frac{\sin(\alpha\pi)}{\alpha\pi} A \left(\int_0^b (t^{1/\alpha} I + A)^{-1}\,\mathrm{d}t + \int_b^{\infty} (t^{1/\alpha} I + A)^{-1}\,\mathrm{d}t \right)$$

where

$$\left\| \int_b^{\infty} (t^{1/\alpha} I + A)^{-1}\,\mathrm{d}t \right\| \le \frac{2 b^{1-1/\alpha}}{1/\alpha - 1}. \tag{4.10}$$

By using the inequality (4.10), we obtain the following:

$$\left\| \int_r^\infty F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| = \left\| \frac{\sin(\alpha\pi)}{\alpha\pi} A \int_b^\infty (t^{1/\alpha} I + A)^{-1}\,\mathrm{d}t \right\| \le \frac{\sin(\alpha\pi)\|A\|}{\pi(1-\alpha)} 2b^{1-1/\alpha} \tag{4.11}$$

where $r = \operatorname{asinh}(\log(b)/\alpha)$.

The main tool for estimating the truncation error in Lemma 4.1 is the Neumann series expansion. Based on the idea, we first derive a new upper bound on the second term in the RHS of (4.9) whose rate is the same as that of upper bound (4.11), but whose constant factor is smaller than that of (4.11). After that, we obtain an upper bound on the first term in the RHS of (4.9).

We show a new upper bound on the second term in the RHS of (4.9) in the following lemma:

Lemma 4.2. *Let A have no eigenvalues on the closed negative real axis and $\alpha \in (0,1)$. Then, for $b \ge (2\|A\|)^\alpha$, we have*

$$\left\| \int_b^\infty F(t)\,\mathrm{d}t \right\| \le \frac{\sin(\alpha\pi)(3-2\alpha)\|A\|}{\pi(1-\alpha)(2-\alpha)} b^{1-1/\alpha} \tag{4.12}$$

where $F(t)$ is defined in (4.2)*. Therefore,*

$$\left\| \int_r^\infty F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \le \frac{\sin(\alpha\pi)(3-2\alpha)\|A\|}{\pi(1-\alpha)(2-\alpha)} b^{1-1/\alpha} z \tag{4.13}$$

where $r = \operatorname{asinh}(\log(b)/\alpha)$.

Proof. The outline of this proof is similar to that of Lemma 4.1. For all $t \in [b,\infty)$ where $b \ge (2\|A\|)^\alpha$, it holds that $t \ge (2\|A\|)^\alpha$, and hence $\|t^{-1/\alpha}A\| \le 1/2$ (< 1). By applying the Neumann series expansion to $(t^{1/\alpha}I + A)^{-1}$, we get

$$(t^{1/\alpha}I + A)^{-1} = t^{-1/\alpha}[I - (-t^{-1/\alpha}A)]^{-1} = t^{-1/\alpha} \sum_{k=0}^\infty (-1)^k t^{-k/\alpha} A^k.$$

Therefore, the integral can be rewritten as follows:

$$\begin{aligned}
\int_b^\infty (t^{1/\alpha}I + A)^{-1}\,\mathrm{d}t &= \int_b^\infty \left[t^{-1/\alpha} \sum_{k=0}^\infty (-1)^k t^{-k/\alpha} A^k \right] \mathrm{d}t \\
&= \sum_{k=0}^\infty \left[(-1)^k A^k \int_b^\infty t^{-(k+1)/\alpha}\,\mathrm{d}t \right] \\
&= \sum_{k=0}^\infty (-1)^k A^k \frac{\alpha}{k+1-\alpha} b^{1-(k+1)/\alpha} \\
&= \frac{\alpha}{1-\alpha} b^{1-1/\alpha} I + \sum_{k=1}^\infty (-1)^k A^k \frac{\alpha}{k+1-\alpha} b^{1-(k+1)/\alpha}.
\end{aligned}$$

By using the triangle inequality and consistency of the norm, it follows

$$\begin{aligned}
\left\| \int_b^\infty F(t)\,\mathrm{d}t \right\| &= \left\| \frac{\sin(\alpha\pi)}{\alpha\pi} A \int_b^\infty (t^{1/\alpha} I + A^{-1})\,\mathrm{d}t \right\| \\
&= \left\| \frac{\sin(\alpha\pi)}{\alpha\pi} \left[\frac{\alpha}{1-\alpha} b^{1-1/\alpha} A + A \sum_{k=1}^{\infty} (-1)^k A^k \frac{\alpha}{k+1-\alpha} b^{1-(k+1)/\alpha} \right] \right\| \\
&\le \frac{\sin(\alpha\pi)}{\alpha\pi} \left[\left\| \frac{\alpha}{1-\alpha} b^{1-1/\alpha} A \right\| + \|A\| \left\| \sum_{k=1}^{\infty} (-1)^k A^k \frac{\alpha}{k+1-\alpha} b^{1-(k+1)/\alpha} \right\| \right] \\
&\le \frac{\sin(\alpha\pi)}{\pi} \|A\| \left[\frac{1}{1-\alpha} b^{1-1/\alpha} + \sum_{k=1}^{\infty} \|A\|^k \frac{1}{k+1-\alpha} b^{1-(k+1)/\alpha} \right]. \qquad (4.14)
\end{aligned}$$

For the second term in the bracket on the RHS of (4.14), we have:

$$\begin{aligned}
\sum_{k=1}^{\infty} \|A\|^k \frac{1}{k+1-\alpha} b^{1-(k+1)/\alpha} &= b^{1-1/\alpha} \sum_{k=1}^{\infty} \frac{1}{k+1-\alpha} \|b^{-1/\alpha} A\|^k < b^{1-1/\alpha} \sum_{k=1}^{\infty} \frac{1}{2-\alpha} \left(\frac{1}{2}\right)^k \\
&= \frac{1}{2-\alpha} b^{1-1/\alpha}. \qquad (4.15)
\end{aligned}$$

We have (4.12) by substituting (4.15) in (4.14), and we obtain (4.13) by applying $t(x) = \exp(\alpha \sinh(x))$ to (4.12). □

Remark 4.3. *The RHS of* (4.12) *is smaller than that of* (4.11) *because*

$$\begin{aligned}
\frac{2\sin(\alpha\pi)\|A\|}{\pi(1-\alpha)} b^{1-1/\alpha} - \frac{\sin(\alpha\pi)(3-2\alpha)\|A\|}{\pi(1-\alpha)(2-\alpha)} b^{1-1/\alpha} &= \frac{\sin(\alpha\pi)\|A\|}{\pi(1-\alpha)} b^{1-1/\alpha} \left(2 - \frac{3-2\alpha}{2-\alpha}\right) \\
&= \frac{\sin(\alpha\pi)\|A\|}{\pi(1-\alpha)} b^{1-1/\alpha} \frac{1}{2-\alpha}. \\
&> 0.
\end{aligned}$$

Next, we obtain an upper bound on the first term in the RHS of (4.9) as the following lemma.

Lemma 4.4. *Let A have no eigenvalues on the closed negative real axis and $\alpha \in (0,1)$. For $a \le (2\|A^{-1}\|)^{-\alpha}$, we have*

$$\left\| \int_0^a F(t)\,\mathrm{d}t \right\| \le \frac{\sin(\alpha\pi)[\alpha + (1+\alpha)\|I\|]}{\alpha\pi(1+\alpha)} a, \qquad (4.16)$$

and therefore

$$\left\| \int_{-\infty}^{l} F_{\mathrm{DE}}(x)\,\mathrm{d}x \right\| \le \frac{\sin(\alpha\pi)[\alpha + (1+\alpha)\|I\|]}{\alpha\pi(1+\alpha)} a \qquad (4.17)$$

where $l = \operatorname{asinh}(\log(a)/\alpha)$.

Proof. The outline of this proof is similar to that of Lemma 4.2. For all $t \in [0, a]$ where $a \le (2\|A^{-1}\|)^{-\alpha}$, it holds that $t \le (2\|A^{-1}\|)^{-\alpha}$, and therefore $\|t^{1/\alpha} A^{-1}\| \le 1/2$ (< 1). By applying the Neumann series expansion to $(t^{1/\alpha} I + A)^{-1}$ we get the following:

$$(t^{1/\alpha} I + A)^{-1} = A^{-1}[I - (-t^{1/\alpha} A^{-1})]^{-1} = A^{-1} \sum_{k=0}^{\infty} (-1)^k A^{-k} t^{k/\alpha}.$$

Therefore, the integral can be rewritten as follows:

$$\begin{aligned} A\int_0^a (t^{1/\alpha}I + A)^{-1}\,\mathrm{d}t &= A\int_0^a A^{-1}\left[\sum_{k=0}^{\infty}(-1)^k A^{-k}t^{k/\alpha}\right]\mathrm{d}t \\ &= \sum_{k=0}^{\infty}(-1)^k A^{-k}\left[\int_0^a t^{k/\alpha}\,\mathrm{d}t\right] \\ &= \sum_{k=0}^{\infty}(-1)^k A^{-k}\left(\frac{1}{k/\alpha+1}a^{k/\alpha+1}\right) \\ &= aI + \sum_{k=1}^{\infty}(-1)^k A^{-k}\left(\frac{1}{k/\alpha+1}a^{k/\alpha+1}\right). \end{aligned}$$

Thus, we get the following:

$$\begin{aligned} \left\|\int_0^a F(t)\,\mathrm{d}t\right\| &= \left\|\frac{\sin(\alpha\pi)}{\alpha\pi}A\int_0^a (t^{1/\alpha}I + A)^{-1}\,\mathrm{d}t\right\| \\ &= \frac{\sin(\alpha\pi)}{\alpha\pi}\left\|aI + \sum_{k=1}^{\infty}(-1)^k A^{-k}\left(\frac{1}{k/\alpha+1}a^{k/\alpha+1}\right)\right\| \\ &\le \frac{\sin(\alpha\pi)}{\alpha\pi}\left[a\|I\| + \sum_{k=1}^{\infty}\left\|A^{-1}\right\|^k\left(\frac{1}{k/\alpha+1}a^{k/\alpha+1}\right)\right]. \end{aligned} \tag{4.18}$$

Furthermore, we have:

$$\sum_{k=1}^{\infty}\left\|A^{-1}\right\|^k\left(\frac{1}{k/\alpha+1}a^{k/\alpha+1}\right) = \sum_{k=1}^{\infty}\frac{\alpha a}{k+\alpha}\left\|a^{1/\alpha}A^{-1}\right\|^k \le \sum_{k=1}^{\infty}\frac{\alpha a}{1+\alpha}\left(\frac{1}{2}\right)^k = \frac{\alpha}{1+\alpha}a \tag{4.19}$$

because $a \le (2\|A^{-1}\|)^{-\alpha}$. We get (4.16) by substituting (4.19) in (4.18), and we obtain (4.17) by applying $t(x) = \exp(\alpha\sinh(x))$ to (4.16). □

Finally, from the estimates (4.17) and (4.13), we obtain the upper bound on the interval truncation error in the following proposition:

Lemma 4.5. *Let A have no eigenvalues on the closed negative real axis and $\alpha \in (0,1)$. For given $[l,r]$, let $a = \exp(\alpha\sinh(l))$ and $b = \exp(\alpha\sinh(r))$. Then, if $a \le (2\|A^{-1}\|)^{-\alpha}$ and $b \ge (2\|A\|)^{\alpha}$, it holds that*

$$\left\|A^{\alpha} - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\| \le \frac{\sin(\alpha\pi)[\alpha + (1+\alpha)\|I\|]}{\alpha\pi(1+\alpha)}a + \frac{\sin(\alpha\pi)(3-2\alpha)\|A\|}{\pi(1-\alpha)(2-\alpha)}b^{1-1/\alpha}. \tag{4.20}$$

4.2.2 Setting the integral interval

To develop an algorithm for computing A^{α} based on the DE formula, we need to determine the appropriate finite integration interval, $[l,r]$ in advance. In the following proposition, we show a method for selecting the finite interval so that the relative truncation error in 2-norm is smaller than or equal to the given tolerance, $\varepsilon > 0$.

Proposition 4.6. *For given $\varepsilon > 0$, let*

$$l = \operatorname{asinh}(\log(a)/\alpha), \quad r = \operatorname{asinh}(\log(b)/\alpha),$$

where

$$a = \min\left\{\frac{\varepsilon}{2}\frac{\pi\alpha(1+\alpha)\rho(A)^\alpha}{\sin(\alpha\pi)(1+2\alpha)}, (2\|A^{-1}\|_2)^{-\alpha}\right\},$$
$$b = \max\left\{\left[\frac{\varepsilon}{2}\frac{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha}{\sin(\alpha\pi)(3-2\alpha)\|A\|_2}\right]^{\alpha/(\alpha-1)}, (2\|A\|_2)^\alpha\right\}.$$

Then, it holds that

$$\frac{\left\|A^\alpha - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\|_2}{\|A^\alpha\|_2} \le \varepsilon. \tag{4.21}$$

Proof. From the definition of a and b, it is true that $a \le (2\|A^{-1}\|_2)^{-\alpha}$ and $b \ge (2\|A\|_2)^\alpha$. Therefore, the interval $[l, r]$ satisfies the assumptions for Lemma 4.5. By dividing the inequality (4.20) with $\rho(A)^\alpha \le \|A^\alpha\|_2$ [2], it follows:

$$\frac{\left\|A^\alpha - \int_l^r F_{\mathrm{DE}}(x)\,\mathrm{d}x\right\|_2}{\|A^\alpha\|_2} \le \frac{\sin(\alpha\pi)(1+2\alpha)}{\alpha\pi(1+\alpha)\rho(A)^\alpha}a + \frac{\sin(\alpha\pi)(3-2\alpha)\|A\|_2}{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha}b^{1-1/\alpha}. \tag{4.22}$$

From the definition of a and b, we have

$$\frac{\sin(\alpha\pi)(1+2\alpha)}{\alpha\pi(1+\alpha)\rho(A)^\alpha}a \le \frac{\varepsilon}{2}, \quad \frac{\sin(\alpha\pi)(3-2\alpha)\|A\|_2}{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha}b^{1-1/\alpha} \le \frac{\varepsilon}{2}. \tag{4.23}$$

We obtain (4.21) by substituting (4.23) in (4.22). □

4.2.3 Algorithm

As a summary of subsections 4.2.1 and 4.2.2, we show an algorithm which is designed to compute A^α using the m-point DE formula on a finite interval whose interval truncation error is smaller than or equal to the given tolerance ε. We show the algorithm in Algorithm 5.

Algorithm 5 Computing A^α based on the m-point DE formula

Input A, $\alpha \in (0,1)$, $\varepsilon > 0$, The number of nodes $m \ge 2$.
1: Set $\tilde{F}_{\mathrm{DE}}(x) := \exp(\alpha\sinh(x))\cosh(x)[\exp(\sinh(x))I + A]^{-1}$
2: Compute $\|A\|_2$, $\|A^{-1}\|_2$ and the spectral radius $\rho(A)$
3: $a = \min\left\{\frac{\pi\alpha(1+\alpha)\rho(A)^\alpha}{2\sin(\alpha\pi)(1+2\alpha)}\varepsilon, (2\|A^{-1}\|_2)^{-\alpha}\right\}$
4: $b = \max\left\{\left[\frac{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha}{2\sin(\alpha\pi)(3-2\alpha)\|A\|_2}\varepsilon\right]^{\alpha/(\alpha-1)}, (2\|A\|_2)^\alpha\right\}$
5: $l = \mathrm{asinh}(\log(a)/\alpha)$
6: $r = \mathrm{asinh}(\log(b)/\alpha)$
7: $h = (r-l)/(m-1)$
8: $T = \frac{h}{2}\tilde{F}_{\mathrm{DE}}(l) + \frac{h}{2}\tilde{F}_{\mathrm{DE}}(r) + \sum_{j=1}^{m-2} h\tilde{F}_{\mathrm{DE}}(l+jh)$
Output $\frac{\sin(\alpha\pi)}{\pi}AT \approx A^\alpha$.

[2] For the maximum absolute eigenvalue of A, $\lambda_{\max}$, it holds that $|\lambda^\alpha| = |\lambda|^\alpha$. Hence, we have $\rho(A)^\alpha = \rho(A^\alpha)$ $(\le \|A^\alpha\|_2)$

When ε is sufficiently small, accurate computation of $\|A\|_2$, $\|A^{-1}\|_2$ and $\rho(A)$ in Step 2 of Algorithm 5 may not be required because the errors that stem from these values have little effect on the accuracy of A^α. We give more detail in the following paragraph.

Assume that ε in Algorithm 5 is sufficiently small, and that a in Step 3 and b in Step 4 are chosen as

$$a = \frac{\pi\alpha(1+\alpha)\rho(A)^\alpha}{2\sin(\alpha\pi)(1+2\alpha)}\varepsilon, \quad b = \left[\frac{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha}{2\sin(\alpha\pi)(3-2\alpha)\|A\|_2}\varepsilon\right]^{\alpha/(\alpha-1)}.$$

Let Δ_ρ and $\Delta_{\|A\|_2}$ be the relative errors of $\rho(A)^\alpha$ and $\|A\|_2$ respectively. Then, the computational result of a is equal to

$$\frac{\pi\alpha(1+\alpha)\rho(A)^\alpha(1+\Delta_\rho)}{2\sin(\alpha\pi)(1+2\alpha)}\varepsilon = \frac{\pi\alpha(1+\alpha)\rho(A)^\alpha}{2\sin(\alpha\pi)(1+2\alpha)}\varepsilon(1+\Delta_\rho)$$

and that of b can be estimated as

$$\begin{aligned}&\left[\frac{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha(1+\Delta_\rho)}{2\sin(\alpha\pi)(3-2\alpha)\|A\|_2(1+\Delta_{\|A\|_2})}\varepsilon\right]^{\alpha/(\alpha-1)}\\ &\quad\approx \left[\frac{\pi(1-\alpha)(2-\alpha)\rho(A)^\alpha}{2\sin(\alpha\pi)(3-2\alpha)\|A\|_2}\varepsilon(1+\Delta_\rho-\Delta_{\|A\|_2})\right]^{\alpha/(\alpha-1)}\end{aligned}$$

by considering the first-order approximation of $(1+\Delta_\rho)/(1+\Delta_{\|A\|_2})$. Therefore, the upper bound on the truncation error ε computed with the relative errors $\Delta_\rho, \Delta_{\|A\|_2}$ will be at most the upper bound on the truncation error when the tolerance is set as $\varepsilon(1+|\Delta_\rho|+|\Delta_{\|A\|_2}|)$. For example, when $\Delta_\rho, \Delta_{\|A\|_2} = 10^{-2}$, $\varepsilon(1+|\Delta_\rho|+|\Delta_{\|A\|_2}|) = 1.02\varepsilon$, which means that the upper bound on the truncation error changes by approximately 2%. If ε is sufficiently small, the effect of these errors will be negligible.

The computational cost of Algorithm 1 for dense A is $(2m+2)n^3 + \mathcal{O}(n^2)$. When A is sparse, evaluating $A^\alpha\boldsymbol{b}$ using Algorithm 1 has computational cost $mc_{\text{abscissa}} + c_{\text{mul}} + c_{\text{param}}$, where c_{abscissa} is the computational cost of computing $\tilde{F}_{\text{DE}}(x)\boldsymbol{b}$, c_{mul} is the computational cost of a matrix-vector multiplication, and c_{param} is the computational cost of computing parameters $\|A\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$. If the parameters are computed approximately and $\tilde{F}_{\text{DE}}(x)\boldsymbol{b}$ is computed accurately, then c_{param} will be smaller than c_{abscissa}. Therefore, the computational cost will largely depend on m.

In practical situations, one may improve the accuracy of the m-point DE formula by adding $m-1$ abscissas to the centers of subintervals $l + ih/2$ $(i = 1, \dots, m)$. We can estimate the trapezoidal error from the difference between the m-point approximation and $2m-1$ point approximation, i.e., we can use the DE formula as adaptive quadrature. See Appendix 4.A for more details.

4.3 Convergence analysis of the DE formula

It is important to know the convergence speed of quadrature formulas because we can choose the fastest convergence quadrature formula and reduce the computational cost. In [28], convergence rates of the GJ quadrature for (4.3) and (4.4) are calculated under the assumption that A is an SPD matrix. In this section, following [28], we estimate the convergence rate of the DE formula (4.6) for SPD matrices. After that, we compare it to that of the Gaussian quadratures.

When A is an SPD matrix, we can reduce the convergence analysis of the DE formula for A^α to that of the DE formula for the scalar fractional power because it holds that

$$\left\| \int_{-\infty}^{\infty} F_{\mathrm{DE}}(x)x - \sum_{i=-\infty}^{\infty} hF_{\mathrm{DE}}(ih) \right\|_2 = \max_{\lambda\in\Lambda(A)} \left| \int_{-\infty}^{\infty} f_{\mathrm{DE}}(x,\lambda)x - \sum_{i=-\infty}^{\infty} hf_{\mathrm{DE}}(ih,\lambda) \right|,$$

where h is mesh size, $\Lambda(A)$ is the spectrum of A, and

$$f_{\mathrm{DE}}(x,\lambda) = \frac{\sin(\alpha\pi)\lambda}{\pi} \frac{\exp(\alpha\sinh(x))\cosh(x)}{\exp(\sinh(x)) + \lambda}. \tag{4.24}$$

In Subsection 4.3.1, we first estimate the convergence rate of the trapezoidal rule for $f_{\mathrm{DE}}(x,\lambda)$, i.e., the convergence rate of the DE formula for λ^α. In Subsection 4.3.2, we discuss the convergence rate of the DE formula for A^α. In Subsection 4.3.3, we compare the convergence speed of the DE formula and that of the Gaussian quadratures.

4.3.1 Convergence of the DE formula for λ^α

In this subsection, we estimate the convergence rate of the trapezoidal rule for $f_{\mathrm{DE}}(x,\lambda)$ based on Theorem 2.5. In order to apply Theorem 2.5 to f_{DE}, we need a strip region in which f_{DE} is analytic. We show such a strip region in the following proposition:

Proposition 4.7. *For $\lambda > 0$, $f_{\mathrm{DE}}(z,\lambda)$ defined in (4.24) is analytic in the strip $\{z' \in \mathbb{C} : |\Im(z')| < d_0(\lambda)\}$ where*

$$d_0(\lambda) = \arcsin\left(\sqrt{\frac{(\log(\lambda))^2 + \pi^2 + 1 - \sqrt{[(\log(\lambda))^2 + \pi^2 + 1]^2 - 4\pi^2}}{2}} \right). \tag{4.25}$$

Proof. In this proof, as done in Proposition 3.6, we show Proposition 4.7 by calculating

$$d_0(\lambda) = \min_{\omega\in\Omega(\lambda)} |\mathrm{Im}(\omega)|, \tag{4.26}$$

where $\Omega(\lambda)$ is the set of the singular points of $f_{\mathrm{DE}}(z,\lambda)$. In the calculation, we focus on the zeros of the denominator of $f_{\mathrm{DE}}(z,\lambda)$, i.e., $\exp(\sinh(z)) + \lambda$, because the numerator $\exp(\alpha\sinh(z))\cosh(z)$ is analytic in $\mathbb{C}$. The zeros of the denominator are the solutions of the equation

$$\exp(\sinh(z)) = -\lambda. \tag{4.27}$$

Therefore, we can obtain (4.26) by considering the imaginary parts of the solutions of (4.27).

Let x and y be the real part and the imaginary part of z in (4.27), respectively. Then, the LHS and the RHS of (4.27) can be rewritten as

$$\exp(\sinh(z)) = \exp\left(\sinh(x)\cos(y) + \mathrm{i}\cosh(x)\sin(y)\right), \tag{4.28}$$

$$-\lambda = \exp(\log(\lambda) + \mathrm{i}k\pi) \quad (k = \pm1, \pm3, \pm5 \dots) \tag{4.29}$$

By comparing the RHS of (4.28) with that of (4.29), we have

$$\begin{cases} \sinh(x)\cos(y) = \log(\lambda), \\ \cosh(x)\sin(y) = k\pi. \end{cases} \tag{4.30}$$

By squaring the left and right of both equation of (4.30), we obtain

$$\begin{cases} \sinh^2(x)\cos^2(y) = (\log(\lambda))^2, \\ \cosh^2(x)\sin^2(y) = k^2\pi^2. \end{cases}$$

By substituting $\cosh^2(x) = 1 + \sinh^2(x)$ and $\cos^2(y) = 1 - \sin^2(x)$, we have

$$\left(1 + \frac{(\log(\lambda))^2}{1 - \sin^2(y)}\right)\sin^2(y) = k^2\pi^2. \tag{4.31}$$

Let us consider the minimum absolute solution of (4.31). The equation (4.31) has just one solution in $(0, \pi/2)$ for each k because $s(y) := [1 + (\log\lambda)^2/(1 - \sin^2 y)]\sin^2 y$ monotonically increases in $(0, \pi/2)$, and $s(y)$ satisfies $s(0) = 0$ and $\lim_{y\to\pi/2} s(y) = \infty$. In addition, the absolute minimum solution of (4.31) lies in $(0, \pi/2)$ because $s(y)$ is an even function. Furthermore, when $k^2 = 1$ we have the minimum solution among all odd number k because $s(y)$ monotonically increases. After all, the minimum absolute solution of (4.31) is obtained as the RHS of (4.25) by solving (4.31) with $k = 1$. In conclusion, we proved Proposition 4.7. □

Next, we make sure that Theorem 2.5 is applicable to f_{DE}. The following discussions are in a similar way as done in Subsection 3.3.1. Consider $\tilde{D} := \{z' \in \mathbb{C} : |\mathrm{Im}(z')| < d_0(\lambda) - \delta\}$ for a sufficiently small $\delta > 0$, and assume that $z = x + \mathrm{i}y \in \tilde{D}$. Then we have

$$|f_{\mathrm{DE}}(x + \mathrm{i}y, \lambda)| = \begin{cases} \mathcal{O}\left(\exp\left(-\dfrac{\alpha - 1}{2}\mathrm{e}^x\cos y\right)\right) & (x \longrightarrow \infty), \\ \mathcal{O}\left(\exp\left(-\dfrac{\alpha}{2}\mathrm{e}^{-x}\cos y\right)\right) & (x \longrightarrow -\infty) \end{cases}.$$

For more details of the calculations, see Appendix 4.B. In addition, there exists c such that

$$\int_{-\infty}^{\infty} |f_{\mathrm{DE}}(x + \mathrm{i}y)|\,\mathrm{d}x \le c$$

because $|f_{\mathrm{DE}}(z, \lambda)|$ is bounded in $\tilde{D}$ and $f_{\mathrm{DE}}(x+\mathrm{i}y, \lambda)$ decays faster than $\exp(-|x|)$ as $x \longrightarrow \pm\infty$. Therefore, Theorem 2.5 is applicable to f_{DE}, and for sufficiently small h, the convergence of the DE formula for $\log(\lambda)$ is approximately exponential with rate $\exp(-2\pi d_0(\lambda))$.

Similar to the DE formula for $\log(\lambda)$, we have two observations for the convergence. The first one is that the convergence rate of the DE formula for λ^α is approximately equal to that of the DE formula for $(1/\lambda)^\alpha$ because $d_0(\lambda) = d_0(1/\lambda)$. The second one is that for two positive scalar λ and $\tilde{\lambda}$ with $|\log(\lambda)| < |\log(\tilde{\lambda})|$, the convergence of the DE formula for λ^α is faster than that of the DE formula for $\tilde{\lambda}^\alpha$. This is because $\mathrm{d}d_0(\lambda)/\mathrm{d}(\log(\lambda))^2 > 0$ for $\lambda < 1$ and $\mathrm{d}d_0(\lambda)/\mathrm{d}(\log(\lambda))^2 < 0$ for $\lambda > 1$. We illustrate $d_0(\lambda)$ in Figure 4.1 for reference, and the graph in Figure 4.1 supports our observations.

4.3.2 Convergence of the DE formula for A^α

In Subsection 3.1, we learned that the convergence speed of the DE formula for λ^α is slow when $|\log(\lambda)|$ is large, i.e., λ is far away from 1. Therefore, for sufficient small h, it holds that

$$\left\| A^\alpha - \sum_{i=-\infty}^{\infty} hF_{\mathrm{DE}}(ih) \right\|_2 \le \max_{\lambda\in\{\lambda_{\max},\lambda_{\min}\}} \left| \lambda^\alpha - \sum_{i=-\infty}^{\infty} hf_{\mathrm{DE}}(ih, \lambda) \right|,$$

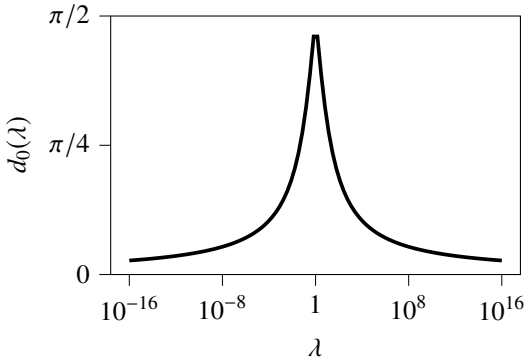


Figure 4.1: The width $d_0(\lambda)$ (defined in (4.25)) of a strip region in which $f_{\mathrm{DE}}(z, \lambda)$ is analytic.

where $\lambda_{\max}$ and $\lambda_{\min}$ is the largest and the smallest eigenvalue of A, respectively. If we assume that $\lambda_{\max}\lambda_{\min} = 1$, i.e., $|\log(\lambda_{\max})| = |\log(\lambda_{\min})|$, we have

$$\left\| A^{\alpha} - \sum_{i=-\infty}^{\infty} hF_{\mathrm{DE}}(ih) \right\|_2 \leq \mathcal{O}\left(\exp\left(-\frac{2\pi d_0(\lambda_{\max})}{h} \right) \right) = \mathcal{O}\left(\exp\left(-\frac{2\pi d_0(\sqrt{\kappa(A)})}{h} \right) \right)$$

because $d_0(\lambda_{\max}) = d_0(1/\lambda_{\max}) = d_0(\lambda_{\min})$. Note that the condition $\lambda_{\max}\lambda_{\min} = 1$ can be satisfied by computing $(\gamma_* A)^{\alpha}$ where $\gamma_* = 1/\sqrt{\lambda_{\max}/\lambda_{\min}}$, and can accelerate the convergence of the DE formula for matrices such that $\lambda_{\max}\lambda_{\min}$ is far away from 1.

If we further assume that we have a finite interval $[l, r]$ whose truncation error is smaller than the trapezoidal error, then the trapezoidal error can be represented using the number of abscissas $m = (r - l)/h + 1$:

$$\left\| A^{\alpha} - \sum_{i=1}^{m-2} hF_{\mathrm{DE}}(l + ih) - \frac{h}{2}(F_{\mathrm{DE}}(l) + F_{\mathrm{DE}}(r)) \right\|_2 \leq \mathcal{O}\left(\exp\left(-\frac{2\pi d_0\left(\sqrt{\kappa(A)}\right)}{r - l} m \right) \right). \quad (4.32)$$

4.3.3 Comparison of the convergence speed

In this subsection, we compare convergence speed of the following three quadrature formulas: the DE formula (`DE`), the GJ quadrature for (4.3) (`GJ1`), and the GJ quadrature for (4.4) (`GJ2`). Under the assumption that $\lambda_{\max}\lambda_{\min} = 1$, the error of the `GJ2` is

$$\left\| A^{\alpha} - \sum_{i=1}^{m} w_i F_{\mathrm{GJ2}}(x_i) \right\|_2 \leq \mathcal{O}\left(\exp\left(-2\log(\tau_{\mathrm{GJ2}}) m \right) \right). \quad (4.33)$$

where x_i and w_i are nodes and weights of m-point GJ quadrature [3], and

$$F_{\mathrm{GJ2}}(x) = \frac{2\sin(\alpha\pi)}{\pi} A\left[(1 - x)I + (1 + x)A \right]^{-1}, \quad \tau_{\mathrm{GJ2}} = \frac{1 + [\kappa(A)]^{1/4}}{|1 - [\kappa(A)]^{1/4}|}.$$

The error of the `GJ1` when α is reciprocal of a natural number is

$$\left\| A^{\alpha} - \sum_{i=1}^{m} w_i F_{\mathrm{GJ1}}(x_i) \right\|_2 \leq \mathcal{O}\left(\exp\left(-2\log(\tau_{\mathrm{GJ1}}) m \right) \right), \quad (4.34)$$

[3] Generally, the error of the m-point Gaussian quadrature is represented as $\mathcal{O}(\tau^{-2m})$ for some constant τ. In this paper, we write $\mathcal{O}(\exp(-2\log(\tau)m))$ because we want to compare the convergence of the Gaussian quadrature with that of the DE formula.

where

$$F_{\mathrm{GJ1}}(x) = \frac{2\sin(\alpha\pi)}{\alpha\pi} A\left[(1+x)^{1/\alpha} I + (1-x)^{1/\alpha} A\right]^{-1},$$
$$\tau_{\mathrm{GJ1}} = \frac{1 + [\kappa(A)]^{\alpha/2} + \sqrt{2[\kappa(A)]^{\alpha/2}(1-\cos(\alpha\pi))}}{\sqrt{1 + [\kappa(A)]^{\alpha} + 2[\kappa(A)]^{\alpha/2}\cos(\alpha\pi)}}.$$

See [28] for more details of convergence analysis for `GJ1` and `GJ2`.

Since `DE`, `GJ1`, and `GJ2` converges exponentially, upper bounds on the error of these quadrature formulas can be written in the form of $\mathcal{O}(\exp(-\phi(\kappa(A))m))$. We numerically calculated the coefficient ϕ for `DE`, `GJ1`, and `GJ2`. We set $\alpha = 0.1,\ 0.2, \ldots, 0.9$ and $\kappa_2(A) \in [10^{-16}, 10^{16}]$, and plotted results in Figure 4.2. In this calculation, we obtain a finite interval for `DE` by using Algorithm 5 with $\varepsilon = 2^{-53} \approx 1.1 \times 10^{-16}$. When α is not reciprocal of natural numbers, we skipped computing convergence speed of `GJ1` for (4.3) because the integrand is not analytic on $u = \pm 1$ in such cases.

Figure 4.2 shows that the convergence speed of `DE` is higher than that of `GJ2` when $\kappa(A)$ is larger than about 10^4. Conversely, the convergence speed of `DE` is slower than that of `GJ2` for small $\kappa(A)$. When α is reciprocal of a natural number and $\kappa(A)$ is large, the convergence speed of `GJ1` is higher than the other two quadratures except for $\alpha = 0.5$. By using Fig. 4.2, we can select the fastest convergence quadrature formula. For example, when we compute A^α where $\alpha = 0.5$ and $\kappa(A) = 10^{10}$, `DE` converges fastest among the three quadratures.

4.4 Preconditioning of quadrature-based algorithms based on the relation $A^\alpha = (AP)^\alpha P^{-\alpha}$

From the previous section, we found that the convergence of quadrature formulas for the fractional power of an SPD matrix A are slow when $\kappa(A)$ is large. In order to avoid slow convergence in such a situation, we consider another approach to reducing the number of abscissas based on the relation

$$A^\alpha = (AP)^\alpha P^{-\alpha}. \tag{4.35}$$

Since $A^\alpha = \exp(\alpha \log(A))$, the relation (4.35) holds if the conditions of Theorem 3.7 are satisfied, i.e.,

- all of the eigenvalues of $P \in \mathbb{C}^{n\times n}$ lie in $\mathbb{C} \setminus (-\infty, 0]$,
- $AP = PA$, and
- for every eigenvalue λ_j of A and the corresponding eigenvalue μ_j of P, $|\arg\lambda_j + \arg\mu_j| < \pi$.

In order to reduce the number of abscissas, we choose P so that $\kappa(AP) < \kappa(A)$ and $\kappa(P) < \kappa(A)$, and compute $\log(A)$ via the RHS of (4.35). In this case, we propose $P = (A + \sqrt{\lambda_{\max}\lambda_{\min}} I)^{-1} = \arg\min_{P'\in\Phi}\{\kappa(AP'), \kappa(P')\}$, where $\Phi = \{(A + pI)^{-1} : p > 0\}$ because this is the same situation as discussed in Section 3.4.

We show the procedure of the preconditioning in Algorithm 6. At Step 4 of Algorithm

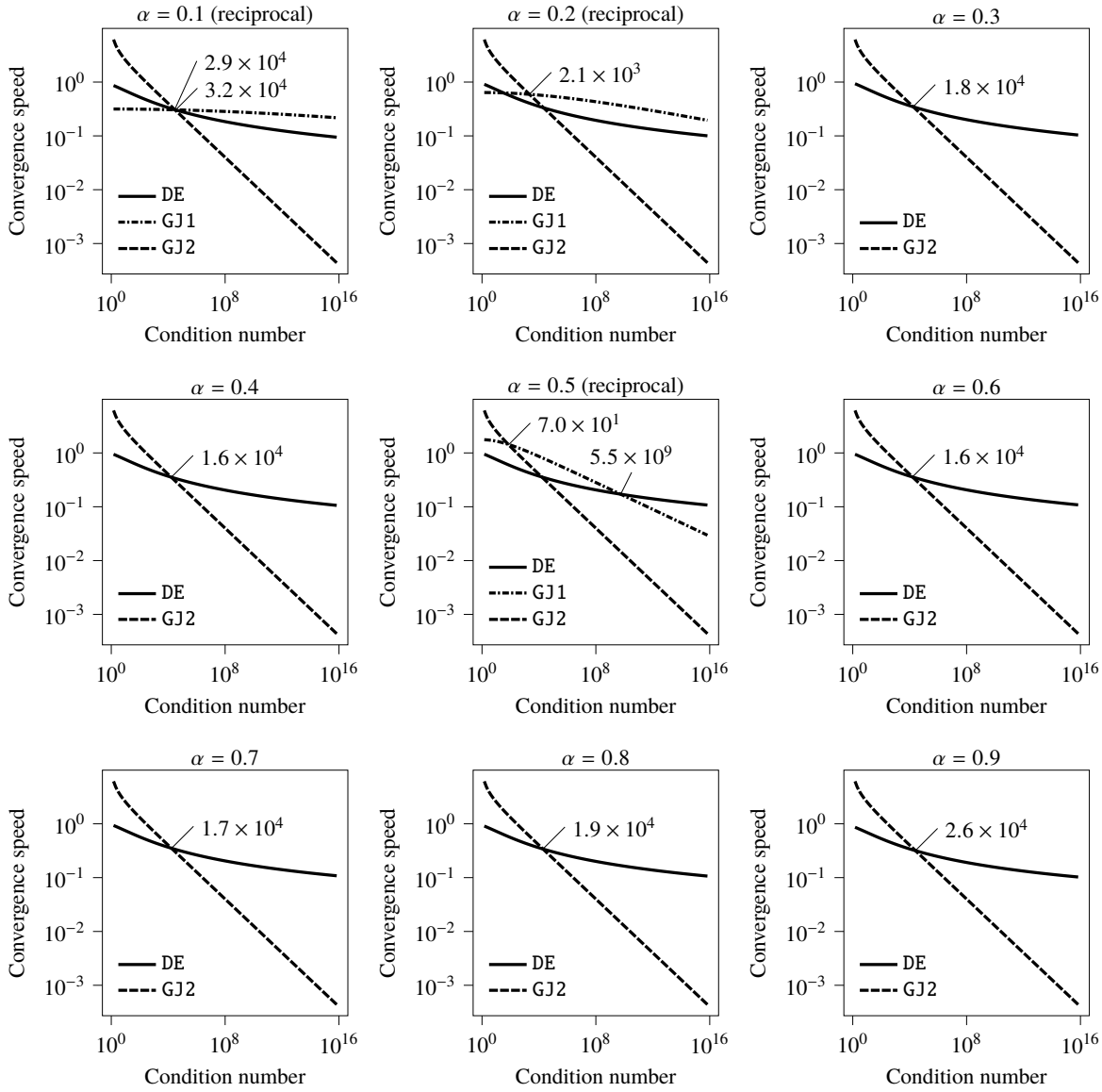


Figure 4.2: The convergence speed of the following quadrature formulas: the DE formula (`DE`), the GJ quadrature for (4.3) (`GJ1`), and the GJ quadrature for (4.4) (`GJ2`). The upper bound on the error for `DE`, `GJ1`, and `GJ2` are shown in (4.32), (4.33), and (4.34) respectively. The vertical axes are the convergence speed, i.e., the constant ϕ in the error of the m-point quadrature formula $\exp(-\phi m)$. Therefore, the convergence of a quadrature formula is fast if the values in graphs are large. We annotated condition numbers when the fastest quadrature formula is changed.

Algorithm 6 Splitting precondition for the fractional power of SPD matrices.

Input: $A \in \mathbb{C}^{n\times n}$, $\alpha \in (0,1)$

1: $\tilde{A} = A/\sqrt{\lambda_{\max}\lambda_{\min}}$

2: $p_{\tilde{A}+I} = 1/\sqrt{2+\sqrt{\kappa}+1/\sqrt{\kappa}}$, $p_{\tilde{A}(\tilde{A}+I)^{-1}} = (1+\sqrt{\kappa})\kappa^{-1/4}$

3: Compute $(\tilde{A}+I)^{\alpha} = p_{\tilde{A}+I}^{-\alpha}[p_{\tilde{A}+I}(\tilde{A}+I)]^{\alpha}$

4: Compute $[(\tilde{A}+I)^{-1}\tilde{A}]^{\alpha} = p_{\tilde{A}(\tilde{A}+I)^{-1}}^{-\alpha}[p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A}]^{\alpha}$

5: Output $A^{\alpha} = (\lambda_{\max}\lambda_{\min})^{\alpha/2}\tilde{A}^{\alpha} = (\lambda_{\max}\lambda_{\min})^{\alpha/2}\left[(\tilde{A}+I)^{-1}\tilde{A}\right]^{\alpha}(\tilde{A}+I)^{\alpha}$

6, we can compute $[(\tilde{A}+I)^{-1}\tilde{A}]^{\alpha}$ without computing $(\tilde{A}+I)^{-1}$ directory:

$$\begin{aligned}\left[(\tilde{A}+I)^{-1}\tilde{A}\right]^{\alpha} &= \frac{\sin(\alpha\pi)}{\alpha\pi}(\tilde{A}+I)^{-1}\tilde{A}\int_0^{\infty}\left[t^{1/\alpha}I+(\tilde{A}+I)^{-1}\tilde{A}\right]^{-1}\mathrm{d}t \\ &= \frac{\sin(\alpha\pi)}{\alpha\pi}\tilde{A}\int_0^{\infty}\left[t^{1/\alpha}(\tilde{A}+I)+\tilde{A}\right]^{-1}\mathrm{d}t\,.\end{aligned}$$

At the last of this chapter, we estimate the effect of the preconditioning. We calculated the convergence speed for $\alpha = 0.1,\ 0.5,\ 0.8$ and illustrated the convergence speed in Figure 4.3. We consider the convergence speed to be the coefficient $\phi(\kappa)$ in the error of a quadrature formula $\mathcal{O}(\exp(-\phi(\kappa)m))$ where m is the number of integrand evaluations. Figure 4.3 shows the effect of the preconditioning. For example, when $\alpha = 0.8$ and $1.3\times10^{2} \lesssim \kappa(A) \lesssim 1.0\times10^{8}$, we expect that we can reduce the number of abscissas by using preconditioned `GJ1`.

4.5 Numerical Experiments

The numerical experiments were carried out using Julia 1.2 on a Core-i7 (3.4 GHz) CPU with 16 GB RAM. We used the IEEE double precision arithmetic unless otherwise stated. When we consider arbitrary precision arithmetic, we implement the code by using the `BigFloat` type of Julia whose machine epsilon is about 1.7×10^{-77}. We computed abscissas and weights in the GJ quadrature with `FastGaussQuadrature.jl`[4]. In this test, we used the same test matrices in Chapter 3 shown in Tab. 3.1.

4.5.1 Test 1: checking the appropriateness of the determined finite interval

In this test, we check that Algorithm 5 selects a finite interval appropriately. We consider the computation of the square root ($\alpha = 0.5$) of test matrices[5]. First, we scaled the test matrix as $\tilde{A} = A/\sqrt{\sigma_{\max}\sigma_{\min}}$ where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and the minimum singular value of A, respectively. Then, we computed the reference solution $\tilde{A}^{1/2}$ by using the scaled Denman–Beavers iteration [39, p. 148] with arbitrary precision. After that, with the tolerance $\varepsilon = 10^{-11}, 10^{-14}$, we computed $\tilde{A}^{1/2}$ via Algorithm 1. For Step 2 of Algorithm 5, we compute the parameters $\|A\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$ in two ways. The one are computed by using `eigvals` and `svdvals` in Julia that are function computing all the eigenvalues and the singular values respectively. The others are computed by using `Arpack.jl` which compute eigenvalues and singular values with restarted Lanczos or Arnoldi iterations. We computed parameters $\|A\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$ with three-digit accuracy. Furthermore, for the steps 7 and 8 of Algorithm 5, we used arbitrary precision arithmetic in order to avoid the error that stems from matrix inversion.

The convergence histories are illustrated in Figure 4.4. Figure 4.4 shows that we obtained approximations with the desired accuracy in all cases. Thus, the truncation error of Algorithm 5 was smaller than a given tolerance. In addition, the approximates are accurate regardless of the accuracy of the parameters $\|A\|_2$, $\|A^{-1}\|_2$, and $\rho(A)$. This results support the discussions in Subsection 3.3.

[4] `https://github.com/JuliaApproximation/FastGaussQuadrature.jl`

[5] We computed $(-\tilde{A})^{1/2}$ for `pores_1` alternatively because the eigenvalues of `pores_1` lie in the set $\mathbb{C}\backslash[0,\infty)$.

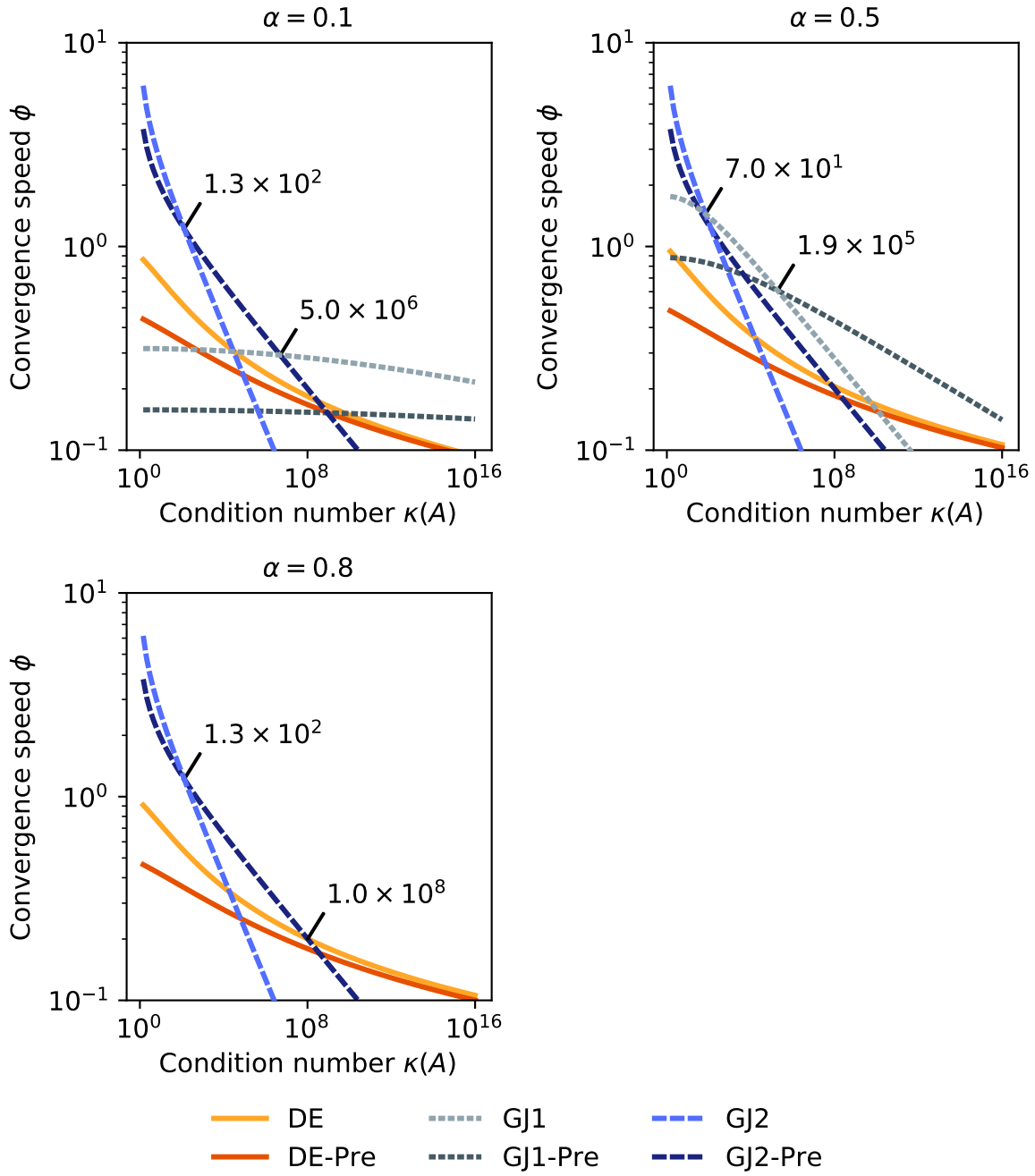


Figure 4.3: Comparison of the convergence speed of the GL quadrature and the DE formula when considering the preconditioning $A^\alpha = (AP)^\alpha P^{-\alpha}$ where $P = (\sqrt{\lambda_{\max}\lambda_{\min}}A + I)^{-1}$. GJ1, GJ2, and DE denote the convergence speed $\phi_{\mathrm{GJ1}}(\kappa)$, $\phi_{\mathrm{GJ2}}(\kappa)$, $\phi_{\mathrm{DE}}(\kappa)$ respectively as in Fig. 4.5. GJ1-Pre, GJ2-Pre, and DE-Pre denote that the convergence speeds of the quadrature formulas with preconditioning respectively, i.e., $\phi_{\mathrm{GJ1}}(\kappa^{1/2})/2$, $\phi_{\mathrm{GJ2}}(\kappa^{1/2})/2$ and $\phi_{\mathrm{DE}}(\kappa^{1/2})/2$. We annotated condition numbers when the fastest quadrature formula is changed.

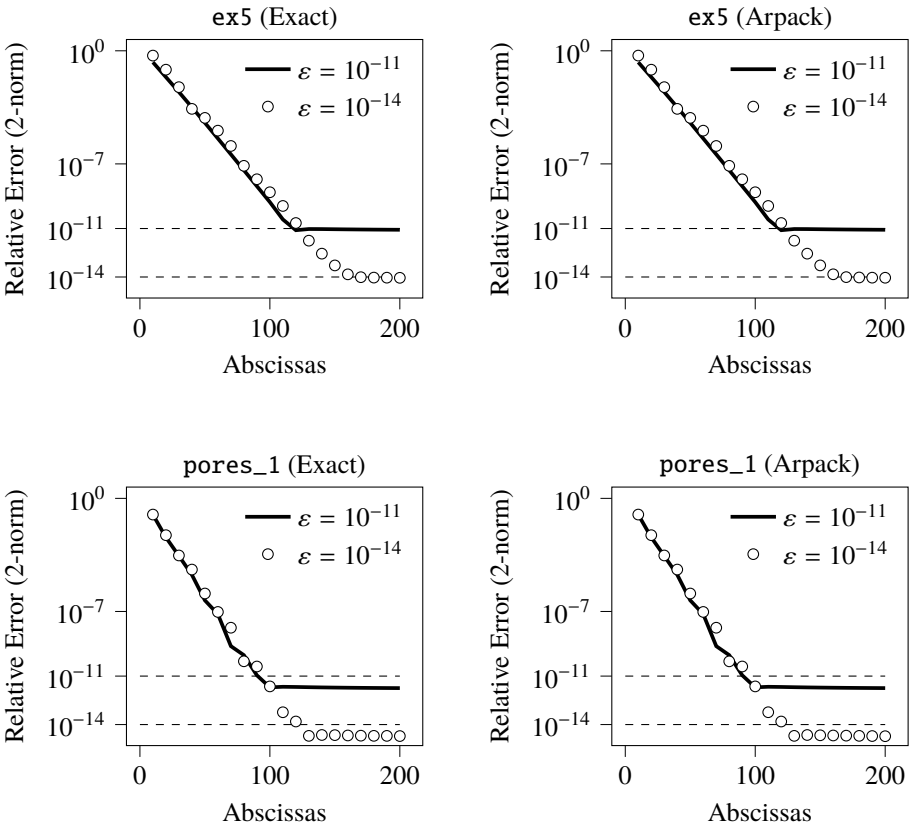


Figure 4.4: Convergence histories of the DE formula for $\tilde{A}^{1/2}$ in Test 1. The titles in each graph stand for the test matrix and the method for computing parameters $\|\tilde{A}\|_2$, $\|\tilde{A}^{-1}\|_2$, and $\rho(\tilde{A})$. "Exact" means that the parameters are computed by using `eigvals` and `svdvals`, and "Arpack" means that the parameters are computed approximately by using `Arapack.jl`. The value ε in legend represents the input parameter for Algorithm 5.

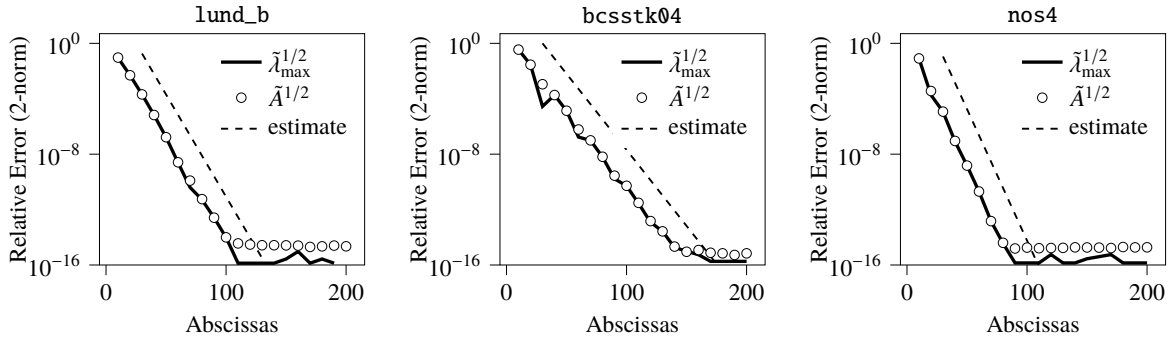


Figure 4.5: Convergence histories of the DE formula for the square root of the test matrices in Tab. 3.1.

4.5.2 Test 2: checking the convergence speed of the DE formula

In this test, we check the convergence speed of the DE formula for the fractional power of SPD matrices. First, we scaled the test matrices $\tilde{A} = A/\sqrt{\lambda_{\max}\lambda_{\min}}$ and compute the reference solution $\tilde{A}^{1/2}$ as done in Test 1. Then, we compute $\tilde{A}^{1/2}$ by using the DE formula (Algorithm 5) with $\varepsilon = 2^{-53} \approx 1.1\times10^{-16}$ in the double precision environment. After that, we compute the square root of the maximum eigenvalue of $\tilde{A}$, $\tilde{\lambda}_{\max}$, by using the DE formula whose integral interval is the same as that of the DE formula for $\tilde{A}^{1/2}$. We plotted the convergence histories of the DE formula for $\tilde{A^{1/2}}$ and $\lambda_{\max}^{1/2}$ in Figure 4.5. In addition, we plotted the estimates of the convergence rate (4.32) in Fig. 4.5.

Figure 4.5 shows that the convergence of the DE formula for $\tilde{A}^{1/2}$ is almost the same as that of the DE formula for $\lambda_{\max}^{1/2}$. Furthermore, the convergence rate of the quadrature formula is approximately equal to the estimate. These results support our convergence analysis in Section 3.

4.5.3 Test 3: comparison the DE formula with the GJ quadrature

In this test, we compare the convergence of the DE formula with that of the GJ quadrature. We computed A^{α} for $\alpha = 0.2, 0.5, 0.8$ and plotted convergence histories in Figure 4.6. Figure 4.6 shows that when α is not reciprocal of natural numbers ($\alpha = 0.8$) and $\kappa(A)$ is large (`SPD_ill` and `NS_ill`), `DE` converged fastest among three quadrature formulas. On the other hand, `GJ1` converges fast when α is reciprocal of a natural number, and `GJ2` converged fast when $\kappa(A)$ is small. These results support that the DE formula is effective when the convergence of the GJ quadrature is slow.

4.5.4 Test 4: checking the effect of the preconditioning

In this test, we check the effect of the preconditioning with test matrices used in Test 2. We computed A^{α} for $\alpha = 0.1, 0.5, 0.8$ by using the GJ quadrature and the DE formula with and without preconditioning (Algorithm 6), and plotted the convergence histories in Figure 4.7. Figure 4.7 shows that preconditioning reduced the number of evaluations of integrand in some cases. For example, when computing $A^{0.8}$ for `lund_b`, `GJ2-Pre` converges fastest among the all quadrature formulas. On the other hand, the preconditioned quadrature formulas are slightly more inaccurate than quadrature formulas without preconditioning. This is the same tendency as the preconditioning of quadrature formulas for $\log(A)$ in Subsection 3.5.4. We leave the discussions on the accuracy for future work.

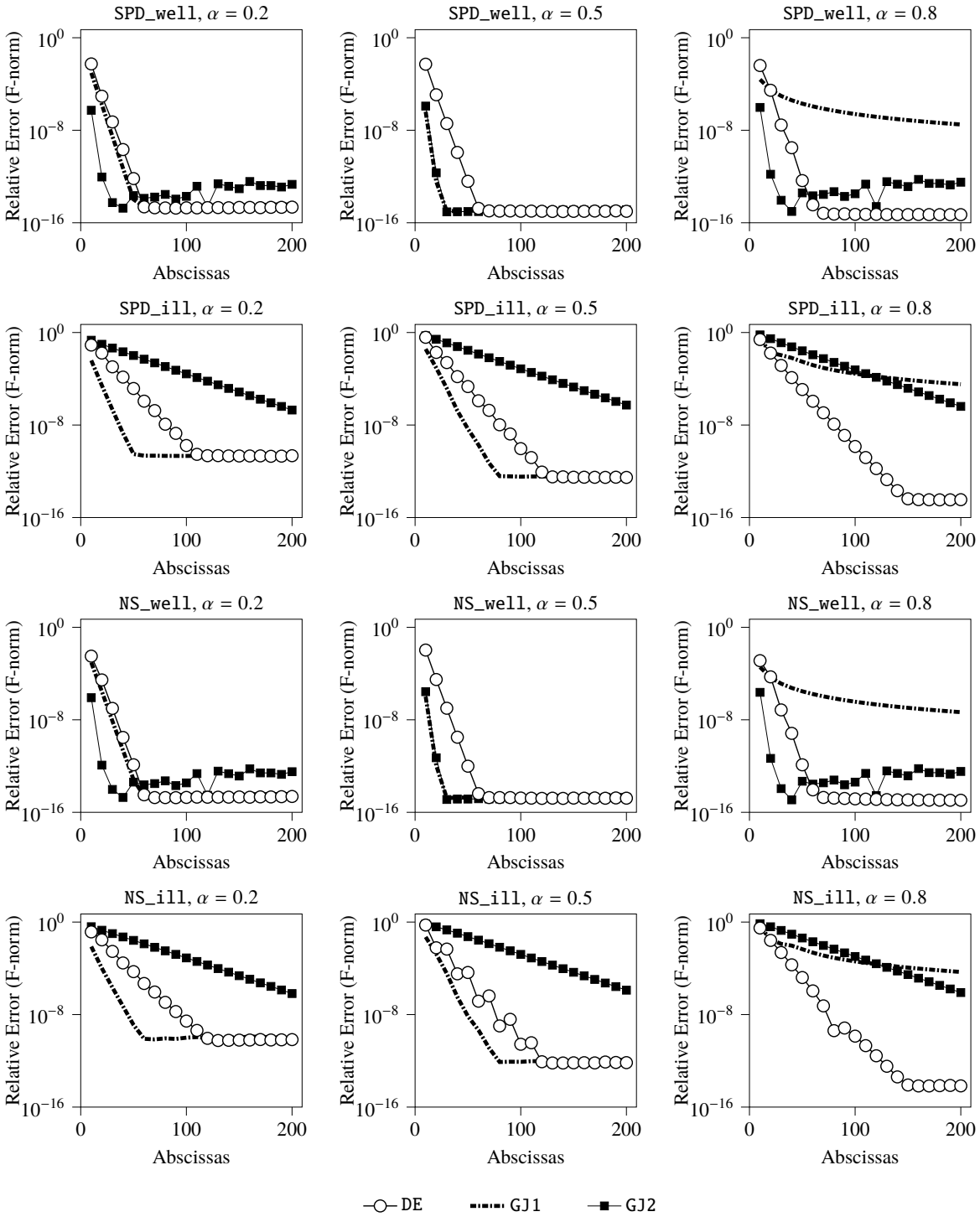


Figure 4.6: Convergence histories of the three quadratures formulas: the DE formula based on Algorithm 5 (`DE`), the GJ quadrature for (4.3) (`GJ1`), and the GJ quadrature for (4.4) (`GJ2`).

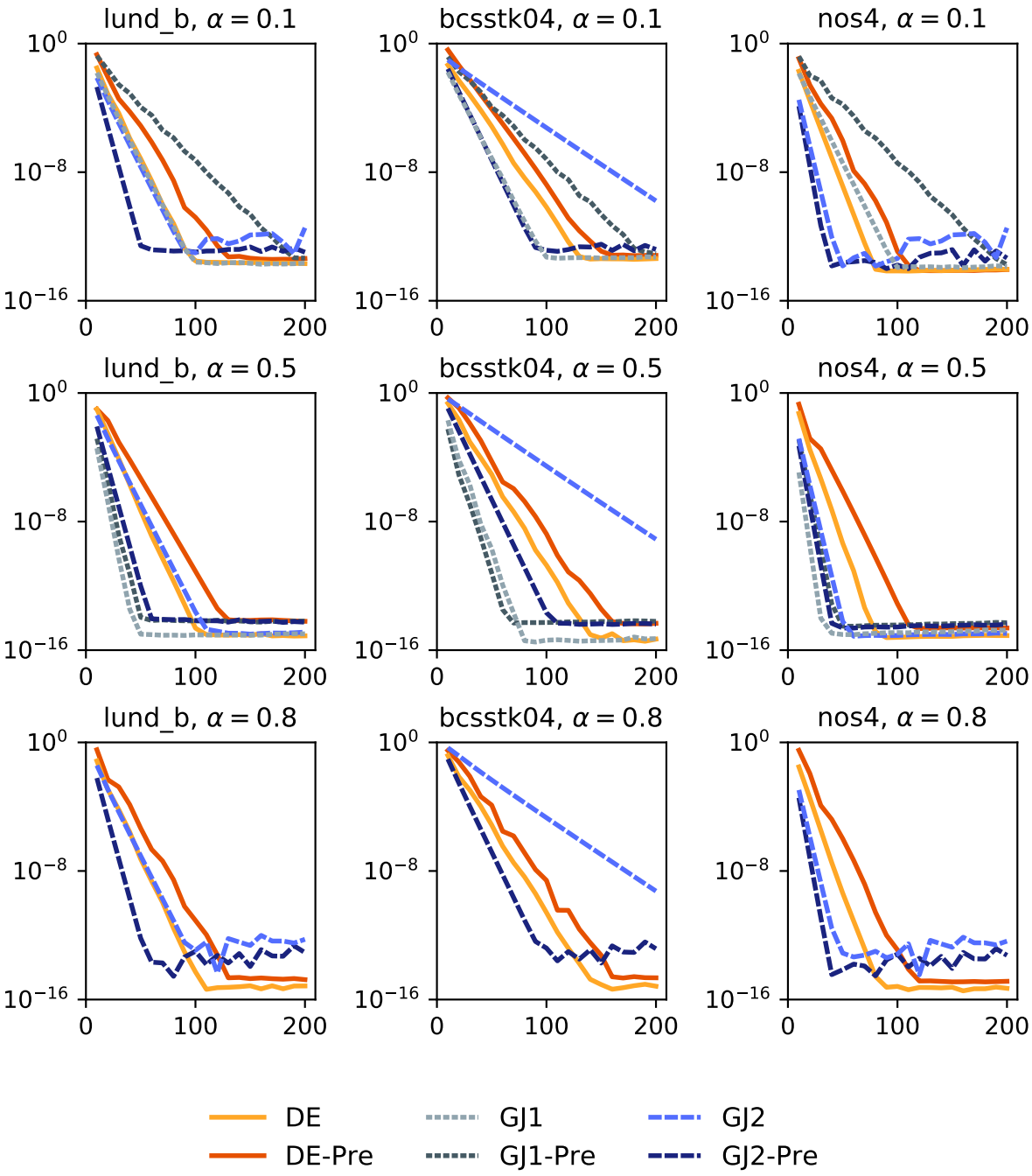


Figure 4.7: Convergence histories of quadrature formulas for Test 4. We consider on the x-axis the number of evaluations of the integrand and on the y-axis the relative error in the Frobenius norm $\|\tilde{A}^{\alpha} - X\|_F / \|\tilde{A}^{\alpha}\|_F$. DE, GJ1, and GJ2 means the DE formula based on Algorithm 5, the GJ quadrature for (4.3), and the GJ quadrature for (4.4) respectively. DE-Pre, GJ1-Pre, and GJ2-Pre mean that we performed the preconditioning based on Algorithm 6 and computed $\tilde{A}^{\alpha}$ via DE, GJ1, GJ2 respectively. When considering the preconditioning, we compute both of $\log(p_{\tilde{A}+I}(\tilde{A}+I))$ and $\log(p_{\tilde{A}(\tilde{A}+I)^{-1}}(\tilde{A}+I)^{-1}\tilde{A})$ by using the $m/2$-point quadrature formula. Each title of the graphs represents the test matrix and the value of α.

4.6 Conclusion

In this chapter, we first focused on the DE formula as a new choice of quadrature formulas for A^α. In order to utilize the DE formula, we proposed a method of selecting an appropriate finite interval on the basis of error analysis. We analyzed the convergence of the DE formula for SPD matrices, and we ensured that when α is not reciprocal of natural numbers and $\kappa(A)$ is large, the DE formula converges faster than the GJ quadratures. In addition, we considered the preconditioning for SPD matrices based on the relation $A^\alpha = (AP)^\alpha P^{-\alpha}$ and showed a choice of P. We also estimated the effect of preconditioning.

We carried out four numerical tests. The first one showed that our algorithm selected a finite interval whose truncation error is smaller than or equal to a given tolerance. The second one showed that our estimation on convergence rate is appropriate. The third one showed that the DE formula is effective when the GJ quadratures converge slow, and the fourth one showed that the preconditioning accelerates the convergence as the estimates.

Our future work will focus on three problems as discussed in Section 3.6. The first one is a further error analysis of quadrature formulas. For example, analysis of the convergence rate of quadrature formulas for nonsymmetric matrices and the analysis of the absolute error which affects the convergence when the number of abscissas is small. The second one is developing more efficient quadrature formulas. It may be done by modifying the variable transformation. The third one is the verification of the practical performance of the presented algorithms when applied to large (sparse) matrices from current research problems.

Appendix 4.A Two algorithms for A^α based on the DE formula

As seen in Section 3.A, we can use the DE formula as an adaptive quadrature algorithm once we have an appropriate finite integral interval. Thus, we establish an adaptive quadrature algorithm based on the DE formula for A^α as done in Section 3.A, and show the algorithm in Algorithm 7. The computational cost of Algorithm 7 for dense A is $(2m_{k+1} + 2)n^3 + \mathcal{O}(n^2) = [2^{k+1}(m_0 - 1) + 4]n^3 + \mathcal{O}(n^2)$, and the computational cost of $\log(A)\boldsymbol{b}$ with sparse A is $[2^k(m_0 - 1) + 1]c_{\text{abscissa}} + c_{\text{mul}} + c_{\text{param}}$.

We also show an algorithm working without selecting the number of abscissas for an SPD matrix A in Algorithm 8 which is based on the computation of the fractional power of the maximum eigenvalue [6].

At the end of this chapter, we show numerical results for checking Algorithm 3. Our test matrices are described in Table 4.1, and the reference solutions are computed by the scaled Denman–Bervers iteration as done in Section 4.5. We applied Algorithm 7 to test matrices with $\varepsilon = 10^{-8}$ and $m_0 = 16$, and we show the results in Table 4.2. Table 4.2 shows Algorithm 3 worked well for all cases.

[6] The idea predicting the number of abscissas of Algorithm 8 is pointed out in [28].

Algorithm 7 Adaptive quadrature algorithm for A^α based on the DE formula.

Input A, $\alpha \in (0,1)$, $m_0 \leq 2$, $\varepsilon > 0$. ▷ For example, one can set $m_0 = 16$.

Set $\tilde{F}_{\mathrm{DE}}(x) := \exp(\alpha \sinh(x)) \cosh(x) [\exp(\sinh(x))I + A]^{-1}$
Compute $\rho(A)$, l, r using Algorithm 1 from step 2 to step 6
$h_0 = (r - l)/(m_0 - 1)$
$T_0 = \frac{h_0}{2}\tilde{F}_{\mathrm{DE}}(l) + \frac{h_0}{2}\tilde{F}_{\mathrm{DE}}(r) + h_0 \sum_{i=1}^{m-2} \tilde{F}_{\mathrm{DE}}(l + ih)$
for $k = 0, 1, 2, \ldots$ until convergence **do**
    $h_{k+1} = h_k/2$
    $T_{k+1} = \frac{1}{2}T_k + h_{k+1} \sum_{i=1}^{m_k - 1} \tilde{F}_{\mathrm{DE}}(l + (2i-1)h_{k+1})$
    $m_{k+1} = 2m_k - 1$
    if $\frac{1}{3}\|T_{k+1} - T_k\|/\rho(A)^\alpha \leq \varepsilon$ **then**
        $T = T_{k+1}$
        break
    end if
end for
Output $\frac{\sin(\alpha\pi)}{\pi} AT \approx A^\alpha$

Algorithm 8 Computing the fractional power of an SPD matrix with the appropriate number of abscissas.

Input: an SPD matrix A, $\alpha \in (0,1)$, $\varepsilon > 0$

Compute $\lambda_{\max}$, $\lambda_{\min}$, the maximum and the minimum eigenvalues of A respectively.
$\tilde{A} = (\lambda_{\max}\lambda_{\min})^{-1/2} A$, and $\tilde{\lambda}_{\max} = (\lambda_{\max}\lambda_{\min})^{-1/2}\lambda_{\max}$.
Compute l, r by using from Step 2 to Step 6 of Algorithm 5 for $\tilde{A}$.
Find the minimum positive integer m satisfying

$$\frac{\left|\tilde{\lambda}_{\max}^\alpha - h(m)\left[\frac{1}{2}f_{\mathrm{DE}}(l, \tilde{\lambda}_{\max}) + \sum_{j=2}^{m-1} f_{\mathrm{DE}}(l + jh, \tilde{\lambda}_{\max}) + \frac{1}{2}f_{\mathrm{DE}}(r, \tilde{\lambda}_{\max})\right]\right|}{\tilde{\lambda}_{\max}^\alpha} \leq \varepsilon$$

where $h(m) = (r - l)/(m - 1)$.
Compute $\tilde{A}^\alpha$ by using the m-point DE formula (Step 7 of Algorithm 5)
Output $(\lambda_{\max}\lambda_{\min})^{\alpha/2}\tilde{A}^\alpha \approx A^\alpha$

Table 4.1: Test matrices.

Matrix	Size n	$\kappa_2(A)$	Structure
`bcsstk02` [19]	66	4.3×10^3	SPD
`bcsstk03` [19]	112	2.3×10^6	SPD
`ck104` [19]	104	5.5×10^3	Nonsymmetric

Table 4.2: Results of numerical test for checking Algorithm 7. Bold numbers mean the numbers of linear systems to be solved, and numbers in parentheses mean relative errors when the algorithm stopped.

α	bcsstk02	bcsstk03	ck104
$\alpha = 0.2$	**145** (2.1×10^{-14})	**289** (7.4×10^{-12})	**145** (1.6×10^{-14})
$\alpha = 0.4$	**145** (4.7×10^{-15})	**289** (6.1×10^{-13})	**145** (6.0×10^{-15})
$\alpha = 0.5$	**145** (2.6×10^{-15})	**289** (1.7×10^{-13})	**145** (3.1×10^{-15})

Appendix 4.B On the behavior of $|f_{\mathrm{DE}}(x+\mathrm{i}y,\lambda)|$ as $x \longrightarrow \pm\infty$

Similar to Appendix 3.B, we check that

$$\begin{aligned}|f_{\mathrm{DE}}(x+\mathrm{i}y,\lambda)| &= \left|\frac{\sin(\alpha\pi)\lambda}{\pi}\frac{\exp(\alpha\sinh(x))\cosh(x)}{\exp(\sinh(x))+\lambda}\right| \\ &= \begin{cases} \mathcal{O}\left(\exp\left(-\dfrac{\alpha-1}{2}\mathrm{e}^{x}\cos y\right)\right) & (x\longrightarrow\infty), \\ \mathcal{O}\left(\exp\left(-\dfrac{\alpha}{2}\mathrm{e}^{-x}\cos y\right)\right) & (x\longrightarrow-\infty) \end{cases}. \end{aligned} \tag{4.36}$$

It is checked that

$$|\cosh(x+\mathrm{i}y)| = \sqrt{\cosh^2 x+\cos^2 y-1} = \mathcal{O}\left(\exp\left(\frac{1}{2}\mathrm{e}^{|x|}\right)\right) \quad (x\longrightarrow\pm\infty) \tag{4.37}$$

in Appendix 3.B. Thus, we only consider $|\exp(\alpha\sinh(x+\mathrm{i}y))|$ and $|\exp(\sinh(x+\mathrm{i}y))+\lambda|$. By defining $a=\sinh x\cos y$ and $b=\cosh x\sin y$, we have

$$\begin{aligned}|\exp(\alpha\sinh(x+\mathrm{i}y))| &= |\exp(\alpha a+\mathrm{i}\alpha b)| = \exp(\alpha a) = \exp(\alpha\sinh x\cos y) \\ &= \mathcal{O}\left(\exp\left(\pm\frac{\alpha}{2}\mathrm{e}^{|x|\cos y}\right)\right) \quad (x\longrightarrow\pm\infty) \end{aligned} \tag{4.38}$$

In addition, it follows that

$$\begin{aligned}|\exp(\sinh(x+\mathrm{i}y))+\lambda| &= |\exp(a+\mathrm{i}b)+\lambda| = \sqrt{(\exp a\cos b+\lambda)^2+\exp(2a)\sin^2 b} \\ &= \sqrt{\exp(2a)+2\lambda\exp a\cos b+\lambda^2} \\ &= \mathcal{O}\left(\exp\left(\frac{1}{2}\mathrm{e}^{x}\cos y\right)\right) \quad (x\longrightarrow\infty), \end{aligned} \tag{4.39}$$

and $\lim_{x\to-\infty}|\exp(\sinh(x+\mathrm{i}y))+\lambda| = \lambda$. By combining (4.37), (4.38) and (4.39), we have (4.36).

Chapter 5

A variant of Newton's iteration for $A^{1/p}$

5.1 Introduction

In this chapter, we consider the computation of the matrix pth root $A^{1/p}$ ($p \in \mathbb{N}$), which is the special cases of the matrix fractional power A^{α}. The restriction of the exponent gives us more variety of computational methods than when computing general fractional power. For example, the Schur method for the matrix square root is generalized in [58], and iterative methods solving the equation $X^p - A = O$ [31, 45, 46]. In this study, we consider Newton's method, since that method is the most fundamental iterative method. In addition, it has been reported that Newton's method gives a more accurate solution than the Schur method for some ill-conditioned matrices [45].

Now, let us recall several results for Newton's method by Smith [58] and Iannazzo [45]. In order to apply Newton's method to the equation $F(X) := X^p - A = O$, we have to compute the inverse map of the Fréchet derivative of F. From the definition of F, it follows that

$$F(X + E) - F(X) = (X + E)^p - X^p = \sum_{i=0}^{p-1} X^i E X^{p-1-i} + \mathcal{O}(\|E\|),$$

therefore the Fréchet derivative of F is

$$L(X, E) = \sum_{i=0}^{p-1} X^i E X^{p-1-i}.$$

Therefore, Newton's iteration for the matrix pth root for an initial guess X_0 is

$$\begin{cases} \text{solve } \displaystyle\sum_{i=0}^{p-1} X_k^i E_k X_k^{p-1-i} = A - X_k^p \text{ for } E_k, \\ X_{k+1} = X_k + E_k, \end{cases} \qquad k = 0, 1, 2, \dots. \tag{5.1}$$

Newton's iteration (5.1) requires solving a generalized Sylvester equation; however, no $\mathcal{O}(n^3)$ flops methods are known for solving this generalized Sylvester equation for $p > 2$ [39, p. 178]. In order to compute Newton's iteration with in $\mathcal{O}(n^3)$ flops, one supposes the commutativity of X_k and A, and simplify the iteration (5.1). In other words, X_0 commutes with A, then X_k commutes with A for all k and the iteration can be simplified as

$$X_{k+1} = \frac{(p-1)X_k + AX_k^{1-p}}{p}, \quad k = 0, 1, 2, \dots. \tag{5.2}$$

Problems for utilizing the simplified Newton's method (5.2) are the selection of initial guess X_0 and numerical instability of the simplified Newton's iteration. In [45], Iannazzo utilized Newton's method for $A^{1/p}$. First, Iannazzo considered the initial guess X_0. Iannazzo showed that if both of the following conditions,

$$\begin{aligned} &\text{all eigenvalues of } A \text{ lie in the set } \{z \in \mathbb{C} : \ \mathrm{Re}(z) > 0,\ |z| \le 1\}, \\ &X_0 = I \end{aligned} \tag{5.3}$$

are satisfied, then Newton's method (5.2) converges to $A^{1/p}$. Next, Iannazzo proposed a preconditioning step, computing $\tilde{A} = A^{1/2}/\left\|A^{1/2}\right\|$ with a consistent norm (say, p-norm, Frobenius norm), because then $\tilde{A}$ satisfies the condition (5.3) for any A. Even if the matrix A is preconditioned, Newton's iteration (5.2) could be unstable in the neighborhood of $A^{1/p}$ [58]. Then, Iannazzo proposed three stable iterations:

$$\begin{cases} X_{k+1} = X_k + H_k,\ F_k = X_k X_{k+1}^{-1}, \\ H_{k+1} = -\frac{1}{p} H_k \left(\sum_{i=0}^{p-2} (i+1) X_{k+1}^{-1} F_k^i \right) H_k, \end{cases} \qquad \left(X_0 = I,\ H_0 = \frac{A - I}{p} \right) \tag{5.4}$$

$$\begin{cases} X_{k+1} = X_k + H_k,\ F_k = X_k X_{k+1}^{-1}, \\ H_{k+1} = -X_k \left(\frac{I - F_k^p}{p} + F_k^{p-1} (F_k - I) \right), \end{cases} \qquad \left(X_0 = I,\ H_0 = \frac{A - I}{p} \right) \tag{5.5}$$

and

$$\begin{cases} X_{k+1} = X_k \left(\dfrac{(p-1)I + N_k}{p} \right), \\ N_{k+1} = \left(\dfrac{(p-1)I + N_k}{p} \right)^{-p} N_k. \end{cases} \qquad \left(X_0 = I,\ N_0 = A \right) \tag{5.6}$$

In particular, iteration (5.4) is called incremental Newton (IN) iteration, and iteration (5.6) is called coupled Newton iteration.

It is known that Newton's method converges quadratically in a neighborhood of the solution, but global convergence of that method is not guaranteed. One way to globalize the convergence of Newton's method is by using damping.[1] From this point of view, it might be possible to apply damping to IN iteration (5.4) and iteration (5.5). Comparing these two iterations, the cost of IN iteration (5.4) is $\mathcal{O}(n^3 p)$ flops per iteration, higher than $\mathcal{O}(n^3 \log p)$ flops for iteration (5.5). On the other hand, the incremental part of IN iteration (5.4) is computed in the form of $H_{k+1} = f_k(H_k)$, in contrast to iteration (5.5). This characteristic of IN iteration (5.4) might provide a new viewpoint for convergence analysis to confirm that H_k converges to O. That is to say, if H_{k+1} explicitly includes H_k, then H_{k+1} is represented as $H_{k+1} = (f_k \circ f_{k-1} \circ \ldots \circ f_0)(H_0)$, and its convergence behavior might be analyzed using composite mapping $(f_k \circ f_{k-1} \circ \ldots \circ f_0)$ and initial matrix H_0. Thus, IN iteration (5.4) is worth considering.

The purpose of this paper is to provide a cost-efficient variant of IN iteration (5.4) whose increment part is computed in the form $H_{k+1} = f_k(H_k)$. In this paper, we reduce the cost of IN iteration (5.4) by finding a specific matrix polynomial in IN iteration (5.4) and proposing a decomposition of the matrix polynomial.

The remainder of this paper is organized as follows. In section 2, a variant of IN iteration is shown, and we numerically estimate its cost at $\mathcal{O}(n^3 \log p)$ flops per iteration. In section 3, we present the results of numerical experiments. We conclude in section 4.

[1] A damped Newton iteration is represented as $X_{k+1} = X_k + \alpha_k H_k$ ($\alpha_k \in (0, 1]$), where α_k is a relaxation factor chosen to reduce residuals.

5.2 Variant of IN iteration

The computational cost for computing the increment part

$$H_{k+1} = -\frac{1}{p} H_k \left(\sum_{i=0}^{p-2} (i+1) X_{k+1}^{-1} F_k^i \right) H_k \tag{5.7}$$

is the highest in IN iteration (5.4), because $(2p + 2/3)n^3 + \mathcal{O}(n^2)$ flops are required for Eq. (5.7), and $(2p + 10/3)n^3 + \mathcal{O}(n^2)$ flops for IN iteration (5.4). In this section, without losing the previous matrix H_k, Eq. (5.7) is rewritten to reduce the number of matrix multiplications whose computational costs are $\mathcal{O}(n^3)$ flops.

5.2.1 Rewriting the increment

From the definition of IN iteration (5.4), the increment H_k is equivalent to $X_{k+1} - X_k$, and thus

$$H_k X_{k+1}^{-1} = (X_{k+1} - X_k) X_{k+1}^{-1} = I - F_k.$$

Substituting this relation into Eq. (5.7) yields

$$\begin{aligned}
H_{k+1} &= -\frac{1}{p} H_k X_{k+1}^{-1} \left(\sum_{i=0}^{p-2} (i+1) F_k^i \right) H_k \\
&= -\frac{1}{p} (I - F_k) \left(\sum_{i=0}^{p-2} (i+1) F_k^i \right) H_k \\
&= -\frac{1}{p} \left[I + F_k + F_k^2 + \cdots + F_k^{p-2} - (p-1) F_k^{p-1} \right] H_k \\
&= -\frac{1}{p} \left\{ \left[-(p-1) F_k + pI \right] \left[I + F_k + F_k^2 + \cdots + F_k^{p-2} \right] - (p-1) I \right\} H_k.
\end{aligned}$$

Introducing the matrix polynomial

$$P_d(X) := I + X + X^2 + \cdots + X^d,$$

enables Eq. (5.7) to be simplified further to

$$H_{k+1} = -\frac{1}{p} \left\{ \left[-(p-1) F_k + pI \right] P_{p-2}(F_k) - (p-1) I \right\} H_k. \tag{5.8}$$

The number of matrix multiplications for Eq. (5.8) is equal to the number of matrix multiplications for $P_{p-2}(F_k)$ plus two. We now define a variant of IN iteration as

$$\begin{cases}
X_{k+1} = X_k + H_k, \; F_k = X_k X_{k+1}^{-1}, \\
H_{k+1} = -\frac{1}{p} \left\{ \left[-(p-1) F_k + pI \right] P_{p-2}(F_k) - (p-1) I \right\} H_k.
\end{cases} \tag{5.9}$$

This new expression motivates us to reduce the number of matrix multiplications for computing $P_{p-2}(F_k)$.

Furthermore, this variant (5.9) is as stable as original IN iteration (5.4). We use the following definition of stability to analyze the variant (5.9).

Definition 5.1 ([39, Definition 4.17]). *Consider an iteration $X_{k+1} = g(X_k)$ with a fixed point X. Assume that g is Fréchet differentiable at X. The iteration is stable in a neighborhood of X if the Fréchet derivative $L_g(X)$ has bounded powers, that is, there exists a constant c such that $\|L_g^i(X)\| \le c$ for all $i > 0$.*

In Definition 5.1, the norm of $L(X)$ is defined as $\|L(X)\| := \max_{E\neq O} \|L(X,E)\|/\|E\|$ [39, p. 56], and $L_g^i(X)$ is ith power of the Fréchet derivative L at X, defined as i-fold composition [39, p. 97]. For example, $L^3(X,E) = L(X, L(X, L(X,E)))$. Then, we show that the variant (5.9) is stable.

Proposition 5.2. *The variant* (5.9) *is stable.*

Proof. The iteration function for the variant (5.9) is

$$G\left(\begin{bmatrix} X \\ H \end{bmatrix}\right) = \begin{bmatrix} X+H \\ -\frac{1}{p}\left\{[-(p-1)F + pI]P_{p-2}(F) - (p-1)I\right\}H \end{bmatrix} \quad \left(F = X(X+H)^{-1}\right) \tag{5.10}$$

and the fixed point is $\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}$. In order to calculate the Fréchet derivative of G at $\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}$, we calculate $G\left(\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}\right)$ and $G\left(\begin{bmatrix} A^{1/p}+E_X \\ O+E_H \end{bmatrix}\right)$, where $\|E_X\|$ and $\|E_H\|$ are sufficiently small. Substituting $X = A^{1/p}$ and $H = O$ into Eq. (5.10),

$$G\left(\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}\right) = \begin{bmatrix} A^{1/p} \\ -\frac{1}{p}\left\{[-(p-1)I + pI]\left[\sum_{n=0}^{p-2} I\right] - (p-1)I\right\}O \end{bmatrix} = \begin{bmatrix} A^{1/p} \\ 0 \end{bmatrix}, \tag{5.11}$$

and substituting $X = A^{1/p} + E_X$ and $H = O + E_H$ into Eq. (5.10),

$$\begin{aligned} G\left(\begin{bmatrix} A^{1/p}+E_X \\ O+E_H \end{bmatrix}\right) &= \begin{bmatrix} A^{1/p}+E_X+E_H \\ -\frac{1}{p}\left\{[-(p-1)F_\Delta + pI]\left[\sum_{i=0}^{p-2} F_\Lambda^i\right] - (p-1)I\right\}E_H \end{bmatrix} \\ &\qquad \left(F_\Delta = (A^{1/p}+E_X)(A^{1/p}+E_X+E_H)^{-1}\right) \\ &= \begin{bmatrix} A^{1/p}+E_X+E_H \\ -\frac{1}{p}\left[I + F_\Delta + F_\Delta^2 + \cdots + F_\Delta^{p-2} - (p-1)F_\Delta^{p-1}\right]E_H \end{bmatrix}. \end{aligned} \tag{5.12}$$

Since $\|E_X\|$ and $\|E_H\|$ are sufficiently small, F_Δ becomes

$$\begin{aligned} F_\Delta &= (A^{1/p}+E_X)(A^{1/p}+E_X+E_H)^{-1} \\ &= \left[A^{1/p}+E_X\right]\left[A^{-1/p} - A^{-1/p}(E_X+E_H)A^{-1/p} + \mathcal{O}\left(\|E_X+E_H\|^2\right)\right] \\ &= I - E_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\,\|E_H\|). \end{aligned} \tag{5.13}$$

Using Eq. (5.13), F_Δ^i becomes

$$\begin{aligned} F_\Delta^i &= \left(I - E_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\,\|E_H\|)\right)^i \\ &= I - iE_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\,\|E_H\|). \end{aligned}$$

Therefore, the lower part of (5.12) can be rewritten as

$$
\begin{aligned}
&-\frac{1}{p}\left[I + F_\Delta + F_\Delta^2 + \cdots + F_\Delta^{p-2} - (p-1)F_\Delta^{p-1}\right] E_H \\
&\quad = -\frac{1}{p}\Big[I + (I - E_H A^{-1/p}) + (I - 2E_H A^{-1/p}) + \cdots + (I - (p-2)E_H A^{-1/p}) \\
&\qquad\qquad - (p-1)(I - (p-1)E_H A^{-1/p}) + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\,\|E_H\|)\Big] E_H \\
&\quad = -\frac{1}{p}\left[\frac{p(p-1)}{2} E_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\,\|E_H\|)\right] E_H \\
&\quad = \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2),
\end{aligned}
$$

and we have

$$
G\left(\begin{bmatrix} A^{1/p} + E_X \\ O + E_H \end{bmatrix}\right) = \begin{bmatrix} A^{1/p} + E_X + E_H \\ \mathcal{O}\left(\|E_X\|^2\right) + \mathcal{O}\left(\|E_H\|^2\right) \end{bmatrix}. \tag{5.14}
$$

From Eq. (5.11) and Eq.(5.14), it holds that

$$
\begin{aligned}
&G\left(\begin{bmatrix} A^{1/\rho}_{O+E_H} + E_X \\ O + E_H \end{bmatrix}\right) - G\left(\begin{bmatrix} A^{1/\rho} \\ O \end{bmatrix}\right) - \begin{bmatrix} I & I \\ O & O \end{bmatrix}\begin{bmatrix} E_X \\ E_H \end{bmatrix} = \begin{bmatrix} O \\ \mathcal{O}\left(|E_X|^2\right) + \mathcal{O}\left(|E_H|^2\right) \end{bmatrix} \\
&\quad = o\left(\left\|\begin{bmatrix} E_X \\ E_H \end{bmatrix}\right\|\right),
\end{aligned}
$$

and we obtain

$$
L_G\left(\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}, \begin{bmatrix} E_X \\ E_H \end{bmatrix}\right) = \begin{bmatrix} I & I \\ O & O \end{bmatrix}\begin{bmatrix} E_X \\ E_H \end{bmatrix}.
$$

The matrix $\left[\begin{smallmatrix} I & I \\ O & O \end{smallmatrix}\right]$ is idempotent because

$$
\begin{bmatrix} I & I \\ O & O \end{bmatrix}^2 = \begin{bmatrix} I & I \\ O & O \end{bmatrix}.
$$

Then, for all $i > 0$, $\left\|L_G^i\left(\left[\begin{smallmatrix} A^{1/p} \\ O \end{smallmatrix}\right]\right)\right\|$ is bounded. From the above, the variant (5.9) is stable.[2] □

In the next subsection, we provide a means of reducing matrix multiplications of $P_{p-2}(F_k)$.

5.2.2 Decomposition of the polynomial.

If $d \geq 3$, the matrix polynomial $P_d(X)$ can be rewritten in a more efficient form:

$$
P_d(X) = \begin{cases} P_{\frac{d-1}{2}}(X^2) \cdot (X + I) & (d \text{ is odd}) \\ P_{\frac{d-2}{2}}(X^2) \cdot (X^2 + X) + I & (d \text{ is even}). \end{cases} \tag{5.15}
$$

On the right-hand side of Eq. (5.15), there is a new matrix polynomial whose variable is X^2 and degree is approximately half of d. This decomposition reduces the number of matrix multiplications by almost a factor of two. Thus, the number of matrix multiplications of $P_d(X)$ is reduced by applying the decomposition (5.15) to $P_d(X)$ repeatedly.

[2] The stability of IN iteration (5.4) can be proved in a similar manner.

Let us show the example of $d = 57$.[3]

$$
\begin{aligned}
P_{57}(X) &= I + X + X^2 + \cdots + X^{57} && (5.16)\\
&= \left\{ P_{28}\left(X^2\right) \right\} \{X + I\}\\
&= \left\{ P_{13}\left(X^4\right)\left(X^4 + X^2\right) + I \right\} \{X + I\}\\
&= \left\{ P_{6}\left(X^8\right)\left(X^4 + I\right)\left(X^4 + X^2\right) + I \right\} \{X + I\}\\
&= \left\{ \left[\left(X^{32} + X^{16} + I\right)\left(X^{16} + X^8\right) + I\right]\left[X^4 + I\right]\left[X^4 + X^2\right] + I \right\} \{X + I\}. && (5.17)
\end{aligned}
$$

In this example, $P_{57}(X)$ of Eq. (5.16) is computed using 56 matrix multiplications by naive implementation. On the other hand, after applying the decomposition (5.15) to Eq. (5.16) four times, Eq. (5.17) can be computed with nine matrix multiplications. In detail, five matrix multiplications are required for constructing five intermediate matrices, X^2, X^4, X^8, X^{16}, and X^{32}, and another four matrix multiplications are required for multiplication of the subpolynomials.

Finally, we combine variant (5.9) with decomposition (5.15) into Algorithm 9 for practice.

Algorithm 9 Newton's method for $A^{1/p}$ with the variant of IN iteration

Input $A \in \mathbb{C}^{n\times n}$, $p \in \mathbb{N}$

Decompose P_{p-2} by applying the decomposition (5.15) repeatedly.

$X_0 = I, \quad H_0 = (A - I)/p$

for $k = 0, 1, 2 \ldots$ until convergence **do**

  $X_{k+1} = X_k + H_k$

  $F_k = X_k X_{k+1}^{-1}$

  Compute $P_{p-2}(F_k)$

  $H_{k+1} = -\frac{1}{p}\left\{ \left[-(p-1)F_k + pI\right] P_{p-2}\left(F_k\right) - (p-1)I \right\} H_k$

end for

Output $X_k \approx A^{1/p}$

5.2.3 Estimation of the computational cost of the variant

We calculated the computational cost of the variant (5.9) for $p \in [5, 100]$ numerically and found that cost to be consistent with $(2\left\lfloor 2\log_2(p-1)\right\rfloor + 8/3)n^3$. Here, the computational cost $8/3n^3$ results from the computation of $F_k (= X_k X_{k+1}^{-1})$ by using the LU decomposition of X_{k+1}. While a proof that the cost of the variant (5.9) is $\mathcal{O}(n^3 \log p)$ flops per iteration is left for future work, this numerical result agrees with that expectation. In addition, we calculated the costs of IN iteration (5.4) and the iteration (5.5) for $p \in [5, 100]$ to compare them with that of variant (5.9). The result is shown in Fig. 5.1. It is clear from the figure that the computational cost of the variant (5.9) is lower than that of IN iteration (5.4) and competitive with that of iteration (5.5). For example, when $d = 59$, the computational cost of variant (5.9) is approximately a quarter of that of IN iteration (5.4) and slightly higher than that of iteration (5.5).

[3]The polynomial $P_{57}(F_k)$ appears when calculating the matrix 59th root.

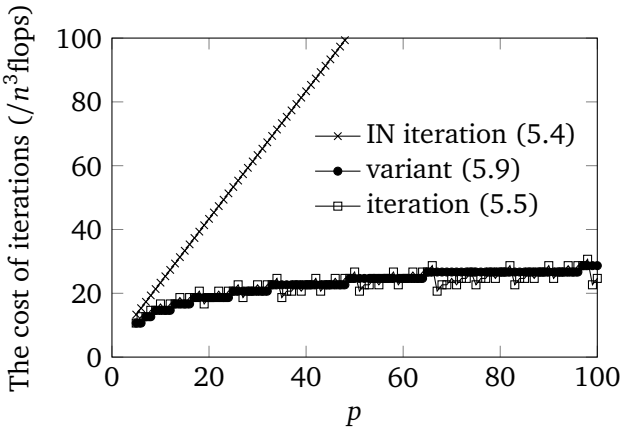


Figure 5.1: The computational costs per iteration for the three iterations

5.3 Numerical experiment

This section describes a numerical experiment in which the principal 59th roots of test matrices are calculated. The test matrices are described in Table 5.1. First, we precondi-

Table 5.1: Test matrices used for numerical experiments of Chapter 5.

	Matrix A	n	Non-zero elements	$\kappa(A)$	Property
1	msc01440 [19]	1440	44998	3.3×10^6	SPD
2	Random matrix	1500	2250000	3.8×10^2	SPD
3	NNC1374 [19]	1374	8606	3.7×10^{14}	Nonsymmetric

tioned the test matrices to satisfy the sufficient condition (5.3) of global convergence in section 1: all eigenvalues of A lie in the set $\{z \in \mathbb{C} : \mathrm{Re}z > 0, |z| \leq 1\}$. Thus, we computed $\tilde{A} = A^{1/2}/\|A^{1/2}\|_F$. Then, we computed $\tilde{A}^{1/59}$ by IN iteration (5.4), variant (5.9) of Algorithm 9, and iteration (5.5). The computational costs of these three iterations are shown in Table 5.2. For this experiment, Python 3.5 was used for programming, and Intel(R) Core™ i7 2.8GHz

Table 5.2: Computational costs for computing the principal 59th root.

Iteration	Computational costs per iteration (flops)
IN iteration (5.4)	$(118 + 10/3)n^3 + \mathcal{O}(n^2)$
variant (5.9)	$(22 + 8/3)n^3 + \mathcal{O}(n^2)$
iteration (5.5)	$(20 + 8/3)n^3 + \mathcal{O}(n^2)$

CPU and 8GB RAM were used for computation.

First, Figure 5.2 shows the ratios of computation time of these three iterations. From Fig. 5.2, the computation time of variant (5.9) is approximately one fourth of that of IN iteration (5.4) and slightly longer than that of (5.5) in all cases. Here it can be seen that both the computation time and the computational cost decreased. Next, Figure 5.3 shows the relative residual defined as $R(X) = \|X^p - A\|_F / \|A\|_F$ for these three iterations. The figure shows that the convergence behavior of variant (5.9) differs little from that of IN iteration (5.4) and iteration (5.5). Since there is some possibility of numerical cancellation of variant (5.9), IN iteration (5.4) is slightly better than variant (5.9) in terms of accuracy.

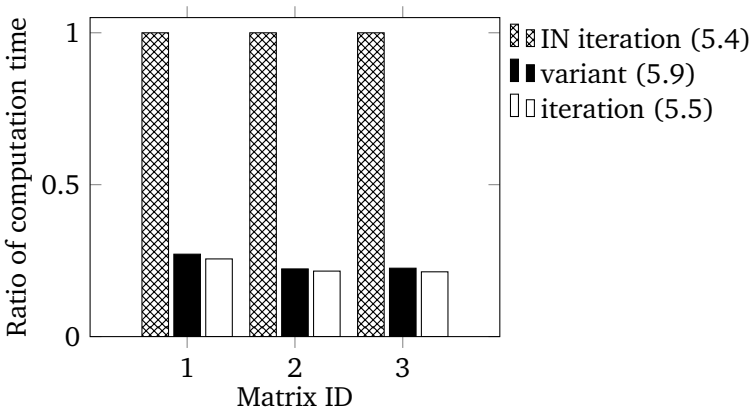
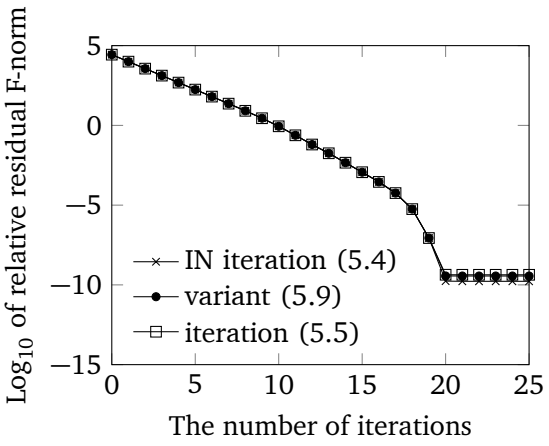
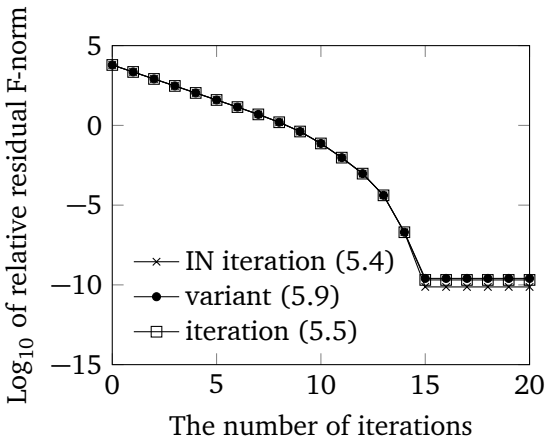


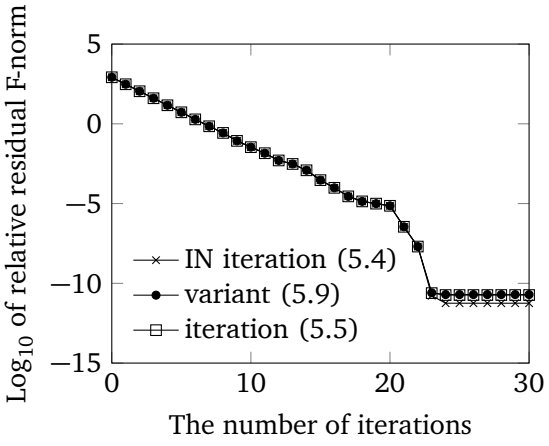
Figure 5.2: Time comparison of the three iterations



(a) Residual Plot of msc01440 (1)



(b) Residual plot of the random matrix (2)



(c) Residual plot of NNC1374 (3)

Figure 5.3: Convergence histories of the three iterations.

5.4 Conclusion

In this chapter, a variant of the IN iteration was proposed whose computational cost well agreed with $\mathcal{O}(n^3 \log p)$ flops per iteration if p is up to at least 100, and whose increment part still has the form $H_{k+1} = f_k(H_k)$. We have learned from the results of the numerical experiment that the variant is competitive with iteration (5.5) in terms of accuracy and computation time. The proposed variant therefore becomes a choice for practical application.

Our future work will focus on three problems. The first one is to prove that the computational cost of the variant is $\mathcal{O}(n^3 \log p)$. The second one is to develop an algorithm which automatically decomposes the polynomial P_{p-2} for given p. The third one is reducing the computation time of Newton's method for the principal matrix pth root by reducing the number of iterations. However, it is not clear how to choose a better initial guess than the conventional initial guess I.

Chapter 6

Conclusion

The computation of matrix functions has been developing because of the great efforts of many contributors. In this thesis, several algorithms to enhance the computation of matrix functions are proposed. We briefly summarize the results of each chapter and refer to future work.

Summary

Chapter 3

We considered the computation of the matrix logarithm $\log(A)$ via numerical quadrature. Since the computational cost largely depends on the number of abscissas, it is important to reduce the number of abscissas. In previous studies, the GL quadrature is used for numerical quadrature. Though the convergence of the GL quadrature is fast when $A \approx I$, the convergence is slow when A is far away from I, e.g., when $\kappa(A)$ is large.

In order to avoid slow convergence, we first considered applying the DE formula for $\log(A)$. The difficulty in utilizing the DE formula is the truncation of the infinite integral interval which affects the accuracy. In this study, we analyzed the error stemmed from the truncation and proposed a method for determining a finite interval so that the truncation error is smaller than or equal to a given tolerance. After that, we analyzed the convergence of the DE formula when A is an SPD matrix. We found that the DE formula convergence faster than the GL quadrature when $\kappa(A) \gtrsim 3.7 \times 10^3$. In addition, we considered preconditioning of numerical quadrature for an SPD matrix A based on the relation $\log(A) = \log(AP) - \log(P)$. We proposed a choice of $P = (A + \sqrt{\lambda_{\max}\lambda_{\min}}I)^{-1}$ so that $\kappa(AP)$ and $\kappa(P)$ are smaller than $\kappa(A)$ and we can compute $\log(AP)$ and $\log(P)$ quickly. The estimate showed the preconditioning accelerate the GL quadrature when $1.3 \times 10^2 \leq \kappa(A) \leq 2.5 \times 10^6$ and the DE quadrature when $\kappa(A) > 1.0 \times 10^{14}$. Numerical experiments supported our discussions.

Chapter 4

We considered the computation of the matrix fractional power A^α via numerical quadrature. The difficulties on numerical quadrature are the integral representation of A^α is improper integral, and the value of α affected the efficiency of numerical quadrature as well as the properties of A. In the previous studies, A^α is computed with the GJ quadrature after applying some variable transformations; however, when

$\kappa(A)$ is large and α is not reciprocal of natural numbers, the convergence of the GJ quadrature becomes slow.

In this study, we first considered applying the DE formula for A^α as done in Chapter 3. We analyzed the truncation error and proposed an algorithm for practical computation. We also analyzed the convergence rate of the DE formula for an SPD matrix A. We found that the DE formula converges faster than the GJ quadrature when $\kappa(A)$ is large and α is not reciprocal of natural numbers. For example, the DE formula converges faster than the GJ quadrature when $\kappa(A) \geq 1.6 \times 10^4$ and $\alpha = 0.8$. After that, we consider preconditioning for numerical quadrature when A is an SPD based on the relation $A^\alpha = (AP)^\alpha P^{-\alpha}$. We show a choice of P, and estimate the effect of preconditioning as done in Chapter 3. Numerical results support our discussions.

Chapter 5

In Section 5, we considered Newton's method for the matrix pth root. The IN iteration is a numerically-stable variant of Newton's iteration for $A^{1/p}$, and its computational cost is $\mathcal{O}(n^3 p)$. In this study, we proposed a cost-efficient variant of IN iteration by rewriting the original iteration so that the number of matrix-matrix multiplication is reduced. We showed that our variant iteration is also numerical stable and the computational cost is $\mathcal{O}(n^3 \log(p))$ when $p \leq 100$. Numerical results showed that the computation time is reduced as well as the reduced computational cost, and our iteration is stable.

Future work

In this thesis, we proposed several algorithms for the computation of matrix functions. We expect that these algorithms will be improved by further study. For example,

- a sharp lower bound on $\|\log(A)\|_2$ reduces the computational cost of the DE formula for $\log(A)$,
- developing a more efficient quadrature formula reduces the computational cost of the quadrature-based algorithms,
- analysis of the convergence rate of the quadrature-based algorithms for general matrix and analysis of absolute error of the quadrature-based algorithms allows us to choose more appropriate quadrature formula for practical computation,
- an automatic decomposition algorithm of the matrix polynomial $I + A + A^2 + \cdots + A^d$ simplifies writing a code of our variant of Newton's iteration for $A^{1/p}$.

An interesting topic of quadrature-based algorithms for $f(A)\boldsymbol{b}$ is methods for computing liner systems in the integrands. The computation of the integrand on each abscissa comes down to solve shifted linear systems. If we select a suitable method for solving the shifted linear systems, we can save computation time and resources.

It is also worth working on customizing the algorithms computing matrix functions for practical problems.

Bibliography

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel. Implementation of a restarted Krylov subspace method for the evaluation of matrix functions. *Linear Algebra Appl.*, 429(10):2293–2314, 2008.

[2] A. H. Al-Mohy and N. J. Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 31(3):970–989, 2009.

[3] A. H. Al-Mohy and N. J. Higham. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM J. Sci. Comput.*, 34(4):C153–C169, 2012.

[4] A. H. Al-Mohy, N. J. Higham, and S. D. Relton. New algorithms for computing the matrix sine and cosine separately or simultaneously. *SIAM J. Sci. Comput.*, 37(1):A456–A487, 2015.

[5] B. Andrews and C. Hopper. *The Ricci flow in Riemannian geometry*, volume 2011 of *Lecture Notes in Mathematics*. Springer, Heidelberg, 2011.

[6] S. Aoki, M. Fukugita, S. Hashimoto, K. I. Ishikawa, N. Ishizuka, Y. Iwasaki, K. Kanaya, T. Kaneko, Y. Kuramashi, M. Okawa, N. Tsutsui, A. Ukawa, N. Yamada, and T. Yoshié. An exact algorithm for any-flavor lattice QCD with Kogut–Susskind fermion. *Computer Physics Communications*, 155(3):183–208, Nov. 2003.

[7] M. Aprahamian and N. J. Higham. The matrix unwinding function, with an application to computing the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 35(1):88–109, 2014.

[8] Å. Björck and S. Hammarling. A Schur method for the square root of a matrix. *Linear Algebra Appl.*, 52/53:127–140, 1983.

[9] K. Burrage, N. Hale, and D. Kay. An efficient implicit FEM scheme for fractional-in-space reaction-diffusion equations. *SIAM J. Sci. Comput.*, 34(4):A2145–A2172, 2012.

[10] J. R. Cardoso. Computation of the matrix pth root and its Fréchet derivative by integrals. *Electron. Trans. Numer. Anal.*, 39:414–436, 2012.

[11] J. R. Cardoso and R. Ralha. Matrix arithmetic-geometric mean and the computation of the logarithm. *SIAM J. Matrix Anal. Appl.*, 37(2):719–743, 2016.

[12] J. R. Cardoso and A. Sadeghi. Computation of matrix gamma function. *BIT*, 59(2):343–370, 2019.

[13] A. Cayley. II. A memoir on the theory of matrices. *Phil. Trans. R. Soc.*, 148:17–37, Jan. 1858.

[14] J. Chen and E. Chow. A Stable Scaling of Newton-Schulz for Improving the Sign Function Computation of a Hermitian Matrix. 2014.

[15] A. Cordero, F. Soleymani, J. R. Torregrosa, and M. Z. Ullah. Numerically stable improved Chebyshev-Halley type schemes for matrix sign function. *J. Comput. Appl. Math.*, 318:189–198, 2017.

[16] P. I. Davies and N. J. Higham. A Schur-Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.*, 25(2):464–485, 2003.

[17] P. I. Davies and N. J. Higham. Computing $f(A)b$ for matrix functions f. In *QCD and numerical analysis III*, volume 47 of *Lect. Notes Comput. Sci. Eng.*, pages 15–24. Springer, Berlin, 2005.

[18] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Computer Science and Applied Mathematics. Academic Press, Inc., Orlando, FL, second edition, 1984.

[19] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):Art. 1, 25, 2011.

[20] E. Deadman, N. J. Higham, and R. Ralha. Blocked Schur Algorithms for Computing the Matrix Square Root. In P. Manninen and P. Öster, editors, *Applied Parallel and Scientific Computing*, pages 171–182, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[21] E. D. Denman and A. N. Beavers, Jr. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2(1):63–94, 1976.

[22] L. Dieci, B. Morini, and A. Papini. Computational techniques for real logarithms of matrices. *SIAM J. Matrix Anal. Appl.*, 17(3):570–593, 1996.

[23] K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson. Scalable Log Determinants for Gaussian Process Kernel Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6327–6337. Curran Associates, Inc., 2017.

[24] M. Fasi. *Computing matrix functions in arbitrary precision arithmetic*. PhD thesis, The University of Manchester, 2019.

[25] M. Fasi and N. J. Higham. Multiprecision algorithms for computing the matrix logarithm. *SIAM J. Matrix Anal. Appl.*, 39(1):472–491, 2018.

[26] M. Fasi and N. J. Higham. An arbitrary precision scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 40(4):1233–1256, 2019.

[27] M. Fasi, N. J. Higham, and B. Iannazzo. An algorithm for the matrix Lambert W function. *SIAM J. Matrix Anal. Appl.*, 36(2):669–685, 2015.

[28] M. Fasi and B. Iannazzo. Computing the weighted geometric mean of two large-scale matrices and its inverse times a vector. *SIAM J. Matrix Anal. Appl.*, 39(1):178–203, 2018.

[29] A. Frommer, S. Güttel, and M. Schweitzer. Efficient and stable Arnoldi restarts for matrix functions based on quadrature. *SIAM J. Matrix Anal. Appl.*, 35(2):661–683, 2014.

[30] A. Frommer and B. Hashemi. Verified computation of square roots of a matrix. *SIAM J. Matrix Anal. Appl.*, 31(3):1279–1302, 2009.

[31] C.-H. Guo and N. J. Higham. A Schur-Newton method for the matrix pth root and its inverse. *SIAM J. Matrix Anal. Appl.*, 28(3):788–804, 2006.

[32] V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned Stochastic Tensor Optimization. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1842–1850, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR.

[33] S. Güttel and Y. Nakatsukasa. Scaled and squared subdiagonal Padé approximation for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 37(1):145–170, 2016.

[34] N. Hale, N. J. Higham, and L. N. Trefethen. Computing $\mathbf{A}^\alpha$, $\log(\mathbf{A})$, and related matrix functions by contour integrals. *SIAM J. Numer. Anal.*, 46(5):2505–2523, 2008.

[35] I. Han, D. Malioutov, and J. Shin. Large-scale log-determinant computation through stochastic Chebyshev expansions. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 908–917, Lille, France, July 2015. PMLR.

[36] A. Heßelmann and A. Görling. Random phase approximation correlation energies with exact Kohn–Sham exchange. *Molecular Physics*, 108(3-4):359–372, Feb. 2010.

[37] N. J. Higham. Newton's method for the matrix square root. *Math. Comp.*, 46(174):537–549, 1986.

[38] N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005.

[39] N. J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, PA, Jan. 2008.

[40] N. J. Higham and E. Deadman. A Catalogue of Software for Matrix Functions. Version 2.0. Jan. 2016.

[41] N. J. Higham and L. Lin. A Schur-Padé algorithm for fractional powers of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(3):1056–1078, 2011.

[42] N. J. Higham and L. Lin. An improved Schur-Padé algorithm for fractional powers of a matrix and their Fréchet derivatives. *SIAM J. Matrix Anal. Appl.*, 34(3):1341–1360, 2013.

[43] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numer.*, 19:209–286, 2010.

[44] Z. Huang and L. V. Gool. A Riemannian Network for SPD Matrix Learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 2036–2042, Feb. 2017.

[45] B. Iannazzo. On the Newton method for the matrix pth root. *SIAM J. Matrix Anal. Appl.*, 28(2):503–523, 2006.

[46] B. Iannazzo. A family of rational iterations and its application to the computation of the matrix pth root. *SIAM J. Matrix Anal. Appl.*, 30(4):1445–1462, Sept. 2008.

[47] B. Iannazzo and C. Manasse. A Schur logarithmic algorithm for fractional powers of matrices. *SIAM J. Matrix Anal. Appl.*, 34(2):794–813, 2013.

[48] M. Ilic, F. Liu, I. Turner, and V. Anh. Numerical approximation of a fractional-in-space diffusion equation. I. *Fract. Calc. Appl. Anal.*, 8(3):323–341, 2005.

[49] C. Kenney and A. J. Laub. Condition estimates for matrix functions. *SIAM J. Matrix Anal. Appl.*, 10(2):191–209, 1989.

[50] S. Miyajima. Fast verified computation for the matrix principal pth root. *J. Comput. Appl. Math.*, 330:276–288, 2018.

[51] S. Miyajima. Verified computation for the matrix Lambert W function. *Appl. Math. Comput.*, 362:124555, 15, 2019.

[52] S. Miyajima. Verified computation for the matrix principal logarithm. *Linear Algebra Appl.*, 569:38–61, 2019.

[53] S. Miyajima. Verified computation of the matrix exponential. *Adv. Comput. Math.*, 45(1):137–152, 2019.

[54] S. Nakamura, K. Ozawa, and C. Hirota. Scaling and modified squaring method for the matrix exponential. *JSIAM Lett.*, 8:65–68, 2016.

[55] R. F. Rinehart. The equivalence of definitions of a matric function. *Amer. Math. Monthly*, 62:395–414, 1955.

[56] J. Sastre, J. Ibáñez, P. Alonso, J. Peinado, and E. Defez. Two algorithms for computing the matrix cosine function. *Appl. Math. Comput.*, 312:66–77, 2017.

[57] T. Schenk, I. M. Richardson, M. Kraska, and S. Ohnimus. Modeling buckling distortion of DP600 overlap joints due to gas metal arc welding and the influence of the mesh density. *Computational Materials Science*, 46(4):977–986, Oct. 2009.

[58] M. I. Smith. A Schur algorithm for computing matrix pth roots. *SIAM J. Matrix Anal. Appl.*, 24(4):971–989, 2003.

[59] M. Sugihara. Optimality of the double exponential formula—functional analysis approach. *Numer. Math.*, 75(3):379–395, 1997.

[60] H. Takahasi and M. Mori. Double exponential formulas for numerical integration. *Publ. Res. Inst. Math. Sci.*, 9:721–741, 1973.

[61] F. Tatsuoka, T. Sogabe, Y. Miyatake, and S. Zhang. A cost-efficient variant of the incremental Newton iteration for the matrix pth root. *J. Math. Res. Appl.*, 37(1):97–106, 2017.

[62] F. Tatsuoka, T. Sogabe, Y. Miyatake, and S.-L. Zhang. A Note on Computing the Matrix Fractional Power Using the Double Exponential Formula (in Japanese). *Trans. Jpn. Soc. Ind. Appl. Math.*, 28(3):142–161, 2018.

[63] F. Tatsuoka, T. Sogabe, Y. Miyatake, and S.-L. Zhang. Algorithms for the computation of the matrix logarithm based on the double exponential formula. *J. Comput. Appl. Math.*, page 112396 (doi: 10.1016/j.cam.2019.112396), Aug. 2019.

[64] 立岡文理, 曽我部知広, 張紹良. 数値積分に基づく行列実数乗の計算. 計算数理工学レビュー, 2019-2:45–55, 2019.

[65] L. N. Trefethen and J. A. C. Weideman. The exponentially convergent trapezoidal rule. *SIAM Rev.*, 56(3):385–458, 2014.

[66] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv:1907.10121*, July 2019.

[67] A. Wehrl. General properties of entropy. *Rev. Mod. Phys.*, 50(2):221–260, Apr. 1978.

[68] T. Yamamoto. A method for finding sharp error bounds for Newton's method under the Kantorovich assumptions. *Numer. Math.*, 49(2-3):203–220, 1986.

[69] W. Zhang and N. J. Higham. Matrix Depot: an extensible test matrix collection for Julia. *PeerJ Comput. Sci.*, 2:e58, Apr. 2016.