Numerische Mathematik (2024) 156:2179–2219
https://doi.org/10.1007/s00211-024-01441-8

Numerische Mathematik

Complex matrix inversion via real matrix inversions

Zhen Dai[1] · Lek-Heng Lim[1] · Ke Ye[2]

Received: 9 October 2023 / Revised: 10 September 2024 / Accepted: 7 October 2024 /
Published online: 5 November 2024

Abstract

We study the inversion analog of the well-known Gauss algorithm for multiplying complex matrices. A simple version is $(A+iB)^{-1} = (A+BA^{-1}B)^{-1} - iA^{-1}B(A+BA^{-1}B)^{-1}$ when A is invertible, which may be traced back to Frobenius but has received scant attention. We prove that it is optimal, requiring fewest matrix multiplications and inversions over the base field, and we extend it in three ways: (i) to any invertible $A+iB$ without requiring A or B be invertible; (ii) to any iterated quadratic extension fields, with $\mathbb{C}$ over $\mathbb{R}$ a special case; (iii) to Hermitian positive definite matrices $A+iB$ by exploiting symmetric positive definiteness of A and $A+BA^{-1}B$. We call all such algorithms Frobenius inversions, which we will see do not follow from Sherman–Morrison–Woodbury type identities and cannot be extended to Moore–Penrose pseudoinverse. We show that a complex matrix with well-conditioned real and imaginary parts can be arbitrarily ill-conditioned, a situation tailor-made for Frobenius inversion. We prove that Frobenius inversion for complex matrices is faster than standard inversion by LU decomposition and Frobenius inversion for Hermitian positive definite matrices is faster than standard inversion by Cholesky decomposition. We provide extensive numerical experiments, applying Frobenius inversion to solve linear systems, evaluate matrix sign function, solve Sylvester equation, and compute polar decomposition, showing that Frobenius inversion can be more efficient than LU/Cholesky decomposition with negligible loss in accuracy. A side result is a generalization of Gauss multiplication to iterated quadratic extensions, which we show is intimately related to the Karatsuba algorithm for fast integer multiplication and multidimensional fast Fourier transform.

✉ Lek-Heng Lim
lekheng@uchicago.edu

Zhen Dai
zhen9@uchicago.edu

Ke Ye
keyk@amss.ac.cn

1 Computational and Applied Mathematics Initiative, University of Chicago, Chicago, IL 60637-1514, USA

2 KLMM, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

Mathematics Subject Classification 65F05 · 15B33 · 68W30

1 Introduction

The article is a sequel to our recent work in [16], where we studied the celebrated Gauss multiplication algorithm $(A+iB)(C+iD) = (AC-BD)+i[(A+B)(C+D)-AC-BD]$ for multiplying a pair of complex matrices with just three real matrix multiplications. Such methods for performing a complex matrix operation in terms of real matrix operations can be very useful as floating point standards such as the IEEE-754 [41] often do not implement complex arithmetic natively but rely on software to reduce complex arithmetic to real arithmetic [59, p. 55]. Here we will analyze and extend an inversion analogue of Gauss algorithm: Given a complex invertible matrix $A+iB \in \mathbb{C}^{n\times n}$ with $A, B \in \mathbb{R}^{n\times n}$, it is straightforward to verify that its inverse is given by

$$(A+iB)^{-1} = (A+BA^{-1}B)^{-1} - iA^{-1}B(A+BA^{-1}B)^{-1} \tag{1}$$

if A is invertible, a formula that can be traced back to Georg Frobenius [72]. In our article we will refer to all such algorithms and their variants and extensions as *Frobenius inversions*. While Gauss multiplication has been thoroughly studied (two representative references are [45, Section 4.6.4] in Computer Science and [37, Section 23.2.4] in Numerical Analysis, with numerous additional references therein), the same cannot be said of Frobenius inversion — we combed through the research literature and found only six references, all from the 1970s or earlier, which we will review in Sect. 1.3.

Our goal is to vastly extend and thoroughly analyze Frobenius inversion from a modern perspective. We will extend it to the general case where only $A+iB$ is invertible but neither A nor B is (Sect. 4.2), and to the important special case where $A+iB$ is Hermitian positive definite, in a way that exploits the symmetric positive definiteness of A and $A+BA^{-1}B$ (Sect. 4.3). We will show (Sect. 3) that it is easy to find complex matrices $A+iB$ with

$$\max\bigl(\kappa_2(A), \kappa_2(B), \kappa_2(A+BA^{-1}B)\bigr) \ll \kappa_2(A+iB), \tag{2}$$

where the gap between the left- and right-hand side is arbitrarily large, i.e., $A+iB$ can be arbitrarily ill-conditioned even when A, B, $A+BA^{-1}B$ are all well-conditioned — a scenario bespoke for (1).

Frobenius inversion obviously extends to any quadratic fields of the form $\Bbbk[\sqrt{a}]$, i.e., x^2+a is irreducible over $\Bbbk$, but we will further extend it to any arbitrary quadratic field, and any iterated quadratic extensions including constructible numbers, multi-quadratics, and towers of root extensions (Sect. 2.4). In fact we show that for iterated quadratic extensions, Frobenius inversion essentially gives the multidimensional fast Fourier transform. We will prove that over any quadratic field Frobenius inversion is optimal in that it requires the least number of matrix multiplications and inversions over its base field (Sects. 2.3, 4.3, and 4.2).

For complex matrix inversion, we show that MATLAB's built-in inversion algorithm, i.e., directly inverting a matrix with LU or Cholesky decomposition *in complex arithmetic*, is slower than applying Frobenius inversion with LU or Cholesky decomposition *in real arithmetic* (Theorem 4.1, Propositions 4.2 and 4.6). More importantly, we provide a series of numerical experiments in Sect. 5 to show that Frobenius inversion is indeed faster than MATLAB's built-in inversion algorithm in almost every situation and, despite well-known exhortations to avoid matrix inversion, suffers from no significant loss in accuracy. In fact methods based on Frobenius inversion may be more accurate than standard methods in certain scenarios (Sect. 5.2).

1.1 Why not invert matrices

Matrix inversion is frown upon in numerical linear algebra, likely an important cause for the lack of interest in algorithms like Frobenius inversion. The usual reason for eschewing inversion [37] is that in solving an $n \times n$ nonsingular system $Ax = b$, if we compute a solution $\widehat{x}_{\mathsf{inv}}$ by inverting A through LU factorization $PA = LU$ and multiplying A^{-1} to b, and if we compute a solution $\widehat{x}_{LU}$ directly through the LU factors with backward substitutions $Ly = Pb$, $Ux = y$, the latter approach is both faster, with $2n^3$ flops for $\widehat{x}_{\mathsf{inv}}$ versus $2n^3/3$ for $\widehat{x}_{LU}$, and more accurate, with backward errors

$$|b - A\widehat{x}_{\mathsf{inv}}| \le n|A||A^{-1}||b|\mathsf{u} + O(\mathsf{u}^2) \quad \text{versus} \quad |b - A\widehat{x}_{LU}| \le 3n|\widehat{L}||\widehat{U}||\widehat{x}_{LU}|\mathsf{u} + O(\mathsf{u}^2). \tag{3}$$

Here u denotes unit roundoff and where $|\cdot|$ and $\le$ applies componentwise. As noted in [37], usually $\||\widehat{L}||\widehat{U}|\|_\infty \approx \|A\|_\infty$ and so $\widehat{x}_{LU}$ is likely more accurate than $\widehat{x}_{\mathsf{inv}}$ when $\|x\|_\infty \ll \||A^{-1}||b|\|_\infty$.

Another common rationale for avoiding inversion is the old wisdom that many tasks that appear to require inversion actually do not — an explicit inverse matrix $A^{-1} \in \mathbb{C}^{n\times n}$ is almost never required because upon careful examination, one would invariably realize that the same objective could be accomplished with a vector like $A^{-1}b$ or $\operatorname{diag}(A^{-1}) \in \mathbb{C}^n$ or a scalar like $c^{\mathsf{T}}A^{-1}b$, $\|A^{-1}\|$, or $\operatorname{tr}(A^{-1}) \in \mathbb{C}$. These vectors and scalars could be computed with a matrix factorization or approximated to arbitrary accuracy with iterative methods [30], which are often more amenable to updating/downdating [29] or better suited for preserving structures like sparsity.

Caveat: We emphasize that Frobenius inversion, when applied to solve a system of complex linear equations $(A + iB)z = c + id$, will *not* involve actually computing an explicit inverse matrix $(A + iB)^{-1}$ and then multiplying it to the vector $c + id$. In other words, we do not use the expression in (1) literally but only apply it in conjunction with various LU decompositions and back substitutions over $\mathbb{R}$; the matrix $(A + iB)^{-1}$ is never explicitly formed. The details are given in Sect. 3 alongside discussions of circumstances like (2) where the use of Frobenius inversion gives more accurate results than standard methods, with numerical evidence in Sect. 5.2.

1.2 Why invert matrices

We do not dispute the reasons in Sect. 1.1 but numerical linear algebra is a field that benefits from a wide variety of different methods for the same task, each suitable for a different regime. There is no single method that is universally best in every instance. Even the normal equation, frown upon in numerical linear algebra like matrix inversion, can be the ideal method for certain least squares problems.

In fact, if we examine the advantages of computing $\widehat{x}_{LU}$ over $\widehat{x}_{\mathsf{inv}}$ in Sect. 1.1 more closely, we will find that the conclusion is not so clear cut. Firstly, the comparison in (3) assumes that accuracy is quantified by backward error $|b - A\widehat{x}|$ but in reality it is the forward error $|x - \widehat{x}|$ that is far more important and investigations in [17, 18], both analytical and experimental, show that the forward errors of $\widehat{x}_{LU}$ and $\widehat{x}_{\mathsf{inv}}$ are similar. Secondly, if instead of solving a single linear system $Ax = b$, we have p right-hand sides $b_1, \dots, b_p \in \mathbb{C}^n$, then it becomes $AX = B$ where $B = [b_1, \dots, b_p] \in \mathbb{C}^{n\times p}$ and we seek a solution $X \in \mathbb{C}^{n\times p}$. In this case the speed advantage of computing $\widehat{X}_{LU}$ over $\widehat{X}_{\mathsf{inv}}$ disappears when $p = O(n)$: Note that the earlier flop count $2n^3/3$ for $\widehat{x}_{LU}$ ignores the cost of two backsubstitutions but when there are $2p$ backsubstitutions, these may no longer be ignored and are in fact dominant, making the cost of computing $\widehat{X}_{\mathsf{inv}}$ and $\widehat{X}_{LU}$ comparable. In [17], it is shown that because of data structure complications, computing $\widehat{X}_{\mathsf{inv}}$ can be significantly faster than $\widehat{X}_{LU}$.

Moreover, the old wisdom that one may avoid computing explicit inverse matrices, while largely true, is not always true. There are situations, some of them alluded to in [37, p. 260], where computing an explicit inverse matrix is inevitable or favorable:

Mimo radios In such radios, explicit inverse matrices are implemented in hardware [18, 21, 71]. It is straightforward to hardwire or hardcode an explicit inverse matrix but considerably more difficult to do so in the form of "LU factors with permutations and backsubstitutions," which can require more gates or code space and is more prone to implementation errors.

Superconductivity In the so-called KKR CPA algorithm [33], one needs to integrate the KKR inverse matrix over the first Brillouin zone, necessitating an explicit inverse matrix.

Linear modeling The inverse of a matrix often reveals important statistical properties that could only be discerned when one has access to the full explicit inverse [51, 52], i.e., we do not know which entries of A^{-1} matter until we see all of them. For a specific example, take the ubiquitous model $y = X\widehat{\beta} + \varepsilon$ with design matrix X and observed values $y_1, \dots, y_n$ of the dependent variable y [52], we understand the regression coefficients $\widehat{\beta}$ through the values its covariance matrix $\Sigma := \sigma^2 \cdot (X^{\mathsf{T}}X)^{-1}$ where σ^2 is the variance of the dependent variable [52]. To see which values are large (positively correlated), small (negatively correlated), or nearly zero (uncorrelated) in relation to other values, we need access to all values of Σ.

Statistics For an unbiased estimator $\widehat{\theta}(X)$ of a parameter θ, its Cramer–Rao lower bound is the inverse of its Fisher information matrix $I(\theta)$. This is an important quantity that gives a lower bound for the covariance matrix [15, 62] in the sense of $\operatorname{cov}_\theta\bigl(\widehat{\theta}(X)\bigr) \succeq I(\theta)^{-1}$ where $\succeq$ is the Loewner order. In some Gaussian processes, this lower bound could be attained [44]. We need the explicit matrix inverse $I(\theta)^{-1}$ to understand the limits of certain statistical problems and to design optimal estimators that attain the Cramer–Rao lower bound.

Graph theory The inverses of the adjacency matrix, forward adjacency matrix, and various graph Laplacians of a graph G contain important combinatorial properties about G [57, 60, 61, 76] that are only revealed when one examines all entries of their explicit inverse matrices.

Symbolic computing Matrix inversions do not just arise in numerical computing with floating point operations. They are routinely performed in finite field arithmetic over a base field of the form $\mathbb{k} = \mathrm{GF}(p^n)$ in cryptography [39, 69], combinatorics [46], information theory [2], and finite field matrix computations [11]. They are also carried out in rational arithmetic over transcendental fields [22, 23, 28], e.g., with a base field of the form $\mathbb{k} = \mathbb{Q}(x_1, \ldots, x_n, e^{x_1}, \ldots, e^{x_n})$ and an extension field of the form $\mathbb{F} = \mathbb{Q}[i](x_1, \ldots, x_n, e^{x_1}, \ldots, e^{x_n})$, or with finite fields in place of $\mathbb{Q}$ and $\mathbb{Q}[i]$. With such exact arithmetic, the considerations in Sect. 1.1 become irrelevant.

In summary, the Frobenius inversion algorithms in this article are useful (i) for problems with well-conditioned A, B, and $A + BA^{-1}B$ but ill-conditioned $A + iB$; (ii) in situations requiring an explicit inverse matrix; (iii) to applications involving exact finite field or rational arithmetic.

1.3 Previous works

We review existing works that mentioned the inversion formula (1) in the research literature: [24, 25, 66, 68, 72, 78]—we note that this is an exhaustive list, and all predate 1979. We also widened our search to books and the education literature, and found [19, 50] in engineering education publications, [8, pp. 218–219], [37, Exercise 14.8], and [47, Chapter II, Section 20], although they contain no new material.

The algorithm, according to [72], was first discovered by Frobenius and Schur although we are unable to find a published record in their Collected Works [27, 65]. Since "Schur inversion" is already used to mean something unconnected to complex matrices, and calling (1) "Frobenius–Schur inversion" might lead to unintended confusion with Schur inversion, it seems befitting to name (1) after Frobenius alone.

The discussions in [24, 66, 78] are all about deriving Frobenius inversion. From a modern perspective, the key to these derivations is an embedding of $\mathbb{C}^{n\times n}$ into $\mathbb{R}^{2n\times 2n}$ as a subalgebra via

$$A + iB \mapsto \begin{bmatrix} A & -B \\ B & A \end{bmatrix} =: M,$$

and noting that if A is invertible, then $(A+iB)^{-1}$ corresponds to M^{-1}, given by the standard expression

$$M^{-1} = \begin{bmatrix} A^{-1} - A^{-1}B(M/A)^{-1}BA^{-1} & A^{-1}B(M/A)^{-1} \\ -(M/A)^{-1}BA^{-1} & (M/A)^{-1} \end{bmatrix},$$

where $M/A := A + BA^{-1}B$ denotes the Schur complement of A in M. The two right blocks of M^{-1} then yield the expression

$$(A + iB)^{-1} = (M/A)^{-1} - iA^{-1}B(M/A)^{-1},$$

which is (1). The works in [47, 68] go further in addressing the case when both A and B are singular [47] and the case when A, B, $A + B$ or $A - B$ are all singular [68]. However, they require the inversion of a $2n \times 2n$ real matrix, wiping out any computational savings that Frobenius inversion affords. The works [25, 78] avoided this pitfall but still compromised the computational savings of Frobenius inversion. Our method in Sect. 4.2 will cover these cases and more, all while preserving the computational complexity of Frobenius inversion.

1.4 Notations and conventions

Fields are denoted in blackboard bold fonts. We write

$$\begin{aligned}
\mathrm{GL}_n(\mathbb{F}) &:= \{X \in \mathbb{F}^{n\times n} : \det(X) \neq 0\},\\
\mathrm{O}_n(\mathbb{R}) &:= \{X \in \mathbb{R}^{n\times n} : X^{\mathsf{T}}X = I\},\\
\mathrm{U}_n(\mathbb{C}) &:= \{X \in \mathbb{C}^{n\times n} : X^{\mathsf{H}}X = I\}
\end{aligned}$$

for the general linear group of invertible matrices over any field $\mathbb{F}$, the orthogonal group over $\mathbb{R}$, and the unitary group over $\mathbb{C}$ respectively. Note that we have written X^{T} for the transpose and X^{H} for conjugate transpose for any $X \in \mathbb{C}^{m\times n}$. Clearly, $X^{\mathsf{H}} = X^{\mathsf{T}}$ if $X \in \mathbb{R}^{m\times n}$. We will also adopt the convention that $X^{-\mathsf{T}} := (X^{-1})^{\mathsf{T}} = (X^{\mathsf{T}})^{-1}$ and $X^{-\mathsf{H}} := (X^{-1})^{\mathsf{H}} = (X^{\mathsf{H}})^{-1}$ for any $X \in \mathrm{GL}_n(\mathbb{C})$. Clearly, $X^{-\mathsf{H}} = X^{-\mathsf{T}}$ if $X \in \mathrm{GL}_n(\mathbb{R})$.

For $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, we write $\|X\| := \sigma_1(X)$ for the spectral norm of $X \in \mathbb{F}^{m\times n}$ and $\kappa(X) := \sigma_1(X)/\sigma_n(X)$ for the spectral condition number of $X \in \mathrm{GL}_n(\mathbb{F})$. When we speak of norm or condition number in this article, it will always be the spectral norm or spectral condition number, the only exception is the max norm defined and used in Sect. 5.

2 Frobenius inversion in exact arithmetic

We will first show that Frobenius inversion works over any quadratic field extension, with $\mathbb{C}$ over $\mathbb{R}$ a special case. More importantly, we will show that Frobenius inversion is optimal over any quadratic field extension in that it requires a minimal number of matrix multiplications, inversions, and additions (Theorem 2.5).

The reason for the generality in this section is to show that Frobenius inversion can be useful beyond numerical analysis, applying to matrix inversions in computational number theory [12, 13], computer algebra [55, 56], cryptography [39, 69], and finite fields [53, 54] as well. This section covers the symbolic computing aspects of Frobenius inversion, i.e., in exact arithmetic. Issues related to the numerical computing aspects, i.e., in floating-point arithmetic, including conditioning, positive definiteness, etc, will be treated in Sects. 3–5.

Recall that a field $\mathbb{F}$ is said to be a *field extension* of another field $\mathbb{k}$ if $\mathbb{k} \subseteq \mathbb{F}$. In this case, $\mathbb{F}$ is automatically a $\mathbb{k}$-vector space. The dimension of $\mathbb{F}$ as a $\mathbb{k}$-vector space is called the *degree* of $\mathbb{F}$ over $\mathbb{k}$ and denoted $[\mathbb{F} : \mathbb{k}]$ [64]. A degree-two extension is also called a *quadratic extension* and they are among the most important field extensions. For example, in number theory, two of the biggest achievements in the last decade were the generalizations of Andrew Wiles' celebrated work to real quadratic fields [26] and imaginary quadratic fields [10]. Let $\mathbb{F}$ be a quadratic extension of $\mathbb{k}$. Then it follows from standard field theory [64] that there exists some monic irreducible quadratic polynomial $f \in \mathbb{k}[x]$ such that

$$\mathbb{F} \simeq \mathbb{k}[x] \big/ \langle f \rangle,$$

where $\langle f \rangle$ denotes the principal ideal generated by f and $\mathbb{k}[x]/\langle f \rangle$ the quotient ring. Let $f(x) = x^2 + \beta x + \tau$ for some $\beta, \tau \in \mathbb{k}$. Then, up to an isomorphism, f may be written in a normal form:

- $\operatorname{char}(\mathbb{k}) \neq 2$: $\beta = 0$ and $-\tau$ is not a complete square in $\mathbb{k}$;
- $\operatorname{char}(\mathbb{k}) = 2$: either $\beta = 0$ and $-\tau$ is not a complete square in $\mathbb{k}$, or $\beta = 1$ and $x^2 + x + \tau$ has no solution in $\mathbb{k}$.

2.1 Gauss multiplication over quadratic field extensions

Let ξ be a root of $f(x)$ in an algebraic closure $\overline{\mathbb{k}}$. Then $\mathbb{F} \simeq \mathbb{k}[\xi]$, i.e., any element in $\mathbb{F}$ can be written uniquely as $a_1 + a_2\xi$ with $a_1, a_2 \in \mathbb{k}$. Henceforth we will assume that $\mathbb{F} = \mathbb{k}[\xi]$. The product of two elements $a_1 + a_2\xi$, $b_1 + b_2\xi \in \mathbb{k}[\xi]$ is given by

$$(a_1+a_2\xi)(b_1+b_2\xi) = \begin{cases} (a_1b_1 - \tau a_2b_2) + (a_1b_2 + a_2b_1)\xi & \text{if } f(x) = x^2 + \tau, \\ (a_1b_1 - \tau a_2b_2) + (a_1b_2 + a_2b_1 - a_2b_2)\xi & \text{if } f(x) = x^2 + x + \tau. \end{cases} \tag{4}$$

The following result is well-known for $\mathbb{C} = \mathbb{R}[i]$ but we are unable to find a reference for an arbitrary quadratic extension $\mathbb{k}[\xi]$.

Proposition 2.1 *(Complexity of multiplication in quadratic extensions). Let $\mathbb{k}$, f, τ, ξ be as above. Then there exists an algorithm for multiplication in $\mathbb{F} = \mathbb{k}[\xi]$ that costs three multiplications in $\mathbb{k}$. Moreover, such an algorithm is optimal in the sense of bilinear complexity, i.e., it requires a minimal number of multiplications in $\mathbb{k}$.*

Proof *Case I* $f(x) = x^2 + \tau$. The product in (4) can be computed with three $\mathbb{k}$-multiplications $m_1 = (a_1 - a_2)(b_1 + \tau b_2)$, $m_2 = a_1b_2$, $m_3 = a_2b_1$, since

$$a_1b_1 - \tau a_2b_2 = m_1 - \tau m_2 + m_3, \quad a_1b_2 + a_2b_1 = m_2 + m_3. \tag{5}$$

Case II $f(x) = x^2 + x + \tau$. The product in (4) can be computed with three $\mathbb{k}$-multiplications $m_1 = a_1b_1$, $m_2 = a_2b_2$, $m_3 = (a_1 - a_2)(b_1 - b_2)$, since

$$a_1b_1 - \tau a_2b_2 = m_1 - \tau m_2, \quad a_1b_2 + a_2b_1 - a_2b_2 = m_1 - m_3. \tag{6}$$

To show optimality in both cases suppose there is an algorithm for computing (4) with two $\Bbbk$-multiplications m_1' and m_2'. Then

$$a_1b_1 - \tau a_2b_2,\ a_1b_2 + a_2b_1 - \delta a_2b_2 \in \operatorname{span}\{m_1', m_2'\},$$

where $\delta = 0$ in Case I and $\delta = 1$ in Case II. Clearly $a_1b_1 - \tau a_2b_2$ and $a_1b_2 + a_2b_1 - \delta a_2b_2$ are not collinear; thus

$$m_1', m_2' \in \operatorname{span}\{a_1b_1 - \tau a_2b_2, a_1b_2 + a_2b_1 - \delta a_2b_2\}$$

and so there exist $p, q, r, s \in \Bbbk$, $ps - qr \neq 0$, such that

$$\begin{aligned} m_1' &= p(a_1b_1 - \tau a_2b_2) + q(a_1b_2 + a_2b_1 - \delta a_2b_2) = pa_1b_1 + qa_1b_2 + qa_2b_1 \\ &\quad + (-\tau p - \delta q)a_2b_2, \\ m_2' &= r(a_1b_1 - \tau a_2b_2) + s(a_1b_2 + a_2b_1 - \delta a_2b_2) = ra_1b_1 + sa_1b_2 + sa_2b_1 \\ &\quad + (-\tau r - \delta s)a_2b_2. \end{aligned}$$

As $ps - qr \neq 0$, at least one of p, q, r, s is nonzero. Since m_1' is a $\Bbbk$-multiplication, we must have $m_1' = (\lambda_1a_1 + \lambda_2a_2)(\mu_1b_1 + \mu_2b_2)$ for some $\lambda_1a_1 + \lambda_2a_2, \mu_1b_1 + \mu_2b_2 \in \Bbbk$. Therefore

$$p(-\tau p - \delta q) = q^2, \qquad r(-\tau r - \delta s) = s^2.$$

For Case I, the left equation reduces to $\tau p^2 + q^2 = 0$ and thus $p = q = 0$ as $-\tau$ is not a complete square in $\Bbbk$; likewise, the right equation gives $r = s = 0$, a contradiction as p, q, r, s cannot be all zero. For Case II, the left equation reduces to $\tau p^2 + pq + q^2 = 0$. We must have $p \neq 0$ or else $q = 0$ will contradict $ps - qr \neq 0$; but if so, substituting $q' = q/p$ gives $q'^2 + q' + \tau = 0$, contradicting the assumption that $x^2 + x + \tau = 0$ has no solution in $\Bbbk$. □

For the special case when $\Bbbk = \mathbb{R}$ and $f(x) = x^2 + 1$, we have $\xi = i$ and $\mathbb{F} = \Bbbk[\xi] = \mathbb{C}$ and the algorithm in (5) is the celebrated Gauss multiplication of complex numbers, $(a_1 + ia_2)(b_1 + ib_2) = (a_1b_1 - a_2b_2) + i[(a_1 + a_2)(b_1 + b_2) - a_1b_1 - a_2b_2]$, whose optimality is proved in [58, 75]. Proposition 2.1 may be viewed as a generalization of Gauss multiplication to arbitrary quadratic extensions.

In the language of tensors [48, Example 3.8], multiplication in $\Bbbk[\xi]$ is a bilinear map over $\Bbbk$,

$$m : \Bbbk[\xi] \times \Bbbk[\xi] \to \Bbbk[\xi], \quad (a_1 + a_2\xi, b_1 + b_2\xi) \mapsto (a_1 + a_2\xi)(b_1 + b_2\xi),$$

and therefore corresponds to a tensor in $\mu \in \Bbbk[\xi] \otimes \Bbbk[\xi] \otimes \Bbbk[\xi]$. An equivalent way to state Proposition 2.1 is that the tensor rank of μ is exactly three.

As is the standard assumption in algebraic computational complexity theory [9], our complexity results in the whole of Sect. 2 count only the number of multiplications of variables $(x, y) \mapsto xy$ and not multiplications by constant scalars $x \mapsto ax$ for any fixed $a \in \mathbb{k}$. We have explained this in our earlier works [16, 48, 77] for a numerical linear algebra readership but given its importance we will briefly review the reasons here.

A simple reason is that scalar multiplication by a fixed constant can be hard-coded into software or hard-wired into hardware but not so for multiplication of variables, which depends on the inputs. A more sophisticated reason is that any part of the algorithm involving variable multiplications can often be recursively applied (see Sect. 2.4). Whatever the case, the two kinds of multiplications need to be distinguished, and is consistent with standard practice in algebraic computational complexity. For example, the celebrated $O(n^{\log_2 7})$-complexity of matrix-matrix product discovered by Strassen [70], gradually improved to the current $O(n^{2.371552})$ [73], assumes that only variable multiplications are counted.

Yet a third reason is that the number of scalar multiplications may be increased or decreased with a linear change-of-coordinates but the number of variable multiplications is invariant under any linear change-of-coordinates. For example, there is no way one may obtain Gauss multiplication from standard matrix multiplication or Frobenius inversion from standard matrix inversion by taking linear combinations of the coordinates of the input matrices. Counting variable multiplications shows that they are truly different algorithms. This invariance is why a complexity measure based on the the number of variable multiplications is favored in algebraic computational complexity theory.

2.2 Gauss matrix multiplication over quadratic field extensions

We extend the multiplication algorithm in the previous section to matrices. Notations will be as in the last section. Let $\mathbb{F}^{n\times n}$ be the $\mathbb{F}$-algebra of $n \times n$ matrices over $\mathbb{F}$. Since $\mathbb{F} = \mathbb{k}[\xi]$, we have $\mathbb{F}^{n\times n} = \mathbb{k}^{n\times n} \otimes_{\mathbb{k}} \mathbb{F}$ [48, p. 627]. Thus an element in $X \in \mathbb{F}^{n\times n}$ can be written as $X = A + \xi B$ where $A, B \in \mathbb{k}^{n\times n}$.

By following the argument in the proof of Proposition 2.1, we obtain its analogue for matrix multiplication in $\mathbb{F}^{n\times n}$ via matrix multiplications in $\mathbb{k}^{n\times n}$.

Proposition 2.2 *(Gauss matrix multiplication). Let $\mathbb{k}, \mathbb{F}, n, f, \tau, \xi$ be as before. Let $X = A + \xi B$, $Y = C + \xi D \in \mathbb{F}^{n\times n}$ with $A, B, C, D \in \mathbb{k}^{n\times n}$. If $f(x) = x^2 + \tau$, then XY can be computed via*

$$\begin{aligned} &M_1 = (A - B)(C + \tau D), && M_2 = AD, && M_3 = BC; \\ &N_1 = M_1 - \tau M_2 + M_3, && N_2 = M_2 + M_3; && XY = N_1 + \xi N_2. \end{aligned} \tag{7}$$

If $f(x) = x^2 + x + \tau$, then XY can be computed via

$$\begin{aligned} &M_1 = AC, && M_2 = BD, && M_3 = (A - B)(C - D); \\ &N_1 = M_1 - \tau M_2, && N_2 = M_1 - M_3; && XY = N_1 + \xi N_2. \end{aligned} \tag{8}$$

The algorithms for forming XY in (7) and (8) use a minimal number of matrix multiplications in $\mathbb{k}^{n\times n}$.

Proof It is straightforward to check that (7) and (8) give XY. To see minimality, we repeat the proof of Proposition 2.1 noting that the argument depends only on $\mathbb{F}$ as a two-dimensional free $\mathbb{k}$-module, and that $\mathbb{F}^{n\times n}$ is also a two-dimensional free $\mathbb{k}^{n\times n}$-module. □

2.3 Frobenius matrix inversion over quadratic field extensions

Let $A+\xi B \in \mathrm{GL}_n(\mathbb{F})$ with $A, B \in \mathbb{k}^{n\times n}$. Then $(A+\xi B)^{-1} = C+\xi D$ if and only if

$$(A+\xi B)(C+\xi D) = I, \tag{9}$$

from which we may solve for $C, D \in \mathbb{k}^{n\times n}$. As we saw in (4), multiplication in $\mathbb{F}$ and thus that in $\mathbb{F}^{n\times n}$ depends on the form of f. So we have to consider two cases corresponding to the two normal forms of f.

Lemma 2.3 *Let $\mathbb{k}, \mathbb{F}, n, f, \tau, \xi$ be as before. Let $A+\xi B \in \mathrm{GL}_n(\mathbb{F})$ with $A, B \in \mathbb{k}^{n\times n}$.*

(i) If $f(x) = x^2+\tau$, then $A+\tau BA^{-1}B \in \mathrm{GL}_n(\mathbb{k})$ whenever $A \in \mathrm{GL}_n(\mathbb{k})$.
(ii) If $f(x) = x^2+x+\tau$, then $\tau B + AB^{-1}A - A \in \mathrm{GL}_n(\mathbb{k})$ whenever $B \in \mathrm{GL}_n(\mathbb{k})$.

Proof Consider the case $f(x) = x^2+\tau$. By (9), $AC-\tau BD = I$ and $AD+BC = 0$. So $(A+\tau BA^{-1}B)C = I$. Hence $A+\tau BA^{-1}B$ is invertible. A similar argument applies to the case $f(x) = x^2+x+\tau$ to yield (ii). □

Let the matrix addition, multiplication, and inversion maps over any field $\mathbb{F}$ be denoted respectively by

$$\begin{aligned}
\mathsf{add}_{n,\mathbb{F}} &: \mathbb{F}^{n\times n}\times\mathbb{F}^{n\times n} \to \mathbb{F}^{n\times n}, & \mathsf{add}_{n,\mathbb{F}}(X,Y) &= X+Y;\\
\mathsf{mul}_{n,\mathbb{F}} &: \mathbb{F}^{n\times n}\times\mathbb{F}^{n\times n} \to \mathbb{F}^{n\times n}, & \mathsf{mul}_{n,\mathbb{F}}(X,Y) &= XY;\\
\mathsf{inv}_{n,\mathbb{F}} &: \mathrm{GL}_n(\mathbb{F}) \to \mathrm{GL}_n(\mathbb{F}), & \mathsf{inv}_{n,\mathbb{F}}(X) &= X^{-1}.
\end{aligned}$$

We will now express $\mathsf{inv}_{n,\mathbb{F}}$ in terms of $\mathsf{inv}_{n,\mathbb{k}}$, $\mathsf{mul}_{n,\mathbb{k}}$, and $\mathsf{add}_{n,\mathbb{k}}$.

Lemma 2.4 *(Frobenius inversion over quadratic fields) Let $\mathbb{k}, \mathbb{F}, n, f, \tau, \xi$ be as before. Let $X = A+\xi B \in \mathrm{GL}_n(\mathbb{F})$ with $A, B \in \mathbb{k}^{n\times n}$. If $f(x) = x^2+\tau$ and $A \in \mathrm{GL}_n(\mathbb{k})$, then*

$$X^{-1} = (A+\tau BA^{-1}B)^{-1} - \xi A^{-1}B(A+\tau BA^{-1}B)^{-1}. \tag{10}$$

If $f(x) = x^2+x+\tau$ and $B \in \mathrm{GL}_n(\mathbb{k})$, then

$$X^{-1} = (B^{-1}A-I)(AB^{-1}A-A+\tau B)^{-1} - \xi(AB^{-1}A-A+\tau B)^{-1} \tag{11}$$

Proof *Case I* $f(x) = x^2+\tau$. From (9), we get

$$AC-\tau BD = I, \qquad AD+BC = 0.$$

Case II $f(x) = x^2 + x + \tau$. From (9), we get

$$AC - \tau BD = I, \qquad AD + BC - BD = 0.$$

In each case, solving the equations for C and D gives us the required expressions (10) and (11). □

We could derive alternative inversion formulas with other conditions on A and B. For example, in the case $f(x) = x^2 + \tau$, instead of (10), we could have

$$X^{-1} = B^{-1}A(AB^{-1}A + \tau B)^{-1} - \xi(AB^{-1}A + \tau B)^{-1},$$

conditional on $B \in \mathrm{GL}_n(\Bbbk)$; in the case $f(x) = x^2 + x + \tau$, instead of (11), we could have

$$X^{-1} = (A + \tau B(A - B)^{-1}B)^{-1} - \xi(A - B)^{-1}B(A + \tau B(A - B)^{-1}B)^{-1},$$

conditional on $A - B \in \mathrm{GL}_n(\Bbbk)$. There is no single inversion formula that will work universally for all $A + \xi B \in \mathrm{GL}_n(\mathbb{F})$. Nevertheless, in each case, the inversion formula (10) or (11) works almost everywhere except for matrices $A + \xi B$ with $\det(A) = 0$ or $\det(B) = 0$ respectively. In Sect. 4.2, we will see how to alleviate this minor restriction algorithmically for complex matrices.

We claim that (10) and (11) allow $\mathsf{inv}_{n,\mathbb{F}}$ to be evaluated by invoking $\mathsf{inv}_{n,\Bbbk}$ twice, $\mathsf{mul}_{n,\Bbbk}$ thrice, and $\mathsf{add}_{n,\Bbbk}$ once. To see this more clearly, we express them in pseudocode as Algorithms 1 and 2 respectively.

Algorithm 1 Frobenius Inversion with ξ a root of $x^2 + \tau$

```
Input: $X = A + \xi B$ with $A \in \mathrm{GL}_n(\Bbbk)$
1: matrix invert $X_1 = A^{-1}$;
2: matrix multiply $X_2 = X_1 B$;
3: matrix multiply $X_3 = B X_2$;
4: matrix add $X_4 = A + \tau X_3$;
5: matrix invert $X_5 = X_4^{-1}$;
6: matrix multiply $X_6 = X_2 X_5$;
Output: inverse $X^{-1} = X_5 - \xi X_6$
```

A few words are in order here. A numerical linear algebraist may balk at inverting A and then multiplying it to B to form $A^{-1}B$ instead of solving a linear system with multiple right-hand sides. However, Algorithms 1 and 2 should be viewed in the context of *symbolic computing* over arbitrary fields. To establish complexity results like Theorem 2.5 and Theorem 2.10, we would have to state the algorithms purely in terms of algebraic operations in $\Bbbk^{n\times n}$, i.e., $\mathsf{inv}_{n,\Bbbk}$, $\mathsf{mul}_{n,\Bbbk}$, and $\mathsf{add}_{n,\Bbbk}$. The *numerical computing* aspects specific to $\Bbbk = \mathbb{R}$ and $\mathbb{F} = \mathbb{C}$ will be deferred to Sects. 4–5, where, among other things, we would present several numerical computing variants of Algorithm 1 (see Algorithms 3, 5, 6, 7). We also remind the reader that a term like $X_5 - \xi X_6$ in the output of these algorithms does not entail matrix addition; here ξ

plays a purely symbolic role like the imaginary unit i, and X_5 and $-X_6$ are akin to the 'real part' and 'imaginary part.'

Algorithm 2 Frobenius Inversion with ξ a root of $x^2 + x + \tau$

```
Input: $X = A + \xi B$ with $B \in \mathrm{GL}_n(\Bbbk)$
1: matrix invert $X_1 = B^{-1}$;
2: matrix multiply $X_2 = X_1 A - I$;
3: matrix multiply $X_3 = A X_2$;
4: matrix add $X_4 = X_3 + \tau B$;
5: matrix invert $X_5 = X_4^{-1}$;
6: matrix multiply $X_6 = X_3 X_5$;
Output: inverse $X^{-1} = X_6 - \xi X_5$
```

The addition of a fixed constant (i.e., independent of inputs A and B) matrix $-I$ in Step 2 of Algorithm 2 has no effect on its computational complexity, as explained at the end of Sect. 2.1.

As we mentioned earlier, $\mathbb{F}^{n\times n}$ is a $\Bbbk^{n\times n}$-bimodule. We prove next that Algorithms 1 and 2 have optimal computational complexity in terms of matrix operations in $\Bbbk^{n\times n}$.

Theorem 2.5 *(Optimality of Frobenius Inversion). Algorithm 1 and 2 for* $\mathsf{inv}_{n,\mathbb{F}}$ *require the fewest number of matrix operations in* $\Bbbk^{n\times n}$*: two* $\mathsf{inv}_{n,\Bbbk}$*, three* $\mathsf{mul}_{n,\Bbbk}$*, and one* $\mathsf{add}_{n,\Bbbk}$*, i.e., there is no algorithm for matrix inversion in* $\mathbb{F}^{n\times n}$ *that takes four or fewer matrix operations in* $\Bbbk^{n\times n}$*.*

Proof If $n = 1$, then this reduces to Proposition 2.1. So we will assume that $n \geq 2$. We will restrict ourselves to Algorithm 1 as the argument for Algorithm 2 is nearly identical.

Clearly, we need at least one $\mathsf{add}_{n,\Bbbk}$ to compute $\mathsf{inv}_{n,\mathbb{F}}$ so Algorithm 1 is already optimal in this regard. We just need to restrict ourselves to the numbers of $\mathsf{inv}_{n,\Bbbk}$ and $\mathsf{mul}_{n,\Bbbk}$, which are invoked twice and thrice respectively in Algorithm 1. We will show that these numbers are minimal. In the following, we pick any $A, B \in \mathrm{GL}_n(\Bbbk)$ that do not commute.

First we claim that it is impossible to compute $(A+\xi B)^{-1}$ with fewer than two $\mathsf{inv}_{n,\Bbbk}$ even with no limit on the number of $\mathsf{add}_{n,\Bbbk}$ and $\mathsf{mul}_{n,\Bbbk}$. By (10), $(A+\xi B)^{-1}$ comprises two $\Bbbk^{n\times n}$ matrices $(A + \tau BA^{-1}B)^{-1}$ and $A^{-1}B(A + \tau BA^{-1}B)^{-1}$, which we will call its 'real part' and 'imaginary part' respectively, slightly abusing terminologies. We claim that computing the 'real part' $(A + \tau BA^{-1}B)^{-1}$ alone already takes at least two $\mathsf{inv}_{n,\Bbbk}$. If $(A + \tau BA^{-1}B)^{-1}$ can be computed with just one $\mathsf{inv}_{n,\Bbbk}$, then $A(A + \tau BA^{-1}B)^{-1}$ can also be computed with just one $\mathsf{inv}_{n,\Bbbk}$ as the extra factor A involves no inversion. However, if it takes only one $\mathsf{inv}_{n,\Bbbk}$, then we must have an expression

$$A(A + \tau BA^{-1}B)^{-1} = f(A, B, g(A, B)^{-1})$$

for some noncommutative polynomials $f \in \Bbbk\langle x, y, z\rangle$ and $g \in \Bbbk\langle x, y\rangle$. Now observe that

$$A(A + \tau BA^{-1}B)^{-1} = (I + \tau(BA^{-1})^2)^{-1}.$$

To see that the last two expressions are contradictory, we write $X{:=}BA^{-1}$ and expand them in formal power series, thereby removing negative powers for an easier comparison:

$$\begin{aligned}\sum_{k=0}^{\infty}(-\tau)^k X^{2k} &= (I + \tau X^2)^{-1} = f(A, XA, g(A, XA)^{-1})\\ &= f\Bigl(A, XA, \sum_{k=0}^{\infty}\bigl(I - g(A, XA)\bigr)^k\Bigr).\end{aligned}$$

Note that the leftmost expression is purely in powers of X, but the rightmost expression must necessarily involve A—indeed any term involving a power of X must involve A to the same or higher power. The remaining possibility that X is a power of A is excluded since A and B do not commute. So we arrive at a contradiction. Hence $(A + \tau BA^{-1}B)^{-1}$ and therefore $(A + \xi B)^{-1}$ requires at least two $\mathsf{inv}_{n,\Bbbk}$ to compute.

Next we claim that it is impossible to compute $(A + \xi B)^{-1}$ with fewer than three $\mathsf{mul}_{n,\Bbbk}$ even with no limit on the number of $\mathsf{add}_{n,\Bbbk}$ and $\mathsf{inv}_{n,\Bbbk}$. Let the 'real part' and 'imaginary part' be denoted

$$Y{:=}\,(A + \tau BA^{-1}B)^{-1}, \qquad Z{:=}\,A^{-1}B(A + \tau BA^{-1}B)^{-1} = (B + \tau AB^{-1}A)^{-1}.$$

Observe that we may express $BA^{-1}B$ in terms of Y and $AB^{-1}A$ in terms of Z using only $\mathsf{add}_{n,\Bbbk}$ and $\mathsf{inv}_{n,\Bbbk}$:

$$BA^{-1}B = \tau^{-1}(Y^{-1} - A), \qquad AB^{-1}A = \tau^{-1}(Z^{-1} - B).$$

So computing both $BA^{-1}B$ and $AB^{-1}A$ take the same number of $\mathsf{mul}_{n,\Bbbk}$ as computing both Y and Z. However, as A and B do not commute, it is impossible to compute both $BA^{-1}B$ and $AB^{-1}A$ with just two $\mathsf{mul}_{n,\Bbbk}$. Consequently $(A + \xi B)^{-1} = Y + \xi Z$ requires at least three $\mathsf{mul}_{n,\Bbbk}$ to compute. □

A more formal way to cast our proof above would involve the notion of a *straight-line program* [9, Definition 4.2], but we prefer to avoid pedantry given that the ideas involved are the same.

2.4 Frobenius inversion over iterated quadratic extensions

Repeated applications of Algorithms 1 and 2 allow us to extend Frobenius inversion to an *iterated quadratic extension*:

$$\Bbbk =: \mathbb{F}_0 \subsetneq \mathbb{F}_1 \subsetneq \cdots \subsetneq \mathbb{F}_m := \mathbb{F}, \tag{12}$$

where $[\mathbb{F}_k : \mathbb{F}_{k-1}] = 2$, $k = 1, \dots, m$. By our discussion at the beginning of Sect. 2, $\mathbb{F}_k = \mathbb{F}_{k-1}[\xi_k]$ for some $\xi_k \in \mathbb{F}_k$. Let $f_k \in \Bbbk[x]$ be the minimal polynomial [64] of ξ_k. Then f_k is a monic irreducible quadratic polynomial that we may assume is in normal form, i.e.,

$$f_k(x) = x^2 + \tau_k \quad \text{or} \quad f_k(x) = x^2 + x + \tau_k, \quad k = 1, \dots, m.$$

Since $[\mathbb{F} : \Bbbk] = \prod_{k=1}^{m} = [\mathbb{F}_k : \mathbb{F}_{k-1}] = 2^m$, any element in $\mathbb{F}$ may be written as

$$\sum_{\alpha \in \{0,1\}^m} c_\alpha \xi^\alpha \tag{13}$$

in *multi-index* notation with $\alpha = (\alpha_1, \dots, \alpha_m) \in \{0, 1\}^m$, $\xi^\alpha := \xi_1^{\alpha_1} \cdots \xi_m^{\alpha_m}$, and $c_\alpha \in \Bbbk$. Moreover, we may regard $\mathbb{F}$ as a quotient ring of a multivariate polynomial ring or as a tensor product of m quotient rings of univariate polynomial ring:

$$\mathbb{F} \simeq \Bbbk[x_1, \dots, x_m] \big/ \langle f_1, \dots, f_m \rangle = \bigotimes_{k=1}^{m} \big(\Bbbk[x] \big/ \langle f_k \rangle \big). \tag{14}$$

There are many important fields that are special cases of iterated quadratic extensions

Example 2.6 (Constructible numbers) One of the most famous instance is the special case $\Bbbk = \mathbb{Q}$ with the iterated quadratic extension $\mathbb{F} \subseteq \mathbb{R}$. In which case the positive numbers in $\mathbb{F}$ are called *constructible numbers* and they are precisely the lengths that can be constructed with a compass and a straightedge in a finite number of steps. The impossibillity of trisecting an angle, doubling a cube, squaring a circle, constructing n-sided regular polygons for $n = 7, 9, 11, 13, 14, 18, \dots$, etc, were all established using the notion of constructible numbers.

Example 2.7 (Multiquadratic fields). Another interesting example is $\mathbb{F} = \mathbb{Q}[\sqrt{q_1}, \dots, \sqrt{q_m}]$. It is shown in [6] that

$$\mathbb{Q} \subsetneq \mathbb{Q}[\sqrt{q_1}] \subsetneq \mathbb{Q}[\sqrt{q_1}, \sqrt{q_2}] \subsetneq \cdots \subsetneq \mathbb{Q}[\sqrt{q_1}, \dots, \sqrt{q_m}]$$

is an iterated quadratic extension if the product of any nonempty subset of $\{\sqrt{q_1}, \dots, \sqrt{q_m}\}$ is not in $\mathbb{Q}$. In this case, we have $\mathbb{F}_k = \mathbb{Q}[\sqrt{q}_1, \dots, \sqrt{q}_k]$ and $f_k(x) = x^2 - q_k$, $k = 1, \dots, m$.

Example 2.8 (Tower of root extensions of non-square). Yet another commonly occurring example [20, Section 14.7] of iterated quadratic extension is a 'tower' of root extensions:

$$\mathbb{Q} \subsetneq \mathbb{Q}[q^{1/2}] \subsetneq \mathbb{Q}[q^{1/4}] \subsetneq \cdots \subsetneq \mathbb{Q}[q^{1/2^m}]$$

where $q \in \mathbb{Q}$ is not a complete square.

Since $\mathbb{k}^{n\times n}$ and $\mathbb{F}$ are both free $\mathbb{k}$-modules, we have $\mathbb{F}^{n\times n} = \mathbb{k}^{n\times n} \otimes_{\mathbb{k}} \mathbb{F}$ as tensor product of $\mathbb{k}$-modules. Hence the expression (13) may be extended to matrices, i.e., any $X \in \mathbb{F}^{n\times n}$ may be written as

$$X = \sum_{\alpha\in\{0,1\}^m} C_\alpha \xi^\alpha \tag{15}$$

with $C_\alpha \in \mathbb{k}^{n\times n}$, $\alpha \in \{0, 1\}^m$. Note that the c_α in (13) are scalars and the C_α in (15) are matrices. On the other hand, in an iterated quadratic extension (12), each $\mathbb{F}_k$ is an $\mathbb{F}_{k-1}$-module, $k = 1, \dots, m$, and thus we also have the tensor product relation

$$\mathbb{k}^{n\times n} \otimes_{\mathbb{k}} \mathbb{F} = \mathbb{k}^{n\times n} \otimes_{\mathbb{k}} \mathbb{F}_1 \otimes_{\mathbb{F}_1} \mathbb{F}_2 \otimes_{\mathbb{F}_2} \cdots \otimes_{\mathbb{F}_{m-1}} \mathbb{F},$$

recalling that $\mathbb{F}_0 := \mathbb{k}$ and $\mathbb{F}_m := \mathbb{F}$. Hence any $X \in \mathbb{F}^{n\times n}$ may also be expressed recursively as

$$\begin{aligned} X &= A_0 + \xi_m A_1, \\ A_\beta &= A_{0,\beta} + \xi_{m-k} A_{1,\beta}, \quad \beta \in \{0, 1\}^k, \quad k = 1, \dots, m-1, \end{aligned} \tag{16}$$

with $A_\beta \in \mathbb{F}^{n\times n}_{m-|\beta|}$. The relation between the two expressions (15) and (16) is given as follows.

Lemma 2.9 *Let $X \in \mathbb{F}^{n\times n}$ be expressed as in (15) with $C_\alpha \in \mathbb{k}^{n\times n}$, $\alpha \in \{0, 1\}^m$, and as in (16) with $A_\beta \in \mathbb{F}^{n\times n}_{m-|\beta|}$, $\beta \in \{0, 1\}^k$. Then for any $k \in \{1, \dots, m\}$,*

$$X = \sum_{\beta\in\{0,1\}^k} A_\beta \xi_{m-k+1}^{\beta_1} \cdots \xi_m^{\beta_k},$$

and for any $\beta \in \{0, 1\}^k$,

$$A_\beta = \sum_{\gamma\in\{0,1\}^{m-k}} C_{\gamma,\beta} \xi_1^{\gamma_1} \cdots \xi_{m-k}^{\gamma_{m-k}}.$$

In particular, $C_\alpha = A_\alpha$.

Proof We proceed by induction on k. Clearly the formula holds for $k = 1$ by (16). Assume that the first expression holds for $k = s$, i.e.,

$$X = \sum_{\beta\in\{0,1\}^s} A_\beta \xi_{m-s+1}^{\beta_1} \cdots \xi_m^{\beta_s}.$$

To show that it also holds for $k = s + 1$, note that $A_\beta = A_{0,\beta} + \xi_{m-s} A_{1,\beta}$, so

$$X = \sum_{\beta\in\{0,1\}^s} (A_{0,\beta} + \xi_{m-s} A_{1,\beta}) \xi_{m-s+1}^{\beta_1} \cdots \xi_m^{\beta_s} = \sum_{\gamma\in\{0,1\}^{s+1}} A_\gamma \xi_{m-s}^{\gamma_1} \cdots \xi_m^{\gamma_{s+1}}$$

completing the induction. Comparing coefficients in (15) and (16) yields the second expression. □

The representation in Lemma 2.9, when combined with Gauss multiplication, gives us a method for fast matrix multiplication in $\mathbb{F}^{n\times n}$, and, when combined with Frobenius inversion, gives us a method for fast matrix inversion in $\mathbb{F}^{n\times n}$.

Theorem 2.10 *(Gauss multiplication and Frobenius inversion over iterated quadratic extension) Let $\mathbb{F}$ be an iterated quadratic extension of $\Bbbk$ of degree $[\mathbb{F} : \Bbbk] = 2^m$. Then*

(i) one may multiply two matrices in $\mathbb{F}^{n\times n}$ with 3^m multiplications in $\Bbbk^{n\times n}$;
(ii) one may invert a generic matrix in $\mathbb{F}^{n\times n}$ with $3(3^m - 2^m)$ multiplications and 2^m inversions in $\Bbbk^{n\times n}$.

If we write $N = 2^m$, this multiplication algorithm reduces the complexity of evaluating $\mathsf{mul}_{n,\mathbb{F}}$ from $O(N^2)$ to $O(N^{\log_2 3})$ $\mathsf{mul}_{n,\Bbbk}$.

Proof Let $X, Y \in \mathbb{F}^{n\times n}$. By (16), we may write

$$X = A_0 + \xi_m A_1, \quad Y = B_0 + \xi_m B_1,$$

and thus compute XY in terms of $A_0, A_1, B_0, B_1 \in \mathbb{F}_{m-1}^{n\times n}$ using three $\mathsf{mul}_{n,\mathbb{F}_{m-1}}$ by Proposition 2.2. Each $\mathsf{mul}_{n,\mathbb{F}_{m-1}}$ in turn costs three $\mathsf{mul}_{n,\mathbb{F}_{m-2}}$ by Proposition 2.2. Repeating the argument until we arrive at $\mathsf{mul}_{n,\mathbb{F}_0} = \mathsf{mul}_{n,\Bbbk}$, we see that the total number of $\mathsf{mul}_{n,\Bbbk}$ is 3^m.

Now if X above is generic, depending on whether $f_m(x) = x^2 + \tau_m$ or $x^2 + x + \tau_m$, Algorithm 1 or Algorithm 2 takes three $\mathsf{mul}_{n,\mathbb{F}_{m-1}}$ and two $\mathsf{inv}_{n,\mathbb{F}_{m-1}}$. As in the multiplication case, the argument applies recursively to $m, m-1, \dots, 2, 1$. Writing $\#(\mathsf{op})$ for the number of operation op, we have

$$\#(\mathsf{inv}_{n,\mathbb{F}_k}) = 3\,\#(\mathsf{mul}_{n,\mathbb{F}_{k-1}}) + 2\,\#(\mathsf{inv}_{n,\mathbb{F}_{k-1}}), \qquad k = 1, \dots, m.$$

By Proposition 2.2, we have $\#(\mathsf{mul}_{n,\mathbb{F}_k}) = 3\,\#(\mathsf{mul}_{n,\mathbb{F}_{k-1}})$. Hence we obtain

$$\#(\mathsf{inv}_{n,\mathbb{F}}) = 3(3^m - 2^m)\,\#(\mathsf{mul}_{n,\Bbbk}) + 2^m\,\#(\mathsf{inv}_{n,\Bbbk})$$

as required. □

Slightly abusing terminologies, we will call the multiplication and inversion algorithms in the proof of Theorem 2.10 Gauss multiplication and Frobenius inversion for iterated quadratic extension respectively. Both rely on the general technique of *divide-and-conquer*. Moreover, Gauss multiplication for iterated quadratic extension is in spirit the same as the Karatsuba algorithm [42] for fast integer multiplication and multidimensional fast Fourier transform [67]. Indeed, all three algorithms may be viewed as fast algorithms for modular polynomial multiplication in a ring

$$\Bbbk[x_1, \dots, x_m]/\langle x_1^d + \tau_1, \dots, x_m^d + \tau_m\rangle \simeq \Bbbk[x_1]/\langle x_1^d + \tau_1\rangle \otimes \cdots \otimes \Bbbk[x_m]/\langle x_m^d + \tau_m\rangle$$

for different choices of $m, d, \tau_1, \dots, \tau_m$. To be more specific, we have

(a) Karatsuba algorithm: $m = 1$, $d = 0$, $\tau_1 = -1$.
(b) Multidimensional fast Fourier transform: $m \in \mathbb{N}$, $d \in \mathbb{N}$, $\tau_1 = \cdots = \tau_m = -1$.
(c) Gauss multiplication for iterated quadratic: $m \in \mathbb{N}$, $d = 2$, $\tau_1, \dots, \tau_m$ as in (12)–(14).

2.5 Moore–Penrose and Sherman–Morrison

One might ask if Frobenius inversion extends to pseudoinverse. In particular, does (10) hold if matrix inverse is replaced by Moore–Penrose inverse? The answer is no, which can be seen by taking $\mathbb{k} = \mathbb{R}$, $\mathbb{F} = \mathbb{C}$, in which case (10) is just (1). Let

$$X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & i \end{bmatrix}, \qquad X^{\dagger} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -i \end{bmatrix}, \qquad Y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where Y denotes the right-hand side of (1) with Moore–Penrose inverse in place of matrix inverse. Clearly $X^{\dagger} \neq Y$.

One may be led to think that Frobenius inversion (10) is a consequence of Sherman–Morrison–Woodbury-type identities such as

$$\begin{aligned}(A+B)^{-1} &= A^{-1} - A^{-1}(B^{-1} + A^{-1})^{-1}A^{-1} = A^{-1} - A^{-1}(AB^{-1} + I)^{-1} \\ &= A^{-1} - (A + AB^{-1}A)^{-1} = A^{-1} - A^{-1}B\,(A+B)^{-1}\end{aligned}$$

but it is not. The point to note is that such identities invariably involve at least one matrix inversion in $\mathbb{F}^{n\times n}$ whereas (10) is purely in terms of matrix inversions in $\mathbb{k}^{n\times n}$.

3 Solving linear systems with Frobenius inversion

We remind the reader that the previous section is the only one about Frobenius inversion over arbitrary fields. In this and all subsequent sections we return to the familiar setting of real and complex fields. In this section we discuss the solution of a system of complex linear equations

$$(A + iB)(x + iy) = c + id, \qquad A, B \in \mathbb{R}^{n\times n}, \quad c, d \in \mathbb{R}^n \tag{17}$$

for $x, y \in \mathbb{R}^n$ with Frobenius inversion in a way that does not require computing explicit inverse and the circumstances under which this method is superior. For the sake of discussion, we will assume throughout this section that we use LU factorization, computed using Gaussian Elimination with Partial Pivoting, as our main tool, but one may easily substitute it with any other standard matrix decomposition.

The most straightforward way to solve (17) would be directly as a complex linear system with coefficient matrix $A + iB \in \mathbb{C}^{n\times n}$. As we mentioned at the beginning of this article, the IEEE-754 floating point standard [41] does not support complex floating point arithmetic and relies on software to convert them to real floating point arithmetic [59, p. 55]. For greater control, we might instead transform (17) into a real linear system with coefficient matrix $\left[\begin{smallmatrix} A & -B \\ B & A \end{smallmatrix}\right] \in \mathbb{R}^{2n\times 2n}$. Note that the condition numbers of the coefficient matrices are identical:

$$\kappa_2(A + iB) = \kappa_2\left(\begin{bmatrix} A & -B \\ B & A \end{bmatrix}\right).$$

However, an alternative that takes advantage of Frobenius inversion (1) would be Algorithm 3. For simplicity, we assume that A is invertible below but if not, this can be easily addressed with a simple trick in Sect. 4.2. Algorithm 3 shares the same two-level structure in block LU decomposition [31, Section 3.2.11] but captures the more special block matrix $\left[\begin{smallmatrix} A & -B \\ B & A \end{smallmatrix}\right]$ has a more special block structure coming from the quadratic field extension.

Algorithm 3 Linear system with Frobenius inversion and LU factorization

```
Input: $A + iB \in \mathrm{GL}_n(\mathbb{C})$ with $A \in \mathrm{GL}_n(\mathbb{R})$, $c, d \in \mathbb{R}^n$
1: LU factorize $A = P_1^{\mathsf{T}} L_1 U_1$;
2: forward and backward substitute for $X_1$ in $L_1 U_1 X_1 = P_1 B$;
3: matrix multiply and add $X_2 = A + B X_1$;
4: LU factorize $X_2 = P_2^{\mathsf{T}} L_2 U_2$;
5: forward and backward substitute for $x_1, y_1$ in $L_2 U_2 [x_1, y_1] = P_2 [c, d]$;
6: forward and backward substitute for $x_2, y_2$ in $L_1 U_1 [x_2, y_2] = P_1 B [y_1, x_1]$;
7: vector add $x = x_1 + x_2$, $y = y_2 - y_1$;
Output: solution of $(A + iB)(x + iy) = c + id$
```

One may easily verify that Algorithm 3 gives the solution as claimed, by virtue of the expression (1) for Frobenius inversion. Observe that Algorithm 3 involves only the matrices A, B, and $A + BA^{-1}B$, unsurprising since these are the matrices that appear in (1). In the rest of this section, we will establish that there is an open subset of matrices $A + iB \in \mathbb{C}^{n\times n}$ with

$$\max\bigl(\kappa_2(A), \kappa_2(B), \kappa_2(A + BA^{-1}B)\bigr) \ll \kappa_2(A + iB). \tag{18}$$

A consequence is that ill-conditioned complex matrices with well-conditioned real and imaginary parts are common; in particular, there are uncountably many and they occur with positive probability with respect to any reasonable probability measure (e.g., Gaussian) on $\mathbb{C}^{n\times n}$. In fact we will show in Theorems 3.3 and 3.3 that $A + iB \in \mathbb{C}^{n\times n}$ or $\left[\begin{smallmatrix} A & -B \\ B & A \end{smallmatrix}\right] \in \mathbb{R}^{2n\times 2n}$ can be arbitrarily ill-conditioned when A and B are well-conditioned or even perfectly conditioned, a situation that is tailor-made for Algorithm 3.

Lemma 3.1 *Let $A, B \in \mathbb{R}^{n\times n}$ with $A = GH$ for some $G, H \in \mathrm{GL}_n(\mathbb{R})$. Let $N := H^{-1}BG^{-1}$. Then*

$$\kappa_2(G)\kappa_2(I + iN)\kappa_2(H) \geq \kappa_2(A + iB) \geq \max\left\{\frac{\|(I + iN)^{-1}\|}{\kappa_2(H)}, \frac{\|(I + iN)^{-1}\|}{\kappa_2(G)}\right\}.$$

Proof Let $X{:=}A+iB=H(I+iN)G$. Then

$$\kappa_2(X)\le\kappa_2(H)\kappa_2(I+iN)\kappa_2(G).$$

For the other inequality, since $X^{-1}=G^{-1}(I+iN)^{-1}H^{-1}$,

$$\kappa_2(X)=\|X\|\|X^{-1}\|=\|A+iB\|\|G^{-1}(I+iN)^{-1}H^{-1}\|. \tag{19}$$

As $\|Xv\|=\|Av+iBv\|\ge\|Av\|$ for all $v\in\mathbb{R}^n$, we have

$$\|X\|\ge\|A\|. \tag{20}$$

Since the spectral norm is submultiplicative,

$$\begin{aligned}\|H\|&=\|G^{-1}A\|\le\|A\|\|G^{-1}\|,\\ \|(I+iN)^{-1}\|&=\|GG^{-1}(I+iN)^{-1}H^{-1}H\|\le\|G\|\|G^{-1}(I+iN)^{-1}H^{-1}\|\|H\|,\end{aligned}$$

and we obtain

$$\|G^{-1}(I+iN)^{-1}H^{-1}\|\ge\frac{\|(I+iN)^{-1}\|}{\|G\|\|H\|}\ge\frac{\|(I+iN)^{-1}\|}{\|G\|\|G^{-1}\|\|A\|}=\frac{\|(I+iN)^{-1}\|}{\kappa_2(G)\|A\|}. \tag{21}$$

Assembling (19)–(21), we get

$$\kappa_2(X)\ge\|A\|\frac{\|(I+iN)^{-1}\|}{\kappa_2(G)\|A\|}=\frac{\|(I+iN)^{-1}\|}{\kappa_2(G)}.$$

Swapping the roles of G and H, we get $\kappa_2(X)\ge\|(I+iN)^{-1}\|/\kappa_2(H)$. □

Choosing specific matrix decompositions $A=GH$ and imposing conditions on N in Lemma 3.1 allows us to deduce better bounds for $\kappa_2(A+iB)$.

Corollary 3.2 *Let $A\in\mathrm{GL}_n(\mathbb{R})$ and $B\in\mathbb{R}^{n\times n}$. Let $A=QR$ be a QR decomposition with $Q\in\mathrm{O}_n(\mathbb{R})$ and $R\in\mathbb{R}^{n\times n}$ upper triangular. If $N=Q^{\mathsf{T}}BR^{-1}$ is a normal matrix with eigenvalues $\lambda_1,\dots,\lambda_n\in\mathbb{C}$, then*

$$\kappa_2(A)\frac{\max_{k=1,\dots,n}|1+i\lambda_k|}{\min_{k=1,\dots,n}|1+i\lambda_k|}\ge\kappa_2(A+iB)\ge\max_{k=1,\dots,n}\frac{1}{|1+i\lambda_k|}.$$

Proof It suffices to observe that $\kappa_2(A)=\kappa_2(R)$, $\|A\|=\|R\|$, and N is unitarily diagonalizable. □

We may now show that for any well-conditioned $A\in\mathbb{R}^{n\times n}$, there is a well-conditioned $B\in\mathbb{R}^{n\times n}$ such that $A+iB\in\mathbb{C}^{n\times n}$ is arbitrarily ill-conditioned (i.e., $\gamma\to\infty$).

Theorem 3.3 *(Ill-conditioned matrices with well-conditioned real and imaginary parts) Let $A \in \mathrm{GL}_n(\mathbb{R})$ and $\gamma \geq 1$. Then there exists $B \in \mathbb{R}^{n \times n}$ such that*

$$\kappa_2(A) \geq \kappa_2(B), \qquad \kappa_2(A + iB) \geq \gamma.$$

Proof Consider the normal matrix

$$N = \begin{bmatrix} 0 & -t & 0 & \cdots & 0 \\ t & 0 & 0 & \cdots & 0 \\ 0 & 0 & t & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & t \end{bmatrix} \in \mathbb{R}^{n \times n}$$

where $t \geq 0$ is a real parameter to be chosen later. The eigenvalues of N are $\pm it$ and t so $\kappa_2(N) = 1$. Let $A = QR$ be the QR decomposition of A and set $B{:=}QNR$. Then $\kappa_2(B) \leq \kappa_2(N)\kappa_2(A) = \kappa_2(A)$. By Corollary 3.2, we have $\kappa_2(A + iB) \geq 1/(1-t)$. Hence if t is chosen in the interval $[1 - 1/\gamma, 1)$, we get $\kappa_2(A + iB) \geq \gamma$. □

Interestingly, we may use the Frobenius inversion formula to push Theorem 3.3 to the extreme, constructing an arbitrarily ill-conditioned complex matrix with perfectly conditioned real and imaginary parts.

Proposition 3.4 *Let $\gamma \geq 1$. There exists $A + iB \in \mathbb{C}^{n \times n}$ with $\kappa_2(A + iB) \geq \gamma$ and $\kappa_2(A) = \kappa_2(B) = 1$.*

Proof Let $Q \in \mathrm{O}_n(\mathbb{R})$. The Frobenius inversion formula (1) gives

$$\begin{aligned}(I + iQ)^{-1} &= Q^{\mathsf{T}}(Q^{\mathsf{T}} + Q)^{-1} - i(Q + Q^{\mathsf{T}})^{-1} = (Q^{\mathsf{T}} - iI)(Q + Q^{\mathsf{T}})^{-1} \\ &= (I - iQ)(Q^2 + I)^{-1}.\end{aligned}$$

An orthogonal matrix must have an eigenvalue decomposition of the form $Q = U\Lambda U^{\mathsf{H}}$ for some $U \in \mathrm{U}_n(\mathbb{C})$ and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{C}^{n \times n}$ with $|\lambda_1| = \cdots = |\lambda_n| = 1$. Therefore $I + iQ = U\,\mathrm{diag}(1 + i\lambda_1, \ldots, 1 + i\lambda_n)U^{\mathsf{H}}$ and

$$\begin{aligned}(I - iQ)(Q^2 + I)^{-1} &= U\,\mathrm{diag}\left(\frac{1 - i\lambda_1}{1 + \lambda_1^2}, \ldots, \frac{1 - i\lambda_n}{1 + \lambda_n^2}\right)U^{\mathsf{H}} \\ &= U\,\mathrm{diag}\left(\frac{1}{1 + i\lambda_1}, \ldots, \frac{1}{1 + i\lambda_n}\right)U^{\mathsf{H}}.\end{aligned}$$

Since the spectral norm is unitarily invariant,

$$\|I + iQ\| = \max_{k=1,\ldots,n} |1 + i\lambda_k|, \qquad \|(I - iQ)(Q^2 + I)^{-1}\| = \max_{k=1,\ldots,n} \left|\frac{1 - i\lambda_k}{1 + \lambda_k^2}\right|,$$

from which we deduce that

$$\kappa_2(I + iQ) = \left(\max_{k=1,\ldots,n} |1 + i\lambda_k|\right) \cdot \left(\max_{k=1,\ldots,n} \left|\frac{1 - i\lambda_k}{1 + \lambda_k^2}\right|\right) \geq \sqrt{2} \max_{k=1,\ldots,n} \left|\frac{1}{1 + i\lambda_k}\right|.$$

The last inequality follows from the fact that λ_k's are unit complex numbers. Now observe that if we choose λ_k to be sufficiently close to i, then $\kappa_2(I + iQ)$ can be made arbitrarily large. □

Note that Frobenius inversion (1) and Algorithm 3 avoids $A + iB$ and work instead with the matrices A, B, and $A + BA^{-1}B$. It is not difficult to tweak Theorem 3.3 to add $A + BA^{-1}B$ to the mix.

Theorem 3.5 *(Ill-conditioned matrices for Frobenius inversion). Let $A \in \mathrm{GL}_{2n}(\mathbb{R})$ and $\gamma \geq 1$. Then there exists $B \in \mathbb{R}^{2n\times 2n}$ such that*

$$\kappa_2(A) \geq \max\bigl(\kappa_2(B), \kappa_2(A + BA^{-1}B)\bigr), \qquad \kappa_2(A + iB) \geq \gamma.$$

In other words, for any invertible matrix A there exists a well-conditioned B such that $A + BA^{-1}B$ is well-conditioned but $A + iB$ is arbitrarily ill-conditioned.

Proof Consider the skew-symmetric (and therefore normal) matrix

$$N = \begin{bmatrix} 0 & -t & \cdots & 0 & 0 \\ t & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -t \\ 0 & 0 & \cdots & t & 0 \end{bmatrix} \in \mathbb{R}^{2n\times 2n},$$

where $t \geq 0$ is a real parameter to be chosen later. The eigenvalues of N are $\pm it$ so $\kappa_2(N) = 1$. Let $A = QR$ be the QR decomposition of A and set $B := QNR$. Then $\kappa_2(B) \leq \kappa_2(N)\kappa_2(A) = \kappa_2(A)$. By Corollary 3.2, we have $\kappa_2(A + iB) \geq 1/(1 - t)$. Hence if t is chosen in the interval $[1 - 1/\gamma, 1)$, we get $\kappa_2(A + iB) \geq \gamma$. We also have

$$A + BA^{-1}B = QR + (QNR)(R^{-1}Q^{\mathsf{T}})(QNR) = Q(I + N^2)R$$

and as $I + N^2 = (1 - t^2)I$, we see that $\kappa_2(I + N^2) = 1$ and so

$$\kappa_2(A + BA^{-1}B) \leq \kappa_2(I + N^2)\kappa_2(A) \leq \kappa_2(A).$$

□

Theorems 3.3 and 3.5 show the existence of arbitrarily ill-conditioned complex matrices with well-conditioned real and imaginary parts (and also $A + BA^{-1}B$ in the case of Theorem 3.5). We next show that such matrices exist in abundance — not only are there uncountably many of them, they occur with nonzero probability, showing that there is no shortage of matrices where Algorithm 3 provides an edge by avoiding the ill-conditioning of $A + iB$ or, equivalently, of $\left[\begin{smallmatrix} A & -B \\ B & A \end{smallmatrix}\right]$.

Proposition 3.6 *Let $\mathcal{S}_n := \{A + iB \in \mathrm{GL}_n(\mathbb{C}) : A, B \in \mathrm{GL}_n(\mathbb{R})\}$. For any $1 < \beta \leq \gamma < \infty$,*

$$\begin{aligned}
&\big\{A + iB \in \mathcal{S}_n : \kappa_2(B) \leq \kappa_2(A) \leq \beta,\ \kappa_2(A + iB) \geq \gamma\big\},\\
&\big\{A + iB \in \mathcal{S}_{2n} : \max\big(\kappa_2(B)\kappa_2(A + BA^{-1}B)\big) \leq \kappa_2(A) \leq \beta,\ \kappa_2(A + iB) \geq \gamma\big\}
\end{aligned}$$

have nonempty interiors in $\mathbb{C}^{n\times n}$ and $\mathbb{C}^{2n\times 2n}$ respectively.

Proof Consider the maps $\varphi_1 : \mathcal{S}_n \to [1,\infty) \times \mathbb{R} \times [1,\infty)$ and $\varphi_2 : \mathcal{S}_{2n} \to [1,\infty] \times \mathbb{R} \times \mathbb{R} \times [1,\infty)$ defined by

$$\begin{aligned}
\varphi_1(A + iB) &= \big(\kappa_2(A), \kappa_2(A) - \kappa_2(B), \kappa_2(A + iB)\big),\\
\varphi_2(A + iB) &= \big(\kappa_2(A), \kappa_2(A) - \kappa_2(B), \kappa_2(A) - \kappa_2(A + BA^{-1}B), \kappa_2(A + iB)\big)
\end{aligned}$$

respectively. These are continuous since the condition number κ_2 is a continuous function on invertible matrices. For any $\gamma \geq \beta > 1$, the preimage $\varphi_1^{-1}([1,\beta] \times [0,\infty) \times [\gamma,\infty)) \neq \varnothing$ by Theorem 3.3 and $\varphi_2^{-1}((1,\beta] \times [0,\infty) \times [0,\infty) \times [\gamma,\infty)) \neq \varnothing$ by Theorem 3.5. Note that these preimages are precisely the required sets in question and by continuity of φ_1 and φ_2 they must have nonempty interiors. $\square$

One may wonder if there is a flip side to Theorems 3.3 and 3.5, i.e., are there complex matrices whose condition numbers are controlled by their real and imaginary parts? We conclude this section by giving a construction of such matrices.

Proposition 3.7 *Let $A, B \in \mathrm{GL}_n(\mathbb{R})$. If $\sigma_n(A) = \mu\sigma_1(B)$ for some $\mu > 1$, then*

$$\frac{\kappa_2(A) - 1}{2} < \kappa_2(A + iB) \leq \kappa_2(A) + \frac{\kappa_2(A) + 1}{\mu - 1}$$

In particular, if A is well-conditioned and $\mu \gg 1$, then $A + iB$ is also well-conditioned.

Proof We first show a more generally inequality that holds for arbitrary $X, Y \in \mathbb{C}^{n\times n}$. Recall that singular values satisfy

$$\sigma_{i+j-1}(X + Y) \leq \sigma_i(X) + \sigma_j(Y), \quad 1 \leq i + j - 1 \leq n.$$

In particular, we have $\sigma_1(X + Y) \leq \sigma_1(X) + \sigma_1(Y)$ and $\sigma_1((X + Y) + (-Y)) \leq \sigma_1(X + Y) + \sigma_1(Y)$, and therefore

$$\sigma_1(X) - \sigma_1(Y) \leq \sigma_1(X + Y) \leq \sigma_1(X) + \sigma_1(Y).$$

Also, we have $\sigma_n(X + Y) \leq \sigma_n(X) + \sigma_1(Y)$ and $\sigma_n((X + Y) + (-Y)) \leq \sigma_n(X + Y) + \sigma_1(Y)$, and therefore

$$\sigma_n(X) - \sigma_1(Y) \leq \sigma_n(X + Y) \leq \sigma_n(X) + \sigma_1(Y).$$

If $\sigma_n(X) > \sigma_1(Y)$, then

$$\frac{\sigma_1(X) - \sigma_1(Y)}{\sigma_n(X) + \sigma_1(Y)} \le \frac{\sigma_1(X+Y)}{\sigma_n(X+Y)} \le \frac{\sigma_1(X) + \sigma_1(Y)}{\sigma_n(X) - \sigma_1(Y)}.$$

Rewriting in terms of condition number,

$$\frac{\kappa_2(X)\sigma_n(X) - \sigma_1(Y)}{\sigma_n(X) + \sigma_1(Y)} \le \kappa_2(X+Y) \le \frac{\kappa_2(X)\sigma_n(X) + \sigma_1(Y)}{\sigma_n(X) - \sigma_1(Y)}.$$

Hence

$$\kappa_2(X) - (\kappa_2(X)+1)\left[\frac{\sigma_1(Y)}{\sigma_n(X) + \sigma_1(Y)}\right] \le \kappa_2(X+Y)$$
$$\le \kappa_2(X) + (1+\kappa_2(X))\left[\frac{\sigma_1(Y)}{\sigma_n(X) - \sigma_1(Y)}\right].$$

Since $\sigma_n(X) > \sigma_1(Y)$, we have

$$\frac{\sigma_1(Y)}{\sigma_n(X) + \sigma_1(Y)} < \frac{1}{2}$$

and so $\kappa_2(X+Y) > (\kappa_2(X) - 1)/2$. If we set $X = A$, $Y = iB$, and substitute $\sigma_n(A) = \mu\sigma_1(B)$, the required inequality follows. □

4 Computing explicit inverse with Frobenius inversion

We have discussed at length in Sect. 1.2 why computing an explicit inverse for a matrix is sometimes an inevitable or desirable endeavor. Here we will discuss the numerical properties of inverting a complex matrix using Frobenius inversion. The quadratic extension $\mathbb{C}$ over $\mathbb{R}$ falls under Algorithm 1 (as opposed to Algorithm 2) and here we will compare its computational complexity with the complex matrix inversion algorithm based on LU decomposition, the standard method of choice for computing explicit inverse in MATLAB, Maple, Julia, and Python.

Algorithm 4 Inversion with LU decomposition

```
Input: $X \in \mathrm{GL}_n(\mathbb{C})$
1: LU factorize $X = P^{\mathsf{T}}LU$;
2: backward substitute for $X_1$ in $UX_1 = I$;
3: forward substitute for $X_2$ in $X_2L = X_1$;
Output: inverse $X^{-1} = X_2P$
```

Strictly speaking, Algorithm 4 computes the left inverse of the input matrix X, i.e., $YX = I$. We may also compute its right inverse, i.e., $XY = I$, by swapping the order of backward and forward substitutions. Even though the left and right inverse of a matrix are always equal mathematically, i.e., in exact arithmetic, they can be different numerically, i.e., in floating-point arithmetic [37, Cover picture and Figure 14.1]. We will discuss this in Sect. 5.7. Nevertheless, any subsequent mentions of Algorithm 4 would also hold with its right inverse variant.

4.1 Floating point complexity

In Sect. 2, we discussed computational complexity of Frobenius inversion for $\mathsf{inv}_{n,\mathbb{F}}$ in units of $\mathsf{inv}_{n,\Bbbk}$, $\mathsf{mul}_{n,\Bbbk}$, $\mathsf{add}_{n,\Bbbk}$, which are in turn treated as black boxes. Here, for the case of $\mathbb{F} = \mathbb{C}$ and $\Bbbk = \mathbb{R}$, we will count actual real flops, i.e., real floating point operations; we will not distinguish between the cost of real addition and real multiplication since there is no noticeable difference in their latency on modern processors—each would count as a single flop. With this in mind, we do not use Gauss multiplication for complex *numbers* since it trades one real multiplication for three real additions, i.e., more expensive if real addition costs the same as real multiplication. We caution our reader that this says nothing about Gauss multiplication for complex *matrices* since real matrix addition (n^2 flops) is still much cheaper than real matrix multiplication ($2n^3$ flops).

Our implementation of Frobenius inversion in Algorithm 1 requires real matrix multiplication and real matrix inversion as subroutines. Let $\mathscr{A}_{\mathsf{inv}}$ and $\mathscr{A}_{\mathsf{mul}}$ be respectively any two algorithms for real matrix inversion and real matrix multiplication, with real flop counts $T_{\mathsf{inv}}(n)$ and $T_{\mathsf{mul}}(n)$ on real $n \times n$ matrix inputs. There is little loss of generality in making two mild assumptions about the inversion algorithm $\mathscr{A}_{\mathsf{inv}}$:

(i) $\mathscr{A}_{\mathsf{inv}}$ also works for complex matrix inputs at the cost of a multiple of $T_{\mathsf{inv}}(n)$, the multiple being the cost of a complex flop in terms of real flops;
(ii) the number of complex additions and the number of complex multiplications in $\mathscr{A}_{\mathsf{inv}}$ applied to complex matrix inputs both take the form $cn^k +$ lower order terms, i.e., same dominant term but lower order terms may differ.

Note that these assumptions are satisfied if $\mathscr{A}_{\mathsf{inv}}$ is chosen to be Algorithm 4, even if we replace the LU decomposition in them by other decompositions like QR or Cholesky (if applicable).

Theorem 4.1 *(Frobenius inversion versus standard inversion). Let $\lambda > 0$ be such that the cost of computing $A^{-1}B$ for any $A \in \mathrm{GL}_n(\mathbb{R})$, $B \in \mathbb{R}^{n\times n}$ is bounded by $\lambda T_{\mathsf{mul}}(n)$. Algorithm 1 with subroutines $\mathscr{A}_{\mathsf{inv}}$ and $\mathscr{A}_{\mathsf{mul}}$ on real inputs A and B is asymptotically faster than directly applying $\mathscr{A}_{\mathsf{inv}}$ on complex input $A + iB$ if and only if*

$$\lim_{n\to\infty} \frac{T_{\mathsf{inv}}(n)}{T_{\mathsf{mul}}(n)} > \frac{2+\lambda}{3}.$$

Proof The first two steps of Algorithm 1 computes $A^{-1}B$, which costs $\lambda T_{\mathsf{mul}}(n)$ operations. Thereafter, computing $BA^{-1}B$ costs one matrix multiplication, $A + BA^{-1}B$ one matrix addition, $S = (A + BA^{-1}B)^{-1}$ one matrix inversion, and finally $A^{-1}BS$ one matrix multiplication. We disregard matrix addition since it takes $O(n^2)$ flops and does not contribute to the dominant term. So the cost in real flops of Algorithm 1 is dominated by $T_{\mathsf{inv}}(n) + (2 + \lambda)T_{\mathsf{mul}}(n)$ for n sufficiently large.

Now suppose we apply $\mathscr{A}_{\mathsf{inv}}$ directly to the complex matrix $A + iB$. Each complex addition costs two real flops (real additions) and each complex multiplication costs six real flops (four real multiplications and two real additions). Also, by assumption (ii), $\mathscr{A}_{\mathsf{inv}}$ has the same number of real additions and real multiplications, ignoring lower order terms. So the cost in real flops of $\mathscr{A}_{\mathsf{inv}}$ applied directly to $A + iB \in \mathbb{C}^{n\times n}$ is dominated by $4T_{\mathsf{inv}}(n)$ for n sufficiently large, i.e., the 'multiple' in assumption (i) is 4.

Hence Algorithm 1 is faster than $\mathscr{A}_{\mathsf{inv}}$ if and only if

$$4T_{\mathsf{inv}}(n) > T_{\mathsf{inv}}(n) + (2+\lambda)T_{\mathsf{mul}}(n)$$

for n sufficiently large, i.e., $\lim_{n\to\infty} T_{\mathsf{inv}}(n)/T_{\mathsf{mul}}(n) > (2+\lambda)/3$. □

As we discussed in Sect. 2.3, Algorithm 1 is written in a general form that applies over arbitrary fields and to both symbolic and numerical computing. However, when restricted to $\Bbbk = \mathbb{R}$, $\mathbb{F} = \mathbb{C}$, and with numerical computing in mind, we may state a more specific version Algorithm 5 involving LU decomposition and backsubstitutions for $AX = B$.

Algorithm 5 Frobenius inversion with LU decomposition

```
Input: $X = A + iB \in \mathrm{GL}_n(\mathbb{C})$ with $A \in \mathrm{GL}_n(\mathbb{R})$
1: LU factorize $A = P_1^{\mathsf{T}} L_1 U_1$;
2: forward and backward substitute for $X_1$ in $L_1 U_1 X_1 = P_1 B$;
3: matrix multiply and add $X_2 = A + BX_1$;
4: LU factorize $X_2 = P_2^{\mathsf{T}} L_2 U_2$;
5: forward and backward substitute for $X_3, X_4$ in $[X_3, X_4] P_2 L_2 U_2 = [I, X_1]$;
Output: inverse $X^{-1} = X_3 - iX_4$
```

Note that Steps 1 and 2 in Algorithm 5 are essentially just Algorithm 4 with a different right-hand side, and likewise for Steps 4 and 5. Hence we may regard Algorithm 5 as Algorithm 1 with $\mathscr{A}_{\mathsf{inv}}$ given by Algorithm 4 and $\mathscr{A}_{\mathsf{mul}}$ given by standard matrix multiplication. These choices allow us to assign flop counts to illustrate Theorem 4.1.

Proposition 4.2 *(Flop counts) Algorithm 5 has a real flop count of* $28n^3/3$ *whereas applying Algorithm 4 directly to a complex matrix has a real flop count of* $32n^3/3$.

Proof These come from a straightforward flop count of the respective algorithms, dropping lower order terms. □

With these choices for $\mathscr{A}_{\mathsf{inv}}$ and $\mathscr{A}_{\mathsf{mul}}$, the cost of computing $A^{-1}B$ is asymptotically bounded by $\frac{4}{3}T_{\mathsf{mul}}(n)$ — one LU decomposition plus $2n$ forward and backward substitutions. So $\lambda = 4/3$ and $(2+\lambda)/3 = 10/9 < 4/3 = \lim_{n\to\infty} T_{\mathsf{inv}}(n)/T_{\mathsf{mul}}(n)$. Hence inverting a complex matrix via Algorithm 5 is indeed faster than inverting it directly with Algorithm 4, as predicted by Theorem 4.1; we will also present numerical evidence that supports this in Sect. 5.

While we have other choices for $\mathscr{A}_{\mathsf{inv}}$ and $\mathscr{A}_{\mathsf{mul}}$, notably those based on Strassen-type algorithms [1, 7, 14, 70, 73, 74], it is difficult to determine their precise complexity.

For example, while we know that the current best known algorithm [73] for $\mathscr{A}_{\mathsf{mul}}$ has complexity $cn^{2.371552}$, we have no idea what the value of c is. Another reason is that they are mostly impractical. As far as we know, apart from Strassen's original algorithm in [70], none of the others has ever been implemented, let alone used; and their numerical stability is still a complete mystery. Given such practical considerations regarding the choices of $\mathscr{A}_{\mathsf{mul}}$ and $\mathscr{A}_{\mathsf{inv}}$, Proposition 4.2 is the most useful immediate result we may deduce from Theorem 4.1.

Proposition 4.2 also tells us that variations of Frobenius inversion formula like the one proposed in [78] can obliterate the computational savings afforded by Frobenius inversion. These variants all take the form $X^{-1} = (ZX)^{-1}Z$ for some $Z \in \mathrm{GL}_n(\mathbb{C})$ and the extra matrix multiplications incur additional costs. As a result, the variant in [78] takes $34n^3$ real flops, which exceeds even the $32n^3/3$ by standard methods (e.g., Algorithm 4).

4.2 Almost sure Frobenius inversion

One obvious shortcoming of Frobenius inversion is that (1) requires the real part A to be invertible. It is easy to modify (1) to

$$(A + iB)^{-1} = B^{-1}A(AB^{-1}A + B)^{-1} - i(AB^{-1}A + B)^{-1}$$

if B is invertible. Nevertheless we may well have circumstances where $A + iB$ is invertible but neither A nor B is, e.g., $\left[\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right]$. Here we will extend Frobenius inversion to any invertible $A + iB$ in a way that preserves its computational complexity — this last qualification is important. As we saw in Proposition 4.2, the speed improvement of Frobenius inversion comes from the constants, i.e., it inverts an $n \times n$ complex matrix with $28n^3/3$ real flops whereas Algorithm 4 takes $32n^3/3$ real flops. As we noted in Sect. 1.3, prior attempts such as those in [25, 78] at extending Frobenius inversion to all $A + iB \in \mathrm{GL}_n(\mathbb{C})$ invariably require the inversion of a real matrix of size $2n \times 2n$, thereby obliterating any speed improvement afforded by Frobenius inversion.

Our approach avoids any matrices of larger dimension by adding a simple randomization step that in turns depend on the following observation.

Lemma 4.3 *Let $A + iB \in \mathrm{GL}_n(\mathbb{C})$ with $A, B \in \mathbb{R}^{n\times n}$. Then there exist at most n values of $\mu \in \mathbb{R}$ such that $A - \mu B$ is singular.*

Proof As $f(t) := \det(A + tB)$ is a polynomial of degree at most n and $f(i) = \det(A + iB) \neq 0$, f has at most n zeros in $\mathbb{C}$. So $A - \mu B$ is invertible for all but at most n values of $\mu \in \mathbb{C} \supseteq \mathbb{R}$. $\square$

Algorithm 6 is essentially Frobenius inversion applied to $(1+\mu i)(A+iB)$ for some random μ. Note that $A - \mu B$ is exactly the real part of $(1 + \mu i)(A + iB)$ and therefore invertible for all but at most n values of μ by Lemma 4.3. The interval $(0, 1)$ is chosen so that we may generate μ from the uniform distribution but we could have also used $\mathbb{R}$ with standard normal distribution, both of which are standard implementations in numerical packages.

Algorithm 6 Almost sure Frobenius inversion

```
Input: $X = A + iB \in \mathrm{GL}_n(\mathbb{C})$
1: randomly generate $\mu \in [0, 1]$;
2: matrix add $X_1 = A - \mu B$, $X_2 = \mu A + B$;
3: Frobenius invert $X_3 + iX_4 = (X_1 + iX_2)^{-1}$;    ▷ calls Algorithm 5
4: matrix add $X_5 = X_3 - \mu X_4$, $X_6 = \mu X_3 + X_4$;
Output: inverse $X^{-1} = X_5 + iX_6$
```

Note that Algorithm 5 fails on a set of real dimension $2n^2 - 1$, i.e., when the real part of the input is singular, but Algorithm 6 has reduced this to a set of dimension zero.

Proposition 4.4 *Algorithm 6 has the same asymptotic time complexity as that of Algorithm 5, i.e., Frobenius inversion. Algorithm 6 works with probability one if μ is chosen randomly from* $[0, 1]$ *with any non-atomic probability measure.*

Proof The time complexity of Algorithm 6 is that of Algorithm 5 plus the matrix additions in Steps 2 and 4 that cost a total of $4 \times 2n^2$ real flops. By Proposition 4.2, the time complexity of Algorithm 6 is dominated by $28n^3/3$, and therefore the lower order term $8n^2$ may be ignored asymptotically.

By Lemma 4.3, $X_1 = A - \mu B$ in Step 2 is invertible with probability one since any finite subset of $[0, 1]$ is a null set with a non-atomic probability measure. Thus Algorithm 5 is applicable to $X_1 + iX_2$ and we have

$$\begin{aligned}(X_3 - \mu X_4) + i(\mu X_3 + X_4) &= (1 + \mu i)(X_1 + iX_2)^{-1}\\ &= (1 + \mu i)\big(X(1 + \mu i)\big)^{-1} = X^{-1}.\end{aligned}$$

The almost sure correctness of Algorithm 6 follows. □

We provide Algorithm 6 and Proposition 4.4 as a proof of concept, i.e., the requirement of nonsingularity of A in Frobenius inversion is not a serious limitation and can be circumvented without compromising computational complexity. In numerical linear algebra, the singularity of a matrix is not a binary notion (yes or no) but a continuous one captured by its condition number. Indeed we will not use Algorithm 6 anywhere in our actual numerical experiments in Sect. 5. A numerically relevant version of Algorithm 6 would involve selecting μ as a preconditioning parameter but this requires careful study and is listed among our future works in Sect. 6.

4.3 Hermitian positive definite matrices

The case of Hermitian positive definite $A + iB$ deserves special consideration given their ubiquity. We will propose and analyze a new variant of Frobenius inversion that exploits this special structure of $A + iB$. The happy coincidence is that a Hermitian positive definite $A + iB \in \mathbb{C}^{n\times n}$ must necessarily have symmetric positive definite A and $A + BA^{-1}B \in \mathbb{R}^{n\times n}$ as well as a skew-symmetric $B \in \mathbb{R}^{n\times n}$—precisely the matrices we need in Frobenius inversion. Various required quantities may thus be computed via Cholesky decompositions $A = R_1^{\mathsf{T}} R_1$ and $A + BA^{-1}B = R_2^{\mathsf{T}} R_2$:

$$\begin{aligned} A^{-1}B &= R_1^{-1}R_1^{-\mathsf{T}}B, \\ BA^{-1}B &= BR_1^{-1}R_1^{-\mathsf{T}}B = -(R_1^{-\mathsf{T}}B)^{\mathsf{T}}(R_1^{-\mathsf{T}}B), \\ (A+BA^{-1}B)^{-1} &= \big(A-(R_1^{-\mathsf{T}}B)^{\mathsf{T}}(R_1^{-\mathsf{T}}B)\big)^{-1} = R_2^{-1}R_2^{-\mathsf{T}}, \\ A^{-1}B(A+BA^{-1}B)^{-1} &= A^{-1}BR_2^{-1}R_2^{-\mathsf{T}}. \end{aligned} \tag{22}$$

Lemma 4.5 *Let $A+iB \in \mathbb{C}^{n\times n}$ be a Hermitian positive definite matrix with $A, B \in \mathbb{R}^{n\times n}$. Then*

(i) A is symmetric positive definite and B is skew-symmetric;
(ii) $A+BA^{-1}B$ is symmetric positive definite.

In particular, A is always invertible and so there is no need for an analogue of Algorithm 6.

Proof Let $X = A+iB$ and write $X \succ 0$ to indicate positive definiteness. Then since $A = (X+\overline{X})/2$ and $B = (X-\overline{X})/2i$, A is symmetric and B is skew-symmetric given that X is Hermitian. Since $X \succ 0$, for any $x \in \mathbb{R}^n$,

$$x^{\mathsf{T}}Ax = \frac{1}{2}x^{\mathsf{H}}(X+\overline{X})x = \frac{1}{2}x^{\mathsf{H}}Xx + \frac{1}{2}\overline{x^{\mathsf{H}}Xx} = x^{\mathsf{H}}Xx \geq 0,$$

with equality if and only if $x=0$, showing that A is positive definite. Again since $X \succ 0$, for any $z \in \mathbb{C}^n$,

$$z^{\mathsf{H}}\overline{X}z = \overline{\overline{z}^{\mathsf{H}}X\overline{z}} \geq 0,$$

with equality if and only if $z=0$; so $\overline{X}$ is also Hermitian positive definite. Now observe that $A^{-\frac{1}{2}}XA^{-\frac{1}{2}} = I + iA^{-\frac{1}{2}}BA^{-\frac{1}{2}} \succ 0$ and

$$A+BA^{-1}B = A^{\frac{1}{2}}\big[I+(A^{-\frac{1}{2}}BA^{-\frac{1}{2}})(A^{-\frac{1}{2}}BA^{-\frac{1}{2}})\big]A^{\frac{1}{2}}.$$

Therefore it suffices to establish (ii) for $A=I$. As

$$I+iB \succ 0, \quad I-iB \succ 0, \quad I+B^2 = (I-iB)^{\frac{1}{2}}(I+iB)(I-iB)^{\frac{1}{2}},$$

it follows that $I+B^2 \succ 0$. An alternative way to show (ii) is to use Lemma 2.4, which informs us that $(A+BA^{-1}B)^{-1}$ is the real part of X^{-1}, allowing us to invoke (i). Then $X \succ 0 \Rightarrow X^{-1} \succ 0 \Rightarrow (A+BA^{-1}B)^{-1} \succ 0 \Rightarrow A+BA^{-1}B \succ 0$. □

With Lemma 4.5 established, we may turn (22) into Algorithm 7, which essentially replaces the LU decompositions in Algorithm 5 with Cholesky decompositions, taking care to preserve symmetry and positive definiteness.

The standard method for inverting a Hermitian positive definite matrix is to simply replace LU decomposition in Algorithm 4 by Cholesky decomposition, given in Algorithm 8 for easy reference.

Algorithm 7 Frobenius inversion with Cholesky decomposition

```
Input: $X = A + iB$ with $A \in \mathrm{GL}_n(\mathbb{R})$
1: Cholesky decompose $A = R_1^\mathsf{T} R_1$;
2: backward substitute for $X_1$ in $R_1^\mathsf{T} X_1 = B$;
3: forward substitute for $X_2$ in $R_1 X_2 = X_1$;
4: matrix multiply $X_3 = X_1^\mathsf{T} X_1$;
5: matrix add $X_4 = A - X_3$;
6: Cholesky decompose $X_4 = R_2^\mathsf{T} R_2$;
7: backward substitute for $X_5$ in $R_2^\mathsf{T} X_5 = I$;
8: forward substitute for $X_6$ in $R_2 X_6 = X_5$;
9: matrix multiply $X_7 = X_2 X_6$;
Output: inverse $X^{-1} = X_6 - iX_7$
```

Algorithm 8 Inversion with Cholesky decomposition

```
Input: $A \in \mathrm{GL}_n(\mathbb{C})$
1: Cholesky decompose $A = R^\mathsf{H} R$;
2: backward substitute for $X_1$ in $R^\mathsf{H} X_1 = I$;
3: forward substitute for $X_2$ in $R X_2 = X_1$;
Output: inverse $A^{-1} = X_2$
```

With this, we obtain an analogue of Proposition 4.2. The flop counts below show that Algorithm 7 provides a 22% speedup over Algorithm 8. The experiments in Sect. 5.6 will attest to this improvement.

Proposition 4.6 *(Flop counts). Algorithm 7 has a real flop count of* $23n^3/3$ *whereas applying Algorithm 8 directly to a complex matrix has a real flop count of* $28n^3/3$.

Proof Algorithm 8 performs one Cholesky decomposition, n backward substitutions, and n forward substitutions, all over $\mathbb{C}$. So its flop complexity is dominated by $n^3/3 + n^3 + n^3 = 7n^3/3$ complex flops and thus, by the same reasoning in the proof of Theorem 4.1, $28n^3/3$ real flops. On the other hand, Algorithm 7 performs two Cholesky decompositions, $2n$ backward substitutions, $2n$ forward substitutions, and two matrix multiplications, all over $\mathbb{R}$. Moreover, the symmetry in $X_1^\mathsf{T} X_1$ allows the matrix multiplication in Step 4 to have a reduced complexity of n^3 real flops. Hence its flop complexity is dominated by $2n^3/3 + 2n^3 + 2n^3 + 3n^3 = 23n^3/3$ real flops. □

We end with an observation that the discussions in this section apply as long as $A \succ 0$ and $A + BA^{-1}B \succ 0$. Indeed, another important class of matrices with this property are the $A + iB \in \mathbb{C}^{n\times n}$ with symmetric positive definite real and imaginary parts, i.e., $A \succ 0$ and $B \succ 0$ [37, p. 209]. By Lemma 4.5, such matrices are not Hermitian positive definite except in the trivial case when $B = 0$. However, since such matrices must clearly satisfy $A + BA^{-1}B \succ 0$, Algorithm 7 and Proposition 4.6 will apply verbatim to them.

5 Numerical experiments

We present results from numerical experiments comparing the speed and accuracy of Frobenius inversion (Algorithms 3, 5, 7) with standard methods via LU and Cholesky

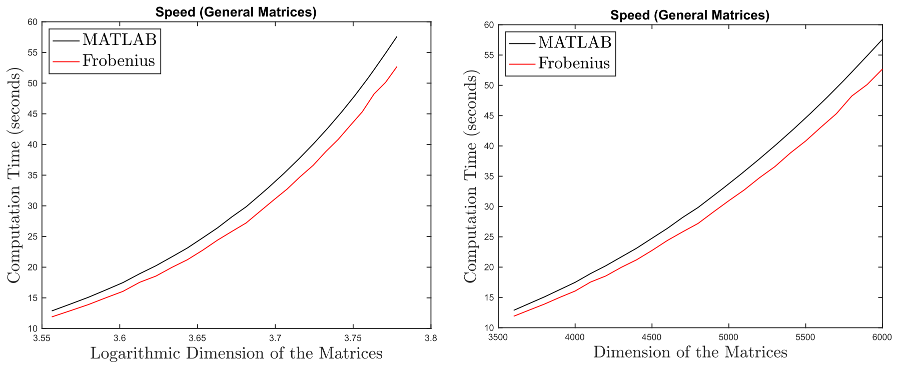


Fig. 1 Time taken versus log-dimension (*left*) and dimension (*right*) of matrix

decompositions (Algorithms 4, 8). We begin by comparing Algorithms 4 and 5, followed by a variety of common tasks: linear systems, matrix sign function, Sylvester equations, Lyapunov equations, polar decomposition, and rounding up with a comparison of Algorithms 7 and 8 on the inversion of Hermitian positive matrices. These results show that algorithms based on Frobenius inversion are more efficient than standard ones based on LU or Cholesky decompositions, with negligible loss in accuracy, confirming Theorem 4.1, Propositions 4.2 and 4.6. In all experiments, we repeat our random trials ten times and record average time taken and average forward or backward error. All codes are available at https://github.com/zhen06/Complex-Matrix-Inversion.

5.1 Matrix inversion

For our speed experiments, we generate $X = A + iB \in \mathbb{C}^{n\times n}$ with entries of $A, B \in \mathbb{R}^{n\times n}$ sampled uniformly from $[0, 1]$ and n from 3600 to 6000.

Figure 1 shows the times taken for MATLAB's built-in inversion (Algorithm 4), i.e., directly performing LU decomposition in complex arithmetic, and Frobenius inversion with LU decomposition in real arithmetic (Algorithm 5). They are plotted against matrix dimension n, using two different scales for the horizontal axis. As predicted by Proposition 4.2, Frobenius inversion is indeed faster than MATLAB's inversion, with a widening gap as n grows bigger.

For our accuracy experiments, we want some control over the condition numbers of our random matrices to reduce conditioning as a factor affecting accuracy. We generate a random $A \in \mathbb{R}^{n\times n}$ with condition number κ: first generate a random orthogonal $Q \in \mathrm{O}_n(\mathbb{R})$ by QR factoring a random $Y \in \mathbb{R}^{n\times n}$ with entries sampled uniformly from $[-1, 1]$; next generate a random diagonal $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n) \in \mathbb{R}^{n\times n}$ with $\lambda_1 = \pm\kappa$, $\lambda_n = \pm 1$, signs assigned randomly, and $\lambda_2, \dots, \lambda_{n-1} \in [-\kappa, -1] \cup [1, \kappa]$ sampled uniform randomly; then set $A{:=}Q\Lambda Q^{\mathsf{T}}/\|\Lambda\|_{\mathsf{F}}$. We generate $B \in \mathbb{R}^{n\times n}$ in the same way. So $\kappa_2(A) = \kappa_2(B) = \kappa$. We also check that $\kappa_2(X)$ is on the same order of magnitude as κ or otherwise discard X. In the plots below, we set $\kappa = 10$ and increase n from 2 through 4096.

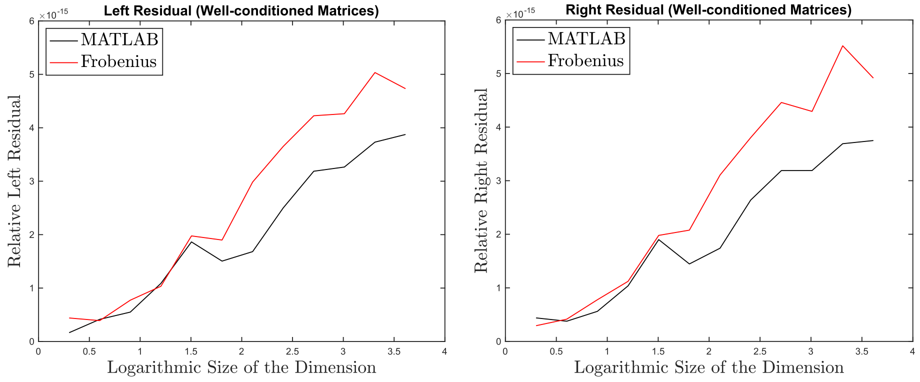


Fig. 2 Relative left and right residuals of Frobenius inversion versus MATLAB built-in inversion. Note that scale of the vertical axis is 10^{-15}

Accuracy is measured by left and right *relative residuals* defined respectively as

$$\operatorname{res}_L(X, \widehat{Y}) := \frac{\|\widehat{Y}X - I\|_{\max}}{\|X\|_{\max}\|\widehat{Y}\|_{\max}}, \qquad \operatorname{res}_R(X, \widehat{Y}) := \frac{\|X\widehat{Y} - I\|_{\max}}{\|X\|_{\max}\|\widehat{Y}\|_{\max}} \tag{23}$$

where $\widehat{Y}$ is the computed inverse of X and the *max norm* is

$$\|A + iB\|_{\max} := \max(\|A\|_{\max}, \|B\|_{\max}) := \max\Big(\max_{i,j=1,\dots,n} |a_{ij}|, \max_{i,j=1,\dots,n} |b_{ij}|\Big). \tag{24}$$

Figure 2 shows the left and right relative residuals computed by MATLAB's built-in inversion (Algorithm 4) and Frobenius inversion (Algorithm 5), plotted against matrix dimension n. At first glance, Frobenius inversion is less accurate than MATLAB's inversion. But one needs to look at the scale of the vertical axis — the two algorithms give essentially the same results to machine precision (15 decimal digits of accuracy), any difference can be safely ignored for all intents and purposes.

5.2 Solving linear systems

It is remarkable that Frobenius inversion shows nearly no loss in accuracy as measured by backward error. For the matrix dimensions in Sect. 5.1, forward error experiments are too expensive due to the cost of finding exact inverse. To get a sense of the forward errors, we look at a problem intimately related to matrix inversion—solving linear systems.

We use the same matrices generated in Sect. 5.1 and generate two vectors $x, y \in \mathbb{R}^n$ with entries sampled uniformly from $[-1, 1]$. We set $c + id := (A + iB)(x + iy)$ and solve the complex linear system $(A + iB)(x + iy) = c + id$ to get a computed solution $\widehat{x} + i\widehat{y}$ using three methods: (i) Frobenius inversion (Algorithm 3), (ii) complex LU factorization, and (iii) augmented system $\begin{bmatrix} A & -B \\ B & A \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}$; we rely on MATLAB's `mldivide` (i.e., the 'backslash' operator) for the last two.

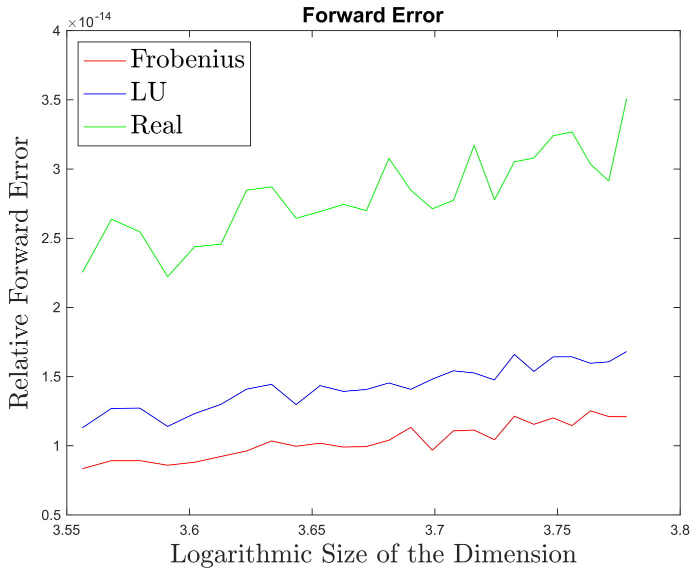


Fig. 3 Linear systems with Frobenius inversion and MATLAB's backslash

Figure 3 shows the relative forward errors $\|\widehat{x} + i\widehat{y} - (x + iy)\|_{\max}/\|x + iy\|_{\max}$ plotted against matrix dimension. The conclusion is clear: Frobenius inversion gives the most accurate result.

5.3 Matrix sign function

The matrix sign function appears in a wide range of problems such as algebraic Riccati equation [63], Sylvester equation [36, 63], polar decomposition [36], and spectral decomposition [3–5, 40, 49]. For $X \in \mathrm{GL}_n(\mathbb{C})$ with Jordan decomposition $X = ZJZ^{-1}$ where its Jordan canonical form $J = \begin{bmatrix} J_+ & 0 \\ 0 & J_- \end{bmatrix}$ is partitioned into $J_+ \in \mathbb{C}^{p\times p}$ with positive real part and $J_- \in \mathbb{C}^{q\times q}$ with negative real part, its matrix sign function is defined to be

$$\mathrm{sign}(X) = Z \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} Z^{-1}. \tag{25}$$

Since the Jordan decomposition cannot be determined in finite precision [32], its definition does not offer a viable way of computation. The standard way to evaluate the matrix sign function is to use Newton iterations [38, 63]:

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \quad k = 0, 1, 2, \dots, \quad X_0 = X. \tag{26}$$

This affords a particularly pertinent test for Frobenius inversion as it involves repeated inversion. Our stopping condition is given by the relative change in X_k: We stop when $\|X_k - X_{k-1}\|_{\max}/\|X_k\|_{\max} \le \varepsilon = 10^{-3}$ or when $k \ge k_{\max} = 100$.

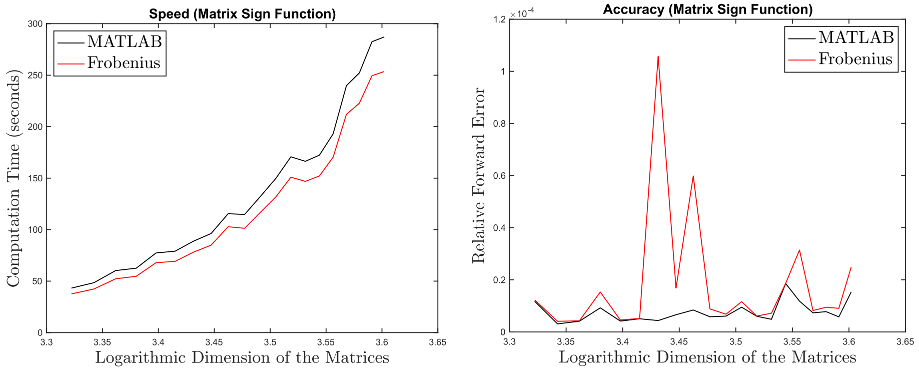


Fig. 4 Matrix sign function with Frobenius inversion and MATLAB's inversion

The definition via Jordan decomposition is useful for generating random examples for our tests: We generate a random diagonal $J \in \mathbb{C}^{n\times n}$ whose first $p \approx n/2$ diagonal entries have positive real parts and the rest have negative real parts, avoiding near zero values, and with n from 2100 to 4000. We generate a random $Z \in \mathrm{GL}_n(\mathbb{C})$ with real and imaginary parts of its entries z_{ij} sampled uniformly from $[-1, 1]$. We set $X := ZJZ^{-1}$. In this way we obtain $\operatorname{sign}(X)$ via (25) as well.

In each iteration of (26), we compute X_k^{-1} with MATLAB's inversion in complex arithmetic (Algorithm 4) and Frobenius inversion in real arithmetic (Algorithm 5). Accuracy is measured by relative forward error $\|\operatorname{sign}(X) - \widehat{S}\|_{\max}/\|\operatorname{sign}(X)\|_{\max}$. From Fig. 4, we see that Frobenius inversion offers an improvement in speed at the cost of slightly less accurate results.

5.4 Sylvester and Lyapunov equations

One application of the matrix sign function is to seek, for given $A \in \mathbb{C}^{p\times p}$, $B \in \mathbb{C}^{q\times q}$, and $C \in \mathbb{C}^{p\times q}$, a solution $Y \in \mathbb{C}^{p\times q}$ for the *Sylvester equation*:

$$AY + YB = C,$$

or its special case with $B = A^{\mathsf{H}}$, the *Lyapunov equation* [37]. As noted in [36, 63], if $\operatorname{sign}(A) = I_p$ and $\operatorname{sign}(B) = I_q$, then

$$\operatorname{sign}\left(\begin{bmatrix} A & -C \\ 0 & -B \end{bmatrix}\right) = \begin{bmatrix} I_p & -2Y \\ 0 & -I_q \end{bmatrix}.$$

Thus the Newton iterations (26) applied to $X_0 = \left[\begin{smallmatrix} A & -C \\ 0 & -B \end{smallmatrix}\right]$ will converge to $\left[\begin{smallmatrix} I & -2Y \\ 0 & -I \end{smallmatrix}\right]$, yielding the solution Y of Sylvester equation in the limit.

As usual, we 'work backwards' to generate $A \in \mathbb{C}^{p\times p}$ with $\operatorname{sign}(A) = I_p$, $B \in \mathbb{C}^{q\times q}$ with $\operatorname{sign}(B) = I_q$, and $C \in \mathbb{C}^{p\times q}$ with p and q taking values between 1050 and 2000. First we generate a random $Z \in \mathrm{GL}_p(\mathbb{C})$ by sampling the real and imaginary parts of its entries in $[-1, 1]$ uniformly; next we generate a random diagonal $J \in \mathbb{C}^{n\times n}$

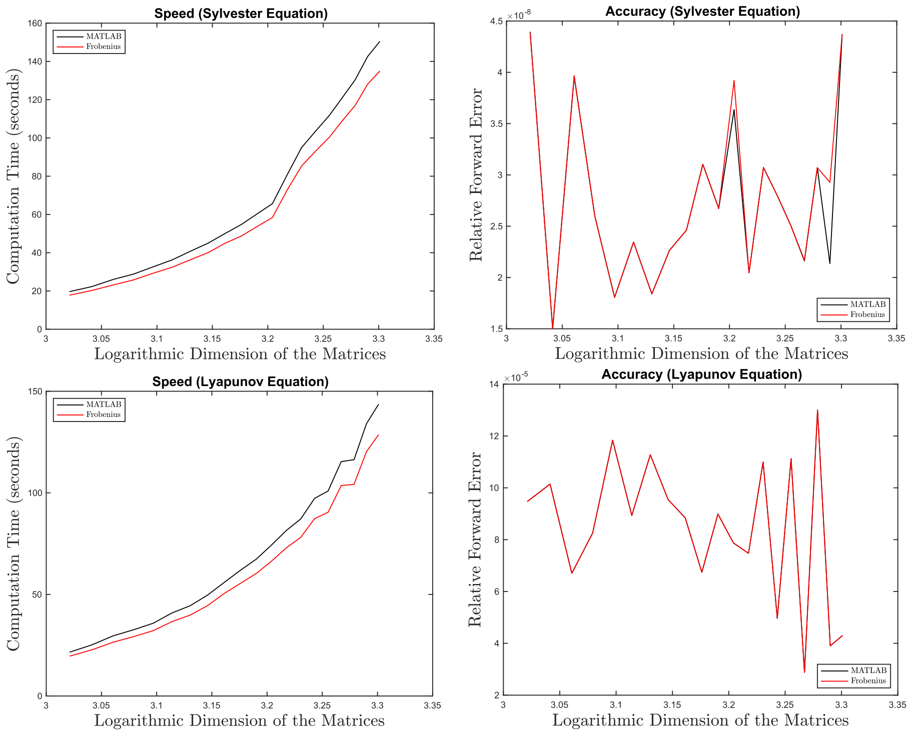


Fig. 5 Sylvester (*top*) and Lyapunov (*bottom*) equations with Frobenius inversion and MATLAB's inversion. Note there are two graphs in the bottom right plot

whose diagonal entries have positive real parts sampled from the interval [9, 10]; then we set $A:=ZJZ^{-1} \in \mathbb{C}^{p\times p}$. We generate $B \in \mathbb{C}^{q\times q}$ in the same way. We generate a random $Y \in \mathbb{C}^{p\times q}$ with real and imaginary parts of its entries sampled uniformly from $[-1, 1]$ and set $C:=AY + YB$.

Using the same stopping condition in Sect. 5.3 with a tolerance of $\varepsilon = 10^{-1}$ and $k_{\max} = 100$ maximum iterations, we compute a solution $\widehat{Y}$ with the Newton iterations (26). Accuracy is measured by relative forward error $\|Y - \widehat{Y}\|_{\max}/\|Y\|_{\max}$.

Figure 5 gives the results for Sylvester and Lyapunov equations, showing that in both cases Frobenius inversion is faster than MATLAB's inversion with no difference in accuracy. Indeed, at a scale of 10^{-5} for the vertical axis, the two graphs in the accuracy plot for Lyapunov equation (bottom right plot of Fig. 5) are indistinguishable. The accuracy plot for Sylvester equation (top right plot of Fig. 5) uses a finer vertical scale of 10^{-8}; but had we used a scale of 10^{-5}, the two graphs therein would also have been indistinguishable.

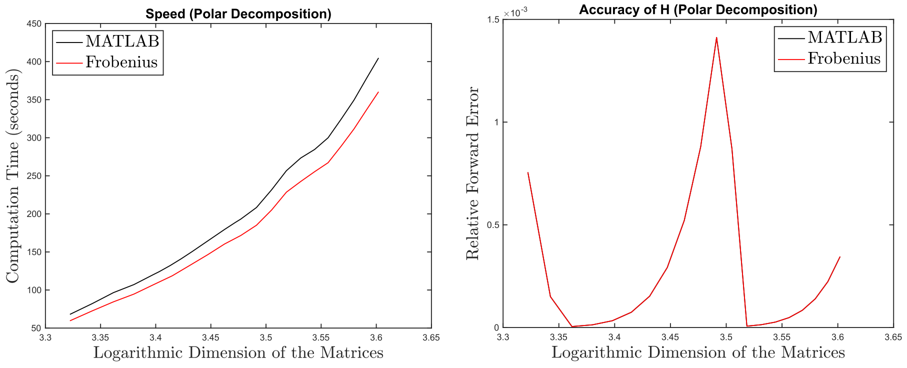


Fig. 6 Polar decomposition with Frobenius inversion and MATLAB's inversion. Note that there are two graphs in the right plot

5.5 Polar decomposition

Another application of the matrix sign function is to polar decompose a given $X \in \mathbb{C}^{n\times n}$ into $X = QP$ with $Q \in \mathrm{U}_n(\mathbb{C})$ and $P \in \mathbb{C}^{n\times n}$ Hermitian positive semidefinite. This is based on the observation [34, 36, 43] that

$$\operatorname{sign}\left(\begin{bmatrix} 0 & X \\ X^{\mathsf{H}} & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & Q \\ Q^{\mathsf{H}} & 0 \end{bmatrix}.$$

Here the Newton iterations (26) take a slightly different form

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-\mathsf{H}}), \quad k = 0, 1, 2, \dots, \quad X_0 = X. \tag{27}$$

We generate random $Y, Z \in \mathbb{C}^{n\times n}$ with real and imaginary parts of its entries sampled uniformly from $[-1, 1]$. We then QR factorize $Y = UR$ and set $P{:=}Z^{\mathsf{H}}Z$ and $X = UP$. The value of n runs from 2100 to 4000.

Using the same stopping condition in Sect. 5.3 with a tolerance of $\varepsilon = 10^{-3}$ and $k_{\max} = 100$ maximum iterations, we compute a solution $\widehat{Q}$ with the Newton iterations (27), with $X_k^{-\mathsf{H}}$ computed by either Frobenius inversion or MATLAB's inversion. We then set $\widehat{P} = \widehat{Q}^{\mathsf{H}}X$. Accuracy is measured by relative forward errors $\|Q - \widehat{Q}\|_{\max}/\|Q\|_{\max}$ and $\|P - \widehat{P}\|_{\max}/\|P\|_{\max}$.

Figure 6 again shows that Frobenius inversion is faster than MATLAB's built-in inversion with near-identical accuracy. Indeed, at a scale of 10^{-3} for the vertical axis, the two graphs in the accuracy plot (right plot of Fig. 6) are indistinguishable.

5.6 Hermitian positive definite matrix inversion

We repeat experiments in Sect. 5.1 on Hermitian positive definite matrices for our variant of Frobenius inversion (Algorithm 7) and MATLAB's built-in inversion based on

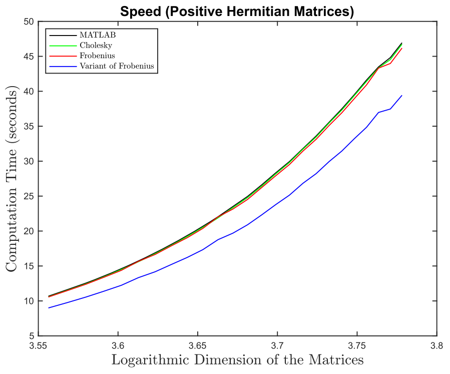
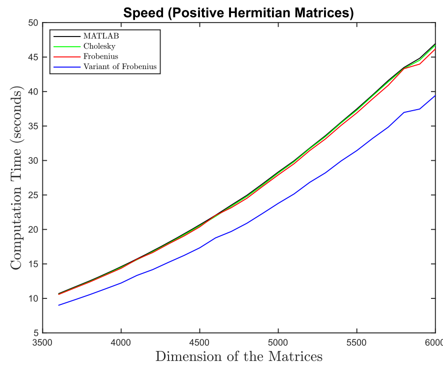
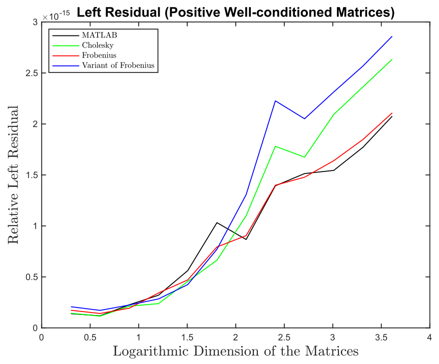


Fig. 7 Time taken versus log-dimension (*left*) and dimension (*right*) of matrix





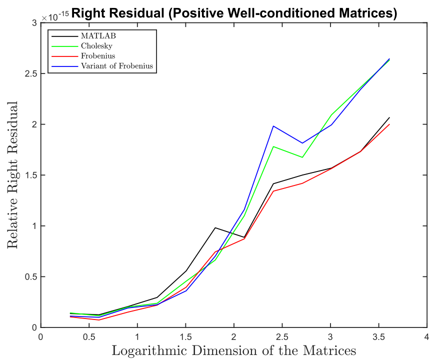


Fig. 8 Relative left and right residuals of Algorithms 4, 5, 7, 8. Note that scale of the vertical axis is 10^{-15}

Cholesky decomposition (Algorithm 8). For comparison, we also include Algorithms 4 and 5 that do not exploit Hermitian positive definiteness.

For our speed experiments, we generate a random Hermitian positive definite $X:=(A+iB)^{\mathsf{H}}(A+iB)+0.01I \in \mathbb{C}^{n\times n}$ with $A, B \in \mathbb{R}^{n\times n}$ sampled uniformly from $[-1, 1]$ and n from 3600 to 6000. We plot the results in Fig. 7, with two different scales for the horizontal axis.

For our accuracy experiments, we control the condition numbers of our matrices to reduce conditioning as a factor affecting accuracy. To generate a random Hermitian positive definite $X \in \mathbb{C}^{n\times n}$ with condition number κ, first we generate a random unitary $Q \in \mathrm{U}_n(\mathbb{C})$ by QR factoring a random $Y \in \mathbb{C}^{n\times n}$ with real and imaginary parts of its entries sampled uniformly from $[-1, 1]$; next we generate a random diagonal $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n) \in \mathbb{R}^{n\times n}$ with $\lambda_1 = \kappa$, $\lambda_n = 1$, and $\lambda_2, \dots, \lambda_{n-1} \in [1, \kappa]$ sampled uniform randomly; then we set $X:=Q\Lambda Q^{\mathsf{H}}/\|\Lambda\|_{\mathsf{F}}$. So $\kappa_2(X) = \kappa$. In the plots below, we set $\kappa = 10$ and increase n from 2 through 4096.

Accuracy is measured by left and right relative residuals as defined in equation (23), with results plotted in Fig. 8, which shows the left and right relative residuals computed by Algorithms 4, 5, 7, 8 plotted against matrix dimension n. The important thing to

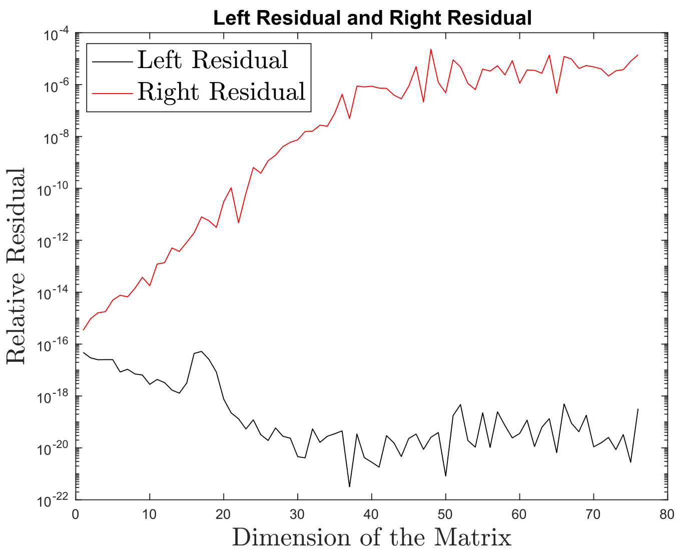


Fig. 9 A complex variant of the figure that appears on the cover of [37]

note is the scale of the vertical axes—all four algorithms give essentially the same results up to machine precision.

5.7 Left and right inverses

We dedicate this article to the memory of Nick Higham. The picture on the cover of his magnum opus [37], which also appears as Figure 14.1 therein, illustrates that the computed left and right inverses of a real matrix can be very different if the matrix is ill-conditioned. Figure 9 shows a variant of this picture for complex matrices, where we directly invert $A + iB$. Our results in Fig. 8, showing nearly matching left and right residuals, indicate that such anomalies may sometimes be avoided with care. In the Frobenius inversion algorithms, it does not matter if the matrix to be inverted $A + iB$ is ill-conditioned because these algorithms only work with its well-conditioned real and imaginary parts A and B.

6 Conclusion

We hope our effort here will rekindle interest in this beautiful algorithm. In future work, we plan to provide rounding error analysis for Frobenius inversion, discuss its relation with Strassen-style algorithms, and its advantage in solving linear systems with a large number of right-hand sides.

There are also interactions between results in different sections that we would like to consider in the future. For example, one may argue that for actual numerical computations, the "non-atomic probability measure" assumption in Proposition 4.4 does not hold since there are only a finite number of floating point numbers, each representing a set of real numbers that round to it via some standard rules [41, 59]. Indeed, in the context of numerical computations, it is critical to avoid not just singular matrices but any matrices that are near singular, in other words, ill-conditioned matrices. With this in mind, we will need to choose $\mu \in \mathbb{R}$ in a way that takes in account our conditioning results in Sect. 3.

Acknowledgements KY acknowledges the support of the National Key R & D Program of China 2023YFA1009401, CAS Project for Young Scientists in Basic Research YSBR-008 and National Natural Science Foundation of China 12288201. ZD acknowledges the support of DARPA HR00112190040 and NSF ECCF 2216912. LHL acknowledges the support of DARPA HR00112190040, NSF DMS 1854831, and a Vannevar Bush Faculty Fellowship ONR N000142312863. All three authors thank the anonymous reviewers for their very helpful comments and suggestions.

References

1. Alman, J., Vassilevska Williams, V.: A refined laser method and faster matrix multiplication. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 522–539. SIAM, Philadelphia, PA (2021)
2. Althaus, H., Leake, R.: Inverse of a finite-field Vandermonde matrix (corresp.). IEEE Trans. Inform. Theory **15**(1), 173–173 (1969)
3. Bai, Z., Demmel, J., Dongarra, J., Petitet, A., Robinson, H., Stanley, K.: The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers. SIAM J. Sci. Comput. **18**(5), 1446–1461 (1997)
4. Bai, Z., Demmel, J.W.: Design of a Parallel Nonsymmetric Eigenroutine Toolbox. University of Kentucky, Lexington (1992)
5. Beavers, A.N., Jr., Denman, E.D.: A computational method for eigenvalues and eigenvectors of a matrix with real eigenvalues. Numer. Math. **21**, 389–396 (1973)
6. Besicovitch, A.S.: On the linear independence of fractional powers of integers. J. Lond. Math. Soc. **15**, 3–6 (1940)
7. Bini, D., Capovani, M., Romani, F., Lotti, G.: $O(n^{2.7799})$ Complexity for $n \times n$ approximate matrix multiplication. Inform. Process. Lett. **8**(5), 234–235 (1979)
8. Bodewig, E.: Matrix Calculus. North-Holland, Amsterdam (1959)
9. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic complexity theory. Grundlehren der mathematischen Wissenschaften, vol. 315. Springer-Verlag, Berlin (1997)
10. Caraiani, A., Newton, J.: On the modularity of elliptic curves over imaginary quadratic fields. arXiv:2301.10509 (2023)
11. Casacuberta, S., Kyng, R.: Faster sparse matrix inversion and rank computation in finite fields. In 13th Innovations in Theoretical Computer Science Conference, ITCS,: 33:1–33:24, p. 2022. Dagstuhl Publishing, Germany (2022)
12. Cohen, H.: A course in computational algebraic number theory. Graduate Texts in Mathematics. Springer-Verlag, Berlin (1993)
13. Cohen, H.: Advanced topics in computational number theory. Graduate Texts in Mathematics. Springer-Verlag, New York (2000)
14. Coppersmith, D., Winograd, S.: On the asymptotic complexity of matrix multiplication. SIAM J. Comput. **11**(3), 472–492 (1982)
15. Cramér, H.: Mathematical Methods of Statistics. Princeton mathematical series, Princeton University Press, Princeton (1946)
16. Dai, Z., Lim, L.-H.: Numerical stability and tensor nuclear norm. Numer. Math. **155**(3–4), 345–376 (2023)
17. Ditkowski, A., Fibich, G., Gavish, N.: Efficient solution of $Ax^{(k)} = b^{(k)}$ using A^{-1}. J. Sci. Comput. **32**(1), 29–44 (2007)
18. Druinsky, A., Toledo, S.: How accurate is $\mathrm{inv}(A) * b$? arXiv:1201.6035 (2012)

19. Dudeck, K.: Solving complex systems using spreadsheets: a matrix decomposition approach. In Proceedings of the 2005 ASEE Annual Conference and Exposition: The Changing Landscape of Engineering and Technology Education in a Global World, ASEE, Washington, DC pp. 12875–12880 (2005)
20. Dummit, D.S., Foote, R.M.: Abstract Algebra, 3rd edn. John Wiley, Hoboken (2004)
21. Eberli, S., Cescato, D., Fichtner, W.: Divide-and-conquer matrix inversion for linear MMSE detection in SDR MIMO receivers. In Proceedings of the 26th IEEE Norchip Conference, IEEE, New York, NY pp. 162–167 (2008)
22. Eberly, W.: Processor-efficient parallel matrix inversion over abstract fields: two extensions. In Proceedings of the 2nd International Symposium on Parallel Symbolic Computation, PASCO '97, ACM, New York, NY pp. 38–45 (1997)
23. Eberly, W., Giesbrecht, M., Giorgi, P., Storjohann, A., Villard, G.: Faster inversion and other black box matrix computations using efficient block projections. In Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation, ISSAC '07, ACM, New York, NY, USA pp. 143–150, (2007)
24. Ehrlich, L.W.: Complex matrix inversion versus real. Comm. ACM **13**, 561–562 (1970)
25. El-Hawary, M.: Further comments on "a note on the inversion of complex matrices". IEEE Trans. Automat. Contr. **20**(2), 279–280 (1975)
26. Freitas, N., Le Hung, B.V., Siksek, S.: Elliptic curves over real quadratic fields are modular. Invent. Math. **201**(1), 159–206 (2015)
27. Frobenius, F.G.: Gesammelte Abhandlungen. III. Springer-Verlag, Berlin, Bände I, II (1968)
28. Giesbrecht, M.: Nearly optimal algorithms for canonical matrix forms. SIAM J. Comput. **24**(5), 948–969 (1995)
29. Gill, P.E., Golub, G.H., Murray, W., Saunders, M.A.: Methods for modifying matrix factorizations. Math. Comp. **28**, 505–535 (1974)
30. Golub, G. H.: Matrix computation and the theory of moments. In Proceedings of the International Congress of Mathematicians, Birkhäuser, Basel. vol. 2, pp. 1440–1448 (1995)
31. Golub, G.H., Van Loan, C.F.: Matrix computations, 4th edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore (2013)
32. Golub, G.H., Wilkinson, J.H.: Ill-conditioned eigensystems and the computation of the Jordan canonical form. SIAM Rev. **18**(4), 578–619 (1976)
33. Heath, M.T., Geist, G.A., Drake, J.B.: Early experience with the Intel iPSC/860 at Oak Ridge National Laboratory. Int. J. High Perform. Comput. Appl. **5**(2), 10–26 (1991)
34. Higham, N.J.: Computing the polar decomposition–with applications. SIAM J. Sci. Statist. Comput. **7**(4), 1160–1174 (1986)
35. Higham, N.J.: Stability of a method for multiplying complex matrices with three real matrix multiplications. SIAM J. Matrix Anal. Appl. **13**(3), 681–687 (1992)
36. Higham, N.J.: The matrix sign decomposition and its relation to the polar decomposition. Linear Algebra Appl. **212**(213), 3–20 (1994)
37. Higham, N.J.: Accuracy and Stability of Numerical Algorithms, 2nd edn. SIAM, Philadelphia (2002)
38. Higham, N.J.: Functions of matrices. SIAM, Philadelphia (2008)
39. Hoffstein, J., Pipher, J., Silverman, J.H.: An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics, 2nd edn. Springer, New York (2014)
40. Howland, J.L.: The sign matrix and the separation of matrix eigenvalues. Linear Algebra Appl. **49**, 221–232 (1983)
41. IEEE standard for floating-point arithmetic. In IEEE Std 754-2019 (Revision of IEEE 754-2008), pages 1–84. IEEE, New York, NY (2019)
42. Karatsuba, A.A.: The complexity of computations. Trudy Mat. Inst. Steklov. **211**, 186–202 (1995)
43. Kenney, C., Laub, A.J.: On scaling Newton's method for polar decomposition and the matrix sign function. SIAM J. Matrix Anal. Appl. **13**(3), 698–706 (1992)
44. Klein, A., Mélard, G.: Computation of the Fisher information matrix for time series models. J. Comput. Appl. Math. **64**(1–2), 57–68 (1995)
45. Knuth, D.E.: The Art of Computer Programming, 3rd edn. Addison-Wesley, San Francisco (1998)
46. Krattenthaler, C.: A new matrix inverse. Proc. Amer. Math. Soc. **124**(1), 47–59 (1996)
47. Lanczos, C.: Applied Analysis. Prentice-Hall, Englewood Cliffs (1956)
48. Lim, L.-H.: Tensors in computations. Acta Numer. **30**, 555–764 (2021)

49. Lin, C.-C., Zmijewski, E.: A Parallel Algorithm for Computing the Eigenvalues of an Unsymmetric Matrix on an Simd Mesh of Processors. University of California, Santa Barbara (1991)
50. Lo, K.: Several numerical methods for matrix inversion. Int. J. Electr. Eng. Educ. **15**(2), 131–141 (1978)
51. Maindonald, J.H.: Statistical Computation. Series in probability and mathematical statistics: applied probability and mathematical statistics, John Wiley, New York (1984)
52. McCullagh, P., Nelder, J.A.: Generalized Linear Models. Monographs on Statistics and Applied Probability. Chapman & Hall, London (1989)
53. McEliece, R.J.: Finite fields for computer scientists and engineers. International Series in Engineering and Computer Science, Kluwer, Boston (1987)
54. Menezes, A.J., Blake, I.F., Gao, X., Mullin, R.C., Vanstone, S.A., Yaghoobian, T.: Applications of finite fields. International Series in Engineering and Computer Science, vol. 199. Kluwer, Boston, MA (1993)
55. Mignotte, M.: Mathematics for Computer Algebra. Springer-Verlag, New York (1992)
56. Mishra, B.: Algorithmic Algebra. Texts and Monographs in Computer Science. Springer-Verlag, New York (1993)
57. Mukherjee, P., Satish, L.: On the inverse of forward adjacency matrix. arXiv:1711.09216 (2017)
58. Munro, I.: Some results concerning efficient and optimal algorithms. In Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC, ACM, New York, NY. 71, pp. 40–44 (1971)
59. Overton, M.L.: Numerical computing with IEEE floating point arithmetic. SIAM, Philadelphia (2001)
60. Panda, S.K.: Inverses of bicyclic graphs. Electron. J. Linear Algebra **32**, 217–231 (2017)
61. Pavlíková, S.: A note on inverses of labeled graphs. Australas. J. Combin. **67**, 222–234 (2017)
62. Rao, C.R.: Information and the accuracy attainable in the estimation of statistical parameters. Bull. Calcutta Math. Soc. **37**, 81–91 (1945)
63. Roberts, J.D.: Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. Int. J. Control **32**(4), 677–687 (1980)
64. Roman, S.: Field Theory of Graduate Texts in Mathematics, 2nd edn. Springer, New York (2006)
65. Schur, I.: Gesammelte Abhandlungen, vol. I. III. Springer-Verlag, Berlin, II (1973)
66. Schur, J.: Über potenzreihen, die im innern des einheitskreises beschränkt sind. J. Reine Angew. Math. **148**, 122–145 (1918)
67. Smith, W.W., Smith, J.: Handbook of real-time fast Fourier transforms. IEEE, New York (1995)
68. Smith, W. W., Jr., Erdman, S.: A note on the inversion of complex matrices. IEEE Trans. Automatic Contol, AC-19:64 (1974)
69. Stinson, D.R., Paterson, M.B.: Cryptography. Textbooks in Mathematics, 4th edn. CRC Press, Boca Raton (2019)
70. Strassen, V.: Gaussian elimination is not optimal. Numer. Math. **13**, 354–356 (1969)
71. Studer, C., Fateh, S., Seethaler, D.: ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation. IEEE J. Solid-State Circuits **46**(7), 1754–1765 (2011)
72. Tornheim, L.: Inversion of a complex matrix. Comm. ACM **4**, 398 (1961)
73. Vassilevska Williams, V., Xu, Y., Xu, Z., Zhou, R.: New bounds for matrix multiplication: from alpha to omega. In Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 3792–3835. SIAM, Philadelphia, PA (2024)
74. Winograd, S.: A new algorithm for inner product. IEEE Trans. Comput. **17**(7), 693–694 (1968)
75. Winograd, S.: On the number of multiplications necessary to compute certain functions. Comm. Pure Appl. Math. **23**, 165–179 (1970)
76. Ye, D., Yang, Y., Mandal, B., Klein, D.J.: Graph invertibility and median eigenvalues. Linear Algebra Appl. **513**, 304–323 (2017)
77. Ye, K., Lim, L.-H.: Fast structured matrix computations: tensor rank and Cohn-Umans method. Found. Comput. Math. **18**(1), 45–95 (2018)
78. Zielinski, A.: On inversion of complex matrices. Internat. J. Numer. Methods Engrg. **14**(10), 1563–1566 (1979)