Advances in Linear Algebra & Matrix Theory, 2015, 5, 1-15
Published Online March 2015 in SciRes. http://www.scirp.org/journal/alamt
http://dx.doi.org/10.4236/alamt.2015.51001

Scientific Research Publishing

New Approach for the Inversion of Structured Matrices via Newton's Iteration

Mohammad M. Tabanjeh

Department of Mathematics and Computer Science, Virginia State University, Petersburg, VA, USA
Email: mtabanjeh@vsu.edu

Received 14 January 2015; accepted 5 February 2015; published 10 February 2015

Abstract

Newton's iteration is a fundamental tool for numerical solutions of systems of equations. The well-known iteration $X_{i+1} = X_i(2I - MX_i)$, $i \geq 0$ rapidly refines a crude initial approximation X_0 to the inverse of a general nonsingular matrix. In this paper, we will extend and apply this method to $n \times n$ structured matrices M, in which matrix multiplication has a lower computational cost. These matrices can be represented by their short generators which allow faster computations based on the displacement operators tool. However, the length of the generators is tend to grow and the iterations do not preserve matrix structure. So, the main goal is to control the growth of the length of the short displacement generators so that we can operate with matrices of low rank and carry out the computations much faster. In order to achieve our goal, we will compress the computed approximations to the inverse to yield a superfast algorithm. We will describe two different compression techniques based on the SVD and substitution and we will analyze these approaches. Our main algorithm can be applied to more general classes of structured matrices.

Keywords

Newton Iteration, Structured Matrices, Superfast Algorithm, Displacement Operators, Matrix Inverse.

1. Introduction

Frequently, matrices encountered in practical computations that have some special structures which can be exploited to simplify the computations. In particular, computations with dense structured matrices are ubiquitous in sciences, communications and engineering. Exploitation of structure enables dramatic acceleration of the computations and a major decrease in memory space, but sometimes it also leads to numerical stability problems.

How to cite this paper: Tabanjeh, M.M. (2015) New Approach for the Inversion of Structured Matrices via Newton's Iteration. *Advances in Linear Algebra & Matrix Theory*, **5**, 1-15. http://dx.doi.org/10.4236/alamt.2015.51001

The best-known classes of structured matrices are Toeplitz and Hankel matrices, but Cauchy and Vandermonde types are also quite popular. The computations with such matrices are widely applied in the areas of algebraic coding, control, signal processing, solution of partial differential equations and algebraic computing. For example, Toeplitz matrices arise in some major signal processing computations and Cauchy matrices appear in the study of integral equations and conformal mappings. The complexity of computations with $n\times n$ dense structured matrices dramatically decreases in comparison with the general $n\times n$ matrices, that is, from the order of n^2 words of storage space and n^{ω} arithmetic operations (ops) with $2.37<\omega\le 3$ in the best algorithms, to $O(n)$ words of storage space and to $O(n\log^2 n)$ ops (and sometimes to $O(n\log n)$ ops) (**Table 1**). The displacement rank r for an $m\times n$ Toeplitz and Hankel matrix is at most 2. A matrix is Toeplitz-like or Hankel-like if r is small or bounded by a small constant independent of m and n. Those matrices can be represented by $(m+n)r$ entries of its short displacement generators instead of mn entries which leads to more efficient storage of the entries in computer memory and much faster computations [1]. In this paper, we will focus our study on Newton's iteration for computing the inverse of a structured matrix and we will analyze the resulting algorithms. This iteration is well-known for its numerically stability and faster convergence. Newton's iteration

$$X_{i+1}=X_i(2I-MX_i),\qquad i\ge 0 \tag{1}$$

for matrix inversion was initially proposed by Schultz in 1933 and studied by Pan and others. The authors of [2] described quadratically convergent algorithms for the refinement of rough initial approximations to the inverse of Toeplitz and Toeplitz-like matrices and to the solutions of Toeplitz and Toeplitz linear systems of equations. The algorithms are based on the inversion formulas for Toeplitz matrices. The Cauchy-like case was studied in [3]. Since those matrices can be represented by their short generators, which allow faster com- putations based on the displacement operators tool, we can employ Newton's iteration to compute the inverse of the input structured matrices. However, Newton's iteration destroys the structure of the structured matrices when used. Therefore, in Section 5 we will study two compression techniques in details, namely, truncation of the smallest singular values of the displacement and the substitution of approximate inverses for the inverse matrix into its displacement expression to preserve the structure. Finally, each iteration step is reduced to two multiplications of structured matrices and each iteration is performed by using nearly linear time and optimal memory space which leads to superfast algorithm, especially if the input matrix is a well-conditioned structured matrix. Our main algorithm of Section 6 can be applied to more general classes of structured matrices. We will support our analysis with numerical experiments with Toeplitz matrices in Section 7.

2. Definition of Structured Matrices

Definition 1 *A matrix* $T=\left[t_{i,j}\right]_{i,j=0}^{n-1}$ *is a Toeplitz matrix if* $t_{i,j}=t_{i+1,j+1}$ *for every pair of its entries* $t_{i,j}$ *and* $t_{i+1,j+1}$. *A matrix* $H=\left[h_{i,j}\right]_{i,j=0}^{n-1}$ *is a Hankel matrix if* $h_{i,j}=h_{i-1,j+1}$ *for every pair of its entries* $h_{i,j}$ *and* $h_{i-1,j+1}$. *For a given vector* $\boldsymbol{v}=(v_i)_{i=0}^{n-1}$, *the matrix* $V=V(\boldsymbol{v})$ *of the form* $V=\left[v_i^j\right]_{i,j=0}^{n-1}$ *is called a Vandermonde matrix. Given two vectors* $\boldsymbol{s}$ *and* $\boldsymbol{t}$ *such that* $s_i\ne t_j$ *for all* i *and* j, *the* $n\times n$ *matrix* $C=C(\boldsymbol{s},\boldsymbol{t})$ *is a Cauchy* (*generalized Hilbert*) *matrix where* $C(\boldsymbol{s},\boldsymbol{t})=\left[\frac{1}{s_i-t_j}\right]_{i,j=0}^{n-1}$.

Example 1 $T=\begin{pmatrix}-3&4&6\\1&-3&4\\2&1&-3\end{pmatrix}$, $H=\begin{pmatrix}2&1&-3\\1&-3&4\\-3&4&6\end{pmatrix}$, $V(\boldsymbol{v})=\begin{pmatrix}1&2&4\\1&-1&1\\1&3&9\end{pmatrix}$, *and* $C(\boldsymbol{s},\boldsymbol{t})=\begin{pmatrix}\frac{1}{4}&-1&\frac{1}{5}\\\frac{-1}{2}&\frac{-1}{7}&-1\\\frac{1}{10}&\frac{1}{5}&\frac{1}{11}\end{pmatrix}$

are 3×3 *Teoplitz, Hankel, Vandermonde, and Cauchy matrices respectively that can be represented by the* 5*-dim vector* $\boldsymbol{T}=(6\quad 4\quad -3\quad 1\quad 2)^{\mathrm{T}}$, *the 5-dim vector* $\boldsymbol{H}=(2\quad 1\quad -3\quad 4\quad 6)^{\mathrm{T}}$, *the* 3*-dim vector* $\boldsymbol{v}=(2\quad -1\quad 3)^{\mathrm{T}}$, *and the* 3*- dim vectors* $\boldsymbol{s}=(1\quad -5\quad 7)^{\mathrm{T}}$ *and* $\boldsymbol{t}=(-3\quad -2\quad -4)^{\mathrm{T}}$ *respectively.*

We refer the reader to [4] and [5] for more details on the basic properties and features of structured matrices.

3. Displacement Representation of Structured Matrices

The concept of displacement operators and displacement rank, initially introduced by Kailath, Kung, and Morf in 1979, and studied by other authors such as Bini and Pan, is a powerful tool for dealing with matrices with structure. The displacement rank approach was intended for more restricted use when it was initially introduced in [6] (also [7]), namely, to measure how "close" to Toeplitz a given matrix is. Then the idea turned out to be even deeper and more powerful, thus it was developed, generalized and extended to other structured matrices. In this paper, we consider the most general and modern interpretation of the displacement of a matrix M (see **Definition 2**). The main idea is, for a given structured matrix M, we need to find an operator L that transforms the matrix M into a low rank matrix $L(M)$, such that one can easily recover M from its image $L(M)$ and operate with low rank matrices instead. Such operators that shift and scale the entries of the structured matrices turn out to be appropriate tools for introducing and defining the matrices of Toeplitz-like, Hankel-like, Vandermonde-like, and Cauchy-like types which we will introduce in **Definition 4**.

Definition 2 *For any fixed field* F (*say the complex field* C) *and a fixed pair* $\{A,B\}$ *of operator matrices, we define the linear displacement operators* $L: F^{m\times n} \to F^{m\times n}$ *of Sylvester type* $L = \nabla_{\{A,B\}}$:

$$L(M) = \nabla_{A,B}(M) = AM - MB \tag{2}$$

and Stein type $L = \Delta_{\{A,B\}}$:

$$L(M) = \Delta_{\{A,B\}}(M) = M - AMB. \tag{3}$$

The image $L(M)$ of the operator L is called the displacement of the matrix M. The operators of Sylvester and Stein types can be transformed easily into one another if at least one of the two associated operator matrices is nonsingular. This leads to the following theorem.

Theorem 1 $\nabla_{\{A,B\}} = A\Delta_{\{A^{-1},B\}}$ *if the operator matrix* A *is nonsingular, and* $\nabla_{\{A,B\}} = -\Delta_{\{A,B^{-1}\}}B$ *if the operator matrix* B *is nonsingular.*

Proof. $\nabla_{\{A,B\}}(M) = AM - MB = AM - AA^{-1}MB = A(M - A^{-1}MB) = A\Delta_{\{A^{-1},B\}}(M)$, and

$$\nabla_{\{A,B\}}(M) = AM - MB = AMB^{-1}B - MB = (AMB^{-1} - M)B = -(M - AMB^{-1})B = -\Delta_{\{A,B^{-1}\}}(M)B. \qquad \square$$

The operator matrices that we will use are the unit *f*-circulant matrix

$$Z_f = \begin{pmatrix} 0 & \cdots & 0 & f \\ 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} = (\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{n-1}, f\boldsymbol{e}_0),$$ and the diagonal matrix $D(\boldsymbol{v}) = (\mathrm{diag}(v_i))_{i=1}^{n-1}$. Note that

$Z_0 = Z = (\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{n-1}, 0)$ is unit lower triangular Toeplitz matrix.

$$Z_f(\boldsymbol{u}) = \sum_{i=0}^{n-1} u_i Z_f^i = \begin{pmatrix} u_0 & fu_{n-1} & \cdots & fu_1 \\ u_1 & u_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & fu_{n-1} \\ u_{n-1} & \cdots & u_1 & u_0 \end{pmatrix}$$ is an *f*-circulant for a vector $\boldsymbol{u} = (u_i)$. For example,

$Z_0(\boldsymbol{u}) = Z(\boldsymbol{u})$ is a lower triangular matrix.

Table 1. Parameters and flops (floating point arithmetic operations) count.

$m\times n$ Matrix M	Number of parameters	flops for Mv
General	mn	$2mn - m - n$
Toeplitz	$m+n-1$	$O((m+n)\log(m+n))$
Hankel	$m+n-1$	$O((m+n)\log(m+n))$
Vandermonde	m	$O((m+n)\log^2(m+n))$
Cauchy	$m+n$	$O((m+n)\log^2(m+n))$

Example 2 (*Toeplitz Matrices*). *Let* $T=\left[t_{i-j}\right]_{i,j=0}^{n-1}$. *Then if we apply the displacement operator of Sylvester type* (*2*) *to Toeplitz matrix* T *using the shift operator matrices* Z_1 *and* Z_0, *we will get*:

$$\nabla_{\{Z_1,Z_0\}}(T)=Z_1T-TZ_0=\begin{pmatrix} t_{n-1} & t_{n-2} & \cdots & t_0 \\ t_0 & t_{-1} & \cdots & t_{1-n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n-2} & t_{n-1} & \cdots & t_{-1} \end{pmatrix}-\begin{pmatrix} t_{-1} & \cdots & t_{1-n} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ t_{n-3} & \cdots & t_{-1} & 0 \\ t_{n-2} & \cdots & t_0 & 0 \end{pmatrix}=\begin{pmatrix} t_{n-1}-t_{-1} & \cdots & t_1-t_{1-n} & t_0 \\ 0 & \cdots & 0 & t_{1-n} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & t_{-1} \end{pmatrix}.$$

This operator displaces the entries of the matrix T *so that the image can be represented by the first row and the last column. Furthermore, the image is a rank* 2 *matrix*

Notice that the operator matrices vary depending on the structure of the matrix M. We use shift operator matrices (such as Z_f and Z_f^{T} for any scalar f) for the structures of Toeplitz and Hankel type, scaling operator matrices (such as $D(\boldsymbol{x})$) for the Cauchy structure, and shift and scaling operator matrices for the structure of Vandermonde type. One can restrict the choices of f to 0 and 1. However, other operator matrices may be used with other matrix structures (see **Table 2**).

Definition 3 *For an* $m\times n$ *structured matrix* M *and an associated operator* $L=\nabla_{\{A,B\}}$, *or* $L=\Delta_{\{A,B\}}$, *the number* $r=\operatorname{rank}(L(M))$ *is called the* L*-rank or the displacement rank of the matrix* M.

The constant r in **Definition 3** usually remains relatively small as the size of M grows large, that is, r is small relative to $\min(m,n)$, say $r=O(1)$ or $r=O(\min\{m,n\})$ as m and n grow large. In this case, we call the matrix L-like and having structure of type L. Now let us recall the linear operators L of (2) and (3) that map a matrix M into its displacements:

$$L(M)=\nabla_{\{A,B\}}(M)=AM-MB=GH^{\mathrm{T}} \tag{4}$$

and

$$L(M)=\Delta_{\{A,B\}}(M)=M-AMB=GH^{\mathrm{T}} \tag{5}$$

where A and B are operator matrices of **Table 2**, $G=(\boldsymbol{g}_1,\cdots,\boldsymbol{g}_l)$ and $H=(\boldsymbol{h}_1,\cdots,\boldsymbol{h}_l)$ are $l\times n$ matrices. Then the matrix pair (G,H) is called a (nonunique) L-generator (or displacement generator) of length l for M, $l\geq r=\operatorname{rank}(L(M))$ (the L-rank or the displacement rank of M). If M is an $m\times n$ matrix and l is small relative to m and n, then M is said to have L-structure or to be an L-structured matrix. The displacement $L(M)=GH^{\mathrm{T}}$ (as a matrix of small rank) can be represented with a short generator defined by only a small number of parameters. For instance, the singular value decomposition (SVD) of the matrix $L(M)$ of size $m\times n$ gives us its orthogonal generator that consists of two generator matrices G and H having orthogonal columns and of sizes $m\times r$ and $n\times r$ respectively, that is a total of $(m+n)r$ entries. The matrix pair (G,H) is called an L-generator (or displacement generator) of length r for M. It is also a generator of length r for $L(M)$. Note that the pair (G,H) is nonunique for a fixed $L(M)$. In particular, for Toeplitz, Hankel, Vandermonde, and Cauchy matrices, the length r of the associated generators (4) and (5) is as small as 1 or 2 for some appropriate choices of the operator matrices A and B of **Table 2**. The four classes of structured matrices are naturally extended to Toeplitz-like, Hankel-like, Vandermonde-like, and Cauchy-like matrices, for which r is bounded by a fixed (not too large) constant.

Definition 4 *An* $m\times n$ *matrix* M *is Toeplitz-like if* $r=\operatorname{rank}(L(M))$ *is small relative to* $\{m,n\}$ *and if*

Table 2. Operator Matrices for $\nabla_{\{A,B\}}(M)$ and $\Delta_{\{A,B\}}(M)$.

Structured Matrices	Operator Matrices for $\nabla_{\{A,B\}}$	Operator Matrices for $\Delta_{\{A,B\}}$	Rank of $\nabla_{\{A,B\}}$ or $\Delta_{\{A,B\}}$
Toeplitz	(Z_e,Z_f) or $(Z_e^{\mathrm{T}},Z_f^{\mathrm{T}})$, $e\neq f$	(Z_e^{T},Z_f) or (Z_e,Z_f^{T}), $ef\neq 1$	rank ≤ 2
Hankel	(Z_e,Z_f^{T}) or (Z_e^{T},Z_f), $ef\neq 1$	$(Z_e^{\mathrm{T}},Z_f^{\mathrm{T}})$ or (Z_e,Z_f), $e\neq f$	rank ≤ 2
Vandermonde	$(D(\boldsymbol{x}),Z_f)$ or $(D^{-1}(\boldsymbol{y}),Z_f^{\mathrm{T}})$	$(D^{-1}(\boldsymbol{y}),Z_f)$ or $(D(\boldsymbol{x}),Z_f^{\mathrm{T}})$	rank ≤ 1
Cauchy	$(D(\boldsymbol{x}),D(\boldsymbol{y}))$	$(D(\boldsymbol{x}),D^{-1}(\boldsymbol{y}))$ or $(D^{-1}(\boldsymbol{x}),D(\boldsymbol{y}))$	rank ≤ 1

M *is given with its* L*-generator of length* $l=O(r)$, *where* $L=\nabla_{Z_e,Z_f}$, $L=\nabla_{Z_e^{\mathrm{T}},Z_f^{\mathrm{T}}}$, $L=\nabla_{Z_e,Z_f^{\mathrm{T}}}$, *and* $L=\nabla_{Z_e^{\mathrm{T}},Z_f}$ *for a fixed pair of scalars* e *and* f. *A matrix* M *is Hankel-like if* MJ *or* JM *is Toeplitz-like where* $J=\begin{pmatrix}0&\cdots&0&1\\ \vdots&\iddots&\iddots&0\\ 0&\iddots&\iddots&\vdots\\ 1&0&\cdots&0\end{pmatrix}=(\boldsymbol{e}_{n-1},\cdots,\boldsymbol{e}_0)$ *is the reflection matrix.*

Therefore, the problems of solving Toeplitz-like and Hankel-like linear systems are reduced to each other.

Remark 1 *In the case where the operator* L *is associated with Toeplitz, Hankel, Vandermonde, or Cauchy matrices* M, *we call an* L*-like matrix Toeplitz-like, Hankel-like, Vandermonde-like, or Cauchy-like matrix, respectively, and we say that the matrix has the structure of Toeplitz, Hankel, Vandermonde, or Cauchy type respectively. To take advantage of the matrix structure, we are going to COMPRESS the structured input matrix* M *via its short L-generators based on* ***Theorem*** **2** *below, then we will OPERATE with L-generators rather than with the matrix itself, and finally RECOVER the output from the computed short L-generators.*

Definition 5 *A linear operator* L *is nonsingular if the matrix equation* $L(M)=0$ *implies that* $M=0$.

Definition 6 *We define the norm of a nonsingular linear operator* L *and its inverse* L^{-1} *as follows*: $\mu=\mu_{r,d}(L)=\sup_M\left(\frac{\|L(M)\|_d}{\|M\|_d}\right)$ *and* $\mu^-=\mu_{r,d}\left(L^{-1}\right)=\sup_M\left(\frac{\|M\|_d}{\|L(M)\|_d}\right)$, *for* $d=1,2,\infty$, *or* F, *and where the supremum is taken over all matrices* M *having a positive* L*-rank of at most* r. *In this case the condition number* κ *of the operator* L *is defined as* $\mathrm{cond}(L)=\kappa=\mu\mu^-=\mu_{r,d}(L)\mu_{r,d}^-\left(L^{-1}\right)$

Theorem 2 [8] *Let* G *and* H *be a pair of* $n\times l$ *matrices,* $G=(\boldsymbol{g}_1,\cdots,\boldsymbol{g}_l)$ *and* $H=(\boldsymbol{h}_1,\cdots,\boldsymbol{h}_l)$, *where* $\boldsymbol{g}_i$ *and* $\boldsymbol{h}_i$ *denote the* i*-th columns of* G *and* H *respectively. Let* $L(M)=GH^{\mathrm{T}}$. *Then we have one of the following Toeplitz type matrices*: $M=\frac{1}{e-f}\sum_{j=1}^{l}Z_e(\boldsymbol{g}_j)Z_f(J\boldsymbol{h}_j)$, *where* $L=\nabla_{Z_e,Z_f}$, $e\neq f$.

$M=\frac{1}{1-ef}\sum_{j=1}^{l}Z_e(\boldsymbol{g}_j)Z_f^{\mathrm{T}}(\boldsymbol{h}_j)$, *where* $L=\Delta_{Z_e,Z_f^{\mathrm{T}}}$, $ef\neq 1$. $M=\frac{1}{e-f}\sum_{j=1}^{l}Z_e(\boldsymbol{g}_j)Z_f(\boldsymbol{h}_j)J$, *where* $L=\nabla_{Z_e,Z_f^{\mathrm{T}}}$, $e\neq f$. $M=\frac{1}{1-ef}\sum_{j=1}^{l}Z_e(\boldsymbol{g}_j)Z_f^{\mathrm{T}}(J\boldsymbol{h}_j)J$, *where* $L=\Delta_{Z_e,Z_f^{\mathrm{T}}}$, $ef\neq 1$.

Theorem 3 *Let M be a nonsingular matrix, then we have the following*:

1) $\nabla_{A,B}\left(M^{-1}\right)=-M^{-1}\nabla_{B,A}(M)M^{-1}$.
2) $\Delta_{A,B}\left(M^{-1}\right)=AM^{-1}\Delta_{B,A}(M)A^{-1}M^{-1}$, *if* A *is a nonsingular matrix.*
3) $\Delta_{A,B}\left(M^{-1}\right)=M^{-1}B^{-1}\Delta_{B,A}(M)M^{-1}B$, *if* B *is a nonsingular matrix.*

Proof.

1) $\nabla_{A,B}\left(M^{-1}\right)=AM^{-1}-M^{-1}B=-\left(M^{-1}B-AM^{-1}\right)=-\left(M^{-1}BMM^{-1}-M^{-1}MAM^{-1}\right)$
$=-M^{-1}(BM-MA)M^{-1}=-M^{-1}\nabla_{B,A}(M)M^{-1}$,
2) $\Delta_{A,B}\left(M^{-1}\right)=M^{-1}-AM^{-1}B=AA^{-1}M^{-1}-AM^{-1}BMM^{-1}=AM^{-1}MA^{-1}M^{-1}-AM^{-1}BMAA^{-1}M^{-1}$
$=AM^{-1}(M-BMA)A^{-1}M^{-1}=AM^{-1}\Delta_{B,A}(M)A^{-1}M^{-1}$,
3) $\Delta_{A,B}\left(M^{-1}\right)=M^{-1}-AM^{-1}B=M^{-1}B^{-1}B-AM^{-1}B=M^{-1}B^{-1}MM^{-1}B-M^{-1}MAM^{-1}B$
$=M^{-1}B^{-1}MM^{-1}B-M^{-1}B^{-1}BMAM^{-1}B=M^{-1}B^{-1}(M-BMA)M^{-1}B=M^{-1}B^{-1}\Delta_{B,A}(M)M^{-1}B.$ □

Theorem 4 *Let a pair of* $n\times l$ *matrices G and* H *form a* $\nabla_{A,B}$*-generator of length* l *for a nonsingular matrix* M. *Write* $M^{-1}G=-U$ *and* $H^TM^{-1}=W^{\mathrm{T}}$. *Then* $\nabla_{B,A}\left(M^{-1}\right)=UW^{\mathrm{T}}$.

Proof. By **Theorem 3**, $\nabla_{B,A}\left(M^{-1}\right)=-M^{-1}\nabla_{A,B}(M)M^{-1}=-M^{-1}GH^{\mathrm{T}}M^{-1}=UW^{\mathrm{T}}$, where $M^{-1}G=-U$ and $H^{\mathrm{T}}M^{-1}=W^{\mathrm{T}}$. □

4. Newton's Iteration for General Matrix Inversion

Given an $n\times n$ nonsingular matrix A and an initial approximation X_0 to its inverse A^{-1} that satisfies the bound $\rho=\|I-X_0A\|<1$ for some positive ρ and some fixed matrix norm, Newton's iteration (1) can be written in the form

$$X_{i+1}=X_i\left(2I-AX_i\right)=2X_i-X_iAX_i=\left(2I-X_iA\right)X_i;\quad i\geq 0 \tag{6}$$

and rapidly improves the initial approximation X_0 to the inverse. X_0 can be supplied by either some computational methods such as preconditioning conjugate gradient method or by an exact solution algorithm that requires refinement due to rounding errors.

Lemma 1 *Newton's iteration* (6) *converges quadratically to the inverse of an* $n\times n$ *nonsingular matrix* A.

Proof. From (6), since $X_{i+1}=2X_i-X_iAX_i$, $i\geq 0$, then

$I-X_{i+1}A=I-2X_iA+X_iAX_iA=\left(I-X_iA\right)^2=\left(I-X_{i-1}A\right)^4=\cdots=\left(I-X_0A\right)^{2^{i+1}}$. We define the error matrices $E_i=A^{-1}-X_i$, $i\geq 0$ and the residual matrices $R_i=E_iA=I-X_iA$, $i\geq 0$. Since $\rho=\|I-X_0A\|<1$, then $\|I-X_iA\|<\rho^{2^i}$ and hence the norm of the residual matrices R_i is bounded above by ρ^{2^i}. This proves that Newton's iteration converges quadratically. □

If $\epsilon>0$ is a given residual norm bound, the bound

$$\|I-X_iA\|<\epsilon \tag{7}$$

is guaranteed in $\left\lceil \log_2\left(\frac{\log\epsilon}{\log\rho}\right)\right\rceil$ recursive steps (6), where $\lceil x\rceil$ stands for the smallest integer not exceeded by x. (7) also implies that $\|A\|\cdot\|A^{-1}-X_i\|\leq\|I-X_iA\|<\epsilon$. Since $\|A^{-1}\|\cdot\|A\|\geq 1$, it follows that $\frac{\|A^{-1}-X_i\|}{\|A^{-1}\|}<\epsilon$, so that the matrix X_i approximates A^{-1} with a relative error norm less than ϵ.

Remark 2 *Each step of Newton's iteration* (6) *involves the multiplication of* A *by* X_i, *the addition of* 2 *to all of the diagonal entries of the product, which takes* n *additions, and the pre-multiplication of the resulting matrix by* X_i. *The most costly steps are the two matrix multiplications; each step takes* n^3 *multiplications and* n^3-n^2 *additions for a pair of general* $n\times n$ *input matrices* A *and* X_0. *Of course, this cost will be reduced if the input matrix is structured* (*see Section* 5).

Example 3 (*Newton's iteration versus linear convergent schemes.*) *Let* $\rho=1/2$ *and* $\epsilon=10^{-6}$. *Then by quadratic convergence of Newton's iteration, we only need* 5 *iterative steps of* (6) *to compute* X_5 *that satisfies the bound* $\|I-X_5A\|<\epsilon=10^{-6}<\rho^{2^i}=(1/2)^{32}$ *and hence* $\|A^{-1}-X_5\|<\epsilon\|A^{-1}\|$. *If we consider any linearly convergent scheme such as Jacobi's or Gauss-Seidels's iteration* ([9], p. 611) *for the same input matrix, the norm of the error matrix of the computed approximation to* A^{-1} *decreases by factor* 2 *in each iteration step. The error norm decreases to below* 2^{-i-1} *in* i *iteration steps, so this will take* 19 *iteration steps to ensure the bound* (7) *for the same* $\epsilon=10^{-6}$.

5. Newton's Iteration for Structured Matrix Inversion

5.1. Newton-Structured Iteration

In Section 4, we showed that for a general input matrix A, the computational cost of iteration (6) is quite high because matrix products must be computed at every step (see **Remark 2**). However, for structured matrices M, matrix multiplication has a lower computational cost of $O\left(r_1r_2n\log^\beta n\right)$ flops for $\beta\leq 2$, provided that we operate with displacement generators of length r_1 and r_2 for the two input matrices M and X_0 respectively. So if M and X_0 are structured matrices, say Toeplitz or Toeplitz-like, the computations required for the Newton's iteration can be carried out more efficiently. In general, for the four classes of structured matrices of **Definition 1**, we usually associate various linear displacement operators L of Sylvester type $\nabla_{A,B}(M)$ and/or Stein type $\Delta_{A,B}(M)$. In this case, Newton's iteration can be efficiently applied and rapidly improves a rough initial approximation to the inverse, but also destroys the structure of the four classes of

structured matrices. Therefore, Newton's iteration has to be modified to preserve the initial displacement structure of the input structure matrix during the iteration and maintain matrix structure throughout the computation. The main idea here is to control the growth of the length of the short displacement generators so that we can operate with matrices of low rank and carry out the computations much faster. This can be done based on one of the following two techniques. The first technique is to periodically chop off the components corresponding to the smallest singular values in the SVDs of the displacement matrices defined by their generators. This technique can be applied to a Toeplitz-like input matrix. For a Cauchy-like input matrix C, there is no need to involve the SVD due to the availability of the inverse formula for C^{-1} [10]. The second technique is to control the growth of the length of the generators. This can be done by substituting the approximate inverses for the inverse matrix into its displacement. Here we assume that the matrices involved can be expressed via their displacement generators. Now, let us assume that the matrices M and X_0 have short generators $\Delta_{A,B}$ and $\Delta_{B,A}$ respectively. So by operating with short Δ-generators of the matrices M, X_i, and MX_i (or X_iM), we will have to modify Newton's iteration and use one of the above two techniques to control the length of a $\Delta_{B,A}$-generator of X_i, which seems to triple at every iterative step of (6). The same technique applies in the case when $\nabla_{A,B}$ and $\nabla_{B,A}$-generators are used. In this paper, we restrict our presentation to ∇-generators (see **Theorem 1**).

5.2. SVD-Based Compression of the Generators

For a fixed operator L and a matrix M, one can choose the orthogonal (SVD-based) L-generator matrices to obtain better numerical stability [11].

Definition 7 *A matrix $U \in C^{m\times m}$ is said to be unitary iff $U^*U = I$. For real matrices, U is orthogonal iff $U^{\mathrm{T}}U = I$. If $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m]$ is orthogonal, then $\boldsymbol{u}_i$ form an orthogonal basis for C^m.*

Definition 8 *Every $m\times n$ matrix W has its SVD* (*singular value decomposition*):

$$W = U\Sigma V^*, \quad U \in C^{m\times r}, \quad V \in C^{n\times r} \tag{8}$$

$$U^*U = V^*V = I_r, \quad \Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_r) \tag{9}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0 \tag{10}$$

where $r = \operatorname{rank}(W)$, $\sigma_1, \cdots, \sigma_r$ are singular values of the matrix W, U^* and V^* denote the Hermitian (conjugate) transpose of the orthogonal matrices U and V, respectively. That is, $U^* = \overline{U}^{\mathrm{T}}$ and $V^* = \overline{V}^{\mathrm{T}}$.

Note that if U and V are real matrices, then $U^* = U^{\mathrm{T}}$ and $V^* = V^{\mathrm{T}}$. Based on the SVD of a matrix W, its orthogonal generator of the minimum length is defined by

$$G = U, \quad H = V\Sigma \tag{11}$$

However, one can use an alternative diagonal scaling, in this case (11) can be written as

$$G = U\Sigma^{1/2}, \quad H = V\Sigma^{1/2}. \tag{12}$$

Note that if $W = L(M)$ for a matrix M, then the pair (G, H) is an orthogonal L-generator of minimum length for the matrix M.

Example 4 *Consider the following matrix* $W = \begin{pmatrix} 0.96 & 1.72 \\ 2.28 & 0.96 \end{pmatrix} = \begin{pmatrix} 0.6 & -0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{pmatrix}^{\mathrm{T}} = U\Sigma V^{\mathrm{T}}$.

Using Matlab command "$[U,S,V] = svd(W)$" produces the above results. Clearly, $\sigma_1 = 3$ is the largest singular value and $\sigma_2 = 1$ is the smallest singular value. $r = \operatorname{rank}(W) = 2$, $\|W\|_2 = \sigma_1 = 3$, and $\|W\|_F^2 = \sigma_1^2 + \sigma_2^2 = 10$ (Frobenius norm). Furthermore, by (12), $G = U\Sigma^{1/2} = \begin{pmatrix} 1.0392 & -0.8 \\ 1.3856 & 0.6 \end{pmatrix}$ and $H = V\Sigma^{1/2} = \begin{pmatrix} 1.3856 & 0.6 \\ 1.0392 & -0.8 \end{pmatrix}$. Note that $W = L(M) = GH^{\mathrm{T}} = \begin{pmatrix} 0.96 & 1.72 \\ 2.28 & 0.96 \end{pmatrix}$.

Remark 3 *In numerical computation of the SVD with a fixed positive tolerance ε to the output errors caused by rounding (ε is the output error bound which is also an upper bound on all the output errors of the*

singular values σ_i *and the entries of the computed matrices* U *and* V *and for* $W = L(M)$*), one truncates* (*sets to zero*) *all the smallest singular values* σ_i *up to* ε*. This will produce a numerical orthogonal generator* $(G_\varepsilon, H_\varepsilon)$ *of* ε*-length* r_ε *for a matrix* W*, which is also a numerical orthogonal L-generator for* M*. Here,* r_ε *is the* ε*-rank of the matrix* $W = L(M)$*, which is equal to the minimum number of the singular values of the displacement* $L(M)$ *exceeding* ε*,* $r_\varepsilon \le r = \operatorname{rank}(W)$*.*

Theorem 5 ([9], p. 79) *Given a matrix* W *of rank* r *and a nonnegative integer* k *such that* $k < r$*, it holds that* $\sigma_{k+1} = \min_{\operatorname{rank}(S)\le k} \|W - S\|_2$*. That is, the error in the optimal choice of an approximate of* W *whose rank does not exceed* k *is equal to the* $(k+1)$*-th singular value of* W*.*

Theorem 6 *If M is a nonsingular matrix, then* $\operatorname{rank}\left(\nabla_{A,B}(M)\right) = \operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right)$*.*

Proof. $\operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right) = \operatorname{rank}\left(BM^{-1} - M^{-1}A\right) = \operatorname{rank}\left(B - M^{-1}AM\right)$. On the other hand, $\operatorname{rank}\left(\nabla_{A,B}(M)\right) = \operatorname{rank}(AM - MB) = \operatorname{rank}\left(M^{-1}AM - B\right) = \operatorname{rank}\left(-\left(B - M^{-1}AM\right)\right) = \operatorname{rank}\left(B - M^{-1}AM\right)$. This implies that $\operatorname{rank}\left(\nabla_{A,B}(M)\right) = \operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right)$. □

Definition 9 *Let* $k = k_i = \operatorname{rank}\left(\nabla_{B,A}(X_i)\right)$*,* $k \le 3r$*,* $\forall i$*. We define*

$$e_{d,i} = \left\|X_i - M^{-1}\right\|_d, \quad d = 1,\ 2,\ \infty \quad \text{and} \quad e_i = \left\|X_i - M^{-1}\right\| \tag{13}$$

$$\hat{e}_{d,i} = \left\|Y_i - M^{-1}\right\|_d, \quad d = 1,\ 2,\ \infty \quad \text{and} \quad \hat{e}_i = \left\|Y_i - M^{-1}\right\|. \tag{14}$$

6. Our Main Algorithm and Results

Outline of the Techniques

By **Theorem 6**, we assume that $r = \operatorname{rank}\left(\nabla_{A,B}(M)\right) = \operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right)$ and $r_0 = \operatorname{rank}\left(\nabla_{A,B}(X_0)\right)$. The matrices X_i that approximate M^{-1} for large i have a nearby matrix of $\nabla_{B,A}$-displacement rank r for large i. This fact motivates the following approach to decrease the computational cost of the iteration in (6): We will replace X_i in (6) with a nearby matrix Y_i having $\nabla_{B,A}$-displacement rank of at most r and then restart the iteration with Y_i instead of X_i. The advantage of this modification is the decrease of the computational cost at the i-th Newton step and at all subsequent steps. However, the disadvantage is a possible deterioration of the approximation, since we do not know M^{-1}, and the transition from X_i to Y_i may occur in a wrong direction. But since X_i and Y_i are supposed to lie close to M^{-1}, hence they both lie close to each other. This observation enables us to bound the deterioration of the approximation relative to the current approximation error, so that the quadratic improvement in the error bound at the next step of Newton's iteration will immediately compensate us for such a deterioration. Furthermore, the transition from X_i to a nearby matrix Y_i of a small displacement rank can be done at a low computational cost. In the next main algorithm we describe this approach for Sylvester type operators only. The same results can be extended to Stein operators L using **Theorem 1** with slight modifications to the technique and the theory.

Algorithm 1 (*Newton's Iteration Using Sylverster Type Operators*)

INPUT: An integer $r > 0$, two matrices A and B defining the operator $\nabla_{A,B}$, an $n \times n$ nonsingular structured matrix M having $\nabla_{A,B}$-rank r and defined by its $\nabla_{A,B}$-generator (G,H) of length at most r, a sufficiently close initial approximation Y_0 to the inverse M^{-1} given with its $\nabla_{B,A}$-generator of length at most r, an upper bound λ on the number of Newton's iteration steps, and a compression subalgorithm $C1$ or $C2$ for the transition from a $\nabla_{B,A}$-generator of length at most $3r$ for an $n \times n$ matrix approximating M^{-1} to $\nabla_{B,A}$-generator of length at most r for a nearby matrix.

OUTPUT: A $\nabla_{B,A}$-generator of length at most r for a matrix Y_λ approximating M^{-1}.

COMPUTATION: For $i = 0, \cdots, n-1$, recursively compute a $\nabla_{B,A}$-generators of length at most $3r$ for the matrices

$$X_{i+1} = Y_i\left(2I - MY_i\right) \tag{15}$$

and then apply the compression subalgorithm $C1$ or $C2$ (transition from X_{i+1} to Y_{i+1}) to the matrix X_{i+1} to compute $\nabla_{B,A}$-generators of length at most r for the matrices Y_{i+1}.

At the i-th step of (15) of **Algorithm 1**, a $\nabla_{B,A}$-generator of length at most $3r$ for the matrix X_{i+1} can be computed using $O\left(rn\log^{\beta}n\right)$ flops, for $\beta \le 2$. As mentioned earlier, the main idea is to control the growth of the length of the short displacement generators. So the question is how can we control the length of the computed L-generators? That is, we need to compress the length of the generators. Therefore, to complete **Algorithm 1**, we need to introduce two compression subalgorithms, namely, subalgorithm $C1$ and $C2$. First, we describe the compression subalgorithm $C1$ based on the SVD of the displacement $\nabla_{B,A}\left(X_i\right)$. To do this, we compute the SVD based orthogonal generator of the displacement $W = L\left(X_i\right) = \nabla_{B,A}\left(X_i\right)$ and decrease the length of the L-generator to r by zeroing (truncating) all singular values σ_i for $i \ge r+1$. Then output the resulting orthogonal $\nabla_{B,A}$-generator of at most r for a nearby matrix Y_i which is unique and lies near X_i.

Remark 4 *By* ***Theorem*** **5** *we can deduce that* $\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2 \le \left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(M^{-1}\right)\right\|$ *because* $\operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right) \le r$ *and this also implies that* Y_i *lies near* X_i *if the operator* $\nabla_{B,A}$ *is invertible.*

Lemma 2 (1) $\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2 = \sigma_{r+1}\left(\nabla_{B,A}\left(X_i\right)\right)$, (2) $\left\|\nabla_{B,A}\left(M^{-1}\right)-\nabla_{B,A}\left(X_i\right)\right\| \le \left(\|A\|+\|B\|\right)e_i$, *where* e_i *as in* (13).

Proof. 1) Since $\|M\|_2 = \sigma_1(M)$ (largest singular value), then (1) follows from this fact. 2) Since $\nabla_{B,A}\left(M^{-1}\right) = BM^{-1} - M^{-1}A$ then $\nabla_{B,A}\left(X_i\right) = BX_i - X_iA$. Now,

$$\begin{aligned}\left\|\nabla_{B,A}\left(M^{-1}\right)-\nabla_{B,A}\left(X_i\right)\right\| &= \left\|BM^{-1}-M^{-1}A-BX_i+X_iA\right\| = \left\|B\left(M^{-1}-X_i\right)-\left(M^{-1}-X_i\right)A\right\| \\ &= \left\|\left(X_i-M^{-1}\right)A-B\left(X_i-M^{-1}\right)\right\| \le \left\|X_i-M^{-1}\right\|\cdot\|A\|+\|B\|\cdot\left\|X_i-M^{-1}\right\| \\ &\le \left(\|A\|+\|B\|\right)\left\|X_i-M^{-1}\right\| = \left(\|A\|+\|B\|\right)e_i.\end{aligned}$$

□

Lemma 3 $\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2 \le \left(\|A\|_2+\|B\|_2\right)e_{2,i}$, *where* $e_{2,i}$ *as in* (13).

Proof. Using the estimate from ([9], p. 442), that is, for any two matrices A and E, $\left|\sigma_j\left(A+E\right)-\sigma_j\left(A\right)\right| \le \sigma_1\left(E\right) = \|E\|_2$ for $j = 1,\cdots,n$ and conclude that $\left|\sigma_j\left(\nabla_{B,A}\left(X_i\right)\right)-\sigma_j\left(\nabla_{B,A}\left(M^{-1}\right)\right)\right| \le \left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(M^{-1}\right)\right\|_2$ where $\sigma_j(W)$ are defined in (8) - (10) of **Definition 8**. For all $j > r$ we have $\sigma_j\left(\nabla_{B,A}\left(M^{-1}\right)\right) = 0$ and then we obtain $\left|\sigma_j\left(\nabla_{B,A}\left(X_i\right)\right)\right| \le \left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(M^{-1}\right)\right\|_2$. Now, we use (2) of **Lemma 2** and deduce $\left|\sigma_j\left(\nabla_{B,A}\left(X_i\right)\right)\right| \le \left(\|A\|_2+\|B\|_2\right)e_{2,i}$, for $j > r$. If we combine the latest inequality with equation (1) of **Lemma 2** for $j = r+1$, then we conclude that $\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2 \le \left(\|A\|_2+\|B\|_2\right)e_{2,i}$. □

Now let us use **Lemma 2** and **Lemma 3** to prove the bound in the next theorem.

Theorem 7 *If* $\nabla_{B,A}$ *is a nonsingular linear operator and* $e_{2,i}$ *is a positive scalar defined in* (13), *then the bound* $\left\|Y_i-M^{-1}\right\| \le \left(1+\left(\|A\|_2+\|B\|_2\right)\mu^-\right)e_{2,i}$ *holds for* $\mu^- = \mu_{r,2}\left(\nabla_{B,A}^{-1}\left(M\right)\right)$ *of* ***Definition*** **6**.

Proof. By **Definition 6** replacing L by $\nabla_{B,A}$, d = 2, and M by $X_i - Y_i$ we have $\mu^- = \sup_M\left(\dfrac{\left\|X_i-Y_i\right\|_2}{\left\|\nabla_{B,A}\left(X_i-Y_i\right)\right\|_2}\right)$, which implies that $\left\|X_i-Y_i\right\|_2 \le \mu^-\left\|\nabla_{B,A}\left(X_i-Y_i\right)\right\|_2 \le \mu^-\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2$ by linearity of $\nabla_{B,A}$. Now, $\left\|M^{-1}-Y_i\right\|_2 = \left\|M^{-1}-X_i+X_i-Y_i\right\|_2 \le \left\|M^{-1}-X_i\right\|_2 + \left\|X_i-Y_i\right\|_2$. Then replace $e_{2,i} = \left\|M^{-1}-X_i\right\|_2$ and $\left\|X_i-Y_i\right\|_2 \le \mu^-\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2$ to deduce $\left\|M^{-1}-Y_i\right\|_2 \le e_{2,i} + \left\|X_i-Y_i\right\|_2 \le e_{2,i} + \mu^-\left\|\nabla_{B,A}\left(X_i\right)-\nabla_{B,A}\left(Y_i\right)\right\|_2$. Finally, use the inequality of **Lemma 3** to conclude that $\left\|M^{-1}-Y_i\right\|_2 \le e_{2,i} + \mu^-\left(\|A\|_2+\|B\|_2\right)e_{2,i} = \left(1+\left(\|A\|_2+\|B\|_2\right)\mu^-\right)e_{2,i}$. This completes the proof. □

Subalgorithm 1 ($C1$: *Compression of Displacement by Truncation of their Smallest Singular Values.*)

INPUT: An integer $r > 0$, two matrices A and B defining the operator $\nabla_{A,B}$ for an $n \times n$ nonsingular structured matrices M, $r = \operatorname{rank}\left(\nabla_{A,B}\left(M\right)\right) = \operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right)$, and a $\nabla_{B,A}$-generator $\left(G_i, H_i\right)$ of length

at most $k = k_i$ for a matrix X_i such that $r \le k$, $\nabla_{B,A}(X_i) = G_i H_i^{\mathrm{T}}$.

OUTPUT: A $\nabla_{B,A}$-displacement generator of length at most r for a matrix Y_i such that $\left\|Y_i - M^{-1}\right\|_2 \le \left(1 + \left(\|A\|_2 + \|B\|_2\right)\mu^-\right)e_{2,i}$, where $e_{2,i}$ as in (13) and $\mu^- = \mu_{r,2}\left(\nabla_{A,B}^{-1}\right)$ of **Definition 6**.

COMPUTATIONS: Compute the SVD of the displacement $\nabla_{B,A}(X_i) = U_i \Sigma_i V_i^{\mathrm{T}}$ (the computation is not costly since it is performed with $G_i H_i^{\mathrm{T}}$, where G_i, $H_i \in C^{n \times k_i}$ and since k_i is small, see [11]). Set to zero the diagonal entries $\sigma_{r+1}, \cdots, \sigma_k$ of Σ_i, that is, turning Σ_i into a diagonal matrix of rank at most r. ($\sigma_{r+1}, \cdots, \sigma_k$ are $k - r$ smallest singular values of the matrix $\nabla_{B,A}(X_i)$.) Compute and output the matrices G_i^*, H_i^* obtained from $U_i \Sigma_i$ and V_i respectively, by deleting their last $k - r$ columns.

Example 5 *For $e \ne f$, consider the operator $L = \nabla_{Z_e, Z_f}$, $(e - f)X_i = \sum_{j=1}^{l} Z_e(\boldsymbol{g}_j) Z_f(J\boldsymbol{h}_j)$ where the pair $G = (\boldsymbol{g}_j)_{j=1}^{l}$, $H = (\boldsymbol{h}_j)_{j=1}^{l}$ is the orthogonal, SVD based generator for the displacement $L(X_i)$. Let $\operatorname{rank}\left(L\left(M^{-1}\right)\right) = \operatorname{rank}\left(\nabla_{Z_f, Z_e}(M)\right) = r \le l$. Then **Subalgorithm 1** is applied to the matrix X_i and outputs the matrix $Y_i = (e - f)^{-1} \sum_{j=1}^{r} Z_e(\boldsymbol{g}_j) Z_f(J\boldsymbol{h}_j)$. The computational cost of performing **Subalgorithm 1** is dominated by the computation of the SVD of the displacement which requires $O(ir^2 n \log^{\beta} n)$ flops for $\beta \le 2$.*

Next, we will present the second compression technique based on substitution which does not depend on the SVD. By **Theorem 4** we conclude that $L\left(M^{-1}\right) = \nabla_{B,A}\left(M^{-1}\right) = UW^{\mathrm{T}} = -M^{-1} G H^{\mathrm{T}} M^{-1}$, and by **Theorem 3** we also have $L\left(M^{-1}\right) = \Delta_{B,A}\left(M^{-1}\right) = BM^{-1}\Delta_{A,B}(M)B^{-1}M^{-1}$ if B is nonsingular, and $L\left(M^{-1}\right) = \Delta_{B,A}\left(M^{-1}\right) = M^{-1}A^{-1}\Delta_{A,B}(M)M^{-1}A$ if A is nonsingular. Clearly this expresses the displacement of M^{-1} via the displacement of M, the operator matrices A and B, and the product of the inverse with $2r$ specified vectors, where $r = \operatorname{rank}(L(M))$ for $L = \nabla_{A,B}$ or $L = \Delta_{A,B}$. Now since $\nabla_{B,A}\left(M^{-1}\right) = -M^{-1} G H^{\mathrm{T}} M^{-1}$, let us substitute X_i for M^{-1} on the right hand side and define a short $\nabla_{B,A}$-generator for the matrix Y_i:

$$\nabla_{B,A}(Y_i) = -X_i G H^{\mathrm{T}} X_i = U_i W_i^{\mathrm{T}}, \quad U_i = -X_i G \in C^{n \times r}, \quad W_i^{\mathrm{T}} = H^{\mathrm{T}} X_i \in C^{r \times n} \tag{16}$$

Since $X_i \approx M^{-1}$, we expect $Y_i \approx M^{-1}$ because $\nabla_{B,A}(Y_i) \approx \nabla_{B,A}\left(M^{-1}\right)$.

Subalgorithm 2 ($C2$: *Compression of the displacement by substitution.*)

INPUT: An integer $r > 0$, a pair of $n \times n$ operator matrices A and B defining the operator $\nabla_{B,A}$, a $\nabla_{A,B}$-generator of length r for a nonsingular $n \times n$ matrix M where $r = \operatorname{rank}\left(\nabla_{A,B}(M)\right) = \operatorname{rank}\left(\nabla_{B,A}\left(M^{-1}\right)\right)$, and a $\nabla_{B,A}$-generator (G_{i+1}, H_{i+1}) of length at most $3r$ for a matrix X_{i+1} of Newton's iteration (6).

OUTPUT: A $\nabla_{B,A}$-generator (U_{i+1}, W_{i+1}) of length at most r for a matrix Y_{i+1} satisfying the equation $\nabla_{B,A}(Y_{i+1}) = U_{i+1} W_{i+1}^{\mathrm{T}}$ with the bound

$$\hat{e}_{i+1} = \left\|Y_{i+1} - M^{-1}\right\| \le \nu_i e_i \tag{17}$$

where $\nu_i = \mu^- \left\|G H^{\mathrm{T}}\right\| \left(e_i + 2\left\|M^{-1}\right\|\right)$, $\mu^- = \mu_{r,l}\left(\nabla_{B,A}^{-1}\right)$ of **Definition 6**, $\hat{e}_{i+1}$ of (14), and for e_i of (13).

COMPUTATIONS: Compute the matrix products

$$U_{i+1} = Y_i (MY_i - 2I) G = -X_{i+1} G \tag{18}$$

and

$$W_{i+1}^{\mathrm{T}} = H^{\mathrm{T}} Y_i (MY_i + 2I) = H^{\mathrm{T}} X_{i+1}. \tag{19}$$

Now the pair (U_{i+1}, W_{i+1}) is a $\nabla_{B,A}$-generator of length at most r for a matrix Y_{i+1} satisfying the equation $\nabla_{B,A}(Y_{i+1}) = U_{i+1} W_{i+1}^{\mathrm{T}}$. Here we assume the operator $L = \nabla_{B,A}$, then the matrix Y_{i+1} is uniquely defined by its L-generators U_{i+1}, W_{i+1} and thus (18) and (19) completely define Newton's process without involving the matrix X_{i+1} in $X_{i+1} = Y_{i+1}(2I - MY_i)$. This subalgorithm achieves compression because $\operatorname{rank}\left(U_{i+1} W_{i+1}^{\mathrm{T}}\right) \le \operatorname{rank}\left(G H^{\mathrm{T}}\right)$, and preserves approximation. Since $X_{i+1} \approx M^{-1}$, we deduce from **Theorem 3** that

$U_{i+1}W_{i+1}^{\mathrm{T}} \approx -M^{-1}GH^{\mathrm{T}}M^{-1} = \nabla_{B,A}\left(M^{-1}\right)$. The computation of the $n\times r$ matrices of (16) is reduced to multiplication of the matrix X_{i+1} by the $n\times(2r)$ matrix $(-G,H)$ which requires $O\left(r^2 n\log^{\beta} n\right)$ flops for $\beta \le 2$.

Now, we need to prove bound (17).

Theorem 8 *For the matrices* $U_j = -X_j G$, $W_j^{\mathrm{T}} = H^{\mathrm{T}}X_j$, $E_j = UW^{\mathrm{T}} - U_j W_j^{\mathrm{T}}$, *and for* $e_j = \left\|X_j - M^{-1}\right\|$, *we have* $\left\|E_j\right\| = \left\|U_j W_j^{\mathrm{T}} - UW^{\mathrm{T}}\right\| \le \left\|GH^{\mathrm{T}}\right\| e_j\left(e_j + 2\left\|M^{-1}\right\|\right)$.

Proof. Recall the matrix equations $U = -M^{-1}G$, and $W^{\mathrm{T}} = H^{\mathrm{T}}M^{-1}$. Then we can write $-U_j = X_j G = X_j G - M^{-1}G + M^{-1}G = \left(X_j - M^{-1}\right)G + M^{-1}G$, and $W_j^{\mathrm{T}} = H^{\mathrm{T}}X_j = H^{\mathrm{T}}X_j - H^{\mathrm{T}}M^{-1} + H^{\mathrm{T}}M^{-1} = H^{\mathrm{T}}\left(X_j - M^{-1}\right) + H^{\mathrm{T}}M^{-1}$. Now, let us write $E_j = UW^{\mathrm{T}} - U_j W_j^{\mathrm{T}}$, then

$$E_j = -M^{-1}GH^{\mathrm{T}}M^{-1} + \left(X_j - M^{-1}\right)GH^{\mathrm{T}}\left(X_j - M^{-1}\right) + M^{-1}GH^{\mathrm{T}}\left(X_j - M^{-1}\right) + \left(X_j - M^{-1}\right)GH^{\mathrm{T}}M^{-1}$$
$$+ M^{-1}GH^{\mathrm{T}}M^{-1} = \left(X_j - M^{-1}\right)GH^{\mathrm{T}}\left(X_j - M^{-1}\right) + M^{-1}GH^{\mathrm{T}}\left(X_j - M^{-1}\right) + \left(X_j - M^{-1}\right)GH^{\mathrm{T}}M^{-1}.$$

Now, take the norm of E_j:

$$\left\|E_j\right\| \le \left\|X_j - M^{-1}\right\|\cdot\left\|GH^{\mathrm{T}}\right\|\cdot\left\|X_j - M^{-1}\right\| + \left\|M^{-1}\right\|\cdot\left\|GH^{\mathrm{T}}\right\|\cdot\left\|X_j - M^{-1}\right\| + \left\|X_j - M^{-1}\right\|\cdot\left\|GH^{\mathrm{T}}\right\|\cdot\left\|M^{-1}\right\|$$
$$= e_j\left\|GH^{\mathrm{T}}\right\|e_j + \left\|M^{-1}\right\|\cdot\left\|GH^{\mathrm{T}}\right\|e_j + e_j\left\|GH^{\mathrm{T}}\right\|\cdot\left\|M^{-1}\right\|.$$

Simplifying the latest inequality yield the following bound:

$$\left\|E_j\right\| = \left\|U_j W_j^{\mathrm{T}} - UW^{\mathrm{T}}\right\| \le \left\|GH^{\mathrm{T}}\right\| e_j\left(e_j + 2\left\|M^{-1}\right\|\right) \quad \text{for} \quad e_j = \left\|X_j - M^{-1}\right\| \tag{20}$$

□

Theorem 9 *For* $\hat{e}_i = \left\|M^{-1} - Y_i\right\|$ *and* $\nu_i = \mu^{-}\left\|GH^{\mathrm{T}}\right\|\left(e_i + 2\left\|M^{-1}\right\|\right)$ *for* e_i *in* (13) *and* $\mu^{-} = \mu\left(\nabla_{B,A}^{-1}\right)$ *of* ***Definition*** **6**. *Then we have* $\hat{e}_{i+1} \le \nu_{i+1}e_{i+1}$ *for* $i \ge 0$.

Proof. Recall that $\hat{e}_{i+1} = \left\|Y_{i+1} - M^{-1}\right\| \le \mu^{-}\left\|\nabla_{B,A}\left(Y_{i+1} - M^{-1}\right)\right\|$, and since the operator $\nabla_{B,A}$ is linear, then we have $\hat{e}_{i+1} \le \mu^{-}\left\|\nabla_{B,A}\left(Y_{i+1}\right) - \nabla_{B,A}\left(M^{-1}\right)\right\| \le \mu^{-}\left\|U_{i+1}W_{i+1}^{\mathrm{T}} - UW^{\mathrm{T}}\right\| \le \mu^{-}\left\|E_{i+1}\right\|$. If we use (20) for $j = i+1$, we get $\hat{e}_{i+1} \le \mu^{-}\left\|E_{i+1}\right\| \le \mu^{-}\left\|GH^{\mathrm{T}}\right\|e_{i+1}\left(e_{i+1} + 2\left\|M^{-1}\right\|\right) = \nu_{i+1}e_{i+1}$. Therefore, $\hat{e}_{i+1} \le \nu_{i+1}e_{i+1}$. □

7. Numerical Experiments

Algorithm 1 was tested numerically for $n\times n$ Toeplitz input matrices M by applying the compression **Subalgoritm 1**. We use Matlab 8.3.0 on a Window 7 machine to run our tests. The Matlab SVD subroutine was used to obtain the singular value decomposition. In general, the complexity of the SVD on an $n\times n$ square matrix is $O\left(n^3\right)$ arithmetic operations. However, it is actually applied to structured matrices with smaller rank, so it is not really expensive. In addition, CLAPACK is one of the faster SVD algorithms that can be applied on real and complex matrices with less time complexity (see [11]). The tests results are summarized in the following example.

Example 6 N *refers to number of Newton's steps,* n *is the matrix size,* K *is number of times we ran the test,* ε *is a selected bound, and* $\kappa(M) = \mathrm{cond}_2(M) = \sigma_1(M)/\sigma_r(M)$ *is the condition number of* M, *where* $\sigma_i(M)$ *is the* i*-th condition number of the matrix* M. *We restricted our numerical experiments to the following classes of* $n\times n$ *Toeplitz matrices*:

1) *Real symmetric tridiagonal Toeplitz matrices*: $\left(t_{i,j}\right)_{i,j=0}^{n-1}$, $t_{i,j} = 0$ *where* $|i-j| > 1$, $t_{i+1,j} = t_{i,j+1} = 1$, $t_{i,i} = 4$ (*the first table of* **Table 3**) *or* -2 (*the second table of* **Table 3**) (*see* **Figure 1**).

2) *The matrices* $\left(\dfrac{1}{1+|i-j|}\right)_{i,j=0}^{n-1}$ (*third table represents symmetric positive definite and the fourth table represents symmetric positive indefinite of* **Table 3**) (*see* **Figure 1**).

3) *Randomly generated Toeplitz matrices, whose entries are chosen randomly from the interval* $[0,1)$ *and*

are uniformly distributed with mean 0 *and standard deviation of* 1 (*see* **Table 4** *and* **Figure 2**).

For a symmetric positive definite matrix M, the initial matrix $X_0 = Y_0 = \frac{I}{\|M\|_F}$ *was used, whereas for unsymmetric and symmetric indefinite Toeplitz matrices M, Newton's iteration was initialized with* $X_0 = Y_0 = \frac{M^{\mathrm{T}}}{\|M\|_1 \|M\|_\infty}$.

Table 3. First top two tables represent Tridiagonal Matrices of Class 1 and the last two tables represent matrices of Class 2.

n	κ	N
50	2.9924	6
100	2.9981	6
150	2.9991	6
200	2.9995	6
250	2.9997	6
300	2.9998	6
350	2.9998	6
n	κ	N
50	1.0535×10^{3}	20
100	4.1336×10^{3}	22
150	9.2402×10^{3}	23
200	1.6373×10^{4}	24
250	2.5533×10^{4}	24
300	2.6719×10^{4}	25
350	4.9931×10^{4}	25

n	κ	N
50	16.2215	11
100	19.6417	12
150	21.6801	12
200	23.1380	12
250	24.2739	12
300	25.2047	12
350	25.9933	13
n	κ	N
50	16.2215×10^{3}	8
100	19.6417×10^{3}	8
150	21.6801×10^{3}	9
200	23.1380×10^{4}	9
250	24.2739×10^{4}	9
300	25.2047×10^{4}	9
350	25.9933×10^{4}	9

Table 4. Random Matrices of Class 3.

	$n=100$	$\varepsilon=0.050$	
K	κ	ε	N
3	93.8994	0.050	20
7	97.8369	0.050	20
12	23.5604	0.050	17
15	35.1519	0.050	18
190	103.3290	0.050	20
20	82.4028	0.050	20
23	89.0855	0.050	21
	$n=100$	$\epsilon=0.013$	
K	κ	ε	N
1	102.0477	0.013	21
4	24.8330	0.013	17
8	58.9206	0.013	19
14	299.1225	0.013	25
17	643.5464	0.013	28
22	258.7756	0.013	25
25	144.0435	0.013	22

	$n=100$	$\epsilon=0.025$	
K	κ	ε	N
1	102.0477	0.025	21
3	93.8994	0.025	20
4	24.8330	0.025	17
8	58.9206	0.025	19
10	42.4488	0.025	18
11	139.8939	0.025	23
22	258.7756	0.025	24
	$n=100$	$\epsilon=0.00001$	
K	κ	ε	N
1	102.0477	0.00001	23
3	93.8994	0.00001	21
4	24.8330	0.00001	18
7	97.8369	0.00001	22
10	42.4488	0.00001	20
17	643.5464	0.00001	29
22	89.0855	0.00001	22

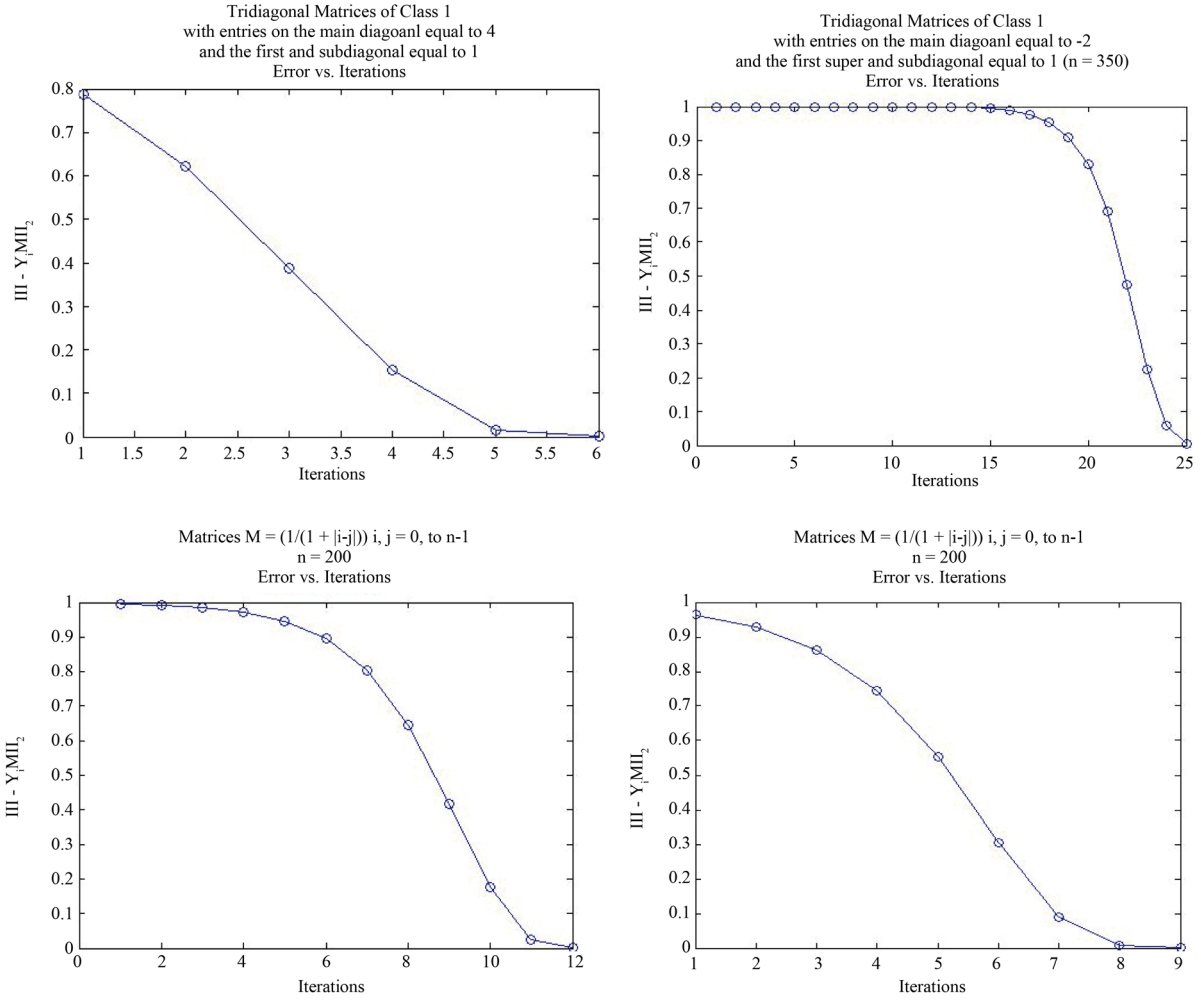

Figure 1. Error verses iterations.

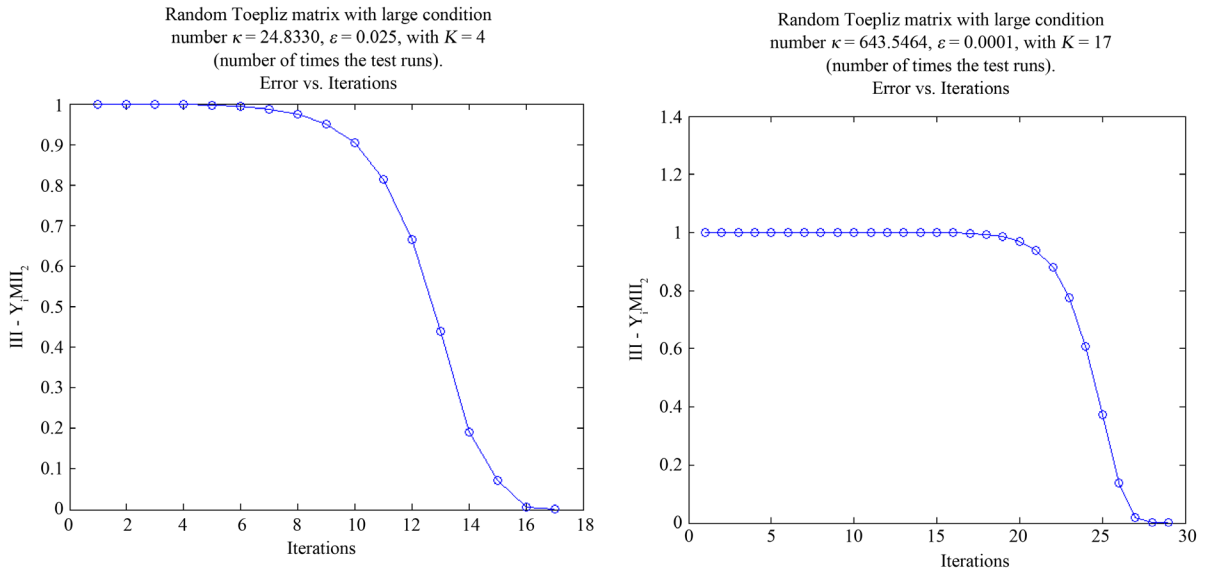

Figure 2. Error verses iterations.

Acknowledgements

We thank the editor and the referees for their comments.

References

[1] Kailath, T. and Sayed, A. (1999) Fast Reliable Algorithms for Matrices with Structure. Society for Industrial and Applied Mathematics, Philadelphia. http://dx.doi.org/10.1137/1.9781611971354

[2] Pan, V.Y., Branham, S., Rosholt, R. and Zheng, A. (1999) Newton's Iteration for Structured Matrices and Linear Systems of Equations, SIAM Volume on Fast Reliable Algorithms for Matrices with Structure. Society for Industrial and Applied Mathematics, Philadelphia.

[3] Pan, V.Y., Zheng, A.L., Huang, X.H. and Dias, O. (1997) Newton's Iteration for Inversion of Cauchy-Like and Other Structured Matrices. *Journal of Complexity*, **13**, 108-124. http://dx.doi.org/10.1006/jcom.1997.0431

[4] Bini, D. and Pan, V.Y. (1994) Polynomial and Matrix Computations, Vol. 1 Fundamental Algorithms. Birkhäuser, Boston.

[5] Pan, V.Y. (2001) Structured Matrices and Polynomials: Unified Superfast Algorithms. Birkhäuser, Boston.

[6] Kailath, T., Kung, S.-Y. and Morf, M. (1979) Displacement Ranks of Matrices and Linear Equations. *Journal of Mathematical Analysis and Applications*, **68**, 395-407. http://dx.doi.org/10.1016/0022-247X(79)90124-0

[7] Kailath, T. and Sayed, A.H. (2002) Displacement Structure: Theory and Applications. *SIAM Review*, **37**, 297-386. http://dx.doi.org/10.1137/1037082

[8] Pan, V.Y. and Rami, Y. (2001) Newton's Iteration for the Inversion of Structured Matrices. In: Bini, D., Tyrtyshnikov, E. and Yalamov, P., Eds., *Structured Matrices*: *Recent Developments in Theory and Computation*, Nova Science Publishers, New York, 79-90.

[9] Golub, G.H. and Van Loan, C.F. (2013) Matrix Computations. 4th Edition, John Hopkins University Press, Baltimore.

[10] Heinig, G. (1995) Inversion of Generalized Cauchy Matrices and the Other Classes of Structured Matrices. *The IMA Volume in Mathematics and Its Applications*, **69**, 63-81.

[11] Pan, V.Y. (1993) Decreasing the Displacement Rank of a Matrix. *SIAM Journal on Matrix Analysis and Application*, **14**, 118-121. http://dx.doi.org/10.1137/0614010

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.